ECE 477 Final Report
Spring 2007

Team Code Name: Digi-Spi Team ID: 1

Team Members (#1 is Team Leader):

#1: Brad Sokola Signature: _____ Date: _____

#2: Karl Herb Signature: _____ Date: _____

#3: Warren Santner Signature: _____ Date: _____

#4: Justin Lanning Signature: _____ Date: _____

REPORT EVALUATION

Component/Criterion	Score	Multiplier	Points
Abstract	0 1 2 3 4 5 6 7 8 9 10	X 1	
Project Overview and Block Diagram	0 1 2 3 4 5 6 7 8 9 10	X 2	
Team Success Criteria/Fulfillment	0 1 2 3 4 5 6 7 8 9 10	X 2	
Constraint Analysis/Component Selection	0 1 2 3 4 5 6 7 8 9 10	X 2	
Patent Liability Analysis	0 1 2 3 4 5 6 7 8 9 10	X 2	
Reliability and Safety Analysis	0 1 2 3 4 5 6 7 8 9 10	X 2	
Ethical/Environmental Impact Analysis	0 1 2 3 4 5 6 7 8 9 10	X 2	
Packaging Design Considerations	0 1 2 3 4 5 6 7 8 9 10	X 2	
Schematic Design Considerations	0 1 2 3 4 5 6 7 8 9 10	X 2	
PCB Layout Design Considerations	0 1 2 3 4 5 6 7 8 9 10	X 2	
Software Design Considerations	0 1 2 3 4 5 6 7 8 9 10	X 2	
Version 2 Changes	0 1 2 3 4 5 6 7 8 9 10	X 1	
Summary and Conclusions	0 1 2 3 4 5 6 7 8 9 10	X 1	
References	0 1 2 3 4 5 6 7 8 9 10	X 2	
Appendix A: Individual Contributions	0 1 2 3 4 5 6 7 8 9 10	X 4	
Appendix B: Packaging	0 1 2 3 4 5 6 7 8 9 10	X 2	
Appendix C: Schematic	0 1 2 3 4 5 6 7 8 9 10	X 2	
Appendix D: Top & Bottom Copper	0 1 2 3 4 5 6 7 8 9 10	X 2	
Appendix E: Parts List Spreadsheet	0 1 2 3 4 5 6 7 8 9 10	X 2	
Appendix F: Software Listing	0 1 2 3 4 5 6 7 8 9 10	X 2	
Appendix G: FMECA Worksheet	0 1 2 3 4 5 6 7 8 9 10	X 2	
Technical Writing Style	0 1 2 3 4 5 6 7 8 9 10	X 8	
CD of Project Website	0 1 2 3 4 5 6 7 8 9 10	X 1	
		TOTAL	

Comments:

TABLE OF CONTENTS

Abstract

This report details the semester long development of the Digi-Spi intelligent listening device. The device is an audio surveillance system that is housed discreetly inside of a lamp. The device edits out non-speech audio and allows the user to save this audio to an SD card and play the audio back for listening using a headphone jack. The device can be accessed to playback live audio via a cellular phone module. Recorded audio from an SD card can be accessed by listening to playback from the headphone jack. The desire to create a simple surveillance device incorporating audio processing was the motivation behind this project. Virtually the entire design process from component selection through future "version 2" changes is considered in this document.

1.0 Project Overview and Block Diagram

The Digi-Spi is an intelligent audio surveillance system that is housed discreetly inside of a lamp.

Figure 1.1 – Photograph of Completed Digi-Spi

The Digi-Spi system has a microphone jack for audio input. As audio is captured the Digi-Spi makes a determination as to whether the audio contains speech or not. If captured audio is determined to contain speech, it is stored on an SD card. The Digi-Spi can be accessed in three ways. First, a GSM module that allows users to call in and listen live. Second, the SD card can be removed. Finally, users can play back the audio and listen through the headphone jack. The intended application for the Digi-Spi is as a discreet form of home surveillance. Parents could place the Digi-Spi in a common room and then would be able to call the device to hear live audio or review past speech captured on the SD card. This would be especially useful for parents wary of leaving their children alone in the house.

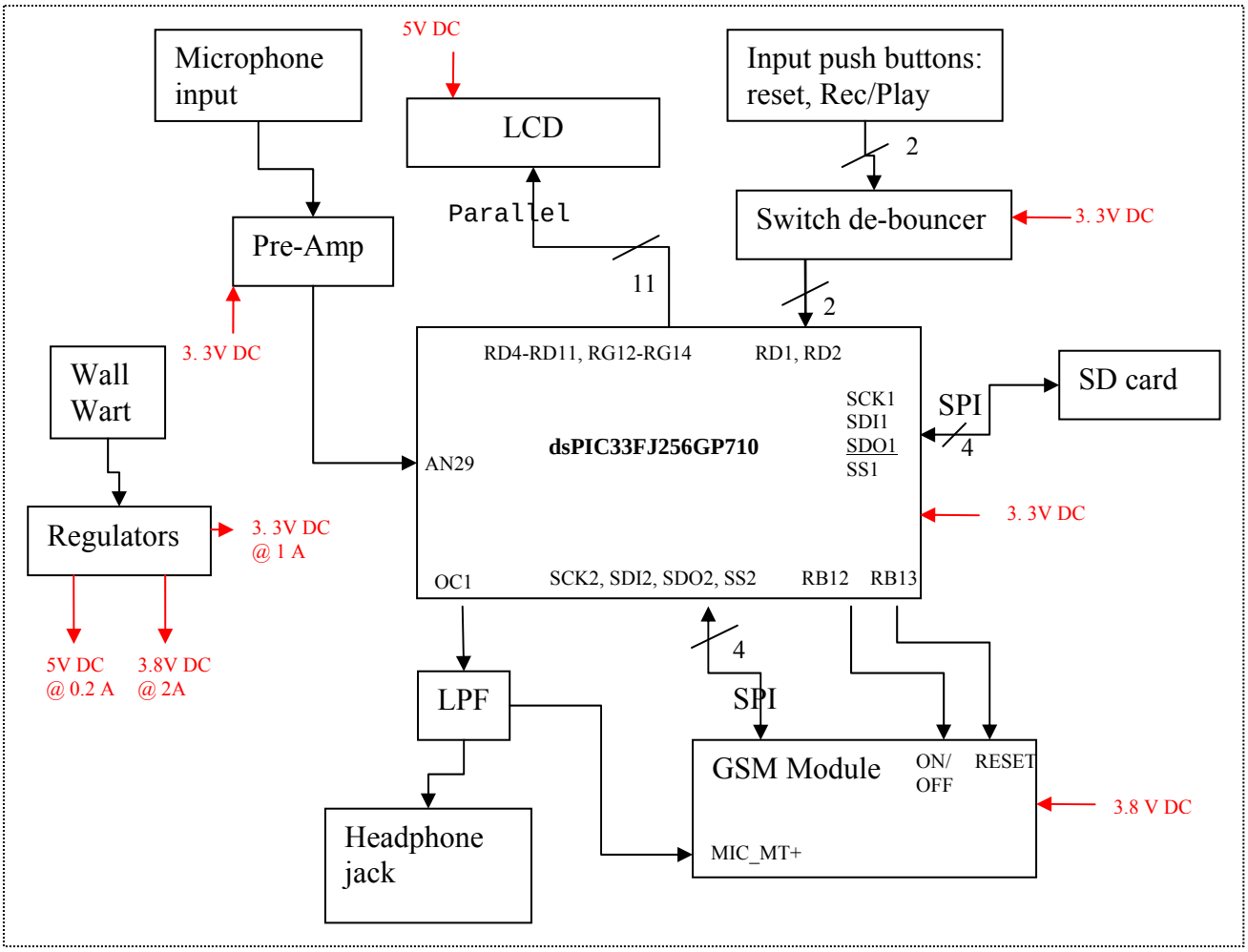


Figure 1.2 – Block Diagram

Figure 2, above, shows a block diagram of the final system including the pin usage, interfaces, relevant supply voltages, and regulator requirements. The decisions necessary to realize the successful development of this device will be considered in the following sections. Note that when PWM is mentioned in the following sections, it refers to the pin OC1.

2.0 Team Success Criteria and Fulfillment

1. An ability to record an audio signal

   The Digi-Spi device successfully samples audio from an A-to-D port. This is demonstrated by outputting the audio sample with the PWM.

2. An ability to modify or filter an audio signal

   The Digi-Spi device contains an algorithm using threshold detection to determine if speech is present, and displays if the device is "Listening" or "Recording" based upon the signal relation to the threshold. If the threshold is exceeded it is recorded to the SD card. This is demonstrated by the playback of one's voice in which the pauses during their speech are removed.

3. An ability to display and modify current settings via the LCD and push-buttons

   The LCD is able to display the current recording status (recording, listening, or playback), the current status of the cellular module, the current microphone being used, and the up-time of the device. The user can change the device from recording mode to playback mode by pressing one push button. The other button allows the user to reset the device which returns the up-time to 0 and initializes any SD card operations to start at the first block of data.

4. An ability to save speech signals to external memory

   An SPI interface to the SD card is implemented and this is shown by successfully initializing the card, outputting data to it, and re-obtaining it. This is demonstrated alone with demonstration boards, and also in the project by saving the audio signal from the A-to-D and then being able to re-obtain it for output on the PWM.

5. An ability to play back the recorded sound (via headphone jack or line-level output)

   The Digi-Spi device is able to play back what is stored on the SD card through the PWM pin output electrically connected to a low pass filter and a headphone jack. This is demonstrated by leaving the device in recording mode and speaking into the microphone. Then the device is switched to playback mode, and the SD card's recorded speech is heard on the headphones.

3.0 Constraint Analysis and Component Selection

The Digi-Spi device was designed to integrate a microcontroller with digital signal processing capabilities, an LCD driven user interface, a Bluetooth host controller, an 802.11b/g wireless web-server, a GSM Cellular Modem, an SD Memory Card interface, and analog line-level input/output. In the following section, the important design constraints the group considered will be discussed.

3.1 Design Constraint Analysis

The following sections summarize the computation, interface, on-chip and off-chip peripherals, power, cost, and packaging constraints and requirements for the project. All of these criteria were considered when selecting parts and designing the embedded application.

The focus of this project's constraints was interface and peripheral requirements. Interface requirements were critical due to the number of different peripherals that were being supported. Serial and SPI capabilities are important for communication between these devices. Microcontroller selection was also heavily considered to insure that signal processing algorithms for speech processing could be completed in real-time.

3.1.1 Computation Requirements

The computational requirements of the project can be broken into: digital input/output sample buffering, and signal processing computation and transfer to external memory.

In order for the project to be useful as an audio listening and processing device, it is necessary for the embedded application to store audio samples for future calculations. The samples will be 8 bits wide. Samples will be buffered into SRAM with a frequency of 8 kHz. The speech detection algorithm makes a determination of whether speech is present on a block by block basis. The initial version of the algorithm used up to half a second or 4000 bytes of data to make a determination. The final algorithm used only 512 bytes of data (roughly 0.06 seconds of audio) to make a determination. A circular buffer of at least 2 blocks is necessary to allow the processing of each block to be completed without having to worry about data being over written before it is stored to the external memory card. This means that the audio processing requires a minimum of 1024 bytes of memory.

The signal processing algorithm and output to the SD card are embedded into the main program loop, while the sample buffering is interrupt-driven. When a full block is ready to be processed, a flag is set by the interrupt and processed in the main loop. The signal processing

algorithm consists simply of comparing the average energy of the block to a threshold set at system start ups and at resets. From an experimental test of our code, the comparison and a 512 byte write to the SD card took roughly 10ms to complete. Since it takes roughly 60ms for a block to be sampled this gives plenty of time for a block to be processed before a new one is ready.

3.1.2 Interface Requirements

To accommodate the spectrum of peripherals external to the microcontroller, a variety of general purpose I/O systems will be necessary. A summary of the I/O requirements is provided in Figure 1.

The SPI bus is utilized to interface the microcontroller with the WiPort Wi-Fi Server, Telit GSM Cellular Module, and Secure Digital Memory Card [1], [2], [3]. Communication with the Ezurio Bluetooth Module was designed to be accomplished with the standard RS-232 SCI protocol [4]. Two general purpose digital inputs were used for operator interface control. The operator interface display (16 x 2 LCD) utilized an 8-bit parallel interface [5]. Finally, an analog ADC input accommodated microphone input, while a PWM output allowed for digital-to-analog conversion.

The design requires a total of 27 I/O pins. Two separate SPI lines were utilized because the GSM module's SPI interface is implemented in Python and only allows operation in master mode. So the microcontroller has one SPI port configured in master mode and one in slave mode.

I/O Requirements		
Peripheral Description	**I/O Interface**	**Number of Pins**
Wiport 802.11b/g Wi-Fi Server	SCI (RS-232)	2 (input, output)
Hitachi 16x2 LCD	Parallel	11(8 data lines, chip select, clock)
Telit GSM Cellular Module	SPI	4 (input, output, chip select, clock)
Ezurio BISM2 Bluetooth Module	SCI (RS-232)	2 (input, output)
Secure Digital Card	SPI	4 (input, output, chip select, clock)
Maxim Switch Debouncer	GPIO	2 (CMOS level)
Analog Audio Input	A-to-D	1
Analog Audio Output	PWM	1

Figure 3.1 - I/O Requirements

3.1.3 On-Chip Peripheral Requirements

The on-chip peripheral requirements are dictated by the selection of external components outlined in this document. As described in section 2.2 and Figure 1.1, five on-chip peripherals were necessary: SPI, RS-232 Serial, 1 channel of 10-bit ATD, 1 channel of 8-bit PWM, and 13 channels of general purpose I/O.

3.1.4 Off-Chip Peripheral Requirements

To provide the extended and flexible functionality of the embedded system, several off-chip peripherals were necessary. Wireless communication was designed to be managed through three off-chip devices: The WiPort 802.11b/g Wi-Fi Server, the Telit GSM Cellular Module, and the Ezurio BISM2 Bluetooth Module [1], [2], [4]. A Secure Digital (SD) Card provides extended non-volatile memory for audio storage [3].

3.1.5 Power Constraints

Again, the variety of embedded components created several power management constraints. The Wiport Wi-Fi Server, Bluetooth Module, and dsPIC Microcontroller can all operate at a nominal 3.3V input [1], [4], [6] . The GSM cellular, though, must have a voltage supply of 3.4 to 4.2 volts [2]. Finally, the Hitachi LCD operates at 5V [5]. Special care must also be taken to accommodate large power consumption spikes when the wireless radios are transmitting. The WiPort and GSM Cellular modem can have current pulses of 400 and 1900 milliamps respectively [1], [2].

The embedded system will be stationary and powered through a “wall wart” and three linear regulators running at 3.3V, 3.8V, and 5V. The 3.3V and 3.8V are derived from two LT1528s [7]. The 5V is provided by an LT1121 [23].

3.1.6 Packaging Constraints

This surveillance system is intended to be hidden in household furniture, specifically a lamp. The footprint of the device must accommodate this requirement. Also, no active cooling can be utilized to maintain the “stealth” silence of the system.

3.1.7 Cost Constraints

The Digi-Spi will be competing in a niche marketing focused on individuals and companies that require high-end remote surveillance needs. Baby monitoring systems enable

close range monitoring in the $100 price range, but they do not integrate the technology to enable surveillance from anywhere in the world.

3.2 Component Selection Rationale

3.2.1 Microprocessor Selection

Early in the design process, a criterion was created for the selection of the microcontroller. First, selecting a platform that the team was familiar with was determined to be critical. This immediately narrowed the field of microcontrollers to Freescale and Microchip.

Next, to accommodate the buffering and processing of audio, the amount of onboard RAM was considered. The Freescale MC9S12 E128 has 16K of RAM, while the MicroChip dsPIC33FJ256GP710 has 30K of SRAM. SPI and RS-232 Serial interfaces are also required, but both processors provide ample support for the necessary serial communication and GPIO [8], [6]. The economic factors of the microcontroller were also carefully weighed. The MC9S12 E128 can be purchased from DigiKey for $10.48, while the dsPIC costs $19.80 in single unit quantities.

The determining factor, though, came down to the vast possibilities for the integrated DSP functionality of the dsPIC. The algorithm for voiced/unvoiced speech detection was developed in parallel with the hardware and thus the computation requirements were unknown at the time of microcontroller selection. The dsPIC has single-cycle multiplies and built-in DSP functions that would be more than sufficient for any algorithm developed [6].

3.2.2 Bluetooth Module Selection

The Ezurio BISM2 was compared to the MITSUMI Bluetooth WML-C40 module. The two modules provided very similar functionality. Both were Bluetooth 2.0 qualified with RS-232 serial interfaces. A PCM audio interface is also present on both devices. The similarities continue with their integrated ceramic antennas and 300 kbps max data transfer rate [4], [9].

There are some differentiating qualities, though. The current draw on the Ezurio model is 30 mA, compared to 90 mA for the MITSUMI [4], [9]. The Ezurio also handles a wider input voltage range: 3.3V-6.0V compared to 3.2V-4.0V. The BISM2, though, costs $99.99 in single-unit quantities while the MITSUMI is only $64.95 in the same quantities.

The deciding factor, though, was the availability of documentation and support. Ezurio's website has easy access to all the necessary technical documentation. MITSUMI, on the other hand has an English Website that is under construction and individual requests must be made for

all documentation. Considering the condensed timeline of the project, the Ezurio was chosen for its superior support and lower power consumption despite its higher cost.

3.2.3 Wi-Fi Module Selection

The project constraints dictate a small, discrete form factor that can be hidden in some household or office furniture. As a result, 802.11b/g Ethernet bridges were immediately taken out of consideration. Initial research revealed an embedded wireless server from Lantronix that provided 802.11b/g connectivity with flexible RS-232 Serial and SPI interfaces as well as 11 GPIO pins [1].

Continued research revealed that a Purdue University ECE Senior Design Team in the fall of 2006 investigated a competing product, the DPAC Technologies Airborne WLNG Adaptor [12]. The DPAC Technologies Airborne, though, draws more power (575 mA compared to 400 mA), lacks a flexible Ethernet port, and costs more compared to the Wiport [10], [11]. Both modules do contain a surplus of GPIO pins that will be unnecessary for the scope of this project. The best solution, based on cost, power constraints, and flexibility is clearly provided by Lantronix.

3.3 Summary

The success of this project relies on seamless and reliable system integration. The desire to be very versatile lead to the selection of a microcontroller (the dsPIC33F) with more memory and processing power than our project would likely need. The desire for seamless integration led to the selection of a Bluetooth module with documentation in English. Finally, a Wi-Fi module, the Lantronix Wiport was selected based on cost, lower power and interface flexibility.

4.0 Patent Liability Analysis

There are a few areas of potential patent infringement. The primary area for infringement is the speech detection algorithm; however, the file format used to store audio, and the file system used on the external memory will also be considered. The rest of the device's functionality comes primarily from interfacing with components that would have any potential licensing fees already included in their price. Since the speech detection functionality appears to have the most potential for infringement, this report will focus primarily on that aspect of the device. However, a brief analysis regarding the file format and file system is included.

The system will store the sample values directly onto the SD card. When the user removes the SD card and uses the card reader and software provided, the sample values will be written to a Sun Audio file for the user to listen to. The Sun Audio (.au) file format is an open file format, so there should not be any potential for infringement. The Digi-Spi device will not be utilizing a recognized file system on the SD card, but instead will simply be writing directly to memory and then will have a card reader and software on the user's computer that will output the audio files. Since no real file system is being used, there is no way of infringing on just writing data to an SD card.

In order to assess the potential patent infringement of the speech detection algorithm, specifics about how the Digi-Spi speech algorithm functions must be known. The speech detection algorithm consists of a simple threshold comparison. A threshold is set based on the average level over roughly a second when the device is turned on. An offset is added to that threshold to prevent random variation in noise levels from triggering the device to record. Finally, blocks of audio samples are compared to the threshold and a determination is made.

4.1 Results of Patent and Product Search

A number of recording devices advertising voice activated modes were found. Two representative commercial devices are included in the following sections. Many patents involving speech detection were found. An analysis of the three most similar to Digi-Spi device has been included. Patents were found using Google Patent Search with terms "speech", "detection", and "voice activity detection".

4.1.1 33 Hour Mini Digital Recorder [13]

This device is a small recording device that features a voice activated record mode. According to the product description, "the easy to use Voice Activated Mode will only begin

recording when a sound is heard and it will stop recording after a few seconds where no sounds are present" [13]. It is difficult to tell how the Spytronix device actually determines voiced sound from unvoiced sounds. It appears most likely that this would be implemented with a simple fixed threshold to determine if voice is present. There is no indication of a patent associated with this device on the website cited.

4.1.2 Coby – CX-R55 Voice-activated cassette recorder [14]

This is a small, battery powered cassette recorder that features a, "Voice Activated system with Adjustable Sensitivity control" [14]. It features a built in microphone as well as a jack for an external one. The adjustable sensitivity control appears to change the threshold that the voice activated system uses. Once again there is little indication of how the device actually decides if audio is voiced or not. This device likely also just uses a threshold comparison. However, in this case, the threshold can be changed via the adjustable sensitivity control.

4.1.3 VAD/CNG software [15]

This product is a software package that producers of VoIP phone systems could include in their product. A number of VoIP phones, such as the Cisco SPA901 [16], were found that mentioned voice activity detection as one of the voice features. Unfortunately, none of the products found had a description of what the voice activity detection consisted of. This lead to the determination that a software package marketed towards VoIP phone manufacturers would perhaps be a better product to look at in terms of analyzing the actual functioning of the detection. Although the description of the VAD software is still sparse, it does state that the VAD algorithm "analyses voice activity to detect silence intervals". The detecting of silence intervals would seem to indicate some type of threshold comparison.

4.1.4 US Patent number 6,453,285, Issued September 17th, 2002, "Speech activity detector for use in noise reduction system" [17]

A system and method for removing noise from a signal containing speech (or a related, information carrying signal) and noise. A speech or voice activity detector (VAD) is provided for detecting whether speech signals are present in individual time frames of an input signal. The VAD comprises a speech detector that receives as input the input signal and examines the input signal in order to generate a plurality of statistics that represent characteristics indicative of the presence or absence of speech in a time frame of the input signal, and generates an output based on the plurality of statistics representing a likelihood of

speech presence in a current time frame; and a state machine coupled to the speech detector and having a plurality of states. The state machine receives as input the output of the speech detector and transitions between the plurality of states based on a state at a previous time frame and the output of the speech detector for the current time frame.

This patent uses a variety of statistics to determine the likelihood of speech. The key claim for this patent is claim 1. Claim 1 describes these statistics as including an energy change statistic and a spectral deviation change statistic.

4.1.5 US Patent number 6,757,651, Issued June 29th, 2004, "Speech detection system and method" [18]

The method first receives a sound signal and determines if the energy value of the sound signal is above a threshold energy value. If the energy level of the signal is above the threshold energy value, the method determines a predictive signal of the received signal, subtracts the predictive signal from the signal, and determines if the result of the subtraction indicates the presence of speech. If it is determined that no presence of speech is indicated, the threshold energy value is set to the energy level of the present received signal. If it is determined that the result of the subtraction indicates the presence of speech, the received signal is sent to a speech recognition engine.

The second patent detects speech based on an initial threshold and generation of and comparison to a predictive signal. The key claim for this patent is claim 1. This claim mentions using a threshold and also using a prediction algorithm to determine if the audio contains speech.

4.1.6 US Patent number 5,826,230, Issued October 20th, 1998, "Speech detection device." [19]

The device detects the beginning and ending portions of speech contained within an input signal based on the variance of smoothed frequency band limited energy and the history of the smoothed frequency band limited energy within the signal. The use of the variance allows detection which is relatively independent of an absolute signal-to-noise ratio with the signal, and allows accurate detection within a wide variety of backgrounds such as music, motor noise, and background noise, such as other voices.

The third patent detects speech based on frequency band energies. In this case, Claim 2 is the most important. It mentions a means for selecting portions of the signal having frequencies within a pre-selected range and a means to determine the values of frequency band limited energy [19].

4.2 Analysis of Patent Liability

4.2.1 Literal Infringement

According to an article from *JOM* about patent infringement, "Every requirement of each claim must be considered to see if each thing set out in the claim also appears in the accused practice. If one or more things set forth in a claim is not present in the practice being reviewed, there is no infringement of that claim" [20]. Using this as a basis, the most similar and relevant claim for each patent is considered and then it is determined whether what is set out in that claim also appears in the Digi-Spi. If the most similar and relevant claim differs from the product being considered, then no literal infringement exists.

The most similar claim for the first patent discussed mentioned energy change and spectral deviation change statistics. The Digi-Spi device uses only an average energy statistic over a block of data, with no carry over between blocks. Thus it does not literally infringe on this claim.

The most similar claim for the second patent discussed mentioned a threshold comparison and the generation of a predictive signal for comparison. The Digi-Spi does not generate a predictive signal of any kind, so it does not infringe on this claim.

The third patent considered mentions detection based on frequency band energies. Its first claim refers to a "means for determining a variance of smoothed frequency band limited energy; and means for determining the beginning and ending points of speech within the signal based on the variance of the smoothed frequency band limited energy and past history of the smoothed frequency band limited energy" [19]. The Digi-Spi device only uses an energy threshold comparison. This does not include what is mentioned in claim 1 and thus does not literally infringe.

Based on this analysis, the Digi-Spi device does not literally infringe on any of the speech detection patents discussed.

4.2.2 Infringement under the Doctrine of Equivalents

Under the doctrine of equivalents, a device can be considered to be infringing on a patent if it performs substantially the same function in substantially the same way even if it does not literally infringe on the claims.

The first patent analyzed [17] mentions using energy change and spectral change statistics to determine a likelihood of speech. Since the Digi-Spi uses a much simpler algorithm

that consists only of comparison to an average energy threshold, the detection of speech appears to be done in a substantially different way.

The second patent analyzed [18] mentions a predictive comparison in addition to the initial threshold comparison. This is also significantly different than the energy threshold that the Digi-Spi uses to determine which parts of the input audio contain speech.

The third patent analyzed [19] utilizes frequency band energies to determine if speech is present. The Digi-Spi does not make any estimation of frequency content and therefore appears to be significantly different from the algorithm described in this patent.

4.3 Summary and Action Recommended

In terms of the speech detection algorithm, there appears to be reasonably low likelihood of any infringement on the patents discussed. This is because of the relatively simplistic nature of the algorithm being used compared to the patents discussed. Also, the amount of prior art involving use of thresholds for detection seems to make any infringement very unlikely. Important to note, however, is that any change in future detection algorithms used by the device could impact this status.

Two issues make it unlikely that our product will encounter any infringement issues. First of all, our product will likely be a niche product, so economic gain from any type of infringement suit would be small for the other company. Second, there is a wide range of prior art involving speech detection utilizing energy thresholds making any case of infringement difficult.

5.0 Reliability and Safety Analysis

When considering the reliability of the Digi-Spi components, it is logical to immediately look into the three "wireless" modules utilized in its design. However, due to the "black box" nature of these devices, they do not lend themselves to easy reliability analysis and have therefore been excluded from the following report. The remaining components within the Digi-Spi that will pose reliability issues are the microcontroller, LT1528 LDO voltage regulator, LT1121 LDO voltage regulator, and microphone pre-amplifier. The failure rate and mean time to failure will be discussed for each of these microcircuits, as well as possible measures to increase their reliability.

Essential to the safety and quality of any device is prior consideration of how this device could possibly fail. By doing this analysis during the design process, an engineer can take measures to ensure that their product fails in a safe and predictable manor. This analysis is best done by breaking the device schematic into functional blocks and looking at failures within each section. The blocks to be considered within the Digi-Spi are the microcontroller and off-board peripherals, the audio and general purpose I/O, the on-board "wireless" modules, and the power supply. The power supply block is most critical to the design because it has the highest potential for causing harm to the user or irreparable damage to other components.

5.1 Reliability Analysis

An important part of any design process is a thorough reliability analysis, where the potential weak points of the circuit are analyzed, and their failure rates are estimated. Aside from its "wireless" modules, the Digi-Spi consists of number of components worth consideration; these include the dsPIC33F microcontroller, LT1528 LDO voltage regulator, LT1121 LDO voltage regulator, and MAX9812L microphone pre-amplifier. These components were chosen because aside from the "wireless" modules, which do not lend themselves to such an analysis, they make up the primary non-passive elements in the circuit.

The Military Handbook for Probability Prediction of Electronic Equipment [21] was used to calculate the failure rate (λp) and mean time to failure (MTTF) for each component. All equations and parameter values were chosen based on the guidelines outlined in this document. The four parts to be analyzed all fall under the MIL-HDBK-217F microcircuit model. The following equation defines this model: $\lambda_P = (C_1\pi_T + C_2\pi_E)\pi_Q\pi_L$. C_1 is the microcircuit die

complexity failure rate, π_T is the temperature factor, C_2 is the package failure rate, π_E is the environmental factor, π_Q is the quality factor, and π_L is the learning factor (measure of chip manufacturing maturity).

For each of the four components π_E, π_Q, and π_L are all the same value due to a number of universal assumptions. The first of these being that the Digi-Spi is assumed to be operating in a temperature and humidity controlled environment with a fixed ground. Table 3-2 of the MIL-HDBK-217F defines these conditions to be ground benign, giving all components a π_E value of 0.5 (MIL-HDBK-217F Section 5.10). The second assumption is that the Digi-Spi will have the quality of a commercial product, giving them a π_Q value of 10.00 (MIL-HDBK-217F Section 5.10). The final assumption is that all of these components have been in production over two years, giving them a π_L value of 1.00 (MIL-HDBK-217F Section 5.10). Having stated these universal assumptions, the only variables left to consider are C_1, π_T, and C_2. The following set of tables outlines the λp and MTTF calculation for each selected component. All justifications for the C_1, π_T, and C_2 values are from the component data sheets unless they are expressly assumed.

Parameter	Value	Justification
C_1	0.56	16-bit with 40-bit extended precision, Assumed 32-bit MOS microprocessor
π_T	0.95	Digital MOS, T_J = +85 degrees C
C_2	0.13	100 pin TQFP, eq. 3 was used = 3.0 E 5 * 100^1.82
π_E	0.50	Assumed ground benign condition
π_Q	10.00	Commercial component
π_L	1.00	Years in production ≥ 2.0

λp	5.97 Failures/Million Hours
MTTF	1.675E+5 hrs or 19.12 years

Table 5.1 dsPIC33FJ256GP710 [6] MTTF Parameters

Parameter	Value	Justification
C_1	0.02	Assumed Linear MOS device, # of Transistors = 264
π_T	32.00	BiCMOS, T_J = +150 degrees C

C_2	0.002	Hermetic 6-pin SC70-6 package, eq. 1 = 2.8 E -4 * 6^1.08
π_E	0.50	Assumed ground benign condition
π_Q	10.00	Commercial component
π_L	1.00	Years in production ≥ 2.0

λp	6.41 Failures/Million Hours
MTTF	1.56E+5 hrs or 17.81 years

Table 5.2 MAX9812L Microphone Pre-Amplifier [22] MTTF Parameters

Parameter	Value	Justification
C_1	0.01	Linear MOS device, # of Transistors assumed to be < 100
π_T	180	Linear MOS, T_J = +150 degrees C
C_2	.00092	Hermetic 3-pin SOT23-6 package, eq. 1 = 2.8 E -4 * 3^1.08
π_E	0.50	Assumed ground benign condition
π_Q	10.00	Commercial component
π_L	1.00	Years in production ≥ 2.0

λp	18.00 Failures/Million Hours
MTTF	5.56E+4 hrs or 6.34 years

Table 5.3 LT1121 LDO Linear Regulator [23] MTTF Parameters

Parameter	Value	Justification
C_1	0.01	Linear MOS device, # of Transistors assumed to be < 100
π_T	58.0	Linear MOS, T_J = +125 degrees C
C_2	.0016	Hermetic 5-pin DD package, eq. 1 = 2.8 E -4 * 5^1.08
π_E	0.50	Assumed ground benign condition
π_Q	10.00	Commercial component
π_L	1.00	Years in production ≥ 2.0

λp	5.81 Failures/Million Hours
MTTF	1.72E+5 hrs or 19.63 years

Table 5.4 LT1528 LDO Linear Regulator [7] MTTF Parameters

Overall, most of these components seem to have a sufficient MTTF taking into account the scope of the Digi-Spi's function (less than 10 years), with the exception being the LT1121 (6.34 years MTTF). A regulator with a smaller worst case junction temperature would significantly increase the MTTF, and is something that should definitely be looked at in future designs. While the other components seem to have sufficient MTTF, it is still important to take steps in order to decrease this number because of the fact that the failure rate is constant for a chip's lifespan. By properly heat sinking these devices the worst case junction temperature can be greatly reduced. This is especially relevant to the MAX9812L, LT1528, and LT1121 which have π_T values of 32.0, 58.0, and 180 respectively. As for the dsPIC33F, one possible (but small) improvement that could be made would be to reconsider its model number to see a part with a smaller number of pins can be found that will still meet our device's needs.

5.2 Failure Mode, Effects, and Criticality Analysis (FMECA)

The functional block breakdown of the Digi-Spi schematic can be viewed in Appendix A. These blocks include Microcontroller and Off-Board Peripherals (A), Audio and GPIO (B), On-Board "Wireless" Modules (C), and Power Management (D). The following criticality definitions will be used for this analysis. High criticality is defined as anything potentially harmful to the user; this type of failure has a $\lambda p \leq 10^{-9}$. Medium criticality is defined as any failure that affects the device's primary function of recording speech or causes irreparable damage to any component; this includes any failures in the range of $10^{-9} < \lambda p < 10^{-6}$. Finally, low criticality is defined as any failure that partially limits device function but does not affect the recording of speech or causes reparable damage; this includes failures with $10^{-6} < \lambda p < 10^{-3}$.

A full FMECA report can be viewed in Appendix G; after investigation of this report it is obvious that the Power Management Block is most critical to safety of operation. Every failure within this block has a rating of at least medium criticality because of the block's ability to propagate its failures to other systems. Also, it should be noted that the three failures of Block D can be viewed as six failures since there are two different supply voltages. Depending on which supply voltage is affected, the consequences throughout the circuit are quite different. Within this block a way of addressing the high criticality failure of excess voltage would be to place a zener diode on the wall wart output in order to limit the supply voltage to an acceptable value. In regards to other blocks, it is planned to use the web server interface and LCD screen to

provide feedback to the user on the status of components such as the SD card, Bluetooth module, GSM module, and Wiport.

5.3 Summary

After analyzing the reliability and safety of the Digi-Spi, it is clear that component failures and system failures modes must always be on the product developer's mind. The parts reviewed in the reliability section, while not possibly the most critical considering the inability to perform this analysis on the Digi-Spi's larger modules, provide a good insight into the expected product lifetime. Steps can be made to increase their reliability by properly heat sinking the components, thus reducing the temperature factor in each. This will have a profound affect on the MAX9812L and the LT1528. As for the FMEC analysis, measures can be put in place during the design process to ensure that the most critical failures occur with minimal probability. The block of most importance, Power Management, can be given a better chance to fail safely by placing protection circuitry on the wall wart output, therefore reducing the likelihood of device overheating and possible harm to the user.

6.0 Ethical and Environmental Impact Analysis

The Digi-Spi device is recording voice data. The potential misuse of this data is an important ethical concern. Environmentally, various pollutants and power utilizations are considered. The following discusses these concerns and analyzes the ethical and environmental impact of the Digi-Spi remote listening device.

6.1 Ethical Impact Analysis

Since the Digi-Spi is a listening device, the natural ethical question stemming from this is the use of the recorded voice data. This is a current topic seen in the media, with the current United States government security measures under question. Furthermore, the data being recorded can be streamed in real time over the air using a cell phone or computer connection. The data is then stored in persistent storage in a well-defined format within a portable and scaleable component: the SD card. This ability to stream and record what was said when can be of positive use. Ethically, it is important to examine how this utilization can be a negative aspect.

To solve this ethical issue, restricted access and operating environment of the device is essential. When the target market of the device under development (if it would later be produced) is to be a toy or amateur novelty application, a liability for 'clean' use does not exist. If the device is targeted for professional or military use, however, the device would need to be more secure. Fortunately, the device is not targeted for this market, and therefore a heightened level of security is not essential. If the device was to be made more secure, it would ideally have a single phone number, have a lock on the SD card, have its wireless network uniquely encrypted, and have a digital lock on the control panel. Hence, in order to avoid this potential issue of using a non-secure and amateur device in a secure or professional area, a warning label must be placed on the device saying it is not intended for any professional or military use, but solely as a novelty. This would also be stated in the user manual. In addition, the manufacturer and even the developers of the device are held responsible when the component is used in a negative manner. Therefore, in the user manual, there will be a statement claiming that the developers are not reliable for any harm brought upon someone through an unethical use of the device. This would even include reusing the data on a computer to manipulate it, or hold it against a person.

Adding to the military and professional use discussion, the device is not designed for rigorous use. Ethically, a device used in a professional environment needs to be fail safe and highly portable to operate under a variety of environmental conditions. In addition its software design needs to be secure, scaleable, and safe from tampering. A ***Military Embedded Articles*** states that the C language is not favorable for the development of secure applications [25]. Other software languages allow for easier designing of secure software. If this device were to be designed for use in a professional setting, the choice of programming language or the construction of the software should be reconsidered with an eye towards security. Therefore, in addition to the warning labels and user manual outlined above, the device has been designed to fit into a common household item and function off of a wall wart. In addition, if the product was to be marketed, the system would be sealed and unable to be opened (outside of removing the SD card). This way the PCB could not be removed and used in a different application. To prevent this issue, the user manual will state that the developers are not responsible when the PCB is used in a different application than what it was designed for.

Finally, the ethical issues of electricity, power, and electrical signals need to be considered. The PCB is enclosed in a common household device, such as a lamp, to protect it from the elements. As stated above, if the device would be manufactured, it would be sealed. This not only deters reuse, but keeps the device clean and reduces a risk of electrical injury. The worst issue is electro-magnetic-fields (EMF) from the GSM and WiPort. In fact, an ***EE Times*** article states that "pulsed electromagnetic energy generated by electronic devices causes stress to the human body" [26]. The EMF generated is part of the WiPort and GSM module, and therefore their operation is kept within there limits. To aid with this, a warning label will be placed on the device noting the EMF factor and in the user manual a note will be made stating that while the device abides by all relevant regulations regarding EMF emissions there is still a potential hazard.

6.2 Environmental Impact Analysis

This device is designed to be a novelty item and not for a specific portable or professional application. Therefore, the worst environmental impacts come from the manufacturing and disposal of the PCB and components.

During manufacturing, the environmental impact of the device is due to the PCB manufacturing. PCB manufacturing consists of etching of the traces onto a metal. This uses

numerous natural resources including metals (copper, gold, nickel, and tin) and water [27]. In addition, the process "generates large amounts of metal-bearing effluent solutions that require expensive treatment before they are discharged into the environment" [27]. Fortunately, this product is only a prototype, but if the product were put into production, it would be necessary to consider the cost of alternative PCB manufacturing technologies that reuse the manufacture metals and reduce pollution. With effective capturing procedures, the large amount of metal-bearing solution is less likely to be released into the environment, and a portion can be reused for another PCB.

Once the PCB is etched and populated, it is operational. During the operational phase the main environmental impact of the product is EMF from the wireless modules. This so called "eSmog" has an environmental impact in the sense that it harms living organisms, including humans, in the environment [26]. The radiation of EMF can cause "breaks in DNA sequences and inhibit the body's ability to repair these breaks" [26]. For this reason, there are the ethical liability and labels mentioned above. These also double to warn the user that the device can be harmful. Unfortunately, the data for EMF harm is still not fully backed, but it could be a potential risk. In addition, our device is small and only uses Wi-Fi, Bluetooth, and cellular modules. These will not come in close proximity to a human as a cellular phone would. If this environmental need would be addressed and the device is manufactured, in addition to the labeling it would be best to include an optional Ethernet connection with the ability to disable the wireless.

Over time, the device will breakdown, and when it is time for disposable, it would be best to recycle the device. Efficient disposal is an important step to prevent potentially harmful e-waste from just being put in the trash. This is evident from a military document summary that states PCBs "generally have no usefulness once they are removed from the electrical component in which they were installed" [28]. In our case, this would be when it is no longer powered and the board has burnt-out. The large amounts of lead and the energy used to make the device should not be wasted. Instead, companies exist that recycle the device, such as NEC Corp. in Japan [29]. This plant separates the components of the PCB and materials, including copper, are reclaimed from the board. This is in the hopes of reusing the materials. Clearly, this is not an inexpensive procedure, and for a single board, this would not be the best solution. For a set of manufactured boards, it would be ideal. For this reason, the user manual will contain a return

to manufacturer label for proper disposal and a warning label will be on the package as well. This way the PCB can be recycled, or taken care of if another avenue exists at the time.

For disposal of the remaining device, i.e. the lamp and plastics, it would be best to recycle these items. Unfortunately, much of the materials are thrown away in American society, but it would be best to reuse any parts, such as screws and spacers, for future use. Therefore, the disposal of the entire package will be marked as returnable to the manufacturer for proper disposal. In order to not be offensive and incur any costs, the warning label is only a suggestion for a marketable product. But, this is a single prototype, and therefore the return to manufacturer is a demand.

6.3 Summary

As is shown, the Digi-Spi has several ethical and environmental issues that must be addressed. Its impact ranges from basic PCB recycling to the ethics of recording sound data. With a correct design focus for amateur use and warning labels this device will not pose an ethical liability. By adding to that a non-battery design, involvement in device disposal, and alternative manufacturing methods the device will have a low impact on the environment as well.

7.0 Packaging Design Considerations

The DigiSpi is enclosed inside a typical household lamp. The lamp will feature a hidden compartment where the DigiSpi LCD can be viewed, SD Card accessed, and recording turned on/off by way of push buttons.

7.1 Commercial Product Packaging

After considerable market research two families of similar product types emerged: the compact digital voice recorder and the ultra high frequency transmitter-receiver surveillance system. The specific products chosen to represent these types are the Spytronix 33hr Mini Digital Recorder [13], and the MSCSpytek UHF Transmitter-Receiver [30].

7.2 Spytronix 33hr Mini Digital Recorder

The Spytronix Mini Digital Recorder, as seen in Figure 1, is a very compact and portable voice recorder. The design is very simple having only one external push button and a small, one-line LCD display which shows the current recording status. Also not visible in the graphic is the USB jack, which is used to download the recorded messages to the user's PC, and the built-in speaker which can be used for instant playback. The device runs on 2 AA batteries with a battery life of approximately 12 hrs [13]. The compact design of this device makes it very portable and concealable, and the external interface is quite simple since it only involves one push button and a small LCD display. However, one obvious negative of this product's design is that it will not be confused for anything other than a digital recorder, thus if found by the person being spied on, they would immediately know the situation. Also its single push button design allows for only limited recording options, as there is no other way to interface with the device (i.e. wirelessly). The final negative is that the product's recording time is significantly limited by its battery constraints.

Figure 7.1

Our product packaging is similar to the Spytronix device in that it will include a small external LCD to view the current recording status, and also in our minimal use of push buttons (only 2). Due to a number of large, on-board wireless modules (Wi-Fi, Bluetooth, and GSM) our packaging needs to be a little larger, and also the use of AC power is desired. Taking into account these two factors it has been decided to conceal the device in a custom lamp, due to its

ability to mask the device's true functionality. This is an improvement over the Spytronix recorder because it allows for long, uninterrupted surveillance and effectively hides the product's true intention.

7.3 MSCSpytek UHF transmitter-receiver

The MSCSpytek UHF transmitter-receiver, as seen in Figure 2, is a two component wireless surveillance system. The compact transmitter and receiver are designed with modest external interfaces. This is indicative of the product's simplified operating mode. The transmitter features an antenna for sound transmission, and an on/off switch. The receiver features an antenna, on/off switch, and a jack for listening to the received audio. The transmitter is powered by a lithium ion battery, which can sustain up to five days of continuous use, and the receiver employs a PP3 battery for up to 4 days of continuous use [30].

Figure 7.2

The positives of this product's packaging design are that both of these components are relatively compact, and the simplified designs allow for ease of operation. However, much like the Spytronix device, the transmitter does little to conceal its actual function if found. The battery life of this device is much improved over the previously compared product, but it still remains a limiting factor. Also another negative is that there is no digital storage, therefore this must either be setup external to the receiver or the user must be listening at all times.

The most important feature the DigiSpi takes from this device is its wireless capability. The function of the on-board GSM module, which allows for live listening, is very similar to the UHF system and it in fact has an even greater range. The DigiSpi will also feature antennas, but they will be internal in order to maintain the product's ability to hide its true function.

7.4 Project Packaging Specifications

When considering the packaging options for the DigiSpi there is one overlying factor that guided the ultimate decision: the desire to use the AC power source of a wall outlet. Considering this, the best option for packaging the product is to hide it in the base of a wooden or ceramic lamp. Lamps are common household items found in nearly every room and would serve as a good base station for the DigiSpi hardware.

When designing the specifications for the lamp, the only area of interest is the base which will house the PCB, LCD, and internal antennas for the Wi-Fi and GSM modules. The base will be 8 in. x 10 in. x 12 in. The 8 in. x 10 in. base is adequate size when taking into account the initial PCB size estimate of 6 in. x 6 in., and the 12 in height of the base also gives plenty of room for housing the antennas which must be enclosed in order to mask the device's functionality. This will be on the larger end of typical household lamps, so some innovative packaging design will needed to make the final product seem decorative. The initial dimensions for the lamp shade are 8 in. x 10 in. x 12 in.; the height of the shade is arbitrary and the other two dimensions were chosen to match the base, however this is not a requirement.

The lamp's base will be constructed using wood casing, and the shade will be purchased. A hollow metal cylinder will protrude from the base upwards into the shade and at the end will be a Compact Fluorescent Bulb. Taking into account the PCB, the major on-board components, and the lamp apparatus, the estimated weight is 8 lbs, and the cost is $522.81.

7.5 PCB Footprint Layout

When considering the DigiSpi's PCB dimensions, the first thing the designer must take into account is the three large onboard modules (Wi-Fi, Bluetooth, and GSM). The size of these modules dictated the initial estimate of the board size to be 6 in. x 6 in. The dsPIC33FJ256GP710 microcontroller comes in a 12 mm x 12 mm x 1 mm,100-pin TQFP package; there are two packages offered for this model, the other being a 14 mm x 14 mm x 1 mm 100-pin TQFP, the later was chosen due to its smaller size [6]. This is roughly centered on the board so that it may easily interface with all the various peripherals. The three main modules are located adjacent to each other below the microcontroller. The WiPort, GSM module, and Bluetooth module have very similar footprints as each of these larger components interface with the board through a smaller surface mounted, high pitch connector. Each module has a manufacturer defined connector that it uses [1] [2] [4]. These small connectors have no influence on the PCB; it is instead the footprint of the module itself which influences board dimensions.

Most of the DigiSpi's other large parts are not onboard (i.e. LCD, SD card), and therefore do not have a large influence on the size of the board. The rest of the board is taken up primarily by smaller passive components, regulators, and IC's, and while they individually do not have a large effect on board size, they should be factored in collectively when making a PCB size estimate.

7.6 Summary

Taking into account the desire to use AC power, the PCB size, and the need to mask the device's true function, the best packaging design for the DigiSpi is to hide it in a common household item that is large enough to hide all internal components and antennas. The lamp is an ideal choice for this because it can be designed in a number of ways that can house the DigiSpi hardware. While this is quite different than other products on the market, the DigiSpi still has some similarities in packaging design. However, its increased functionality requires a larger PCB, and thus it cannot be packaged as compactly as other products.

8.0 Schematic Design Considerations

This useful embedded device consists of a number of different components; therefore several design considerations must be examined. The primary considerations considered are that the power circuitry must be able to supply the 3 voltages required (3.3V, 3.8V, and 5V) and the microcontroller must be interfaced using a variety of methods with the other components. Although not all modules were integrated into the final design, the design issues related to each module have been left in this section to more clearly show the design process.

8.1 Theory of Operation

8.1.1 Power Supply

The DigiSpi will be conveniently concealed in a lamp. This will allow the device to draw power from a conventional wall outlet. A 7V DC wallwart will be used. Three low drop out linear regulators will be used to produce the 3.3V, 3.8V and 5V voltages required for our components [7]. Due to impulse currents from wireless transmission, the power supply must be able to source significant current. For example, the data sheet for the GSM module mentions "relative current peaks as high as 2A". [6] The two LT1528 linear regulators used to supply the 3.3V and 3.8V rails are rated as being able to source up to 3A of current allowing us a sufficient margin above our maximum current requirements (2A for the 3.8V rail and 1A for the 3.3V rail). The LT1121 used for the 5V rail is rated as being able to source 150mA [23]. This is sufficient to power the LCD which requires only 120mA [12]. The Microcontroller and Wi-Fi Server will be powered by the 3.3V rail. The GSM module and Bluetooth module will be powered by the 3.8V rail. The LCD will be powered off of the 5V rail

8.1.2 Microcontroller

The microcontroller that will be used is the dsPIC33FJ256GP710 [6]. It will be powered from the 3.3V supply rail and will operate at a core frequency of 120 MHz and an instruction frequency of 33 MHz. The choice of operating frequency is made such that the micro is able to run multiple timer modules and have ample time to perform all of its primary functions, including sampling and analyzing audio. The processing frequency is I/O bound based upon the rate of writing to the SD card. In the final project, this needs to be as fast as possible. This means a timing to write 512 blocks of data is about 10 milliseconds. The ATD sampling has a frequency of around 60 milliseconds for 512 samples. Therefore, the operating frequency is at

least five times faster than it needs to be. This is warranted in this design in case other functionality needs to be implemented (aka "version 2"). Since the device will be plugged into the wall, lowering the operating frequency to conserve power is not necessary. The microcontroller will be responsible for accepting audio input, push button inputs, writing to and reading from the SD card, interfacing with the Wi-Fi and GSM modules, outputting audio to the headphone jack, as well as performing filtering operations on the sampled audio.

8.1.3 LCD

The LCD will provide users with recording information such as the current up-time and the amount of speech recorded so far. This feature is especially useful for users who want a quick way to confirm that the device is functioning. A user could start speaking and observe the change in the amount of speech recorded to confirm that the device is functioning. It is also a useful way to tell if the SD card needs to be replaced in order to continue recording. The LCD used in this application will be the Xiamen Ocular GDM1602K [12]. It will communicate with the microcontroller via a parallel connection. According to the LCD data sheet, it can run off of a supply voltage of 5V. The LCD will be powered by the 5V rail.

8.1.4 WiPort Wi-Fi Server

The WiPort will provide wireless access for users to connect to. This is an important function since it will allow users within range to connect and change recording settings, such as threshold levels and monitor recording status. It requires a 3.135V to 3.45V supply [1], so it will be supplied by the 3.3V rail. It will communicate with the Microcontroller via an SPI connection that will be shared with the GSM module.

8.1.5 GSM Module

The GSM module will provide a means of world wide call-in access to hear what is happening live. The GSM module requires a 3.4-4.2V supply [2]. Since 3.3V will not be sufficient, it will be powered from the 3.8V rail. The peak impulse current for this module can be as high as 2A.

8.1.6 Bluetooth module

The Bluetooth module allows for better recording and more discreet placement of the device since it will enable the device to connect with a small, Bluetooth microphone situated anywhere in the room. The Bluetooth module requires a 3.6V to 7V supply, so it will be

supplied by the 3.8V rail [4]. It will communicate with the microprocessor via a serial interface. The audio from the Bluetooth microphone will be sent to a PCM codec [24] which will output an analog signal that can then be fed into an A-to-D input on the microcontroller.

8.2 Hardware Design Narrative

The microcontroller accomplishes many important tasks. First, it will receive input audio and process that audio to determine if speech is present. Second, it will interface with the SD card to save and read back audio. Third, it coordinates the operations of the rest of the components. Fourth, the microcontroller will be interfacing with the LCD and push buttons to display and change the current settings. Finally, the microcontroller will be interfacing with the GSM module to turn it on and monitor its status.

Components will be interfaced via SPI, parallel, serial, as well as some discrete inputs/outputs. The modules of the microcontroller that will be used are the Timing module, PWM, SPI, Serial (UART) and the Analog to Digital converter.

8.2.1 SPI

The SPI subsystem of the microcontroller will be used to communicate with the GSM module and SD card. The microcontroller will communicate in slave mode with the GSM module as the master on SPI port 2 on the following pins: SS2, SCK2, SDI2, and SDO2. The microcontroller will be the master for communication with the SD card on SPI port 1, utilizing port pins SCK1, SDI1, SDO1, and SS1.

8.2.2 Serial (UART)

The Serial (UART) subsystem will be used to communicate with the Bluetooth module and Wi-Fi module. The Bluetooth module will use port pins U1RX and U1TX with a baud rate of 115200. The Wi-Fi module will use U2RX and U2TX and communicate using a baud rate of 115200.

8.2.3 PWM

The PWM subsystem will be used to for the headphone output. The first of the available PWM channels will be used on port pin OC1. After being output from the PWM, the signal will pass through an analog low pass filter before being output to the headphone jack.

8.2.4 Analog-to-Digital Converter

The Analog-to-Digital Converter will be used to sample the wired microphone. This will occur on one of the available analog input pins, such as AN0. A sampling rate of 8 kHz will be used. The dsPIC can provide samples with 10 or 12 bit resolution. In this case 10 bit resolution is used and then those samples are shifted down to 8 bits. After processing, 8 bit samples will be stored to the SD card.

8.2.5 Timing module

The timing module will be used to generate interrupts to check push buttons and to keep track of system up-time. The system up-time information will be used for outputting to the display.

8.2.6 Summary

The primary challenges that this design presents are the high maximum current ratings for the wireless communications modules and the variety of interfaces. A very durable and high current wallwart along with regulators that have high current ratings will be necessary to accommodate the short, high-power bursts. With the device adhering to the outlined hardware design constraints, a durable and discreet embedded device is the result.

9.0 PCB Layout Design Considerations

A functional PCB layout will be critical to accommodate the plethora of necessary components in a discrete form-factor. Mechanical concerns for board-to-board connections; noise management to maximize audio signal integrity; and stable power distribution throughout the PCB will all be carefully analyzed.

9.1 PCB Layout Design Considerations – Overall

To enable the extreme flexibility of the DigiSpi surveillance device, many components will have to be seamlessly integrated. Wise component layout will be important to manage EMI.

The following strategy will be used in part placement: digital, analog, and mixed signal devices will be separated to reduce noise and interference [36]. If possible, high frequency parts will be segregated from devices that operate at a clock speed lower than 40 MHz [36]. In the DigiSpi PCB layout, the power supply electronics will be isolated in a separate quadrant of the PCB. The dsPIC and GM862 GSM Cellular modem will both be handling analog audio, so they will be placed near each other in the mixed signal section of the board. The dsPIC will also be located on the analog power and ground planes since SAR (successive approximation register) A/D conversion will be utilized [36]. The Ezurio Bluetooth module and WiPort Wi-Fi server will only be handling digital signals, so they will be grouped in a digital section of the PCB.

Device placement is also constrained by factors other than noise reduction. The SD Memory Card slot will need to be positioned on the edge of the board for access to the spring-loaded slot. The Bluetooth module has an integrated ceramic antenna. To maximize signal propagation, it must be placed parallel to the plane of operation [4]. Finally, headers for the external LCD and pushbuttons will be placed in order to minimize tricky cabling within the packaging.

Trace size, length, and routing considerations must be considered in conjunction with part placement and layout. As a general rule of thumb, traces should be as short and direct as possible to reduce undesirable coupling [36]. Also, digital parts will be placed closest to board-to-board and other connectors to prevent interference and signal degradation. To further suppress the noise, special care will be taken to insure that high frequency signals (the audio and clock traces) will not be placed close to high impedance input A/D input pins on the dsPIC and GM862 GSM cellular modem. As per the course recommendation, signal traces and spacing will

be 12 mils, but they will be tapered down to 8 mils when approaching the fine pitch pines of the dsPIC TQFP. Power traces will be larger, a minimum of 70 mils to handle up to 3 amps of instantaneous current [37]. The trace widths are calculated using the PCB manufacturer's online calculator. First, the trace area is calculated (192.9 square mils for 3 amps using 2 oz. copper and a 10 degree temperature rise), and then the associated trace width (70 mils) is computed. Power traces will also be routed in a hub-and-spoke configuration to avoid inductive loops [38].

To reduce power supply noise, bypass capacitors will be placed at the power supply and as close as possible to all active devices [36]. Care will be taken to reduce the lead length on the bypass capacitors located at the input voltage pins of peripheral components. The Ezurio Bluetooth module and WiPort Wi-Fi server have internal voltage regulation, so bypass capacitors would not be necessary for these devices. However, bypass capacitors will be utilized in this design to maintain a stable power rail [4], [1]. Specific bypass capacitor requirements for the power supply and GM862 GSM cellular modules are discussed in the power supply requirements section of this document.

Secure and stable physical mounting of the three wireless modules will be important. The Ezurio Bluetooth module is contained on a separate PCB that will mate with the core logic board through a 40-pin connector [4]. Drill-holes will be necessary for plastic stand-offs that secure the Bluetooth module. The GM862 GSM Cellular module is encased in a metallic shield. Pins protruding from this case will secure the device to the PCB [2]. Again, drill-holes matching the module's footprint are required. The WiPort, also contained in a metallic shield, will mate with the PCB like the GM862 [1].

9.2 PCB Layout Design Considerations – Microcontroller

The extended flexibility of the dsPIC family of microcontrollers allows for simplified board layout and design. Utilization of the onboard oscillator eliminates costly external circuitry as well as the complications of these supporting components [6].

Attention must still be paid to some basic design rules, though. First, placement of bypass capacitors close to V_{dd} and V_{ss} pins will insure a stable supply voltage for the processor. Next, the microcontroller will be oriented to minimize signal traces to the WiPort, GM862 GSM Module, and Ezurio Bluetooth board. Finally, attention must be paid to routing noisy signals away from the high impedance ADC inputs to avoid coupling [36].

9.3 PCB Layout Design Considerations - Power Supply

Due to the impulsive power requirements from transmitting wireless devices, an efficient PCB design will needed to maintain stable voltage rails and minimize noise.

The design of the power supply's traces is very important. Intuitively, trace widths must be the proper width to handle the necessary current. Trace widths for the DigiSpi design have been specified to be a minimum to 70 mils to handle 3 amps of current. As an added safety factor and per the suggestion of the course staff, the traces will be enlarged to 100 mils [37]. The traces to high power components like the GM862 and WiPort must also be kept as short as possible. Although the voltage drop and power loss across a long trace might be acceptable for a device that is not battery powered, the induced noise is not tolerable. The current impulses from the GM862 occur at a frequency of 216 Hz [2]. Care must be taken to insure that noise from these impulsive current spikes does not affect the rest of the embedded system.

To minimize noise, a common ground plane will be utilized. This introduces some complexity in the layout since more signals will have to be routed on a single layer, but will provide better noise tolerance for our mixed signal application. Care in ground plane routing, though, must be taken to insure that high current return paths do not cross noise sensitive lines like the analog audio signal lines near the microcontroller and GM862 GSM module [36].

As previously mentioned, the power supply system will utilize bypass capacitors placed close to all active components. Specifically, the GM862 requires a low ESR bypass capacitor located very close to the part to reduce the current absorption peaks—a 100 uF tantalum capacitor is recommended [2]. The PCB layout strategy will prioritize the close placement of all these bypass capacitors.

9.4 Summary

To accommodate the DigiSpi's diverse set of components, a multitude of PCB layout strategies will be used to design a functional and efficient core logic board. Digital and analog component separation; intelligent trace routing; bypass capacitor placement; and component mounting techniques will be focused on. EMI will also be reduced through the use of a copper ground plane. The combination of these steps should result in a reliable device with manageable noise levels.

10.0 Software Design Considerations

Figure 10.1 shows the overall block diagram for the dsPIC30F. The dsPIC33F used in our final PCB is very similar, with the major difference being a larger Data SRAM, 64KB in our case. From this diagram and the dsPIC30F family overview document, it is evident that the Data SRAM and the Flash Program Memory are separate entities, allowing an instruction frequency of 33.3MHz and a timer interrupt level of 66.6MHz [31], [32], [34]. The different high frequencies enable the software to function without an external oscillator, and to keep a fast processing and interrupt rate.

10.1 Code and Memory Setup

The Program Memory is Flash programmable and contains sixteen 16-bit registers [31]. The most important is the Data SRAM setup due to the importance of buffering data in this project. The direct reason for the choice of a dsPIC device was to facilitate DSP instructions if they would be necessary. In our project they were not.

For the project, a C compiler develops the assembly code, and many specifics are left to the compiler through macro definitions in the code placed after a variable definition (the keyword *__attribute_*) [33]. This will be discussed following the figure.

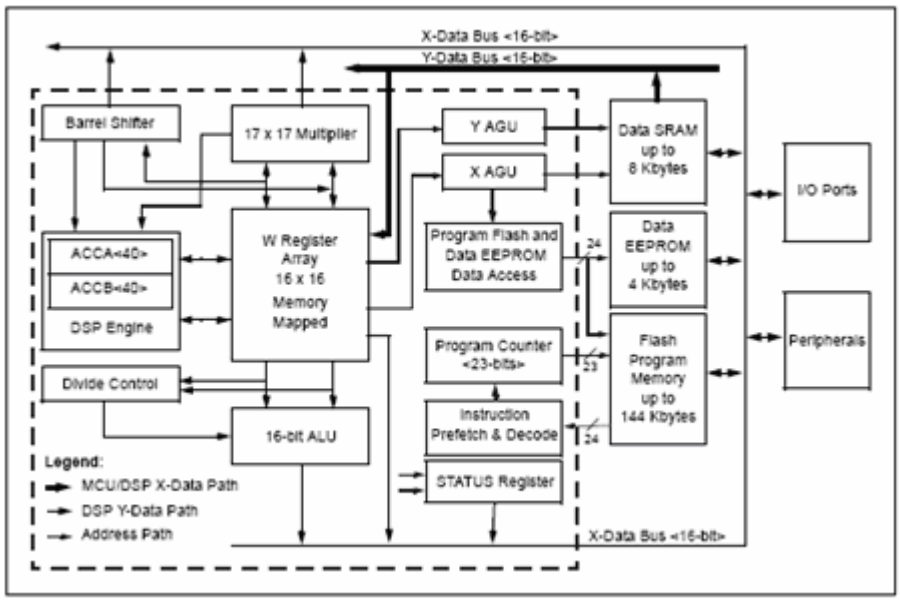


Figure 10.1 Block Diagram of Microcontroller [31]

With respect to the memory, many functions were inline to reduce the need for a stack. This way the maximum amount of memory is available for buffering and software complexity is greatly reduced. The heap is not used; instead a general data memory is used for storing flags,

temporary values and buffers, loop control variables, and pointers. Again, these will be global, reducing the need for specific management in the software.

The complexity of the memory mapping is further reduced using the C Compiler. From Microchip this is the ***MPLAB C30 Compiler for dsPIC30F and dsPIC33F Families***. The following table shows where each code and data section is in memory and the preprocessor definitions used for compiler recognition [33]. The far means an addressable memory further from the base of the pointer, while near (.ndata) is the default setup for all values. Therefore, only the sample buffer needs a special setup.

Table 10.1 Memory Setup

Item	Location	Compiler Call
Executable Code	.text	Setup by the C30 compiler
Global Variables	.ndata	*_attribute_(space(ndata))*
Sample Buffer	.far	*_attribute_(space(far))*
Stack	.Stack	Setup by the C30 compiler

10.2 Code Organization

The code is ultimately interrupt driven, with the main signal processing algorithm embedded into the main program loop. The design is partially polling as the main program loop waits for a signal from the interrupt service routine that a set can be processed. Therefore, this is a flag driven, or a "hybrid" organization of the software. For maximum organization and scale-ability, the main program is designed as a state machine seen in ***Appendix F***. In the appendix section is also the code listing, i.e. the tree. Several files were used in addition to main for setup, LCD I/O, and SD I/O. These contained the low level routines necessary and presented the main program with a high level interface to achieve the desired result.

To aid in the design, timers are used to achieve functionality. Timer 1 is used to setup the heartbeat debug that pulses an LED every 0.5 second. This insures the board is working. Timer 2 is used by the PWM, allowing it a 10 bit sensitivity and 8 kHz frequency. Timer 3 drives the ATD and its time out triggers a sample every 0.125 millisecond. Timer 4 is used to poll the pushbuttons and this is done every **2** milliseconds.

For interrupts, Timer 2 and 3 are used to control the ATD sample and PWM frequency, so only Timer 1 and Timer 4 have user defined interrupts. Timer 1 toggles the LED on port RD1, and timer 4 looks for a button to toggle from low to high and back to low meaning it has been pushed. When the red pushbutton, on port RD2, is pushed it sets a *src_flag*, and the black

pushbutton, on port RD1, sets a *reset_flag*. These are used to reset the device or toggle its mode. In addition, the ATD sample conversion to an unsigned integer value begins once a sample is obtained. The completion of the conversion causes an interrupt. Within this ATD interrupt is the buffering of the sample and/or output of a respective data depending on the operation setup (see more in subsequent paragraphs). When 512 bytes of data are read or written from the buffer, a *sample_flag* is set to signal for more data or there is data that could potentially written to the SD card. Block diagrams for each of these interrupt service routines are seen in ***Appendix F.***

10.3 Processing Summary

The software has three different states. The first is a reset mode and is entered whenever a state is finished and the *reset_flag* is set. The reset mode sets up the registers, initializes the SD cards, sets the timing of the registers and timers, and sets up the buffers. If the source is the ATD (i.e. data is being recorded), the buffer is left alone, but if data is being read from the SD card, the buffer is filled before continuing. It then enables appropriate interrupts before clearing the *reset_flag* and setting the current state to a run state. The run state and the update state are the two operation modes. During the run state, the main program loop is polling the *sample_flag*. When that flag is set and the mode is setup to read from the ATD port and write data, an algorithm is run to determine if speech is present, and if it is, the data is written to the SD card. The various pointers and interrelation is seen in the diagram after this paragraph and on the left. More of the processing is explained through flowcharts in ***Appendix F***. Currently the number of buffers is six, and the data has the potential to be written every 64 millisecond (512 bytes * 0.125 milliseconds per each new one). Therefore, the SPI operates for SD card I/O as fast as possible, allowing an I/O time of around 10 milliseconds.

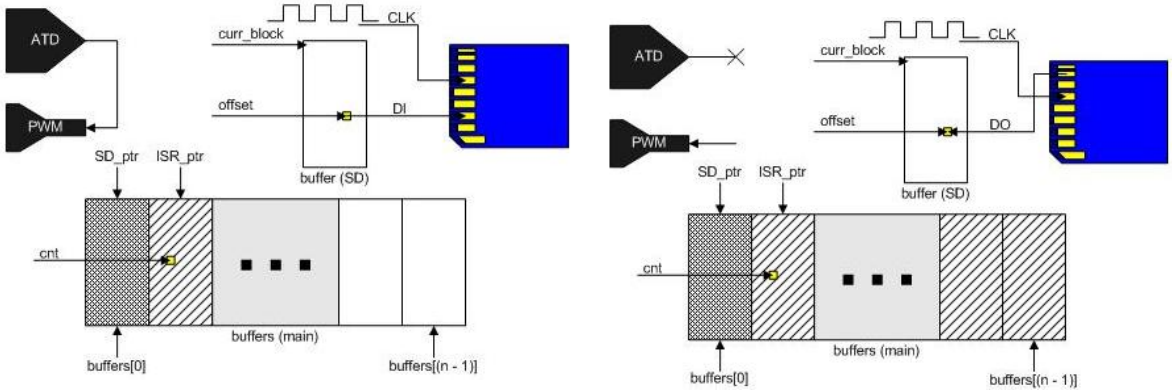


Figure 10.2 Interrelation of Software and SD card

If the *sample_flag* is set and the mode is setup to read from the SD card, data is read from the SD card into the buffer with the same timing constraints as before. This diagram is seen above this paragraph and is the illustration on the right.

The final mode is an update mode. In this mode the ATD and timer 1 interrupts are disabled if the *src_flag* is set and the mode is changed to read from the SD card if it is buffering from the ATD or visa versa. If the SD card will be read, the buffers are first filled with all the data they can hold before continue. Thereafter, ATD and timer 1 interrupts are re-enabled, and the state returns to run with the new mode.

In order to determine if the block contains voice activity, the algorithm computes two values for a given block of audio and compares those values against thresholds. The two values are number of zero-crossings and average power. The number of zero-crossings is calculated by multiplying each value by its previous value and adding up the number of those results that are below zero (because consecutive positives or consecutive negatives both result in positive values when multiplied.) The average power of the signal is calculated by summing up the square of each value. The threshold for average is set during reset. The process consists of measuring the average power for a given period of time and then using a constant offset to scale the threshold slightly above the recorded ambient noise level.

10.4 Summary

The speech processing algorithm, state machine design, timer and interrupt driven, and the memory layout of the software creates a tractable and scaleable design. This successfully fulfills the necessary timing criteria observed herein, and enables a compact design for efficiency.

11.0 Version 2 Changes

After spending the semester learning about and designing our project there are a number of changes that would be made to version 2. This would include changes to our schematic, custom footprints, layout, component selection and software implementation.

First, a few changes would be made to the schematic to improve audio quality. We would purchase a more sensitive microphone and look into possibly doing some sort of automatic gain control. Also, a volume control would be added for the line level output, even though this is designed primarily for debugging. Changes would also be made to the GSM microphone input circuitry. The signal for the GSM microphone input could be taken after the input LPF, rather than from the PWM output. This would likely improve audio quality. The PCB as a whole should also be better shielded from the GSM antenna to prevent interference.

Second, there were a few layout and footprint errors that we would change to make construction easier. The two audio jack connectors were located near but not quite on the edge of the board. In a second revision these connectors should be moved so that they sit flush on the edge of the board to make connecting microphones and headphones easier and removing any possibility of snapping a connector off the board while trying to plug in a cable. The through holes on the footprint for our audio connectors were a bit tight. We were able to still mount the connectors, however a little more room would make the fit much better. Two of our footprints (PCM Codec and Bluetooth module) had the pin-out reversed leading to some bending of pins and fly wiring. These footprints also would be changed.

Knowing now what we have decided to go with in terms of a speech detection algorithm, our microprocessor choice is really overkill for this type of design. A cheaper micro with less memory and fewer pins could be utilized for this project.

Finally, given additional time, support for the FAT file system would be included in the micro so that the files could be read from any SD card reader instead of using the Digi-Spi SD card companion device.

These are the most obvious and useful changes to make. However, that does not mean that they are the only ones. A likely inclusion in a second generation would be a custom-built Bluetooth microphone to be included with the Digi-Spi device. Unfortunately, most available Bluetooth microphones are designed with less sensitive microphones. This application requires a more sensitive microphone to pick up conversations within a reasonable area of the microphone.

12.0 Summary and Conclusions

This course certainly utilizes a variety of skills. As a group, we managed to design, create and package a consumer oriented device. With this type of a project, a number of different aspects must be considered from software and hardware to safety and reliability. This course, above all, gave us a better appreciation for what all goes into the development of a product.

Designing and creating a working PCB layout was a major accomplishment. As a group, our previous experience with this aspect of the design process was very limited. Needless to say we learned a lot about taking a schematic and turning it into a usable printed circuit board.

We learned how to work with new IDE and interfaced with a variety of different peripherals using UART, Parallel, GPIO, and SPI to communicate. We also gained a greater appreciation for the engineering design process. Throughout the semester our team improved on our project management and team work skills. Hardware skills such as creating a schematic and board layout were also acquired. Although we did not implement the FAT file system, a fair amount was learned about low level memory access on the SD card and how things would likely work if we did implement a file system.

13.0 References

[1] Lantronix, "WiPort Data Sheet," [Online Document], unknown publication date, [accessed January 31, 2007], http://www.lantronix.com/pdf/WiPort_DS.pdf.

[2] Telit, "GM862-QUAD-PY Hardware User Guide," [Online Document], November, 2006, [accessed January 31, 2007], http://www.telit.co.it/data/uploads_EN/products/1vv0300692_GM862-QUAD_Hardware_User_Guide_r4.pdf.

[3] SanDisk, "SD Card Physical Layer Specification," [Online Document], May, 2001, http://www.sandisk.com/Assets/File/OEM/Manuals/SD_Physical_specsv101.pdf.

[4] Ezurio, "BISM2 Bluetooth Version 2.0 Serial Module," [Online Document], unknown publication date, [accessed January 31, 2007], http://www.ezurio.com/dl/?id=88.

[5] Hitachi, "HD44780U (LCD-II) (Dot Matrix Liquid Crystal Display Controller/Driver)," [Online Document], 1999, [accessed January 23, 2007] http://www.sparkfun.com/datasheets/LCD/HD44780.pdf.

[6] Microchip, "dsPIC33F Family Data Sheet," [Online Document], 2007, [accessed Jan 23, 2007] http://ww1.microchip.com/downloads/en/DeviceDoc/70165E.pdf.

[7] Linear Technology, "LT1528 - 3A Low Dropout Regulator for Microprocessor Applications," [Online Document], 1995, [accessed Jan 23, 2007] http://www.linear.com/pc/downloadDocument.do;jsessionid=F59qCKyrESmNZ0TWONP6ijwiS76IRLOd9IO2RcOp75XgkbDKyYm7!108404241?navId=H0,C1,C1003,C1040,C1055,P1048,D1350.

[8] Freescale, "MC9S12E128 Data Sheet," [Online Document], October, 2005, [accessed Jan 23, 2007] http://www.freescale.com/files/microcontrollers/doc/data_sheet/MC9S12E128V1.pdf.

[9] Mitsumi, "Bluetooth Module WML-C40 Class 1," [Online Document], unknown publication date, [accessed January 23, 2007], http://www.sparkfun.com/datasheets/Wireless/Bluetooth/Bluetooth-SMD-Module.pdf.

[10] N. Bedwell, "Homework 3: Design Constraint Analysis and Component Selection Rationale," [Online Document], September, 2006, [accessed January 23, 2007], http://cobweb.ecn.purdue.edu/~dsml/ece477/Webs/F06-Grp02/files/hw3.doc.

[11] DPAC Technologies, "Airborne Embedded Wireless Device Server Module", [Online Document], August, 2006, [accessed Jan 23, 2007] http://www.dpactech.com/docs/wireless_products/g_ab_wireless_device_server_module.pdf.

[12] XIAMEN OCULAR, "GDM1602K - Basic 16x2 Character LCD Datasheet," [Online Document], unknown publication date, [accessed January 31, 2007], http://www.sparkfun.com/datasheets/LCD/GDM1602K.pdf

[13] Spytronix.com, "33hr USB Digital Voice and Phone Recorder," [Online Document], 2007 Jan 27, [accessed January 27, 2007], http://www.spy-tronix.com/33hourphonerecorder.html

[14] COBY Online, "CXR55 Voice Activated Cassette Recorder," [Online Document], unknown publication date, [accessed April 7, 2007] http://www.cobyusa.com/_en/prod_item.php?item=CXR55&pcat=portaudio&pscat=recorder&pscat2

[15] SoftRISC, Inc, "VoIP and Multimedia Software," [Online Document], unknown publication date, [accessed April 7, 2007] http://www.softrisc.com/products/vad.html

[16] Keenzo, "Cisco SPA901," [Online Document], unknown publication date, [accessed April 7, 2007] http://www.keenzo.com/showproduct.asp?ID=800524&ref=FRG0

[17] Anderson, David V., "Speech activity detector for use in noise reduction system," US Patent 6 453 285, September 17th, 2002,

[18] Vergin; Julien Rivarol, "Speech detection system and method," US Patent 6 757 651, June 29, 2004

[19] Reaves; Benjamin Kerr, "Speech detection device," US Patent 5 826 230, October 20, 1998

[20] Blenko, Walter J., "The Doctrine of Equivalents in Patent Infringement," *JOM*, vol. 42 (5) p. 59 1990 http://www.tms.org/pubs/journals/JOM/matters/matters-9005.html

[21] Department of Defense, "Military Handbook, Reliability Prediction of Electronic Equipment," [Online Document], 1991, Available: http://cobweb.ecn.purdue.edu/~dsml/ece477/Homework/Fall2006/Mil-Hdbk-217F.pdf

[22] Maxim, **"**Tiny, Low-Cost, Single/Dual-Input, Fixed-Gain Microphone Amplifiers with Integrated Bias," [Online Document], 2003, Available: http://datasheets.maxim-ic.com/en/ds/MAX9812-MAX9813L.pdf

[23] Linear Technology, "LT1121 Micropower Low Dropout Regulators with Shutdown," [Online Document], 1994, [accessed April 9, 2007] http://www.linear.com/pc/downloadDocument.do?navId=H0,C3,P1370,D2188

[24] Freescale Semiconductor, "MC145481 3V PCM Codec-Filter," [Online Document], 1998, [accessed April 9, 2007] http://www.freescale.com/files/timing_interconnect_access/doc/data_sheet/MC145481.pdf

[25] R.B.K. Dewar and R. Chapman, "Building secure software: Your language matters!," Military Embedded Systems, Winter, pp. 30-33, 2006. http://www.mil-embedded.com/PDFs/AdaCore.Win06.pdf

[26] David Benjamin, "Is EMF radiation the cigarette of the 21st century?," EE Times Online, [Online Document], 13 February 2007, [accessed February 21, 2007], http://www.eetimes.com/showArticle.jhtml?articleID=197005857&printable=true

[27] intellectuk.org, "'World First' makes printed circuit board production more sustainable," [Online Document], unknown publication date, [accessed February 21, 2007], http://www.intellectuk.org/markets/pcb_production.asp

[28] "Printed Circuit Board Recycling: Overview," [Online Document], unknown publication date, [accessed February 21, 2007], http://p2library.nfesc.navy.mil/P2_Opportunity_Handbook/2_II_8.html

[29] Techmonitor.net, "Waste Management : Jan-Feb 2006; Electronic Wastes; Recycling printed circuit boards," [Online Document], 2006, [accessed February 21, 2007], http://www.techmonitor.net/techmon/06jan_feb/was/was_electronic.htm

[30] MSCSpytek, "UHF Crystal Controller Transmitters and Receivers," [Online Document], Jan 27 2007, [accessed Jan 27 2006], http://www.mscspytek.com/uhf_spy ing_equipment.htm

[31] Microchip.com, "dsPIC30F3014/4013 Data Sheet," [Online Document] , unknown publication date, [accessed February 07, 2007], http://ww1.microchip.com/downloads/en/DeviceDoc/70138E.pdf .

[32] Microchip.com, "dsPIC30F Family Overview," [Online Document], unknown publication date, [accessed February 07, 2007], http://ww1.microchip.com/downloads/en/DeviceDoc/70043F.pdf .

[33] Microchip.com, "MPLAB C30 C Compiler User's Guide," [Online Document], unknown publication date, [accessed February 07, 2007], http://ww1.microchip.com/downloads/en/DeviceDoc/C30_Users_Guide_51284e.pdf .

[34] Microchip.com, "Section 7. Oscillator," [Online Document] , unknown publication date, [accessed February 07, 2007], http://ww1.microchip.com/downloads/en/DeviceDoc/70054d.pdf .

[35] Wikipedia.org, "Voice Frequency," [Online Document], unknown publication date, [accessed January 23, 2007], http://en.wikipedia.org/wiki/Voice_frequency .

[36] Microchip, "Layout Tips for 12-Bit A/D Converter Application", [Online Document], 1999, [accessed February 2, 2007], http://ww1.microchip.com/downloads/en/AppNotes/00688b.pdf

[37] D.G. Meyer, "Module X: PCB Fabrication Process and Layout Basics", [Online Document], 2006, [accessed January 23, 2007], http://cobweb.ecn.purdue.edu/~dsml/ece477/Notes/PDF/5-PCB_DESIGN.pdf

[38] Motorola, "Motorola Semiconductor Application Note AN1259", [Online Document], 2005 September, [accessed January 23, 2007], http://cobweb.ecn.purdue.edu/~dsml/ece477/Homework/Fall2006/AN1259.pdf

Appendix A: Individual Contributions

A.1 Contributions of Brad Sokola:

I served as the leader for team Digi-Spi. As part of this I was responsible for communication among the group, reviewing and submitting the initial team homework assignments, organizing parts acquisition, having the TCSP power points edited and ready to go and generally keeping everyone up to date about the project deliverables and due dates. This project gave us the chance to experience so many different aspects of the design process. This made the leadership role all the more challenging. Of primary concern with a project of this magnitude was organization. There were numerous deliverables throughout the semester and the timeliness and quality of these deliverables greatly impacts the final product.

Warren and I purchased and sampled the bulk of the components that went into the final project. I kept an updated bill of materials list generated from our final schematic and kept track of the status of all of our parts.

Initially, I was responsible for creating the detailed block diagram including voltage requirements and pin usage.

In addition to the organizational contributions, I was responsible for creating a detailed block diagram including voltage requirements and pin usage. This naturally led to me being the point person for the first schematic and later revisions of the schematic. I created footprints in OrCad Layout for some of the custom parts we used. Once the PCB was returned, I was involved in soldering a few of the easier packages including some electrolytic capacitors as well as a variety of 1206 packages. The majority of my soldering however was with headers and connectors for the LCD and SD card.

In terms of the software, I adapted code for a generic LCD to be used with our specific display and created function for the various screen updates including the up-time and peripheral status. I also was involved in the general debugging and testing of the code. I was also responsible for getting familiar with the GSM module and writing the Python code associated with it. Finally, I was involved in organizing the final documentation, specifically editing and stitching together the final report.

A.2 Contributions of Karl Herb:

I am the only computer engineer on the team, so I naturally assumed the role of software and website maintenance. Throughout the course of the project I setup the website, added new descriptions, and on conclusion of the project, wrote the descriptions, and compiled the pictures for it.

My initial tasking consisted of filling in where was necessary. In addition to brainstorming and proofreading, I began to design the power supply. I selected parts, ordered some samples, and attempted to draw a schematic for the power design. My design was not used in the final project, but a power plug jack I sampled was mounted on the PCB.

After I wrote the software constraints homework and the ethical and environmental analysis homework, I began initial software development on the dsPIC30F. I learned how the dsPIC's ATD, PWM, timers, and other modules worked and used this to make a proof of concept of the necessary components we would need in the design.

When the concept was done, I began migrating code over to the dsPIC33F. I solved problems using Microchip's help with the setup of MPLAB (development software) and how to initialize the ATD in the 33F. I expanded the code to allow more timers, run at the fastest rate possible, and to be ready for an SD card interface.

I began testing the SD card interface on the 30F board, and after several weeks of testing and debugging, I was reading and writing correctly from the SD card. I then migrated this over to the 33F. This was difficult due to the SPI setup, but once the correct ordering of the code was known it was working. I then cleaned the SD code and I designed an initial buffering scheme for ATD samples. It used a combination of a state machine in the main function, several interrupt service routines, and flags that when set low or high directed the functionality of the main code. When working it was further developed to toggle between the ATD sourcing the PWM and buffering data into the SD card and the SD card outputting data to be sourced on the PWM (thus ignoring the ATD). When this was working I wrote in where Brad needed to call various LCD functions and where Justin needed to do signal processing. I handed the code off to them and they put the finishes touches on it and removed several bugs.

The 30F experimentation with the SD card led to the development of a Python program and intercommunication protocol over the SCI port between the Python and the microcontroller. This was in the hopes of making an *.au file from the data on the SD card for future playback.

The data was correctly obtained from the SD card, but the *.au file was unable to produce a decent playback. Despite a circuit being built to interface the SD card, SCI port, and the dsPIC30F and a software program compiled to communicate with it; this design was abandoned in the end. I was not able to have it work at a level suitable to present as an integration of the component.

Upon conclusion of the project, I developed the poster, documented all of the software with diagrams, listings, and revisions to the initial homework, and finalized the website for archiving.

A.3 Contributions of Warren Santner:

The successful completion of our project was certainly a team effort, but we all focused our areas of expertise on certain aspects of the design. My two core responsibilities were the PCB board layout and packaging.

For the PCB layout, I rendered a large number of footprints with the help of Brad Sokola and Justin Lanning. The board placement and routing was entirely my responsibility. I completed three revisions of the board following some major schematic changes.

With the assistance of Brad Sokola, I constructed the lamp housing and hidden operator interface within the lamp. This task included lots of woodworking, cutting, staining, and wiring of the components.

Besides these main focus areas, I assisted with various tasks as issues and problems arose. I provided assistance with the initial schematic; did significant research on the Bluetooth integration and PCM codec selection; soldered the majority of our components on the PCB; provided debugging assistance with the SPI initialization and LCD contrast control; and debugged the issues plaguing the power-up control of our GSM module.

A.4 Contributions of Justin Lanning:

Throughout the semester I took on a number of roles, varying from both hardware and software. Aside from the individual homework's that I completed, the first half of the semester was spent mostly assisting with schematic creation and circuit design. When it came time to do the layout, Brad and I assisted Warren by creating and verifying dimensions on the custom

footprints required for our design. Also when the team divided up the peripherals, I became the WiPort expert and I spent time working with this device on a development kit that we had purchased. I was able to setup the wireless web server, and connect to it. I embedded on the server a Java applet which created a serial terminal that could be used for wireless serial communication. Ultimately this was not integrated into our final device as it did not add significant functionality to warrant its inclusion.

Following the receipt of our PCB, I was significantly involved with populating the various components and debugging circuit problems on our board. One of these circuit issues proved to be quite significant as it was hindering our ability to accurately sample audio input. After the population of our board was complete my focus turned to helping with software development.

My software contributions include working with Brad on the LCD code, debugging issues with our input buffering and SD card writing (solved by speeding up SPI clock and restructuring buffer), push button setup, SD initialization routine debugging (caught critical error involving the use of an SD only command on a MMC), SPI-GSM communication/LCD status output, and speech processing algorithm integration. Overall, I am very pleased with my role in this project as I was able to contribute in a variety of roles which gave me a comprehensive understanding of the engineering design process.

Appendix B: Packaging

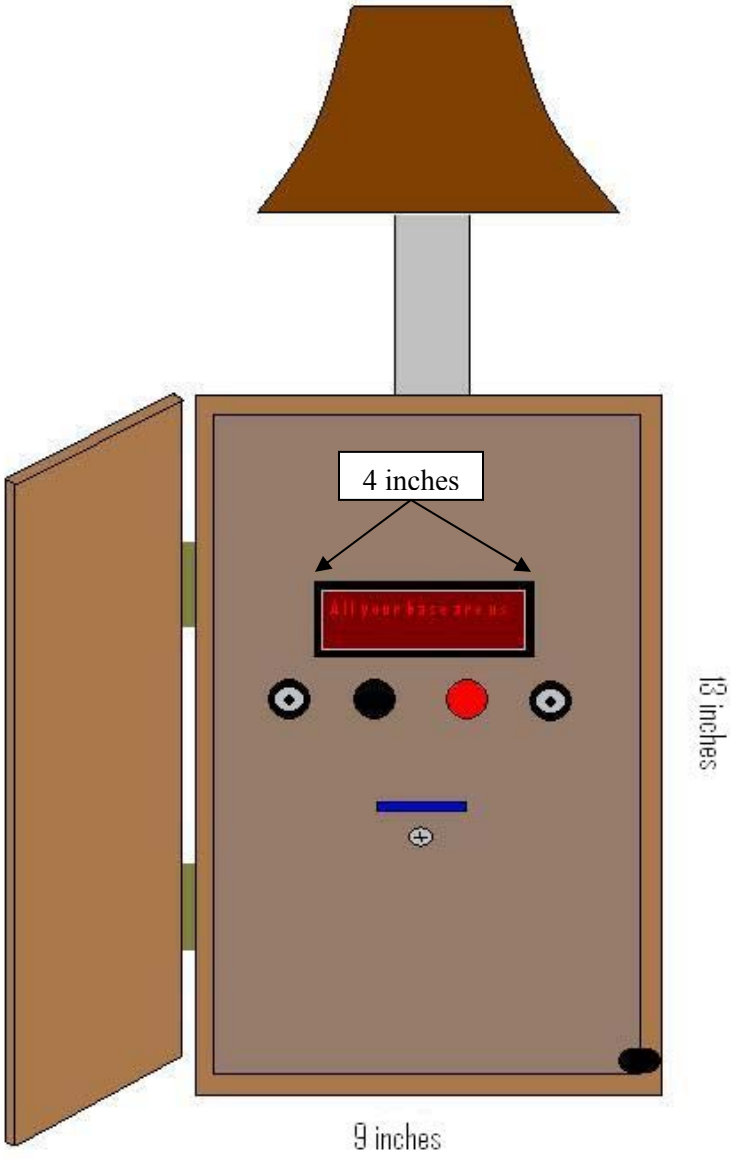


Figure B-1 Front View (With Panel)

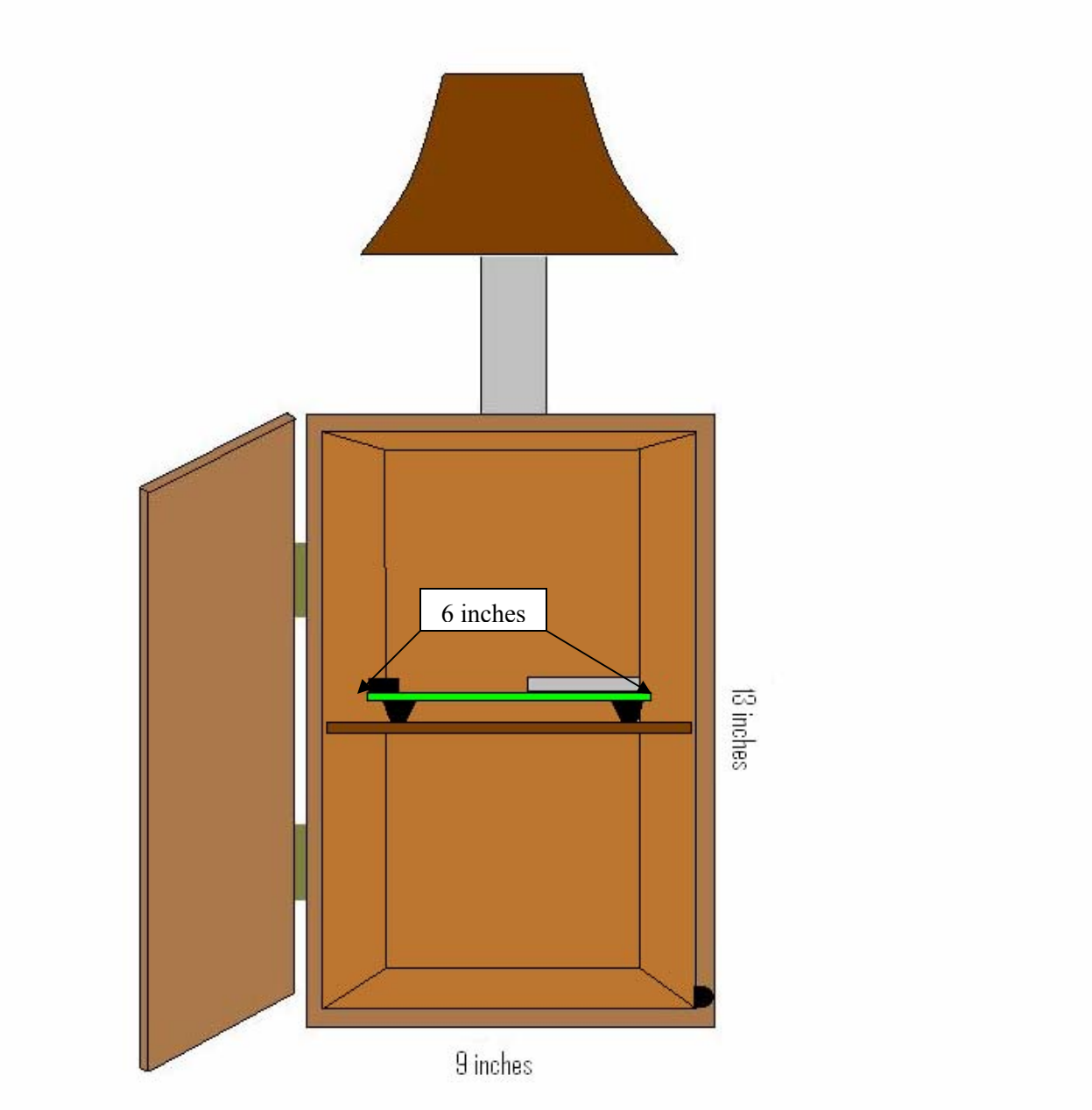


Figure B-2 Front View (No Panel)

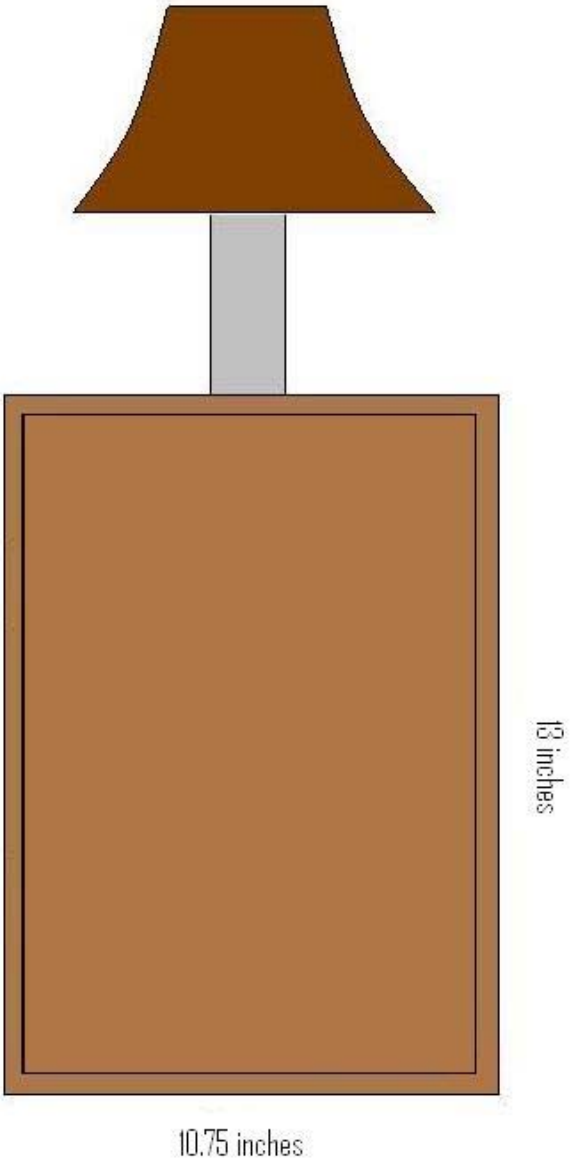


Figure B-3 Side View

Appendix C: Schematic

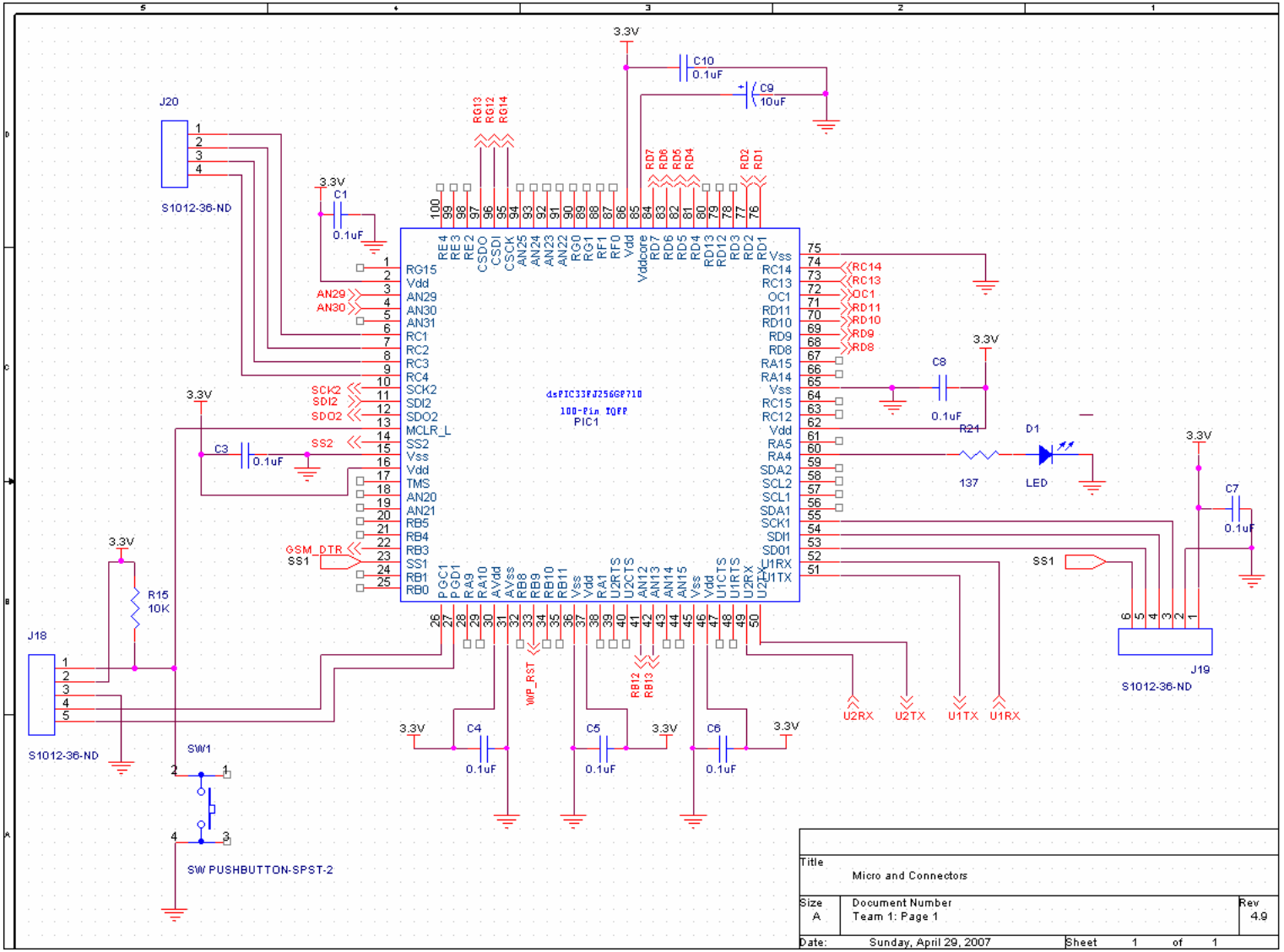


Figure C-1. Microcontroller (Block A in FMECA)

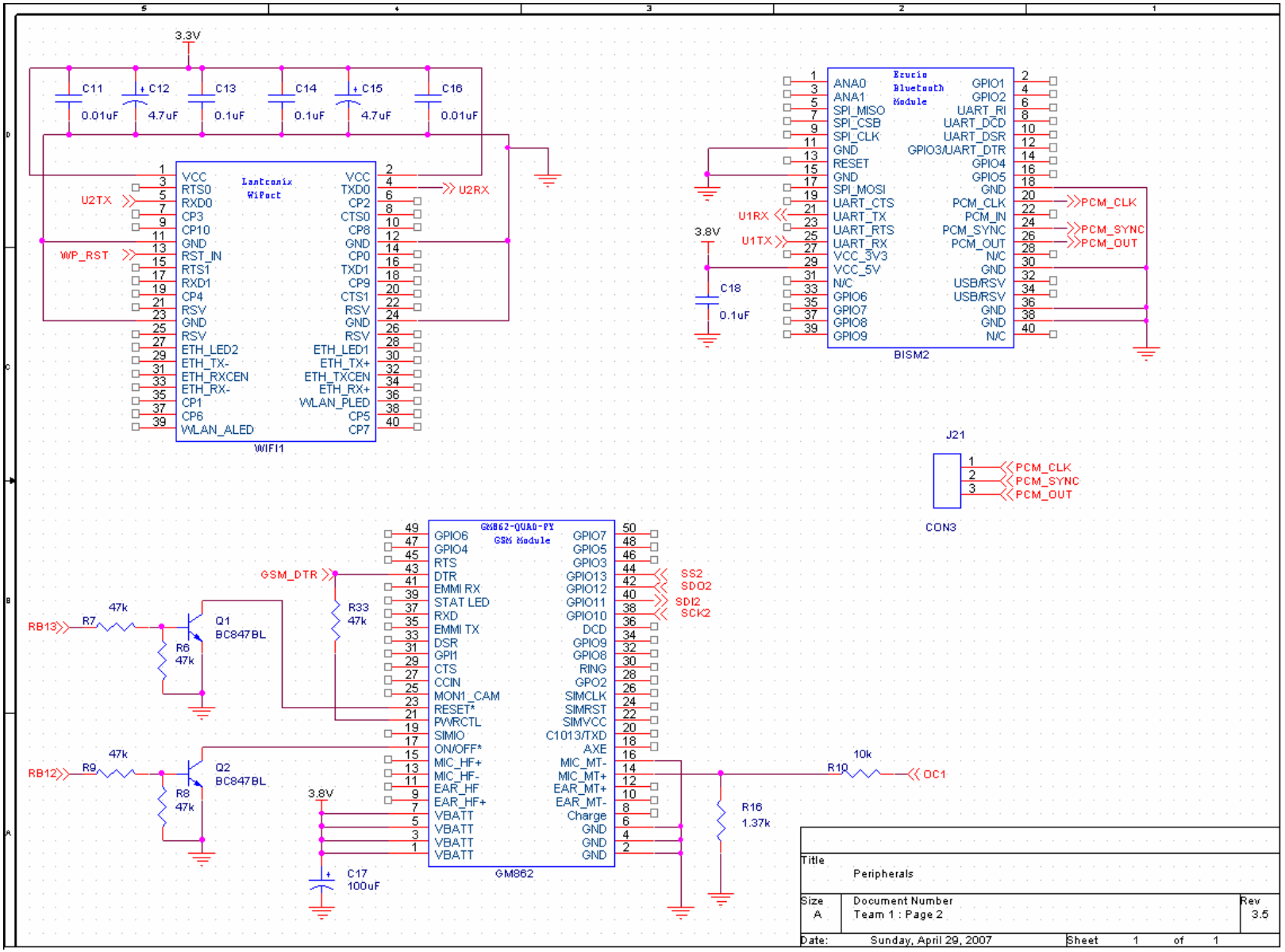


Figure C-2. Peripherals (Block C in FMECA)

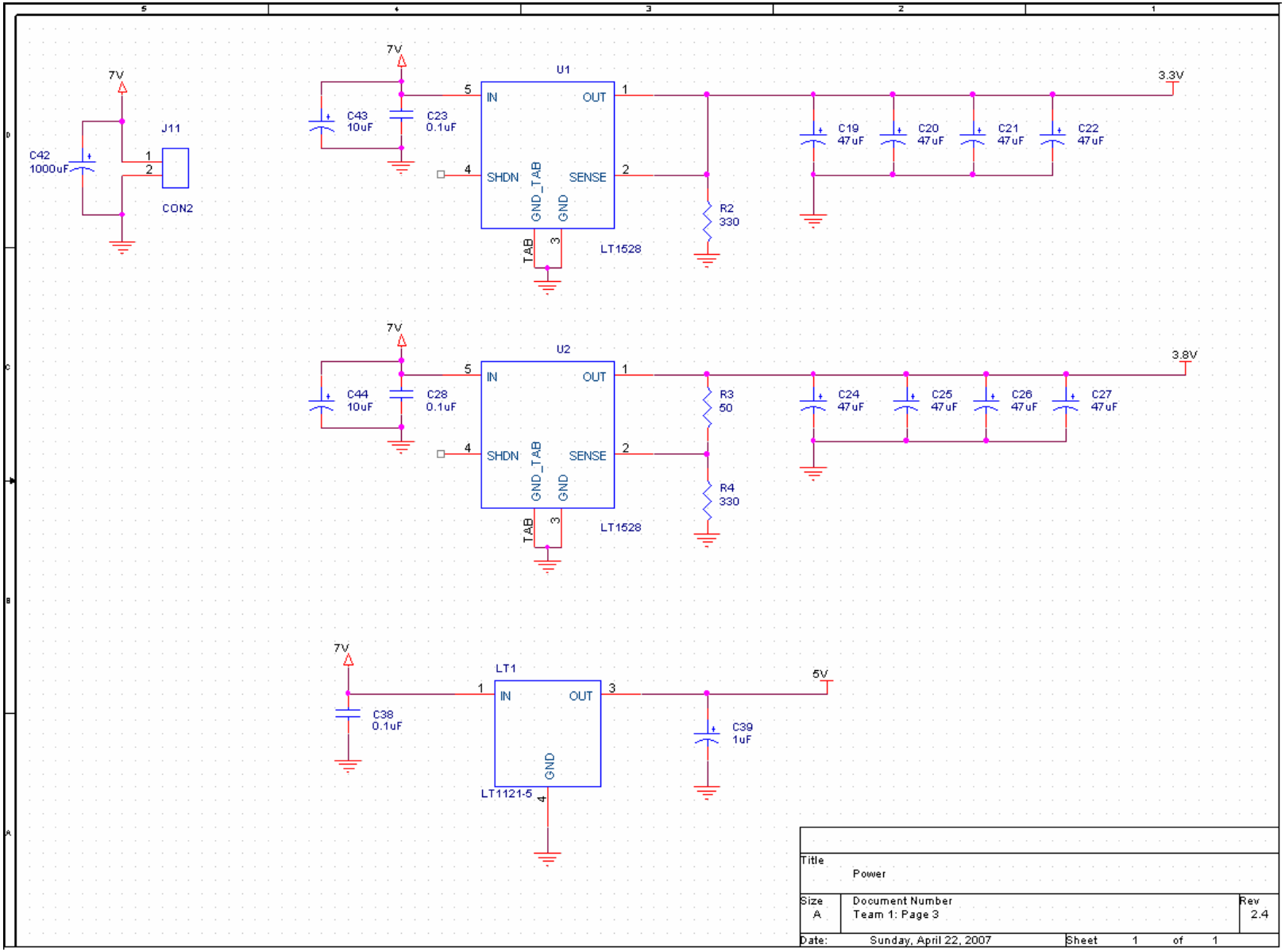


Figure C-3. Power (Block D in FMECA)

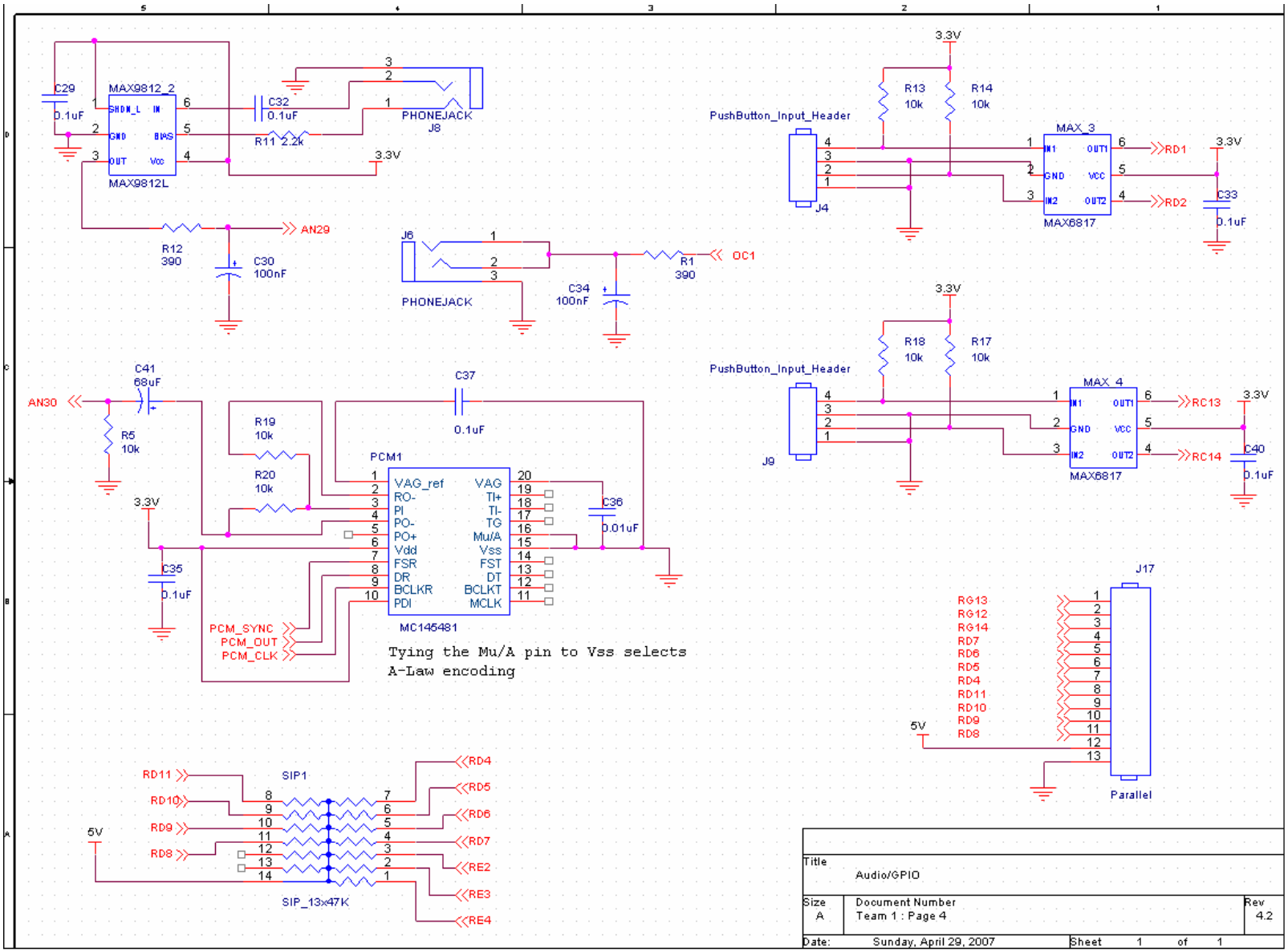


Figure C-4. Audio/GPIO (Block B in FMECA)

Appendix D: PCB Layout Top and Bottom Copper

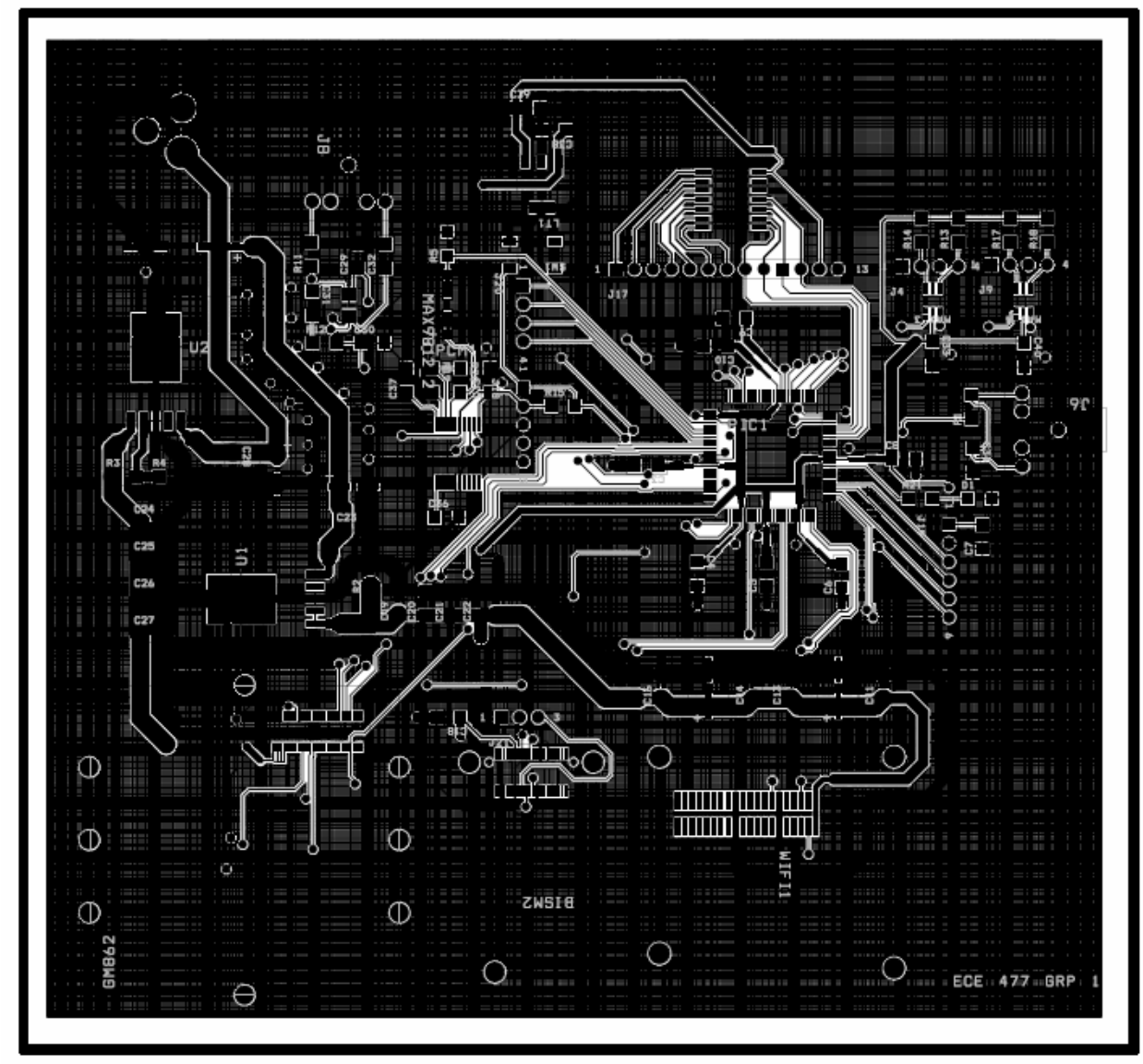


Figure D-1. PCB top copper

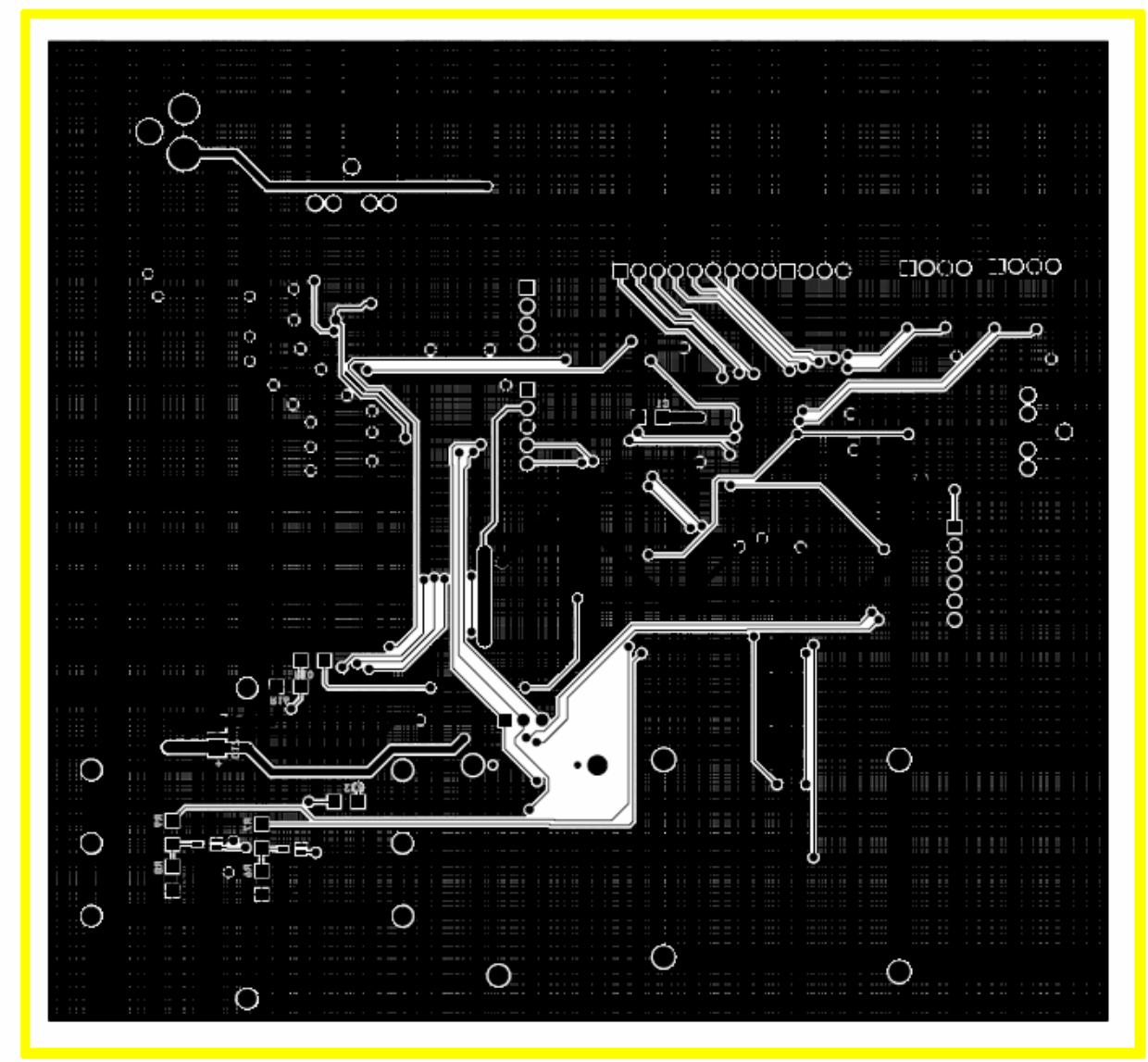


Figure D-2. PCB bottom copper

Appendix E: Parts List Spreadsheet

Vendor	Manufacturer	Part No.	Description	Unit Cost	Quantity	Total Cost
TekGear	Ezurio	BISM2 w/ integrated antenna	Bluetooth Module	$99.00	1	$99.00
SparkFun	Telit	GM862-QUAD-PY	GSM Module	$139.95	1	$139.95
Digi-Key	Linear Technologies	LT1121CST-5#PBF-ND	5V Regulator	$3.25	1	$3.25
Digi-Key	Maxim	MAX6817	Switch Debouncer	$3.52	2	$7.04
Digi-Key	Maxim	MAX9812L	Audio Amplifier	$0.67	1	$0.67
Digi-Key	Freescale Semiconductor	MC145481SDR2CT-ND	PCM Codec	$3.85	1	$3.85
Microchip	Microchip	dsPIC33FJ256GP710	dsPIC Microcontroller	$7.66	1	$7.66
Digi-Key	Linear Technologies	LT1528CQ#PBF-ND	3.3V and 3.8V regulators	$7.13	2	$14.26
Lantronix	Lantronix	Wiport	WiPort WiFi Module	$119.00	1	$119.00
SparkFun	4UCON	PCB-SDMMC	SD Card Holder	$14.95	1	$14.95
SparkFun	Hitachi	GDM1602K	16 x 2 LCD	$14.95	1	$14.95
			Total Cost for Major Component			$424.58

Table E-1. Major Components

Vendor	Manufacturer	Part No.	Description	Unit Cost	Quantity	Total Cost
Digi-Key	Kemet	399-1259-1-ND	0.1uF	$0.12	21	$2.52
Digi-Key	Taiyo Yuden	587-1333-1-ND	10uF	$0.42	1	$0.42
Digi-Key	Yageo Corp.	311-1174-1-ND	0.01uF	$0.08	3	$0.25
Digi-Key	Panasonic - ECG	PCE4293CT-ND	4.7uF	$0.27	2	$0.54
Digi-Key	Kemet	399-3318-1-ND	100uF	$1.30	1	$1.30
Digi-Key	Rohm	511-1452-1-ND	47uF	$0.28	8	$2.26
Digi-Key	AVX Corp.	478-3797-1-ND	100nF	$0.39	2	$0.77

Digi-Key	Kemet	399-1255-1-ND	1uF	$0.19	1	$0.19
Digi-Key	Panasonic - ECG	PCE3806CT-ND	68uF	$0.34	1	$0.34
Digi-Key	Panasonic - ECG	PCE3621CT-ND	1000uF	$1.89	1	$1.89
Digi-Key	Panasonic - ECG	PCE4179CT-ND	10uF	$0.17	2	$0.35
Digi-Key	Panasonic - SSG	P11532CT-ND	Red Status LED	$0.63	1	$0.63
SparkFun	4UCON	PRT-08032	Audio Jack 3.5mm	$0.95	2	$1.90
Digi-Key	CUI Inc	PJ-002A	Power Connector	$0.38	1	$0.38
Digi-Key	Diodes Inc.	BC847BW-FDICT-ND	NPN Transistor	$0.14	2	$0.28
Digi-Key	Yageo Corp.	311-390ERCT-ND	390 ohm	$0.08	2	$0.16
Digi-Key	Yageo Corp.	311-330ERCT-ND	330 ohm	$0.08	2	$0.16
Digi-Key	Yageo Corp.	311-49.9FRCT-ND	50 ohm	$0.09	1	$0.09
Digi-Key	Yageo Corp.	311-10.0KFRCT-ND	10k ohm	$0.09	9	$0.79
Digi-Key	Yageo Corp.	311-47KERCT-ND	47k ohm	$0.08	5	$0.39
Digi-Key	Vishay/Dale	541-2.20KFCT-ND	2.2k ohm	$0.05	1	$0.05
Digi-Key	Rohm	RHM1.37KFCT-ND	1.37k ohm	$0.05	1	$0.05
Digi-Key	Rohm	RHM137FCT-ND	137 ohm	$0.05	1	$0.05
Digi-Key	CTS Corp.	768-141-R47K-ND	SIP 13x47K	$0.79	1	$0.79
Mouser	ALPS	688-SKQGAB	Surface Mount Push Button	$0.71	1	$0.71
Radio Shack		275-1556	Momentary Pushbutton Switch2-Pk.	$2.99	1	$2.99
			Total Cost for Discrete Components			$20.24

Table E-2. Discrete Components

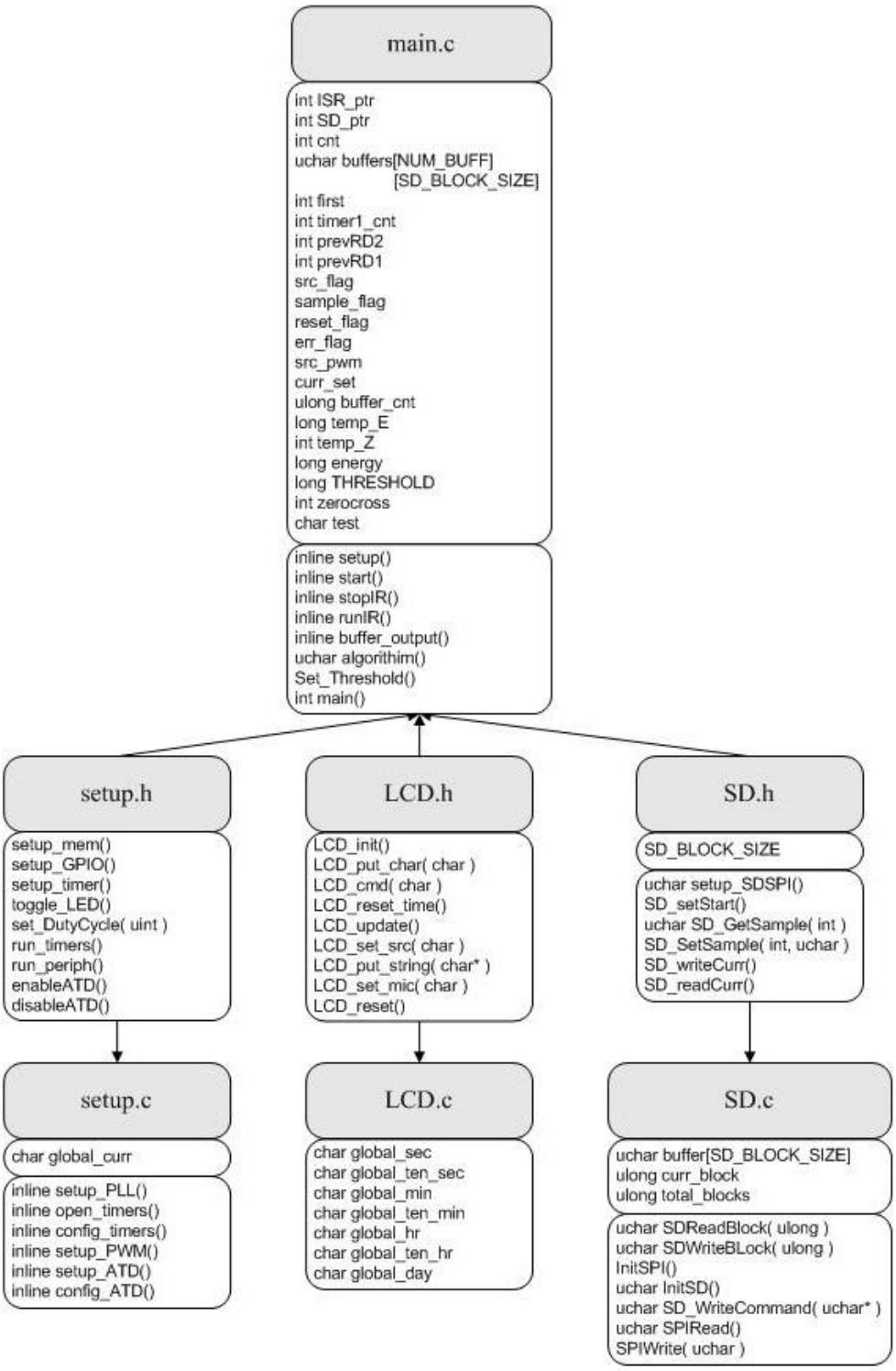
Vendor	Manufacturer	Part No.	Description	Unit Cost	Quantity	Total Cost
Home Depot	Unknown	N/A	Wood (12in x 4ft x 1in)	$8.25	1	8.25
Wal-Mart	Unknown	Unkown	Lamp Cord, base and socket	$9.99	1	9.99
Target	GE	Unkown	Compact Fluorescent Light bulb	$5.90	1	5.9
Wal-Mart	Unknown	Unkown	Lamp shade	$7.99	1	7.99
Radio Shack	Radio Shack	42-2497	12-Inch Shielded Stereo Audio Cable	$4.99	2	9.98
Target	Unknown	Unkown	Extension Cord	$2.99	1	2.99
Radio Shack	Radio Shack	273-1696	Wall wart	$29.99	1	29.99
Mouser	Tyco Electronics / AMP	571-1042574	5 pin connectors for LCD, LCD headers	$0.29	2	0.58
Mouser	Tyco Electronics / AMP	571-5-103957-1	2 pin connector for LCD	$0.38	1	0.38
Mouser	Tyco Electronics / AMP	571-1042575	6 pin connectors for SD card, LCD, LCD headers	$0.34	4	1.36
Mouser	Tyco Electronics / AMP	571-1042573	4 pin connectors for Push Buttons	$0.29	2	0.58
*Wire for the connectors was acquired from the Senior Design Lab						
			Total Cost of Miscellaneous Components			77.99
			Total Cost of Project			$522.81

Table E-3. Miscellaneous Components

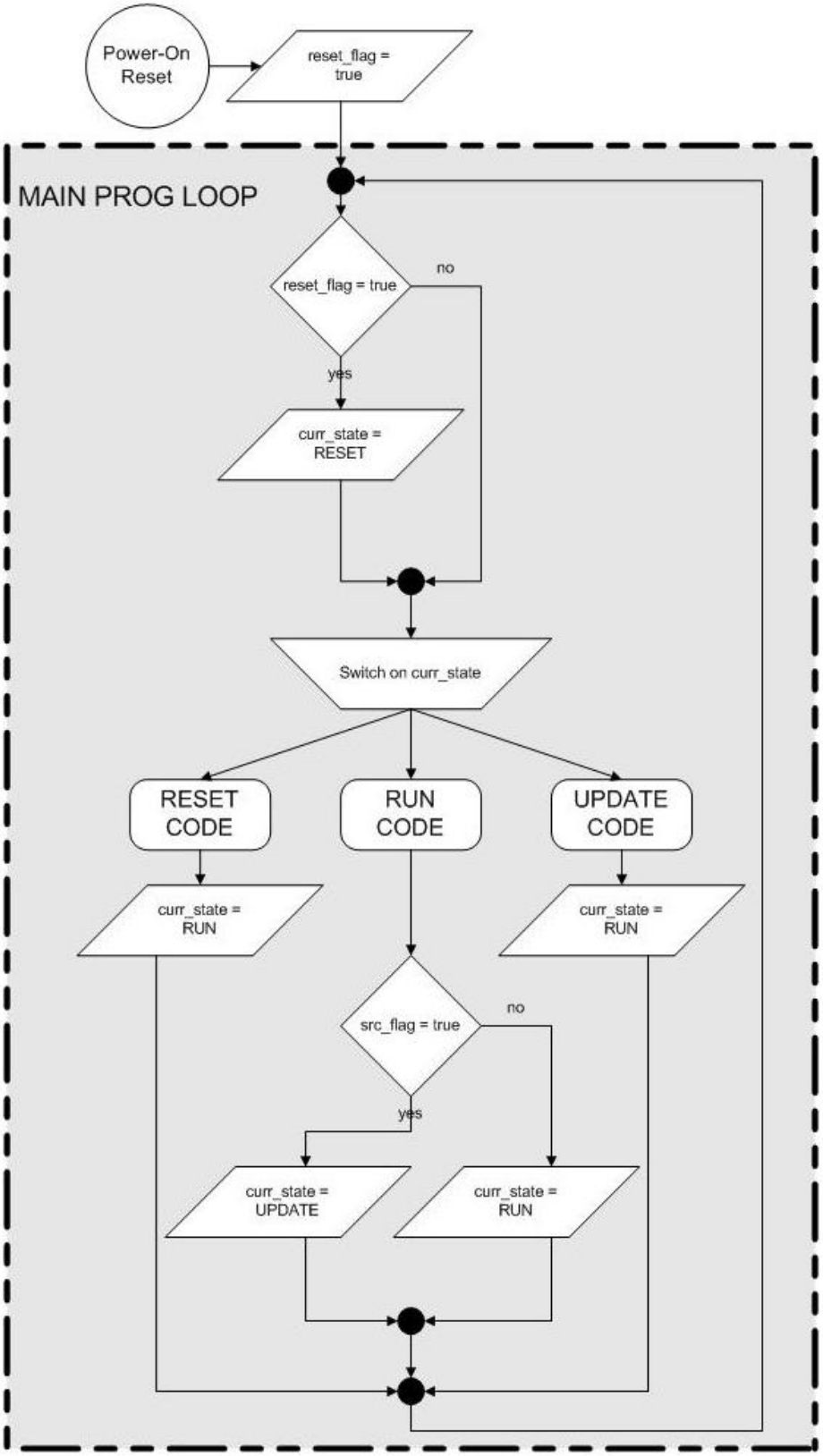
Appendix F: Software Listing

Code organization

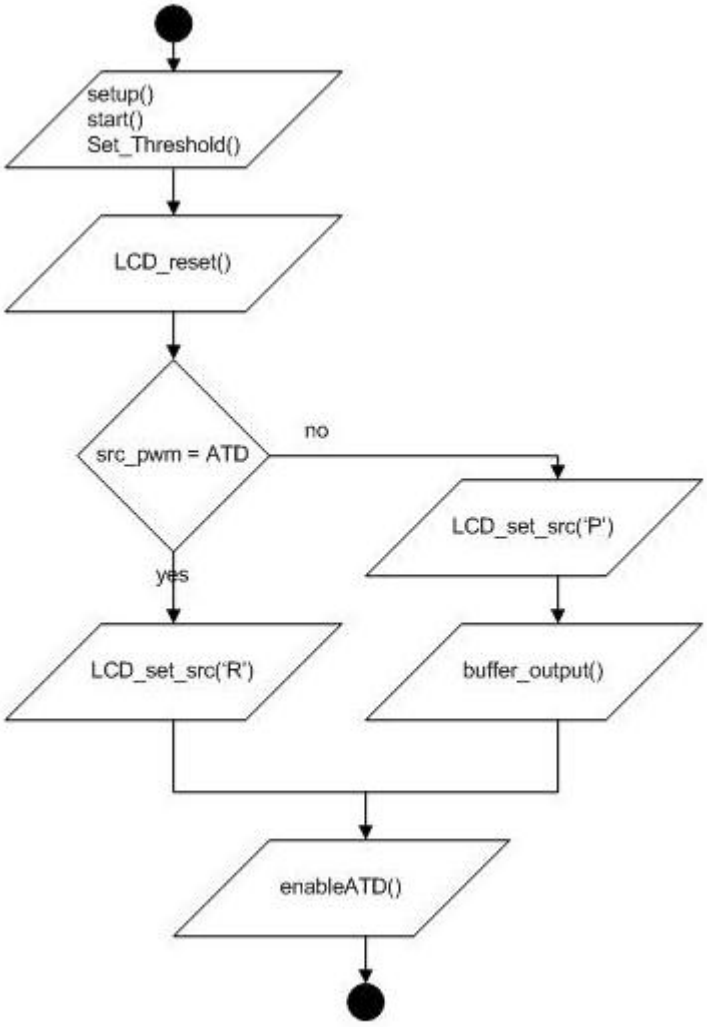
Item at the head of the arrow has control over the functions and data at the tail. The organization is the name (in the gray box), the global variables, and then the functions either declared and/or defined in that file.



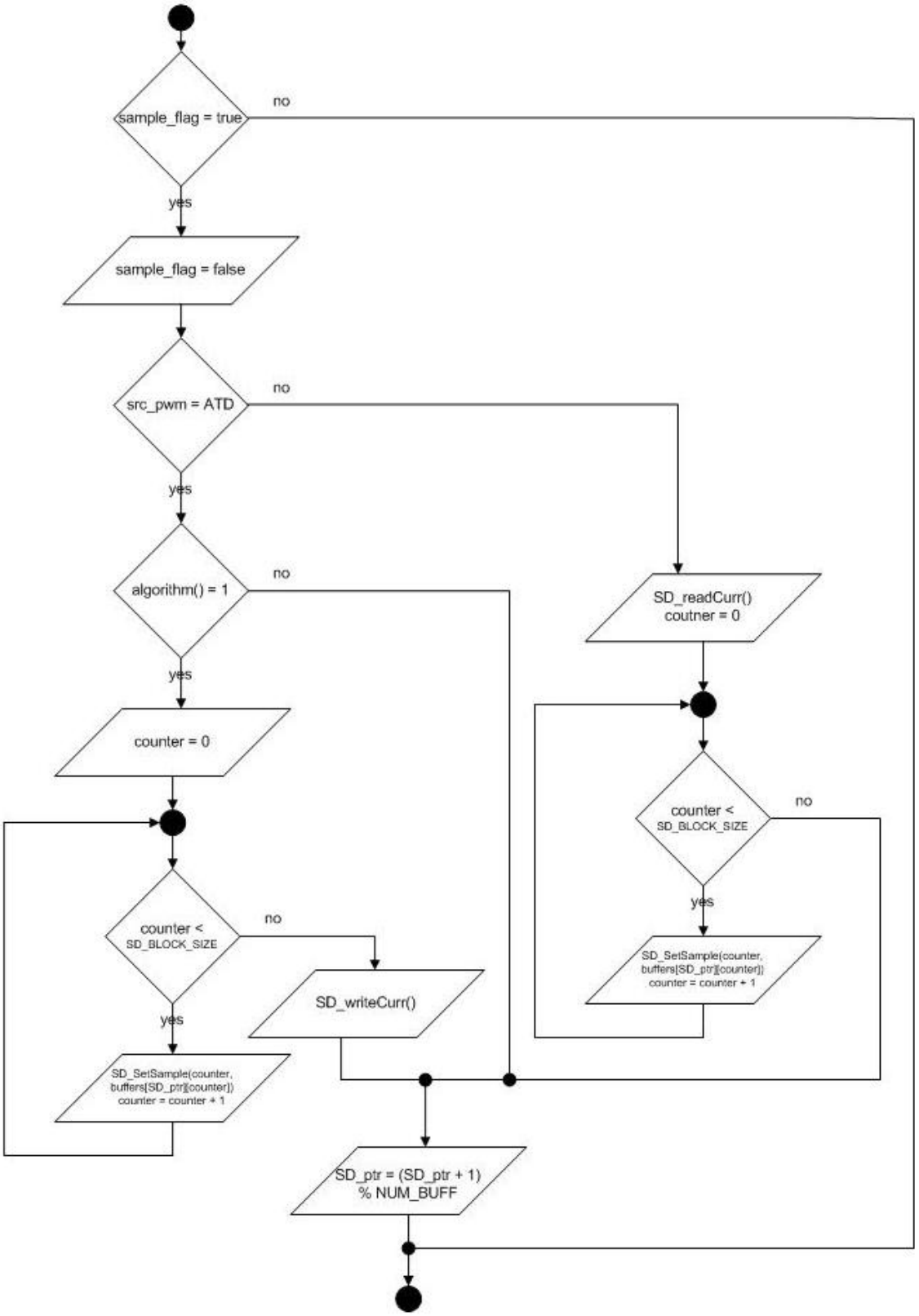
Main Program Loop (State Machine) Flow Chart



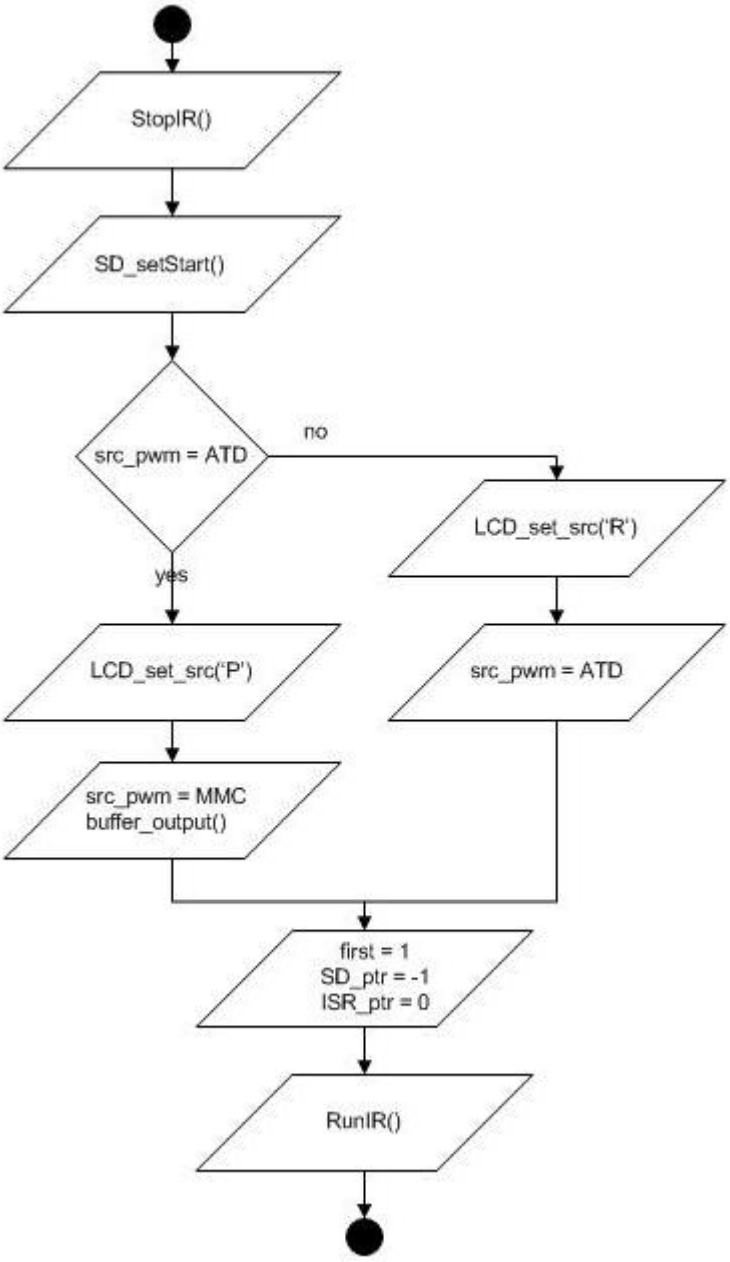
RESET CODE Flow Chart



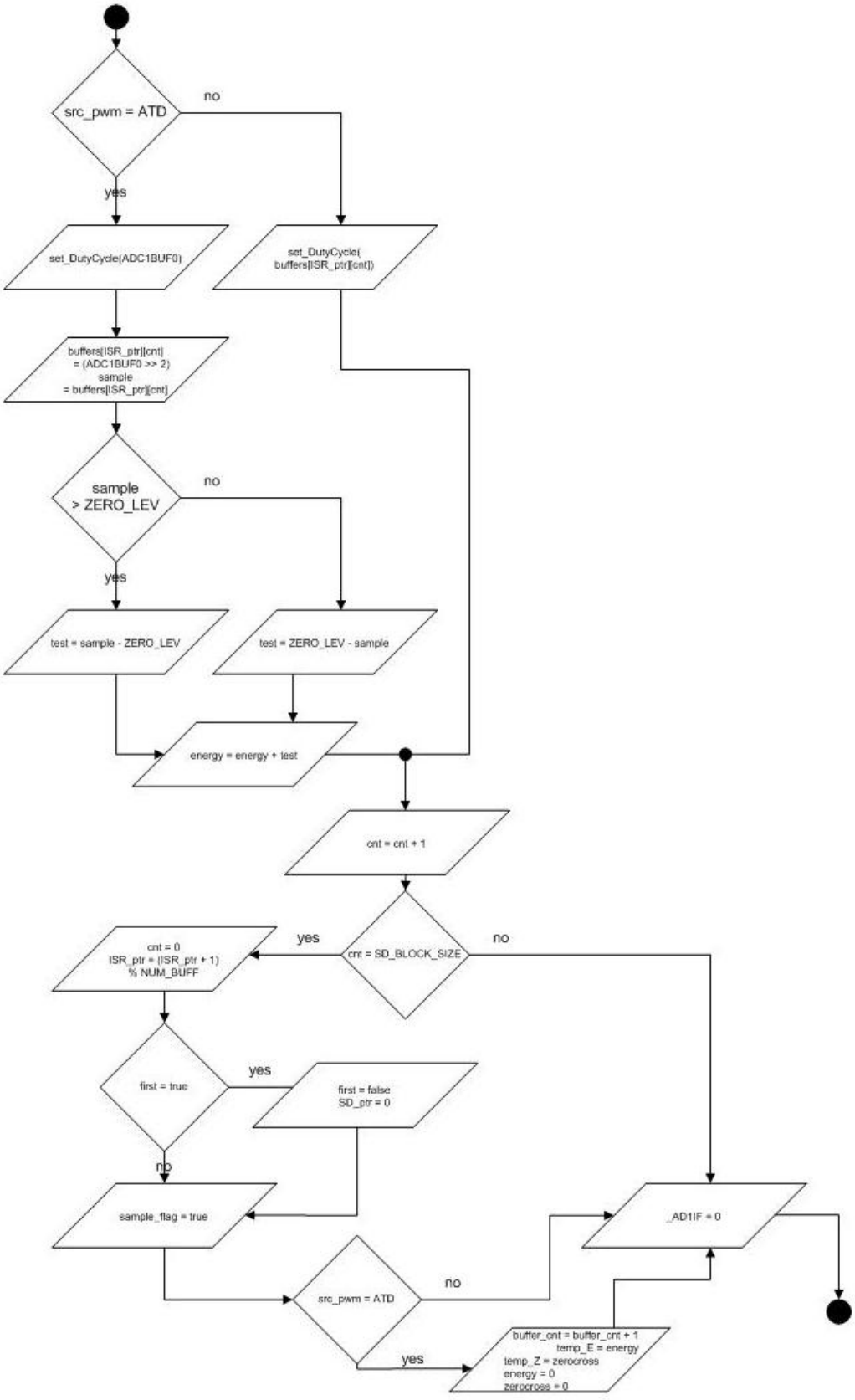
RUN CODE Flow Chart



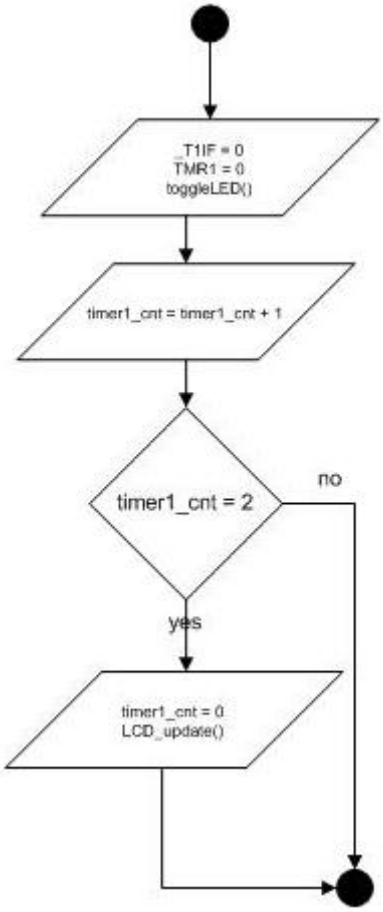
UPDATE CODE Flow Chart



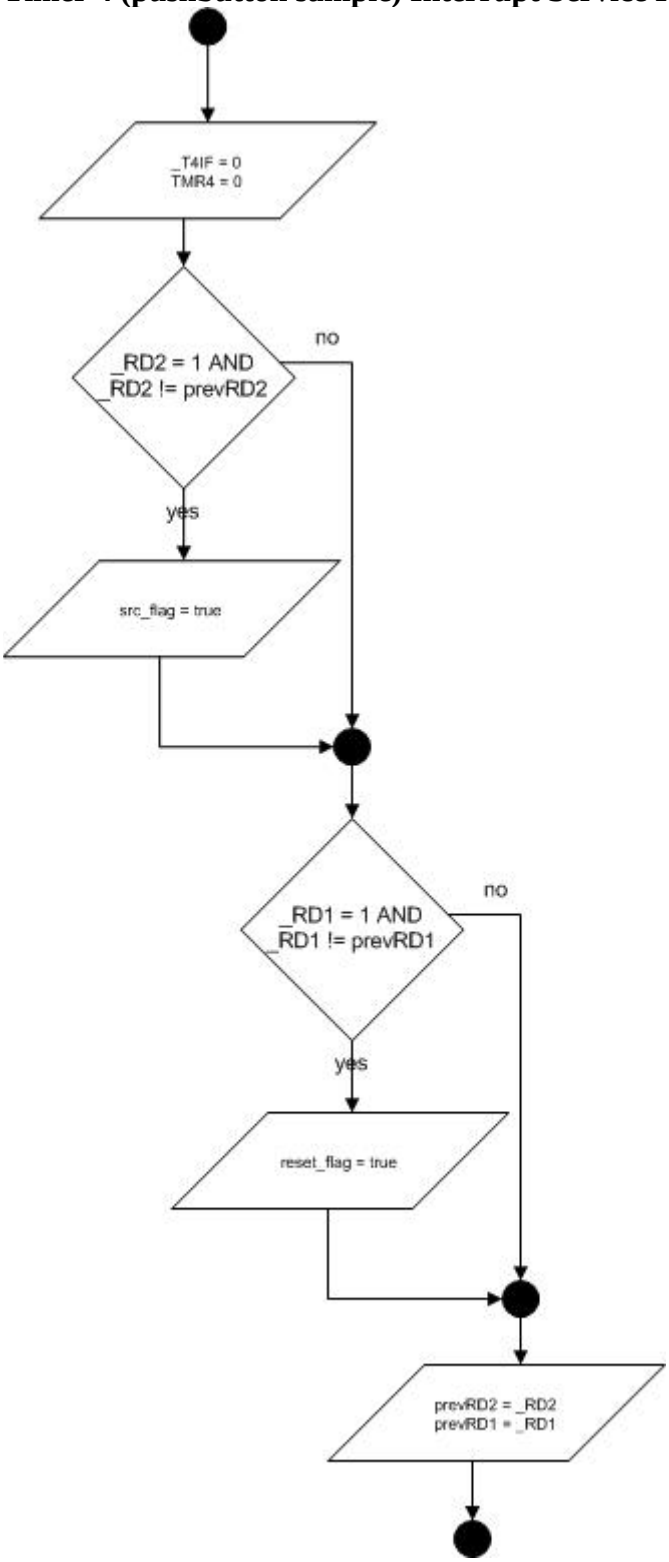
ATD Conversion Interrupt Service Routine Flow Chart



Timer 1 (heartbeat, ½ second) Interrupt Service Routine Flow Chart



Timer 4 (pushbutton sample) Interrupt Service Routine Flow Chart



Software Files

main.c

```
/*
    main.c
    Author: Karl Herb
    Contributors: Justin Lanning, Brad Sokola, Warren Santner
    Core software file for project.  Designed for a dsPIC33FJ25GP710
    and for building to assembly and binary with Microchip's MPLAB C30
    compiler.
    This code contains inline functions for setting up the microcontroller,
    and times all of the devices according to timing outlined in setup.c
    (main.c pulls in setup.h to gain access to the setup configurations).
    The code contains a main polling loop and is a hybrid model with
    interrupt service routines for timer 1, timer 4, and the completion
    of an ATD conversion.
*/

/*
    Header files
*/
#include <p33FJ256GP710.h>
#include "setup.h"
#include "SD.h"
#include "LCD.h"

/*  configuration register setup
    these must be defined here and must
    use the macros found in p33FJ.... .h
    this makes the device have an IF of 33.3MHz and
    clock of 66.6MHz
*/
_FBS(RBS_NO_RAM     &BSS_NO_FLASH  &BWRP_WRPROTECT_OFF      );
_FSS(RSS_NO_RAM     &SSS_NO_FLASH  &SWRP_WRPROTECT_OFF      );
_FGS(GSS_OFF        &GCP_OFF       &GWRP_OFF                );
_FOSCSEL(FNOSC_FRCPLL&IESO_OFF      &TEMP_OFF               );
_FOSC(FCKSM_CSDCMD  &OSCIOFNC_ON   &POSCMD_NONE             );
_FWDT(FWDTEN_OFF   &WINDIS_OFF & WDTPRE_PR32 & WDTPOST_PS1  );
_FPOR(FPWRT_PWR1);

/*  setting the size of the sample buffers
*/
#define NUM_BUFF (6)
#define PIVOT (5)
/* Zero Voltage Level (aka threshold)*/
#define ZERO_LEV (0x73)

/*  Controls for the buffering.  A large array is mantained of
    the specified number of buffers each with a block size for
    SD card I/O (see SD.h for SD_BLOCK_SIZE define).  The ISR_ptr
    is controled by AD1 ISR, and SD_ptr is controlled by the
    main polling loop and is prepped on its first necessary use.
    cnt is used in the AD1 ISR.
*/
int ISR_ptr = 0;
int SD_ptr = 0;
int cnt = 0;
```

```
__attribute__ ((far)) unsigned char buffers[NUM_BUFF][SD_BLOCK_SIZE];


/* flags for prepping SD_ptr and for threshold setup */
int first = 1;
/* Controls whether time displayed gets updated on LCD */
char threshold_setting = 0;

/*  counts for timer 1 */
int timer1_cnt= 0;
int prevRD2 = 1;  //active low
int prevRD1 = 1;  //active low

/*  flags for transitions */
/*  TRUE = Need to address (like _IF), FALSE = nothing to respond to */
enum flag_typ { TRUE, FALSE };
/*  TRUE if want to change the SRC of the PWM and buffering I/O */
enum flag_typ src_flag;
/*  TRUE to get or set more data to/from SD */
enum flag_typ sample_flag;
/*  TRUE if a reset is desired */
enum flag_typ reset_flag;
/*  TRUE if buffer overrrun  */
enum flag_typ err_flag;

/*  PWM src control */
/*  MMC is confusing, but SD is used as a file name (SD.h, SD.c)
*/
enum pwmSrc_typ { ATD, MMC };
enum pwmSrc_typ src_pwm;

/*  Algo Variables*/
unsigned char buffer_cnt = 0;
long int temp_E = 0;
int temp_Z = 0;
long int energy = 0;
long int THRESHOLD = 0;
int zerocross = 0;
/*  Sample value with offset subtracted*/
char amplitude;

/*  heartbeat and uptime count (every second)
    event happens every 1/2 sec (see setup.h)
*/
void _ISR _T1Interrupt( void )
{
    unsigned int spi_data;
    /*clear flags, toggle heartbeat, and inc cnt*/
    _T1IF = 0;
    TMR1  = 0;
    toggleLED();
    timer1_cnt++;
    /*has 1 sec passed?*/
    if(timer1_cnt == 2)
    {
        timer1_cnt = 0;
        /*Check to make sure we aren't doing a threshold setting*/
        if(!threshold_setting)
```

```
        {   /*1 second has passed, Update LCD uptime, and send to GSM*/
            LCD_update();
            spi_data = ReadSPI2();
            LCD_gsm_stat(spi_data-48);
        }
    }
    return;
}

/*  check for event on pushbuttons, timing is per setup.h
*/
void _ISR _T4Interrupt( void )
{
    _T4IF = 0;
    TMR4 = 0;
    /*completes a push down push up cycle?
    if so set the appropriate flag based on which
    push button
    */
    if(_RD2 == 1 && _RD2 != prevRD2)
    {
        src_flag = TRUE;
    }
    if(_RD1 == 1 && _RD1 != prevRD1)
    {
        reset_flag = TRUE;
    }
    /*update previous for next use */
    prevRD2 = _RD2;
    prevRD1 = _RD1;
    return;
}

/*ATD conversion complete ISR, this also duals as control of the PWM, and if
the ATD is not buffering, its conversion is ignored*/
void _ISR _ADC1Interrupt( void )
{
    int sample;

    if(src_pwm == ATD)
    {
        /*output the sample, and make the 10bit sample into an 8bit
        sample to gain maximal use of the SD space*/
        set_DutyCycle(ADC1BUF0);
        buffers[ISR_ptr][cnt] = (ADC1BUF0 >> 2);
        sample = buffers[ISR_ptr][cnt];
        /*adjust the amplitude for the processing algorithm on each
        sample, and total the energy approrpiately*/
        if (sample > ZERO_LEV)
        {
            amplitude = sample - ZERO_LEV;
        }
        else
        {
            amplitude = ZERO_LEV - sample;
        }
        energy += amplitude;
    }
```

```
    else
    {   /*output the value in the current buffer position to the
        PWM for audio output (remembering to make the 8bit back into
        10bit for near consistency*/
        unsigned char theSamp = buffers[ISR_ptr][cnt];
        set_DutyCycle(theSamp << 2);
    }

    /*adjust pointers and look for events*/
    cnt++;
    if(cnt == SD_BLOCK_SIZE)
    {
        /*adjust count and if necessary ISR ptr
        and request for more data/data output if need be*/
        cnt = 0;
        ISR_ptr = (ISR_ptr + 1) % NUM_BUFF;
        if(first == 1)
        {   /*prep SD ptr*/
            first = 0;
            SD_ptr = 0;
        }
        /*flag the event and setup for the signal processing
        algo if need be*/
        sample_flag = TRUE;
        if(src_pwm == ATD)
        {
            buffer_cnt++;
            temp_E = energy;
            temp_Z = zerocross;
            energy = 0;
            zerocross = 0;
        }
    }

    _AD1IF = 0;
    return;
}

/*Function for algorithm
*/
void Set_Threshold( void )
{
    enableATD();
    /*take control for the time being...*/
         while(buffer_cnt < 20)
         {
                  if(sample_flag == TRUE)
                  {
                           sample_flag = FALSE;
                           THRESHOLD += temp_E;
        }
    }
    disableATD();
    /*set up pointers again in case any sideffects*/
    buffer_cnt = 0;
    first = 1;
    ISR_ptr = 0;
    SD_ptr = -1;
```

```
    cnt = 0;
    /*set the threshold for processing*/
    THRESHOLD = THRESHOLD / 20;
    return;
}

/*state machine for the main loop*/
enum state_typ { RESET, RUN, UPDATE };
enum state_typ curr_state;

/*
SETUP FUNCTION
*/
inline void setup( void )
{
    /*internal setups (setup.c defined functions)*/
    setup_mem();
    setup_GPIO();
    setup_timing();
    setup_periph();

    /*Initializes LCD*/
    LCD_init();
    /*sets cursor at head of first line*/
    LCD_cmd(0x80);
    LCD_put_string("Insert");
    /*sets cursor at head of second line*/
    LCD_cmd(0b11000000);
    LCD_put_string("SD Card");

    /*SD card*/
    setup_SDSPI();
    SD_setStart();

    /*switches and flags*/
    src_flag = FALSE;
    sample_flag = FALSE;
    reset_flag = FALSE;
    err_flag = FALSE;
    src_pwm = ATD;
    prevRD1 = 1;
    prevRD2 = 1;
    first = 1;

    /*ptrs*/
    ISR_ptr = 0;
    SD_ptr = -1;
    cnt = 0;

    /*Initializes LCD to insert SD screen*/
    LCD_init();
    LCD_cmd(0x80);
    LCD_put_string("Setting");
    LCD_cmd(0b11000000);
    LCD_put_string("Threshold");

    return;
}
```

```
/*
RUNTIME ENABLE
*/
inline void start( void )
{
    run_timers();
    run_periph();
    return;
}
/*fills up the buffer at the start of the transition to MMC output
*/
inline void buffer_output( void )
{
          int buffer_num = 0;
    int samp_num = 0;
          prints(READ_MSG);
          for(buffer_num = 0; buffer_num < NUM_BUFF; buffer_num++)
          {
                    SD_readCurr();
                    for(samp_num = 0; samp_num < SD_BLOCK_SIZE; samp_num++)
                    {
                              buffers[buffer_num][samp_num] = SD_GetSample(samp_num);
                    }
          }
          return;
}

/*check for the energy being above the threshold*/
unsigned char algorithm( void )
{
          if(temp_E >= (THRESHOLD+150))
          {
                    return 1;
          }
          return 0;
}

/*
MAIN FUNCTION
*/
int main ( void )
{
          int curr_block = 0;
          int counter = 0;
          int GSM_init = 1;
          /*force start*/
          reset_flag = TRUE;
          /*main loop is a giant state machine*/
          while(1)
          {
                    if(reset_flag == TRUE)
                    {
                              curr_state = RESET;
                              reset_flag = FALSE;
                    }
                    switch(curr_state)
                    {
```

```
 case RESET:
          setup();
          start();
          /*Only toggle the On/Off pin on the GSM
 if this is a hardware reset*/
          if(GSM_init)
          {
          if(GSM_init)
          {
                    _RB12 = 1;         /*Set On/Off pin high
                              for 1 sec*/
                    threshold_setting = 1;
                    Set_Threshold();  /*Set the energy
                              threshold for algo*/
                    threshold_setting = 0;
                    LCD_reset();
                    /*It has been about a second since RB12
          was set high to turn on the GSM module
                    so set it low now*/
                    _RB12 = 0;
                    GSM_init = 0;
          }
else
          {
                    threshold_setting = 1;
          /*Set the energy threshold for algorithm*/
                    Set_Threshold();
                    threshold_setting = 0;
                    LCD_reset();
          }

          if(src_pwm == ATD)
          {
          /*set LCD to show recording from ATD*/
                    LCD_set_src('L');
                    curr_state = RUN;
          }
          else
          {
          /*set LCD to show playback from SD card*/
                    LCD_set_src('P');
                    buffer_output();
                    curr_state = RUN;
          }
          enableATD();
          break;
 case RUN:
          while(sample_flag == TRUE)
          {
                    sample_flag = FALSE;
                    if(src_pwm == ATD)
                    {
                    /*need to save sample?*/
                    if( algorithm() == 1)
                    {/*have speech (most likely)*/
                              /*copy buffer to SD scope*/
                              for(curr_block = 0;
                              curr_block < 1;
```

```
                                        curr_block++)
                                                {
                                                for(counter = 0;
                                                counter
                                            < SD_BLOCK_SIZE;
                                                counter++)
                                                {
                                        SD_SetSample(counter,
                                        buffers[SD_ptr][counter]);
                                                }
                                                SD_ptr = (SD_ptr + 1)
                                                % NUM_BUFF;
                                        /*write sample block*/
                                                SD_writeCurr();
                        /*set LCD to show writing block to SD card*/
                                                LCD_set_src('R');
                                        }
                                }
                                else
                                {
                                        SD_ptr = (SD_ptr + 1) %
                                        NUM_BUFF;
                                        /*set LCD to show listening*/
                                        LCD_set_src('L');
                                }
                                }
                                else
                                {
                                /*read in a block*/
                                        SD_readCurr();
                                /*copy block to main space*/
                                        for(counter = 0;
                                        counter < SD_BLOCK_SIZE;
                                        counter++)
                                        {
                                        buffers[SD_ptr][counter] =
                                        SD_GetSample(counter);
                                        }
                                        SD_ptr = (SD_ptr + 1) %
                                        NUM_BUFF;
                                }
                        /*need to update source (PWM or SD?)*/
                                if(src_flag == TRUE)
                                {
                                        src_flag = FALSE;
                                /*disable IR function in
                                state machine, called earlier yes*/
                                        disableATD();
                                        DisableIntT1;
                                curr_state = UPDATE;
                                }
                        }
                        break;
                case UPDATE:
                /*go to beginning of card*/
                        SD_setStart();
                        if(src_pwm == ATD)
                        {
```

```
                                        /*PWM now, prime with buffer output func*/
                                        /*set LCD to show playback from SD card*/
                                                LCD_set_src('P');
                                                src_pwm = MMC;
                                                buffer_output();
                                        }
                                        else
                                        {       /*ATD now*/
                                        /*set LCD to show listening on ATD*/
                                                LCD_set_src('L');
                                                src_pwm = ATD;
                                        }
                                        /*reset pointers and flags*/
                                        first = 1;
                                        SD_ptr = -1;
                                        ISR_ptr = 0;
                                        curr_state = RUN;
                                        EnableIntT1;
                                        enableATD();
                                        break;
                                default:
                                        curr_state = RUN;
                                break;
                        }

                }

}

/* end main.c */
```

setup.h

```
/*
    setup.h
    Author: Karl Herb
    This code is derived from modules, such as timer.h, adc12.h, etc.
    included with the compiler.  Definitions have literally been copied
    and pasted and used for our setup.  This way 1) the amount of header
    files is greatly reduced and 2) only what is necessary is contained
    herein.
    This setups the timers, ATD, PWM, serial (only for debug), SPI
    (for the GSM), and GPIO (except for the LCD).
*/
#include <p33FJ256GP710.h>
#ifndef SETUP_H
#define SETUP_H

/*timer1 defines, setup for tick every 1/2 of a second*/
#define T1_OFF              0x7fff      /*Timer1 OFF */
#define T1_IDLE_CON         0xdfff      /*operate during sleep */
#define T1_GATE_OFF         0xffbf      /*Timer Gate time accum off*/
#define T1_PS_1_256         0xffff      /*1:256 */
#define T1_SYNC_EXT_OFF     0xfffb      /*Do not synch external clk input*/
#define T1_SOURCE_INT       0xfffd      /*Internal clock source */
#define T1_SETUP
T1_OFF&T1_IDLE_CON&T1_GATE_OFF&T1_PS_1_256&T1_SYNC_EXT_OFF&T1_SOURCE_INT
#define T1_PERIOD                   0xfda7
#define T1_INT_OFF          0xfff7      /*Interrupt Disable */
#define T1_INT_PRIOR_3      0xfffb      /*011 = Interrupt is priority 3*/
#define T1_CONFIG T1_INT_OFF&T1_INT_PRIOR_3
#define EnableIntT1                   asm("BSET IEC0,#3")
#define DisableIntT1                  asm("BCLR IEC0,#3")

/*timer 2 defines, setup for 10 bit precision PWM; use with OC*/
#define T2_OFF              0x7fff      /*Timer2 OFF */
#define T2_IDLE_CON         0xdfff      /*operate during sleep */
#define T2_GATE_OFF         0xffbf      /*Timer Gate time accum off*/
#define T2_PS_1_1           0xffcf      /*Prescaler 1:1*/
#define T2_32BIT_MODE_OFF   0xfff7
#define T2_SOURCE_INT       0xfffd      /*Internal clock source */
#define T2_SETUP
T2_OFF&T2_IDLE_CON&T2_GATE_OFF&T2_PS_1_1&T2_32BIT_MODE_OFF&T2_SOURCE_INT
#define T2_PERIOD                   0x03ff
#define T2_INT_OFF          0xfff7      /*Interrupt Disable */
#define T2_CONFIG T2_INT_OFF
#define DisableIntT2                  asm("BCLR IEC0,#7")

/*timer 3 defines, setup for tick every 8 kHz; use with ATD*/
#define T3_OFF              0x7fff      /*Timer4 OFF */
#define T3_IDLE_CON         0xdfff      /*operate during sleep */
#define T3_GATE_OFF         0xffbf      /*Timer Gate time accum off*/
#define T3_PS_1_1           0xffcf      /*Prescaler 1:1   */
#define T3_SOURCE_INT       0xfffd      /* Internal clock source */
#define T3_SETUP
T3_OFF&T3_IDLE_CON&T3_GATE_OFF&T3_PS_1_1&T2_32BIT_MODE_OFF&T3_SOURCE_INT
#define T3_PERIOD                   0x0fff
#define T3_INT_OFF          0xfff7      /*Interrupt Disable */
#define T3_INT_PRIOR_2      0xfffa      /*011 = Interrupt is priority 3 */
```

```
#define T3_CONFIG T3_INT_OFF&T3_INT_PRIOR_2
#define DisableIntT3                    asm("BCLR IEC0,#7")

/*timer 4 defines for pushbutton sampling every 2ms*/
#define T4_OFF              0x7fff      /*Timer4 OFF */
#define T4_IDLE_CON         0xdfff      /*operate during sleep */
#define T4_GATE_OFF         0xffbf      /*Timer Gate time accum off*/
#define T4_PS_1_256         0xffff      /*Prescaler 1:256   */
#define T4_32BIT_MODE_OFF   0xfff7
#define T4_SOURCE_INT       0xfffd      /*Internal clock source */
#define T4_SETUP
T4_OFF&T4_IDLE_CON&T4_GATE_OFF&T4_PS_1_256&T4_32BIT_MODE_OFF&T4_SOURCE_INT
#define T4_PERIOD                     0x00ff
#define T4_INT_PRIOR_5      0xfffd      /*101 = Interrupt is priority 5 */
#define T4_INT_OFF          0xfff7      /*Interrupt Disable */
#define T4_CONFIG T4_INT_OFF&T4_INT_PRIOR_5
#define EnableIntT4                     asm("BSET IEC1,#11")
#define DisableIntT4                    asm("BCLR IEC1,#11")

/*PWM defines*/
#define OC_IDLE_CON                0xdfff
#define OC_TIMER2_SRC              0xfff7
#define OC_PWM_FAULT_PIN_DISABLE   0xfffe
#define PWM_SETUP OC_IDLE_CON&OC_TIMER2_SRC&OC_PWM_FAULT_PIN_DISABLE

/*ATD interrupt control*/
#define EnableIntADC1                    asm("BSET IEC0,#13")
#define DisableIntADC1                   asm("BCLR IEC0,#13")

/*
external function definitions
*/
void setup_mem( void );
void setup_GPIO( void );
void setup_timing( void );
void setup_periph( void );
void toggle_LED( void );
void set_DutyCycle(unsigned int);
void run_timers( void );
void run_periph( void );
void printbyte( unsigned char );
void prints(char *);
void enableATD( void );
void disableATD( void );

/*ReadSPI.Read byte/word from SPIBUF register */
unsigned int ReadSPI2();

/*WriteSPI. Write byte/word to SPIBUF register */
void WriteSPI2( unsigned int data_out);

#endif
```

setup.c

```
/*
    setup.c
    Author: Karl Herb
    This code is derived from modules, such as timer.h, adc12.h, etc.
    included with the compiler.  Definitions have literally been copied
    and pasted and used for our setup.  This way 1) the amount of header
    files is greatly reduced and 2) only what is necessary is contained
    herein.
    This setups the timers, ATD, PWM, serial (only for debug), SPI
    (for the GSM), and GPIO (except for the LCD).
*/
#include "setup.h"
#include "LCD.h"

/*internal definitons for ATD setup*/
#define  SAMP_BUFF_SIZE                      8
#define  NUM_CHS2SCAN                        1

/*debug functions*/
void printhexdigit(unsigned char digit);

/*global for GPIO with the LED*/
char global_curr;

/*internal inline functions*/
inline void setup_PLL()
{
    /*pg 153, fig 8-2 in osc doc, also have divide by 4 factor (not shown)
    exp. shows a IF of 66.6MHz so a clock freq of 33.3MHz*/
    _PLLPRE  = 2 ;
    _PLLDIV  = 150;
    _PLLPOST = 2 ;
}
/*timing (periods and op factors)*/
inline void open_timers()
{
    TMR1  = 0;
    PR1   = T1_PERIOD;
    T1CON = T1_SETUP;
    TMR2  = 0;
    PR2   = T2_PERIOD;
    T2CON = T2_SETUP;
    TMR3  = 0;
    PR3   = T3_PERIOD;
    T3CON = T3_SETUP;
    TMR4  = 0;
    PR4   = T4_PERIOD;
    T4CON = T4_SETUP;
}
/*interrupt setup*/
inline void config_timers()
{
    IFS0bits.T1IF = 0;
    IPC0bits.T1IP = (T1_CONFIG &0x0007);
    IEC0bits.T1IE = (T1_CONFIG &0x0008)>>3;
    IFS0bits.T2IF = 0;
```

```
    IPC1bits.T2IP = (T2_CONFIG &0x0007);
    IEC0bits.T2IE = (T2_CONFIG &0x0008)>>3;
    IFS0bits.T3IF = 0;
    IPC2bits.T3IP = (T3_CONFIG &0x0007);
    IEC0bits.T3IE = (T3_CONFIG &0x0008)>>3;
    IFS1bits.T4IF = 0;
    IPC6bits.T4IP = (T4_CONFIG &0x0007);
    IEC1bits.T4IE = (T4_CONFIG &0x0008)>>3;
}
/*PWM aka. output compare*/
inline void setup_PWM( void )
{
  OC1CONbits.OCM = 0;
  OC1RS = 0;
  OC1R = 0x03FF;
  OC1CON = PWM_SETUP;
}
/*ATD setup, adapted from FAQ site on Microchip*/
inline void setup_ATD( void )
{
    AD1CON1bits.FORM   = 0;
    /*Sample Clock Source: GP Timer 3 starts conversion*/
    AD1CON1bits.SSRC   = 2;
    /*ADC Sample Control: Sampling begins immediately after conversion */
    AD1CON1bits.ASAM   = 1;
    /*10 bit ADC operation*/
    AD1CON1bits.AD12B  = 0;
    /*Scan Input Selections for CH0+ during Sample A bit, this must
    be on, makes no sense as only converting CH0, but o well*/
    AD1CON2bits.CSCNA = 1;
    AD1CON2bits.CHPS  = 0;
    /*ADC Clock is derived from Systems Clock  */
    /*with a fast sample time (Tad = Tcy*(ADCS+1) = (1/40M)*64 = 1.6us)*/
    AD1CON3bits.ADRC = 0;
    AD1CON3bits.ADCS = 63;
    /*setup scanning and _AN29 is only audio input*/
    AD1CON2bits.SMPI    = (NUM_CHS2SCAN-1);
    AD1CSSHbits.CSS29 = 1;
    AD1CSSL = 0x0000;
    AD1PCFGL=0xFFFF;
    AD1PCFGH=0xFFFF;
    AD1PCFGHbits.PCFG29 = 0;
    return;
}
/*ATD interrupt*/
inline void config_ATD( void )
{
    /*Clear the A/D interrupt flag bit*/
    IFS0bits.AD1IF = 0;
    /*Enable A/D interrupt*/
    IEC0bits.AD1IE = 0;
    /*Turn on the A/D converter*/
    AD1CON1bits.ADON = 0;
}

/*externally visible functions*/
void setup_mem( void )
{
```

```
    /*set up global vars*/
    global_curr = 0;
    return;
}

void setup_GPIO( void )
{
    /*set up RA4 and RD2 (LED and pushbutton)*/
    TRISAbits.TRISA4 = 0;
    LATAbits.LATA4  = 0;
    PORTAbits.RA4 = 0;

    /*Set up push buttons as inputs*/
    /* We don't have to explicitly do this because the
    TRIS registers default to 1 (input) in a reset, but we
    will do it anyway as a good programming practice */
    TRISDbits.TRISD1 = 1;
    TRISDbits.TRISD2 = 1;

    /*Set-up output to GSM pins*/
    TRISBbits.TRISB12 = 0; /*Set up GSM On/Off pin (RB12) as an output */
    PORTBbits.RB12 = 1;
    return;
}

void setup_timing( void )
{
    setup_PLL();
    open_timers();
    config_timers();
    return;
}

void setup_periph( void )
{
         setup_PWM();
         setup_ATD();
         /* Set Up UART */
         U1BRG=216;      /* 9600Baud for 16MIP (See FRM Tables)       */
                   /* Change U2BRG to suit your clock frequency */
         U1MODE=0x8000;  /* Enable, 8data, no parity, 1 stop    */
         U1STA =0x8400;  /* Enable TX                           */

         /*set SPI port to slowest setting
         master mode
         8 bit
         Idle state for Clock is high level
         Primary prescaler 64:1
         Secondary prescaler 1:1*/
         /*must be in this order*/
         SPI2BUF = 0;
         SPI2CON1 = 0x00C0;
         SPI2CON2 = 0x0000;
         SPI2STAT = 0x8000; /*enable SPI port*/
}

void toggleLED( void )
{
```

```
    global_curr = !global_curr;
    _RA4 = global_curr;
    return;
}

void set_DutyCycle(unsigned int dutycycle)
{   /*taken from header file of compiler*/
    /*check if OC is in PWM Mode*/
    if((OC1CONbits.OCM & 0x06) == 0x06)
        OC1RS = dutycycle;   /* assign to OCRS */
}

/*self explanatory*/
void enableATD( void )
{
    _AD1IF = 0;
    EnableIntADC1;
    AD1CON1bits.ADON = 1;
    return;
}

/*self explanatory*/
void disableATD( void )
{
    _AD1IF = 0;
    DisableIntADC1;
    AD1CON1bits.ADON = 0;
    return;
}


/*debug stuff (SCI)*/
void printbyte(unsigned char val)
{
    unsigned char nibble;
    unsigned char res;
    /*build rep of high byte and send*/
    nibble = val / 16;
    printhexdigit(nibble);
    nibble = val % 16;
    printhexdigit(nibble);
    return;
}
/*makes the non-ASCII digit into a hex digit
and prints it*/
void printhexdigit(unsigned char digit)
{
    if(digit >= 0x0A && digit <= 0x0F)
    {
        /*have A - F*/
        digit = digit + 0x37;
    }
    else if(digit <= 0x09)
    {
        /*have 0 - 9*/
        digit = digit + 0x30;
    }
    else
```

```
    {
        digit = 0x30;
    }
    LCD_put_char(digit);

  return;
}
void prints(char *msg)
{
         char *sptr;
         sptr=msg;
         while(*sptr != '\0')
         {
                  while(U1STAbits.UTXBF==1); /* Wait TX buf read for new data */
                  U1TXREG=*sptr;
                  sptr++;
         }
         return;
}

/*SPI2 for GSM*/
unsigned int ReadSPI2()
{
    /* Check for Receive buffer full status bit of status register*/
    if (SPI2STATbits.SPIRBF)
    {
        SPI2STATbits.SPIROV = 0;

        if (SPI2CON1bits.MODE16)
            return ( SPI2BUF );        /* return word read */
        else
            return (SPI2BUF & 0xff);   /* return byte read */
    }
         return -1;                      /* RBF bit is not set return error*/
}

void WriteSPI2(unsigned int data_out)
{
    if (SPI2CON1bits.MODE16)                  /* word write */
        SPI2BUF = data_out;
    else
        SPI2BUF = data_out & 0xff;   /*  byte write  */
}

/* end setup.c */
```

SD.h

```
/*
    SD.h
    Author: Karl Herb
    This code sets up the SD card for I/O and reads and writes blocks
    of 512 bytes to and from the card using an SPI interface.  This
    contains the external function definitions.
*/
#include <p33FJ256GP710.h>

/*legacy defines*/
#define OK 0x00
#define PRINT_ERR 0x01
#define INIT_ERR 0x02
#define SD_ERR 0x03

/*define the block size as 512, the standard for SD card interface*/
#define SD_BLOCK_SIZE 512

/*external functions for setup, reseting to the
beginning of the card, and I/O*/
unsigned char setup_SDSPI( void );
void SD_setStart( void );
unsigned char SD_GetSample( int );
void SD_SetSample( int, unsigned char );
void SD_writeCurr( void );
void SD_readCurr( void );
```

SD.c- built around functions generated by MpAMModules (MidiMaestro)

```
/*
    SD.c
    Author: Karl Herb
    SD vs. MMC card bug fix: Justin Lanning
    This code sets up the SD card for I/O and reads and writes blocks
    of 512 bytes to and from the card using an SPI interface.  This is
    developed from the MidiMaestro code and uses much of their original
    design and defaults.
    NOTE: legacy = defined in original, and has not been kept uptodate
    and therefore the code or variable has fallen out of importance
*/
#include "SD.h"
/*need this to call appropriate functions for debug (SCI)*/
#include "setup.h"
/*include LCD.h so can interrupt when SD card removed */
#include "LCD.h"

/*internal macros, some legacy*/
#define HEX 16
#define READ_CMD 17
#define DUMMY 0xFF
#define START_BLOCK_TOKEN 0xFE
/*R1 Response Codes (from SD Card Product Manual v1.9 section 5.2.3.1)*/
#define R1_IN_IDLE_STATE    (1<<0)   /*The card is in idle state
                                     and running initializing process.*/
#define R1_ERASE_RESET      (1<<1)   /*An erase sequence was cleared
                                     before executing because of an
                                     out of erase sequence
                                     command was received.*/
#define R1_ILLEGAL_COMMAND  (1<<2)  /*An illegal command code detected*/
#define R1_COM_CRC_ERROR    (1<<3)  /*The CRC check of the
                                     last command failed.*/
#define R1_ERASE_SEQ_ERROR  (1<<4)  /*An error in the sequence of
                                     erase commands occured.*/
#define R1_ADDRESS_ERROR    (1<<5)  /*A misaligned address,
                                     which did not match the block length
                                     was used in the command.*/
#define R1_PARAMETER        (1<<6)  /*The command's argument (e.g.
                                     address, block length) was out of
                                     the allowed range for this card.*/

/*ports redefined for descriptive use*/
#define SD_CS PORTCbits.RC4
#define SD_CS_DIR TRISCbits.TRISC4
#define SDI PORTFbits.RF7
#define SDI_DIR TRISFbits.TRISF7
#define SCK PORTFbits.RF6
#define SCK_DIR TRISFbits.TRISF6
#define SDO PORTFbits.RF8
#define SDO_DIR TRISFbits.TRISF8
#define SD_Enable() SD_CS = 0
#define SD_Disable() SD_CS = 1

/*internal types and definitions
*/
typedef unsigned char buffer_typ;
```

```
/*local buffer for the current sample*/
buffer_typ buffer[SD_BLOCK_SIZE];
/*global pointers for blocks*/
unsigned long curr_block;
unsigned long total_blocks;
/*legacy flag*/
unsigned char synced;

/*internal function declarations*/
unsigned char SDReadBlock( unsigned long );
unsigned char SDWriteBlock(unsigned long );
void InitSPI( void );
unsigned char InitSD( void );
unsigned char SD_WriteCommand(unsigned char* cmd);
unsigned char SPIRead( void );
void SPIWrite(unsigned char data);

/*functions definitions*/
unsigned char setup_SDSPI( void )
{
    unsigned char res = 0;
    unsigned long start_block = 0xD6D6;

    InitSPI();
    res = InitSD();
    do
    {
        while(res)
        {
            res = InitSD();
        }
        res = SDReadBlock(start_block);
    } while(res);

    /*internal for block pointing to at the time, set to 1 as
    future implementations may have block 0 contain information
    (metadata) about the samples*/
    curr_block = 1;
    synced = 0;

    return OK;
}

/*reset the SD card to beginning, see above why it is set to 1*/
void SD_setStart( void )
{
    curr_block = 1;
    return;
}

/*obtain the sample in the local array at the value specified
by count (like an object's data encapsulation function)*/
unsigned char SD_GetSample( int cnt )
{
    return buffer[cnt];
}

/*similar to above, just adds a single value to the internal buffer*/
```

```
void SD_SetSample( int cnt, unsigned char sample )
{
    buffer[cnt] = sample;
    return;
}

/*signal to write the local buffer to the SD card at the block
pointed to by the internal pointer*/

void SD_writeCurr( void )
{
    SDWriteBlock(curr_block);
    curr_block++;
    return;
}

/*signal to read the current pointer's block to the local array*/
void SD_readCurr( void )
{
    SDReadBlock(curr_block);
    curr_block++;
    return;
}

/*read a block of 512 bytes (SD_BLOCK_SIZE) from the SD card*/
unsigned char SDReadBlock(unsigned long block)
{
    buffer_typ* theData;
    unsigned char read_cmd[6];
    unsigned char status = 0x0;
    unsigned int offset = 0;
    unsigned char res = 1;

    /*best to do a quick init in case anything is off*/
    while(res)
    {
        res = InitSD();
    }

    /*send the read command and the block*/
    block = block * SD_BLOCK_SIZE; /*need to be correct offset*/
    read_cmd[0] = READ_CMD;
    read_cmd[1] = ((block & 0xFF000000) >> 24);
    read_cmd[2] = ((block & 0x00FF0000) >> 16);
    read_cmd[3] = ((block & 0x0000FF00) >> 8 );
    read_cmd[4] = ((block & 0x000000FF)      );
    read_cmd[5] = DUMMY;
    SD_Enable();
    status = SD_WriteCommand(read_cmd);
    if(status != 0)
    {
        return SD_ERR;
    }

    /*find the start of the read*/
    do
    {
        status = SPIRead();
```

```
    }while(status != START_BLOCK_TOKEN);

    /*read the bytes*/
    theData = buffer;
    for(offset = 0; offset < SD_BLOCK_SIZE; offset++)
    {
        *theData = SPIRead();
        theData++;
    }
    SD_Disable();

    /*pump for eight cycles according to spec*/
    SPIWrite(0xFF);

    return OK;
}

/*write a block of 512 bytes (SD_BLOCK_SIZE) to the SD card*/
unsigned char SDWriteBlock(unsigned long block)
{
    buffer_typ* theData;
    unsigned int i;
    unsigned char status;
    unsigned char res = 1;

    while(res)
    {
        res = InitSD();
    }

    /*setup the write command and block*/
    unsigned char CMD24_WRITE_SINGLE_BLOCK[] = {24,0x00,0x00,0x00,0x00,0xFF};
    block = block * SD_BLOCK_SIZE; /*need to be correct offset*/
    CMD24_WRITE_SINGLE_BLOCK[1] = ((block & 0xFF000000) >> 24);
    CMD24_WRITE_SINGLE_BLOCK[2] = ((block & 0x00FF0000) >> 16);
    CMD24_WRITE_SINGLE_BLOCK[3] = ((block & 0x0000FF00) >> 8);
    CMD24_WRITE_SINGLE_BLOCK[4] = ((block & 0x000000FF));

    SD_Enable();

    /*Send the write command*/
    status = SD_WriteCommand(CMD24_WRITE_SINGLE_BLOCK);
    if(status != 0)
    {
         return 1;
    }

    /*write data start token*/
    SPIWrite(0xFE);

    /*write all the bytes in the block*/
    theData = buffer;
    for(i = 0; i < SD_BLOCK_SIZE; ++i)
    {
         SPIWrite(*theData);
         theData++;
    }
```

```
    /*Write CRC bytes*/
    SPIWrite(0xFF);
    SPIWrite(0xFF);

    /*wait to complete*/
    status = SPIRead();
    while(status != 0xFF)
    {
         status = SPIRead();
    }

    SD_Disable();

    /*wait 8 clock cycles*/
    SPIWrite(0xFF);

    return(0);

}

/*internal function definitions */
/*this taken from MidiMaestro directly, not augmented much*/
unsigned char InitSD( void )
{
     unsigned int i = 0;
     unsigned char status;

     SD_Disable();

     // Wait for power to really go down
     for(i = 0; i; i++);
     for(i = 0; i; i++);
     for(i = 0; i; i++);
     for(i = 0; i; i++);

     // Turn on SD Card
     //SD_PowerOn();

     // Wait for power to really come up
     for(status = 0; status < 10; ++status)
     {
        for(i = 0; i; i++);
        for(i = 0; i; i++);
        for(i = 0; i; i++);
        for(i = 0; i; i++);
     }

     //We need to give SD Card about a hundred clock cycles to boot up
     for(i = 0; i < 16; ++i)
     {
       SPIWrite(0xFF); // write dummy data to pump clock signal line
     }

     SD_Enable();

     //This is the only command required to have a valid CRC
     // After this command, CRC values are ignore
     //unless explicitly enabled using CMD59
```

```
    unsigned char CMD0_GO_IDLE_STATE[] = {0x40,0x00,0x00,0x00,0x00,0x95};

    // Wait for the SD Card to go into IDLE state
    i = 0;

    do
    {
      status = SD_WriteCommand(CMD0_GO_IDLE_STATE);

        // fail and return
        if(i++ > 50)
        {
                return 1;
        }
    } while( status != 0x01 );

    // Wait for SD Card to initialize
    unsigned char CMD1_SEND_OP_COND[] = {0x01,0x00,0x00,0x00,0x00,0xFF};

    i = 0;
    do
    {
        status = SD_WriteCommand(CMD1_SEND_OP_COND);
        if(i++ > 50)
        {
                return 2;
        }
    } while( (status & R1_IN_IDLE_STATE) != 0 );

    // Send CMD55, required to precede all "application specific" commands
    unsigned char CMD55_APP_CMD[] = {55,0x00,0x00,0x00,0x00,0xFF};
    status = SD_WriteCommand(CMD55_APP_CMD); // Do not check response here

    // Send the ACMD41 command to
    //initialize SD Card mode (not supported by MMC cards)
    i = 0;
    unsigned char ACMD41_SD_SEND_OP_COND[] = {41,0x00,0x00,0x00,0x00,0xFF};
    do
    {
      status = SD_WriteCommand(ACMD41_SD_SEND_OP_COND);
      // Might return 0x04 for Invalid Command if MMC card is connected
        if(i++ > 50)
        {
                return 3;
        }
    } while( (status & R1_IN_IDLE_STATE) != 0 );

    // Set the SPI bus to full speed now that
    //SD Card is initialized in SPI mode
    SD_Disable();

    return 0;
}

/*this taken from MidiMaestro directly, not augmented much*/
void InitSPI(void)
{
         //SD_PowerOff();
```

```
          //SD_PWR_DIR = 0;   // output
          //SD_PowerOff();

          SD_Disable();
          SD_CS_DIR = 0; // output
          SD_Disable();

          SDI_DIR = 1; // input
          SCK_DIR = 1;
          SDO_DIR = 1;

          // set SPI port to slowest setting
          // master mode
          // 8 bit
          // Idle state for Clock is high level
          // Primary prescaler 64:1
          // Secondary prescaler 1:1
          SPI1CON1 = 0x007C;
        SPI1CON2 = 0x0000;
          SPI1STAT = 0x8000; // enable SPI port
}

/*this taken from MidiMaestro directly, not augmented much*/
unsigned char SD_WriteCommand(unsigned char* cmd)
{
          unsigned int i;
          unsigned char response;
          unsigned char savedSD_CS = SD_CS;

          // SD Card Command Format
          // (from Section 5.2.1 of SanDisk SD Card Product Manual v1.9).
          // Frame 7 = 0
          // Frame 6 = 1
          // Command (6 bits)
          // Address (32 bits)
          // Frame 0 = 1

          // Set the framing bits correctly (never change)
          cmd[0] |= (1<<6);
          cmd[0] &= ~(1<<7);
          cmd[5] |= (1<<0);

          // Send the 6 byte command
          SD_Enable();
          for(i = 0; i < 6; ++i)
          {
                    SPIWrite(*cmd);
                    cmd++;
          }

          // Wait for the response
          i = 0;
          do
          {
                    response = SPIRead();

                    if(i > 60000)  //instead of 100
                    {
```

```
                        break;
                }
                i++;
        } while(response == 0xFF);

        SD_Disable();

        //Following any command, the SD Card needs 8 clocks
      //to finish up its work.
        //(from SanDisk SD Card Product Manual v1.9 section 5.1.8)
        SPIWrite(0xFF);

        SD_CS = savedSD_CS;
        return(response);
}

/*this taken from MidiMaestro directly, not augmented much*/
void SPIWrite(unsigned char data)
{
        // DO NOT WAIT FOR SPITBF TO BE CLEAR HERE
        // (for some reason, it doesn't work on this side of the write data).

        // Write the data!
        SPI1BUF = data;
        // Wait until send buffer is ready for more data.
        while(SPI1STATbits.SPITBF);
}

/*this taken from MidiMaestro directly, not augmented much*/
unsigned char SPIRead(void)
{
        unsigned char data;

        if(SPI1STATbits.SPIRBF)
        {
                //already have some data to return, don't initiate a read
                data = SPI1BUF;

                SPI1STATbits.SPIROV = 0;
                return data;
        }

        // We don't have any data to read yet, so initiate a read
        SPI1BUF = 0xFF;  // write dummy data to initiate an SPI read
        while(SPI1STATbits.SPITBF); // wait until the data is finished reading
        data = SPI1BUF;

        SPI1STATbits.SPIROV = 0;
        return data;
}

/* end SD.c */
```

LCD.h

```
/*
    LCD.h
    Authors: Brad Sokola, Justin Lanning
    This contains the definitions and interface necessary for working
    with the LCD.  The necessary cmds are hardcoded into main.c.  Look
    at it for information.
*/
#include <p33FJ256GP710.h>

/* internal defines*/
#define CLEAR_LCD                   0b00000001
#define RETURN_HOME_LCD             0b00000010

/*externally visible functions */
void LCD_init(void);
void LCD_put_char(char ch);
void LCD_cmd(char ch);
void LCD_reset_time(void);
void LCD_update(void);
void LCD_set_src(char src);
void LCD_put_string(char *string);
void LCD_set_mic(char mic);
void LCD_reset( void );
```

LCD.c

```
/*
    LCD.c
    Authors: Brad Sokola, Justin Lanning
    This contains the definitions and interface necessary for working
    with the LCD.  It outputs characters and has a function for updating
    the runtime of the device from the seconds to the days.
*/
#include "LCD.h"

/* internal variables, for up-time count */
char global_sec;
char global_ten_sec;
char global_min;
char global_ten_min;
char global_hr;
char global_ten_hr;
char global_day;

void LCD_init(void)
{
         int i;
         int b;
         for(i=0;i<10;i++)
         {
                  for(b=0;b<10000;b++);       /*Wait for the LCD to come up*/
         }

         /*clearing screen*/
         _LATG12 = 0;
         _LATG13 = 0;
         _LATG14 = 0;
         _LATD4 = 0;
         _LATD5 = 0;
         _LATD6 = 0;
         _LATD7 = 0;
         _LATD8 = 0;
         _LATD9 = 0;
         _LATD10 = 0;
         _LATD11 = 0;
         _TRISG12 = 0;      /*RW pin - Just leave this 0 the whole time*/
         _TRISG14 = 0;      /*RS pin*/
         _TRISG13 = 0;      /*EN pin*/

         /*Write command to data port 0b00111000*/
         LCD_cmd(0b00111000);        /*Sets to 8 bit operation
                             and 2Line 5x8 dot char font*/
         /*Write command to data port (0b00001110);*/
         LCD_cmd(0b00001100);        /*Turns on display and cursor off,
                             blinking off*/
         /*Write command to data port (0b00000110);*/
         LCD_cmd(0b00000110);        /*Sets cursor mode direction
                             and display shift*/
         LCD_cmd(CLEAR_LCD);
         for(i=0;i<15000;i++);   /*wait...*/

         return;
```

```
}


void LCD_reset( void )
{
         LCD_reset_time();
         LCD_cmd(0x80); /*sets cursor at head of first line*/
         LCD_put_string("Stat> W Rec Off");
         LCD_cmd(0xC0); /*sets cursor at head of second line*/
         LCD_put_string("Up-> 00:00:00:00");
         return;
}

void LCD_put_char(char ch)
{
         int i;
         _TRISD11 = (ch&0b10000000) >> 7;
         _TRISD10 = (ch&0b01000000) >> 6;
         _TRISD9 =  (ch&0b00100000) >> 5;
         _TRISD8 =  (ch&0b00010000) >> 4;
         _TRISD7 =  (ch&0b00001000) >> 3;
         _TRISD6 =  (ch&0b00000100) >> 2;
         _TRISD5 =  (ch&0b00000010) >> 1;
         _TRISD4 =   ch&0b00000001;
         _TRISG14 = 1;      /*RS pin high*/
         /*Clock the cmd in*/
         _TRISG13 = 1;      /*EN pin - Clock high*/
         for(i=0;i<125;i++);
         _TRISG13 = 0;      /*EN pin - Clock low*/
         _TRISG14 = 0;      /*RS pin low*/
         for(i=0;i<125;i++);
         return;
}

void LCD_cmd(char ch)
{
         int i;
         /*All control bits to 0*/
         _TRISG12 = 0;      /*RW pin - Just leave this 0 the whole time*/
         _TRISG14 = 0;      /*RS pin*/
         _TRISG13 = 0;      /*EN pin*/
         /*Command*/
         _TRISD11 = (ch&0b10000000) >> 7;
         _TRISD10 = (ch&0b01000000) >> 6;
         _TRISD9 =  (ch&0b00100000) >> 5;
         _TRISD8 =  (ch&0b00010000) >> 4;
         _TRISD7 =  (ch&0b00001000) >> 3;
         _TRISD6 =  (ch&0b00000100) >> 2;
         _TRISD5 =  (ch&0b00000010) >> 1;
         _TRISD4 =   ch&0b00000001;
         /*Clock the cmd in
         _TRISG13 = 1;      /*EN pin - Clock high*/
         for(i=0;i<125;i++);
         _TRISG13 = 0;      /*EN pin - Clock low*/
         for(i=0;i<125;i++);
         return;
}
```

```
void LCD_reset_time(void)
{
          global_sec = 0;
          global_ten_sec = 0;
          global_min = 0;
          global_ten_min = 0;
          global_hr = 0;
          global_ten_hr = 0;
          global_day = 0;
          return;
}

void LCD_set_src(char src)
{
          /*Update Recording Status*/
          LCD_cmd(0x88); /*sets cursor at Rec/Playback place*/
          if(src == 'P'){
                    /*Playback mode*/
                    LCD_put_string("Ply");
          }
          else if (src == 'R'){
                    /*Recording mode*/
                    LCD_put_string("Rec");
          }
          else if (src == 'L'){
                    /*Listening but not recording*/
                    LCD_put_string("Lis");
          }
          else {
                    /*Off*/
                    LCD_put_string("Off");
          }
          return;
}

void LCD_set_mic(char mic)
{
          /*Update Microphone Status*/
          LCD_cmd(0x86); /*sets cursor at Microphone place*/
          LCD_put_char(mic);
          return;
}

void LCD_gsm_stat(char gsm)
{
          /*Update Microphone Status*/
          LCD_cmd(0x8C); /*sets cursor at GSM place*/

          switch(gsm)
          {
          case 0: LCD_put_string("NoS");
                    break;
          case 1: LCD_put_string("Con");
                    break;
          case 2: LCD_put_string("Sch");
                    break;
          case 3: LCD_put_string("NoS");
                    break;
```

```
        case 4: LCD_put_string("Ukn");
                break;
        case 5: LCD_put_string("Rom");
                break;
        default: LCD_put_string("Off");
                break;
        }

        return;
}

void LCD_update(void)
{
        global_sec++;
        if(global_sec == 10){
                global_sec = 0;
                global_ten_sec++;
                if(global_ten_sec == 6){
                        global_ten_sec = 0;
                        global_min++;
                        if(global_min == 10){
                                global_min = 0;
                                global_ten_min++;
                                if(global_ten_min == 6){
                                        global_ten_min = 0;
                                        global_hr++;
                                        if(global_hr == 10){
                                                global_hr = 0;
                                                global_ten_hr++;
                                                if(global_ten_hr == 24){
                                                        global_ten_hr = 0;
                                                        global_day++;
                                                        LCD_cmd(0xC6);
                                                LCD_put_char(global_day+48);
                                                }
                                        LCD_cmd(0xC8); /*sets cursor
                                        at ten hour place*/
                                        LCD_put_char(global_ten_hr+48);
                                        }
                                LCD_cmd(0xC9); /*sets cursor at hour place*/
                                LCD_put_char(global_hr+48);
                                }
                        LCD_cmd(0xCB); /*sets cursor at ten minutes place*/
                        LCD_put_char(global_ten_min+48);
                        }
                        LCD_cmd(0xCC); /*sets cursor at minutes place*/
                LCD_put_char(global_min+48);
                }
        LCD_cmd(0xCE); /*sets cursor at ten seconds place*/
        LCD_put_char(global_ten_sec+48);
        }
        LCD_cmd(0xCF); /*sets cursor at seconds place*/
        LCD_put_char(global_sec+48);
        return;
}


void LCD_put_string(char *string)
```

```
{
    while(*string)/*Write data to LCD up to null*/
    {
        LCD_put_char(*string);/*Write character to LCD*/
        string++;           /*Increment buffer*/
    }
    return;
}

/* end LCD.c */
```

p33FJ256GP710.h- not listed as obtained from Microchip's compiler

Appendix G: FMECA Worksheet

Failure No.	Failure Mode	Possible Causes	Failure Effects	Method of Detection	Criticality	Remarks
A1	Inability to read / write to SD card	Software, corrupt SD, micro, or poor connection to headers.	Speech will not be recorded.	Check SD with an external reader	Medium	Medium criticality is given due to the recording of audio being the device's main function.
A2	Micro failure	Software, shorted bypass capacitors causing micro to operate below 3.3V.	Unpredictable behavior and loss of functionality (partial or complete).	Observation	Low - Medium	Low - Medium criticality given because of variable functionality loss.

Table G-1 - Microcontroller and Off-Board Peripherals (Block A) FEMCA

Failure No.	Failure Mode	Possible Causes	Failure Effects	Method of Detection	Criticality	Remarks
B1	No audio output	PWM software error, shorting of LPF capacitor.	Inability to use PWM output for debugging purposes.	Observation	Medium	Medium criticality is given because this feature allows playback of stored audio.
B2	Push button malfunction	MAX6817 no longer de-bounces switch, or mechanical failure in push button, software.	No longer able to turn on/off recording, or alter settings	Observation	Low	Low criticality because no vital components will be damaged and recording can still occur.
B3	No audio input	MAX9812L damage, or passive component breakdown / short	Inability to sample audio from on-board mic.	Observation	Medium	Medium criticality is given because this feature is critical to primary device function.
B4	LCD display malfunction (blank or incorrect)	Software, corrupt LCD, micro, or poor connection to headers, bad SIP	Current status of device will not be displayed.	Observation	Low	Low criticality is awarded because none of main functionality is lost.

Table G-2 - Audio and GPIO (Block B) FEMCA

Failure No.	Failure Mode	Possible Causes	Failure Effects	Method of Detection	Criticality	Remarks
C1	No Wi-Fi access	Software, internal Wiport damage, bypass capacitor short, or pin connection error.	User loses ability to access embedded web server.	Observation (trying to log into the device)	Low	Low criticality because device does not lose recording functionality
C2	No Bluetooth audio streaming	Software, internal BSIM2 damage, bypass capacitor short, or pin connection error.	Failure to record audio.	Observation (scope PCM output)	Medium	Medium criticality because devices loses ability to record audio.
C3	No GSM access	Software, internal GM862 damage, bypass capacitor short, or pin connection error.	Inability to call into the device and access live audio.	Observation	Low	Low criticality because user can find other ways to access audio.
C4	GSM will not turn on/off or reset	Software, internal damage, external BJT damage.	May not be able to call, unnecessary power drain.	Observation	Low	Low criticality because user can find other ways to access audio.

Table G-3 - On-Board "Wireless" Peripherals (Block C) FEMCA

Failure No.	Failure Mode	Possible Causes	Failure Effects	Method of Detection	Criticality	Remarks
D1	V_{OUT} = 0 V	LT1528/ LT1121 is damaged, short of peripheral capacitor	3.3 V - Device failure to function 3.8 V- GSM and Bluetooth failure 5 V - LCD	Observation, measurement of voltage	Medium	Medium criticality because device loses all functionality
D2	V_{OUT} > 3.3 V, 5 V, or 3.8 V	LT1528/ LT1121 is damaged, passive component failure (resistors)	Damage to major components (i.e. micro, GSM, Wiport, Bluetooth, LCD)	Observation, excess heat	High	High criticality because of its potential to cause harm to user
D3	V_{OUT} does not meet tolerance values for normal operation	LT1528/ LT1121 is damaged, passive component failure (resistors)	Unpredictable behavior from MCU and other powered components	Observation	Medium	Medium criticality due to potential loss of all device functionality

Table G-4 - Power Management (Block D) FEMCA