|  |
| --- |
| Final Project I |
| Terminal Node Controller Project |
| Project Proposal Report |

|  |
| --- |
| Thomas Cort  9/19/2012 |

# Executive Summary

This is the summary.

# Table of Contents

Here

# Introduction

I propose developing a Terminal Node Controller (TNC). A TNC is a radio modem commonly used in the amateur radio service. Its purpose is to modulate and demodulate signals in order to connect an amateur radio operator to packet radio networks. Packet radio networks are useful for keyboard to keyboard communications (i.e. chat, instant messaging, etc.), reading and posting messages (i.e. e-mail, bulletin boards, etc.), emergency communications when normal communications mediums are down, beaconing and receiving position and weather reports, and much more. Just within the national capital region there are several weather stations, bulletin board systems, and many other fixed and mobile packet radio nodes. The international space station is even equipped with its own terminal node controller. There are several shortcomings which exist in current terminal node controllers. Improving on those designs is my motivation for this project. The goal of this project is to build a modern reliable low-cost easy to use terminal node controller with an open design in a small form factor that can be used with a computer and also portable without a computer.

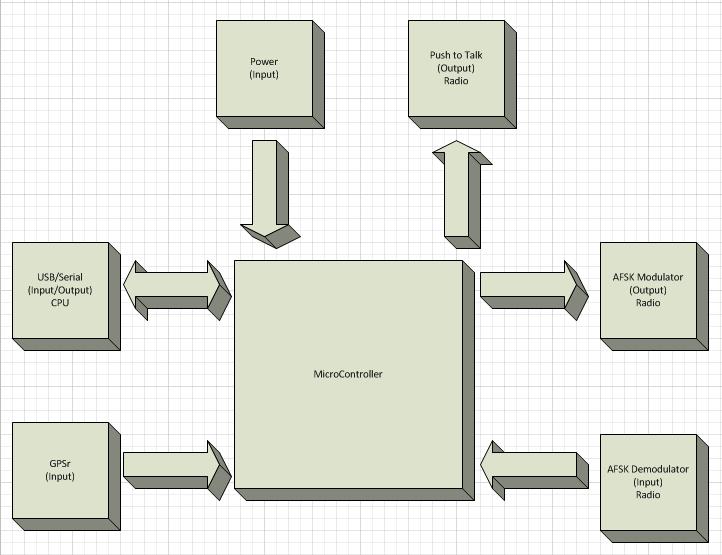
Terminal Node Controllers operate at the lowest layers of the Open Systems Interconnection (OSI) model (ISO/IEC 7498-1). At the physical level, TNCs implement the Bell 202 Modem standard to send and receive data to and from a radio. The radio itself is responsible for transforming the audio input and output into radio frequency signals. The Bell 202 Modem standard is simply Audio Frequency Shift Keying (AFSK) at 1200-baud with continuous phase using 1200 Hz and 2200 Hz audio tones. This scheme is also used in North America to transmit telephone caller data over the public switched telephone network to call display systems. In amateur radio, non-return to zero inverted (NRZI) encoding is used. A one ('1') is encoded as a change in frequency, and a zero ('0') is encoded as no change in frequency. NRZI encoding is not specific to Terminal Node Controllers; it is also used in Universal Serial Bus (USB Mode 1). The interface to the computer is left up to the implementer, but it has traditionally been RS232 (serial). At the data-link layer, computers can send and receive protocol frames to and from the TNC via the TNC KISS protocol, a modified version of Serial Line Internet Protocol (SLIP) commonly used for telephone modems. The KISS encapsulation is stripped away by the TNC and just the raw frames are sent out over the air. The frames are implemented using the AX.25 protocol. AX.25 is derived from the X.25 protocol with some changes to accommodate radio networks. While in principle AX.25 supports nearly all network layer protocols including Internet Protocol, in practice most communications happen through raw text data sent via AX.25. However, several application layer protocols such as Automatic Packet Report System (APRS) do exist and are in common use. Some TNCs are mostly self-contained and implement AX.25 internally. Those TNCs can function without a computer by using a dumb terminal. Other TNCs implement just the KISS protocol and let a full featured computer deal with AX.25 and higher level protocols. Special purpose TNCs exist which only connect to a radio and act as beacon controllers and/or provide telemetry capabilities. These purpose built TNCs are often used in tracking/control applications such as high altitude weather balloon tracking and telemetry.

While a wide range of terminal node controllers have been in existence for more than two decades, none exist that meet the proposed project’s goal of building a modern reliable low-cost easy to use terminal node controller with an open design in a small form factor that can be used with a computer and also portable without a computer. There are currently two broad categories of TNCs, commercial TNCs and homebrew TNCs. Commercial TNCs don’t meet my objective of low cost (e.g. Kantronics KPC-3+ lists for $229.95, PacComm TINY-2 MK-II lists for $190.00, etc.) nor is there a commercial TNC that can be used both as a portable APRS tracker and a computer connected data modem. I hypothesize that the high costs are due to low volume nature of amateur radio equipment sales and the manpower needed to design and produce a terminal node controller. Additionally, many are based on older designs which need more integrated circuits; 10 or 20 years ago, it wasn’t possible to get a 20MHz microcontroller for under $10. In the homebrew TNC space, there are over half a dozen microcontroller based designs published on the internet. However, they all lack some aspect of my project goals. Here are some examples. AVR\_UI\_TNC and ATmega-TNC don’t have open designs, they rely on rare modem ICs, their source code is written entirely in assembly, they don’t provide a USB interface, and they don’t support KISS which limits their usefulness. WhereAVR doesn’t really have a computer interface; it’s used for high-altitude balloon tracking projects. Other homebrew TNCs such as the TNC-X require a computer and can’t be used portably with a GPS. As you can see, the project I am proposing is unique in that it can be used as both a tracker and a data modem. Additionally, my TNC will have many desirable features that some TNCs lack such as a USB interface, an open design, and low cost. Based on my bill of materials, printed circuit board fabrication costs, and enclosure costs, the final product will cost under $100 with GPSr option and about $50 without GPSr option.

The following sections of this document will provide a more in-depth discussion of the proposed project. The functional description will outline the functional blocks of the design and provide a specification of the project as a whole. The circuit design section will present the circuits used to implement the essential terminal node controller functions as well as discuss the design process and decisions made along the way. The circuit design will be followed by a proposed circuit diagram and bill of materials. Finally, an approximate project timeline will be presented.

# Functional Description

The project design can be broken down into a series of several functional blocks. At its core is a microcontroller. The microcontroller interacts with a radio to send and receive AFSK modulated signals, a GPS to receive position information, a computer to provide a user interface, and a DC voltage source to power the device.



Figure

## Power Input

The power section will take an input DC voltage source and provide a regulated 5V output to power the microcontroller and other active devices in the circuit.

## Push to Talk

The push to talk section will be used to key (i.e. turn on or off) the radio’s transmitter. Circuitry will be used as a switch. An input to this section is a control line from the microcontroller. The other input is the Push-to-Talk line from the radio. The closing of the switch will short the Push-to-Talk line to ground which will turn on the radio’s transmitter. When the control line is HIGH (‘1’), the switch will close. When the control line is LOW (‘0’), the switch will open.

## AFSK Modulator

Audio Frequency Shift Keying (AFSK) requires the generation of audio tones at 1200 Hz and 2200 Hz. Since the phase has to be continuous, it isn’t enough to build two independent oscillators at 1200 Hz and 2200 Hz and switch between the two. Instead, I propose using a digital to analog converter (DAC) which is driven by the microcontroller. The output of the DAC needs to be scaled to 1Vpp; additionally, it needs to have any DC offset removed so that the signal is centered on 0V.

## AFSK Demodulator

The AFSK demodulator performs the reverse of the modulator. It adds a DC offset of 2.5V and amplifies the signal to 5Vpp, a signal that the microcontroller can process.

## GPSr

A global positioning system receiver (GPSr) will feed the microcontroller position information serially.

## USB/Serial

The USB/Serial interface provides communications between the computer and the microcontroller. In between the computer and the microcontroller will be a translator between the microcontroller’s TTL UART interface and the computers USB interface.

# Circuit Design

There are several pieces to the TNC hardware which are technically interesting. Those include modulating and demodulating the AFSK signals, interfacing with the radio, and interfacing with the computer. To tie all of these pieces together in an elegant way, a microcontroller is needed. For this project, I am proposing to use an Atmel ATMega1284p class microcontroller because of its low cost, ubiquity, availability of parts, availability of free / open source development tools, availability of documentation, and ease of use.

Modulating AFSK Signals

Modulating AFSK signals is very straight forward, especially at low frequencies. An R-2R network will be used with several output pins of the microcontroller to create a digital to analog converter. Using an R-2R network as a digital to analog converted (DAC) is cost effective as it can be constructed using just one or two different resistor values. The microcontroller will go through a table of pre-computed sine wave values. Depending on the frequency (1200 Hz or 2200 Hz), it will iterate over the values faster or slower. Using the same table and index for both frequencies will allow the phase to be continuous despite changing frequencies, a requirement for this project. If the storage size of the table becomes problematic, only the first quarter of the sine wave needs to be stored, the rest can be computed using simple mathematic operations. A coupling capacitor will be used to remove the DC offset introduced by the R-2R DAC which can only produce voltages between ground and 5V. The output voltage will also be attenuated to fit within the range used by most audio devices (1Vpp) and buffered with an operational amplifier to reduce the output impedance. I chose not to use an application specific integrated circuit such as the MX614 modem on a chip due to the cost and availability of the part. I also chose to build the R-2R ladder out of 1% resistors instead of a packaged resistor network. The only packaged 8 bit R-2R networks I could find were in DIP-16 packages, weren’t well stocked, and didn’t offer a cost advantage over using resistors.

Demodulating AFSK Signals

There are several methods for decoding AFSK signals. One common method in TNC implementations is to use an integrated circuit specifically designed for 1200 baud AFSK. However, such chips are often expensive and have limited availability. For example, the NJM2211 AFSK Demodulator costs over $3.50. That's more expensive than an ATMega32 micro-controller, and the NJM2211 only does decoding. Since low-cost is one of the goals of this project, a dedicated demodulator chip will not be used.

There are several (nearly) pure software solutions to AFSK demodulation. Using an analog to digital converter, one can sample the incoming signal to determine the frequency and hence the value, space or mark. Given the Nyquist theorm, one must sample the input signal at least twice as often as the input frequency. That is to say, the sample rate must be at least 4.4 kHz in our application. The samples are then fed through an algorithm to decide whether they represent a 1 or a 0 or no signal. Popular software AFSK demodulation techniques include the Goertzel algorithm, finite impulse response filters, and infinite impulse response filters. The Goertzel algorithm works best when the sample rate is an integer multiple of the target frequencies. That would mean sampling at 13.2 kHz and doing computations every handful of samples. That decoding method is rather CPU intensive. Especially considering that at the same time, data is being buffered and sent to the computer. A more powerful microcontroller could be used, but that would increase cost.

There are several hardware solutions to the AFSK decoding problem. Some have used peak detectors. They split the signal into two paths, one with a band pass filter centered at 1200Hz and the other with a band pass filter centered at 2200 Hz. The signals at the filter outputs are compared to see which one is more dominant. This has the disadvantage of requiring several active components with some passive components.

The solution I decided on uses a zero cross detector. Based on the time between zero crossings, the microcontroller can determine the frequency of the input signal and thus the encoded information. The 1Vpp input audio signal from the radio will be given an offset and amplified with a summing amplifier such that the signal going into the microcontroller is centered at 2.5V and ranges from about 0V to 5V. The input capture pin will trigger interrupts on rising edges. An input capture interrupt handler will be used to compute the time since the last rising edge. A variable will be used to track the current frequency. A timer based interrupt handler will decode the incoming information. Toggles (i.e. changes in frequency) can be used to synchronize the 1200 baud sample clock. When it is time to sample, the timer based interrupt handler will shift the data into a variable. It will look for the AX.25 frame delimiter (0x7e – 0b01111110) for synchronization, to know where bytes begin/end.

Interfacing with the Radio

Most TNCs get their input from the radio's speaker jack. The TNCs send their output via the microphone connection. The microphone connector typically has an audio component and a push-to-talk (PTT) line. Closing the PTT circuit activates the transmitter. This can be done by using a transistor in a switch configuration. A common connecter used for radio interfacing is DIN5. It has enough pins for audio in, audio out, ground, and push to talk. Since it’s a common connector and nothing stands out as being superior to it, I went with DIN5 in my design. I had considered RJ-11 (telephone style plug) because they are so common and cheap. However, I reconsidered after I thought of the potential harm/damage it could cause if someone accidentally plugged it into a telephone line.

Interfacing with the Computer

Current TNC designs use RS232 serial connections to interface the TNC and the computer. However, fewer and fewer new computers are being built with serial ports. Most peripherals now use the Universal Serial Bus (USB). USB-to-Serial adapters are available, but they cost around $15. Developing a USB interface seems like the answer, but it adds complexity and it closes the door to compatibility with existing software. Fortunately, we can have our cake and eat it too. I propose using an FT232RL (usb to serial IC) in the TNC. It would provide a USB connection to the computer, and the TNC would still show up as a serial device on the host computer. The connector would be a USB Mini B receptacle.

Miscellaneous Hardware

The device will be powered through an external DC power source. This will likely be a 12V "wall wart" style power supply connected with a standard barrel connector. 12V is a common voltage for amateur radio gear. There will be an on/off switch. The positive side of the power input will go through a protection diode (1N4001) which will protect the circuit from reversed polarity input and excessive current. Then, the input voltage will be stepped down to 5V with a 7805.

A clock frequency of 14.7456 MHz will be used for the microcontroller as it will give the lowest error rate for both AFSK and USART. The 14.7456 MHz clock can be divided to 1200.00 Hz, 9600.00 Hz, and 2199.52 Hz (very close to 2200 Hz).

A JTAG debug / In-circuit programming port will be present as well to ease development.

To summarize, there will be the following external connectors: Computer Port (USB Mini B), Radio Port (DIN5 female), JTAG/ICP (CONN-2x5), +9V (Barrel Connector). All of the connectors will mount along the same edge of the printed circuit board (PCB) to allow for the greatest flexibility in enclosures. The majority of devices will be surface mount. This reduces costs and allows for more flexibility in printed circuit board design. Where possible, 0805 and SOIC type component packages will be selected. 0603 and TQFP will also be accepted when larger packages aren't available. Anything that cannot be hand soldered will not be used.

# Proposed Circuit Diagram

Insert it here

# Proposed Bill of Materials

Insert Here

# Approximate Timeline

Insert Gantt chart here

# Conclusion

In conclusion, …

There are several challenging aspects of this project. Most of them are software related. In the firmware, the key areas of difficulty will be buffer management and AFSK demodulation. The TNC KISS protocol specifies how frames should be buffered, how to know when it is safe to transmit, and what to do when the transmit buffer gets full. Managing frames going into and out of the transmit buffer is going to be difficult because of the parallel nature of the code, queuing and de-queuing is done via interrupt service routines. The AFSK demodulation will also be challenging due to the complexity; there are 2 threads of execution, the input capture interrupt handler and the timer interrupt handler. Doing the state management for demodulating the signal will require a lot of thought and in depth focus. My plan of attack is to design the software in advance on paper with a flow chart or UML, provide gratuitous comments in the code to ease readability and understanding, and heavily test the resulting code. For AFSK demodulation I will test the code using the “TNC Test CD” which provides a track containing tens of minutes of recording of amateur packet radio activity in Los Angeles during rush hour. The track is commonly used to test AFSK demodulators in order to directly compare performance. The transmit buffer management can be tested by sending various amounts of data with a computer and observing the output line.

There are some possible features which are not feasible to implement in the first revision of this project. These tasks have a high cost in terms of development time and a low reward in terms of usefulness to potential users. In other words, these are features which don’t significantly impact the normal operation of the project. I will keep track of these items in a list, and if I am ahead of schedule I will consider implementing some of them. These tasks include supporting additional AFSK baud rates (i.e. 300 for HF, 9600 for UHF), supporting full duplex operation, and support for multiple ports. There is only a fraction of on the air activity at 300 baud and 9600 baud. In the initial project, only 1200 baud will be supported since it captures the widest part of the market. Similarly, full duplex is only used in special situations (e.g. digipeaters) and would require expensive hardware like a duplexer and a duplex capable radio (or two separate radios). Not supporting duplex doesn’t limit normal operation by a person, so it won’t be implemented in the first pass. Support for multiple ports is another feature that’s rarely used and only implemented on a few TNCs. Again, this feature isn’t useful in to the general population of packet radio operators.

# References

List references here

# Appendices