Speech Aid with Recordable Speech Output

Group D9

Anurag Singla (03d07026)<anurag@ee.iitb.ac.in> BY Vinay Kumar (03d07035)<vinayby@ee.iitb.ac.in> Vivek Mishra (03d07038)<vivekm@ee.iitb.ac.in>

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

Abstract

In this project, we have developed a playback device. The unit has been implemented using a Mixed Signal Processor MSP430F1610 of the MSP430 family. The MSP430 can be programmed using a JTAG interface. An SD-Flash card of 128MB has been interfaced with the MSP430 using SPI interface. As of now, only raw data in the form of .wav files has been written onto the flash card. The flash card can be programmed using a computer using a standard USB card reader. The processor has been equipped with a Nokia LCD for better user interface, and a minimal keypad for playing the tracks. At the output end an audio amplifier takes the input from the inbuilt DAC of the MSP430 and directs the signal to the speaker. The speaker volume can be controlled through software.

1. Introduction

The inspiration behind this project has been to develop a portable playback device which can record data from the PC, store it in a reliable and easy to interface medium and play it back whenever necessary.

The complete block diagram of the unit is shown in Fig 1. The output amplifier needs a 5 V supply and the rest of the circuit works on 3.3 V supply. The MSP430 was chosen for this application since it has some matchless features listed below:

- A12 bit ADC
- Inbuilt 12 bit DAC(2 Channel)
- Serial Interfaces (USART/SPI/I2C)
- 3 DMA channels
- 5 KB RAM and 32KB ROM
- 6 Ports (with Port 1 and 2 with interrupt capability on each pin)
- USART0 and USART1 (which also support SPI or I2C[™] interface)
- Low power dissipation

The MSP was programmed using a JTAG board and using IAR embedded Workbench. The schematic for the JTAG board has been included in Appendix C. The IAR workbench is a highly user friendly tool with in system programming and debugging capabilities using the FET debugger through JTAG. All the codes used in the project are written in C.



Figure 1. Block Diagram for the unit

Since the MSP430 has an inherent support for an SPI interface, SD card was the most suitable choice for this application. Moreover, an SD card works on a 3.3V power supply. The data transfers between the SD card and the MSP430¹ are managed by DMA channels. Two DMA channels are utilized in this process since SPI requires that a byte be sent for a byte that is received by the MSP. The 128MB SD card is divided into pages of 512 Bytes.

All the tracks stored in the flash card are displayed on the LCD when the device is powered up. LCD is an effective way to flip through the tracks and also adds to the aesthetic looks of the unit. The keypad just helps the user to browse through the track listing and play the appropriate item.

The output stage constitutes an audio amplifier. The amplifier derives its input from the DAC channel-0. Once the DMA transfer from the flash card is completed, the entire block of 512 Bytes is stored in a buffer in the RAM and is then transmitted byte by byte

¹ ref. TI's application note and one another good one by F. Foust, \Secure Digital Card Interface for the MSP430", Application Note, Dept. of Electrical and Computer Engineering, Michigan State University, 2004.

to the DAC channel. The amplifier collects the signal from the DAC channel and transmits it to a speaker.

2. Design Approach

a. Hardware Design

JTAG Programmer

A standard JTAG programming board was borrowed from a previous batch. All the schematics for making the board have been included in the Appendix.

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. 2

The SD-card connector was not available, so a card holder from a USB card reader was removed and was soldered to the general purpose board. We used a miniSD card, which along with its adapter is equivalent to a normal SD card.



Fig2. SD card connections with MSP

LCD

The LCD panel we used was a Nokia 3310, 84x48 pixels black and white [2]. 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 4.7μ F capacitor between V_{out} and the Ground (refer Fig 3.). This capacitor is essential for the working of the LCD.



Fig 3. Back View of the LCD

Audio Amplifier

A TDA8551 1W Bridge Tied Load amplifier [5] was used as the output amplifier stage. The circuit diagram is shown in Fig 4. 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.

Power Supply Unit

The power supply consists of two voltage regulators.

- 1. LM7805 for +5 V
- 2. TPS7430 for +3.0 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. Thus the entire unit can work on a single 9 V battery. A 9 V adapter is being used for now.



Fig 4. Audio Amplifier

b. Software Design

SD Flash Card

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 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.

The SD card controller is implemented as a simple low-level block driver with calls for initialization, block reads, block writes, and a minimal set of status functions. The function sd_send_command () is the single most important function. It uses the MSP430's onboard SPI controller for sending and receiving bytes. A more detailed description on the SD functions has been included in the Appendix A.

Nokia 3310 gLCD

Port 6(bit0...4) is being used for LCD. Lookup tables are 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 (black-on-white-on-black) any displayed character.

E.g. to print a character "a" at the current cursor position(re-adjustable with $lcd_gotoxy(x, y)$ function), we send bytes 5 bytes 0x7F, 0x48, 0x44, 0x44, 0x38 from the lookup table, to clear LCD display we send 84x6 = 504 bytes of 0x00 one after the other.

The lcd_init () takes care of initializing nokia3310 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.

<u>KEYPAD</u>

The keypad consisting for 4 keys has been connected to the Port 2 of the MSP430. All pins on the Port 2 have interrupt capability; hence any key press can be detected easily, by just back tracking the port pin which has been interrupted. The current setup also allows for expandability to a 4x4 keypad if needed, only with a small software support.

Working with SD card and PC

The **usb-storage** class module along with **sd/mmc** driver in Linux Kernel enables us to mount the SD card as a normal block device (like a hard-disk). A file is created in the devices tree with an entry /dev/sda (or, sdb etc).We can either format it with some file system (say fat16) or access it in raw mode using the fopen() or fread() functions, now that the device is equivalent to a file.

We used a standard unix utility `dd` to copy blocks of 512 bytes, to and from the card. Command reference:

```
$ fdisk -1
#lists all the devices/partitions. Can subsequently be
#used to partition the flashcard
$ dd if=somefile.wav of=/dev/sda bs=512 count=101
#here the arguments: if=inputfile, of=outputfile
#bs=blocksize and count tells how many such blocks are to
#be transferred, we can also <u>skip</u> some blocks, or <u>seek</u>
# some position before transfer.
```

3. Circuit Design Specifications

Appended below is a list of major components of the circuit along with the labels in the schematic which is given later

- a. MSP430F1610 microcontroller
- b. Nokia 3310 LCD
- c. 128MB miniSD Flash Card
- d. A minimal 4 keys-pad
- e. TDA8551 audio amplifier "IC3"
- f. 74125N tri-state quad buffers "IC1A"
- g. LM7805 voltage regulator "IC1B"
- h. TPS7430 voltage regulator
- i. Decoupling capacitors "C7 and C4"

The overall circuit schematic is shown in Figure 5. The PCB layouts are included in the Appendix B

4. Results

Speech data which has been stored on a computer is transferred to the SD card through a USB card reader. The LCD shows the status of the SD card when the unit is powered up. It retrieves the file names and displays the track listing on the LCD. The user can select any track which is then played out using the speaker. As it stands, we have only been able to write data in the raw format on the SD card. A file system has not been implemented due to time constraints.

5. Future Scope

The following features can be implemented as a next stage

- A proper file system is to be implemented.FAT16 can be easily implemented and it is a common file system with Flash Card based applications. An even better open source filesystem Journaling Flash File System (JFFS) can be used.
- The unit can be extended to act as a portable MP3 player by adding an MP3 decoder chip to the output of the DAC.
- Write a good application note for MSP430 usage with reference to this project.

Appendix

A. SD-Flash card

Card Initialization

The initialization process is very a specific and an involved process. Card initialization starts by setting the SPI clock to 400 kHz. Next, at least 74 clocks must be issued by the master (i.e. MSP430) before any attempt is made to communicate with the card. This allows the card to initialize any internal state registers before card initialization proceeds. Next, the card is reset by issuing the command CMD0 while holding the SS\ pin low. This both resets the card and instructs it to enter SPI mode. Next, the card is continuously polled with the commands CMD55 and ACMD41 until the idle bit becomes clear, indicating that the card is fully initialized and ready to respond to general commands. Next, the command CMD58 is used to determine if the card supports the processor's operating voltage. CMD58 returns a bit field containing the allowed operating voltage ranges, typically between 2.7V and 3.6V. It is assumed that the MSP430 is using a voltage supply of 3.3V. Finally, the SPI clock is set to the maximum rate allowed. The initialization sequence is implemented by the function sd_initialize().

Block Read

The block read command is a bulk data command. The command response is followed by a delay, then followed by a "start of block" token, and then followed by the actual block itself. To enable fast block transfers the block read function uses Direct Memory Access (DMA) channels available on the MSP. First, the command is sent and the response is

received. Then, the function waits until the start token is received. When it is received, the function starts a DMA transfer. Since SPI requires that a byte be sent for a byte to be received, two DMA channels are used to complete the transfer.

The function sd_read_block() implements the block read. The function will return immediately and normal program execution can continue while the block transfer finishes. sd_wait_notbusy() can be used to synchronously wait for any pending block transfers to finish.

Block Write

The block write is similar to the block read function in that it uses a DMA transfer and also starts with a data token. However, since no bytes need to be received during the block transfer, the block transfer only requires one DMA trigger. The destination for the DMA transfer is the USART receive buffer, U0RXBUF. The source for the DMA transfer is the data buffer. The source is set to byte-wide, with an increment. The count is fixed at 512, the default block size for a typical SD card. The function sd_write_block() implements the block write. The function will return immediately and normal program execution can continue while the block transfer finishes. sd_wait_notbusy() can be used to synchronously wait for any pending block transfers to finish.

B.PCB Layout



Figure 5: PCB Schematic



Figure 6 . PCB layout



Figure 7. Top Layer



Figure 8. Bottom Layer

C. JTAG Programming Tool

The JTAG emulation [3] is a high speed in system programming (ISP) tool. We borrowed the board from a previous batch. The tool allows real time debugging when coupled with IAR Embedded Workbench.



References

- [1] F.Foust, "Secure Digital Card Interface for MSP430", Department of Electrical and Computer Engineering, Michigan State University
- [2] <u>www.microsyl.com</u>, for Nokia LCD pinout.
- [3] <u>www.olimex.com/dev</u>, for JTAG Board Schematic
- [4] MSP430161x User Manual http://www-s.ti.com/sc/psheets/slau049f/slau049f.pdf
- [5]TDA8551 datasheet <u>http://www.semiconductors.philips.com/pip/TDA8551T_N1.html</u>