Speech Aid with a Recordable Speech Output

Group D9

Anurag Singla (03*d*07026) (anuragsingla@iitb.ac.in)
B. Y. Vinay Kumar (03*d*07035) (vinayby@iitb.ac.in)
Vivek Mishra (03*d*07038) (vivek.mishra@iitb.ac.in)

Supervisors: P. C. Pandey, V. K. Tandon

Abstract

In this project, we have developed a device which can be used to record and playback speech, which can be used as a speech aid. It can be used as an audio player/voice recorder. The unit has been implemented using a Mixed Signal Processor MSP430F1610 of the TI's MSP430 family. A *miniSD*, 128 MB Flash card interfaces with the processor's SPI port. FAT16 has been implemented as a medium for file operations. The standard input to the device is the 4x4 matrix keypad, Nokia 3310 LCD being the display. The UART module is also made functional which can be used to further extend its I/O capabilities., eg. for the differently-abled people, some special input-pad could be designed and interfaced through UART, for whom the device could be used as a speech-aid.

1 Introduction

The aim of the project was to develop a portable device, with speech recording and playback capabilities, for the differently-abled persons. In EDL-I, the audio playback part was implemented and demonstrated. The previous design had some shortcomings, like — occasional unpredictable behaviour during playback, ad-hoc keypad implementation, lack of a file system, which made the process of storing files to SD card very tedious [5].

Most of the source code has been refactored with good documentation and a consistent coding style, which helped in removing all the shortcomings from the previous design. Additionally, a minimal FAT file system has also been implemented which only has functions tailored to cater to the project's need. Other music/audio files can also be dropped onto this filesystem through a PC, for playback.

Speech recording uses MSP's inbuilt 12-bit ADC. This samples the amplified speech input on channel 0 (A0) of the processor, and a DMA channel continuously stores the converted samples into two different buffers (without CPU intervention), which are then written to the FAT filesystem in a round-robbin fashion.

The inbuilt DAC has both 8-bit and 12-bit capabilities, and both are being used: 8-bit playback mode for 8-bit mono music files, and 12-bit mode for the recorded 12-bit voice data.

The UART module is also functional which enables furthur extensions to the device, eg. a specially designed input pad for differently-abled people can be interfaced to it. For now, it has been interfaced to PC to provide a simple utility to view all the content on the flash card on PC. A Block diagram of the system is shown in Fig. 1.

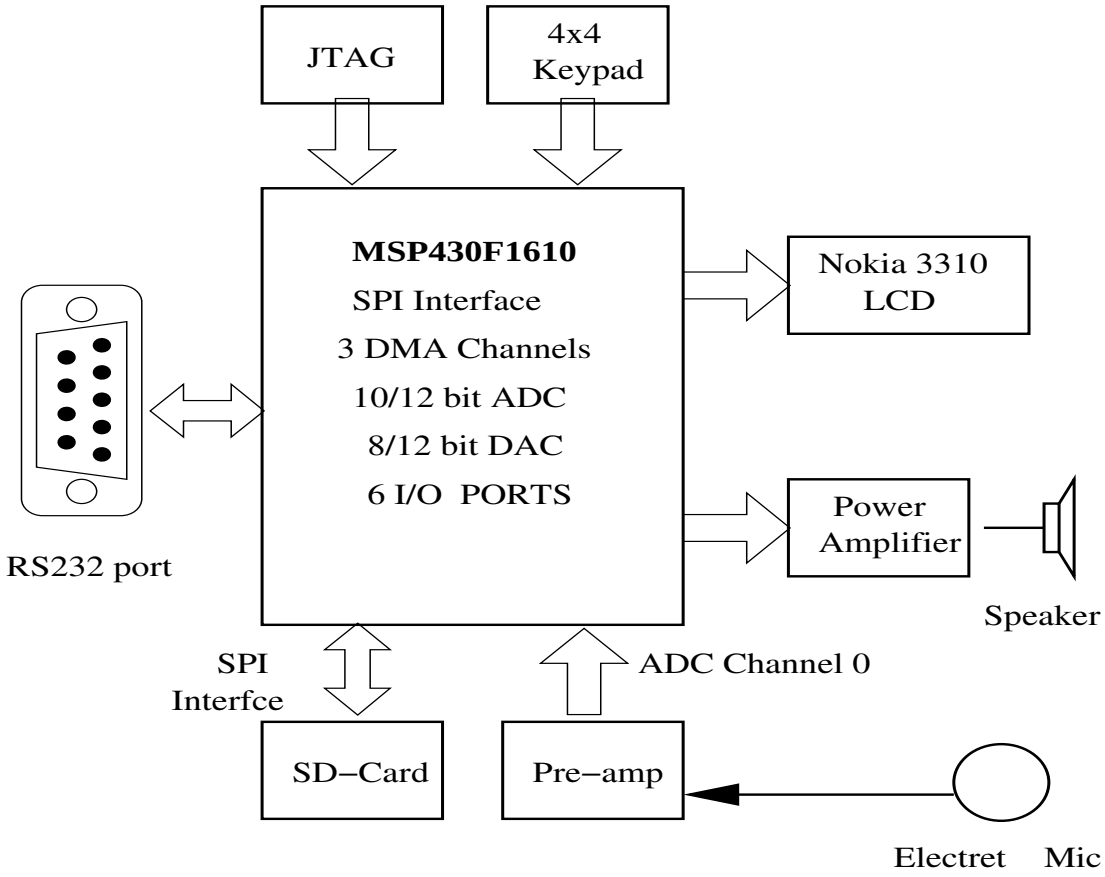


Figure 1: System Block Diagram

2 Circuit Design

The major components used in the circuit are the following.

1. MSP430F1610 microcontroller
2. Nokia 3310 LCD

3. 128MB miniSD Flash Card
4. 4x4 matrix keyad
5. TDA8551 audio amplifier
6. 74HC125 tri-state quad buffers
7. LM7805 voltage regulator
8. TPS7433 voltage regulator
9. TLV2252 Dual Opamp.
10. MAX232 level convertor.
11. SN74HC244 Octal Buffer.

2.1 User Interface

A 4x4 matrix keypad has been used for taking user input. The keypad has 8 pins, four of which are connected to the port 2 of the MSP430 (port 2 has interrupt capabilities, both rising and falling edge) and the other four are connected to the port 4, as shown in Fig. 2. The processor waits for a key press while it is in a low power sleep mode. The key-mapping

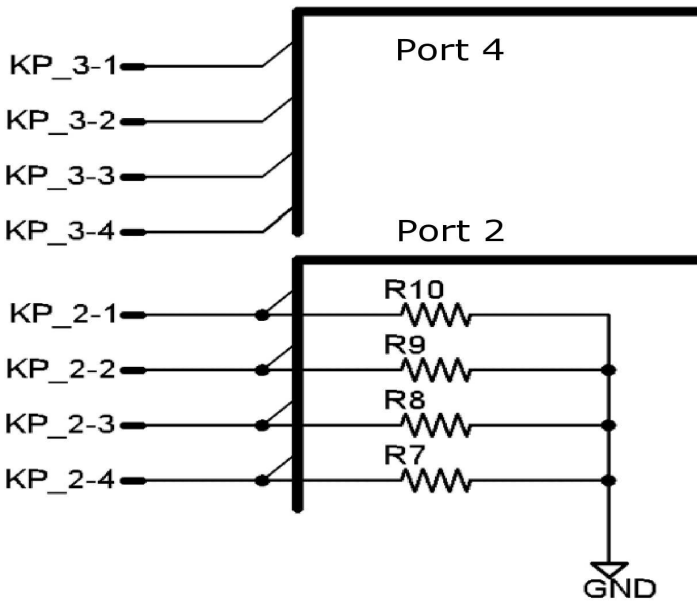


Figure 2: key mapping for the keypad

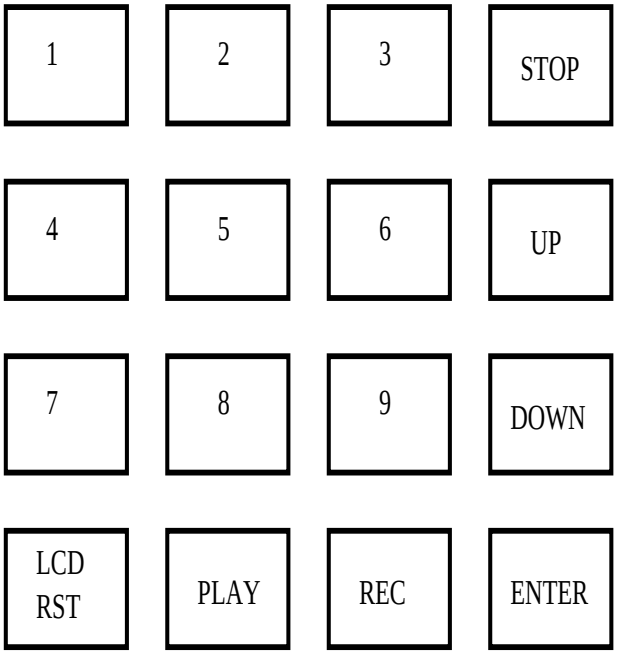


Figure 3: Key mapping for the keypad

for the keypad, the default user-interface, is as shown in Fig. 3.

When the system is switched on, it goes into sleep mode (MSP's Low power mode 0) and is woken up by pressing ENTER key. The next screen displays a listing of files on the SD card, four at a time. The Keys UP and DOWN can be used to scroll up or down the list, respectively. At any point, the record mode can be accessed by pressing the REC key, where the recording starts by pressing ENTER key and can be stopped by pressing STOP. The default recording time, before STOP is pressed, is nearly 150 seconds (corresponding to 5 MB of wav audio data, and a sampling rate of 16 Ksps). The LCD_RST key was used only as a makeshift, to reset the LCD display. To demonstrate the working of numeric keys, the sequence of keys $1-6-9$ has the effect of putting the device in standby.

3 Implementation Details

3.1 SD Card

The SD-card [1] is a removable flash storage device, specifically designed for portable applications. The communication between the microprocessor and the SD Card occurs byte wise through the Serial Peripheral Data Register (SPDR). On every byte shifted from the Master SPDR through SIMO, a byte is shifted back from the Slave SPDR to MISO in a circular fashion. The pin out for the SD card is as shown in Fig. 4.

SD protocol in SPI mode is a simple command response protocol. All commands are initiated by the master which in this case is the MSP430. The SD card responds to the

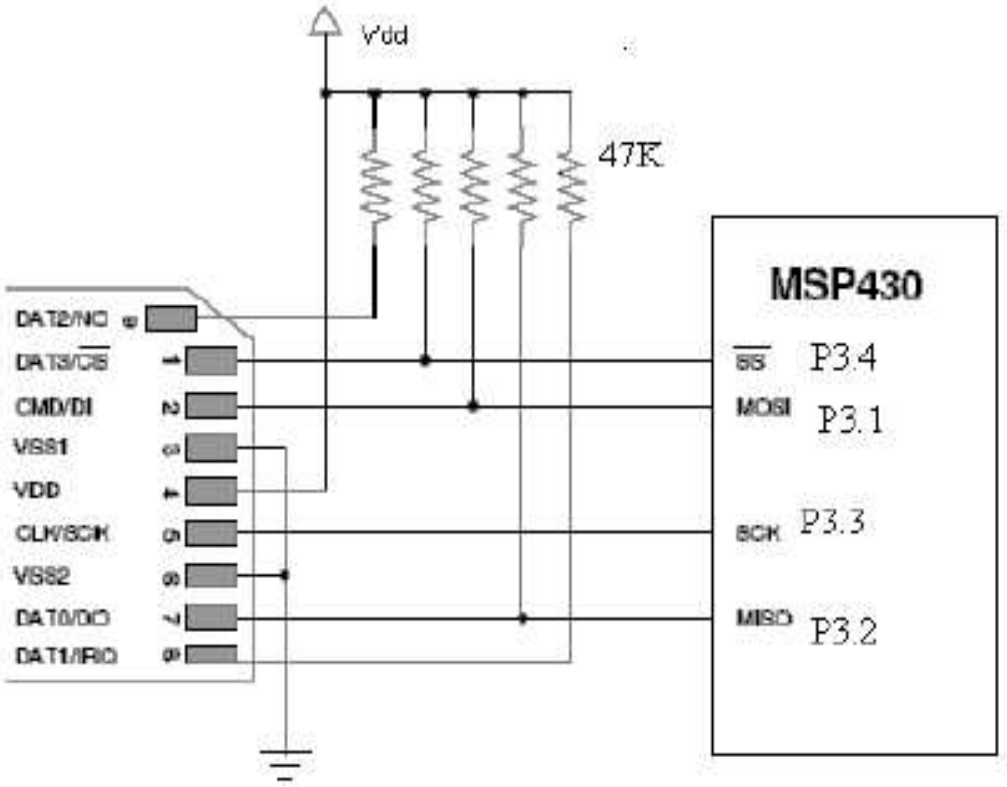


Figure 4: SD Card and MSP430 interfacing

command with a response frame and then, depending upon the command, may be followed by a data token indicating the beginning of a bulk data transfer or an error condition. For any response of the SD card the highest order bit(i.e. the 7th bit) designated as the start bit must be 0. A simple low-level block driver with calls for initialization, block reads, block writes, and a minimal set of status functions are implemented.

3.2 Nokia 3310 LCD

For the LCD panel, we used a Nokia 3310 [8], 84x48 pixels black and white. It works on a 3.3 V supply which is same as the main MSP power supply. The main reason for selecting this LCD was that it worked on the same supply voltage as the MSP. Moreover, this LCD is low-cost and easy to find. It also has low power consumption which makes it especially suitable for battery operated systems. The only external component used is a 4.7μF capacitor between V_{out} and the ground [8], as shown in Fig 5.

LCD uses port 6 (bit0...4). Lookup tables for all printable ASCII characters are being used to print characters and graphics. To get LCD working, one only needs to call the lcd_init () function followed by either the lower-level lcd_write_data(char) or any of the print character/string functions. The function invert () could be used to invert

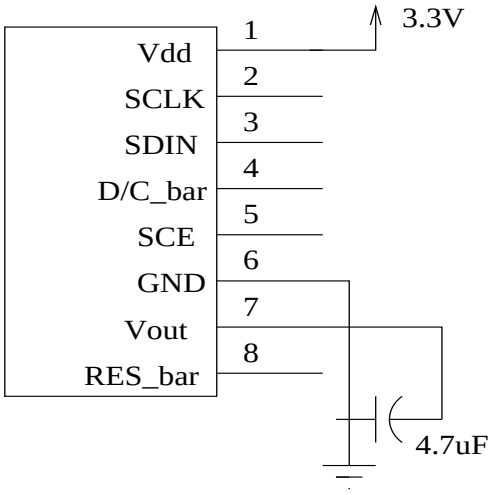


Figure 5: LCD backside view

(black-on-white or white-on-black) any displayed character. The lcd_init () takes care of initializing Nokia 3310 LCD, setting up mode (left-to-right, vertical addressing etc.,), operating bias etc. At a lower level a "byte", be it a command byte or a data byte, is sent by lcd_write_dorc() routine, which sends it out with 8 clock SCLK pulses.

3.3 FAT16 file-system

The previous implementation of the project did not have any file-system. It used SD Card in raw mode, which is a very involved procedure. The File Allocation Table (FAT) file system [6, 7] is considered relatively uncomplicated, and is consequently supported by virtually all existing operating systems for personal computers. So, FAT16 was implemented with minimal features to meet the requirements of the project.

The FAT16 functions implemented and used for the project are discussed below. To simplify things, we assume no sub-directories. All the files are stored in the topmost root directory. This assumption means that we can have no more than 512 files on the SD Card, a limitation imposed by FAT16 specification.

3.3.1 FAT16 functions

The following is a list of the main FAT operations. Refer to the source-code for more details of other internal functions.

1. *FAT_Init* reads the Master Boot Record (MBR) from the SD Card, and gathers information into relevant structures needed by other functions for any file operations.

2. FAT_CreateFile makes an entry of the file to be created in the root directory The requires the caller to pass a 'file' structure with name and 'next unused cluster' (using

an internal function). This then creates the file and sets the size to 0. (It is important to close the file after it's creation, otherwise it will not be visible). Refer appendix for more details.

3. *FAT_CloseFile* , in our implemention is used after a voice recording session. It updates the file's root directory entry with file size with inputs from the just concluded recording session.

4. *FAT_Readfile* and *FAT_ReadNo* read the files by name and number respectively. Since the application assumes all files to be audio (wav) files, the 'read'ing action defaults to file playback. This is different from the conventional 'read' functions provided by operating systems.

5. *FAT_DeleteFile* and *FAT_DeleteNo* can be used to remove the files by name and number respectively.

6. *FAT_ListDir* lists the files in the root directory on the gLCD.

As an example, the following is the sequence of operations performed to create and close a voice recording session–

- Fill the required attributes of the *file* structure.
- FAT_CreateFile then is used to create its entry in the filesystem. At this point the global file pointer points to this newly created entry.
- The the two buffers being supplied by DMA , are written to the SDcard using the current file pointers. A count of the number of bytes written is also maintained.
- After the recording session, the file is closed with FAT_CloseFile, which updates the file's size attribute.
- A subsequent FAT_ListDir shows this newly added file.

3.4 Microphone/pre-amplifier unit

For the purpose of recording a voice signal we have used an electret Microphone and a pre-amplifier stage, shown in Fig. 6. The electret microphone picks up the voice and converts it into an analog signal of nearly 2 to 3 mV. The analog voice signal is then amplified by an operational amplifier. We have used the TLV2252 [4] which is a low-voltage and low-power dual opamp, one of which is used for microphone signal amplification. TLV2252 can operate at 3 Volts with a low operating current. One of the reasons for choosing TLV2252 (an SMD package) is because of its low-power operation. The amplified analog signal is bandwidth limited by a simple RC filter to the required voice spectrum before it is input to the integrated A/D converter of the MSP.

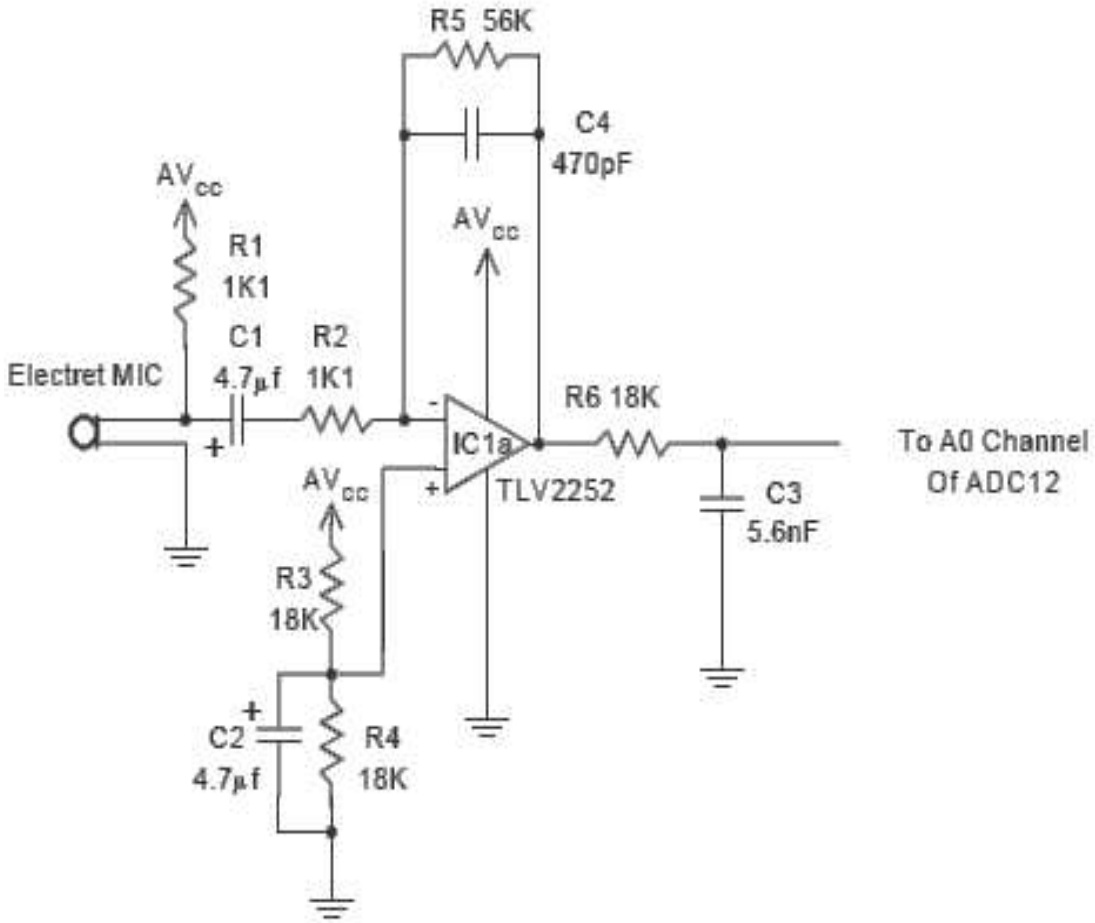


Figure 6: Microphone Pre-Amplifier circuit

The capacitor C4 across the feedback path also provides some high frequency roll-off. Technically this filter is the antialiasing filter and is required to avoid frequency aliasing of the input signal after sampling.

The output of the preamp stage gives the analog signal in the range $0-2.5$ V. The ADC of the processor is configured to a V_{ref-} of AV_{SS} (analog ground) and a internal V_{ref+} of 2.5 V. This is fed to the Channel A0 of the ADC controller. As the ADC clock (ADC12CLK), SMCLK [2] was selected which is sourced from a 8 Megahertz crystal. A sample rate of 16 kilo samples per second was set using observations by recording and playing back various frequencies of sine waves(below 3 kHz).

3.5 Audio Amplifier unit

A TDA8551 1W Bridge Tied Load amplifier [3] was used as the output amplifier stage. The circuit diagram is shown in Fig 7. Volume can be increased / decreased by giving positive

or zero voltage pulses respectively, through software or hardware at pin 1. The amplifier works on 5 V power supply. We have implemented the volume control through software. To increase the volume with positive pulses we used a tri state buffer so that after the pulse is applied, the pin is in the floating condition instead of zero volts.

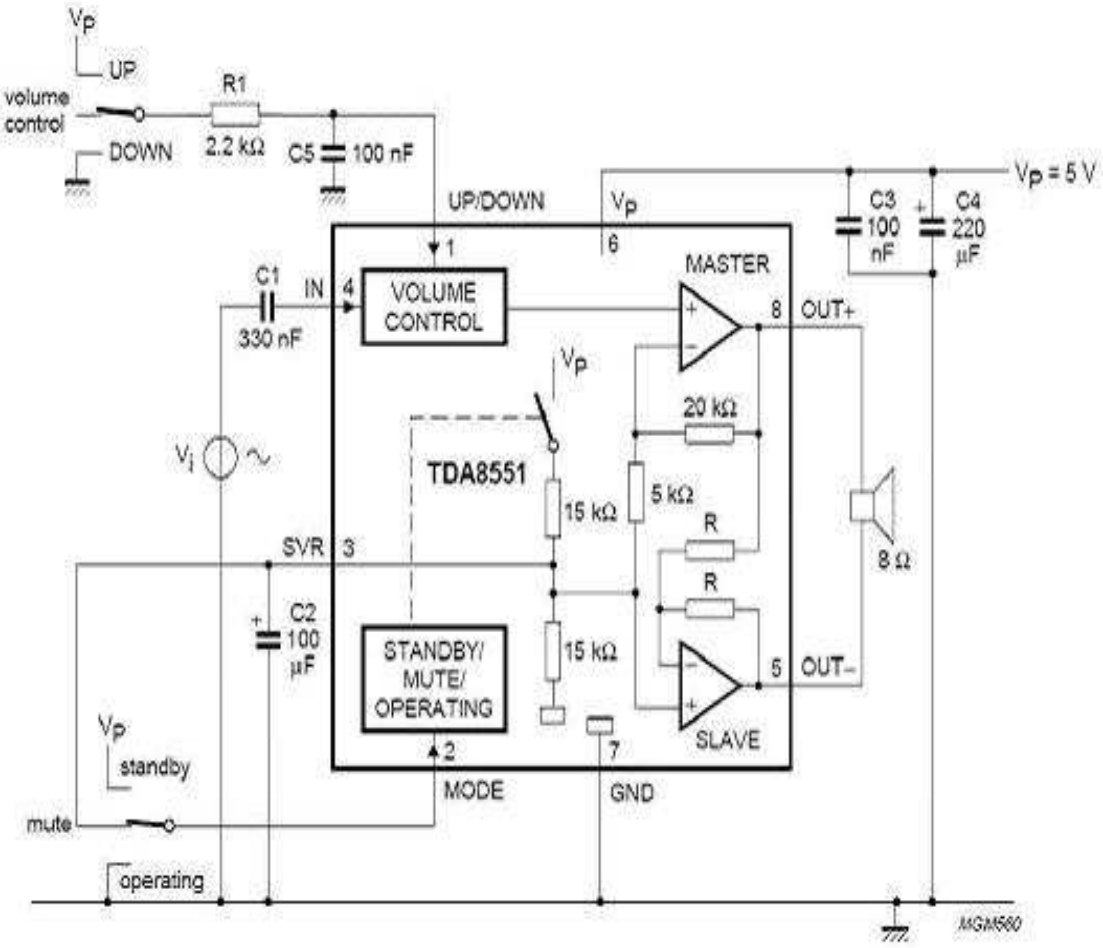


Figure 7: Audio amplifier circuit [3]

3.6 UART

In order to interface the UART of the microcontroller to an RS232 port of the PC a level converter was used to convert the data at RS232 levels to digital logic levels (of MSP). For this purpose the MAX232 chip was used. An SN74HC244 was used to convert the digital output level of MAX232(which is 5V) to 3.3V.
The UART is used to enhance the expansion of the unit for other purposes as explained below.

- Through the UART, the unit can be connected to a computer and it can accept commands serially.
- The unit can be made in-system programmable by programming the flash memory of the processor.
- A generic serial keypad may be connected.

. In our case, we have demonstrated the use of the UART by displaying a list of files when a command is given from the PC.

3.7 Power Supply Unit

The power supply consists of two voltage regulators.

1. LM7805 for +5 V
2. TPS7433 for +3.3 V

The SD card requires a power supply of 3.3 V and sinks current up to 40 mA. The LCD requires a power supply of 3.3 V.

3.8 Software aspects

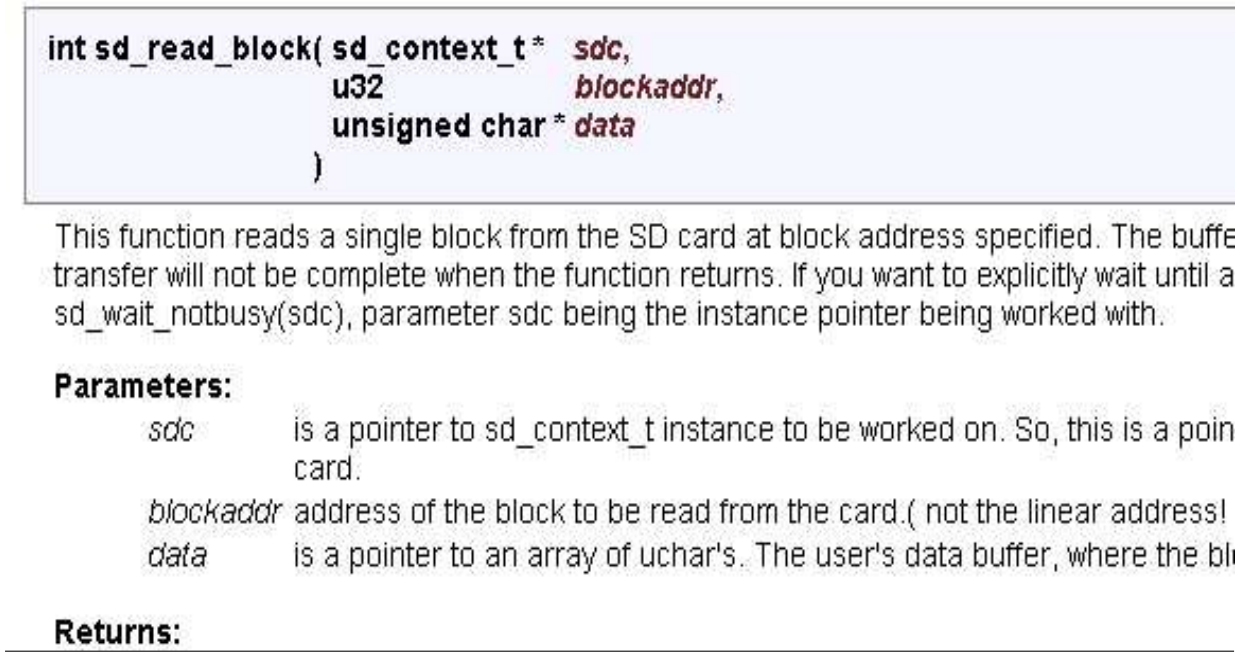
The lack of some consistent structure and documentation of the source-code made it difficult for one member of the project team to debug/modify the code written by the other. Understanding the importance of good documentation and a consistent coding style, most of the source code has been refactored. Doing this actually helped us in fixing some persistent bugs in the previous design. The code was also made self-documenting(a sample screenshot Fig 8) using a widely used tool called *Doxygen*, which is a tool that parses C source files and creates HTML pages based on the comment blocks that have been tailored with special doxygen tags.

4 Further Extensions

All the functionality required for this project is being provided by MSP430 (like ADC12, DAC12, DMA). A dedicated speech codec IC could be used in place of these inbuilt modules, which would greatly improve the quality of playback and recording. Specially designed keypads can be interfaced to the UART.

A new file system can be implemented which will overcome some of the limitations of the FAT16 file system as mentioned in the caveats section on the following page.

The device size can be considerably reduced by replacing all the ICs used on the PCB by their SMD version.



```
int sd_read_block( sd_context_t *  sdc,
                   u32             blockaddr,
                   unsigned char * data
                 )
```

This function reads a single block from the SD card at block address specified. The buffe transfer will not be complete when the function returns. If you want to explicitly wait until a sd_wait_notbusy(sdc), parameter sdc being the instance pointer being worked with.

Parameters:

sdc	is a pointer to sd_context_t instance to be worked on. So, this is a poin card.
blockaddr	address of the block to be read from the card.(not the linear address!
data	is a pointer to an array of uchar's. The user's data buffer, where the bl

Returns:

Figure 8: Self-documenting code, (partial HTML screenshot)

Caveats/Issues

1. For the MSP recorded file (i.e the file recorded using the ADC of the MSP), windows operating system shows the file size correctly but it does not allow that file to be copied to Windows, as a result an MSP recorded file cannot be renamed using Windows. The FAT16 specification is proprietary, and this minimal FAT16 implementation was done according to [6, 7]. Nevertheless, this implementation can read a Windows file. In order to rename an MSP recorded file (from the format rec_xx to a meaningful name), the following things can be done.

   (a) Access the SD Card in raw mode (as done in EDL-I [5]), and edit the FAT16 directory table to change the filename. This would involve copying that particular sector to PC, where the directory table containing the filename to be edited exists, and writing it back to the same location after editing the filename.

   (b) Upgrade the keypad for entering alphanumeric keys, so that the filename can be entered before saving a recorded session.

   (c) For the demonstration, the file names were hardcoded for each recording.

2. Though the file delete function in this FAT16 implementation works, some undefined behavior was observed during further read/write on SD Card after deleting a particular file.

3. Above issues with FAT16 can be overcome with a primitive raw filesystem imple-

mented with some simplifications like– contiguous files and a directory table with only filename, start address and filesize as the entries. But this will not allow copying files from a Windows PC. This can also be overcome by writing a custom software with the ability to interpret this new format.

A Hardware

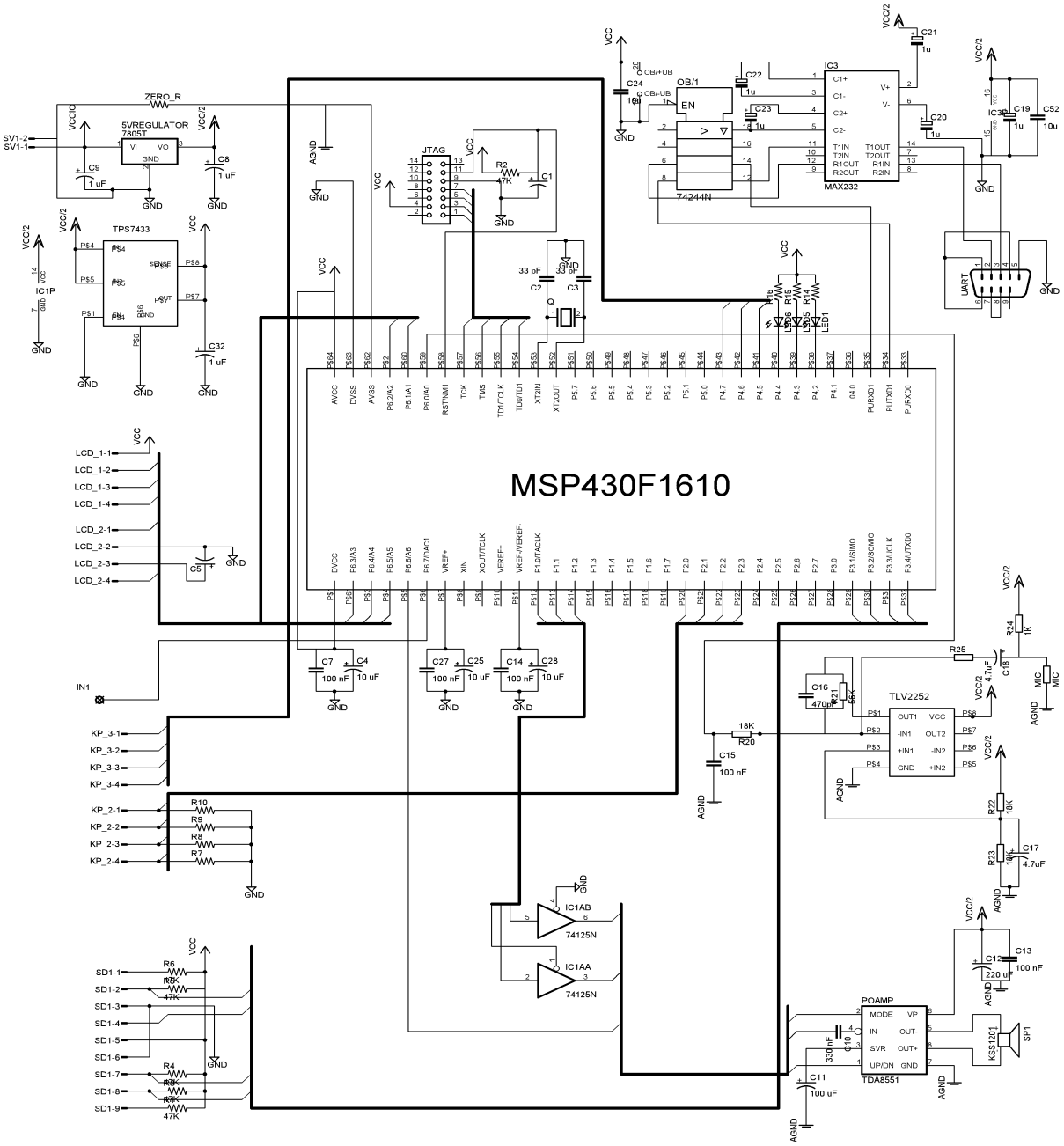


Figure 9: Schematic

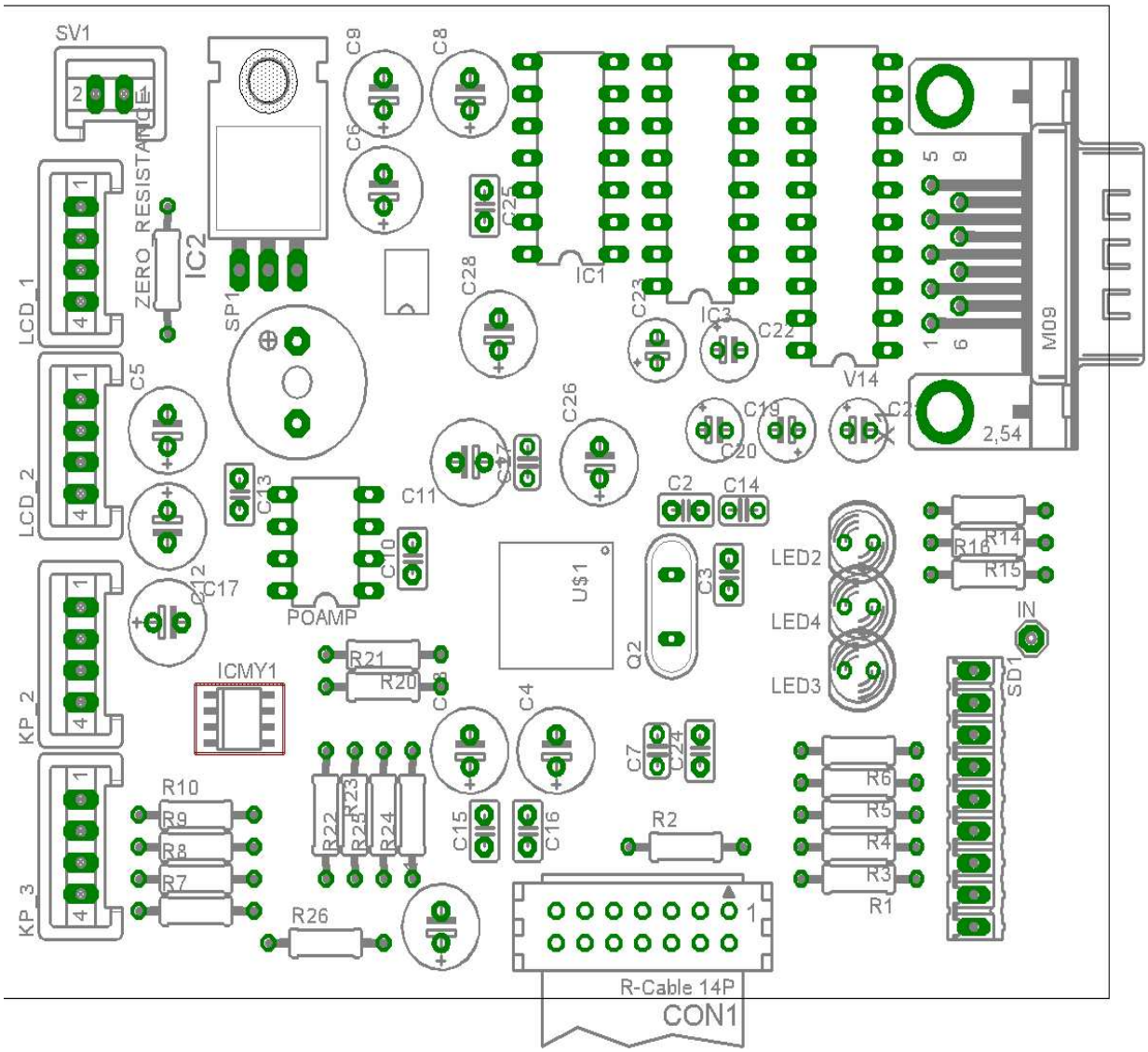


Figure 10: Component View

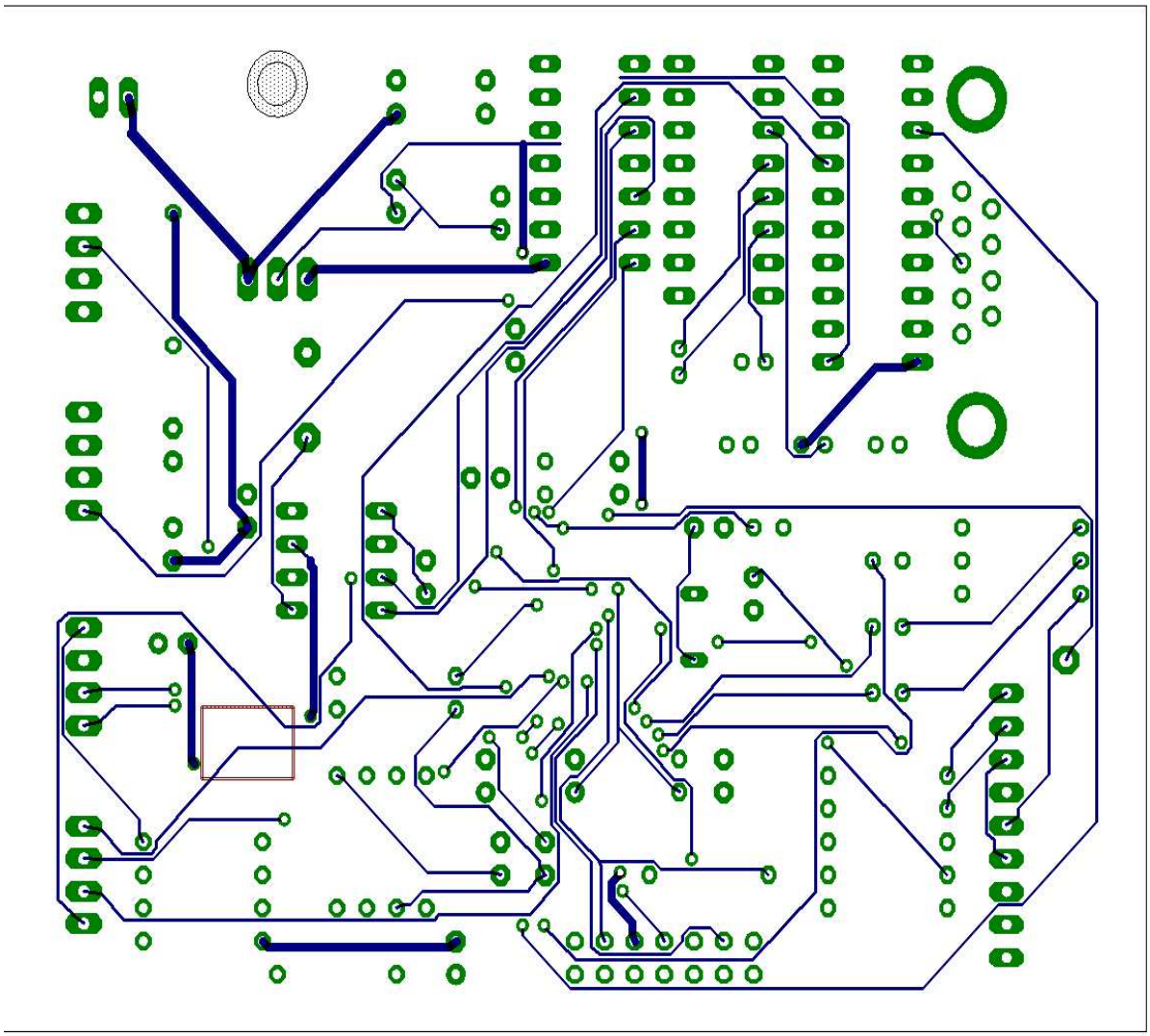


Figure 11: TOP layer

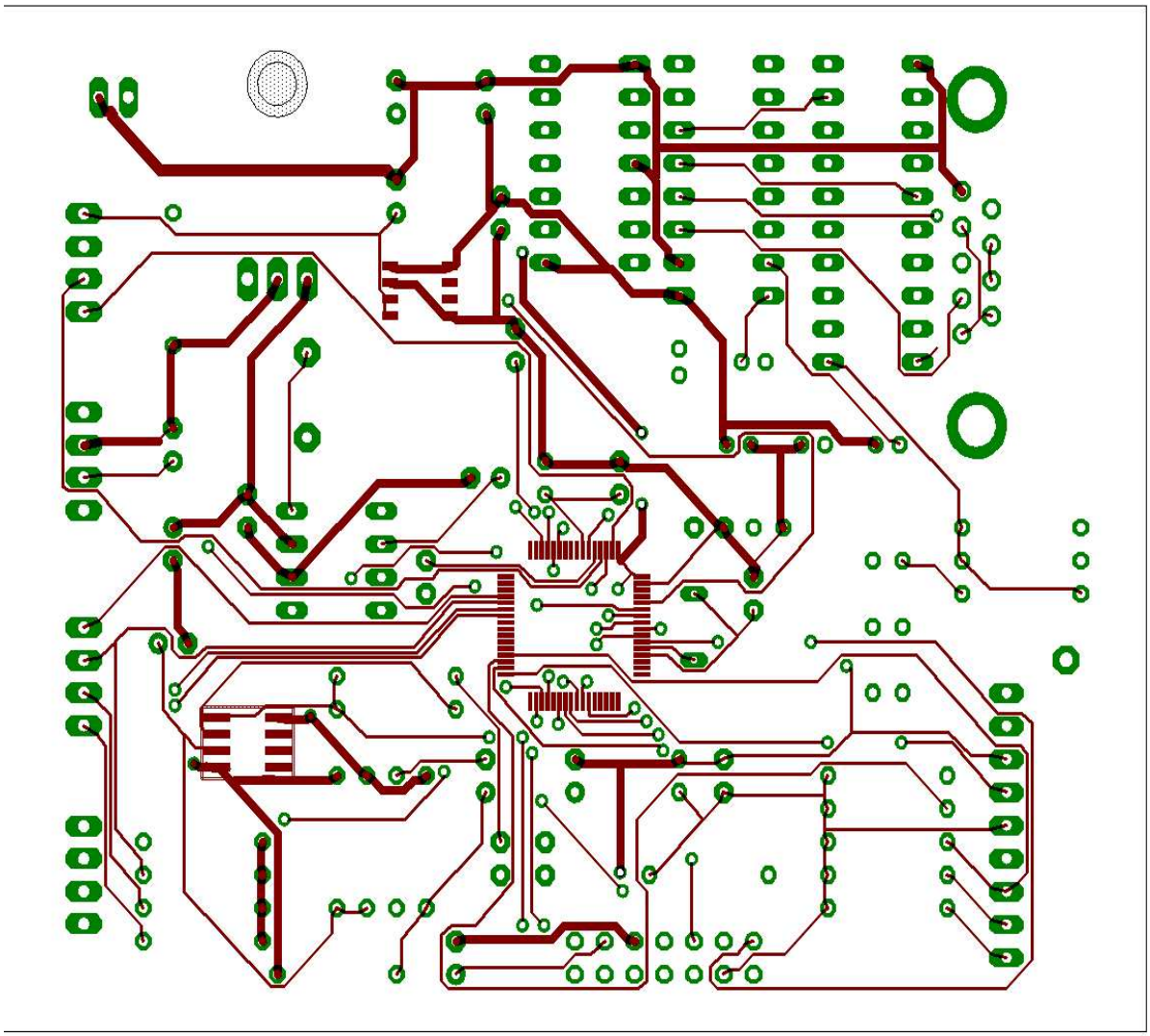


Figure 12: BOTTOM layer

References

[1] F. Foust, "Secure Digital Card Interface for MSP430", Department of Electrical and Computer Engineering, Michigan State University

[2] Texas Instruments, MSP430161x User Manual,http://www-s.ti.com/sc/psheets/slau049f/slau049f.pdf.

[3] NXP (philips), TDA8551 Datasheet, http://www.nxp.com/pip/TDA8551T_N1.html

[4] Texas Intruments, Datasheet(Rev. D), TLV2252, http://focus.ti.com/docs/prod/folders/print/tlv2252.html

[5] Anurag Singla, B. Y. Vinay Kumar, Vivek Mishra, EDL-I Project Report, “Speech aid with a Recordable Speech Output”, April 2006, http://sharada.ee.iitb.ac.in/ edlab/edl06s/edl06s_d09.pdf.

[6] Wikipedia.org, "FAT filesystems", http://en.wikipedia.org/wiki/File_Allocation_Table.

[7] Microsoft, white paper, “FAT16 filesystem”, www.microsoft.com/technet/prodtechnol/windowsserver2003/library/TechRef.

[8] www.microsyl.com, Datasheet, Nokia 3310 LCD datasheet.

[9] www.olimex.com/dev, Board Schematic, JTAG for MSP430.