

DESIGN OF AUDIO PLAYER AND RECORDER USING STM32F4 DISCOVERY BOARD

A Thesis report submitted in partial fulfilment of the requirements for the

award of the degree in

Master of Technology (Dual Degree)

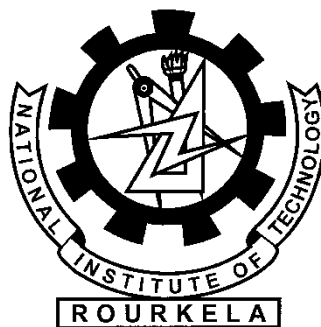
In

“VLSI Design and Embedded Systems”

Vimal Kumar

(Roll No. 710EC2120)

May, 2015



Department Of Electronics & Communication Engineering

National Institute Of Technology, Rourkela

DESIGN OF AUDIO PLAYER AND RECORDER USING STM32F4 DISCOVERY BOARD

A Thesis report submitted in partial fulfilment of the requirements

for the award of the degree in

Master of Technology (Dual Degree)

In

“VLSI Design and Embedded Systems”

By

Vimal Kumar

(Roll No. : 710EC2120)

May, 2015

Under the guidance of

Prof. A.K Swain



Department of Electronics and Communication Engineering

National Institute of Technology

Rourkela-769008



DEPARTMENT OF ELECTRONICS AND
COMMUNICATION ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY, ROURKELA
ODISHA, INDIA -769008

CERTIFICATE

This is to certify that the thesis entitled “**Design of Audio Player and Recorder on STM32F4 Discovery Board**”, submitted by **Vimal Kumar (Roll No. 710EC2120)** in partial fulfilment of the requirements for the award of **Master of Technology (Dual Degree)** in “**VLSI Design and Embedded Systems**” during session 2014-2015 at National Institute of Technology, Rourkela.

The candidate has fulfilled all the prescribed requirements. The Thesis is an authentic work, based on candidates’ own work.

To my knowledge, this thesis is up to the standard required for the award of a Bachelor of Technology in Electronics and Instrumentation Engineering & Master of Technology in VLSI Design and Embedded Systems (Dual Degree) degree.

Place: Rourkela

Prof. A.K Swain

Assistant Professor

Department of Electronics and Communication Engineering

National Institute of Technology

Rourkela -769008

ACKNOWLEDGEMENTS

I wish to express my heartfelt gratitude to my supervisor **Prof. A.K Swain**, Assistant Professor, Department of Electronics & Communication Engineering, National Institute of Technology, Rourkela for his valuable support, guidance, and time throughout my project. I also appreciate the freedom provided to me by provided by **Prof. Ayas Kanta Swain** to explore new ideas in the field of my project.

I am also grateful to **Prof. Sunil Kumar Sarangi**, Director, National Institute of Technology, Rourkela for providing me with outstanding facilities in the institute for my research.

I would also like to thank Prof. K. K Mahapatra, Head of Department, Department of Electronics & Communication Engineering, National Institute of Technology, Rourkela for providing facilities during this project work.

Finally, I want to thank my parents and the almighty god for their backing, without which this would not have been conceivable.

Vimal Kumar

Roll No. 710EC2120

TABLE OF CONTENTS

Topics	Page Number
ABSTRACT.....	1
LIST OF FIGURES	2
LIST OF TABLES.....	3
1 INTRODUCTION	5
1.1 Project Overview.....	5
1.2 Organisation Of Thesis.....	6
2 ARM MICROCONTROLLERS AND TOOLCHAINS.....	8
2.1 Reduced Instruction Set Computer (RISC).....	8
2.1.1 Characteristics:	8
2.1.2 RISC vs CISC.....	9
2.1.3 Advantages of RISC over CISC.....	9
2.1.4 Disadvantages of RISC over CISC.....	10
2.2 ARM Processors.....	10
2.2.1 History.....	11
2.2.2 Operating System Support.....	11
2.2.3 ARM Cortex M4.....	11
2.2.3 STM32F4 Discovery board	12
2.3 Keil uvision MDK pro Tool chain	14
3 FILE FORMATS	18
3.1 Wave Audio File Format.....	18
3.2.1 Wave file header.....	19
3.2.2 Format Chunk.....	19
3.2.3 Data chunk.....	20
4. COMMUNICATION PROTOCOLS	23
4.1 Inter Integrated Circuit (I2C) Protocol	23
4.2 Universal Serial Bus (USB)	26
4.3.1 USB Communication	27
4.3.2 USB Advantages and Disadvantages.....	28

5	FAT 32 FILE SYSTEM.....	30
	5.1 FAT 32 File System	30
6	AUDIO PLAYBACK AND RECORDING APPLICATION	34
	6.1 Application Overview	34
	6.2 Audio Playback Application	35
	6.3 Audio Record Application	37
7	CONCLUSION AND FUTURE WORK	40
	7.1 Conclusion	40
	7.2 Future Prospects	40
8	REFERENCES	42

ABSTRACT

With the advancement in semiconductor technology, scope for development of embedded systems has increased manifolds. New processors with improved computing capabilities and low power consumption have further accelerated the developments in embedded domain. Consumers are looking for affordable multimedia devices with high performance and durability making embedded developers to think creatively and use all resources at hand to meet the desired user specifications. This is one such attempt by designing an Audio Player and Recorder to play the wave audio files from a USB flash drive and recording the audio in USB flash drive in the same format. In this application MEMS microphone is used for recording the audio data.

LIST OF FIGURES

Figure No.	Figure Title	Page No.
Figure 1	STM32F4 Discovery Board.....	12
Figure 2	Hardware block diagram of STM32F4 Discovery board	13
Figure 3	STM32F4 Discovery MCU.....	14
Figure 4	Run time environment of Keil MDK pro	15
Figure 5	Source code editor	16
Figure 6.	WAV File Format Layout.....	18
Figure 7	Schematic connection of two devices to I2C bus.....	23
Figure 8	Start and stop condition in I2C bus	25
Figure 9	Data stability condition	25
Figure 10:	USB Flash Drive	26
Figure 11.	Logical Connections between USB Host Clients and USB Device Endpoints	27
Figure 12	Master Boot Record [11].....	30
Figure 13	16 byte Partition entry	31
Figure 14	Critical fields of FAT 32 Volume ID.....	31
Figure 15	Schematic of Audio peripherals connection on STM32F4 discovery	34
Figure 16	Audio playback and record architecture.....	35

LIST OF TABLES

Table No.	Table Title	Page No.
Table 1	Difference between RISC and CISC	9
Table 2	Wave File Header	19
Table 3:	Data chunk.....	21
Table 4	Condition of I2C bus while read/write operation.....	24
Table 5	Bus condition while writing 2 bytes to slave	24
Table 6	Bus condition while reading 2 bytes from slave	25
Table 7	Advantages and Disadvantages of USB	28
Table 8	Main Variables of Volume ID	32

CHAPTER 1

INTRODUCTION

- Motivation
- Organisation of thesis

1 INTRODUCTION

1.1 Project Overview

An Audio player has been implemented on the stm32f4 discovery board which will play the audio data (wave) stored in a USB flash drive. The audio data is read from the external USB drive of the STM32F407VG microcontroller using the DMA and also can be recorded in the same format on the usb drive. Playing wave audio files which are the basic high quality audio format because these are also uncompressed files, the quality of the output is nowhere compromised. The recording application has been implemented using ST MP45DT02 MEMS microphone which is present on the discovery board, with a PDM audio software decoding library. The MEMS microphone produces PDM data which is then converted to PCM data and stored into the usb flash drive. Thus if one is seeking to play music and record audio with a single device, the device thus developed may come very handy.

With the improvements in processor technology and development of software design environments for the same, designing of an embedded system has not remained a very tedious task as it was a decade ago. Processors such ARM which have very good computing performances are becoming more popular among embedded developers and are also coming at low prices.

1.2 Organisation of Thesis

Chapter 1: This chapter includes brief introduction and organisation of thesis

Chapter 2: This chapter deals with ARM microcontrollers giving brief idea of RISC methodology of processor design, development board used and the keil uvision MDK pro which is a design environment for ARM processors

Chapter 3: This chapter deals with the file formats that needed to worked upon describing in detail about the WAVE audio format.

Chapter 4: This chapter deals with various communication protocols that have been used for making components communicate with processor and among each other.

Chapter 5: This chapter deals with the idea of FAT32 file system and SD card interfacing.

Chapter 6: This chapter explores the audio playback and recording applications in detail and each operation has been carefully explained with the help of flowcharts

Chapter 7: This chapter provides the various references used in this project work.

CHAPTER 2

ARM MICROCONTROLLERS

- Reduced Instruction Set Computer (RISC)
- ARM Processors
- STM32F4 Discovery Board

2 ARM MICROCONTROLLERS AND TOOLCHAINS

2.1 Reduced Instruction Set Computer (RISC)

RISC is a design methodology for developing CPUs based on the belief that a simple instruction set combined with a processor architecture equipped to execute these instructions provides enhanced performance with less processor cycle per instruction thus consuming less circuitry (transistors) [1]. Its most common trait is the use of Load/Store architecture in which memory is generally used only through particular instruction only instead of using it as a part of some other instruction. Due to reduced no. of cycles per instruction (CPI) it provides higher speed. This is done by optimizing every instruction on CPU and Pipelining.

2.1.1 Characteristics:

- Fewer instructions rather than a very large and complex instruction set.
- Fewer addressing modes making it more flexible and user friendly.
- Operations required to be performed takes place within CPU itself.
- Every instruction is executed in same amount of time (cycles) and hence enhances the speed of the system.
- In this design methodology processors have sufficiently large number of registers and a considerably more productive instruction pipeline.
- Instructions are of constant length and easy to decode.

2.1.2 RISC vs CISC

Following table listing the difference between two processor design methodologies.

Table 1 Difference between RISC and CISC

RISC	CISC
Stress on Software	Stress on Hardware
Fixed instruction size and very few formats.	Various instruction sizes and formats
Large number of registers	Fewer registers
Lesser addressing modes	Wide variety of addressing modes
complex compiler	Large scale use of microprogramming.
All instructions take only 1 processor cycle for execution.	Different time of executions for different instructions.
easy pipelining	Pipelining difficult.

2.1.3 Advantages of RISC over CISC

- Since a streamlined instruction set takes into account a pipelined, superscalar plan RISC processors frequently attain to 2 to 4 times the execution of CISC processors utilizing tantamount semiconductor innovation and the same clock rates.
- Since the instruction set of a RISC processor is so basic, it uses up substantially less chip space; additional capacities, for example, memory management units or floating point arithmetic units, can likewise be set on the same chip. Littler chips permit a semiconductor producer to place more parts on a solitary silicon wafer, which can bring down the every chip cost significantly.
- Since RISC processors are more straightforward than relating CISC processors, they can be developed more rapidly, and can exploit other technological

advancements faster than their CISC counterparts, prompting more noteworthy jumps in performance in upcoming eras.

2.1.4 Disadvantages of RISC over CISC

- Performance of a processor developed with RISC methodology is greatly determined by the quality of code which it has to execute. A poor job on programmer's part or compiler can lead to processor spending lot of time stalling. Since instruction scheduling is a tedious task, most of programmers choose high level languages such as C or C++, leaving the job of scheduling to compilers. Thus compiler has to be chosen carefully to generate quality code, making RISC processors compiler dependent.
- Instruction scheduling makes debugging a tedious task. When IS is taken into account then machine language equivalent of a line of source can appear as in between other instructions of source code.
- Code expansion can be a problem with RISC processors as a complex operation can consume many instruction which can be done by a single instruction with CISC processors.
- Very fast Memory systems are required with RISC processors which are capable of feeding instructions to processor at fast rates. Thus RISC systems are required of having big memory caches generally within the chip. This incorporation is also called First-Level cache.

2.2 ARM Processors

This is a family of 32-bit microcontrollers developed on RISC (Reduced Instruction set computing) methodology developed by a company from Great Britain ARM Holdings. Unlike its CISC (Complex Instruction Set Computing) counterparts it consists of fewer transistors thus reducing cost, heat dissipation and power consumption, which are desirable for designing light, battery powered devices such as smart phones , tablets etc. The company designs instruction sets and architecture but not product itself.

2.2.1 History

Development of first ARM processor dates back to 1980s when Acorn RISC Machine Architecture (ARM) was developed by British Acorn Computers, its first product being coprocessor modules for a series of computers BBC Micro. It was developed using VLSI Technology as silicon partners. It included data bus(32 bits), address space(26 bits) and 27 registers (32 bits). Out of all bits of PC(program counter) register, 8 were available for other purposes, 6 for Status Flags and 2 bits were used for choosing setting mode.

2.2.2 Operating System Support

ARM architecture supports a huge number of embedded and real time operating systems like Linux, embedded C, freeRTOS etc. Both 32 and 64 bit operating system are supported on ARM architecture.

2.2.3 ARM Cortex M4

ARM Cortex M4 is a 32-bit microcontroller based on RISC (Reduced Instruction Set Computer) architecture specifically designed for efficient signal processing applications. It highlights extended Multiply- Accumulate (MAC) instructions, enhanced SMID arithmetic instructions and a Floating Point Unit (FPU) [2]. Basically ARM M4 is nothing but M3 with special Digital Signal Processing (DSP) instructions. The blend of high-proficiency signal transforming utility with low-power consumption, minimal expenses and wide usability of the Cortex-M group of processors is intended to fulfil the emerging class of adaptable solutions particularly focusing on the engine control, car, power optimization, embedded application and modern robotic markets.

Key features of ARM Cortex M4 can be summarised as follows

- ARMv7E Architecture
- Three stage Pipelining
- Instruction set contains

- Thumb Instructions which are a subset of ARM instruction in 16 bit compressed mode.
- 32 bit hardware multiply
- 32 bit hardware divide
- DSP support
- 1 to 240 interrupts with Non Maskable Interrupts (NMI)
- Sleep modes are also available

2.2.3 STM32F4 Discovery board

STM32F4 Discovery board is a product of ST Microelectronics which has a STM32 microcontroller based on ARM Cortex M4 architecture. It includes an ST-LINK/V2 embedded debug tool interface, ST MEMS digital accelerometer, ST MEMS digital microphone, audio DAC with integrated class D speaker driver, LEDs, push buttons and a USB OTG micro AB connector.



Figure 1 STM32F4 Discovery Board

It is a low cost easy to use development kit to quickly start development with powerful STM32 microcontrollers. Tool chain used is Keil uvision MDK pro which provides inbuilt libraries

and a user friendly design environment for STM32F4. Hardware block diagram of the discovery board is shown below.

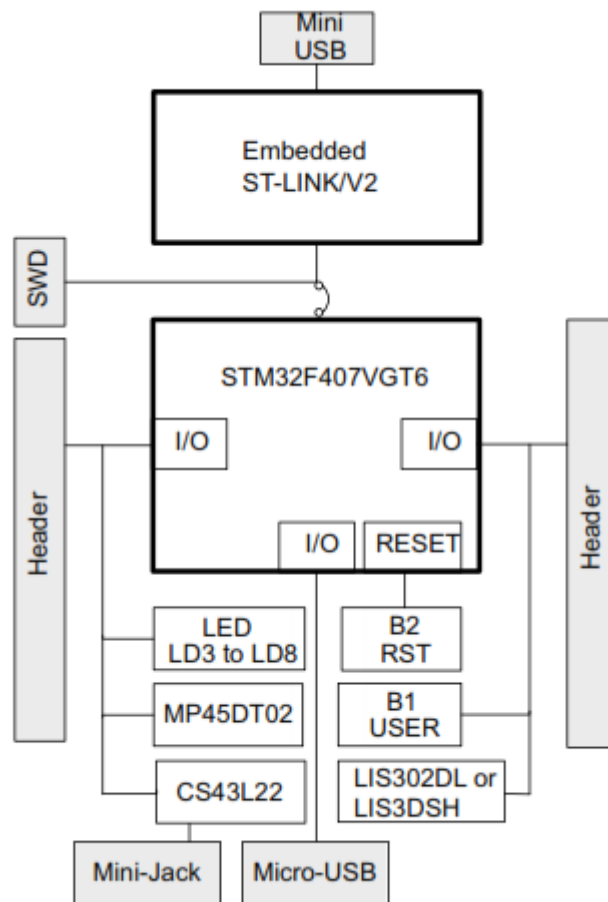


Figure 2 Hardware block diagram of STM32F4 Discovery board

It has ARM Cortex M4 32 bit MCU which is characterised by 210 DMIPS, up to 1 MB flash/192+4 KB RAM, USB OTG HS/FS, Ethernet, 3 ADCs and 15 communication interfaces.

Following figure describes the overall MCU structure of STM32F4.

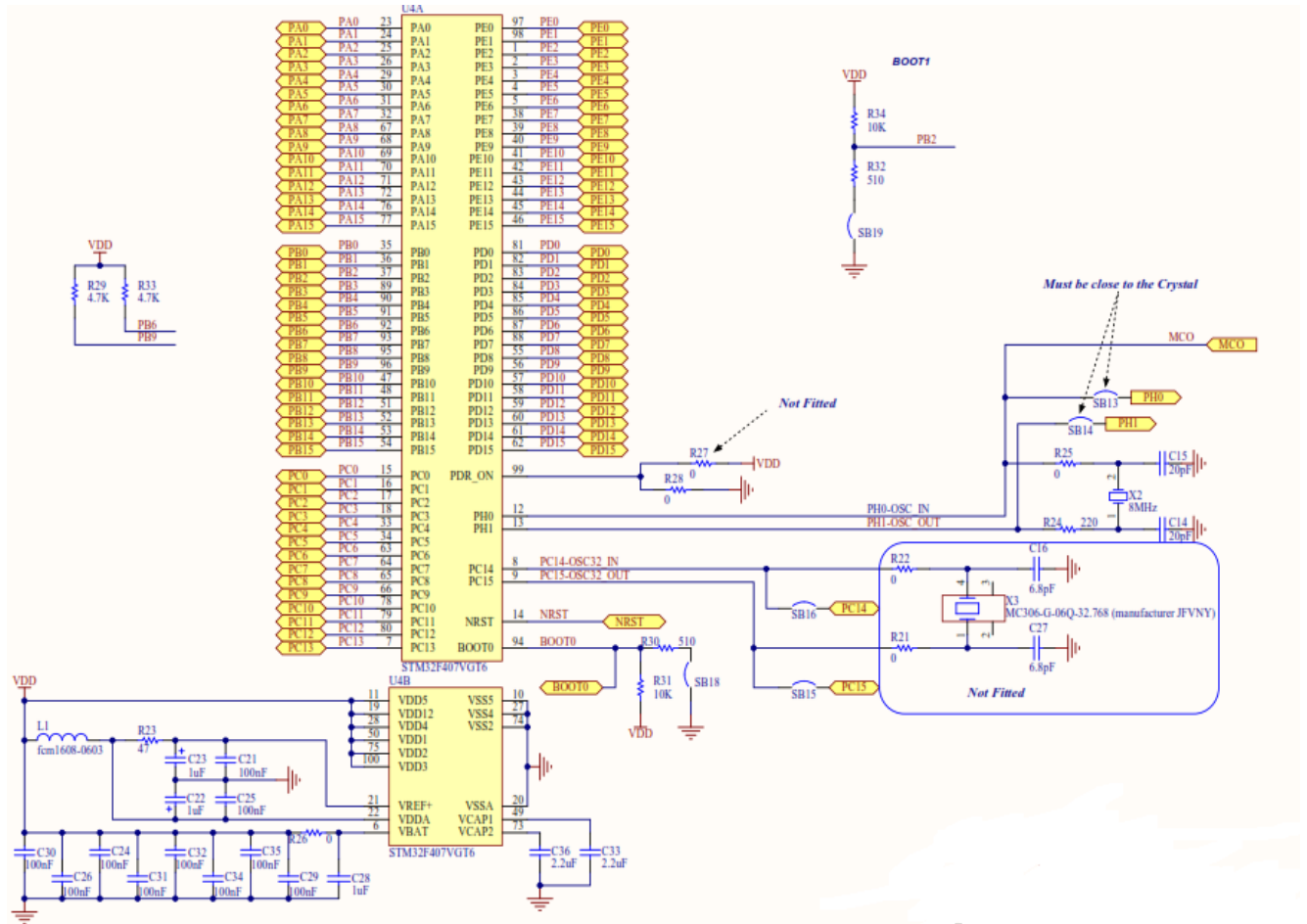
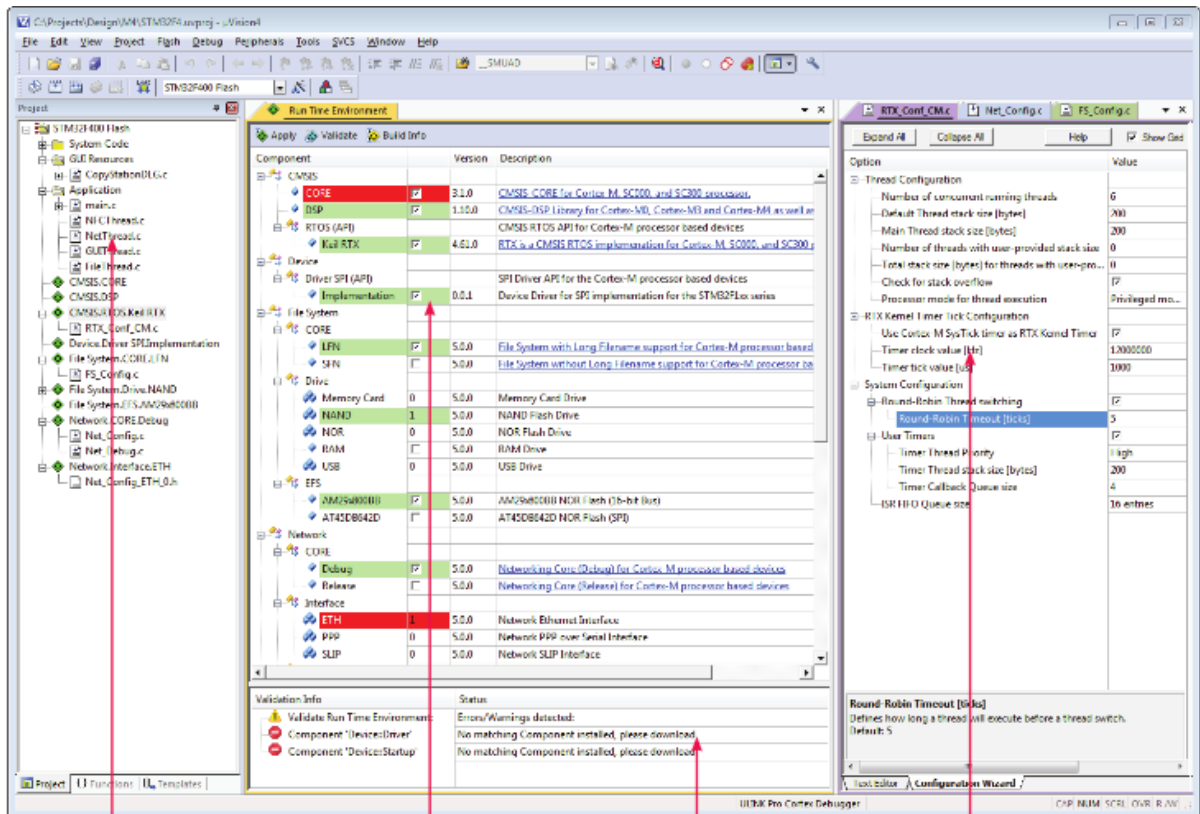


Figure 3 STM32F4 Discovery MCU

2.3 Keil uvision MDK pro Tool chain

Keil MDK pro is a complete software development environment for a variety of ARM based microcontrollers including Cortex M and Cortex R series. It includes uvision IDE/Debugger, ARM C/C++ compiler and all the additional middleware components required for ARM based embedded development [3]. It also includes Keil RTX which is a Real Time Operating System (RTOS). It also features a complete GUI library for designing Graphical User Interface (GUI) for embedded systems.

Run time environment window shows all software components that are compatible with the selected device. These pre designed software components helps to design embedded systems faster. It creates complete design environment for the chosen device.



- The Project Window shows application source files of selected software components.
- Create the Run-Time Environment from Software Packs with pre-built software components.
- Inter-dependencies of software components are clearly identified with validation messages.
- The Configuration Wizard simplifies the setup for selected software components.

Figure 4 Run time environment of Keil MDK pro

It has a powerful source code editor which enhances productivity and is a user friendly environment for compiling source codes.

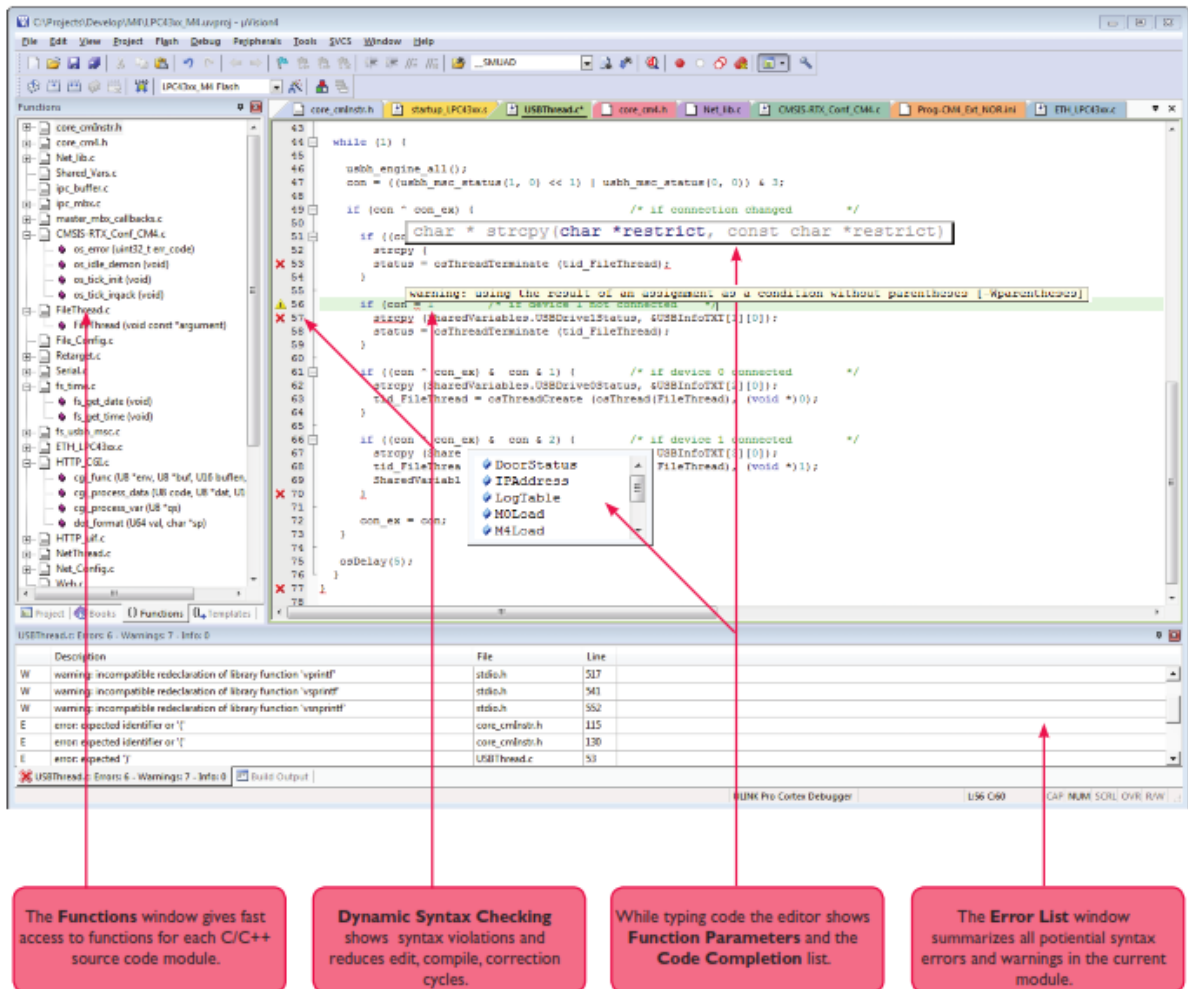


Figure 5 Source code editor

CHAPTER 3

WAVE FILE FORMAT

3 FILE FORMATS

3.1 Wave Audio File Format

The Wave file format is Windows' local file format for storing computerized digital audio information. It has turned into a standout amongst the most broadly upheld computerized audio formats on the PC because of the notoriety of Windows and the immense number of projects composed for the stage. In this file format the arrangement of data is little-endian that is the least significant byte has to appear first.

Wave files make use of a general Resource Interchange File Format (RIFF) [6] structure which stores the contents of the file in to different “chunks”. The chunk header tells the application about the type and the length of data bytes stored in the chunk. This method of organisation makes a program powerful by allowing it to skip over unrecognisable types of chunk and continue the processing of known data chunks. Following figure shows the basic wave file layout.

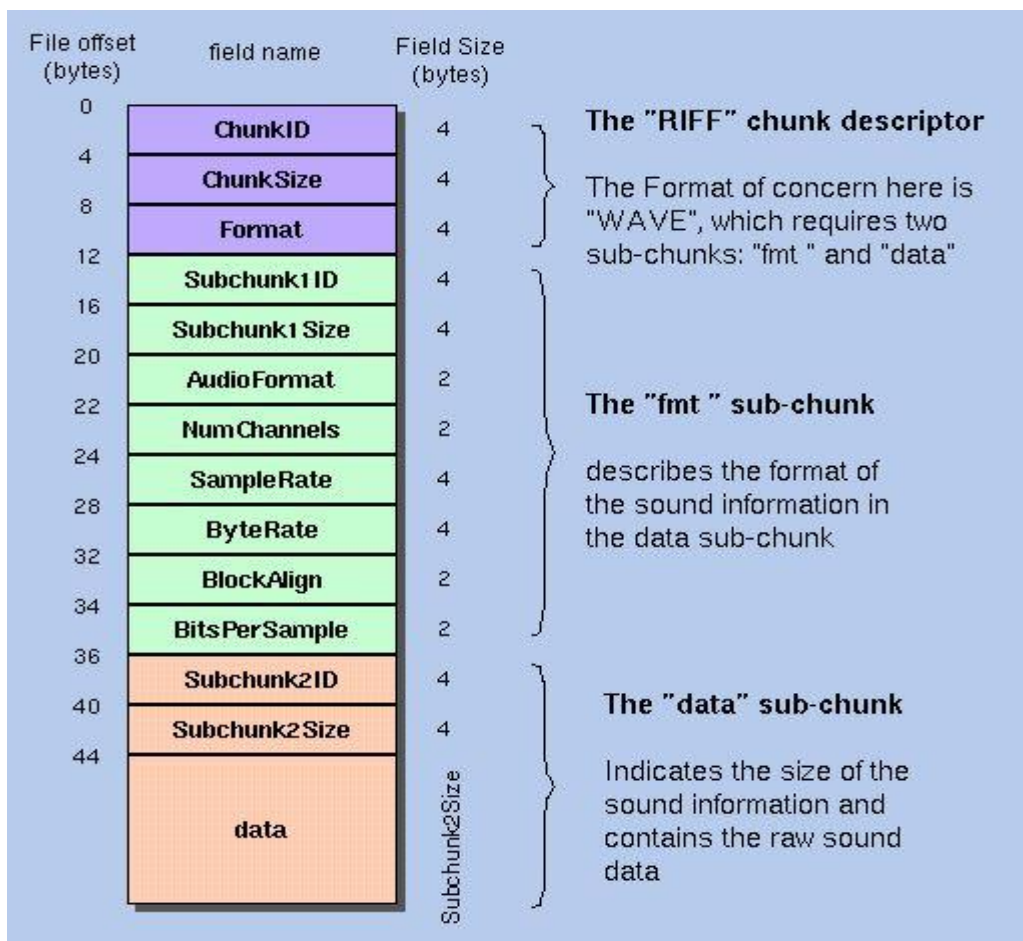


Figure 6. WAV File Format Layout

3.2.1 Wave file header

It follows standard RIFF structure. The very first 8 bytes of the file is a RIFF chunk header which has an ID of "RIFF" and chunk size which is same as file size minus 8 bytes for header. WAV files are stored uncompressed, therefore they can get quite large, but they cannot exceed 4 gigabytes due to the fact that the file size header field is a 32-bit unsigned integer.

A WAVE file is often just a RIFF file with a single "WAVE" chunk which consists of two sub-chunks -- a "fmt" chunk specifying the data format and a "data" chunk containing the actual sample data.

The following table gives the details of the wave file header.

Table 2 Wave File Header

BYTE NUMBER	SIZE	DESCRIPTION	VALUES
0-3	4	Chunk ID	"RIFF" (0x52494646)
4-7	4	Chunk Data Size	8 (file size)
8-11	4	RIFF type	"WAVE" (0x57415645)

3.2.2 Format Chunk

Format chunk has detailed information about storing the data waveform and playing it including the mode of compression, channel number, rate of sampling, number of bits in each sample and other properties. Following table provides good insight into format chunk.

Channel – An independent waveform in the audio data. The number of channels is important: one channel is "Mono," two channels is "Stereo" – there are different waves for the left and right speakers. 5.1 surround sound has 5 channels, one of which is for the lowest sounds and is usually sent to a subwoofer. Again, each channel holds audio data that is independent of all the other channels, although all channels will be the same overall length.

Frame – A frame is like a sample, but in multichannel format – it is a snapshot of all the channels at a specific data point.

Sampling_Rate/Sample_Rate – The number of samples (or frames) that exist for each second of data. This field is represented in Hz, or “per second.” For example, CD-quality audio has 44,100 samples per second. A higher sampling rate means higher fidelity audio.

Bit_Depth/Bits_per_Sample – The number of bits available for one sample. Common bit depths are 8-bit, 16-bit and 32-bit. A sample is almost always represented by a native data type, such as byte, short, or int. A higher bit depth means each sample can be more precise, resulting in higher fidelity audio.

3.2.3 Data chunk

BYTE NUMBER	SIZE (Bytes)	DESCRIPTION	VALUES
0-3	4	Chunk ID	"fmt " (0x666D7420)
4-7	4	Chunk Data Size	Length Of format Chunk (always 0x10)
8-9	2	Compression Code	Always 0x01
10-11	2	Channel Number	0x01=mono 0x02=stereo
12-15	4	Sample Rate	Binary , in Hertz
16-19	4	Bytes Per Second	
20-21	2	Bytes per Sample	1=8 bit mono, 2=8 bit stereo, 3=16 bit mono, 4= 16 bit stereo
22-23	2	Bits Per Sample	

It contains the main digital audio data which has to be decoded by utilizing the format and compression scheme described by the wave format chunk. Following is the detailed structure of data chunk.

Table 3: Data chunk

BYTE NUMBER	SIZE	DESCRIPTION	VALUE
0-3	4	Chunk ID	"data" (0x64617461)
4-7	4	Chunk Data Size	Length of data to follow
8-end		Data	Sound Samples

CHAPTER 4

COMMUNICATION PROTOCOLS

- Inter Integrated Circuit (I2C) Protocol
- Universal Serial Bus

- Control bits for identification of start and end of data transfer, determining the direction of data transfer, and acknowledgement system.

According to this protocol the IC initiating data transfer acts as bus master while all other ICs connected to bus act as bus slaves. Firstly the master IC issues a “START” condition which acts as an “ATTENTION” for all other devices connected to the bus thus making them ready for receiving data on the bus [10]. After this the master device puts the address of the slave device it wants to communicate with, on the bus along with specifying whether it is a Read or Write operation. Slave devices compares this address with their own unique address. If it matches then the corresponding slave device gives an acknowledgement response. Other devices whose address does not matches with the address issued by master wait in idle state until they get stop signal on bus. When master sense this acknowledgement signal it starts sending or receiving data.

A typical I2C transfer can be explained by the condition of the bus while reading / writing two bytes of data.

Table 4 Condition of I2C bus while read/write operation

START	Slave Address	Read/Write	ACK	DATA	ACK	DATA	ACK	STOP
1 bit	7 bits	1 bit	1 bit	8 bits	1 bit	8 bits	1 bit	1 bit

The condition of the bus while writing 2 bytes of data to the slave device is shown below where shaded part are the data which is put on the bus by the master device.

Table 5 Bus condition while writing 2 bytes to slave

START	Slave Address	0	0	DATA	0	DATA	0	STOP
1 bit	7 bits	1 bit	1 bit	8 bits	1 bit	8 bits	1 bit	1 bit

The condition of the bus while reading 2 bytes of data from slave is also shown below where shaded part are the data which is put on the bus by the master device.

Table 6 Bus condition while reading 2 bytes from slave

START	Slave Address	1	0	DATA	0	DATA	1	STOP
1 bit	7 bits	1 bit	1 bit	8 bits	1 bit	8 bits	1 bit	1 bit

The conditions of “START” and “STOP” closely depends on the physical structure of bus.

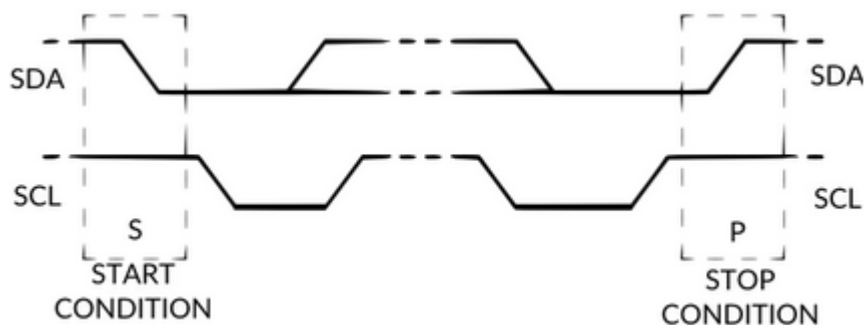


Figure 8 Start and stop condition in I2C bus

According to the specifications of I2C protocol, data on SDA line can alter only when the clock line, SCL, is at low level. Also the data is stable on SDA only when SCL is high.

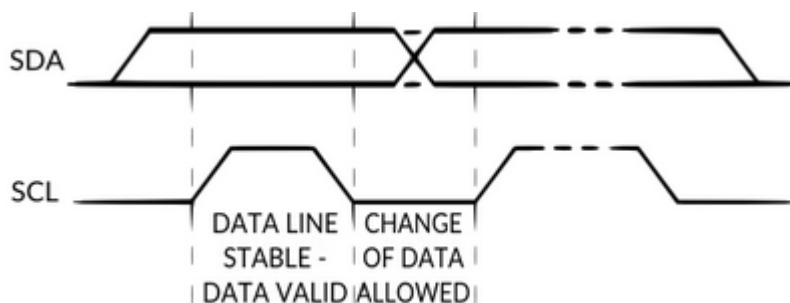


Figure 9 Data stability condition

4.2 Universal Serial Bus (USB)

USB, or the Universal Serial Bus Interface is now well established as an interface for computer communications.

In many areas it has completely overtaken RS232 and the parallel or Centronics interface for printers, and it is also widely used for memory sticks, computer mice, keyboards and for many other functions.

One of the advantages of USB is its flexibility: another is the speed that USB provides.

USB gives quite fast serial data transfer method for data communications, however power can also be obtained through connectors and this has further enhanced the popularity of USB as many low power computer accessories. USB find a wide utility from memory to disk drives.

The development USB interface was as a result of the demand for a data transfer interface that was easy to use and one that supports higher data rates which is a key requirement for the computer and peripherals industries.



Figure 10: USB Flash Drive

With USB 1.0 well established, faster data transfer rates were required, and accordingly a new specification, USB 2 was released. With the importance of USB already established it did not take long for the new standard to be adopted.

With USB shaping its place in the computer market, other improvements of the standard had to be researched. With the need for portability in many areas of the electronics industry taking off, the next predictable move for USB could have been to employ a wireless interface. In making it possible USB will need to employ a flexible methodology that has proved to a success for a wired interface. Moreover the wireless USB interface has to be able to send data at rates higher than the wired USB 2 connections.

4.3.1 USB Communication

USB is a serial bus, in which all the data transfer and receiving is initiated by the USB Host. The data is transmitted to or from endpoints in an USB Device. The client in the USB Host stores data in buffers, but does not have endpoints. As shown below different layers of data transmission can be seen. The interaction across different layers are logical Host-Device connection between each horizontal layer. Between the logical connections data is transferred using pipes.

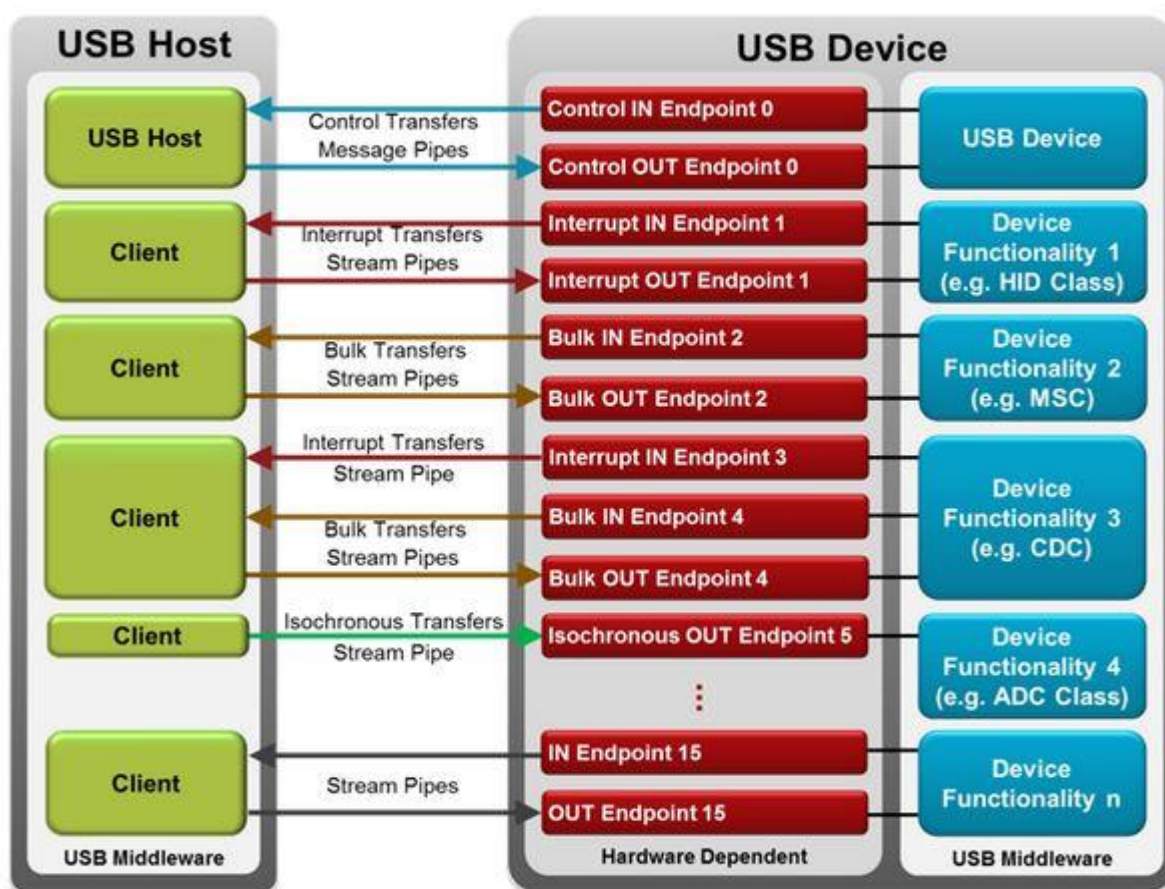


Figure 11. Logical Connections between USB Host Clients and USB Device Endpoints

4.3.2 USB Advantages and Disadvantages

USB has many advantages when compared to other technologies, but it also has a number of disadvantages which need to be considered when deciding on a technology to be used.

Table 7 Advantages and Disadvantages of USB

Advantages	Disadvantages
User friendly	Data transfer speed is not as fast as other Communication Protocols
Many applications can find its utility with data transfer speed in range.	Capabilities and speed is limited.
Connector system is robust	
Variety of connector types available	
Cost is low	

USB has many advantages and this is why it is so widely used. However, its simplicity and ease of use, mean that it is not always applicable in applications where more sophisticated interfaces are required for very high speed data transfer.

CHAPTER 5

FAT 32 FILE SYSTEM

5 FAT 32 FILE SYSTEM

5.1 FAT 32 File System

File Allocation Table (FAT) is a file system developed by Microsoft for MS-DOS and was the primary file system for consumer versions of Microsoft Windows up to and including Windows Me. FAT file system is relatively easy and is virtually supported by all major operating systems. This makes it an ideal file system for organization of data in hard disks, floppy disks, and memory cards.

FAT comes in three variations namely FAT 12, FAT16 and FAT 32. The numbers here corresponds to the number of bits that are being used in defining a cluster. Here in this project the SD card is using FAT 32 file system for storing the data [12].

The very first sector of the drive is known as Master Boot Record (MBR) with its first 446 bytes being used for booting the system. Then follows a partition table of length 64 bytes with last 2 bytes always being 0x55 and 0xAA which are usually used for checking the integrity of MBR.

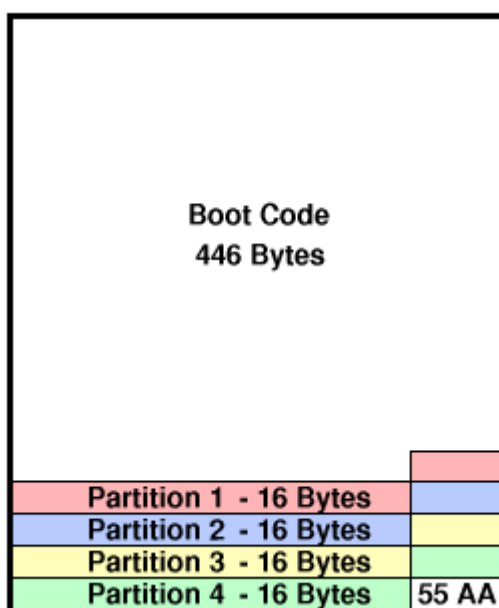


Figure 12 Master Boot Record [11]

Every partition is of 16 bytes length but most of the bytes can be ignored. Fifth byte is known as “Type Code” which gives information about the type of file system that is supposed to be contained in each partition. Ninth through twelfth bytes is “LBA begin” which is the address where that partition begins on the disk.

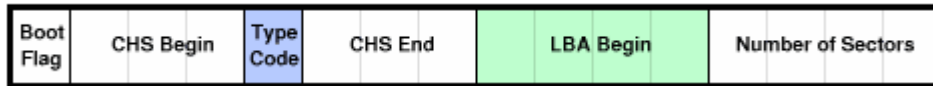


Figure 13 16 byte Partition entry

Generally only “type code” is looked upon for 0x0B or 0x0C and then “LBA begin” is read to know the location of FAT 32 File System on the disk.

While reading FAT 32 File System, first sector has to be read known as Volume ID which is supposed to be read by using “LBA begin” which is to be found from partition table. This sector gives detailed information about the layout of the FAT 32 File system. But many of these information are not of interest here. Only four variables are desired and another three for ensuring their authenticity.

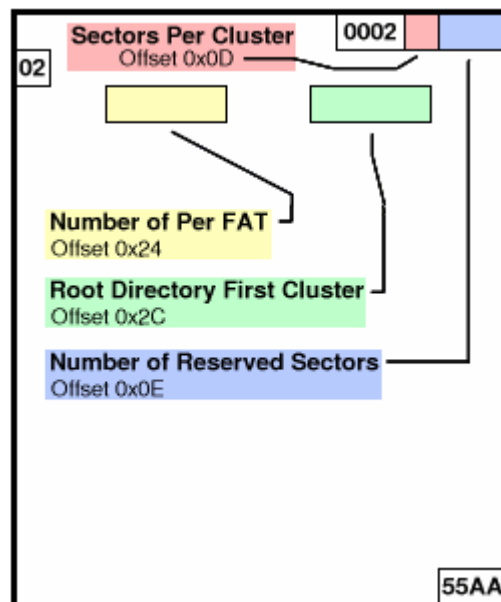


Figure 14 Critical fields of FAT 32 Volume ID

Following table dig into these seven variables and their significance while using FAT 32 File System.

Table 8 Main Variables of Volume ID

FIELD		MICROSOFT'S NAME	OFFSET	SIZE	VALUE
Bytes Per Sector		BPB_BytsPerSec	0x0B	16 bits	Always 512 Bytes
Sectors Per Cluster		BPB_SecPerClus	0x0D	8 bits	1,2,4,8,16,32,64,128
Number Of Reserved Sectors		BPB_RsvdSecCnt	0x0E	16 bits	Usually 0x20
Number Of FATs		BPB_NumFATs	0x10	8 bits	Always 2
Sectors Per FAT		BPB_FATSz32	0x24	32 bits	Depends on size of the disk
Root Directory First Cluster		BPB_RootClus	0x2C	32 bits	Usually 0x00000002
Signature		None	0x1FE	16 bits	Always 0xAA55

Arrangement of FAT 32 File System is very simple. Volume ID is always the first sector followed by some empty space known as Reserved Sectors. After this reserved sector there exists two set of File Allocation Tables (FATs). The remaining file system is data organised in clusters with a very small unused space after last cluster. Majority of space is being used by clusters used to store files. Cluster numbering is that first cluster is named cluster 2, where there is no cluster 0 or cluster 1.

CHAPTER 6

AUDIO PLAYBACK AND RECORDING

- Application Overview
- Audio Playback Application
- Audio Record Application

6 AUDIO PLAYBACK AND RECORDING APPLICATION

6.1 Application Overview

Wave audio files can be played using STM32F4 with USB storage device as the file destination. The application make use of MEMS microphone, audio DAC, headphone and USB key. [15]

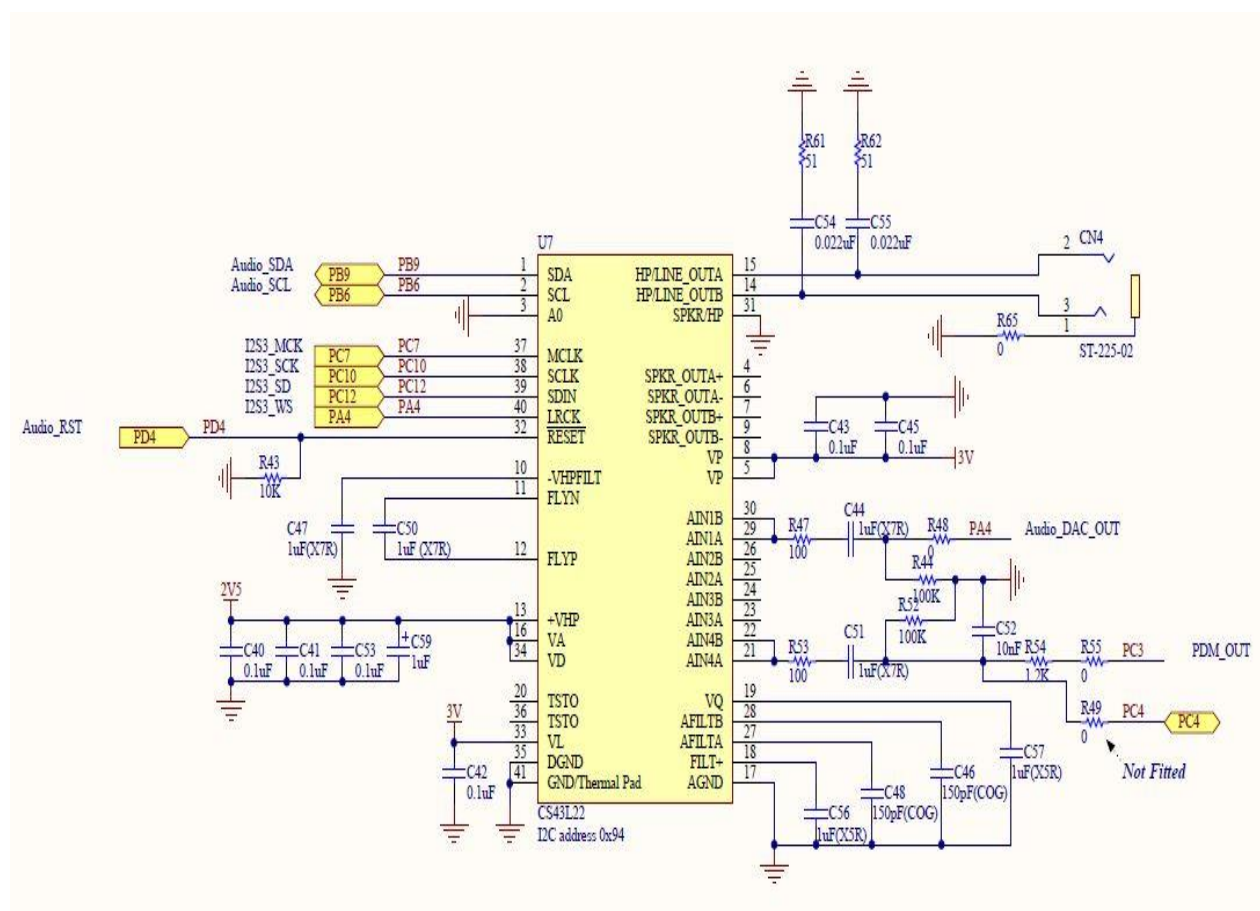


Figure 15 Schematic of Audio peripherals connection on STM32F4 discovery

USB peripheral has to be configured in host mode. Mass Storage Class (MSC) is used to send/receive audio data to/from USB. I2S peripheral is configured in master transmitter mode and used to transmit audio data to the external audio codec (DAC). DMA is used to send from buffers to I2S peripherals which efficiently reduces the load on CPU [12]. I2C peripheral is used to control external devices like audio codec and obtain data from that device. User button is for monitoring the operation (playback or recording).

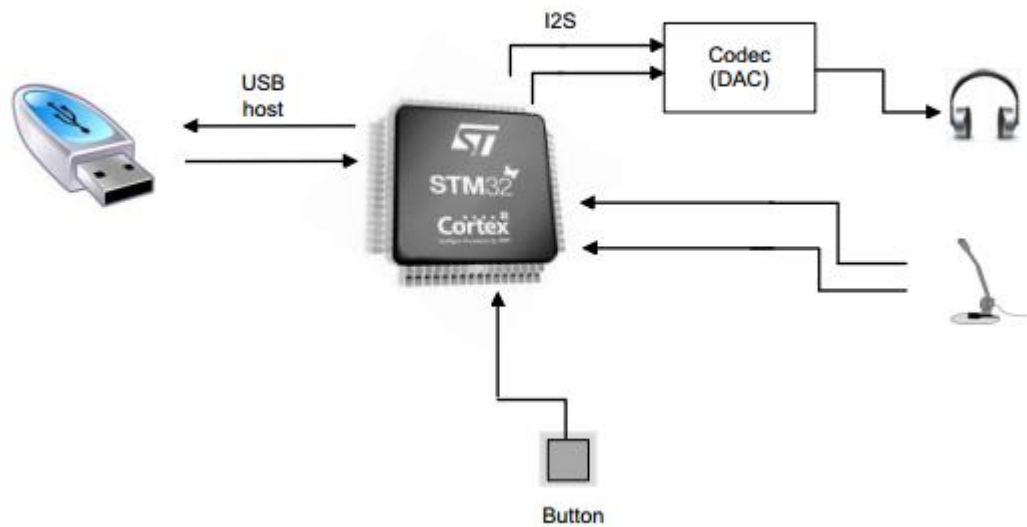


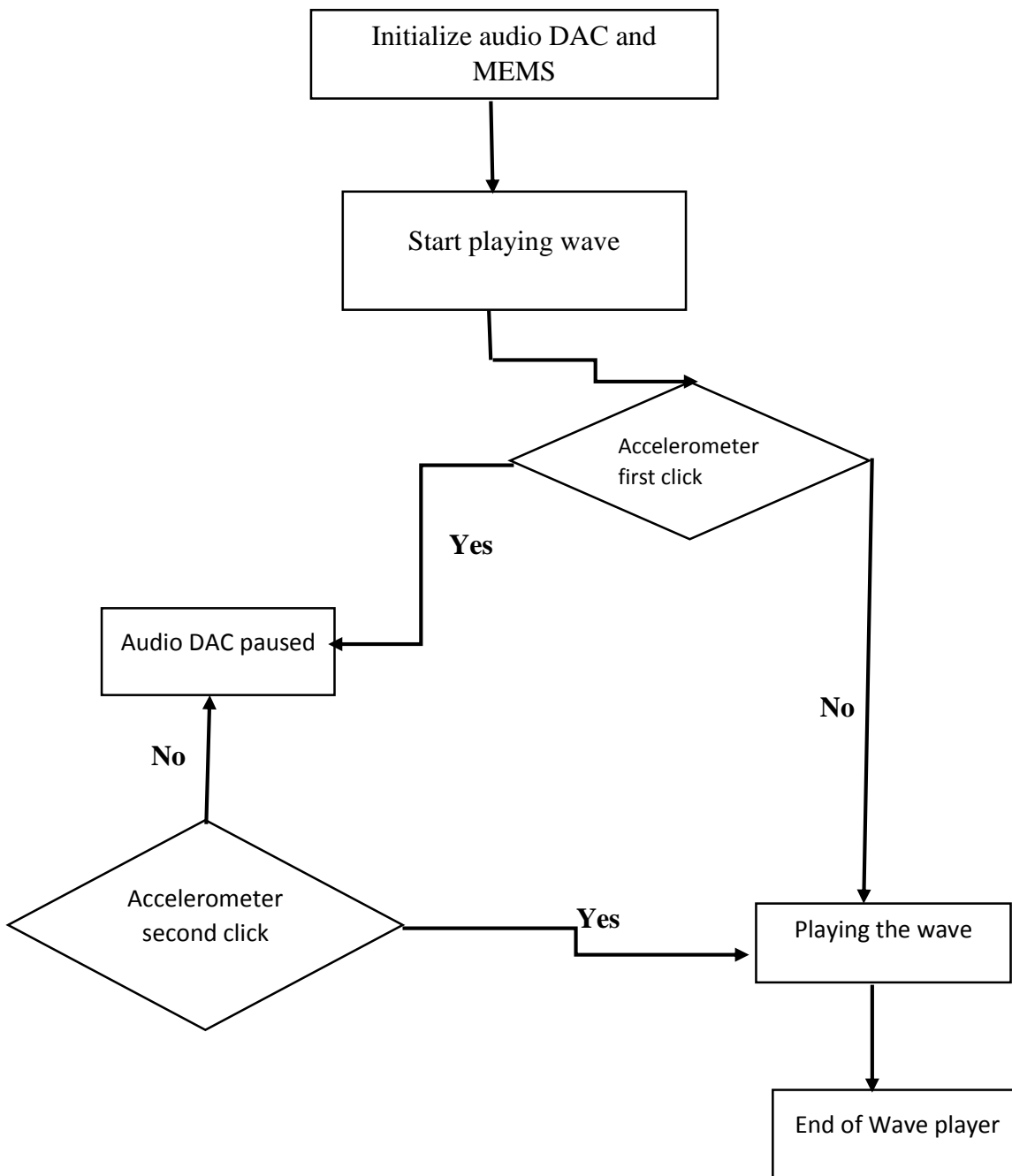
Figure 16 Audio playback and record architecture

6.2 Audio Playback Application

Procedure for the audio play application on the board

1. Initialize the USB, FatFs File Systems, Audio DAC, MEMS
2. Transfer the WAV file from USB storage to internal SRAM of MCU, block by block (1024 bytes) using DMA in its first buffer.
3. DMA sends its data to I2S Peripheral which transfers it to external audio codec (DAC).
4. In the meantime data is stored from USB to DMA in its second buffer.
5. These two buffers are swapped indefinitely till the end of audio file.

Following flowchart describes the audio playback application.



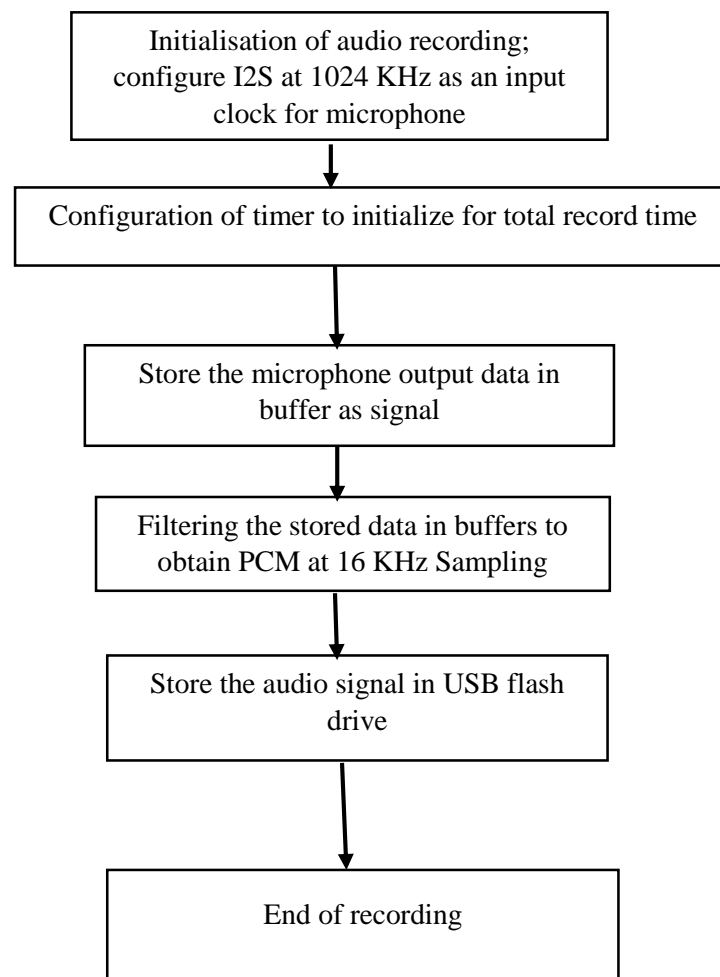
MEMS accelerometer is used as pause/resume support. When the playback is running first click on the board stops it playing and second click resumes the playing of wave file.

6.3 Audio Record Application

Procedure for the audio recording:

1. Audio Record initialisation: Configure I2S as 1024 KHz as an input clock for MEMS microphone
2. Timer configuration to initialize recording time
3. Store the microphone in buffer as signal
4. Filter the stored data to obtain a PCM signal at 16 KHz sampling frequency
5. Store the filtered signal in the USB mass media.

Following flowchart describes the record application [15]



I2S peripheral has been configured in master mode in order to generate the correct clock cycle (1.024 MHz). The 1.024 MHz clock can be calculated from the output audio streaming (16 KHz) and the decimation factor (64) chosen for this application ($16000 \text{ Hz} \times 64 = 1.024 \text{ MHz}$)[15].

CHAPTER 7

**CONCLUSION AND FUTURE
PROSPECTS**

7 CONCLUSION AND FUTURE WORK

7.1 Conclusion

An audio player and recorder has been thus implemented for playing the wave audio file stored in a USB Flash Drive and recording audio in the same format on the USB flash drive. For playing the audio file from USB Key firstly, wav data is transferred from the USB flash drive to internal SRAM of the microcontroller block by block in the first DMA buffer. Once this buffer is filled:

- DMA sends this data to I2S peripheral which transfers this to external audio codec DAC to decode the audio data.
- The data from USB is stored in the second buffer.

Then these buffers are swapped indefinitely till the end of the audio file.

7.2 Future Prospects

The implemented audio player is only able to play wav audio files not any other file formats from the USB Flash drive. It can be made to play some audio formats like mp3. If someone wants to interface the USB with the microcontroller they can use it as their reference. A graphic LCD can be connected to the board and a GUI interface can be made. It will show the content of the USB drive on the screen and will make the application more user friendly in terms of accessing the content of the USB Flash Drive.

CHAPTER 8

REFERENCES

8 REFERENCES

- [1] Sivarama P Dandamudi, *Guide to RISC Processors for programmers and engineers*, Springer International Series in Engineering and Computer Science,2005
- [2] Joseph Yiu , *Definitive guide to ARM cortex M3 and ARM cortex M4 Processors*, Newnes Publishers,2013
- [3] www.keil.com/product/brochures/uv4.pdf
- [4] topherlee.com/software/pcm-tut-waveformat.htm
- [5] www.itk.ilstu.edu/faculty/javila/datatypes/soundwave.htm
- [6] <http://soundfile.sapp.org/doc/WaveFormat/>
- [7] <http://en.wikipedia.org/wiki/WAV>
- [8] Miroslav Popovic, *Communication Protocol Engineering*, CRC Press,2009
- [9] www.byteparadigm.com/application/introduction-to-i2c-and-spi-protocols
- [10] Richard Lai, Ajin jirachiefpattana, *Communication Protocol Specification and Verification*, Spinger US, 1998
- [11] <http://educyclopedia.karadimov.info/electronics/I2C.htm>
- [12] elm-chan.org/fsw/ff/00index_e.html
- [13] www.pjrc.com/tech/8051/ide/fat32.html
- [14] <http://www.inversereality.org/files/dmaprogramming.pdf>
- [15] www.st.com/stwebu/static/active/en/resource/technical/document/active/technical.pdf