A Prototype Device for Isolating and Wirelessly Transmitting Neural Action Potentials

Eric Christopher Slominski

Thesis Submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
In
Mechanical Engineering

Alfred L. Wicks, Chairmen

William R. Saunders

Charles F. Reinholtz

Robert E. Hampson

December 5, 2003

Blacksburg, Va

Keywords: electrophysiology, instrumentation, bluetooth, microcontroller, action potential

A Prototype Device for Isolating and Wirelessly Transmitting Neural Action Potentials

By

Eric C. Slominski

Abstract

An electrophysiology research laboratory at the Wake Forest University School of Medicine in the Physiology/Pharmacology Department currently carries out memory research by recording neural signals from laboratory animals with a wire tethering the animal to nearby signal conditioning and recording equipment. A wireless neural signal recording system is desirable because it removes the cumbersome wires from the animal, allowing it to roam more freely. The result is an animal that is more able to behave as it would in its natural habitat, thus opening the possibility of testing procedures that are not possible with wired recording systems. While there are wireless neural recording systems in existence, this thesis presents a new approach to recording neural signals wirelessly.

The firings of neurons in the hippocampus are manifested as action potentials or voltage "spikes" on the order of 100 to 400uV in magnitude. Though the information content of the neural signal is riding on these action potentials, the spikes comprise a small fraction of the complete neural signal. A unique feature of the neural signal transceiver presented in this thesis is its ability to digitally isolate and transmit the action potentials, leaving out the remaining, unimportant part of the neural signal. This approach to recording neural signals makes efficient use of the limited bandwidth available with portable short range wireless devices. This thesis will present the spike isolating neural transmitter, which was built using commercially available electronic components. Then, the proper function of assembly language code written for a PIC18F458 microcontroller will be demonstrated. Finally, a discussion of the performance of the neural signal transmitter will be provided.

Acknowledgments

I would like to thank Dr. Hampson and my advisor Dr. Wicks for giving me the very special opportunity to work on my masters project in Winston-Salem, North Carolina at a Wake Forest University School of Medicine Phys./Pharm. lab. This rare arrangement allowed me to work directly with some recognized scientists involved in memory research and gave me access to the resources necessary to set-up a respectable electronics laboratory.

Thanks to Jim Kemerling of Triad Semiconductor for giving much appreciated electrical engineering guidance.

I would like to thank my family. The emotional and financial support that they provided made my transition from undergraduate student to graduate student much more comfortable than it might have otherwise been. I'm sorry I didn't visit more on the weekends, but soon I'll have more time for playing paintball and helping clean up the destruction from Hurricane Isabel.

Recognition is due to Joanne K. Despite her busy schedule, she was always willing to help me with whatever my needs were. Thanks to Lucy F. for being so patient with my many purchase orders, even the ones from Finland.

Finally, I would like to give recognition to the friends that I made during my stay in Winston-Salem. You made my time in WS very enjoyable and provided me with a much needed escape from the stresses of graduate school.

Disclaimer

Elements of the wireless neural recording system described herein are included in provisional patent application no. 60/417,350. The title of this provisional patent application is "Wireless Systems and Methods for the Detection of Neural Events Using Onboard Processing."

Table of Contents

CHAPTER 1 . INTRODUCTION AND BACKGROUND 1

1.1 **INTRODUCTION** 1
1.2 **THE NEURON** 1
1.3 **MEMORY RESEARCH** 3
1.4 **ELECTROPHYSIOLOGY RECORDING SYSTEMS** 4
1.4.1 ***Wired Electrophysiology Recording Systems*** *4*
1.4.2 ***WFUSM Phys./Pharm. Department Instrumentation*** *6*
1.4.3 ***Advantages of Wireless Electrophysiology Recording Systems*** *7*
1.4.4 ***Neuralynx Wireless Electrophysiology System*** *7*

CHAPTER 2 . SPIKE ISOLATING NEURAL SIGNAL TRANSMITTER 9

2.1 **MOTIVATION AND RATIONALE** 9
2.2 **REQUIREMENTS** 9
2.4 **DESCRIPTION OF THE NEURAL SIGNAL TRANSMITTER** 11
2.4.1 ***Neural Signal Transmitter*** *11*
2.4.2 ***Receive and Display Components*** *14*

CHAPTER 3 . COMPONENTS OF THE NEURAL SIGNAL TRANSMITTER 18

3.1 **POWER SUPPLY CIRCUITRY AND CIRCUIT BOARD** 18
3.2 **MICROCONTROLLER AND ASSEMBLY LANGUAGE PROGRAM** 20
3.2.1 ***Data Memory and Program Memory*** *21*
3.2.2 ***The Stack*** *21*
3.2.3 ***Interrupts*** *22*
3.2.4 ***Pointers*** *22*
3.2.5 ***Assembly Language Code*** *22*
3.3 **BLUETOOTH** 27
3.3.1 ***Bluegiga WRAP THOR Bluetooth Module*** *27*
3.3.2 ***Digital Radio Frequency Communication and Bluetooth*** *29*

CHAPTER 4 . PERFORMANCE OF THE NEURAL SIGNAL TRANSMITTER 34

4.1 **MICROCHIP PICDEM 2 PLUS DEMO BOARD** 34
4.2 **BK PRECISION FUNCTION GENERATOR** 35
4.3 **PIC CODE TESTING** 35
4.4 **NEURAL SIGNAL TRANSMITTER TESTING** 37
4.5 **FUTURE WORK** 41

CHAPTER 5. CONCLUSION 43

References *44*
Appendix A *Companies That Provide Neural Signal Instrumentation* *46*
Appendix B *Assembly Language Code* *47*
Vita *58*

List of Figures

Figure 1-1 An artists rendering of a neuron 2
Figure 1-2 Rat with a "hat". 3
Figure 1-3 An electrode picking up neural firings from four neurons 4
Figure 1-4 A typical wired electrophysiology recording system 5
Figure 1-5 Shown is a rat in a testing box with a commutator 6
Figure 1-6 WFU Phys./Pharm. departments electrophysiology recording system 7
Figure 1-7 The transmitter board of the wireless data acquisition system by Neuralynx .. 8
Figure 2-1 What a sampled action potential should look like for given requirements 10
Figure 2-2 Breakdown of the prototype wireless instrumentation system 11
Figure 2-3 The PIC board 12
Figure 2-4 Programming new firmware onto the prototype. 12
Figure 2-5 The Bluetooth board 13
Figure 2-6 Segment of assembly language code for setting up the PICs ADC module. . 14
Figure 2-7 Bluegiga Evaluation Board and PC 15
Figure 2-8 The LabVIEW user interface. 16
Figure 3-1 Picture and schematic of custom circuit board 19
Figure 3-2 Construction of custom circuit board 20
Figure 3-3 A flow diagram of the PIC's assemble language code 25
Figure 3-4 RTS/CTS flow control. 27
Figure 3-5 Digital Radio communication diagram 29
Figure 3-6 The contents of a Bluetooth packet 30
Figure 3-7 A few examples of baseband coding 31
Figure 3-8 Bluetooth protocol stack arrangement 32
Figure 4-1 The Microchip PICDEM2 PLUS Demo Board 34
Figure 4-2 Screen shots of data collected with LabVIEW interface 36
Figure 4-3 Screen shot illustrating buffering of five samples 37
Figure 4-4 Screen shot illustrating where data has been lost 38
Figure 4-5 "File size test" diagram 38
Figure 4-6 A Bluegiga WRAP THOR module with integrated antenna is shown 42

List of Tables

Table 1 The codes that may be used to control the neural signal transmitter 17
Table 2 The possible codes and their effects on the MCU. 23
Table 3 Selected commands from the WRAP THOR ASCII interface users manual. 27
Table 4 The collected data from the "file size test" 40
Table 5 The averages and standard deviations of data collected from the "file size test" 40

Chapter 1 . Introduction and Background

This chapter is intended to give a brief introduction to this thesis, followed by some background information on the neuron and an explanation of memory research and the equipment used to record signals from neurons for the purpose of memory research. Wired and wireless recording instrumentation is introduced, and the motivation for a wireless system is explained.

1.1 **Introduction**

Memory research on laboratory animals at an electrophysiology research laboratory at the Wake Forest University School of Medicine in the Physiology/Pharmacology (WFUSM Phys./Pharm.) Department is carried out by recording the animal's neurological signals using wires that tether the animal to nearby signal conditioning and recording equipment. A wireless recording system is desirable because it removes the constraining wires from the animal, allowing it to move freely, opening the possibility for improved testing procedures. Unfortunately, wireless instrumentation has limited throughput when compared to wired systems and are relatively power hungry, which necessitates the use of a bulky battery.

The information content of the neurological signal is contained in voltage outbursts called action potentials or "spikes". Since these spikes comprise a small fraction of the total neurological signal, the requirements on the radio transmitter would be relaxed if the action potentials were isolated before radio transmission. This method of neural signal recording would reduce the throughput and power required to sample a single channel. This thesis presents a spike isolating neural signal transmitter constructed from commercially available electronic components.

In this chapter, the action potential generating neuron will be introduced. Memory research will be discussed followed by wired and wireless instrumentation used to carry out this research. The advantages of a wireless system will be presented. In Chapter 2, the spike isolating neural signal transmitter will be introduced, starting with a discussion of the motivation and rationale for such a transmitter. In Chapter 2, the discussion will be focused more on the device's overall system level operation. More details of the custom etched circuit board, PIC18F458 Microcontroller Unit (MCU), and Bluegiga Bluetooth module will be given in Chapter 3 where the discussion is switched to the component level. The performance of the spike isolating neural signal transmitter is discussed in Chapter 4. In Chapter 4, the proper operation of the PIC assembly language code is demonstrated and the performance of the Bluetooth wireless link is assessed. Chapter 5 provides a brief conclusion.

1.2 **The Neuron**

Neurons, or nerve cells, are cells of the central and peripheral nervous system that provide the pathways through which the brain's electrical signals are distributed. An artist's rendering of a neuron is shown in Figure 1-1. Though there are different types of neurons, they all have the same basic function. Chemical neurotransmitters are received by the dendrites and have a depolarizing effect on the cell body [1]. When the

resting potential of the cell body surpasses a certain threshold, the neuron "fires". This is known as an "action potential". To fire, the neuron produces a pulse or "spike" of electrical activity and releases some of its own chemical neurotransmitter [1]. The neurotransmitter travels across a tiny gap called a synapse, where it is received by the dendrites of another neuron (see inset of Figure 1-1), which acts to change the potential of another neuron's cell body. Thus the process is repeated. Interestingly, the magnitude of a particular neuron's action potential is always the same, regardless of the amount of depolarization the cell body experiences. This is known as the "all or nothing" principal [1]. For a colorful and detailed description of how the action potential works, see [1].

The billions of discrete neurons of the brain are interconnected with other neurons through the axons and dendrites. In this way, firing patterns propagate throughout the brain and enable the brain to manage complex tasks, such as provide perceptions of the world, control the body's mechanical motion, and enable memories to be stored and retrieved [2].

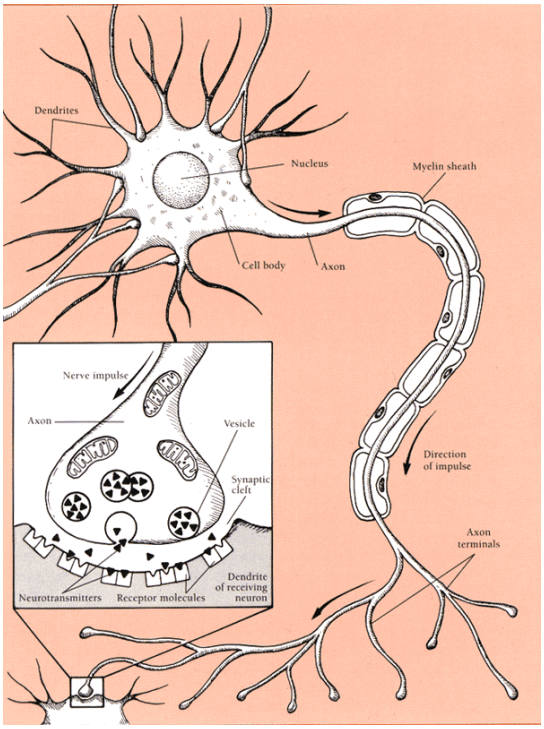


Figure 1-1 An artists rendering of a neuron. The synaptic gap is shown in the inset picture. [4]

The branch of physiology that studies the relationship between electric phenomena and bodily processes is known as electrophysiology [3]. Understanding the mechanisms through which information is passed by nerve cells throughout the brain is the first step in understanding the complex data processing performed by the brain [2]. Memory research is a branch of electrophysiology research in which the patterns of neural action potentials firings are statistically correlated to memory. Memory research is explained in further detail in the next section.

1.3 **Memory Research**

It has long been recognized that the part of the brain known as the hippocampus plays a part in the brains ability to memorize [12]. For this reason, memory research is carried out by recording neural electrical activity of the hippocampus and statistically correlating this activity to memory. Researchers at Wake Forest University School of Medicine are currently involved in electrophysiological memory research using laboratory rats.

In this research with laboratory rats, the animals are placed in a special experimental chamber and are taught certain tasks. The animals learn how to receive a reward, such as water, by memorizing what actions to take based on visual cues. An automated electro-mechanical testing apparatus presents the reward to the animal if the tasks are correctly completed. Once the animal has learned the task, it is fitted with a special hat to make its hippocampal neural signals accessible. Firing patterns of the hippocampal neurons are then recorded while the animal performs the learned task. The action potentials of the recorded neural signal are electronically isolated and an attempt is made to identify the neural mechanisms responsible for memory.

To give researchers convenient long term access to the electrical signals generated by a rat's neurons, the rat must be surgically fitted with a special cap, often called a hat, that allows for a direct electrical connection with the neural signals. To start, the rat is anesthetized. The animal's head is secured and the skull is exposed. Small holes are drilled through the skull over the areas of the brain where an array of electrodes is to be dropped into the soft brain tissue. During the surgery, a micromanipulator is used to make fine adjustments to the depth of the electrodes in the brain, while an audio amplifier amplifies the signal from the electrodes, allowing the surgeon to audibly verify the electrodes proximity to firing hippocampal neurons. Once the array is in place, dental cement is used to carefully seal the array to the skull. Figure 1-2 shows a picture of a rat that has had a hat installed

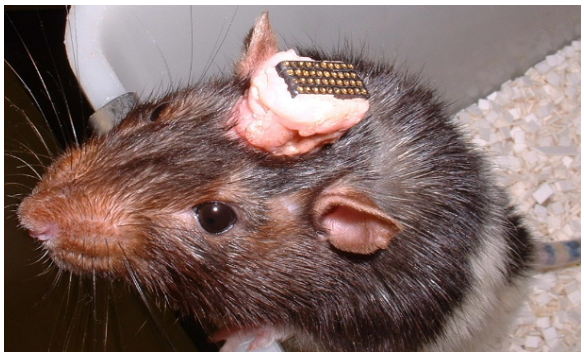


Figure 1-2 This rat has undergone surgery to have his hippocampal neural signals made accessible.

Once the animal has had electrodes surgically implanted, the electrical activity of some of its hippocampal neurons is conducted through the metallic electrodes to the hat, where a connector may be plugged in to access them. A single electrode picks up signals from any neuron in its proximity, as shown in Figure 1-3. Typically, up to four neurons may be close enough to the electrode to have a signal significantly larger than

the noise floor. The shorter the distance between a neuron and an electrode, the greater the amplitude of that neurons action potential. For that reason, and because of the "all or nothing" principle, a particular neuron's action potential will always have the same shape (over the short term), which allows neurons to be individually identified by analyzing the amplitudes of the action potentials picked up by the electrode. We will call this neuron differentiation "spike sorting".

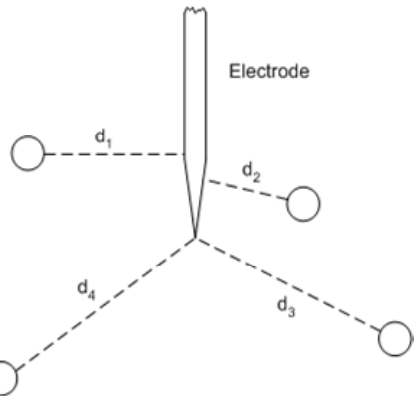


Figure 1-3 The electrode picks up neural firings from any brain cell in its immediate surroundings. The four circles represent four different neurons, each having a different distance from the electrode.

Because the neural signals in their raw form are too small in amplitude to record directly, electronic equipment must be used to filter and amplify the signal before the spikes may be isolated, digitized, sorted, and analyzed. The next section introduces the electronics that are designed for this task.

1.4 **Electrophysiology Recording Systems**

There are several companies around the world that design hardware and software dedicated to amplifying, filtering and processing of neural electrical signals. For a list of companies that design electrophysiology research equipment, see Appendix A. This list is not intended to be comprehensive; rather it is merely a starting point for the interested reader. A laboratory in the Wake Forest University School of Medicine Physiology/Pharmacology (WFUSM Phys./Pharm.) Department uses instrumentation from Plexon Inc.

In this section, general wired neural instrumentation will be discussed first. Then, the WFUSM system will be introduced along with a discussion of the drawbacks of wired systems. Finally, a wireless system from Neuralynx will be shown.

1.4.1 **Wired Electrophysiology Recording Systems**

Shown in Figure 1-4, is a flow diagram of a typical wired electrophysiology recording system. In the arrangement shown in the figure, the rat's hat provides access to multiple channels of neural signals from the animal's brain. A head-stage, or front-end, is plugged into the rats hat to buffer the high impedance output of the rats brain waves

from the low impedance wires. The head-stage may also provide varying amounts of filtering and amplification depending on the particular system. After passing through the front-end, the neural signal is filtered, amplified, digitized, and processed to isolate the spikes. Digitization is the process of turning analog signals into binary format and is performed by Analog to Digital Converters (ADCs). The Personal Computer (PC) may be used to adjust the filtering and amplification characteristics, as well as the parameters of the spike isolation and sorting process. After being processed, the action potentials are then saved to a file and displayed by the PC.

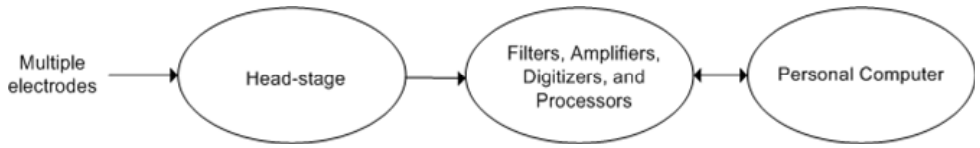


Figure 1-4 In a typical wired electrophysiology recording system, multiple channels of neural data are buffered by a head-stage and lead off the rat through a wire where the signals are, filtered, amplified, digitized, processed, and finally displayed by a Personal Computer.

In a wired system, a wire bundle is used to lead the signals from the front-end to the other electronics in the system. This wire must be prevented from twisting as the animal moves about the test chamber. To prevent a torque from building up in the wire a special rotating electrical connection called a commutator is used. Figure 1-5 shows a rat in an experimental chamber. It can be seen that a commutator acts to prevent the wire bundle from getting twisted. In this picture, the output of the commutator is led to a pre-amplifier (blue box), which is part of the electrophysiology recording equipment from Plexon inc. Instrumentation from Plexon is discussed more thoroughly in the next section, as it is the equipment used in the WFUSM Phys./Pharm. Department laboratory in which I worked.

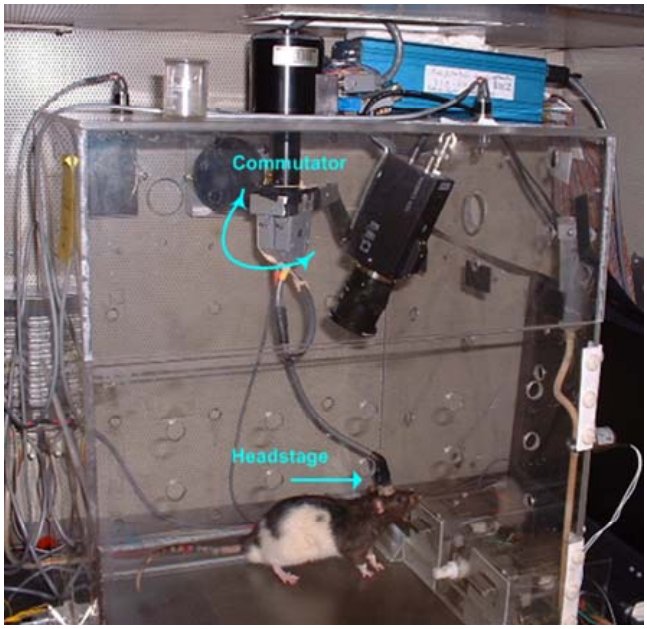
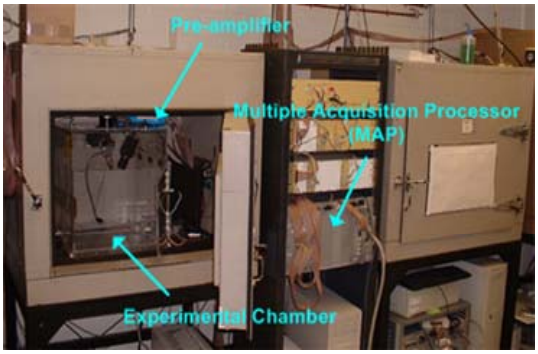


Figure 1-5 Shown is a rat in an experimental chamber. A commutator prevents the animal from twisting the wire bundle.

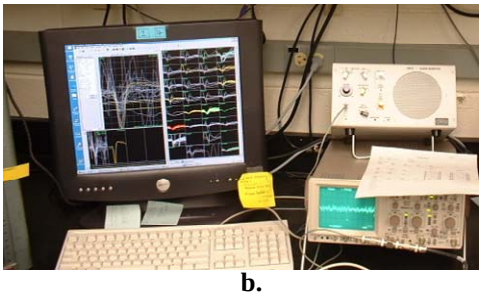
1.4.2 **WFUSM Phys./Pharm. Department Instrumentation**

A research lab in the Phys./Pharm. department at Wake Forest University School of Medicine uses instrumentation from Plexon inc. The system, shown in Figure 1-6, includes a unity gain head-stage, followed by a pre-amplifier, Multiple Acquisition Processor (MAP), and finally the PC. In Figure 1-6a, the testing box can be seen with a Plexon head-stage, pre-amplifier, and MAP. Between the head-stage and the pre-amplifier, a commutator prevents the wire bundle from getting twisted. In Figure 1-6b, the PC and oscilloscope outputs can be seen. Figure 1-6c shows a close-up of an action potential on the oscilloscope display.

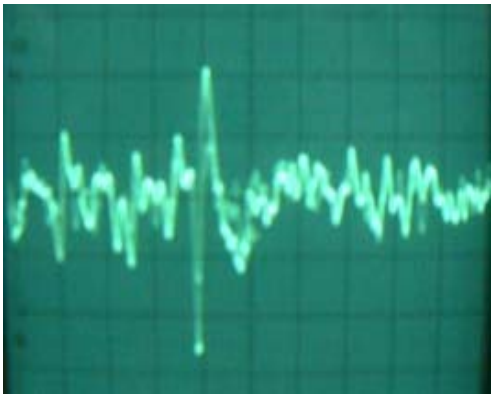
The head-stage consists of operational amplifiers in a negative feedback voltage follower arrangement for unity gain buffering. Next, the preamplifier band-pass filters the signal with a 2-pole high-pass filter and 4-pole low-pass filter and applies a gain of 1000 (60dB). The low-cut frequency is 100Hz and the high-cut frequency is 8kHz. The MAP is a system of modular plug in circuit boards housed in a stand-alone box capable of handling from 16 to 128 channels of neural signals. At the particular Wake Forest University Phys/Pharm department laboratory in which I worked, 32 channels are used. The MAP allows for programmable amplification, filtering, digitization, and real time spike sorting using Digital Signal Processors (DSPs). During the sorting process, up to four separate neurons from each electrode may be identified through the use of adjustable threshold settings. Digitization rates as high as 40,000 samples a second are possible on each channel with 12-bit resolution. Once through the MAP, the filtered, amplified, and digitized signals are led into a computer where the data is saved and displayed.



a.



b.



c.

Figure 1-6 Wake Forest University Phys/Pharm departments electrophysiology recording system is shown. **a.** A testing box and a Plexon pre-amplifier and MAP. **b.** On the PC display is the output of Plexon software. **c.** In this magnification of the oscilloscope display, an action potential can be clearly seen after filtering.

1.4.3 **Advantages of Wireless Electrophysiology Recording Systems**

Wired electrophysiology recording systems have several drawbacks when compared to wireless systems. First, the wire puts geometric constraints on the design of test chambers. Since the wire should never pull too tight on the animals head and must never hang too slack, experimental chambers have to be designed in certain ways. Second, wired systems prevent the animal from tunneling under objects, as the wire would get hung up. Third, the electrical commutator used to provide a torque free connection between the animal and the off-animal electronics inevitably adds noise to the signal. For these reasons, wireless systems are attractive alternatives to wired systems. The use of radio transmitters is particularly attractive for electrophysiological research with birds [13]. There are wireless systems in existence. One such system is made by Neuralynx, inc. The Neuralynx wireless system is the topic of the next section.

1.4.4 **Neuralynx Wireless Electrophysiology System**

The Neuralynx wireless electrophysiology recording system uses analog FM modulation in the 902-928 Mhz band to wirelessly transmit amplified and filtered neural signals. The complete system consists of head-stages, a transmitter board, and a receiver system.

Each head-stage has eight separate channels. Up to four of the front-ends may be used for a maximum of 32 total channels. The head-stages have to provide sufficient amplification to allow the neural signal to be effectively modulated onto a carrier wave and demodulated on the receiving end. The head-stages have a gain of 1500 and are 2-pole band-pass filters with a low cut frequency of 600 Hz and a high cut frequency of 6 kHz.

Once filtered and amplified by the front-ends, the signals plug into the transmitter board, shown in Figure 1-7. In addition to an FM transmitter, the transmitter board has the electronics that multiplex the incoming filtered and amplified neural signal so that they may all be transmitted on a single FM channel. The transmitter is reprogrammable for different numbers of channels through a programming connector on the board. A total maximum throughput of about 1 Mhz is possible with the Nerualynx system (26 kHz sample rate for 32 channels). The transmitter board transmits the entire neural signal and spike isolation is performed on a computer once the signal has been received.

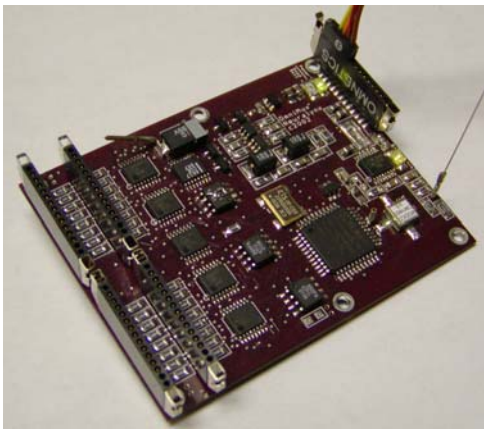


Figure 1-7 Shown is the transmitter board of the wireless data acquisition system by Neuralynx [5].

The receiving system for the device uses a technique called spatial diversity to improve the quality of the wireless link. Instead of a single antenna, eight separate antennas are placed around the room to increase the likelihood that at least one antenna will pick up a strong signal. Neuralynx hardware monitors each of the antennas and switches to the antenna that has the greatest received signal strength. The hardware also reassembles the multiplexed data. More information on the Neuralynx wireless data acquisition system may be found at [5]. The first section of the next chapter will discuss the motivation behind a wireless device capable of digitally isolating action potentials *before* transmission occurs.

Chapter 2 . Spike Isolating Neural Signal Transmitter

The first section of Chapter 2 will discuss the motivation and rationale behind a spike isolating neural signal transmitter. Second, requirements of a spike isolating transmitter, appropriate for a proof-of-concept device will be discussed. Finally, the spike isolating wireless neural signal recording system will be presented.

2.1 **Motivation and Rationale**

While wireless systems are attractive in the sense that they eliminate the geometric constraints on experimental chambers, they have several problems. First, the systems are relatively heavy and bulky and require a battery, which adds even more weight [13]. Second, wireless systems usually have very few channels [13] because of their limited throughput compared to wired systems.

The information content of brain activity is contained in the outbursts of electrical charge called action potentials. Typically, up to four hippocampal neurons may be close enough to an electrode for its signal to be identifiable over the noise floor. Since the firing rates of neurons are relatively slow, a large percent of the total neural signal consists of nothing but background noise. If the action potentials could be isolated from the rest of the neural signal *before* transmission, the throughput required to sample a channel would be significantly reduced. A spike isolating neural signal transmitter would increase the number of channels that could potentially be sampled, and reduce the amount of power necessary to record a single channel. The next section describes requirements for the spike isolating transmitter.

2.2 **Requirements**

The requirements for the second device are the attributes it was felt would be sufficient to prove the concept of a device that could digitally isolate and transmit action potentials for the purpose of throughput reduction. These requirements, given in this section, are not stringent in terms of sample rate, or total throughput, but they are appropriate for a proof-of-concept device.

The first and most important requirement was that the prototype needed to be able monitor an input signal with neural spikes, isolate action potentials, and transmit them to a computer for display. Spikes were to be differentiated from the rest of the neural signal by comparing the voltage level of the neural signal to defined threshold levels. Digitization of the neural signal by an Analog to Digital Converter (ADC) would be necessary to make the comparison possible. The device should begin transmitting when a digitized sample is found to be higher than an upper threshold level, or smaller than a lower threshold value. Once transmitting begins, transmission would continue for a fixed interval of 1.5ms, which is enough time to ensure any hippocampal action potential is completely sampled and transmitted. Once the transmission is complete, the device is to resume sampling and monitoring for threshold crossings.

A second requirement was that when the device was not transmitting, a 5 sample buffer was to be continually maintained. The result is that when a threshold crossing occurs, five extra samples in addition to the samples taken over the 1.5 ms time span

would be included in the transmitted data. This feature would make it so that some of the waveform before the threshold crossing could be observed on the computer display, which is a helpful for spike sorting.

A final requirement of the device is that it needed to be able to sample a single channel at no less than 20 kHz. Over a time span of 1.5 ms, a sample rate of 20 kHz yields a total of 30 samples. Figure 2-1 illustrates how a sampled spike should appear given the requirements from this section. The amplitude and frequency of the action potential are roughly 260 mV (after amplification by 60 db) and about 1000 Hz, respectively, which is an accurate representation of a typical spike. The sample rate is about 20 kHz, as there are 30 samples taken over the 1.5 ms sample period. Also shown are the five samples taken before the upper threshold crossing. The lower threshold crossing does not trigger another 1.5ms sampling session, as the upper threshold was crossed first.

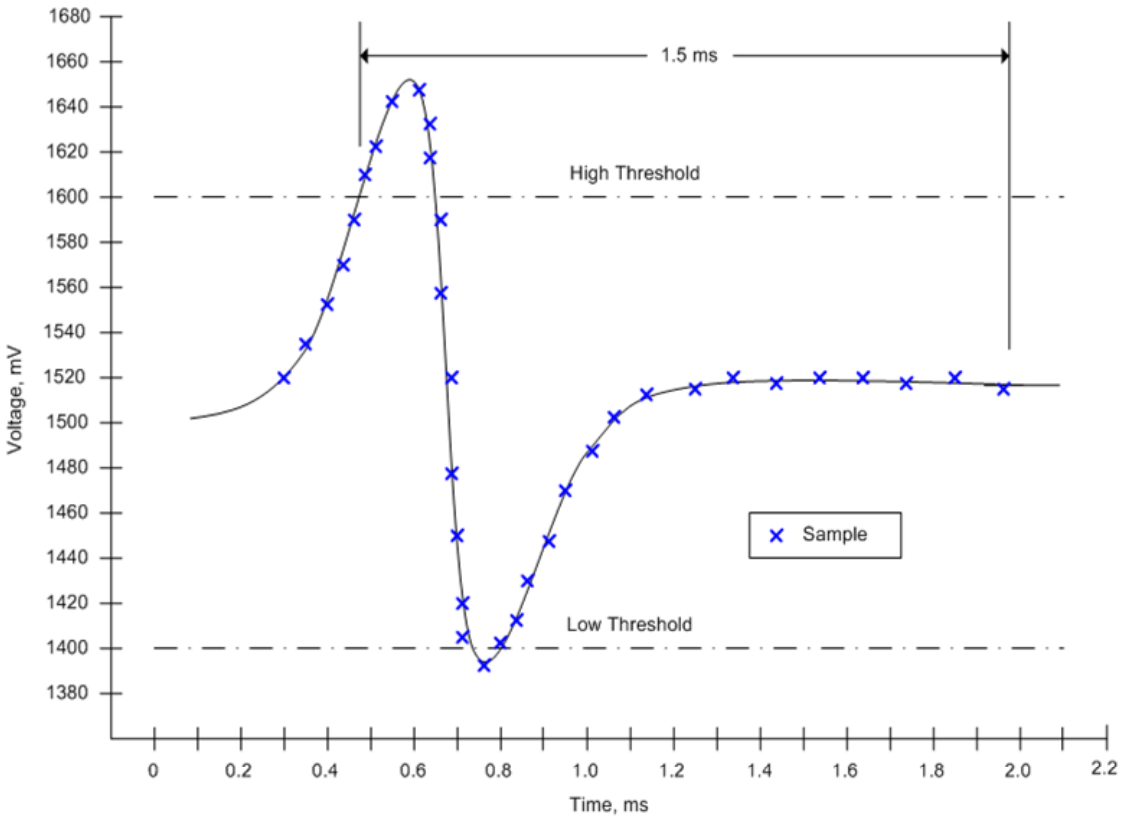


Figure 2-1 Shown is picture of what a sampled action potential should look like given the requirements from this section. Notice that there are five samples before the threshold crossing.

Because the device was to be a proof-of-concept, the requirements were relaxed in two major ways. First, time stamps were not necessary. The second respite in the requirements was applicable if a wireless connection of less than 20 kHz was used. In the case that the throughput of the wireless link was less than the desired 20 kHz sampling rate, either one of two things could happen. Either the sampling rate would need to be reduced, or directly after a spike was sampled, time could be set aside to allow the samples to be wirelessly transmitted. This break would be taken with the

understanding that if an action potential were to occur during the lull time, it would be ignored and not sampled or transmitted. As it was desired to have a sampling rate of at least 20 kHz, the possibility of missing an action potential was seen as an acceptable limitation, and the latter of the two options would be chosen in the case of a slower than desired wireless connection.

2.4 **Description of the Neural Signal Transmitter**

The complete neural signal transmitter system is shown in Figure 2-2. The figure shows how neural data, originating at the rat, is passed among the different elements of the system. The brown box represents the rat from which four channels of neural data are taken. In red, is the front-end circuit that filters and amplifies the animal's neural signals and passes them to the Microchip PIC18F458 Microcontroller Unit (MCU) in green. Radio Frequency (RF) transmission is handled by the Bluegiga Bluetooth module, also in green, which interfaces with the PIC through a Universal Asynchronous Receiver Transmitter (UART) connection. The communication and display equipment, shown in blue, receives data through a wireless link from the Bluegiga Bluetooth module. To control the PIC MCU, commands may be wirelessly transmitted to the Bluetooth module using the communication and display components.

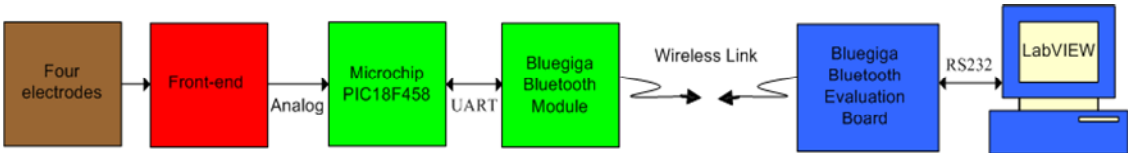


Figure 2-2 Shown is a breakdown of the wireless instrumentation system. The neural signal transmitter is shown in green and the communication and display components are in blue. Brown represents the lab rat and red corresponds to the head-stage.

The neural signal transmitter, is the primary topic of this thesis, and consists of the components in Figure 2-2 that are shown in green. In Chapter 3, some details of these components will be discussed. In this chapter, the overall system will be explained in more general terms, starting with the neural signal transmitter. After that, the communication and display components, shown in blue, will be explained. While a power supply for a head-stage was built into the neural signal transmitter, a front-end circuit is not discussed in this thesis.

2.4.1 **Neural Signal Transmitter**

The neural signal transmitter consists of the Microchip PIC18F458 and the Bluegiga WRAP THOR Bluetooth module. The PIC and the Bluetooth module are soldered onto two separate circuit boards that lay flat against one another. The PIC18F458 is soldered onto a 4 layer Printed Circuit Board (PCB) jointly designed by WFUSM and Triad Semiconductor and modified by me for the purposes of the neural signal transmitter. We will call this PCB the PIC board. The Bluegiga Bluetooth module is soldered onto a custom designed and fabricated circuit board, along with the power supply circuitry. The neural signal transmitter requires 3.5 – 10V and draws about 50 to 190mA of current.

Shown in Figure 2-3 is the PIC board. The Microchip PIC18F458 MCU may be seen in the figure. A programming connector is used to download firmware onto the PIC MCU and a UART connector may be used to access its UART pins. Also seen is the 40 MHz clock oscillator that provides timing for the internal instruction cycle of the PIC.

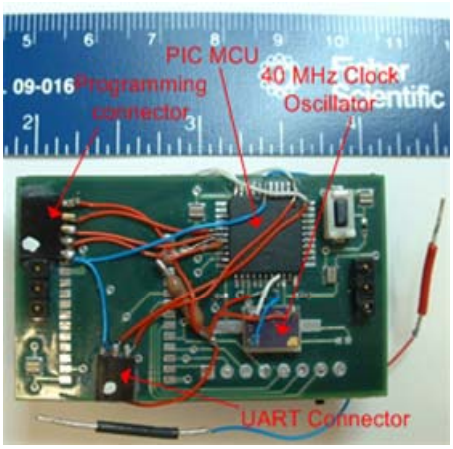


Figure 2-3 The PIC board is shown here.

The Microchip In-Circuit Serial Debugger 2 (ICSD2) may be used for programming or debugging the prototype device, as shown in Figure 2-3. When used as a programmer, the ICSD2 downloads firmware developed in the Microchip Integrated Development Environment (IDE) to the PIC MCU. Though the ICSD2 may be used to provide power to the neural signal transmitter during programming, it is recommended that a 3.7 V power supply be used to provide power to the prototype during programming, as using the ICSD2 to power the circuit reverse biases voltage regulators on the neural signal transmitter and may potentially damage them through a phenomenon known as latch-up [16].

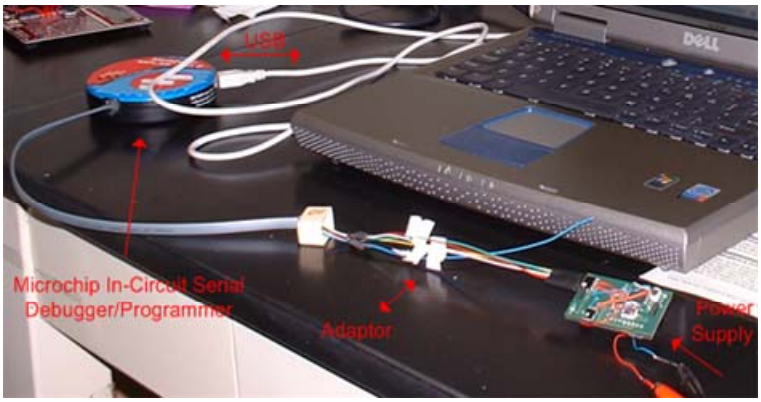


Figure 2-4 The Microchip ICSD2 and a custom adaptor may be used to flash new firmware onto the prototype.

In Figure 2-5, the custom designed and fabricated circuit board is shown. We will refer to this circuit board as the Bluetooth board. The WRAP THOR module from Bluegiga, which is responsible for wireless communication, is soldered to the board.

As can be seen in the figure, gold plated break-away header connectors, which are soldered to the PIC board, extend through holes in the Bluetooth board. These connectors provide access to six pins on the PIC MCU. Four of the pins are electrically connected to different input channels of the PIC's ADC, while the remaining two pins provide access to the PIC's external voltage reference pins for the ADC.

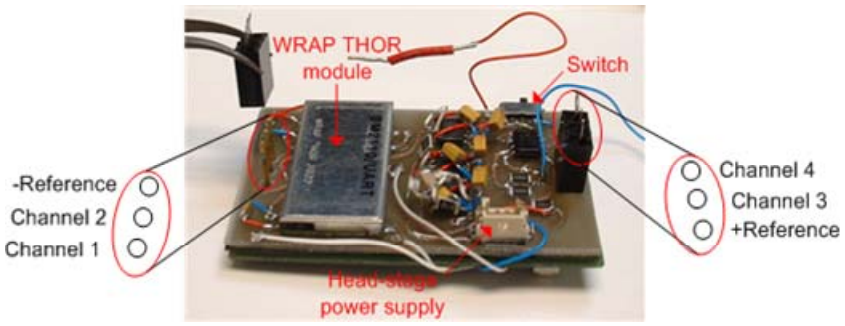


Figure 2-5 The Bluetooth board is shown here.

The PIC MCU uses its Analog to Digital Converter (ADC) to convert the neural signal on its ADC pins into 8-bit binary format. An ADC must have upper and lower reference voltages to orient the assignment of binary values to analog voltages. For example, let's consider that we have an 8 bit ADC, which means that the ADC uses 8 binary bits to represent a sampled value. An 8 bit ADC has 2^8 or 256 (0-255) discrete values that it may assign to an analog voltage. If the ADC has an upper reference of 3.0 V and a lower reference of 0V (GND), this means that analog voltages of 3V and GND get assigned binary values of 255 and 0, respectively. Any analog voltages between 0 and 3V get assigned binary numbers relative to the reference voltages. In this case, the resolution of the ADC would be

$$\frac{3V - 0V}{256} = 0.01172 \frac{volts}{division}$$

where a division defines the smallest possible change in quantization level.

The default ADC references used by the PIC are its supply voltages of 3.3V and GND. Solder pads on the Bluetooth board supply external reference voltages in case their use is desired to improve the resolution of the ADC. Figure 3-1b shows where these reference voltages are located on the Bluetooth circuit board. The default external reference voltages are 2V and 1V, but these may be changed to any value between 3V and GND. For the PIC to use the external references, the contents of the register "ADCON1" needs to be changed. A segment of the PIC's assembly language code is shown in Figure 2-6. The commented lettering in red describes how to make the PIC use the voltages on its reference pins as references for its ADC, instead of its internal supply voltages. Once the change is made to the code, it has to be recompiled, and downloaded to the MCU as shown in Figure 2-4 for the changes to take effect.

```
ADC_init
        movlw       b'01000000'         ; for external Vrefs: b'01001000'
```

```
movwf     ADCON1,0          ; configure for left justification, appropriate ADC clock source and
                            ; A/D port configuration, use internal voltage references

bsf       CVRCON,CVREN,0    ; disconnects VREF- from VSS to avoid excessive current draw
                            ; output impedance of VREF source must be less than 20 ohms (use
                            ; voltage follower). ("see PIC18FXX8 Rev.B4 Silicon/Data Sheet Errata"
                            ; problem number 7)
                            ; ADC is NOT turned on here to save power.
                            ; ADC module will be turned on when it is needed.
return
```

Figure 2-6 Here is the **s**egment of assembly language code responsible for setting up the PICs ADC module.

A front-end that uses bi-polar amplifiers needs positive, negative, and GND supply voltages. Because the PIC's ADC cannot accept voltages outside of its voltage references, the circuit board is equipped with a head-stage power supply, shown in Figure 2-5, that when used to power a head-stage, biases the output of the front-end to be within the PIC's reference voltages. The output is accessible through a three prong Hirose connector that supplies regulated 3.0V, 1.5V and GND. When used as a supply for a bi-polar head-stage, the 1.5 V is used as front-end GND, the 3V as positive 1.5V, and the GND as -1.5 V.

During operation, four separate channels of filtered and amplified neural signals are applied on the PIC's ADC channel pins seen in Figure 2-5. The prototype device may function in two different modes of operation. In "continuous mode", the device digitizes a channel of neural data at about 11 kHz and transmits it to the communications and display equipment. In "waveform mode", the device isolates action potentials, samples them at about 30 kHz, and transmits them. Action potentials are isolated by comparing the voltage level of the incoming signal to that of upper and lower threshold values. Because the wireless link has a relatively low throughput, while in waveform mode the device stops monitoring for spikes in the neural signal for a period of 5ms immediately following the sampling of an action potential. This pause is necessary to give the Bluetooth module time to transmit the sampled action potential.

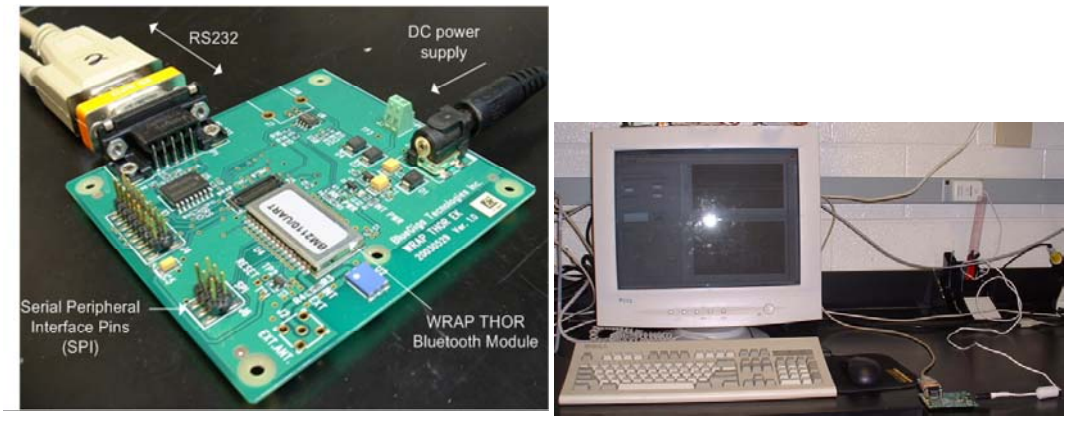
The neural signal transmitter device is capable of transmitting one channel of data at a time. The sampled channel, as well as the threshold values, and the devices mode of operation may be changed by using the communications and display equipment. These components will be discussed in the next section.

2.4.2 **Receive and Display Components**

The receive-and-display components were used to wirelessly communicate with the neural signal transmitter. To accomplish this, a Bluegiga WRAP THOR evaluation board, shown in Figure 2-7a served as a wireless interface between a computer and the prototype. The Bluegiga evaluation board is made by Bluegiga for the purposes of evaluating their WRAP THOR Bluetooth module. The board has a variety of break out pins for accessing I/O and programming pins on the wireless module. A line driver on the evaluation board converts the WRAP THOR modules UART data into RS232 so that the board may be plugged directly into a computers serial port.

Once a wireless connection is established, any data that the evaluation board receives from the neural signal transmitter is sent through a serial cable to the computer.

Alternatively, any data that the evaluation board receives from the computer is wirelessly transmitted to the neural signal transmitter. In this way, the computer, which is running a LabVIEW user interface code, may be used to display data from the neural activity transmitter, as well as control the device. Figure 2-7b shown the Bluegiga WRAP THOR evaluation board hooked up to a computer running the LabVIEW interface code.



a. **b.**

Figure 2-7 The Bluegiga evaluation board is used to wirelessly communicate with the neural signal transmitter. **a.** A close-up of the Evaluation board. **b.** The evaluation board is connected to the PC through a serial cable. The LabVIEW interface code is seen on the computer display.

A user interface was needed on the computer to both display received data and send commands to the neural signal transmitter. National Instruments LabVIEW was chosen for this task, as it provides a versatile and easy to learn Graphical User Interface (GUI) programming environment. LabVIEW enables easy access to computer serial ports through the use of built in Virtual Instruments (VIs). There is a Virtual Instrument for initiating a connection to a serial port, one to determine the number of bytes in the serial port buffer, one to read the contents of the buffer, and one to close the serial port connection.

The LabVIEW program's front panel is shown in Figure 2-8. There are two different types of waveform displays. The waveform graph is used to display a single set of samples received from the computers RS232 buffer. In other words, if 45 samples are gathered from the buffer by the port read VI, 45 samples will be displayed on the waveform graph. The horizontal yellow lines on the waveform graph indicate the threshold levels that are currently being used by the neural signal transmitter. These references may be adjusted manually using the "Threshold High" and "Threshold Low" controls on the front panel. The waveform chart scrolls a fixed number of samples across the display. It is useful for observing the history of the collected data as it may show samples from several sets of received data.

The "milliseconds to wait" control allows the user to adjust the amount of time between each successive read of the computer serial port buffer. Changes to the "milliseconds to wait" field can have a major effect on the appearance of the data shown in the waveform graph display. A larger "milliseconds to wait" value means that more time is allowed for samples to enter the buffer before a read so that on the next

read cycle more samples will be received and displayed on the waveform graph. Alternatively, a smaller "milliseconds to wait" value will result in fewer samples displayed in the waveform graph display.

The port number control allows the user to change the serial port from which the LabVIEW program will send and receive data. The program saves all collected data to a file specified in the file path field. The "bytes in the buffer" display shows the number of samples received by the port read VI and the "string read" display shows the received string of data in ASCII format.

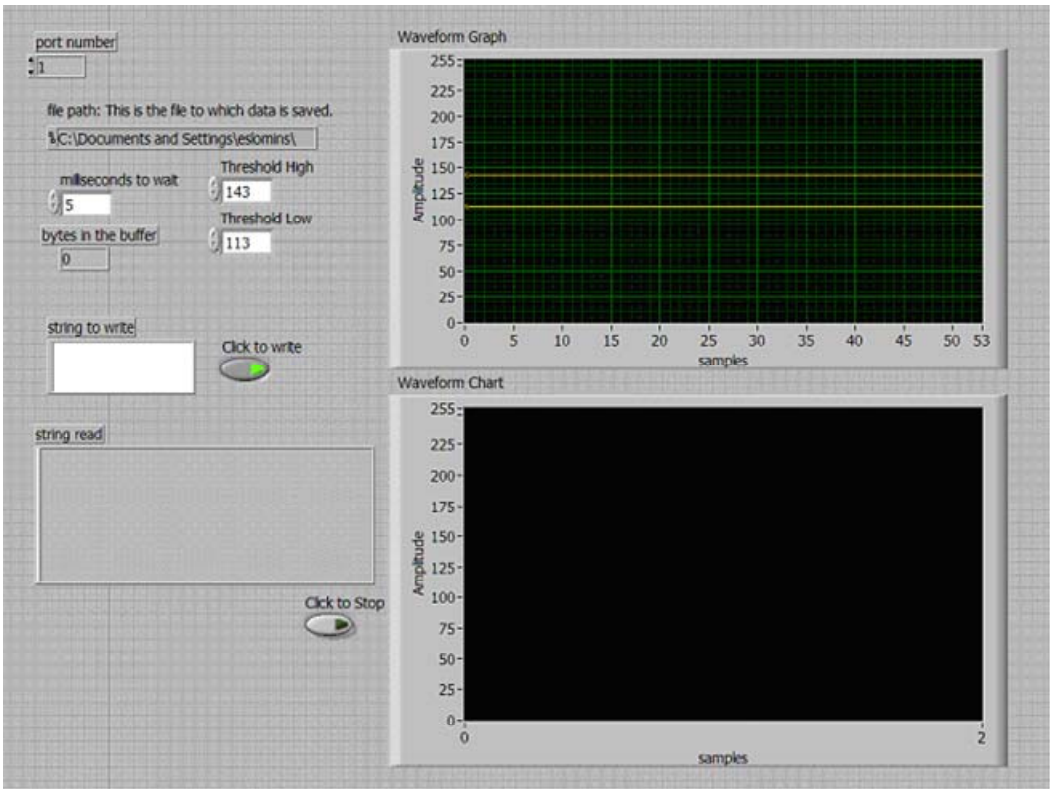


Figure 2-8 The LabVIEW user interface.

The "string to write" field may be used to send 8-bit codes to the evaluation board through the serial port. These codes are transmitted to the neural signal transmitter and may be used to control the device. All of the available control codes and their descriptions are given in Table 1. Though the default format of the "string to write" field is hexadecimal, ASCII format may also be used.

To change the threshold values used by the neural signal transmitter, "SETUP" is entered by typing 35 to the "sting to write" field and clicking, "click to write". When this command reaches the neural signal transmitter, the device responds by transmitting the string "Enter ThreholdH", which is displayed on the "string read" display on the front panel. The new desired threshold level, which is entered as an 8-bit integer in either hexadecimal or ASCII format, may be entered into the "sting to write" field. For convenient conversions from decimal format, a decimal to hexadecimal chart may be accessed on the front panel by scrolling down. Once the new upper threshold level is received, the prototype then transmits the string "Enter ThresholdL", at which time the

new lower threshold level may be entered into the "string to write" field and transmitted. Once the prototype has received both threshold levels, it goes into "OFF" mode to allow the user to choose which channel is desired to be sampled.

Table 1 Shown are the codes that may be used to control the neural signal transmitter

Code name	BINARY	HEX	ASCII	Decimal	Effect
OFF	00110000	30	0	48	Turns the neural signal transmitter completely off.
CH1	00110001	31	1	49	Causes neural activity transmitter to sample channel 1.
CH2	00110010	32	2	50	Causes neural activity transmitter to sample channel 2.
CH3	00110011	33	3	51	Causes neural activity transmitter to sample channel 3.
CH4	00110100	34	4	52	Causes neural activity transmitter to sample channel 4.
SETUP	00110101	35	5	53	Enter mode whereby new threshold levels may be entered and transmitted to the prototype.
SWITCH	00110110	36	6	54	Alternates between "continuous" and "waveform" modes.

Chapter 3 . Components of the Neural Signal Transmitter

This chapter takes a closer look at the Microchip PIC18F458 and the Bluegiga WRAP THOR Bluetooth module, shown in green in Figure 2-2. Also to be discussed is the circuitry that provides power to these devices, and the custom designed and etched circuit board that provides a platform for the Bluetooth module and the power supply circuitry. The first things to be discussed are the power supply circuitry and the custom circuit board. The PIC18F458 MCU and its assembly language program are to be discussed next. Finally, the Bluegiga WRAP THOR Bluetooth module and Bluetooth wireless protocol are discussed.

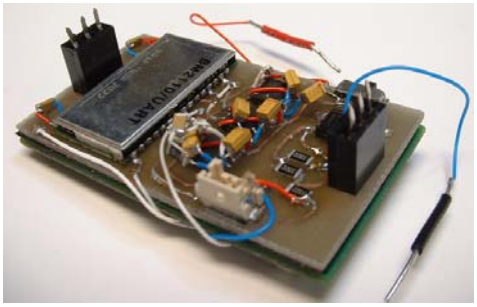
3.1 **Power Supply Circuitry and Circuit Board**

The custom circuit board, shown in Figure 3-1 provides a platform on which to solder the Bluegiga wireless module and power supply circuitry that provides power to the wireless module, the PIC, and a head-stage. Figure 3-1a shows an actual photograph of the board, while Figure 3-1b shows a schematic of the circuit board with the external components of the supply circuitry removed to make it easier to follow the copper traces of the circuit board.

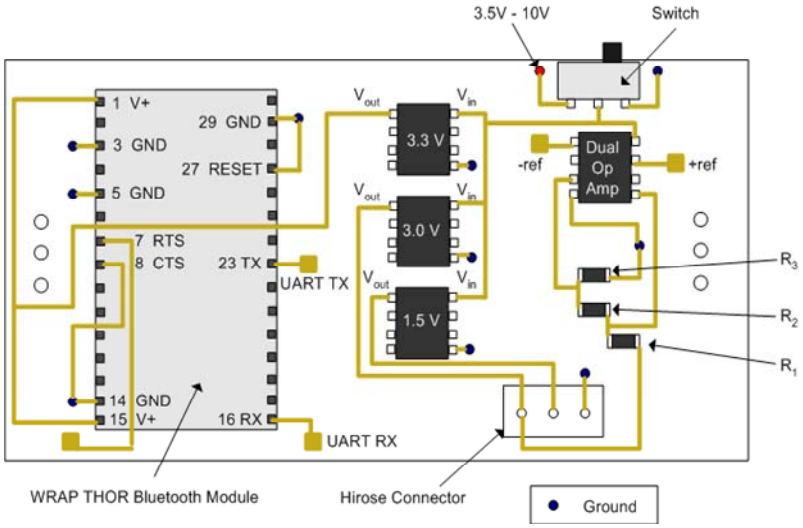
Three National Semiconductor LP2986 Low Dropout Out (LDO) voltage regulators are used to provide 3.3V for the Bluegiga Bluetooth module and the PIC, and 3.0V and 1.5V for a front-end. The WRAP THOR module is required to have a supply of 3.3 ± 0.1 V. Even though the PIC has a voltage range of 2.0-5.5V, its supply is matched to the Bluetooth modules so that each device will communicate using the same voltage levels through their respective UARTs to avoid latch-up.

The power supply circuitry also provides two reference voltages from the 3.0V regulator for the PIC's ADC in case the use of external voltage references is desired. These reference voltages are set by three 1206 sized chip resistors, which form a resistive voltage divider and are shown in Figure 3-1b as R1, R2, and R3. The default resistors used on the device are each 3.3 kΩ, which result in upper and lower reference voltages of 2 and 1 volts respectively. The two references are buffered using a Texas Instruments dual op amp chip, with both amps in the unity gain voltage follower configuration.

An E-Switch 1257 2 position slide switch is on the board and is used to turn the device on and off. Voltages of 1.5 V, 3.0 V, and GND are supplied for a head-stage. These three voltage levels are conveniently accessible through the Hirose Surface Mount (SMT) 3 lead 1.25 mm pitch connector.



a

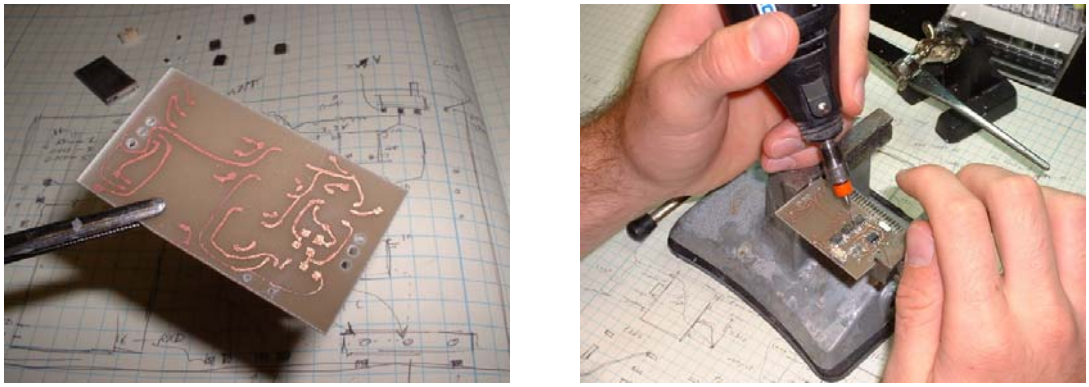


b.

Figure 3-1 a. A picture of the new circuit board is shown (top) attached to the PIC board (underneath). **b.** In this schematic drawing, the external components are omitted to make the wire traces easier to see.

The new circuit board was constructed from a two sided ½ oz copper clad 1/16 in. epoxy glass circuit board. One side of the board serves as a ground plane, while the other has the traces etched out on it. To etch the traces on the board a PC Board Kit from Radio Shack was used. The kit included an etchant resistant pen, etching solution (Ferric Chloride), and copper clad circuit boards. First, a copper clad board was cut into a rectangle with the same dimensions as the PIC board. Next, the desired circuit traces were drawn on the board using the etchant resistant pen and the ground plane side of the circuit board was covered with electrical tape. Finally, the circuit board was submersed in a container of etchant solution, which slowly dissolved the exposed copper. The finished product is shown in Figure 3-2a.

Once the circuit board was etched, it was ready to be populated. Using a Dremel Tool, holes were drilled where the circuits needed access to the ground plane. All electronics were carefully soldered onto the circuit board. Figure 3-2b shows a picture of the board being populated.



a. b.

Figure 3-2 a. The traces of the circuit board are freshly etched out. **b.** A Dremel is used to drill through holes for access to the ground plane.

3.2 **Microcontroller and Assembly Language Program**

The PIC18F458 is a general purpose MCU. It has several convenient on chip modules. In particular, an on-chip Analog to Digital Converter (ADC) eliminates the need to design separate ADC circuitry, and an on-chip Universal Asynchronous Receive Transmit (UART) module makes asynchronous communication with the Bluetooth module relatively simple. Both the ADC and UART modules operate in parallel with program memory. In other words, once they are told what to do, the ADC or UART modules can operate almost independently while the PIC's program runs.

The PIC MCU and Bluetooth module communicate with one another using UART communication. UART communication is a digital data packet communication scheme. The data packet consists of eight binary bits of information. A start bit is sent as a prefix to the eight bits of data and either one or two stop bits may be appended to the data packet. The PIC and the Bluetooth module communicate using a single stop bit for a total packet length of ten bits. RS232 and UART communication are exactly the same except that the voltage levels used to represent binary ones and zeros are different. A line driver is an electronic device that may be used to switch between UART and RS232.

The PICs ADC module can handle up to 12 separate channels but can only sample one channel at a time. A 10-bit Successive Approximation Register (SAR) is used to do the conversion. Using the UART module is a simple matter of configuring the UART module and loading the appropriate registers with the data to be exchanged. A variety of baud rates may be used, but the PIC's UART baud rate was set as close as possible to the baud rate of the WRAP THOR module's UART baud rate of 115.2 kbaud. The PIC's UART baud rate was configured for 113.6 kbaud, which results in a baud rate error of

$$BR_{error} = \frac{115.2 - 113.6}{115.2} = 1.4\%$$

when matched to 115.2 kbaud.

Before discussing the assembly language code written for the PIC18F458, some basic Microcontroller concepts that may be helpful for understanding the assembly language code will be introduced. The concepts that will be introduced, which were all used in the development of the code, are data memory and program memory, the hardware stack, interrupts, and pointers.

3.2.1 **Data Memory and Program Memory**

The two different kinds of memory used on the PIC for the prototype device were data memory (RAM), and program memory (ROM). Data memory (RAM) is divided into two different types of eight bit registers: general purpose and special function. General Purpose Registers (GPRs) may be used to store variables, while Special Function Registers (SFRs) are used to configure and control the MCU. Names may be assigned to general purpose registers for convenience. For example the name "Bufferedsamples" may be assigned to data memory location 0x503 by using the "equ" command. Any reference to "Bufferedsamples" will automatically be associated with data memory location 0x503. The PIC18F458s data memory is divided into 16 different banks, each containing 256 registers. Six of the 16 banks contain GPRs and one of the 16 banks contain SFRs. The remaining 9 banks are unused. Before reading or writing data to any register in data memory, the bank in which the register resides must first be selected. In the PIC18F458, banks are selected by loading the appropriate binary identifier into the first four bits of the Bank Select Register (BSR), which is an SFR dedicated to the purpose of bank selection.

Operations that tell the MCU what to do are called Operational Codes (Op-Codes) and they are stored in program memory [14]. In the PIC18F458, there is nearly 2 Mb of addressable program memory, which gets executed sequentially from the top down. Only 32 kbytes of which may be used to store the user program. For more information on the 77 available Op-Codes available for the PIC18F458, the interested reader should download the data sheet from [11]. Code may be written anywhere in the accessible program memory by using the "org" command. For example, using the command "org 004h", will cause the next instruction to be written to program memory location 0004h. A program counter, which consists of three SFRs, keeps track of the line of code in ROM that is being executed at any given time.

3.2.2 **The Stack**

Next, it is good to know the details of the STACK and the difference between the "call" and "goto" commands. The STACK is a stack of registers that is used to save the value of the program counter so these locations of program memory may be returned to. When a "call" command is executed, the top of the STACK is loaded with the current value of the program counter and the program vectors to the location in ROM specified by the "call" command. When a "return" command is encountered, the value loaded into the STACK is then popped off and the program vectors to that location in program memory. The "goto" command does nothing to the STACK, it simply causes the program to vector to whatever location in ROM specified by the "goto" command. In

the PIC18F532, the STACK is 32 levels deep, which means that 32 “call” commands may be issued before a “return” instruction is encountered.

3.2.3 **Interrupts**

Interrupts are useful because they allow the PIC to execute code at full speed until some external or internal event triggers the interrupt. When an interrupt is triggered, the STACK is loaded with the current program counter value, and the program disables interrupts and vectors to the location in ROM that is dedicated to servicing interrupts. In the case of the PIC18F458, this area of program memory is located at address 0x008. When the interrupt has been serviced, a “retfie” command may be used to return to the location in ROM saved by the STACK. The “retfie” command also automatically re-enables the interrupts. If the “retfie” command is not used the STACK is not unloaded and the stack must be manually unloaded using the “pop” Op-Code to remove the top layer of the stack and avoid overflow. Opting to not use the “retfie” command also means that the interrupts have to be manually re-enabled.

In the neural transmitter program, UART-receive and UART-transmit interrupts are used. The use of UART-receive interrupts means that when a channel code is received on the UART receive pin by the Bluetooth module, an interrupt is triggered, the stack is loaded, and the program vectors to the interrupt service address in program memory. In this case, the interrupt is serviced by saving the received value into data memory location “ch_sel” and then vectoring to loop1. Similarly, the use of UART-transmit interrupts means that when the UART transmit register is empty, an interrupt is triggered. In the code used for the neural signal transmitter, this type of interrupt is used in the area of code called “waveform_mode”.

3.2.4 **Pointers**

Pointers are used to accomplish an addressing technique known as “indirect addressing”. In indirect addressing, a register’s address is loaded into a “File Select Register” (FSR). Then, by using certain SFRs, the contents of the register pointed to by the FSR may be read and/or modified. This addressing technique is useful because depending on the SFR used to read the contents of the register pointed to by the FSR, the FSR may be automatically incremented or decremented. This makes the use of the FSR an efficient way to read or save data to a sequential list of data. In the code used for the neural signal transmitter, FSRs are used to save data into RAM until the Bluetooth module is ready to transmit them. The SFR POSTINCn, where n is a number between 0 and 2, signifies the number of the FSR, and is used to perform an operation on the register pointed to by the FSR then automatically increment the FSR to be pointing at the next register.

3.2.5 **Assembly Language Code**

The assembly language code written for the PIC18F458 uses all of the concepts discussed thus far in section 3.2. In Appendix B, the entire assembly language program is shown. This code was developed in the Microchip Integrated Development

Environment (IDE) version 6.10, which may be downloaded from [11]. Instead of stepping through the entire assembly language program, which is relatively long and cumbersome, a simplified flow diagram of the code will be discussed.

Shown in Figure 3-3 is a simplified flow diagram of the assembly language code. The oval labeled "start" in Figure 3-3 indicates where the program begins. The first thing that the program does is initialize the MCU's timer, ADC, and UART modules, setup interrupts, and send the proper commands to the Bluegiga module to setup a wireless connection. More information on these commands will be given in section 3.3. Interrupts are configured for interrupt on receive, in other words, when the receive register of the UART becomes loaded with a received value, a receive interrupt is triggered and the program vectors to the area of program memory labeled "Interrupt Service". Next, the program branches to "OFF" and waits for receive interrupt to occur.

When a receive interrupt occurs, the interrupt is serviced in the "Interrupt Service" routine and the program branches to "Loop 1". Depending on the received value, the program will branch to another area of program memory. Table 2 shows the possible codes and the action that will be taken by the MCU for each received code. These codes are sent to the PIC using the communications and display equipment that was the topic of section 2.4.2.

Table 2. Shown are the possible codes and their effects on the MCU.

Code name	BINARY	HEX	ASCII	Decimal	Effect
OFF	00110000	30	0	48	Program branches to "OFF"
CH1	00110001	31	1	49	Program branches to "Channel 1", where the ADC is configured for the channel of interest. From there, the program branches to "ON"
CH2	00110010	32	2	50	Same as above except that ADC is configured for Channel 2
CH3	00110011	33	3	51	Same as above except that ADC is configured for Channel 3
CH4	00110100	34	4	52	Same as above except that ADC is configured for Channel 4
SETUP	00110101	35	5	53	Enter mode whereby new threshold levels may be received.
SWITCH	00110110	36	6	54	Alternates between "continuous" and "waveform" modes.

If the program receives code 030h, the program will branch to “OFF” where the ADC and the timer are turned off to save power. If the program receives codes 031h through 034h, the program will initialize the ADC with the appropriate channel configurations and branch to the “ON” loop. Once in the “ON” loop the program will digitize a single 8-bit sample of the neural signal with its ADC. If the MCU is currently configured for “Continuous” mode the program will simply send the sample through the UART by placing the sample in the UART transmit register, first making sure that the Bluetooth module is ready and the UART transmit register is not still in use from the previous send. If the MCU is configured for “Waveform” mode, the program first saves the sample into a five sample buffer. Then, the sample is compared to upper and lower threshold values. If the sample is greater than the upper threshold value, or smaller than the lower threshold value, the program branches to “Waveform_mode”, otherwise another sample is digitized from the neural signal, starting the process over again.

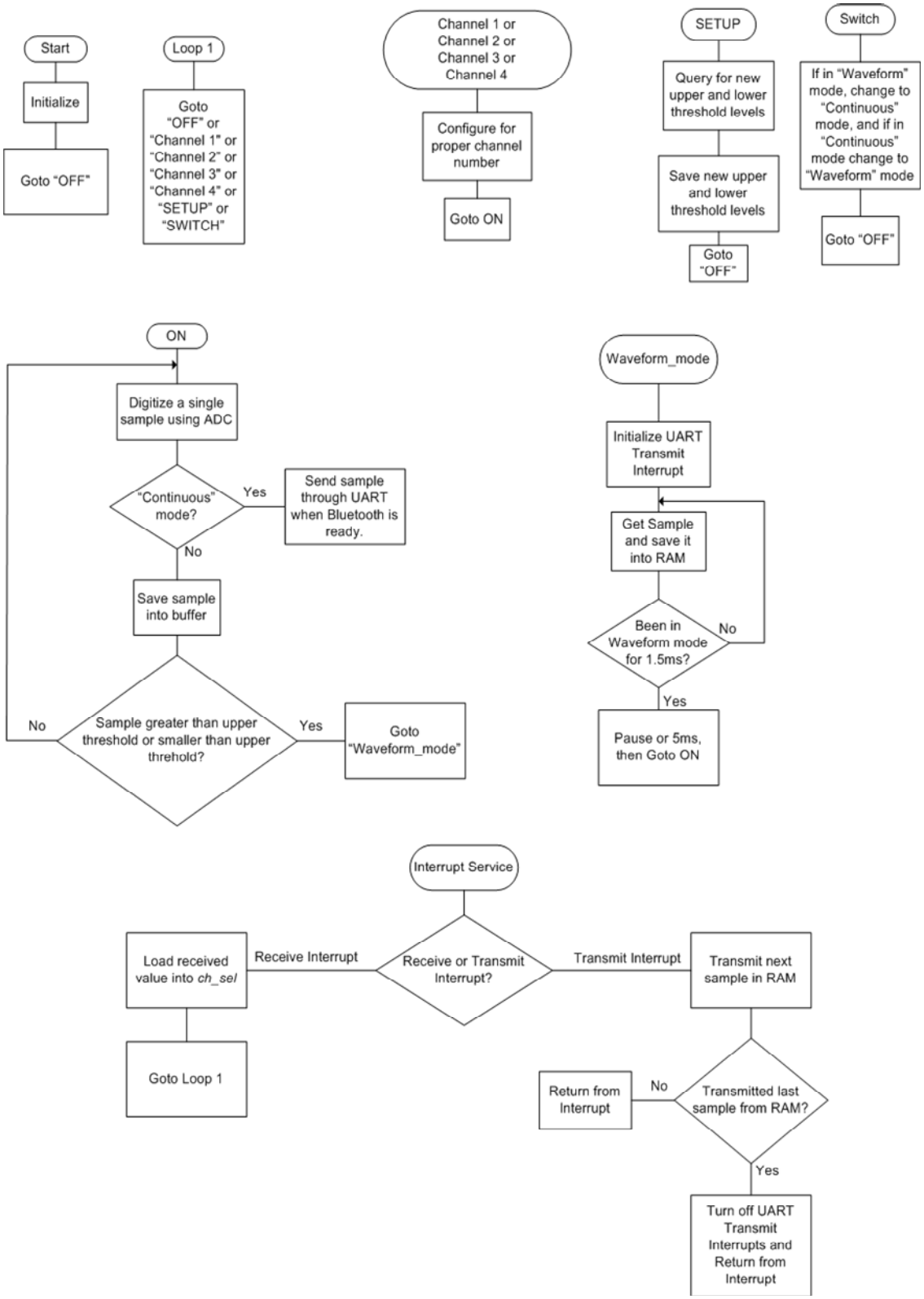


Figure 3-3 A flow diagram of the PIC's assemble language code is shown.

Once in "Waveform_mode" the program first initializes the UART transmit interrupt. What this means is that when the UART transmit register is empty, an interrupt is triggered and the program branches to "Interrupt Service". A UART transmit interrupt is serviced by first sending the next sample saved in RAM through the UART, then checking to see if the last sample saved in RAM has been sent. If not, the program returns from the interrupt, which means that the program returns to the line of code after the last code executed before the interrupt occurred. If the last sample has been sent, then transmit interrupts are disabled and the program returns from the interrupt.

After initializing transmit interrupts, the "Waveform_mode" branch of code then gets another sample and saves it into RAM. After saving another sample, it checks to see if it has been sampling in "Waveform_mode" for more than 1.5ms. If the MCU has not been digitizing and saving samples into RAM for more than 1.5ms the program simply continues sampling. If it has been running for more than 1.5ms, the program pauses for 5ms to give the UART time to transmit all of the saved samples, then goes back to the ON loop to repeat the process of buffering samples and checking for threshold crossings.

While in waveform mode The PIC's ADC is allowed to run at full speed which samples the neural signal at about 30 kHz. This prediction was made based on conversion times given in the data sheet found at [17], as well as instruction cycle times. For a sample rate of 30 kHz , over the sample period of 1.5 ms, a total of

$$\frac{30\times10^{3}\,samples}{\text{sec}}\left(1.5\times10^{-3}\,s\right)=45 samples$$

will on average be collected. This number of samples, plus the 5 sample buffer, takes the following amount of time to send through the UART at 113.6 kbaud

$$50 samples\left(\frac{1}{113.6 kbaud}\right)\frac{10 baud}{samples/\text{sec}}=0.0044\,\text{sec}\approx 4.4 ms$$

To account for possible delays, the code pauses for 5ms after each waveform sample to ensure that the UART has enough time to transmit all of the samples.

Because the Bluetooth module supports a serial flow control technique called Ready to Send (RTS) Clear to Send (CTS) flow control, the PIC was programmed to follow the RTS/CTS protocol. The RTS/CTS flow control scheme, shown in Figure 3-4, is simple. If a device is able to receive data, it clears its RTS pin. If a device wants to send data, it must first make sure that its CTS pin is clear. In this way, data is not lost due to overflowing buffers.

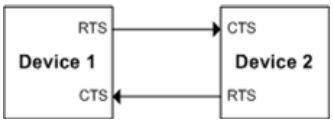


Figure 3-4 RTS/CTS flow control.

The PICs PortB pin 3 (RB3) is connected to Bluetooths RTS pin and thus serves as the PICs CTS pin. The program implements flow control by checking RB3 before it sends data through the UART to the Bluetooth module. Since the PIC is not saving data received from the Bluetooth module, there is no concern that the Bluetooth will overflow the PIC's buffers. Therefore, the Bluetooth modules CTS pin was tied to ground.

3.3 **Bluetooth**

This section will first introduce the WRAP THOR Bluetooth module from Bluegiga. Next, Digital Radio Frequency (RF) communication will be discussed along with some specifics of the Bluetooth protocol stack.

3.3.1 **Bluegiga WRAP THOR Bluetooth Module**

For the wireless link, a WRAP THOR Bluetooth module from Bluegiga technologies was used. The modules transmit at 2.4 GHz in the unlicensed Industrial Scientific Medical (ISM) band. These modules require no external components and are preinstalled with Bluetooth enabling software and a Bluegiga proprietary ASCII interface code. While running Bluegiga's proprietary ASCII interface program, all communication with the WRAP THOR module goes through its configurable UART. Special strings of ASCII characters are sent to the wireless module to command it to take certain actions. For example, the module may be instructed to look for other Bluetooth devices in its vicinity by sending it the command "INQUIRY [*time*]" where *time* is a number representing the number of seconds the module should spend inquiring. Table 3 is an excerpt of a table in the WRAP THOR ASCII interface manual and shows some of the available commands. For more information, the complete manual may be downloaded from [6].

Table 3. Selected ASCII commands from the WRAP THOR ASCII interface user's manual.

ASCII STRING	Action Taken by WRAP THOR module
`INQUIRY [time]`	Device starts an inquiry for [time] seconds and returns all discovered devices in the following form: INQUIRY [BD_ADDR] [class_of_device]
`CALL [BD_ADDR] [uuid] RFCOMM`	Device calls the device with BD_ADDR for [uuid] connection. The supported uuids are: (Note that these have to be given without the 0x- prefix!)

	Serial Port Connection 0x1101 Object Push 0x1105 (not supported in version 1.0) Object File Transfer 0x1106 (not supported in version 1.0) Example for Serial Port Profile connection to machine 00:07:80:80:00:79 call 00:07:80:80:00:79 1101 rfcomm NOTE: After successful call, the device goes to data mode
SET BT AUTH * [pin]	Sets the default personal identification code (PIN) used for the Bluetooth connections.
SET	Shows the settings.
HELP	Prints the available commands.
CLOSE [link_id]	Closes the given connection identified by link_id

The MCU communicates with the WRAP THOR module through a UART connection at about 113.6kbaud and is able to instruct the embedded Bluetooth device to setup a wireless connection using the commands from Table 3. Once a wireless connection is setup, the PIC MCU may send sampled neural data through the UART connection to the WRAP THOR module on the custom circuit board. When the embedded wireless device receives the data, it assembles it into packets and transmits it using Bluetooth protocol. More details on Bluetooth packets and protocol will be discussed in the next section. When the RF data is received by the evaluation board, it is reassembled into UART format and line driven to levels readable by the computers RS232 serial port. The net effect is that the WRAP THOR modules act as wireless serial ports operating at 115.2kbaud. At 115.2 kbaud, the maximum obtainable sample rate is

$$\frac{115.2 kbits}{\text{sec}}\left(\frac{sample}{10 bits}\right)=\frac{11.52 ksamples}{\text{sec}}$$

which is slightly slower than the desired sampling rate of about 20 ksamples/sec. To compensate for the slow sampling rate, the PIC processor was programmed to save samples into registers until the wireless module is ready to transmit them. This technique allows the PIC to sample at an acceptable sample rate while the throughput is only 115.2kbaud. The cost of this sampling technique is that any action potential that happens to occur during the 5ms pause directly after an action potential is sampled will be ignored by the neural signal transmitter. As previously stated, this limitation is acceptable for a proof-of-concept device.

The WRAP THOR modules built-in communication protocol simplifies the integration of wireless Bluetooth technology. The next section goes into some of the details of digital RF communication and the Bluetooth protocol to give the reader an appreciation of how the wireless module works.

3.3.2 **Digital Radio Frequency Communication and Bluetooth**

The goal of any digital Radio Frequency (RF) system is to wirelessly transfer digital data from one location to another. This goal may be accomplished using a digital RF system represented by the block diagram shown in Figure 3-6. The transmitter receives some source data, which is encoded, modulated onto a carrier, amplified, and finally transmitted. The receiver receives the RF energy by its antenna, downconverts the signal, and decodes it to reconstruct the original data. The source data, in the case of the neural signal transmitter, consists of digital 8-bit samples. Before it is modulated onto the RF carrier, the source data must be properly formatted so that it can be effectively transmitted, received, and decoded [7]. This formatting is accomplished by the encoder.

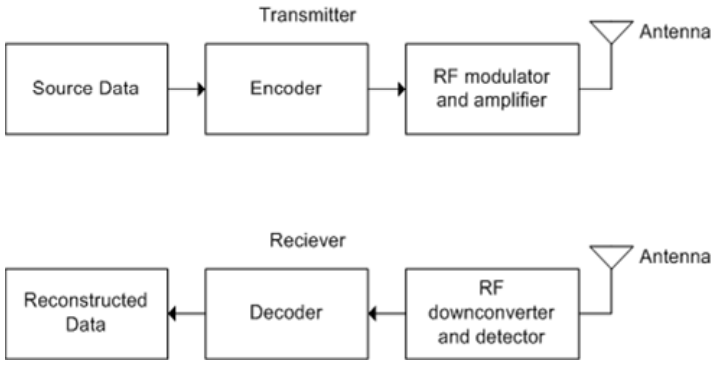


Figure 3-5 Digital Radio communication diagram [7]

In a digital system, one way that the encoder formats the source data is by adding certain information to it. This information, represented by bits of data assembled into packets, serves several purposes, for example, ensure that the receiver will not mistake a false signal for a real one [7]. Packets allow error free data communications to be performed in a highly automatic way, and are used extensively in two-way data communications [8]. There are many different types of data packets but they generally contain bits for the purpose of identification, packet correction, synchronization, and of course, data transfer [7]. Figure 3-3 shows a Bluetooth message frame or packet. Bluetooth data packets are relatively complicated, containing information to synchronize clocks, control errors, provide addresses, maintain a secure connection, and identify the capabilities of their hosts. We will not go into the details of each field of the data packet but it is shown here to give the reader an idea of a Bluetooth packet's complexity.

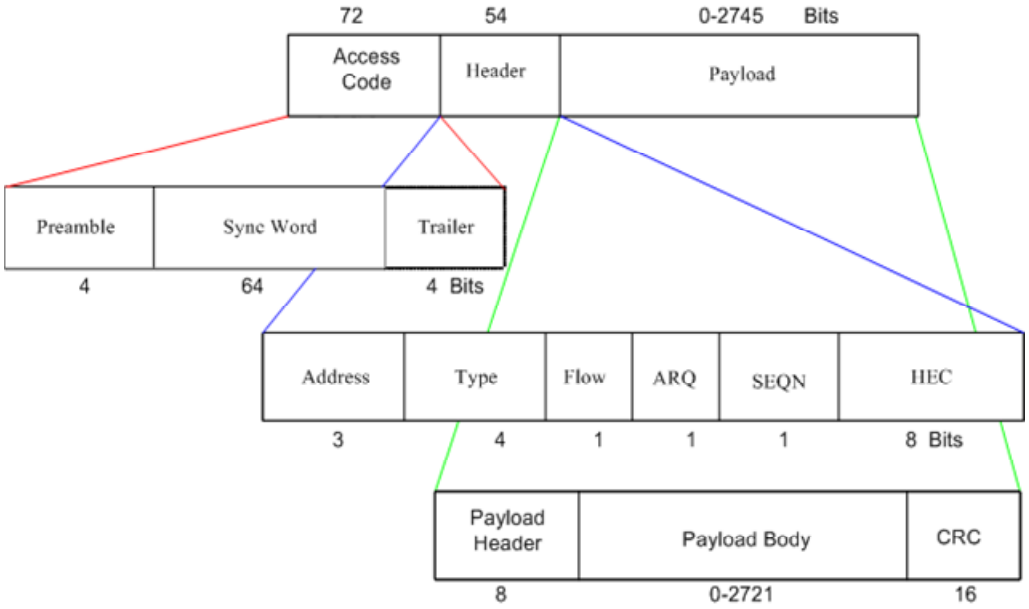


Figure 3-6 Shown are the contents of a Bluetooth packet [9].

The second way the encoder formats the data is by adjusting the "offs'" and "ons'" in certain patterns to represent digital ones and zeros. This formatting is called baseband coding. Some of the ways digital signals may be baseband encoded are shown in Figure 3-8. The Non Return to Zero (NRZ) encoding is likely the most familiar to the reader, as it is used in wired serial communication such as RS-232. Once the baseband signal is formatted by the encoder, it is modulated onto a carrier wave by the modulator, amplified and transmitted. Bluetooth uses the modulation scheme of Gaussian Frequency Shift Keying (GFSK) at a signal rate of 1 Mbps [9]. The specifics of GFSK are beyond the scope of this thesis but basically, a binary 1 is represented by a positive frequency deviation from the nominal carrier frequency, and a binary 0 is represented as a negative frequency deviation.

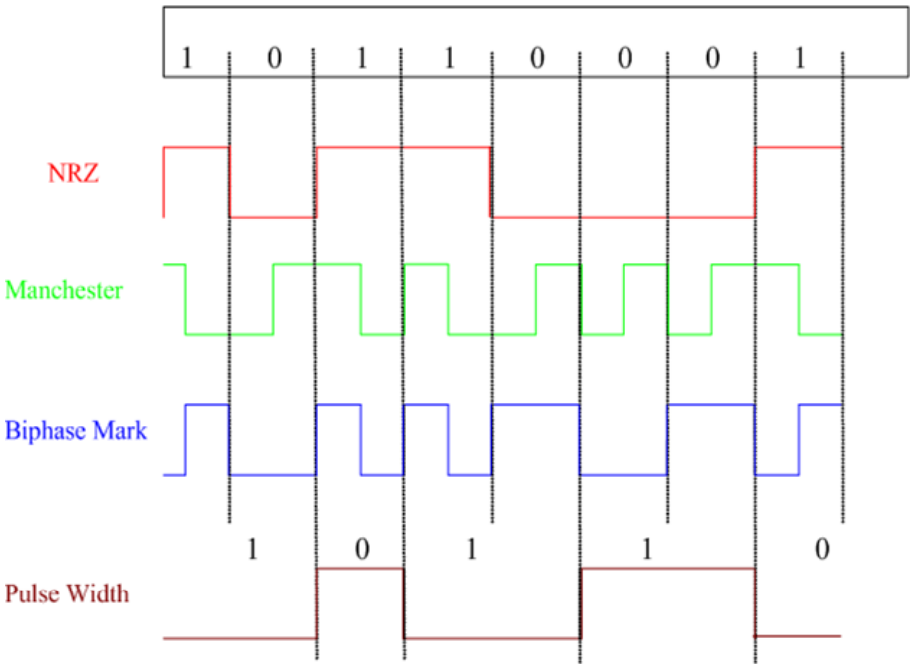


Figure 3-7 Shown above are a few examples of baseband coding (7).

When Bluetooth transmits a signal, it hops between 79 different frequencies at a rate of 1600 hops/s. This technique is called Frequency Hopping Spread Spectrum (FHSS). FHSS mitigates the effects of interference and multi-path fading, as data that is lost using a channel with temporary interference, may be retransmitted at a different frequency. A certain amount of security is also added since it is more difficult to intercept a signal that is hopping among different frequencies. For FHSS to properly work, the transmitter and receiver have to be synchronized. This synchronization takes place when wireless connections are initialized.

Now that the basic concepts of digital packets, encoding, and modulation have been presented, the Bluetooth software stack will be introduced. A software stack is a way to visualize a complicated software hierarchy where different layers have different functions. Higher layers generally have broader control over operations while lower layers control more specific processes. The layers of the Bluetooth stack work together to establish connections through the exchange of BT addresses, PINs, and device names and defining the procedural paths to be taken in the exchange of this information depending on modes of discoverability, connection-ability, pairing, and security [15]. The stack is responsible for controlling everything about the wireless connections and user interfaces that makes Bluetooth what it is. Thus, Bluetooth is not just a definition of radio hardware; it also defines the protocol architecture of the software stack.

Bluetooth functionality is dependent on the implementation of the Bluetooth software stack used on a particular Bluetooth chip. The Bluetooth Special Interest Group (SIG) creates detailed rules on the implementations of Bluetooth stacks and the specific configurations of each layer of the stack to achieve that functionality. The interested reader is encouraged to browse the SIG's web site at [15].

A Bluetooth software stack is shown in Figure 3-9. The highest layer of the stack is the Application Code, which provides an interface. The lowest layer is the physical radio. Although this isn't always the case, this protocol stack's layers are all loaded onto the Bluetooth module so that the wireless module may operate independently, interfacing to the host hardware through the application code. Because of the added burden that this configuration puts on Bluetooth's on-board MCU, the wireless modules functionality is generally limited and the throughput of the wireless link is restricted. The Bluegiga WRAP THOR Bluetooth module uses this configuration, interfacing to host hardware (PIC18F458) through Bluegiga proprietary ASCII interface code, which frees the PIC MCU from having any Bluetooth functionality at all.

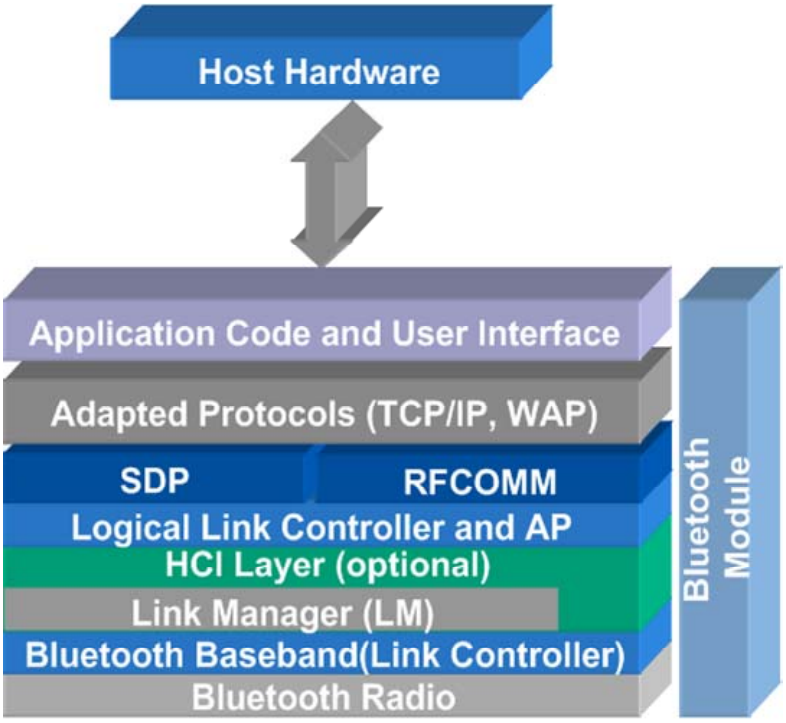


Figure 3-8 Shown is the Bluetooth protocol stack arrangement [10].

The functionality of the Bluetooth chip defines what sort of wireless connections it may make as well as the chip's available "profiles". Profiles define communication protocols that Bluetooth is capable of emulating. For example, the Bluetooth Serial Port Profile (SPP) may allow a Bluetooth chip to function as if it were a serial port. In other words, a Bluetooth module with the SPP is capable of accepting data from a standard serial port, transmit the data using Bluetooth protocol, and reassemble the data into standard serial port configuration on the other end. The hardware interfacing to the Bluetooth module doesn't even necessarily need to know that it's talking to a Bluetooth module and the wires are in fact missing entirely. This profile is useful for wireless serial port emulation or simply general wireless data transfer. The SPP is the profile that the WRAP THOR module runs, along with a proprietary application code to enable simple control of the wireless module. Other examples of profiles currently supported

by Bluetooth are Dial up Networking (DUN), FAX, Headset, LAN access, and File Transfer. For more information on Bluetooth profiles see [9].

Bluetooth was created with the intention of creating an environment of interoperability [9]. Devices that have a Bluetooth wireless chip in them are said to be Bluetooth "enabled". The idea behind Bluetooth is that Bluetooth enabled devices, may they be a laptop, cell phone, PDA, printer, mouse, or other electronic device will all be able to communicate with each other, automatically setting up wireless connections depending on the functionality one device has in common with another. Thus, Bluetooth is really intended to be a wireless "always on", "always working" wireless networking solution. The Bluegiga WRAP THOR module is then a very simple point-to-point implementation of Bluetooth technology.

This section discussed some specifics of digital RF communication and Bluetooth. The Bluegiga Bluetooth module was used for the neural signal transmitter because it automatically takes care of all of the issues of digital wireless communication and Bluetooth protocols. Further, virtually no external components were needed to use the Bluegiga Bluetooth module, simplifying the modules integration into a custom designed circuit board. Also, low power wireless products are subject to regulation [8], but since Bluetooth operates in the unlicensed ISM band, there was no concern of the need to be licensed. It is significant that the module may be used to communicate wirelessly using Bluetooth technology without having intimate knowledge of digital RF or the Bluetooth protocol stack. The next chapter will discuss the performance of the neural signal transmitter.

Chapter 4 . Performance of the Neural Signal Transmitter

In this chapter, the performance of the neural signal transmitter will be discussed. First, the performance of the PIC assembly language code was verified using a Microchip demonstration board. Then the performance of the wireless link of the neural signal transmitter was tested by performing a "file size test". In the file size test, the amount of data collected for a given amount to time was determined for a wired baseline and the wireless link, giving a quantitative measure of the quality of the wireless link. Two tools that helped in evaluating the prototype were the Microchip demo board and the BK Precision function generator. These tools will be introduced before discussing the PIC code and wireless link performances.

4.1 **Microchip PICDEM 2 Plus Demo Board**

Throughout the development process, one of the tools that made testing code more convenient was the PICDEM 2 Plus Demo Board from Microchip, pictured in Figure 4-1. The demo board was used to test the PIC code in place of the actual prototype device whenever possible for several reasons. First, there is a larger chance that a custom made board, with a hand soldered TQFP PIC on it, will introduce extraneous variables that make it more difficult to assess the performance of the code itself. Second, using the neural signal transmitter inevitably reduced the life span of the device because of wear and tear and the fact that PICs have a limited number of program write cycles. Finally, the electronics on the PICDEM board were not as small and difficult to work with as those on the neural signal transmitter.

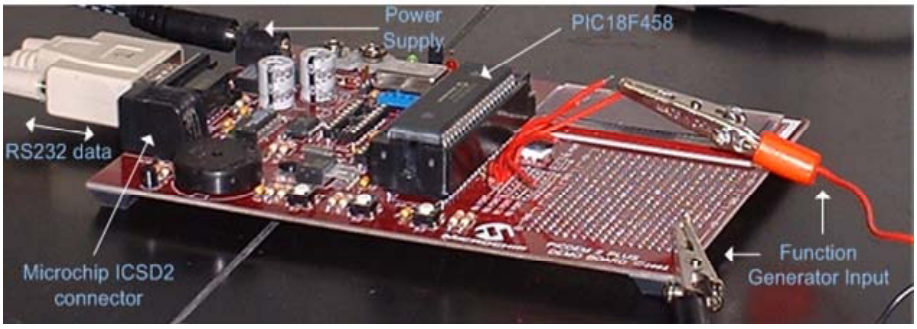


Figure 4-1 The Microchip PICDEM2 PLUS Demo Board came in handy during testing of the PIC code and was used as a wired baseline comparison for the file size tests.

The Microchip demo board has several features that make it convenient for testing code. First, the PCB has an on-board line driver to convert the MCU's UART data to RS232. The board could be plugged into a computer and the same LabVIEW program used to communicate with the prototype could be used to communicate with the demo board. The board also provides for easy access to I/O ports on the MCU through breakout pins on the PCB, which meant that the inputs to the ADC could be easily accessed. The demo board accommodates a variety of Microchip MCUs including the Dual In-line Package (DIP) PIC18F458. Except for its size, the DIP PIC18F458 is identical to the TQFP PIC18F458 that is used on the neural signal

transmitter. The board also has an ICSD2 connector, which made it possible to program and debug the MCU using the ICSD2. Seen on the right hand side of Figure 4-1 is the function generator input. This input was created by the BK Precision function generator, which is introduced in the next section.

4.2 **BK Precision Function Generator**

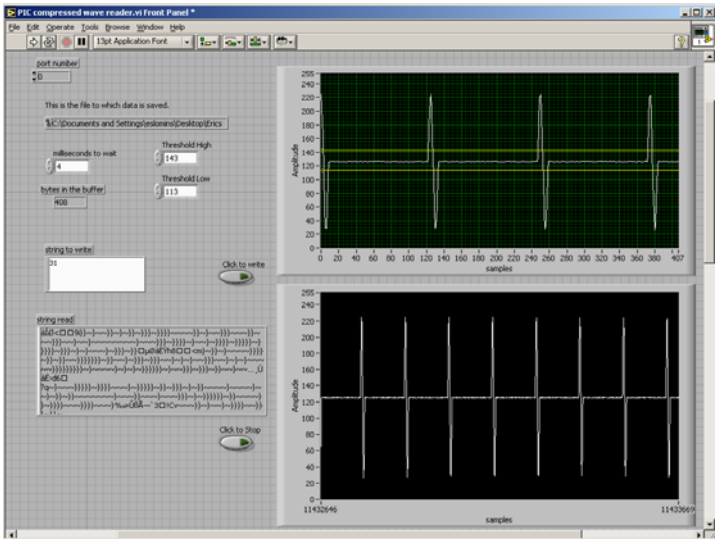
A BK Precision 4070A Multi Function Arbitrary Waveform Generator was used to create an idealized head-stage output signal. To model action potentials, the function generator was used in burst mode to create single-period sine waves at regular intervals. The output impedance of the function generator, at 50Ω, is very low, which was assumed to be similar to that of a head-stage because the output impedance of front-ends are low by design. The function generator output was ideal because the noise in the signal was very low and the "action potentials" occurred at a regular, adjustable interval.

The clean and regular nature of the function generator output signal allowed some aspects of the prototypes performance to be more accurately determined. For example, having a clean input signal meant that any output data that was noisy was indicative of noise introduced by the system, as it was not introduced by the input. In addition, having a sine wave "action potential" occur at a regular interval meant that the output data should also show the spikes occurring at regular intervals.

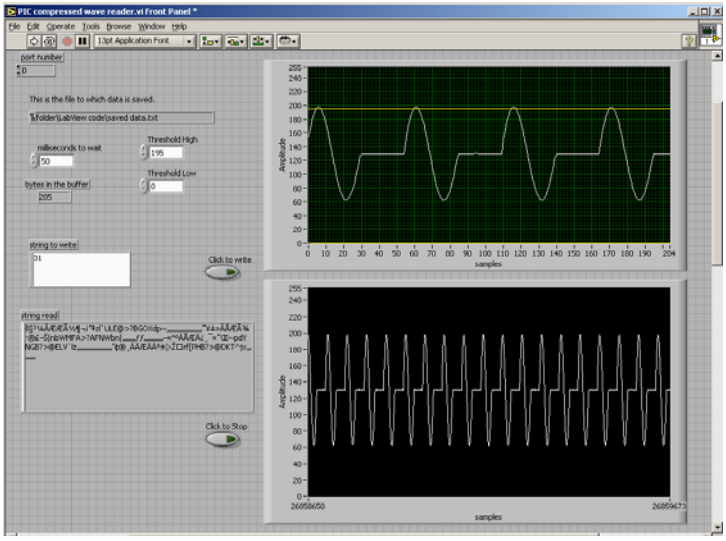
4.3 **PIC Code Testing**

The function generator output was adjusted to create a 1000 Hz sine wave at an interval of 10 ms with amplitude between binary 60 and 200 relative to the reference voltages of the PIC18F458 on the Microchip demo board. This signal was applied to one of the ADC inputs of the PIC. Shown in Figure 4-2, is the data collected on the PC.

In Figure 4-2a, is the data collected when the neural signal transmitter is operating in continuous mode. In continuous mode, the device samples at a constant rate of about 11 kHz. For a 1 ms burst sine wave, roughly 11 samples will be collected. Shown in Figure 4-2b, is the data collected from the prototype operating in waveform mode. In this mode, the device samples at about 30 kHz and keeps a five sample buffer; transmission begins upon threshold crossing. After transmission begins, sampling occurs for a fixed interval of 1.5 ms, after which there is a 5ms pause to allow the UART to finish transferring all of the samples. It can be seen that the device successfully isolates and transmits only the "action potentials" in the signal. The same test was performed for a variety of input signals, all resulting in the same, good results.



a.



b.

Figure 4-2 Shown are actual screen shots of data collected while the device wirelessly transmits the signal while in "continuous" mode (a) and "waveform" mode (b).

The ability of the neural signal transmitter to buffer five samples is illustrated in Figure 4-3. The figure shows a screen shot of the data collected when the function generator was adjusted to create a 1000 Hz sine wave every 500 ms with amplitude between 20 and 242. The Threshold level on the microcontroller was set to quantization level 240. An overlaid magnification of the signal clearly shows that there are five samples collected before the threshold level. Though it is not shown here, five buffered samples could also be observed when the input signal crossed the lower threshold level.

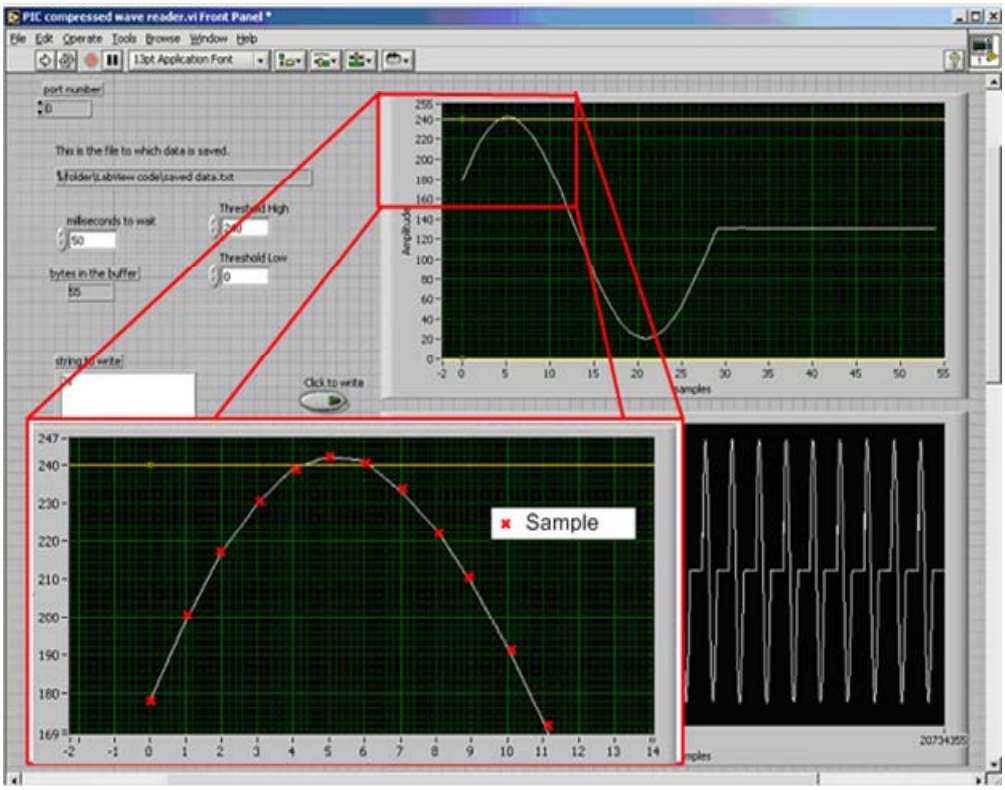


Figure 4-3 Shown is an actual screen shot with an overlaid magnification of the peak of the signal. It can be seen that 5 samples that occurred before threshold crossing have been buffered and transmitted by the prototype.

4.4 **Neural Signal Transmitter Testing**

It was apparent that during wireless operation data was occasionally lost. This loss of data was more apparent during continuous mode operation because it was easy to see that the gaps between each spike were sometimes not symmetrical as they should have been. This point is illustrated in the screen shot shown in Figure 4-4.

To get a more quantitative measurement of the quality of the wireless link of the neural signal transmitter, a test was devised to compare the amount of data collected using a wired baseline to that of the data collected over the wireless link. In this test, illustrated in Figure 4-5, the Microchip Demo board was fitted with a DIP PIC18F458 programmed with assembly language code identical to the code on the neural signal transmitter device. Identical function generator signals were applied to each device, and the sizes of the files collected over the course of 60 seconds were compared. This test was run in both continuous and waveform sampling modes, and was run with the wireless devices with both 2ft and 8ft of separation. During testing, a 3.7V DC power supply was used to provide power to the neural signal transmitter to eliminate the possible reduction in performance occurring if batteries were unable to provide enough peak current during times of high current draw.

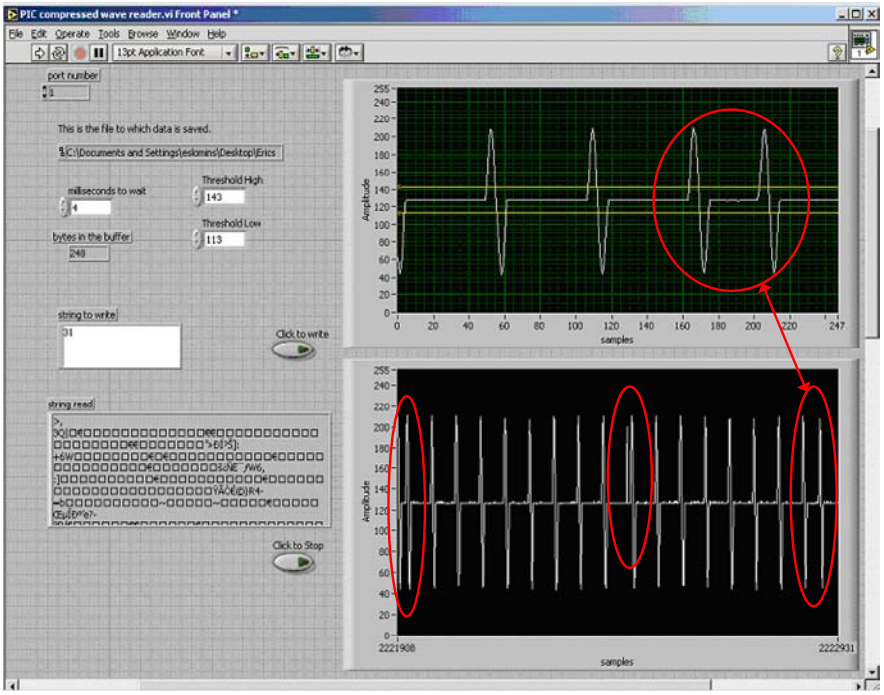


Figure 4-4 This screen shot has overlaid circles showing areas of the signal where data has been lost.

The output of the BK precision function generator was adjusted to produce a 1000 Hz sine wave periodically every 20 ms. Since the references of the PIC's ADCs for the Microchip demo board and neural signal transmitter were 0V and 5V, and 0V and 3.3V, respectively, the amplitude of the function generators output had to be scaled for each device. This was done so that the threshold settings would trip waveform sampling at the same relative voltage and to get the same amplitude signals on the output graphs of the LabVIEW front panel.

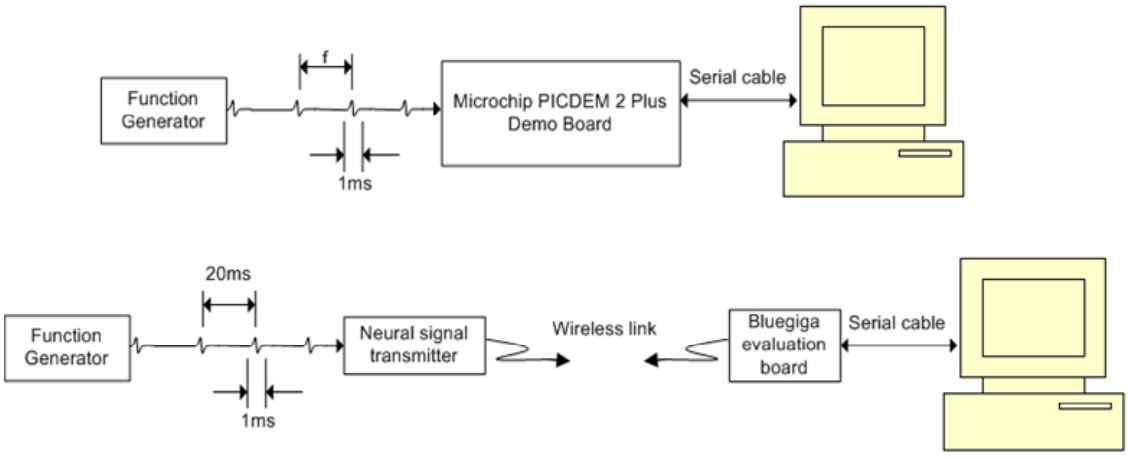


Figure 4-5 The performance of the prototype (bottom) was compared to the performance of the wired setup (top).

Shown in Table 4, is the data collected from 8 trials run for the wired baseline and the wireless tests at 2 ft. and 8 ft. of separation. The average number of samples collected during the wired continuous mode and wired waveform mode tests were 661 and 152 Kbytes, respectively. Since a byte and a sample are each 8-bits of data, the two words may be used interchangeably. For the wired continuous test, the average sampling rate was

$$661\frac{ksamples}{60s} = 11.02\frac{ksamples}{s} \approx 11kHz$$

which is close to the PIC's UART baud rate of 11.3 kHz. The discrepancy likely comes from the fact that, though the PIC's UART is sending information at 11.3 kHz, it cannot send constantly since there has to be some finite setup time. For example, loading the contents of the TXREG into the Transmit Shift (TSR) register, which shifts each bit through the UART transmit pin, has to take a certain amount of time[18]. Though the burst sample rate for the waveform test was around 30 kHz, the average sample rate was

$$152\frac{ksamples}{60s} = 2.53\frac{ksamples}{s} \approx 2.5kHz$$

which demonstrates how spike isolation reduced the average sample rate and thus, reduced the overall throughput. This reduction in throughput occurred despite a 30 kHz burst sampling rate in waveform mode as opposed to an 11 kHz sampling rate in continuous mode.

Over the course of a minute, for a spike every 20ms, the total number of spikes is

$$\frac{60s}{\frac{20\times10^{-3}s}{spike}} = 3000spikes$$

If the sample rate is 30 kHz as predicted, and the number of bytes buffered is 5, then the total number of samples per spike would be

$$30\frac{ksamples}{s}(0.0015s) + 5samples = 50samples$$

For 3000 spikes sampled over the course of a minute, then the expected file size would be

$$3000\frac{spikes}{\min}\left(50\frac{samples}{spike}\right) = 150\frac{ksamples}{\min} = 150\frac{kbytes}{\min}$$

which is actually a couple Kbytes fewer than the average for any of the waveform mode file size tests. This probably means that the sampling rate in waveform mode is actually slightly faster than 30 kHz. However, the predicted sample rate of 30 kHz was very close to the actual tested value. For the wired waveform mode test, a sample rate of 30.4 kHz results in 152 samples being collected, which is the average number of bytes collected in the wired waveform mode trials.

Table 4. Shown is the collected data from the file size test.

	File size (Kbytes)					
	Wired		Wireless			
			2ft		8ft	
	continuous	Waveform	continuous	Waveform	continuous	waveform
trial 1	660	152	646	148	400	148
trial 2	664	152	444	148	384	150
trial 3	659	152	561	148	602	150
trial 4	656	152	587	150	540	150
trial 5	663	152	436	148	652	150
trial 6	665	151	634	150	659	148
trial 7	662	153	516	149	656	148
trial 8	659	154	648	150	655	160
Avg.	661	152	559	149	569	151
std. dev.	3.0237158	0.8864053	86.276963	0.99103121	116.29273	3.964125

Shown in Table 5 are the averages and standard deviations for each of the wired and wireless tests. It can be seen that the average number of samples collected for the continuous mode tests was about 100 Kbytes greater for the wired baseline test than it was for the wireless trials. However, in waveform mode, the wireless trials only sampled a few less Kbytes of samples. This suggests that the wireless link may have been unable to cope with a constant transmission rate. It is unclear where the problem may have arisen, as there are many factors to consider.

Interestingly, the average number of samples collected for the wireless transmission across 2ft was on average slightly lower than for 8ft for both the continuous and waveform mode trials. One may intuitively reason that a signal that must travel farther to reach its destination would experience more attenuation, and thus a larger transmission distance would result in a poorer wireless link. This is not necessarily the case. When a signal is transmitted, especially indoors, the signal is reflected off ceilings, floors, furniture, and people, which causes the received signal strength to be the result of not one, but a vector sum of a multitude of RF signals [7]. We will call this the multi-path phenomenon. Multi-path introduces variation of signal strength, frequency distortion, and fading [7], which are all factors that affect the quality of the received signal. Thus, while transmission distance does play a part in the quality of the RF link, it is not the only factor. For more information on these topics, see [7].

Table 5 Shown are the averages and standard deviations for the wired, wired with a displacement of 2ft, and wired with a displacement of 8ft tests for both continuous sampling and waveform sampling modes.

	Continuous			Waveform		
	Wired	Wireless 2ft	wireless 8ft	Wired	wireless 2ft	wireless 8ft
average (kB)	661	559	569	152	149	151
standard deviation (kB)	3	86	116	1	1	4

Standard deviation is a measure of the spread of the sampled data and was calculated using the formula

$$s = \sqrt{\frac{\sum_{i}^{n} (x_i - \bar{x})^2}{n-1}}$$

where x is the number of Kbytes collected for a particular trial, $\bar{x}$ is the mean of the entire set, and n is the total number of trials or 8. It can be seen that the standard deviation for the continuous trials were greater than that for the waveform tests. This makes sense because the continuous test had a higher average throughput than that of the waveform test which would make variability in the testing times have more of an effect. For example, if an extra 0.5s was added to the recording time, in continuous sampling mode there would be an extra

$$11\frac{ksamples}{s}(0.5s) = 5.5ksamples = 5.5kbytes$$

of data. In waveform sampling mode there would only be an extra

$$2.5\frac{ksamples}{s}(0.5s) = 1.25ksamples = 1.25kbytes$$

of data. Since the trial times were timed by hand, it is reasonable to expect some variability in time and thus a greater standard deviation for the continuous mode tests.

What is interesting is how the standard deviations for the wireless continuous mode trials, at 86 and 116 Kbytes for 2ft. and 8ft. respectively were so much greater than the standard deviation of the wired continuous mode test. This again suggests a problem with the RF link during continuous transmission. The standard deviation in number of samples collected for the wireless trail at 8ft. is greater than for the wireless trial at 2 ft. This was likely the result of the added transmission distance, though it is not clear how the standard deviation in the number of samples collected at 8ft was greater, yet the average number of samples collected was higher as well.

4.5 **Future Work**

It was speculated that Bluetooth's robust packet transmission protocol, which includes error correction and packet re-transmission capabilities, would lessen the need for an antenna. Therefore, the neural signal transmitter was built without an antenna for

simplicity. Since the building of the neural signal transmitter, Bluegiga has began offering its WRAP THOR module on a circuit board with an integrated antenna, as seen in Figure 4-6. Rebuilding the neural signal transmitter with this module could potentially improve the radio performance of the prototype. However, a higher throughput wireless link would be needed to eliminate the necessity to pause after sampling an action potential, or if it was desired to sample multiple channels simultaneously.

Figure 4-6 The Bluegiga WRAP THOR module with integrated antenna is shown [6].

While etching a custom circuit board was an inexpensive and fast way to build the prototype neural signal transmitter, it also is prone to cause problems. For example, some of the etched traces initially had breaks in them due to inconsistencies in the somewhat crude etching method. Also, many of the passive components on the Bluetooth board were simply soldered to the proper electrical pads by wires instead of mounting them directly to the circuit board. This made construction of the circuit board simpler, as fewer traces needed to be etched, but it also made for less reliable hand soldered electrical connections. These factors may have contributed to the somewhat reduced performance of the prototype while in continuous mode of operation. To eliminate the potential problems introduced into the system by an improvised circuit board, a more compact, higher quality PCB could be made for a nominal fee from a PCB fabricating company such as ExpressPCB [19]. These companies can provide prototyping quantities of PCBs for a reasonable price.

Finally, the addition of time stamps would add more practical functionality to the neural signal transmitter. Time stamps could easily either be appended to the outgoing sampled action potentials by the PIC or assigned to incoming sampled waveforms by the LabVIEW software.

Chapter 5. Conclusion

Memory research would be greatly aided by the introduction of miniature wireless electrophysiology instrumentation. Unfortunately, wireless technology is somewhat limited in bandwidth which sometimes offsets its inherent benefits. This thesis presented a wireless neural signal transceiver that was constructed for the purpose of demonstrating throughput reduction through digital spike isolation *before* RF transmission. This technique makes efficient use of the limited bandwidth available in short range wireless devices, thus making wireless technology a more viable alternative to wired systems.

This proof-of-concept device was constructed using inexpensive commercially available electronic components. A Microchip PIC MCU was used for analog to digital conversion, processing of the neural signal to isolate spikes, and communication with a Bluegiga WRAP THOR Bluetooth module. The PIC18F458 MCU and its assembly language code were described in detail. Also described was the Bluegiga WRAP THOR Bluetooth module. Digital RF communication and the Bluetooth protocol were discussed not because these concepts were needed to implement the module, but because these concepts were needed to appreciate the module, as its simplicity is deceptive. The PIC code written for the PIC18F458 device was demonstrated to work properly, and the performance of the neural signal transceiver was measured.

Though it was shown that some data was lost over the radio frequency link during uninterrupted transmission in continuous mode, results showed that very little data was lost while isolating spikes in waveform mode. Possible reasons for the lost data were given and some future work was suggested for improving the prototype device. As electronics continue to become miniaturized and wireless technology improves, as it inevitably will, the applications for wireless technology will undoubtedly proliferate.

References

[1] University of Washington, "Lights, Camera, Action potential!!" http://faculty.washington.edu/chudler/ap.html

[2] Kandel, Eric R., Schwarts, James H. and Jessell, Thomas M., *Essentials of Neural Science and Behavior*, Appleton and Lange, 1995.

[3] Dictionary.com, http://dictionary.reference.com/

[4] Society for Neuroscience, "How do nerve cells communicate?", http://web.sfn.org/

[5] Neuralynx, "Wireless data acquisition", http://www.neuralynx.com/

[6] Bluegiga Technologies, "Products", 2003, http://www.bluegiga.com/

[7] Bensky, Alan, *Short Range Wireless Communication*, LLH Technology Publishing, 2000.

[8] RF Monolithics, Inc., *Virtual Wire Development Kit Manual for DR1200-DK and DR1201-DK*, Rev: B, 1999.03.05

[9] Bray, Jennifer, Sturman, Charles F., *Bluetooth Connect Without Cables*, Prentice Hall, 2001.

[10] Cambria Silicon Radios (CSR), "Training", http://www.csr.com/home.htm

[11] Microchip., "PICMicro Microcontrollers", 2003, http://www.microchip.com/1010/index.htm

[12] Nadel, Lynn, Cooper, Lynn A., Culicover, Peter, Harnish, R. Michael, editors, *Neural Connections, Mental Computation*, The MIT Press, 1989.

[13] Nicolelis, Miguel, A.L, editor, *Methods for Neural Ensemble Recordings*, CRC Press, 1999.

[14] Stiffler, Kent A., *Design with Microprocessors for Mechanical Engineers*, McGraw-Hill, 1992.

[15] Official Bluetooth Web Site, http://www.bluetooth.com/

[16] National Semiconductor Data Sheet, LP2986 Micropower 200mA Ultra Low Dropout Fixed or Adjustable Voltage Regulator, "Electrical Characteristics", Jan 2002.

[17] Microchip Data Sheet, PIC18FXX8, “Compatible 10-Bit Analog-To-Digital (A/D) Converter Module”, 2003

[18] Microchip Data Sheet, PIC18FXX8, “Addressable Universal Asynchronous Synchronous Receiver Transmitter (USART)”, 2003

[19] ExpressPCB, http://www.expresspcb.com/

Appendix A Companies That Provide Neural Signal Instrumentation

Plexon Inc., http://www.plexoninc.com/

Triangle BioSystems, Inc., http://www.tbsi.biz/

Bionic Technologies, http://www.bionictech.com/

Cambridge Electronic Design, http://www.ced.co.uk/smu.shtml

DataWave Tech., http://www.dwavetech.com/MainFrm.asp?area=zhome%5Chome

InstruTECH, http://www.instrutech.com/

MultiChannel Systems, http://www.multichannelsystems.com/

Neuralynx, http://www.neuralynx.com/

Panasonic MED, http://www.med64.com/

RC Electronics, http://www.rcelectronics.com/

Appendix B Assembly Language Code

```
        LIST P=18f458
        radix hex
        #include  <P18F458.INC>


;--------------RAM equates (All in Bank 5)----------------------
Position            equ     500h        ; table position pointer
Outer_count         equ     501h        ; used to store a counter value for delay routines
Inner_count         equ     502h        ; used to store a counter value for delay routines
BufferedSamples equ     503h        ; keeps track of the number of samples in the buffer
thresholdL          equ     504h        ; the lower threhold value
thresholdH          equ     505h        ; the upper threhold value
wavesample#         equ     506h        ; to store number of wavesamples in RAM
store_C             equ     507h        ; used to store the C bit of the STATUS reg on interrupt
Callz               equ     508h        ; number of times to call
Incwire             equ     509h        ; number of times to inquire
TimeOut             equ     50Ah        ; used to store a counter value for delay routines
ch_sel              equ     50Bh        ; data recieved by UART is saved to this register
counter             equ     50Ch        ; used to store variable for delay routines
shift               equ     50Dh        ; variable used for buffering routine
switch              equ     50Eh        ; used to switch between "continuous" and "waveform" modes
store_W             equ     50Fh        ; used to store the working register on interrupt
;-----------------------------------------------------------------------
        org     0x00                    ; reset vector
        goto    START

        org     0x0008
        goto    INT_SERVICE_1           ; primary interrupt vector
;-----------------------------------------------------------------------
START
        movlb   b'00000101'             ; bank 5
        call    USART_init              ; set-up the USART module
        movlw   d'1'                    ; number of times to inquire
        movwf   Incwire
        movlw   d'1'                    ; number of times to call
        movwf   Callz
        call    PC_BT_Connection        ; set-up wireless connection with PC BlueTooth
        call    ADC_init                ; set-up the ADC
        call    Timer_init              ; set-up the 16-bit timer
        movlw   d'143'
        movwf   thresholdH              ; set the HIGH threshold level
        movlw   d'113'
        movwf   thresholdL              ; set the LOW threshold level
        bcf             switch,0        ; start in "continuous" mode
        call    Interrupt_init          ; set-up the interrupts
        movlw   0x30
        movwf   ch_sel                  ; move 0x30 to 'ch_sel' to choose "Off" loop
        goto    loop1
;-----------------------------------------------------------------------
loop1
        movlb   b'00000101'             ; bank 5
```

```
        movf    ch_sel,W            ; get 'ch_sel' value for channel select
        sublw   0x30                ; compare to literal 30h, "Off"
        bz      Off                 ; branch if zero to Off

        movf    ch_sel,W
        sublw   0x31                ; compare to literal 31h, "CH1"
        bz      CH1                 ; branch if zero to CH1

        movf    ch_sel,W
        sublw   0x32                ; compare to literal 32h, "CH2"
        bz      CH2                 ; branch if zero to CH2

        movf    ch_sel,W
        sublw   0x33                ; compare to literal 33h, "CH3"
        bz      CH3                 ; branch if zero to CH3

        movf    ch_sel,W
        sublw   0x34                ; compare to literal 34h, "CH4"
        bz      CH4                 ; branch if zero to CH4

        movf    ch_sel,W
        sublw   0x35                ; compare to literal 35h, "SETUP"
        bz      SETUP               ; branch if zero to SETUP

        movf    ch_sel,W
        sublw   0x36                ; compare to literal 36h, "CHANGE_MODE"
        bz      CHANGE_MODE         ; branch if zero to CHANGE_MODE


        goto    loop1               ; return to the top of loop1
;-----------------------------------------------------------------------
Off
        bcf     ADCON0,ADON,0       ; turn off the ADC to save power
        bcf     T0CON,TMR0ON,0      ; stop the 16-bit timer
        goto    Off
;-----------------------------------------------------------------------
CH1
        movlw   b'10000001'         ; configures conversion clock, turns on ADC
                                    ; and selects channel (AN0)
        goto    ON
CH2
        movlw   b'10001001'         ; (AN1)
        goto    ON
CH3
        movlw   b'10100001'         ; (AN4)
        goto    ON
CH4
        movlw   b'10101001'         ; (AN5)
        goto    ON
;-----------------------------------------------------------------------
CHANGE_MODE
        btfsc   switch,0
        goto    $+8
        bsf     switch,0
        goto    Off
        bcf     switch,0
```

```
        goto    Off
;-------------------------------------------------------------------------
SETUP
        bcf     INTCON,GIE,0        ; turn off global interrupts
Message_1
; write 'Enter\sThresholdH/s/s/s/s/s'
        clrf    Position            ; initialize table position pointer
        call    Message_1_TABLE
        incf    Position
        incf    Position
        btfss   PIR1,4,0            ; Is TXREG empty? use access reg
        goto    $-2
        btfsc   PORTB,3,0           ; make sure BlueGiga is ready to recieve
        goto    $-2
        movwf   TXREG,0
        movf    Position,w
        sublw   d'42'
        BZ      $+8
        nop
        nop
        goto    $-20
        btfss   PIR1,RCIF,0
        goto    Message_1
        movff   RCREG,thresholdH

        ;---- clear flag-----
        movf    RCREG,W,0           ; to clear recieve flag, the RCREG must be empty (read to empty)
        btfsc   PIR1,RCIF,0         ; check that recieve interrupt flag is clear
        goto    $-4                 ; read RCREG again (RCREG is a double buffered register)
        bcf     RCSTA,CREN,0        ; toggle continuous recieve enable bit to clear possible
        bsf     RCSTA,CREN,0        ; overflow error and reenable receive

Message_2
; write 'Enter\sThresholdL/s/s/s/s/s'
        clrf    Position
        call    Message_2_TABLE
        incf    Position
        incf    Position
        btfss   PIR1,4,0            ; Is TXREG empty? use access reg
        goto    $-2
        btfsc   PORTB,3,0           ; make sure BlueGiga is ready to recieve
        goto    $-2
        movwf   TXREG,0
        movf    Position,w
        sublw   d'42'
        BZ      $+8
        nop
        nop
        goto    $-20
        btfss   PIR1,RCIF,0
        goto    Message_2
        movff   RCREG,thresholdL

        ;---- clear flag-----
        movf    RCREG,W,0           ; to clear recieve flag, the RCREG must be empty  (read to empty)
        btfsc   PIR1,RCIF,0         ; check that recieve interrupt flag is clear
```

```
        goto    $-4                 ; read RCREG again (RCREG is a double buffered register)
        bcf     RCSTA,CREN,0        ; toggle continuous recieve enable bit to clear possible
        bsf     RCSTA,CREN,0        ; overflow error and reenable receive
        bsf     INTCON,GIE,0        ; reset global interrupts
        goto    Off
;-----------------------------------------------------------------------
ON
        movwf   ADCON0  ; configures conversion clock, selects channel and turns on the ADC module
        clrf    BufferedSamples     ; initilize the bufferedsample reg
convert1
        call    Tacq            ; wait required acquisition time (to charge the sample and hold capacitor)
        bsf     ADCON0,2,0      ; start A/D conversion
Wait                            ; waits for A/D conversion to be completed
        btfsc   ADCON0,2,0
        goto    Wait

        btfsc   switch,0
        goto    Add_to_buffer
        goto    Continuous_mode

Add_to_buffer       ; This routine shifts down the first five wavesample regs and adds the new sample to
                    ;the top.  The oldest sample is then in register 000h and the newest  sample is saved in
                    ;register 004h
        movlw   d'4'
        movwf   shift
        lfsr    FSR0,000h       ; load FSR0 with 000h to initialize it
        lfsr    FSR2,001h       ; load FSR2 with 001h to initialize it
again
        movff   POSTINC2,POSTINC0   ; move file pointed to by FSR1 to file pointed to by FSR0
                                                                    ; and increment FSR1 and FSR0
        decfsz  shift,f             ; done shifting?
        goto    again               ; no, shift again
        movff   ADRESH,004h         ; yes, add newest sample to the top
        movlw   d'5'
        subwf   BufferedSamples,w
        BZ              Threshold   ; if bufferedsamples = 5, skip
        incf  BufferedSamples       ; one more buffered sample

Threshold       ; This routine checks to see if the aquired sample is outside of the
                ; threshold range defined by thresholdH and thresholdL.
        movf    ADRESH,w,0      ; move the upper byte of the ADC value into the working reg
         subwf  thresholdH,w
        btfss   STATUS,C,0
        goto    Waveform_mode
        movf    ADRESH,w,0
        subwf   thresholdL,w
        btfsc   STATUS,C,0
        goto    Waveform_mode
        goto    convert1                ; repeat

Continuous_mode
        btfss   TXSTA,TRMT,0        ; make sure the transmit shift register is empty
        goto    $-2
        btfsc   PORTB,3,0           ; make sure BlueGiga is ready to recieve
        goto    $-2
        movff   ADRESH,TXREG
```

```
        goto    convert1
;-------------------------------------------------------------------------
Waveform_mode
        movf    BufferedSamples,w       ; move BufferedSamples to the w reg
        sublw   d'5'                    ; subtract w from literal '5'
        movwf   FSR1L,0                 ; initialize the register pointer for the USART
        lfsr    FSR0,005h               ; initialize the register pointer for the ADC

        movff   BufferedSamples, wavesample#    ; initialize the wave sample number register
        clrf    TMR0H,0                 ; reset the upper (buffered) byte of the timer
        clrf    TMR0L,0                 ; resets the lower byte of the timer and moves
;TMR0H into the high byte of the timer
        bsf     T0CON,TMR0ON,0      ; enable the 16-bit timer (turn it on)
another_sample
        call    Tac             ; wait required acquisition time (to charge the sample and hold capacitor)
        bsf     ADCON0,2,0      ; start A/D conversion
        btfsc   ADCON0,2,0              ; waits for A/D conversion to be completed
        goto    $-2
        movff   ADRESH,POSTINC0         ; add new wave sample value
        incf    wavesample#             ; add one to wavesample#

        movf    wavesample#,w               ; This routine is to ensure 6 samples are saved
        sublw   d'6'            ; before UART may transmit.  (this value MUST be 6 or greater or
                                ; UART transmit will eventually not get enabled!)
        BNZ     $+4
        bsf     PIE1,TXIE       ; enable USART transmit interrupt bit (Interrupt is tripped when the
                        ; USART transmit buffer is empty and cleared when TXREG is written).
;---------check for time-out ------------
        movf    TMR0L,w,0       ; read lower byte of timer, which moves upper byte of timer into...
                                                        ; ...TMR0H regisiter.
        sublw   d'58'           ; subtact w from literal
        btfss   STATUS,C,0      ; sampled for 1.5ms?
        goto    ADC_done        ; yep, git outta here!
        bra     another_sample  ; nope
;-----------------------------------
UART_done
        bcf     PIE1,TXIE,0     ; disable TX interrupt
        retfie                  ; return from interrupt (auto resets the global interrupt bit)
ADC_done;
        bcf     T0CON,TMR0ON,0      ; disable the 16-bit timer (turn it off)
        call    Pause               ; pause for 5ms
        call  Clear_regs
        clrf    BufferedSamples      ; initilize the bufferedsample reg
        goto    convert1            ; go back to 'On' loop
;-------------------------------------------------------------------------------
INT_SERVICE_1

check_UART_RX
        btfss   PIR1,RCIF,0         ; UART recv interrupt?
        bra     check_UART_TX       ; not UART recv?

service_UART_RX
        btfsc   RCSTA,FERR,0        ; check for a framing error
        goto    Framing_err         ; there is a framing error (possible spurious input)
        movf    RCREG,W,0           ; get value from recv register
        movwf   ch_sel              ; save as 'ch_sel'
```

```
clear_flag
        movf    RCREG,W,0       ; to clear recieve flag, the RCREG must be empty (read to empty)
        btfsc   PIR1,RCIF,0     ; check that recieve interrupt flag is clear
        goto    clear_flag      ; read RCREG again (RCREG is a double buffered register)
        bcf     RCSTA,CREN,0        ; toggle continuous recieve enable bit to clear possible…
        bsf     RCSTA,CREN,0                    ; …overflow error and reenable receive
        pop                             ; discard the top of the hardware stack
        bsf     INTCON,GIE,0            ; manually reset global interrupts
        goto    loop1                   ; go check what ch selection was recieved

check_UART_TX
        btfss   PIR1,TXIF               ; UART trsmt interrupt?
        retfie          ; return from interrupt (global interrupt bit is automatically ( reset)
service_UART_TX
        bcf             store_C,0       ; save the carry bit from the STATUS reg

        btfsc   STATUS,C,0
        bsf     store_C,0
        movwf   store_W                 ; save the w reg

        btfsc   PORTB,3,0               ; make sure BlueGiga is ready to recieve
        goto    $-2
        movff   POSTINC1,TXREG
        movf    FSR0L,w,0                 ; move wavesample# to the w reg
        subwf   FSR1L,w,0
        BZ      UART_done

        bcf     STATUS,C,0              ; restore the carry bit
        btfsc   store_C,0               ; bcf doesn't affect C or Z bits
        bsf             STATUS,C,0
        movf    store_W, w

        retfie          ; return from interrupt (global interrupt bit is automatically reset)
;-------------------------------------------------------------------------------
Tacq
         ; This routine allows time for the sample and hold capacitor to charge.  This delay is 114 cycles
; long (including the call command) which is 11.4 micro-seconds with a 40 Mhz oscillator.

         movlw   D'36'
        movwf   counter
        decfsz  counter,f
        goto    $-1
        return
;-------------------------------------------------------------------------------
Pause
                        ; This delay is 5.02 ms with a 40 Mhz oscillator.

        movlw   d'65'
Outer_loop
        movwf   Outer_count
Inner_loop
        clrf    Inner_count
        decfsz  Inner_count
        goto    $-1
        decfsz  Outer_count
        goto    Inner_loop
```

```
        return
;-------------------------------------------------------------------------
Clear_regs
        lfsr    FSR2,000h       ; initialize register pointer
next
        clrf    POSTINC2
        movlw   d'60'
        subwf   FSR2L,w
        BZ              done
        goto    next
done
                return
;-------------------------------------------------------------------------
Framing_err
        bcf     RCSTA,CREN,0    ; toggle continuous recieve enable bit to clear possible
        bsf     RCSTA,CREN,0    ; overflow error and reenable recieve
        movf    RCREG,W,0       ; RCREG must be empty to clear recieve flag (PIR1,RCIF)
        btfss   PIR1,RCIF,0     ; check if recieve flag is clear
        retfie
        goto    Framing_err     ; read RCREG again (RCREG is double buffered)
;-------------------------------------------------------------------------
Timer_init
        movlw   b'00000111'
        movwf   T0CON,0         ; set 1:256 prescale value (upper limit on timer is ~1.677
                                ; seconds with 40 MHz oscillator), 16 bit timer/counter,
                                ; internal clock = fosc/4, timer disabled(not started)
        return
;-------------------------------------------------------------------------
Interrupt_init
        bcf     PIR1,RCIF,0     ; clear USART recieve interrupt flag
        movlw   b'11000000'
        movwf   INTCON,0        ; set global and peripheral interrupt bits
        bsf     PIE1,RCIE,0     ; enable USART recieve interrupt
        return
;-------------------------------------------------------------------------
USART_init
  movlw         b'10000000'
        movwf   TRISC,0         ; configure Port_C bits 6 and 7 for USART (inputs)
        movlw   b'10100100'
        movwf   TXSTA,0         ; configure for master mode, asynchronous mode
                                ; and high speed baud rate generator, enable transmit
        movlw   d'21'
        movwf   SPBRG,0         ; 113.636 kbaud @ 40MHz clock
                                ; (1.36% error when matched to 115.2 kbaud)
        movlw   b'10010000'
        movwf   RCSTA,0         ; enable USART and commence continuous recieve
        return
;-------------------------------------------------------------------------
ADC_init
        movlw   b'01000000'     ; for external Vrefs:         b'01001000'
        movwf   ADCON1,0        ; configure for left justification, appropriate ADC clock source and
                                ; A/D port configuration, use internal voltage references

        bsf    CVRCON,CVREN,0   ; disconnects VREF- from VSS to avoid excessive current draw
                                ; output impedance of VREF source must be less than 20 ohms (use
                        ; voltage follower). ("see PIC18FXX8 Rev.B4 Silicon/Data Sheet Errata"
```

```
                        ; problem number 7)
        ; ADC is NOT turned on here to save power. ADC module will be turned on when it is needed.

        return
;-------------------------------------------------------------------------------
PC_BT_Connection

;  Pause to give the Bluetooth Module time to start up and  spit out:
;  "WRAP THOR AI ($Name: AI-0-0-2 $ Jun  2 2003)
;  Copyright (c) 2003 Bluegiga Technologies Inc.
;  READY." (about 100 characters which takes about 8.6ms to transmit)

        call    QuietTime

INQUIRY
; Send "INQUIRY\s5\n" to the Bluetooth Module
        call    Delay
        clrf    Position
        call    INQUIRY_TABLE
        incf    Position
        incf    Position
        btfss   PIR1,4,0        ; Is TXREG empty? use access reg
        goto    $-2
        btfsc   PORTB,3,0       ; make sure BlueGiga is ready to recieve
        goto    $-2
        movwf   TXREG
        movf    Position,w
        sublw   d'20'
        BZ      wait
        nop
        nop
        goto    $-20
wait
        call    QuietTime
        decfsz  Incwire
        goto    INQUIRY
        goto  CALL_device

; This routing delays for about 5.3s

QuietTime

        movlw   d'3'
        movwf   counter
        movlw   d'175'
        movwf   TimeOut
        call    Delay
        decfsz  TimeOut
        goto    $-6
        decfsz  counter
        goto    $-10
        return


CALL_device
; Send "CALL\s00:03:0d:01:0f:6e\s1101\srfcomm\n" to the Bluetooth module
```

```
        call    QuietTime
        clrf    Position
        call    CALL_TABLE
        incf    Position
        incf    Position
        btfss   PIR1,4,0        ; TXREG empty?
        goto    $-2
        btfsc   PORTB,3,0       ; make sure BlueGiga is ready to recieve
        goto    $-2
        movwf   TXREG
        movf    Position,w
        sublw   d'70'
        BZ      $+8
        nop
        nop
        goto    $-20
        decfsz  Callz
        goto    CALL_device
        return

;  This delay is 10.04 ms with a 40 Mhz oscillator.

Delay
        movlw   d'130'
        movwf   Outer_count
        clrf    Inner_count
        decfsz  Inner_count
        goto    $-1
        decfsz  Outer_count
        goto    $-9
        return

;-----------------------------------------------------------------
        org 0x0400

INQUIRY_TABLE
        movlw   0x04
        movwf   PCLATH
        movf    PCL,w           ; read PCL to move contents of PCLATH into PCH
        movf    Position,w
        addwf   PCL,f
        retlw   A'I'
        retlw   A'N'
        retlw   A'Q'
        retlw   A'U'
        retlw   A'I'
        retlw   A'R'
        retlw   A'Y'
        retlw   0x20    ; '\s' (space)
        retlw   A'5'
        retlw   0x0a    ; '\n' (newline)

CALL_TABLE
        movlw   0x04
        movwf   PCLATH
        movf    PCL,w           ; read PCL to move contents of PCLATH into PC
```

```
        movf    Position,w
        addwf   PCL,f
        retlw   A'C'
        retlw   A'A'
        retlw   A'L'
        retlw   A'L'
        retlw   0x20    ; '/s' (space)
        retlw   A'0'
        retlw   A'0'
        retlw   A':'
        retlw   A'0'
        retlw   A'3'
        retlw   A':'
        retlw   A'0'
        retlw   A'd'
        retlw   A':'
        retlw   A'0'
        retlw   A'1'
        retlw   A':'
        retlw   A'0'
        retlw   A'f'
        retlw   A':'
        retlw   A'6'
        retlw   A'e'
        retlw   0x20    ; '/s' (space)
        retlw   A'1'
        retlw   A'1'
        retlw   A'0'
        retlw   A'1'
        retlw   0x20    ; '/s' (space)
        retlw   A'r'
        retlw   A'f'
        retlw   A'c'
        retlw   A'o'
        retlw   A'm'
        retlw   A'm'
        retlw   0x0a    ; '/n' (newline)
;-------------------------------------------------------------------------
        org 0x0500

Message_1_TABLE
        movlw   0x05
        movwf   PCLATH
        movf    PCL,w           ; read PCL to move contents of PCLATH into PCH
        movf    Position,w
        addwf   PCL,f
        retlw   A'E'
        retlw   A'N'
        retlw   A'T'
        retlw   A'E'
        retlw   A'R'
        retlw   A' '
        retlw   A'T'
        retlw   A'H'
        retlw   A'R'
        retlw   A'E'
```

```
        retlw   A'S'
        retlw   A'H'
        retlw   A'O'
        retlw   A'L'
        retlw   A'D'
        retlw   A'H'
        retlw   A' '
        retlw   A' '
        retlw   A' '
        retlw   A' '
        retlw   A' '

Message_2_TABLE
        movlw   0x05
        movwf   PCLATH
        movf    PCL,w           ; read PCL to move contents of PCLATH into PCH
        movf    Position,w
        addwf   PCL,f
        retlw   A'E'
        retlw   A'N'
        retlw   A'T'
        retlw   A'E'
        retlw   A'R'
        retlw   A' '
        retlw   A'T'
        retlw   A'H'
        retlw   A'R'
        retlw   A'E'
        retlw   A'S'
        retlw   A'H'
        retlw   A'O'
        retlw   A'L'
        retlw   A'D'
        retlw   A'L'
        retlw   A' '
        retlw   A' '
        retlw   A' '
        retlw   A' '
        retlw   A' '
;-------------------------------------------------------------------------
        END
```

Vita

Eric Christopher Slominski was born on January 13th 1980 in Williamsburg, VA. He is the son of Christopher and Cynthia Slominski. After graduating from York High School in Yorktown, VA, Eric began pursuing a degree in Mechanical Engineering from the Virginia Polytechnic Institute and State University in the fall of 1998. He graduated with the class of 2002 with a B.S in Mechanical Engineering and stayed at Virginia Tech to begin graduate school in the fall of 2002. Eric received a Masters of Science degree from Virginia Tech in December 2003.