March 10, 2008

Mr. Patrick Leung
School of Engineering Science
Simon Fraser University
8888 University Drive
Burnaby, British Columbia
V5A 1S6

Re: ENSC 440/305 Design Specification for *RockIt*

Dear Mr. Leung,

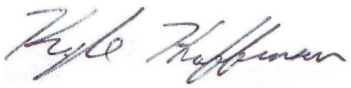
The enclosed document is entitled *Design Specification for Motion-Control Guitar Effects*. The document will further outline the design details of our capstone ENSC 440/305 project, entitled *RockIt*.

Currently, our prototype is in the development stage with several key elements complete. The remaining elements have been designed and the algorithms are awaiting implementation, barring some minor details and testing. Upon completion, our product will broaden the ranges of guitarists' capabilities by creating sounds based on the musicians' motions.

The design specification will outline details of our user interface and physical packaging with well-illustrated algorithms of the processing. Power consumption and component selection are also detailed in this document. Further details can be found in the appendix that specify component specifications, source code, and circuit schematics.

Perceptum Technologies consists of four engineers: Kyle Huffman, Daniel Galeano, Paul Carriere, Ben Shewan. Each team member is a well-accomplished engineer. If you have any questions or concerns about our product, please contact us through our email address: ensc440-spring08-perceptum@sfu.ca

Sincerely,



Kyle Huffman

Director of QA, Perceptum Technologies

PERCEPTUM TECHNOLOGIES

RockIt Design Specification

Project Members:	Kyle Huffman Paul Carriere Ben Shewan Daniel Galeano
Contact Person:	Kyle Huffman ensc440-spring08-perceptum@sfu.ca
Submitted To:	Steve Whitmore, ENSC 305 Patrick Leung, PEng, ENSC 440 School of Engineering Science Simon Fraser University
Issued:	March 10, 2008
Version:	A.1

Executive Summary

To further musicians' capabilities and stage theatrics, Perceptum Technologies aims to create a new motion-controlled sound effects system entitled *RockIt*. By placing sound controls directly on the guitar, the restriction of floor pedals is eliminated. By having an accelerometer placed on the neck of the guitar, musicians motions can be correlated to adjustable sound effects; this will better convey emotions and nuances from music.

RockIt will be built in two stages. Stage 1 is the functioning prototype whereas Stage 2 is the refined and streamlined product for sale. Stage 1 will have the following features:

1. 3D Output ADXL330 accelerometer controlling variable levels of two adjustable sound effects via MIDI signals
2. 4 Force Sensing Resistors (FSRs) controlling the accelerometer state and toggling sound effects
3. Two PIC18F2420 processors. One processor accepting user inputs from the accelerometer and FSRs and a second processor receiving commands to control the PODXT Live via MIDI signals while updating the user through an LCD display

Stage 2 of the project will have the following elaborated features in addition to the Stage 1 design:

1. Wireless communication between the two processors
2. Streamline housing for the FSRs, accelerometer, and pre-processor

Stage 1 will be completed by April 15th, 2008 where a demonstration will occur. This document focuses solely design issues surrounding on Stage 1.

Table of Contents

1 Introduction 7
1.1 Scope 7
1.2 Intended Audience 7
2 System Overview 7
3 Microcontroller 8
3.1 Pin-Out 9
3.2 PIC18F2420 hardware highlights 10
3.2.1 Analog comparator 10
3.2.2 Analog-to-Digital Converters 10
3.2.3 Memory 10
3.3 PIC18F2420 Programming 10
3.3.1 In-Circuit Programmer/Debugger 10
4 MIDI 10
5 Accelerometer 13
5.1 Physical Issues 14
5.1.1 Placement 14
5.1.2 Package 14
5.1.3 Cabling 15
5.1.4 Power 16
5.2 Signal Processing 16
5.2.1 Precision and Bandwidth 17
5.3 Tilt Sensing 18
5.4 Dynamic Motion 20
5.5 Optimization and Simulation 23
5.6 Potential Issues 24
6 Force Sensitive Resistors 25

6.1 Theory of FSR 25
6.2 FSR Detection 26
6.3 Algorithm 28
6.4 Physical Package 29
6.5 Potential Problems 30
7 Microcontroller Communication 31
7.1 Communication Protocol 31
7.1.1 Baud Rate 31
7.1.2 Frames 31
7.1.3 Packets 32
7.2 Hardware 35
7.3 Software 36
8 User Interface 36
9 Power Management 41
10 Conclusion 42
11 Appendix 43
11.1 Glossary 43
12 Bibliography 44

List of Figures

Figure 1: High Level Overview of Rockit System 8
Figure 2 Microcontroller Pin-Out 9
Figure 3: A typical MIDI message 12
Figure 4 Placement of accelerometer 14
Figure 5 Accelerometer Enclosure 15
Figure 6 Ribbon Cable Configuration 15
Figure 7 Accelerometer Signal Flow Diagram 17
Figure 8 Tilt Sensing Usage 19
Figure 9 Theory for Determining Gravity Projection 19
Figure 10 Tilt Sensing State Diagram 20
Figure 11 Dynamic Motion Sensing Axis 21
Figure 12 Dynamic Motion State Diagram 22

Figure 13 MATLAB Equivalent of Accelerometer 23
Figure 14 Debugging Platform 24
Figure 15 FSR Composition 26
Figure 16: FSR Force to resistance diagram 26
Figure 17 FSR Comparator Simulation Circuit 27
Figure 18 FSR Subsystem Signal Flow Diagram 28
Figure 19 PIC comparator and external input schematics 28
Figure 20 Flow Chart of FSR Timer Interrupt Algorithm 29
Figure 21 FSR Housing and Location 30
Figure 22 Frame format 32
Figure 23 FSR packet 33
Figure 24: Reset Packet 33
Figure 25 Accelerometer packet 34
Figure 26 Header frame in Accelerometer packet 34
Figure 27 Data frame in accelerometer packet 35
Figure 28 Microcontroller communication 35
Figure 29 Receiver State Diagram 36
Figure 30: User Interface 39
Figure 31: User Interface (continued) 40
Figure 32 Power Management Diagram 42

List of Tables

Table 1: PIC Pin-out Table 9
Table 2 FSR Simulation Comparator Component Details 27
Table 3 Framing Bits 32

1 Introduction

This document details our Design Specifications for *RockIt*, a motion-controlled sound effects system. Our product is being prototyped for a guitar, but the idea can be adapted for many uses. There are certain key restrictions existing for guitarists today that can be eliminated with our new sound control system and there are key advances that our new system allows. These advances are centred on the control of sound effects for better stage performance and enhanced freedom of expression.

1.1 Scope

This document specifies the proof-of-concept version of *RockIt*. An overview of the system is detailed with supporting sections to describe the exact design choices and reasons for those design choices. For highly documented information, the appendix contains more detailed specifications.

1.2 Intended Audience

This document is intended for the design team responsible for creating the *RockIt* prototype. It will serve as a reference for design decisions and testing procedures. Engineers will use this document during the prototype, optimization and final design phase, ensuring that the intended functionalities are achieved.

2 System Overview

RockIt consists of seven main parts:

- Four FSRs (Force Sensing Resistors)
- One accelerometer
- Pre-processing PIC
- MIDI controlling PIC
- Wall-mount Power
- LCD Display
- POD XT Live effects processor

As outlined in Figure 1, *RockIt* can be broken into two key sub-systems; the pre-processor and the MIDI controller. The pre-processor handles the logic from the FSRs and accelerometer. Using the processed accelerometer data, the guitarists' motions will update status memory to send to the MIDI processor. The four FSRs will toggle on/off the polling of the accelerometer in addition to toggling requests for sound effects. Both sets of sensors, FSRs and accelerometer, will have their statuses updated in the pre-processor and requests will be sent to the MIDI processor to control the POD XT Live.

The MIDI processor receives the status of sensors and simply generates MIDI signals to communicate with the POD. In addition, there exists an LCD display to update the user of the current sensor setting.

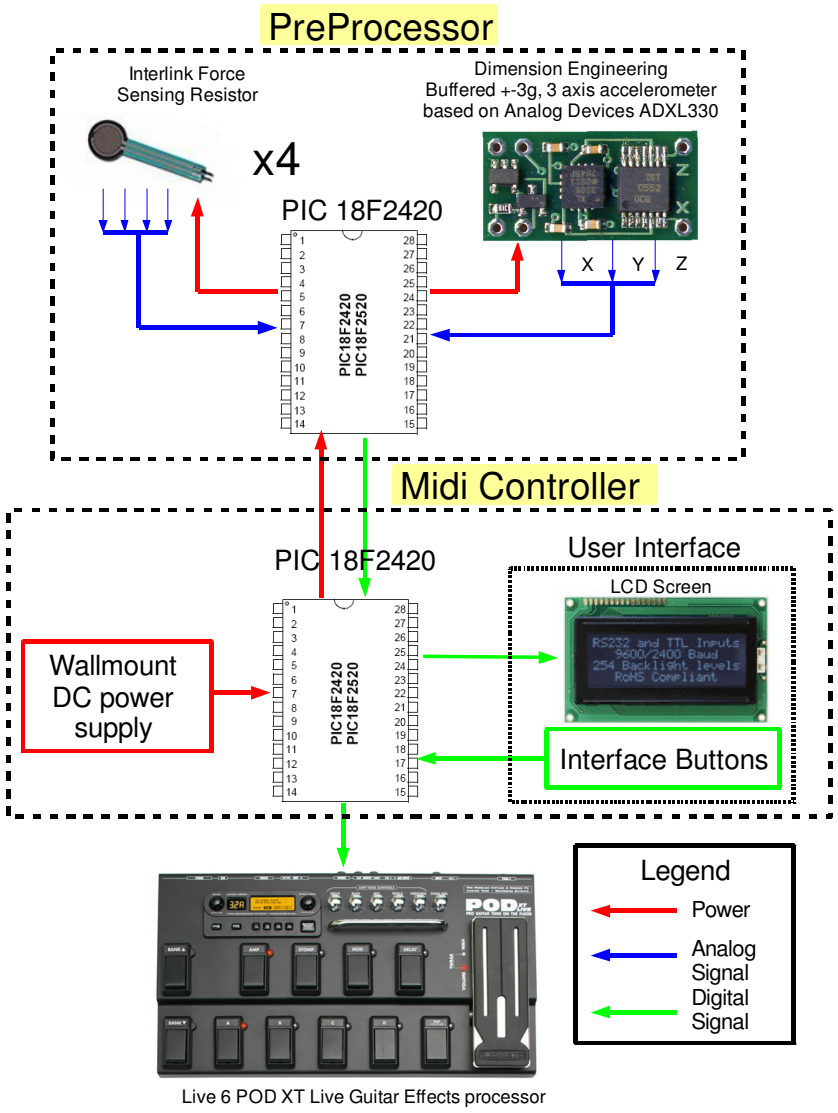


Figure 1: High Level Overview of Rockit System

(Interlink, 2008),(Dimension Engineering, 2008), (Microchip, 2007), (Line 6, 2007), (Netmedia , 2008)

3 Microcontroller

At the core of the Pre-Processor and MIDI Controller subsystems lies a dedicated microcontroller. The PIC18F2420 from Microchip was found to be the most appropriate microcontroller to be used in both subsystems; its selection was based on several hardware and software features required by *RockIt*.

3.1 Pin-Out

The PIC18F2420 has 28 pins, including three power pins and a master reset. The following diagram shows the pin distribution for each subsystem.

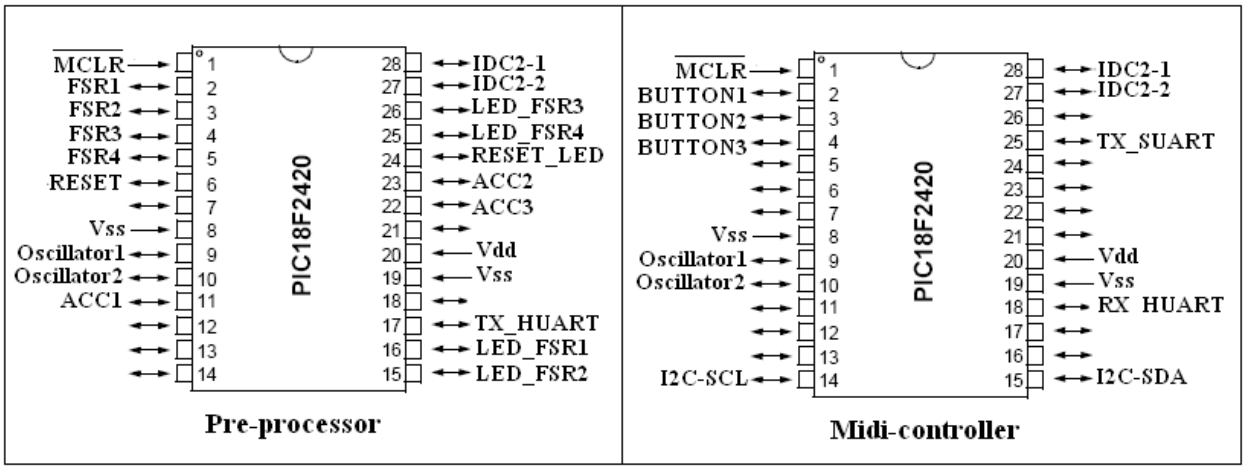


Figure 2 Microcontroller Pin-Out

(Microchip, 2007)

Pins 9 and 10 in both microcontrollers are used to configure a 10 MHz external oscillator. Pins 29 and 27 also have the same functionality in both microcontrollers; they are dedicated to the in-circuit programmer/debugger ICD2. The following table presents the pin-out on the microcontrollers.

Microcontroller	***Pin***	***Name***	***Description***
MIDI-controller	2,3,4	BUTTON1,2,3	User interface buttons
	14	I2C_SCL	LCD control signal
	15	I2C_SDA	LCD control signal
	18	RX_HUART	Hardware UART receiver
	25	TX_SUART	Software UART transmitter
Pre-processor	2,3,4,5	FSR1,2,3,4	FSR signals
	11,22,23	ACC1,2,3	Accelerometer signals
	15,16,25,26	LED_FSR1,2,3,4	FSR status LEDs
	17	TX_HUART	Hardware UART transmitter

Table 1: PIC Pin-out Table

3.2 PIC18F2420 hardware highlights

The following subsections outline the main product-specific features that make the PIC18F2420 optimal for this project. These specifications are taken from (Microchip, 2007)

3.2.1 Analog comparator

An important peripheral highlight of the PIC18F2420 is the dual analog comparators that allow the digitization of the FSR data by comparing their analog signal with a reference voltage. This feature is implemented in the pre-processor microcontroller and permits a more integrated design.

3.2.2 Analog-to-Digital Converters

The analog-to-digital converter module in the PIC18F2420 allows processing of the signals from the accelerometer. This module provides up to 10 dedicated analog input pins to allow conversion with 10-bit precision. Unimplemented analog input pins can be used to add a second accelerometer in future designs.

3.2.3 Memory

The data memory of the PIC18F2420 has 768 bytes of SRAM that provides enough data storing capacity for both the pre-processor and the MIDI-controller. In the pre-processor subsystem, the microcontroller uses the SRAM memory for data processing. The MIDI-controller uses the SRAM for the LCD strings and MIDI commands supported by *RockIt*.

3.3 PIC18F2420 Programming

MPLAB is an IDE from Microchip that is compatible with the PIC18F2420 and which also offers important debugging and programming tools. MPLAB is compatible with a student edition C-compiler called C18, which is also provided by Microchip. The main motivation for programming in C is to take advantage of the development libraries. The chosen C compiler (C18) provides software libraries that simplify the programming of UART, A/D conversions and timers. The math library provided in C18 also simplifies the coding of signal processing algorithms that are needed in the pre-processing subsystem.

3.3.1 In-Circuit Programmer/Debugger

The use of an in-circuit programmer greatly simplifies the code development by allowing a step-by-step debugging of the program. ICD 2 from Microchip is the in-circuit programmer/debugger we used. It's important to note that this in-circuit programmer uses some of the microcontroller resources such as two I/O pins (pins 28 and 27), stack space, and program and data memory (Microchip, 2007) . The design of the RockIt already accounts for these trade-offs.

4 MIDI

Perhaps the most fundamental part of the *RockIt* system is the ability to control a wide range of commercially available music effects. The Musical Instrument Digital Interface (MIDI) is an industry standard protocol which allows electronic music instruments and other equipment to synchronize and communicate with each other. First introduced in the early 80's, this highly successful protocol has

remained almost entirely unchanged and been adopted by most major brands in the electronic music industry. While many people associate MIDI with poor quality video game music, it is actually the music synthesis equipment that limited the capabilities of these early MIDI applications. MIDI is simply a protocol for asynchronous serial digital communications between different devices. As such, the interpretations and actions taken in response to MIDI signals are entirely dependent upon the device receiving these messages. Aside from the more conventional applications of the protocol (such as keyboard controllers), MIDI has been utilized for the control of lighting and other stages effects, as well as the synchronization, composition and recording of music in a studio environment. (Borg, 2002)

MIDI signals are based upon a current loop connecting the transmitting and receiving devices. This current loop provides a one-way communications path between two devices; thus, to realize two-way communications, a path must be provided for both MIDI IN and MIDI OUT signals. The flow of current in the signal path is nominally between 5 and 10mA, indicating a digital 0, whereas the absence of current indicates a digital 1. Because MIDI does not utilize a common clock signal between devices, it is necessary to frame a data byte with start and stop bits. A start bit is defined as a digital 0, and will allow the receiving device to prepare for incoming data bits. Since the idle state of the line is a digital 1, the presence of the start bit will unambiguously inform the receiving device of the impending transmission. Following the start bit, a byte of data will be transmitted, followed by a stop bit indicating the end of the transmission. MIDI data is transmitted at a baud rate of 31250 (alternatively, the duration of a single bit is 32µS) and is generally composed of one, two, or three data bytes. To prevent ground loops and undesirable signal coupling, optical-isolators are used to receive a MIDI signal. (Borg, 2002)

Since the current driving a MIDI communications circuit is generally less than 10mA, the physical length of cabling is limited without the use of repeaters. MIDI controllable devices are also capable of being daisy-chained together to allow information to propagate between multiple devices. The maximum number of devices connected in this fashion is ultimately limited by the propagation delay incurred at each node. (Borg, 2002)

Up to 16 different devices are able to be controlled with MIDI messaging, and the specification of a channel number (0-15) will serve as a form of addressing to a specific device. Some MIDI messages are intended for a certain receiver, so the transmitter must specify the channel number this receiver has been set to. Other messages may be addressed to all connected devices, and MIDI is therefore provided with a broadcast message type that will be received regardless of channelization. Since we only need to control a single device in the *RockIt* system, the MIDI protocol will be more than adequate in this respect.

The parameters of the PODxt Live are able to be dynamically controlled in real time via MIDI messages. In this way the musician will be able to utilize sensor signals to control their desired effects. The various effects implemented on the PODxt are all associated with a unique MIDI control code that allows this effect to be changed via MIDI commands. There are 128 unique control codes (0-127) that may be sent in the MIDI protocol and each of the PODxt`s effects is assigned to one of these codes. The

interpretation of a received control code is dependent on the manufacturer's specification and generally varies from product to product.

A typical MIDI message that may be sent to the PODxt is presented below.

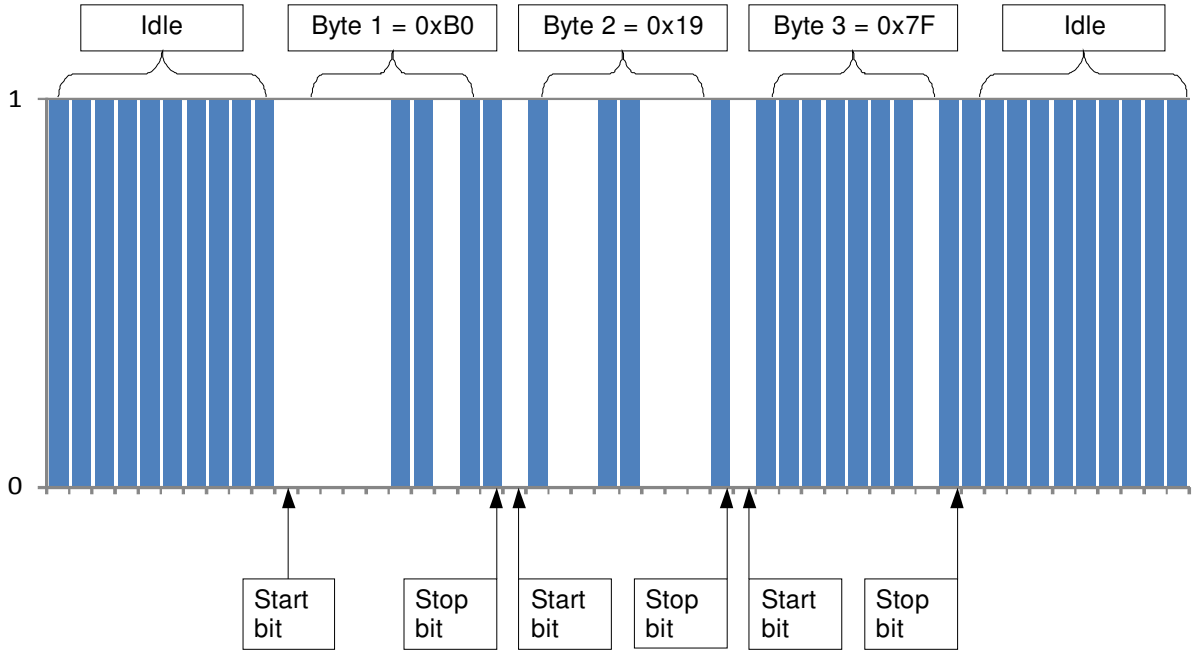


Figure 3: A typical MIDI message

The above message sets the device volume to maximum and consists of 3 bytes of information. The first byte represents status and serves to specify the channel and message type. Since we are sending a control message on channel 0, this byte will have a value of 176 or B0 in hexadecimal. The 8 bits of information in each byte are framed by start and stop bits. The second and third bytes respectively select the volume as the control parameter and set the volume level. Since volume is control code 25 on the PODxt Live, the 2nd byte will contain a 19 in hexadecimal. The 3rd byte will set the volume to some value between 0 and 127. Since we wish to maximize the volume, a value of 127 (or 7F in hex) is sent in the last byte. It should also be noted that the binary representation of these values is always sent with the least significant bit first.

The generation of MIDI messages using the PIC18F2420 microcontroller is a reasonably straightforward process. Since MIDI is an asynchronous serial communication protocol, it is practical to use the USART (Universal Synchronous Asynchronous Receiver Transmitter) module provided with the PIC18F series. By setting onboard control registers, we are able to setup the desired baud rate and framing bit information. The use of pre-compiled C libraries makes the task of outputting bytes onto the USART transmit-pin very simple. Since we are only transmitting commands to the effects device, there is no need to read MIDI messages at any time. The I/O pins on the PIC18F series are capable on sourcing

25mA of current, which makes them suitable for directly driving a MIDI circuit. Because we do not need to use buffering circuits or amplifiers, the system simplicity is maintained and the cost is minimized.

5 Accelerometer

At the heart of the *RockIt* system lays an analog sensor which transforms motion into modulated guitar effects like wah-wah or delay. To implement this functionality, the use of gyroscopes, strain gauges and exotic potentiometers was considered; all of which have been used in electro-musical systems.(SensorWiki, 2008) But considering the constraints on the user while playing guitar, we decided that our best solution would be the use of an accelerometer.

Having selected a sensor type, various MEMS-based accelerometers are offered by Analog Devices, Freescale and MSI were researched. The sensor must provide:

- Analog Output, as '*analog accelerometers are usually preferred for music interaction systems*' (SensorWiki, 2008)
- Sensitivity in the 5g range. This value was corresponds to limits of human motion, defined (SensorWiki, 2008)
- Bandwidth which is tuneable using off-chip bypass capacitors
- Low costs
- A readily available development package which is simple to interface and implement
- 3 measurement axes

When we measured these criteria against possible choices, the Analog Devices ADXL330 accelerometer packaged by Dimension Engineering (DE) was the obvious choice. The device includes analog output, sensitivity of ±3g, tuneable bandwidth and 3 measurement axes. The chip is wrapped in a DIP package along with an on-board power regulator and output impedance buffer. These added features are essential to the proper operation of accelerometer, as the PIC cannot process the native output impedance of the accelerometer, and power regulation is crucial to accurate reading of the accelerometer (Texas Instruments, 2004). Also, this accelerometer is used in the Nintendo Wii Nun-Chuck, confirming its applicability to the *RockIt* system.

The ADXL330 implements a capacitive sensing output dependent on the distance between two planar surfaces. (ADXL330, 2007)This method of sensing is known for its high accuracy and stability(SensorWiki, 2008). The output is represented by a voltage between 0 and V_{cc} where $V_{cc}/2$ represents the zero g position. The accelerometer is capable of sensing both the effects of gravity and the effects of dynamic motion. The DC value of the sensor represents the effects of gravity, whereas dynamic motion is represented by an AC waveform. While using the on-board regulator, we expect a sensitivity of 333mV/g (Dimension Engineering, 2008) Note: the accelerometer convention is to use **g** as the unit of acceleration. This unit represent acceleration normalized to earth's gravity, which is 9.81m/s^2

5.1 Physical Issues

The following section presents a short discussion of the physical issues and design solutions regarding the accelerometer, including placement, packaging, cabling and power.

5.1.1 Placement

Using the simple rules of levers, we can maximize our acceleration signal by placing the sensor on the head of the guitar. Besides an amplified movement-to-acceleration ratio, placing the sensor on the head of the guitar makes the sensor transparent to the user, since the head is seldom touched during a performance. Placing the sensor on the head is intuitive for the user, since most guitarists already tilt and shake their guitars during performances.

Considering how the guitar is operated, we matched the sensor axes with the guitars natural axes. An image of the accelerometer orientation is shown in Figure 4 below:

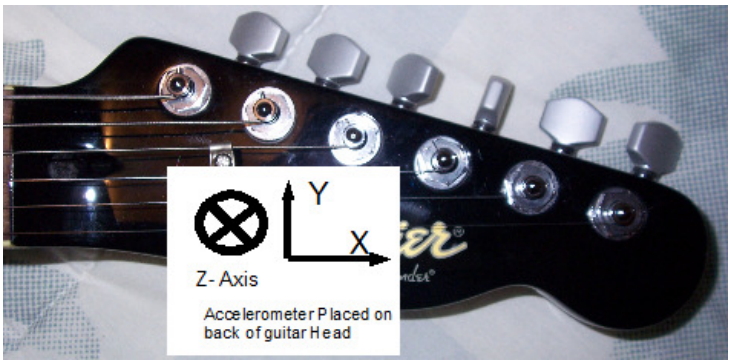


Figure 4 Placement of accelerometer

(jtxdriggers, 2004)

5.1.2 Package

The DE development board can be easily soldered to a thru-hole break-out PCB. For early prototyping purposes, this board is mounted in a DIP socket. It is crucial that the accelerometer is securely fastened (Texas Instruments, 2004). This requirement forces us to solder the development board directly to the break-out PCB as the accuracy requirements increase.

The entire accelerometer sensor will be housed in a standard electronics enclosure, which can be purchased through many local hobby shops. This enclosure will provide protection while allowing the accelerometer to be fastened to the head. The PCB will be mounted within the enclosure using machine screws and stand-offs, such that breakout PCB is lifted yet secured. The enclosure will then be mounted to the head using a combination of double sided tape, and zip ties. This redundancy will guarantee that our sensor is securely fastened to the guitar. A rough hand sketch of the package is shown below in Figure 5

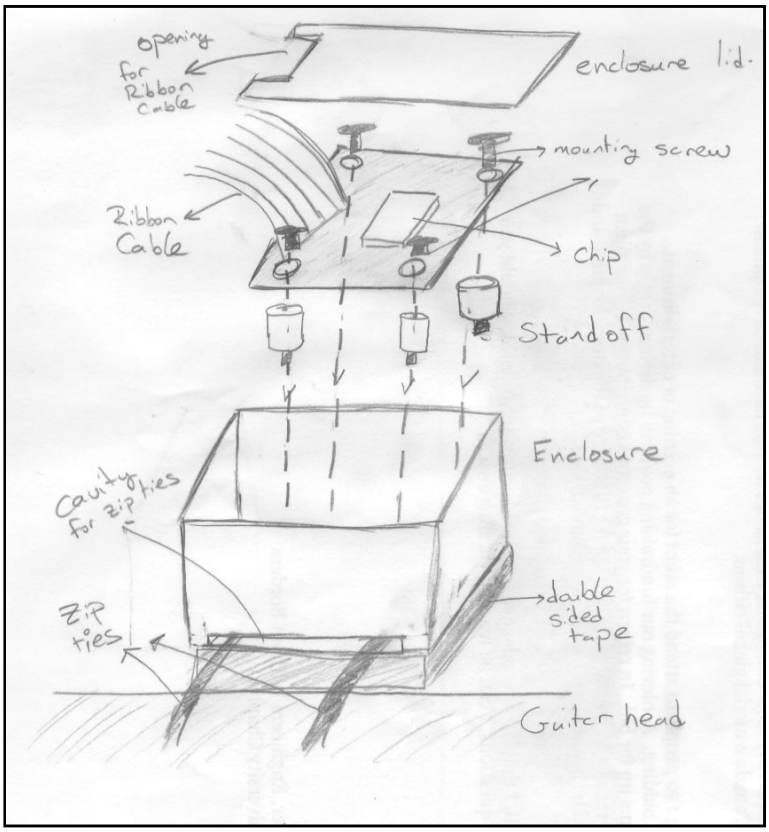


Figure 5 Accelerometer Enclosure

5.1.3 Cabling

In order to reduce clutter and improve signal fidelity, the accelerometer will connect to the microprocessor using an 8-channel ribbon cable. The ribbon cable configuration is shown in Figure 6 below:

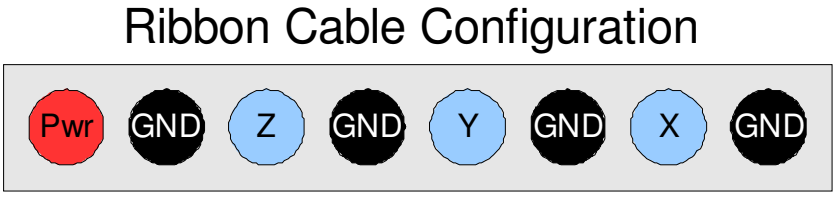


Figure 6 Ribbon Cable Configuration

Isolating each signal with ground paths will reduce the crosstalk. This configuration also reduces the conductor loop area, thereby minimizing magnetic pickup. If noise/precision becomes an issue, we can incorporate a larger ribbon cable and reduce crosstalk by placing 2 ground channels between each signal

channel. We can also reduce the capacitive pickup by wrapping the ribbon cable in a grounded conductor like tinfoil.

The ribbon cable will run along the guitar strap since the strap already connects the user to the guitar head. This will result in an unobtrusive connection between the accelerometer and the pre-processor.

5.1.4 Power

A major advantage of the DE package is the on-board power regulator. The power regulator has an operating voltage of 3.5-15V(Dimension Engineering, 2008) implying that we can power the accelerometer using the microprocessor voltage plane.

It is important to note that the operating voltage of the regulated accelerometer is a unipolar 3.33 volts. Matching voltage with the ADC is desirable since it will maximize our readout precision. Unfortunately, we are using the ADC V_{ref+} pin of the microprocessor for the FSR sensors, with no pin flexibility. Therefore, the ADC operating voltage is defined by the operating voltage of the PIC and should be set to the minimum 4.2 volts (Microchip, 2007). This solution maximizes our precision by not introducing additional amplify-and-offset circuitry.

5.2 Signal Processing

With three orthogonal sensor axes each capable of outputting a DC and AC value, the follow-up question is: what does that information do? The signal processing must account for issues like aliasing, noise, scaling and precision; making it a crucial component of *RockIt*.

For our prototype system, we will implement 2 motion-to-music processes: tilt sensing and dynamic sensing. Each of these processes is discussed in the following sections. A generalized signal processing flow diagram for each sensor is shown in Figure 7 below. The sole difference between the tilt sensing and the motion sensing is the last 2 blocks of the signal flow diagram.

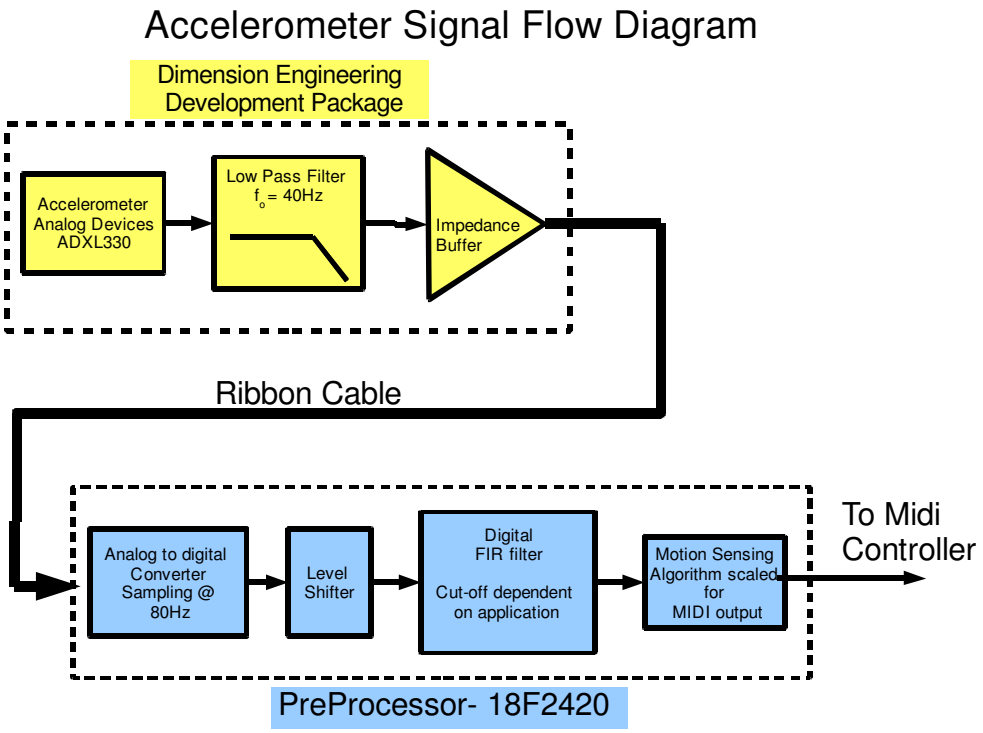


Figure 7 Accelerometer Signal Flow Diagram

As Figure 7 shows, we will not require additional signal conditioning circuitry. Although shown in Figure 7, it is important to highlight the following details:

- 40Hz Low-pass analog filter implemented with bypass capacitors on the DE dev. board
- ADC samples at 80Hz
- Human motion has a maximum frequency component of 12Hz (SensorWiki, 2008) therefore we are over-sampling the desired signal
- Further bandwidth reduction and tuning will be implemented in the digital domain

It should be noted that sensor output should be scaled as a percentage of the voltage input for maximum accuracy. (Texas Instruments, 2004)

Also, the maximum possible precision using MIDI is 7 bits, and therefore all our signal processing should not exceed this limitation. From the perspective of signal processing, every effect output only has 2 parameters: On/Off (defined by the touch sensors), and value between 0-127 (defined by our sensing algorithms).

5.2.1 Precision and Bandwidth

A discussion of precision, noise and bandwidth must be presented, as these factors determine the limitations of the *RockIt* motion-sensing system. As defined in the accelerometer specification sheet (ADXL330, 2007), the worst case noise density is 350µg/$\sqrt{Hz}$. Therefore, using an analog bandwidth of

40Hz, along with peak-to-peak versus RMS ratio of 4:1 (which corresponds to the noise exceeding the nominal peak-to-peak only 4.6% of the time), we can calculate the RMS noise as defined in(ADXL330, 2007) to be:

$$Rms\ Noise < \mu g > = \ 4 * Noise\ Density * \sqrt{BW * 1.6} \quad (1)$$

Using equation 1, we have calculated the noise strength to be 11.2mg. Using the accelerometer sensitivity of 333mV/g (Dimension Engineering, 2008),we can transform the RMS noise to a corresponding voltage of 3.73mV.

In terms of tilt sensitivity, it is important to highlight that the acceleration-to-tilt relationship is non-linear, since gravity is projected onto the measurement axes. Practically speaking, the acceleration-to-tilt ratio is a minimum when the measurement axis is parallel to gravity. In this worst case scenario, we expect one degree of tilt to correspond to approximately 15mg of acceleration (or 5mV). (Texas Instruments, 2004)

Lastly, the voltage sensitivity of the PIC 10-bit ADC operating at 0->4.2V is 4.08mV.

The figures presented above highlight the importance of properly selecting our bandwidth. Unfortunately, the RMS noise is approximately equal to our worst-case readout sensitivity and our ADC capabilities. Therefore, we must reduce our bandwidth in either the analog or digital domain. Considering that our microcontroller possesses an 8x8 bit hardware multiplier, we have opted to reduce this bandwidth using finite impulse response (FIR) digital filters. These filters are easily designed using the MATLAB DSP toolbox. Using a FIR solution reduces our circuit complexity; and allows simple bandwidth modification during testing and optimization.

A digital low-pass filter with a bandwidth of approximately 0.7 Hz would correspond to a 4:1 RMS noise of approximately 0.5mV. We believe that this value would give us sufficient accuracy to read 1 degree of tilt in the worst-case scenario.

For the types of effects we wish to control with dynamic motion (wah-wah and delay), we are confident that we can easily reduce the MIDI accuracy to 5 or 6 bits. Reproducibility is crucial for any musical performance, and we believe the maximal 7-bit precision could compromise this requirement. Referencing the frequency of human movement, (SensorWiki, 2008) we will design a band-pass filter with cut-offs at 1Hz and 15Hz. DC rejection will negate any effects of gravity being projected onto our measurement axis and therefore will make the system more robust against un-wanted triggering. Using this bandwidth, we can expect an RMS noise of 2.2mV.

5.3 Tilt Sensing

Because of their change-and-hold nature, effects like volume, pan, and gate decay time are best implemented using the tilt sensors. For the purpose of this discussion, we are only concerned with how

the MIDI value is modulated between 0 and 127. The below discussion relates to the accelerometer signals *after* they have been low-pass filtered.

The images in Figure 8 below shows how the X and Y sensor axis will be used to determine tilt.

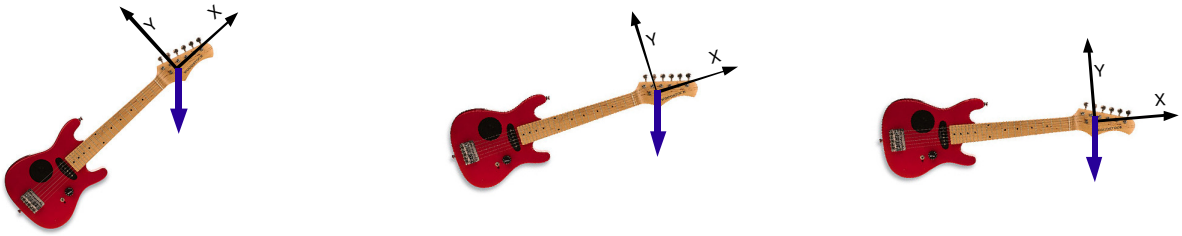
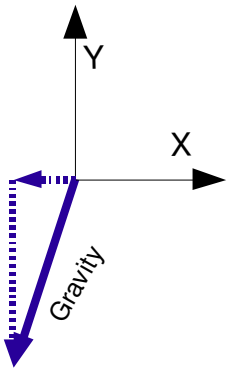


Figure 8 Tilt Sensing Usage

(Time, 2006)

Once the zero-g voltage level is determined (see section: *Possible Issues)*, it is mathematically simple to determine the projection of gravity onto each axis. This is shown in the Figure 9 below.



$$Projection\ of\ gravity_{x,y} = Current\ Voltage - Zero\ G\ voltage$$

Figure 9 Theory for Determining Gravity Projection

Once the projection of gravity onto each axis is known, we will take the arctangent of y/x to find the corresponding angle of the guitar tilt with respect to the x-axis. The MPLab C library supports 2 arctangent functions with a range of [–π, π] and [–π/2,π/2]. For our purposes, the former range is appropriate.

Once the angle has been determined, it will be used in the following state diagram detailed in Figure 10.

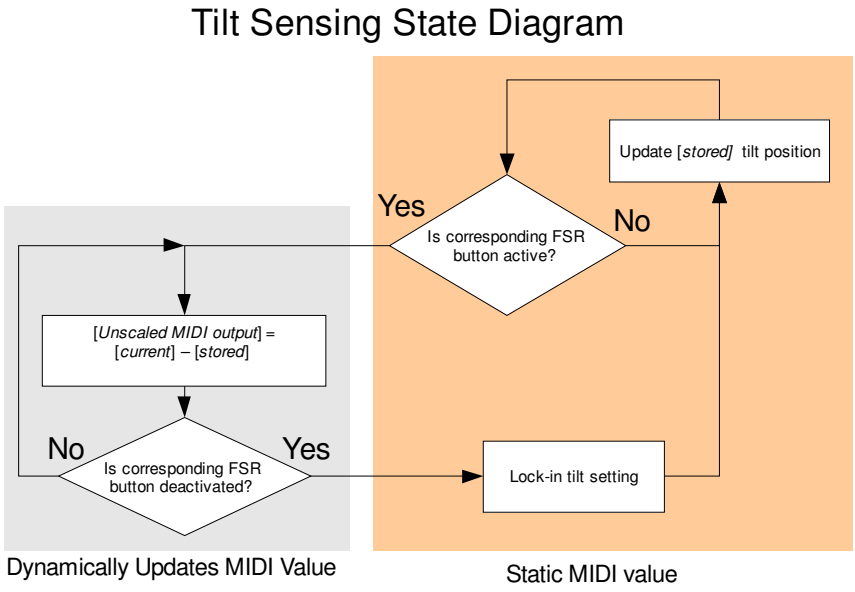


Figure 10 Tilt Sensing State Diagram

For simplicity, certain details in the above diagram have been left out which include:

- Only one conditional statement will be evaluated per sample
- [*stored*] represents the average angle over the past 40 cycles
- [*current*] represents the most recent sampled angle
- [*Unscaled MIDI Output*] represents a value which will have to be processed according to user preference and the 7-bit range of MIDI. Precision in MIDI output might be sacrificed depending on precision of readout, arctangent function, noise, etc.
- *Lock-in tilt setting* implies that the most recent MIDI value will be stored and no longer updated.

The above discussion assumes that gravity will always be in the plane of the guitar. This assumption depends on the mounting of the accelerometer along with the user's posture. This assumption is strengthened by a simple geometrical fact: that the effects of gravity are minimal when the axis of measurement is in plane with gravity. This effect led to the worst-case sensitivity discussed in the *Precision and Bandwidth* section and is discussed in (Texas Instruments, 2004). This fact implies that a small angle between the x-y plane and gravity should have negligible effects on the output. Also, the effects of tilt between the x-y plane and gravity will be mitigated by the common mode rejection, implicit in subtracting the stored value from the current value.

5.4 Dynamic Motion

Effects like wah-wah are generally used in a pulsed fashion. These types of effects are a perfect match for our dynamic motion sensor. Our measurement axis and orientation is shown in Figure 11 below.

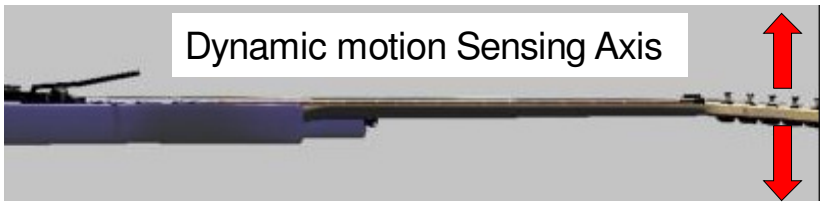


Figure 11 Dynamic Motion Sensing Axis

(Falling Pixel, 2007)

The above diagram shows that rocking the guitar back and forth will modulate the effects. As discussed in the human motion. *Bandwidth and precision* section, the signal will be band-pass filtered with corner frequencies at 1Hz and 15Hz. The lower cut-off frequency should cancel out any influence of gravity while the upper cut-off frequency is chosen to minimize bandwidth while maintaining the ability to monitor

The state diagram for band-pass filtered signals used in the dynamic sensors is shown in Figure 12 below.

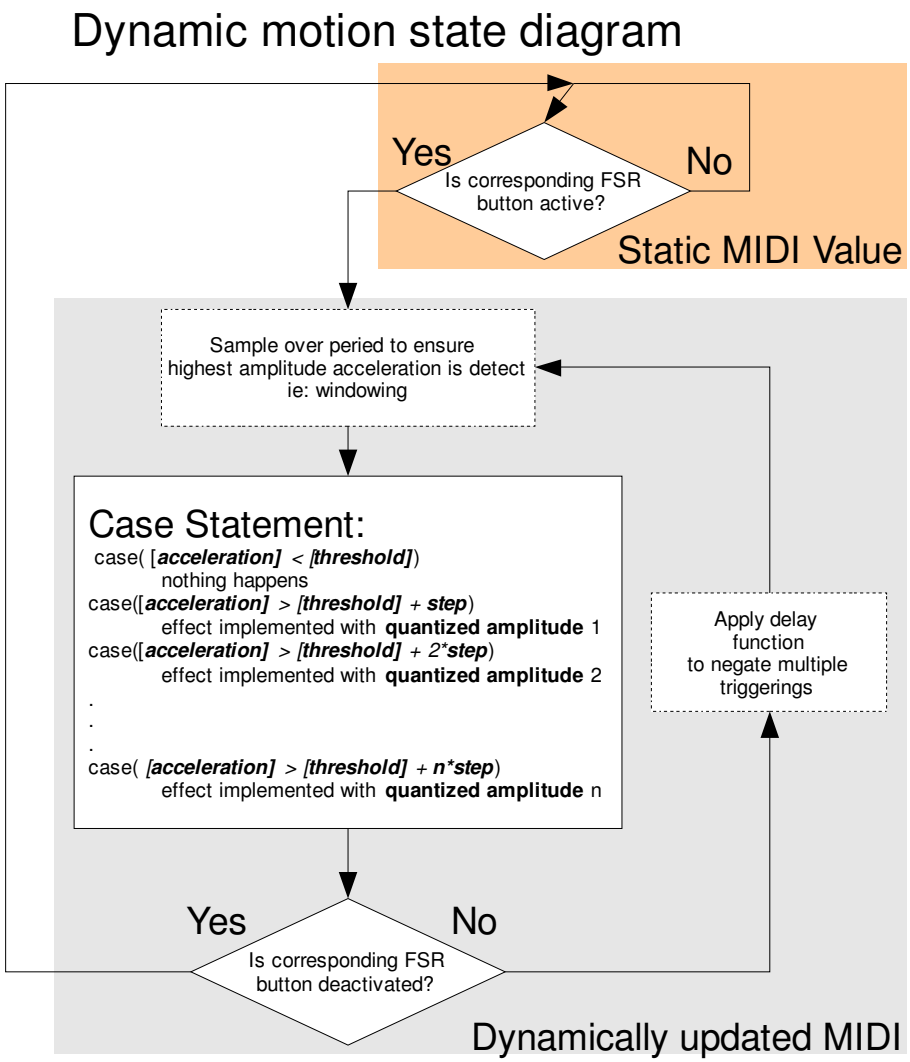


Figure 12 Dynamic Motion State Diagram

For simplicity, certain details in the above diagram have been left out, which include:

- Only one conditional statement will be evaluated per sample
- [*threshold]* will be determined experimentally
- [*step*] and [*quantized Midi amplitude*] and number of steps [*n*] will depend on our detection accuracy and MIDI restrictions
- *[Unscaled MIDI Output*] represents a value which will have to be processed according to user preference and the 7-bit range of MIDI. Precision in MIDI output might be sacrificed depending on precision of readout, arctangent function, noise, etc.
- The dotted boxes are provisions that may be implemented depending on the results of experimentation

The motion sensing algorithm will be robust against the effects of gravity. Even though gravity is normal to the z-axis, we cannot assume that gravity's effects will be negligible. Because the sensors are most responsive to gravity when the z-axis is normal, we must have a bandpass filter to reject effects of gravity (which are represented by a DC value).

5.5 Optimization and Simulation

The algorithms suggested in the tilt and motion sensing sections were motivated by an Analog Devices design reference. (Weinberg, Using the ADXL202 accelerometer as a multifunciton sensor, 2002). Even with these references, many revisions will be made and a great deal of experimentation will be required. This is because music is a very delicate system which requires a good deal of tweaking to get *'just right'*.

Also, algorithm debugging in MPLab is nearly impossible since the program does not support analog signal simulation. Therefore, if the system does not operate as expected, it will be difficult to track the reason for failure.

These reasons have motivated our decision to mimic the motion sensing system in MATLAB. Using MATLAB, we will simulate the accelerometer using stimulus signals which mimic gravity and motion projected onto the measurement axes. This approach is demonstrated in Figure 13 below.

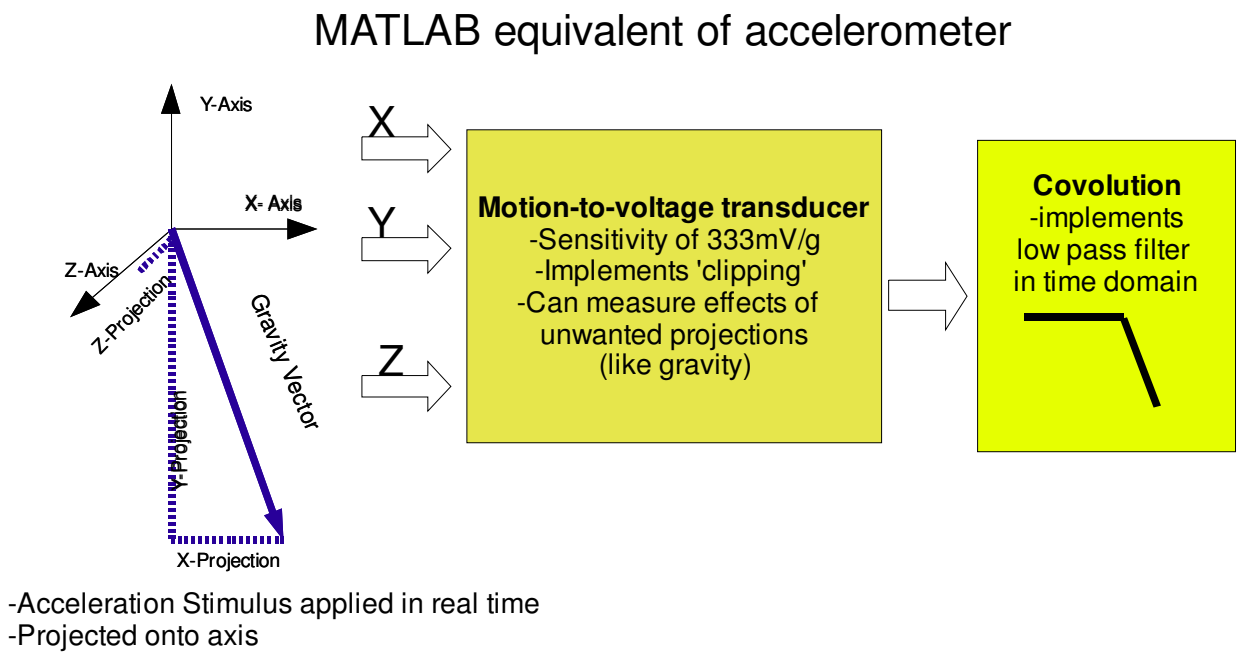


Figure 13 MATLAB Equivalent of Accelerometer

The above functional block will mimic our accelerometer. An analog-to-digital converter is easily created in MATLAB, along with the corresponding FIR filters. Cascading these blocks together while using

MATLAB's animation capabilities will give us an identical platform to the *RockIt* system. The MATLAB simulation platform is shown below in Figure 14

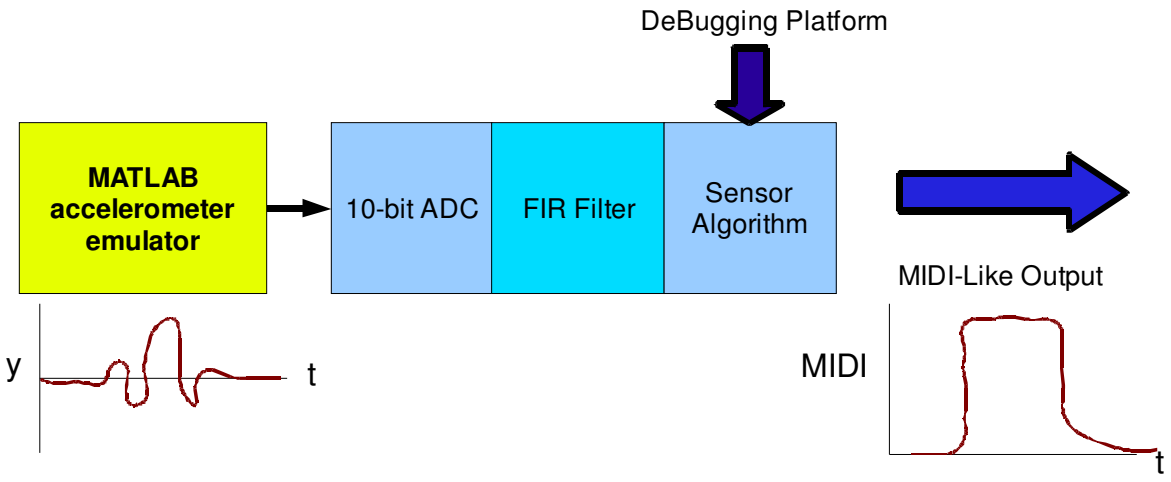


Figure 14 Debugging Platform

By taking advantage of MATLAB's advanced capabilities, we will be better equipped to debug and tweak our algorithms. And since MATLAB and MPLab C18 are both C-based programming languages, codes can be easily ported.

Using MATLAB, we will be able to observe and account for issues like:

- Dynamic level shifting
- Time delay
- Appropriate MIDI scaling
- Off-axis acceleration vectors(like gravity)
- System accuracy

It will be crucial to maintain the limited computational power of the microprocessor when simulating our signal processing algorithms in MATLAB.

5.6 Potential Issues

Besides the issues discussed above, there are a few other problems that might arise during the implementation of *RockIt*. A brief list of possible motion sensing problems is listed below:

- **Temperature drift**: according to the ADXL330 datasheet, the accelerometer can suffer zero-g drift of 1.0mg/^{o}C and sensitivity drift of 0.015%/ ^{o}C.(ADXL330, 2007) Adequate sampling and calibration techniques can remedy this issue.(Weinberg, Temperature Compensation Techniques for Low g iMEMS® Accelerometers, 2007)

- **Acoustic noise:** It is possible that the accelerometer could pick-up the acoustic signal generated by audio equipment. Choosing a very low cut-off frequency limits this effect, and damping foam incorporated into the accelerometer enclosure would solve this issue.
- **Processing Time:** If the motion sensing algorithms become too lengthy to implement in 1/80 seconds, we can: reduce the algorithm accuracy, utilize look-up tables for functions like arctangent, increase the microprocessor clock, or distribute the processing between the Pre-Processor and the MIDI controller.
- **Precision**: If experimentation shows that we need to increase the precision of the accelerometer read-out, we can: improve ribbon cable shielding, increase the operating voltage of the accelerometer or optimize the sensor bandwidth.

It is our hope that none of these issues will become detrimental to our system. We have already considered these issues and how to remedy them.

6 Force Sensitive Resistors

To toggle sound effects, *RockIt* has four FSRs (Force-Sensitive Resistors). Each FSR can be programmed to control four user-chosen effects. An important component of the FSR system is the housing which will be mounted on the guitar for easy-use. Each FSR will drive a variable-force threshold switch circuit with buffered output. This output will connect to the analog I/O pins on the PIC18F2420.

6.1 Theory of FSR

We ordered our FSRs from National Ergonomic Supply Inc., a supplier located in Penticton, BC. The cost of the sensors was comparable between suppliers but we chose to order from National Ergonomic based on their close proximity. Delivery from Penticton to Vancouver was promised within 5 business days. The sensors would allow room for elaboration that a simple toggle-switch cannot. Since the FSRs are thin (~0.46mm) (Interlink, 2008), they can be mounted easily on a guitar without hindering the guitarists' freedom to play.

Force Sensing Resistors are made from a Polymer Thick Film that exhibits a decrease in resistance when the force upon the sensing area is increased. The specified resistance can vary up to 25%. (Interlink, 2008)This is not a problem since we are using the sensors simply as lighter and thinner toggle switches. The fact that these FSRs can detect amounts of variable pressure is a secondary reason for not choosing toggle switches or buttons. FSRs can detect amounts of pressure and this will allow for further elaboration of our product in the future. Each FSR is made of four layers(SensorWiki, 2008)

- A layer of electrically insulating plastic
- An *active area* consisting of patterned conductors
- A plastic *spacer*
- A flexible substrate coated with a thick polymer conductive film, aligned with the active area.

An image of this layout is shown in Figure 15 below:

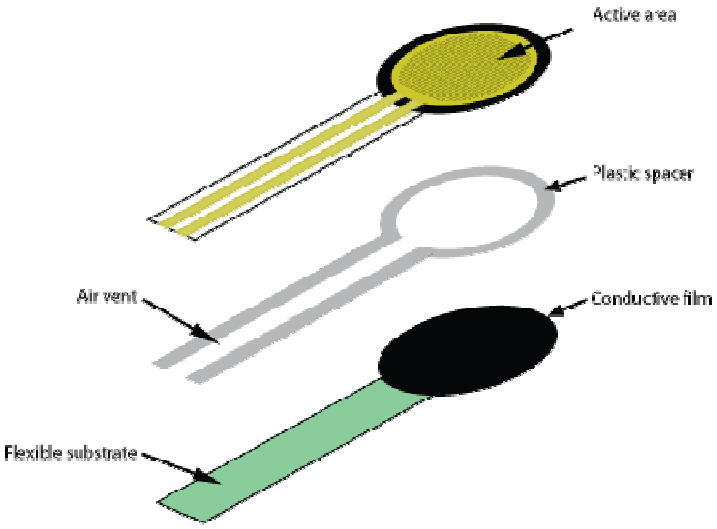


Figure 15 FSR Composition

(SensorWiki, 2008)

The FSR force-to-resistance relationship approximately exhibits a 1/R relationship as shown in Figure 16.

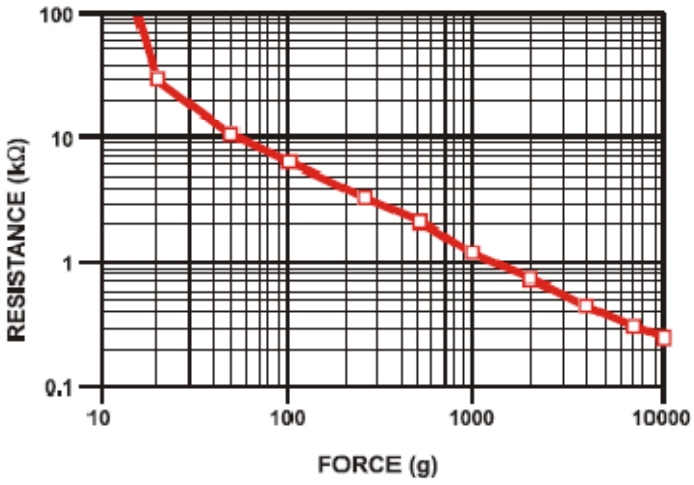


Figure 16: FSR Force to resistance diagram

(Interlink, 2008)

6.2 FSR Detection

Since human touch tends to be inaccurate and inconsistent, we use a threshold detection circuit that only detects actual 'touching', with a Schmitt Trigger circuit. The conceptual schematic is shown in Figure 17.

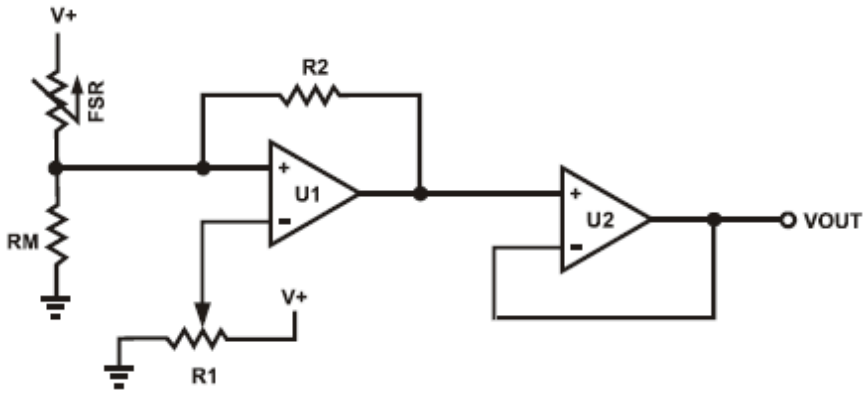


Figure 17 FSR Comparator Simulation Circuit

R2 is a hysteresis resistor and U2 is an output buffer to strengthen the signal and reduce output impedance to an acceptable value. This is a simple comparator circuit with U1 acting as the comparator, driven by the voltage divider to the negative input terminal(Interlink, 2008). The basic functions are outlined below in table 2.

Table 2 FSR Simulation Comparator Component Details

U1 outputs high or low (0V or 5V)
U1 and U2 are LM324N op-amps
The parallel combination of R2 and RM is approximately 47kΩ
R1 is the threshold adjustment sink
R2 is the hysteresis resistor

To date, we are testing our system with this external circuitry but hysteresis and V_{ref} will be handled internally by the microcontroller. These details are further outlined in the "Algorithm" section. The general FSR subsystem signals will operate as illustrated in Figure 18.

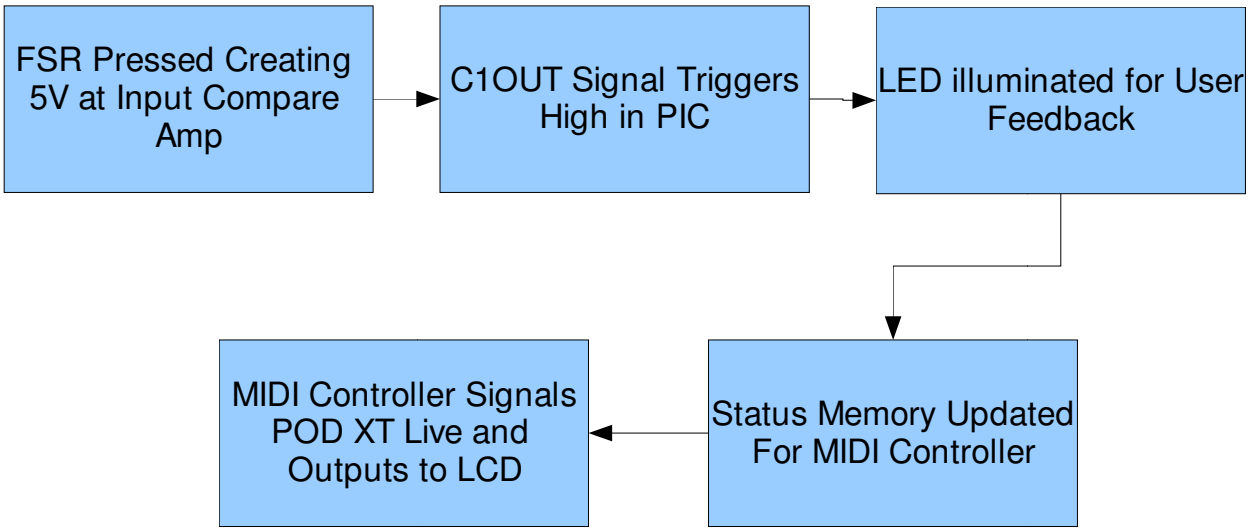
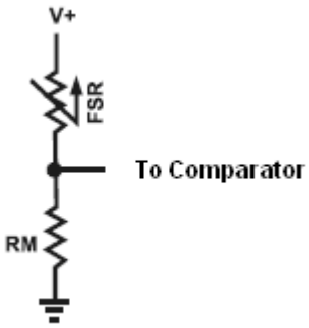
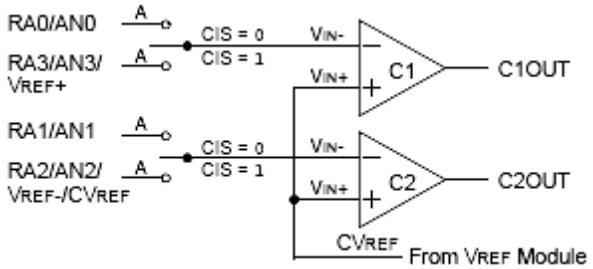


Figure 18 FSR Subsystem Signal Flow Diagram

6.3 Algorithm

The touch detection will be sampled at 80Hz. Because of the nature of the comparator input to the PIC, this frequency is reduced to 40Hz. This is because our PIC can receive four inputs for comparison, but not without cycling the Comparator Input Switch (CIS) which acts 4-2 MUX. This is shown in Figure 19. V_{ref} can be set in software, and is currently defined as 3.125V.





(Microchip, 2007)

Implementation of FSR Circuit PIC internal Comparator Circuit

Figure 19 PIC comparator and external input schematics

At the analog input pins (RA0 to RA3), the voltage source impedance should not exceed 10k (Microchip, 2007) Thus, when selecting R_M, the value will be less than 10kOhm. The V_{ref} and hysteresis are handled in the C source code. However, our PIC does not have the hysteresis R_2 resistor as illustrated in Figure 17. Therefore, we implement a counting scheme within the Timer ISR. The FSR sampling algorithm is detailed in Figure 20.

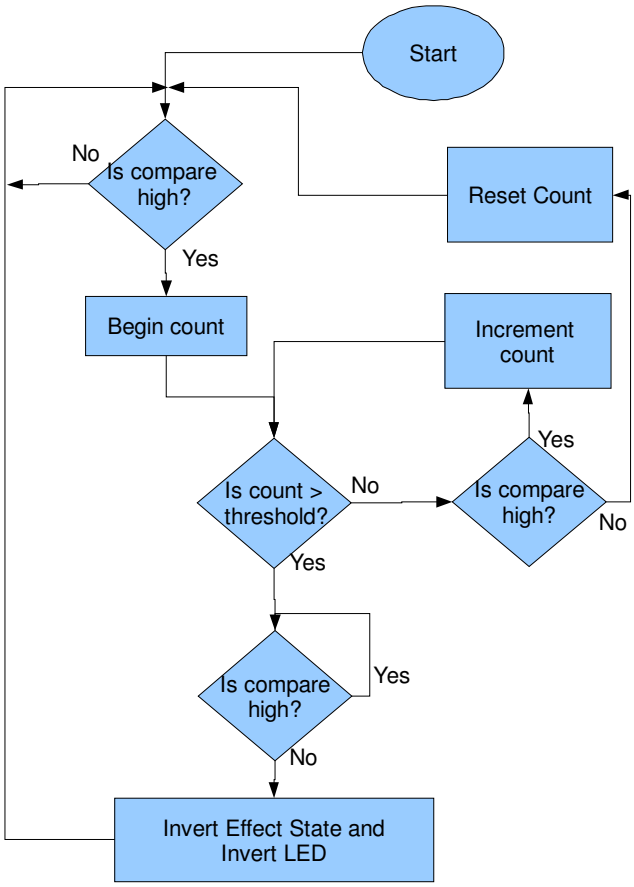


Figure 20 Flow Chart of FSR Timer Interrupt Algorithm

Figure 20 illustrates the polling algorithm for one FSR but in reality, exists in the source code as being capable of handling four FSRs. The basic idea of the algorithm is to toggle pre-set effects with each push of the FSR. If the FSR is held, no action will occur until the FSR is released; concurrently, the FSR needs to be pressed for a time greater than a set threshold. It is necessary to check the output of the comparator for an input 'high' and subsequently an input 'low' for reducing hysteresis. If the FSR is not pressed for a set time and subsequently released, the algorithm will revert back to its constant polling stage. It is important to note that the threshold count is equivalent to 0.05s. Further testing is being done to determine the optimal threshold count.

6.4 Physical Package

The four FSRs will be packaged in a casing which is mounted on the face of the guitar body as shown in Figure 21. Within the FSR housing will be the connections from the FSRs to the ribbon cable. The ribbon cable will be extended from the housing to the pre-processor enclosure located on the guitarist.

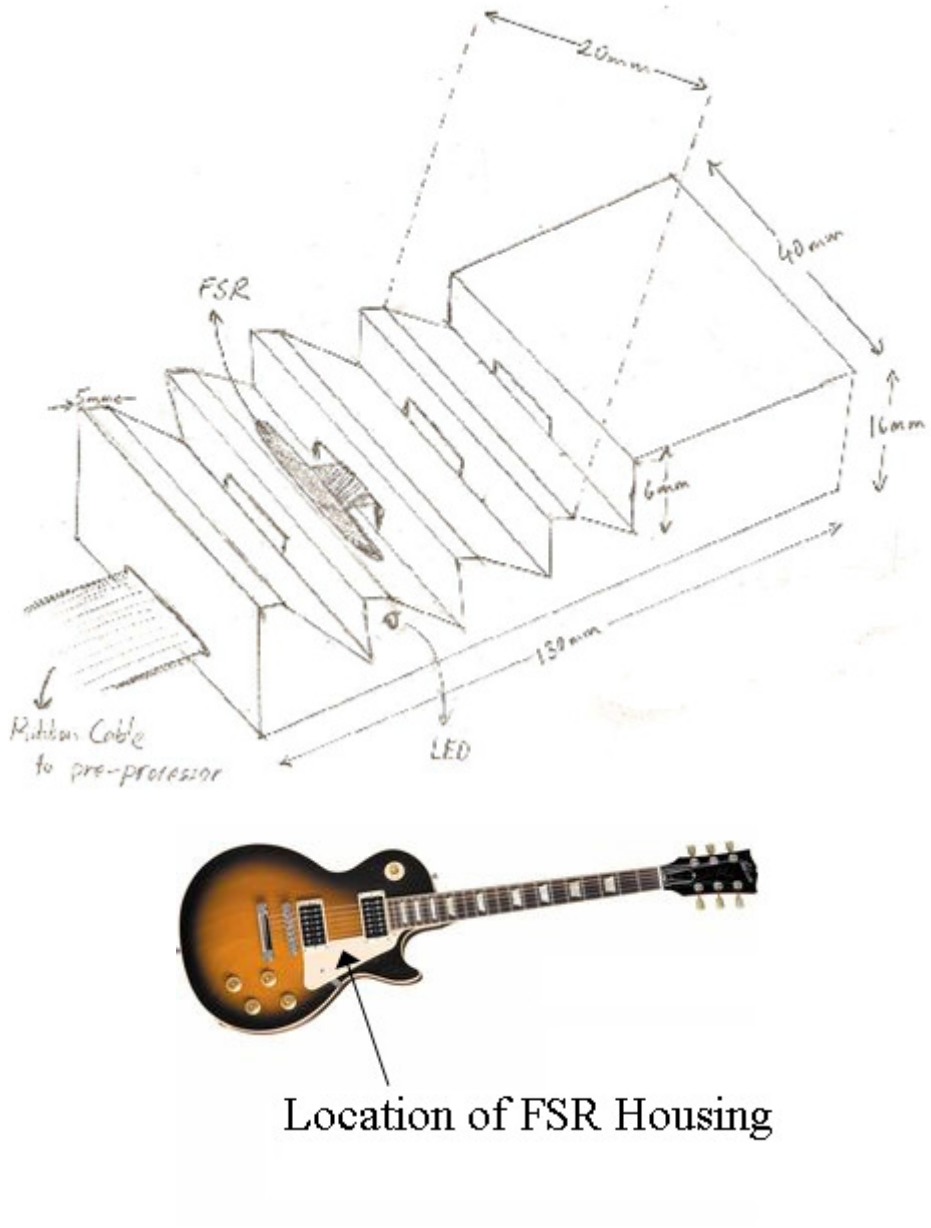


Figure 21 FSR Housing and Location

(MusiciansFriend, 2008)

The FSR enclosure is specially designed for streamlined fitting on the guitar. It is thin enough to fit on the guitar without impeding the musicians' ability to play.

6.5 Potential Problems

A list of potential issues related to FSR usage is listed below:

- **Breakage** The radius of curvature is very small and allows for great flexibility. The maximum radius of curvature is approximately 2.5mm (Interlink, 2008). If the tail is bent too far, the conductive leads inside the active area can break and the air vent could deform (SensorWiki, 2008) It is recommended that the sensor be mounted on a firm and flat surface; this is no problem for our intended use but limits the FSRs in functionality of other potential designs if placed on fingertips, for example. It is important to note that the tail can be bent, but it cannot be creased or kinked (SensorWiki, 2008).
- **Repeatability** With varying resistance from sensor to sensor, high repeatability and accuracy can be a problem. For our use as a threshold detect, these manufacturing variations should not be noticeable especially due to the lifetime of the sensors (approximately 10 million actuations) (Interlink, 2008)
- **Adhesives** It is not recommended to solder directly to the pins of the FSR. The heat could melt the substrate. It is also not recommended to use cyanoacrylate adhesives such as Krazy Glue because the substrate could degrade(Interlink, 2008).

7 Microcontroller Communication

The communication between the pre-processor and the MIDI-controller is achieved through UART. This one-way data transfer allows the pre-processor to send the current status data of the sensors to the MIDI-controller. The simplicity of this communication allows us to avoid hand shaking and therefore speed up the data transfer. Even though the communication between these two microcontrollers in the proof of concept is wired, having an asynchronous communication will open the possibility to implement a wireless connection between the pre-processor and the MIDI-controller in the commercial design.

7.1 Communication Protocol

A set of rules has been developed in order to characterize the format of the data frames transmitted from the pre-processor to the MIDI-controller. This communication protocol assumes that the transmission link uses UART.

7.1.1 Baud Rate

The baud rate is set to 4800 and which is supported by the microcontroller hardware UART. This baud rate is extensively supported by wireless link modules.

7.1.2 Frames

Each frame is 10 bits long and uses a logic low and logic high as starting and stopping bits respectively. Idle state is represented by logic high in order to make this protocol consistent with the MIDI protocol used to interact with the stomp box. Figure 22 presents this frame format.

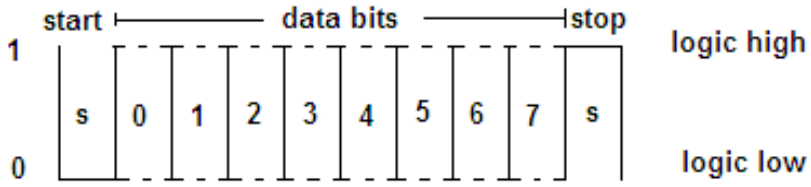


Figure 22 Frame format

This figure will be referenced throughout this section as a naming convention for the different bits in a frame.

7.1.3 Packets

A packet contains the information required by the MIDI-controller to generate a MIDI messages. There are three types of packets: FSR, Accelerometer and Reset. While the FSR and accelerometer packets contain updating data for the sensors, the reset packet is generated every time users want to turn off all the effects that are currently being implemented. The following table shows the logic used to identify the packet type.

Data bit	***Name***	***Logic high***	***Logic low***	***Packet***
0	ACC	ACC packet	Reset or FSR packet	All of them
1	RESET	RESET packet	FSR packet	FSR and RESET

Table 3 Framing Bits

7.1.3.1 FSR packet

FSRs packets are one frame long and contain status information about the four FSRs. Data bits 2,3,4 and 5 contain the status of the four FSRs. Logic high indicates that the FSR is on, while logic low indicates that the FSR is off. The remaining data bits in the packet are unimplemented. The following figure shows the format of the FSR packet.

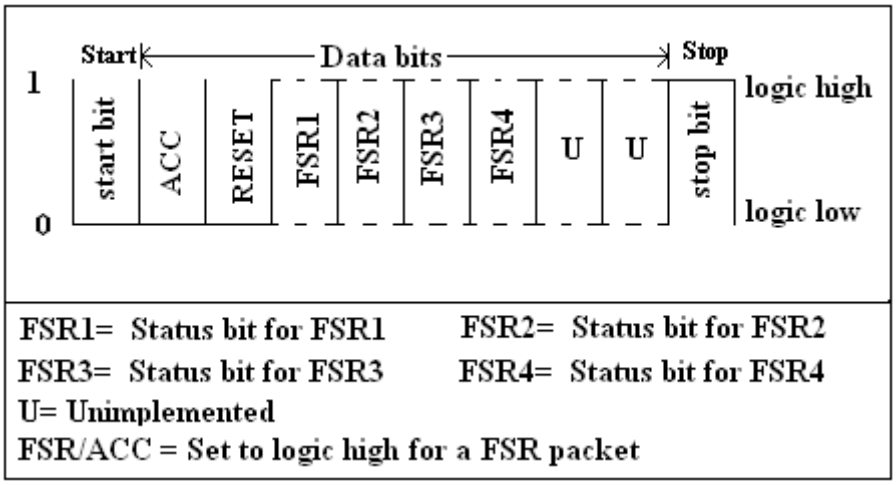


Figure 23 FSR packet

7.1.3.2 Reset packet

The reset packet is sent to the MIDI-controller to suppress all the sound effects that are being implemented. The following figure is the format of the reset packet

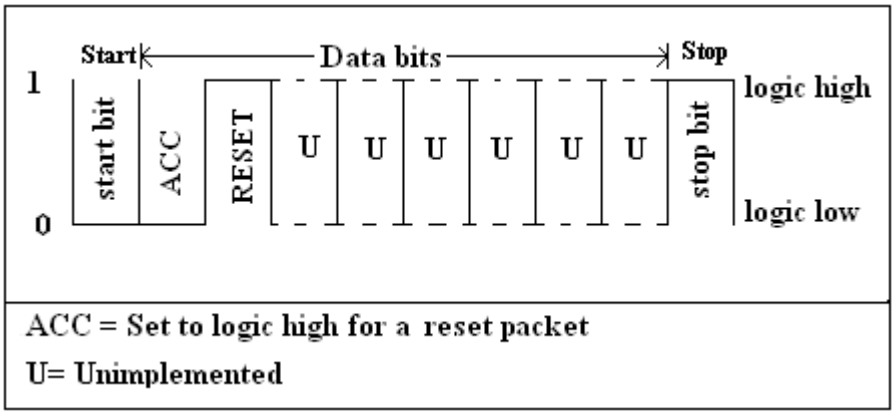


Figure 24: Reset Packet

7.1.3.3 Accelerometer packet

The format of the accelerometer packet is more elaborated due to the higher complexity of the status data. The data bits in this packet contain the MIDI control amplitude (a number between 0 and 127) for either: the tilt in the x-y plane, or the position in the z-axis. Since the precision of the MIDI control amplitudes is 7 bits, the accelerometer packet requires two frames. The first frame contains header bits and the second frame contains a MIDI value between 0 and 127.

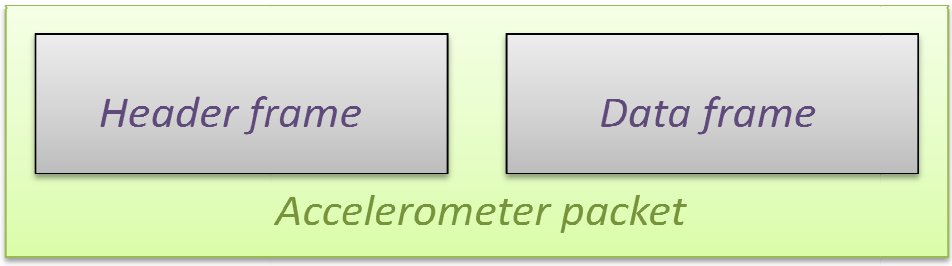


Figure 25 Accelerometer packet

7.1.3.3.1 Header frame

Data bit 1 in the header frame is named 'T/Z' and is used to indicate if the data frame contains a MIDI control amplitude of the tilt or the position in the z-axis. If set to logic high, the T/Z bit indicates that the MIDI control amplitude corresponds to the tilt in the x-y plane; otherwise it corresponds to the position in the z-axis is being updated. The remaining 6 bits in this header frame are unimplemented.

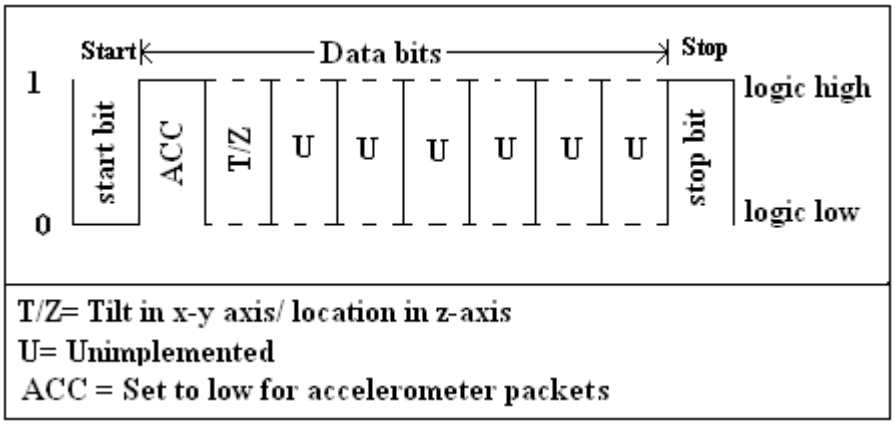


Figure 26 Header frame in Accelerometer packet

7.1.3.3.2 Data frame

The data frame contains the MIDI control amplitude defined by the header frame. Data bits from 0 to 6 are used to represent this amplitude and data bit 7 is unimplemented.

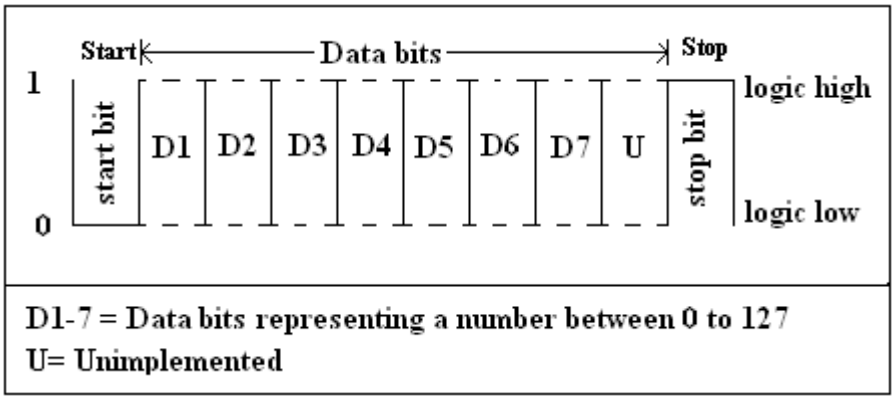


Figure 27 Data frame in accelerometer packet

7.2 Hardware

As explained in section 3, the chosen microcontroller has a dedicated hardware EUSART module that can be configured to transmit and receive data asynchronously. Implementing UART with hardware instead of software allows us to buffer data at both the transmitter and the receiver sides. This feature is essential to minimize data loss while achieving a more robust communication for a potential wireless implementation. Another hardware feature of the microcontroller is the capability to auto-awake upon character reception. Auto-awake is essential to reducing power consumption by eliminating the need for continuous polling.

As shown in Figure 28, only one pin is used in each of the microcontrollers in order to facilitate a one-way data transfer. Pins #17 and #18 are the dedicated UART transmitter and receiver ports respectively.

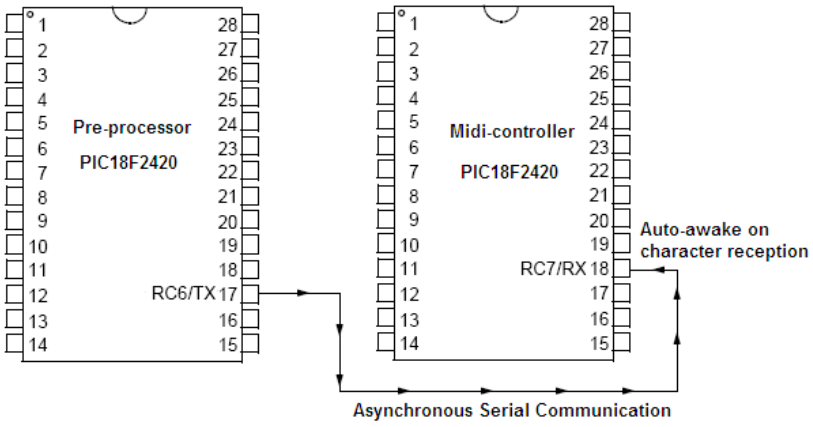


Figure 28 Microcontroller communication

(Microchip, 2007)

7.3 Software

The microcontroller communication is coded using the UART functions library of the Microchip complier MPLAB C18. A state machine is implemented to control the transmission and reception of the packets. The following diagram presents the state machine algorithm of the receiver/MIDI controller side.

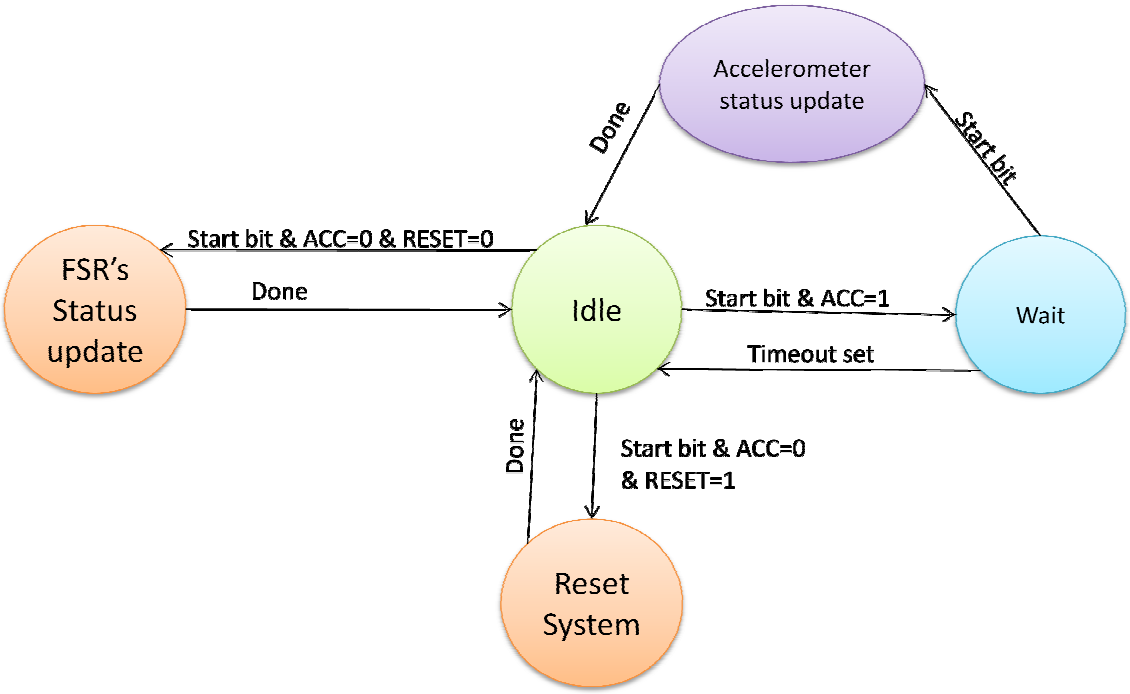


Figure 29 Receiver State Diagram

The receiver stays in idle state until a starting bit is received. Once a starting bit is detected, the ACC bit is checked and if found to be high, the packet contains information about the Accelerometer. The program goes into the "*Wait*" state to wait for the next frame as accelerometer packets are expected to have two frames. A timeout is used in this waiting state in case the second frame is never received. If the timeout is triggered, the accelerometer packet is assumed to be lost and the system returns to the idle state. If a starting bit is received before the timeout is set, then the system goes into the "*Accelerometer status update"*, where the data frame of the accelerometer packet is mapped into a MIDI command. The system returns to idle state once the accelerometer data is processed.

If the ACC bit contains a logic low, then RESET bit determines whether the received packet is a RESET or a FSR packet. After the MIDI-controller handles the reset or the FSR packet, the system goes back to idle state.

8 User Interface

The *RockIt* system is meant to be used with a wide variety of commercially available effects-processors. As there is no universally accepted MIDI signal interpretation, the system must be capable of allowing

the user to select the specific effects (i.e. MIDI control codes) that the sensor signals will control. There are many available devices that will accept MIDI messages as control information, although the interpretation of a given MIDI message may vary widely between different brands and models. Also, since there are many controllable effects available on a device such as the PODxt Live, it is desirable to allow user control of the sensor functionality. Suppose that the musician wishes to control a delay effect for a specific performance; the "on/off" functionality of this effect could be controlled with one of four FSR sensors, while the amount of delay time could be controlled via the tilt of the guitar body. Providing the flexibility to allow a musician this level of control necessitates the development of a simple user interface.

There are 128 different MIDI control codes that may be assigned to the control of a device specific effect. Some of these effects will have a simple "on/off" functionality, while others will be modulated over some range. Because the RockIt system could be used with any effects or DSP module capable of MIDI communications, it will be necessary for the user to specify the MIDI code that they wish to associate with a given guitar sensor. So long as a value between 0 and 127 is specified, the MIDI controller PIC will send the selected MIDI command whether it is recognized by the receiving unit or not. It may be that the selected MIDI control code is not utilized in a certain device and will not result in any action upon receipt. It will be the responsibility of the user to look up device specific MIDI codes in the manufacturer's user manual, and to ensure that they are sending MIDI commands appropriate for a given device. To overcome this lack of standardization, a commercially available version of the *RockIt* system would have a wide variety of pre-programmed settings that would allow the user to select their desired device from a menu. If supported, the unlabelled control codes will be replaced by the name of their device dependant functionality. Should the desired device not be supported, the system will revert to the basic programming method.

Another consideration will be the type of effect the user is assigning a certain sensor to control. For example, volume control is an inherently "analog" type of signal in that it is altered in a series of quantized steps. The *RockIt* system will not prevent the user from attempting to control a volume parameter with an FSR sensor, an inherently "on/off" control mechanism. Rather, the system will simply associate the minimum and maximum volume settings as the "on" and "off" states. Although this type of control will not be particularly practical, it will still generate valid MIDI messages that can be interpreted by a connected device. In this way, the oversight of the user should become readily apparent upon use of the system in this configuration. This basic form of feedback will allow for straightforward analysis and debugging of unintended behaviours. The converse of the previous error may be encountered as well; a user may attempt to control an inherently "on/off" effect with an analog parameter. Since MIDI scale "analog" signals range from 0 to 127 (in integer steps), the MIDI controller will associate values 0-63 with the "off" state and values 64-127 with "on". Again, the error will be easily diagnosed upon use of the system.

The proposed user interface will be a part of the MIDI controller and will consist of an LCD and three buttons. The Modtronix 2S-162YGN LCD has 2 lines at 16 characters per line, and is an ideal candidate

for a basic interface. The LCD can be controlled via I2C, a serial communications protocol supported by the PIC18F2420 and many other peripheral devices. Requiring only two pins, the Master Synchronous Serial Port (MSSP) module found on the PIC18F series can be easily configured to act as an I2C master. User menus will be navigated with a simple, three button interface. Connected to digital I/O pins available on the microcontroller, these buttons will be labelled as “back”, “select” and “next”.

A start menu will appear upon power up to indicate that the device is operational, and will prompt the user to press the “select” button. After pressing “select”, the user will be presented with an available sensor list that will allow the selection of guitar sensors to be mapped to MIDI control codes. Different sensor options can be cycled through by pressing the “back” or “next” buttons, while the “select” button will display the sensor settings screen. Once the user has accessed the sensor settings screen, it will be necessary to select the desired MIDI control code to associate with that sensor. These codes range from 0 to 127 and will be selected by using the “back”, “next” and “select” keys. Should the user attempt to enter an invalid MIDI control code, an error message will be presented and the user will be returned to the sensor settings screen. Once the control code has been properly set, the user will be directed to a confirmation screen where the setting will be verified by pressing “select”. The user will then be returned to the available sensor list, and may proceed to assign control codes to the remaining sensors. All menus will display a blinking cursor to indicate that the device is active and running correctly.

An overview of the menu system is presented in Figure 30 and Figure 31 below. These figures incorporate the actual LCD display, along with the effect of each button being pressed.

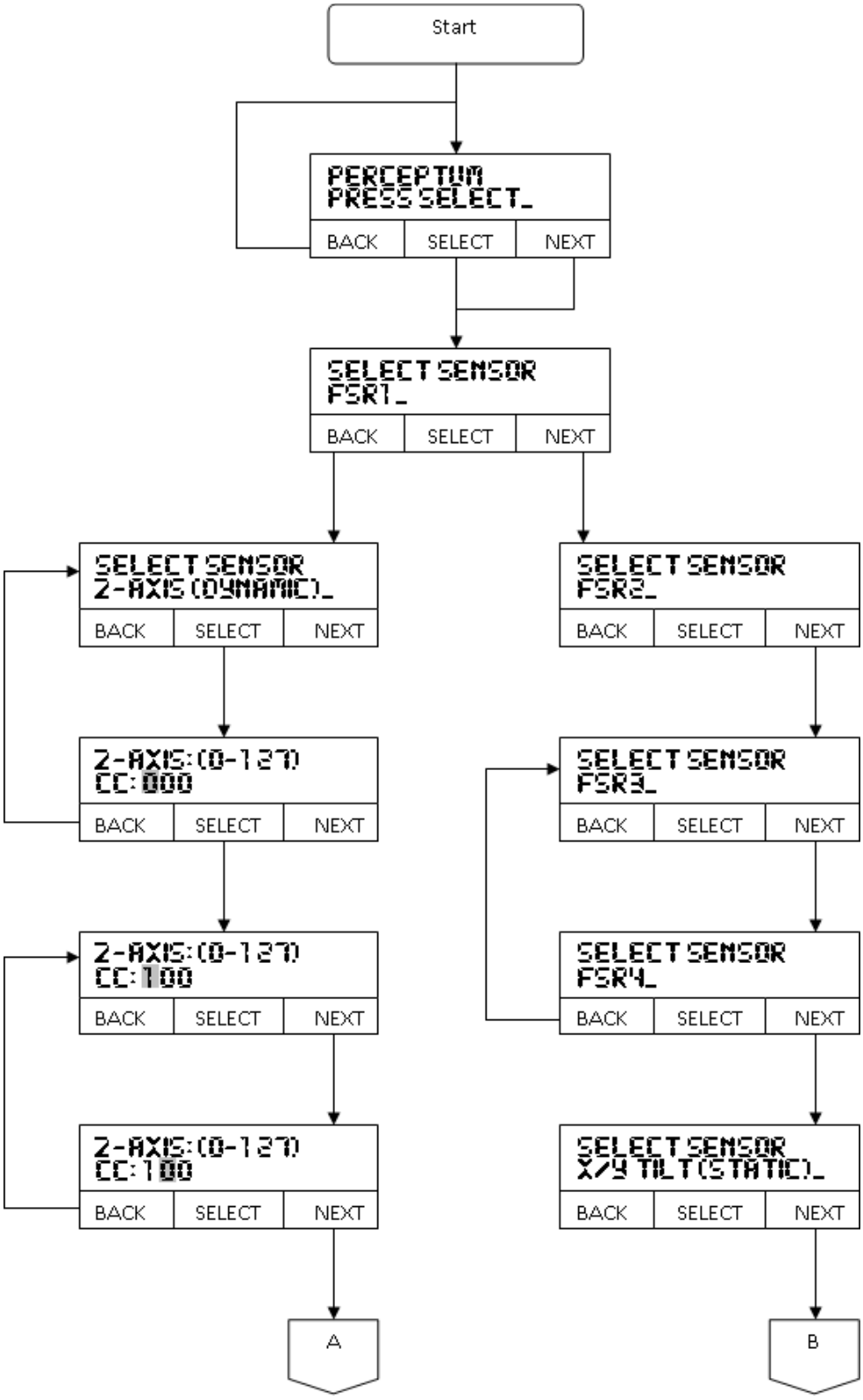


Figure 30: User Interface

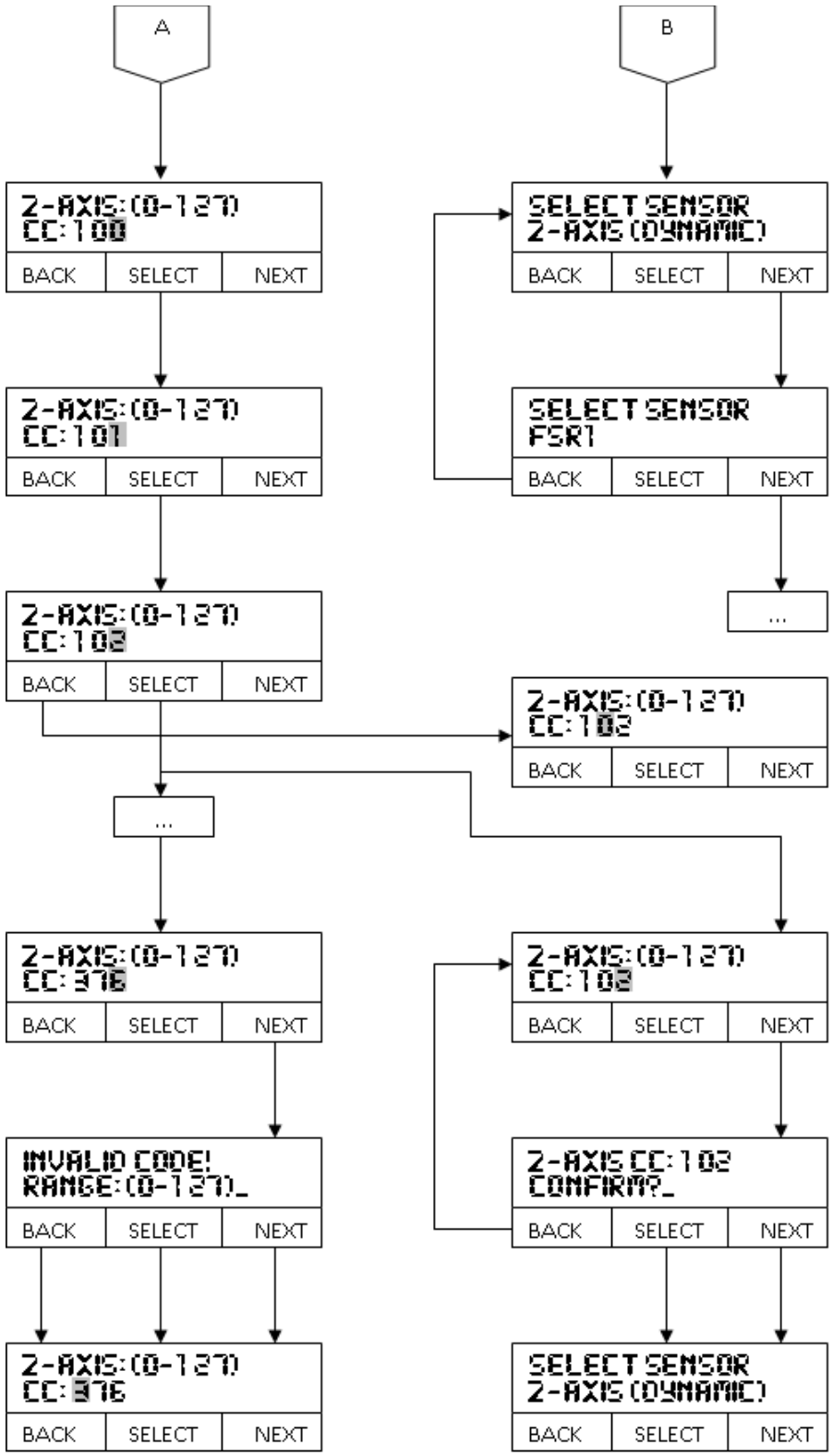


Figure 31: User Interface (continued)

9 Power Management

The power management of the *RockIt* will be broken into three separate sub-systems. Each of these sub-systems will be powered with a dedicated regulator IC that takes its input from the unregulated voltage rail. The unregulated voltage will be provided by a wall mount AC/DC switching converter that can provide a maximum of 1000mA without significant voltage drop.

The first regulator is a 3.3V IC that is included with the accelerometer development board. Since the accelerometer is sensitive to variation in the supply voltage, the use of a dedicated regulator is justified. Both the pre-processor and the MIDI controller will be powered by separate regulators outputting a nominal 4.5V. The reason for this choice of V_{DD} is that the pre-processor PIC should be powered as close to 3.3V as possible to maximize the dynamic range of the onboard ADC. Since the PIC18F2420 cannot be powered by a voltage lower than 4.2V (Microchip, 2007),a value of 4.5V is selected to ensure that the PIC does not enter a “brown-out” state. Although the MIDI controller PIC is not subject to these constraints, a 4.5V regulator is still used to ensure compatible signalling levels between the microcontrollers.

The TK71645 regulator IC has been chosen to provide a 4.5V output from the unregulated voltage. The justification for this regulator choice is that the IC provides a high efficiency with an output voltage that is not strongly dependent upon the current draw.

The power management scheme for RockIt is shown in Figure 32 below

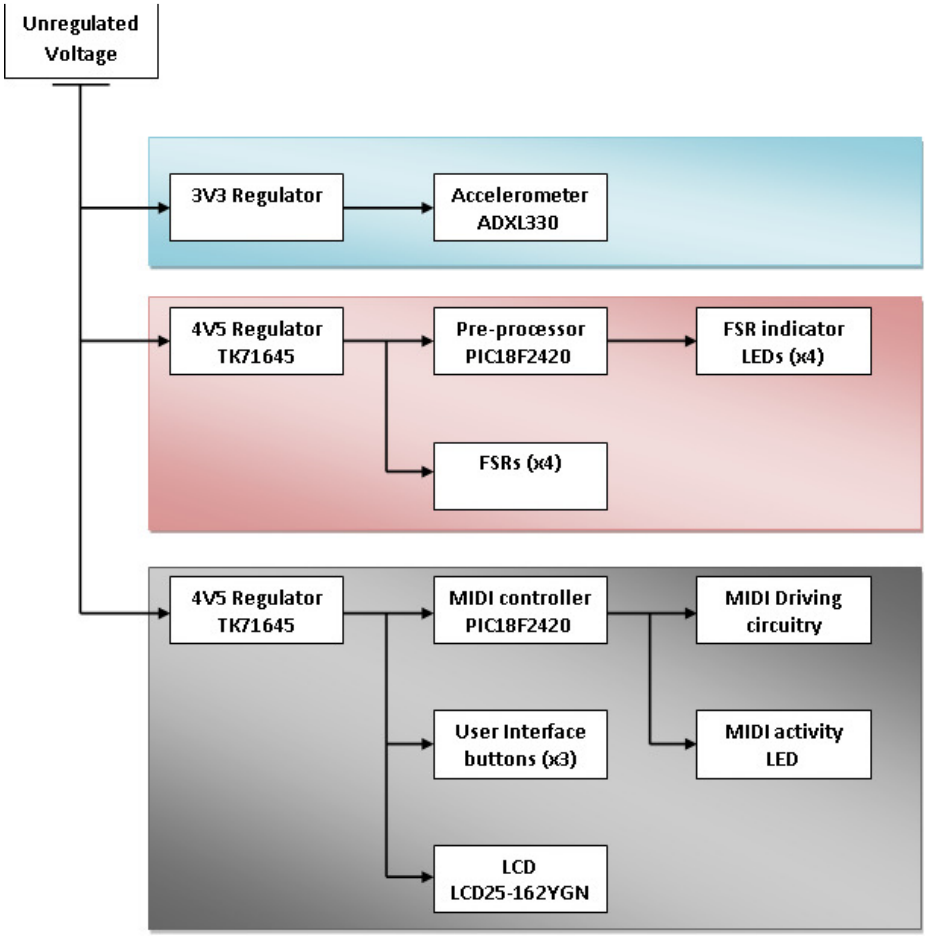


Figure 32 Power Management Diagram

10 Conclusion

The above document outlines our technical design details for implementing the *RockIt* effects system. We discussed our microcontroller choice, MIDI protocol, user interface, FSR sensor, accelerometer sensors, power management and communication protocol. The issues surrounding all these topics have been addressed and possible problems have been identified.

Considering the complexity of our project, unforeseen problems might arise, which require us to change the design outlined in this document.

11 Appendix

11.1 Glossary

ADC: Analog to digital converter

DE: Dimension Engineering, the company that makes our accelerometer development board

FIR: finite impulse response

FSR: Force Sensing Resistor

g: a measure of gravity. 1g represents the earths gravitational acceleration, which is 9.8m/s^2

MIDI: Musical Instrument Digital Interface

I2C: Inter Intergrated Circuit. A 2-wire serial communication protocol

FSR: Force Sensitive Resistor

ICD2: In-Circuit Debugger. Created by Microchip, used to program and debugged PIC microprocessors

12 Bibliography

ADXL330. (2007). *Analog Devices-ADXL330.* Retrieved from http://www.analog.com/en/prod/0,2877,ADXL330,00.html

Borg. (2002). *The MIDI Specification.* Retrieved from http://www.borg.com/~jglatt/tech/midispec.htm

Dimension Engineering. (2008). *RobotShop: ADXL330 development board.* Retrieved from http://www.robotshop.ca/PDF/DE-ACCM3D.pdf

Falling Pixel. (2007). *3D electric Guitar Image Gallery.* Retrieved from http://www.fallingpixel.com/product.php/744

Interlink. (2008). *400 Force Sensing Resistor 0.2" Circle.* Retrieved from http://www.ergo-tech.ca/frame.cfm?ProductID=457&CategoryID=19

jtxdriggers. (2004). *Squier by Fender Master Series Chambered Telecaster HH*. Retrieved from http://jtxdriggers.com/guitar/head.jpg

Line 6. (2007). *POD XT Live Photogallery.* Retrieved from http://line6.com/podxtlive/photogallery.html

Microchip. (2007). *PIC 18F2420 Datasheet.* Retrieved from http://ww1.microchip.com/downloads/en/DeviceDoc/39631a.pdf

MusiciansFriend. (2008). *Gibson Les Paul Classic Electric Guitar with Antique Mahogany Top and Scroll Logo.* Retrieved from http://www.musiciansfriend.com/product/Gibson-Les-Paul-Classic-Electric-Guitar-with-Antique-Mahogany-Top-and-Scroll-Logo?sku=514564&src=3SOSWXXA

Netmedia . (2008). *Netmedia 4X20 ROHS Serial LCD.* Retrieved from http://www.robotshop.ca/home/products/robot-parts/electronique-lcd/netmedia-4x20-rohs-lcd-blue.html

SensorWiki. (2008). *Accelerometers.* Retrieved from http://www.sensorwiki.org/index.php/Accelerometer

Texas Instruments. (2004). Accelerometers and How they work. *Texas Instruments PowerPoint Slides* .

Time. (2006). *Time web shopping 2006 recommendations.* Retrieved from http://www.time.com/time/2006/techguide/shoppingguide/webshopping/kids2.html

Weinberg, H. (2007). *Temperature Compensation Techniques for Low g iMEMS® Accelerometers.* Retrieved from http://www.analog.com/UploadedFiles/Application_Notes/826890745AN598.pdf

Weinberg, H. (2002). *Using the ADXL202 accelerometer as a multifunciton sensor.* Retrieved from Analog Devices:

http://www.analog.com/UploadedFiles/Application_Notes/503243645710974349543215284957 30car_app.pdf