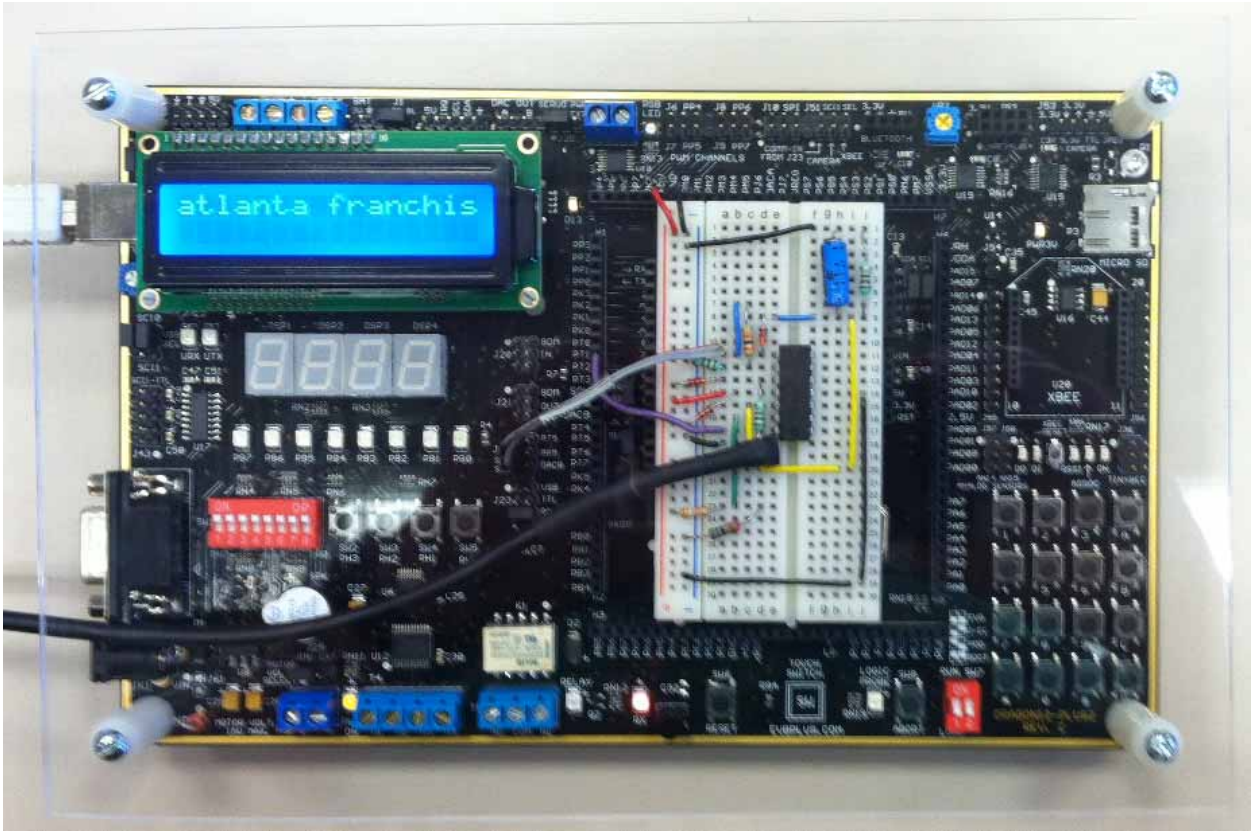
Morse Code Decoder



Created by: Shawn Lawson

Sinclair Community College

Electronics and Robotics

Electronics Project Capstone, EET-2278,

Spring 2014

Professor Russo

Contents

Acknowledgments

My wife Meggan – I could not have completed this project without her love, support, and understanding. There were many days where I spent more time at school than at home, but all that hard work has finally paid off.

Work – If I had not been given the freedom to take a vacation day every Friday so I could attend class during the day this capstone project would not have been possible. Special thanks to my boss Mr. Glenn Greet who was extremely supportive and flexible during the three years I went to school.

Kenzie Grogean – Give this man a raise! His help was invaluable, and his patience was amazing. Thank you so much for all of the help, I couldn't have done it without you.

Professor Russo – Thank you for all of the technical advice, and for making the entire capstone experience fun and enjoyable.

Professor Reeder – I appreciate your assistance with all of my assembly coding questions. I truly learned so much from you in the many courses you were my professor.

Fellow EET students – Thank you for all of your assistance, advice, and support.

The entire EET staff – The past three years have been a wonderful learning experience. I would highly recommend the Sinclair EET program and all of its educators to any student interested in the electronics field.

Overview

My capstone project is a Morse code decoder. Morse code is an on-off keyed carrier, and when viewed in an oscilloscope an off state looks like 0V DC, and an on state is a sine wave transmitted at a frequency specified by the user or transmission equipment. An analog Morse code signal will be played through an app on a smartphone, and then processed through some electronic components to condition the waveform from a sine wave to individual pulses. The pulses will be the same width as the transmitted dit or dah, and then the resultant pulses and spaces will be measured to determine whether it makes up a part of character, a full character, or a space.

"International Morse Code
Recommendation ITU-RM.1677-1"
Referenced from Wikipedia

Background

My initial idea when asked to decide upon a capstone project was to do something related to RFID since I view the topic as interesting and the wave of the future within various industries such as credit card companies. I decided against building an RFID project when I considered the fact that I have never had any formalized training in the subject, and I learn better with instruction compared to being self-taught.

I finally decided to do a project related to Morse code since it was my very first job in the United States Air Force, and ever since learning Morse code at Fort Devins, Massachusetts in early 1993 I have had an interest in the subject. To me Morse code is very similar to knowing a foreign language and I hope to one day become a ham radio operator so I can practice my code copying abilities.

Upon completion of the Basic and Advanced Morse Code Operators Course all students at the time were given a cassette tape to help maintain proficiency. The original thought for the project was to play this analog cassette, digitize the waveform by some means, and interpret the corresponding characters. I could not locate my 20+ year old cassette tape, but I was able to find a $5 app on iTunes which played the code way better than any old tape ever could.

After viewing the Morse code signal in an oscilloscope it was noted that there were clear and distinct characteristics of each dit and dah used to transmit information. As along as the speed remained constant then each dit, dah, and spaces were set lengths. Since Morse code is an on-off keyed transmission system I noted that when data was being transmitted a sine wave was active at a specified frequency. I conducted research on how to turn the dits and dahs into individual pulses from which a single rising and falling edge for each pulse could be measured.

The hardest decision in the entire process was whether to use Labview or the Dragon board 12 – Plus 2 for the decoding aspects of the project. I finally decided to purchase the Dragon board 12 – Plus 2 since I had just finished EET-2261 the previous semester, and both assembly language programming and the circuit board itself were still fresh in my mind.

Principles of Design

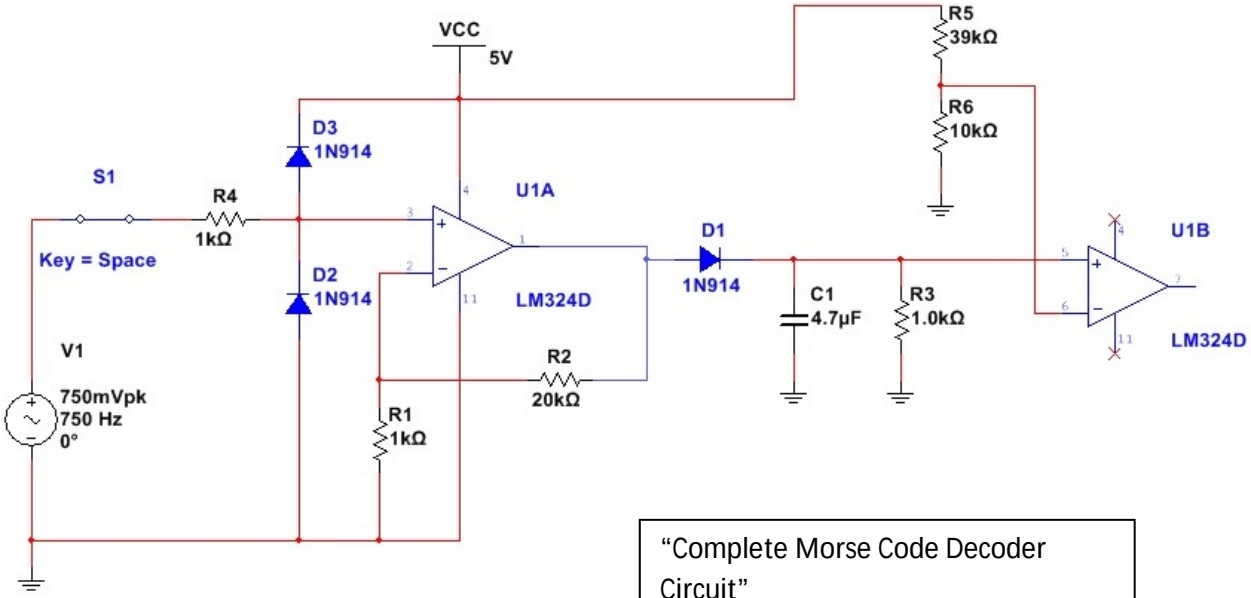
I consider the project to have three distinct sections: externals, circuitry and coding. Externals include the laptop, iPhone, and cable input to the circuit board. Circuitry is all of the components that are located on the Dragon 12 – Plus 2 breadboard. Coding is the program that was written in CodeWarrior to decode the Morse code characters utilizing the Dragon 12 – Plus 2 circuit board.

As stated in the background section, I originally planned on playing an analog cassette tape containing Morse code. Fortunately, I was unable to locate the cassette tape. The reason I say this is fortunate is that it enabled me to conduct some research and find a great smartphone app called AA9PW Ham Morse. The app cost $4.99, and plays on my iPhone or iPod touch. Since the Morse code is an audio signal I decided to use the headphone jack on the iPhone to output the Morse code. To accomplish this I needed to purchase a mono audio cable with one end stripped so I could access the positive and negative wires. Each wire was soldered individually to a single pin on a 2-pin connector designed to fit into breadboard holes.

For the circuitry component of the project the following references were used: the Principles of Electric Circuits book (EET-1150/EET-1155), the Electronic Devices book (EET-2201), and the multisim simulation software. It was noted that the voltage level output from the headphone jack was around 700mv which would not be enough voltage to drive the designed circuit so a method to boost the input signal was required, and I decided to use an op-amp. I only wanted the positive levels of voltage so some form of rectification would be needed. After the signal was rectified I needed something to turn the waveform into smooth pulses, and I decided to use a comparator.

CodeWarrior was the interface we learned to create, debug, and test any code that we wrote in the EET-2261 Microprocessor course. Since I had a familiarity with the software and the Dragon 12 – Plus 2 development board I decided to use them in my project rather than a Raspberry Pi or Arduino board. CodeWarrior can use either assembly or C languages, but I chose to work in assembly since that's what we learned in EET-2261 as well as the fact that I had no experience in writing code in C. Overall I found the CodeWarrior software to be very user friendly and easy to use.

Design Details

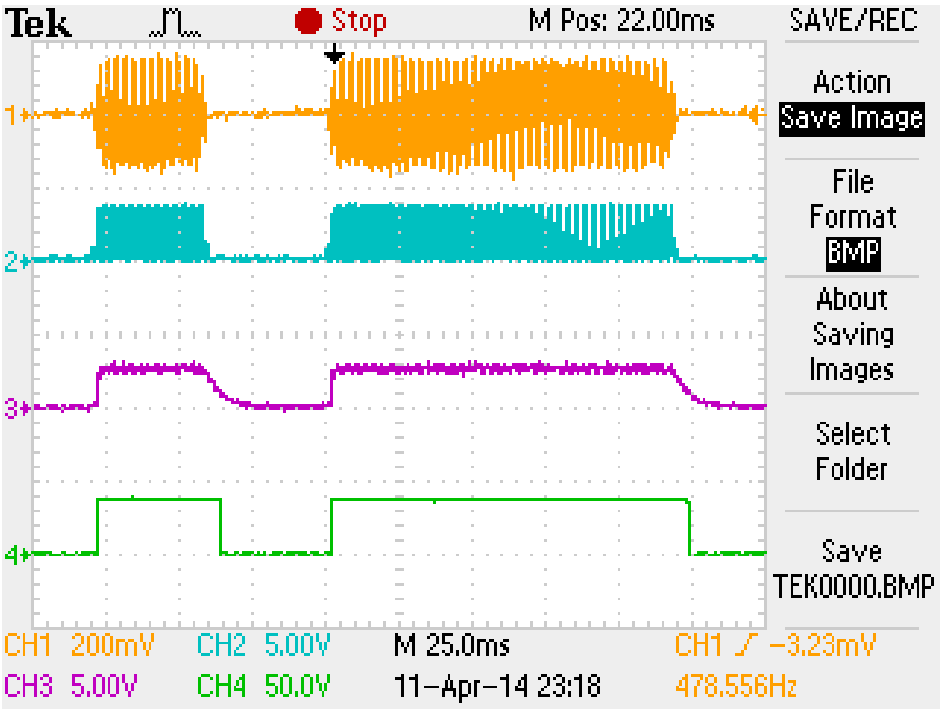


"Complete Morse Code Decoder Circuit"

Pictured above is a multisim drawing of the finalized circuit. A 750mVpk @ 750Hz sine wave used to simulate the Morse code, and the switch was used to simulate the inherent on-off keyed conditions of the Morse code.

Diodes D2 and D3 were placed in the circuit to ensure no power surges or other unexpected noise would interfere or harm the circuitry and the Dragon board.

With only approximately 0.7V being output from the iPhone headphone jack there needed to be a way to amplify the signal. I decided upon an LM324D op-amp, which is a 4 channel op-amp. Using a negative feedback loop from the output pin 1 of the op-amp and R2 and R1 I was able to increase the gain by 21, although due to the +5V VCC the max gain was limited to about 4.3V. One unexpected yet completely welcome surprise was that the LM324 op-amp acts as a rectifier, eliminating the negative voltage levels. I was going to build a rectifier into the circuit but did not have to thanks to this built in feature of the op-amp. D1, C1, and R3 make up the AM detect portion of the circuit. What the AM detect circuit does is conditions the sine wave to give it a more DC appearance. There is still a ripple caused by the capacitor discharging at this point so I added a comparator to completely smooth out the resultant wave into nice distinct pulses. For the comparator I used the +5V VCC, R5, R6, along with the second channel of the LM-324 op-amp. The final output is the pulsed signal.



"A dit and a dah at various points in the circuit"

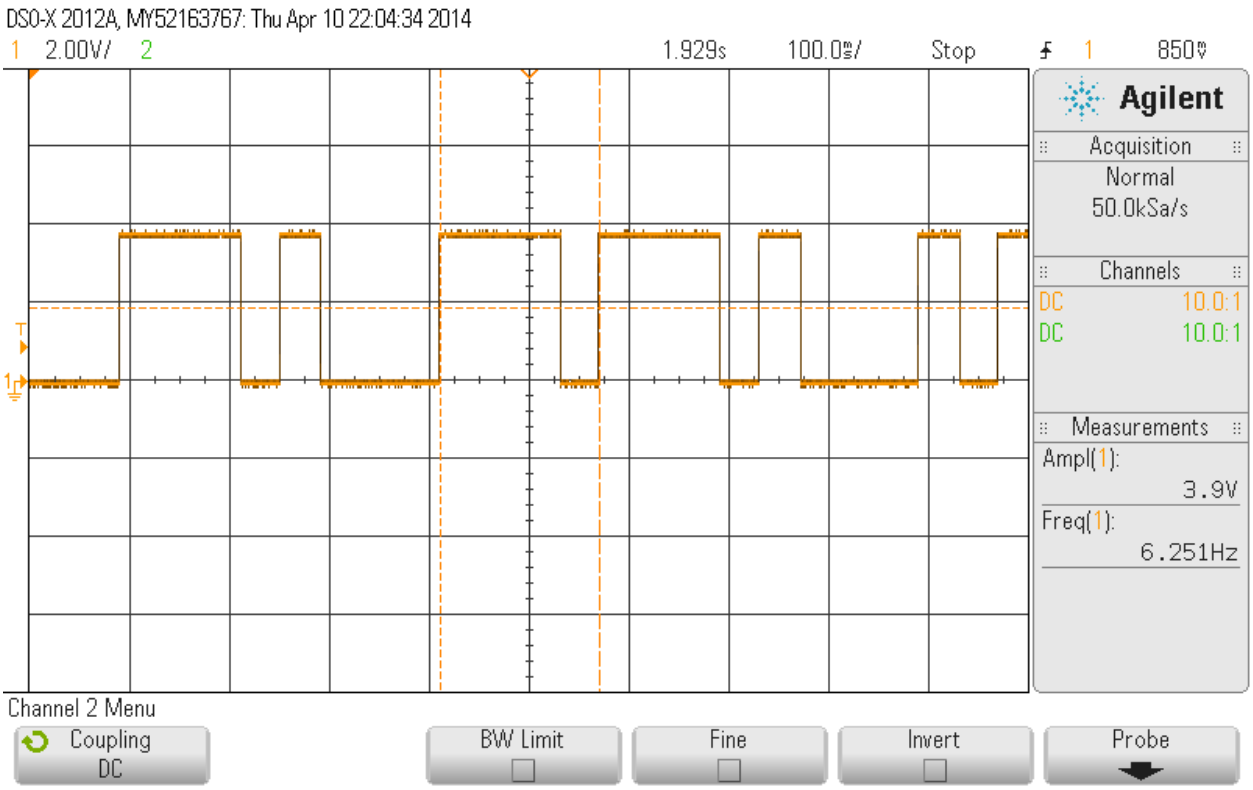
The picture directly above was taken from Tektronix TDS2014B four channel oscilloscope, and clearly depicts the various stages of the waveform as it is processed through the circuit on the Dragon development board. This smaller pulse shown is a single dit being transmitted, and the longer pulse is a single dah being transmitted.

Channel 1 of the oscilloscope shows the Morse code as it is originally input into the circuit from the iPhone. If the time increment was decreased the sine wave when in an on state would become visibly apparent.

Channel 2 of the oscilloscope shows the signal after it has been processed through the op-amp. The rectified signal is visible and it should be noted that there is no longer any negative voltage levels.

Channel 3 of the oscilloscope is the waveform after it goes through the AM detect portion of the circuit. The discharging capacitor is still highly visible at the end of each pulse, but the signal is starting to look more like the desired output.

Channel 4 of the oscilloscope is the final output pulsed waveform after the output from the AM detect was ran through a comparator. This output will be input to PT1 on the Dragon 12 – Plus 2 to start determining what characters are being transmitted.



Picture showing dits, dahs, space between a dit/dah, and a space between a character. A space between words is not shown.

Pin PT1 is now receiving pulses so run the program to decode the Morse code through CodeWarrior. See the Operation section of the written report for step by step instructions to run a program in CodeWarrior.

The first part of the program measures rising edge to falling edge in order to determine if a dit or dah was generated. A dah is 3 times longer than a dit. In the picture above the first pulse is a dah, and it is followed by a dit.

Measuring falling edge to rising edge determines if it is a space between dits/dahs, a space between characters, or a space between words. In the picture above there is an example of a space between dits/dahs, i.e. the space between the first dah and the following dit. A space between characters is longer than the space between a dit/dah, and it is shown between the dah – dit character and the dah dah dit character. A space between words is the longest break, but an example is not shown above. The dit/dah is stored in memory and shifted left each time a new dit/dah is sent until the character is finished, at which point the final stored character bit value is referenced in a look up table and displayed on to the LCD screen as an ASCII character.

Construction

The construction of the project was not very labor intensive once the circuit was designed and confirmed to work correctly. The only electrical components required were 6 resistors, 3 diodes, 1 capacitor, and one op-amp, I decided to use the breadboard already attached to the Dragon 12 – Plus 2 board rather than building a circuit-board in dip-trace. All the other desired components such as positive voltage to bias the op-amp, ground, and a speaker were conveniently already built into the development board. I did make sure to take extra care in cutting the wires lengths and component leads as short as possible to ensure a neat and tidy appearance.

Since I am utilizing the LCD screen attached to the Dragon12 board I needed to ensure that the audience would be able to read the text scrolling across the screen. I considered building a box out of various materials including wood, metal, and acrylic, but ultimately I decided to simply use an acrylic shield that was connected by screws at the four corners of the circuit-board. The screen rises above the board by the length of the screws, but they are hidden by plastic spacers to make it look more professional. Underneath the board the screws are attached to bolts with enough space left to set the remaining screw length on rubber stoppers to support the board above the surface the project is resting on.

For the purpose of this project the Dragon-board needs to be connected via USB to a laptop computer to power the circuit and to run the Code-warrior software. There was an additional 9V power source that could be plugged into an electrical outlet included in the purchase of the development board. If I were to have utilized this 9V power source I would have needed to save my program in Flash memory rather than RAM. I chose the USB/CodeWarrior option because I felt like it gave me greater flexibility to troubleshoot and diagnose problems.

Testing

This section was by far the most frustrating aspect of building and designing my own project. There were issues with multisim simulation, input pins on the Dragon 12 – Plus 2 board, and coding the LCD to only display text on a single line.

Multisim was a useful tool for giving a clean visual representation of the designed circuit, but the program had an issue properly simulating the circuit. Using the probe feature in multisim I was getting voltage readings that I shouldn't have been getting. Specifically, since the VCC input voltage was +5V I should not have been able to get a voltage out of the op-amp much higher than 4.3V due to the limitations of the chip. Multisim was simulating the full gain of 21 which caused issues with all the other voltage levels within the circuit. The circuit did work very well on the breadboard however, with the correct voltage levels at each test point.

In EET-2261 we used a program which counted the number of rising edges from input pulses, and I used this program as an initial code test to ensure the Dragon board would be able a identify the pulse edges, and count them correctly. The program from class worked great as a test but I decided not to use it in my final code since I had another way to accomplish what I wanted to do with the pulses. The input on the board used for this program was Port T pin 0.

Now that I knew the Dragon board could properly identify the pulse edges I wrote a section of code that would measure the pulse width to determine if a dit or dah had been transmitted. A single dit in Morse code is an E, and a single dah is a T, so I wrote a test program I called "E" & "T" test. When a dit was transmitted an E would be displayed on the LCD and when a dah was transmitted a T would be displayed. I could not get this program to identify the dits and dahs correctly, and both Kenzie and Professor Reeder look at the code, to which they could find no problems. I had been using Port T pin 0 and as a last resort I decided to use Port T pin 1 since I remembered a pin problem from class. Port T pin 1 worked as it should have. I don't know why pin 0 didn't work properly, especially considering it worked fine for the pulse accumulator test I first used to test the board.

The last major issue I encountered was displaying the scrolling text on only one line of the LCD. Using the Microprocessor book as a reference I tried numerous variations of logical shifts and display shifts but none would work as desired. Kenzie really helped me out in this section and suggested I try applying an offset and counter so when it went past the 20^{th} character it wouldn't automatically jump down to the second screen, but would jump to the beginning of the display instead. Offset the 16 bits from the beginning and the character would display in the upper right position just as the previous 24 characters did, and then continue to scroll.

Operation

To operate the Morse code decoder start by generating Morse code through the AA9PW Ham Morse app on any smartphone. For the purposes of this project I am using an iPhone. The app has the ability to transmit letters, numbers, words, and news feeds. There are other transmission options such as punctuation only, but in this project I will solely be using one of the news feeds so the audience can view readable text as they hear the Morse code coming out of the speaker on the Dragon board. A modified audio cable is the output from the iPhone to the breadboard on the Dragon board.

The positive and negative leads of the audio cable are input to the proper holes on the breadboard and the attached circuitry on the breadboard conditions the waveform to pulses so that CodeWarrior can run its program.

CodeWarrior a free download from www.freescale.com, and can be loaded on any computer. The software is loaded on the computers at Sinclair Community College, but for portability and display reasons I decided to load the software on to my personal laptop to run the project. Within the folder where the program is written double click on the CodeWarrior Project file (.mcp) to launch CodeWarrior. Once the software launches double click on the main.asm file to edit/run/debug the program. Locate the Debug tab and click it once to compile the program. A new window will launch. Press the Run tab to start the program.

The end user should listen for Morse code coming out of the speaker and look for text scrolling across the LCD screen. If there is no sound or text then there is a problem and troubleshooting should be accomplished. The most common problem I've found is the headphone volume is not turned up high enough. The volume control on the phone acts as a sort of gain, and the volume needs to be maxed out so that the program will have a high enough input voltage to drive the circuit, therefore driving the speaker and causing the text to scroll across the LCD screen. The Run tab may need to be pushed once more because I noticed that the CodeWarrior program will shut itself off if there is not an input.

Conclusions

I don’t think I could have hoped for much better of an outcome than I was able to achieve. Since I had never done a project on the same scale as my desired result there were initial concerns whether or not what I wanted to build was even possible. Morse code being a personal interest of mine certainly helped me maintain focus and stay on task. There were frustrating sections, especially when there was no logical reason why something was not working like the PT0 input pin issue referenced in the testing section.

The greatest positive outcome to me was the sense of accomplishment that went with undertaking such a larger project through the entire design and building process. I asked a lot of questions of Kenzie, but he was very skilled at steering me in the right direction without straight giving me the answers. The fact that I had to research the subjects myself rather than just being given the answer made it feel that much more worthwhile at the end when everything fell into place and worked as envisioned.

Getting the circuit to transform the initial Morse code signal when in an on state and turning that into a smooth pulse turned out better than predicted. The pulses are so smooth they look like they are machine generated. The comparator part of the circuit worked really well.

The greatest disappointment was that I ran out of time in the semester to be able to decode multiple speeds of Morse code. I only specified the program being able to do one speed in my initial proposal so I met the requirement I outlined for myself, but it would have been nice to take the project one step further.

Recommendations

If I had to do it all over again I know for certain that I would not have picked a different project. I found my project to be personally interesting, rewarding, and challenging. If another student were considering building the same project that I chose I would suggest having some background in Morse code. Being able to listen to the code and know what a certain character was supposed to be with regards to dits and dahs definitely helped me troubleshoot and ensure the proper output.

For someone who wants to build the best capstone project of their own design I would highly recommend picking a subject that they have an interest in. If a project had been picked for me I might have found the subject boring or uninteresting, but since I was given the freedom to choose my own project I had a vested interest in its success.

Make sure to use available time wisely. There were points where I had concerns that the project would not work, but because I had allotted myself the proper time I was able to work through problems as they arose. There WILL be problems. Not I or any of my fellow students that I talked to were able to make it through the entire semester without running into at least one difficulty or problem. If I had procrastinated I don't know if I would have been able to complete everything that needed to be done. Even the final written report and slides took considerable time to complete, so no part is insignificant.

Use all available resources. I personally utilized the open lab as often as I possibly could. This served a dual purpose for me: I was at school so it was easier for me to talk myself into working on what needed to be done compared to being at home where it is easy to get wrapped up in life, and also because Kenzie is such a valuable resource. From what I've heard from students who attended larger schools it was almost impossible to get the individualized attention that you can get from the Sinclair EET department. All the professors were excellent resources as well: Professor Russo was full of great tips to push me in the right direction when brainstorming, and Professor Reeder was willing to sit down and diagnose coding problems.

Bibliography

Mazidi, Muhammad Ali, and Causey, Danny. *HCS 12 Mircocontroller and Embedded Systems Using Assembly and C with CodeWarrior.* Upper Saddle River: Pearson Prentice Hall, 2009. Print.

Floyd, Thomas L. *Electronic Devices – Conventional Current Version.* Upper Saddle River: Pearson Prentice Hall, 2008. Print.

Floyd, Thomas L. *Principles of Electric Circuits – Conventional Current Version.* Upper Saddle River: Pearson Prentice Hall, 2010. Print.

Addenda

Addendum #1 Project Proposal

Addendum #2 Journal Summary

Addendum #3 Complete Parts List

Addendum #4 LM324 Datasheet (page 1)

Addendum #5 Copy of Assembly program code

Addendum #6 Presentation Slides

Addendum #7 Project Report Guidance and Requirements