



Alexander Kniebeiß

Daniel Fuhrmann

Stefan Kirchengast

Inhaltsverzeichnis

1.)Bestehendes Projekt	3
Projektbeschreibung	3
Kenndaten:	4
Hauptplatine.....	5
Umschaltungsplatine.....	6
Verstärker	7
LCD-Programmierung	8
Programmierung SPI und SD-Karte	10
Hauptprogramm.....	12
2.) verworfenes Projekt.....	13
Projektbeschreibung	13
Kenndaten	13
VLSI VS1053 Datenblatt.....	14
Hauptplatine.....	15
Programmierboard für den AT89C51RD2	16
3.)Projekttagebuch.....	17
4.)Probleme	20
5.)Resümee.....	20

1.)Bestehendes Projekt

Projektbeschreibung

Es ist ein MP3-Player welcher die Musikdateien von einer SD/MMC-Karte bezieht und sie auf internen Lautsprechern abspielt. Die jeweiligen Titel werden auf einem LC-Display angezeigt. Das Projekt besteht aus 4 Teilen:

- der Hauptplatine mit dem LC-Display
- einem primitiven Hama SD Mp3-Player
- der Umschaltung zwischen
 - Zugriff des Humerboards auf die SD-Karte
 - Zugriff des Mp3-Players auf die SD-Karte
- und einem Verstärker



Kenndaten:

SD-Karte: Kingston 512MB

LC-Display: Powertip 1601-A

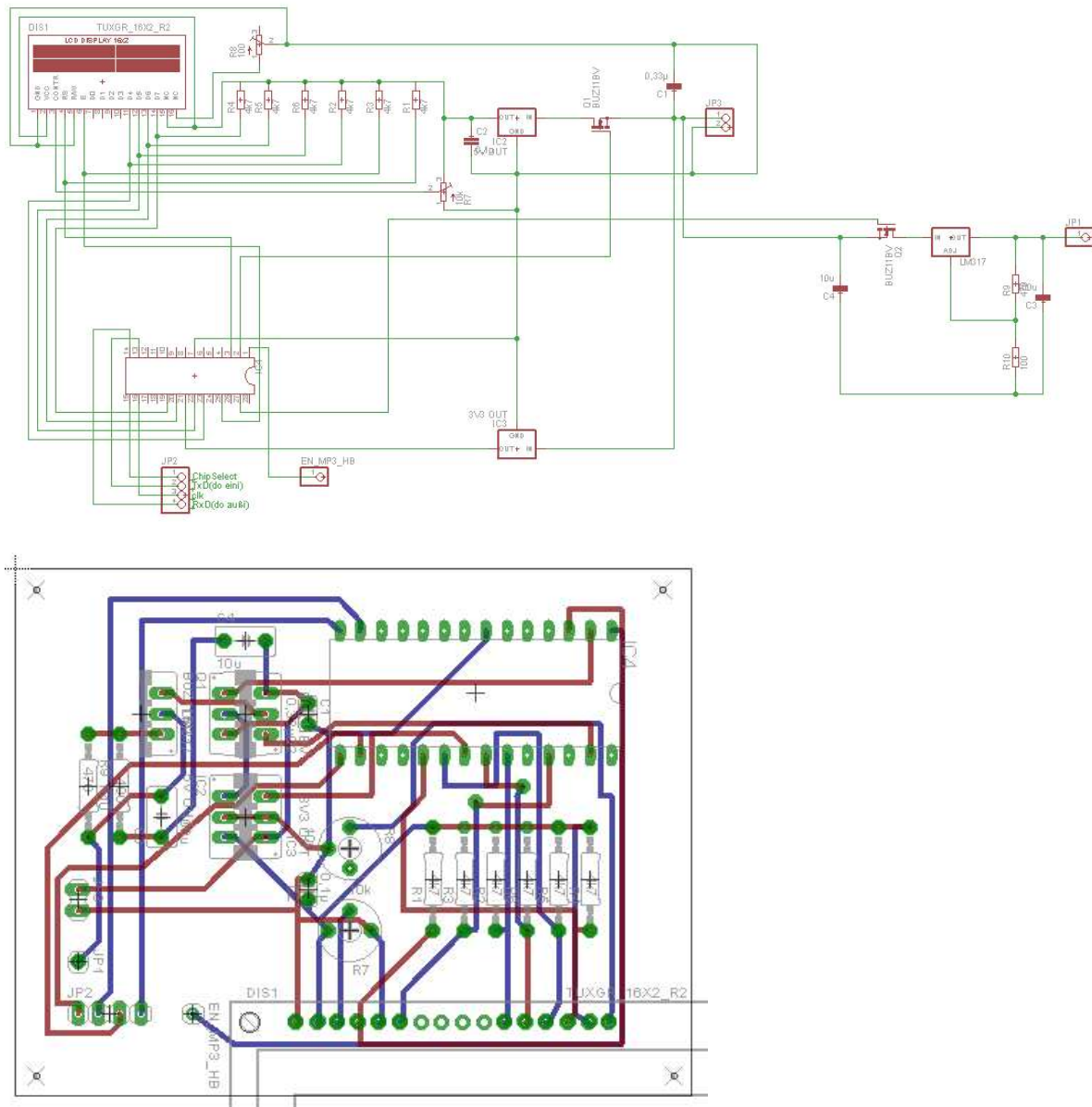
Spannungsversorgung: 2x 9V Block seriell (18V runtergeregelt auf 15V)

4 Potentiometer:

- Lautstärke Linker-Kanal
- Lautstärke Rechter-Kanal
- Kontrast vom Display
- Helligkeit der Displayhintergrundbeleuchtung

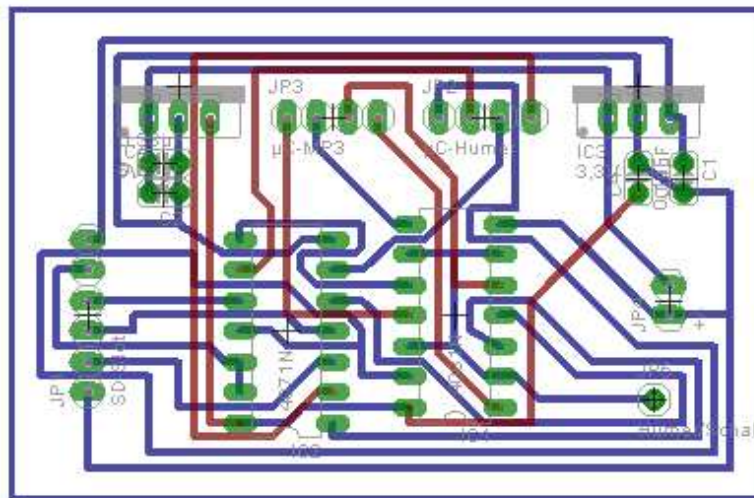
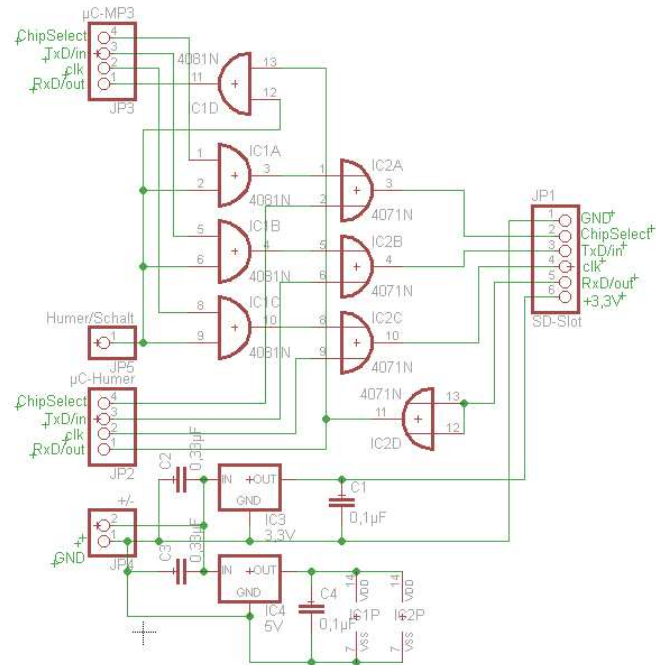
NLB 935 (Humerboard)

Hauptplatine



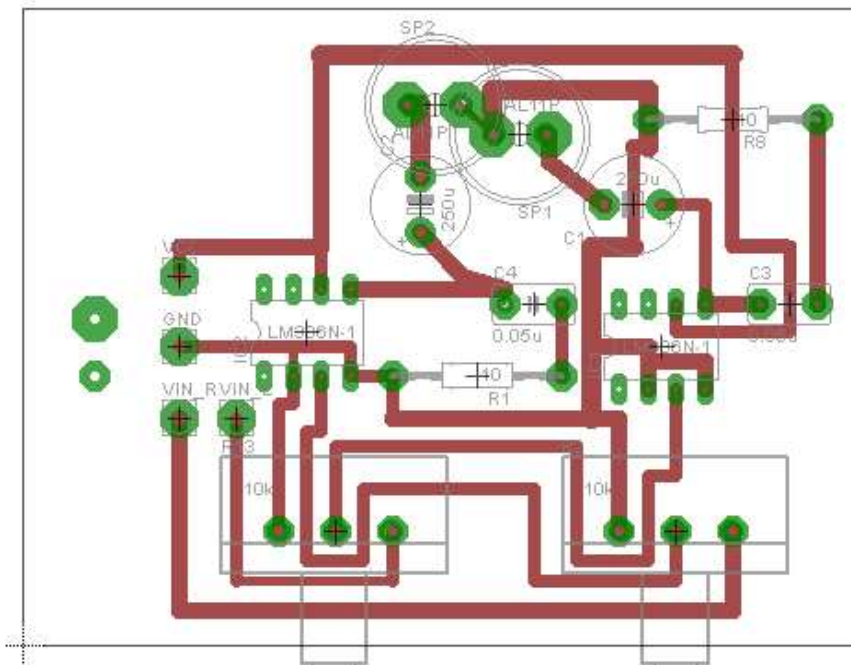
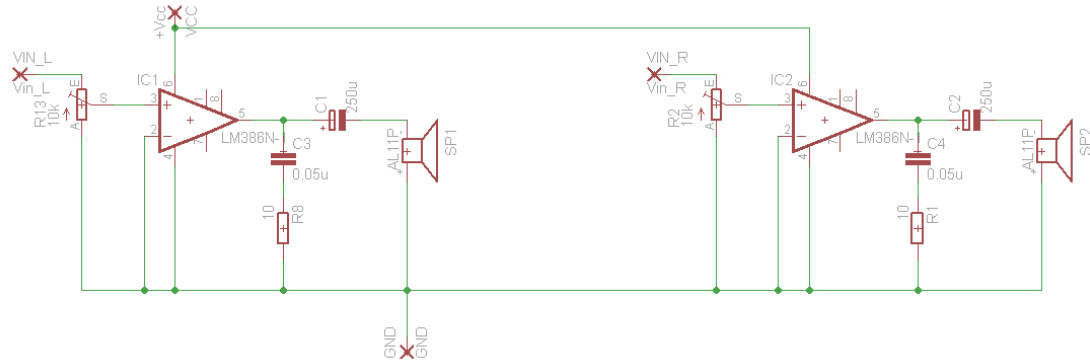
Als Display wird ein Powertip 1601-A verwendet und im 4-BIT Modus verwendet. Außerdem sind 3,3V und 5V Spannungsregler für das LCD und das Humerboard vorhanden, sowie Steckerleisten zum Verbinden des Humerboards mit der SD/MMC-Karte. Das LCD und der MP3-Player können über die Software durch zwei MOS-FET neugestartet werden.

Umschaltungsplatine



Die Umschaltung wird dazu verwendet um den gleichzeitigen Zugriff des Humerboards und des Mp3-Players zu unterdrücken, somit wird verhindert, dass die beiden Mikrocontroller sich gegenseitig beschädigen. Durchgeführt wird dies mit einem Und-(74LS09) und einem Oder-Gatter(74LS32). Hier werden die Pins MISO, MOSI, CLK, und CS des Humerboards und des Hama-MP3-Players zusammengeführt.

Verstärker



Verwendet werden zwei LM386N-1, jeweils einer für einen Kanal. Mit den Potentiometern kann man die Lautstärke für jeden Kanal extra einstellen.

LCD-Programmierung

4-Bit Modus

Eingebunden in ein eigenes Headerfile

Routinen:

Initialisierung, Byte ausgeben, String ausgeben, Delay, Cursor zur Anfangsposition setzen und Datenregister löschen

Code:

```
#include <REG935.H>
#include <stdio.h>

#define LCD_RS 0x01    //Bit0
#define LCD_E  0x04    //Bit2

#define SET_E(x) {if(x) P0|=LCD_E;else P0&=~LCD_E;}
#define E_Puls  {SET_E(1);SET_E(0);}           //Enable Impuls senden
#define SET_RS(rs){if(rs) P0|=LCD_RS;else P0&=~LCD_RS;}
#define LCD_Clr LCDwriteB(0,0x01)    //Textpuffer löschen
#define LCD_Home LCDwriteB(0,0x02)    //Cursor zum Anfang

void LCDwriteB(char, char);
void LCD_INIT(void);
void LCD_Delay(char);
void LCD_WriteS(char *);

void LCD_INIT(void)
{
    P0=0x00;        //alle Ports auf low

    LCD_Delay(15); //min 15ms warten wegen rset vom lcd

    P0=(0x30);      //8bit mod
    E_Puls;
    LCD_Delay(5);
```



```
P0=(0x30); //8bit mod
    E_Puls;
LCD_Delay(1);

P0=(0x30); //8bit mod
    E_Puls;
LCD_Delay(1);

P0=(0x20); //4bit mod
    E_Puls;
LCD_Delay(1);

LCDwriteB(0,0x28); //2zeilen 5*8 dot
LCDwriteB(0,0x08); //display aus
LCDwriteB(0,0x01); //display löschen
LCDwriteB(0,0x06); //cursor nach rechts kein display shift
LCDwriteB(0,0x0C); //Display ein
}

void LCD_Delay(char delay)
{
    int i;
    for(i=0;i<700*delay;i++)
    {
    }
}

void LCDwriteB(char rs,char c)
{
    P0=c&0xF0; //oberes Nibble
    SET_RS(rs);
    E_Puls;

    LCD_Delay(2);

    P0=c << 4; // unteres Nibble
    SET_RS(rs);
    E_Puls;
    LCD_Delay(2);
}

void LCD_WriteS(char* string) // String aufs LCD ausgeben
{
    int j=0;
    while(string[j]!=0)
    {
        LCDwriteB(1, string[j]);
        j++;
    }
}
```

Informationen darüber, wie man LCDs ansteuert, haben wir aus dem Internet und von Herrn Prof. Holzer erhalten.

Quellen: <http://www.sprut.de/electronic/lcd/>
Prof. Holzer

Ausgelagert in ein Headerfile

Problem: Bei der Initialisierung erhalten wir immer eine ERROR Meldung der SD-Karte

```
#include <REG935.H> |
#include "lcd_driver.h"
#include "SD_Driver.h"

void SD_Init(void);
void SD_Read(void);
void Send_Cmd(char);
void SPI_Init(void);

sbit CS=P2^4;

#define CMD0    (0x40+0)    /* GO_IDLE_STATE */
#define CMD1    (0x40+1)    /* SEND_OP_COND */
#define CMD8    (0x40+8)    /* SEND_IF_COND */
#define CMD9    (0x40+9)    /* SEND_CSD */
#define CMD10   (0x40+10)   /* SEND_CID */
#define CMD12   (0x40+12)   /* STOP_TRANSMISSION */
#define CMD16   (0x40+16)   /* SET_BLOCKLEN */
#define CMD17   (0x40+17)   /* READ_SINGLE_BLOCK */
#define CMD18   (0x40+18)   /* READ_MULTIPLE_BLOCK */
#define CMD24   (0x40+24)   /* WRITE_BLOCK */
#define CMD25   (0x40+25)   /* WRITE_MULTIPLE_BLOCK */
#define CMD41   (0x40+41)   /* SEND_OP_COND (ACMD) */
#define CMD55   (0x40+55)   /* APP_CMD */
#define CMD58   (0x40+58)   /* READ_OCR */

char CRC=0;
char value;

void SPI_Init()
{
    P2|= 0xC3;    // POrts auf High
    SPCTL=0xD6;   // SSIG Mstr auf High clk/64
}

void Send_Cmd(char cmd)
{
    SS = 0; // SPI Slave-Signal aktivieren
    SPDAT = cmd; // Wert schreiben
    while((SPSTAT & 0x80) == 0x00); // Warten bis Wert gesendet
    SPSTAT = 0xFF; // SPIF zuruecksetzen
    value=SPI_IN;
    _SS = 1; // SPI Slave-Signal deaktivieren
}
```

```
void SD_Init()
{
    int i;
    MOSI=1;
    CS=1;

    //74 clk cyrcles warten = 322µs
    for(i=0;i<500;i++)
    {
        //und ich warte^^
    }

    CS=0;
    Send_Cmd(CMD0); // sendet das Kommando Go-Idle-State
    if(value!=0x01)
    {
        LCD_WriteS("ERROR");
    }
}

}
```

Informationen zur SPI-Schnittstelle erhielten wir durch www.c51.de.

Quelle: http://c51.de/c51.de/Dateien/MEBA/LPC900_Kap10Erg_SPI.zip

SD-Kartenansteuern:

<http://www.mikrocontroller.net/topic/13434#new>

<http://www.ulrichradig.de/home/index.php/avr/mmc-sd>

http://elm-chan.org/docs/mmc/mmc_e.html

Hauptprogramm

```
#include <REG935.H>
#include "lcd_driver.h"
#include "SD_Driver.h"

void Init_Ports(void);
sbit EN_LCD;
sbit EN_HB_MP3
sbit EN_MP3;

void main()
{
    EN_LCD=0;
    EN_LCD=1;                //LCD RESET
    EN_HB_MP3=0;             //Humerboard greift auf die SD-Karte zu
    EN_MP3=0;                //Hama ist aus;
    Init_Ports();
    LCD_INIT();
    SPI_Init();
    SD_Init();
    // jetzt sollte das Verzeichnis der SD-Karte ausgelesen(*.mp3) werden und in einem Feld gespeichert werden
    EN_HB_MP3=1; // Hama kann auf die SD-karte zugreifen
    EN_MP3=1
    LCD_Delay(5000) //Hama einschalten
    EN_MP3=0;           //

    while(1)
    {
        // es sollte hier dann die erhaltenen Dateinamen ausgegeben werden.
    }
}

void Init_Ports()
{
    POM1=1;
    POM2=1;                //LCD OpDrain
    P2M1=0;                // SD KArt + enable eingänge QuBi
    P2M2=0;
    P3M1=1;                //INon für Play Next und Prev
    P3M2=0;
}
```

2.) verworfenes Projekt

Projektbeschreibung

Das Projekt sollte von einer SD-Karte unter Verwendung des VS1053 von VLSI Musikdateien (egal welches Format) dekodieren und auf Lautsprechern wiedergeben. Die Titel hätten, wie bei dem aktuellen Projekt, auf einem LCD angezeigt werden sollen.

Kenndaten

Display: Powertip 1601-A

MP3-Player-Baustein: VS1053

Mikrocontroller : AT89C51RD2 Atmel 8051

Spannungsversorgung: 12V

VS1053b - Ogg Vorbis/MP3/AAC/WMA/MIDI AUDIO CODEC

Features

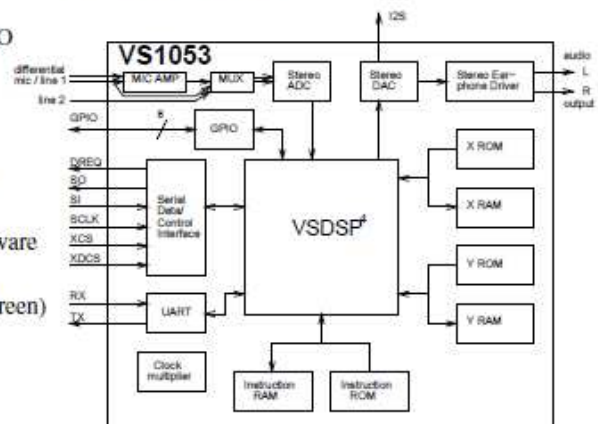
- Decodes Ogg Vorbis;
MPEG 1 & 2 audio layer III (CBR +VBR +ABR); layers I & II optional;
MPEG4/2 AAC-LC(+PNS),
HE-AAC v2 (Level 3) (SBR + PS);
WMA 4.0/4.1/7/8/9 all profiles (5-384 kbps);
WAV (PCM + IMA ADPCM);
General MIDI 1 / SP-MIDI format 0 files
- Encodes Ogg Vorbis with software plugin (available Q4/2007)
- Encodes IMA ADPCM from mic/line (stereo)
- Streaming support for MP3 and WAV
- EarSpeaker Spatial Processing
- Bass and treble controls
- Operates with a single 12..13 MHz clock
- Can also be used with a 24..26 MHz clock
- Internal PLL clock multiplier
- Low-power operation
- High-quality on-chip stereo DAC with no phase error between channels
- Zero-cross detection for smooth volume change
- Stereo earphone driver capable of driving a 30 Ω load
- Quiet power-on and power-off
- I2S interface for external DAC
- Separate voltages for analog, digital, I/O
- On-chip RAM for user code and data
- Serial control and data interfaces
- Can be used as a slave co-processor
- SPI flash boot for special applications
- UART for debugging purposes
- New functions may be added with software and upto 8 GPIO pins
- Lead-free RoHS-compliant package (Green)

Description

VS1053b is a single-chip Ogg Vorbis/MP3/AAC/WMA/MIDI audio decoder and an IMA ADPCM and user-loadable Ogg Vorbis encoder. It contains a high-performance, proprietary low-power DSP processor core VS_DSP⁴, working data memory, 16 KiB instruction RAM and 0.5+ KiB data RAM for user applications running simultaneously with any built-in decoder, serial control and input data interfaces, upto 8 general purpose I/O pins, an UART, as well as a high-quality variable-sample-rate stereo ADC (mic, line, line + mic or 2xline) and stereo DAC, followed by an earphone amplifier and a common voltage buffer.

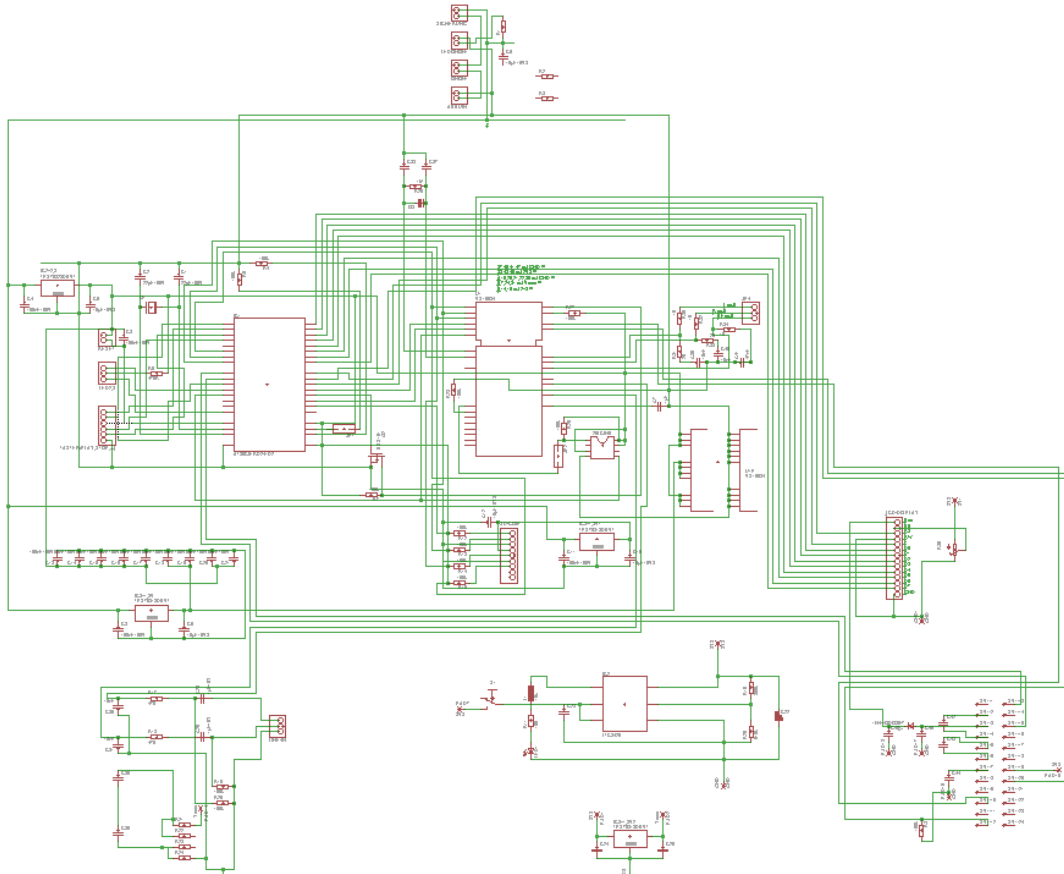
VS1053b receives its input bitstream through a serial input bus, which it listens to as a system slave. The input stream is decoded and passed through a digital volume control to an 18-bit over-sampling, multi-bit, sigma-delta DAC. The decoding is controlled via a serial control bus. In addition to the basic decoding, it is possible to add application specific features, like DSP effects, to the user RAM memory.

Optional factory-programmable unique chip ID provides basis for digital rights management or unit identification features.



<http://www.vlsi.fi/fileadmin/datasheets/vlsi/vs1053.pdf>

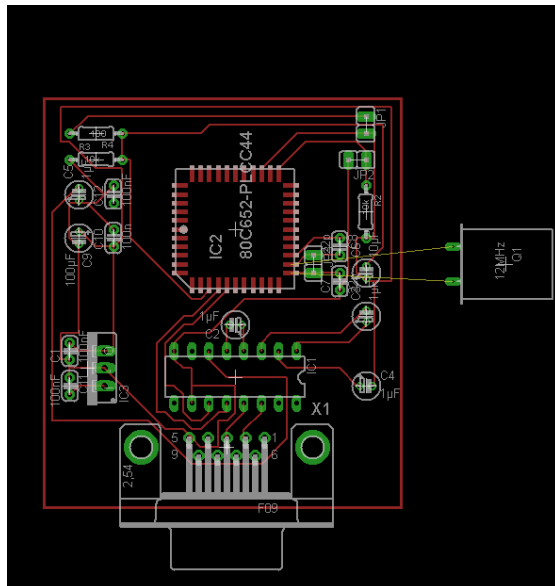
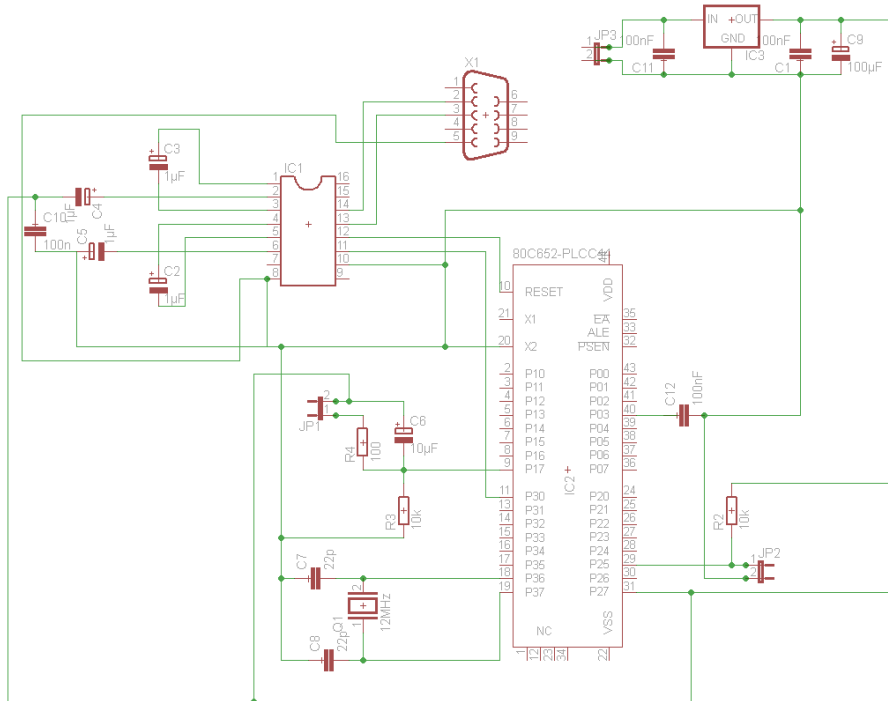
Hauptplatine



Das Layout hätte noch erstellt werden müssen.

Wir haben erst relativ spät bemerkt, dass der VS1053 inkl. Versand zirka 55€ gekostet hätte und der versand vier bis sechs Wochen gedauert hätte. Nach zirka 11 Projekttagen haben wir dieses Projekt aufgegeben.

Programmierboard für den AT89C51RD2



Wurde angefertigt, um den AT89C51RD2 zu flashen.

3.)Projekttagbuch

2. März Projektplanung

9. März Projektplanung

16. März Schaltung für MP3-Player (VS1053)
Bauteilliste

23. März Schaltung für MP3-Player

30. März Schaltung für MP3-Player

6. April Schaltung – Fehlerkorrektur
SD-Karten Spezifikation recherchiert

13. April Schaltung – Fehlerkorrektur
SPI Recherche

20. April Schaltung – Fehlerkorrektur
Schaltung und Layout für Programmierboard
Humerboard – Datenblatt studieren

27. April Schaltung – Fehlerkorrektur
Beginn Layoutentwurf für MP3-Player

4. Mai Layoutentwurf für MP3-Player

 Programmierboard geätzt

 LCD - Datenblatt studieren

11. Mai Layoutentwurf für MP3-Player

 Programmierboard gebohrt und Bauteile eingelötet

 Studieren des Datenblattes des LCDs

18. Mai Besprechung der Lage

 Aufgabe des Projekts, da der MP3-Player-Baustein zu teuer war

 Planung des neuen Projekts

25. Mai 1. Schaltung vom Verstärker

 LCD-Treiber (8-Bit)

 SD-Kartenansteuerung

 Schaltungsentwurf der Umschaltung

1. Juni LCD-Ansteuerung – Schaltung

 Umschalter geätzt und gebohrt

 LCD-Treiber (4-Bit)

 SD-Kartenansteuerung

8. Juni LCD-Ansteuerung – Schaltung

2. Schaltung vom Verstärker

Layout vom Verstärker

Bauteile vom Umschalter eingelötet

LCD-Treiber-Korrektur (Fertigstellung)

SD-Kartenansteuerung

15. Juni Layout vom Verstärker fertiggestellt

LCD-Ansteuerung – Schaltung – Korrektur, ätzen, bohren

Verstärkerplatine ätzen und bohren

Dokumentation, Präsentation entwerfen

SPI-Interface + SD-Kartenansteuerung

Löten der Verbindung zwischen MP3-Player und Umschaltung

22. Juni Voraussichtlich: Verstärkerbauteile einlöten

LCD-Ansteuerung: Bauteile einlöten

Präsentation

4.)Probleme

- Eingeschränkte Lieferbarkeit und die hohen Kosten für den VS1053 bzw. für den Versand des VS1053
- Wir haben 11 Tage verloren, da ein neues Projekt gestartet werden musste.

Wir haben das Problem mit der Initialisierung der SD-Karte nicht mehr beheben können und des Weiteren auch das nicht fertig stellen Hauptprogramm können. Da wir nicht auf die SD-Karte zugreifen konnten, konnten wir auch nicht das Inhaltsverzeichnis der SD-Karte auslesen. Außerdem gibt es noch Probleme mit der Hauptplatine, welche unsere Spannung von 15V auf 3V einbrechen lassen.

Somit haben wir unser Projektziel eine funktionsfähige „JukeBox“ zu entwickeln nicht erreicht.

5.)Resümee

Wir haben uns mit unserem Projekt zu viel vorgenommen, haben dies jedoch erst spät bemerkt. Dadurch haben wir gelernt, wie wichtig Arbeitsaufteilung im Team und Zeitmanagement ist. Wir sind jetzt auch routinierter im Umgang mit Datenblättern und vor allem mit Timing-Diagrammen. Aber die wichtigste Lektion die uns dieses Projekt lehrte ist, dass man sich genug Zeit nehmen muss, um ein Projekt gründlich und sorgfältig zu planen, damit spätere Probleme, wie Zeitdruck, von vornherein verhindert werden.

