I, Tyler S Witt, hereby submit this original work as part of the requirements for the degree of Master of Science in Electrical Engineering.

It is entitled:
A Modular, Wireless EEG Platform Design

Student's name: Tyler S Witt

This work and its defense approved by:

Committee chair: Fred Beyette, Ph.D.

Committee member: Carla Purdy, Ph.D.

Committee member: Paul Talaga,
A Modular, Wireless EEG Platform Design

A thesis submitted to the
Division of Research and Advanced Studies
of the University of Cincinnati
in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

in the School of Electronics and Computing Systems
of the College of Engineering and Applied Science

by

Tyler Witt
B.S. Electrical Engineering
University of Cincinnati, Cincinnati, Ohio, USA, 2013.

Thesis Advisor and Committee Chair: Dr. Fred R. Beyette

May 5, 2014
Abstract

Electroencephalography (EEG) is the measurement of electrical activity that takes place in the brain. EEG and the study of the waveforms thereby acquired are useful in the clinical and medical range. In addition to this, commercial applications and electronics are ever approaching the use of such technology as well. Moving forward in time, these technologies (as all technologies in general) are moving more towards becoming solutions composed entirely of embedded technology. As the technologies get smaller and more complex, the capabilities of current medical solutions are continually improving. The work presented in this thesis shows how embedded technologies can interface to completely comprise an entire EEG system.

Among all things, this design is modular and wireless. This novel design concept provides ways to take an existing solution and improve its portability and adaptability, without compromising the accuracy of the system as a whole.

The design itself is based on three main components: an Analog Front End, a 16-bit Microcontroller, and a Bluetooth Transceiver. This work will show that by interfacing these three main embedded devices, the results of a proven clinical EEG system can be replicated. Additionally, this work will show that this design can exceed in value the existing system due to its vast superiority in portability and adaptability.
Acknowledgement

I take great pride in and have a great appreciation for everyone who has helped me throughout the course of my Master’s studies at the University of Cincinnati.

I am first off very appreciative of my immediate family. My mother Judy, my father Dennis, and my brother Michael have always been supportive of my academic endeavors. Through the ups and downs, and all the delays, they were always there to push me as far as I could go.

I would also like to thank Dr. Fred Beyette for being so patient and helpful throughout this entire process. Under Dr. Beyette’s guidance, I have been able to expand my thoughts and interests in the embedded processing applications space, especially focusing on medical technologies. I would also like to thank any and every professor at the University of Cincinnati who imparted knowledge and wisdom to me in my years here. Specifically, I would like to extend a special thanks to Dr. Carla Purdy and Dr. Paul Talaga for serving on my defense committee and showing an interest in my work.

I would also like to thank all of my close friends that I have known throughout my time at the University of Cincinnati. When I needed a break from anything (or everything), I could always count on at least one of my friends to be there to give balance to my life.

The final group of people I would like to thank are those who were in the Point of Care System Design Laboratory (POCSDL) with me for my Masters studies and all the help and knowledge they imparted to me throughout this process.

So to all, a sincere, “Thank you.”
# Contents

1. **Introduction** ................................................................. 1
   1.1. Motivation ........................................................................... 2
   1.2. Background ....................................................................... 3
   1.3. Problem Definition ............................................................. 8
   1.4. Thesis Outline ................................................................... 9

2. **Platform Hardware Description** ....................................... 11
   2.1. High Level Description ....................................................... 11
   2.2. Analog Front End ............................................................. 13
   2.3. Microcontroller ............................................................... 16
   2.4. Bluetooth Transceiver ...................................................... 17
   2.5. Target Device ................................................................... 19
   2.6. Platform Hardware Final Design ........................................ 20

3. **Firmware/Software Description** ..................................... 23
   3.1. High Level Description ....................................................... 23
   3.2. Microcontroller Firmware ................................................. 24
   3.3. Target Device Viewer Software ........................................... 29

4. **Experimental Results** ...................................................... 32
   4.1. Experimental Setup ........................................................... 32
   4.2. Comparative Results ........................................................ 35
   4.3. Acquired Results ............................................................. 40

5. **Conclusion** ...................................................................... 45
   5.1. Confirmation of Results ..................................................... 45
   5.2. Future Works ................................................................. 48
5.3. Final Thoughts .......................................................................................................................... 51

A. Microcontroller Firmware ........................................................................................................ 53
B. Target Device Viewer Software ............................................................................................... 62
C. Additional Results .................................................................................................................... 65
D. Board Schematics and Layout ................................................................................................. 68
# List of Figures

1.1. Sections of the Brain ...........................................................................................................3  
1.2. Breakdown of the Forebrain ..............................................................................................4  
1.3. Baseline Brain Waves ..........................................................................................................5  
1.4. P300 6x6 Matrix ..................................................................................................................7  
2.1 Block Diagram of Platform Hardware ..............................................................................12  
2.2 Logical Block Diagram of ADS1299 ................................................................................13  
2.3 ADS1299 Input Multiplexer Block (Single Channel) .........................................................14  
3.1 UART Baud Rate Generation Equations ........................................................................25  
3.2 System Timing Diagram ....................................................................................................27  
4.1 10-20 Placement System ....................................................................................................33  
4.2 Final Hardware Design .......................................................................................................34  
4.3 g.USBamp Results – Complete Setup .............................................................................36  
4.4 g.USBamp Results – 8 Main Electrodes ............................................................................37  
4.5 g.USBamp Results – FP1 and FP2 EMG Events .................................................................37  
4.6 g.USBamp Results – FP1 ....................................................................................................38  
4.7 g.USBamp Results – FP2 ....................................................................................................38  
4.8 g.USBamp Results – O1 and O2 Electrodes .......................................................................39  
4.9 g.USBamp Results – O1 ......................................................................................................40  
4.10 g.USBamp Results – O2 ....................................................................................................40  
4.11 Designed System Results – FP1 .....................................................................................41  
4.12 Designed System Results – FP2 .....................................................................................42  
4.13 Designed System Results – O1 .......................................................................................42  
4.14 Designed System Results – O2 .......................................................................................42
4.15 Designed System Results (Filtered) – O1 ................................................................. 43
4.16 Designed System Results (Filtered) – O2 ................................................................. 43
5.1 Occipital Lobe Electrode – Comparative Result ......................................................... 46
5.2 Occipital Lobe Electrodes – Acquired Result .............................................................. 46
5.3 FP1 EMG Events – Comparative Result ...................................................................... 47
5.4 FP1 EMG Events – Acquired Result ........................................................................... 47
5.5 TI HRM Laptop Application ....................................................................................... 49
C.1 FP1 Electrode Filtered – Eye Blink Events ................................................................. 65
C.2 FP2 Electrode Filtered – Eye Blink Events ................................................................. 66
C.3 FP1 Electrode – Eye Blink Events .............................................................................. 66
C.4 FP2 Electrode – Eye Blink Events .............................................................................. 67
List of Tables

2.1 Data Rate(s)................................................................................................................................. 19

3.1 ADS1299 SPI Commands.................................................................................................................. 26

3.2 ADS1299 Registers.............................................................................................................................. 26
Chapter One

Introduction

Electroencephalography (EEG) is used for many clinical, medical applications, and is becoming widely used in the development of commercial applications. As a further expansion of this, medical procedures and technologies are starting to incorporate more embedded technologies into application designs. As this evolution of biomedical technologies begins to take shape, many proven and existing medical technologies are being improved upon based on the capabilities that newer technologies present. The work presented in this thesis is a prime example of improving existing biomedical systems by introducing newer technologies to the platform.

The research and design presented in this thesis aims to improve on existing systems by improving modularity, portability, and adaptability, while maintaining accuracy. To improve the modularity, a simple Analog Front End (AFE) with little pre-processed filtering is employed along with a simple 16-bit microcontroller and a Bluetooth transceiver. These hardware components are used to replicate what many other custom systems do, in a much more simple and modular design. Portability is improved by allowing the viewing device to not be physically tethered to the remaining portion of the device. Finally, this design is made to be as adaptable as possible. With comparable firmware and software applied to different target devices (as will be discussed later), this design can be used with a plethora of different end use devices. Therefore, by retaining the accuracy of the device and improving upon its modularity, portability, and adaptability, a novel design can be created to replace existing systems.
Motivation

As stated previously, EEG is a technology that is used widely within the medical, research, and commercial realms. In the medical realm, EEG is a very powerful and useful way to read biopotential signals from the human brain and relate these to a number of neurological pathologies [1]. A short list of these neurological pathologies includes epilepsy, coma, brain death, and a number of sleep disorders. As the diagnostics for any one of these conditions can be a difficult procedure, EEG has been shown to contribute meaningful information to aid in the diagnosis of these neurological pathologies.

In the commercial realm, EEG technologies show novel opportunities in a number of fields. One prime example is the vast number of systems that have been put into a category of “mind control” games. Some examples range from the simple Mattel’s MindFlex and the Star Wars Force Trainer from Uncle Milton, to the more complex, multiple electrode Emotiv EPOC. Each one of these devices uses the brain’s biopotential signals as “fuel” for the game, whether it be a reaction to an external signal or “self generated” actions [2].

While EEG is clearly a well developed technology, there are still improvements that can be made. Existing EEG systems (including the examples mentioned earlier) are all physically tethered by a wiring harness to electronics that can display the EEG signals visually to a doctor or other professional and/or do post acquisition processing of the EEG signals. Similarly, all of the existing EEG systems have power requirements that demand either wall plug connections or a battery pack that makes them impractical for sustained use of extended periods of time.

While these limitations present opportunities for system improvement in the design described here, it is still necessary to benchmark system performance against an existing state-of-
the-art EEG platform. In this thesis, we will use the g.USBamp 3.0 system [3] manufactured by g.tec Inc. as a baseline system for EEG signal capture. The g.USBamp allows for up to sixteen channels of EEG data acquisition per USB amplifier. This system also provides 24-bit resolution and will simultaneously sample all channels up to a maximum sampling rate of 38.4 kSPS. The design described in this work provides a replication of result quality on up to eight EEG channels, with improvements which make the designed system much more modular, portable, and adaptable, thus making this system an improvement on the basic system structure as a whole.

**Background**

To understand the functions and results of EEG testing, it is helpful to have a basic understanding of the sections of the brain, and their functions. The brain is comprised of three main sections, the forebrain, the midbrain, and the hindbrain. Figure 1.1 shows these sections of the brain and where they are located in respect to the brain as a whole. Thought production and thus the majority of EEG signals are produced in the forebrain.

Figure 1.1 – Sections of the Brain [4]
Forebrain, midbrain, and hindbrain (left to right)
As is shown in Figure 1.2, the forebrain is separated into a number of lobes and cortexes, each of which holds its own function. The first lobes to consider are the frontal lobes. As the picture shows, they are in charge of thinking, planning, and accomplishing executive functions. These lobes are located directly behind the forehead, and carry out the aforementioned functions by acting as a short-term storage site, keeping a single idea in mind while other ideas are being considered [4].

In the rear portion of the frontal lobe are the motor cortexes. As the name dictates, the motor cortexes are in charge of all voluntary movement of the body. Moving further back in the brain, the parietal lobes are found. The parietal lobes contain the sensory cortex which controls information about temperature, taste, as well as any other sensations the body can undergo. Each of these lobes also contributes to the reading and arithmetic functions that are carried out by the brain. Again, moving farther backward in the brain, the occipital lobes are found at the rear of the forebrain. The occipital lobes are extremely important as they are the sections of the brain that are in charge of comprehending visual stimuli and processing images into stored memory. The final lobes, found underneath parietal and frontal lobes, are the temporal lobes. The top of these lobes are responsible for receiving auditory information from the ears. Cognition of sounds and music are
also examples of the upper part of the temporal lobes. The undersides of these lobes are pivotal for forming and retrieving memories [4].

Now that a basic understanding of the different portions of the brain has been established, the basic details of electroencephalography (EEG) can be described. EEG is a non-invasive method of measuring the biopotential signals of the brain from the scalp [6]. When neurons (brain cells) are activated, they create a local current flow. EEG measures the flow of these local currents during synaptic excitations. To obtain basic brain patterns, also known as baseline measurements, subjects close their eyes and are told to “relax.” The baseline measurements appear sinusoidal and normally range from 0.5 to 100 µV in amplitude (around 100 times lower than ECG signals) [7].

By using a Fourier transform of the raw EEG, the power spectrum can be derived. Although the spectrum can range much higher, the baseline measurements of the subject have certain frequencies that are dominant. These frequencies are depicted in Figure 1.3. The alpha waves have been studied most extensively and can be usually observed better from the posterior and occipital lobes of the brain. These waves are induced through relaxation of the brain, as the magnitude of beta waves will drop and the magnitude of alpha waves will increase. Although the precise origin of this rhythm is unknown, alpha waves are usually attributed to “summed dendrite potentials” [8].

Figure 1.3 – Baseline Brain Waves [9]
As Figure 1.3 also explains, EEG is sensitive to a number of different states of brain usage. For example, when the eyes are open and the subject is alert, beta waves are dominant over the others. In a state of relaxation, the alpha waves take over as the dominant frequency range. Past regular awake relaxation, deep relaxation shows the theta waves as the dominating frequency, while a deep dreamless sleep shows the lowest frequency range, the delta waves, as being dominant [7].

Using an understanding of these brain wave characteristics, many EEG applications can be realized. A short list of conditions which can be medically monitored as a result of EEG includes coma and brain death, location of damaged areas of the brain following injury or stroke, and monitoring of epilepsy and location of seizure origin [8]. To undertake these applications, the proper analysis of the Fourier transformed frequency content must be done. A further application of EEG testing is to use the event related potential (ERP) technique. ERPs are significant voltage fluctuations resulting from evoked neural activity. These potentials are initiated by external or internal stimuli [8]. ERPs can be used to study both normal and abnormal cognitive processes. Perception, selective attention, language processing, and memory can all be assessed using ERPs with EEG. The amplitudes of ERPs tend to be much smaller than those of spontaneous EEG components, so they must be extracted from the set of baseline EEG signals. ERP thus reflects patterns of neuronal activity with high temporal resolution [7].

One main example of an ERP application is the P300 ERP. A P300-based application, such as the P300 Speller, has been shown to be an effective communication device when trained on a limited amount of data. To achieve results, a 6x6 matrix of letters and numbers, as shown in Figure
1.4, are flashed, in sequence either lighting up a single column or a single row. As the row or column containing the letter on which the subject is focusing flashes, the subject’s brain generates an ERP that can signify to the software which letter the subject is attempting to spell with [10]. This implementation, plus many others, sets a baseline for other brain-computer interface (BCI) technologies which can either focus on ERPs from the brain, or which can analyze the frequency content of the baseline EEG signals of an individual.

One final application of EEG which is the main focus of the application space of the design presented in this work is ambulatory EEG. Ambulatory EEG systems are designed to be portable and wearable devices. Ambulatory EEG is a portable platform designed for long term monitoring and offline evaluation of data. As such, it is ideally suited for applications such as epilepsy patient monitoring where occurrence/timing of neurologic events that occur during normal life activities is important to diagnosis and treatment. Unfortunately, existing systems can be heavy and bulky, and are comprised of multiple units, which include local storage as well as large battery packs. Another downfall of current ambulatory EEG systems is that they do not incorporate immediate viewing possibilities, and all of the data is stored for offline evaluation.

<table>
<thead>
<tr>
<th>DICE (D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABCDEFG</td>
</tr>
<tr>
<td>GHILJKL</td>
</tr>
<tr>
<td>MNOPQR</td>
</tr>
<tr>
<td>STUVWX</td>
</tr>
<tr>
<td>YZ1234</td>
</tr>
<tr>
<td>56789-</td>
</tr>
</tbody>
</table>

Figure 1.4 – P300 6x6 Matrix [10]
Problem Definition

It has been described in the previous sections, EEG is a vitally important form of medical diagnosis. Further, it has been noted that EEG is becoming a popular base technology for commercial applications, especially “mind control” games. In the vast majority of these cases, the EEG is either cheap and portable but lacks signal quality (like the MindFlex and Emotiv EPOC systems) or is accurate but remains bulky and expensive. Additionally, in the case of the higher performing systems, the hardware is often physically tethered to a bench top data acquisition and processing platform. To overcome these limitations, the EEG system developed here focuses on realizing an accurate EEG capture system that can wirelessly transmit EEG data to a laptop or handheld electronic device. Ultimately, this platform will be useful for a number of EEG and/or BCI applications. The formal definition of this problem is divided into three main properties in which the system should excel.

a. Quality Acquisition and Digitization of EEG Signals: This property serves to ensure that the accuracy and overall quality of the newly designed system can be compared to a known accurate system. For the case of this project, the g.tec g.USBamp 3.0 system will serve as a baseline for quality testing.

b. A Simplified, Modular Design to Improve Adaptability of the System: This property brings adaptability to the forefront of the design. Namely, the modularity of the system itself will cause it to be much more adaptable to other hardware and/or software which may be used with the system. This allows the end applications of the system to be widely varied, and the platform as a whole can become much more portable and useful.
c. Accurate, Wireless Transmission of Digital EEG Signals to a Target Device: This property is the main focus of the project. As the accuracy and quality of the system are an expectation, the novel portion of this design is its wireless nature. This property could lead to the realization of a subject being able to be monitored remotely. Along with this, the wireless nature of the device makes it much more portable than the existing cumbersome tethered systems.

**Thesis Outline**

**Chapter Two** will take an in-depth look at the hardware of the platform in its entirety. It explains why each of the individual main components were chosen, why they were configured the way they are, and how the system as a whole will be able to replicate the quality of signal transmission that existing systems demonstrate.

**Chapter Three** will describe the firmware and software that are a part of the basic system itself. This will detail the communication protocols used by the Microcontroller to control the peripheral devices as well as the basics of the viewing software.

**Chapter Four** will contain a detailed results comparison. The results of this system, as well as those that were captured using the g.tec system, will be compared to show the accuracy of the designed system. All customized parameters used in this data acquisition (such as which electrodes will be used) will also be detailed, allowing these experimental results to be replicated.

**Chapter Five** will detail the conclusions and future work that can be demonstrated with this system, as well as a few extensions that can be made to further improve the system.
Chapter References:


[4] “Brain Basics: Know Your Brain” Internet:


[9] Internet:

http://3.bp.blogspot.com/_tOcXPdhWUYA/SxO9H2i6CuI/AAAAAAAAAAABw/L250i2VIJTo/s320/alphabetatheta.jpg, 2013 [27 Dec 2013].

Chapter Two

Platform Hardware Description

High Level Description

The platform hardware described by this body of work was designed to simplify and improve existing designs. This is especially true in regards to portability, adaptability, and modularity. The platform hardware can be thought of as two main modules. On one end is the Acquisition and Transmission Module (AT Module), which – due to its apt naming convention – will acquire and transmit the acquired brainwave signals from the patient to the second main module. This AT Module can be further broken down into its main components, the Analog Front End, the Microcontroller, and the Bluetooth Transceiver. The final portion of the platform hardware that will be detailed in this Chapter is the second main module. This module is named the Target Device. This is the final end application device on which the data is received and the EEG waves can be viewed.
Figure 2.1 – Block Diagram of Platform Hardware

Figure 2.1 above is a simple block diagram of the entire Platform Hardware as well as the communication protocols that are used between the different components/modules. As can be seen from Figure 2.1, the user will wear a standard EEG cap with passive electrodes (this setup is further detailed in Chapter 4 under the subsection detailing the Experimental Setup). This EEG cap is then directly connected to the inputs of the Analog Front End. The Analog Front End then transmits the digitized brainwaves via a Serial Peripheral Interface (SPI) connection with the Microcontroller. The Microcontroller then continues the chain of events and transmits the digitized signals to the Bluetooth Transceiver via UART. The Bluetooth Transceiver finalizes the data path and transmits the brainwaves via Bluetooth to the Target Device. The Target Device then takes the digitized brainwaves and plots them in a capacity such that they can be viewed.
Analog Front End

The purpose of the Analog Front End is to acquire the evoked potentials created by the brain and transform these analog values into a digital value that can be processed and transmitted by the rest of the AT Module. There are dedicated EEG acquisition integrated circuits (ICs) commercially available which can provide the necessary amplification and analog-to-digital conversion [1]. For this design, the Texas Instruments ADS1299 was chosen to act as the Analog Front End for the system. The ADS1299 is a low-noise, multichannel, simultaneous-sampling, 24-bit delta-sigma analog-to-digital converter (ADC) with built-in programmable gain amplifiers (PGA), internal reference, and an onboard oscillator [2]. There are a total of eight channels.

Figure 2.2 – Logical Block Diagram of ADS1299
which can be simultaneously sampled at a rate of up to 16 kSPS. Figure 2.2 shows the ADS1299’s logical block diagram, including its general inputs and outputs.

A major advantage for the ADS1299 is the adaptability and configurability that it allows. Each input channel can be multiplexed and has many configurable signal-switching options. Figure 2.3 shows the input multiplexer block for a single channel. The inputs SRB1, SRB2, and BIAS_IN are common to all eight blocks of inputs, and VINP and VINN are separate in each of the eight blocks. The CHnSET registers (the Configuration Registers of the ADS1299 are detailed in the next Chapter) allow for further internal multiplexing. From device noise measurements, to test signals, and temperature sensing, the input multiplexing of the ADS1299 offers much adaptability to its front end. In addition to this, the CHnSET registers allow for each

Figure 2.3 – ADS1299 Input Multiplexer Block (Single Channel)
channel to have its PGA set to a specific gain level. This again lends to the adaptability of the ADS1299 [2].

Another reason the ADS1299 was chosen was for its accuracy and ability to take detailed analog readings. With the help of a digital decimation filter, the amount of filtering can be adjusted to either preserve the highest resolution of detail, or to increase the data rate. In the case of this device, the digital filter on each channel consists of a third-order sinc filter. The sinc filter decimation ratio can be adjusted for all eight channels in the CONFIG1 register. This implies that all eight channels will operate at the same data rate in a device [2].

As was referred to previously, the ADS1299 has a SPI-compatible interface. This is a very simple SPI interface that consists of four signals: /CS, SCLK, DIN, and DOUT. The /CS signal is an active low signal which serves as the “Chip Select” signal and must remain low for the entire serial communication duration. The SCLK signal is the SPI serial clock. SCLK is used to shift in commands and shift out data from the device. The DIN and DOUT signals are the data in and data out signals, respectively. The DIN signal is used to communicate with the ADS1299 (as will be outlined in the next chapter), while the DOUT signal is used to read out conversion and register data from the ADS1299. The data is shifted out on the rising edge of the SCLK signal and will only output data when /CS is held low. When /CS is high, the DOUT signal goes to a high-impedance state. This simple SPI module will allow the ADS1299 to communicate with the next piece of the Platform Hardware, namely the Microcontroller.
Microcontroller

The Microcontroller is the workhorse of the Platform Hardware. It is in charge of controlling all communications in and out of each of the rest of the devices in the AT Module, as well as any data conversion and data processing. In the case of this design, this incorporates communication via SPI to the AFE and communication via UART to the Bluetooth Transceiver. As such, this section of this Chapter will focus on the capabilities and peripherals of the Microcontroller that are relevant to this design.

The Microcontroller chosen for this design is the Microchip dPIC33FJ256GP710. This is a 16-bit Microcontroller with 256 KB of Flash memory and 30 KB of RAM. The dsPIC33 is capable of processing at 40 MIPS, which allows it to operate at a speed that is sufficient to carry out all of the communication, data conversion and data processing operations necessary. The clock is managed with an internal oscillator with programmable phased locked loops (PLLs) and oscillator clock sources. Not only does this device meet the speed requirements of the system, but it also meets the communication protocol requirements, namely SPI and UART communication protocols [3].

Both the SPI and UART modules are controlled simply through a combination of control registers and interrupt vectors. In addition to this, the SPI and UART modules on board the dsPIC33 are more than adequately expedient in their data transmit speeds. Specifically, the SPI module on the dsPIC33 is able to run at a top speed of 20Mbps (assuming clock speed of 40MHz, maximum speed of SPI module is one half the main clock frequency). The SPI module then uses a 4-wire system where one wire is dedicated as the data in (DIN), another dedicated as data out (DOUT), a third as the system clock (SCK), and the final as the chip select (CS) wire.
The SPI module is activated by lowering of the CS wire (an active low signal) and then all data transmitted via the DIN or DOUT wires is recognized as real transmissions.

Conversely, the UART module uses just two wires, transmit (TX) and receive (RX). Additional hardware control may be implemented using the UART’s clear to send (CTS) and the ready to send (RTS) pins on the dsPIC33. However, for this application, the hardware control of data flow was not needed. The UART module on the dsPIC33 is capable of speeds up to 10Mbps, which far exceeds the maximum bandwidth for applications of interest and the possible speeds of the RN-42 (as will be discussed in a later section).

Another important peripheral that is used on the dsPIC33 is the Direct Memory Access (DMA) Controller. This controller is a bus master module useful for data transfers between different devices without CPU intervention. Both the SPI and UART communication modules of the dsPIC33 can be used with DMA. This particular DMA controller has four channels, which will either auto-increment the source or destination address registers depending on the wants of the user. This will be useful after data is ready to be transmitted via the Bluetooth Module. No CPU intervention will slow the process down, and the highest possible transfer speed will be achieved, thus giving the best performance for the entire system [3]. The specific configuration of the DMA controllers will be discussed in the next Chapter in the section describing the Firmware of the Microcontroller.

Bluetooth Transceiver

The third and final portion of what makes up the AT Module is the Bluetooth Transceiver. The Bluetooth Transceiver is a vitally important part of the overall design and must
be consistently able to operate reliably in a fashion such that the EEG signals being captured and processed by the earlier sections are being sent successfully to the Target device. Not only must this transmission be reliable, but it also must be able to operate at speeds that will allow for detailed sampling rates, which will ensure a sufficient accuracy for the system as a whole.

For this portion of the AT Module, the Roving Networks RN-42 was selected [4] [5]. This particular transceiver was chosen due to its simple interfacing and programming via a microcontroller. The RN-42 can be controlled via two different communication protocols, UART or USB. In addition to this, while the USB communication can only be used for the Human Computer Interface (HCI) communication interface, UART can be used for either HCI or for Serial Port Profile (SPP). When the device is setup for the HCI communication interface, a software Bluetooth stack must be designed and implemented on the host controller (in this case the dsPIC33). To forego this requirement, the RN-42 has a firmware based, pre-programmed on-board Bluetooth stack that can be utilized when the device uses the SPP interface. This allows the controller to send data via UART to the RN-42 and the RN-42 will transmit this data via Bluetooth, and vice versa for incoming data from the Target Device (mainly control commands).

Beyond the simplicity of control, as was alluded to earlier, the speed and reliability of the Bluetooth transceiver is of the utmost importance. When the device is placed in SPP mode, the highest data rate that can be achieved (while the RN-42 is in the required slave mode), is 240Kbps [4] [5]. Conversely, if the device is placed in HCI mode, it can reach data rates nearing 1.5Mbps in sustained communications. Clearly the HCI mode is capable of retaining the necessary quality of signal required to be as accurate if not more accurate than existing EEG systems. The real question for this system was if the SPP would be sufficient in retaining enough signal quality such that the captured signals would be sufficiently accurate in comparison to
existing systems. Table 2.1 shows the mathematical side of figuring in the maximum samples per second (SPS) that could be driven from the ADS1299, through the RN-42 in SPP slave mode. This shows that the ADS1299 would be able to transmit eight channels (plus status bits) at 1000SPS in SPP mode. This sampling rate is sufficient for quality EEG detection and recording.

### Target Device

The final piece of hardware that is required for this EEG platform is a final target device on which the EEG can be viewed. For this purpose, no one specific device is required. In fact this system is designed to be as adaptable as possible for any number of extensions. As such, any Bluetooth enabled device can be used as the Target Device for this platform. That is to say, if proper software were created along the lines of that created for this EEG platform (as will be detailed in the following chapter), a personal smartphone, an iPad, or another Bluetooth capable device could be used as a viewer for the EEG signals.
Platform Hardware Final Design

To implement this entire system, a number of design goals were set. The first of these goals was to use as little hardware as possible to keep the system simple and modular. The second goal was to design the hardware platform so that it could be customized or applied to other applications with different hardware. To achieve this, the design employs a Microchip standard for Experimenter’s Boards, which includes a main Motherboard and Daughter Cards. The Motherboard and Daughter Cards board layout files were designed with EAGLE CAD software, then these files were sent to Advanced Circuits to be fabricated. These boards were then populated in house and tested for functionality.

The Motherboard design was based off of the Microchip Explorer 16 Experimenter’s Board [6], after having removed extraneous components and systems that were not required for the end application. Thus, the Motherboard was comprised of headers for a Microchip Plug-in Module (PIM), a basic programming interface for the Microchip ICD (In-circuit Debugger) 3, power supplies (including +3.3V and +5V supplies), and two daughter card slots. The main point of focus for this board should be and is the daughter card slots. These slots are designed based off of a standard put in place by Microchip so that the on-board supplies, as well as every single pin on the PIM will be accessible by each daughter card slot. Having access to all the pins is very helpful for prototyping/design purposes. The end result was a simple board with two Daughter Card slots that can be reliably and effectively programmed for any number of applications. As far as the Daughter Cards are concerned, only an ADS1299 Daughter Card was designed. For the Bluetooth side of the design, an already created Evaluation Module (EVM) was used. Initially, a
previously created Daughter Card was to be used. The RN-42 on this card could only operate in HCI mode, so the EVM was employed in its place.

The design of the ADS Daughter Card took advantage of the Motherboard’s capability to produce a 5V supply and used this for a unipolar supply for the ADS1299. As such, it was simply created to handle the inputs through a RS232-like port and communicate via the dsPIC33 directly. However, the design of this Daughter Card proved to have many erratum. Included in this was a mistake in the reference design and in the clock select pin (which was left floating). Although the design and fabrication of this board comprised a large amount of effort and time, the end result was reverting back to the initial EVM designed by TI.

The final prototype setup can be described as the daughter board from the TI ADS1299 EVM connected to the EVM motherboard for power only. This daughter board was also connected via air wires to a prototyping Daughter Card that used air wires to connect the communication and ground pins between the EVM and the Motherboard. From the Motherboard was another prototyping Daughter Card that had all the necessary connections with the RN-42 EVM via air wires as well. All of this was then paired to the Target Device, a laptop running a MATLAB program which would acquire, store, and display the data on the screen.

Although this setup is not entirely ideal, it was sufficient to achieve test results necessary for validation of the design concepts. A shortened timeline and last minute discovery of very important design flaws caused compromises in the overall look of the design. However, these changes did not affect the performance and quality of the system as a whole.
Chapter References:


Chapter Three

Firmware/Software Description

High Level Description

This section describes all of the coding (both firmware and software, heretofore noted as the System Code) that was involved in the use of this device. While the software communicated here only serves a small purpose, which was vital for the testing of the overall system, the System Code can be replicated on a number of different end target hardware platforms and can be modified simply to adapt to end user applications. The adaptability and reuse of this code was paramount as this system is set up to be a “starting point” for more intricate usage of EEG and BCI technologies in the future.

The System Code is comprised of two main aspects, the Microcontroller Firmware (or Firmware) and the Target Device Viewer Software (or Software). The main aspects of the Firmware were described by the functionalities of the AT section of the Platform Hardware Description. Namely, the Microcontroller (or MCU) is first initialized and then set up to use its peripheral modules to initialize the other hardware, then receive data and finally transmit the received data onto the Target Device.

The Software was set up on a laptop computer. The only requirements for this Software are that the laptop (or other computing device) be Bluetooth enabled and have MATLAB installed. MATLAB provides an environment where a large portion of data processing can be
done quite easily and on the fly if needed, which is very helpful for many EEG applications. The Software described does not do any filtering or processing of the data, but rather checks the data (more specifically, the status bits) for accuracy and displays the raw EEG of the different channels on a graph. Further implementations may be done to analyze the frequency content, as will be described later.

In short, the entirety of the System Code is designed to be a “bare bones” implementation from which many things can be built up. By simplifying the design, the results achieved are as raw and real as can be possibly acquired. In addition to its simplicity, the System Code can also be adapted for a variety of uses. These adaptations will be described in detail in the upcoming sections.

**Microcontroller Firmware**

As stated previously, the MCU Firmware is packaged and created to be as simple and adaptable as possible. The Firmware code (which is attached as an Appendix) is comprised of three main C source files and two C header files. It is broken down in such a way that there is an initial init.c file that will handle the initialization of all systems and modules used in the entire system. The second C source file is ADS.c, which contains all the commands that serve as an Application Programming Interface (API) specific to this ADS and the applications that it can be used for. The final C source file is the main.c file. This file not only calls all the commands of the previous two files, but also is in charge of packaging the data and sending it in a way that the Target Device Viewer Software is familiar with.
To be specific, the init.c file initializes the clock system, all of the required General Purpose I/O Pins (GPIO’s), the SPI module, the UART module, and the Direct Memory Access (DMA) module. The clock system is setup in a way where the external 8.00MHz crystal is the main clock source for the device, and this is fed into the device’s Phase Locked Loop (PLL) to achieve a final oscillation frequency (Fosc) of 80MHz. The operating frequency (Fcy) of the device is halved to 40MHz. The setup of the GPIO’s is very straightforward. Each peripheral is required to have certain pins be inputs and others to be outputs. Setting the device’s TRISx registers initializes the input/output orientation of the peripheral pins. Holding pins high or low can be achieved by setting the proper bits in the LATx registers. The SPI module was set up by dividing the input frequency (Fcy) by eight to achieve the fastest possible communication with the ADS (5MHz). As will be discussed later, the SPI was not setup on an interrupt basis, as it could be controlled by the MCU much more effectively. The UART was then setup in addition to the DMA. The UART baud rate was setup using the equations in Figure 3.1. The TX and RX interrupts for the UART module are also enabled and the system then sets up the DMA module. The DMA is then mapped to the UART Transmit Register (U1TXREG) and functionality, set up as a continuous reading of the DMA RAM and transmits until the appropriate number of requests has been completed. The number of requests is determined by the number of channels and the number of samples being taking per transmission [1].

\[
\text{Band Rate} = \frac{F_{CY}}{16 \times (UxBRG + 1)} \quad \text{(1)}
\]

\[
UxBRG = \frac{F_{CY}}{16 \times \text{Band Rate}} - 1 \quad \text{(2)}
\]

Note: \( F_{CY}\) denotes the instruction cycle clock frequency (Fosc/2).

Figure 3.1 – UART Baud Rate Generation Equations [1]
### Table 3.1 – ADS1299 SPI Commands [2]

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>First Byte</th>
<th>Second Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAKEUP</td>
<td>Wake-up from standby mode</td>
<td>0000 0010 (02h)</td>
<td></td>
</tr>
<tr>
<td>STANDBY</td>
<td>Enter standby mode</td>
<td>0000 0100 (04h)</td>
<td></td>
</tr>
<tr>
<td>RESET</td>
<td>Reset the device</td>
<td>0000 0110 (06h)</td>
<td></td>
</tr>
<tr>
<td>START</td>
<td>Start and restart (synchronize) conversions</td>
<td>0000 1000 (08h)</td>
<td></td>
</tr>
<tr>
<td>STOP</td>
<td>Stop conversion</td>
<td>0000 1010 (0Ah)</td>
<td></td>
</tr>
</tbody>
</table>

### Data Read Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>First Byte</th>
<th>Second Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDATAC</td>
<td>Enable Read Data Continuous mode. This mode is the default mode at power-up (1)</td>
<td>0001 0000 (10h)</td>
<td></td>
</tr>
<tr>
<td>SDATAC</td>
<td>Stop Read Data Continuously mode</td>
<td>0001 0001 (11h)</td>
<td></td>
</tr>
<tr>
<td>ROATA</td>
<td>Read data by command; supports multiple read back.</td>
<td>0001 0010 (12h)</td>
<td></td>
</tr>
</tbody>
</table>

### Register Read Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RREG</td>
<td>Read ( n \text{ mm} ) registers starting at address ( r \text{ mm} ).</td>
</tr>
<tr>
<td>WREG</td>
<td>Write ( n \text{ mm} ) registers starting at address ( r \text{ mm} ).</td>
</tr>
</tbody>
</table>

(1) When in RDATAC mode, the RREG command is ignored.
(2) \( n \text{ mm} \) = number of registers to be read or written – 1. For example, to read or write three registers, set \( n \text{ mm} = 0 \) (0010). \( r \text{ mm} = \) starting register address for read or write opcodes.

As for the API (application programming interface) set up in ADS.c, there are a number of functions that are outlined in the ADS1299 Datasheet. First, the power up sequencing that is required to power up the ADS1299 chip is initialized. Specifically, the ADS1299 should first be allowed a second to have the power supplies properly settle. Next, the RESET pin of the ADS is held low for at least two ADS clock cycles, then re-raised and held high for at least eighteen ADS clock cycles before progressing further, per the specifications of the ADS. Also included in the API set up is the ability to send any of the commands listed in Table 3.1 [2]. For this application, only the START, STOP, and SDATAC commands are used.

The final use of the API set up is to allow the writing and reading of all the ADS1299 registers. These registers are listed in Table 3.2, which shows the specific register settings, and a

### Table 3.2 – ADS1299 Registers [2]

<table>
<thead>
<tr>
<th>Register</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG1</td>
<td>0x95</td>
<td>250SPS, Daisy Chain Mode, OSC output disabled</td>
</tr>
<tr>
<td>CONFIG2</td>
<td>0xD0</td>
<td>Default setting, internal test signals</td>
</tr>
<tr>
<td>CONFIG3</td>
<td>0xE0</td>
<td>Reference buffer enabled, no BIAS measurements</td>
</tr>
<tr>
<td>MISC1</td>
<td>0x20</td>
<td>SRB1 switches closed</td>
</tr>
<tr>
<td>GPIO</td>
<td>0x0F</td>
<td>Default setting, all GPIO’s input</td>
</tr>
<tr>
<td>CHnSET</td>
<td>0x60</td>
<td>Normal operation, max PGA gain, normal electrode input</td>
</tr>
</tbody>
</table>
short description of what the settings mean.

The final portion of code is the main.c file, which is the controlling piece of Firmware for the system. After the functions of the other files are called with appropriate timing, the main.c file initializes a pointer to the start of the DMA RAM. This pointer serves as a place marker for the SPI module to write to. Once the UART has completed its transmission (which will be caused by a DMA interrupt), the pointer will be reset to the start of DMA RAM. The entire system is controlled via UART at this point. Unless the UART module receives a certain character, no data will be sent. Upon the specified character reception, in this application the ‘^’ character, a port pin will be toggled to enable/disable transmission of data. In this way, the ‘^’ character serves as a flow control for the host Target Device to control the MCU and ADS (as will be discussed in the following section). When the MCU and ADS are enabled for transmission, the system will continuously transmit data via UART. Because the SPI module is operating at such a higher speed than the UART, it is forced to wait on the UART to complete its transmissions before continuing to transmit data. Figure 3.2 shows a timing diagram for this scenario.

![System Timing Diagram]

Figure 3.2 – System Timing Diagram
As can be seen, the system has no communication before ‘^’ is received via UART. Once this happens, both the SPI and UART are released, and the data streams through in 24-bit packets. This Timing Diagram shows the case when a single sample is taken and simultaneously transmitted via UART with only four channels enabled. Once the ‘^’ is received, the SPI module sends a Read Data Command (“C”) to the ADS1299, which will allow the ADS to send the sample via SPI and, by using the pointer referenced earlier, saving the sample data in DMA RAM. When the SPI gets the entire sample, it will not receive or send anything until the DMA completes the UART transfer of the data already received. Once the DMA completes, the SPI will send another command to receive a new sample. This continues until another ‘^’ is received via UART, at which point the system ceases all communications and the pointers for the DMA and SPI are re-initialized.

In this way, the Firmware packages and communicates the EEG signals with the Target Device. Another thing to note is that the ADS (and, therefore, the MCU) will output this data as a two’s compliment of the actual value, allowing for both positive and negative values to be represented [2]. As such, the Viewer Software will know the formatting and packaging of the data that it is receiving and will be able to process and display the data as needed. The Firmware described previously was very simply done. As can be seen from the Timing Diagram, the Firmware can be adapted to a certain amount of input channels and the number of consecutive samples that will be taken. This was accomplished so that a number of different EEG or BCI applications can be achieved by the system.
Target Device Viewer Software

Similar to the Firmware, the Target Device Viewer Software was created with simplicity and adaptability at the crux of its design. The Viewer Software itself is comprised of only two pieces of code (this code is also attached as an Appendix). All of the code for this Software is written in MATLAB, which was chosen for its ability to easily create extension programs for future usage (as will be detailed in later sections). The first piece of code is ADS_serial.m which controls the creation of the serial object and the real-time plotting of the input signals and display on the Target Device. The second piece of code is twos2decimal.m which takes the two’s compliment data that the MCU sends to the Target Device and turns it into proper decimal values. Outside of the Viewer Software, one extra piece of code was created and used to derive and display the frequency content of the signal, which is often the more important portion of the EEG signals.

As stated previously, the first piece of code, ADS_serial.m, controls the entire Viewing Software. In detail, it creates a plot space based first on the number of inputs and the sampling rate of the ADS1299. Each of these parameters can be adapted for the individual specific application that may be created. After this plot space is created (with each individual signal separated for correct viewing, a serial object is created and opened within MATLAB to pair the Target Device with the RN-42 of the system. After this point, modifications would need to be made to accommodate more than two inputs (as it was written up). Once everything has been setup and the COM port opened, the Viewer Software sends the ‘^’ flow control character to the MCU. The next section of the code takes the incoming data, separates it into the three 24-bit values (status word and two data words) then converts the data words into decimal values using
twos2decimal.m. twos2decimal.m is a function that can be called and performs the necessary calculations on the binary data to convert it into the decimal equivalent. Following this procedure, the status bits are checked for accuracy (as this value should be known based on the setup information on the MCU) and the data is plotted in real time. The time duration of this plotting sequence can be adapted using the numSamp parameter in the earlier part of the code.

Once all the data has been captured and saved locally (as part of the real-time plotting algorithm) the ‘^’ flow control character is sent again, thus stopping all communications. The Software then disconnects from the COM port and cleans up the objects that were created.

Obviously, this code could be extended and adapted to be able to serve a particular application. For this system, it was created for the simple purpose of reading in the data values for two channels and plotting these real-time values in MATLAB for a short period of time. This amount of data acquisition time was sufficient to run the tests and retrieve the results that will be shown in the next chapter. In addition to this, more data processing could be done on the data, some of which will be described in the next chapter, and others (not implemented) that will be described and detailed in the Future Work section of the final chapter of this report.
Chapter References:

dsPIC33FJXXXGPX06/X08/X10 Data Sheet, 2009.

Chapter Four

Experimental Results

Experimental Setup

Before delving into the Experimental Results themselves, it is prudent to first understand the experimental setup for the working prototype of the design cataloged in this body of work. This portion of the chapter will walk through the specifics of the experimental setup, including an overview of the selection of EEG electrode placement and the final design considerations and implementations which gave the following results. After this, an introduction will be made to the processes in which the comparative results were created and the software methods that were used to do so.

The full design of this project included a very basic board design. As was detailed in Chapter 2 previously, the premise behind this board design was to create a platform that would be able to easily interface with any future projects and ventures. For the purpose of the design described in this thesis, one of these slots was used to interface with the RN-42 Bluetooth transceiver, and the other was used to interface with the ADS1299 Analog Front End. In this way, the Motherboard served as a full prototyping solution which could be adjusted “on the fly” at any point in time. The schematics and board layouts are shown in the Appendices of this document.
The next design consideration to discuss is the selection of the active EEG electrode placements that were chosen for the designed hardware being spoken of. To decide which positions to populate, the first consideration was to consult the International 10-20 System of Electrode Placement [1]. This system is the most popular way to describe the positions of the electrode, and the entire 10-20 placement system is shown to the right in Figure 4.1. The simple explanation of these placements is that the letters stand for the following; “F” & “FP” electrodes – frontal lobe, “T” electrodes – temporal lobe, “C” electrodes – central lobe, “P” electrodes – parietal lobe, and “O” electrodes – occipital lobe. Additionally, the “G” electrode is placed in the center of the forehead to provide a ground reference for the electrodes, while the “A” electrodes serve as reference points. Specifically for this body of work, the frontal and occipital lobes were focused on, while also highlighting the “Z” electrodes along the brain’s midline. Thus, the eight electrode placements selected for the design outlined here were the following: FP1, FP2, F3, F4, O1, O2, CZ, and FZ (with also the addition of the required ground and reference electrodes).

![Figure 4.1 – 10-20 Placement System [2]](image)
To sum up the final design of the system being described here, a picture was taken of the device, just before it was put into use. This picture is shown in Figure 4.2 to the left. As can be seen from the picture, there are multiple boards that are being utilized by the platform. On the far left is a breakout board for the ADS1299 designed and sold by Texas Instruments. In the middle resides the “Barebones” designed board which houses the dsPIC and holds the daughter cards for development and prototyping. The final (very small) board is just the RN-42 evaluation board which, like the ADS1299 board, is wired to the prototyping breadboard daughter cards. What is not pictured in Figure 4.2 is the EEG headcap which has the entire 10-20 Placement System of electrodes populated on it.

The system that was chosen to serve as the baseline of comparison for functionality of the system described above and pictured in Figure 4.2 was the g.tec g.USBamp. A single g.USBamp system features “16 DC-coupled wide-range input channels” and provides “24-bit resolution with simultaneous sampling of all channels with up to 38.4 kHz” sampling speed [2]. Additionally, multiple g.USBamp systems can be used in parallel to increase the number of channels that can be displayed and analyzed at any one time. These devices are physically tethered and have the incoming data transmitted via USB to a PC for further capture and analysis. To do this further analysis and capture, the BCI2000 software suite was chosen. The
BCI2000 software suite has been designed and fitted to support a number of data acquisition systems, including the g.tec g.USBamp [3]. This software suite was chosen because it allows for simple interactions with other software, such as MATLAB. Also included in the BCI2000 software suite is the capability to not only view the EEG signal as it is being acquired, but also the capability to store the results and view them at a later time. Thus, data analysis can be performed after the fact and compared with other systems. All of the results shown from both of the systems come from the capabilities of each system to have a PC application store the data for future viewing and analysis.

**Comparative Results**

In the proceeding figures, the comparative results of the g.tec system will be displayed. To begin, the total acquisition of the g.tec system will be shown (a total of sixteen channels at once). Following this, these results will be narrowed down to show the response from the eight channels that were chosen from the 10-20 Placement System for this research (mentioned in the earlier section in this chapter). Immediately after this, the results will be broken down further into a format that will best be comparable to the results acquired by the designed system which this thesis is based upon. In more detail, the particular electrodes which will be focused upon will be two from the frontal lobe (FP1 and FP2) and two from the occipital lobe (O1 and O2). As each set of results is displayed, a short description and explanation will go along with the individual result which is presented.
As stated previously, the first set of results is a snapshot of the entire results from the g.tec system as were acquired and stored for later viewing by the BCI2000 software suite. Figure 4.3 below is the snapshot of all nineteen populated EEG electrode channels that were included in the comparative setup. This was achieved by tethering two g.USBamp systems together to achieve more than the base sixteen channels of EEG. As can be seen, along the x-axis of this plot is the time (in minutes:seconds) for which the event was captured. Along the y-axis of this plot is the channel name, as can be related to the 10-20 Placement System. The y-axis will also serve as the amplitude measurement of each individual signal, as measured in microvolts (uV). Additionally, the scale in the bottom left corner, near the label for electrode T6 shows a comparative amplitude of 100uV. Each of the proceeding result screenshots within the comparative results section will hold to this same format.

Figure 4.3 – g.USBamp Results – Complete Setup
The above result screenshot, Figure 4.4, shows the g.USBamp’s results across the eight specific or “main” electrodes that were chosen for the research defined by this body of work. Even more specifically, in the lower of the screenshots, Figure 4.5, the results from just the FP1
and FP2 electrodes are shown over a period of close to six seconds. This section of the entire capture period was chosen for these electrodes because of the presence of eye blink events at the 2:08 mark. The appearance of these EMG events are important baseline capture points to ensure that the EEG is operational and that the electrodes have good electrical contact with the scalp. To show more detail on each of these electrodes, Figures 4.6 and 4.7 below separate each of these results into their own plot. From these Figures, the EMG events are obvious from the baseline measurements, showing in some cases a response greater than 300uV in amplitude while the rest of the signal remains well below 100uV in amplitude. These EMG capture events then confirm that the g.tec g.USB amp can successfully capture the baseline event of an eye blink.

Figures 4.6 and 4.7 – g.USBamp Results – Frontal Lobe Electrodes

*FP1 results on the left, FP2 results on the right*
Moving forward with the results, the two results from the two electrodes that were placed on the occipital lobe, O1 and O2, are shown in the resultant screenshot above, Figure 4.8. This span of nearly ten seconds of time was chosen for focus because of the distinct appearance of alpha waves in the signals. These alpha waves, like the eye blink EMG events are also important baseline capture points to ensure proper operation of an EEG system. As can be easily noticed in Figure 4.8, from 1:54 to around 2:01 there is a very apparent resonant presence of alpha waves which taper off shortly thereafter at 2:02. This confirms that the g.tec system was correctly capturing the alpha waves from these electrodes. Taking a bit more of an in-depth look at each of these electrode position results, Figures 4.9 and 4.10 separate each of these electrode positions into its own plot. These more detailed screenshots show the alpha waves strong presence, having an amplitude greater than 100\(\mu\)V, and one much greater than the baseline EEG readings shown in the tail section of Figure 4.8 (around 2:02). At this point, the Figures and results portrayed in this
Acquired Results

Similar to the previous section, the proceeding figures from this section will encapsulate those that were acquired by the design on which this thesis is based. These results were not garnered as a result of the use of the BCI2000 software suite, but, as mentioned in a previous chapter, were acquired through a custom MATLAB script. To make these results more succinct, each screenshot will appear as a single electrode’s resultant signal, and the result will be focused on the two baseline capture points that were shown in the comparative results. Additionally, the

Figures 4.9 and 4.10 – g.USBamp Results – Occipital Lobe Electrodes

O1 results on the left, O2 results on the right

chapter have set a baseline for what we should expect from the forthcoming results of the system for which this work is based upon.
resultant captures and plots shown in this section will appear quite different from the previous results from the BCI2000 software suite. For these results, the x-axis will have a time domain value, but this value is not explicitly related to the exact time, but rather the number of samples that were captured by the MATLAB program. The y-axis on these results will also be a bit more arbitrary than the results from the g.tec system, however, they relate more closely to the digital value that was assigned to the signal via the ADS1299. In this way, the y-axis can be thought of similarly to the comparative results as the amplitude of the signal in microvolts (uV).

To begin, the first set of results that will be displayed are the results that were procured from the frontal lobe. Specifically, the eye blink events which were spoken of and shown in the previous section of comparative results. Figure 4.11 to the right shows the eye blink EMG events on the FP1 electrode position.
Along with the previous result in Figure 4.11, the result in Figure 4.12 to the right shows the eye blink results from the FP2 electrode position from the same run of data. This data also explicitly shows the eye blink results as is expected per the comparative results from the g.tec system.

Moving onward, the below figures, Figures 4.13 and 4.14, show the unfiltered results obtained from the two occipital lobe electrode positions O1 and O2. Since these results are a bit difficult to interpret on first glance, the raw data was run through a basic low pass filter (set to 60Hz) and high pass filter (set to 1Hz) to achieve a result that could be better interpreted.

\[ \text{Figure 4.12 – Designed System Results – FP2} \]

\[ \text{Figures 4.13 and 4.14 – Designed System Results – Occipital Lobe Electrodes} \]

\[ O1 \text{ results on the left, O2 results on the right} \]
The resultant filtered results are shown above in Figures 4.15 and 4.16. These waveforms more accurately portray the alpha waves that were expected to be acquired from these two electrode positions.
Chapter References:

[1] “The International 10-20 System of Electrode Placement” Internet:


[3] BCI2000: System Features” Internet:
Chapter Five

Conclusion

Confirmation of Results

In order to confirm the results, and thus the functionality of the system designed by this work, two main parallels between the results must be shown between the designed system and the commercial baseline system. Specifically, it is important to show that the designed system can acquire EEG data that is consistent with a commercial EEG system (i.e. the g.tec system). To enable comparison between the two systems we will look at the ability to successfully capture both alpha waves and eye blink events. These two EEG features are among the most basic characteristics found in a set of EEG data. Upon confirming that the results from both systems not only show these artifacts, but are presented in a similar, repeatable fashion, then the results can be deemed sufficient to prove the efficacy of the system which was designed in this body of work.

The first item to address is confirmation of the appearance of alpha waves in the results. As reported in the first chapter, alpha waves are induced by relaxing the brain. This causes the magnitude of the beta waves, which are at a higher frequency, to drop and show the alpha waves more clearly. Referring explicitly to Figure 5.1, which shows the EEG signal collected by the g.tec system from the electrodes at positions O1 and O2. In this Figure, it was shown that the characteristic appearance of alpha waves were present during the point in time when the subject
was relaxed during baseline testing. In Figure 5.2, the results from this same position for the designed system also shows the appearance of alpha waves from the acquired raw EEG data. Further, the alpha waves which are present in each of these sets of results occur similarly. To expand upon this, in Figure 5.1, the cyclical nature of the magnitude of the EEG signal when alpha waves are present shows very thoroughly. Continuing on to the results in Figure 5.2, the cyclical nature of the magnitude is also present, thus showing the similarity of the recorded results. Therefore, the capture of alpha waves is verified by both the commercial and design systems.

Moving forward to the more explicit results from the frontal lobe are the occurrence of eye blink events. These are easily noticed in both sets of data, as they consist of EMG events whose magnitude greatly outweighs that of the baseline EEG signal. To compare the two systems look first at Figure 5.3, which shows the appearance of three consecutive eye blink events captured with the g.tec system. These EMG events correlated directly to the physical blinking events during the baseline testing. By comparison consider Figure 5.4, which also show the occurrence of three consecutive eye blink events captured using the designed system. Since
these events are so overwhelming in their magnitude in comparison to anything else, it is also
easy to confirm that the results shown in both systems appear in a similar fashion to one another,
thus confirming the second point for the efficacy of the system designed by this body of work.

The appearance of these two very telling and basic EEG characteristic features confirms
that the designed system on which this body of work resides is fully functional and successful in
providing the desired EEG capture capabilities. Since these results align, and the comparative
system (the g.tec system) is widely known as an accurate and successful system, it is safe to
deed the system designed in this work as also being accurate and successful in obtaining
meaningful EEG signals. These two tests and results by no means categorize this device as a
fully medically-licensed product, but for the means of proving the functionality and efficacy of
the system as a whole are sufficient.

Figures 5.3 and 5.4 – FP1 EMG Events
g.tec results on the left, designed system results on the right
Note: these figures are also noted as 4.5 and 4.11 (respectively) in Chapter 4
**Future Works**

The work described in this thesis was for a basic platform which could be more intricately expanded upon in future works. The first and foremost of these design improvements for future work would be to incorporate the design into a single custom printed circuit board. This future work would not be too much of an extension of the current application, but would improve the overall modularity and portability of the design, and allow for the design to be fully battery powered as opposed to requiring direct powering. A simple and proven application that could be implemented in this situation would be to add a Lithium-Ion Polymer battery and a charging integrated circuit to the design [1]. This, when paired with a buck-boost converter attached between the charging circuitry/battery and the other components on the board, would ensure proper powering of the entire system.

Another future work for the platform described in this work would be to make the end application solution more user friendly. In a vast majority of medical applications and laptop/tablet based applications in general, “user-friendliness” is of paramount importance. A user friendly end application will allow doctors and other medical professionals to quickly and accurately read and analyze the data with the efficacy that medical applications demand. A specific way to do this on the user interface front would be to develop an application on either a laptop or, ideally, a tablet which would display the EEG signal real-time on a time-based axis. In addition to making the data easily available and legible, a dedicated application would be able to take MATLAB out of the equation. As the design stands now, MATLAB is used in the foreground to pull in communications and display the results. For users unfamiliar with using in-line MATLAB, the design can be a bit “clunky” and difficult to use. If a graphical user interface
were implemented to take over the responsibilities of controlling the communications and mathematical calculations carried out currently by MATLAB, the design would be much more successful. One prime example of an application already in use (and in further development) is the solution that is designed by Texas Instruments (TI) for their Heart Rate Monitor (HRM) solution [2]. Figure 5.5 below depicts a typical screenshot of the application in use on a laptop. The TI HRM design also supports an iPad app which shows a very similar display.

![Figure 5.5 – TI HRM Laptop Application [2]](image)

As with all technological designs, improvements in hardware can be done to vastly increase the efficacy and viability of a solution. One instance of this for this design would be to implement a different Bluetooth transceiver solution which could transmit data at a faster rate out to the target device via the already established UART connection. The current solution described in this thesis implements the Bluetooth communications with the Roving Networks RN-42. The RN-42, in the current mode setting (using the system-on-chip (SOC) Bluetooth stack) can only operate with a baud rate of 115,200 bps (bits per second). While this data rate is sufficient for the goals set out by this design, further advancement in transmission speed could be utilized to sample the data more frequently and achieve even better results. One example of such a
Bluetooth transceiver is the Texas Instruments CC2564 Bluetooth and Dual Mode Controller [3]. The CC2564 has a UART module on board which can handle UART baud rates of up to 4Mbps. This baud rate can be matched by the UART module on board the dsPIC used in the platform design of this body of work and would thus allow for data rates that are nearly thirty-five times faster than the current solution.

In addition to the possible hardware improvements that can be made, a number of software improvements can be made to the design. A few of these were spoken to previously in this chapter as far as the user interface is concerned. However, further filtering and data pre-processing could be done to improve the quality and robustness of the signal. The first improvement that could be made algorithmically would be to filter out the high frequency noise. As was mentioned in Chapter 1, a vast majority of the useful signal lies in very low frequencies (between 1Hz and 30Hz). Since minimal hardware filtering is employed to keep the EEG signals as raw and true as possible, pre-processing and filtering the high frequency noise out of the system would be helpful for getting the raw EEG into a form which is more easily analyzed by eye. Conversely, as could be seen by the results shown in the previous chapter, the raw EEG signals were showing quite a bit of DC noise (very low frequencies). This can be confusing when trying to analyze data, especially when trying to distinguish the alpha waves in baseline testing. Similar techniques for the high frequency filtering can be employed to further increase the accuracy of the results. A number of these frequency filters were applied to the raw EEG results shown in the previous chapter as post-processing of the data.

Another further extension of this project that deals with a software improvement to the design would be to employ some sort of motion rejection algorithm to the system software. This could be employed either as pre or post-processing of the data and would also greatly increase
the signal integrity of the system when applied to a moving person. One example of such a motion rejection software solution is one that is described by Gwin et. al. [4]. In this solution, they use “channel-based template regression procedure[s]” and “time-window averaging phase-locked data” to remove artifacts typical to the actions of both walking and running. Such advances would make EEG technologies much more robust and applicable in wearable technology solutions.

**Final Thoughts**

Although there seem to be a number of improvements to the system designed in this work, this does not diminish in any way the efficacy and value of the design itself. The EEG and BCI field of study is constantly changing and more advanced technology and software algorithms are being created every day. In addition to this, there is constantly more silicon designs which are focused on individual solutions. As the embedded solutions for these specific fields of study become more commonplace, a modular system as is described in this thesis becomes even more valuable. As the hardware becomes more advanced, the modular components can be replaced with minimal re-design requirements needed. Additionally, as software solutions become more readily available, these can be easily reproduced using either an embedded software design on the dsPIC, or on the end application side of the laptop within MATLAB or a similar program. Thus, the simple platform design that is outlined and detailed in this thesis can serve as an easy starting point for many advances and further extensions and applications of EEG and BCI technologies.
Chapter References:


Appendices

Appendix A: Microcontroller Firmware

Below is a (formatted) copy of the microcontroller firmware. This firmware consists of five files; two C header files and three C source files. Each will be introduced in a generic header with its filename and a short description of its purpose.

```c
#include "p33Fxxxx.h"
extern unsigned char *SPI_read_ptr; // Contains current SPI read address

void Wait_Xms(int ms) {
    unsigned int i, j;
    for(j=0; j<22*ms; j++) {
        for(i=0; i<261; i++) {
            Nop();
        }
    }
}

void PowerUp_ADS(void) {
    short i;
    Wait_Xms(1000); // wait for 1s for ADS power up correctly
    LATC1 = 0; // Lower /RESET
    for(i=0; i<20; i++) // Keep /RESET low for 2 ADS clock cycles
```
Nop(); // 1 ADS Clock cycles = 10 dsPIC clock cycles

_LATC1 = 1; // Re-raise /RESET
for(i=0;i<180;i++) // Keep /RESET high for 18 ADS clock cycles
    Nop();
}

void write_SPI(int command) {
    int temp;

    temp = SPI1BUF; // dummy read of the SPI1BUF
    SPI1BUF = command; // write the data out through SPI1
    while (!SPI1STATbits.SPIRBF) // wait for the TX buffer to be ready
        ;
}

char read_SPI(void) {
    char temp;

    temp = SPI1BUF; // dummy read of the SPI1BUF
    SPI1BUF = 0x00;
    while (!SPI1STATbits.SPIRBF) // wait for the data to be received
        ;

    return SPI1BUF;
}

void ADS_command(int command) {
    write_SPI(command); // Send command byte
    Nop(); // Space out commands
    Nop();
    Nop();
}

*************************************************************************
* Note: The below function is set up to read ONLY 1 REGISTER. To adapt this *
* to read more than one register, consult the TI ADS1299 Datasheet          *
*************************************************************************/
char read_Register(int address) {
    int i;
    char out;

    //Send Command To Read Register
    i = 0x20 + address;
    write_SPI(i); // only read 1 register
    write_SPI(0x00); // write the data out through SPI1

    //Read Register Contents
    out = read_SPI();
    return out;
}

void write_Register(char address, char value) {
    char i;

// Send Command to Write Register
i = 0x40 + address;
write_SPI(i);
write_SPI(0x00);    // only write 1 register

// Send Value to be Written
write_SPI(value);

// ADS Register Macros
#define _ID         0x00
#define _CONFIG1    0x01
#define _CONFIG2    0x02
#define _CONFIG3    0x03
#define _LOFF      0x04
#define _CH1SET    0x05
#define _CH2SET    0x06
#define _CH3SET    0x07
#define _CH4SET    0x08
#define _CH5SET    0x09
#define _CH6SET    0x0A
#define _CH7SET    0x0B
#define _CH8SET    0x0C
#define _RLD_SENSP  0x0D
#define _RLD_SENSN  0x0E
#define _LOFF_SENSP 0x0F
#define _LOFF_SENSN 0x10
#define _LOFF_FLIP  0x11
#define _LOFF_STATP 0x12
#define _LOFF_STATN 0x13
#define _GPIO       0x14
#define _PACE       0x15
#define _RESP       0x16
#define _CONFIG4    0x17
#define _WCT1       0x18
#define _WCT2       0x19

void ADS_Config(void) {
    char temp;
    temp = read_Register(_ID);    // Read out the device ID for the ADS1298
    write_Register(_CONFIG1,0x96); // Write and read out values for
    temp = read_Register(_CONFIG1); // registers
    write_Register(_CONFIG2,0xD0);
    temp = read_Register(_CONFIG2);
    write_Register(_CH1SET,0x05);
    temp = read_Register(_CH1SET);
    write_Register(_CH2SET,0x00);
    temp = read_Register(_CH2SET);
    write_Register(_CH3SET,0x80);
    temp = read_Register(_CH3SET);
    write_Register(_CH4SET,0x80);
    temp = read_Register(_CH4SET);
    write_Register(_CH5SET,0x80);
    temp = read_Register(_CH5SET);
    write_Register(_CH6SET,0x80);
    temp = read_Register(_CH6SET);
}
write_Register(_CH7SET, 0x80);
temp = read_Register(_CH7SET);
write_Register(_CH8SET, 0x80);
temp = read_Register(_CH8SET);
}
#define ADS_Read()     ADS_command(0x12)
void ADS_ReadSingle(void) {
    int i;
    char out;
    ADS_Read();                     // Send RDATA opcode
    // 24 Status bits + 8 24-bit Channel Samples = 27 reads
    for(i=0; i<27; i++) {
        if(i<=2 || i>=9) // Ignore status and other inputs
            out = read_SPI();
        else             // Save good values to DMA RAM
            *SPI_read_ptr++ = read_SPI();
    }
}
/***************************************************************************/
/******************************************************************************/
/* Filename: ADS.h
* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Author            Date      Comments on this revision
* ~~~~~~~~~~~~~~~~~~~~~~
* wittts  07/31/13  Initial Code Work
* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* PROGRAM DESCRIPTION:
* This file contains all the ADS centered functions needed for the
* application
* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
*/
#ifndef ADS_H
#define ADS_H
#ifdef __cplusplus
extern "C" {
#endif
void Wait_Xms(int ms);
void PowerUp_ADS(void);
void write_SPI(int command);
char read_SPI(void);
void ADS_command(int command);
char read_Register(int address);
void write_Register(char address, char value);
void ADS_Config(void);
void ADS_ReadSingle(void);
#ifdef __cplusplus
}
#endif
#endif /* ADS_H */
/* ******************************************************************************
 * Filename: init.c
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * Author            Date      Comments on this revision
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * wittts    07/31/13  Initial code work
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * PROGRAM DESCRIPTION:
 * This file contains all of the initialization routines and functions
 * for the main application
 *******************************************************************************/
#include "p33Fx.c"

#define Fcy    40000000
#define BAUD   115200
#define BRGVAL ((Fcy/BAUD)/4)     // Works out to 86.8, do not subtract one
extern unsigned char TX_Buff[60] __attribute__((space(dma)));
                // Buffer for DMA TX (10 Samples)

void Init(void) {
    RCONbits.SWDTEN = 0;          // Disable Watch Dog Timer

    // Configure Oscillator to operate the device at 40Mhz
    // Fosc= Fin*M/(N1*N2), Fcy=Fosc/2
    // Fosc= 8M*40/(2*2)=80Mhz for 8M input clock
    PLLFB = 38;                    // M=40
    CLKDIVbits.PLLPOST = 0;        // N1=2
    CLKDIVbits.PLLPRE = 0;         // N2=2

    // Clock switching to incorporate PLL
    __builtin_write_OSCCONH(0x03);   // Initiate Clock Switch to Primary
    __builtin_write_OSCCONL(0x01);   // Oscillator with PLL (NOSC=0x03)

    while (OSCCONbits.COSC != 0x03) // Wait for Clock switch to occur
        while(OSCCONbits.LOCK != 1) {};
    // Wait for PLL to lock

    TRISA = 0x0;                    // set PORTA to all outputs

    // Only pins needed for UART1:
    // UIRX = RF2, U1TX = RF3, U1CTS = RD14, U1RTS = RD15
    TRISF = 0x0004;                 // All outputs except for RF2 (UIRX)
    TRISD = 0x0000;                 // All outputs
    _LATD14 = 0;                    // U1CTS and U1RTS should be held low
    _LATD15 = 0;                    // (See RN-42 User's Guide)

    // SPI1 Port Pin Configuration:
    // SD1 = RF7; SD0 = RF8; SCK1 = RF6; SS1 = RB2
    // TRISF = 0x0080;                // set PORTF to output except for SDI1
    _TRISB2 = 0;                    // Set Slave select line as an output
    _LATB2 = 1;                     // raise the slave select line
    _TRISC1 = 0;                    // Set GPIO as output for ADS /RESET
    _LATC1 = 1;                     // Raise /RESET
    _TRISC3 = 1;                    // Set GPIO as input for ADS /DRDY

    // Extra GPIO's
}
void Init_SPI(void) {
    //Setup SPI1
    SPI1STAT = 0x0;  // disable SPI1
    SPI1CON1 = 0x0023;  // input clock = Fcy/8 = 5MHz
    SPI1CON1bits.CKE = 0x00;  // Output data changes on rising edge of SCK
    SPI1CON1bits.CKP = 0x00;  // Clock polarity set so low is idle
    SPI1STAT = 0x8000;  // enable SPI1
    _LATB2 = 0;  // lower the slave select line
}

void InitDMA0(void) {
    DMA0REQ = 0x000C;  // Select UART1 Transmitter
    DMA0PAD = (volatile unsigned int) &U1TXREG;

    DMA0CONbits.AMODE = 0x00;  // Register indirect, post-increment
    DMA0CONbits.MODE = 0x00;  // Continuous, ping-pong disabled
    DMA0CONbits.DIR = 0x01;  // read from RAM, write to peripheral
    DMA0CONbits.SIZE = 0x01;  // send a byte at a time

    DMA0CNT = 59;  // 60 DMA requests
    DMA0STA = 0x7800;

    DMA0CONbits.CHEN = 1;  // Enable DMA0 Channel

    // Enable DMA Interrupts
    IFS0bits.DMA0IF = 0;  // Clear DMA Interrupt Flag
    IEC0bits.DMA0IE = 1;  // Enable DMA interrupt
}

void InitUART1() {
    U1MODE = 0x0008;  // No hardware control, BRGH set high
    U1BRG = BRGVAL;  // 40Mhz osc, 115200 Baud
    U1STA = 0x00;

    IFS0bits.U1TXIF = 0;  // Clear the Transmit Interrupt Flag
    IEC0bits.U1TXIE = 1;  // Enable Transmit Interrupts
    IFS0bits.U1RXIF = 0;  // Clear the Receive Interrupt Flag
    IEC0bits.U1RXIE = 1;  // Enable Receive Interrupts

    U1MODEbits.UARTEN = 1;  // Turn on UART1
    U1STAbits.UTXEN = 1;
}

/***************************************************************************/
/**
 * Filename: init.h
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * Author      Date      Comments on this revision
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * wittts      07/31/13  Initial code work
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * PROGRAM DESCRIPTION:
 * This file contains all of the initialization routines and functions
 * for the main application
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~/

#ifndef INIT_H
#define INIT_H

#ifdef __cplusplus
extern "C" {
#endif

void Init(void);
void Init_SPI(void);
void InitDMA0(void);
void InitUART1();

#ifdef __cplusplus
}
#endif

#endif /* INIT_H */
FILE: main.c

AUTHOR: wittts

DATE: 02/22/13

COMMENTS: Copy of Original ADS Setup Code

DATE: 07/10/13

COMMENTS: Checkout on custom PCBs

DATE: 07/31/13

COMMENTS: Implementation of pointer and reading into DMA RAM

PROGRAM DESCRIPTION:
This program is intended to not only setup, but then pull data from the ADS1298 AFE to the dsPIC33FJ256GP710. The code and setup for this code example will comprise of the entire front end of an EEG end application.

The first thing that this code does is set up the ADS by writing all of the important registers, thus configuring the ADS for proper operation. After the registers have been written, conversions are resumed and single data reads are performed.

The read data is stored in DMA RAM. When block of DMA RAM is filled, it will then be overwritten by new data. For the purpose of this example, the status bits will be ignored, as will the last six inputs (which are disabled in this example).

ADDITIONAL NOTES:
The Processor starts with the External Crystal with PLL enabled and then the Clock is switched to PLL Mode.

CLOCK FREQUENCIES:
Fcy = 40 MIPS; SCK = Fcy/8 = 5 MHz

#include "p33Fxxxx.h"
#include "init.h"
#include "ADS.h"

#include "ADS.h"

_FOSCSEL(FNOSC_PRIPLL & IESO_ON); // Primary clock is 8MHz Crystal with PLL
_FOSC(FCKSM_CSECMD & OSCIOFNC_OFF & POSCMD_XT); // Clock Switching is enabled and Fail Safe Clock Monitor is disabled
 // Primary Oscillator Mode: XT Crystal (8MHz)
_FWDT(FWDTEN_OFF); // Watchdog Timer Enabled/disabled by user software
_FICD(JTAGEN_OFF & ICS_PGD1); // Disable JTAG pins & Setup program pins

unsigned char TX_Buff[60] __attribute__((space(dma))); // Buffer for DMA TX (10 Samples)

unsigned char *SPI_read_ptr; // Contains current SPI read address
int main(void) {
    Init();                // Init clock and port pins
    PowerUp_ADS();         // Power up sequence for ADS
    Init_SPI();            // Init the SPI peripheral
    ADS_StopCont();        // Stop continuous sampling
    ADS_Stop();            // Stop the ADC's conversions
    ADS_Config();          // Write and read back all important ADS registers
    ADS_Start();           // Restart conversions
    SPI_read_ptr = 0x7800;  // Initialize pointer at start of DMA RAM
    InitDMA0();            // Initialize DMA0 for U1TX from RAM
    InitUART1();           // Initialize UART1 for 115200,8,N,1 TX/RX

    char i;
    while(1) {
        for (i=0;i<10;i++) {       // Save 100 Samples
            while(!_RC3);           // Wait for DRDY to go low
            ADS_ReadSingle();      // Read a single sample
        }
        while(!_RG0);             // Hold and wait for DMA interrupt
        SPI_read_ptr = 0x7800;     // Reset pointer to start of DMA RAM
        _LATG0 = 0;
    }
}

// Hardware Interrupts
void __attribute__((interrupt, no_auto_psv)) _U1RXInterrupt(void) {
    LATA = U1RXREG;
    IFS0bits.U1RXIF = 0;
}
void __attribute__((interrupt, no_auto_psv)) _U1TXInterrupt(void) {
    IFS0bits.U1TXIF = 0;
}
void __attribute__((interrupt, no_auto_psv)) _DMA0Interrupt(void) {
    _LATG0 = 1;
    IFS0bits.DMA0IF = 0;
}

/***************************************************************************/
Appendix B: Target Device Viewer Software

Below is a (formatted) copy of the target device viewer software. This software utilizes functionalities built into MATLAB to connect to the COM port for the system (as chosen once synced to the Target Device) and stream the data onto the Target Device. The Target Device will then plot this data in real time. Two pieces of code were used for this software, a MATLAB function file, and a MATLAB code file (which uses the function file). Each file will start off with an introduction of what it is.

```matlab
% ADS_serial.m            this code will connect to the assigned COM port,
%                          stream data in through said COM port, and plot
%                          the data in real time

% for plotting the data
data2 = cell(1000,9);
i=0;
j=0;
drawFlag=0;

numTrace   = 2;    % < traces
sampPerWin = 500;  % < data/trace
numSamp    = 1000; % < total number of samples

% for input short signal
%minYlim = -1350;
%maxYLim = -1000;

% for test signal
%minYlim = -5000;
%maxYLim = 3000;

% for normal electrode
minYlim = -9000000;
maxYLim =  9000000;

% prepare the plot
axes('xlim',[1,sampPerWin],'ylim',[minYlim,maxYLim]);
x=1:sampPerWin;
y=-inf*ones(size(x));
y2=-inf*ones(size(x));
lh=line(x,y,'Color', 'g');
lh2=line(x,y2,'Color', 'r');
lb=line([inf,inf],[minYlim,maxYLim],'Color', 'b');
shg;
```
% Create a serial port object.
obj1 = instrfind('Type', 'serial', 'Port', 'COM5', 'Tag', '');

% Create the serial port object if it does not exist
% otherwise use the object that was found.
if isempty(obj1)
    obj1 = serial('COM5', 'BAUD', 125000, 'FlowControl', 'software', ...
                   'Timeout', 10); % WORKING
else
    fclose(obj1);
    obj1 = obj1(1);
end

get(obj1)

% Connect to instrument object, obj1.
 fopen(obj1);

fprintf('Waiting for device to respond...
')
data1 = fread(obj1, 17);
fprintf('Waking up: %c%c%c
', data1(1), data1(2), data1(3))
fprintf('Device ID: %X
', data1(4))
fprintf('CONFIG1 : %X
', data1(5))
fprintf('CONFIG2 : %X
', data1(6))
fprintf('CONFIG3 : %X
', data1(7))
fprintf('CH1SET  : %X
', data1(8))
fprintf('CH2SET  : %X
', data1(9))
fprintf('CH3SET  : %X
', data1(10))
fprintf('CH4SET  : %X
', data1(11))
fprintf('CH5SET  : %X
', data1(12))
fprintf('CH6SET  : %X
', data1(13))
fprintf('CH7SET  : %X
', data1(14))
fprintf('CH8SET  : %X
', data1(15))
fprintf('MISC1   : %X
', data1(16))

tic
% gather the data and plot in <real-time>...
for i=1:numSamp
    [data1 numReads msg ] = fread(obj1, (3+(numTrace*3)));
    if (numReads~= (3+(numTrace*3)))
        fprintf('Error: msg %s
', msg)
    else
        count = 0;
        for j=1:3:(3+(numTrace*3))
            count = count+1;
            temp1 = dec2bin(data1(j) ,8);
            temp2 = dec2bin(data1(j+1),8);
            temp3 = dec2bin(data1(j+2),8);
            data2(i,count)= {strcat(temp1, temp2, temp3)};
        end
        temp = bin2dec(data2(i,2)); % <- new data
        if temp > 8388607
            temp = twos2decimal(temp,24);
        end
        temp4 = bin2dec(data2(i,3)); % <- new data

end
if temp4 > 8388607
    temp4 = twos2decimal(temp4,24);
end
if (~strncmp(data2(i,1),'110000000000000000000000',24))
    fprintf('Error: Status incorrect\n')
else
    ix = rem(i-1,sampPerWin)+1;
    y(ix) = temp;
    y2(ix) = temp4; % second trace
    if ~ishandle(lh)
        break % break in case you close the figure
    end
    set(lh,...
        'xdata',x,...
        'ydata',y);
    set(lh2,...
        'xdata',x,...
        'ydata',y2);
    set(lb,...
        'xdata',[ix,ix]);
    drawFlag=drawFlag+1;
    if(drawFlag==7)
        drawFlag=0;
        drawnow;
    end
    end
end
toc
disconnect from instrument object, obj1.
fclose(obj1);

% Clean up all objects.
delete(obj1);

function [decimal] = twos2decimal(x,bits)
% twos2decimal(data,bits) convert 2s complement to decimal
% data - single value or array to convert
% bits - how many bits wide is the data (i.e. 8 or 16)
decimal=zeros(length(x),1);
for i=1:length(x),
    if bitget(x(i),bits) == 1,
        decimal(i) = (bitxor(x(i),2^bits-1)+1)*-1;
    else
        decimal(i) = x(i);
    end
end
Appendix C: Additional Results

A few additional resultant plots were taken from the designed system outside of the results that were shown previously. These results are a bit superfluous to those in the previous chapter and were thus saved for this Appendix.

Figure C.1 – FP1 Electrode Filtered – Eye Blink Events
Figure C.2 – FP2 Electrode Filtered – Eye Blink Events

Figure C.3 – FP1 Electrode – Eye Blink Events
Figure C.4 – FP2 Electrode– Eye Blink Events
Appendix D: Board Schematics and Layout

As was discussed previously, a custom board was created and used in this design. The proceeding Appendix will contain the schematics and layout views of this board.