Music Driven Graphical Visualization System

By

Glenn Bruner

Senior Project Report

ELECTRICAL ENGINEERING DEPARTMENT

California Polytechnic State University

San Luis Obispo

2012

TABLE OF CONTENTS

PAGE

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to first off express my deep thanks to my wife Georgette and my family for all countless moments of sacrifice they had to endure while I pursued my engineering degree.

Next I would like to express thanks to Matthias Domin (www.mdgames.de) and Scott Walters for their technical support during the development and integration of this project. Their background and knowledge of the Jaguar system was instrumental in over-coming several issues during development of this system.

I would like to acknowledge the team at Atari who developed the Jaguar game system. Of particular I would like to give special mention to John Mathieson and Tim Dunn (two of the principle designers of the Jaguar's GPU/DSP processors and system) for their willingness to answer technical questions regarding the Jaguar system despite its 1996 departure from the consumer market. I would like to extend a similar acknowledgement to Jeff Minter and Ian Bennett authors of the Virtual Light Machine (VLM) software that became an integral part of this project.

And lastly I would like to honor my father, Ronald L. Bruner, who sparked my in interest in electronics, computers, engineering and math which has led me to the pursuit of my Electrical Engineering degree.

ABSTRACT

In today's society, we are able to enjoy listening to music on an assortment of high-tech devices such as home computers, or multimedia players, or personal audio players. A common characteristic of typical audio media players is that they lack the ability to provide visual stimulation in addition to the provided audio stimulus. Large public events, such as concerts, could be enhanced if they included visual effects in addition to audio stimulation. The goal of this project is to provide a graphical visualization that is automatically synchronized to an audio signal with a goal of enhancing the overall musical experience.

This system features a self-contained hardware and software solution that facilitates a quick setup time and is easy to transport. This system accepts stereo audio signals using a simple cable connection to a music player or audio processing equipment. The music visualizer portion of this system uses existing technology to provide a low-cost solution to satisfy the need and features predefined visual settings, or banks, along with an edit mode to allow customization of any of the pre-defined visualizations. The music visualizer employs an intuitive to use controller to allow easy operation and switching between animations during a musical event.

I. INTRODUCTION

This report covers the details concerning the design and build of the Music Driven Graphical Visualization System. I chose this system to be my senior design project in Summer 2011. I chose to do this project because of my lifelong passion for video game systems and hardware design. From the early days of when Atari created the video / arcade gaming industry I was inspired by the engineers who developed innovative and entertaining electronics that have reshaped our society. This eventually led me to pursue a degree in engineering.

One of the goals of this project was to build a visualization system that could provide a low cost solution for people who want to incorporate graphical and entertaining displays to a party or large public event. This project's focus has been to develop a cartridge that would replace the compact-disc (CD) add-on unit available for the Atari Jaguar gaming system. The CD add-on contained a built-in music visualizer function to enhance the user's experience with the gaming system. But this add-on to the game system was very limited in its production numbers; less than 20,000 units were manufactured compared to the 250,000 game systems produced. This makes the CD add-on hard to locate and expensive to acquire due to its rarity. In addition, the CD add-on could only accept music in the form of a CD media which restricts to using only pre-recorded music. This project achieves this goal by eliminating the CD unit and provides an audio input port that opens up the system to accept from a wider variety of sources for audio information that is used to drive a graphical visualization.

II. BACKGROUND

BRIEF HISTORY OF MUSIC DRIVEN VISUALIZATION SYSTEMS

In 1976, Atari developed and marketed a system to perform graphical visualization animation to music. The product was called Video Music and was invented by Robert J. Brown [6]. Allan Alcorn, Pong Inventor and Chief Engineer for Atari, was asked during a recent email exchange the question of "how did the product idea originate?" Mr. Alcorn responded with this, "*We had just finished shipping Home Pong containing our first custom chip. That chip was created by a small team, Harold Lee, Bob Brown, Carl Nielson and myself. The idea for Video Music came from Bob Brown with support from Harold. They were passionate about it so we did it. It was somewhat ironic that it came from Bob Brown as he seemed to be the most conservative "adult" engineer we had*." Despite the fact the product was unique and innovative, it was considered a failed product due to its limited and short product life and lack of interest from distributors at the Consumer Electronics Show.

The Atari Jaguar game console marketed by Atari Corporation in the mid-1990's represented the next attempt to include a music visualization feature. Known as the Virtual Light Machine (VLM), this feature was built into the CD unit add-on later released for the game console. VLM was principally the work of Jeff Minter and Ian Bennett [20]. The CD unit interfaced with the game console through the cartridge port of the system. The VLM software employs a Fast Fourier Transformation (FFT) technique to sample the music played through the CD unit. Due to the timing of the release of the first Sony Playstation and cash strapped condition at Atari, the VLM had a limited market impact.

Figure 1. Jaguar with CD add-on unit

Jeff Minter later took the VLM software and developed VLM-2 for the Nuon DVD chip [20], an integrated circuit processor developed by former Atari Jaguar engineers. Once again, the product saw

limited market impact because the Nuon media processor was never popular with consumers. Now, modern music visualizer's are typically found embedded into PC's, Mac's and video game consoles and is worth noting that Jeff Minter programmed the Neon visualizer software currently used in the Microsoft Xbox 360 game console [20].

III. REQUIREMENTS

The system comprises of an integrated hardware and software solution that provides a graphical visualization that responds to a music input. The system accepts audio input in the form of left and right channels, known as stereo, that fall within the frequency range of 20-20,000 Hertz which corresponds to the frequency range of human hearing and thus recorded music. The system audio input operates with a high input impedance to prevent distortion of the audio signal by loading affects. The video output of the system supports a variety of television signal formats such as NTSC channel 3/4 RF, composite video, Super-Video and RGB which are formats found on any modern television or projection systems. The system uses a common 3.5mm connector that is securely mounted to the printed circuit board (PCB) for audio input. The system provides the user with a selection of 81 predefined visualization animation options that support quick and easy changes using an intuitive joypad interface that contains a familiar telephone keypad. The system also contains an option for non-volatile storage memory which is currently unsupported due to software integration issues. And lastly, the system uses an external wall-mounted transformer that can either provide 115VAC for operation in the United States/Canada or 220V for operation in Europe that are commonly found in home and commercial environments.

In order to provide maximum marketability for the system, the system supports several important marketing requirements. The system is light-weight, easy to transport, relatively affordable to own and operate, and requires minimal setup and operation procedures. The pre-configured visualizations support a wide range of music styles such as classical, rock, R&B, Jazz and hip-hop. The

system also includes a custom visualization editor that can be used to edit any of the 81 preconfigured visualizations. But as previously noted, the ability to save that custom visualization is unsupported feature. And above all, the system supports a concept of sustainability of reusing existing technology to reduce generating more electronic waste and extend the useful life of an existing technology [14].

Figure 2 and Table 1 provide a level-0 functional decomposition of the system. Figure 2 shows the inputs on the left and the outputs on the right. Table 1 provides details about the inputs and outputs and text to describe the function of the system.

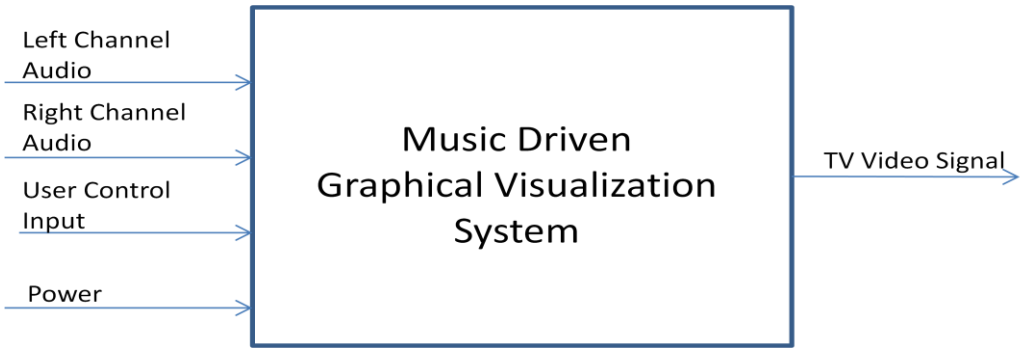


Figure 2. Level 0 Design Block Diagram

Table 1. Level 0 Design Functional Requirements

Module	**Music Driven Graphical Visualization System**	
Inputs	***Input Signal Name***	***Description***
	Left & Right Channel Audio	Line-audio input signal. Nominal signal amplitude 0.316Vrms (-10dBV). Maximum amplitude of 1.0Vrms (0dBV). Minimum input impedance 10k Ω.
	User Control Input	An encoded signal(s) that provides input to the visualization software to control menu selections and enter parameters for user of system to interact with system.
	Power	110/120V AC rms, 60Hz
Outputs	***Output Signal Name***	***Description***
	TV Video Signal	Frequency modulated analog signal; composite-video (chrominance, luminance and sync) or separated-video (separate chrominance, luminance and sync), RGB (red, green, blue and sync) signal formats; NTSC or PAL broadcast standards
Functionality	The stereo audio channel input signals frequency spectrums are sampled and drive a graphical visualization animation. User input control contains encoded information that represents user commands to interact with the visualization software. User has the ability to navigate menus, readjust parameters when in edit mode, and change predefined visualization displays. TV signal output is compatible with television or overhead projector systems.	

Section IV and APPENDIX B. SPECIFICATIONS AND REQUIREMENTS provide more details regarding the design and the original design requirements and specifications, respectively, which were generated during the initial proposal stage of this project. APPENDIX B. SPECIFICATIONS AND REQUIREMENTS, Table 8 captures the general requirements highlighted in previous paragraph. Additionally these requirements are individually addressed with detailed specifications and the specific marketing requirements they support.

IV. DESIGN

The core of the hardware design is the analog-to-digital conversion (ADC) circuitry. This circuitry is required to simultaneously convert a left and right audio channel into 16-bit digital form and multiplex the audio data over an Inter-Integrated circuit Sound (I2S) bus. This hardware employs an analog front-

end and digital circuit back-end which must be carefully separated to prevent signal interference between the two electrical domains. The fact that the host-platform chosen for this project was limited to a 16-bit I2S bus digital audio input provided a design constraint on available ADC chips to choose from. This constraint was brought to my attention when consulting with John Mathieson, co-designer of the Jaguar's DSP. John replied to my technical inquiry with this, "*looking at the netlist it will capture the first 16 bits after a word strobe edge, which is the MSBs I believe. It goes into an overflow condition after it captures 16 bits which should stop further capture.*" And lastly, the ADC circuit is supported by two 78L05 +5V DC regulators and a collection of external passive components in order to perform its intended function.

The core of the software design has focused on extracting the necessary components of the VLM software from the CD add-on ROM memory and writing a loader program that would transfer the VLM software and user interface code into the main memory of the system. The VLM software is an integral part of the CD add-on and requires the use of the CD front-end user interface in order to properly function for this project. But even with the separation of the CD front-end user interface software code from the CD add-on hardware the software is still functional and stable in its operation with the VLM software.

I disassembled the VLM software in order to gain better understanding about its structure and data organization. One of the design requirements was to incorporate a feature to save user configurations of the VLM visualizations edited by the user of the system. In order to add this feature I required source code for the VLM software and the only way to obtain this source code was to disassemble the code. This disassembly of the VLM software deepened my understanding the inner workings of the VLM software and its needs for dependent code (e.g., CD front-end and CD BIOS). This contributed to the success of writing a functional loader program for the project. But as of this moment

in time, no code alteration has been done regarding the VLM or CD front-end software due to complications found with attempting to incorporate the save feature requirement. The visualization configuration data stored within the software is disjointed and would require extensive rework of the source code to implement a save feature. But a good note to indicate, despite the lack of a save feature, is that all of the software employed by this system occupies only one third of a single 512k byte ROM chip, which is mounted on the cartridge board with the ADC hardware. This provides plenty of room to expand the software in case a coding solution could be engineered to fulfill the save feature requirement.

Figure 3 presents the level 1 functional decomposition of the system design. In particular, Figure 3 shows key system components that support the functionality described by above thru the requirements and level 0 descriptions given in Figure 2 and Table 1 above. The audio processing section contains the component that conditions the incoming audio signals and converts to a digital pulse-code modulated (PCM) data format. The CPU/Graphics Hardware block in Figure 3 represents a generalized architecture diagram for the Jaguar game console. The Jaguar system is the host platform for the music input hardware and VLM software. Engineering specification #7 in Table 8, Appendix B, specified the host platform for this project which did not require the design and build of the CPU/Graphics Hardware block shown in the level 1 diagram. The functional description of the individual subsystems of the host platform is provided in order to help facilitate understanding of how the audio processing and program storage component interface with the host platform. Table 2 through Table 7 provide further details about the level 1 diagram components.

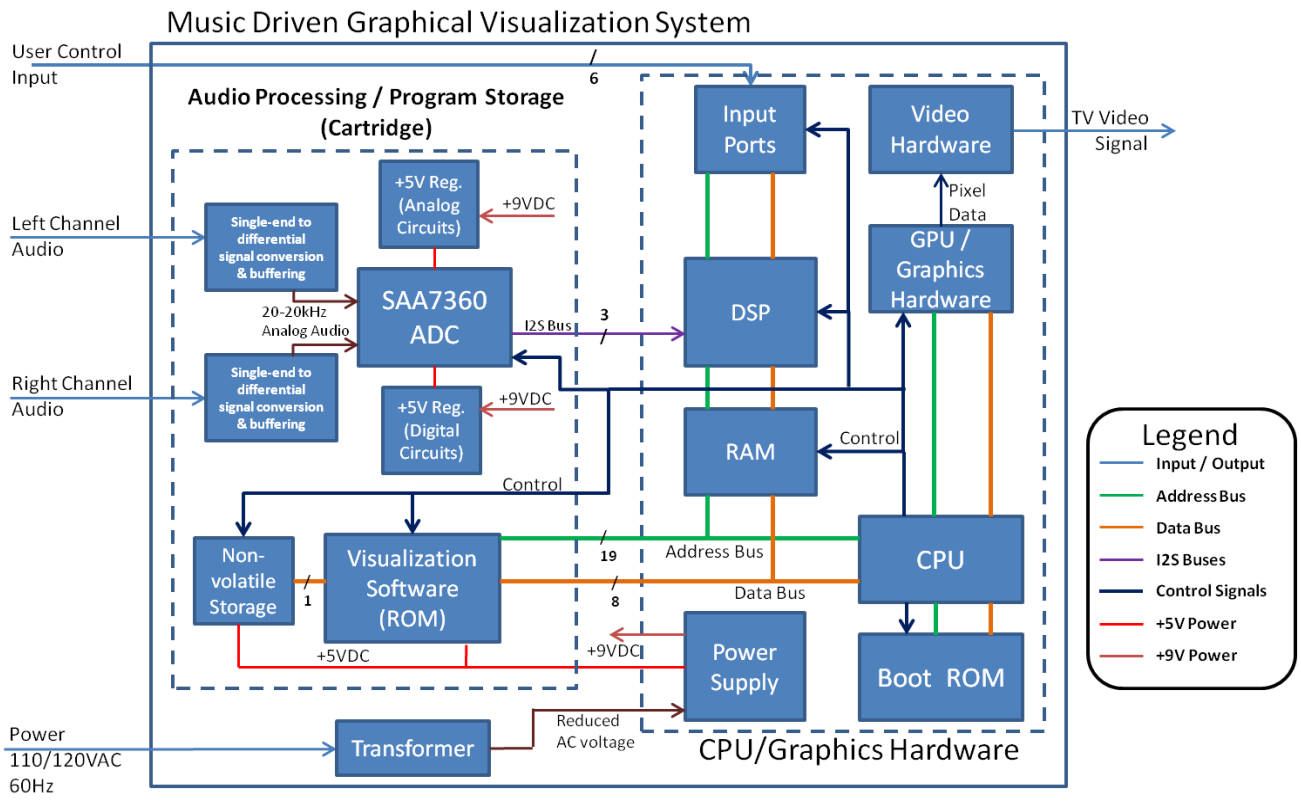


Figure 3. Level 1 Design Block Diagram

Table 2 describes the details about the audio signal processing that converts and buffers the incoming audio data. Table 3 describes the details about the analog-to-digital conversion circuitry. Table 4 describes functionality details required of the visualization software storage ROM and Table 5 describes the functionality for the non-volatile storage memory used for storing parameters associated with the user's custom visualizations.

Table 2. Level 1 Single-end to Differential Signal Conversion Functionality

Module	**Single-end to Differential Signal Conversion (L) & (R)**	
Inputs	***Input Signal Name***	***Description***
	Left & Right Channel Audio	Line-audio input signal. Nominal signal amplitude 0.316Vrms (-10dBV). Maximum amplitude of 1.0Vrms (0dBV). Minimum input impedance 10k Ω.
	DC Power	Single rail +5 VDC voltage.
	Output Signal Name	***Description***
Outputs	20-20kHz Analog Audio	Differential analog audio signal information with amplitude of 1.0Vrms.
Functionality	The stereo audio channel input signals frequency spectrum is reduced to the passband for human hearing range (20 – 20k Hz). The input signal is converted into a differential form before application to the differential analog-to-digital converter circuit.	

Table 3. Level 1 A/D Converter Functionality

Module	**Analog-to-Digital (A/D) Converter**	
Inputs	***Input Signal Name***	***Description***
	20 – 20k Hz Analog Audio Channels	Analog left and right audio differential voltage signal information with maximum amplitude of 1.0Vrms.
	Inter-Integrated circuit Sound (I^2S) Bus (Clock Signals)	Clock signals providing bit clock and Word Select for data sample rate for A/D circuitry [15].
	DC Power	Two +5 VDC voltage rails – analog section and digital section
	Output Signal Name	***Description***
Outputs	I^2S Bus (16-bit Serial Data)	Digital sampled data stream of stereo analog passband signal. Data is pulse-coded modulated alternating pair of digital serial data representing the sampled amplitude level of input audio signal. Input offset voltage is removed. Transferred over Inter-Integrated circuit Sound (I^2S) bus [15].
Functionality	The stereo audio filtered channel input signals are processed thru an analog-to-digital 128x oversampled 3rd order sigma-delta modulated conversion process to produce a 16-bit digital stream of audio data for use by visualization software. The clock inputs provide necessary signals for A/D conversion and serial data transfer to host system.	

Table 4. Level 1 Visualization Software ROM Functionality

Module	**Visualization Software ROM**	
Inputs	***Input Signal Name***	***Description***
	Address Bus	Eighteen digital address lines to address the specific memory location within ROM chip storing visualization software within the cartridge memory space of the Jaguar system.
	Control Signals	Control signal that provides chip select and output enable signal to access stored program code for visualization system.
	DC Power	Single rail +5 VDC voltage.
	Output Signal Name	***Description***
Outputs	Data Bus	Eight digital bus lines for transferring visualization software to host system.
Functionality	ROM hardware used to hold visualization software, CD front-end user interface and necessary loader code. Host platform system will access ROM after boot code has initialized system and load visualization software into main memory for execution by CPU, DSP and GPU processors.	

Table 5. Level 1 Non-volatile Storage Functionality

Module	**Non-volatile Storage**	
Inputs	***Input Signal Name***	***Description***
	General Purpose Input / Output (GPIO) Bus (Serial Data Input)	Serial data input into non-volatile storage memory to store visualization software parameters. Serial length of eight bits.
	GPIO Bus (Control Signals)	Two control signals associated with GPIO bus that provide serial data clock and chip select.
	DC Power	Single rail +5 VDC voltage.
	Output Signal Name	***Description***
Outputs	GPIO Bus (Serial Data Output)	Serial data output from non-volatile storage memory to retrieve stored visualization software parameters. Serial length of eight bits.
Functionality	Non-volatile storage memory used to save a user custom created visualization parameters for later recall by the user. Memory will retain contents when DC power is removed. Memory size requirements will be determined later depending on visualization software storage needs.	

Table 6 describes the basic function of the power rectifier which steps down the input 120VAC and rectifies an unregulated DC voltage level required by the host platform power supply circuitry. Table 7 describes the input and output signals that Jaguar system provide and accept to support the intended function of the system.

Table 6. Level 1 Power Rectifier Functionality

Module	**Power Rectifier**	
Inputs	***Input Signal Name***	***Description***
	Power	110/120V AC rms, 60Hz
	Output Signal Name	***Description***
Outputs	DC voltage	AC voltage stepped down and rectified to 9VDC at 1.2A
Functionality	External AC step down and rectifier power circuitry that connects to host platform and provides unregulated DC voltage input to DC regulation power supply inside host platform. This voltage is used by the two 78LM05 to provide isolated +5V DC power to the SAA7360GP ADC digital and analog sections.	

Table 7. Level 1 CPU/Graphics Hardware Functionality

Module	**Jaguar System (CPU Graphics Hardware) [9]**	
Inputs	***Input Signal Name***	***Description***
	User Control Input	An encoded signal(s) that will provide input to visualization software to control menu selections and enter parameters for user of system to interact with system.
	GPIO Bus (Serial Data Input) [9]	Serial data input from non-volatile storage memory retrieved stored data. Serial length of 8bits.
	I^2S Bus (Serial Data) [15]	Digital sampled data stream of stereo analog passband signal. Data is pulse-coded modulated alternating pair of digital serial data representing the sampled amplitude level of input audio signal. Input offset voltage is removed. Transferred over Inter-Integrated circuit Sound (I^2S) bus.
	Data Bus	Digital bus for transferring data from addressed peripheral to CPU.
	DC Voltage	Unregulated DC voltage for regulated power supply contained within Jaguar system.
	Output Signal Name	***Description***
Outputs	TV Video Signal	Frequency modulated analog signal; composite-video (chrominance, luminance and sync) or separated-video (separate chrominance, luminance and sync), RGB (red, green, blue and sync) signal formats; NTSC or PAL broadcast standards
	GPIO Bus (Serial Data Output)	Serial data output to non-volatile storage memory to data storage. Serial length of eight bits.
	GPIO Bus (Control Signals)	Two control signals associated with GPIO bus that provide serial data clock and chip select.
	I^2S Bus (Clock Signals)	Clock signals providing master clock for A/D circuitry, serial clock for data synchronization and Left/Right clock for data sample rate.
	Address Bus	Digital address bus to specify address of memory location currently under access by host platform CPU.
	Data Bus	Digital bus for transferring data from CPU to addressed peripheral.
	Control Signals	Control systems produced by host platform system to provide chip select, output enable and clock signals for synchronizing the transfer of parallel and serial data within system.
Functionality	The Jaguar host platform will provide user input control input function, GPIO and I^2S busses, data/address bus & control signals for system operation, TV signal generation and power supply for rectifying and regulation of DC power.	

V. TEST PLANS

The core tests for this project evolve around testing of the ADC circuitry to test the functionality of the hardware to provide a digital stream of music data with minimal amount of signal distortion. Concepts were initially developed to test the various hardware design decisions and test the software components individually to ensure they were going to meet requirement performance specifications for eventual integration into the overall system.

Hardware testing begins with checking power supply rails are within design limits. Then the digital bit clock, word select and data output lines on the I2S bus would be measured and compared against the signal specifications found in the SAA7360GP datasheet. Using a signal generator perform a frequency sweep thru the audio passband and into the stopband to characterize the transfer function for the ADC input circuitry to validate that the input signal is band-limited to the audio frequency range.

Software testing of the ROM storage is done on the Jaguar system using cartridge ROM checksum calculating utility and comparing the result against the checksum determine during ROM burning. The boot process is cycled multiple times to ensure software consistently starts up and shows no instability issues. Then VLM and CD front-end code is exercised, with music streaming to the system, to observe that all 81 visualizations can be selected and no anomalous software glitches are observed to occur.

VI. DEVELOPMENT AND CONSTRUCTION

Development began with the search for a suitable ADC chip that can fulfill the requirements and specifications for the system. The first ADC to for design consideration was the CS5330 made by Cirrus Loigc [12]. The CS5330 is an 18-bit stereo ADC that supports I2S bus data streaming and is packaged in a small 8-pin surface mount package. The chip only requires simple lowpass filtering using external operational amplifier circuits. An initial circuit prototype, Figure 4, was assembled to test the feasibility of the ADC to integrate with the Jaguar's I2S bus. I used the suggested circuit provided in [12] for the prototype. The analog circuitry, ADC analog side and MC33202 op-amps were powered with a 7805 regulator powered from the Jaguar's 9V power supply. The digital side was powered using the Jaguar's +5V power supply. The analog input to the chip did function adequately as it provided good isolation for the audio source and a band limited input to the ADC. The chief problem with the circuit was separation between the analog and digital circuits. When the analog and digital buses were connected the digital I2S bus signals were overwhelmed and badly distorted. Since the chip was no longer supported, I was unable to elicit help from Cirrus Logic.

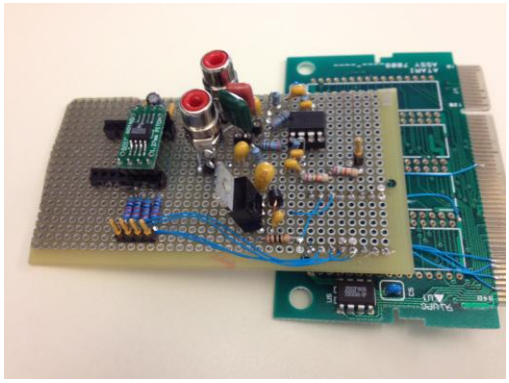


Figure 4. CS5330 Prototype

The next two ADC's considered were the Analog Devices AD1871 [22] and the WM8782 made by Wolfson Microelectronics [21]. The AD1871 provided features to support the needs of the system: 16-bit output word size, I2S bus support, delta-sigma conversion. But drawback to the chip involved finding a way to interface the chip the Jaguar's GPIO interface. The default power-up operational configuration for the AD1871 almost matches the required settings needed for operation with the Jaguar's DSP processor. Additional commanding by the Jaguar's CPU would be required in order to configure the AD1871's registers to which complicates the design. The WM8782 proved to be a better alternative to

the AD1871 as it supported the same features as the AD1871 and CS5330 but was configurable using external pull-up / pull-down resistors. A PCB prototype was designed and ordered due to the small 0.65mm pin spacing on the WM8782 and the fact the project was pushing up against a time constraint to complete a design solution. I attempted to elicit design help from Wolfson but could not get answers to my phone messages and emails. The prototype had similar issues as the CS5330, intense interference between the analog and digital (Figure 5) which I failed to understand and plan for in my PCB layout. Due to the lack of support I was receiving from Wolfson, I abandoned the chip and went with an older chip made by Philips.

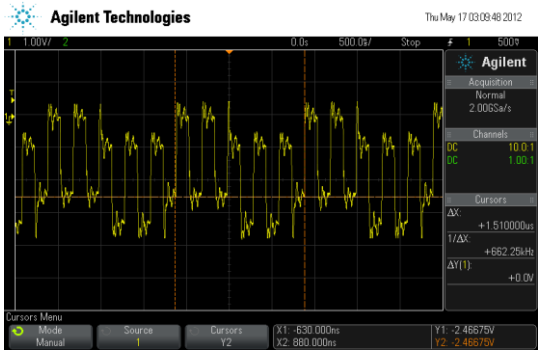


Figure 5. Analog Distortion to Digital Bit Clock

The Philips SAA7360 [11] is the ADC that managed to conform this project into a working and functional system. I discovered this chip on a system test board that was used for repair and refurbishment of Jaguar game systems. Thru my research of the Jaguar system this ADC was used during developmental testing of the Jerry DSP processor. Since the system CD ROM player was developed by Philips Corporation, this ADC is potentially of compatible with the game system. Similar constraint previously noted with the Wolfson chip, there was not enough time to prototype a circuit to verify the performance of the SAA7360. I instead proceeded ahead with designing and manufacture of a PCB to test this ADC. This ADC incorporated operational amplifier circuits for the input end that conveniently help transform the single-end voltage signals into differential voltage for the sigma-delta ADC circuit. The output end is specifically designed to support the I2S bus used by the Jaguar DSP. And hardware configuration of the ADC is completely configurable using external pull-up / pull-down resistors. Lastly, the signals input / output pins on the ADC package for digital and analog can be easily separated on the PCB which contributed to the success of this project.

I used Eagle CAD, distributed by Cadsoft USA, for all PCB development. I used a tutorial provided on Cadsoft's website to learn how to use the software. I then drew a schematic for the intended circuit design (Figure 15 thru Figure 18 in APPENDIX E. CIRCUIT SCHEMATICS) followed by a PCB layout based upon the schematic (Figure 19 thru Figure 22 in APPENDIX F. PC BOARD AND PART LAYOUT). Component library files were drawn for the ADC chip and 3.5mm audio connector chosen for this project for their inclusion in the schematic diagram and PCB layout. The remaining components used were available from the in the Eagle CAD component libraries. The circuit components were manually placed in order to provide clean separation between the digital memory (program / non-volatile) circuits and the ADC circuit. I first placed down the power connections, followed by the ADC component connections and then finished up with the Program / Non-volatile circuit placement. The last feature added was the ground planes for the analog and digital grounds using a polygon shape and area copper fill. The copper fill is a critical feature of the PCB layout to provide good decoupling for analog and digital power supplies, improve ground signal isolation and reduce chances of ground loops forming in the circuit [22].

The VLM software is comprised of Motorola 68000 assembly code for overall management of GPU & DSP reduced instruction set computer (RISC) code components, GPU RISC code is used for graphic generation of the visualizations and DSP RISC runs an FFT code for sampling the digital audio information streaming into the system. Through my dissection of the VLM software I discovered that the VLM required access to the CD Front-End software in the CD ROM boot code along with the CD BIOS code. From this I was able to write a loader program (Listing 1 in Appendix G) that is assembled and linked for operation from cartridge memory space and then turns control over to the VLM software.

To assist with disassembly of the RISC code components, I resurrected a disassembler found on the internet (Listing 2 thru Listing 5 in Appendix G). I modified and added features to the C code to tailor

the disassembly output to make it compatible with the assembler development tools (e.g., MADMAC and SMAC) for the Jaguar system. This improved the disassembly of the VLM software since the debugger's disassembler produced inferior output that would have required much more labor to produce the same disassembly. And it was thru the development of this tool that I had discovered the FFT algorithm that was embedded within the VLM software.

One feature requirement that I was not able to develop was the visualization save feature. Thru my analysis of the VLM software disassembly I found that visualization settings were not conveniently stored in easily identifiable regions of memory. Since the original design of the VLM software did not intend to include an end-user editing feature, this information is haphazardly stored in many different locations from what I can determine from the disassembly. Given enough time I could probably identify where the data is stored but that would have impacted the completion of this project. Consequently, this requirement was sacrificed in order to obtain a working system.

VII. INTEGRATION AND TEST RESULTS

Testing began with the ADC circuit. First test involved testing the analog and digital +5 voltage regulators. A simple DMM measurement validated that the regulator outputs were within datasheet specifications. The analog regulator measured at 4.90V and the digital regulator output measured at 4.98V. The tolerance according to the ST Microelectronics datasheet is ±0.4V while the ADC tolerance is ±0.5V, well within specifications for both chips. And since the voltage is within design limits this indicated to me that both voltage regulators are not significantly loaded by the ADC chip and are working within their design current limits.

Testing then proceeded with viewing the digital signals associated with the CPU clock signal and I2S bus signals. Figure 6 shows the oscilloscope measurement of the CPU Clock signal to the ADC circuit. The frequency measured 13.31MHz with is half of the Jaguar's main crystal oscillator frequency of 26.6MHz. The high voltage level measured at around 5.2V. The ADC expects a high level that is 0.7 times of VDD (minimum of 3.5V) and VDD + 0.5V (maximum of 5.5V), where VDD is the digital +5V supply.

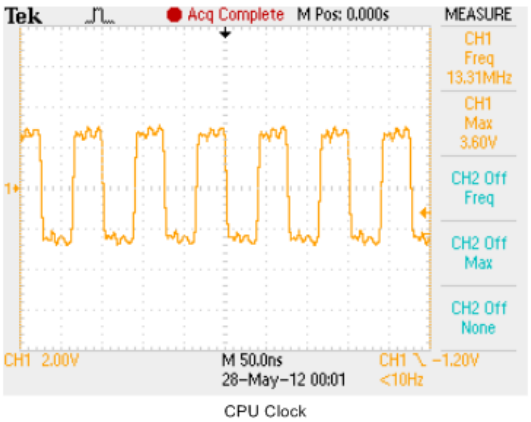
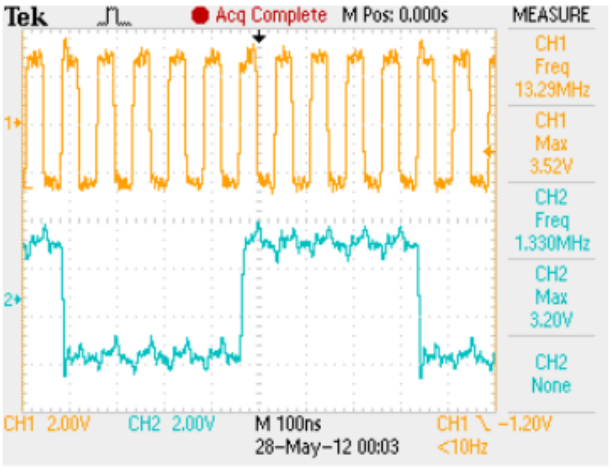


Figure 6. Jaguar CPU Clock Signal Measure

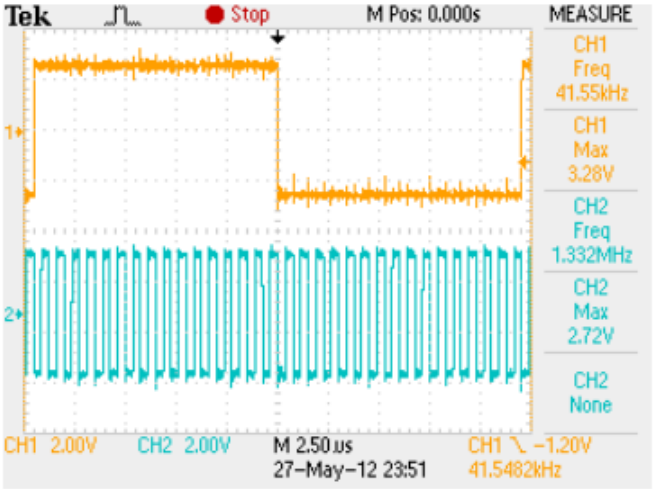
In Figure 7 the I2S serial clock is compared to the CPU Clock. SCLK is operating at one-tenth of CPU clock frequency and shows proper rise and fall synchronization.



CPU Clock (Ch.1) - SCLK (Ch.2)

Figure 7. CPU Clock Vs. I2S Serial Clock (SCLK)

In Figure 8 the I2S word select (SWSI) is compared to the SCLK signal. Each half-cycle of SWSI shows 16 cycles of the serial clock indicating that the ADC is operating in the 16-bit word size mode. The SCLK and SWSI signals are showing good alignment and synchronization.



SWSI (Ch.1) - SCLK (Ch.2)

Figure 8. SWSI Vs. SCLK Measurement

Figure 9 compares the SWSI (word select) signal with the data out (SDO) signal. Similar to the signals above, SWSI and SDO showed proper synchronization and alignment to each other.

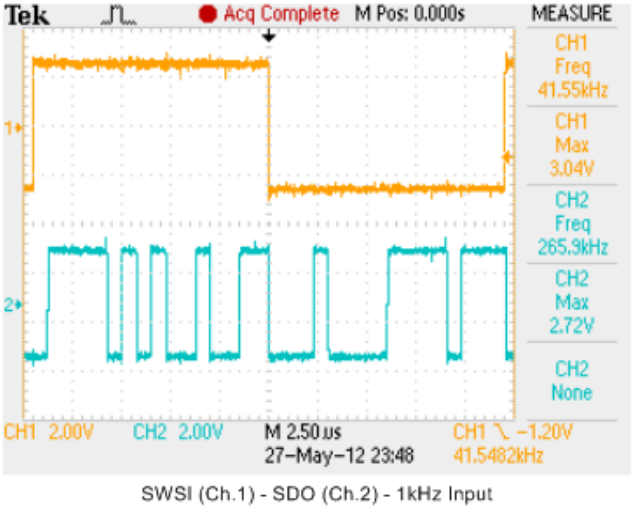
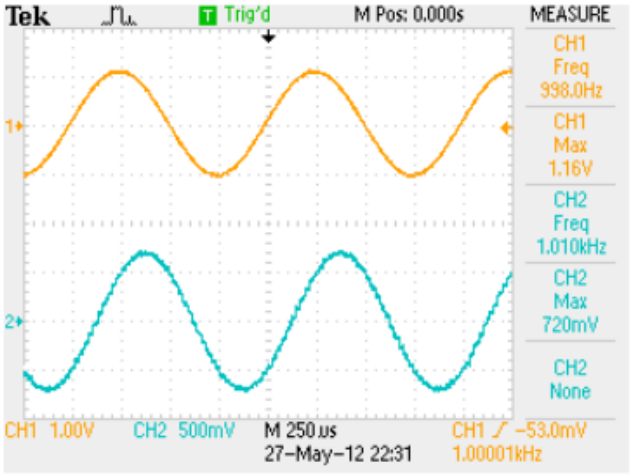


Figure 9. SWSI Vs. SDO Measurement

In order to verify the accuracy of the ADC's conversion I chose to measure the analog audio signal that was generated by the Jaguar's VLM software and DSP hardware. I felt this was a good way to measure both the ADC circuit and the system as a whole. Attempting to measure the digital word produce by the ADC matched the particular frequency input would be an overkill of a measurement since the accuracy of the ADC output was not a design requirement.

Figure 10 shows the measurement of the right channel output of the system with a 1 kHz sine wave input injected into the system. The output signal showed a 12 Hz difference or 1.2% difference from the input which is adequate performance for the system.



Signal Input (R) vs. Output (R)

Figure 10. 1kHz Signal Input Vs. System Output

The next measurement taken was a frequency response of the system, from input to output. Figure 11 shows a plot of the magnitude versus frequency measured for the system. This was to verify that the coverage of the audio spectrum was adequately covered by the ADC circuitry. The output signal spectrum -3dB point is measured at 17.6 kHz showing an adequate coverage of the human hearing range of frequencies. Since the VLM software is configured to trigger off frequencies that fall within the more sensitive region of human hearing (1 kHz to 4 kHz) the lower levels seen above 17.6kHz to 20kHz does not affect system performance.

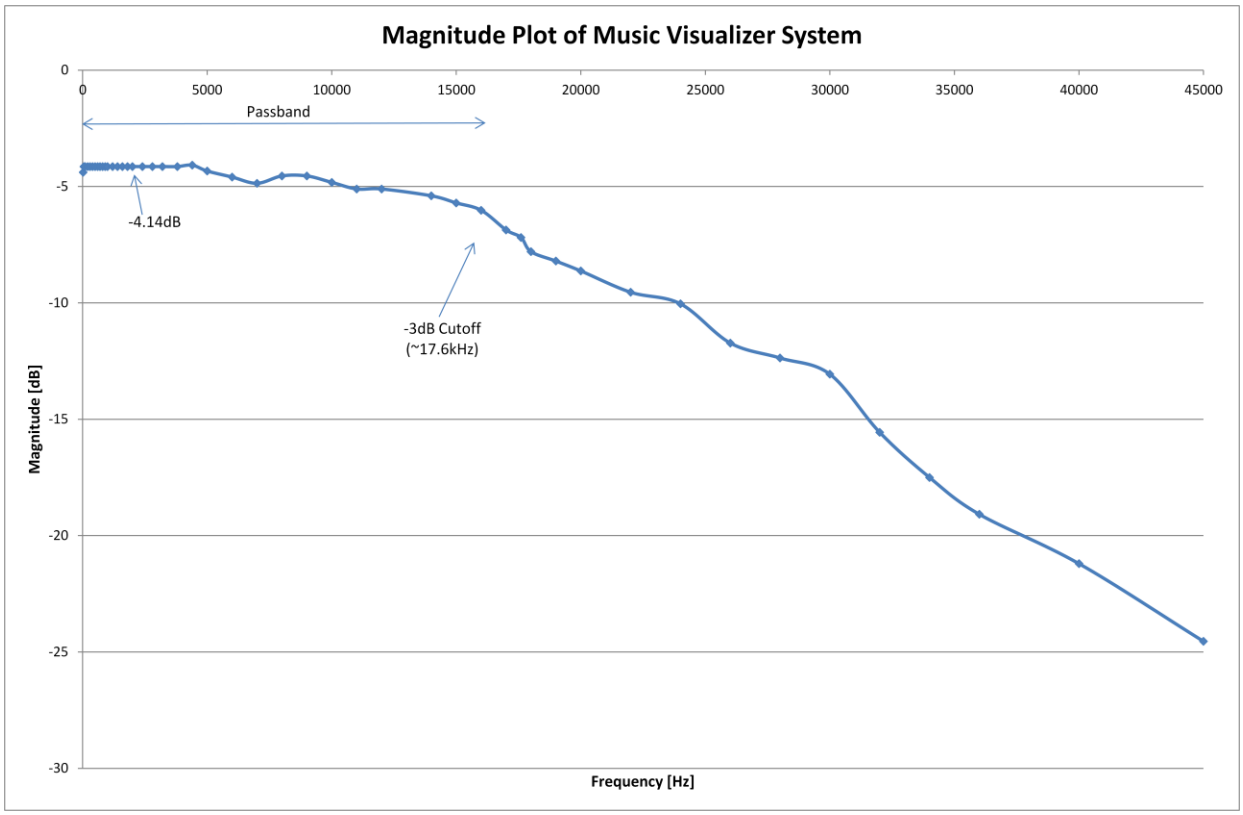


Figure 11. Frequency Response of Music Visualizer System

A ROM checking utility that I have available on my Jaguar development system was used for verification of the signal lines between the 8-bit 27C4001 ROM and Jaguar cartridge interface hardware. The EPROM burner calculated a checksum of 0xC266 (base 16) for the ROM image. When I ran the utility a checksum of the cartridge ROM was calculated to 0xC266 indicating that all address, control and data lines were connected correctly to the system. To test the signal trace connections to the 93C86 EEROM chip a verification utility written by Matthias Domin was used on the system to verify that the EEROM could be properly addressed and read by the Jaguar's CPU. No further testing was done of the EEPROM was done since read/write code for the non-volatile storage was not incorporated into the system.

Final testing of the system involved testing the controls and features of the VLM software to see if any observable anomalies could be detected with the system. The VLM software 81 different

visualizations were individually exercised to verify that they could be activated, respond to input signals and that the edit mode could be accessed. All features of the system and menu cycling performed as expected and no anomalies were noted with the system. The system remained stable in operation thru out testing. Figure 12 and Figure 13 show pictures taken during this testing.

Figure 12. VLM Software Responding to Music Input

Figure 13. Edit Mode Activated

VIII. CONCLUSION

There are several conclusions I can take away with this project. First and foremost is dedicating time for research. My developmental issues with getting a working ADC circuit were plagued by circuit layout issues. My failing to understand, during the design phase of this project, the need for strict separation between the analog and digital side of the ADC circuitry. Most of the datasheets and other documentation assume you already understand this and don't necessarily mention it. It wasn't until I found the Analog Devices AD1871 datasheet [22] that I learned about the importance and method of doing the analog and digital separation on my PCB layout. By doing more thorough research, and planning for the time to do this research, should help you discover problematic issues you may face with a design.

Another takeaway is allowing for time to educate yourself on software tools you will need to accomplish a project. When I initially worked with Eagle CAD I jumped right into and struggled with using the program. This resulted in me wasting time and making numerous mistakes. It wasn't until I forced myself to sit down and work thru a tutorial for the software that I became more skilled with the software and improved my efficiency of producing schematics and PCB layouts. My first boards produced for this project using the SAA7360 ADC chip resulted in about 12 un-routed traces. This could have been avoided if I took the time to properly educate myself on the software.

My last takeaway is having a methodical checklist for performing quality assurance on schematic and PCB layouts. As mentioned above I had un-routed signals. Having a process such as printing out and reviewing every pin connection would have allowed me to catch the errors before I sent the design files off for manufacturing. I also misjudged the drill hole size needed for the socket eyelets I planned on using for flush mounting of the program ROM on the PCB. I was able to correct this

easily using a drill press in the Mustang 60 Machine Shop to drill the holes with a number 58 drill bit. Another reason for having a well thought out means of doing quality assurance and sticking with it – don't assume it is correct, verify it.

IX. REFERENCES

[1] R. Ford and C. Coulston, “Properties of Engineering Requirement,” in *Design for Electrical and Computer Engineers*, McGraw-Hill, 2007, p. 37

[2] *IEEE Std 1233, 1998 Edition*, p. 4 (10/36), DOI: 10.1109/IEEESTD.1998.88826

[3] A. Ambardar, “Analog and Digital Signal Processing,” 2nd ed., Pacific Grove, CA: Brooks/Cole Publishing Co., 1999, pp. 398-439, 762-797.

[4] S. Franco, “Design with Operational Amplifiers and Analog Integrated Circuits,” 3rd ed. , New York: McGraw-Hill, 2002, pp. 106-204.

[5] S.C. Nanayakkara, E. Taylor, L. Wyse, and S.H. Ong, "Towards building an experiential music visualizer," *Information, Communications & Signal Processing, 2007 6th International Conference on*, vol., no., pp.1-5, 10-13, December 2007. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4449609&isnumber=4449533 [Accessed October 12, 2011]

[6] R. Brown, “Audio Activated Video Display,” U.S. Patent 4 081 829, March 28, 1978. Available: http://www.google.com/patents/about/4081829_Audio_activated_video_display.html?id=GCIvAAAAEBAJ [Accessed October 12, 2011]

[7] W.A. Serdijn, M. Broest, J. Mulder, A.C. Van Der Woerd, and A.H.M. Van Roermund , "A low-voltage ultra-low-power translinear integrator for audio filter applications," *Solid-State Circuits, IEEE Journal of* , vol.32, no.4, pp.577-581, Apr 1997. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=563680&isnumber=12250. [Accessed October 12, 2011]

[8] J.W. Pierre, R.F. Kubichek, and J.C. Hamann, "Reinforcing the understanding of signal processing concepts using audio exercises," *Acoustics, Speech, and Signal Processing, 1999 Proceedings, 1999 IEEE International Conference on*, vol.6, no., pp.3577-3580 vol.6, 15-19 March 1999. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=757616&isnumber=16348 [Accessed October 12, 2011]

[9] M. Brennan, T. Dunn and J. Mathieson, “Jaguar Technical Reference Manual – Tom & Jerry,” 8th ed., Sunnyvale, CA:Atari Corp., February 28, 2001. Available: http://www.hillsoftware.com/files/atari/jaguar/jag_v8.pdf [Access October 12, 2011]

[10] E. Nisley, “ADC and DAC Bandwidth Sampled Signals,” *Circuit Cellar*, no. 185, pp. 52-55, December 2005.

[11] "SAA7360 Bitstream Conversion ADC for Digital Audio Systems product specification," Philips Semiconductor, April 24, 1995. Available: http://www.alldatasheet.com/datasheet-pdf/pdf/19058/PHILIPS/SAA7360GP.html. [Accessed October 15, 2011]

[12] "8-pin, Stereo A/D Converter for Digital Audio CS5330A/CS5331A datasheet," Cirrus Logic, March 1999. Available: http://www.datasheetarchive.com/CS5330-datasheet.html. [Accessed October 12, 2011]

[13] "Low Voltage Rail-to-Rail Operational Amplifiers MC33201/33202/33204 datasheet," Motorola, 2nd ed., 1996. Available: http://www.datasheetarchive.com/MC33201-datasheet.html. [Accessed October 12, 2011]

[14] "Waste = Food (An inspiring documentary on the Cradle to Cradle design concept)," 2006 [Podcast television program], Directed by R. van Hattum. The Netherlands: VPRO. Available: http://www.indybay.org/newsitems/2007/05/15/18416351.php. [Accessed November 11, 2010].

[15] "I2S Bus Specification," Philips Semiconductor, February 1986. Rev. June 5, 1996. [Online]. Available: http://www.classic.nxp.com/acrobat_download2/various/I2SBUS.pdf. [Accessed November 11, 2011]

[16] R. Ford and C. Coulston, "The IEEE Code of Ethics," in *Design for Electrical and Computer Engineers*, McGraw-Hill, 2007, p. 216.

[17] R. Ford and C. Coulston, "Duration Estimation (1) & Cost Estimation (6) equations," in *Design for Electrical and Computer Engineers*, McGraw-Hill, 2007, pp. 198, 205.

[18] "State Electricity Profiles Ranked by Average Retail Price," U.S. Energy Information Agency, 2009 Ed. [Online]. Available: http://www.eia.gov/cneaf/electricity/st_profiles/profiles_sum.html. [Accessed: November 26, 2011].

[19] Marrkula Center for Applied Ethics, "A Framework for Thinking Ethically," Santa Clara University, 2010. [Online]. Available: http://www.scu.edu/ethics/practicing/decision/framework.html. [Accessed: November 19, 2011].

[20] "Lightsynths," Llamasoft – Home of the Virtual Light Machine, 2005. [Online]. Available: http://minotaurproject.co.uk/lightsynths.php. [Accessed: November 26, 2011].

[21] "WM8782 24-bit, 192kHz Stereo ADC Datasheet," Wolfson Microelectronics, 2002. Available: http://www.wolfsonmicro.com/products/adcs/WM8782/. [Accessed March 30, 2012]

[22] "Stereo Audio, 24-bit, 96 kHz, Multibit Sigma-Delta ADC Datasheet," Analog Devices, April 2010. Available: http://www.analog.com/en/analog-to-digital-converters/audio-ad-converters/ad1871/products/product.html. [Accessed March 21, 2012]

APPENDIX A – SENIOR PROJECT ANALYSIS

Project Title: Music Driven Graphical Visualization System

Student's Name: Glenn S. Bruner **Student's Signature:** 

Advisor's Name: Dr. Bryan Mealy **Advisor's Initials: BJM** **Date: 8 Jun 2012**

1. Summary of Functional Requirements

The system will accept a stereo music source, filtered and sampled into a digital form, to drive a graphical visualization display system. The system will accept the stereo audio signal input thru common RCA coaxial cables from any source capable of providing a line-level audio signal. The user of the system will have an input control to access menus to control the visualization software and enter parameters to control how the graphic visualizations respond to the music. The system will feature an edit mode in order to create unique visualizations that tailor the animation to different ranges of the audio frequency spectrum. The system will also feature the ability to save a custom created visualization for later recall by the user. The system will provide standard video signal output formats commonly found on televisions and overhead projectors including composite or super-video signal formats.

2. Primary Constraints

Software design will be a critical constraint for this project. The aspect of this project that involves hardware will be rather straight forward – passband filtering of incoming signal, convert to digital streaming format for input to visualization software. Developing visualization software from scratch is a major undertaking for this project idea; capitalizing on existing software will lessen the overall risk to completing the project. Integration with the hardware will require considerable time and

effort in order to achieve all proposed system specifications and will be key to making this project a success.

Time is another important constraint for this project; careful consideration must be given to time management to prevent this project from falling behind schedule. Completing graduation requirement necessitates finding a balance between both academic needs and outside responsibilities such as family issues, and personal financial obligations. Technical challenges with the prototyping of the system's components and final product assembly are other considerations which need to be taken into account in order to complete the project. Developing skills, or sharpening existing skills, can be achieved by taking additional courses or workshops here at Cal Poly. Finding and soliciting knowledge and experience among faculty members and fellow students can also provide valuable resources to overcome technical and administrative obstacles.

3. Economic

This project has some foreseeable economic impacts regarding human, manufactured and natural capital. For the human capital, this project will directly impact my education and growth as an engineer during the development of the system. The project will also involve supervision by a faculty advisor, direct support of librarians with research assistance, and support from department technicians regarding access to test equipment providing assistance in the system's development. Indirectly there will be human and manufactured capital utilized outside of the immediate realm of Cal Poly involved with transportation, warehousing of parts, fabrication, software and documentation employed during the development of this project. The development and build cycles of this project will consume natural resources in the form of gasoline consumed while transporting in parts and materials as well as transporting me to and from school. This project will also consume natural resources like silicon, aluminum, tin, paper, lead and precious metals, such as gold.

Although human, manufactured and natural resources will be consumed during this project, their consumption will help generate revenue either by their direct or indirect involvement with this project. The underlying theme to all of this is to ensure natural resource consumption is carefully monitored in order to minimize their use in this project; the benefit of saving natural resources will help ensure the ultimate success of this project.

The economic support for this project will come from me and accrue during the development and building phases of this project. The estimated parts cost is around $210. I estimate the equipment costs required to complete this project to be the thousands of dollars; these costs are primarily associated with tools and test equipment required to complete the project. The immediate economic benefit and profit will manifest with the completion of my education. After graduation the experience I gain thru this project will help benefit any future employer with an engineer ready to assist or manage a design project.

The system has no expected maintenance costs. The cost of operation will be those costs associated with the electricity used to operate the system, audio and projection equipment. And as for manufacturing plans, there are no plans to produce any systems other than the one used for demonstration of this project.

4. If manufactured on a commercial basis:

Should the system go into production, I estimate that 20 systems would be sold during a single year with a projected product life of approximately five to seven years. The initial parts estimate is around $210 for a system and around $50 in labor of assembling the system. An initial estimated price point for this system is $360. Further analysis is required to conduct a market research to find the optimal consumer price point balanced with the potential sales to quickly recoup development costs. A maximum profit of $2,000 would yield from the sale of 20 units. The monthly operating cost estimate is

around $1.20 in electricity usage using a high retail electricity rate of 21.21 cents [18]. This is based on an average use of four hours per week for the system and a large screen television. Using a worst case of 50W and 301W power consumption for the system and a plasma television, respectively, this would equate to a kilowatt hour (kWh) usage of 0.05 kWh and 0.301 kWh. This totals to a monthly amount of 5.62 kWh of electricity consumed which was then multiplied by the 21.21 cent kWh rate to come up with the operating cost.

5. Environmental

The environmental impacts of this project are widespread and thus difficult to precisely quantify. I estimate that the gasoline consumed in transportation of parts, materials and myself are the largest environmental impact contributor during this project. Gasoline consumption introduces hydrocarbons and other gases into the atmosphere. Many chemicals are also used in the manufacture of plastics and electronic components needed for this project. Natural resources directly consumed will be in the form of paper, meat, vegetables and fruit (food) during the development of this project. The consumption of these natural resources will impact the ecosystem by this project along with producing waste material that will end up in a landfill. Impact to the ecosystem is minimized through the making of wise choices regarding the amount of natural resources consumed and how they are disposed of. Recycling waste material generated by this project is just one of many possible means by which to minimize the impact.

6. Manufacturability

Keeping the final design light weight and utilizing electronic components that have a viable and lasting product life are two of the manufacturing challenges that this project faces. A relative light weight is one of the design goals of the system. If other design requirements present their own unique

challenges and limitations the system could possibly exceed the weight requirement, impacting one of the marketing requirements. Moreover, electronic components, such as integrated circuits, can become obsolete quickly. Semiconductor manufacturers only produce these parts as long as there is a viable demand to for the part. The system could no longer be manufactured in its current design should a chosen component no longer become available and no suitable substitute parts exist. The only option then is to redesign the system in order to integrate new components into the system.

7. Sustainability

Some of the sustainability issues facing the system are associated with the safe storage and operation of the system. When in use the system should be placed on a sturdy table and flat surface and use surge protectors to avoid electrical damage during operation. Electricity is the only resource consumed directly by the complete system. The system is sustainable if the electricity is sourced from renewable sources like wind, hydroelectric or solar energies. If the system were upgraded to work with the more modern digital television systems, by way of HDMI video connectivity, the sustainability of this system could be extended considering that the composite and super-video video input formats may eventually become obsolete. Including HDMI capability would have the affect of raising development cost though, while eventually the projection equipment that supports HDMI should come down in price as its use becomes more widespread. At that time then it would benefit this project to consider upgrade to support HDMI for product life sustainability.

Another sustainability goal of this system, thru the rethinking of the term “sustainability”, is to try and re-use existing technology that otherwise would be considered “waste” and turn the technology into “food” for a new system, such as this project. This project will use parts that do not naturally become food for the decomposers, the organisms within the ecosystem that breakdown waste and convert into nutrients for larger organisms. The ecosystem could benefit if this product can help extend

the usefulness of an existing system and prevent it from prematurely entering a landfill. As the prospect of recycling methods hopefully improve in the future, then maybe the system saved from entering a landfill will become 100% recyclable as expressed by William McDonough in the "Waste = Food" documentary [14].

8. Ethical

An ethical framework for the use of this product is with the notion that life in community is good in itself and our actions should contribute to that life [19]. Music is a common experience enjoyed by humans and is often enjoyed best in a public environment. These public environments help bring together people in a social experience and contribute positive experience to their individual lives. The intent of the final product from this project is to enhance the social experience gained during these public events. Thus, as long as the social experience is positive then there is a hope that this project ultimately enhances the overall quality of life in the community.

An ethical issue that could be considered with this product is if it was used by people involved with illegal drug use. For better or worse, there are people out in society who believe that using illegal drugs enhance how they experience life. The development of this system is not intended to promote illegal drug use in any way; the eventual enhancement of a musical performance with images and animations that further stimulate the visual senses is left up to those attending to use it as they will.

There could be a perceived ethical dilemma with the use of other people's work in the course of academic work. This project is based on the work of many hardware and software engineers whose hard work had produced a product that unfortunately had a short product life. The system has only been sustained thru homebrew development activity. It will be the intent of this project to ensure proper recognition is given to the authors and engineers whose intellectual work will contribute to this project, as identified in rule 7 of IEEE Code of Ethics. It is strictly the intent of this project to develop an

improved function to the game system and gain project management experience associated with this project. There is no intent to market this project.

9. Health and Safety

Safety concerns for this project are associated with soldering and printed circuit board fabrication. Soldering can produce fumes that contain dangerous gases and breathing these fumes should be avoided. The use of a soldering iron involves temperatures around 500° to 700° Fahrenheit. Safety practices learned in IME 156 regarding the proper use of a soldering iron should be observed to avoid painful first or second-degree burns. Safety glasses should be employed to protect eyes from the splash of hot solder due to the soldering and tip cleaning activities and a bench top smoke extractor to absorb the fumes generated from the smoke and particulates from the solder fluxes.

10. Social and Political

The social benefits envisioned with the use of this system are thru the enhancement of an entertainment event such as a concert or an event hosted by a disc jockey. By enhancing the entertainment experience the system is improving the social experience for those attending. The direct stakeholders are the companies that provide the materials used to make this system. This system provides means by which the manufacturers of the parts generate income for their businesses. The parts, if manufactured outside the United States, help generate income and jobs in locations where they are manufactured. This provides benefit to the people who live in those locations with the means of earning an income to support them and their families. Indirectly, businesses that provide services for the companies that directly manufacture the parts of the system also benefit in the economic activity indirectly generated by this project.

At the same time though, this project does contribute to the use of dangerous chemicals and methods used in the manufacture of semiconductor components. If companies that manufacture the components do not operate in an ethical manner then the disposing of the hazardous waste generated would eventually lead to health, safety and social issues for those living around the parts manufacturer. This eventually will cause political issues for the government where the parts manufacturing plant is located. This project requires consideration in the parts used to ensure parts that they come from manufacturers who demonstrate that they operate under ethical guidelines. That they provide proper compensation for their employees hard work, provide benefits to the health and safety of their employees and their families and ensure they do not negatively impact the ecosystems thru their manufacturing processes.

11. Development

The sophomore and junior level Cal Poly EE courses provide a foundation of skills and knowledge required to successfully complete this project, but there are some senior level courses that I am currently taking or scheduled to take that will facilitate the exploration of design options for this project. The EE409/449 courses provide necessary background for using active analog filters which relates to the required passband filter this project needs. The EE 419/459 – Digital Signal Processing (DSP) can provide the expanded knowledge for developing the analog-to-digital converter circuit and using MATLAB for DSP development. Continuing to enhance my skills associated with simulation tools such as PSpice, LT Spice and MATLAB to utilize the power these analytical tools can provide to this project. Finding courses outside of the EE department, such as IME 322 – Leadership and Project Management, can enhance the management aspect of this project beyond what has been taught during EE460. Moreover, IME 458 - Microelectronics and Electronics Packaging is a course that can provide additional

instruction on printed circuit board development which will be needed during the build phase of this project.

Development tools that will be required to assist in this development will entail using a Jaguar system modified as a development system. An Alpine Development board containing battery backed static RAM and PC parallel port interface, dis-assembler, macro assembler, linker and debugger tools will provide the tools required for software development. Skills developed during CPE 129/169, CPE229/269 and CPE 336, along with my thirty years of assembly language programming on Radio Shack, Atari and Macintosh computers provides an important foundation to help ensure that all unforeseen problems that may arise do not significantly impact the various milestones of this project.

APPENDIX B. SPECIFICATIONS AND REQUIREMENTS

Table 8. Music Driven Graphical Visualization System Requirements and Specifications

Marketing Requirements	Engineering Specifications	Justification
2, 6	1. Total parts should cost less than $300 and system shall utilize existing hardware technology to reduce costs as much as possible.	The system cost needs to be priced such that it provides an attractive option for users to purchase. Using existing technologies that have the necessary graphics, video, CPU and memory hardware capabilities that support the needs of the system reduces development time; re-use of existing technology support concepts of sustainability and reduce hardware costs.
1, 3, 4	2. System should be easy to setup and activate a graphical visualization within an average time period of ten minutes.	The system should require minimal setup time. A typical user would have many other tasks to do in setting up their DJ equipment. The user needs system that is quick to setup and place into operation.
1	3. System should be easy to transport with a maximum weight of ten pounds.	Disc jockeys (DJs) typical work requires an extensive amount of equipment to transport and setup. Adding more heavy equipment to carry would detract from using this system.
3	4. System should accept stereo audio signal input, with a frequency range of 20 to 20k Hz and nominal amplitude level of 0.316Vrms (-10dBV) and maximum of 1.0Vrms (0 dBV).	The frequency range is considered the range of human hearing and recorded music. The 0.316Vrms level is considered standard signal level for consumer audio equipment.
3	5. System should have single audio channel input impedance of at least 10k Ω.	The interface between the source and input needs to maximize voltage transfer of the audio signal and function as an impedance bridge to prevent loading of the audio source that would result in distortion of the audio signal. Research has found that minimum impedance for a line input port, driven by a 100 Ω line output, is 10k Ω.
2, 3	6. System should use industry standard coaxial RCA style audio connectors for audio source input.	This connector is a very common and accepted type of connection used for audio signals on home and commercial audio equipment. Using common connection standards helps reduce equipment cost.
All	7. System will be hosted on Atari Jaguar video game platform [9].	This system provides ideal hardware platform for generation of visualization graphics, TV signal generation, user control input and hardware input & output ports to support this system. Supports all of the marketing requirements and platform is an open platform

Marketing Requirements	Engineering Specifications	Justification
		system released into the public domain. But the systems VLM software will require significant modification.
5	8. System should have ability to create and edit a user created custom graphic visualization.	This provides the user the ability to create a custom visualization that can be customized to a specific style of music or sound environment.
5	9. System should have capability to save and recall a user created custom graphic visualization.	This is needed in order to save a custom visualization so that the visualization parameters do not require re-entry before use.
1, 4, 5	10. System should allow user to control and interact with graphical visualization software for changing displays, save and recall custom settings and enter operating parameters.	A simple controller device, ergonomically compatible and intuitive to use, that allows selection of visualization options and supports user entry of visualization parameters.
2, 3	11. System should output standard analog television video formats including composite, RGB or separated-video.	These are common video signal input standards found on televisions, monitors and overhead projector equipment. Using common video connection standards helps reduce equipment cost and provides versatile options in its use.
2	12. System should operate on a voltage input of 110 to 120V AC with a power consumption under 50 watts.	Unit should be able to operate from standard voltage sources that would be found in locations where similar equipment would use the same voltage sources.

Marketing Requirements

1. Easy to use, quick setup and easy to transport
2. Affordable to own and operate
3. Compatible with home/commercial audio and video equipment
4. Comes with choices of preconfigured graphic visualization options for a wide range of music styles
5. Allows creation of a custom configured visualization setting that can be saved and recalled for later use
6. Supports sustainability by re-using existing technologies to extend its useful life of operation

The requirements and specifications table format derives from [1], Chapter 3.

APPENDIX C. PARTS LIST AND COSTS

Table 9. Parts List and Cost

Part Reference	Part Value	Package	Price
C1	100uF	C/6032-28W	$1.06
C3	27pF	C0805	$0.09
C4	68pF	C0805	$0.09
C5	27pF	C0805	$0.09
C6	68pF	C0805	$0.09
C8	100uF	C/6032-28W	$1.06
C9	4.7uF	C/6032-28W	$0.75
C10	4.7uF	C/6032-28W	$0.75
C11	2.2uF	C1206	$0.65
C12	2.2uF	C1206	$0.65
C13	0.1uF	C0805	$0.14
C14	0.1uF	C0805	$0.14
C15	47uF	C/6032-28W	$0.70
C16	47uF	C/6032-28W	$0.70
C17	47uF	C/6032-28W	$0.70
C18	47uF	C/6032-28W	$0.70
C19	47uF	C/6032-28W	$0.70
C20	47nF	C0805	$0.57
C21	47nF	C0805	$0.57
C22	47nF	C0805	$0.57
C23	47nF	C0805	$0.57
C24	47nF	C0805	$0.57
C25	47uF	C/6032-28W	$0.70
C26	47nF	C0805	$0.57
PCB			$33.00
IC1	78L05	SO08	$0.38
IC2	M27C4001B	DIL32	$5.45
IC3	78L05	SO08	$0.38
IC4	SAA7360GP	QFP44_SOT205	$6.09
IC5	93C86	SO-08	$1.03
J1	J1	STX3790	$1.93
JP2		wire	
JP3		wire	
L1	10uH	L1812	$0.31
L2	10uH	L1812	$0.31
R5	150	R0805	$0.03
R6	150	R0805	$0.03
R7	10k	R0805	$0.03
R8	10k	R0805	$0.03
R9	10	R0805	$0.03
R10	15k	R0805	$0.07
R11	10	R0805	$0.03
R12	150	R0805	$0.03
R13	150	R0805	$0.03
R14	10k	R0805	$0.03
R15	10k	R0805	$0.03
R16	10	R0805	$0.03
R17	4.7	R0805	$0.08
R18	10	R0805	$0.03
R19	4.7k	R0805	$0.08
RN2	4.7k	SOMC16	$0.42
IC2 Socket	Socket Eyelets (32 total)		$0.96
		Total	$64.03

APPENDIX D. SCHEDULE – TIME ESTIMATES

This appendix contains Table 10 thru Table 14 that provides more details of tasks listed in Figure 14. Project Gantt Chart timeline. These tables cover the project planning, final report phases, design phase, build and integration phase and project closeout tasks.

Table 10 provides cost estimates for the project over the total development time of the project. A preliminary analysis of required parts for the system was conducted during the EE460 course. This analysis provided a basis to compile the high, medium and low material cost estimates. The optimistic materials estimate assumes a 15% reduction from the priced part while the pessimistic estimate assumes a 30% increase of part prices. Labor estimates are based on a $25/hr wage estimate. According to Figure 14, the Gantt chart projects the project to last approximately 190 days. The estimate for Table 10 labor hours was assessed as one hour per day over the project period. The optimistic labor estimate is 150 hours and the pessimistic estimate is double of the optimistic amount, or approximately 300 hours. The three low, mid and high estimates are combined using the formulas from [17].

Table 10. Project Cost Estimates

Cost Estimates for Music Driven Graphical Visualization System Development			
Item	**Most Optimistic Price [$cost_a$]**	**Most Likely Price [$cost_m$]**	**Most Pessimistic Price [$cost_b$]**
Integrated Circuits	$15.70	$18.47	$24.01
Passive Components			
Capacitors	$7.89	$9.28	$12.06
Resistors	$1.77	$2.08	$2.70
Connectors	$2.11	$2.48	$3.22
Host System Platform	$119.00	$140.00	$182.00
Plastic Case	$4.25	$5.00	$6.50
Printed Circuit Board	$28.05	$33.00	$42.90
Materials Estimate Subtotals	$178.76	$210.31	$273.40
Labor Hours Estimates	150	190	300
Shipping Estimates	$34.00	$40.00	$57.00
Labor Hours Estimate*	202		
Combined Material Estimate Total**	$215.57		
Labor Cost Estimate Total**	$5,041.67		
Combined Shipping Estimate Total**	$41.83		
Material, Labor, Shipping Estimate Total	**$5,299.07**		

* Formula for labor estimate total [17]: $t_e = \frac{t_a + 4 \cdot t_m + t_b}{6}$

** Formula for cost estimate total [17]: $Cost = \frac{cost_a + 4 \cdot cost_m + cost_b}{6}$

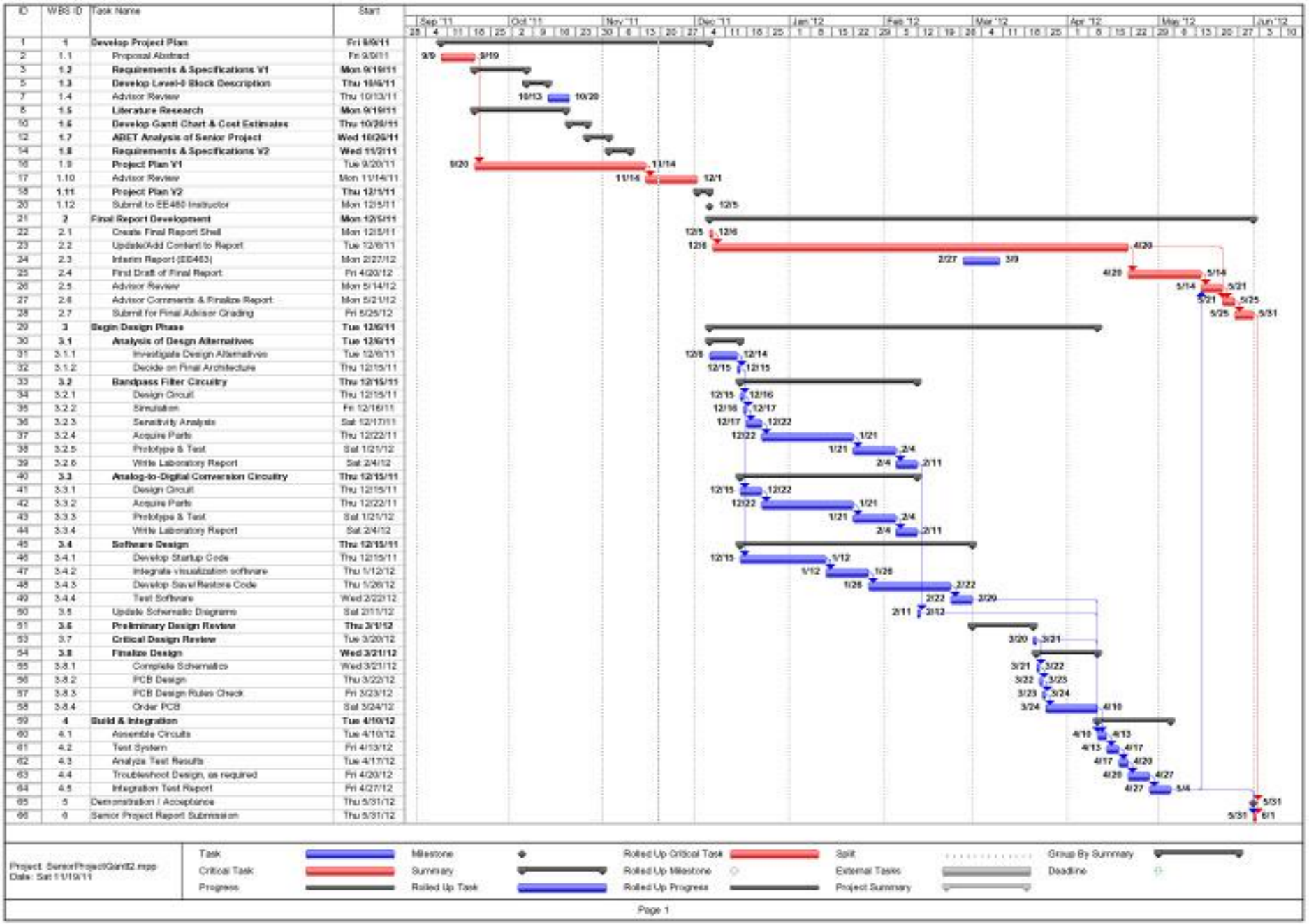


Figure 14. Project Gantt Chart

Table 11. Project WBS - Project Plan & Final Report Phases

WBS ID	Activity	Description	Deliverables / Checkpoints	Duration	Resouces	Predecessors
1	**Develop Project Plan**					
1.1	**Proposal Abstract**	Identifies need and how the system fulfills the need	• Abstract	7 days		
1.2	**Requirements & Specifications V1**	Defines project capability & test criteria	• Engineering Specifications • Marketing Requirements	13 days	• PC	
1.2.1	*Peer Review & Incorporate Comments*			17 edays		1.1
1.3	**Develop Level-0 Block Description**	Describes system input & output functions	• Level-0 Functional Description	5 days	• PC	
1.3.1	*Peer Review & Incorporate Comments*			7 edays		1.2.1
1.4	**Advisor Review**	Obtain feedback and project supervision acceptance		7 edays		1.3.1
1.5	**Literature Research**	Gains background material for project development	• Background material	22 days	• PC	
1.5.1	*Peer Review & Incorporate Comments*			30 edays		1.1
1.6	**Develop Gantt Chart & Cost Estimates**	Provides roadmap for project & initial cost projections	• Project roadmap	4 days	• PC	
1.6.1	*Peer Review & Incorporate Comments*			6 edays		1.5.1,1.4
1.7	**ABET Analysis of Senior Project**	Appendix A for Project Plan	• ABET Project Plan Appendix A	5 days	• PC	
1.7.1	*Peer Review & Incorporate Comments*			7 edays		1.6.1
1.8	**Requirements & Specifications V2**	Continued refinement of project capabilities & test criteria	• Level-1 Functional Description	5 days	• PC	
1.8.1	*Peer Review & Incorporate Comments*			7 edays		1.7.1
1.9	**Project Plan V1**	Introduces the project, explains the marketing reqs and engineering specs, explains the functional decomposition at levels 0 &l 1	• Project Plan Draft	40 days	• PC	1.1
1.10	**Advisor Review**	Obtain feedback on project plan		17 edays		1.9
1.11	**Project Plan V2**	Revision of project plan	• Revised Project Plan	2 days	• PC	
1.11.1	*Incorporate Advisor Comments*	Revise project plan based on supervisor review		4 edays		1.10 1.8.1
1.12	**Submit to EE460 Instructor**	Submit for grading for EE460 completion	• Completion of EE460	0 edays		1.11.1
2	**Final Report Development**					
2.1	**Create Final Report Shell**	Develop report outline based on senior design handbook	• Final Report Layout	1 eday		1.11.1
2.2	**Update/Add Content to Report**	Capture project details over course of development	• Progress Documenting	136 edays	• PC	2.1
2.3	**Interim Report (EE463)**	Provide project status at the end of EE463 to project advisor	• Progress Status	10 days	• PC	
2.4	**First Draft of Final Report**	Initial cut of final report	• Final Report for Review	24 edays	• PC	2.2
2.5	**Advisor Review**	Supervisor review of final report		7 edays		4.5 2.4
2.6	**Advisor Comments & Finalize Report**	Revise report based on supervisor review	• Final Report	4 edays	• PC	2.5 2.2
2.7	**Submit for Final Advisor Grading**	Final report submission for EE464 completion		6 edays	• PC	2.6

Table 12. Project WBS - Design Phase (Part 1 of 2)

WBS ID	Activity	Description	Deliverables / Checkpoints	Duration	Resouces	Predecessors
3	**Begin Design Phase**					
3.1	**Analysis of Desgn Alternatives**			8 days		
3.1.1	*Investigate Design Alternatives*	Investigate design options and determine pros and cons to each design		7 days	• PC	1.11.1
3.1.2	*Decide on Final Architecture*	Pick best design	• Design Approach	1 day	• PC	3.1.1
3.2	**Bandpass Filter Circuitry**			41 days		
3.2.1	*Design Circuit*	Develop circuit and define required components	• Input Filter Circuit	1 eday	• PC • MATLAB	3.1.2
3.2.2	*Simulation*	Conduct analysis of ideal circuit	• Filter Performance	1 eday	• LT Spice/ PSpice	3.2.1
3.2.3	*Sensitivity Analysis*	Test circuit using component tolerances, temperature variations, input & output conditions	• Filter Environmental Performance	5 edays	• LT Spice/ PSpice	3.2.2
3.2.4	*Acquire Parts*	Source and purchase parts	• Place Order • Receive Parts	30 edays		3.2.3
3.2.5	*Prototype & Test*	Develop circuit prototype and test using criteria developed in project plan	• Test data for Analysis	14 edays	• Parts • Power Supply • Oscilloscope • Function Generator • Test Bench • DMM	3.2.4
3.2.6	*Write Laboratory Report*	Write test report and submit to advisor. Incorporate into final report.	• Progress update to advisor	7 edays	• PC	3.2.5
3.3	**Analog-to-Digital Conversion Circuitry**			41 days		
3.3.1	*Design Circuit*	Develop circuit, define required components, simulate using available simulation tools	• Digital Conversion Circuit	7 edays	• PC • MATLAB	3.1.2
3.3.2	*Acquire Parts*	Source and purchase parts	• Place order • Receive parts	30 edays		3.3.1
3.3.3	*Prototype & Test*	Develop circuit prototype and test using criteria developed in project plan	• Test data for Analysis	14 edays	• Parts • DMM • Function Generator • Oscilloscope • Power Supply • Proto-board • Test Bench	3.3.2
3.3.4	*Write Laboratory Report*	Write test report and submit to advisor. Incorporate into final report.	• Progress update to advisor	7 edays	• PC	3.3.3

Table 13. Project WBS - Design Phase (Part 2 of 2)

WBS ID	Activity	Description	Deliverables / Checkpoints	Duration	Resouces	Predecessors
3	**Begin Design Phase (Continued)**					
3.4	**Software Design**			54 days		
3.4.1	*Develop Startup Code*	Program code use to startup host system and load visualization software	• Boot code	28 edays	• PC • Assembler • Linker • Development Board • Jaguar System • TV Monitor	3.1.2
3.4.2	*Integrate visualization software*	Modify visualization software for project needs	• Integrated Boot & Visualization Software	14 edays	• PC • Assembler • Linker • Development Board • Jaguar System • System Test Board • TV Monitor	3.4.1
3.4.3	*Develop Save/Restore Code*	Program code to save and restore users custome visualization settings	• Complete System Software	27 edays	• PC • Assembler • Linker • Development Board • Jaguar System	3.4.2
3.4.4	*Test Software*	Test software to ensure fulfills requirements, stable and has no performance issues	• Test data for Analysis	7 edays	• PC • Assembler • Linker • Development Board • Jaguar System • System Test Board • TV Monitor	3.4.3
3.5	**Update Schematic Diagrams**	Hardware schematic drawings	• System Schematic	1 eday		3.2.6 3.3.4
3.6	**Preliminary Design Review**	Initial advisor & peer review of system design	• Design for Review	14 days		
3.6.1	*Incorporate Advisor Comments*	Take comments from PDR and make changes to design		14 days		3.5 3.4.4
3.7	**Critical Design Review**	More detailed review of design	• Final Design for Review	1 eday		3.6.1
3.8	**Finalize Design**			14 days		
3.8.1	*Complete Schematics*	Make final schematic changes required for PCB design	• Final Schematic	1 eday	• LT Spice/ PSpice	3.7
3.8.2	*PCB Design*	Circuit board device layout	• PCB layout and drill files	1 eday	• PC • Eagle CAD	3.8.1
3.8.3	*PCB Design Rules Check*	Verify board meets manufacturing and component restrictions	• Verified PCB Files	1 eday	• PC • Eagle CAD	3.8.2
3.8.4	*Order PCB*	Send out board design for manufacturing	• Place order • Receive PCB	17 edays	• PC	3.8.3

Table 14. Project WBS - Build & Integration and Report Closeout

WBS ID	Activity	Description	Deliverables / Checkpoints	Duration	Resouces	Predecessors
4	**Build & Integration**					
4.1	**Assemble Circuits**	Build system	• Integrated System	3 edays	• Parts • Test Bench • PCB • EPROM Burner • Solder Iron	3.8.4 3.7 3.5 3.4.4
4.2	**Test System**	Integration testing to collect data for analysis	• Test data for Analysis	4 edays	• Test Bench • DMM • CD Player • Jaguar System • Function Generator • Oscilloscope • TV Monitor	4.1
4.3	**Analyze Test Results**	Analyze data to determine if system fulfills requirements	• Passed & Failed Criteria	3 edays	• PC • MATLAB	4.2
4.4	**Troubleshoot Design, as required**	Determine underlying cause of failed requirements and make required design changes	• Root cause • Fix action	7 edays		4.3
4.5	**Integration Test Report**	Capture test results ans submit to advisor. Incorporate into final report.	• Progress update • Final report Data	7 edays	• PC	4.4
5	**Demonstration / Acceptance**	Senior project demonstration in EE department building	• Showcase of Project	0 edays	• Jaguar System • CD Player • TV Monitor	2.6
6	**Senior Project Report Submission**	Archive report in Kennedy libarary and fullfill graduation requirements	• Archived report	19 edays	• PC	2.6 4.5

APPENDIX E. CIRCUIT SCHEMATICS

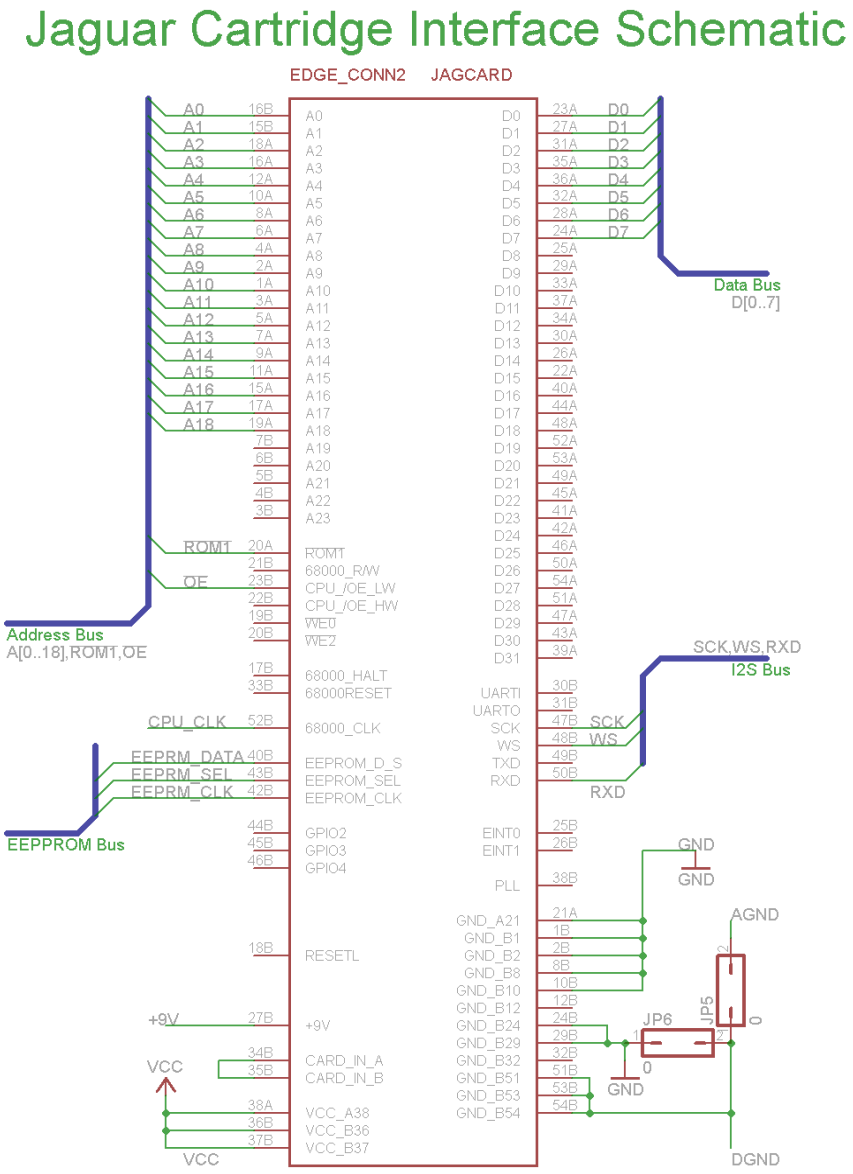


Figure 15. Jaguar Cartridge Interface

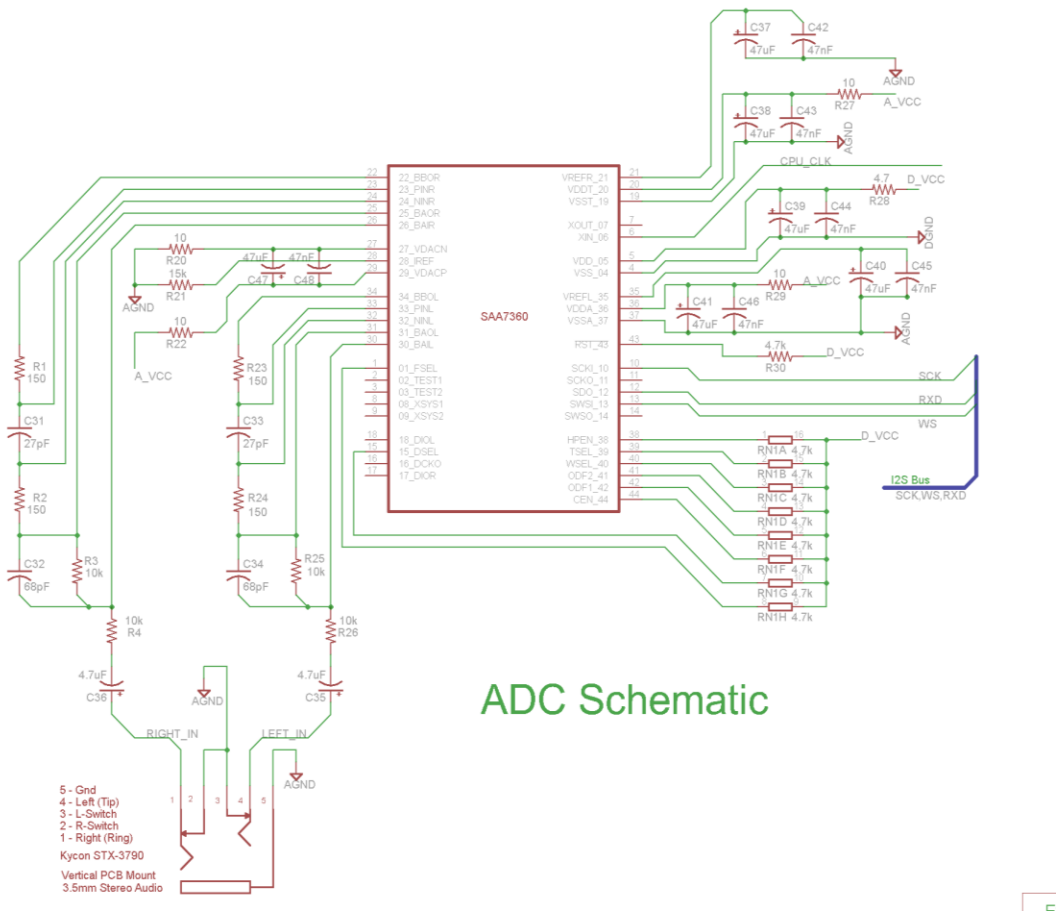


Figure 16. ADC Schematic

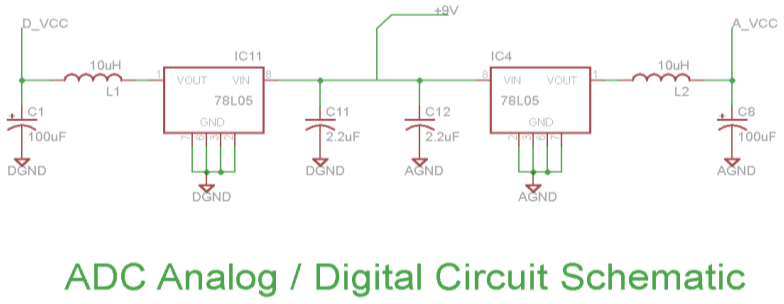


Figure 17. ADC Power Supply

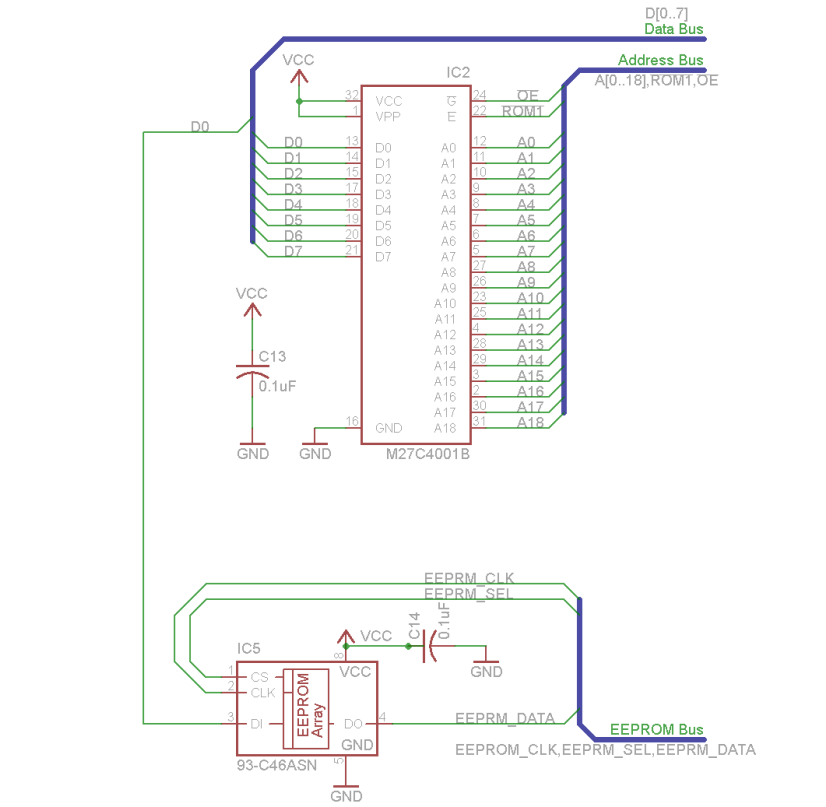


Figure 18. Program / Non-volatile Storage Schematic

APPENDIX F. PC BOARD AND PART LAYOUT

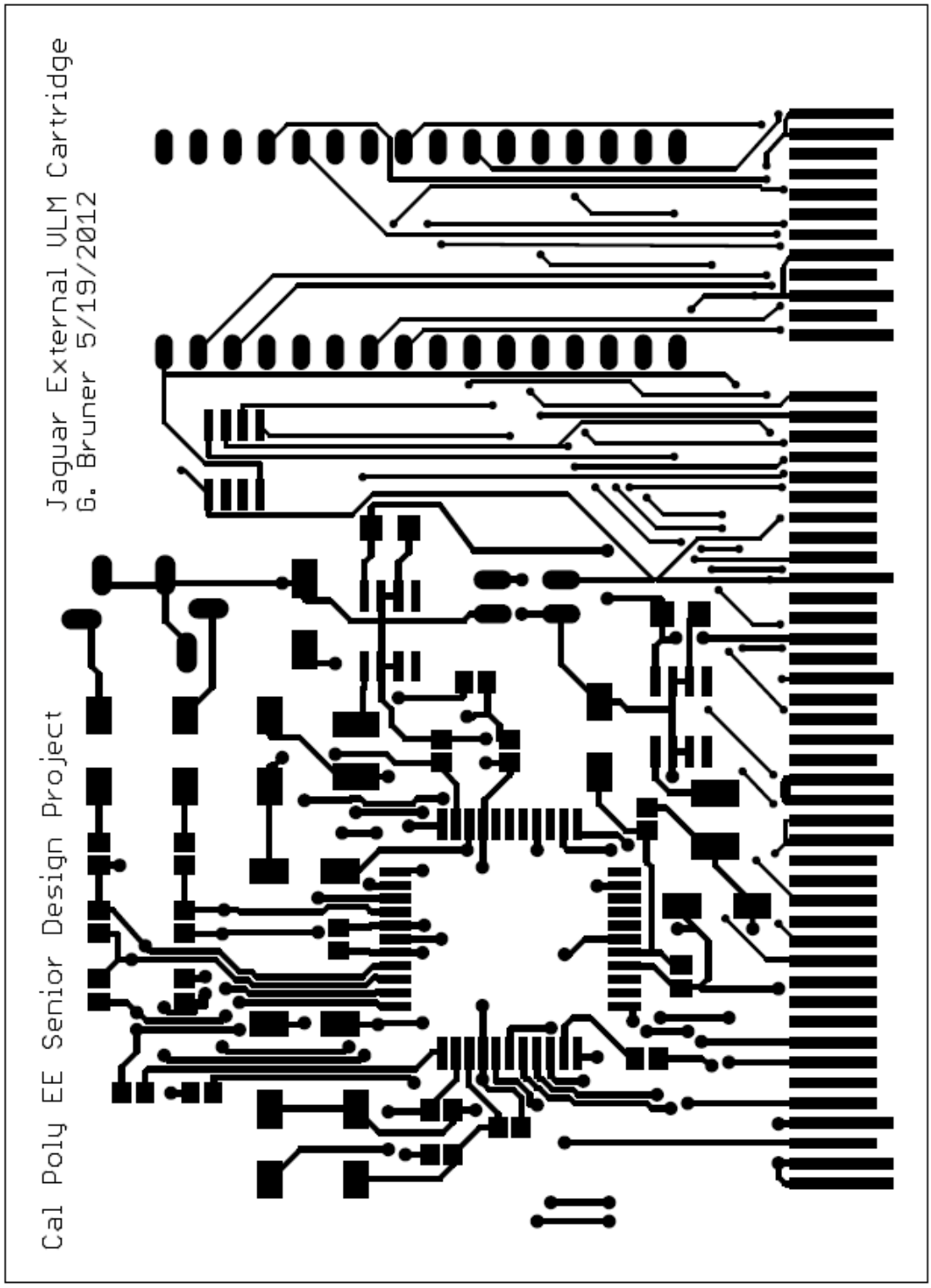


Figure 19. PCB Top Layer Layout

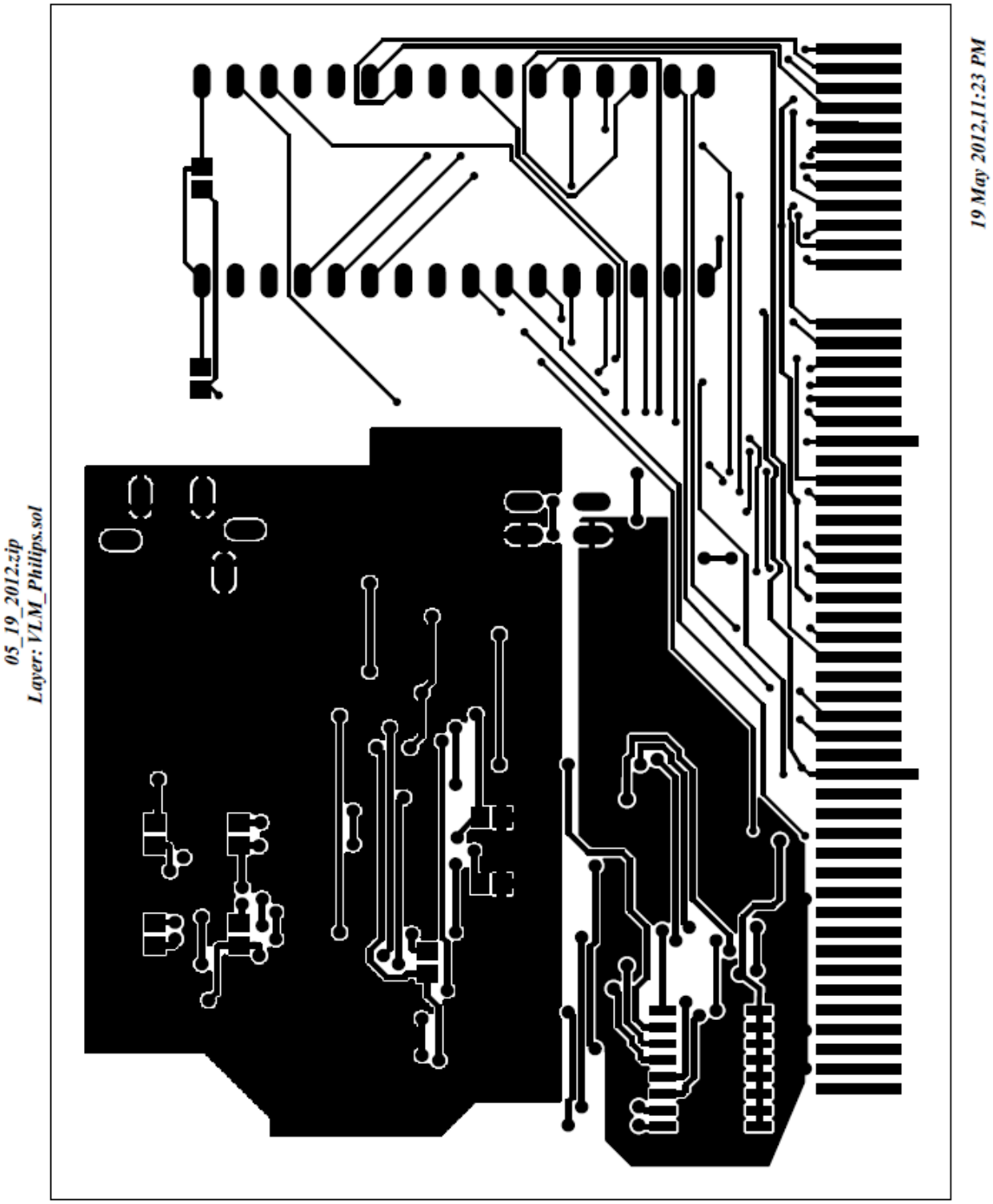


Figure 20. PCB Bottom Layer Layout

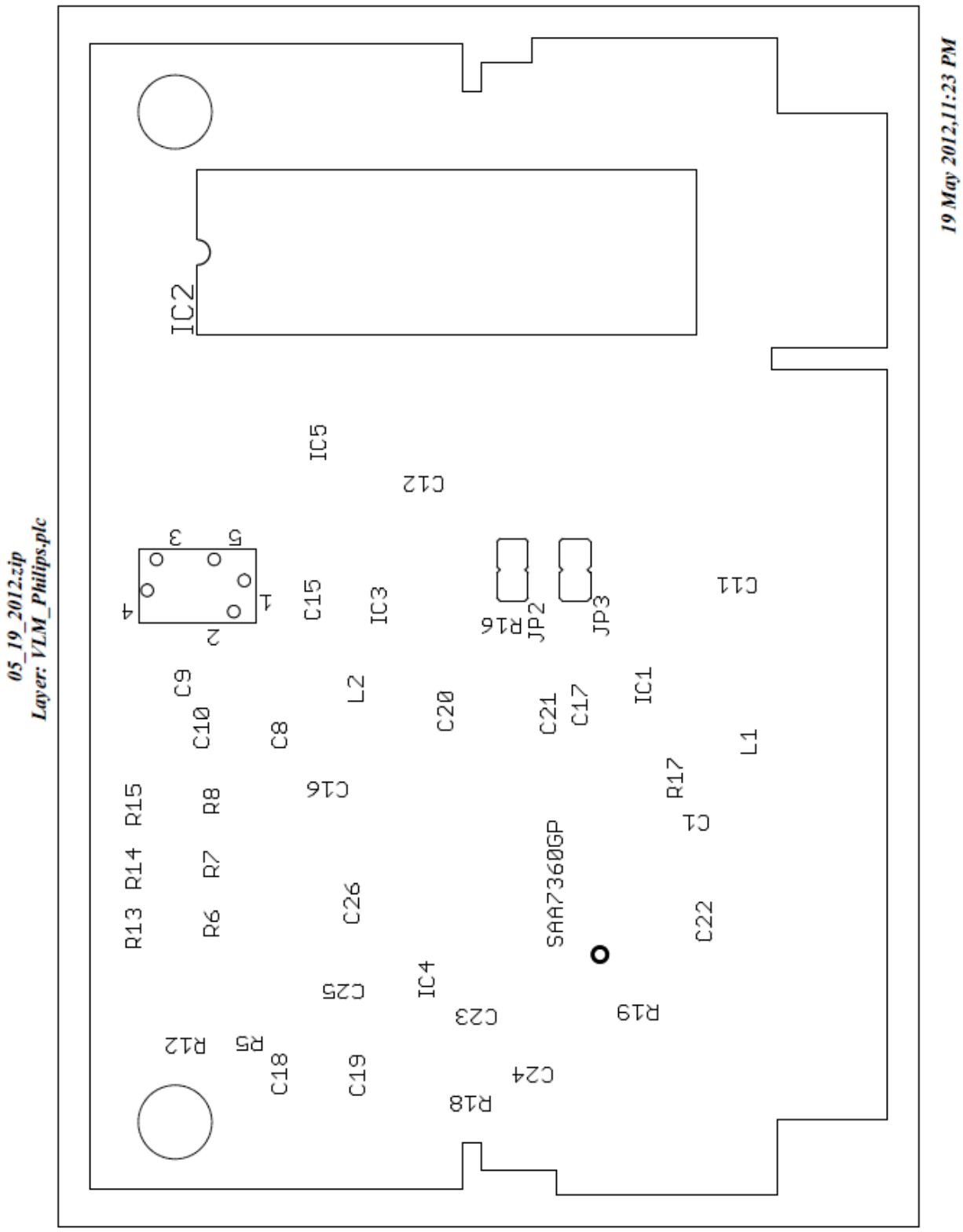


Figure 21. PCB Part Placement (Top Layer)

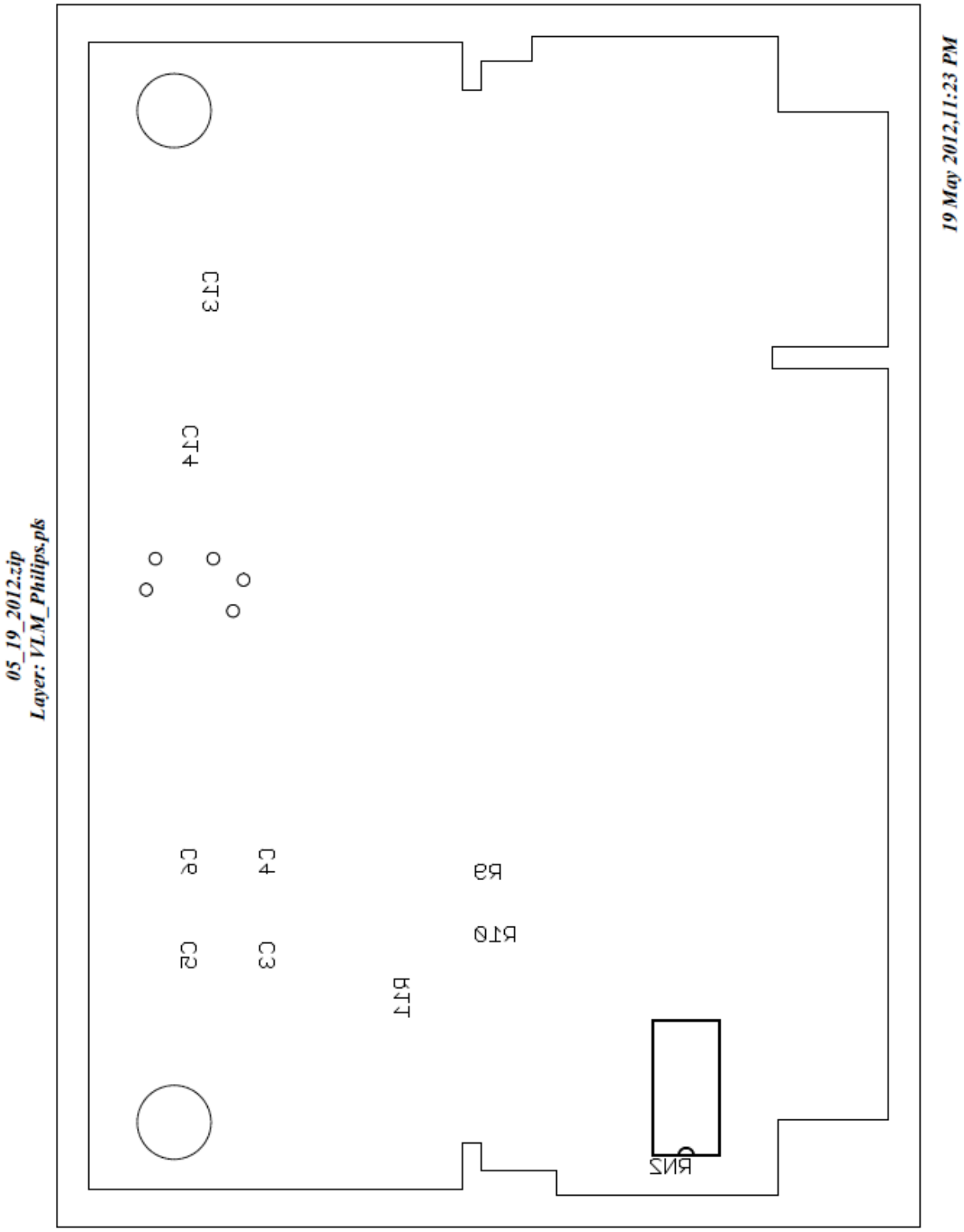


Figure 22. PCB Part Placement (Bottom Layer)

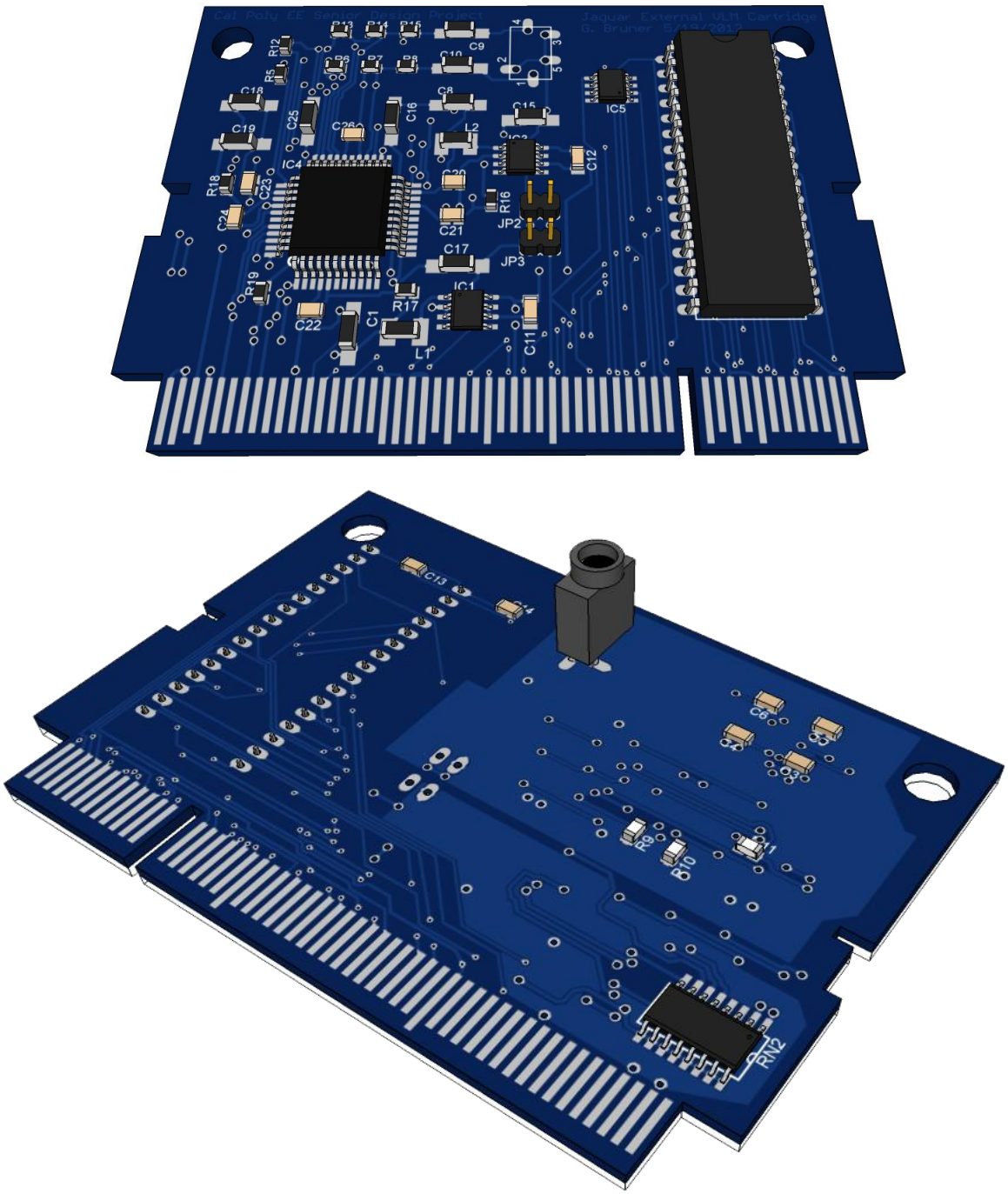


Figure 23. Google Sketch Up 3D Images of Final PCB Layout and Design

APPENDIX G. PROGRAM LISTINGS

Listing 1. VLM Software Loader Program

```
;*****************************************
; Jaguar Virtual Light Machine (VLM) Loader
;       By Glenn Bruner
;               CA Polytechnic State University
;               Senior Project - Spring 2012
;       Project Title:  Music Driven Graphical Visualization System
;
;       Description:  Cartridge based visualization system that takes an
;               external audio feed and processes the audio thru a audio bandpass
;               filter (20-20kHz) to an analog-to-digital converter to the DSP
;               I2S bus.  The VLM software has a fast-fourier transform program
;               running that takes the incoming digital audio stream to convert
;               the data from the time domain to the frequency domain.  The frequency
;               spectrum of the audio data is used to trigger visual effects that
;               are generated by the VLM software.  The user will be able to use
;               a game controller to switch between different banks of visual
;               effects.
;
;*****************************************
; Program Includes & External Variables
;*****************************************
                        .include    'jaguar.inc'
;*****************************************
; Program Equates
;*****************************************
Ramtop          .equ    $200000         ; Top address of dRAM
VLM_dRAM        .equ    $192000          ; Beginning load address of VLM software
cdf_dRAM        .equ    $80000          ; Load address for CD front end code
cdb_dRAM        .equ    $3000           ; Load address for CD BIOS
relocad         .equ    $50000          ; Final org address of this loader in dRAM
JPIT3_read      .equ    $f1003a         ; Programmable timer 3 read-only location

; Some locations within CD Front code
RanGetEE        .equ    $80000
RanPutEE        .equ    $80004
randfAD         .equ    $80008
CDfront         .equ    $8000C          ; Run address for CD front end code

free            .equ    $192018         ;address of freerun mode of VLM
;freerun        .equ    $1987d0         ;.w write a 2 here to stop freerun mode of VLM
skid            .equ    $1aa05e         ;.w VLM mode selection 0-9
vlm_mode        .equ    $1ae02a         ;.w 0 = no ctrl, 1 = ctrl

                .text
;*****************************************
; Main Program
;*****************************************
_start:
                move.l  #$70007,G_END   ; GPU in big endian data organization
                move.l  #$70007,D_END   ; DSP in big endian data organisation

                move.l  #0,G_CTRL               ; Stop the GPU
                move.l  #0,G_FLAGS      ; GPU Flags register
                move.l  #0,D_CTRL               ; Stop the DSP
                move.l  #0,D_FLAGS      ; DSP Flags register

                movea.l  #Ramtop,a7     ; Set stack point to top of ram
                moveq   #-1,d0          ; start timer, just in case it's 0'd
                move.l  d0,JPIT3                ; Timer 2 Pre-scalar
;
;  Load the VLM software into main RAM
;
                lea     vlm,a0          ; Get address of beginning of VLM code
                lea     VLM_dRAM,a1     ; Address in RAM where VLM needs to go
                lea     vlm_x,a2                ; Get address of end of VLM code
vlmset:
                move.l  (a0)+,(a1)+
```

```
                cmp.l   a2,a0           ; Check to see if we reach the end of code
                bcs     vlmset          ; Keep going until we reach end of data
;
;  Load cdfront
;
                lea     cdf,a0
                lea     cdf_dRAM,a1
                lea     cdf_x,a2
cdfset:
                move.l  (a0)+,(a1)+
                cmp.l   a2,a0
                bcs     cdfset
;
;  Now, load the cdbios
;
                lea     cdbios,a0
                lea     cdb_dRAM,a1
                lea     cdbios_x,a2
cdbioset:
                move.l  (a0)+,(a1)+
                cmp.l   a2,a0
                bcs     cdbioset

                bsr     moveLoader      ; Time to move remaining loader code into RAM

                reladj  .equ  _start-relocad   ; Calculate length of relocation move

                lea     newstart-reladj,a0     ; This is the address we continue execution in dRAM
jumper:
                jmp     (a0)            ; Leave 8-bit cartridge space behind

moveLoader:
                lea     _start,a0       ; Beginning of the loader code
                lea     relocad,a1      ; Address to relocate to in main RAM
                lea     loader_x,a2     ; This is end of loader code
reloop:
                move.l  (a0)+,(a1)+
                cmp.l   a2,a0
                bcs     reloop
                rts
newstart:
                lea     randf(pc),a0    ; CD front would like this address
                move.l  a0,randfAD      ;   to a random number generator

                move.w  #$1865,MEMCON1 ; Place cartridge space in 32-bit wide mode

                move.w  #1,vlm_mode     ; We want VLM to activate with controls enabled
                move.w  #5,skid         ; Start with display 5
                jsr     free            ; Activate VLM code in free running mode

                jmp     CDfront         ; Jump to CD front code
;
;  ----------------------------------------
;    Get a random long in d0 after each call
;
randf:
                movem.l a0/d1-d5,-(sp)
;
;  The following random generator provides exactly the same
;   results as Landon Dyer's version in the Atari ST extended BIOS.
;
                movem.l seed(pc),d0-d1 ; get seed & constant
                moveq   #0,d2           ; clear neg flag
rerand:
                tst.l   d0              ; check seed
                bgt     ov2             ; branch if positive, non-zero (seed ok)
                bne     ovx             ; branch if negative, non-zero (need positive)
;
;  there is a zero seed--we need to fix this using a programmable timer source
;
                move.l  JPIT3_read,d0  ; use JPIT3 as a random seed source
```

```
                swap    d0
                bra     rerand          ; the seed is guaranteed non-zero now
ovx:
                neg.l   d0              ; make positive
                addq    #1,d2           ; set negative flag
ov2:
                move.l  d0,d3
                mulu    d1,d3
;
                move.l  d0,d4
                swap    d4
                mulu    d1,d4
;
                move.l  d1,d5
                swap    d5
                mulu    d5,d0
;
                add.w   d4,d0
                swap    d0
                clr.w   d0
                add.l   d3,d0
                tst.w   d2
                bne     ov1
                neg.l   d0
ov1:
                addq.l  #1,d0
                lea     seed(pc),a0     ; Use a relocatable technique to find address
                move.l  d0,(a0)         ; save seed for next time
;
                movem.l (sp)+,a0/d1-d5
                rts
;
seed:
                dc.l    $33ba0359       ; seed
constant:
                dc.l    $44bf19d3       ; constant

loader_x:
                dc.l    0

                .long
;****************************************
; VLM Code
;****************************************
vlm:
                .incbin         'vlm.abs'       ; VLM software binary from CD boot ROM
vlm_x:
                .long
cdf:
                .incbin         'cdfront.bin'   ; CD front-end code from CD boot ROM
cdf_x:
                .long
cdbios:
                .incbin         'cdbios45.bin'  ; CD BIOS binary from developer file library
cdbios_x:
                .end
```

Listing 2. RISC Disassembler Source – Global.h, Main.h and Main.C

```
/*******************************************************
* File: global.h
*
* Description: Definitions for variable assignments and type
*
* Original Author:  anarko <anarko@telia.com>
*
* Revisions:
*
*******************************************************/

/* global.h */

#ifndef _GLOBAL_H
#define  GLOBAL H


#define GPU_TYPE 0
#define DSP TYPE 1

unsigned char  *mem_ram;

#endif


/*******************************************************
* File: main.h
*
* Description: Function prototype definitions
*
* Original Author:  anarko <anarko@telia.com>
*
* Revisions:
*
*******************************************************/

/* main.h */

#ifndef _MAIN_H
#define _MAIN_H

int   main(int argc, char *argv[]);
void  load_gui();

#endif


/*******************************************************
* File: main.c
*
* Description: Main program loop for RISC disassembler.
*
* Original Author:  anarko <anarko@telia.com>
*
* Revisions:  Glenn Bruner, December 2011
*    Added command line switch command to pass code offset
*  value for any GPU / DSP code that did not start at assumed beginning
*  memory locations for respective processors.  Updated the version number
*  of the program to identify this program from the original.
*
*******************************************************/

/* main.c */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
#include "main.h"
#include "functs.h"
#include "global.h"
#include "risc dis.h"
#include "risc_mem.h"
#include "risc_reg.h"


#define  VERSION  "Jaguar DSP/GPU RISC disasm v1.0 by anarko <anarko@telia.com>"
#define  VER2 "Expanded functions by Glenn Bruner, 21Dec2011. v1.3"

int main(int argc, char *argv[]) {

    //Variables used in main
    unsigned long rPC,rSTART,rLEN,filelen;
    unsigned long offset,rMEM;
    unsigned int result=0;
    char msg[80],oplen,cpu_type;

    //Does command line have the required amount of arguments?  If not, display help.
    if (argc != 4) {
        printf("%s\n",VERSION);
        printf("%s\n\n",VER2);
        printf("Syntax: %s data [-d | -g] offset\n",argv[0]);
        printf("Offset is hexdecimal integer value.");
        exit(0);
    }

    if     (strcmp(argv[2],"-d")==0) {
        cpu_type=DSP_TYPE;
        rMEM = 0xF1B000; // Beginning memory location for DSP RAM in Jaguar
    }
    else if(strcmp(argv[2],"-g")==0) {
        cpu_type=GPU_TYPE;
        rMEM = 0xF03000; // Beginning memory location for GPU RAM in Jaguar
    }
    else { printf("Unrecognized option: %s\n",argv[2]); exit(0); }

    //load the file and make reference to source file name and amount of bytes loaded
    filelen=load_rawfile(argv[1],0x0000);
    printf(";Loaded %s (%ld bytes)\n\n",argv[1],filelen);

    //convert offset hex string value to an integer
    result=xtoi(argv[3], &offset);

    //Did xtoi return an error condition - something wrong with offset value?
    if(result>1) {
        printf("Offset value is not a valid hexidecimal value!\n");
        exit(0);
    }

    rPC=0;          //Pointer to where the next instruction for disassembly is
    rSTART=rPC;     //  to be pulled from
    rLEN=filelen;   //Keep track of how much code requires disassembly
    rMEM+=offset;   //Maintain a psuedo memory pointer for generating
                    //  addresses for movei and jr instructions

    //Typical section of program filespace where assembled code goes
    printf("\t.text\n\t");

    //Assembler directive for which type of CPU assembly was generated
    if(cpu_type==GPU_TYPE) { printf(".gpu\n\t"); }
    else { printf(".dsp\n\t"); }

    //Print the assembler direction that identifies where in Jaguar RAM the DSP/GPU code goes
    printf(".org $%07X\n",rMEM);


    while (rPC<rLEN) {
        //Disassemble a line of code and return the opcode length (2, 4 or 6 words)
        oplen=disasm_risc(msg,rPC,rMEM,cpu_type);
```

```
        printf("\t%s\t",msg);  //Print a tab, followed by disassembled opcode and a remark

        //Did we produce a 2 byte opcode, or a larger one?
        if(oplen==2) printf("\t\t; %041X: %04X           \n",rMEM,risc_get16(rPC));
        else         printf("\t; %041X: %04X %04X %04X
\n",rMEM,risc_get16(rPC),risc_get16(rPC+2),risc_get16(rPC+4));

        rPC+=oplen;     //Increment the file data pointer based upon opcode length
        rMEM+=oplen;    //Increment our memory pointer for assembly reference
    }

    //Place an assembler directive to return to 68000 default for instruction assembly
    printf("\t.68000\n");

    //Free up assigned RAM for used for storage of binary file data
    free(mem_ram);
    return 0;
}
```

Listing 3. RISC Disassembler - FUNCTS.h and FUNCTS.c

```
/*********************************************************
* File: functs.h
*
* Description: Variable definition for retrieve opcodes
*
* Original Author:  anarko <anarko@telia.com>
*
* Revisions:  Glenn Bruner, December 2011
*           Added function prototype for xtoi (hex to integer)
*     for conversion of switch offset hex string to numerical form
*     to assist with code cleanup after disassembly.
*
*********************************************************/

/* functs.h */

#ifndef _FUNCTS_H
#define _FUNCTS_H

long  load_rawfile(char *file, long offs);
unsigned int xtoi(const char* xs, unsigned int* result);

#endif


/*********************************************************
* File: functs.c
*
* Description: Jaguar DSP / GPU RISC opcode disassembler.  This program
*   retrieves 2 to 6 bytes of opcode/data and decodes the instruction.
*   The program outputs readiable text information and generates in a form
*   that is compatiable with the origianl Jaguar assembler (MADMAC) and the
*   recently written SMAC assemblers used for Jaguar development.
*
* Original Author:  anarko <anarko@telia.com>
*
* Revisions:  Glenn Bruner, December 2011
*   Added hexidecimal to integer conversion for converting a hexidecimal
*  text value passed to the program thru the DOS command.  This converts a hex value
*  that identifies a required memory offset for the RISC code within the memory space
*  for GPU / DSP code.  This way any code that is disassembled that does not start
*  at the assumed beginning of GPU or DSP memory can be properly offset in the disassembly.
*
*********************************************************/

/* functs.c */
```

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stdarg.h>
#include <ctype.h>

#include "functs.h"
#include "global.h"
#include "main.h"

//Routine to load file to be disassembled into RAM and placed
//  within a character array named mem_ram
long load_rawfile(char *file, long offs)
{
    FILE    *in;
    long    filelen=0;

    in=fopen(file, "rb");
    if(in==NULL) {
        printf("Couldn't find %s.\n",file);
        return 0;
    }

    rewind(in);
    fseek(in,0,SEEK_END);
    filelen=ftell(in);


    fseek(in,0,SEEK_SET);
    switch (offs) {
        case 0x000000:
            mem_ram=(unsigned char *)calloc(filelen,sizeof(char));
            if(mem_ram==NULL) { printf("Couldn't allocate enough memory.\n"); exit(0); }
            if (!fread(mem_ram,1,filelen,in)) {
                fclose(in);
                printf("[LOAD] Failed to read from %s.\n",file);
                return 0;
            } break;
    }
    fclose(in);

    return filelen;
}

// Converts a hexadecimal string to integer
unsigned int xtoi(const char* xs, unsigned int* result)
{
     size_t szlen = strlen(xs);
     int i, xv, fact;

     if (szlen > 0)
     {
      // Converting more than 32bit hexadecimal value?
      if (szlen>8) return 2; // exit

      // Begin conversion here
      *result = 0;
      fact = 1;

      // Run until no more character to convert
      for(i=szlen-1; i>=0 ;i--)
      {
       if (isxdigit(*(xs+i)))
       {
        if (*(xs+i)>=97)
        {
         xv = ( *(xs+i) - 97) + 10;
        }
        else if ( *(xs+i) >= 65)
        {
         xv = (*(xs+i) - 65) + 10;
```

```
        }
        else
        {
         xv = *(xs+i) - 48;
        }
        *result += (xv * fact);
        fact *= 16;
       }
       else
       {
        // Conversion was abnormally terminated
        // by non hexadecimal digit, hence
        // returning only the converted with
        // an error value 4 (illegal hex character)
        return 4;
       }
      }
     }

     // Nothing to convert
     return 1;
}
```

Listing 4. RISC Disassembler - RISC_MEM.h and RISC_MEM.c

```
/*********************************************************
* File: risc_mem.h
*
* Description: Variable definition for retrieve opcodes
*
* Original Author:  anarko <anarko@telia.com>
*
* Revisions:
*
*********************************************************/

/* risc mem.h */

#ifndef _RISC_MEM_H
#define _RISC_MEM_H

unsigned int risc_get16(unsigned int where);

#endif


/*********************************************************
* File: risc_mem.c
*
* Description: Function that retrieves two bytes of RISC opcade.
*
* Original Author:  anarko <anarko@telia.com>
*
* Revisions:
*
*********************************************************/

/* risc_mem.c */

#include "global.h"
#include "functs.h"
#include "risc_mem.h"
#include "risc_reg.h"

// This routine pulls 2 bytes from mem_ram array for disassembly
// This only deals with GPU/DSP programs that fall within range of their respective
//  internal RAM ares (GPU 4k & DSP 8k)
unsigned int risc_get16(unsigned int where) {
```

```
    unsigned short tmp=0;

    if ((where>=0x0000)&&(where<=0x1FFF)) {
        tmp=((mem_ram[where]<<8)|mem_ram[where+1]);
        }

    return tmp;
}
```

Listing 5. RISC Disassembler - RISC_REG.h, RISC_DIS.h and RISC_DIS.c

```
/*********************************************************
* File: risc_reg.h
*
* Description: Structure defintion for RISC registor bits
*    embedded within instruction opcode
*
* Original Author:  anarko <anarko@telia.com>
*
* Revisions:
*
*********************************************************/

/* risc_reg.h */

#ifndef _RISC_REG_H
#define  RISC_REG_H

typedef union {
    struct {
        unsigned lo : 5;
        unsigned mid: 5;
        unsigned op : 6;
    } b;
    unsigned short s;
} risc_matrix;

risc_matrix risc_opcode;

#endif



/*********************************************************
* File: risc_dis.h
*
* Description: RISC disassembler function prototype
*
* Original Author:  anarko <anarko@telia.com>
*
* Revisions:
*
*********************************************************/

/* risc dis.h */

#ifndef _DISC_DIS_H
#define  DISC_DIS_H

char  disasm_risc(char *msg,unsigned long rPC,unsigned long rMEM, char type);

#endif


/*********************************************************
* File: risc_dis.c
*
```

```
* Description: Jaguar DSP / GPU RISC opcode disassembler.  This program
*   retrieves 2 to 6 bytes of opcode/data and decodes the instruction.
*   The program outputs readiable text information and generates in a form
*   that is compatiable with the origianl Jaguar assembler (MADMAC) and the
*   recently written SMAC assemblers used for Jaguar development.
*
* Original Author:  anarko <anarko@telia.com>
*
* Revisions:  Glenn Bruner, December 2011
*   Corrected bugs found in original code that improperly
*  decoded opcodes.  Added condition code types that were
*  not in original code and identified in Jaguar tech reference
*  manual.  Fixed relative jump address calculations for jump relative
*  (JR) instructions.
*
*********************************************************/

/* risc_dis.c */

#include <stdio.h>
#include <stdlib.h>

#include "global.h"
#include "risc_dis.h"
#include "risc mem.h"
#include "risc_reg.h"

//Character array holding all the standard register names
char *risc_r[32] = {
     "r0", "r1", "r2", "r3", "r4", "r5", "r6", "r7",
     "r8", "r9","r10","r11","r12","r13","r14","r15",
    "r16","r17","r18","r19","r20","r21","r22","r23",
    "r24","r25","r26","r27","r28","r29","r30","r31"
};

//Character array holding all the condition code names based on Jag tech reference manual
definitions
char *risc_cc[32] = {
    "T,","NE,","EQ,","??,","CC,","HI,","CC NE,","??,",
    "CS,","CS NE,","LE,","??,","??,","??,","??,","??,",
    "??,","??,","??,","??,","PL,","PL NE,","PL EQ,","??,",
    "MI,","MI NE,","MI EQ,","??,","??,","??,","??,","NT,"
};


#define Rm   risc_r[risc_opcode.b.mid]  //Definition for source register ID #
#define Rn   risc_r[risc_opcode.b.lo]   //Definition for destination register ID #
#define CC   risc_cc[risc_opcode.b.lo]  //Definition for condition code value
#define ii   risc_opcode.b.mid          //Definition for index value for jump relative
instruction


char disasm_risc(char *msg,unsigned long rPC, unsigned long rMEM, char type) {

    unsigned long t32,j32,d16;      //Define variables needed for this subroutine
    char oplen=2;                   //Most opcodes are 2 bytes in length
    risc_opcode.s=risc_get16(rPC);

    /* for MOVEI & JR instructions */
    t32=(risc_get16(rPC+4)<<16)|risc_get16(rPC+2);  //Immediate load value generation
    j32=rMEM+2+((ii>16)?0xFFFFFFE0+ii:ii)*2;        //Jump relative address calculation
    d16=risc_get16(rPC);                            //Get 16-bit word for item that doesn't
translate to an instruction

    // Switch statement below decodes all variations of Jaguar DSP and GPU instructions.  When
    //   the correct opcode is identified (by case) then a text output is produced to generate a
    //   single line of assembly code to the output device (usually the command window or
    //   redirection to a text file using DOS command). Case 0x3E and 0x3F replace invalid opcode
    //   for DSP with an assembler directive to turn that opcode value into a data word to
    //   preserve the opcode at that location in the disassembly.
    switch (risc_opcode.b.op) {
```

```
case 0x00: sprintf(msg,"add      %s,%s",Rm,Rn); break;
case 0x01: sprintf(msg,"addc     %s,%s",Rm,Rn); break;
case 0x02:
            if(ii==0) sprintf(msg, "addq     #$20,%s",Rn);
            else sprintf(msg,"addq     #$%02X,%s",ii,Rn);
            break;
case 0x03:
            if(ii==0) sprintf(msg, "addqt    #$20,%s",Rn);
            else sprintf(msg,"addqt    #$%02X,%s",ii,Rn);
            break;
case 0x04: sprintf(msg,"sub      %s,%s",Rm,Rn); break;
case 0x05: sprintf(msg,"subc     %s,%s",Rm,Rn); break;
case 0x06:
            if(ii==0) sprintf(msg,"subq     #$20,%s",Rn);
            else sprintf(msg,"subq     #$%02X,%s",ii,Rn);
            break;
case 0x07:
            if(ii==0) sprintf(msg,"subqt    #$20,%s",Rn);
            else sprintf(msg,"subqt    #$%02X,%s",ii,Rn);
            break;

case 0x08: sprintf(msg,"neg      %s",Rn); break;
case 0x09: sprintf(msg,"and      %s,%s",Rm,Rn); break;
case 0x0A: sprintf(msg,"or       %s,%s",Rm,Rn); break;
case 0x0B: sprintf(msg,"xor      %s,%s",Rm,Rn); break;

case 0x0C: sprintf(msg,"not      %s",Rn); break;
case 0x0D: sprintf(msg,"btst     #$%02X,%s",ii,Rn); break;
case 0x0E: sprintf(msg,"bset     #$%02X,%s",ii,Rn); break;
case 0x0F: sprintf(msg,"bclr     #$%02X,%s",ii,Rn); break;

case 0x10: sprintf(msg,"mult     %s,%s",Rm,Rn); break;
case 0x11: sprintf(msg,"imult    %s,%s",Rm,Rn); break;
case 0x12: sprintf(msg,"imultn   %s,%s",Rm,Rn); break;
case 0x13: sprintf(msg,"resmac   %s",Rn); break;

case 0x14: sprintf(msg,"imacn    %s,%s",Rm,Rn); break;
case 0x15: sprintf(msg,"div      %s,%s",Rm,Rn); break;
case 0x16: sprintf(msg,"abs      %s\t",Rn); break;
case 0x17: sprintf(msg,"sh       %s,%s",Rm,Rn); break;

case 0x18: sprintf(msg,"shlq     #$%02X,%s",(32-ii),Rn); break;
case 0x19:
            if(ii==0) sprintf(msg,"shrq     #$20,%s",Rn);
            else sprintf(msg,"shrq     #$%02X,%s",ii,Rn);
            break;
case 0x1A: sprintf(msg,"sha      %s,%s",Rm,Rn); break;
case 0x1B:
            if(ii==0) sprintf(msg,"sharq    #$20,%s",Rn);
            else sprintf(msg,"sharq    #$%02X,%s",ii,Rn);
            break;

case 0x1C: sprintf(msg,"ror      %s,%s",Rm,Rn); break;
case 0x1D:
            if(ii==0) sprintf(msg,"rorq     #$20,%s",Rn);
            else sprintf(msg,"rorq     #$%02X,%s",ii,Rn);
            break;
case 0x1E: sprintf(msg,"cmp      %s,%s",Rm,Rn); break;
case 0x1F: sprintf(msg,"cmpq     #$%02X,%s",ii,Rn); break;

case 0x20:
           if(type==GPU_TYPE) sprintf(msg,"sat8     %s    ;GPU",Rn);
           if (type==DSP_TYPE)
                if(ii==0) sprintf(msg,"subqmod  #$20,%s  ;DSP",Rn);
                else sprintf(msg,"subqmod  #$%02X,%s  ;DSP",ii,Rn);
           break;
case 0x21: if(type==GPU_TYPE) sprintf(msg,"sat16    %s    ;GPU",Rn);
           if(type==DSP_TYPE) sprintf(msg,"sat16s   %s    ;DSP",Rn); break;
case 0x22: sprintf(msg,"move     %s,%s",Rm,Rn); break;
case 0x23: sprintf(msg,"moveq    #$%02X,%s",ii,Rn); break;
```

```
        case 0x24: sprintf(msg,"moveta   %s,%s",Rm,Rn); break;
        case 0x25: sprintf(msg,"movefa   %s,%s",Rm,Rn); break;
        case 0x26: sprintf(msg,"movei    #$%08lX,%s",t32,Rn); oplen=6; break;
        case 0x27: sprintf(msg,"loadb    (%s),%s",Rm,Rn); break;

        case 0x28: sprintf(msg,"loadw    (%s),%s",Rm,Rn); break;
        case 0x29: sprintf(msg,"load     (%s),%s",Rm,Rn); break;
        case 0x2A: if(type==GPU_TYPE) sprintf(msg,"loadp    (%s),%s   ;GPU",Rm,Rn);
                   if(type==DSP_TYPE) sprintf(msg,"sat32s   %s   ;DSP",Rn); break;
        case 0x2B:
                   if(ii==0) sprintf(msg,"load     (r14+32),%s",Rn);
                   else sprintf(msg,"load     (r14+%d),%s",ii,Rn);
                   break;

        case 0x2C:
                   if(ii==0) sprintf(msg,"load     (r15+32),%s",Rn);
                   else sprintf(msg,"load     (r15+%d),%s",ii,Rn);
                   break;
        case 0x2D: sprintf(msg,"storeb   %s,(%s)",Rn,Rm); break;
        case 0x2E: sprintf(msg,"storew   %s,(%s)",Rn,Rm); break;
        case 0x2F: sprintf(msg,"store    %s,(%s)",Rn,Rm); break;

        case 0x30: if(type==GPU_TYPE) sprintf(msg,"storep   %s,(%s)   ;GPU",Rn,Rm);
                   if(type==DSP_TYPE) sprintf(msg,"mirror   %s        ;DSP",Rn); break;
        case 0x31:
                   if(ii==0) sprintf(msg,"store    %s,(r14+32)",Rn);
                   else sprintf(msg,"store    %s,(r14+%d)",Rn,ii);
                   break;
        case 0x32:
                   if(ii==0) sprintf(msg,"store    %s,(r15+32)",Rn);
                   else sprintf(msg,"store    %s,(r15+%d)",Rn,ii);
                   break;
        case 0x33: sprintf(msg,"move     PC,%s",Rn); break;

        case 0x34: sprintf(msg,"jump     %s(%s)",CC,Rm); break;
        case 0x35: sprintf(msg,"jr       %s$%04lX",CC,j32); break;
        case 0x36: sprintf(msg,"mmult    %s,%s",Rm,Rn); break;
        case 0x37: sprintf(msg,"mtoi     %s,%s",Rm,Rn); break;

        case 0x38: sprintf(msg,"normi    %s,%s",Rm,Rn); break;
        case 0x39: sprintf(msg,"nop\t\t"); break;
        case 0x3A: sprintf(msg,"load     (r14+%s),%s",Rm,Rn); break;
        case 0x3B: sprintf(msg,"load     (r15+%s),%s",Rm,Rn); break;

        case 0x3C: sprintf(msg,"store    %s,(r14+%s)",Rn,Rm); break;
        case 0x3D: sprintf(msg,"store    %s,(r15+%s)",Rn,Rm); break;
        case 0x3E: if(type==GPU_TYPE) sprintf(msg,"sat24    %s    ;GPU",Rn);
                  //Invalid opcode for the DSP, create a word data assembler directive instead
                   if(type==DSP_TYPE) sprintf(msg,"dc.w     $%04X",d16);
        case 0x3F: if(type==GPU_TYPE) {
                       if     (ii==0) sprintf(msg,"pack     %s     ;GPU",Rn);
                       else if(ii==1) sprintf(msg,"unpack   %s     ;GPU",Rn);
                  //Invalid opcode for the DSP, create a word data assembler directive instead
                       else           sprintf(msg,"dc.w     $%04X",d16);
                   }
                   if (type==DSP_TYPE)
                        if(ii==0) sprintf(msg,"addqmod  #$20,%s    ;DSP",Rn);
                        else sprintf(msg,"addqmod  #$%02X,%s    ;DSP",ii,Rn);
                   break;
    }
    return oplen;
}
```