Mr. Lakshman One
School of Engineering Science
Simon Fraser University
Burnaby, British Columbia
V5A 1S6

March 8, 2007

Re: ENSC 440 Design Specifications for a Handheld Spectrum Analyzer

Dear Mr. One,

The attached document, *Handheld Spectrum Analyzer Design Specifications*, outlines SonoSense Technologies Inc.'s design specifications for the ENSC 305/440 project.

We are currently developing a handheld device that analyzes various sound inputs from the environment within the human hearing range including frequencies between 20 Hz and 20 kHz. The results of this frequency analysis are displayed visually in terms of certain frequency intervals and their respective power spectrums on the display unit of our Handheld Spectrum Analyzer. Our device also features an interactive menu with a display similar to common audio equalizers, allowing users to observe, save and compare various frequency spectrums.

The purpose of the enclosed document is to outline the design details and test plans for our prototype Handheld Spectrum Analyzer, also denoted as HSA. This document highlights all the design specifications that will be used to complete the proof-of-concept prototype before the demonstration deadline in late April.

SonoSense Technologies Inc. is compromised of four motivated and innovative fourth year engineering students: Sanaz Jahanbakhsh, Johnny Pak, Naureen Sikder, and Kenneth Wong. Should you have any questions or concerns about our functional specifications, please feel free to contact me by phone at (604) 722-0473 or our group e-mail address at ensc-440-project@sfu.ca.

Sincerely,

Sanaz Jahanbakhsh

Sanaz Jahanbakhsh
President and CEO
SonoSense Technologies Inc.

Enclosure: *Handheld Spectrum Analyzer Design Specification*

Handheld Spectrum Analyzer Design Specifications

SonoSense Technologies Inc.

Team: Sanaz Jahanbakhsh
Johnny Pak
Naureen Sikder
Kenneth Wong

Contact: Sanaz Jahanbakhsh
(604) 722-0473
ensc-440-project@sfu.ca

Submitted to: Lucky One
Steve Whitmore
Engineering Science
Simon Fraser University

Revision Date: March 8, 2007

Revision Number: 1.5

Executive Summary

Accurate frequency analysis has primarily been a task meant only for extremely high-tech engineering apparatus. These devices are extremely precise, but are bought at exuberant prices. By giving a mid-cost solution that is accurate enough for consumer use, the applications for these types of devices will expand to a much larger commercial market.

Our frequency analyzer accepts input from both a direct microphone and also a line-in input for data. This data is then processed and transformed into a frequency spectrum, which can then be displayed or analyzed for various purposes. Power levels of various frequencies are also determined by the data processing.

The Handheld Sound Analyzer used for frequency investigation is a portable solution for localized sound spectrum assessment. With the advancement of modern handheld development technology, which is extremely powerful and expandable, creation of this device allows access to this technology at a reasonable price to a large percentage of the commercial population. By incorporating this device design into handheld phones and computers, the HSA can expand this specific type of instrumental testing equipment.

The HSA will be built in a 3-division plan as well as a 2-stage development cycle. The division plan separates the product design into three categories: hardware requirements, software requirements, and integration requirements. The 2 stages of development are separated into an initial prototyping stage to produce an acceptable proof-of-concept and a production stage which involves the advancing of the initial prototype and creation of a marketable product form. Please note that the production stage mentioned in this document combines development phases 2 and 3 that was described in the Functional Specifications.

The primary features that will be available to our device through the first stage of development are:

1. Sound input and digital conversion of sound data.
2. The ability to process incoming data and transfer to the MX1 development board.
3. A bar display of analyzed frequency data onto an LCD display.

The second stage of development would entail more arduous specifications:

1. An encased design to hold the full device together.
2. Advanced GUI functionality and display features.
3. More options of sound data input other than direct microphone input.
4. An increased efficiency in data processing.

The primary stage of development has been planned for an expected completion data by April 25, 2007 and the second stage of development has an expected completion date of April 2008.

Table of Contents

List of Figures

List Tables

Glossary

The following abbreviations are found throughout the entirety of this document. If ever there are terms that may be specific to our product, the acronym or definition will be located here.

ADC	Analog Digital Converter
ENSC	Engineering Science
EVC++	Embedded Visual C++
FFT	Fast Fourier Transform
FFTW	Fast Fourier Transform in the West
GUI	General User Interface
HSA	Handheld Spectrum Analyzer
IDE	Integrated Development Environment
LCD	Liquid Crystal Display
MAPBGA	Mold Array Process-Ball Grid Array
MATLAB	Matrix Laboratory (Mathworks, Inc.)
MCU	Microcontroller Unit
MOSFET	Metal Oxide Silicon Field Effect Transistor
MSDN	Microsoft Developer Network
MSSP	Master Synchronous Serial Port
MX1	M9328MX1ADS/B Freescale Media Extension Development Board
OS	Operating System
PC	Personal Computer
PDA	Personal Digital Assistant
SDRAM	Synchronous Dynamic Random Access Memory
SDK	Software Development Kit
SPI	Serial Peripheral Interface
TFT	Thin Film Transistor
TTL	Transistor – Transistor Logic
TX	Transmission
RX	Receive
UART	Universal Asynchronous Receiver Transmitter
WinCE	Windows Compact Edition

Revision History

Revision	Date	Description	Name
0.1	Mar/04/07	Initial Compilation	Sanaz
0.2	Mar/06/07	All parts integrated	Johnny Pak
1.0	Mar/08/07	First Revision Rough	SonoSense Team
1.1	Mar/08/07	First Revision	Johnny Pak
1.2	Mar/08/07	Second Revision	Kenneth Wong
1.3	Mar/08/07	Third Revision	Sanaz Jahanbakhsh
1.4	Mar/08/07	Fourth Revision	Naureen Sikder
1.5	Mar/08/07	Fifth Revision	SonoSense Team

1 Introduction

The SonoSense Technologies Inc. Handheld Spectrum Analyzer displays the spectrum of received sound data and on its user interface. The main functionality of the Handheld Spectrum Analyzer is the presentation of sound wave power levels at various frequencies, ranging from 20Hz to 20 kHz, intended for audio equipment tuning and statistical purposes. The project's proof-of-concept prototype will be ready for demonstration by late April; for more detailed information regarding the prototype's development cycles please refer to *Handheld Spectrum Analyzer Functional Specification* document [1].

1.1 Scope

While most of the design details throughout the document outline the proof-of-concept prototype's primary features, other pre-production important subsystems like data manipulation, storage and user interface are also outlined, which will be implemented in the later stages of development.

1.2 Intended Audience

This document is primarily targeted towards developers and integration engineers as well as the quality assurance team. Once finalized, this document will serve as the set of guidelines for the SonoSense engineers when designing various subsystems of the device. Successful implementation of any given features will be verified by using the outlined test plan.
In addition, on-time completion of specific detailed features will be used as a monitor of the status and progress of the project.

2 System Requirements

To provide the reader with a better understanding of the device, the overall system's hardware and software are discussed. In addition, the device's subsystems as well as its input and output are explained. Please note that more detailed design specifications of each of these major components are included in the Hardware and Software sections.

2.1 System Overview

Figure 1 displays an overview of the Handheld Spectrum Analyzer outlining its input and output. Audible sound, frequencies ranging from 20Hz to 20 kHz, is considered the input source while the frequency spectrum displayed on the LCD is the system's output. Spectrum analysis results are realized in "real-time" and only sound sources within the audible range are processed. Upon user input, the display of sampling and refresh rates can be changed and captured spectrums can be stored onto the removable memory of the device for later comparisons on the device.

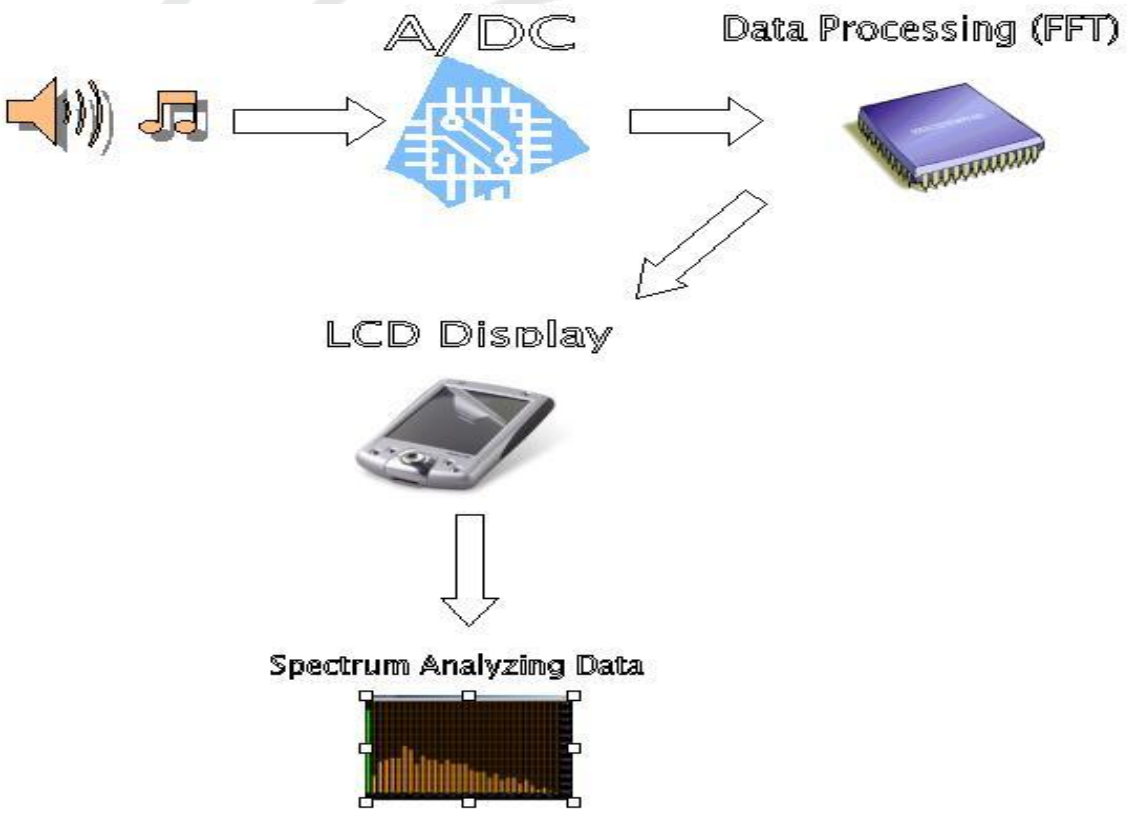

Figure 1: System Overview

2.2 Hardware Overview

Handheld Spectrum Analyzer's hardware consists of five major components: Input amplification circuitry, analog to digital converter (ADC), Microchip PIC microcontroller, serial driver and the main development board, Motorola M9328MX1ADS/B, as shown in Figure 2. Please note that throughout this document MX1 and Motorola M9328MX1ADS/B are used interchangeably.

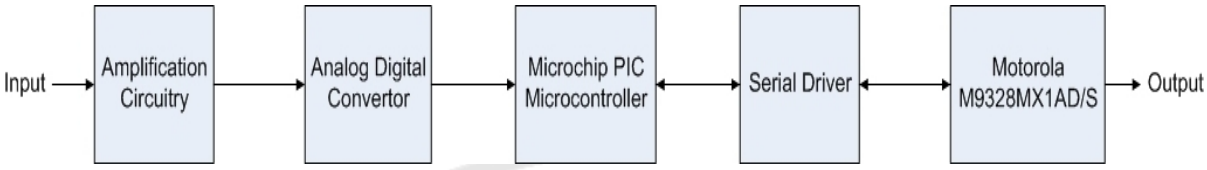

Figure 2: System's Hardware Overview

Input amplification circuitry consists of a high-precision microphone, and an amplifier constructed on a breadboard. The amplifier was designed to maximize the precision of collected input, which is passed onto the ADC. The criteria in selection of our ADC were the frequency sampling capabilities as well as its output precision for audio equipment. The output of the ADC is then passed onto the PIC microcontroller, which communicates with the main development board, Motorola M9328MX1ADS/B, via a serial driver. The spectrum of the processed data is then displayed on the development board's LCD panel. For more detailed information regarding the Motorola M9328MX1ADS/B development board please refer to the Technical Appendices [6].

2.3 Software Overview

The majority of the proof-of-concept's time budget is spent on developing the three main software components: PIC module, Serial module, and HSA application, which are shown in Figure 3.

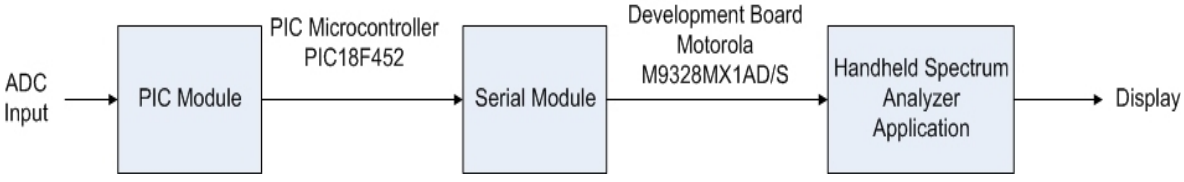

Figure 3: System's Software Overview

Due to the popularity of its user interface and availability of free application development tools such as Embedded Visual C++ (EVC++), Microsoft Windows CE 4.2, also denoted as WinCE, was chosen as the device's OS and development language. In order to speed up

and simplify the testing process, all the developed code is testing on both WinCE 4.2 Standard, as well as a Pocket PC 2003 SDK emulator.

The first two interfacing components, SPI and Serial modules, are designed to transfer data from the ADC to the PIC Microcontroller and then onto the main development board, the Motorola M9328MX1ADS/B. The HSA application consists of an FFT calculation module and the device's GUI designed to display the output and interact with the user. The SPI module is developed in C language on MicroChip's MPLAB IDE. The serial module connecting the PIC and the MX1 and the HSA application are developed using embedded Visual C++ 4.0.

3 System Hardware

There are various components used in the Sound Frequency Analyzer. The M9328MX1ADS/B development board by Freescale Semiconductors Inc. is the major component for data processing of incoming sound data. This development board has a DragonBall MCU, which uses a 200MHz ARM9 microprocessor core. Due to the strict requirements of the project timeline, development of the system prototype is contained to the successful transfer of analog data to the development board, and sending representative data visually to our display.

The Motorola M9328MX1ADS/B is a powerful and functional board with the capability to meet our design needs. It is also highly adaptive, thus allows for use in our specific application. Apart from this board, the other component cores used in the HSA are the microphone input and amplifier circuit, as well as the analog-to-digital converter, which requires a PIC processor unit to link with the MX1 board serially.

Because the MX1 is a development board, we are able to perform most testing and debugging of software and graphical display on a PC connected to the board.

3.1 M9328MX1ADS/B DragonBall Development Board

The MX1 is a development board used typically for smartphone and PDA applications, but works well with the requirements of our product. This development board was chosen mainly due to its accessibility. The MX1 is packaged in a 256 pin MAPBGA (See Glossary). Reviews of the major components of this board, which will be used, are detailed below. Figure 4 below shows the functional block diagram for the MX1. For more specific functionality of the mentioned components please see references [2] and [3].

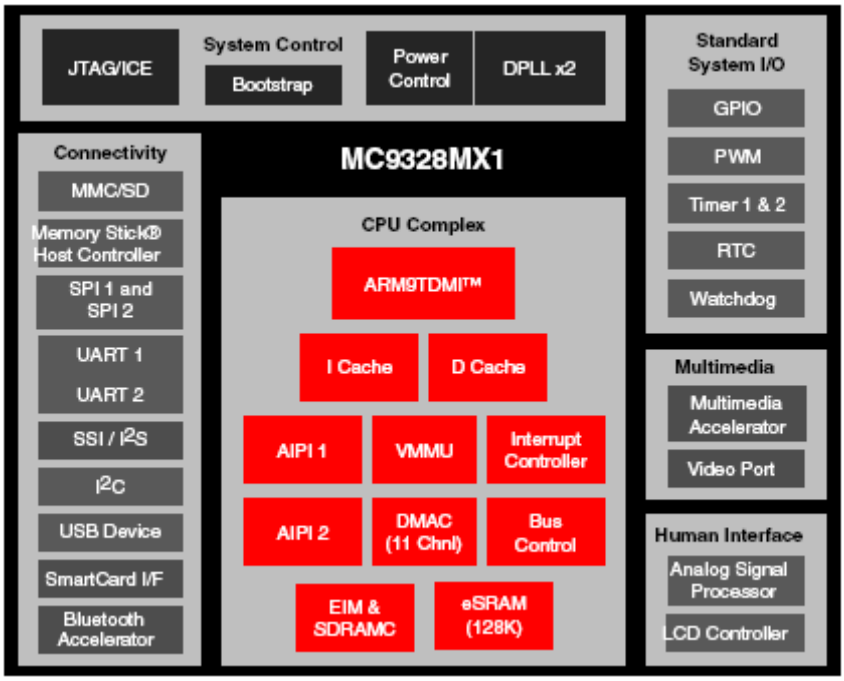

Figure 4: Functional Block Diagram of the MX1 Development Board

3.1.1 ARM920T™ Processor

The main development unit is ARM920T microprocessor that runs at 200MHz and is contained within the MX1's processor chip itself. This processor speed is more than adequate to handle the proposed FFT algorithm.

3.1.2 LCD controller with a LQ035Q7DB02 TFT LCD (Sharp Corporation)

The MX1 processor has a touch screen ADC and the MX1 development board includes an LCD peripheral, the LQ035Q7DB02. This LCD display allows for a very effective display mechanism that is used in our prototype.

3.1.3 USB, SPI and UART Connectivity

The MX1 is also able to process current data transfer devices, including USB connections, SPI data transfers, as well as serial communication through UART ports. These built in connectors allow for serial connection to and from our development board.

3.1.4 SyncFlash and SDRAM controller with onboard memory chips

The MX1 development board also comes with built in SyncFlash and SDRAM memory for OS loading and other memory requirements of our device. The MX1 is capable of controlling various memory types.

3.2 Microphone Input and Preamplifier Circuit

In order to capture data from the surrounding environment, a high-quality wide-frequency band microphone was used. A microphone powering circuit attached to a two-stage amplifying circuit was then implemented to produce an analog signal for our microphone. Figure 5 below shows the initial test schematic used for verification of our circuit.

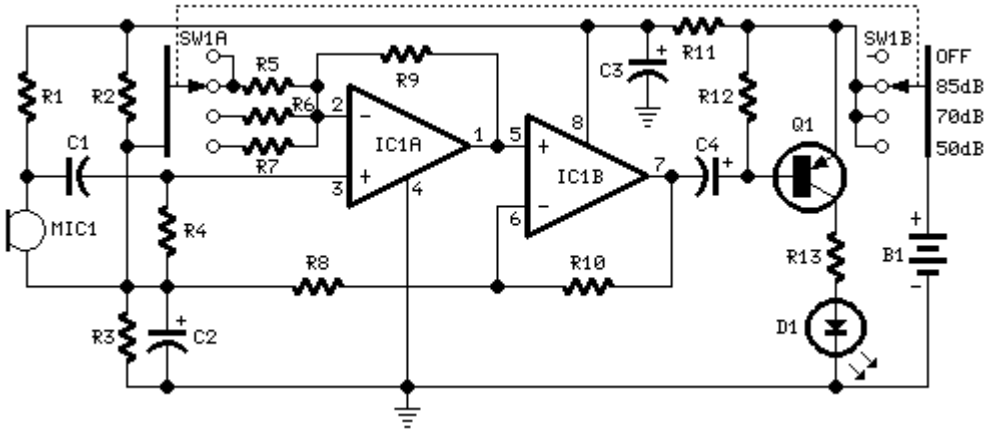

Figure 5: Schematic diagram of a simple sound amplifier [4]

A specification of the functionality of this circuit is described as follows.

3.2.1 Microphone Circuit

We designed our microphone circuit using an electret type condenser microphone which consumes relatively low power. The design of the microphone is based on a diaphragm component connected to a MOSFET, which is encapsulated together. The microphones used are high quality with an accepting frequency range of 20 Hz to 20 kHz.

3.2.2 Amplifier Circuit

The amplifying circuit we created was based on reference research [4] for an applicable type of circuit. The initial test circuit is a basic 2-stage circuit using operational amplifiers in a non-inverting configuration.

3.3 Analog-to-Digital Converter Chip

The AD974 Analog-to-Digital Converter chip is a 16-bit resolution chip that was chosen for our prototype because of its high precision. This chip uses SPI to connect to a master device, which is our PIC processor, and takes in the analog data from our microphone amplifying circuit. Figure 6 shows the functional block diagram for ADC chip that is produced by Analog Devices [5].

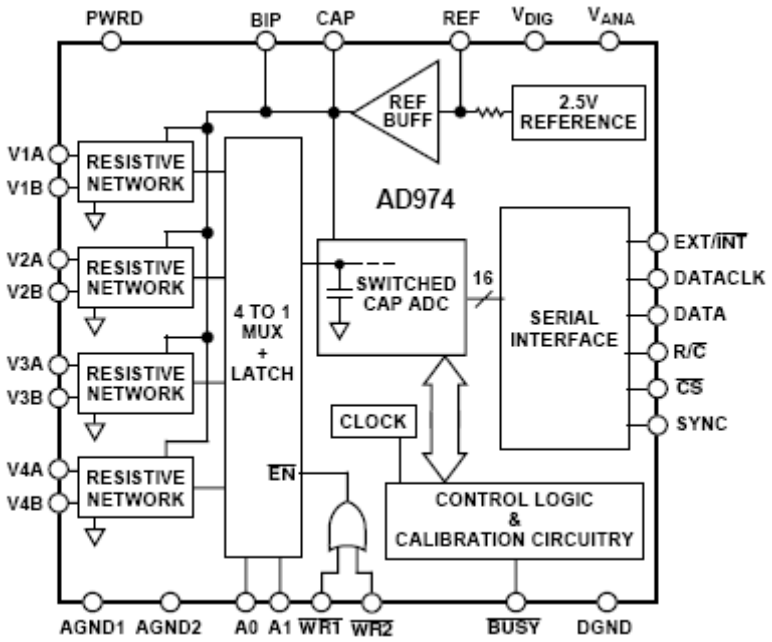

Figure 6: Functional block diagram of the AD974

3.4 PIC18F452 Processor Chip

The PIC18F452 microprocessor chip is the driving force of the data transfer to the MX1 development board. A major motivation for this specific option was due to the difficulty in using the expansion port of the MX1. The PIC18F452 is an 8-bit solution to our data transfer needs [6]. By using the serial capabilities of the PIC processor, we are able to transfer data via the serial port, which is more easily accessible by the MX1. Please refer to Appendix A for the functional and line connection block diagrams of the chip.

3.5 MAX232 Serial Driver Chip

The MAX232 Driver is used to convert the serial data from our PIC processor and made valid to the UART input of the MX1 development board. This driver simply amplifies the voltages of the data and makes it standard for UART input.

4 Interface Design

The Handheld Spectrum Analyzer will have three separate hardware stages, which include the following: Microphone and ADC, Microchip PIC Microcontroller, and the Motorola M9328MX1ADS/B board. The interconnections of these components are shown in Figure 7.

Figure 7: Interconnection of Main Three Stages

4.1 ADC to PIC Interface

In the first stage, our analog circuitry is converted to digital pulses using the Analog Devices AD974 ADC. This ADC is then connected to our PIC microcontroller by utilizing the Serial Peripheral Interface (SPI). There are a number of wires that need to be connected for this interface to work properly, and these are shown in Figure 8.

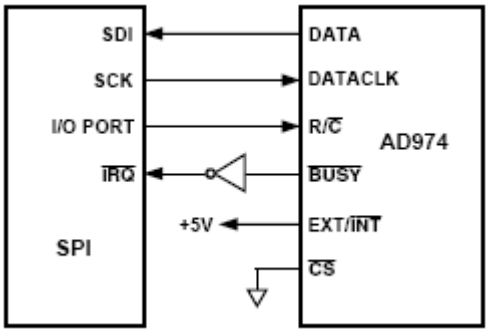

Figure 8: AD974 ADC to PIC Interface Connections [7]

The SPI is a very common method of interconnecting and communicating between various devices. Usage of the SPI only requires physically connecting four lines between the two components, and then initializing pins in a specific way on the master device. Afterwards, data from the ADC is serially transmitted via the SDI line (on the PIC). In our current setup, the PIC microcontroller is set as the master device and the AD974 as the slave device. This method of connecting the ADC to the PIC microcontroller was chosen because of its ease of use, and the large number of compliant 16-bit ADCs that support SPI.

The proposed design, after the completion of the first phase, will have the ADC constantly sending data over the SPI to the PIC once both devices have been powered on. The only instances where this data transfer is interrupted or stopped is when either the PIC is reset, or the device components are turned off.

4.2 PIC Microcontroller to Motorola M9328MX1ADS/B Interface

The next two systems that interconnect are the PIC microcontroller and the Motorola M9328MX1ADS/B and are interfaced using a RS232 serial connection. The serial method of communication was chosen due to lack of documentation and drivers in employing the expansion port of the Motorola board in Windows CE 4.2 environment. In addition, EVC++ 4.0 does not have any existing libraries for accessing such an interface. Hence, a viable solution was to use some of the alternative expansion ports that were included on the board. After weighing the advantages and disadvantages of the RS232 serial port and the USB port, we decided to use the RS232 serial port because of the pre-existing UART libraries in EVC++ 4.0, and also the existing PIC peripherals available when using RS232.

The physical setup of the connection between the PIC microcontroller and Motorola M9328MX1ADS/B is shown in Figure 9. In order to implement the shown solution, the MAX232 Transceiver as well as the female RS232 adapter on the PIC microcontroller were used. The large number of peripherals available for existing PIC Microcontrollers has made this design possible. No additional hardware is required because the RS232 female connector is already present on MX1 board; only a serial cable is required to connect the two components once they are assembled.

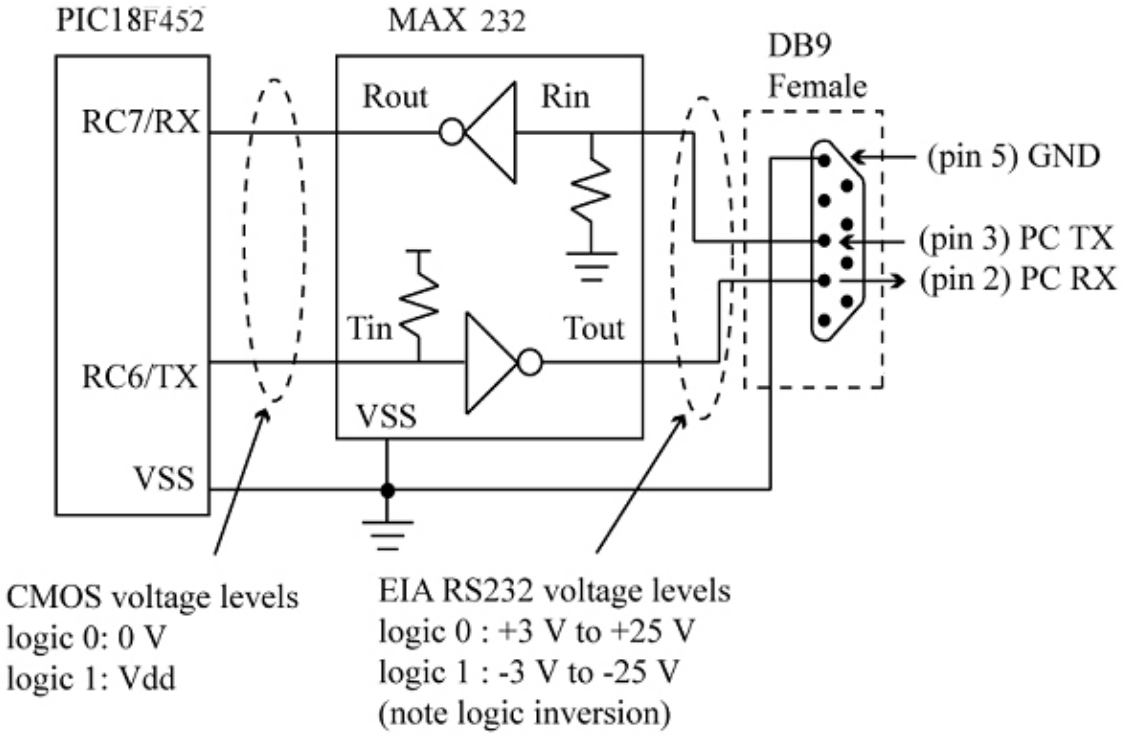

Figure 9: PIC Microcontroller's Serial Port Interconnection [8]

In our software design, the standard UART protocols will be used. The MAX232 transceiver on the PIC side will be utilized to increase or decrease the voltage levels of TTL logic, before being sent or received from the PIC. Each time, 8 bits (1 byte) of data block is then sent or received via the RS232 serial interface.

In our proposed design solution, the MX1 will first initiate the transfer of ADC data by sending a specific code value to the PIC. After receipt of this code, the PIC will start sending the ADC data via the RS232 serial port until it receives the stop code. Other instances where the PIC will stop sending data over the RS232 interface are when the PIC has been reset, or turned off. On initial boot up of the PIC, no data is exchanged via the RS232 interface with the MX1 until a signal is received from the other end.

5 Software Design

As it was outlined in the Software Overview section, the device's software consists of the PIC module, the Serial module and HSA application. These are discussed in more detail in the following sections. Figure 3 illustrates all the layers of software that will be implemented throughout various development phases.

5.1 PIC Module

In coding the PIC microcontroller, the C language was chosen over the more conventional Assembly Language. Programming in C requires less code, contains many pre-built library functions, and also reduces the strain on the developer to constantly keep track of memory locations and stack contents. In developing the software on the PIC, the readily available Microchip MPLAB IDE was used. In addition to this, the add-on MPLAB C18 compiler is used, which has a fully featured free student edition for 60 days [9]. Included in these development packages are also working simulators that contain an emulator for testing the serial port.

The choice of using the Microchip family of development tools over the alternative tools such as the Hi-Tech PICC Compiler was because of the limited number of supported devices in the free version and the high cost of purchasing such a tool [10]. In addition, the final compiled code will be flashed to the PIC microcontroller using the Microchip PICSTART Plus provided in the Simon Fraser University Engineering Science (ENSC) Laboratory 1. This device is well supported using the Microchip MPLAB IDE, which is another one of the critical reasons behind choosing this IDE solution.

The PIC Microcontroller software code can be broken down into three main modules, which consist of: SPI module, Serial RS232 module, and Word control. In this case, a Word is represented by a 16-bit value (2 bytes). Figure 10 illustrates the communication between each of these modules as data from the ADC is being inputted.

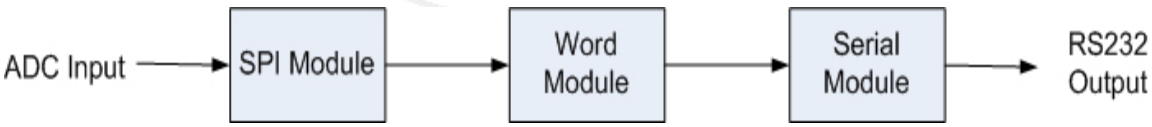

Figure 10: PIC Software Module Overview

5.1.1 SPI Module

In this module the PIC18F452 will be initialized to use the SPI port to accept incoming data. The PIC microcontroller will be setup, as the master device while the Analog Devices AD974 ADC will be set as the slave device. Also, to ensure constant synchronization, the AD974 will be driven by the clock outputted on the SCLK line of the PIC microcontroller.

To initialize the SPI, the correct bits in the MSSP Control and Status Registers should be set. Afterwards, the SSPIF interrupt flag will notify the ADC when new incoming data is ready for processing. The new data can be read from the 8 bits present at the SSPBUF register. Figure 11 illustrates the components used when accessing SPI on PIC18F452.

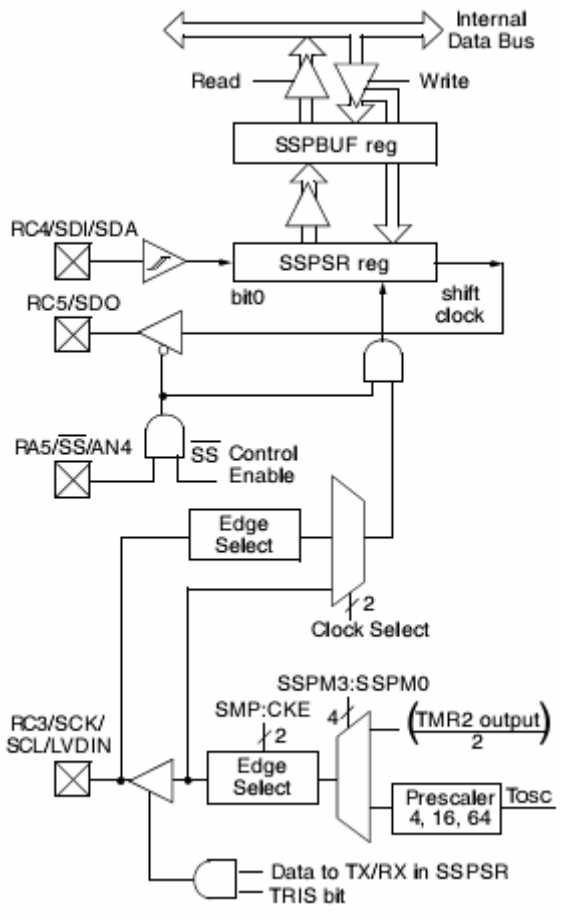

Figure 11: PIC18F452's Internal Interconnections (when using SPI)

In our proposed design, there will only be one-way traffic between the AD974 and the PIC microcontroller. Once the SPI connection is initialized, the AD974 will constantly send 8-bit data to the PIC microcontroller. Since the AD974 is a 16-bit ADC, another module is implemented to handle the extra byte as explained in Section 5.1.3.

5.1.2 Serial RS232 Module

The Serial RS232 Module is implemented on the PIC by sending data through the RC7 (RX) and RC6 (TX) pins, which is then driven to higher voltages using the MAX232 and sent through the DB9 connector to the MX1 board. In this software module, signals from the MX1 will be received and sample bytes will be sent to the development board. The received bytes are be considered as control commands of the PIC microcontroller. With the current proposed design PIC will respond to only two types of control bits:

- Token to Start sending data via RS232
- Token to Stop sending data via RS232

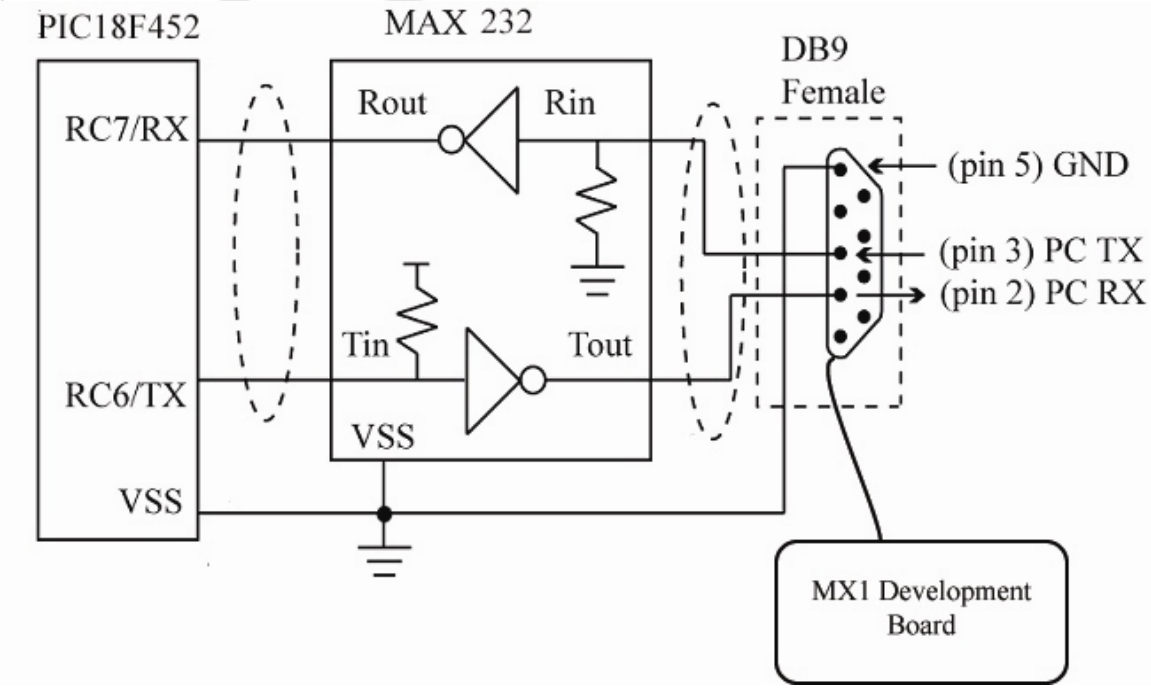

Figure 12: Connections between PIC18F452 and M9328MX1ADS/B

Above, Figure 12 shows the connections for the PIC18F452 and the MX1. On the PIC processor, the appropriate bits located in the TXSTA (Transmit Status and Control Register), RCSTA (Receive Status and Control Register), BRG (Baud Rate Generator) and TRISC (Port C Data Direction Register) are initialized. The BRG will set the baud rate of the transfer according to the frequency of the oscillator being used in the PIC. After the PIC

microprocessor has been appropriately initialized, it can receive data from the M9328MX1ADS/B by reading the RCREG register when the RCIF interrupt flag is set. Similarly, the PIC microcontroller can send data to the M9328MX1ADS/B by first checking that the transmit register is empty and then inserting an 8-bit value onto the TXREG register.

All the transfers taking place on the serial RS232 port are 8-bit transfers. Because incoming data from the 16-bit ADC needs to be transfered to the MX1, the next module will be developed to differentiate between the high and low bytes.

5.1.3 Word Control Module

While the employed ADC has a 16-bit precision, the selected PIC processor can only transfer 8-bits via the SPI; thus Word Control module is designed to transfer the 16-bit samples. An illustration of the Word output obtained from the ADC is shown in Figure 13.

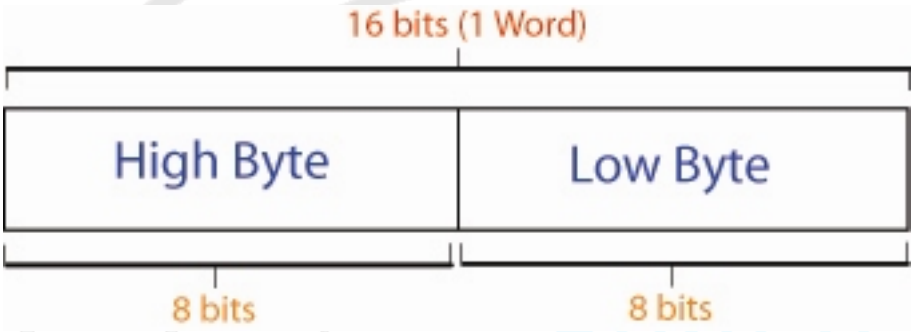

Figure 13: Reconstruction of Sample Word Method

On the first transfer over the SPI, the high byte of the input sample will be sent via the serial port. Afterwards, the low byte of the input sample from the ADC will be sent. Lastly, before the high byte of the next input sample is sent, delimiter value is sent to the MX1 board. This delimiter value will be a special byte such as a binary value of all zeros.

5.2 Serial Module

As it was mentioned before that the C++ language was chosen to create a transfer module between the PIC microcontroller and M9328MX1ADS/B board. Two reasons for the selection of C++ language over alternatives are the availability of the serial port libraries and ease of transfer between the output file and the HSA application.

5.3 HSA Application

Handheld Spectrum Analyzer consists of three main modules: the FFT calculation, data module and user interface. All coding is done in Embedded Visual C++. The functional diagram for the HSA software application is shown in Figure 14 and explained in more detail in the following sections.

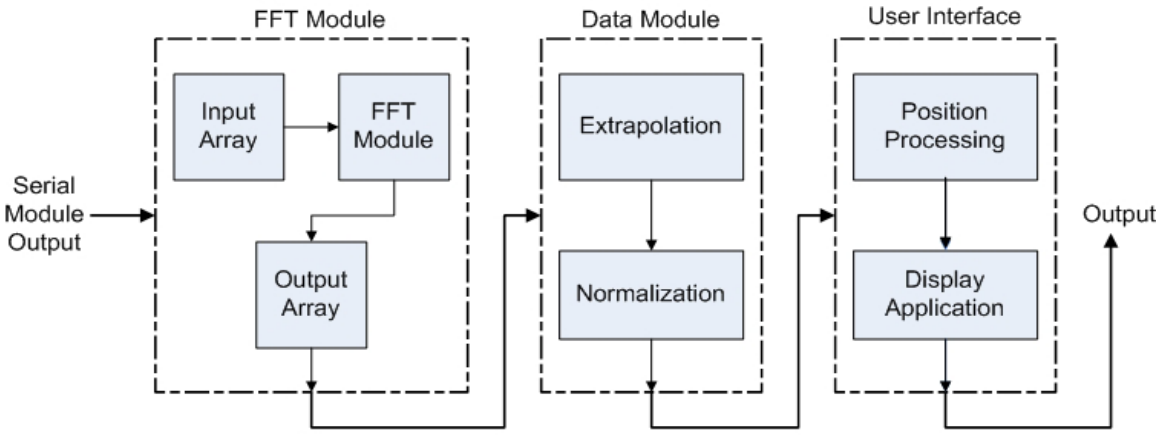

Figure 14: HSA Application Overview

EVC++ 4.0 SP4 environment was chosen due to its compatibility with the device's operating system, WinCE 4.2 and ease of code transferability from the development environment of the PC to the device. In addition, there are numerous libraries and example code available from the MSDN website that are used in writing the serial interfacing and the application's GUI.

5.3.1 FFT Calculation Module

The M9328MX1ADS/B receives 16-bit ADC data samples from the PIC processor on each transfer. Two consecutive bytes of data are combined to formulate 16-bits of input data and each Word of data is separated by a delimiter byte. These ADC Word samples are then fed into the Fourier Transform algorithm in order to calculate the frequency spectrum values.

To determine the weight or power of a specific frequency present of a digitized sound, the obtained samples can be multiplied by a sinusoidal weight function with the same frequency and then should be summed . This summation conveys the quantity of a given frequency present in the given digitized sound sample. A set of such numbers corresponding to various frequencies, is called the Fourier Transform, also known as FT, of the sound [13]. The following formula can be used to get the FT numbers corresponding to various frequencies as k varies:

$$A_k = \sum_{n=0}^{N-1} x_n e^{-2\pi i k n / N} \quad [14]$$

The components of the Fourier Transform of digitized sound are complex numbers with a modulus and a phase. The modulus of a given component describes the intensity at a particular frequency, whereas the phase describes the phase shift with respect to the phase at which it starts the oscillation. This modulus can be used for determining the power of the sound at a given frequency. The phase can be used in the future to implement any signal processing involving multiple channels, for example, comparing two sources of sound waves [15].

With so many options to choose from, two main algorithms considered are Fast Fourier Transform (FFT), and Fastest Fourier Transform in the West (FFTW). Both algorithms will be coded and tested using EVC++ 4.0 in MX1. Based on the output accuracy, functional speed and compatibility with our processor, the better performing program will be used for HSA's calculations.

5.3.1.1 FFT Algorithm

For the usual N-point Discrete Fourier Transform, the N-dimensional vector containing data from the sample buffer is multiplied by N separate basis vectors. This involves N multiplications and N additions, making the total required time proportional to N^2. So the algorithm is of order $O(N^2)$. Using the Fast Fourier Transform (FFT) method, the algorithm can be optimized down to order O(Nlog(N)), which gives a greater efficiency for large N. This method performs the best if N is a power of 2, where it divides the samples into two groups of even and odd, and keeps on dividing recursively to reach a 2-point stage. The algorithm utilizes the fact that the initial division is equivalent to sorting the binary numbers in the buffer by bit-reversed order. At the end of applying the interrelating formula (named 'butterfly') on adjacent pairs of numbers in the buffer separated by a power of 2 at stages until the separation is N/2, the Fourier transform is obtained [16]. An example of this coding algorithm is freely provided by Reliable Software and can be altered for use in the HSA's FFT module.

5.3.1.2 FFTW Algorithm

FFTW algorithm is for computing the discrete Fourier transform (DFT) in one or more dimensions of arbitrary input size, and of both real and complex data (as well as of even/odd data) was developed at Massachusetts Institute of Technology (MIT) and claims to be the fastest in the west [17]. Similar to an FFT, this algorithm also works the best for N of power of 2. This software component consists of a variety of FFT algorithms and implementation strategies, whose combination into a particular plan for a given size can be

determined at runtime and arbitrarily composed to adapt itself to a machine. A code generator to produce highly optimized routines for computing small transforms and explicit divide and conquer style to take advantage of the memory hierarchy provides the fast speed [18]. The source code is published for free as defined by the Free Software Foundation and is distributed under the terms of GNU General Public License [19].

5.3.2 Data Module

As explained in the previous section the processed data from the FFT module is stored into an array and is fed as the input of the data module. In order to eliminate errors and discrepancies of the processed data, an extrapolation algorithm is used. In addition, a normalization module helps in organizing data points efficiently, reducing processing time by eliminating redundancy and improving data consistency. The output of this module is saved into an array file that is being fed into the GUI module.

5.3.3 GUI Module

Two main components of the GUI module are the position processing and the display modules. The position processing module further processes the normalized data into an array, which can be easily understood by the display module functions.

The display module is designed to plot the final frequency spectrum of the input and has a menu bar, as shown in Figure 15, for user interaction.

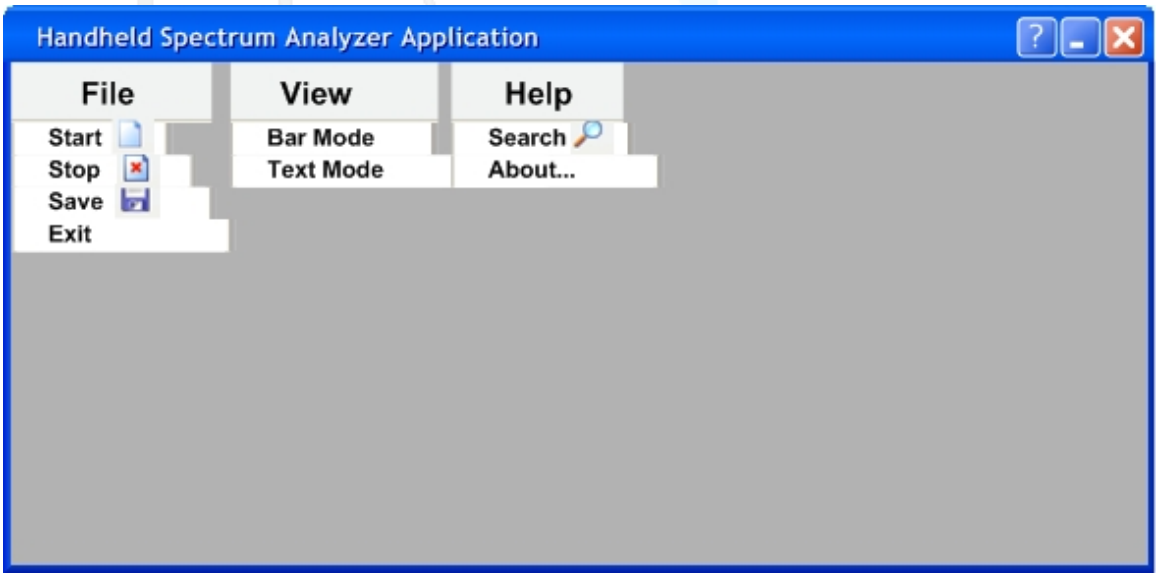

Figure 15: Sample GUI Window

In developing the GUI's window, common WinCE libraries were used to create a standard menu bar. To better illustrate the outline of the application's interface, Figure 16 shows a sample output window when a constant array of data is fed into the GUI module.

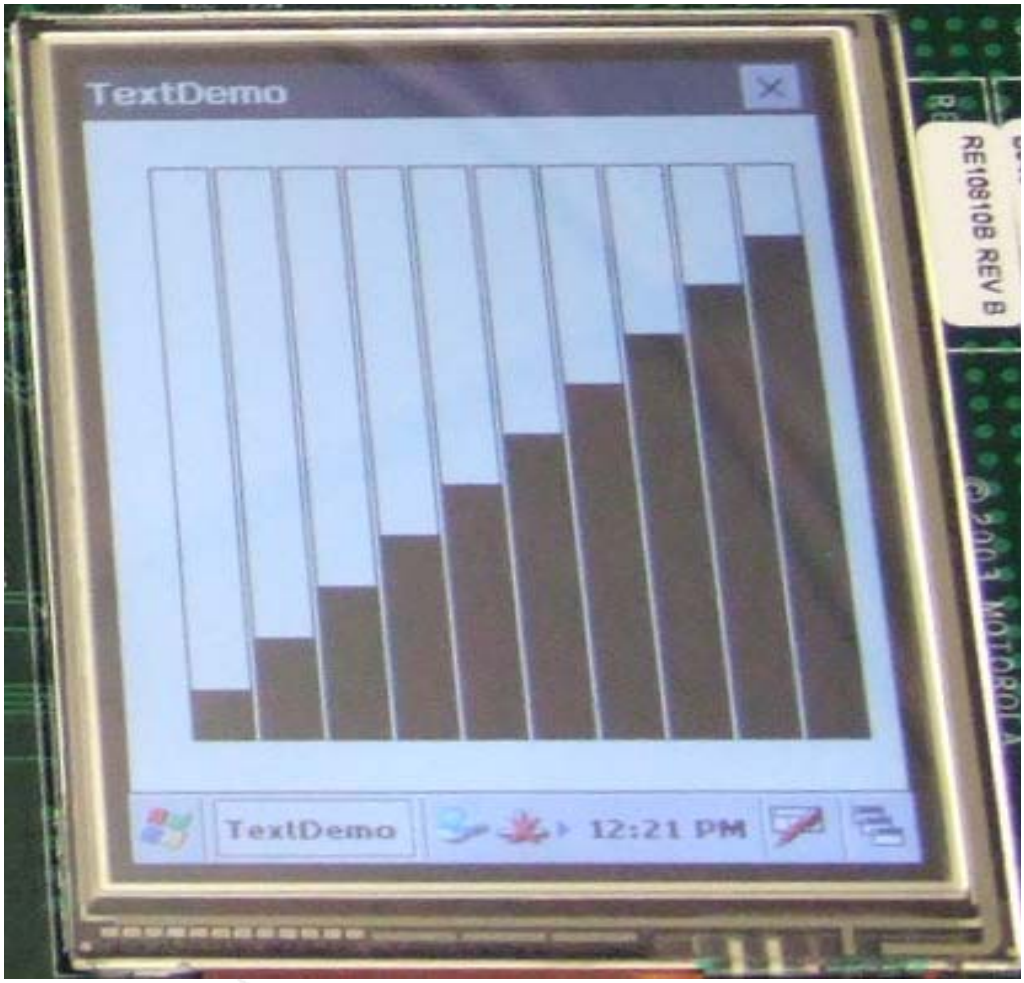

Figure 16: Windows CE Bar display sample.

It should be mentioned that only some of the shown features of the menu bar are programmed during the initial development phase and will be functional on the proof-of-concept prototype. Further menu features will be available on the final product for market.

6 Test Plan

To better organize the quality assurance phase of the product development, testing has been divided into three main sections: Hardware, Software and Integration testing. Please note that while during the initial phases of development these levels of testing are performed sequentially by different development teams. However, all these areas are combined and tested concurrently during the integration phase.

6.1 Hardware Testing

Essential hardware components must be tested individually before approaching the integrated testing. Different parts require different tests to ensure their individual functionality. This section describes the principal testing procedures for the main hardware components. Additional testing and analysis process may be introduced as part of the ongoing Hardware or Integration testing when necessary. Table 1 summarizes the tests that will be performed. More details on the listed tests are provided in the subsequent sections.

Table 1: Hardware Component Test Outline

Component	Test Outline
ADC	- Check the ADC output on the oscilloscope - Feed in a wave (rectangular) and verify the output results
Microphone	- Test the voltage output of the microphone on the oscilloscope
Amplifier	- Verify the gain with PSPICE
SPI connection	- Verify test input and valid serial output
PIC/MX1 Board Serial Port	- Check output using the hyper terminal
MX1 Board	- Load an operating system into the board - Test board's LCD by loading a sample code

6.1.1 ADC Testing

To ensure that the ADC successfully converts an analog signal to digital bits, a voltage wave function (preferably a rectangular function) can be applied to the powered ADC on the breadboard as an input. The corresponding output signal can be observed using an oscilloscope. The test should verify that the ADC outputs serial bits for each sample.

6.1.2 Microphone Testing

The microphone should be powered on a breadboard. The voltage function across the device should be checked with an oscilloscope and it should be verified that the output voltage of the microphone varies as does incoming sound wave.

6.1.3 Amplifier Testing

Using a specific small voltage input, the amplified output can be observed at the oscilloscope. Comparing the output voltage to the input voltage, closed loop gain for the amplifier can be easily measured using the formula:

$$Gain,\ g = \frac{Output\ voltage, V_o}{Input\ voltage, V_i}$$

A set of data for different input frequencies and amplitudes can show the consistency of the gain over frequency and amplitude ranges. The measured gain of the amplifier should also be verified using PSPICE to ensure consistency and adequate amplification of our input source.

6.1.4 SPI Testing

In order to test that the SPI functions properly with the ADC and the PIC , a square wave will be fed into the ADC input. Afterwards, the stored values on the PIC will be checked to see if they are cyclic and approximate the input; if this test is successful, the input across the SPI is valid.

6.1.5 Serial Port Testing

Using a computer's serial port, data can be transferred between the computer and the serial port through Hyper Terminal. An alphanumeric character from the computer will be sent to the PIC microcontroller, and the PIC will return the next alphanumeric character in ASCII form to the computer. For example, if the computer inputs 'a', the PIC will return a 'b'.

6.1.6 MX1 (M9328MX1ADS/B)

The development board M9328MX1ADS/B can be tested by loading an operating system such as Windows CE 4.2 to utilize the board's ARM processor. This operating system should validate that the touch-screen LCD is working. The board's processing power can be tested by running various calculation-intensive programs. Peripheral connectivity components that can be used include the serial port and USB port. These two connectivity ports can be tested by establishing a successful connection to a personal computer using Hyper Terminal and ActiveSync respectively. A headphone can be plugged into the headphone jack and a song can be played to verify that the audio DAC is working.

6.2 Software Testing

The following sections indicate the layers of testing that will be performed to verify the functionality of all the developed software modules.

6.2.1 FFT Algorithm Testing

To ensure the accuracy of the FFT module, the results of an array of sample data is processed by the algorithm and compared against the results obtained from feeding the same array into MATLAB.

6.2.2 GUI Testing

The credibility of the user interface's plotting algorithm will be tested using sample data arrays. In addition, standard debugging practices will be performed by the developer on the emulator. This will eliminate any logical inconsistencies in the code. The final stage of the GUI testing will be to load the application onto the M9328MX1ADS/B board. Numerous test samples will then be run to ensure the functionality of application's menu items.

6.2.3 Integration Testing

The primary task of integration testing is to ensure that the device is producing results as expected and a final output is shown on the LCD display. One final test would be to play a musical instrument near the HSA, and verify that the power-levels of the specific musical note frequency are elevated on the LCD. Further field-testing and verifying product stability will be the major task of the integration.

7 Conclusion

This document has entailed the major outlines of expected design steps and choices for the creation of the initial prototype of the HSA device. Engineers and application developers who read this walkthrough should exercise caution when implementing the designs to ensure the most up-to-date information has been acquired. The designs described are the primary choice of SonoSense, but can be changed if found to be inadequate for the requirements expected. The initial prototype shall be completed by April 25, 2007 and work will continue until the final product is available to market.

8 References

[1] SonoSense Technologies Inc. 2007. *Functional Specifications for a Handheld Spectrum Analyzer* Burnaby, British Columbia.

[2] Freescale Semiconductors, Inc. 2007. i.MX1 Product Summary Page. 8 Feb. 2007. <http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=i.MX1&fsrch=1>

[3] Freescale Semiconductors, Inc. 2007. M9328MX1ADS_B Product Summary Page. 12 Feb. 2007. <http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=M9328MX1ADS_B&parentCode=i.MX1&nodeId=02XPgQ82172973ZrDR>

[4] Electronics-Lab.com.2006. Room Noise Detector. 14 Feb. 2007. <http://www.electronics-lab.com/projects/sensors/012/index.html>

[5] Analog Devices, Inc. 2007. Analog Devices AD974 - 4-Channel, 16-Bit, 200kSPS Data Acquisition System. 2 Mar. 2007. <http://www.analog.com/en/prod/0,2877,AD974,00.html?>

[6] Microchip Technology Inc. PIC18F452. 27 Feb. 2007. <http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1335&dDocName=en010296>

[7] Analog Devices, Inc. 2007. AD974 – Specifications. 1 Mar. 2007. <http://www.analog.com/UploadedFiles/Data_Sheets/693498521AD974_a.pdf>

[8] Mississippi State University. 2006. ECE 3724 Microprocessors Lab Home page. 12 Feb. 2007. <http://www.ece.msstate.edu/~reese/ece3724/lab/>

[9] Microchip Technology Inc. MPLAB C18. 20 Feb. 2007. <http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=140 6&dDocName=en010014>

[10] HI-TECH Software. Freeware Microchip PIC C Compiler. 20 Feb. 2007. <http://www.htsoft.com/products/compilers/PICClite.php>

[11] Douglas Boling. Microsoft Press 2003. Programming Microsoft Windows CE .NET, third Edition. 18 Feb. 2007.

[12] Microsoft Corporation. 2007. Windows CE Home Page. 18 Feb. 2007. <http://msdn2.microsoft.com/en-us/embedded/aa731407.aspx>

[13] Reliable Software, LLC. 2006. Fourier Transforms: Fourier Analysis. 1 Mar. 2007. <http://www.relisoft.com/Science/Physics/fourier.html>

[14] Reliable Software, LLC. 2006. Freeware: How Frequency Analyzer Works. 1 Mar. 2007. <http://www.relisoft.com/Freeware/freq.html>

[15] Reliable Software, LLC. 2006. FFT: Digital Fourier Transform. 1 Mar. 2007. <http://www.relisoft.com/Science/Physics/dft.html>

[16] Reliable Software, LLC. 2006. FFT: Fast Fourier Transform. 1 Mar. 2007. <http://www.relisoft.com/Science/Physics/fft.html>

[17] Matteo Frigo and Steven G. Johnson. 2006. FFTW Homepage. 1 Mar. 2007. <http://www.fftw.org/>

[18] Matteo Frigo and Steven G. Johnson. 2006. Internals of FFTW. 1 Mar. 2007. <http://www.fftw.org/faq/section4.html#howworks>

[19] Matteo Frigo and Steven G. Johnson. 2006. FFTW Introduction and General Information. 1 Mar. 2007. <http://www.fftw.org/faq/section1.html#whatisfftw>

9 Technical Appendices

The following information is contains various images and designs too large to be placed through out the document. References are available for more specific information if required.

9.1 Appendix A

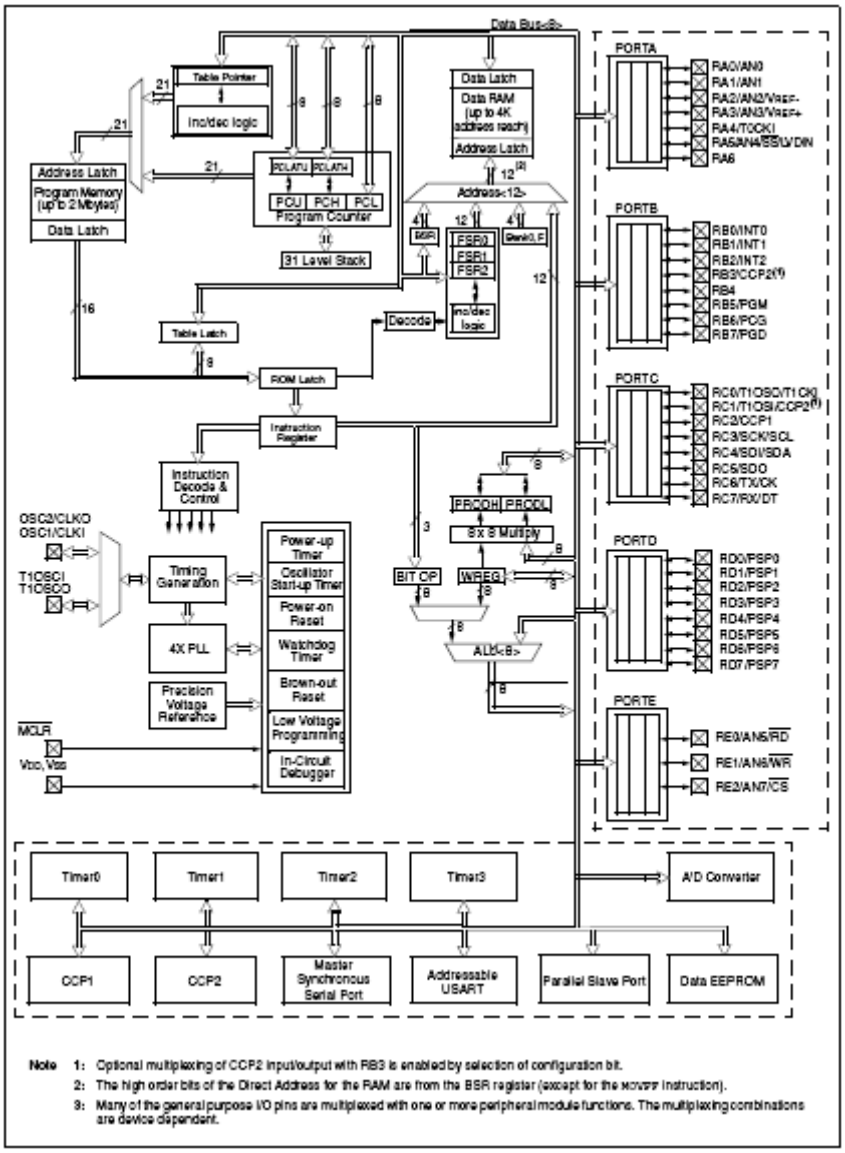

Figure 17: The Functional Block Diagram of the PIC18F452 [6]

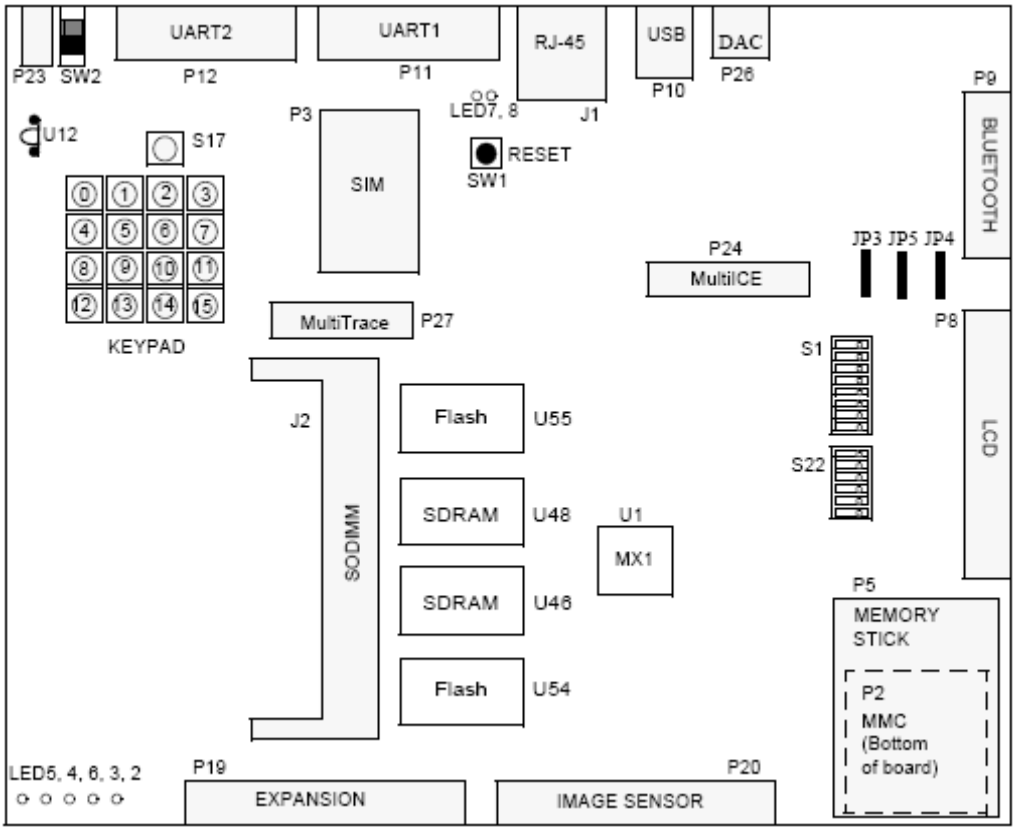

Figure 18: Placement Design of Components Connected to the MX1 Development Board [6]

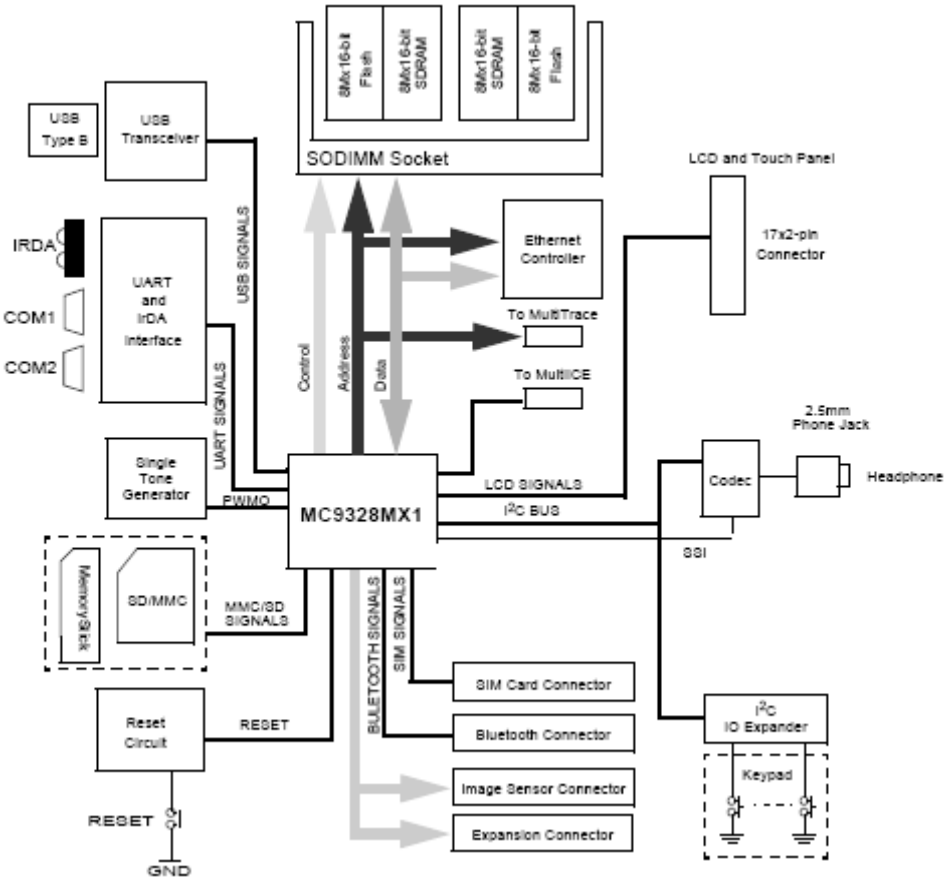

Figure 19: Functional Block Diagram of the M9328MX1 [6]