RHODES UNIVERSITY
Where leaders learn

Department of Physics & Electronics

Real-Time Audio Spectrum Analyser

Research, Design, Development and Implementation using the 32-bit ARM® Cortex-M4 Microcontroller

A dissertation submitted in fulfilment of the
requirement for the degree of

Master of Science in Electronics

of

Rhodes University

by

Stefan Antonio Just

May 2016

Abstract

This thesis describes the design and testing of a low-cost hand-held real-time audio analyser (RTAA). This includes the design of an embedded system, the development of the firmware executed by the embedded system, and the implementation of a real-time signal processing algorithms.

One of the objectives of this project was to design an alternative low-cost audio analyser to the current commercially available solutions. The device was tested with the audio standard test signal (pink noise) and was compared to the expected flat-spectrum response corresponding to a balanced audio system. The design makes use of an 32-bit Reduced Instruction Set Computer (RISC) processor core (ARM Cortex-M4), namely the STM32F4 family of microcontrollers. Due to the pin compatibility of the microcontroller (designed and manufactured by STMicroelectronics), the new development board can also be upgraded with the newly released Cortex-M7 microcontroller, namely the STM32F7 family of microcontrollers. Moreover, the low-cost hardware design features 256kB Random Access Memory (RAM); on-board Micro-Electro-Mechanical System (MEMS) microphone; on-chip 12-bit Analogue-to-Digital (A/D) and Digital-to-Analogue (D/A) Converters; 3.2" Thin-Film-Transistor Liquid-Crystal Display (TFT-LCD) with a resistive touch screen sensor and SD-Card Socket.

Furthermore, two additional expansion modules were designed and can extend the functionality of the designed real-time audio analyser. Firstly, an audio/video module featuring a professional 24-bit 192kHz sampling rate audio CODEC; balanced audio microphone input; unbalanced line output; three MEMS microphone inputs; headphone output; and a Video Graphics Array (VGA) controller allowing the display of the analysed audio spectrum on either a projector or monitor. The second expansion module features two external memories: 1MB Static Random Access Memory (SRAM) and 16MB Synchronous Dynamic Random Access Memory (SDRAM). While the two additional expansion modules were not completely utilised by the firmware presented in this thesis, upgrades of the real-time audio analyser firmware in future revisions will provide a higher performing and more accurate analysis of the audio spectrum.

The full research and design process for the real-time audio analyser is discussed and both Problems and pitfalls with the final implemented design are highlighted and possible resolutions were investigated. The development costs (excluding labour) are given in the form of a bill of materials (BOM) with the total costs averaging around R1000. Moreover, the additional VGA controller could further decrease the overall costs with the removal of the TFT-LCD screen from the audio analyser and provided the external display was not included in the BOM.

Declaration

I hereby declare that this thesis has not been submitted, either in the same or different form, to this or any other university for a degree and that it represents my own work. I know the meaning of plagiarism and declare that all of the work in the thesis, save for that which is properly acknowledged, is my own.

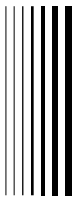
Stefan Antonio Just
May 2016

Acknowledgements

I would like to acknowledge a selected few individuals. Without your help this research project would not have been possible.

- Mr Anthony Sullivan — For supervising this research project, providing guidance and endless patience with me and my need to over complicate the simplest of tasks. Moreover, for allowing me to use, abuse and finally kill your old CRT monitor. R.I.P.
- Mr Andy Youthed — For helping create a backup server which contained all the code, schematic and circuit designs as well as all the resources used for this research. Additionally, for teaching me the process of etching small printed circuit boards for prototyping.
- Mr Dino Giovannoni — For providing your thoughts, opinions and guidance in writing this thesis. Additionally, for your infinite wisdom, knowledge and aid in navigating the intricacies of LaTeX.
- Family — For providing the funding of the first year of this research project, without the financial aid none of this research would be possible. Thank you for your support, sound opinions and suitable suggestions, all were considered.
- Department of Physics and Electronics — For the use of the *Tektronix MSO 2024B Mixed Signal Oscilloscope*, the *Tektronix AFG310 Arbitrary Function Generator* and a work space within the department.
- Microtronix Manufacturing — For aiding in fabricating and populating the printed circuit boards.
- National Research Foundation (NRF) — For funding the second year of this research project. The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

"Craftsmanship is a direct expression of character." - Don Davis



Contents

Contents v

List of Figures ix

List of Tables xii

1 Introduction 1

1.1 What are Audio Analysers and How are They Used . . . 1
1.2 Motivations for the Research Project . . . 1
1.3 Research Goal and Objectives . . . 2
1.4 Research Approach . . . 3
1.4.1 Phase 1 - Research, Theory and Literature Review . . . 3
1.4.2 Phase 2 - Design of Hardware . . . 3
1.4.3 Phase 3 - Development of Firmware . . . 4
1.4.4 Phase 4 - Implementation of the RTAA . . . 4
1.4.5 Phase 5 - Investigate Possible Improvements . . . 4
1.5 Thesis Structure . . . 4

2 Theory and Literature Review 5

2.1 Audio Signals and Systems . . . 5
2.1.1 Types of Audio Signals . . . 5
2.1.2 White Noise . . . 7
2.1.3 Pink Noise . . . 7
2.1.4 Black Noise . . . 8
2.1.5 Audio Equalization . . . 8
2.1.6 The Human Auditory System . . . 10
2.1.7 Equivalent Sound Level Addition . . . 11
2.2 Fourier Transformations . . . 13

2.2.1 Continuous-Time Fourier Transform (CTFT) . . . 13
2.2.2 Discrete-Time Fourier Transform (DTFT) . . . 13
2.2.3 Discrete Fourier Transform (DFT) . . . 14
2.3 Filters . . . 15
2.3.1 Digital Signal Processing Windows . . . 15
2.3.2 Audio Weighting Filters . . . 19
2.4 Simultaneous Real-Valued Fast-Fourier Transform (FFT) . . . 23
2.5 The Decibel (dB) . . . 27
2.5.1 The Decibel in Electronics . . . 27
2.5.2 The Decibel in Acoustics (L_I, L_W, L_P) . . . 28
2.5.3 The Decibel in Audio Electronics (dBV, dBu, dBW ,dBm, dBFS) . . . 28
2.5.4 Adding Decibels and Combining Voltages . . . 32
2.5.5 Microphone Sensitivity (S_V) and Open-Circuit voltage (V_O) . . . 34
2.6 The STM32F4 Series of ARM® Cortex-M4 Microcontrollers . . . 35
2.6.1 STM32F407 Microcontroller (Foundation Line) . . . 35
2.6.2 STM32F429 Microcontroller (Advanced Line) . . . 35

3 Overview of the Real-Time Audio Analyser 37
3.1 Introduction . . . 37
3.1.1 The STM32F429 Real-Time Audio Analyser . . . 38
3.2 Specifications and Capabilities of the Real-Time Audio Analyser . . . 39

4 The Real-Time Audio Analyser Hardware 40
4.1 Power Supply . . . 40
4.2 Crystal Oscillator . . . 40
4.3 STM32F4 Microcontroller . . . 41
4.3.1 Memory and System Bus Architecture . . . 41
4.3.2 Reset and Clock Controller (RCC) . . . 44
4.3.3 General Purpose Input/Output (GPIO) Controller . . . 48
4.3.4 Nested Vectored Interrupt Controller (NVIC) . . . 49
4.3.5 Flexible Memory Controller (FMC) . . . 49
4.3.6 Timers (TIM) . . . 51
4.3.7 Analogue-to-Digital Converter (ADC) . . . 52
4.3.8 Direct Memory Access (DMA) . . . 54
4.3.9 LCD-TFT Display Controller (LTDC) . . . 54
4.4 Debugging LEDs, Switches and Jumpers . . . 55
4.4.1 Debugging LEDs . . . 55
4.4.2 Reset and DFU Bootloader Switches . . . 56
4.4.3 Mode Selection Jumpers . . . 56
4.5 TFT-LCD Screen, Touch Screen Controller and SD Card Socket . . . 56
4.5.1 TFT-LCD Screen . . . 57
4.5.2 Touch Screen Controller . . . 58

4.6 External Memory . . . 59
4.6.1 AS6C8016 SRAM . . . 60
4.6.2 AS4C8M16S SDRAM . . . 62
4.7 Audio Data Converters . . . 64
4.7.1 Internal STM32F429 ADC . . . 64
4.7.2 External CS4272 CODEC . . . 64
4.8 External and Internal Microphones . . . 68
4.8.1 Internal Microphone . . . 68
4.8.2 External Microphone . . . 70
4.9 Video Graphics Array (VGA) Controller . . . 70
4.9.1 Video Digital-to-Analogue Converters (DAC) . . . 72

5 The Real-Time Audio Analyser Firmware 75
5.1 Programming the STM32F4 Microcontroller . . . 76
5.2 CMSIS-DSP Software Library . . . 76
5.3 The Real-Time Audio Analyser Files . . . 77
5.4 The Real-Time Audio Analyser Algorithm . . . 78
5.4.1 Powering up the Real-Time Audio Analyser . . . 79
5.4.2 Initialise RCC . . . 79
5.4.3 Configure the Real-Time Audio Analyser . . . 79
5.4.4 Endless While Loop and DSP Processing . . . 89
5.5 Some Final Words . . . 90

6 Testing and Discussion 91
6.1 Performance Comparison to Commercial Solutions . . . 91
6.1.1 Pure Audio Tone Signal Results . . . 93
6.1.2 Pink Noise Audio Signal Results . . . 95
6.1.3 Final Deliberation . . . 96

7 Future Work 97

8 Conclusion 99

A Schematics, PCB Designs and BOM 101
A.1 DAD Schematics and PCB Designs . . . 102
A.1.1 DAD MCU Board . . . 102
A.1.2 DAD External Memory Board . . . 106
A.1.3 DAD Audio and Video Board . . . 110
A.1.4 DAD Pinout Board . . . 114
A.1.5 Internal MEMS Microphone Breakout Boards . . . 118

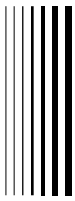
B STM32F4 Peripheral Registers 119
B.1 Reset and Clock Controller (RCC) . . . 119

B.2 General Purpose Input/Output (GPIO) Controller 120
B.3 Nested Interrupt Controller (NVIC) 121
B.4 Advanced Timer 8 (TIM8) 121
B.5 Flexible Memory Controller (FMC) 122
B.6 Analogue-to-Digital Converter (ADC) 123
B.7 Direct Memory Access (DMA) Controllers 123
B.8 LCD-TFT Display Controller (LTDC) 124
B.9 Inter-Integrated Circuit (I^2C) Interface 125
B.10 Serial Peripheral Interface (SPI) 125
B.11 Inter-IC Sound (I^2S) Interface 126

Bibliography **127**

List of Electronic Vendors **132**

Acronyms **134**



List of Figures

2.1 Periodic 100Hz test tone signal calculated and plotted using **Mat**rix **Lab**oratory (MATLAB) 6
2.2 Non-periodic random noise signal calculated and plotted using MATLAB 6
2.3 Front view of DN360 $\frac{1}{3}$ octave graphic equaliser[1] 8
2.4 Addition of correlated audio sources[2] 11
2.5 Addition of uncorrelated audio sources[2] 12
2.6 Rectangle window function for 4096 data points calculated and plotted with MATLAB 16
2.7 Hanning window function for 4096 data points calculated and plotted with MATLAB 16
2.8 Hamming window function for 4096 data points calculated and plotted with MATLAB 17
2.9 Welch window function for 4096 data points calculated and plotted with MATLAB 17
2.10 Bartlett window function for 4096 data points calculated and plotted with MATLAB 18
2.11 Blackman window function for 4096 data points calculated and plotted with MATLAB 18
2.12 Audio weighting functions for 30 bands of $\frac{1}{3}$ octave spaced steps calculated and plotted using MATLAB 23
2.13 Difference between correlated and uncorrelated decibel addition for N sources each with level L calculated and plotted using MATLAB 33
2.14 Difference between correlated and uncorrelated decibel addition for N sources each with level L calculated and plotted using MATLAB 34

3.1 Simplified Real-Time Audio Analyser System Overview 37

4.1 System Bus Architecture on the STM32F429 microcontroller[3] 44
4.2 Clock Tree on STM32F429 microcontroller[3] 45
4.3 GPIO port bit basic structure[3] 48
4.4 FMC Block Diagram[3] 50
4.5 FMC Memory Banks[3] 51
4.6 Timers Features Comparison[3] 52
4.7 Single ADC Block Diagram on the STM32F429 microcontroller[3] 53

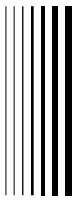
4.8 DMA Block Diagram on the STM32F429 microcontroller[3] 54
4.9 LTDC Block Diagram on the STM32F429 microcontroller[3] 55
4.10 ITDB02-3.2S TFT-LCD Module from ITead Studio[4] 57
4.11 ITDB02-3.2S TFT-LCD Module (Bottom View) from ITead Studio[4] 57
4.12 SRAM Block Diagram[5] 60
4.13 SDRAM Block Diagram[6] 62
4.14 CS4272 Block Diagram[7] 64
4.15 Fully Differential Input Buffer Circuit[7] 66
4.16 Single-Ended Input Buffer Circuit[8] 67
4.17 Differential to Single-Ended Output Buffer[7] 68
4.18 ADMP401 MEMS microphone break-out board[9] 69
4.19 Overview of designed VGA hardware 71
4.20 VGA 15-Pin D-SUB Female Connector[10] 71
4.21 5-bit R-2R DAC Network 73
4.22 Reduced Equivalent R-2R DAC connected to 75Ω display monitor 74
4.23 Potential Divider Circuit 74

5.1 Real-Time Audio Analyser Firmware Project Directory 77
5.2 Real-Time Audio Analyser Firmware Flow Chart 78

6.1 Audio Test System Overview 92
6.2 400Hz tone measured using the DSP30 audio analyser 94
6.3 400Hz tone measured using the real-time audio analyser 94
6.4 Pink noise measured using the DSP30 audio analyser 95
6.5 Pink noise measured with the real-time audio analyser 96

A.1 DAD Microcontroller board silkscreen designed using Easily Applicable Graphical Layout Editor (EAGLE) 102
A.2 DAD Microcontroller board top layer designed using EAGLE 103
A.3 DAD Microcontroller board bottom layer designed using EAGLE 104
A.4 DAD Memory board silkscreen designed using EAGLE 106
A.5 DAD Memory board top layer designed using EAGLE 107
A.6 DAD Memory board bottom layer designed using EAGLE 108
A.7 DAD AV board silkscreen designed using EAGLE 110
A.8 DAD AV board top layer designed using EAGLE 111
A.9 DAD AV board bottom layer designed using EAGLE 112
A.10 DAD PINOUT board silkscreen designed using EAGLE 114
A.11 DAD PINOUT board top layer designed using EAGLE 115
A.12 DAD PINOUT board bottom layer designed using EAGLE 116
A.13 C928A board silkscreen designed using EAGLE 118
A.14 C928A board top layer designed using EAGLE 118
A.15 C928A board bottom layer designed using EAGLE 118
A.16 MP34DT01 board silkscreen designed using EAGLE 118

A.17 MP34DT01 board top layer designed using EAGLE 118
A.18 MP34DT01 board bottom layer designed using EAGLE 118



List of Tables

1.1 Real-Time Audio Analyser System Requirements 2

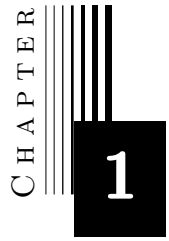
2.1 30 Bands of $\frac{1}{3}$ octave spaced steps calculated using the STM32F429 microcontroller 10
2.2 Types of Fourier Transforms[11] 14
2.3 Audio Weighting functions for 30 bands of $\frac{1}{3}$ octave spaced steps calculated using the STM32F429 microcontroller 22
2.4 Quantization Values for a 12-bit Converter with $3.3V_{FS}$ 31
2.5 Quantization Values for a 24-bit Converter with $3.3V_{FS}$ 31
2.6 Comparison between the STM32F407 and STM32F429 microcontrollers[12][13] . . . 36

4.1 Peripheral Address Organization 42
4.2 Clock Frequencies used by the Real-Time Audio Analyser 47
4.3 Debugging LEDs on the DAD-MCU-BRD 55
4.4 SSD1289 Control and Data Pins 58
4.5 XPT2046 Control and Data Pins 59
4.6 XPT2046 Control and Data Pins 59
4.7 AS6C8016 Control and Data Pins 61
4.8 AS4C8M16S Control and Data Pins 63
4.9 Internal ADC on DAD-MCU-BRD 64
4.10 CS4272 Control Pins 65
4.11 CS4272 Serial Audio Data and Clocks 65
4.12 VGA Connection Pins[10] 71
4.13 Liquid-TFT Display Controller (LTDC) RGB565 Digital Video Output 72

6.1 Comparison between the ADMP401 and 6051 microphones[14][15] 92

A.1 Bill of Materials for DAD Microcontroller Board 105
A.2 Bill of Materials for DAD MEM Board 109
A.3 Bill of Materials for DAD AV Board 113

A.4 Bill of Materials for DAD PINOUT Board 117



Introduction

1.1 What are Audio Analysers and How are They Used

An audio analyser is an electronic measurement tool commonly used by acoustic and sound engineers. Audio analysers provide sound engineers with performance measurements of a sound reinforcement system operating within a specific acoustic environment, such as a concert hall, chapel, amphitheatre or sports stadium. A real-time audio analyser is used to view on a display screen the spectrum or frequency response of the individual sound intensities at selected frequencies within the audible frequency range, namely 20Hz - 20kHz. With the aid of a real-time audio analyser the sound engineer can identity common audio related problems such as over-powering low frequencies, or the detection of high frequency feedback from microphones. The spectrum displayed by the real-time audio analyser allows the sound engineer to obtain the best audio frequency response for the environment by either physically repositioning the loudspeakers of the sound system or through the application of signal processing techniques to the audio signal such as equalization. Ultimately the sound engineer should strive to provide a listener with the best possible audio experience for a sound system operating in a given environment and the real-time audio analyser is the go-to tool for many sound engineers.

1.2 Motivations for the Research Project

A real-time audio analyser is an essential measurement tool used to assess the sound levels at selected frequencies within the audio frequency range of a sound reinforcement system operating within an environment. The cost of a real-time audio analyser corresponds directly to the quality of the audio analysis. Professional hardware audio analysers cost between $1000 and $5000 (Gold Line DSP30[16], Meyer Sound Laboratories SIM 3[17], and Klark Teknik

DN6000[15]). Cheaper yet less sophisticated software implementations can be purchased as software applications ("apps") for smart phones (Apple iPhone, Samsung Galaxy) for a few dollars ($10 - $50). While the cheaper software implementations provide a quick and on-the-go analysis, they lack the professional audio signal chain (professional audio converters, suitable analysis microphones and correctly displayed outputs) found primarily in commercial analysers. Hence, professional audio analysers are not a financially viable option for students, academics and undergraduate sound engineers that wish to study and understand the audio spectrum of a signal or sound system.

1.3 Research Goal and Objectives

The primary objective of this research project is the design of a low-cost hand-held real-time audio analyser capable of displaying the measured audio spectrum of an audio environment to the user. The audio analyser must accept the audio signal to be analysed using an external microphone level input, a line level input, or a high-quality build-in microphone.

A secondary goal of this research project is to review and remove any non-essential hardware costs. The costly LCD screen could be removed and an alternative output viewing display could be developed such as the common Video Graphics Array (VGA) interface. The removal of non-essential hardware costs may allow other hardware features to be incorporated into the audio analyser such as professional 24-bit audio data converters, additional external memory, and/or environment sensors.

The final goal of this project to provide sound educators, audio technicians and engineers with an alternative low-cost real-time audio analysis solution to the currently commercially audio analysis solutions.

In summary, the system requirements are shown in Table 1.1:

System Parameter	**Requirement**
Real Data Input Rate	$\geq$ 48 kHz
Number of Input Bits	$\geq$ 12 bits
Product Size	Small (Hand-held)
Power Supply	Low (1.8V-5V)
Cost	Low
Input Source	Analogue (Microphone)
Output Display	Digital (LCD/VGA)

Table 1.1: Real-Time Audio Analyser System Requirements

It should be noted that there is no clear cut rule or definition as how both the product size and costs are quantified - at least for the purpose of this research project. There is a sense in vagueness which could be open for interpretation as how one can quantify and measure a product cost or product size. Nevertheless, the aim of the designed real-time audio analyser is to be minimal in size and to reduce as much as possible the costs of the product.

1.4 Research Approach

This thesis addresses the development of a low-cost real-time audio analyser in five phases. The five separate phases highlight a developmental plan followed for the implementation of an inexpensive yet dependable real-time audio analysis tool.

- Phase 1 - Research, Theory, and Literature Review
- Phase 2 - Design of Hardware
- Phase 3 - Development of Firmware
- Phase 4 - Implementation of the Real-Time Audio Analyser (RTAA)
- Phase 5 - Investigation and Implementation of Possible Improvements

Each phase provides a clear overview of the path followed by the author in this research project and a full schedule followed by the author during this research project can be obtained upon request.

1.4.1 Phase 1 - Research, Theory and Literature Review

Electronic device manufacturers are often reluctant to reveal the complete inner circuits, workings and schematics of their implemented designs. The first phase is developing an understanding of problem by reviewing previously developed algorithms, techniques and implementations of measuring an audio signal and displaying the correct and desired audio spectrum. Research into the various specifications and features of previously successful implemented audio analyser designs provided an insight into the requirements of a correctly displayed audio spectrum output.

1.4.2 Phase 2 - Design of Hardware

The second phase in the development of a real-time audio analyser is the review of the hardware guidelines from the microcontroller manufacturer and research into the additional external peripheral components such as analogue-to-digital converters (ADC), external memory implementations and different output display options.

The three additional external memory devices that are of interest are:

- Secure Digital Card (SD-Card)
- Static Random Access Memory (SRAM)
- Synchronous Dynamic Random Access Memory (SDRAM)

The two output display devices that are of interest are:

- Thin-Film Transistor Liquid-Crystal Display (TFT-LCD)
- Video Graphics Array (VGA)

1.4.3 Phase 3 - Development of Firmware

The third phase is the development of a suitable digital signal processing algorithm capable of processing an input audio signal from the designed hardware. Once the hardware has buffered and digitised the analogue audio signal, the microcontroller will process the sampled audio signal and output the correctly formatted audio spectrum to the selected display device for interpretation by the user.

1.4.4 Phase 4 - Implementation of the RTAA

The fourth phase combines the designed hardware with the developed firmware algorithms and implements a working prototype for analysing an audio signal such as test audio tones and signals. The implemented real-time audio analyser will also be compared to a commercially available solution using a standard pure tone audio signal and a pink noise audio signal.

1.4.5 Phase 5 - Investigate Possible Improvements

The fifth and final phase discusses both pitfalls and improvements for the designed, developed, and implemented real-time audio analyser. In an ideal world the design, development and implementation of an electronic device such as a real-time audio analyser would undergo several hardware and firmware revisions prior to being made available on the market. Ideally the final electronic device should provide a complete fault-free and bug-free solution to the customer or client.

1.5 Thesis Structure

This thesis is structured in a similar - yet not completely identical - manner to the problem approach with each chapter closely linking each phase. Included in this thesis are several appendices that provide additional background on the digital communication protocols, digital signal processing techniques as well as the final designed schematics, printed circuit boards, and developed firmware code. Moreover, various electronic component datasheets, audio technical manuals, digital signal processing text-books, articles and technical standards are included in a single reference list. A list of both international and local electronic vendors used to complete this research project is provided. Finally, a list of acronyms used in this thesis is included.

CHAPTER 2

Theory and Literature Review

The objective of this chapter is to give the reader some insight into the required background knowledge of the underlying audio theory, research into signal processing algorithms and investigation of electronic hardware. While all attempts have been made to keep this chapter as simple and as concise as possible, the less subtle and more mathematical aspects of the calculations can be obtained from the author upon request.

2.1 Audio Signals and Systems

The properties of sound and the propagation of sound waves through different mediums and environments are well studied in undergraduate physics or applied mathematics. The theory of longitudinal and transverse waves, vibrational physics and even fluid mechanics all provide the framework, theory and models of the fundamental transfer of vibrational energy or rather information through a medium. The analogue representation of an audio signal consisting of currents, voltages, powers and intensities have been explored and experimented for over a century. The digital representation of an audio signal consisting of digital clocks, sampled and binary logic data has been leading the professional audio industry as a new standard in the last few decades.

2.1.1 Types of Audio Signals

An analogue audio signal is measured with a transducer such as a microphone which converts the changes in sound pressure level into an electrical voltage. An audio signal sound pressure level is described as a function over the independent continuous-time variable t and is denoted $x(t)$. A digital audio signal is expressed over the independent time discrete-time variable n and is denoted $x[n]$. Moreover, for a digital signal both the independent variables and the dependant variables are discrete, i.e: both time and amplitude[18].

There are two main types of audio signals:

- Periodic Signals
- Non-Periodic Signals (Random Signals)

As an example, a pure test tone is a periodic signal and is given in Figure 2.1.

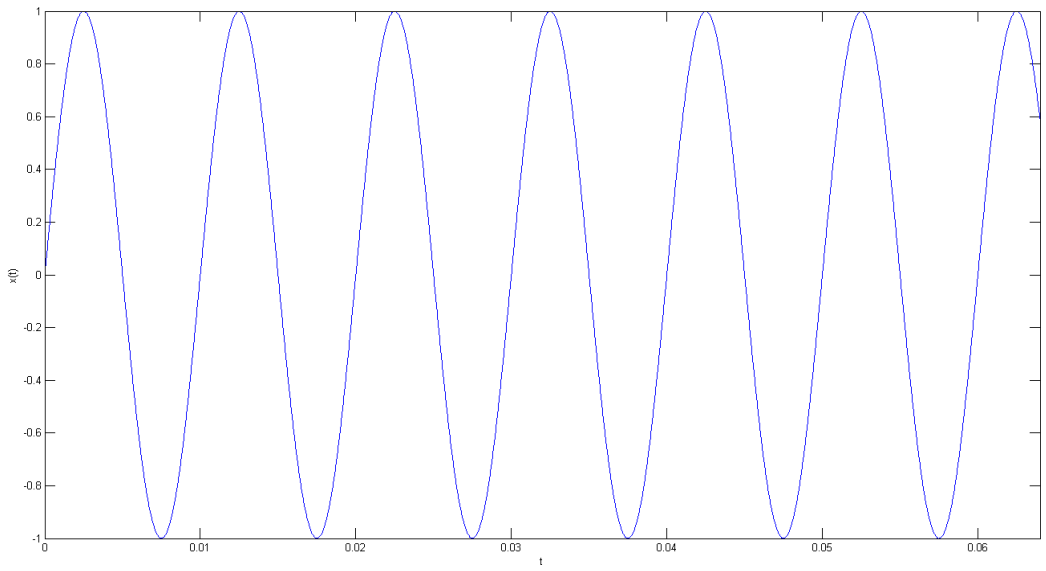


Figure 2.1: Periodic 100Hz test tone signal calculated and plotted using MATLAB

An example of a non-periodic random noise signal is given in Figure 2.2.

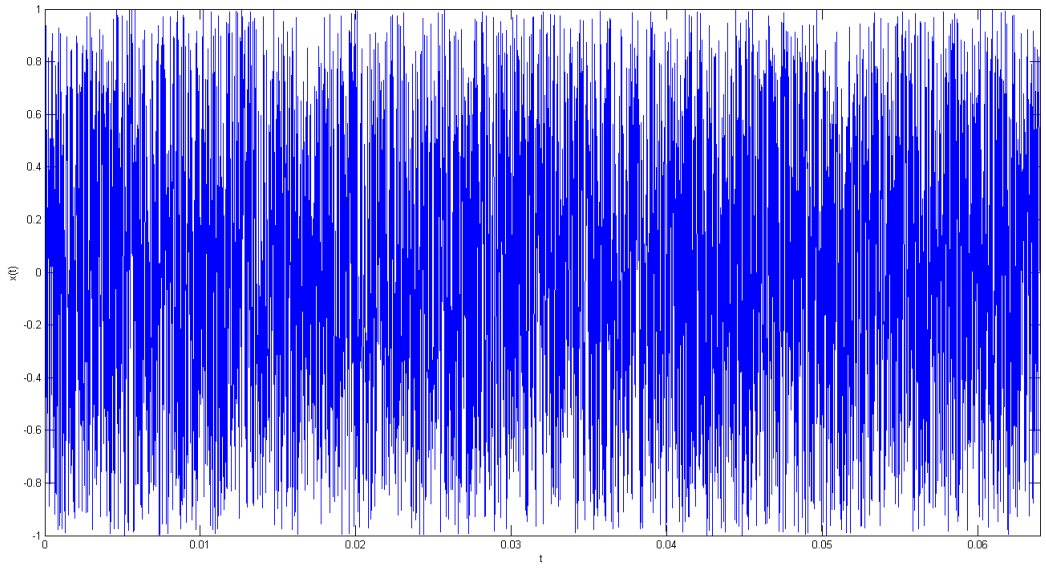


Figure 2.2: Non-periodic random noise signal calculated and plotted using MATLAB

Audio engineers are concerned with three main types of noise and each are attributed to a colour: White, Pink, and Black. The power of a noise signal can be expressed as a homogeneous power-law with a frequency dependence and is given by[19]:

$$P_{noise} \propto f^{-\beta} \tag{2.1}$$

where P_{noise} is the power of the noise signal, f is the frequency, and β is a constant.

2.1.2 White Noise

White noise consists of constant and equal energy per frequency bandwidth[20]. The power spectrum of white noise is constant or 0dB/octave (0dB/decade) on a linear scale and therefore when analysed with a constant-Q audio analyser the display exhibits an increase by 3dB/octave (10dB/decade)[20].

The power of white noise can be expressed with the power law relation where $\beta = 0$ and is given by[19]:

$$P_{whitenoise} \propto f^{-\beta} \tag{2.2}$$
$$\propto f^{-0} \tag{2.3}$$
$$\propto 1 \tag{2.4}$$

2.1.3 Pink Noise

Pink noise consists of constant and equal energy per octave[20]. The power spectrum of pink noise decreases by 3dB/octave (10dB/decade) on a linear scale and therefore when analysed with a constant-Q audio analyser the display exhibits a flat spectrum[20].

The power of pink noise can be expressed with the power law relation where $\beta = 1$ and is given by[19]:

$$P_{pinknoise} \propto f^{-\beta} \tag{2.5}$$
$$\propto f^{-1} \tag{2.6}$$
$$\propto \frac{1}{f} \tag{2.7}$$

Pink noise can be generated by passing white noise through a filter that has a transfer function given by[21]:

$$H(s) = \frac{K}{\sqrt{s}} \tag{2.8}$$

where K is an arbitrary constant.

The frequency response function for Pink noise is given by[21]:

$$H(j\omega) = \frac{K}{\sqrt{j\omega}} \tag{2.9}$$
$$= \frac{K}{\sqrt{\omega}} e^{-j\frac{\pi}{4}} \tag{2.10}$$

where K is an arbitrary constant.

2.1.4 Black Noise

Black noise is often neglected and overlooked in the family of noise colours. Black noise is quite simply silence[20] and consists of zero energy per frequency.

The power of black noise can be expressed with the power relation where $\beta > 2$[19]. As an example, the power of black noise can be expressed with the power law relation where $\beta = 3$ and is given by[19]:

$$P_{blacknoise} \propto f^{-\beta} \tag{2.11}$$

$$\propto f^{-3} \tag{2.12}$$

$$\propto \frac{1}{f^3} \tag{2.13}$$

2.1.5 Audio Equalization

Audio equalization is a process used to alter the frequency response of an audio signal with the use of signal processing techniques such as passing an audio signal through a system of linear filters.

A graphic equalizer allows the sound engineer to add (boost) and/or remove (cut) frequency bands from an input audio signal. A graphic equalizer can be modelled as bank of bandpass filters spread across a logarithmically spaced audio frequency range with a centre frequency f_0, and a bandwidth B. The relationship between the bandwidth B, the range of applicable frequencies, f_1 and f_2, and the centre frequency f_0 is termed the quality factor, Q. The quality factor is given by the following equation:

$$Q = \frac{f_0}{B} \tag{2.14}$$

$$= \frac{f_0}{f_2 - f_1} \tag{2.15}$$

An example of an audio graphic equalizer is the DN360 manufactured by Klark Teknik and is shown in Figure 2.3. The DN360 audio graphic equalizer is a dual channel, 30 band equalizer featuring up to 12dB of cut/boost for frequencies between 25Hz and 20kHz in $\frac{1}{3}$ octave steps[1].

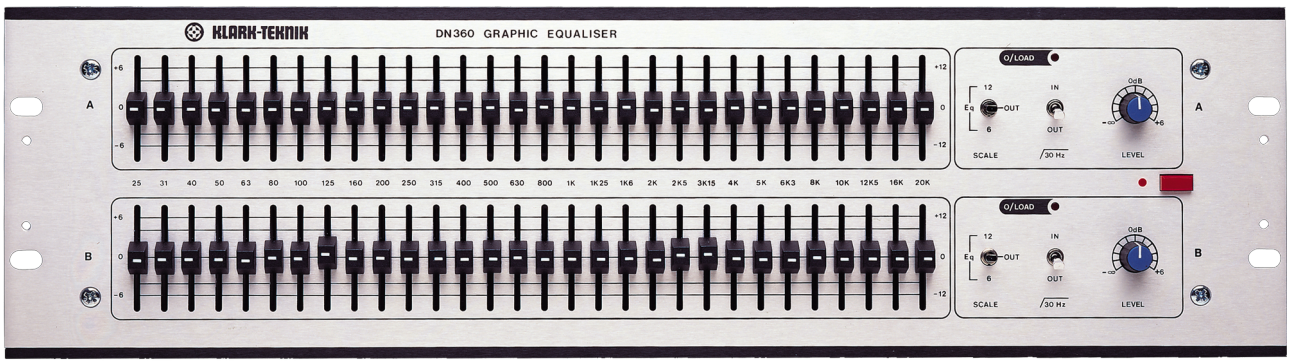


Figure 2.3: Front view of DN360 $\frac{1}{3}$ octave graphic equaliser[1]

The frequency spacing is non-linear yet repetitive with a constant quality factor for each band-pass filter. The centre frequencies for the 30 bands of $\frac{1}{3}$ octave spaced frequency steps are determined according to an international standard. For the purposes of this academic research project a series of out-dated standards were viewed and are referenced appropriately.

The relationship between the centre frequency f_{cent}, the higher f_{high}, and lower f_{low} boundary or bandedge frequencies is given by the ANSI S1.11-2004: Specification for Octave-Band and Fractional-Octave-Band Analog and Digital Filters[22]. There are two octave ratio's specified in the ANSI S1.11-2004 document, namely, the base-10 ratio:

$$G_{10} = 10^{\frac{3}{10}} \approx 1.9953 \tag{2.16}$$

and the base-2 ratio:

$$G_2 = 2 \tag{2.17}$$

The base-10 system is preferred[22]. Once a system base ratio has been chosen the centre frequency, f_{cent}, and bandedge frequencies, f_{low} and f_{high}, can be calculated for a reference frequency, f_{ref}, with the fraction of an octave band defined by the bandwidth designator, $\frac{1}{B}$.

For a reference frequency $f_{ref} = 1000kHz$, the band centre frequency f_{cent} and the bandedge frequencies can be calculated for each band, x, with a $\frac{1}{3}$ bandwidth designator by[22]:

$$f_{cent}(x) = (G^{\frac{x-16}{B}})f_{ref} \tag{2.18}$$

$$f_{low}(x) = (G^{-\frac{1}{2B}})(f_{cent}(x)) \tag{2.19}$$

$$f_{high}(x) = (G^{+\frac{1}{2B}})(f_{cent}(x)) \tag{2.20}$$

where $0 \leq x < 10 \times B$ and $B = 3$.

The 30 band $\frac{1}{3}$ octave spaced centre frequencies with bandedge frequencies are given in Table 2.1.

n	Low Freq (Hz)	Centre Freq (Hz)	High Freq (Hz)	Bandwidth (Hz)	Q
0	22.10	24.80	27.84	5.74	4.32
1	27.84	31.25	35.08	7.24	4.32
2	35.08	39.37	44.19	9.12	4.32
3	44.19	49.61	55.68	11.49	4.32
4	55.68	61.50	70.15	14.47	4.32
5	70.15	78.75	88.39	18.23	4.32
6	88.39	99.213	111.36	22.97	4.32
7	111.36	125.00	140.31	28.95	4.32
8	140.31	157.49	176.78	36.47	4.32
9	176.78	198.43	222.72	45.95	4.32
10	222.72	250.00	280.62	57.89	4.32
11	280.62	314.98	353.55	72.94	4.32
12	353.55	396.85	445.45	91.90	4.32
13	445.45	500.00	561.23	115.78	4.32
14	561.23	629.96	707.11	145.88	4.32
15	707.11	793.70	891.00	183.79	4.32
16	891.00	1000.00	1122.46	231.56	4.32
17	1122.46	1259.92	1414.21	291.75	4.32
18	1414.21	1587.40	1781.80	367.58	4.32
19	1781.80	2000.00	2244.92	463.13	4.32
20	2244.92	2519.84	2828.43	583.50	4.32
21	2828.43	3174.80	3563.59	735.17	4.32
22	3563.59	4000.00	4489.85	926.25	4.32
23	4489.85	5039.68	5656.85	1167.01	4.32
24	5656.85	6349.60	7127.19	1470.34	4.32
25	7127.19	8000.00	8979.70	1852.51	4.32
26	8979.70	10079.37	11313.71	2334.01	4.32
27	11313.71	12699.21	14254.38	2940.67	4.32
28	14254.38	16000.00	17959.39	3705.01	4.32
29	17959.39	20158.74	22627.42	4668.02	4.32

Table 2.1: 30 Bands of $\frac{1}{3}$ octave spaced steps calculated using the STM32F429 microcontroller

2.1.6 The Human Auditory System

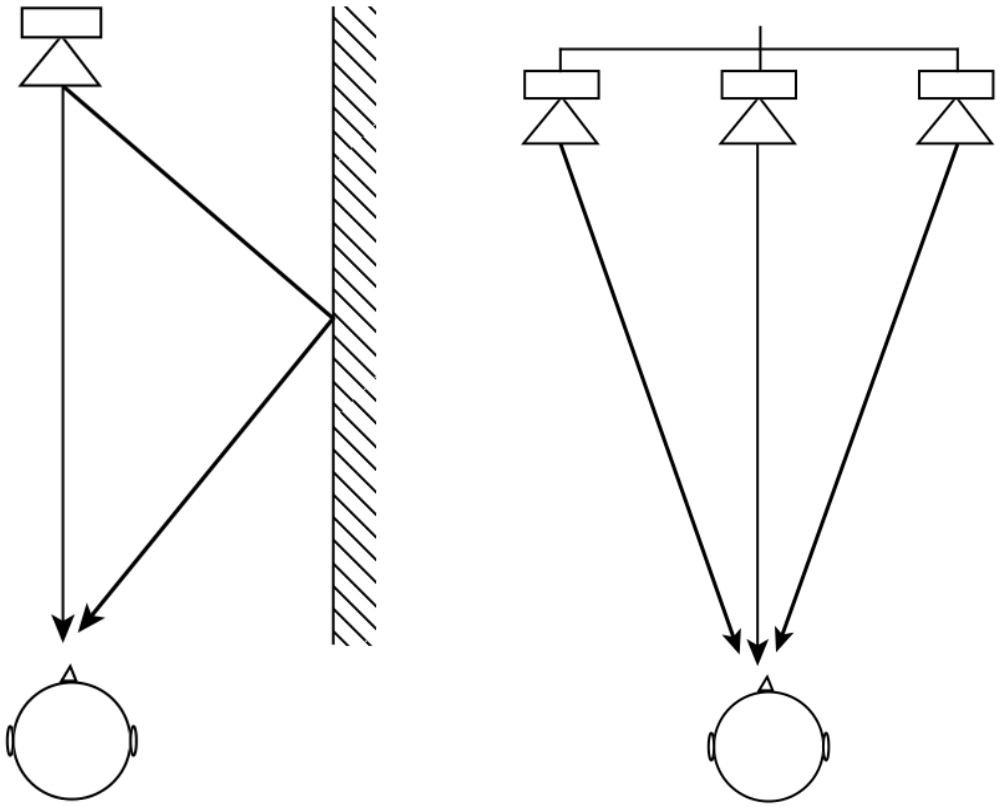
The human hearing system provides a pathway from the external body world through a system of membrane tissues (outer ear), mechanical bone levers (middle ear) and auditory nerve transducer (inner ear) to the brain. The separation of the human hearing system into three distinct different functions provides additional insight into the complex mechanisms with which a listener hears an incoming audio signal from the external environment. Each stage of the human auditory system analyses the audio signal for specific audio properties such as the location of the incoming audio source, the distinct frequency components and the perceived loudness of the audio sound. The complex audio processing by the human hearing system provides the often misguided view that the frequency components are analysed in linearly spaced frequency bands and at equally perceived loudness over the audio spectrum (20Hz - 20kHz).

2.1.7 Equivalent Sound Level Addition

In the ideal scenario, an incoming sound wave does not reflect and interact with objects within the environment and arrives at the listener as a single correlated audio source. In the practical scenario, an incoming sound wave reflects and interacts with objects within the environment resulting in a natural reverberation audio effect on the sound signal which may be interpreted by the human hearing system as a single uncorrelated source. The two different ways of adding and interpreting audio levels from separate sound sources are termed as coherent or correlated level addition, and incoherent or uncorrelated level addition.

2.1.7.1 Coherent and Correlated Audio Sources

An audio source is termed **coherent** or **correlated** with other audio sources if there is either a negligible timing difference between the sources or if the point of origination for each audio source is singular[23]. Figure 2.4 illustrates a coherent or correlated audio source. An audio wave produced by a single source reflects and interacts with the environment producing a negligible time delay or reverberation effect. Moreover, a single audio source which is then split into multiple audio sources such as a public address system or live sound reinforcement system is also considered to be correlated[2].



(a) Correlation due to reflection **(b)** Correlation due to multiple sources

Figure 2.4: Addition of correlated audio sources[2]

A set of coherent or correlated audio sources can be very crudely considered to be a set of very

much the same source with identical frequencies and phase differences. The sum or addition of several coherent or correlated audio sources, L_{corr}, is given by:

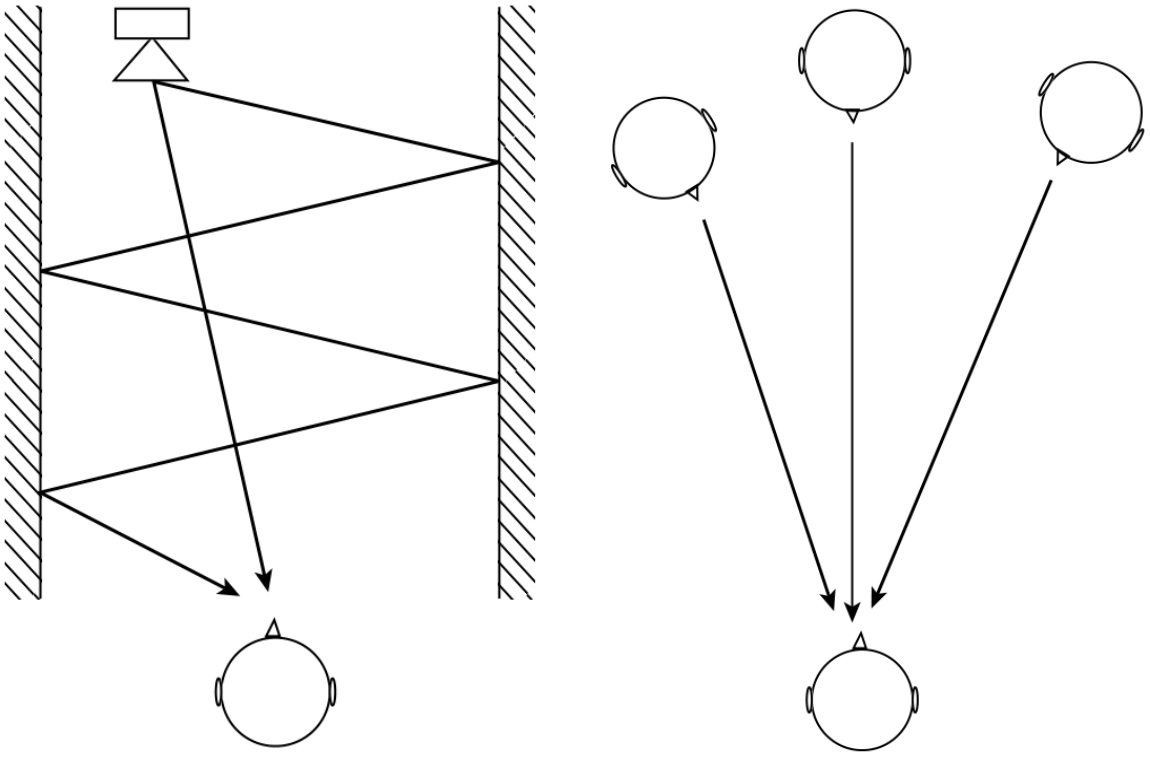
$$L_{corr} = L_1 + \cdots + L_N \tag{2.21}$$

All the individual coherent or correlated sources have the same frequency and a minimal phase difference to other audio sources. A special case of coherent or correlated audio sources exists if each source is of the same level L. The equation for coherent or correlated addition simplifies as follows:

$$\begin{aligned} L_{corr} &= L_1 + \cdots + L_N && (2.22)\\ &= L + \cdots + L \qquad (N \text{ times}) && (2.23)\\ &= L \times N && (2.24) \end{aligned}$$

2.1.7.2 Incoherent and Uncorrelated Audio Sources

An audio source is termed **incoherent** or **uncorrelated** with other audio sources if there is either a significant timing difference between the audio sources or if the point of origination for each audio source is non-singular[23]. Figure 2.5 illustrates an incoherent or uncorrelated audio source. An audio wave produced by a single source reflects and interacts with the environment producing a significant time delay or reverberation effect. Moreover, a set of multiple audio sources consisting of set of distinct audio waves such as a choir or musical instruments arriving at a singular point is also considered to be uncorrelated[2].



(a) Uncorrelated reflection due to long delay **(b)** Uncorrelated multiple sources

Figure 2.5: Addition of uncorrelated audio sources[2]

A set of incoherent or uncorrelated audio sources can be very crudely considered to be a set of distinct or unrelated audio sources with different frequencies and phase differences. The sum or addition of several incoherent or uncorrelated audio sources, L_{uncorr}, is given by:

$$L_{uncorr} = \sqrt{\left((L_1)^2 + \cdots + (L_N)^2\right)} \tag{2.25}$$

All the individual incoherent or uncorrelated sources have different frequencies and phase differences to other audio sources. A special case of incoherent or uncorrelated audio sources exists if each source is of the same level L. The equation for incoherent or uncorrelated addition simplifies as follows:

$$\begin{aligned} L_{uncorr} &= \sqrt{(L_1)^2 + \cdots + (L_N)^2} && (2.26)\\ &= \sqrt{(L)^2 + \cdots + (L)^2} && (2.27)\\ &= \sqrt{L^2 \times N} \qquad (N \text{ times}) && (2.28)\\ &= L\sqrt{N} && (2.29) \end{aligned}$$

2.2 Fourier Transformations

The Fourier Transformations presented here are by no means the only mathematical transformations that were investigated during the research in audio signal processing techniques. Two examples of such transformations are the Hartley Transform and the Constant-Q Transform, both very similar yet very different to the presented Fourier Transforms. Moreover, the Fourier Transforms are primarily useful for quasi-static or stationary signals. Therefore, the analysis of a chirp signal using the Fourier Transforms directly provides very little information on the change in frequency of the signal over a time duration. Alternative techniques such as time-frequency transformations were briefly explored however additional future research will be required to implement a time-frequency display such as an audio spectrogram or waterfall spectrum.

2.2.1 Continuous-Time Fourier Transform (CTFT)

The Continuous-Time Fourier Transform (CTFT) is the formal mathematical description for the continuous frequency-domain spectrum of a continuous-time signal $x(t)$.

The CTFT of the continuous-time signal $x_c(t)$ is defined as follows[11]:

$$X_c(j\omega) = \int_{-\infty}^{+\infty} x_c(t) e^{-j\omega t} \mathrm{d}t \tag{2.30}$$

2.2.2 Discrete-Time Fourier Transform (DTFT)

The Discrete-Time Fourier Transform (DTFT) is a continuous frequency-domain representation of the discrete-time signal $x[n]$.

The DTFT of the discrete-time signal $x[n]$ is defined by[11]:

$$X(e^{j\hat{\omega}}) = \sum_{n=-\infty}^{+\infty} x[n]e^{-j\hat{\omega}n} \tag{2.31}$$

where $\hat{\omega}$ is a continuous-frequency variable and $X(e^{j\hat{\omega}})$ is periodic with period 2π.

The discrete-time sequence $x[n]$ can be obtained by sampling a continuous-time signal $x_c(t)$ with a sampling period T_s by:

$$x[n] = x_c(nT_s) \tag{2.32}$$

2.2.3 Discrete Fourier Transform (DFT)

The Discrete Fourier Transform (DFT) is a discrete frequency-domain representation of the discrete-time sequence $x[n]$.

The DFT of the discrete-time finite-duration sequence $x[n]$ is defined as follows[11]:

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-\frac{2\pi nk}{N}} \tag{2.33}$$

where $0 \leq k \leq N-1$.

Alternatively, the DFT of the discrete-time finite-duration sequence $x[n]$ can be defined by:

$$X[k] = \sum_{n=0}^{N-1} x[n]W_N^{nk} \tag{2.34}$$

where $W_N^{nk} = e^{-\frac{j2\pi nk}{N}} = \cos\left(\frac{2\pi nk}{N}\right) - j\sin\left(\frac{2\pi nk}{N}\right)$ and $0 \leq k \leq N-1$.

It should be noted that the frequency domain for the result of an N-point DFT applied to a sampled discrete-time finite-duration sequence can be determined by:

$$f_i = \frac{Sample\ Rate}{Sample\ Size} \times i \tag{2.35}$$

Thus, the frequency divisions are dependant on both the rate of the sampled data and the total number of data samples.

Table 2.2 illustrates the differences between the Fourier Transformations, namely the Continuous-Time Transform (CTFT), Discrete-Time Fourier Transform (DTFT), and the Discrete Fourier Transform (DFT).

	Discrete-Time	**Continuous-Time**
Discrete-Frequency	DFT, $X[k]$	Fourier Series, $\{a_k\}$
Continuous-Frequency	DTFT, $X(e^{j\hat{\omega}})$	CTFT, $X_c(j\omega)$

Table 2.2: Types of Fourier Transforms[11]

2.3 Filters

The filters presented here have been coded appropriately and are features of the implemented real-time audio analyser. Data windowing functions are commonly studied in digital signal processing courses and are a necessity when analysing digitally sampled signals. Audio weighting filters are found in many audio analysing devices such as an audio analyser and are a necessity when analysing a audio signals.

2.3.1 Digital Signal Processing Windows

A digital signal processing window $w[n]$ is a weighting function that is multiplied with an input digital signal $x[n]$ prior to computing the discrete fourier transform (DFT) of the windowed input signal $x_w[n]$.

$$x_w[n] = x[n] \times w[n] \tag{2.36}$$

where $0 \leq x \leq N - 1$.

The application a window to the input sample results in the reduction of the side-lobes and a widening of the main lobe without altering the centre frequency of each DFT filter[24]. Extensive research has been done on various types and flavours of digital windowing functions[25] and those featured on the implemented real-time audio analyser are discussed in more detail.

It should be noted that the digital window functions are used as a mechanism to smooth the audio signal before performing the audio analysis. The digital audio signal is analysed in short time intervals which is governed by the sampling rate frequency of the analogue to digital converter. The application of a window function to the sampled audio signal results in the suppression of the distant samples and highlights the central samples in the short time interval.

2.3.1.1 Rectangular Window

The most fundamental window is the rectangular (or boxcar) window which does not alter the windowed input signal in any manner. The rectangular window is given by:

$$w[n] = 1 \tag{2.37}$$

where $0 \leq n \leq N - 1$[24].

The rectangular window is given in Figure 2.6 for 4096 data points.

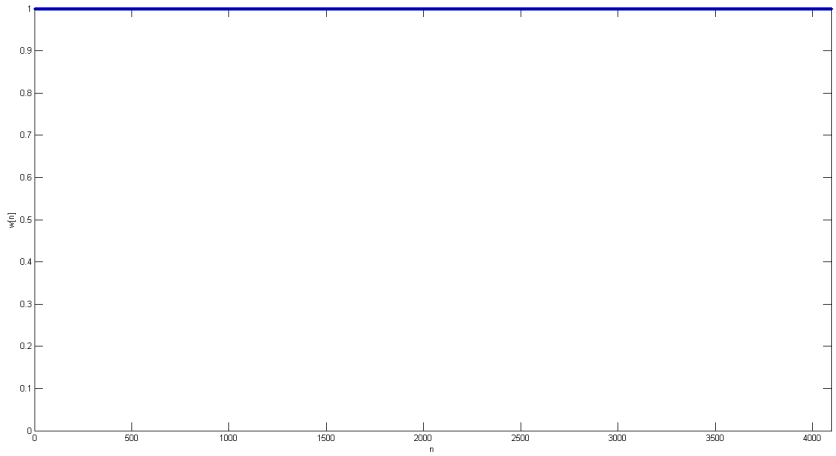


Figure 2.6: Rectangle window function for 4096 data points calculated and plotted with MATLAB

2.3.1.2 Hanning Window

The Hanning window is given by:

$$w[n] = \frac{1}{2}\Big(1 - \cos\big(\frac{2n\pi}{N}\big)\Big) \tag{2.38}$$

where $0 \leq n \leq N - 1$[24].

The Hanning window is given in Figure 2.7 for 4096 data points.

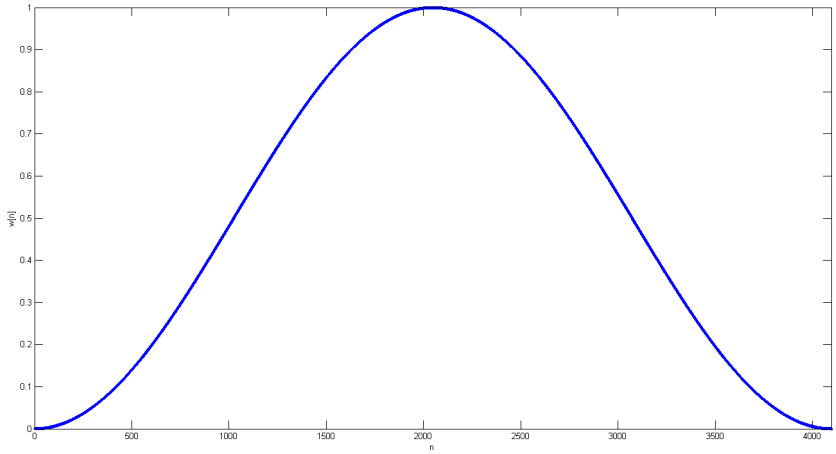


Figure 2.7: Hanning window function for 4096 data points calculated and plotted with MATLAB

2.3.1.3 Hamming Window

The Hamming window is given by:

$$w[n] = 0.54 - 0.46\cos\big(\frac{2n\pi}{N}\big) \tag{2.39}$$

where $0 \leq n \leq N - 1$[24].

The Hamming window is given in Figure 2.8 for 4096 data points.

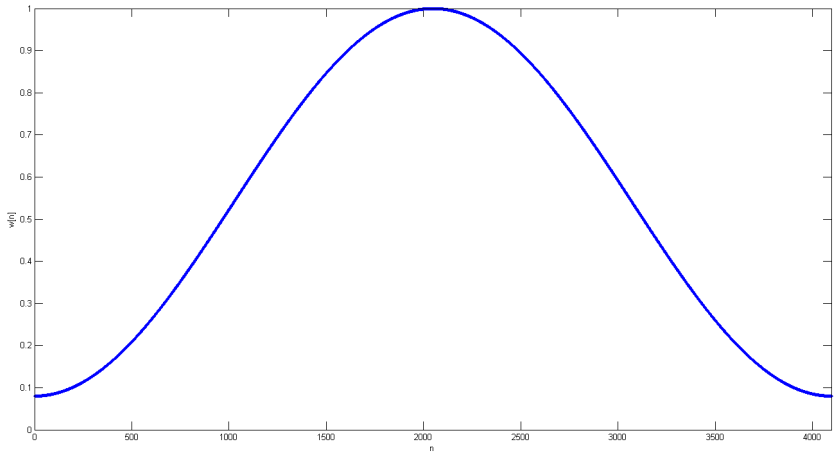


Figure 2.8: Hamming window function for 4096 data points calculated and plotted with MATLAB

2.3.1.4 Welch Window

The Welch window is given by:

$$w[n] = 1 - \left(\frac{n - \frac{N}{2}}{\frac{N}{2}} \right)^2 \tag{2.40}$$

where $0 \leq n \leq N - 1$[26].

The Welch window function is given in Figure 2.9 for 4096 data points.

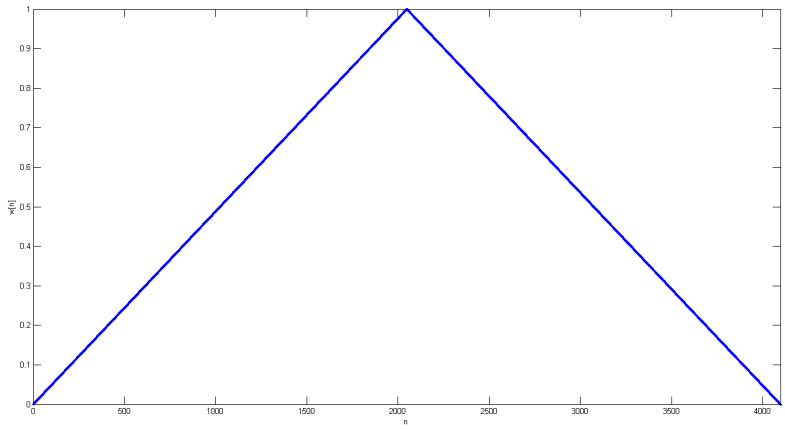


Figure 2.9: Welch window function for 4096 data points calculated and plotted with MATLAB

2.3.1.5 Bartlett Window

The Bartlett window is given by:

$$w[n] = 1 - \left| \frac{n - \frac{N}{2}}{\frac{N}{2}} \right| \tag{2.41}$$

where $0 \leq n \leq N-1$[26].

The Bartlett window is given in Figure 2.10 for 4096 data points.

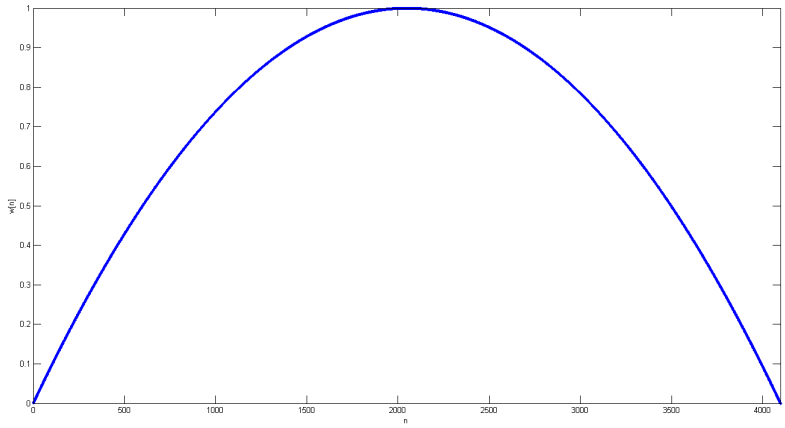


Figure 2.10: Bartlett window function for 4096 data points calculated and plotted with MATLAB

2.3.1.6 Blackman Window

The Blackman window is given by:

$$w[n] = 0.42 - 0.50\left(\cos\left(\frac{2\pi n}{N}\right)\right) + 0.08\left(\cos\left(\frac{4\pi n}{N}\right)\right) \tag{2.42}$$

where $0 \leq n \leq N-1$[24].

The Blackman window is given in Figure 2.11 for 4096 data points.

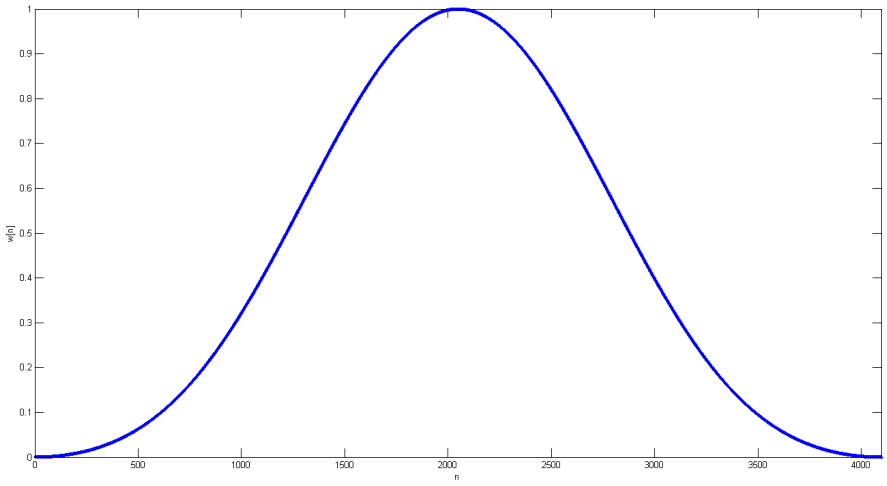


Figure 2.11: Blackman window function for 4096 data points calculated and plotted with MATLAB

It should be noted that while only six data windowing functions have been presented here, there are many others types such as Gaussian, Kasier, and Dolph-Chebyshev. The Rectangle,

Hanning, Hamming, Welch, Bartlett and Blackman windows are the most common data windowing functions found in digital signal analysers as well as being the most common studied by students in a digital signal precessing course.

2.3.2 Audio Weighting Filters

Ultimately the listener will use their ears - the human auditory system - to hear an audio signal or source. Audio measuring devices such as a real-time audio analyser are often designed with the underlying principle of providing a linear frequency response. As a result, an additional weighting filter is commonly applied to the results of an audio analysis which in turn compensates for the non-linearity of the human auditory system. The audio signal is first passed through an appropriate weighting-filter before the root mean square (RMS) or peak value is calculated[27].

From a signal processing point of view, a Linear Time-Invariant (LTI) system or filter can be characterised by its Transfer Function (TF) in the the analogue s-domain or corresponding digital z-domain. The transfer function for each weighting filter can be determined from the contents of the logarithm in the filter weighting level equation and by expressing the frequency f as an angular frequency $\omega = 2\pi f$ and units rad/s.

Each weighting filter has a characteristic pole-zero graphical representation, also called a complex plane pole-zero plot. The pole-zero plot for each weighting filter can be viewed as a flat rubber sheet spanning infinitely on the flat 2-D plane. The flat rubber sheet is initially raised in the z-axis by a fixed constant value: K_A, K_B, K_C. A pole placed at a particular point underneath the flat rubber sheet will cause the previously flat transfer function to be altered and attenuate certain frequencies: f_0, f_1, f_2, f_3, f_4, f_5. A zero placed at a particular point fixes or nails the flat rubber sheet to remain unaltered while points around the zero are free to be altered by any neighbouring poles.

2.3.2.1 A-Weighting

The most common audio frequency weighting filter is the A-weighting filter and is given by[28]:

$$W_A(f) = 20\log\left(\frac{f_4^2 f^4}{(f^2+f_1^2)(f^2+f_2^2)^{\frac{1}{2}}(f^2+f_3^2)^{\frac{1}{2}}(f^2+{f_4}^2)}\right) + K_A \tag{2.43}$$

where $K_A = 1.9997$, $f_1 = 20.598997\,Hz$, $f_2 = 107.65265\,Hz$, $f_3 = 737.86223\,Hz$, and $f_4 = 12194.22\,Hz$.

The transfer function for the A-weighting filter is given by[29]:

$$H_A(s) = \frac{\omega_4^2 s^4}{(s+\omega_1)^2(s+\omega_2)(s+\omega_3)(s+\omega_4)^2} \tag{2.44}$$

where $s = j2\pi f$, $\omega_1 = 129.43\,rad/s$, $\omega_2 = 676.40\,rad/s$, $\omega_3 = 4636.13\,rad/s$, and $\omega_4 = 76618.54\,rad/s$.

Moreover, the transfer function for the A-weighting filter can be expanded and viewed as three cascaded filters:

$$H_A(s) = \frac{\omega_4^2 s^4}{(s+\omega_1)^2(s+\omega_2)(s+\omega_3)(s+\omega_4)^2} \tag{2.45}$$

$$= \frac{\omega_4 s}{(s+\omega_1)^2} \times \frac{s}{(s+\omega_2)} \times \frac{s}{(s+\omega_3)} \times \frac{\omega_4 s}{(s+\omega_4)^2} \tag{2.46}$$

Hence, a digital implementation of the A-weighting filter could be achieved with the aid of the bilinear transform and its relation to a digital biquad filter with poles and zeros located at frequencies f_0, f_1, f_2, f_3, and f_4:

- 4 zeros at $f_0 = 0Hz$
- 2 poles at $f_1 = 20.598997\,Hz$
- 1 pole at $f_2 = 107.65265\,Hz$
- 1 pole at $f_3 = 737.86223\,Hz$
- 2 poles at $f_4 = 12194.22\,Hz$

2.3.2.2 B-Weighting

The B-weighting filter is given by[28]:

$$W_B(f) = 20\log\left(\frac{f_4^2 f^3}{(f^2+f_1^2)(f^2+f_4^2)(f^2+f_5^2)^{\frac{1}{2}}}\right) + K_B \tag{2.47}$$

where $K_B = 0.1696$, $f_1 = 20.598997\,Hz$, $f_4 = 12194.22\,Hz$, and $f_5 = 158.48932\,Hz$.

The transfer function for the B-weighting filter in the s-domain is given by[29]:

$$H_B(s) = \frac{\omega_4^2 s^3}{(s+\omega_1)^2(s+\omega_4)^2(s+\omega_5)} \tag{2.48}$$

where $s = j2\pi f$, $\omega_1 = 129.43\,rad/s$, $\omega_4 = 76618.54\,rad/s$, and $\omega_5 = 995.82\,rad/s$.

Again, the transfer function for the B-weighting filter can be expanded and viewed as three cascaded filters:

$$H_B(s) = \frac{\omega_4^2 s^3}{(s+\omega_1)^2(s+\omega_4)^2(s+\omega_5)} \tag{2.49}$$

$$= \frac{\omega_4 s}{(s+\omega_1)^2} \times \frac{\omega_4 s}{(s+\omega_4)^2} \times \frac{s}{(s+\omega_5)} \tag{2.50}$$

Hence, a digital implementation of the B-weighting filter could be obtained with the aid of the bilinear transform and its relation to a digital biquad filter with poles and zeros located at frequencies f_0, f_1, f_4, and f_5:

- 3 zeros at $f_0 = 0Hz$
- 2 poles at $f_1 = 20.598997\,Hz$
- 1 pole at $f_5 = 158.48932\,Hz$
- 2 poles at $f_4 = 12194.22\,Hz$

2.3.2.3 C-Weighting

The C-weighting filter is given by:

$$W_C(f) = 20\log\left(\frac{f_4^2 f^2}{(f^2+f_1^2)(f^2+f_4^2)}\right) + K_C \tag{2.51}$$

where $K_C = 0.0619$, $f_1 = 20.598997\,Hz$, and $f_4 = 12194.22\,Hz$[28].

The transfer function for the C-weighting filter in the s-domain is given by[29]:

$$H_C(s) = \frac{\omega_4^2 s^2}{(s+\omega_1)^2(s+\omega_4)^2} \tag{2.52}$$

where $s = j2\pi f$, $\omega_1 = 129.43\,rad/s$, and $\omega_4 = 76618.54\,rad/s$.

Again, the transfer function for the C-weighting filter can be expanded and viewed as two cascaded filters:

$$H_C(s) = \frac{\omega_4^2 s^2}{(s+\omega_1)^2(s+\omega_4)^2} \tag{2.53}$$

$$= \frac{\omega_4 s}{(s+\omega_1)^2} \times \frac{\omega_4 s}{(s+\omega_4)^2} \tag{2.54}$$

Hence, a digital implementation of the C-weighting filter could be obtained with the aid of the bilinear transform and its relation to the digital biquad filter with poles and zeros located at frequencies f_0, f_1, and f_4:

- 2 zeros at $f_0 = 0Hz$
- 2 poles at $f_1 = 20.598997Hz$
- 2 poles at $f_4 = 12194.22Hz$

2.3.2.4 Z-Weighting

The Z-weighting filter is sometimes excluded from the family of audio weighing filters commonly found on audio analysing and measuring devices. The Z-weighting is a flat frequency response filter and is sometimes referred to as the zero frequency weighting or flat weighting[30].

The Z-weighting filter is given by:

$$W_Z(f) = 0 \tag{2.55}$$

2.3.2.5 Comparison between the A-, B-, C-, Z-Weighting Filters

The audio attenuation level for each of the audio weighting filter discussed in sections2.3.2.1, 2.3.2.2, 2.3.2.3, 2.3.2.4 can be computed with the 30 band $\frac{1}{3}$ centre frequencies given in Table 2.1. Table 2.3 shows the audio weighting filter attenuation values in decibels calculated on the STM32F429 microcontroller using equations 2.43, 2.47, 2.51, 2.55.

n	**Freq [Hz]**	**A-Weight [dB]**	**B-Weight [dB]**	**C-Weight [dB]**	**Z-Weight [dB]**
0	24.80	-45.01	-20.60	-4.49	0.00
1	31.25	-39.70	-17.23	-3.07	0.00
2	39.37	-34.85	-14.29	-2.04	0.00
3	49.61	-30.42	-11.71	-1.32	0.00
4	61.50	-26.35	-9.44	-0.83	0.00
5	78.75	-22.64	-7.44	-0.51	0.00
6	99.213	-19.25	-5.70	-0.31	0.00
7	125.00	-16.19	-4.23	-0.17	0.00
8	157.49	-13.42	-3.02	-0.09	0.00
9	198.43	-10.93	-2.07	-0.03	0.00
10	250.00	-8.67	-1.36	0.00	0.00
11	314.98	-6.64	-0.85	0.02	0.00
12	396.85	-4.83	-0.51	0.03	0.00
13	500.00	-3.25	-0.28	0.03	0.00
14	629.96	-1.91	-0.13	0.03	0.00
15	793.70	-0.83	-0.04	0.02	0.00
16	1000.00	0.00	0.00	0.00	0.00
17	1259.92	0.59	0.01	-0.03	0.00
18	1587.40	0.98	-0.02	-0.09	0.00
19	2000.00	1.20	-0.09	-0.17	0.00
20	2519.84	1.27	-0.21	-0.30	0.00
21	3174.80	1.20	-0.41	-0.51	0.00
22	4000.00	0.96	-0.73	-0.83	0.00
23	5039.68	0.54	-1.20	-1.31	0.00
24	6349.60	-0.14	-1.92	-2.02	0.00
25	8000.00	-1.15	-2.94	-3.05	0.00
26	10079.37	-2.55	-4.35	-4.46	0.00
27	12699.21	-4.40	-6.21	-6.32	0.00
28	16000.00	-6.71	-8.53	-8.63	0.00
29	20158.74	-9.45	-11.27	-11.38	0.00

Table 2.3: Audio Weighting functions for 30 bands of $\frac{1}{3}$ octave spaced steps calculated using the STM32F429 microcontroller

Figure 2.12 is a plot of the audio weighting filter attenuation values in decibels calculated and plotted with MATLAB using equations 2.43, 2.47, 2.51, 2.55.

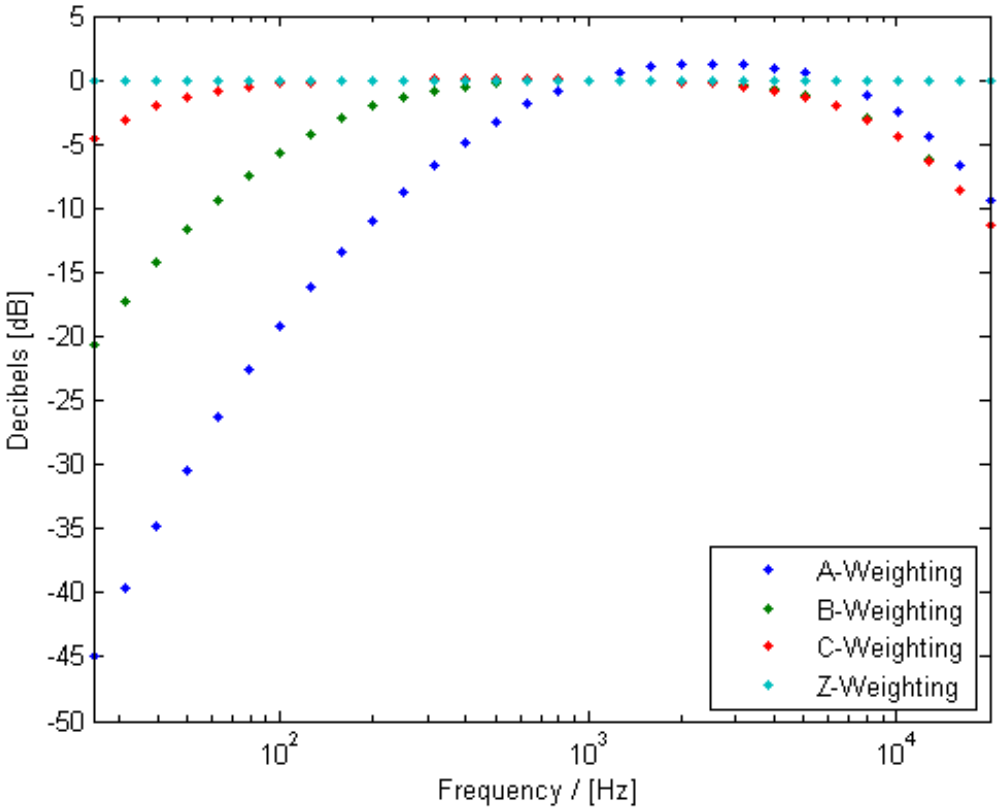


Figure 2.12: Audio weighting functions for 30 bands of $\frac{1}{3}$ octave spaced steps calculated and plotted using MATLAB

Table 2.3 and Figure 2.12 illustrate the audio attenuation levels for each of the audio weighting filters discussed in sections 2.3.2.1, 2.3.2.2, 2.3.2.3,and 2.3.2.4.

2.4 Simultaneous Real-Valued Fast-Fourier Transform (FFT)

This section discusses the underlying mathematical calculations for the audio signal processing algorithm. The developed audio signal processing algorithm is also discussed from a coding point of view in Chapter 5.

Since the input samples are real-valued and not inherently complex-valued, the imaginary part of the complex number is often equated to zero before computing the DFT with an appropriate FFT algorithm. This results in the output of the FFT being mirrored across the frequency domain, and often half of the FFT results are discarded. This method of computing the DFT of a real-valued input may be an option if computing time, memory space and power are not limiting factors of the hardware. An alternative method for determining the FFT of input samples that are real-valued is implemented by the designed real-time audio analyser presented in this research thesis. The algorithm for computing the simultaneous real-valued FFT of sampled data with size 2N is discussed in several referenced sources[24], [26], [31], [32], [33].

Consider $g[n]$, a single real-valued sample of data with a sample size of 2N. The single real-valued sample of data $g[n]$ can be split into $x_1[n]$ and $x_2[n]$, two separate real-valued samples of size N to form the complex-valued input sample $x[n]$ of size N.

The two real-valued samples, $x_1[n]$ and $x_2[n]$ each with sample size N, are defined by the following two equations:

$$x_1[n] = g[2n] \tag{2.56}$$
$$x_2[n] = g[2n+1] \tag{2.57}$$

where $0 \leq n \leq N-1$.

Therefore the complex-valued input $x[n]$ is given by:

$$x[n] = x_1[n] + jx_2[n] \tag{2.58}$$
$$= g[2n] + jg[2n+1] \tag{2.59}$$

where $0 \leq n \leq N-1$.

The DFT of the complex-valued input $x[n]$ is given by:

$$X[k] = \sum_{n=0}^{N-1} x[n]W_N^{nk} \tag{2.60}$$
$$= \sum_{n=0}^{N-1} (x_1[n] + jx_2[n])W_N^{nk} \tag{2.61}$$
$$= \sum_{n=0}^{N-1} (x_1[n]W_N^{nk}) + j\sum_{n=0}^{N-1} (x_2[n]W_N^{nk}) \tag{2.62}$$
$$= X_1[k] + jX_2[k] \tag{2.63}$$

where $0 \leq k \leq N-1$.

The separate components of $X[k]$ are given by:

$$X_1[k] = \frac{1}{2}(X[k] + X^*[N-k]) \tag{2.64}$$
$$X_2[k] = \frac{1}{2j}(X[k]) - X^*[N-k]) \tag{2.65}$$

Now applying the DFT operator directly to the real-valued input sample $g[n]$ with sample size

2N results in:

$$G[k] = DFT\big(g[n]\big) \tag{2.66}$$
$$= \sum_{n-0}^{2N-1} g[n]W_{2N}^{nk} \tag{2.67}$$
$$= \sum_{n=0}^{N-1} g[2n]W_{2N}^{2nk} + \sum_{n=0}^{N-1} g[2n+1]W_{2N}^{(2n+1)k} \tag{2.68}$$
$$= \sum_{n=0}^{N-1} g[2n]W_{2N}^{2nk} + \sum_{n=0}^{N-1} g[2n+1]W_{2N}^{2nk}W_{2N}^{k} \tag{2.69}$$
$$= \sum_{n=0}^{N-1} x_1[n]W_{N}^{nk} + W_{2N}^{k}\sum_{n=0}^{N-1} x_2[n]W_{N}^{nk} \tag{2.70}$$
$$= X_1[k] + W_{2N}^{k}X_2[k] \tag{2.71}$$

where $0 \leq k \leq N-1$.

Substituting the separate components for $X_1[k]$ and $X_2[k]$ from equations 2.64 and 2.65 results in:

$$G[k] = X_1[k] + W_{2N}^{k}X_2[k] \tag{2.72}$$
$$= \frac{1}{2}\big(X[k] + X^*[N-k]\big) + W_{2N}^{k}\frac{1}{2j}\big(X[k] - X^*[N-k]\big) \tag{2.73}$$
$$= \frac{1}{2}\Big(\big(X[k] + X^*[N-k]\big) - jW_{2N}^{k}\big(X[k] - X^*[N-k]\big)\Big) \tag{2.74}$$
$$= \frac{1}{2}\Big(X[k] + X^*[N-k] + \sin\big(\frac{k\pi}{N}\big)\big(X^*[N-k] - X[k]\big) + j\big(\cos\big(\frac{k\pi}{N}\big)\big(X^*[N-k] - X[k]\big)\big)\Big) \tag{2.75}$$

where $0 \leq k \leq N-1$.

The real and imaginary components for $G[k]$ in terms of $X[k]$ can be determined as:

$$\mathrm{Re}(G[k]) = \frac{1}{2}\Big(X[k] + X^*[N-k] + \sin\big(\frac{k\pi}{N}\big)\big(X^*[N-k] - X[k]\big)\Big) \tag{2.76}$$
$$\mathrm{Im}(G[k]) = \frac{1}{2}\Big(\cos\big(\frac{k\pi}{N}\big)\big(X^*[N-k] - X[k]\big)\Big) \tag{2.77}$$

The power of $G[k]$ in terms of $X[k]$ can be determined by:

$$|G[k]|^2 = \Big(\mathrm{Re}(G[k])\Big)^2 + \Big(\mathrm{Im}(G[k])\Big)^2 \tag{2.78}$$
$$= X[k]^2\Big(1 - \frac{1}{2}\sin\big(\frac{k\pi}{N}\big)\Big) + X^*[N-k]^2\Big(1 + \frac{1}{2}\sin\big(\frac{k\pi}{N}\big)\Big) \tag{2.79}$$
$$= X[k]^2A[k] + X^*[N-k]^2B[k] \tag{2.80}$$

where $A[k] = \Big(1 - \frac{1}{2}\sin\big(\frac{k\pi}{N}\big)\Big)$, $B[k] = \Big(1 + \frac{1}{2}\sin\big(\frac{k\pi}{N}\big)\Big)$, and $0 \leq k \leq N-1$.

Now, recall:

$$X[k] = X_1[k] + jX_2[k] \tag{2.81}$$
$$X[N-k] = X_1[N-k] + jX_2[N-k] \tag{2.82}$$
$$X^*[N-k] = X_1[N-k] - jX_2[N-k] \tag{2.83}$$

Substituting equations 2.81, 2.82, and 2.83 into 2.75 results in:

$$G[k] = \frac{1}{2}\Big(X[k] + X^*[N-k] + \sin\big(\frac{k\pi}{N}\big)\big(X^*[N-k] - X[k]\big) + j\big(\cos\big(\frac{k\pi}{N}\big)\big(X^*[N-k] - X[k]\big)\big)\Big) \tag{2.84}$$

$$= \frac{1}{2}\Big(A[k] + \cos(\frac{k\pi}{N})B[k] - \sin(\frac{k\pi}{N})C[k] + j(D[k] - \cos(\frac{k\pi}{N})C[k] - \sin(\frac{k\pi}{N})B[k])\Big) \tag{2.85}$$

where $A[k] = X_1[k] + X_1[N-k]$, $B[k] = X_2[k] + X_2[N-k]$, $C[k] = X_1[k] - X_1[N-k]$, $D[k] = X_2[k] - X_2[N-k]$, and $0 \leq k \leq N-1$.

The real and imaginary components for $G[k]$ in terms of $X_1[k]$ and $X_2[k]$ can be determined as:

$$\mathrm{Re}(G[k]) = \frac{1}{2}\Big(A[k] + \cos(\frac{k\pi}{N})B[k] - \sin(\frac{k\pi}{N})C[k]\Big) \tag{2.86}$$

$$= \frac{1}{2}\Big(X_1[k] + X_1[N-k] + \cos(\frac{k\pi}{N})\big(X_2[k] + X_2[N-k]\big) - \sin(\frac{k\pi}{N})\big(X_1[k] - X_1[N-k]\big)\Big) \tag{2.87}$$

$$\mathrm{Im}(G[k]) = \frac{1}{2}\Big(D[k] - \cos(\frac{k\pi}{N})C[k] - \sin(\frac{k\pi}{N})B[k])\Big) \tag{2.88}$$

$$= \frac{1}{2}\Big(X_2[k] - X_2[N-k] - \cos(\frac{k\pi}{N})\big(X_1[k] - X_1[N-k]\big) - \sin(\frac{k\pi}{N})\big(X_2[k] + X_2[N-k]\big)\Big) \tag{2.89}$$

The power of $G[k]$ in terms of $X_1[k]$ and $X_2[k]$ can be determined as:

$$|G[k]|^2 = \Big(\mathrm{Re}(G[k])\Big)^2 + \Big(\mathrm{Im}(G[k])\Big)^2 \tag{2.90}$$

$$= \frac{1}{2}\Big(X_1[k]^2 + X_1[N-k]^2 + X_2[k]^2 + X_2[N-k]^2 - 2\big(X_1[k]X_1[N-k] - X_2[k]X_2[N-k]\big)\Big) \tag{2.91}$$

where $0 \leq k \leq N-1$.

Now after all those seemingly lengthy and perhaps confusing equations what have we shown? The crux of the matter are the two equations 2.80 and 2.91 which are both expressions for the power of $G[k]$. The first power equation 2.80 expresses the power of $G[k]$ in terms of $X[k]$ and $X^*[N-k]$. Recall that $X[k]$ is the output of the DFT operator applied to the complex-valued sample $x[n]$ but $X[k]$ is also a complex-valued sample. Hence the need for the additional mathematical equations.

The second power equation 2.91 expresses the power of $G[k]$ in terms of $X_1[k]$ and $X_2[k]$. Recall from equation 2.63 that the two terms $X_1[k]$ and $X_1[k]$ are the real and imaginary components of applying the DFT operator to the complex-valued sample $x[n]$ that was constructed using equations 2.59. Hence, the power equation 2.91 is applicable when simultaneously computing the FFT of a real-valued sample $g[n]$ of sample size $2N$ with a single complex-valued sample $x[n]$ of size N since the power equation 2.91 is expressed in terms of the real and imaginary components of applying the DFT operator to the complex-valued sample $x[n]$, namely $X_1[k]$ and $X_2[k]$.

2.5 The Decibel (dB)

Historically the decibel (dB), named after Alexander Graham Bell and the Bell System, is one-tenth of a Bel, which is a dimensionless transmission unit that is closely related to another transmission unit the neper (N_p). The neper however shall not be discussed here.

The decibel is used in many scientific fields and each flavour of decibel is slightly different yet similarly named to another decibel in an alternative scientific field.

The decibel is defined as the base-10 logarithmic unit used to represent and express the ratio between two physical quantities such as intensity levels, power levels or voltage levels[20]. For two physical quantities, Q_1 and Q_2, the base-10 logarithmic ratio between the two quantities in decibels is given by the following equation:

$$dB = 10 \log \left(\frac{Q_1}{Q_2} \right) \tag{2.92}$$

The multiplication factor of 10 produces the "deci" prefix in front of the unit, decibel (dB).

2.5.1 The Decibel in Electronics

There are two kinds of decibels commonly encountered in the field of electronics and they are defined and discussed here. In electronics, most systems can be modelled as a black box, where the exact inner workings or processing contents of the system are not completely known. A single input signal undergoes some kind of internal signal processing (gain or attenuation) within the black-box and the output signal indicates an altered version of the input signal.

2.5.1.1 The Power Gain Decibel

The power gain decibel, G_P, is the base-10 logarithmic ratio between two electrical powers W_1 and W_2, in units W, and is given by the following equation[34]:

$$G_P = 10 \log \left(\frac{W_1}{W_2} \right) \tag{2.93}$$

2.5.1.2 The Voltage Gain Decibel

The voltage gain decibel, A_V, is the base-10 logarithmic ratio between two electrical voltages V_1 and V_2, in units V, and is given by the following equation[34]:

$$A_V = 20 \log \left(\frac{V_1}{V_2} \right) \tag{2.94}$$

Note the difference in the multiplication factor since the power of a signal is related to the voltage squared.

2.5.1.3 Signal-to-Noise Ratio (SNR)

The signal-to-noise ratio (SNR) is a base-10 logarithm ratio between the signal power, P_S, and the noise power, P_N and is given by the following equation:

$$\text{SNR} = 10 \log \left(\frac{P_S}{P_N} \right) \tag{2.95}$$

The signal-to-noise ratio is a dimensionless quantity with decibel units, dB.

2.5.2 The Decibel in Acoustics (L_I, L_W, L_P)

There are three primary kinds of decibels in acoustics and they are outlined and they are discussed here.

2.5.2.1 Acoustic Intensity Level (L_I, SIL)

The acoustic intensity level (L_I), or sound intensity level (SIL), is the base-10 logarithmic ratio between the measured sound intensity I_m, in units W/m^2, and the reference sound intensity level $10^{-12}\, W/m^2$. The acoustic intensity level L_I is given by the following equation[20]:

$$L_I = 10 \log \left(\frac{I_m}{10^{-12}} \right) \tag{2.96}$$

2.5.2.2 Acoustic Power Level (L_W, SWL)

The acoustic power level (L_W), or sound power level (SWL), is the base-10 logarithmic ratio between the measured sound power W_m, in units W, and the reference sound intensity level $10^{-12}\, W$. The acoustic power level L_W is given by the following equation[20]:

$$L_W = 10 \log \left(\frac{W_m}{10^{-12}} \right) \tag{2.97}$$

2.5.2.3 Acoustic Pressure Level (L_P, SPL)

The acoustic pressure level (L_P), or sound pressure level (SPL), is the base-10 logarithmic ratio between the measured sound pressure P_m, in units Pa, and the reference sound pressure level $20\mu\,\text{Pa}$. The acoustic pressure level L_P is given by the following equation[20]:

$$L_P = 20 \log \left(\frac{P_m}{20\mu\, Pa} \right) \tag{2.98}$$

$$= 10 \log \left(\frac{P_m}{20\mu\, Pa} \right)^2 \tag{2.99}$$

2.5.3 The Decibel in Audio Electronics (dBV, dBu, dBW ,dBm, dBFS)

The family decibels that audio and sound engineers use are outlined here.

2.5.3.1 The RMS Voltage Decibel (dBV)

The RMS voltage decibel, dBV, is the base-10 logarithmic ratio between the measured electrical RMS voltage V_m, in units V, across any load resistance and the reference electrical RMS voltage 1V. The RMS voltage decibel dBV is given by the following equation[34]:

$$\text{dBV} = 20 \log \left(\frac{V_m}{1\, V_{RMS}} \right) \tag{2.100}$$

The voltage decibel (dBV) is commonly associated with analogue audio and video signals.

2.5.3.2 The RMS Millivolts Decibel (dBu)

The RMS millivolts decibel or voltage level decibel, dBu, is the base-10 logarithmic ratio between the measured electrical power V_m, in units V, across an open circuit or unloaded resistance and the reference electrical RMS voltage 0.775V. The RMS millivolts decibel dBu is given by the following equation[34]:

$$\text{dBu} = 20 \log \left(\frac{V_m}{0.775\, V_{RMS}} \right) \tag{2.101}$$

Note it is common practice to use the following approximation dBu = dBV + 2.21dB[34].

2.5.3.3 The Watts Decibel (dBW)

The watts decibel or power level decibel, dBW, is the base-10 logarithmic ratio between the measured electrical power W_m, in units W and the reference electrical power 1W. The watts decibel dBW is given by the following equation[35]:

$$\text{dBW} = 10 \log \left(\frac{W_m}{1\, W} \right) \tag{2.102}$$

The watts decibel (dBW) is commonly associated with analogue power audio amplifiers.

2.5.3.4 The Milliwatts Decibel (dBm)

The milliwatts decibel, dBm, is the base-10 logarithmic ratio between the measured electrical power W_m, in units W, across a 600Ω resistance load and the reference electrical power 1mW, or 0.001W. The milliwatts decibel dBm is given by the following equation[35]:

$$\text{dBm} = 10 \log \left(\frac{W_m}{1\, mW} \right) \tag{2.103}$$

It is common for the voltage across the 600Ω load resistance to be $0.775 V_{RMS}$ in order to obtain 1mW reference power. The milliwatts decibel (dBm) is commonly associated with telecommunications and transmission lines.

2.5.3.5 The Full Scale Decibel (dBFS)

The full scale decibel, dBFS, is a relative scale used with analogue-to-digital and digital-to-analogue converters to describe the signal against the full scale range of the data converter.

The full scale range depends on the maximum convertible voltage of the data converters and the bit depth of the data converters[35]. The dynamic range of a data converter is closely related to the signal-to-noise ratio (SNR) discussed in section 2.5.1.3.

An n-bit data converter has a digital dynamic range (DR) that depends on the maximum to minimum signal level that the data converter can handle. Moreover, the dynamic range is a base-10 logarithmic ratio of the bit depth of a data converter and a reference of a single bit.

The dynamic range DR for an n-bit data converter is given by the following equation[35]:

$$DR = 20\log\left(\frac{2^n}{1\ bit}\right) \tag{2.104}$$

$$= 20\log\left(2^n\right) \tag{2.105}$$

$$\approx 6.02 \times n \tag{2.106}$$

The digital quantization step size Q for an n-bit data converter with a maximum full-scale voltage V_{FS} can be determined by the following equation[35]:

$$Q = \frac{V_{FS}}{2^n} \tag{2.107}$$

In all scientific measurements regardless of the field, any measurement taken with an instrument is accompanied by an error or uncertainty. The uncertainty in a measurement is half of the smallest possible measurement.

The quantization error Q_E of an n-bit data converter is determined by the size of the quantization step.

The quantization error, Q_E, is given by the following equation[35];

$$Q_E = \pm\frac{Q}{2} \tag{2.108}$$

The quantization error, Q_E, is random and uniformly distributed about the measured quantization step, Q, within the range of $\pm\frac{Q}{2}$.

The quantization step variance or quantization noise power, σ_E^2, is a measure or feel of the spread of each quantization step and how one step differs from another step. An n-bit data converter with a quantization step size Q and quantization error Q_E, has a quantization noise power, σ_E^2, given by the following equation[27]:

$$\sigma_E^2 = \frac{Q^2}{12} \tag{2.109}$$

The signal-to-quantization noise power ratio (SQNR) indicates the relation of the quantization error to the quantized noise power.

For an n-bit data converter with a maximum full-scale (peak-to-peak) voltage V_{FS} and quan-

tization noise power, σ_E^2, the signal-to-quantization noise power ratio is approximated by[27]:

$$SQNR = 10\log\left(\frac{\left(\frac{\frac{V_{FS}}{2}}{\sqrt{2}}\right)^2}{\sigma_E^2}\right) \tag{2.110}$$

$$= 10\log\left(\frac{\frac{{V_{FS}}^2}{8}}{\frac{Q^2}{12}}\right) \tag{2.111}$$

$$= 10\log\left(\frac{3{V_{FS}}^2}{2Q^2}\right) \tag{2.112}$$

$$= 10\log\left(\frac{3{V_{FS}}^2}{2(\frac{V_{FS}}{2^n})^2}\right) \tag{2.113}$$

$$= 10\log\left(\frac{3\times 2^{2n}}{2}\right) \tag{2.114}$$

$$= 20\log\left(2^n\right) + 10\log\left(\frac{3}{2}\right) \tag{2.115}$$

$$\approx 6.02\times n + 1.76 \tag{2.116}$$

While it won't be discussed in full here, it should be noted that since digital signals are stored as data in memory, there is a slight difference in storing floating-point numbers compared to storing fixed-point numbers in memory[36].

Table 2.4 is a summary of the quantization values for a 12-bit data converter ($n = 12$) and a $3.3V$ full-scale voltage ($V_{FS} = 3.3V$).

Parameter	**Equation**	**Value**
Dynamic Range [dB]	$DR = 20\log(2^n)$	72.25
Signal-to-Quantized Noise Power [dB]	$SQNR = 20\log(2^n) + 1.76$	74.01
Step Size [V]	$Q = \frac{V_{FS}}{2^n}$	805.66×10^{-6}
Error [V]	$Q_E = \pm\frac{Q}{2}$	$\pm 402.83\times 10^{-6}$
Noise Power [W]	$\sigma_E^2 = \frac{Q^2}{12}$	54.09×10^{-9}

Table 2.4: Quantization Values for a 12-bit Converter with $3.3V_{FS}$

Table 2.5 is a summary of the quantization values for a 24-bit data converter ($n = 24$) and a 3.3V full-scale voltage ($V_{FS} = 3.3$V).

Parameter	**Equation**	**Value**
Dynamic Range [dB]	$DR = 20\log(2^n)$	144.49
Signal-to-Quantized Noise Power [dB]	$SQNR = 20\log(2^n) + 1.76$	146.25
Step Size [V]	$Q = \frac{V_{FS}}{2^n}$	178.81×10^{-9}
Error [V]	$Q_E = \pm\frac{Q}{2}$	$\pm 86.4\times 10^{-9}$
Noise Power [W]	$\sigma_E^2 = \frac{Q^2}{12}$	2.66×10^{-15}

Table 2.5: Quantization Values for a 24-bit Converter with $3.3V_{FS}$

For a 24-bit data converter, the dynamic range is approximately 144dB and the SQNR is approximately 146dB. Moreover, the dynamic range (SPL) of the human hearing system is

approximately 140dB since the maximum acoustic pressure is approximated at 200Pa[35]. Alternatively, the dynamic range (SIL and SWL) of the human hearing system is approximately 120dB since the maximum acoustic intensity and power level is approximated at $1W/m^2$ and 1W respectively[35].

2.5.4 Adding Decibels and Combining Voltages

In electronics, signals are a voltage versus time representation with an associated power that is proportional to the square of the voltage. There are two different ways of adding two or more decibel levels together and this difference depends on the nature of the source producing the audio signal.

2.5.4.1 Correlated Decibel Addition

The sum or addition of multiple correlated audio sources represented in decibels, dB_{corr}, is given by[2]:

$$dB_{corr} = 20\log\left(\frac{L_{corr}}{L_{ref}}\right) \tag{2.117}$$
$$= 20\log\left(\frac{L_1 + \cdots + L_N}{L_{ref}}\right) \tag{2.118}$$
$$= 20\log\left(L_1 + \cdots + L_N\right) - 20\log\left(L_{ref}\right) \tag{2.119}$$
$$= 20\log\left(L_{corr}\right) - 20\log\left(L_{ref}\right) \tag{2.120}$$

If each source is of the same level, L, then equation 2.118 simplifies as follows:

$$dB_{corr} = 20\log\left(\frac{L \times N}{L_{ref}}\right) \tag{2.121}$$
$$= 20\log\left(L \times N\right) - 20\log\left(L_{ref}\right) \tag{2.122}$$
$$= 20\log\left(L\right) + 20\log\left(N\right) - 20\log\left(L_{ref}\right) \tag{2.123}$$

2.5.4.2 Uncorrelated Decibel Addition

The sum or addition of multiple uncorrelated audio sources represented in decibels, dB_{uncorr}, is given by[2]:

$$dB_{uncorr} = 20\log\left(\frac{L_{uncorr}}{L_{ref}}\right) \tag{2.124}$$
$$= 20\log\left(\frac{\sqrt{(L_1)^2 + \cdots + (L_N)^2}}{L_{ref}}\right) \tag{2.125}$$
$$= 20\log\left(\sqrt{(L_1)^2 + \cdots + (L_N)^2}\right) - 20\log\left(L_{ref}\right) \tag{2.126}$$
$$= 10\log\left((L_1)^2 + \cdots + (L_N)^2\right) - 20\log\left(L_{ref}\right) \tag{2.127}$$

If each source is of the same level, L, then equation 2.125 simplifies as follows:

$$dB_{uncorr} = 20\log\left(\frac{L\sqrt{N}}{L_{ref}}\right) \tag{2.128}$$

$$= 20\log(L) + 20\log(\sqrt{N}) - 20\log(L_{ref}) \tag{2.129}$$

$$= 20\log(L) + 10\log(N) - 20\log(L_{ref}) \tag{2.130}$$

2.5.4.3 Difference Between Correlated and Uncorrelated Decibel Addition

The difference between correlated decibel addition and uncorrelated decibel addition for multiple sources with the same level is given by:

$$dB_{corr} - dB_{uncorr} = 20\log(N) - 10\log(N) \tag{2.131}$$

$$= 10\log(N) \tag{2.132}$$

Figure 2.13 illustrates the difference between the correlated and uncorrelated decibel addition plotted on a linearly spaced x-axis.

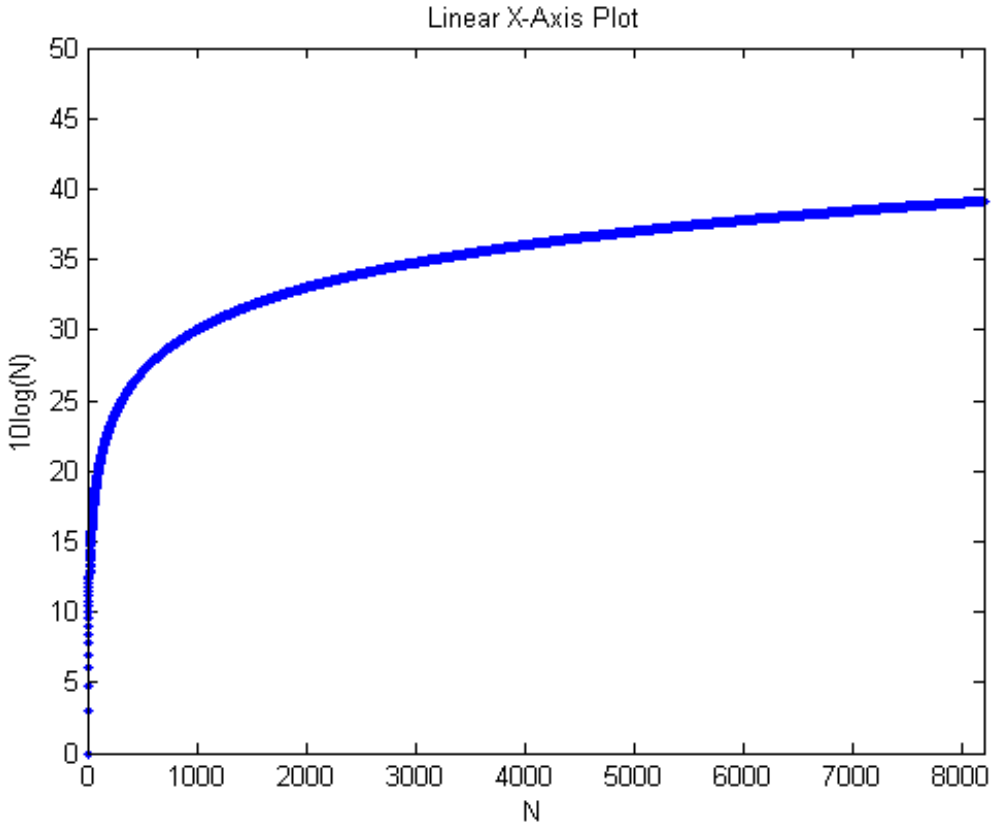


Figure 2.13: Difference between correlated and uncorrelated decibel addition for N sources each with level L calculated and plotted using MATLAB

Figure 2.14 illustrates the difference between the correlated and uncorrelated decibel addition plotted on a logarithmically spaced x-axis.

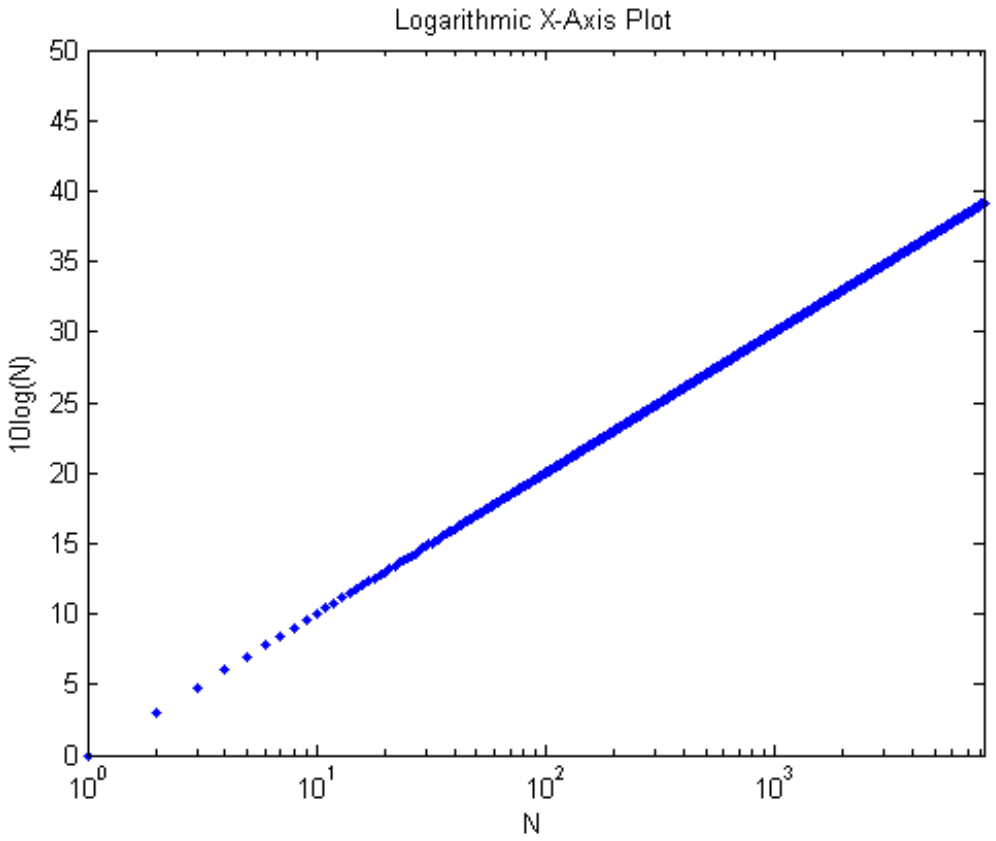


Figure 2.14: Difference between correlated and uncorrelated decibel addition for N sources each with level L calculated and plotted using MATLAB

The difference between correlated and uncorrelated decibel addition of equal level L is dependent on the total number of sources N. Thus, the difference between correlated decibel addition and uncorrelated decibel addition for multiple source, $N = 8192$, with the same noise level, L, is approximately 40dB.

2.5.5 Microphone Sensitivity (S_V) and Open-Circuit voltage (V_O)

The voltage sensitivity of a microphone, S_V, is an open-circuit (no load) voltage measurement across the microphone outputs while the microphone is exposed to an appropriate test signal such as a 1kHz test tone. The open-circuit microphone voltage, V_O, is measured approximately 5ft - 6 ft (1.5m - 1.8m) away from the test source[20].

The voltage sensitivity of the microphone S_V can be determined for a measured open-circuit microphone voltage V_O referenced to 1V for 1Pa acoustic input and is given by the following equation[20]:

$$S_V = 20 \log (V_O) \tag{2.133}$$

Now, the manufacturer of a microphone may only provide the microphone sensitivity, S_V. The open-circuit microphone voltage V_O can be approximated from a given microphone sensitivity

S_V by the following equation:

$$V_O = 10^{(\frac{S_V}{20})} \tag{2.134}$$

As an example, the ADMP401 omnidirectional microphone manufactured by Analog Devices specifies a test condition consisting of a 1 kHz, 94 dB-SPL signal and the microphone produces a typical voltage sensitivity of $S_V = -42dBV$[14].

Hence, the approximated open-circuit voltage for the ADMP401 microphone is given by the following equation:

$$V_O = 10^{\left(\frac{-42dBV}{20}\right)} \tag{2.135}$$

$$\approx 7.94mV \tag{2.136}$$

Alternatively, the voltage sensitivity of a microphone can be expressed in units per μBar. Since 1 Bar = 100000 Pa, then 1μBar = 0.1 Pa. Hence, the approximated open-circuit voltage for the ADMP401 microphone in units per μBar is:

$$V_O \approx 0.794mV \tag{2.137}$$

2.6 The STM32F4 Series of ARM® Cortex-M4 Microcontrollers

STMicroelectronics advertise the STM32F4 series of Cortex-M4 microcontrollers consisting of three different lines: the Access Line, the Foundation Line, and the Advanced Line. Two of the three different STM32F4 lines of microcontrollers were investigated as possible microcontrollers for the designed real-time audio analyser and a brief comparison between the two microcontrollers is given in Table 2.6.

2.6.1 STM32F407 Microcontroller (Foundation Line)

The STM32F407VG microcontroller with Floating Point Unit (FPU) is a 100-pin LQFP operating at a maximum frequency up to 168MHz and features 1MByte Flash Program Memory, 192KBytes System SRAM, 4KBytes of backup SRAM as well as supporting several external memories such as SRAM, PSRAM, Compact Flash, NOR and NAND memories.

2.6.2 STM32F429 Microcontroller (Advanced Line)

The STM32F429ZI microcontroller with Floating Point Unit (FPU) is a 144-pin LQFP operating at a maximum frequency up to 180MHz and features 2MByte Flash Program Memory, 256KBytes System SRAM, 4KBytes of backup SRAM as well as supporting several external memories such as SRAM, PSRAM, SDRAM, Compact Flash, NOR and NAND memories. Moreover, the STM32F429ZI microcontroller also features a LCD-TFT Display Controller (LTDC), a Chrom-ART Graphics Accelerator (DMA2D), and a Serial Audio Interface (SAI).

Table 2.6 provides a comparison between the peripherals featured on both the STM32F407VG and STM32F429ZI microcontrollers.

	STM32F407VG	**STM32F429ZI**
Package	100-pin LQFP	144-pin LQFP
Maximum Frequency (MHz)	168	180
Flash Program Memory (KB)	1024	2048
System SRAM (KB)	192	256
Backup SRAM (KB)	4	4
GPIO Ports	82	114
LCD-TFT Controller	No	Yes
Chrom-ART Graphics Accelerator	No	Yes
Serial Audio Interface	No	Yes
12-bit ADC	3	3
ADC Channels	16	24
12-bit DAC	Yes	Yes
DAC channels	2	2
SPI/I^2S	3/2 (full duplex)	6/2 (full duplex)
I^2C	3	3
USART/UART	4/2	4/4
USB-OTG-FS	Yes	Yes
USB-OTG-HS	Yes	Yes

Table 2.6: Comparison between the STM32F407 and STM32F429 microcontrollers[12][13]

The final designed, developed and implemented hand-held real-time audio analyser features the larger 144-pin STM32F429 microcontroller and a larger printed circuit board was fabricated (DAD-MCU-BRD). A smaller 100-pin STM32F407 microcontroller was investigated at length with the design and fabrication of a smaller printed circuit board (MOM-MCU-BRD). However, the additional features on the STM32F429 microcontroller in particular the increased flash program memory (2048KB), increased system SRAM (256KB), the Serial Audio Interface (SAI) and the LCD-TFT Display Controller (LTDC) provided the necessary resources and communication interfaces for future development of the hand-held real-time audio analyser.

CHAPTER 3

Overview of the Real-Time Audio Analyser

This chapter briefly summarises the hardware and firmware aspects of the implemented hand-held real-time audio analyser. The designed hardware and developed firmware are discussed in detail in Chapter 4 and Chapter 5 respectively.

3.1 Introduction

The designed audio analyser is required to identify the individual audio frequency components of the audio signal and display the output spectrum to the user with a minimal or unnoticeable time delay. A brief overview of the hardware for a real-time audio analyser is given as a block diagram in Figure 3.1.

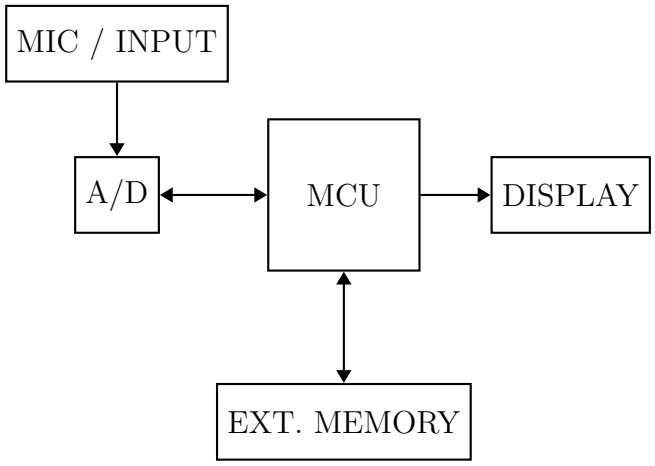


Figure 3.1: Simplified Real-Time Audio Analyser System Overview

An analogue audio signal is the input to the audio analyser and is applied via either an internal

microphone or an external input source connection. The input signal is analysed using appropriate digital signal processing techniques and the resulting audio spectrum is displayed clearly to the user for interpretation. Additional external memory capabilities have been included in the designed hardware to accommodate additional graphics, images, test tones, and other features in future revisions of the developed firmware.

3.1.1 The STM32F429 Real-Time Audio Analyser

The STM32F429 real-time audio analyser hardware design consists three separate printed circuit boards (PCB) that interconnect with one another in a sandwich or stack-like fashion. All designed printed circuit boards including part names and part designators are presented in Appendices A.

3.1.1.1 STM32F429 MCU Board

The STM32F429 microcontroller board, DAD-MCU-BRD for short, is a 131mm x 62mm two layered printed circuit board (PCB). The DAD-MCU-BRD is powered by default, using connector CN8, with a standard 5V USB-mini-B port and is limited to 1.5A . Alternatively, the DAD-MCU-BRD can be externally powered, using connector CN5, by switching jumper J0 (PWR-SEL). The DAD-MCU-BRD supplies all additional printed circuit boards with 1.8V (860mA), 3.3V (1.0A) and 5V (1.1A) via connector CN3. The DAD-MCU-BRD provides complete access to all pins of the STM32F429ZI microcontroller through connectors CN1 (BUS1) and CN2 (BUS2). The TFT-LCD module interfaces with the DAD-MEM-BRD via connector CN4.

3.1.1.2 STM32F429 Memory Board

The STM32F429 external memory board, DAD-MEM-BRD for short, is a 131mm x 62mm two layered printed circuit board (PCB). The DAD-MEM-BRD features two different types of externally accessed memories: 1MByte Static Random Access Memory (SRAM), and 16MByte Synchronous Dynamic Random Access Memory (SDRAM). Both external memories are supplied 3.3V power from the DAD-MCU-BRD and both memory modules interface with the STM32F429 microcontroller using the Flexible Memory Controller (FMC).

3.1.1.3 STM32F429 Audio and Video Board

The STM32F429 audio and video board, DAD-AV-BRD for short, is a 160mm x 62mm two layered printed circuit board (PCB). The DAD-AV-BRD extends the standard real-time audio analyser capabilities with a 24bit, 192kHz sample rate, stereo, audio CODEC. Moreover, the DAD-AV-BRD features a Video Graphics Array (VGA), three internal MEMS microphones, balanced line input/output, and a headphone output connection.

3.2 Specifications and Capabilities of the Real-Time Audio Analyser

The implemented real-time audio analyser is a low-cost alternative to currently available audio analysis solutions with the following specifications:

- Six digital windowing functions:
  - Rectangular Window
  - Hanning Window
  - Hamming Window
  - Welch Window
  - Bartlett Window
  - Blackman Window
- Four audio weighting functions:
  - A-Weighting Function
  - B-Weighting Function
  - C-Weighting Function
  - Z-Weighting Function
- Three external memories:
  - SD-Card
  - 1 MByte Static Random Access Memory (SRAM)
  - 16 MBytes Synchronous Dynamic Random Access Memory (SDRAM)
- Two display outputs:
  - 3.2" TFT-LCD with resistive touch screen
  - Standard VGA controller
- Two input sources:
  - Internal Single-Ended Analogue MEMS microphone
  - External Balanced XLR input
- Two audio converters:
  - Internal 12-bit Successive Approximation ADC
  - External 24-bit, 192kHz Sampling-Rate Stereo Audio CODEC

CHAPTER 4

The Real-Time Audio Analyser Hardware

This chapter discusses the individual hardware components found on the real-time audio analyser board (DAD-MCU-BRD), and the additional extension boards (DAD-AV-BRD and DAD-MEM-BRD).

The designed hardware features the STM32F429 microcontroller with several internal (on-chip) peripheral modules such as: Reset and Clock Controller (RCC), General Purpose Input/Output (GPIO) peripheral, Nested Vectored Interrupt Controller (NVIC), and Direct Memory Access (DMA) controller. The presented hardware design provides a platform for future work with the additional on-board hardware components featured on the extension boards such as the 24-bit stereo CODEC, the Video Graphics Array (VGA) controller, and three external memory options.

4.1 Power Supply

By default the real-time audio analyser is powered by a standard 5V USB-mini-B port via connector `CN8 (VUSB)` and is limited to 1.5A. Alternatively, the real-time audio analyser can be powered by an external power supply via connector `CN5 (VBAT)`, provided VBAT mode is selected by jumper `J1 (PWR-SEL)`. The 5V supplied by either VUSB or VBAT is regulated via the NCP1117LP[37] low-dropout voltage regulator and provides a regulated 5V supply, limited to 1.1A. The regulated 5V is further stepped-down via the AP1117[38] and provides a regulated 3.3V output, limited to 1.0A. The regulated 3.3V can be further stepped-down via the TLV70018[39] to provide a regulated 1.8V, limited to 860mA.

4.2 Crystal Oscillator

The real-time audio analyser requires an 8MHz crystal oscillator, X1, to clock the STM32F429 microcontroller. The crystal oscillator is connected to the STM32F429 microcontroller via pins

PH0 and PH1. The load capacitors connected to the crystal oscillator are placed close to the crystal oscillator in order to minimize the output distortion and start-up stabilization time[12]. Additional guidelines for the external crystal oscillator design is given as an application note for the STM32 family of microcontrollers[40].

4.3 STM32F4 Microcontroller

The STM32F4 series of 32-bit Cortex-M4 microcontrollers with Floating-Point Unit (FPU) is equipped with several common on-chip peripherals such as: Reset and Clock Controller (RCC), Direct Memory Access (DMA) controllers, and 12-bit Analogue-to-Digital Converter (ADC). The STM32F4 series of microcontrollers are also equipped with several less common on-chip peripherals such as: Serial Audio Interface (SAI), LCD-TFT Display Controller (LTDC), and Chrom-Art Accelerator (DMA2D) controller. The hardware related aspects for the STM32F4 series of microcontrollers including the on-chip peripherals used in the implementation of the real-time audio analyser are discussed in subsections 4.3.1, 4.3.2, 4.3.3, 4.3.4, 4.3.5, 4.3.6, 4.3.7, 4.3.8, and 4.3.9.

4.3.1 Memory and System Bus Architecture

There are some minor hardware differences between the STM32F407 microcontroller and the STM32F429 microcontroller. The majority of the internal peripherals featured on the STM32F407 and STM32F429 microcontrollers are identical, such as: General Purpose Input/Output (GPIO) controller, Nested Vectored Interrupt Controller (NVIC), and various Timers. However, there are some subtle differences in the memory organization and system bus architecture, in addition to some minor differences of less common peripherals, such as: Direct Memory Access (DMA) controllers, external memory controllers (FMSC and FMC), and LCD-TFT Display Controller (LTDC).

4.3.1.1 Memory Organization

The memory for the STM32F4 series of microcontrollers is organized as a linear 4GB address space which is divided into 8 main blocks, each consisting of 512MB, and the memory bytes are coded in little endian format[3]. This implies that the lowest byte in a word is the least significant byte of the word and the highest byte is the word is the most significant byte of the word.

The complete memory address mapping for the peripherals on the STM32F4 microcontrollers is given in Table 4.1. The full peripheral and bus name for each short form can be found in the list of acronyms at the end of this thesis.

Memory Address	**Peripheral**	**Bus**
0xA000 000 - 0xA000 0FFF	FSMC (STM32F407) / FMC (STM32F429)	AHB3
0x5006 0800 - 0x5006 0BFF	RNG	AHB2
0x5006 0400 - 0x5006 07FF	HASH	AHB2
0x5006 0000 - 0x5006 03FF	CRYP	AHB2
0x5005 0000 - 0x5005 03FF	DCMI	AHB2
0x5000 0000 - 0x5003 FFFF	USB OTG FS	AHB2
0x4004 0000 - 0x4007 FFFF	USB OTG HS	AHB1
0x4002 B000 - 0x4002 BBFF	DMA2D (STM32F429)	AHB1
0x4002 8000 - 0x4002 93FF	ETHERNET MAC	AHB1
0x4002 6400 - 0x4002 67FF	DMA2	AHB1
0x4002 6000 - 0x4002 63FF	DMA1	AHB1
0x4002 4000 - 0x4002 4FFF	BKPSRAM	AHB1
0x4002 3C00 - 0x4002 3FFF	Flash Interface	AHB1
0x4002 3800 - 0x4002 3BFF	RCC	AHB1
0x4002 3000 - 0x4002 33FF	CRC	AHB1
0x4002 2800 - 0x4002 2BFF	GPIOK	AHB1
0x4002 2400 - 0x4002 27FF	GPIOJ	AHB1
0x4002 2000 - 0x4002 23FF	GPIOI	AHB1
0x4002 1C00 - 0x4002 1FFF	GPIOH	AHB1
0x4002 1800 - 0x4002 1BFF	GPIOG	AHB1
0x4002 1400 - 0x4002 17FF	GPIOF	AHB1
0x4002 1000 - 0x4002 13FF	GPIOE	AHB1
0x4002 0C00 - 0x4002 0FFF	GPIOD	AHB1
0x4002 0800 - 0x4002 0BFF	GPIOC	AHB1
0x4002 0400 - 0x4002 07FF	GPIOB	AHB1
0x4002 0000 - 0x4002 03FF	GPIOA	AHB1
0x4001 6800 - 0x4001 6BFF	LDC-TFT	APB2
0x4001 5800 - 0x4001 5BFF	SAI1	APB2
0x4001 5400 - 0x4001 57FF	SPI6	APB2
0x4001 5000 - 0x4001 53FF	SPI5	APB2
0x4001 4800 - 0x4001 4BFF	TIM11	APB2
0x4001 4400 - 0x4001 47FF	TIM10	APB2
0x4001 4000 - 0x4001 43FF	TIM9	APB2
0x4001 3C00 - 0x4001 3FFF	EXTI	APB2
0x4001 3800 - 0x4001 3BFF	SYSCFG	APB2
0x4001 3400 - 0x4001 37FF	SPI4	APB2
0x4001 3000 - 0x4001 33FF	SPI1	APB2
0x4001 2C00 - 0x4001 2FFF	SDIO	APB2
0x4001 2000 - 0x4001 12FF	ADC1/2/3	APB2
0x4001 1400 - 0x4001 17FF	USART6	APB2
0x4001 1000 - 0x4001 13FF	USART1	APB2
0x4001 0400 - 0x4001 07FF	TIM8	APB2
0x4001 0000 - 0x4001 03FF	TIM1	APB2
0x4000 7C00 - 0x4000 7FFF	UART8	APB1
0x4000 7800 - 0x4000 7BFF	UART7	APB1
0x4000 7400 - 0x4000 77FF	DAC	APB1
0x4000 7000 - 0x4000 73FF	PWR	APB1
0x4000 6800 - 0x4000 6BFF	CAN2	APB1
0x4000 6400 - 0x4000 67FF	CAN1	APB1
0x4000 5C00 - 0x4000 5FFF	I2C3	APB1
0x4000 5800 - 0x4000 5BFF	I2C2	APB1
0x4000 5400 - 0x4000 57FF	I2C1	APB1
0x4000 5000 - 0x4000 53FF	UART5	APB1
0x4000 4C00 - 0x4000 4FFF	UART4	APB1
0x4000 4800 - 0x4000 4BFF	USART3	APB1
0x4000 4400 - 0x4000 47FF	USART2	APB1
0x4000 4000 - 0x4000 43FF	I2S3ext	APB1
0x4000 3C00 - 0x4000 3FFF	SPI3 / I2S3	APB1
0x4000 3800 - 0x4000 3BFF	SPI2 / I2S2	APB1
0x4000 3400 - 0x4000 37FF	I2S2ext	APB1
0x4000 3000 - 0x4000 33FF	IWDG	APB1
0x4000 2C00 - 0x4000 2FFF	WWDG	APB1
0x4000 2800 - 0x4000 2BFF	RTC & BKP	APB1
0x4000 2000 - 0x4000 23FF	TIM14	APB1
0x4000 1C00 - 0x4000 1FFF	TIM13	APB1
0x4000 1800 - 0x4000 1BFF	TIM12	APB1
0x4000 1400 - 0x4000 17FF	TIM7	APB1
0x4000 1000 - 0x4000 13FF	TIM6	APB1
0x4000 0C00 - 0x4000 0FFF	TIM5	APB1
0x4000 0800 - 0x4000 0BFF	TIM4	APB1
0x4000 0400 - 0x4000 07FF	TIM3	APB1
0x4000 0000 - 0x4000 03FF	TIM2	APB1

Table 4.1: Peripheral Address Organization

4.3.1.2 System Bus Architecture

The overall system architecture for the STM32F4 series of microcontrollers consists of the 32-bit multilayer Advanced High-Speed Bus Matrix that interconnects and manages access arbitration between masters and slaves. The system bus organization for the STM32F429 microcontroller is shown in Figure 4.1 and consists of the 32-bit multilayer AHB bus matrix. The AHB bus matrix interconnects 10 masters and 8 slaves:

- Ten Masters:
  - Cortex-M4 with FPU core I-Bus
  - Cortex-M4 with FPU core D-Bus
  - Cortex-M4 with FPU core S-Bus
  - DMA1 Memory Bus
  - DMA2 Memory Bus
  - DMA2 Peripheral Bus
  - Ethernet DMA Bus
  - USB OTG HS DMA Bus
  - LCD Controller DMA Bus
  - DMA2D(Chrom-Art Accelerator) Memory Bus
- Eight Slaves:
  - Internal Flash Memory ICode Bus
  - Internal Flash Memory DCode Bus
  - Internal Main SRAM1 (112KB)
  - Internal Auxiliary SRAM2 (16KB)
  - Internal Auxiliary SRAM3 (64KB)
  - AHB1 Peripherals including AHB to APB bridges and APB peripherals
  - AHB2 Peripherals
  - Flexible Memory Controller (FMC)

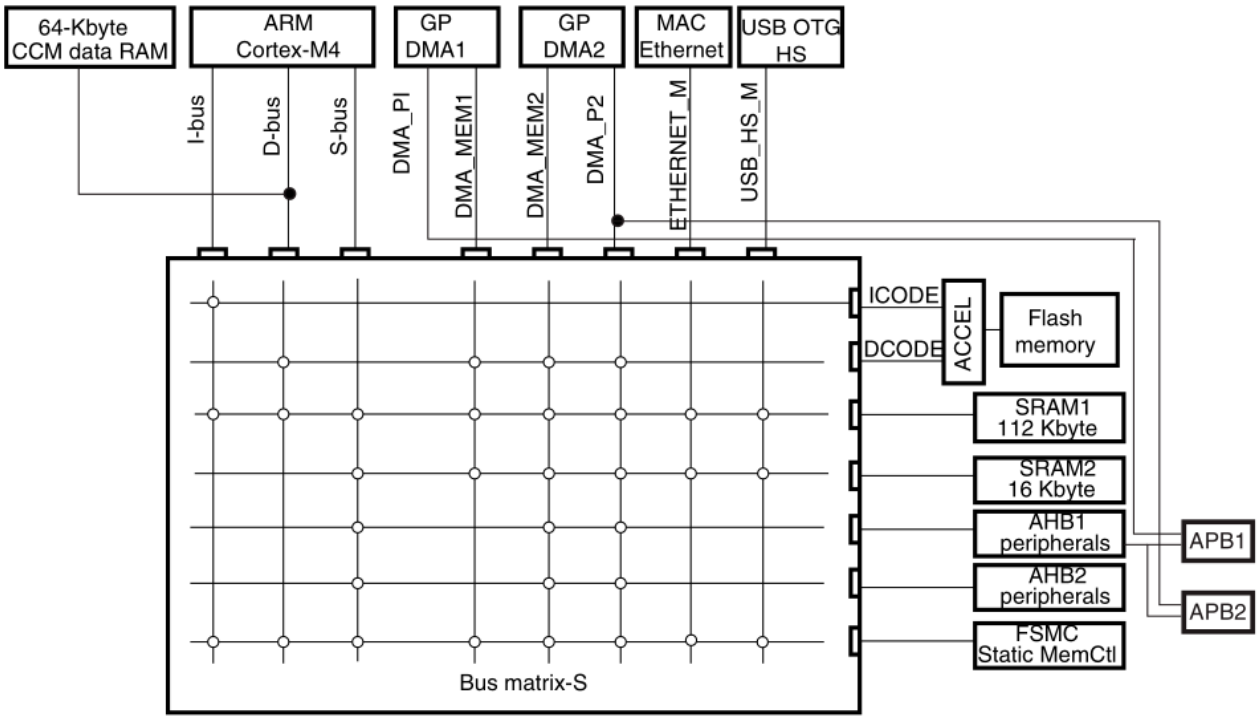


Figure 4.1: System Bus Architecture on the STM32F429 microcontroller[3]

4.3.2 Reset and Clock Controller (RCC)

The use of clocks in digital electronics provides a definite and finite time in which data is transferred on a signal line or bus. The Reset and Clock Controller (RCC) configures the external clock source, the phased-lock loop clocks and all the on-chip peripheral clocks featured by the STM32F429 microcontroller. Figure 4.2 is a block diagram with the external clock sources on the right hand side and the internal peripheral clock sources on the left hand side.

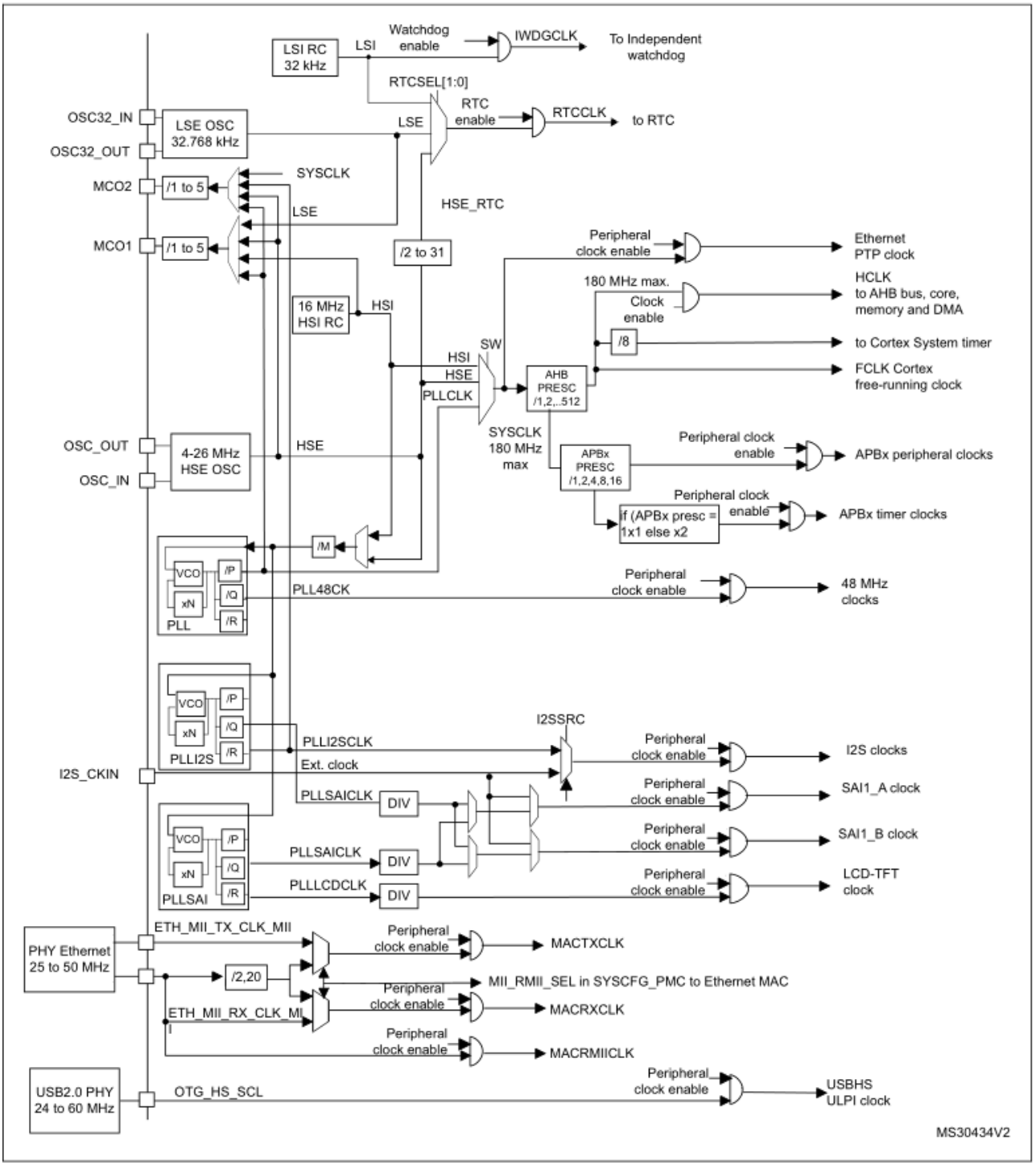


Figure 4.2: Clock Tree on STM32F429 microcontroller[3]

4.3.2.1 External Clock Source

By default, the STM32F429 microcontroller is clocked via the 8MHz external crystal oscillator, X1 on the DAD-MCU-BRD. The external crystal oscillator in Figure 4.2 connects to the STM32F429 microcontroller via pins OSC_{IN} and OSC_{OUT}.

4.3.2.2 Phased-Lock Loops (PLL)

The STM32F429 microcontroller provides three phased-lock loops (PLL) that clock the main system and all peripherals:

- Main PLL clock (PLL) – $SYSCLK$, PLL_{48CK}
- I2S PLL clock ($PLLI2S$) – PLL_{I2SCLK}, PLL_{SAICLK}
- SAI PLL clock ($PLLSAI$) – PLL_{SAICLK}, PLL_{LCDCLK}

The RCC feeds the 8MHz external clock signal through a dividing factor **M** before supplying the Voltage Controlled Oscillator (VCO) within each of the three phased-lock loops with a 2MHz clock source since M = 4.

The three phased-lock loops each have their own voltage controlled oscillators and offer configurable multiplication factor **N** and division factors **P**, **Q**, and **R**. The main PLL only offer the P and Q division factors while the PLLI2S and PLLSAI only feature the Q and R division factors.

The voltage controlled oscillator output frequency is determined by[3]:

$$VCO = \frac{HSE}{M} \times N \tag{4.1}$$

If HSE = 8MHz, M = 4, N = 192, then the voltage controlled oscillator output clock will oscillate at a frequency of 384MHz. Hence, $VCO = 384\text{MHz}$.

The phased-lock loop output frequency is determined by[3]:

$$PLL = \frac{VCO}{P} \tag{4.2}$$

If P = 2, then the phased-lock loop output clock will oscillate at 192MHz. Hence, $PLL = 192\text{MHz}$.

The PLL_{48CK} output frequency is determined by[3]:

$$PLL_{48CK} = \frac{VCO}{Q} \tag{4.3}$$

If Q = 8, then the phased-lock loop output clock will oscillate at 48MHz. Hence, $PLL_{48CK} = 48\text{MHz}$.

It should be noted that the developed firmware code over-clocks the main PLL oscillator and system clock for the STM32F429 microcontroller to a frequency of 192MHz for two reasons. Firstly, the real-time audio analyser processes a large number of samples. The default number of sample obtained from the audio converters is 8192 samples. Moreover, the microcontroller needs to sort through the processed samples before displaying the correctly formatted audio spectrum to the user for interpretation. Hence, the microcontroller needs to be clocked at a high frequency in order to achieve a real-time audio analysis of the input signal.

Secondly, the USB peripheral clock featured on the STM32F429 microcontroller is limited to a 48MHz frequency. However, the maximum system clock speed for the STM32F429 microcontroller is 180MHz which is insufficient to clock the USB peripheral clock.

Alternatively, the system clock speed could be decreased to 168MHz however for the first reason above the STM32F429 microcontroller is over-clocked by a mere 12MHz.

4.3.2.3 System Clock (SYSCLK)

The System Clock (SYSCLK) is the main clock source for the Cortex®-M4 core and all the on-chip peripheral clocks are derived from the system clock except for: USB OTG FS and USB OTG HS clocks, I^2S clock, SAI clock, LTDC clock and Ethernet MAC clocks[3]. The system clock can be driven by one of three different clock sources:

- High-Speed Internal (HSI) Oscillator (16MHz RC oscillator)
- High-Speed External (HSE) Oscillator (4-26MHz external clock)
- Main Phased-Lock Loop (PLL) Oscillator (up to 180MHz)

The RTAA initially uses the 8MHz HSE clock as the system clock and changes to use the main PLL as the system clock once the main PLL has been configured and is stable. Hence, SYSCLK = 192MHz.

The Advanced High-Speed Bus (AHB), Advanced Peripheral Buses 1 and 2 (APB1 and APB2) and CPU Clock (HCLK) are all clocked by the system clock (SYSCLK) and are all over-clocked.

The clock frequencies on the real-time audio analyser are:

Clock	**Frequency**
HSE	8MHz
VCO	384MHz
PLL	192MHz
$SYSCLK$	192MHz
$HCLK$	192MHz
AHB	192MHz
$APB1$	96MHz
$APB2$	48MHz

Table 4.2: Clock Frequencies used by the Real-Time Audio Analyser

Moreover, all ARM Cortex-M4 microcontrollers feature an on-core system timer, the Cortex System Timer (SysTick or STK). The RCC feeds the AHB clock frequency divided by 8. Hence, SysTick = 24MHz.

All the peripherals featured by the STM32F429 microcontroller have their own peripheral clocks. By default all peripheral clocks are turned-off in order to save power consumption.

Hence, the appropriate peripheral enable clock register must be enabled prior to configuring and enabling the relevant peripheral.

4.3.3 General Purpose Input/Output (GPIO) Controller

The General Purpose Input/Output (GPIO) Controller configures each of the I/O ports consisting of six sets of 16 pins used either as inputs or outputs on the STM32F429 microcontroller. Each I/O pins can be configured to operate in one of eight modes, namely[3]:

- Input Floating
- Input Pull-Up
- Input Pull-Down
- Analog
- Output Open-Drain with configurable Pull-Up or Pull-Down
- Output Push-Pull with configurable Pull-Up or Pull-Down
- Alternative Function with configurable Pull-Up or Pull-Down
- Alternative Function with configurable Pull-Up or Pull-Down

Figure 4.3 illustrates the basic structure for an I/O port bit.

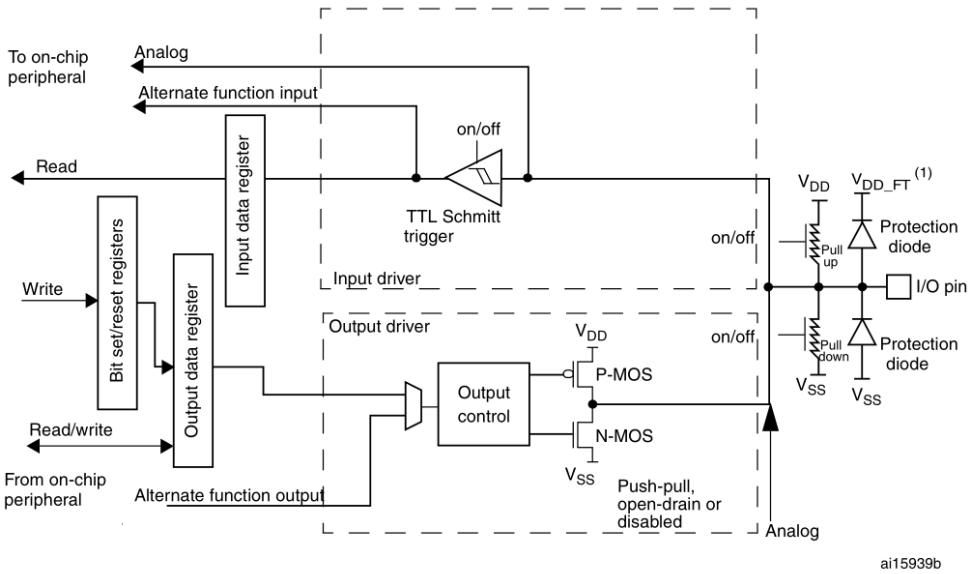


Figure 4.3: GPIO port bit basic structure[3]

4.3.4 Nested Vectored Interrupt Controller (NVIC)

The Nested Vectored Interrupt Controller (NVIC) is common to all ARM Cortex-M4 microcontrollers and is a peripheral of the Cortex-M4 core[41]. The NVIC supports up to 91 maskable interrupt channels on the STM32F429 microcontroller - excluding the 16 interrupt channels for the FPU. The NVIC is located close to the processor core enabling low latency interrupt processing as well as efficient processing of late arriving interrupts[3].

The real-time audio analyser utilizes the NVIC to oversee the flow of data and is discussed in section 5.4.3.5.

4.3.5 Flexible Memory Controller (FMC)

The Flexible Memory Controller (FMC) features three different memory controllers:

- NOR/PSRAM Memory Controller
- NAND/PC Card Memory Controller
- SDRAM Memory Controller

The FMC comprises of five different sections:

- Advanced High-Speed Bus (AHB3) interface and configuration registers
- NOR Flash/PSRAM/SRAM controller
- NAND Flash/PC Card controller
- SDRAM controller
- external device interface

The complete block diagram for the FMC is given in Figure 4.4.

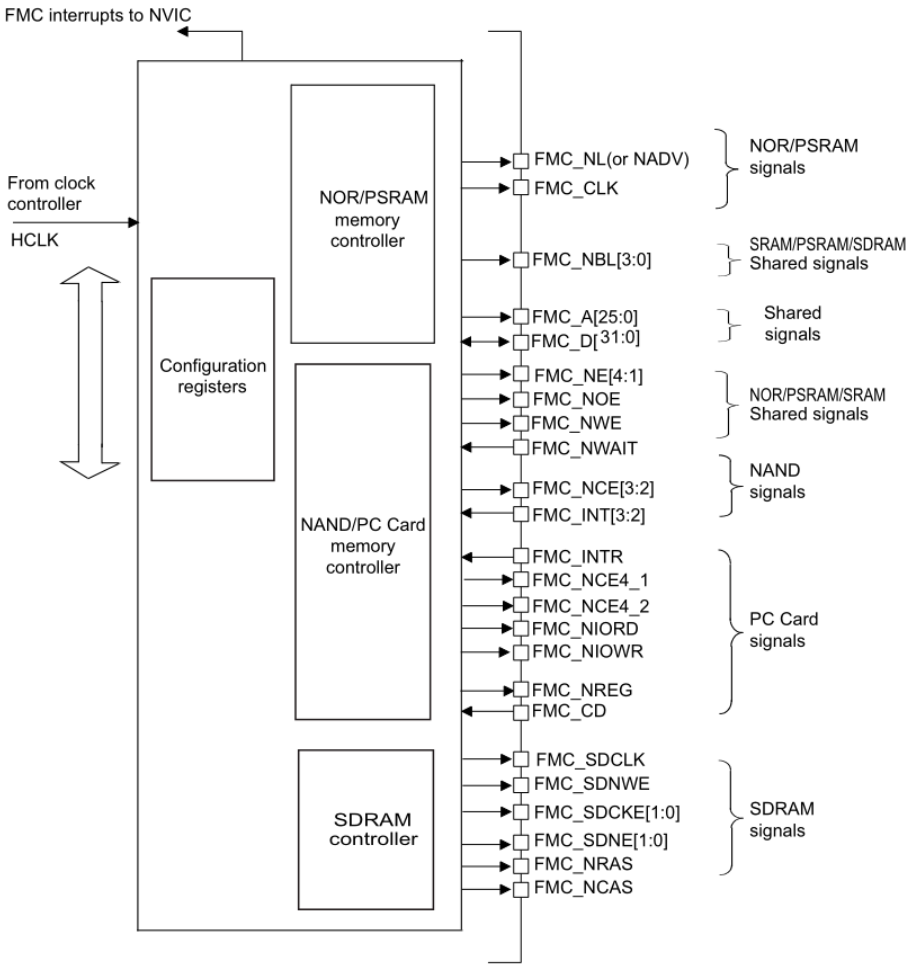


Figure 4.4: FMC Block Diagram[3]

The external memories are accessed by the internal CPU and master peripheral via the AHB3 slave interface. The reference clock for the FMC is the AHB clock (HCLK) which is 192MHz.

The external memory locations comprise of 6 banks of 256MBytes each:

- Bank 1 addresses 4 NOR/PSRAM sub-banks each with a dedicated Chip Select pin
  - Bank 1 - NOR/PSRAM 1
  - Bank 1 - NOR/PSRAM 2
  - Bank 1 - NOR/PSRAM 3
  - Bank 1 - NOR/PSRAM 4
- Bank 2 addresses NAND Flash memory
- Bank 3 addresses NAND Flash memory
- Bank 4 addresses PC Card memory
- Bank 5 addresses SDRAM memory
- Bank 6 addresses SDRAM memory

The complete FMC memory bank locations are given in Figure 4.5.

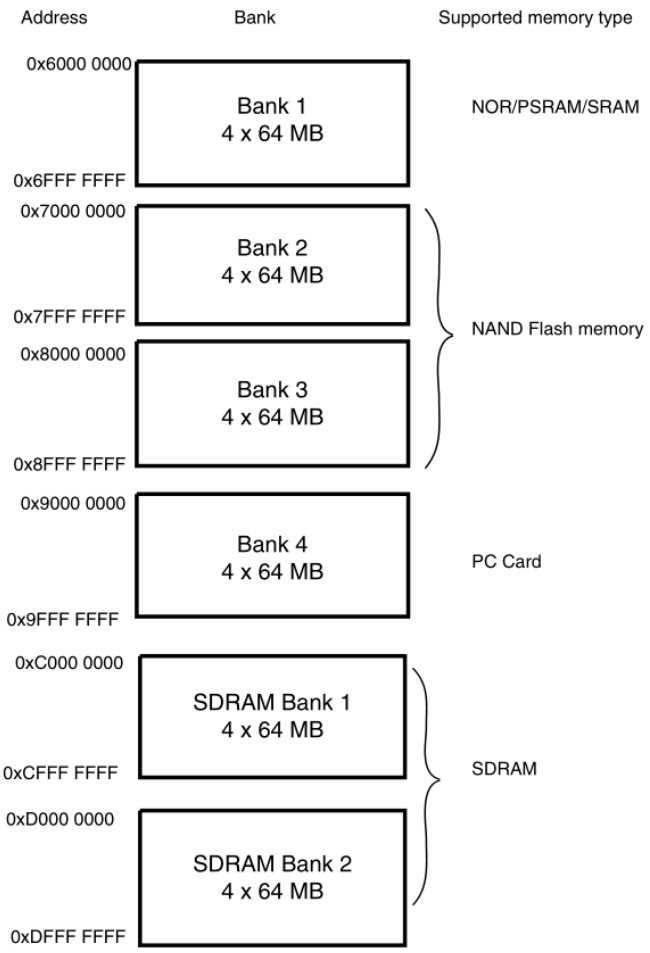


Figure 4.5: FMC Memory Banks[3]

The real-time audio analyser utilizes Bank 1 of the FMC as a data interface for the LCD controller and is discussed in section 5.4.3.1. The additional memory extension board DAD-MEM-BRD utilizes Bank 1 and Bank 5 of the FMC as a data interface for the external memories.

4.3.6 Timers (TIM)

The STM32F4 series of microcontrollers features three different types of timers:

- Basic Timers - TIM6, TIM7
- General Purpose Timers - TIM2, TIM3, TIM4, TIM5, TIM9, TIM10, TIM11, TIM12, TIM13, TIM14
- Advanced Control Timers - TIM1, TIM8

The features for the three different types of timers are compared in Figure 4.6.

Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Complementary output	Max interface clock (MHz)	Max timer clock (MHz) (1)
Advanced -control	TIM1, TIM8	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	Yes	90	180
General purpose	TIM2, TIM5	32-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	45	90/180
	TIM3, TIM4	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	45	90/180
	TIM9	16-bit	Up	Any integer between 1 and 65536	No	2	No	90	180
	TIM10 , TIM11	16-bit	Up	Any integer between 1 and 65536	No	1	No	90	180
	TIM12	16-bit	Up	Any integer between 1 and 65536	No	2	No	45	90/180
	TIM13 , TIM14	16-bit	Up	Any integer between 1 and 65536	No	1	No	45	90/180
Basic	TIM6, TIM7	16-bit	Up	Any integer between 1 and 65536	Yes	0	No	45	90/180

1. The maximum timer clock is either 90 or 180 MHz depending on TIMPRE bit configuration in the RCC_DCKCFGR register.

Figure 4.6: Timers Features Comparison[3]

The real-time audio analyser utilizes the advanced control timer TIM8 to provide the correct sampling rate for the 12-bit on-chip Analogue-to-Digital Converter (ADC). The configuration of TIM8 is discussed in section 5.4.3.6.

4.3.7 Analogue-to-Digital Converter (ADC)

The STM32F429 microcontroller features a successive approximation 12-bit Analogue-to-Digital Converter (ADC) capable of sampling up to 2MHz. The single block diagram indicating the various internal workings and components of the ADC is given in Figure 4.7. The internal ADC featured on the STM32F429 microcontroller provides up to 19 multiplexed input channels consisting of 16 external sources, two internal input sources (Temperature Sensor T_A and V_{REFIN}), and the V_{BAT} input channel. The 16 external input sources are further divided into two separate groups: regular group channels (16 maximum) and injected group channels (4 maximum). The

ADC can operate in four modes: single, continuous, scan, or discontinuous mode. The ADC on the STM32F429 microcontroller can be triggered to begin a single conversion by several on-chip internal timers or by externally triggered pins.

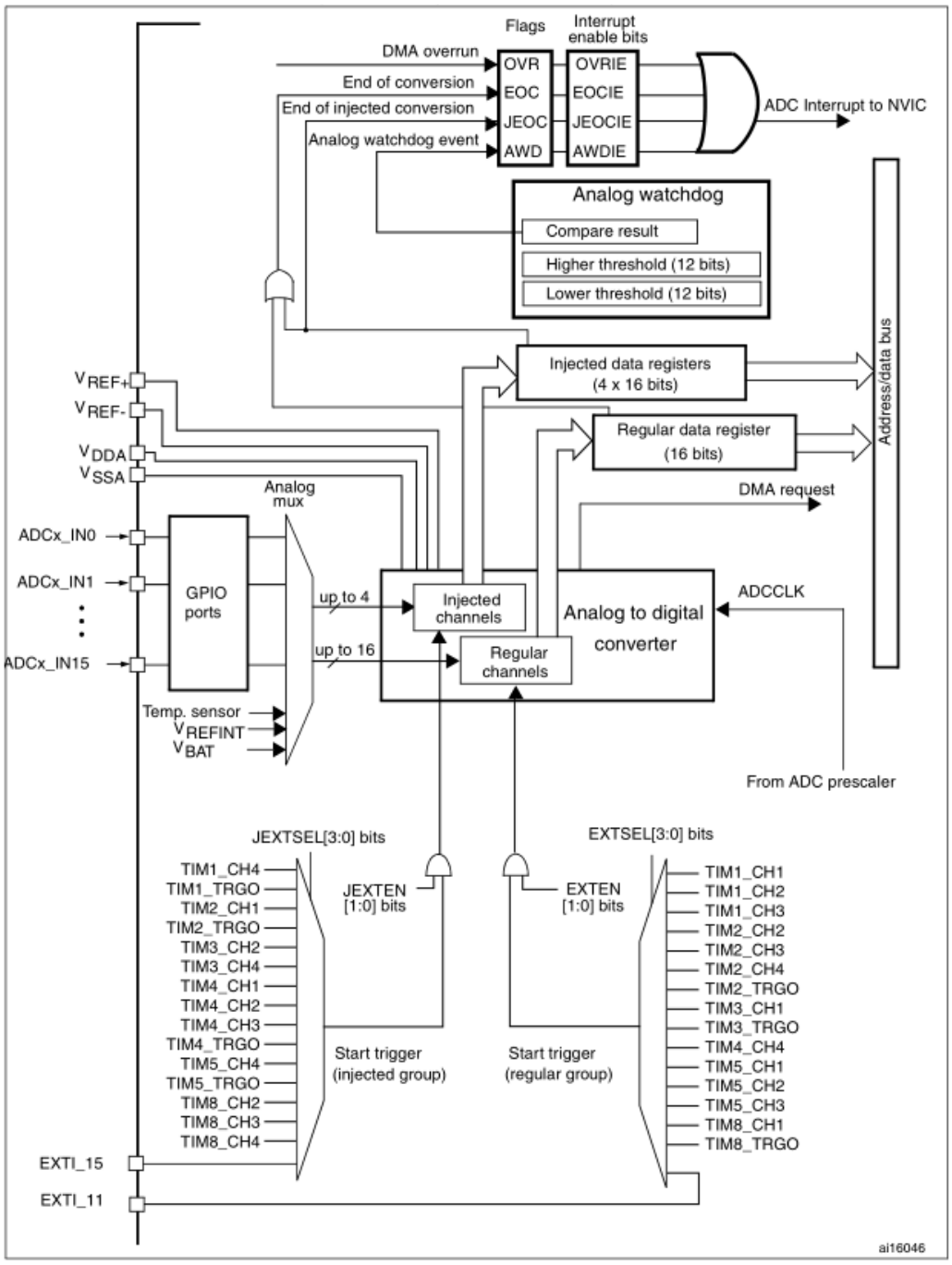


Figure 4.7: Single ADC Block Diagram on the STM32F429 microcontroller[3]

The real-time audio analyser utilizes the internal on-chip 12-bit analogue-to-digital converter with a 3.3V reference voltage to operate in single conversion mode and is discussed in section 5.4.3.3.

4.3.8 Direct Memory Access (DMA)

The Direct Memory Access (DMA) controller provides high-speed data transfer between either peripherals and memory or between memory and memory. The DMA controller transfers the data independently of the current CPU operations and thereby frees CPU resources for other tasks. The DMA controller can transfer data directly as a master on the AHB between a peripheral and memory or between memory and memory. Figure 4.8 is a block diagram indicating the internal components for the DMA controller.

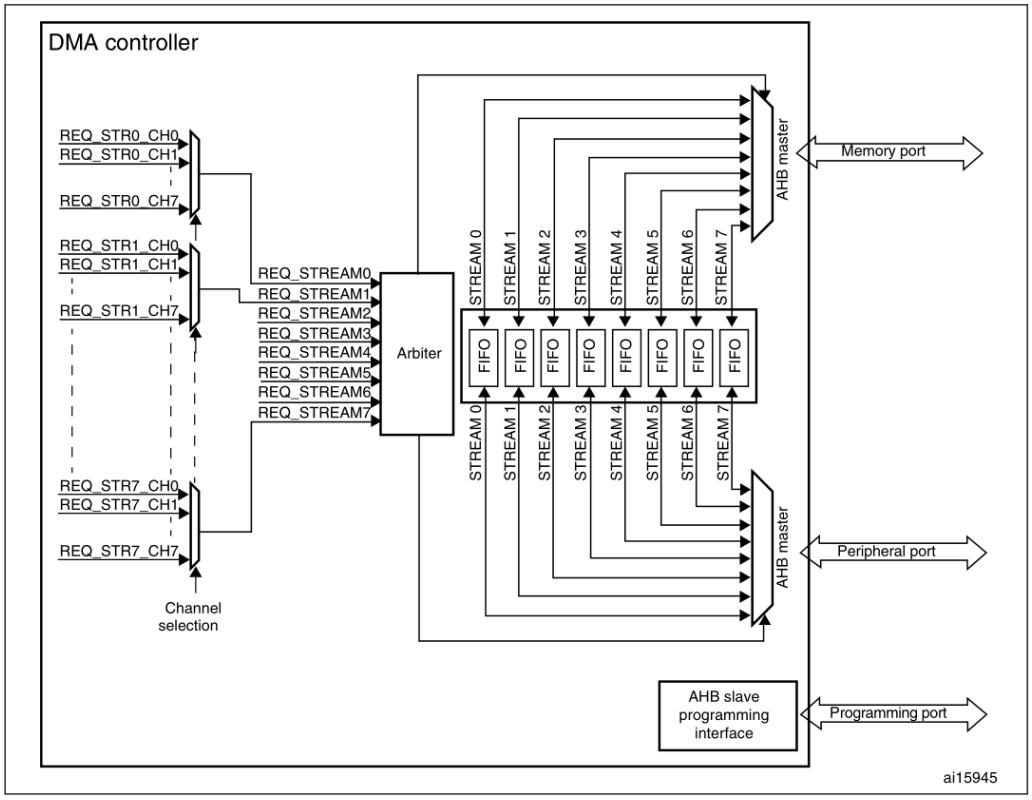


Figure 4.8: DMA Block Diagram on the STM32F429 microcontroller[3]

The real-time audio analyser utilizes the DMA to oversee the transfer of data from the ADC to a memory address and is discussed in section 5.4.3.4.

4.3.9 LCD-TFT Display Controller (LTDC)

The LCD-TFT Display Controller (LTDC) allows the STM32F429 microcontroller to interface with a variety of Thin-Film Transistor Liquid-Crystal Display (LCD-TFT) panels. The block diagram indicating the various internal workings and components of the LTDC is given in Figure 4.9. The LTDC can interface 24-bit parallel data, namely Red, Green, Blue (RGB) formatted as RGB888. The LTDC generates the necessary timing signals for the horizontal and vertical synchronisation and additionally provides outputs for a Pixel clock (`LCD_CLK`) and a Data Enable pin (`LCD_DE`).

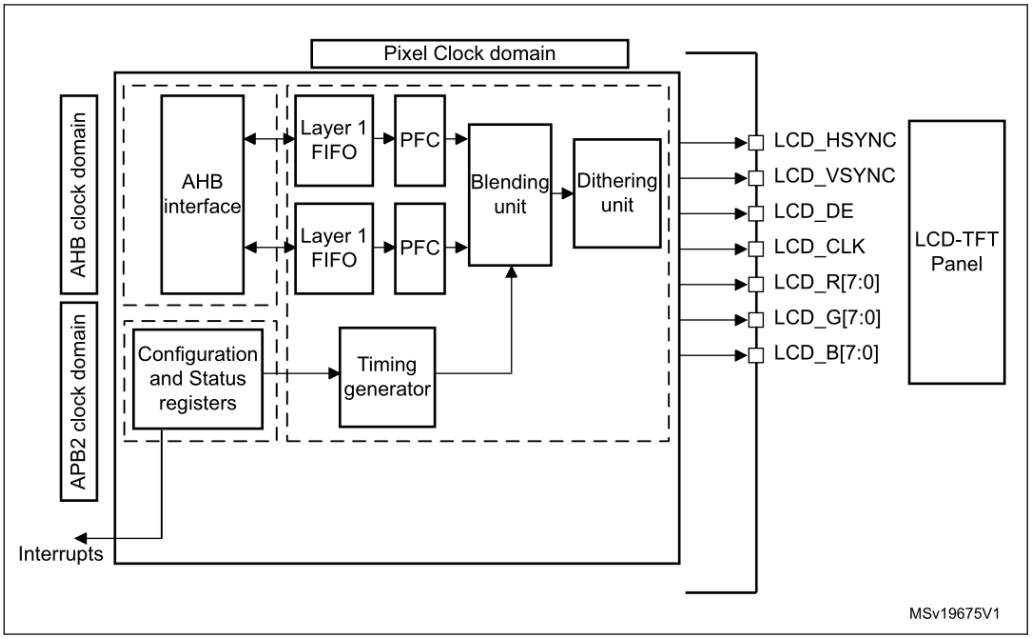


Figure 4.9: LTDC Block Diagram on the STM32F429 microcontroller[3]

The additional audio/video extension board DAD-AV-BRD utilizes the LTDC and is discussed in chapter 7.

4.4 Debugging LEDs, Switches and Jumpers

There are four debugging LEDs, two tactile switches and three selection jumpers featured on the DAD-MCU-BRD and they are shown in Figure A.1.

4.4.1 Debugging LEDs

Each LED is connected to a GPIO pin in series with an appropriate valued resistor to limit the current flow through the LED. The LEDs can be turned on or off (or toggled) at points of interest in the execution of the firmware code by the STM32F429 microcontroller. The debugging LEDs are the next best debugging tool to an in-circuit debugger such as the ST-LINK/V2 in-circuit programmer/debugger from STMicroelectronics. Table 4.3 shows the connections between the debugging LEDs on the DAD-MCU-BRD and the STM32F429 microcontroller.

LED	**GPIO Pin**	Colour
LED1	PC1	Red
LED2	PC4	Orange
LED3	PC5	Yellow
LED4	PG14	Green

Table 4.3: Debugging LEDs on the DAD-MCU-BRD

4.4.2 Reset and DFU Bootloader Switches

There are two tactile switches, SW1 and SW2, featured on the DAD-MCU-BRD and they are shown in Figure A.1.

The first tactile switch, SW1, resets the STM32F429 microcontroller without disconnecting the power supply to the real-time audio analyser. The resetting of the real-time audio analyser may be required if the firmware executed by the device encounters a bug or a fault in the developed code.

The second tactile switch, SW2, allows the user to access the Default Firmware Update (DFU) Bootloader and reload or program the STM32F429 microcontroller with updated firmware. The DFU Bootloader is an alternative way of programming the STM32F429 microcontroller without the use of an in-circuit debugger/programmer such as the ST-LINK/V2 developed by STMicroelectronics.

4.4.3 Mode Selection Jumpers

There are three jumpers, JP1, JP2 and JP3, featured on the DAD-MCU-BRD and are shown in Figure A.1.

The first jumper, JP1 (PWR-SEL), allows the user to select the power supply source for the real-time audio analyser. Pins 1-2 on JP1 selects the 5V USB-mini-B port via connector `CN8 (VUSB)` as the power source. Pins 2-3 on JP1 selects the external power supply via connector `CN5 (VBAT)` as the power source.

The second and third jumpers, JP2 and JP3, are mode selection jumpers for future revisions of the developed firmware executed by the real-time audio analyser. In future revisions the second jumper, JP2, will select the output display mode, either the TFT-LCD or the VGA controller. The third jumper, JP3, will select the mode of operation for the real-time audio analyser in future revisions of the developed firmware, either an audio spectrum analysis or an audio spectrogram (waterfall) analysis.

4.5 TFT-LCD Screen, Touch Screen Controller and SD Card Socket

The Thin-Film Transistor Liquid Crystal Display (TFT-LCD) module used for this project was imported from ITead Studio, and at the time of procurement was a cheaper display option compared to the displays sourced from local suppliers, such as RS-Components (SA), Netram Technologies, Micro Robotics, Riecktron - Embedded Solutions, and Communica.

4.5.1 TFT-LCD Screen

The ITDB02-3.2S module shown in Figures 4.10 and 4.11 feature a 3.2" Thin-Film Transistor Liquid Crystal Display with a 320 x 240 pixel resolution and offers 65K colours[4].

Figure 4.10: ITDB02-3.2S TFT-LCD Module from ITead Studio[4]

The LCD controller for the 3.2" Thin-Film Transistor Liquid Crystal Display is the SSD1289 LCD controller, developed by Soloman Systech. The SSD1289 LCD controller featuring 172800 bytes of built-in GDDRAM communicates 16-bit parallel bi-directional data with wires (DB[0:15]) and is controlled with 4 wires using the 8080-series system bus interface[42][43].

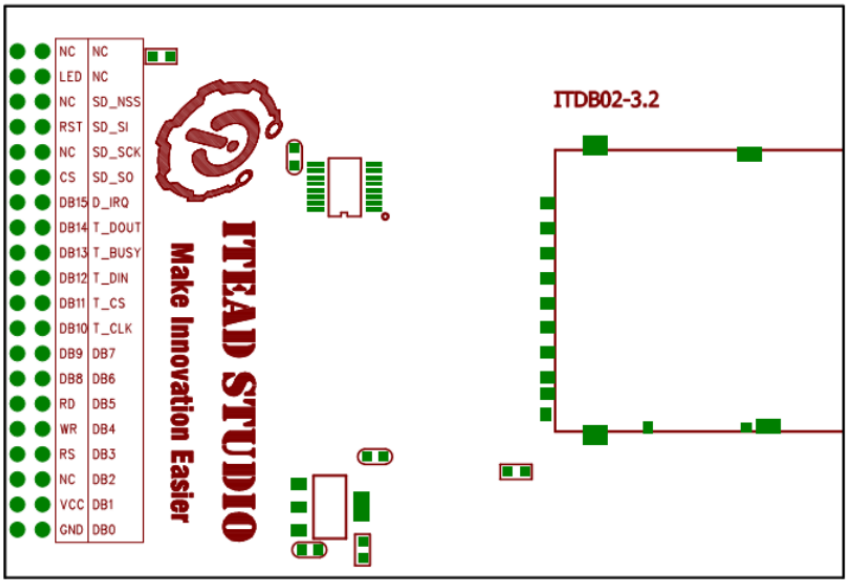


Figure 4.11: ITDB02-3.2S TFT-LCD Module (Bottom View) from ITead Studio[4]

Table 4.4 shows the connections between the SSD1289 LCD controller on the ITDB02-3-2S module and the STM32F429 microcontroller.

ITDB02-3-2S Pin	**Name**	**Description**	**GPIO Pin**
RD	$\overline{\text{RD}}$	Read Data Input	PD4
WR	$\overline{\text{WR}}$	Write Data Input	PD5
RS	D/$\overline{\text{C}}$	Data/Command Input	PG13
CS	$\overline{\text{CS}}$	Chip Select Input	PD7
DB0	Data[0]	Data Input/Output	PD14
DB1	Data[1]	Data Input/Output	PD15
DB2	Data[2]	Data Input/Output	PD0
DB3	Data[3]	Data Input/Output	PD1
DB4	Data[4]	Data Input/Output	PE7
DB5	Data[5]	Data Input/Output	PE8
DB6	Data[6]	Data Input/Output	PE9
DB7	Data[7]	Data Input/Output	PE10
DB8	Data[8]	Data Input/Output	PE11
DB9	Data[9]	Data Input/Output	PE12
DB10	Data[10]	Data Input/Output	PE13
DB11	Data[11]	Data Input/Output	PE14
DB12	Data[12]	Data Input/Output	PE15
DB13	Data[13]	Data Input/Output	PD8
DB14	Data[14]	Data Input/Output	PD9
DB15	Data[15]	Data Input/Output	PD10

Table 4.4: SSD1289 Control and Data Pins

4.5.2 Touch Screen Controller

The TFT-LCD module also features a resistive touch screen controller as a 16-pin TSSOP package shown in Figure 4.11. The XPT2046 touch screen controller, developed by Shenzhen Xptek Technology Limited, interfaces with the STM32F429 microcontroller using the full-duplex, 4-wire, Serial Peripheral Interface (SPI) digital communication protocol developed by Motorola[44]. Moreover, the `SPI1` module featured on the STM32F429 microcontroller interfaces with the XPT2046 touch screen controller. The XPT2046 is fabricated as a 16-pin TSSOP package and requires a 3.3V power supply[45].

Depending on where the TFT-LCD module was manufactured and populated, the ITDB02-3-2S module may feature an alternative touch screen controller. Two possible alternative touch screen controllers are the ADS7846 and the TSC2046 both developed by Texas Instruments. All three touch controllers interface and communicate with the microcontroller using the 3-wire SPI protocol and all three touch controllers feature two additional pins: Pen Interrupt Input ($\overline{\text{T-PENIRQ}}$), and Busy Output (T-BUSY). Table 4.5 shows the connections between the XPT2046 touch screen controller on the ITDB02-3-2S module and the STM32F429 microcontroller.

ITDB02-3-2S Pin	Name	Description	GPIO Pin
T_CLK	T-CLK	Clock	PA5
T_CS	T-CS	Chip Select Input	PA0
T_DIN	T-DIN	Data Input	PA7
T_DOUT	T-DOUT	Data Output	PB4
T_BUSY	T-BUSY	Busy Output	PA2
T_IRQ	$\overline{\text{T-PENIRQ}}$	Pen Interrupt Input	PA9

Table 4.5: XPT2046 Control and Data Pins

4.5.2.1 SD Card Socket

The TFT-LCD module also features an SD card socket[4]. The SD card socket interfaces using the 4-wire SPI protocol, namely the SPI1 module featured on the STM32F429 microcontroller. Table 4.6 shows the connections between the SD Card socket on the ITDB02-3-2S module and the STM32F429 microcontroller.

ITDB02-3-2S Pin	Name	Description	GPIO Pin
SD_SCK	SD-SCK	SD SCK	PA5
SD_NSS	SD-NSS	SD NSS	PA7
SD_SI	SD-SI	SD MOSI	PA0
SD_SO	SD-SO	SD MISO	PA2

Table 4.6: XPT2046 Control and Data Pins

It must be noted that the XPT2046 touch screen controller and the SD card socket both share the SPI1 module featured on the STM32F429 microcontroller. The transmission of data to/from either of the two devices is dependent on the state of the chip-select pin (T-CS) for the XPT2046 touch screen controller and the state of the slave-select pin (SD-NSS) for the SD card socket. Data is transferred to/from the XPT2046 touch screen controller when the chip-select pin (T-CS) is logic 1, while data is transferred to/from the SD card socket when the slave-select pin (SD-NSS) is logic 0.

4.6 External Memory

The real-time audio analyser features two additional external memories, namely 1MB Static Random Access Memory (SRAM) and 16MB Synchronous Dynamic Random Access Memory (SDRAM), both fabricated by Alliance Memory and procured from RS-Components (SA). Both external memories interface using the Flexible Memory Controller (FMC) peripheral featured on the STM32F429 microcontroller.

The additional memory extension board DAD-MEM-BRD utilizes the external memories and is discussed in chapter 7.

4.6.1 AS6C8016 SRAM

The AS6C8016 features a total 8Mbits or 1MByte of low-power CMOS static random access memory and is internally configured as 512K words x 16 bits. The AS6C8016 SRAM is fabricated as a 44-pin TSOP-II package and requires a 3.3V power supply. Figure 4.12 illustrates both the control and data communication connections[5].

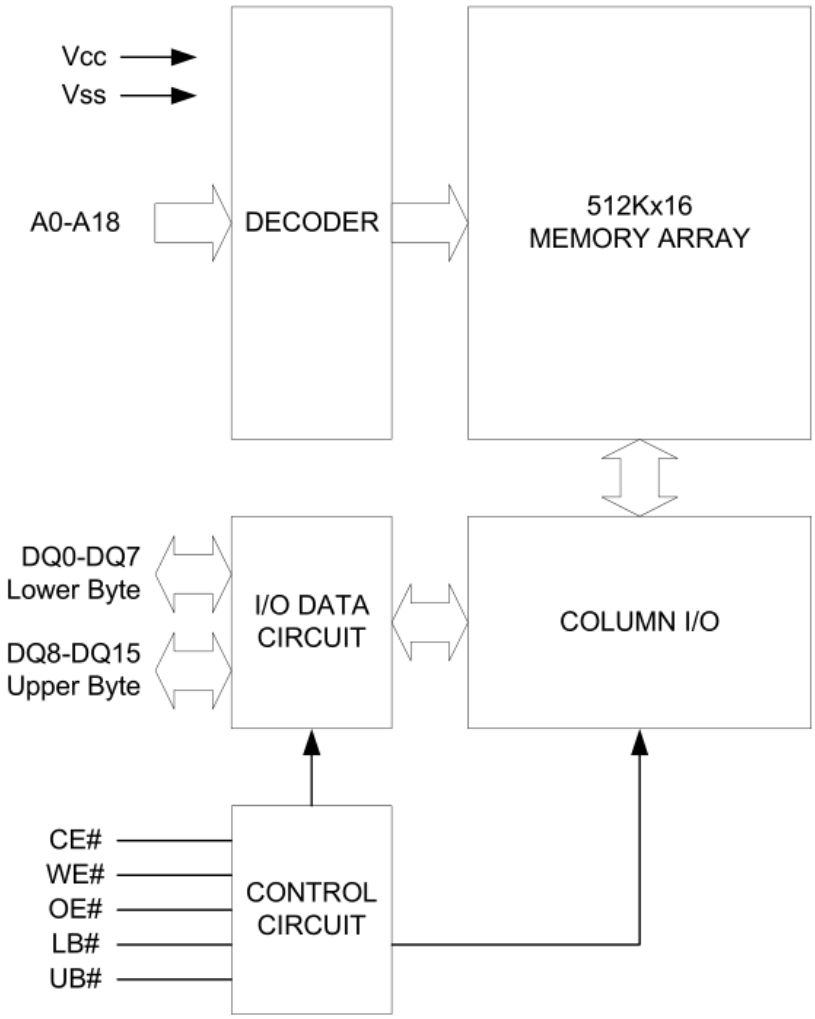


Figure 4.12: SRAM Block Diagram[5]

Table 4.7 shows the connections between the external AS6C8016 SRAM and the STM32F429 microcontroller.

AS6C8016 Pin	**Name**	**Description**	**GPIO Pin**
A0	FMC-A0	Address Input	PF0
A1	FMC-A1	Address Input	PF1
A2	FMC-A2	Address Input	PF2
A3	FMC-A3	Address Input	PF3
A4	FMC-A4	Address Input	PF4
A5	FMC-A5	Address Input	PF5
A6	FMC-A6	Address Input	PF12
A7	FMC-A7	Address Input	PF13
A8	FMC-A8	Address Input	PF14
A9	FMC-A9	Address Input	PF15
A10	FMC-A10	Address Input	PG0
A11	FMC-A11	Address Input	PG1
A12	FMC-A12	Address Input	PG2
A13	FMC-A13	Address Input	PG3
A14	FMC-A14	Address Input	PG4
A15	FMC-A15	Address Input	PG5
A16	FMC-A16	Address Input	PD11
A17	FMC-A17	Address Input	PD12
A18	FMC-A18	Address Input	PD13
D0	FMC-D0	Data Input/Output	PD14
D1	FMC-D1	Data Input/Output	PD15
D2	FMC-D2	Data Input/Output	PD0
D3	FMC-D3	Data Input/Output	PD1
D4	FMC-D4	Data Input/Output	PE7
D5	FMC-D5	Data Input/Output	PE8
D6	FMC-D6	Data Input/Output	PE9
D7	FMC-D7	Data Input/Output	PE10
D8	FMC-D8	Data Input/Output	PE11
D9	FMC-D9	Data Input/Output	PE12
D10	FMC-D10	Data Input/Output	PE13
D11	FMC-D11	Data Input/Output	PE14
D12	FMC-D12	Data Input/Output	PE15
D13	FMC-D13	Data Input/Output	PD8
D14	FMC-D14	Data Input/Output	PD9
D15	FMC-D15	Data Input/Output	PD10
CE#	FMC-NE2	Chip Select	PG9
OE#	FMC-NOE	Output Enable	PD5
WE#	FMC-NWE	Write Enable	PD4
LB#	FMC-NBL0	Lower Byte Control	PE0
UB#	FMC-NBL1	Upper Byte Control	PE1

Table 4.7: AS6C8016 Control and Data Pins

4.6.2 AS4C8M16S SDRAM

The AS4C8M16S features a total 128Mbits or 16MBytes of high-speed CMOS synchronous DRAM and is internally configured as 4 separate banks consisting of 2M words x 16 DRAM. The AS4C8M16S SDRAM is fabricated as a 54-pin TSOP-II package and requires a 3.3V power supply. Moreover, all digital communication signals are positively edge triggered via the external clock signal connected to AS4C8M16S SDRAM[6].

Figure 4.13 illustrates both the control and data communications.

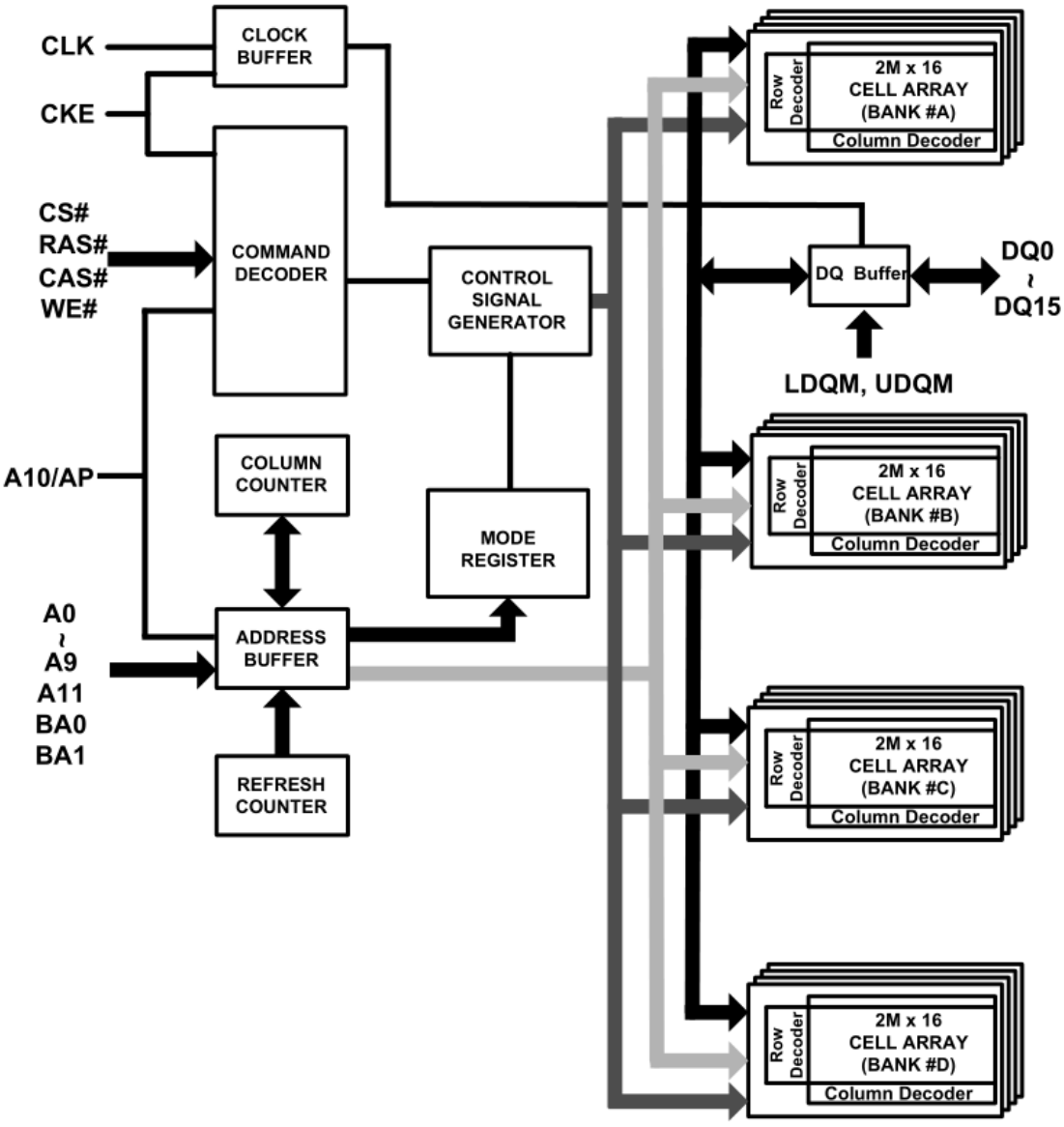


Figure 4.13: SDRAM Block Diagram[6]

Table 4.8 shows the connections between the external AS4C8M16S SDRAM and the STM32F429 microcontroller.

AS4C8M16S Pin	**Name**	**Description**	**GPIO Pin**
A0	FMC-A0	Address Input	PF0
A1	FMC-A1	Address Input	PF1
A2	FMC-A2	Address Input	PF2
A3	FMC-A3	Address Input	PF3
A4	FMC-A4	Address Input	PF4
A5	FMC-A5	Address Input	PF5
A6	FMC-A6	Address Input	PF12
A7	FMC-A7	Address Input	PF13
A8	FMC-A8	Address Input	PF14
A9	FMC-A9	Address Input	PF15
A10	FMC-A10	Address Input	PG0
A11	FMC-A11	Address Input	PG1
DQ0	FMC-D0	Data Input/Output	PD14
DQ1	FMC-D1	Data Input/Output	PD15
DQ2	FMC-D2	Data Input/Output	PD0
DQ3	FMC-D3	Data Input/Output	PD1
DQ4	FMC-D4	Data Input/Output	PE7
DQ5	FMC-D5	Data Input/Output	PE8
DQ6	FMC-D6	Data Input/Output	PE9
DQ7	FMC-D7	Data Input/Output	PE10
DQ8	FMC-D8	Data Input/Output	PE11
DQ9	FMC-D9	Data Input/Output	PE12
DQ10	FMC-D10	Data Input/Output	PE13
DQ11	FMC-D11	Data Input/Output	PE14
DQ12	FMC-D12	Data Input/Output	PE15
DQ13	FMC-D13	Data Input/Output	PD8
DQ14	FMC-D14	Data Input/Output	PD9
DQ15	FMC-D15	Data Input/Output	PD10
BA0	FMC-SDBA0	Bank Activate 0	PG4
BA1	FMC-SDBA1	Bank Activate 1	PG5
CLK	FMC-SDCLK	SDRAM Clock	PG8
CKE	FMC-SDCKE0	Clock Enable	PC2
CS#	FMC-SDNE0	Chip Enable	PC3
CAS#	FMC-SDNCAS	Column Address Strobe	PG15
RAS#	FMC-SDNRAS	Row Address Strobe	PF11
WE#	FMC-SDNWE	Write Enable	PC0
LDQM	FMC-SDNBL0	Lower Byte Control	PE0
UDQM	FMC-SDNBL1	Upper Byte Control	PE1

Table 4.8: AS4C8M16S Control and Data Pins

4.7 Audio Data Converters

The designed real-time audio analyser features two possible audio data converters, namely the CS4272 CODEC and the internal STM32F429 ADC.

4.7.1 Internal STM32F429 ADC

The designed real-time audio analyser features the STM32F429 internal on-chip ADC which is a 12-bit successive approximation analogue-to-digital converter. An external microphone can interface with the internal ADC featured on the STM32F429 microcontroller through one of several input channels shown in Table 4.9.

STM32F429 ADC Input Channel	GPIO Pin
$ADC3_{IN4}$	PF6
$ADC3_{IN5}$	PF7
$ADC3_{IN6}$	PF8
$ADC3_{IN7}$	PF9
$ADC3_{IN8}$	PF10

Table 4.9: Internal ADC on DAD-MCU-BRD

4.7.2 External CS4272 CODEC

The real-time audio analyser DAD-AV-BRD shown in Figure A.7 features a 24bit, 192kHz stereo audio CODEC, namely the CS4272 CODEC fabricated by Cirrus Logic and procured from RS-Components (SA). The CS4272 CODEC is fabricated as a 28-pin TSSOP package and requires a 3.3V digital power supply in addition to the 5V analogue power supply[7]. The CS4272 CODEC shown in Figure 4.14 illustrates the stereo analogue-to-digital (A/D) and digital-to-analogue (D/A) converters both capable of conversions of up to 24-bits of data, at sample rates of up to 192kHz[7].

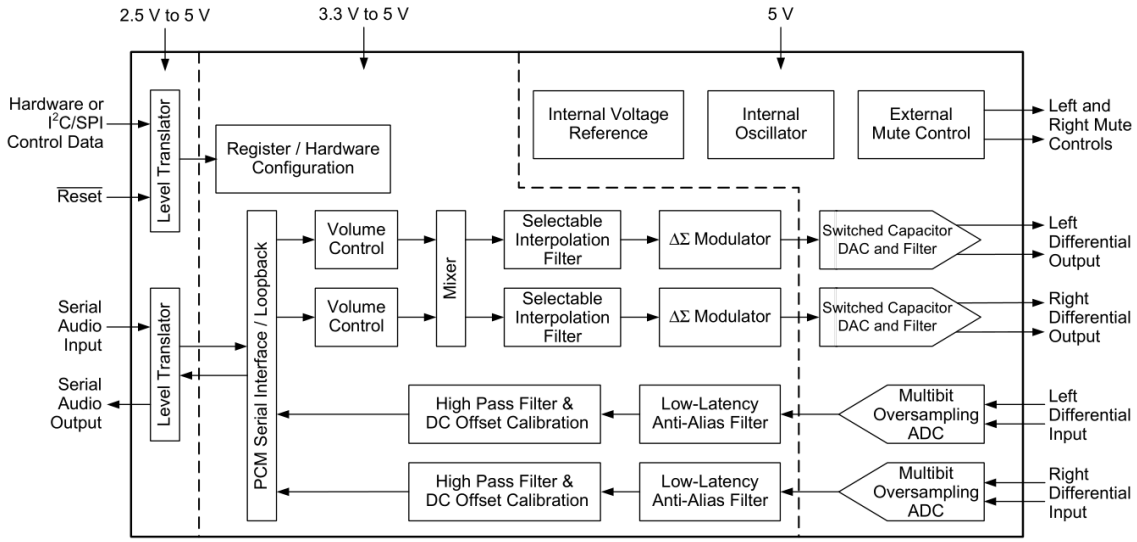


Figure 4.14: CS4272 Block Diagram[7]

The CS4272 CODEC interfaces using the Serial Audio Interface (SAI) digital communication peripheral featured on the STM32F429 microcontroller. The CS4272 CODEC operates in software-mode and therefore offers configurable volume and muting controls which can be accessed in firmware[7]. The CS4272 CODEC is controlled and configured using the serial, half-duplex, 2-wire, bidirectional, Inter-Integrated Circuit (I^2C) digital communication protocol developed by NXP Semiconductors, formerly called Philips Semiconductors and a division of Philips[46]. Moreover, the `I2C1` module featured on the STM32F429 microcontroller interfaces with the CS4272 CODEC. Table 4.10 shows the control connections between the CS4272 CODEC and the STM32F429 microcontroller.

CS4272 Pin	**Name**	**Description**	**GPIO Pin**
Pin 11	SCL/CCLK (M0)	Control Clock Input	PB6
Pin 12	SDA/CDIN (M1)	Control Data Input	PB7
Pin 13	AD0/$\overline{\text{CS}}$ (I2S/$\overline{\text{LJ}}$)	Control Address Bit	0V
Pin 14	$\overline{\text{RST}}$	Reset Input	PB5

Table 4.10: CS4272 Control Pins

The I^2C address for the CS4272 CODEC is `001000` where the LSB of the 7-bit address is determined by the value of pin 13 called AD0[7].

The CS4272 CODEC transmits digital data using the full-duplex, 5-wire, Inter-IC Sound (I^2S) serial audio data communication protocol developed by NXP Semiconductors, formerly called Philips Semiconductors and a division of Philips[47]. The CS4272 CODEC operates in Slave Mode with the STM32F429 microcontroller acting as the master for the serial communication protocol and therefore the STM32F429 microcontroller supplies the CS4272 CODEC with the correctly synchronised clocks sources.

Table 4.11 shows the serial data communication and synchronised clocks used by the I^2S serial data communication protocol featured on the STM32F429 microcontroller.

CS4272 Pin	**Name**	**Description**	**GPIO Pin**
Pin 3	MCLK	Master Clock Input	PE2
Pin 4	LRCK	Left/Right Clock Input	PE4
Pin 5	SCLK	Serial Clock Input	PE5
Pin 6	SDOUT	Serial Data Output	PE3
Pin 7	SDIN	Serial Data Input	PE6

Table 4.11: CS4272 Serial Audio Data and Clocks

As a side note, the I^2S serial data communication protocol can be implemented using two SPI modules simultaneously[48], however this is not a requirement for the designed real-time audio analyser.

4.7.2.1 Analogue/Digital Front-End

The analogue audio signal is passed through an operational amplifier buffering circuit before entering the CS4272 CODEC analogue inputs: AINA+, AINA-, AINB+, and AINB-. The input buffer circuit attenuates any noise from the digital clocks and digital communication signals. Moreover, the input buffer circuit provides a suitable source impedance for the delta-sigma modulators[7].

The real-time audio analyser DAD-AV-BRD features both a fully differential input and a single-ended input. The input buffer circuits make use of the MCP664 and the output buffer circuits make use of the MCP662 low-power operational amplifier manufactured by Microchip Technology. Both the MCP662 and the MCP664 feature a typical gain bandwidth of 60MHz, a slew rate of 32V/μs and require a 3.3V power supply[49]. The only difference between the two operational amplifiers is the number of op-amps housed and fabricated within the silicon package. The MCP664 is fabricated as a 14-pin SOIC package featuring 4 op-amps[49].

The fully differential input featured on the DAD-AV-BRD caters for either an external microphone using `CN8 (XLR_IN)` or an internal differential microphone using `CN7 (MIC3)` providing jumpers `JP1 (IN3+)` and `JP2 (IN3-)` are configured appropriately. The fully differential input buffer circuit given in Figure 4.15 consists of a high-pass RC filter with a DC biasing capacitor and a low-pass op-amp feedback filter with an anti-aliasing capacitor.

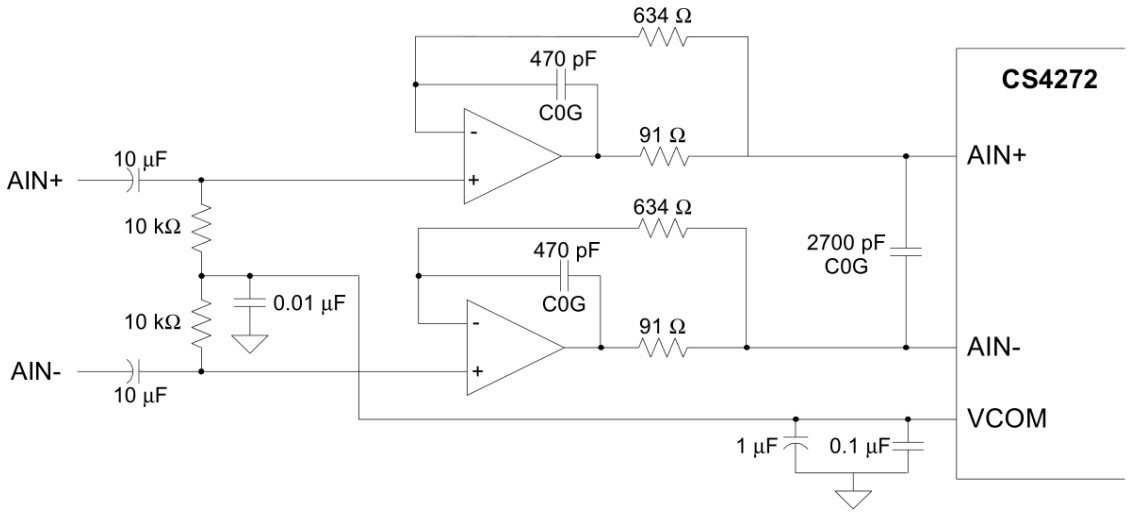


Figure 4.15: Fully Differential Input Buffer Circuit[7]

The corner frequency for a high-pass RC filter can be determined by:

$$f_c = \frac{1}{2\pi RC} \tag{4.4}$$

The 3dB corner frequency for the high-pass RC filter ($R = 10k\Omega$ and $C = 10\mu F$) in Figure 4.15 is $f_c = 1.59Hz$.

The op-amp circuit is a low-pass filter topology with a flat frequency response in the audio spectrum and provides approximately 20dB of noise rejection from the high-frequency digital

clocks [8]. Moreover, the op-amp filter provides a suitable low output impedance for the analogue-to-digital delta-sigma modulators in the CS4272 CODEC[8].

The single-ended input on the DAD-AV-BRD allows an internal single-ended microphone using `CN6 (MIC2)`. The single-ended input buffer circuit, Figure 4.16, consists of a high-pass RC filter with a DC biasing capacitor and a low-pass op-amp feedback filter with an anti-aliasing capacitor. The 3dB corner frequency for the high-pass RC filter ($R = 100k\Omega$ and $C = 1\mu F$) in Figure 4.16 is $f_c = 1.59Hz$.

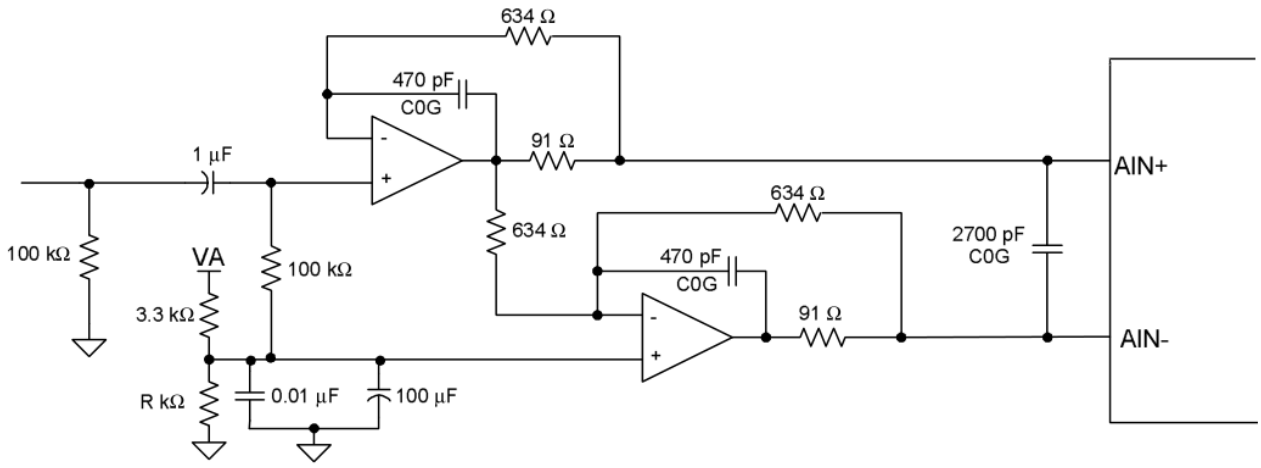


Figure 4.16: Single-Ended Input Buffer Circuit[8]

The op-amp circuit consists of a low-pass filter topology with a flat frequency response in the audio spectrum and provides approximately 20dB of noise rejection from the high-frequency digital clocks[8]. Moreover, the op-amp filter provides a suitable low output impedance for the analogue-to-digital delta-sigma modulators in the CS4272 CODEC[8]. The unvalued resistor R, in Figure 4.16, fixes the optimal DC biased voltage for the CS4272 CODEC single-ended to differential input buffer, which is usually 3.3V.

4.7.2.2 Digital/Analogue Back-End

The analogue audio signal is passed through a buffering circuit after leaving the CS4272 CODEC analogue outputs: AOUTA+, AOUTA-, AOUTB+, and AOUTB-. The output buffer circuit consists of a 2-pole low-pass Butterworth filter with a unity-gain response and is followed by a RC high-pass filter[50]. The output buffer circuit attenuates noise from the CS4272 CODEC and removes common-mode errors[51]. The MCP662 low power operational amplifier is fabricated as a 8-pin SOIC package featuring 2 op-amps[49]. Figure 4.17 shows the schematic for the output buffer circuit.

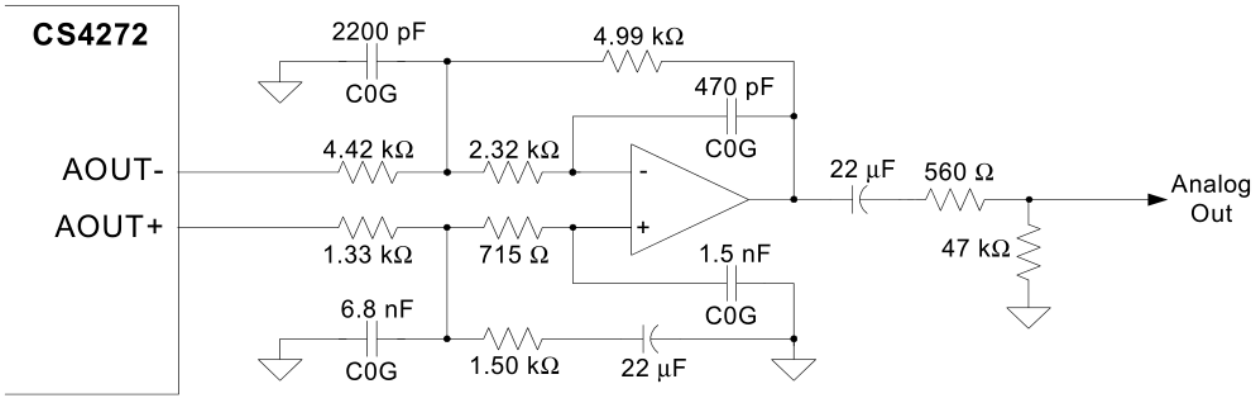


Figure 4.17: Differential to Single-Ended Output Buffer[7]

4.8 External and Internal Microphones

The designed hardware for the real-time audio analyser allows a total of four possible microphone sources to be connected, namely three internal MEMS microphones and one external analogue microphone.

- internal digital single-ended MEMS microphone
- internal analogue single-ended MEMS microphone
- internal analogue differential MEMS microphone
- external analogue differential microphone

4.8.1 Internal Microphone

There are a total of three possible internal microphone connections, namely `CN5` (MIC1), `CN6` (MIC2), and `CN7` (MIC3).

A digital internal MEMS microphone can be connected to the DAD-AV-BRD via `CN5` (MIC1), provided a suitable digital clock is generated using the `I2S` peripheral module featured on the STM32F429 microcontroller. An analogue internal MEMS microphone can be connected to the DAD-AV-BRD via either a single-end input `CN6` (MIC2) or via a differential input `CN7` (MIC3), provided jumpers `JP1` and `JP2` are set appropriately.

Examples of possible MEMS microphones that could be connected to he DAD-AV-BRD are discussed below, namely the ADMP401, the C928A, and the MP34DT01. Only the first ADMP401 MEMS microphone was used in the testing of the designed and implemented real-time audio analyser.

4.8.1.1 ADMP401 Single-Ended Analogue MEMS Microphone

The ADMP401, manufactured by Analog Devices, is a single-ended analogue MEMS microphone featuring a flat frequency response from 100Hz to 15kHz with a high SNR of 62dBA, and sensitivity of -42dBV. Moreover, the ADMP401 is fabricated as a 6-pin LGA surface mount package with a bottom-port omnidirectional microphone and requires a 3.3V power supply[14].

The developed firmware executed by the real-time audio analyser and presented in this research paper was extensively tested using the internal STM32F4 12-bit ADC and the ADMP401 microphone break-out board shown in Figure 4.18, purchased from Riecktron - Embedded Solutions.

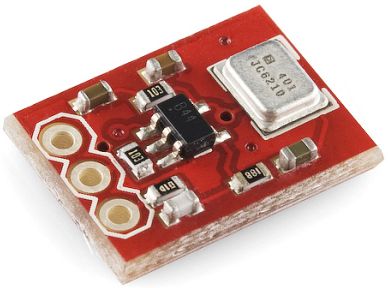


Figure 4.18: ADMP401 MEMS microphone break-out board[9]

The ADMP401 microphone break-out board was connected to the real-time analyser DAD-MCU-BRD via one of the internal ADC input pins shown in Table 4.9. Alternatively, the ADMP401 microphone break-out board can be connected to the real-time audio analyser DAD-AV-BRD via `CN6` (MIC2).

It should be noted that the ADMP401 is no longer fabricated by Analog Devices and has since become obsolete. An alternative MEMS microphone identical to the ADMP401 microphone is the INMP401 MEMS microphone manufactured by InvenSense features the same electrical and mechanical specifications[52].

4.8.1.2 C928A Differential Analogue MEMS Microphone

The C928A manufactured by TDK-EPCOS Corporation is a differential analogue MEMS microphone featuring a very high SNR of 66dBA for the frequency range 20Hz to 20kHz, and sensitivity of -38dBV/Pa. Moreover, the C928A microphone is fabricated as a 6-pin 3.35mm x 2.5mm x 1mm surface mount package with a bottom-port omnidirectional microphone requiring a 3.3V power supply[53].

A break-out board for the C928A microphone was designed using EAGLE and are shown in

Figures A.13, A.14, and A.15. However, the break-out board for the C928A microphone was not fabricated since the ADMP401 microphone break-out provided a sufficient signal level to test the developed firmware code. The designed C928A microphone break-out board will be explored in future work.

4.8.1.3 MP34DT01 Single-Ended Digital MEMS Microphone

The MP34DT01 manufactured by STMicroelectronics is a single-ended digital MEMS microphone featuring a high SNR of 63dBA and sensitivity of -26dBFS. Moreover, the MP34DT01 microphone is fabricated as a 5-pin HCLGA surface mount package with a top-port omnidirectional microphone requiring a 3.3V power supply. Furthermore, the MP34DT01 requires a 2.4MHz input clock frequency generated by the STM32F429 microcontroller `I2S` peripheral and outputs a digital PDM formatted signal[54].

A break-out board for the MP34DT01 microphone was designed using EAGLE and are shown in Figures A.16, A.17, and A.18. However, the break-out board for the MP34DT01 microphone was not fabricated for the same reasons mentioned above, namely the ADMP401 microphone break-out board provided a sufficient signal level and performance to test the developed firmware code. The designed MP34DT01 microphone break-out board will be explored in future work.

4.8.2 External Microphone

An external microphone can be connected to the DAD-AV-BRD via connector `CN8` on the DAD-AV-BRD. It must be noted that the current hardware for the real-time audio analyser does not provide 48V phantom power which is required by condenser microphones. An external condenser microphone will need to source the additional phantom power externally. The inclusion of phantom power will be explored in future revisions of the hardware. Moreover, the future hardware revisions may allow the real-time audio analyser to source the 48V phantom power from an audio mixing desk console.

4.9 Video Graphics Array (VGA) Controller

A Video Graphics Array (VGA) Controller is a common device found on laptops, personal computers and even some tablets. The VGA controller offers alternative viewing capabilities and can also reduce the need for the costly TFT-LCD, provided a monitor or projector screen is available.

A brief overview of the designed hardware for the VGA controller is given in Figure 4.19.

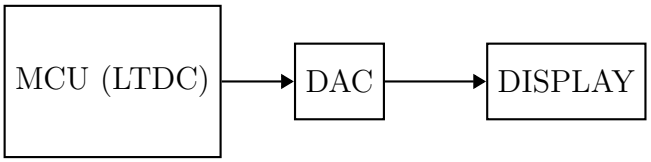


Figure 4.19: Overview of designed VGA hardware

Table 4.12 and Figure 4.20 illustrate the VGA specifications for both the analogue RGB signals and the separate synchronised timing information required to interface with an external video display device such as a CRT monitor or projector screen.

VGA Pin	Name	Description	Impedance
Pin 1	Red	0.7V	75 ohms
Pin 2	Green	0.7V	75 ohms
Pin 3	Blue	0.7V	75 ohms
Pin 4	Reserved (N/A)	-	-
Pin 5	Ground	0V	-
Pin 6	Red Ground	0V	-
Pin 7	Green Ground	0V	-
Pin 8	Blue Ground	0V	-
Pin 9	+5V DC (N/A)	5V	-
Pin 10	Sync Ground	0V	-
Pin 11	Reserved (N/A)	-	-
Pin 12	DDC SDA (N/A)	$\geq$2.4V	-
Pin 13	HSYNC	$\geq$2.4V	-
Pin 14	VSYNC	$\geq$2.4V	-
Pin 15	DDC SCL (N/A)	$\geq$2.4V	-

Table 4.12: VGA Connection Pins[10]

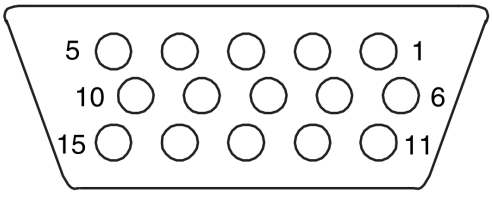


Figure 4.20: VGA 15-Pin D-SUB Female Connector[10]

The designed VGA controller on the DAD-AV-BRD interfaces with the STM32F429 microcontroller via the internal Liquid-TFT Display Controller (LTDC). The LTDC outputs 16-bit data consisting of five red bits, six green bits, and 5 blue bits (RGB565). The digital video data is converted by a series of digital-to-analogue converters into three separate analogue signals, red (R), green (G), and blue (B). Moreover, the digital-to-analogue converters output the analogue video signals with the correct impedance and peak-voltage as specified in Table 4.12. Table 4.13 shows the digital data and separate synchronised timing signals generated by the LTDC featured on the STM32F429 microcontroller.

Name	LTDC Pin	GPIO Pin
R4	Red Data[7]	PG6
R3	Red Data[6]	PB1
R2	Red Data[5]	PA12
R1	Red Data[4]	PA11
R0	Red Data[3]	PB0
G5	Green Data[7]	PD3
G4	Green Data[6]	PC7
G3	Green Data[5]	PB11
G2	Green Data[4]	PB10
G1	Green Data[3]	PG10
G0	Green Data[2]	PA6
B4	Blue Data[7]	PB9
B3	Blue Data[6]	PB8
B2	Blue Data[5]	PA3
B1	Blue Data[4]	PG12
B0	Blue Data[3]	PG11
HSYNC	Horizontal Sync	PC6
VSYNC	Vertical Sync	PA4

Table 4.13: Liquid-TFT Display Controller (LTDC) RGB565 Digital Video Output

4.9.1 Video Digital-to-Analogue Converters (DAC)

The primary hardware for a VGA controller is a digital-to-analogue converter (DAC) which can be implemented using either a Binary-Weight DAC or a R-2R Resistor Network DAC. The R-2R DAC is similar to a binary-weighted DAC however each successive resistor network has resistance values R and 2R. The R-2R DAC provides ease in matching resistor values and the use of resistor packs can reduce the amount of space taken up by the VGA hardware on the printed circuit board (PCB).

The designed R-2R DAC for the VGA display output consists of three separate R-2R DAC networks - one for each analogue signal line with a 75Ω impedance. The analogue signal lines are Red (R), Green (G), and Blue (B). Figure 4.21 illustrates both the Red and Blue 5-bit R-2R DAC networks. The Green 6-bit R-2R DAC network contains an additional 2R and R resistor.

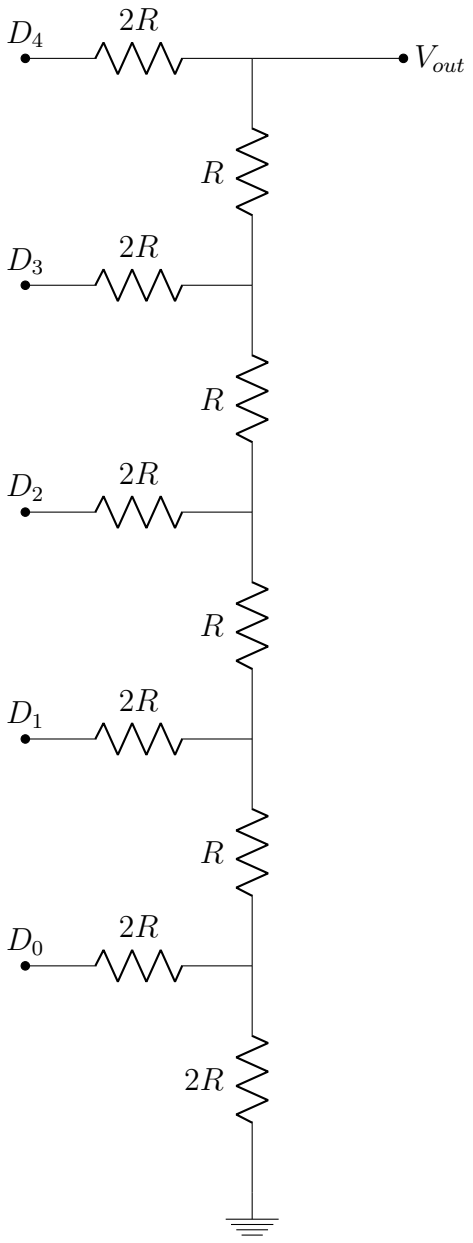


Figure 4.21: 5-bit R-2R DAC Network

Determining the correct R and 2R values is crucial to providing the required analogue signal line impedance of 75Ω. Both the Thevenin and Norton theorems for equivalent circuit analysis aid in determining the necessary impedance values.

Each digital output (D_0, D_1, D_2, D_3, D_4) has the following possible logic values/voltages: $0 \equiv 0V$ and $1 \equiv 3.3V$. When all 5 bits of the R-2R DAC network are logic 0 (0V) then the equivalent resistance is determined to be R by using the properties of equivalent resistances in series and parallel. The equivalent resistance network is illustrated in Figure 4.22 and is connected via V_{out} to the display monitor with an input impedance of 75Ω.

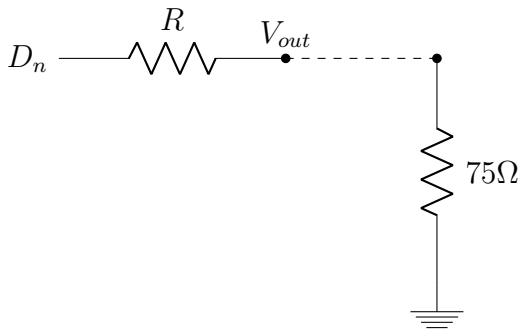


Figure 4.22: Reduced Equivalent R-2R DAC connected to 75Ω display monitor

The 75Ω resistor is the input impedance of the display monitor and is not part of the R-2R DAC network.

Recall, the potential divider circuit illustrated in Figure 4.23.

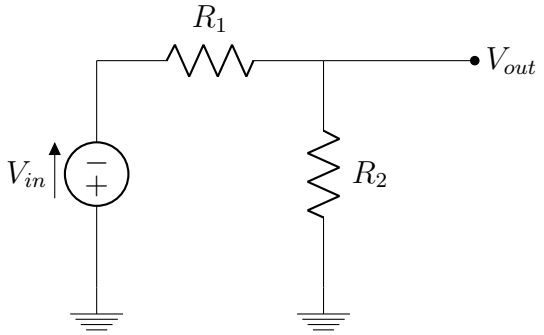


Figure 4.23: Potential Divider Circuit

The output voltage for the potential divider circuit is given by:

$$V_{out} = \left(\frac{R_2}{R_1 + R_2}\right) \times V_{in} \tag{4.5}$$

Hence, rearranging equation 4.5, the value for R_1 is given by:

$$R_1 = \left(\frac{V_{in}}{V_{out}} \times R_2\right) - R_2 \tag{4.6}$$

Now for the equivalent R-2R DAC network: $V_{in} = 3.3V$, $V_{out} = 0.7V$, $R_1 = R$, $R_2 = 75\Omega$. Therefore, $R_1 = R = 278.57\Omega$. The closest resistor values were used in the design of the R-2R DAC network, namely $R = 270\Omega$.

CHAPTER 5

The Real-Time Audio Analyser Firmware

This chapter discusses the developed firmware or C code that is executed by the STM32F4 microcontroller. The firmware loaded onto the STM32F4 microcontroller was written in C - not C++. The C code was initially developed using the Keil Microcontroller Development Kit for ARM based processors (MDK-ARM). The Keil MDK-ARM software provides a complete software development range for ARM, Cortex-M, and Cortex-R processors and microcontrollers. The MDK-ARM software consists of the μVision IDE with built-in debugger and simulator, the ARM C/C++ Compiler, and several other middleware components. Currently the latest version available for download is MDK-ARM V5.16a[1] and the developers do release updates regularly. The only disadvantage of Keil MDK-ARM Evaluation/Lite Edition license is that the code and data will not compile, assemble, or link if the total size of code exceeds 32 KBytes. A license for Keil MDK-ARM without any limitations can be purchased from local company Avnet South Africa at the sizeable cost of between R45 972.98 and R136 470.98 - depending on the exact edition (Cortex-M Edition, Standard Edition, or Professional Edition). While the limitation of the size of code compilation did not initially hinder the development of the firmware code, this limitation did prove to be a nuisance towards the end of this research project.

Fortunately some ARM employees are maintaining and committing regularly to an open-source alternative C compiler, namely a GNU toolchain with a GCC compiler system targeted at embedded ARM, Cortex-M, and Cortex-R based processors and microcontrollers, namely the GCC ARM Embedded Toolchain[2]. Since the creation of the project in 2011, the developers and maintainers have consistently released updates at quarterly year intervals with the most recent version titled GCC ARM Embedded 6-2016-q4-major. Several open source code devel-

[1] `https://www.keil.com/download/product/`
[2] `https://launchpad.net/gcc-arm-embedded/`

opment solutions are freely available, such as emIDE[3], and Code::Blocks[4], however not all code development solutions offer a built-in debugger which allows the microcontroller code to be debugged in real-time.

The final firmware for the implemented real-time audio analyser was developed with the GCC ARM Embedded Toolchain and the C code was written primarily using the free open-source IDE Em::Blocks[5] that no longer has full support. Moreover, the software developers of Em::Blocks have begun beta testing an alternative and newer IDE, EmBitz, and the firmware code complies and has been loaded successfully onto the real-time audio analyser microcontroller.

5.1 Programming the STM32F4 Microcontroller

Electronic vendors that fabricate microcontrollers containing an ARM Cortex core have a linked programmer that allows the microcontrollers to be loaded with the developed code. STMicroelectronics provide the ST-LINK/V2 in-circuit debugger/programmer[55] and can be purchased from RS-Components (SA). The ST-LINK/V2 connects to the target STM32F4 microcontroller and loads the code developed using either the Keil (MDK-ARM) μVision editor or an open source alternative such as Em::Blocks.

The STM32F4 microcontroller is loaded with the firmware or code via a programmer using the Serial Wire Debugging (SWD) protocol. The STM32F4 microcontroller can also be programmed via the Joint Test Action Group debugging protocol however for this research project only the SWD interface was sufficient to load the firmware onto the microcontroller.

An alternative way of programming the STM32F4 microcontroller is to load the firmware code with the Device Firmware Update (DFU) Bootloader. The DFU Bootloader is accessed by pressing and holding SW2 during power-up or after resetting the microcontroller with SW1. Once the bootloader is activated the microcontroller can be programmed directly with DfuSe USB device firmware upgrade application[56] freely available from STMicroelectronics. The only drawback is that the `.hex` or `.bin` file extension created by the compiler needs to be converted into a `.dfu` file extension in order to be loaded onto the microcontroller with the DfuSe application.

5.2 CMSIS-DSP Software Library

The Cortex Microcontroller Software Interface Standard (CMSIS) is a hardware abstraction layer independent of vendor for the Cortex-M series of processors. Several silicon and software vendors share a simple software interface between the processor and the peripherals enabling a reduced learning curve between multiple vendors software and hardware components[57].

[3] `http://www.emide.org/`
[4] `http://www.codeblocks.org/`
[5] `http://www.emblocks.org/`

The CMSIS consists of several components however only the DSP Library collection (CMSIS-DSP) consisting of over 60 functions for both single precision floating point (32-bit) and fix-point (fractional q7, q15, q31) is utilised by the real-time audio analyser. The CMSIS-DSP library located within the `arm_cortexM4lf_math.lib` features the combined Radix-8 Decimation in Frequency (DIF) Complex Fast Fourier Transform (CFFT) floating point processing function, namely `arm_cfft_f32()`[57]. The developed firmware code executed by the implemented real-time audio analyser utilises the `arm_cfft_f32()` function for the implementation of the simultaneous real-valued FFT, discussed in section 2.4.

5.3 The Real-Time Audio Analyser Files

The complete firmware code executed by the STM32F4 microcontroller consists of several developed files and folders illustrated in Figure 5.1 as a directory tree.

```
Project/
├── Target/
│   ├── debug_leds
│   │   ├── debug_leds.c
│   │   └── debug_leds.h
│   ├── delay
│   │   ├── delay.c
│   │   └── delay.h
│   ├── DSP
│   │   ├── DSP.c
│   │   └── DSP.h
│   ├── fonts
│   │   ├── fonts.c
│   │   └── fonts.h
│   ├── LCD_TFT
│   │   ├── LCD_TFT.c
│   │   └── LCD_TFT.h
│   ├── main
│   │   ├── main.c
│   │   └── main.h
│   ├── RTAA
│   │   ├── RTAA.c
│   │   └── RTAA.h
│   └── Touch
│       ├── Touch.c
│       └── Touch.h
├── CMSIS/
│   └── arm_cortexM4lf_math.lib
└── Device/
    ├── startup_stm32f429xx.s
    └── system_stm32f4xx.c
```

Figure 5.1: Real-Time Audio Analyser Firmware Project Directory

It should be noted that all the files in the Target folder illustrated in Figure 5.1 contain the fundamental C code written without the aid of code generators or code wizards. The firmware was written from complete scratch over many months by the author with the aid only the manufacturers reference and datasheet manuals for the STM32F4 microcontroller.

5.4 The Real-Time Audio Analyser Algorithm

The complete flowchart for the developed firmware code executed by the real-time audio analyser is given in Figure 5.2.

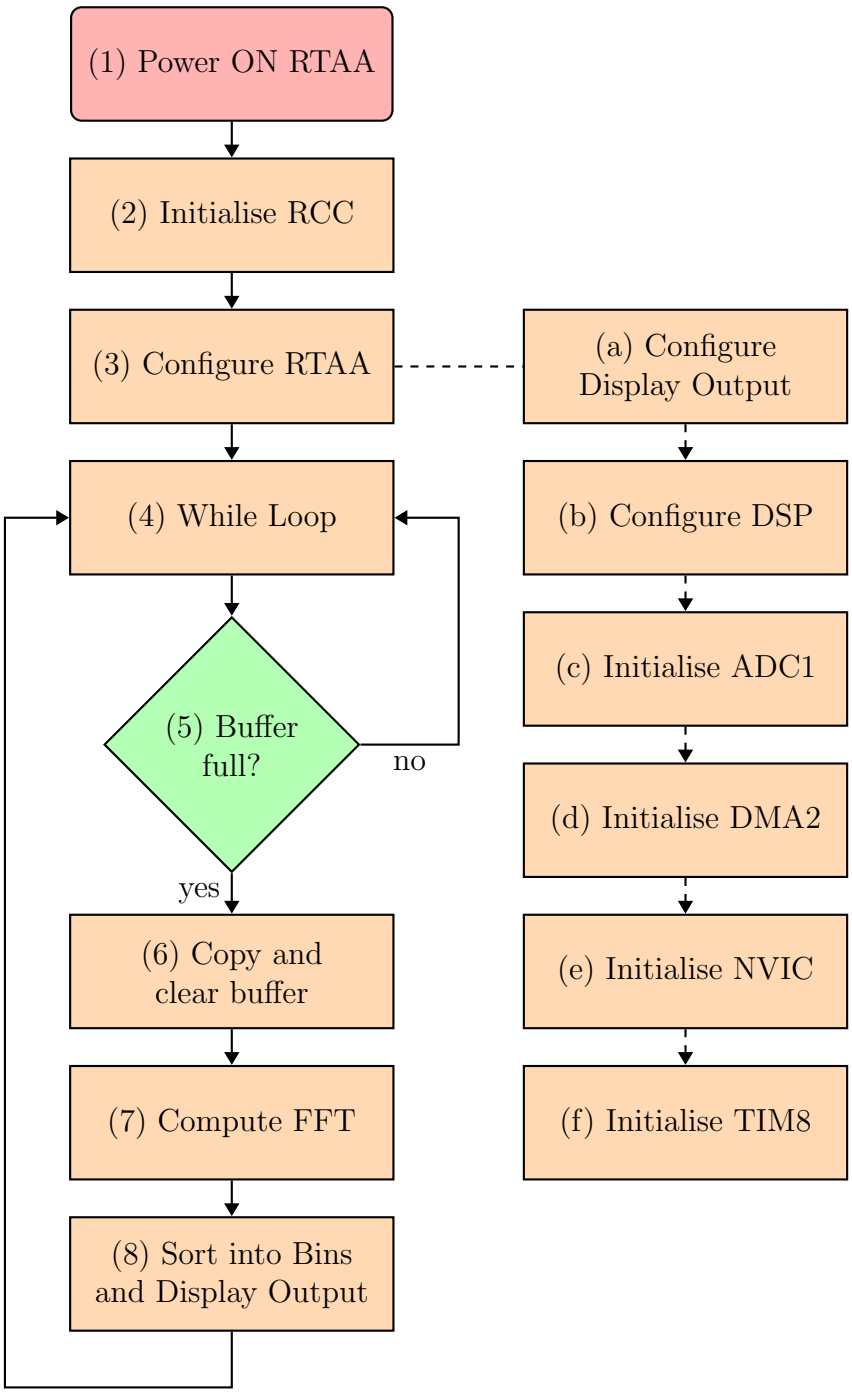


Figure 5.2: Real-Time Audio Analyser Firmware Flow Chart

5.4.1 Powering up the Real-Time Audio Analyser

Upon applying power to the real-time audio analyser, the microcontroller begins to execute the developed firmware routines contained in the project folder. Initially the stack pointer, program counter, and vector interrupt/exception routine addresses are assigned correctly to the STM32F429 microcontroller within the `startup_stem32f429xx.s` file.

5.4.2 Initialise RCC

The STM32F429 microcontroller configures the Reset and Clock Controller (RCC), in particular setting the external clock source and configuring the various system clock frequencies according to Table 4.2 within the `system_stm32f4xx.c` file.

5.4.3 Configure the Real-Time Audio Analyser

The real-time audio analyser setup and configuration routines are located in the `RTAA.c` and `RTAA.h` files within the project folder. Once the system clocks have been correctly configured, the real-time audio analyser is initialised by calling the `RTAA_Init()` function within the `main.c` file.

The `RTAA_Init()` function is the heart of the real-time audio analyser firmware and is located within the `RTAA.c` file. The `RTAA_Init()` function performs six main tasks, namely:

- Initialise the Display Output Device
  - LCD-TFT with resistive touch controller
  - Video Graphics Array (VGA) display
- Configure the Digital Signal Processing (DSP) Routines
- Configure the Analogue-to-Digital Converter (ADC1) Routines
- Configure the Direct Memory Access (DMA2) Routines
- Configure the Nested Vectored Interrupt Controller (NVIC) Routines
- Configure the Timer/Counter (TIM8) Routines

5.4.3.1 Configure Display Output

It must be noted that at the submission of this thesis for evaluation, the developed firmware for the implemented real-time audio analyser only interfaces with the 3.2" TFT-LCD output display option. The use of the resistive touch screen is also not a feature on the developed firmware for the implemented real-time audio analyser. That being said, the alternative display output, namely the standard 640 x 480 VGA, as well as the resistive touch screen have been successfully interfaced with the designed hardware. However, the resistive touch screen and

VGA display output were not integrated into the submitted developed firmware. Therefore, only the submitted developed firmware for the real-time audio analyser as it stands will be discussed.

The LCD-TFT display output configuration routines are located in the `LCD_TFT.c` and `LCD_TFT.h` files within the project folder. The configuration code for the 3.2" TFT-LCD module with SSD1289 LCD controller is located in `LCD_Init(uint16_t orient)` function within the `LCD_TFT.c` file and is given in Listing 5.1.

```
1 void LCD_Init(uint16_t orient){
    LCD_GPIO_Output();
3   LCD_GPIO_Ctrl();
    FMC_Init();
5   LCD_Setup(orient);
  }
```

Listing 5.1: TFT-LCD configuration code

The SSD1289 LCD controller uses the Flexible Memory Controller (FMC) peripheral to communicate data and instructions between the 3.2" TFT-LCD and the STM32F4 microcontroller. The FMC data communication pins are configured by the General Purpose Input/Output (GPIO) controller in the `LCD_GPIO_Output()` function within the `LCD_TFT.c` file and is given in Listing 5.2.

```
void LCD_GPIO_Output(void){
  // Configure Port D 14-15; 0-1 Data/Cmd GPIOs [DB0-3]
  RCC->AHB1ENR |= RCC_AHB1ENR_GPIODEN;     // GPIOD Periph clock enable
  // Configure the GPIO's to Alternate Function Mode and Configure FMC as Alternate Function
  GPIOD->MODER |= GPIO_MODER_MODER14_1;    // [DB0] - [PD14] - [FMC_D14]
  GPIOD->AFR[1] |= 0x0C000000;
  GPIOD->MODER |= GPIO_MODER_MODER15_1;    // [DB1] - [PD15] - [FMC_D15]
  GPIOD->AFR[1] |= 0xC0000000;
  GPIOD->MODER |= GPIO_MODER_MODER0_1;     // [DB2] - [PD0] - [FMC_D0]
  GPIOD->AFR[0] |= 0x0000000C;
  GPIOD->MODER |= GPIO_MODER_MODER1_1;     // [DB3] - [PD1] - [FMC_D1]
  GPIOD->AFR[0] |= 0x000000C0;
  // Configure Port E 7-15 Data/Cmd GPIOs [DB4-12]
  RCC->AHB1ENR |= RCC_AHB1ENR_GPIOEEN;     // GPIOE Periph clock enable
  // Configure the GPIO's to Alternate Function Mode and Configure FMC as Alternate Function
  GPIOE->MODER |= GPIO_MODER_MODER7_1;     // [DB4] - [PE7] - [FMC_D4]
  GPIOE->AFR[0] |= 0xC0000000;
  GPIOE->MODER |= GPIO_MODER_MODER8_1;     // [DB5] - [PE8] - [FMC_D5]
  GPIOE->AFR[1] |= 0x0000000C;
  GPIOE->MODER |= GPIO_MODER_MODER9_1;     // [DB6] - [PE9] - [FMC_D6]
  GPIOE->AFR[1] |= 0x000000C0;
  GPIOE->MODER |= GPIO_MODER_MODER10_1;    // [DB7] - [PE10] - [FMC_D7]
  GPIOE->AFR[1] |= 0x00000C00;
  GPIOE->MODER |= GPIO_MODER_MODER11_1;    // [DB8] - [PE11] - [FMC_D8]
  GPIOE->AFR[1] |= 0x0000C000;
  GPIOE->MODER |= GPIO_MODER_MODER12_1;    // [DB9] - [PE12] - [FMC_D9]
  GPIOE->AFR[1] |= 0x000C0000;
  GPIOE->MODER |= GPIO_MODER_MODER13_1;    // [DB10] - [PE13] - [FMC_D10]
  GPIOE->AFR[1] |= 0x00C00000;
  GPIOE->MODER |= GPIO_MODER_MODER14_1;    // [DB11] - [PE14] - [FMC_D11]
  GPIOE->AFR[1] |= 0x0C000000;
  GPIOE->MODER |= GPIO_MODER_MODER15_1;    // [DB12] - [PE15] - [FMC_D12]
  GPIOE->AFR[1] |= 0xC0000000;
  // Configure Port D 8-10 Data/Cmd GPIOs [DB13-15]
  // Configure the GPIO's to Alternate Function Mode and Configure FMC as Alternate Function
  GPIOD->MODER |= GPIO_MODER_MODER8_1;     // [DB13] - [PD8] - [FMC_D13]
  GPIOD->AFR[1] |= 0x0000000C;
  GPIOD->MODER |= GPIO_MODER_MODER9_1;     // [DB14] - [PD9] - [FMC_D14]
  GPIOD->AFR[1] |= 0x000000C0;
  GPIOD->MODER |= GPIO_MODER_MODER10_1;    // [DB15] - [PD10] - [FMC_D15]
  GPIOD->AFR[1] |= 0x00000C00;
}
```

Listing 5.2: LCD GPIO Data configuration code

The FMC control communication pins are configured by the General Purpose Input/Output (GPIO) controller in the `LCD_GPIO_Ctrl()` function within the `LCD_TFT.c` file and is given in Listing 5.3.

```
void LCD_GPIO_Ctrl(void){
  // Configure LCD_TFT controls GPIOs
  // Configure the GPIO's to Alternate Function Mode and Configure FMC as Alternate Function
  GPIOD->MODER |= GPIO_MODER_MODER7_1;     // [CS] - [PD7] - [FMC_NE1]
  GPIOD->AFR[0] |= 0xC0000000;
  GPIOD->MODER |= GPIO_MODER_MODER4_1;     // [RD] - [PD4] - [FMC_NOE]
  GPIOD->AFR[0] |= 0x000C0000;
  GPIOD->MODER |= GPIO_MODER_MODER5_1;     // [WR] - [PD5] - [FMC_NWE]
  GPIOD->AFR[0] |= 0x00C00000;
  GPIOD->MODER |= GPIO_MODER_MODER11_1;    // [RS] - [PD11] - [FMC_A16]
  GPIOD->AFR[1] |= 0x0000C000;
  GPIOD->MODER |= GPIO_MODER_MODER12_0;    // [RST] - [PD12] - [GPIO]
}
```

Listing 5.3: LCD GPIO Control configuration code

The FMC configuration code is located in the `FMC_Init(void)` function within the `LCD_TFT.c` file and is given in Listing 5.4. Before the peripheral is configured, the peripheral clock for the

FMC must be enabled within the Reset and Clock Controller (RCC). The Memory Bank 1 of the FMC module is configured to interface with the SSD1289 LCD controller.

```
void FMC_Init(void){
  RCC->AHB3ENR |= RCC_AHB3ENR_FMCEN;      // FMC Periph clock enable
  FMC_Bank1->BTCR[0] &= ~0x0000000F;      // Disable Memory Bank 1; Disable Data/Adress MUX;
    Memory Type: SRAM (Default @ Reset 16-bit width)
  FMC_Bank1->BTCR[0] &= ~0x00002000;      // Disbale Wait Enable Bit
  FMC_Bank1->BTCR[0] &= ~0x00000040;      // Disable NOR Flash Memory Access
  FMC_Bank1->BTCR[0] |= FMC_BCR1_MBKEN; // Enable Memory Bank 1
}
```

Listing 5.4: FMC configuration code

The SSD1289 LCD controller configuration code is located in the `LCD_Setup(uint16_t lcd_orient)` function within the `LCD_TFT.c` file and is given in Listing 5.5. The `uint16_t lcd_orient` parameter allows the orientation of the displayed screen to be altered in future revisions of the developed firmware for the real-time audio analyser. The default screen orientation for the real-time analyser is landscape.

```
void LCD_Setup(uint16_t lcd_orient){
  LCD_Reset(); // Reset the LCD controller / screen
  Write_REG_Data(0x00, 0x0001);    // [Oscill. Start] 1 = on, 0 = off
  Write_REG_Data(0x03, 0xA8A4);    // [Pwr. Ctrl. 1]
  Write_REG_Data(0x0C, 0x0000);    // [Pwr. Ctrl. 2]
  Write_REG_Data(0x0D, 0x080C);    // [Pwr. Ctrl. 3]
  Write_REG_Data(0x0E, 0x2B00);    // [Pwr. Ctrl. 4]
  Write_REG_Data(0x1E, 0x00B7);    // [Pwr. Ctrl. 5]
  Write_REG_Data(0x02, 0x0600);    // [Drive Waveform Ctrl.]
  Write_REG_Data(0x10, 0x0000);    // [Sleep Mode] 0 = exit, 1 = enter
  switch(lcd_orient){
    case PORT:
      Write_REG_Data(0x01, 0x2B3F); //delay_ms(1); // [Driver Output Ctrl]
      Write_REG_Data(0x11, 0x6830); //delay_ms(1); // [Entry Mode]
      break;
    case PORT_REV:
      Write_REG_Data(0x01, 0x693F); //delay_ms(1); // [Driver Output Ctrl]
      Write_REG_Data(0x11, 0x6830); //delay_ms(1); // [Entry Mode]
      break;
    case LANDSCAPE:
      Write_REG_Data(0x01, 0x693F); //delay_ms(1); // [Driver Output Ctrl]
      Write_REG_Data(0x11, 0x6838); //delay_ms(1); // [Entry Mode]
      break;
    case LANDSCAPE_REV:
      Write_REG_Data(0x01, 0x2B3F); //delay_ms(1); // [Driver Output Ctrl]
      Write_REG_Data(0x11, 0x6838); //delay_ms(1); // [Entry Mode]
      break;
  }
  Write_REG_Data(0x05, 0x0000);    // [Comp. Reg.] {POR}
  Write_REG_Data(0x06, 0x0000);    // [Comp. Reg.] {POR}
  Write_REG_Data(0x16, 0xEF1C);    // [Horz. Porch] {POR}
  Write_REG_Data(0x17, 0x0003);    // [Vert. Porch] {}
  Write_REG_Data(0x07, 0x0233);    // [Disp. Ctrl.] {}
  Write_REG_Data(0x0B, 0x0000);    // [Frame Cyc. Ctrl.] {}
  Write_REG_Data(0x0F, 0x0000);    // [Gate Scan Pos.] {POR}
  Write_REG_Data(0x41, 0x0000);    // [Vert. Scroll Ctrl.] {POR}
  Write_REG_Data(0x42, 0x0000);    // [Vert. Scroll Ctrl.] {POR}
  Write_REG_Data(0x48, 0x0000);    // [1st Scr. Drive Pos.] {POR}
  Write_REG_Data(0x49, 0x013F);    // [1st Scr. Drive Pos.] {POR}
  Write_REG_Data(0x4A, 0x0000);    // [2nd Scr. Drive Pos.] {POR}
  Write_REG_Data(0x4B, 0x0000);    // [2nd Scr. Drive Pos.] {}
  Write_REG_Data(0x44, 0xEF00);    // [Horz. RAM Addr. Pos.] {POR}
  Write_REG_Data(0x45, 0x0000);    // [Vert. RAM Addr. Pos.] {POR}
  Write_REG_Data(0x46, 0x013F);    // [Vert. RAM Addr. Pos.] {POR}
  Write_REG_Data(0x30, 0x0707); // [Gamma Ctrl.]
  Write_REG_Data(0x31, 0x0204);    // [Gamma Ctrl.]
  Write_REG_Data(0x32, 0x0204);    // [Gamma Ctrl.]
  Write_REG_Data(0x33, 0x0502);    // [Gamma Ctrl.]
  Write_REG_Data(0x34, 0x0507);    // [Gamma Ctrl.]
  Write_REG_Data(0x35, 0x0204);    // [Gamma Ctrl.]
  Write_REG_Data(0x36, 0x0204);    // [Gamma Ctrl.]
  Write_REG_Data(0x37, 0x0502);    // [Gamma Ctrl.]
  Write_REG_Data(0x3A, 0x0302);    // [Gamma Ctrl.]
  Write_REG_Data(0x3B, 0x0302);    // [Gamma Ctrl.]
  Write_REG_Data(0x23, 0x0000);    // [GRAM Write MSK R&G] {POR}
  Write_REG_Data(0x24, 0x0000);    // [GRAM Write MSK B] {POR}
  Write_REG_Data(0x25, 0x8000);    // [Frame Freq. Ctrl.] {POR}
  Write_REG_Data(0x4E, 0x0000);    // [RAM Addr. Set]
  Write_REG_Data(0x4F, 0x0000);    // [RAM Addr. Set]
  LCD_Clear(0);    // Clear GRAM
}
```

Listing 5.5: SSD1289 configuration code

5.4.3.2 Configure DSP

The primary digital signal processing techniques used by the real-time audio analyser is located within the `DSP.c` file. The global variables for the `DSP.c` file are:

- `float input[2*SAMPLE_SIZE]` - real input samples from the ADC/CODEC array

- `uint32_t output[BARS]` - bars displayed on the LCD-TFT/VGA output array
- `float window[SAMPLE_SIZE]` - selected digial window function array
- `float freq_low[BARS]` - lower bandedge frequency array
- `float freq_cent[BARS]` - centre band frequency array
- `float freq_high[BARS]` - higher bandedge frequency array
- `float weight_filter[BARS]` - selected audio weighting filter array
- `float cfft_result[2*SAMPLE_SIZE]` - complex output from the CMSIS CFFT routine
- `float cfft_result_conj[2*SAMPLE_SIZE]` - complex conjugate of the CFFT output
- `float cfft_power[SAMPLE_SIZE]` - power of the CFFT output array
- `float window_sum1` - arithmetic sum of the selected digital window
- `float window_sum2` - square sum of the selected digital window

where `SAMPLE_SIZE` is the number of complex samples for the CFFT routine and `BARS` is the total number of frequency divisions displayed. Default values are `SAMPLE_SIZE` = 4096 and `BARS` = 30.

The `DSP_Init()` function located within the `DSP.c` file and given in Listing 5.6.

```
1 void DSP_Init(void){
    DSP_Freq_Pnts_Config();    // Compute the frequency axis bar divisions
3   DSP_Window_Function();     // Generate Digital Window Function
    DSP_Weighting_Function(); // Generate Audio Weighting Filter
5   DSP_FFT_Select();    // Select correct FFT from CMSIS library
}
```

Listing 5.6: DSP configuration code

The DSP routines are outlined as follows:

- Populate the frequency arrays `freq_low[]`, `freq_cent[]`, and `freq_high[]`
- Generate the selected digital windowing array `window[]`
- Calculate the selected audio weighting array `weight_filter[]`
- Select the correct sized FFT from the CMSIS DSP library

The function `DSP_Freq_Pnts_Config()` located within the `DSP.c` file and given in Listing 5.7 computes the frequency division arrays `freq_cent[]`, `freq_low[]`, and `freq_high[]` according to equations 2.18, 2.19, and 2.20 in section 2.1.5.

```
void DSP_Freq_Pnts_Config(void){
  int32_t k;    // This variable must be an integer!
  if(BINS==3){
    for(k=0;k<BARS;k++){
      freq_cent[k] = (float) ((powf(G,((float)(k-16)/(BINS))))*FREF);
      freq_low[k] = (float) ((powf(G,((float)(-1.0)/(2*BINS))))*(freq_cent[k]));
      freq_high[k] = (float) ((powf(G,((float)(+1.0/(2*BINS)))))*(freq_cent[k]));
    }
  }
  else if(BINS==1){
    for(k=0;k<BARS;k++){
      freq_cent[k] = (float) ((powf(G,((float)((k)-5)/(BINS))))*FREF);
      freq_low[k] = (float) ((powf(G,((float)(-1.0)/(2*BINS))))*(freq_cent[k]));
      freq_high[k] = (float) ((powf(G,((float)(1.0/(2*BINS)))))*(freq_cent[k]));
    }
  }
}
```

Listing 5.7: DSP frequency axis bar divisions

The function `DSP_Window_Function()` located within the `DSP.c` file and given in Listing 5.8 computes the selected digital windowing function `window[]` array according to equations 2.37, 2.38, 2.39, 2.40, 2.41, and 2.42 in section 2.3.1.

```
void DSP_Window_Function(){
  uint32_t n;
  switch(DIGITAL_WINDOW){
    case 1:   // Rectangular Digital Window Function
      for(n=0;n<SAMPLE_SIZE;n++){
        window[n] = 1.0;
      }
      break;
    case 2:   // Hanning Digital Window Function
      for(n=0;n<SAMPLE_SIZE;n++){
        window[n] = (float) (0.5 - 0.5*(cosf((2*PI*n)/(SAMPLE_SIZE))));
      }
      break;
    case 3:   // Hamming Digital Window Function
      for(n=0;n<SAMPLE_SIZE;n++){
        window[n] = (float) (0.54 - 0.46*(cosf((2*PI*n)/(SAMPLE_SIZE))));
      }
      break;
    case 4:   // Welch Digital Window Function
      for(n=0;n<SAMPLE_SIZE;n++){
        window[n] = (float) (1.0 - powf(((n-(0.5*SAMPLE_SIZE))/(0.5*SAMPLE_SIZE)),2.0));
      }
      break;
    case 5:   // Bartlett Digital Window Function
      for(n=0;n<SAMPLE_SIZE;n++){
        window[n] = (float) (1.0 - abs((n-(0.5*SAMPLE_SIZE))/(0.5*SAMPLE_SIZE)));
      }
      break;
    case 6:   // Blackman Digital Window Function
      for(n=0;n<SAMPLE_SIZE;n++){
        window[n] = (float) (0.42-0.50*((cosf((2*PI*n)/(SAMPLE_SIZE)))+(0.08*(cosf((4*PI*n)/(
    SAMPLE_SIZE))))));
      }
      break;
  }
  // Determine the sum of the Digital Window Function
  for(n=0;n<SAMPLE_SIZE;n++){
    window_sum1+=window[n];
    window_sum2+=(powf(window[n],2.0));
  }
}
```

Listing 5.8: DSP digital window fucntion generation

The function `DSP_Weighting_Function()` located within the `DSP.c` file and given in Listing 5.9

populates the `weight_filter[]` array with the selected audio weighting filter according to equations 2.43, 2.47, 2.51, and 2.55 in section 2.3.2.

```
void DSP_Weighting_Function(){
  uint32_t k;
  switch(WEIGHT_FILTER){
    case 1:   // A-Wieghting Filter
      for(k=0;k<BARS;k++){
        weight_filter[k]=(float) (1.9997+20.0*(log10f(((powf(12194.22,2.0)*(powf(freq_cent[k],4.0)))/((powf(freq_cent[k],2.0))+(powf(20.598997,2.0)))*((powf(freq_cent[k],2.0))+(powf(12194.22,2.0)))*(sqrtf(((powf(freq_cent[k],2.0))+(powf(107.65265,2.0)))*((powf(freq_cent[k],2.0))+(powf(737.86223,2.0)))))))));
      }
      break;
    case 2:   // B-Weighting Filter
      for(k=0;k<BARS;k++){
        weight_filter[k]=(float) (0.1696+20.0*(log10f(((powf(12194.22,2.0)*(powf(freq_cent[k],3.0)))/((powf(freq_cent[k],2.0))+(powf(20.598997,2.0)))*((powf(freq_cent[k],2.0))+(powf(12194.22,2.0)))*(sqrtf(((powf(freq_cent[k],2.0))+(powf(158.48932,2.0)))))))));
      }
      break;
    case 3:   // C-Weighting Filter
      for(k=0;k<BARS;k++){
        weight_filter[k]=(float) (0.0619+20.0*(log10f(((powf(12194.22,2.0)*(powf(freq_cent[k],2.0)))/((powf(freq_cent[k],2.0))+(powf(20.598997,2.0)))*((powf(freq_cent[k],2.0))+(powf(12194.22,2.0)))))));
      }
      break;
    case 4:   // Z-Weighting Filter (Flat)
      for(k=0;k<BARS;k++){
        weight_filter[k]=(float) 0.0;
      }
      break;
  }
}
```

Listing 5.9: DSP audio weighting filter generation

5.4.3.3 Configure ADC

The internal on-chip Analogue-to-Digital Converter (ADC), in particular the ADC1, provides the real-time audio analyser with a digital representation of the received audio signal via the selected input source. The ADC1 configuration code is located in the `ADC1_Init()` function within the `RTAA.c` file and is given in Listing 5.10.

Before the peripheral is configured, the peripheral clock for ADC1 must be enabled within the Reset and Clock Controller (RCC). Moreover, a clock prescaler divides the ADC clock by a factor of 4 resulting in ADC_{CLK} = 24MHz. The ADC is configured to allow access via the Direct Memory Access (DMA) Controller and data will be transferred directly from the ADC to a memory location as specified by the DMA configuration code in Listing 5.11. The ADC1 channel input 10 is triggered to begin a conversion on the rising edge of the advanced timer/counter, namely TIM8.

```
void ADC1_Init(){ // GPIO Port C Pin 0 Setup (Input) (ADC1_IN10)
  RCC->AHB1ENR |= RCC_AHB1ENR_GPIOCEN;  // GPIOC Periph clock enable
  GPIOC->MODER |= GPIO_MODER_MODER0;    // Configure PC0 in analoge mode (input)
  RCC->APB2ENR |= RCC_APB2ENR_ADC1EN;   // ADC1 Clock enable
  ADC->CCR |= ADC_CCR_ADCPRE_0;   // ADC clock prescaler set to 1 (APB2/4=24MHz)
  ADC1->CR2 |= ADC_CR2_DMA | ADC_CR2_DDS;   // DMA Access from ADC to memory
  ADC1->CR2 |= ADC_CR2_EXTEN_0;   // External trigger enable (Rising Edge from TIM8)
  ADC1->CR2 |= 0x0E000000;    // External event source TIM8 TRGO event
  ADC1->SQR3 |= 0x0000000A;   // ADC1 - input 10
  ADC1->CR2 |= ADC_CR2_ADON;    // ADC1 ON
}
```

Listing 5.10: ADC1 configuration code

5.4.3.4 Configure DMA

The Direct Memory Access (DMA) controller, in particular the DMA2 controller, controls the flow of data between the analogue-to-digital converter (ADC1) and the `&sample` memory address location. The DMA2 controller configuration code is located in the `DMA2_Init()` function within the `RTAA.c` file and is given in Listing 5.11.

The DMA2 peripheral clock must be enabled within the Reset and Clock Controller (RCC) and the DMA2 controller must be disabled prior to configuring the DMA controller. The DMA2 controller is configured to transfer 32-bit sized data from the ADC1 channel 10 using DMA2 stream 0 to the `&sample` memory address location. Now recall that the `&sample` memory address location is the beginning of the `sample[2*SAMPLE_SIZE]` array, declared in `main.c`. Hence, the DMA2 controller continues to transfer data until a total of `2*SAMPLE_SIZE` transfers have completed between the ADC1 channel 10 and the`&sample` memory address location. An interrupt flag (DMA Transfer Complete) is activated upon the completion of transferring all `2*SAMPLE_SIZE` 32-bit sized data samples between the ADC1 channel 10 and the `&sample` memory address location. Finally, the DMA Stream 0 is enabled and the DMA2 module can maintain and handle the direct memory access transfers between the ADC1 peripheral and the memory address `&sample`.

```
void DMA2_Init(){
  // CONFIGURE DMA DATA TRANSFER WHEN ADC CONVERSION IS COMPLETE [DMA2; Channel 0; Stream 0]
  RCC->AHB1ENR |= RCC_AHB1ENR_DMA2EN;    // Enable DMA2 Clocks
  DMA2_Stream0->CR &= ~DMA_SxCR_EN;      // Ensure that DMA disabled
  DMA2_Stream0->CR |= (DMA_SxCR_PL | DMA_SxCR_MSIZE_1 | DMA_SxCR_PSIZE_1 | DMA_SxCR_CIRC |
    DMA_SxCR_MINC);    // Channel 0 Selected; Peri. to Mem. Trans.; Priority Very High; Mem.
    Size; Peri. Size;
  DMA2_Stream0->M0AR = (uint32_t) (&sample);  //  Memory Address (DESTINATION)
  DMA2_Stream0->PAR = (ADC1_BASE | 0x0000004C);   // Peripheral Address (SOURCE)
  DMA2_Stream0->NDTR = 2*SAMPLE_SIZE;    //  Data Size (TRANSFERED VIA DMA2)
  DMA2_Stream0->CR |= DMA_SxCR_TCIE;     // DMA Transfer Complete Interrupt Enabled
  DMA2_Stream0->CR |= DMA_SxCR_EN;     // Enable DMA2
}
```

Listing 5.11: DMA configuration code

5.4.3.5 Configure NVIC

The Nested Vectored Interrupt Controller (NVIC) overviews the flow of data between the analogue-to-digital converter (ADC) and the `&sample` memory address location with the use of

the Direct Memory Access (DMA) controller. The NVIC configuration code is located in the `NVIC_Init()` function within the `RTAA.c` file and is given in Listing 5.12. The NVIC is configured to monitor the DMA2 Stream 0, namely `DMA2_Stream0_IRQn`, with the highest priority by passing 0 to the second argument of the `NVIC_SetPriority()` function. The NVIC pauses the current CPU operation upon receiving an interrupt request and executes the interrupt service routine, namely `DMA2_Stream0_IRQHandler()`.

```
void NVIC_Init(){ // DMA Transfer Complete Interrupt
  NVIC_SetPriority(DMA2_Stream0_IRQn, 0);    //  Set DMA Transfer Complete IRQ priority highest
  NVIC_EnableIRQ(DMA2_Stream0_IRQn);      // Enable IRQ for ADC in NVIC
}
```

Listing 5.12: NVIC configuration code

It must be noted that the NVIC only detects a general DMA2 interrupt request and cannot determine the exact nature of the interrupt request. Therefore, the nature of the interrupt needs to be probed before executing the DMA interrupt routine. The DMA2 interrupt routine is located in the `DMA2_Stream0_IRQHandler()` function within the `RTAA.c` file and is given in Listing 5.13. The interrupt routine first acknowledges and clears the DMA transfer complete interrupt flag before copying the `sample[2*SAMPLE_SIZE]` array to the `input_cmplx[2*SAMPLE_SIZE]` buffer array.

```
void DMA2_Stream0_IRQHandler(void){
  uint32_t tmp_SR = DMA2->LISR;
  if(tmp_SR & DMA_LISR_TCIF0){     // DMA Data Transfer Complete
    DMA2->LIFCR |= DMA_LIFCR_CTCIF0;     // Clear DMA Transfer Complete Interrupt Flag
    new_sample = 1;    // Enable new sample flag
  }
}
```

Listing 5.13: DMA interrupt routine code

5.4.3.6 Configure Timer/Counter (TIM8)

The advanced timer/counter, in particular Timer 8 (TIM8), provides the real-time audio analyser with a sampling rate for the analogue-to-digital conversion of the received audio signal via the selected input source. The TIM8 configuration code is located in the `TIM8_Init()` function within the `RTAA.c` file and is given in Listing 5.14.

```
void TIM8_Init(){ // Timer 8 Setup
  RCC->APB2ENR |= RCC_APB2ENR_TIM8EN;    // Timer 8 Clock enable
  TIM8->ARR = ((SystemCoreClock/2)/SAMPLE_RATE);
  TIM8->CR1 |= TIM_CR1_ARPE;     // Auto relaod preload enabled
  TIM8->CR2 |= TIM_CR2_MMS_1;    // Select TRGO to be update event (UE)
  TIM8->DIER |= TIM_DIER_UDE;    // Update DMA Request Enabled
  TIM8->CR1 |= TIM_CR1_CEN;    // Enable counting
}
```

Listing 5.14: TIM8 configuration code

The peripheral clock for TIM8 must be enabled within the Reset and Clock Controller (RCC) prior to configuring the TIM8 module. The auto-reload register (ARR) is a value that depends

on the selected sampling rate and the system clock. The default system clock frequency and sampling rate, SYSCLK = 192MHz and `SAMPLE_RATE` = 64kHz. Hence, ARR = 1500. The auto-reload value is preloaded when the timer resets and allows the timer to continuously count up from zero until the auto-reload register value at which point the timer will reset to zero and counting begins again. Moreover, a timer reset triggers an event update on the trigger output pin, TRIGO, and also requests a DMA update. Finally, the sampling timer TIM8 is enabled and the real-time audio analyser can begin sampling the input audio signal.

5.4.4 Endless While Loop and DSP Processing

The main firmware routines executed by the STM32F429 microcontroller are located in the `main.c` file. The `while(1)` or endless loop, located in the `main()` function within the `main.c` file and given in Listing 5.15, is the primary location for the program counter when the developed firmware is successfully executed by the ST32F429 microcontroller.

```
while(1){
  if((new_sample)){
    for(i=0;i<2*SAMPLE_SIZE;i++){
      input[i]=((((float) (sample[i]))-2048)/2048); // Adjust input voltage range [-1;1]
    }
    new_sample = 0;    // Disable new sample flag
    DSP_Process_Data_CMSIS_CFFT();
    }
  }
```

Listing 5.15: Main Endless While Loop

The global flag variable `new_sample` is polled within the endless while loop and indicates the successful transfer of `2*SAMPLE_SIZE` samples between the ADC1 channel 10 using DMA2 stream 0 to the `&sample` memory address location. The samples are adjusted and scaled to a normalized range, namely $[-1; 1]$, and the global `new_sample` flag is reset.

The `DSP_Process_Data_CMSIS_CFFT()` function located within the `DSP.c` file is the core digital signal processing routine function. The adjusted input signal, to the range $[-1; 1]$, is split according to the simultaneous real-valued FFT outline in section 2.4. The selected digital window function (Rectangular, Hanning, Hamming, Welch, Bartlett, or Blackman) is applied to each real-valued input sample before forming the single complex-valued input sample according to equation 2.59. The `arm_cfft_f32()` function located within the `arm_cortexM4lf_math.lib` file and provided by the CMSIS DSP-Library performs an FFT using `SAMPLE_SIZE` data points from the complex-valued windowed input sample. The result of the CFFT, namely array `cfft_result[]`, is adjusted according to equation 2.79 and stored in the array `cfft_power[]`.

The final audio spectrum displayed on the LCD-TFT is determined by sequentially traversing through the `cfft_power[]` array. Each decibel power level in the array `cfft_power[]` is located at a fixed frequency position and can be determined according to equation 2.35. The temporary buffer variable `buf` accumulates the decibel power levels located within each bandwidth according to equations 2.19, and 2.20, and summarised in Table 2.1.

5.5 Some Final Words

The C code presented in this chapter discusses the firmware algorithm executed by the designed real-time audio analyser. It would be impossible to discuss every last line of code presented in Figure 5.1 and hence only the fundamentally important lines of code are presented and discussed. Should any reader of this paper require to complete directory, please contact the author or supervisor.

The applied audio signal is digitized by the internal ADC with a fixed sampling rate generated by the advanced timer. The DMA controller oversees the transfer of digital audio samples between the ADC and a temporary memory location. The completion of data transfer between the ADC and the temporary memory location is governed by the NVIC. The digital samples are processed with the FFT algorithm discussed in section 2.4 and the FMC dictates the output audio spectrum to the LCD screen.

CHAPTER 6

Testing and Discussion

There are many possible testing scenarios, testing standards and commercial devices that the implemented real-time audio analyser could be tested against. Since the developed device was only under investigation in an academic environment and not a commercial setting, the designed instrument will not be tested against all possible audio analysis solutions. Moreover, in a commercial setting the development of an electronic device such as a real-time audio analyser would possibly follow a technical specification guideline or specification standard. Two such examples are the American National Standard published by the American National Specification Institute (ANSI), and the Indian Standard published by the Bureau of Indian Standards (BIS).

6.1 Performance Comparison to Commercial Solutions

The performance of the designed, developed and implemented real-time audio analyser was directly compared to the commercially available DSP30 real-time audio analyser, manufactured by Gold Line[16].

The audio analysis settings for the DSP30 audio analysing device consists of a flat audio weighting filter, a peak analysis, and a fast time-weighting. The real-time audio analyser made use of the Rectangular digital data windowing function and a flat audio weighting filter. It must be noted that the microphones used by each audio analysing device were not identical and some discrepancies between the outputted audio spectrum will result. Ideally both audio analysing devices should use identical audio microphones however the currently designed hardware and developed firmware do not provide the required 48V phantom power for the external condenser microphone used by the DSP30 audio analyser. The implemented real-time audio analyser was tested using the internal ADMP401 single-ended analogue MEMS microphone[14] and the

commercially available DSP30 audio analyser was tested using the 48V phantom powered Klark Teknik 6051 condenser microphone[15].

Table 6.1 shows a comparison between the technical specifications of the two microphones.

	Analog Devices ADMP401	**Klark Teknik 6051**
Frequency Response	Flat from 100Hz to 15kHz	Flat to 15kHz
Sensitivity (per μBar @ 1kHz)	0.794mV	0.5mV
Dynamic Range (SPL)	32 - 120dB	20 - 130dB
Capsule	0.25mm MEMS	0.25" Electret Condenser
Type	Omnidirectional	Omnidirectional
Power Required	1.5V - 3.3V	48V Phantom Power

Table 6.1: Comparison between the ADMP401 and 6051 microphones[14][15]

The sensitivity of the two microphones are not expressed in the same units in their respective datasheets/user manuals. However, the sensitivity for the ADMP401 microphone can be converted to the same units as the 6051 microphone through some simple logarithmic and algebraic manipulations. The complete calculations are discussed in section 2.5.5.

The sound system utilized to output the audio test signals was the 501a Reveal[58] active two-way loud-speaker manufactured by Tannoy Ltd. The 501a Reveal active loud-speaker features an XLR balanced input and a 1/4" unbalanced jack inputs with an input sensitivity of 0.775V$_{RMS}$ for full output. Moreover, the active two-way loud-speaker with a cross-over frequency of 2.3kHz is capable of outputting audio with the built-in power amplifier with a total of LF 40W$_{RMS}$ and HF 20W$_{RMS}$ for a frequency response of 64Hz - 30kHz[58]. The 501a Reveal active loud-speaker provided a suitable point source to output the test tones used to compare the analysed audio spectrum from the implemented real-time audio analyser against a commercially available audio analyser, namely the DSP30.

An overview of the audio test system is given in Figure 6.1. The Device Under Test (DUT) represents the audio analyser exposed to the selected audio test signal and the microphone is placed approximately 5ft - 6ft (1.5m - 1.8m) from the audio test signal source[20].

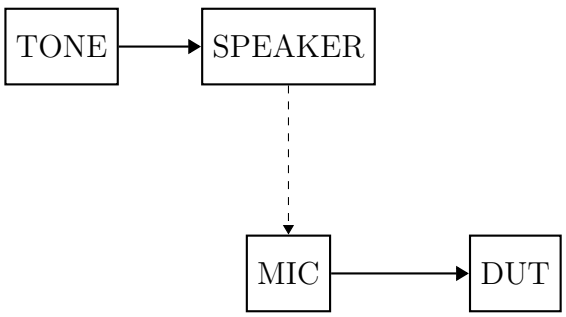


Figure 6.1: Audio Test System Overview

The two real-time audio analysers were tested using two different audio test signals, namely a pure audio tone signal and a pink noise audio signal. The pure audio test tone, also discussed

in section 2.1.1, is a single frequency signal with a stable sound pressure level. A pure audio test tone can be identified by an audio analyser as a single peak in the corresponding frequency bin attributed to the steady frequency of the pure test tone.

The pink noise audio test signal, also discussed in section 2.1.3, is a non-periodic audio signal with constant and equal energy per octave. Pink noise can be identified by a constant-Q audio analyser as a flat level for all frequency bins in the measured frequency range. It must be noted again that the signal chain can alter the output spectrum displayed by the audio analyser. This includes the frequency response of the microphone, the frequency response of the loud speaker that outputs the test signals as well as the operating acoustic environment.

The designed and implemented real-time audio analyser was configured to use a rectangular digital window, a flat or Z-weighting filter, sampled data at 64000Hz, with a 4096 sample sized FFT. The commercial DSP30 audio analyser was configured to use a flat or Z-weighting filter and a fast time decay or 125ms time-weighting analysis[16].

Figures 6.2, 6.3, 6.4, and 6.5 were obtained using an Apple iPhone 4S, cropped and rotated appropriately.

6.1.1 Pure Audio Tone Signal Results

The RaneGain Generator, manufactured by Rane Corporation, outputs a pure audio tone signal with the following specifications[59]:

- Frequency: 400Hz (± 10%) sine wave
- Output Level: 0dBu (± 1dB)

The 501a Reveal loud-speaker was set to output the maximum input level, namely the 400Hz oscillation tone, and the microphone of each audio analysing device was placed 1.5m away, as illustrated by Figure 6.1.

6.1.1.1 Commercial DSP30 Audio Analysis

Figure 6.2 shows the audio spectrum analysis output of a pure 400Hz tone signal measured using the commercial DSP30 audio analysing tool.

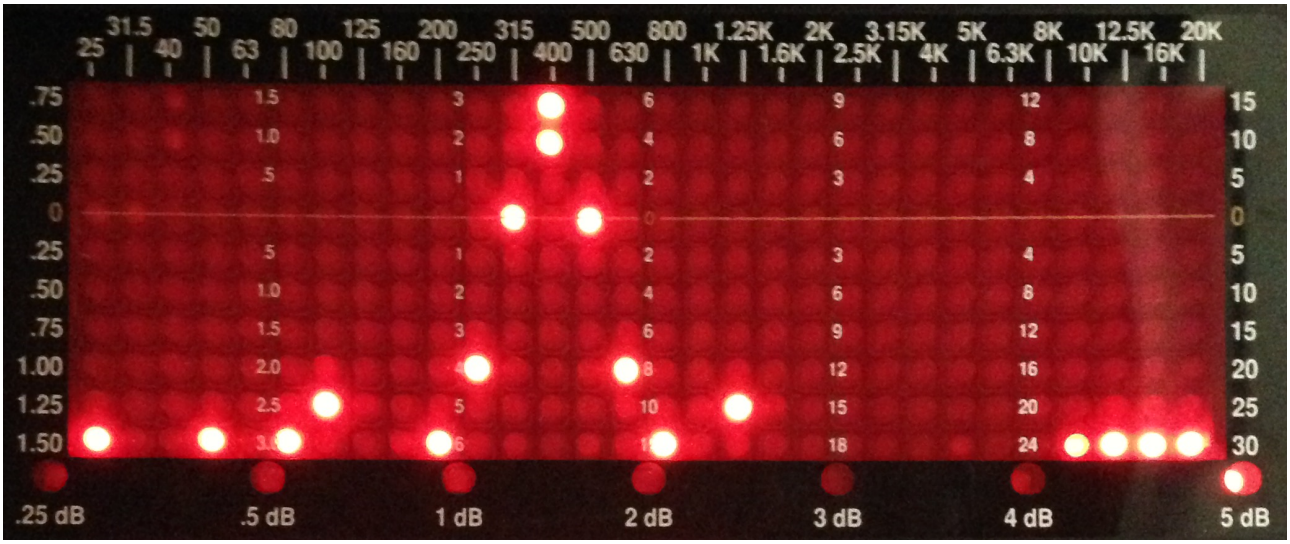


Figure 6.2: 400Hz tone measured using the DSP30 audio analyser

Note the clear increase in sound pressure level measured by the DSP30 audio analyser for the frequency bin corresponding to the pure test tone output frequency of 400Hz.

6.1.1.2 Designed and Implemented Real-Time Audio Analysis

Figure 6.3 shows the audio spectrum analysis output of a pure 400Hz tone signal measured using the designed and implemented real-time audio analyser.

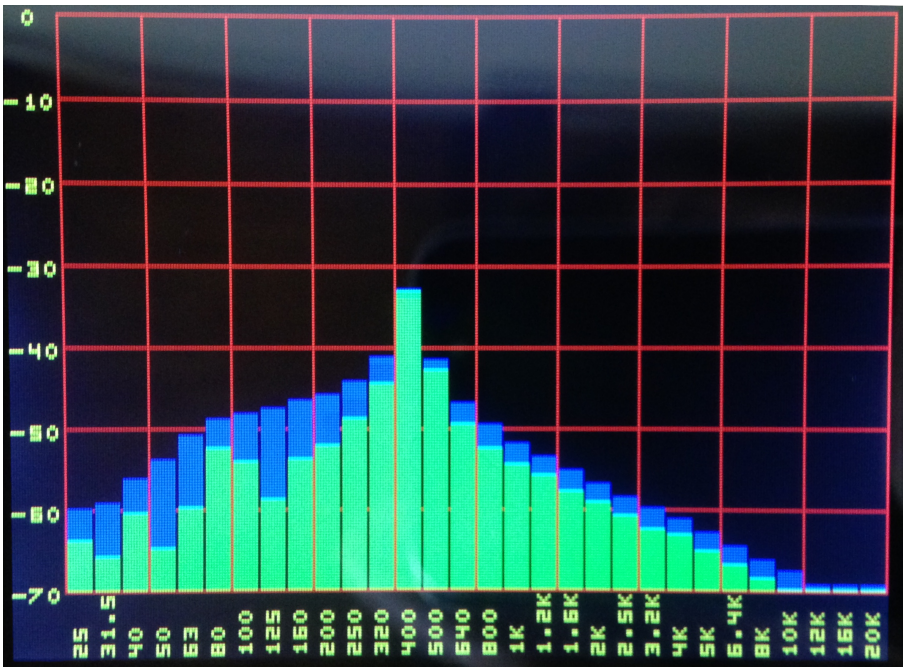


Figure 6.3: 400Hz tone measured using the real-time audio analyser

Note the clear increase in sound pressure level measured by the designed and implemented real-time audio analyser for the frequency bin corresponding to the pure test tone output frequency of 400Hz. While it might appear that the are some differences in Figures 6.2 and 6.3, both are an analysis of the same pure audio tone. The differences in the lower and upper extreme

frequencies are a result of differences in the audio signal chain associated to each audio analyser. Recall, the commercial DSP30 audio analyser uses the Klark Teknik 6051 condenser microphone while the designed and implemented real-time audio analyser uses the Analog Devices ADMP40 MEMS microphone. The difference in the frequency response of each microphone used by each audio analysing device results in a difference in the displayed audio spectrum.

6.1.2 Pink Noise Audio Signal Results

The PN2 Audio Noise Source, manufactured by Gold Line, outputs a pink noise audio test signal with the following specifications[60]:

- Frequency Range: 20Hz - 20kHz
- Output Level: Continuous Pink Noise @ 83dB - 85dB reference; 97dB SPL, 55mV

The 501a Reveal loud-speaker was set to output the maximum input test signal, namely the continuous pink noise generated by the PN2 Audio Noise, and the microphone of each audio analysing device was placed 1.5m away, as illustrated by Figure 6.1.

6.1.2.1 Commercial DSP30 Audio Analysis

Figure 6.4 shows the audio spectrum analysis output of a pink noise signal measured using the commercial DSP30 audio analysing tool.

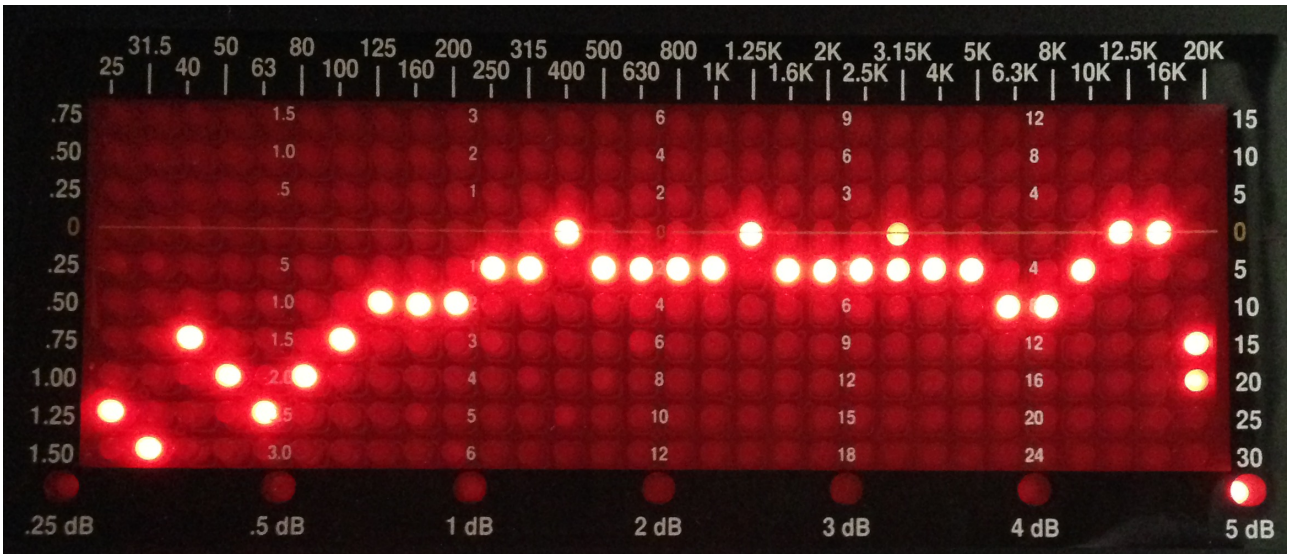


Figure 6.4: Pink noise measured using the DSP30 audio analyser

The lower frequency audio components are attenuated while the mid-frequency is flat and level. The drop in the high frequency component is a characteristic of the frequency response for the 6051 condenser microphone.

6.1.2.2 Designed and Implemented Real-Time Audio Analysis

Figure 6.5 shows the audio spectrum analysis output of a pink noise signal measured using the designed and implemented real-time audio analyser.

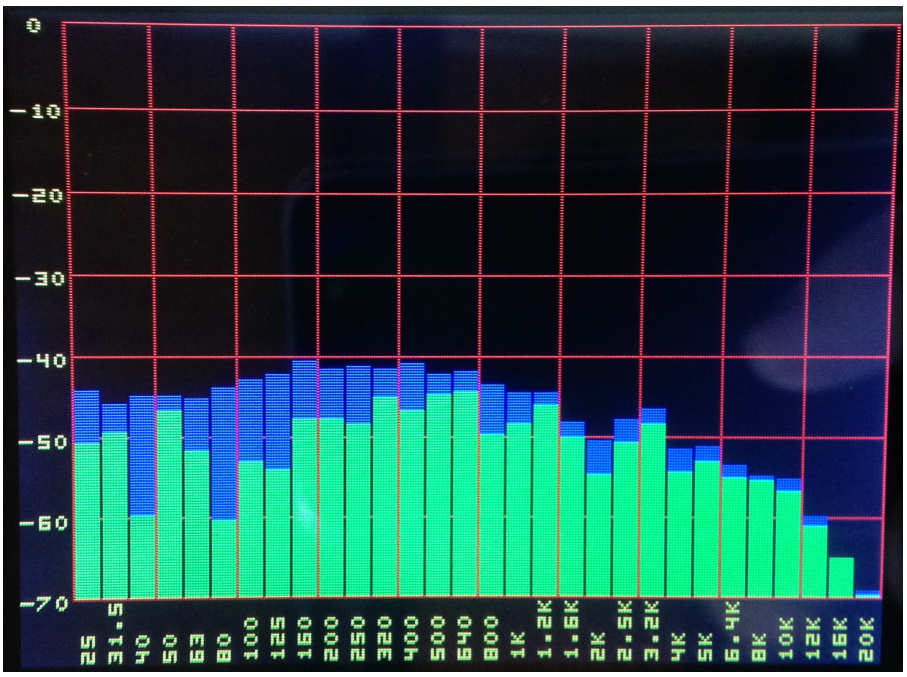


Figure 6.5: Pink noise measured with the real-time audio analyser

The higher frequency audio components of the continuous pink noise signal are poorly analysed by the real-time audio analyser and is a result of the frequency response for the ADMP401 internal microphone.

The differences in the output audio spectrum in Figure 6.4 and Figure 6.5 are a result of a difference in the audio signal chain used by each audio analysing device. The middle frequency bands displayed by each audio analysing device are relatively flat and this is a feature of signal with equal energy per octave.

6.1.3 Final Deliberation

The *Tektronix MSO 2024B Mixed Signal Oscilloscope* and the *Tektronix AFG310 Arbitrary Function Generator* were utilised during the design and development of the real-time audio analyser. Recall, the focus of the research project is to design an alternative audio analyser to a commercial solution and hence the results presented focus on the comparison. The designed, developed and implemented real-time audio analyser was able to output to the display a spectrum analysis for a periodic audio signal and a non-periodic audio signal. The slight differences in the lower and higher frequency bins have been attributed to the difference in the frequency response of the microphones used by each audio analysing device. Moreover, the size of the designed printed circuit boards and the manufacture costs presented in Appendices A were minimized as much as possible.

CHAPTER 7

Future Work

There will always be the need to add/remove minor adjustments to both the hardware and firmware of any designed and developed digital electronic device such as a real-time audio analyser. Throughout this research project the hardware underwent many changes and alterations as the components soon became overwhelmed by the sea of wires and extension proto-boards. The first printed circuit board or "prototype" of the real-time audio analyser consisted of a 100-pin STM32F429VG microcontroller with several components and traces re-soldered due to poor design. Nonetheless, the prototype provided a sufficient starting point for the development of the real-time audio analyser firmware. A list of a few clearly noticeable design faults have since been adjusted and updated in recent revisions and are given as follows:

- PCB traces should ideally travel in the shortest and straightest possible route between pads or component connections
- PCB traces should ideally be large enough to handle the required current flow
- Component names and designators should not be placed over vias or through-hole connections
- Digital traces with high switching frequencies should ideally be located away from analogue traces
- Crystal oscillator traces between the smoothing capacitors should be shorter and the crystal oscillator should be placed closer to the microcontroller.

Moreover, the hardware and firmware presented in this thesis will also undergo several revisions in the near future. A list of some of the hardware upgrades or improvements under consideration are given as follows:

- Calculate the speed of sound with an environment sensor extension board.

- Design and fabricate of a durable 3D printed housing for the real-time audio analyser.
- Replace the STM32F429 microcontroller with the new high performing STM32F7 series of microcontrollers.
- Replace the 3.2" TFT-LCD with a larger 5.0" TFT-LCD, namely the ITDB02-5.0 module with 800 x 480 resolution.

Furthermore, improvements on the developed firmware code are given as follows:

- Integrate the touch screen routines to allow the user to interact with selection menus.
- Program a selection menu allowing the user to change the selected digital data windowing filter, and audio-weighting filter.
- Completion of the Video Graphic Array (VGA) display routines.
- Implement an audio spectrogram (waterfall) plot.
- Use the 24-bit audio CODEC with professional audio signal chain.
- Increase analysed sample sizes and store filter lookup tables with the designed external memory board.
- Generate both the pure and pink noise audio test signals with the internal 12-bit DAC.

CHAPTER 8

Conclusion

In conclusion, this thesis outlines the design, development and implementation of a real-time audio analyser using the 32-bit ARM Cortex-M4 based microcontroller, namely the STM32F4 series of microcontrollers and more specifically the STM32F429 microcontroller manufactured by STMicroelectronics. The literature review presented in Chapter **??** encapsulates the underlying theory, mathematical rigour and processing techniques developed during in the implementation of a real-time audio analyser. Chapter 3 summarises the overall features and provides an overview of the implemented real-time audio analyser. While the full professional audio signal chain was not completely utilised, the designed hardware presented in this thesis does provide a basis and foundation for future development of the necessary firmware to interface with the 24-bit audio CODEC featured on the DAD-AV-BRD. Moreover, the designed DAD-AV-BRD hardware features additional viewing capabilities using either a projector screen or computer monitor with the standard video graphics array (VGA) controller connection, however the firmware is still under development but very near completion. Chapters 4 and 5 discuss in detail the designed hardware and the developed firmware respectively.

It has been shown in Chapter 6 that the implemented real-time audio analyser consisting of the internal 12-bit analogue-to-digital converter featured by the designed DAD-MCU-BRD displayed the correct audio spectrum for both a pure tone audio signal and a pink noise audio signal. Moreover, the implemented real-time audio analyser provided a comparable display output to the commercially available counter-parts, however there were some noted differences in the extreme high and extreme low frequencies. These differences are attributed to the two different audio analysing microphones used by each audio analysing device. Nevertheless, the overall audio analysis between the two devices in the mid-frequencies were comparable in measuring a periodic steady audio signal and non-periodic pink noise signal. Furthermore, the implemented real-time audio analyser satisfied the outlined system requirements given in the introduction to this thesis in Table 1.1.

Improvements in the designed, developed and implemented real-time audio analyser have been outlined in Chapter 7 with the hope of increasing the quality of audio measurements for an environment in future revisions.

Schematics, PCB Designs and BOM

The designed printed circuit boards (PCB) were created using the computer aided design software developed by CadSoft Computer called EAGLE. Moreover, the exact software used was the EAGLE Standard Edition - Version 6.6.0 licensed to the Department of Physics and Electronics, Rhodes University. The bill of materials (BOM) for each designed PCB indicates the cost of the components only and excludes the cost of the PCB fabrication as well as the PCB population. Furthermore, the cost of electronic components in particular semiconductor components such as integrated circuits is very dependant on the current exchange rate. Therefore, future fabrications of the designed hardware presented in this thesis may differ in overall costs.

A.1 DAD Schematics and PCB Designs

A.1.1 DAD MCU Board

The figures that follow illustrate the designed hardware printed circuit boards.

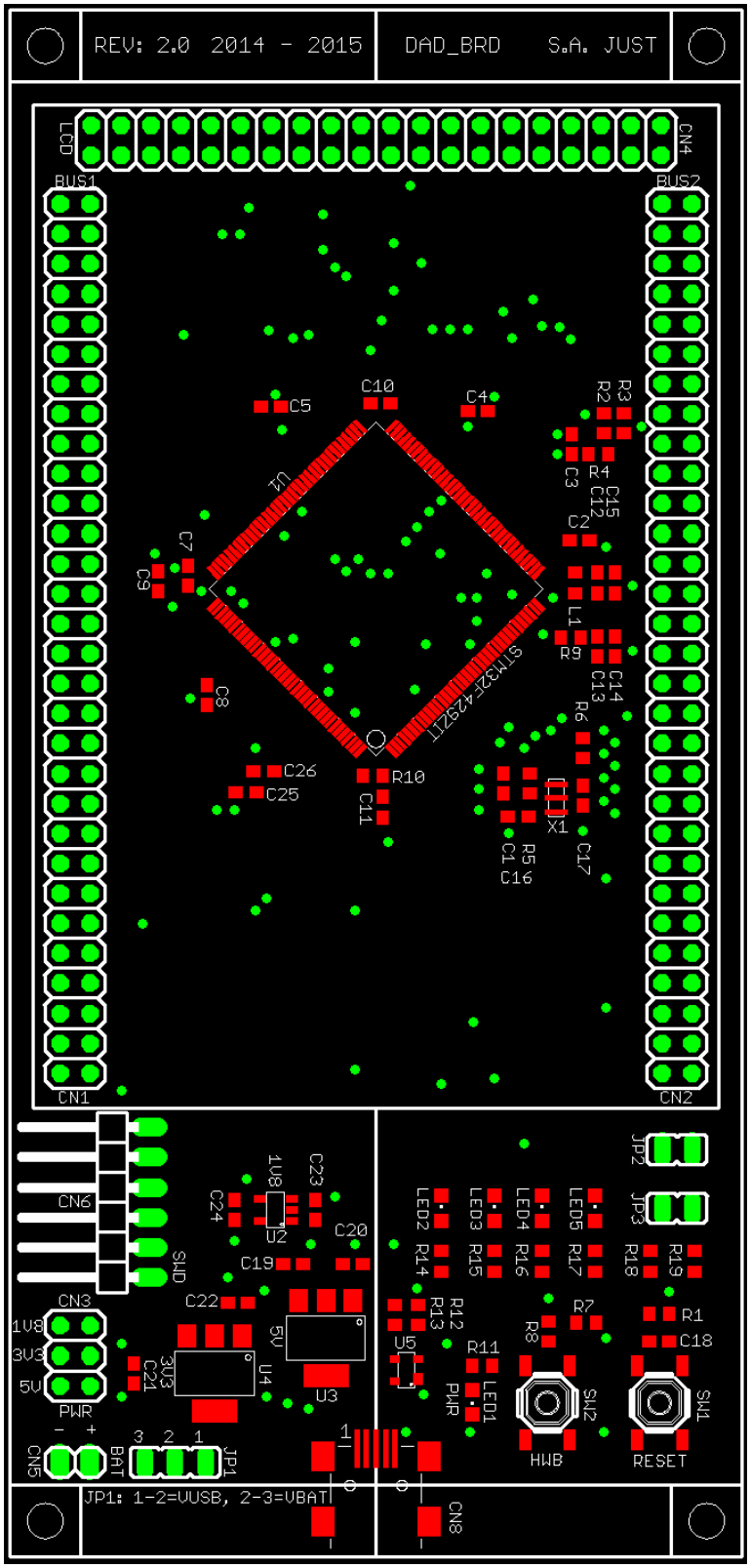


Figure A.1: DAD Microcontroller board silkscreen designed using EAGLE

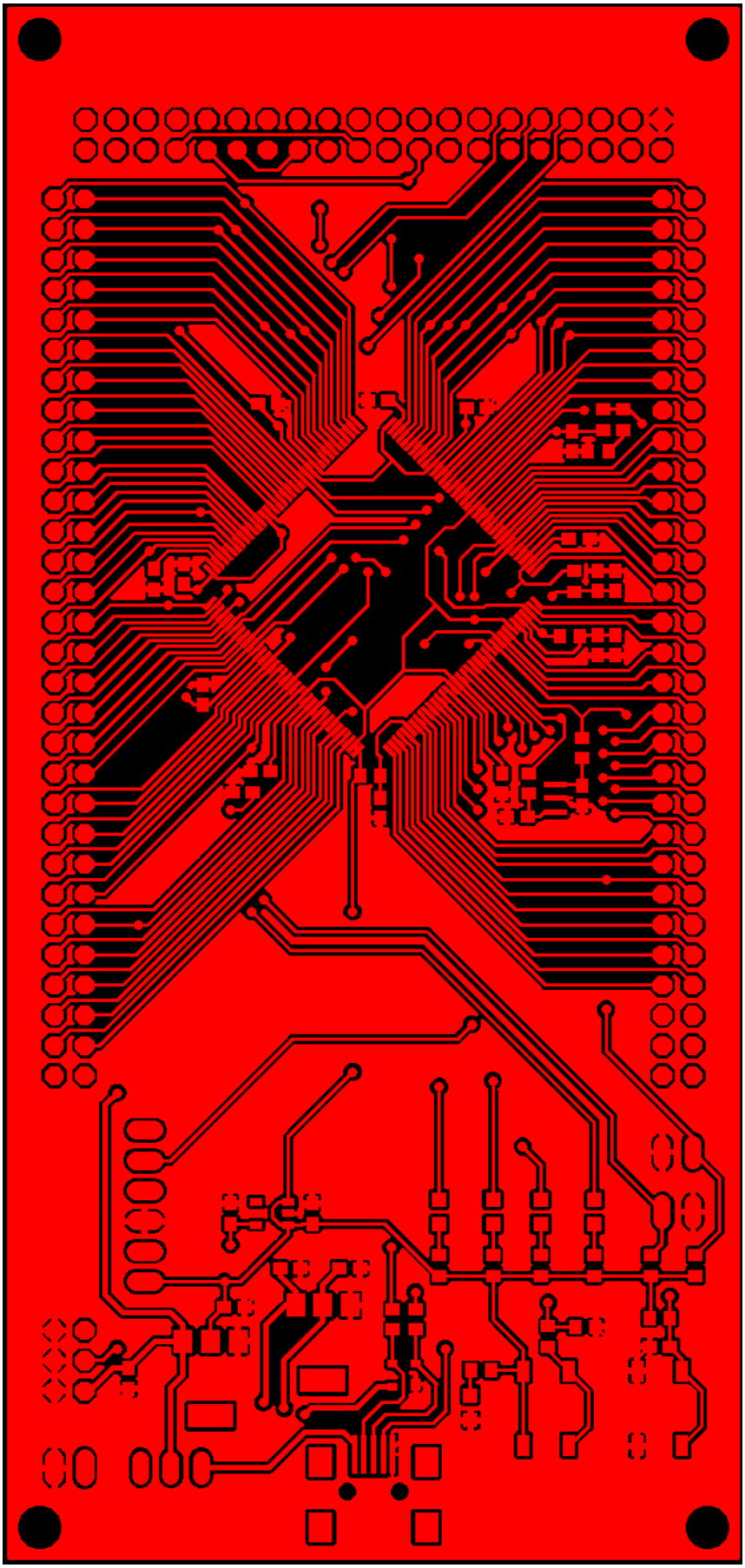


Figure A.2: DAD Microcontroller board top layer designed using EAGLE

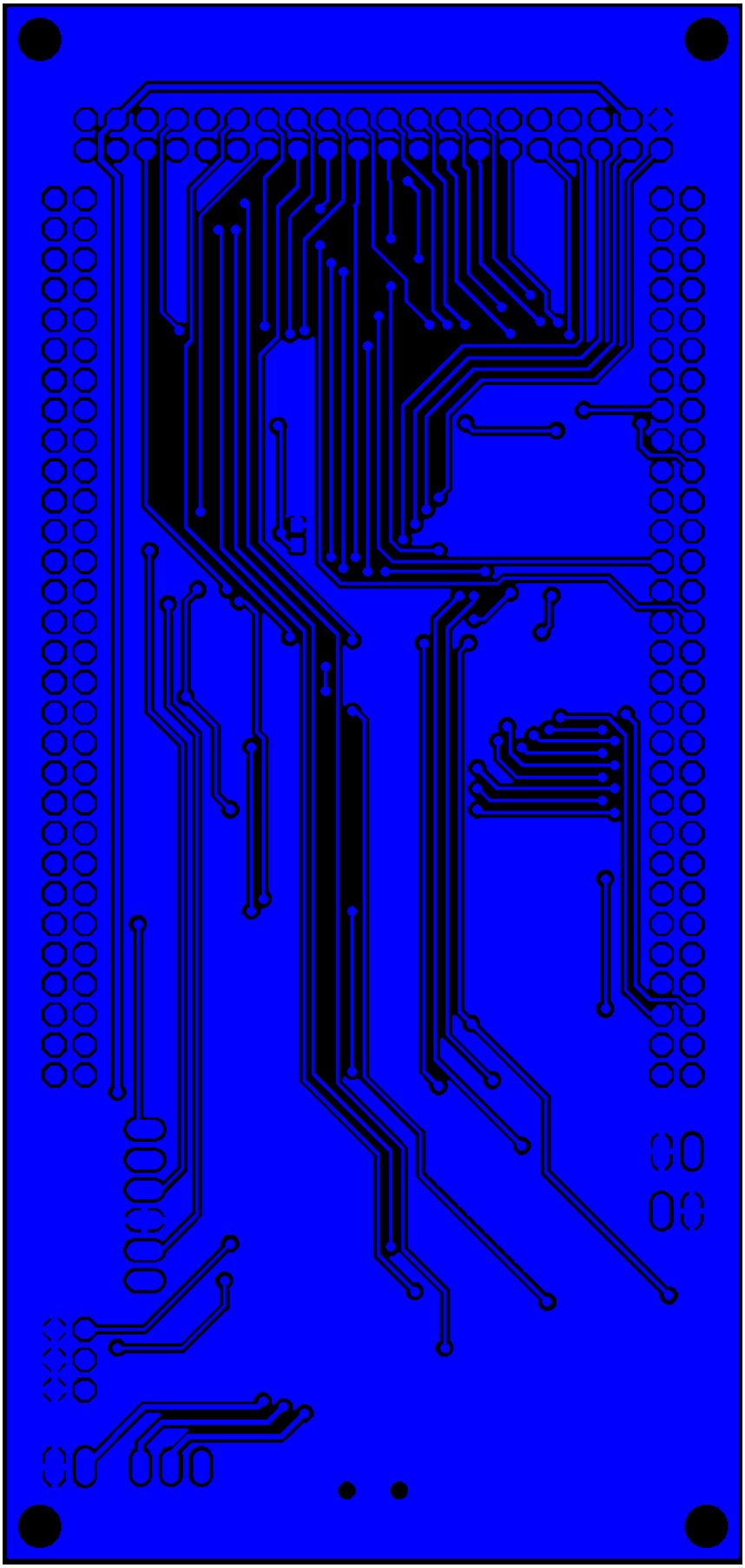


Figure A.3: DAD Microcontroller board bottom layer designed using EAGLE

PART	VALUE	PACKAGE	MANUFACTURER	MANUFACTURER PART	LOCAL SUPPLIER	SUPPLIER CODE	COST PER UNIT [ZAR]	QUANTITY	TOTAL COST [ZAR]	NOTES
U1	STM32F429ZIT	144-LQFP	STMicroelectronics	STM32F429ZIT6	RS Components (SA)	792-6053	R 302.88	1	R 302.88	Pack of 1
U2	TLV70018DDCT	5-TSOT23	Texas Instraments	TLV70018DDCT	RS Components (SA)	796-8329	R 4.18	1	R 4.18	Pack of 25
U3	NCP1117LP	SOT-223	ON Semiconductor	NCP1117LPST50T3G	RS Components (SA)	785-7187	R 3.01	1	R 3.01	Pack of 20
U4	AP1117E33G	SOT-223	Diodes Inc.	AP1117E33G-13	RS Components (SA)	751-2890	R 2.75	1	R 2.75	Pack of 50
U5	CM1293A	SOT-143	ON Semiconductor	CM1293A-02SR	RS Components (SA)	786-2880	R 1.91	1	R 1.91	Pack of 20
X1	XTAL-8MHZ	CSTCE-3PIN	Murata Electronics	CSTCE8M00G55-R0	RS Components (SA)	721-4821	R 0.46	1	R 0.46	Pack of 10
R4, R5, R10	0	R0603	TE Connectivity	CRG0603ZR	RS Components (SA)	213-1982	R 0.08	3	R 0.23	Pack of 50
R12, R13	39	R0603	Vishay	CRCW060339R0FKEA	RS Components (SA)	679-0298	R 0.06	2	R 0.12	Pack of 50
R9	47	R0603	Vishay	CRCW060347R0JNEA	RS Components (SA)	830-6472	R 0.10	1	R 0.10	Pack of 10
R6	220	R0603	Vishay	CRCW0603220RJNEA	RS Components (SA)	832-3152	R 0.13	1	R 0.13	Pack of 10
R8, R11, R14, R15, R16, R17	330	R0603	Panasonic	ERJ3RBD2200V	RS Components (SA)	810-2179	R 0.21	6	R 1.28	Pack of 100
R3	510	R0603	Vishay	CRCW0603510RJNEA	RS Components (SA)	830-6511	R 0.10	1	R 0.10	Pack of 10
R2, R7	10K	R0603	Bourns	CR0603-JW-103GLF	RS Components (SA)	740-8892	R 0.05	2	R 0.10	Pack of 50
R1, R18, R19	100K	R0603	Bourns	CR0603-FX-1003ELF	RS Components (SA)	740-8802	R 0.10	3	R 0.30	Pack of 50
C16, C17	20p	C0603	Murata Electronics	GRM1885C2A200JA01D	RS Components (SA)	815-1525	R 0.22	2	R 0.45	Pack of 200
C1, C2, C3, C4, C5, C6, C7, C8, C14, C15, C18, C26	100n	C0603	TDK	C1608X7R1E104K080AA	RS Components (SA)	788-2916	R 0.06	12	R 0.68	Pack of 50
C11, C12, C13	1u	C0603	AVX	0603YD105KAT2A	RS Components (SA)	698-3336	R 0.17	3	R 0.51	Pack of 100
C9, C10	2.2u	C0603	Murata Electronics	GRM188F51A225ZE01D	RS Components (SA)	723-5503	R 0.62	2	R 1.23	Pack of 100
C25	4.7u	C0603	AVX	06034D475KAT2A	RS Components (SA)	698-3248	R 0.37	1	R 0.37	Pack of 25
C19, C20, C21, C23, C24	10u	C0603	Murata Electronics	GRM188R60J106ME47J	RS Components (SA)	815-1355	R 0.50	5	R 2.48	Pack of 100
C22	22u	C0603	TDK	C1608X5R0J226M080AC	RS Components (SA)	788-2880	R 2.37	1	R 2.37	Pack of 25
L1	600	L0608	Murata Electronics	BLM18AG601SN1D	RS Components (SA)	724-1299	R 0.52	1	R 0.52	Pack of 25
LED1, LED2	RED	2012 (0805)	Lite-On	LTST-C171EKT	RS Components (SA)	692-0931	R 0.56	2	R 1.13	Pack of 50
LED3	ORANGE	2012 (0805)	Lite-On	LTST-C171AKT	RS Components (SA)	692-0929	R 0.57	1	R 0.57	Pack of 50
LED4	YELLOW	2012 (0805)	Lite-On	LTST-C170YKT	RS Components (SA)	692-0925	R 0.59	1	R 0.59	Pack of 50
LED5	GREEN	2012 (0805)	Lite-On	LTST-C170GKT	RS Components (SA)	692-0900	R 0.56	1	R 0.56	Pack of 50
SW1, SW2	RST, HWB	Black Tactile Switch	Wurth Elektronik	430182043816	RS Components (SA)	785-6260	R 5.78	2	R 11.56	Pack of 1
CN1, CN2	BUS1, BUS2	2x30 long pin header (F+M)	-	-	Commmunica (SA)	705100-15MM	R 6.43	12	R 77.16	Pack of 1
CN3	PWR	2x3 long pin header (F+M)	-	-	Commmunica (SA)	705080-15MM	R 5.23	1	R 5.23	Pack of 1
CN4	LCD	2x20 long pin header (F)	-	-	Commmunica (SA)	705100-15MM	R 6.43	4	R 25.72	Pack of 1
CN5	BAT	1x2 pin header (M)	-	-	Netram (SA)	COM-00122	R 0.25	1	R 0.25	Pack of 40
CN6	SWD	1x6 right-angle pin header (M)	-	-	Netram (SA)	COM-00125	R 1.19	1	R 1.19	Pack of 40
CN7	USB-MINI-B	USB-MINI-B-SMD-4PIN	-	-	Netram (SA)	COM-00264	R 19.95	1	R 19.95	Pack of 1
JP1	-	1x3 pin header (M)	-	-	Netram (SA)	COM-00122	R 0.37	1	R 0.37	Pack of 40
JP2, JP3	-	1x2 pin header (M)	-	-	Netram (SA)	COM-00122	R 0.25	2	R 0.50	Pack of 40
LCD_TFT + SD_CARD	ITDB02-3.2S	PCB MODULE	IteadStudio	IM120419005	-	-	R 290.00	1	R 290.00	Pack of 1
							TOTAL	82	R 760.95	1$ = R15.89

Table A.1: Bill of Materials for DAD Microcontroller Board

A.1.2 DAD External Memory Board

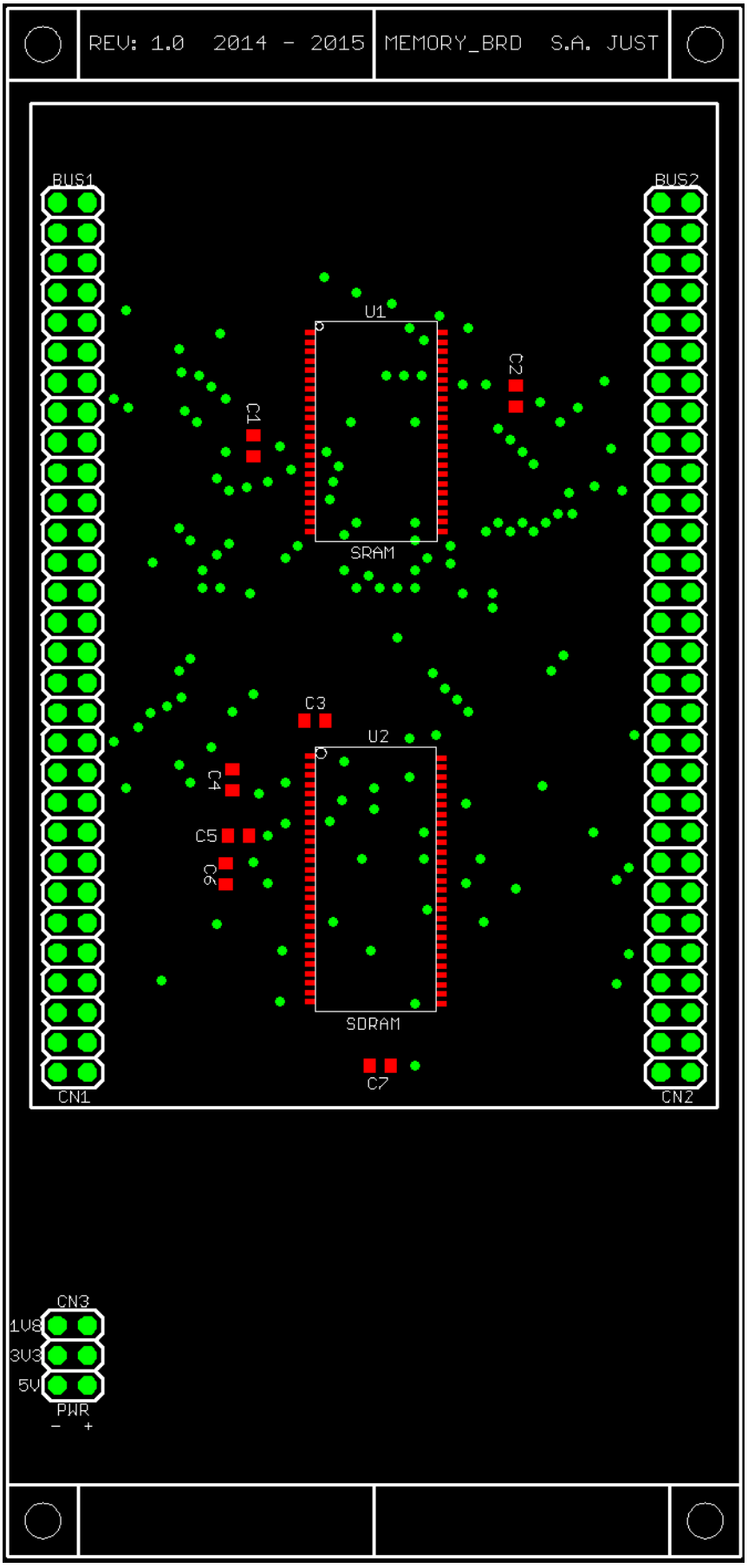


Figure A.4: DAD Memory board silkscreen designed using EAGLE

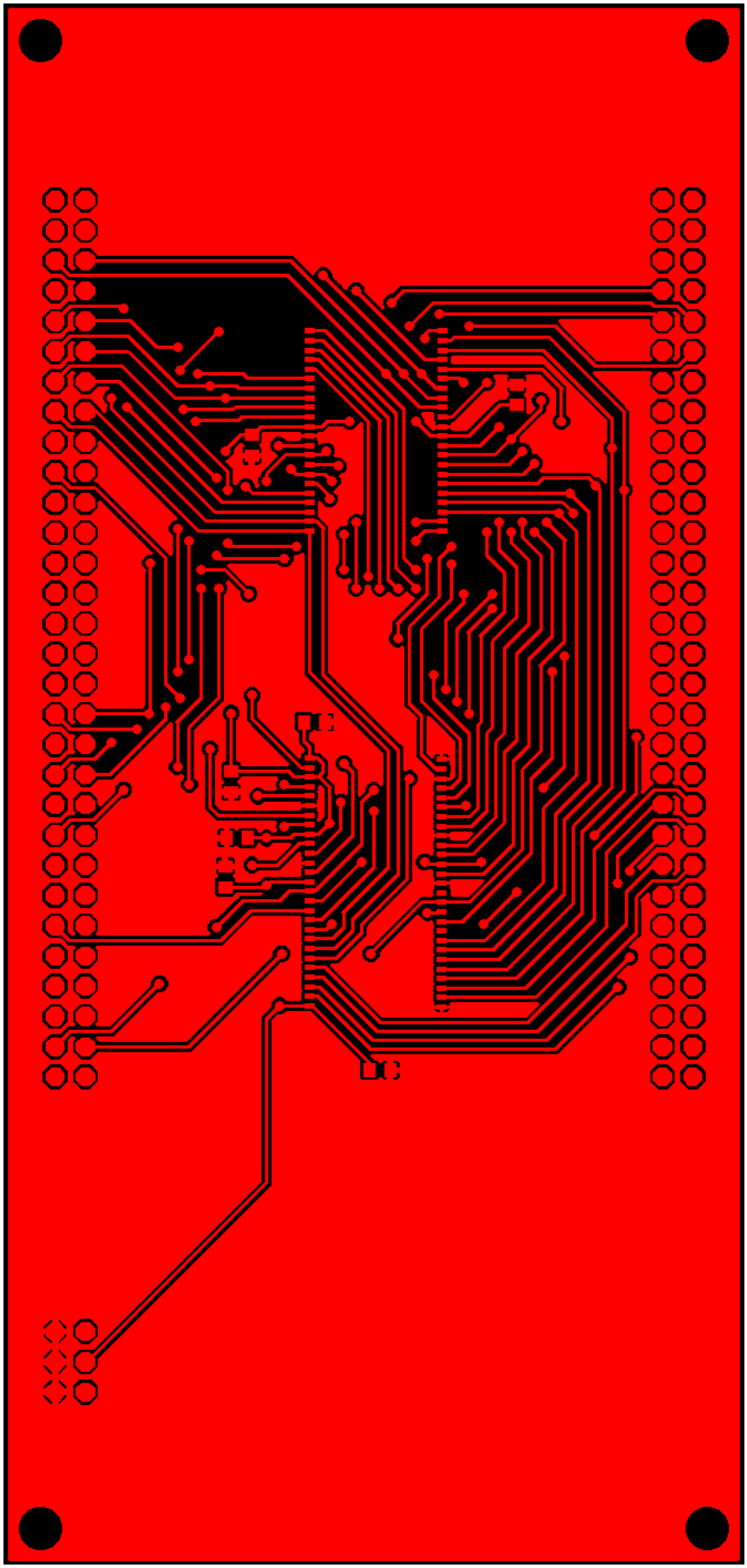


Figure A.5: DAD Memory board top layer designed using EAGLE

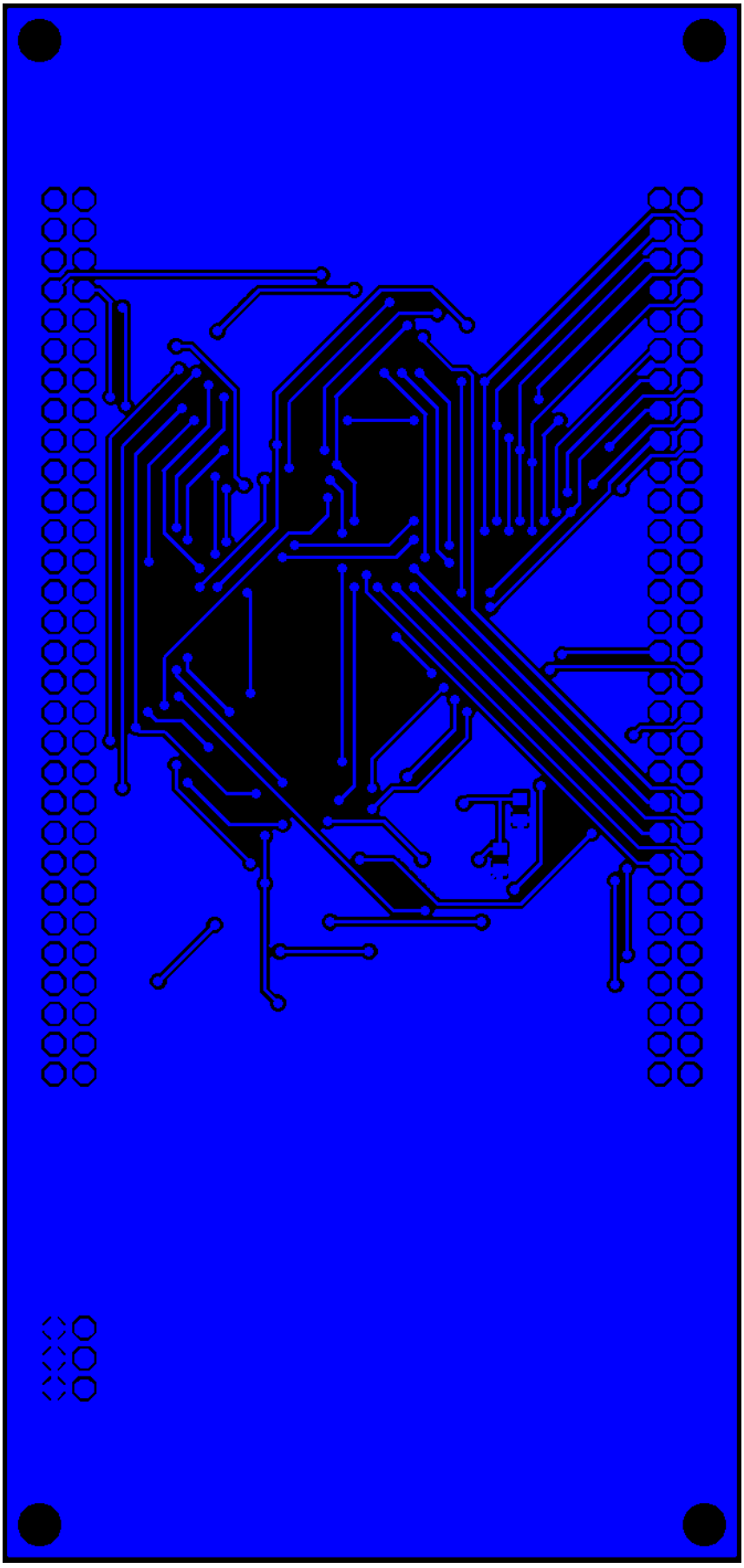


Figure A.6: DAD Memory board bottom layer designed using EAGLE

NAME	VALUE	PACKAGE	MANUFACTURER	MANUFACTURER PART	LOCAL SUPPLIER	SUPPLIER CODE	COST PER UNIT[ZAR]	QUANTITY	TOTAL COST [ZAR]	NOTES
U1	AS6C8016	44-TSOP	Alliance Memory	AS6C8016-55ZIN	RS Components (SA)	538-271	R 158.36	1	R 158.366	Pack of 1
U2	AS4C8M16S	54-TSOP	Alliance Memory	AS4C8M16S-7TCN	RS Components (SA)	744-4536	R 51.75	1	R 51.75	Pack of 2
C1, C2, C3, C4, C5, C6, C7, C8, C9	100n	C0603	TDK	C1608X7R1E104K080AA	RS Components (SA)	788-2916	R 0.06	9	R 0.51	Pack of 50
CN1, CN2	BUS1, BUS2	2x30 long pin header (F+M)	-	-	Commmunica (SA)	705100-15MM	R 6.43	12	R 77.16	Pack of 1
CN3	PWR	2x3 long pin header (F+M)	-	-	Commmunica (SA)	705080-15MM	R 5.23	1	R 5.23	Pack of 1
							TOTAL	15	R 293.01	1$ = R15.89

Table A.2: Bill of Materials for DAD MEM Board

A.1.3 DAD Audio and Video Board

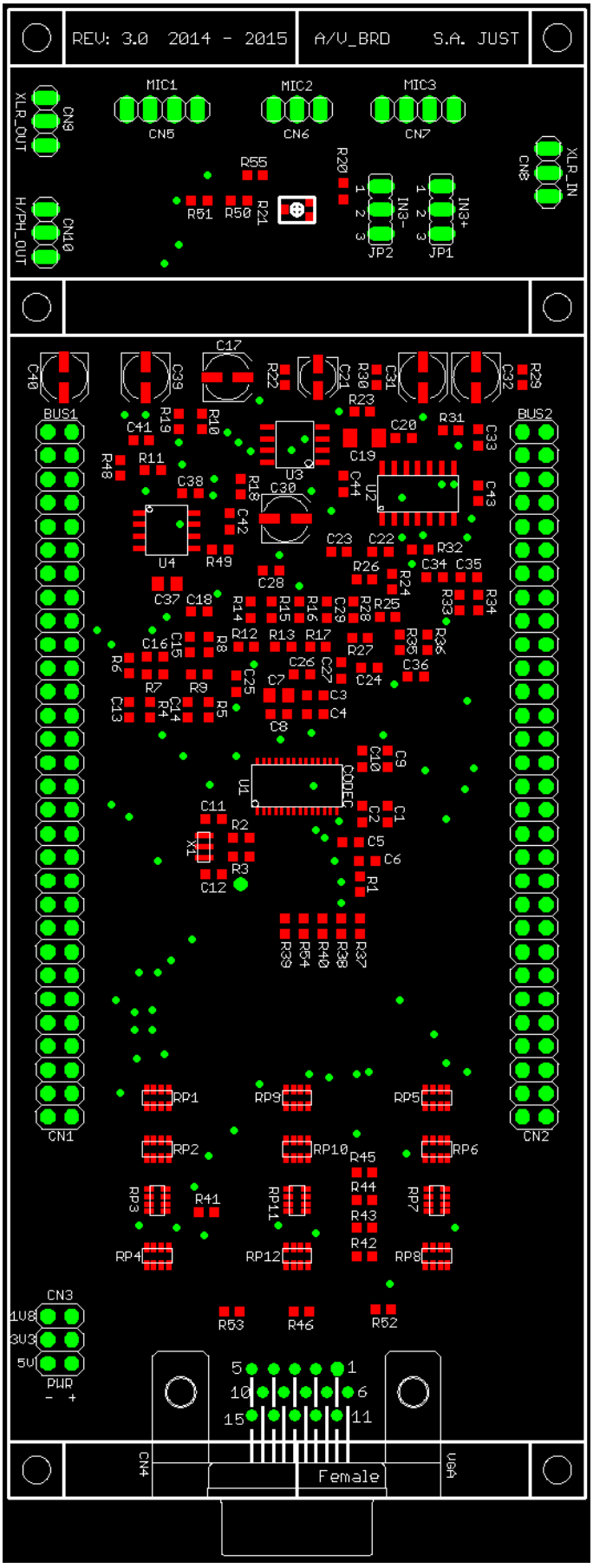


Figure A.7: DAD AV board silkscreen designed using EAGLE

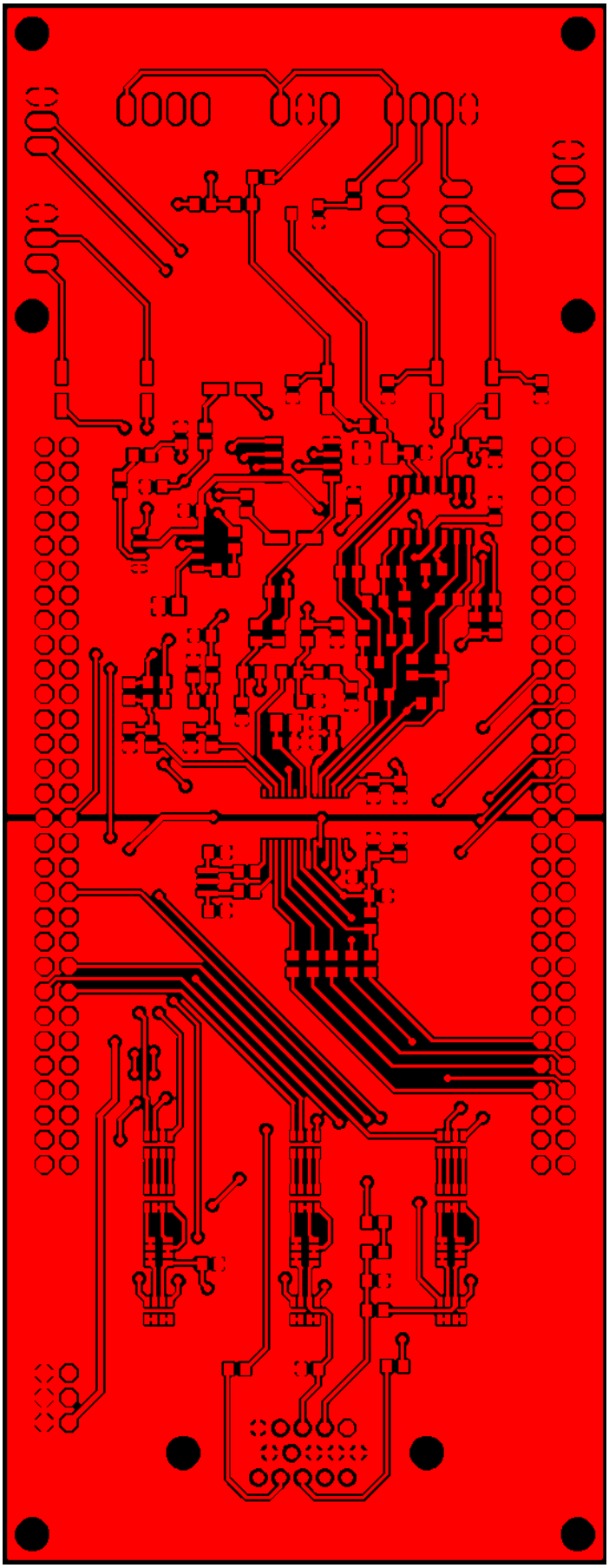


Figure A.8: DAD AV board top layer designed using EAGLE

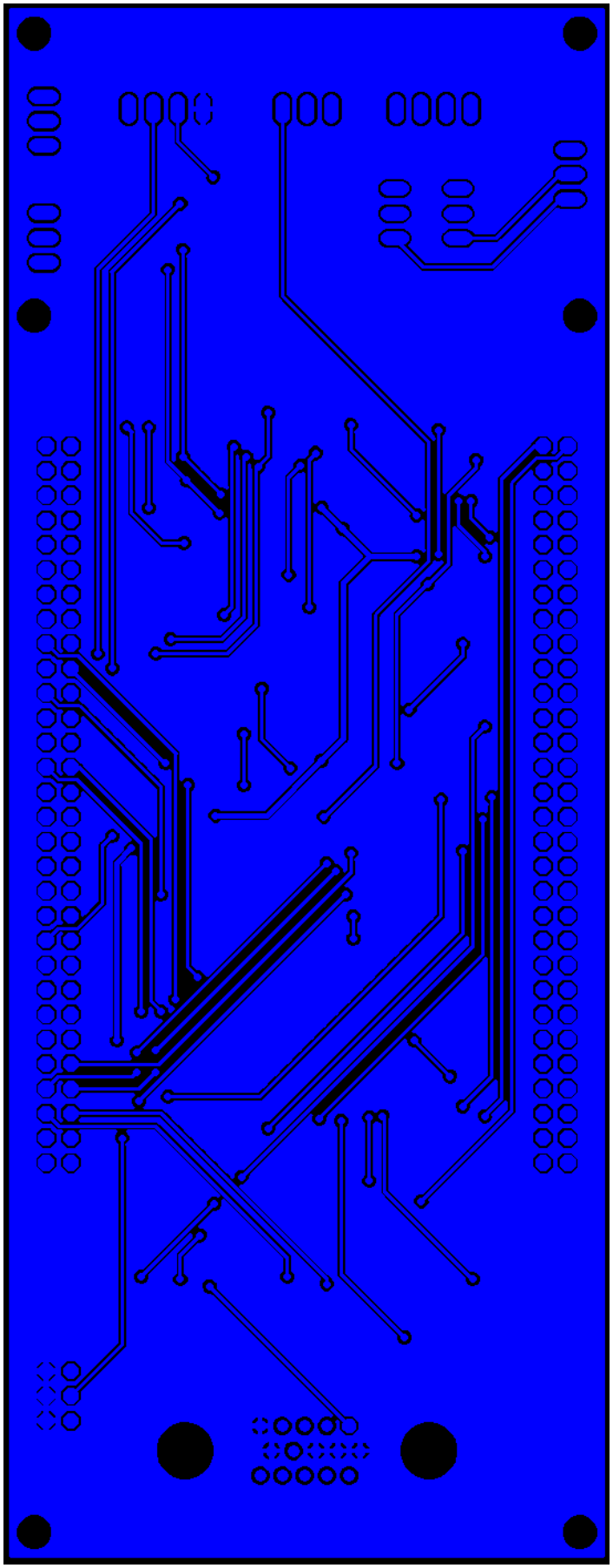


Figure A.9: DAD AV board bottom layer designed using EAGLE

NAME	VALUE	PACKAGE	MANUFACTURER	MANUFACTURER PART	LOCAL SUPPLIER	SUPPLIER CODE	COST PER UNIT [ZAR]	QUANTITY	TOTAL COST [ZAR]	NOTES
U1	CS4272	28-TSSOP	Cirrus Logic	CS4272-CZZ	RS Components (SA)	491-5681	R 187.18	1	R 187.18	Pack of 1
U2	MCP664	14-SOIC	Microchip Technology Inc.	MCP664-E/SL	RS Components (SA)	755-9530	R 29.62	1	R 29.62	Pack of 3
U3	MCP662	8-SOIC	Microchip Technology Inc.	MCP662-E/SN	RS Components (SA)	687-8694	R 25.24	1	R 25.24	Pack of 1
U4	BH3544F	8-SOP	ROHM Semiconductor	BH3544F-E2	RS Components (SA)	700-3551	R 9.82	1	R 9.82	Pack of 1
X1	XTAL-20MHZ	CSTCE-3PIN	Murata Electronics	CSTCE20M0V53-R0	RS Components (SA)	721-4811	R 0.49	1	R 0.49	Pack of 10
RP1, RP2, RP3, RP4, RP5, RP6, RP7, RP8, RP9, RP10, RP11, RP12	220	CAY16-J4-1206	Bourns	CAY16-221J4LF	RS Components (SA)	522-5636	R 0.69	12	R 8.29	Pack of 50
R2, R3, R37, R38, R39, R40, R50, R51, R54, R55	0	R0603	TE Connectivity	CRG0603ZR	RS Components (SA)	213-1982	R 0.08	10	R 0.75	Pack of 50
R25, R28, R34, R36	91	R0603	Vishay	CRCW060391R0FKEA	RS Components (SA)	679-0780	R 0.16	4	R 0.64	Pack of 50
R52, R53	240	R0603	Panasonic	ERJ3RBD2400V	RS Components (SA)	810-2185	R 0.21	2	R 0.43	Pack of 100
R41, R42, R43, R44, R45, R46	270	R0603	Vishay	CRCW06033270RFKEA	RS Components (SA)	679-0071	R 0.10	6	R 0.61	Pack of 50
R10, R18	560	R0603	Vishay	CRCW0603560RJNEA	RS Components (SA)	830-6549	R 0.10	2	R 0.21	Pack of 10
R24, R26, R27, R33, R35	634 (replace with cheaper 620)	R0603	Vishay	CRCW06036620RFKEA	RS Components (SA)	679-0605	R 0.12	5	R 0.60	Pack of 50
R8, R16	715	R0603	Vishay	CRCW06033715RFKEA	RS Components (SA)	679-0686	R 0.12	2	R 0.24	Pack of 50
R5, R13	1K33	R0603	Vishay	CRCW06031K33FKEA	RS Components (SA)	678-9891	R 0.09	2	R 0.18	Pack of 50
R9, R17	1K5	R0603	Panasonic	ERJ3RBD1501V	RS Components (SA)	810-2139	R 0.21	2	R 0.43	Pack of 100
R7, R15	2K32	R0603	Vishay	CRCW06032K32FKEA	RS Components (SA)	679-0122	R 0.21	2	R 0.42	Pack of 50
R20	3K3	R0603	Vishay	CRCW06033K30JNEA	RS Components (SA)	830-6410	R 0.10	1	R 0.10	Pack of 10
R4, R12	4K42	R0603	Vishay	CRCW06034K42FKEA	RS Components (SA)	679-0478	R 0.21	2	R 0.42	Pack of 50
R6, R14	4K99	R0603	Vishay	CRCW06034K99FKEA	RS Components (SA)	679-0490	R 0.12	2	R 0.24	Pack of 50
R31, R32	10K	R0603	Bourns	CR0603-JW-103GLF	RS Components (SA)	740-8892	R 0.05	2	R 0.10	Pack of 50
R1, R11, R19	47K	R0603	Vishay	CRCW060347K0FKEA	RS Components (SA)	679-0425	R 0.26	3	R 0.78	Pack of 50
R22, R23, R29, R30, R48, R49	100K	R0603	Bourns	CR0603-FX-1003ELF	RS Components (SA)	740-8802	R 0.10	6	R 0.59	Pack of 50
C11, C12	20p	C0603	Murata Electronics	GRM1885C2A200JA01D	RS Components (SA)	815-1525	R 0.22	2	R 0.45	Pack of 200
C16, C22, C23, C28, C34, C35	470p	C0603	TDK	C1608C0G1H471J080AA	RS Components (SA)	788-2969	R 0.17	6	R 1.04	Pack of 50
C18, C29	1n5	C0603	KEMET	C0603C152K1RACTU	RS Components (SA)	148-137	R 0.47	2	R 0.93	Pack of 50
C13, C25	2n2	C0603	Murata Electronics	GCM188R71H222KA37D	RS Components (SA)	723-4998	R 0.20	2	R 0.39	Pack of 200
C24, C36	2n7	C0603	Murata Electronics	GRM1885C1H272JA01D	RS Components (SA)	723-5887	R 0.64	2	R 1.27	Pack of 100
C14, C26	6n8	C0603	KEMET	C0603C682K5RACTU	RS Components (SA)	147-825	R 0.22	2	R 0.43	Pack of 50
C20, C33	10n	C0603	TDK	C1608X7R1E103K080AA	RS Components (SA)	788-2907	R 0.06	2	R 0.11	Pack of 50
C1, C3, C5, C8, C10	100n	C0603	TDK	C1608X7R1E104K080AA	RS Components (SA)	788-2916	R 0.06	5	R 0.29	Pack of 50
C2, C4, C6, C9, C41, C42	1u	C0603	AVX	0603YD105KAT2A	RS Components (SA)	698-3336	R 0.17	6	R 1.03	Pack of 100
C38	10u	C0603	Murata Electronics	GRM188R60J106ME47J	RS Components (SA)	815-1355	R 0.50	1	R 0.50	Pack of 100
C15, C27, C43, C44	22u	C0603	TDK	C1608X5R0J226M080AC	RS Components (SA)	788-2880	R 2.37	4	R 9.50	Pack of 25
C7, C37	47u	C0805	Murata Electronics	GRM21BR60J476ME15L	RS Components (SA)	846-7322	R 3.95	2	R 7.90	Pack of 10
C19	100u	C1206	Samsung Electro-Mechanics	CL31A107MQHNNNE	RS Components (SA)	766-1078	R 4.87	1	R 4.87	Pack of 10
C21	1u	CPOL_SMD (4 x 5.8)	NIC Components	NACE1R0M50V4X5.5TR13F	RS Components (SA)	737-9669	R 0.76	1	R 0.76	Pack of 50
C31, C32	10u	CPOL_SMD (5 x 5.8)	NIC Components	NACE100M16V4X5.5TR13F	RS Components (SA)	737-9625	R 1.53	2	R 3.05	Pack of 50
C17, C30	22u	CPOL_SMD (5 x 5.8)	NIC Components	NACE220M16V5X5.5TR13F	RS Components (SA)	737-9634	R 2.36	2	R 4.72	Pack of 50
C39, C40	330u	CPOL_SMD (5 x 5.8)	Nichicon	UWX0J331MCL1GB	RS Components (SA)	739-5535	R 3.95	2	R 7.91	Pack of 50
CN1, CN2	BUS1, BUS2	2x30 long pin header (F+M)	-	-	Commmunica (SA)	705100-15MM	R 6.43	12	R 77.16	Pack of 1
CN3	PWR	2x3 long pin header (F+M)	-	-	Commmunica (SA)	705080-15MM	R 5.23	1	R 5.23	Pack of 1
CN4	VGA	HDF15H-DSUB (DE-15)	ASSMANN WSW	A-HDF 15 A-KG/T	RS Components (SA)	674-0971	R 20.41	1	R 20.41	Pack of 1
CN5, CN7	MIC1, MIC3	1x4 (FEMALE)	-	-	Netram (SA)	COM-00123	R 1.24	2	R 2.49	Pack of 40
CN6, CN8, CN9, CN10	MIC2, XLR_IN, XLR_OUT, H/PH_OUT	1x3 (FEMALE)	-	-	Netram (SA)	COM-00123	R 1.00	4	R 3.98	Pack of 40
JP1, JP2	IN3+, IN3-	1x3 (MALE)	-	-	Netram (SA)	COM-00122	R 0.37	2	R 0.74	Pack of 40
							TOTAL	21	R 422.55	1$ = R15.89

Table A.3: Bill of Materials for DAD AV Board

A.1.4 DAD Pinout Board

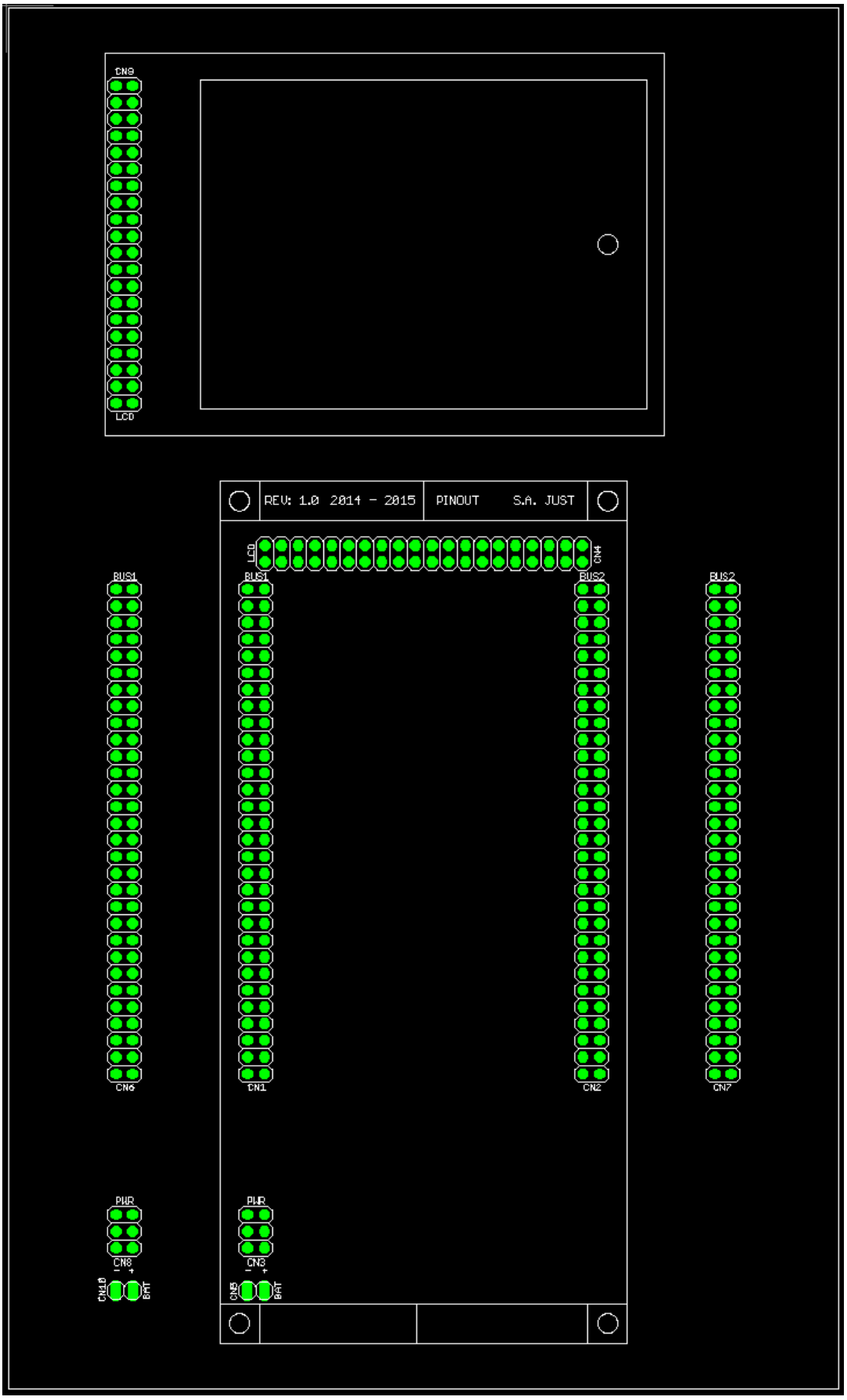


Figure A.10: DAD PINOUT board silkscreen designed using EAGLE

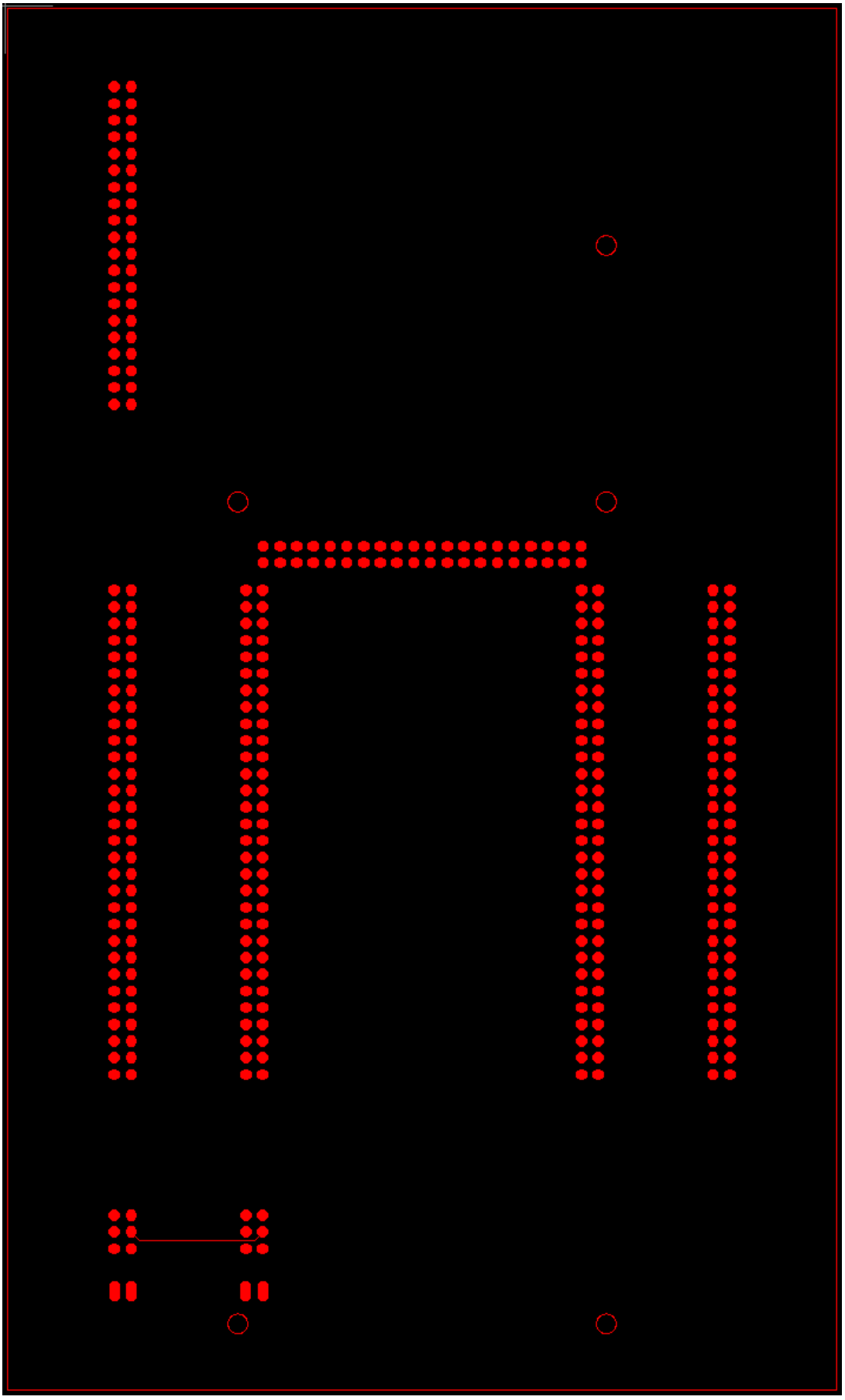


Figure A.11: DAD PINOUT board top layer designed using EAGLE

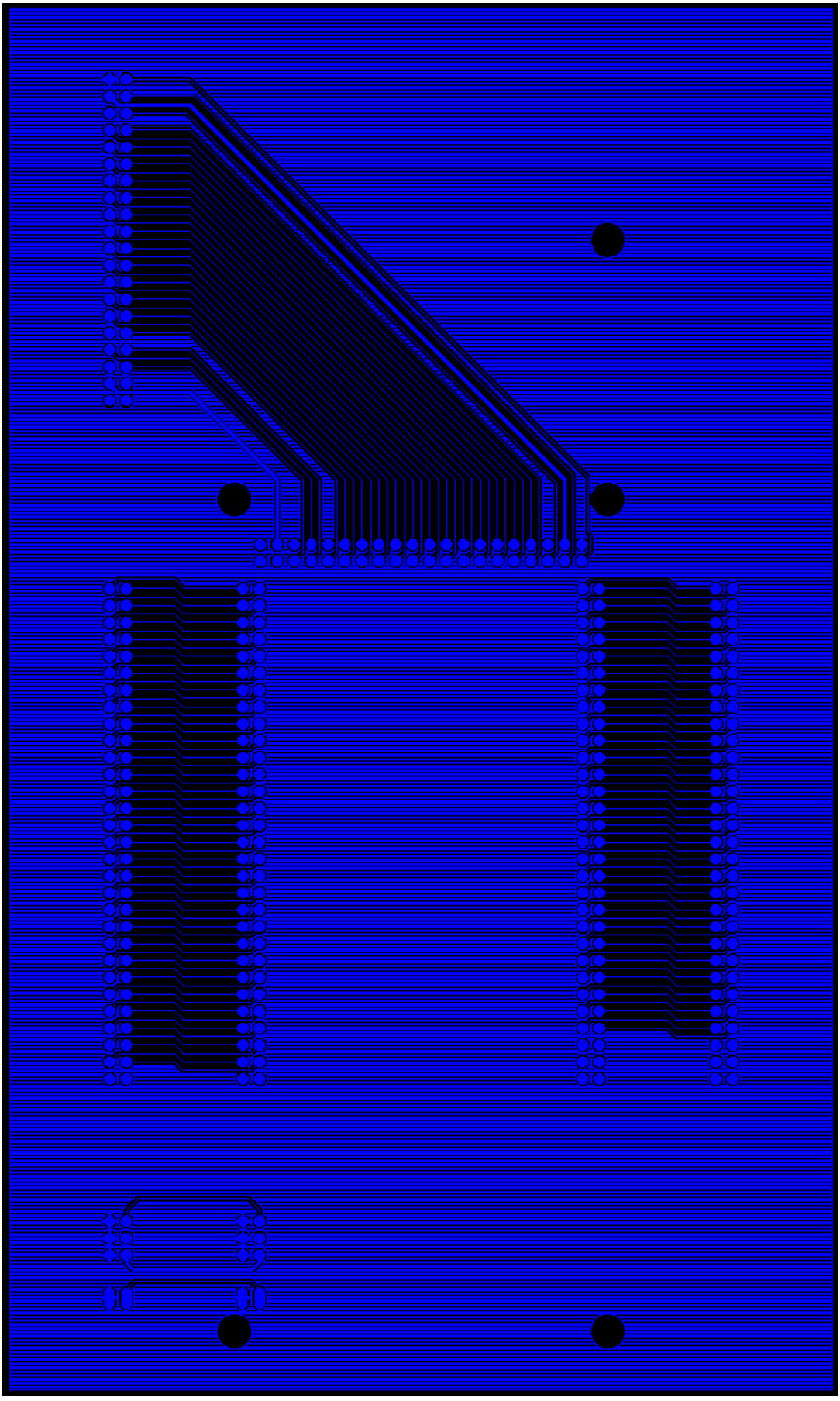


Figure A.12: DAD PINOUT board bottom layer designed using EAGLE

NAME	VALUE	PACKAGE	MANUFACTURER	MANUFACTURER PART	LOCAL SUPPLIER	SUPPLIER CODE	COST PER UNIT [ZAR]	QUANTITY	TOTAL COST [ZAR]	NOTES
CN1, CN2	BUS1, BUS2	2x30 long pin header (F+M)	-	-	Commmunica (SA)	705100-15MM	R 6.43	12	R 77.16	Pack of 1
CN3	PWR	2x3 long pin header (F+M)	-	-	Commmunica (SA)	705080-15MM	R 5.23	1	R 5.23	Pack of 1
CN4	LCD	2x20 long pin header (F)	-	-	Commmunica (SA)	705100-15MM	R 6.43	4	R 25.72	Pack of 1
CN5	BAT	1x2 pin header (M)	-	-	Netram (SA)	COM-00122	R 0.25	1	R 0.25	Pack of 40
CN6, CN7	BUS1, BUS2	2x30 long pin header (F+M)	-	-	Commmunica (SA)	705100-15MM	R 6.43	12	R 77.16	Pack of 1
CN8	PWR	2x3 long pin header (F+M)	-	-	Commmunica (SA)	705080-15MM	R 5.23	1	R 5.23	Pack of 1
CN9	LCD	2x20 long pin header (F)	-	-	Commmunica (SA)	705100-15MM	R 6.43	4	R 25.72	Pack of 1
CN10	BAT	1x2 pin header (M)	-	-	Netram (SA)	COM-00122	R 0.25	1	R 0.25	Pack of 40
							TOTAL	36	R 216.72	1$ = R14.98

Table A.4: Bill of Materials for DAD PINOUT Board

A.1.5 Internal MEMS Microphone Breakout Boards

A.1.5.1 C928A Breakout Board

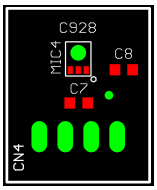


Figure A.13: C928A board silkscreen designed using EAGLE

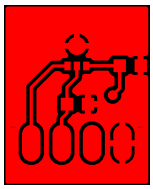


Figure A.14: C928A board top layer designed using EAGLE

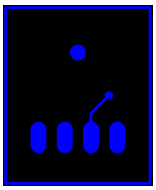


Figure A.15: C928A board bottom layer designed using EAGLE

A.1.5.2 MP34DT01 Breakout Board

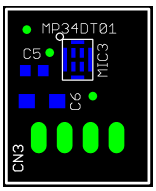


Figure A.16: MP34DT01 board silkscreen designed using EAGLE

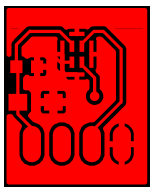


Figure A.17: MP34DT01 board top layer designed using EAGLE

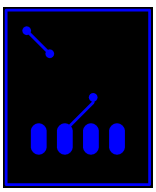


Figure A.18: MP34DT01 board bottom layer designed using EAGLE

STM32F4 Peripheral Registers

A list of the STM32F4 peripheral registers is given below[3][41].

B.1 Reset and Clock Controller (RCC)

The Reset and Clock Controller (RCC) is configured by the following 32-bit registers:

- `RCC_CR` - RCC Clock Control Register
- `RCC_PLLCFGR` - RCC PLL Configuration Register
- `RCC_CFGR` - RCC Clock Configuration Register
- `RCC_CIR` - RCC Clock Interrupt Register
- `RCC_AHB1RSTR` - RCC AHB1 Peripheral Reset Register
- `RCC_AHB2RSTR` - RCC AHB2 Peripheral Reset Register
- `RCC_AHB3RSTR` - RCC AHB3 Peripheral Reset Register
- `RCC_APB1RSTR` - RCC APB1 Peripheral Reset Register
- `RCC_APB2RSTR` - RCC APB2 Peripheral Reset Register
- `RCC_AHB1ENR` - RCC AHB1 Peripheral Clock Enable Register
- `RCC_AHB2ENR` - RCC AHB2 Peripheral Clock Enable Register
- `RCC_AHB3ENR` - RCC AHB3 Peripheral Clock Enable Register
- `RCC_APB1ENR` - RCC APB1 Peripheral Clock Enable Register

- `RCC_APB2ENR` - RCC APB2 Peripheral Clock Enable Register
- `RCC_AHB1LPENR` - RCC AHB1 Peripheral Clock Enable in Low Power Mode Register
- `RCC_AHB2LPENR` - RCC AHB2 Peripheral Clock Enable in Low Power Mode Register
- `RCC_AHB3LPENR` - RCC AHB3 Peripheral Clock Enable in Low Power Mode Register
- `RCC_APB1LPENR` - RCC APB1 Peripheral Clock Enable in Low Power Mode Register
- `RCC_APB2LPENR` - RCC APB2 Peripheral Clock Enable in Low Power Mode Register
- `RCC_BDCR` - RCC Backup Domain Control Register
- `RCC_CSR` - RCC Clock Control & Status Register
- `RCC_SSCGR` - RCC Spread Spectrum Clock Generation Register
- `RCC_PLLI2SCFGR` - RCC PLLI2S Configuration Register
- `RCC_PLLSAICFGR` - RCC PLLSAI Configuration Register
- `RCC_DCKCFGR` - RCC Dedicated Clock Control Register

B.2 General Purpose Input/Output (GPIO) Controller

The General Purpose Input/Output Controller is configured by the following 32-bit registers:

- `GPIOx_MODER` - GPIO port Mode Register
- `GPIOx_OTYPER` - GPIO port Output Type Register
- `GPIOx_OSPEEDR` - GPIO port Output Speed Register
- `GPIOx_PUPDR` - GPIO port Pull-up/Pull-down Register
- `GPIOx_IDR` - GPIO port Input Data Register
- `GPIOx_ODR` - GPIO port Output Data Register
- `GPIOx_BSRR` - GPIO port Bit Set/Reset Register
- `GPIOx_LCKR` - GPIO port Configuration Lock Register
- `GPIOx_AFRH` - GPIO Alternative Function High Register
- `GPIOx_AFRL` - GPIO Alternative Function Low Register

where x = A, B, C, D, E, F, G, H.

B.3 Nested Interrupt Controller (NVIC)

The Nested Vectored Interrupt Controller (NVIC) is configured by the following 32-bit registers:

- `NVIC_ISERx` - Interrupt Set-Enable Registers
- `NVIC_ICERx` - Interrupt Clear-Enable Registers
- `NVIC_ISPRx` - Interrupt Set-Pending Registers
- `NVIC_ICPRx` - Interrupt Clear-Pending Registers
- `NVIC_IABRx` - Interrupt Active Bit Registers
- `NVIC_IPRx` - Interrupt Priority Registers
- `NVIC_STIR` - Software Trigger Interrupt Register

where x = 0, 1, 2.

Alternatively, the NVIC can be configured by the following CMSIS functions:

- `void NVIC_EnableIRQ(IRQn_Type IRQn)` - Enables an interupt or exception
- `void NVIC_DisableIRQ(IRQn_Type IRQn)` - Disables an interrupt or exception
- `void NVIC_SetPendingIRQ(IRQn_Type IRQn)` - Sets the pending status of an interrupt or exception to 1
- `void NVIC_ClearPendingIRQ(IRQn_Type IRQn)` - Clears the pending status of an interrut or exception to 0
- `uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn)` - Returns a non-zero value if the pending status interrupt or exception is set to 1
- `void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)` - Sets the priority of an interrupt or exception with priority level to 1
- `uint32_t NVIC_GetPriority(IRQn_Type IRQn)` - Returns the current priority level of an interrupt or exception

where IRQn is the IRQ number.

B.4 Advanced Timer 8 (TIM8)

The Advanced Timer TIM8 is configured by the following 32-bit registers:

- `TIM8_CR1` - TIM8 Control Register 1

- `TIM8_CR2` - TIM8 Control Register 2
- `TIM8_SMCR` - TIM8 Slave Mode Control Register
- `TIM8_DIER` - TIM8 DMA/Interrupt Enable Register
- `TIM8_SR` - TIM8 Status Register
- `TIM8_EGR` - TIM8 Event Generation Register
- `TIM8_CCMR1` - TIM8 Capture/Compare Mode Register 1
- `TIM8_CCMR2` - TIM8 Capture/Compare Mode Register 2
- `TIM8_CCMR3` - TIM8 Capture/Compare Enable Register
- `TIM8_CNT` - TIM8 Counter Register
- `TIM8_PSC` - TIM8 Prescaler Register
- `TIM8_ARR` - TIM8 Auto-Reload Register
- `TIM8_RCR` - TIM8 Repetition Counter Register
- `TIM8_CCR1` - TIM8 Capture/Compare Register 1
- `TIM8_CCR2` - TIM8 Capture/Compare Register 2
- `TIM8_CCR3` - TIM8 Capture/Compare Register 3
- `TIM8_CCR4` - TIM8 Capture/Compare Register 4
- `TIM8_BDTR` - TIM8 Break and Dead-Time Register
- `TIM8_DCR` - TIM8 DMA Control Register
- `TIM8_DMAR` - TIM8 DMA Address for Full Transfer Register

B.5 Flexible Memory Controller (FMC)

The Flexible Memory Controller (FMC) interfaces with the on-board ITDB02-3.2S module and is configured to use memory Bank 1 by the following 32-bit registers:

- `FMC_BCR1` - SRAM/NOR Flash Bank 1 Chip-Select Control Register
- `FMC_BTR1` - SRAM/NOR Flash Bank 1 Chip-Select Timing Register
- `FMC_BWTR1` - SRAM/NOR Flash Bank 1 Write Timing Register

B.6 Analogue-to-Digital Converter (ADC)

The Analogue-to-Digital Converter (ADC) is configured by the following 32-bit registers:

- `ADC_SR` - ADC Status Register
- `ADC_CR1` - ADC Control Register 1
- `ADC_CR2` - ADC Control Register 2
- `ADC_SMPR1` - ADC Sample Time Register 1
- `ADC_SMPR2` - ADC Sample Time Register 2
- `ADC_JOFRx` - ADC Sample Time Register 2
- `ADC_HTR` - ADC Watchdog Higher Threshold Register
- `ADC_LTR` - ADC Watchdog Lower Threshold Register
- `ADC_SQR1` - ADC Regular Sequence Register 1
- `ADC_SQR2` - ADC Regular Sequence Register 2
- `ADC_SQR3` - ADC Regular Sequence Register 3
- `ADC_JSQR` - ADC Injected Sequence Register
- `ADC_JDRx` - ADC Injected Data Register x
- `ADC_DR` - ADC Regular Data Register
- `ADC_CSR` - ADC Common Status Register
- `ADC_CCR` - ADC Common Control Register
- `ADC_CDR` - ADC Common Regular Data for Dual and Triple Modes

where x = 1, 2, 3, 4.

B.7 Direct Memory Access (DMA) Controllers

The Direct Memory Access (DMA) controller is configured by the following 32-bit registers:

- `DMA_SxCR` - DMA Stream x Configuration Register
- `DMA_SxNDTR` - DMA Stream x Number of Data Register
- `DMA_SxPAR` - DMA Stream x Peripheral Address Register
- `DMA_SxM0AR` - DMA Stream x Memory 0 Address Register

- DMA_SxM1AR - DMA Stream x Memory 1 Address Register
- DMA_SxFCR - DMA Stream x FIFO Control Register
- DMA_LISR - DMA Low Interrupt Status Register
- DMA_HISR - DMA High Interrupt Register
- DMA_LIFCR - DMA Low Interrupt Flag Clear Register
- DMA_HIFCR - DMA High Interrupt Flag Clear Register

where x = 0, 1, 2, 3, 4, 5, 6, 7.

B.8 LCD-TFT Display Controller (LTDC)

The LCD-TFT Display Controller (LTDC) is configured by the following 32-bit registers:

- LTDC_SSCR - LTDC Synchronization Size Configuration Register
- LTDC_BPCR - LTDC Back Porch Configuration Register
- LTDC_AWCR - LTDC Active Width Configuration Register
- LTDC_TWCR - LTDC Total Width Configuration Register
- LTDC_GCR - LTDC Global Control Register
- LTDC_SRCR - LTDC Shadow Reload Configuration Register
- LTDC_BCCR - LTDC Background Colour Configuration Register
- LTDC_IER - LTDC Interrupt Enable Register
- LTDC_ISR - LTDC Interrupt Status Register
- LTDC_ICR - LTDC Interrupt Clear Register
- LTDC_LIPCR - LTDC Line Interrupt Position Configuration Register
- LTDC_CPSR - LTDC Current Position Status Register
- LTDC_CDSR - LTDC Current Display Status Register
- LTDC_LxCR - LTDC Layer x Control Register
- LTDC_LxWHPCR - LTDC Layer x Window Horizontal Position Configuration Register
- LTDC_LxWVPCR - LTDC Layer x Window Vertical Position Configuration Register
- LTDC_LxCKCR - LTDC Layer x Colour Keying Configuration Register

- LTDC_LxPFCR - LTDC Layer x Pixel Format Configuration Register
- LTDC_LxCACR - LTDC Layer x Constant Alpha Configuration Register
- LTDC_LxDCCR - LTDC Layer x Default Colour Configuration Register
- LTDC_LxBFCR - LTDC Layer x Blending Factors Configuration Register
- LTDC_LxCFBAR - LTDC Layer x Colour Frame Buffer Address Register
- LTDC_LxCFBLR - LTDC Layer x Colour Frame Buffer Length Register
- LTDC_LxCFBLNR - LTDC Layer x Colour Frame Buffer Line Number Register
- LTDC_LxCLUTWR - LTDC Layer x CLUT Write Register

where x = 1, 2.

B.9 Inter-Integrated Circuit (I^2C) Interface

The Inter-Integrated Circuit (I^2C) Interface is configured by the following 32-bit registers:

- I2C_CR1 - I2C Control Register 1
- I2C_CR2 - I2C Control Register 2
- I2C_OAR1 - I2C Own Address Register 1
- I2C_OAR2 - I2C Own Address Register 2
- I2C_DR - I2C Data Register
- I2C_SR1 - I2C Status Register 1
- I2C_SR2 - I2C Status Register 2
- I2C_CCR - I2C Clock Control Register
- I2C_TRISE - I2C Time Rise Register
- I2C_FLTR - I2C Filter Register

B.10 Serial Peripheral Interface (SPI)

The Serial Peripheral Interface (SPI) is configured by the following 32-bit registers:

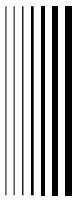
- SPI_CR1 - SPI Control Register 1
- SPI_CR2 - SPI Control Register 2

- `SPI_SR` - SPI Status Register
- `SPI_DR` - SPI Data Register
- `SPI_CRCPR` - SPI CRC Polynomial Register
- `SPI_RXCRCR` - SPI RX CRC Register
- `SPI_TXCRCR` - SPI TX CRC Register
- `SPI_I2SCFGR` - SPI/I2S Configuration Register

B.11 Inter-IC Sound (I^2S) Interface

The Inter-IC Sound (I^2S) Interface is configured by the following 32-bit registers:

- `SPI_CR2` - SPI Control Register 2
- `SPI_SR` - SPI Status Register
- `SPI_DR` - SPI Data Register
- `SPI_I2SCFGR` - SPI/I2S Configuration Register
- `SPI_I2SPR` - I2S Prescaler Register



Bibliography

[1] Klark Teknik. DN360 Graphic Equaliser - Operators Manual. https://media.music-group.com/media/PLM/data/docs/P0AGV/DN360_P0AGV_MANUAL_EN.pdf, August 1999.

[2] D.M. Howard and J. Angus. *Acoustics and Psychoacoustics*. Focal, 2009.

[3] STMicroelectronics. RM0090 - STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced ARM-based 32-bit MCUs. http://www.st.com/st-web-ui/static/active/en/resource/technical/document/reference_misc/DM00031020.pdf, July 2015. Rev. 10.

[4] ITead Studio. Datasheet for ITDB02-3.2S. ftp://imall.iteadstudio.com/IM120419005_ITDB02_3.2S/DS_IM120419005_ITDB02_3.2S.pdf, 4 2012. Rev. 1.

[5] Alliance Memory Inc. AS6C8016 - 512K X 16 BIT SUPER LOW POWER CMOS SRAM. http://www.alliancememory.com/pdf/AS6C8016.pdf, November 2007. Rev. 1.0.

[6] Alliance Memory Inc. AS4C8M16S - 128M - 8M x 16 bit Synchronous DRAM (SDRAM). http://www.alliancememory.com/pdf/dram/128M-AS4C8M16S.pdf, February 2014. Rev. 2.

[7] Cirrus Logic. CS4272 - 24-Bit, 192 kHz Stereo Audio CODEC. http://www.cirrus.com/en/pubs/proDatasheet/CS4272_F1.pdf, August 2005. Rev. F1.

[8] Cirrus Logic. AN421 - Analog Input Buffer Architectures. https://www.cirrus.com/en/pubs/appNote/an241-1.pdf, October 2003. Rev. 1.

[9] Riecktron Embedded Solutions. Breakout Board for ADMP401 MEMS Microphone. https://www.riecktron.co.za/en/image/cache/data/products/09868-01-200x200.jpg.

[10] K. Jack. *Video Demystified: A Handbook for the Digital Engineer*. Demystifying technology series. LLH Technology Pub., 2001.

[11] J.H. McClellan, R.W. Schafer, and M.A. Yoder. *Signal Processing First*. Pearson Education International, 2003.

[12] STMicroelectronics. DS9405 - ARM Cortex-M4 32b MCU+FPU, 225DMIPS, up to 2MB Flash/256+4KB RAM, USB OTG HS/FS, Ethernet, 17 TIMs, 3 ADCs, 20 comm. interfaces, camera & LCD-TFT. `http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00071990.pdf`, November 2015. Rev. 7.

[13] STMicroelectronics. DS8597 - ARM Cortex-M4 32b MCU+FPU, 210DMIPS, up to 1MB Flash/192+4KB RAM, USB OTG HS/FS, Ethernet, 17 TIMs, 3 ADCs, 15 comm. interfaces & camera. `http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00037051.pdf`, October 2015. Rev. 6.

[14] Analog Devices. ADMP401 - Omnidirectional Microphone with Bottom Port and Analog Output. `http://www.analog.com/media/en/technical-documentation/obsolete-data-sheets/ADMP401.pdf`, July 2012. Rev. E.

[15] Klark Teknik. DN6000 - Real Time Audio Analyser - User Manual. `http://bee.mif.pg.gda.pl/ciasteczkowypotwor/stage/Klark%20Teknik/DN6000%20Manual.pdf`, April 2002.

[16] Gold Line. DSP30 Series Analyzers - User Manual. `http://www.gold-line.com/pdf/manual/p_dsp30.pdf`, February 2008.

[17] Meyer Sound Laboratories. SIM 3 Audio Analyzer System - User Guide. `http://www.meyersound.com/pdf/products/integration_tools/SIM3_UG.pdf`, December 2005.

[18] A.V. Oppenheim and R.W. Schafer. *Discrete-Time Signal Processing*. Pearson, 2010.

[19] M.R. Schroeder. *Fractals, Chaos, Power Laws: Minutes from an Infinite Paradise*. W.H. Freeman, 1991.

[20] D. Davis, E. Patronis, and P. Brown. *Sound System Engineering 4e*. Taylor & Francis, 2013.

[21] D.B. Keele. The Design and Use of a Simple Pseudo Random Pink-Noise Generator. `http://www.xlrtechs.com/dbkeele.com/PDF/Keele%20%281973-01%20AES%20Published%29%20-%20Simple%20PR%20Pink%20Noise%20Gen.pdf`, January 1973.

[22] ANSI S1.11: Specifications for Octave-Band and Fractional-Octave-Band Analog and Digital Filters. `https://law.resource.org/pub/us/cfr/ibr/002/ansi.s1.11.2004.pdf`, February 2004.

[23] C. Sujatha. *Vibration And Acoustics*. McGraw-Hill Education (India) Pvt Limited, 2010.

[24] W.W. Smith and J.M. Smith. *Handbook of real-time fast Fourier transforms: algorithms to product testing*. IEEE Press, 1995.

[25] F.J. Harris. On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform. `https://www.utdallas.edu/~cpb021000/EE%204361/Great%20DSP%20Papers/Harris%20on%20Windows.pdf`, January 1978.

[26] W.H. Press. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007.

[27] U. Zölzer. *Digital Audio Signal Processing*. Wiley, 2008.

[28] ANSI American National Standards Institute. ANSI S1.43: Specifications for Integrating-Averaging Sound Level Meters. `https://law.resource.org/pub/us/cfr/ibr/002/ansi.s1.4.1983.pdf`, June 1997.

[29] M.W. Dr and J. McDonough. *Distant Speech Recognition*. Wiley, 2009.

[30] IS:15575-1 - Electroacoustics - Sound Level Meters - Part 1 - Specifications. `https://law.resource.org/pub/in/bis/S04/is.15575.1.2005.pdf`, July 2005.

[31] Texas Instruments. Implementing Fast Fourier Transform Algorithms of Real-Valued Sequences With the TMS320 DSP Platform - Application Report. `http://www.ti.com/lit/an/spra291/spra291.pdf`, August 2001.

[32] R.G. Lyons. *Understanding Digital Signal Processing*. Prentice Hall professional technical reference. Prentice Hall/PTR, 2004.

[33] W.L. Briggs and V.E. Henson. *The DFT: An Owners' Manual for the Discrete Fourier Transform*. SIAM e-books. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 1995.

[34] B. McCarthy. *Sound Systems: Design and Optimization : Modern Techniques and Tools for Sound System Design and Alignment*. Images of America. Focal, 2007.

[35] G. Ballou. *Handbook for Sound Engineers*. Audio Engineering Society Presents Series. Focal, 2008.

[36] E.C. Ifeachor and B.W. Jervis. *Digital Signal Processing: A Practical Approach*. Electronic systems engineering series. Prentice Hall, 2002.

[37] ON Semiconductor. NCP1117LP - 1.0A Low-Dropout Positive Fixed and Adjustable Voltage Regulators. `http://www.onsemi.com/pub_link/Collateral/NCP1117LP-D.PDF`, December 2014. Rev. 4.

[38] Diodes Incorporated. AP1117 - 1A Dropout Positive Adjustable or Fixed-Mode Regulator. `http://www.diodes.com/_files/datasheets/AP1117.pdf`, March 2015. Rev. 24.

[39] Texas Instruments. TLV70018 - TLV700 200-mA, Low-IQ, Low-Dropout Regulator for Portable Devices. http://www.ti.com/lit/ds/symlink/tlv700.pdf, April 2015. Rev. E.

[40] STMicroelectronics. AN2867 - Oscillator design guide for STM8S, STM8A and STM32 microcontrollers. http://www.st.com/web/en/resource/technical/document/application_note/CD00221665.pdf, August 2015. Rev. 10.

[41] STMicroelectronics. PM0214 - STM32F3 and STM32F4 Series Cortex-M4 programming misc. http://www.st.com/st-web-ui/static/active/en/resource/technical/document/programming_misc/DM00046982.pdf, May 2014. Rev. 4.

[42] LTD Success Electronics Co. S95160 - LCD Module Specifications. ftp://imall.iteadstudio.com/LCD_Panel/IM120906008/SPE_IM120906008_FPC.pdf, July 2007. Rev. 3.

[43] Solomon Systech Limited. SSD1289 - TFT LCD Controller Driver integrated Power Circuit, Gate and Source Driver with built-in RAM, April 2007. Rev. 1.3.

[44] Inc. Motorola. S12SPIV3/D - SPI Block Guide, February 2003. Rev. 3.06.

[45] LTD Shenzhen Xptek Technology Co. XPT2046 - Touch Screen Controller. https://ldm-systems.ru/f/doc/catalog/HY-TFT-2,8/XPT2046.pdf, 5 2007. Rev. 2.8.

[46] NXP Semiconductors. UM10204 - I2C-Bus Specification and User misc. http://www.nxp.com/documents/user_manual/UM10204.pdf, April 2014. Rev. 6.

[47] Philips [NXP] Semiconductors. I2SBUS - I2S Bus Specification. https://web.archive.org/web/20060702004954/http://www.semiconductors.philips.com/acrobat_download/various/I2SBUS.pdf, June 1996.

[48] Texas Instruments. SPMA042B - Dual-SPI Emulating I2S on Tiva C Series TM4C123x MCUs. http://www.ti.com/lit/an/spma042b/spma042b.pdf, June 2013.

[49] Microchip Technology. DS22194E - MCP661/2/3/5 - 60MHz, 6mA Op Amps. http://ww1.microchip.com/downloads/en/DeviceDoc/20002194E.pdf, July 2014. Rev. E.

[50] Cirrus Logic. CDB4272 - Evaluation Board For CS4272. http://www.cirrus.com/en/pubs/rdDatasheet/CDB4272-2.pdf, September 2003.

[51] Cirrus Logic. AN048 - Design Notes for a 2-Pole Filter with Differential Input. https://www.cirrus.com/en/pubs/appNote/AN048Rev2.pdf, March 2003. Rev. 2.

[52] InvenSense. INMP401 - Omnidirectional Microphone with Bottom Port and Analog Output. http://43zrtwysvxb2gf29r5o0athu.wpengine.netdna-cdn.com/wp-content/uploads/2015/02/INMP401.pdf, May 2014. Rev. 1.1.

[53] TDK EPCOS. C928 - MEMS Microphone Product Brief. http://en.tdk.eu/blob/731084/download/1/mems-c928-pb.pdf, November 2013.

[54] STMicroelectronics. MP34DT01 - MEMS audio sensor omnidirectional digital microphone. http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00039779.pdf, February 2015. Rev. 12.

[55] STMicroelectronics. DB1275 - ST-LINK/V2 in-circuit debugger/programmer for STM8 and STM32. http://www.st.com/st-web-ui/static/active/en/resource/technical/document/data_brief/DM00027105.pdf, September 2012. Rev. 3.

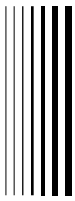
[56] STMicroelectronics. UM0412 - Getting started with DfuSe USB device firmware upgrade STMicroelectronics extension. http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/CD00155676.pdf, July 2009. Rev. 4.

[57] ARM Holdings. Cortex Microcontroller Software Interface Standard. http://www.keil.com/pack/doc/CMSIS/General/html/index.html, October 2015. Ver. 4.5.0.

[58] Tannoy Ltd. 501a/601a Reveal Active - Operator's Manual. http://www.tannoystudio.com/media/1140/reveal-active-manual-lr2010331.pdf, March 2010.

[59] Rane Corporation. RaneGain - RainGain Generator & Transducer Data Sheet. http://www.rane.com/pdf/old/ranegaindat.pdf, March 2011.

[60] Gold Line. PN2/PN2W/PN3A/PN3B Audio Noise Source - Operator's Manual. http://www.gold-line.com/pdf/manual/pa_pn2_3.pdf, February 2008.



List of Electronic Vendors

Alliance Memory http://www.alliancememory.com/.

Analog Devices Founded in 1965. http://www.analog.com/.

ARM Holdings Founded in 1990. http://www.arm.com/.

Atmel Corporation Founded in 1984. http://www.atmel.com/.

Avnet South Africa Founded in 1996. http://www.avnet.co.za/.

CadSoft Computer Founded in 1988. http://www.cadsoftusa.com/.

Cirrus Logic Founded in 1981. http://www.cirrus.com/.

Communica Founded in 1977. http://www.communica.co.za/.

Dallas Semiconductor Founded in 1984 and acquired by Maxim Integrated in 2001.

Diodes Incorporated Founded in 1959. http://www.diodes.com/.

Freescale Semiconductor Founded in 2004 and formally a division of Motorola. http://www.freescale.com/.

Gold Line Founded in 1961. http://www.gold-line.com/.

Infineon Technologies Founded in 1999. http://www.infineon.com/.

InvenSense Founded in 2003. http://www.invensense.com/.

ITead Studio http://www.itead.cc/.

Keil Founded in 1982. http://www.keil.com/.

Klark Teknik Founded in 1974. http://www.klarkteknik.com/.

Maxim Integrated Products Founded in 1983. http://www.maximintegrated.com/.

Meyer Sound Laboratories Founded in 1979. http://www.meyersound.com/.

Micro Robotics https://www.robotics.org.za/.

Microchip Technology Founded in 1989. http://www.microchip.com/.

Microtronix Manufacturing Founded in 1993. http://www.microtronix.co.za/.

Motorola Founded in 1928 and defunct in 2011.

Netram Technologies http://www.netram.co.za/.

NXP Semiconductors Founded in 2006 and formerly a division of Philips. http://www.nxp.com/.

ON Semiconductor Founded in 1999. http://www.onsemi.com/.

Philips Founded in 1891. http://www.philips.com/.

Rabtron Founded in 1991. http://www.rabtron.co.za/.

Rane Corporation Founded in 1981. http://www.rane.com/.

Riecktron - Embedded Solutions Founded in 2005. https://www.riecktron.co.za/.

RS-Components (SA) http://za.rs-online.com/web/.

Shenzhen Xptek Technology Limited http://www.xptek.cn/en/.

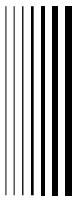
Soloman Systech http://www.solomon-systech.com/.

STMicroelectronics Founded in 1957. http://www.st.com/.

Tannoy Ltd Founded in 1926. http://www.tannoy.com/.

TDK-EPCOS Corporation Founded in 2009. http://www.tdk-epc.us/.

Texas Instruments Founded in 1930. http://www.ti.com/.



Acronyms

A/D Analogue-to-Digital.

ADC Analogue-to-Digital Converter.

AHB Advanced High-Performance Bus (STM32).

ANSI American National Specification Institute.

APB Advanced Peripheral Bus (STM32).

ARM Acorn RISC Machine.

BIS Bureau of Indian Standards.

BOM Bill of Materials.

CAN Controller Area Network (STM32).

CCLK Control Clock (I^2C - CS4272).

CDIN Control Data In (I^2C - CS4272).

CFFT Complex Fast Fourier Transform.

CISC Complex Instruction Set Computer (or Computing).

CMOS Complementary Metal-Oxide Semiconductor.

CODEC **Co**der-**Dec**oder.

CPU Central Processing Unit.

CRC Cyclic Redundancy Check (STM32).

CRT Cathode Ray Tube.

CRYP Cryptographic Processor (STM32).

CTFT Continuous-Time Fourier Transform.

D/A Digital-to-Analogue.

DAC Digital-to-Analogue Converter.

DCMI Digital Camera Interface (STM32).

DDC Display Data Channel.

DFT Discrete Fourier Transform.

DFU Device Firmware Update.

DMA Direct Memory Access.

DMA2D Chrom-Art Accelerator^{TM} Controller (STM32).

DSP Digital Signal Processor (or Processing).

DTFT Discrete-Time Fourier Transform.

EAGLE Easily Applicable Graphical Layout Editor.

ETH Ethernet (STM32).

EXTI External Interrupt/Event Controller (STM32).

FFT Fast Fourier Transform.

Fm Fast-Mode or Fast-Speed (I^2C).

FMC Flexible Memory Controller (STM32).

FMSC Flexible Static Memory Controller (STM32).

FPU Floating Point Unit.

FS Full-Speed (USB).

FSF Free Software Foundation.

GCC GNU Complier Collection.

GDB GNU Debugger.

GDDRAM Graphic Display Data RAM.

GNU **G**NU is **N**ot **U**nix. It is a recursive acronym.

GPIO General Purpose Input/Output (STM32).

GPL General Public License.

HASH Hash Processor (STM32).

HCLGA Holed Cap Land Grid Array.

HCLK CPU Clock.

HS High-Speed (USB).

HSB High-Speed Bus (STM32).

I²C Inter-Integrated Circuit.

I²S Inter-IC Sound.

IC Integrated Circuit.

IDP Integrated Development Platform.

IRQ Interrupt Request.

IWDG Independent Watchdog (STM32).

JTAG Joint Test Action Group (Debugging Protocol).

LCD Liquid Crystal Display.

LGA Land Grid Array.

LGPL Lesser General Public License.

LQFP Low Profile Quad Flat Package.

LR Link Register.

LSB Least Significant Bit.

LTDC LCD-TFT Controller (STM32).

MAC Media Access Controller (STM32).

MATLAB **Mat**rix **Lab**oratory.

MCU Microcontroller.

MEMS Micro-Electro-Mechanical System.

MISO Master Input / Slave Output.

MOSI Master Ouput / Slave Input.

MPU Memory Protection Unit.

MSB Most Significant Bit.

NSS Slave Select.

NVIC Nested Vectored Interrupt Controller.

NXP Next Experience.

OTG USB On-The-Go (USB).

PBA Peripheral Bus A (STM32).

PBB Peripheral Bus B (STM32).

PC Personal Computer.

PC Program Counter.

PCB Printed Circuit Board.

PCM Pulse Code Modulation.

PDM Pulse Density Modulation.

PLL Phased Lock Loop.

PM Power Manager.

PSR Program Status Register.

PSRAM Pseudo-Static Random Access Memory.

PWR Power Controller (STM32).

QFP Quad Flat Package.

RAM Random Access Memory.

RCC Reset and Clock Controller (STM32).

RGB Red Green Blue.

RISC Reduced Instruction Set Computer (or Computing).

RMS Root Mean Squared.

RNG Random Number Generator (STM32).

RTAA Real-Time Audio Analyser.

RTC Real-time Clock (STM32).

SAI Serial Audio Interface (STM32).

SAR Successive Approximation Register.

SCB System Controller Block.

SCK Serial Clock.

SCL Serial Clock line or register (I^2C).

SDA Serial Data line or register (I^2C).

SDIO Secure Digital Input Output Interface (STM32).

SDRAM Synchronous Dynamic Random Access Memory.

SIL Sound Intensity Level.

Sm Standard-Mode or Standard-Speed (I^2C).

SNR Signal to Noise Ratio.

SOIC Small Outline Integrated Circuit.

SOT Small Outline Transistor.

SP Stack Pointer.

SPI Serial Peripheral Interface.

SPL Sound Pressure Level.

SRAM Static Random Access Memory.

SSOP Shrink Small Outline Package.

STK SysTick Timer.

SWD Serial Wire Debug (Debugging Protocol).

SWL Sound Power Level.

SYSCFG System Configuration Controller (STM32).

SYSCLK System Clock.

TC Timer/Counter.

TFT Thin-Film Transistor.

THD Total Harmonic Distortion.

TSOP Thin Small Outline Package.

TSSOP Thin Shrink Small Outline Package.

USART Universal Synchronous Asynchronous Receiver Transmitter (STM32).

USB Universal Serial Bus.

VGA Video Graphics Array.

VIA Vertical Interconnect Access.

WWDG Window Watchdog (STM32).