Final Project I

Terminal Node Controller Project

Project Proposal Report

Thomas Cort
10/18/2012

Executive Summary

I propose developing a Terminal Node Controller (TNC), sometimes called a radio modem or packet controller. I will develop two versions, one that works with a computer to connect to packet radio networks and another with a built-in global positioning system receiver that can encode position information for transmission on packet radio networks. In other words, the end product is a personal tracking system. A hiker can take the portable TNC out on the trail with a small hand held radio; it will determine its location using the GPS network of satellites and transmit the location through the radio. The base TNC will be connected to a radio and a computer. Software I develop will be running on the computer, and it will allow the computer to display the location of the hiker on a map on the screen.

The TNC will be responsible for interfacing with the radio (modulating and demodulating 1200 Baud Audio Frequency Shift Keying with Non-Return to Zero Inverted encoding), interfacing with a computer via USB, and interfacing with a GPS receiver module in the portable version. The only physical difference between the portable version and the base version will be the GPS module; the other differences will be handled in firmware.

Table of Contents

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 zero ('0') is encoded as a change in frequency, and a one ('1') 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. Most 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 the GPSr option and about $50 without the 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.

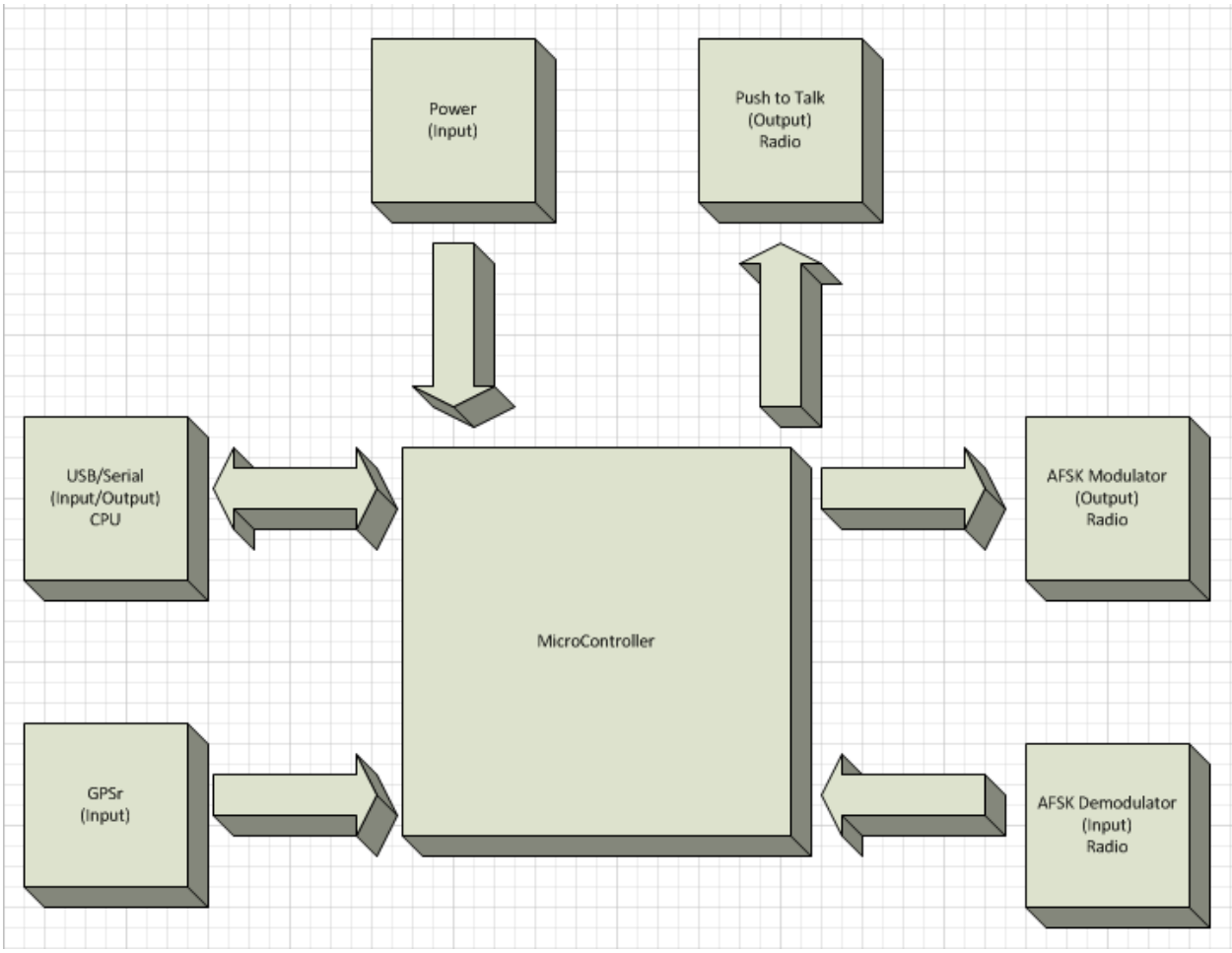


Figure 1

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.

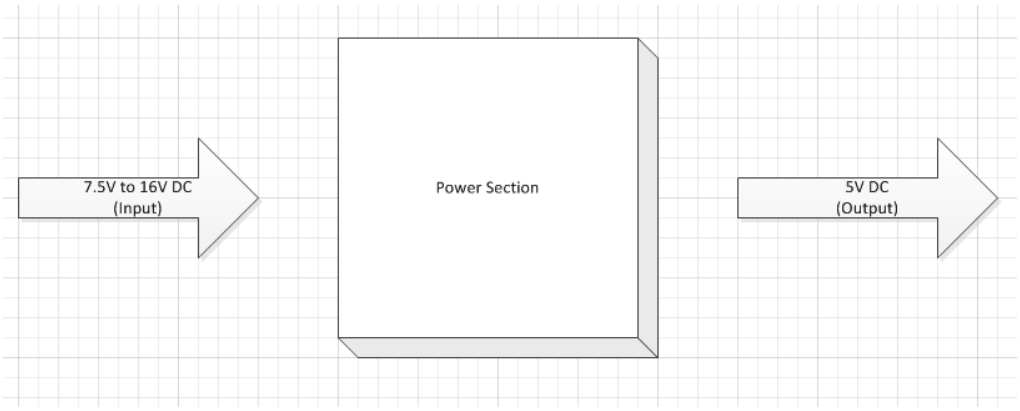


Figure 2

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.

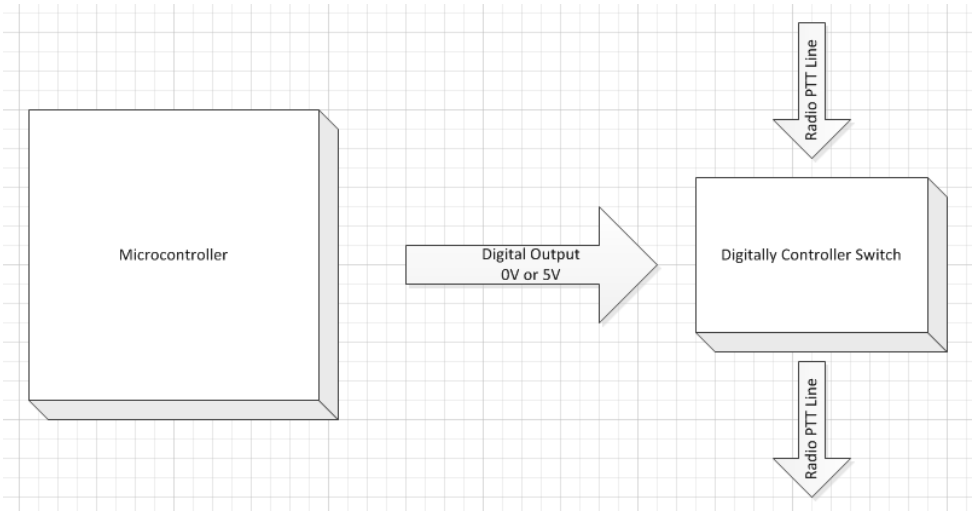


Figure 3

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 about 1Vpp; additionally, it needs to have any DC offset removed so that the signal is centered on 0V.

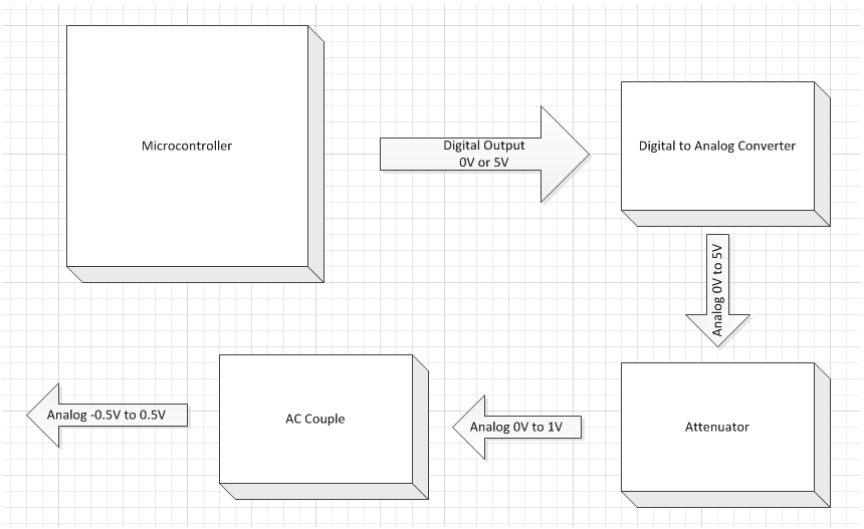


Figure 4

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 sample and process.

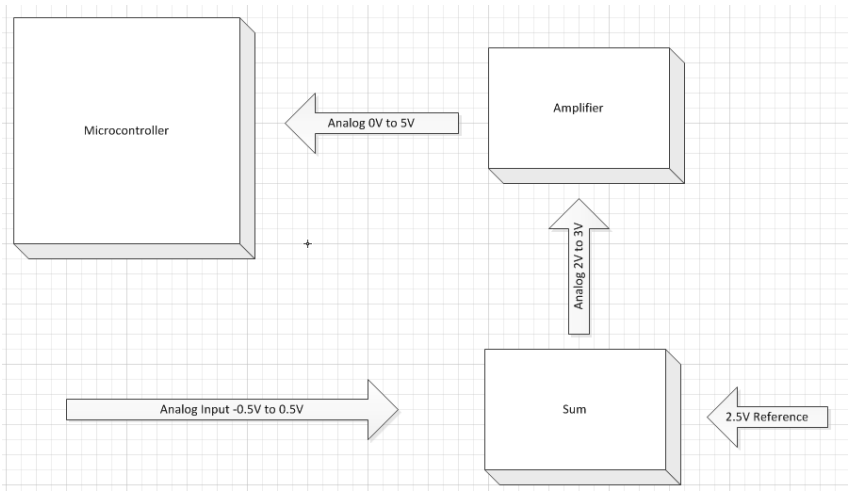


Figure 5

GPSr

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

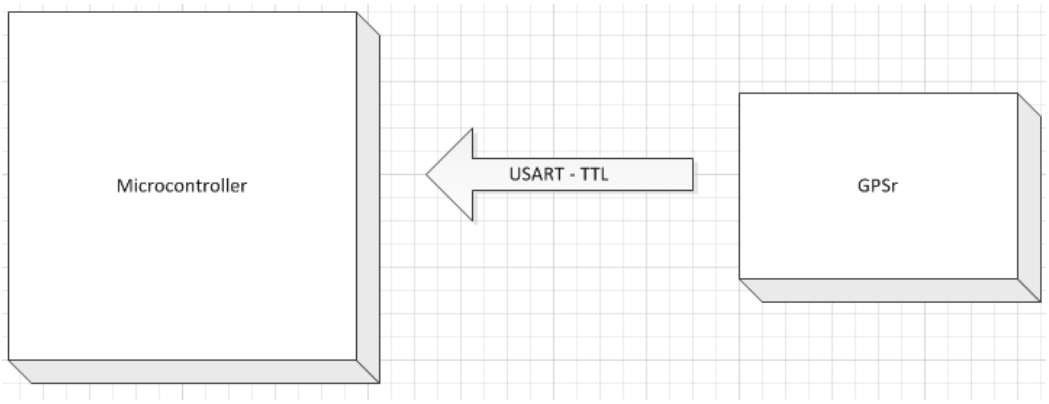


Figure 6

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.

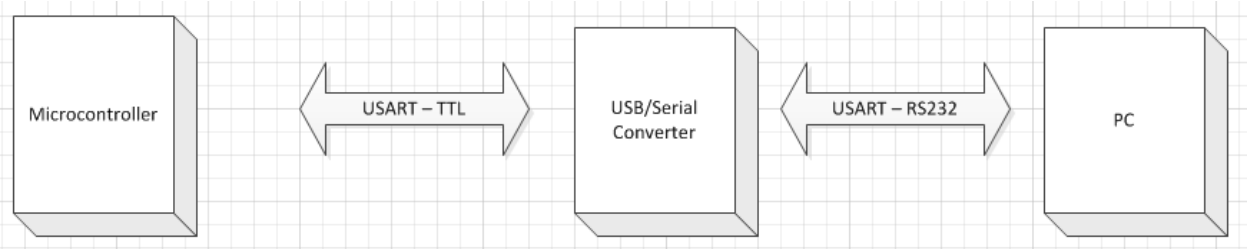

Microcontroller
USART – TTL
USB/Serial
Converter
USART – RS232
PC

Figure 7

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, receiving position information from the GPSr, 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 microcontroller because of its low cost, ubiquity, availability of parts, availability of free / open source development tools, availability of documentation, my prior knowledge/experience with Atmel ATMega chips, 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. The attenuation will be a simple voltage divider. A potentiometer will be used to allow for fine tuning as different radios need different audio levels for optimal performance. 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. I have, in my opinion, a better solution which is explained below. However, as the ADC inputs in my design weren't being used for anything, I decided to connect the RX input signal to one of the analog to digital converter pins so that in the event

I'm wrong, I have the option to explore the viability of some of the sample based software frequency detection algorithms.

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 both hardware and software to implement a zero cross detector. Zero crosses are detected using the ATMega's input capture peripheral. 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) in the stream 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 radio's microphone jack. The microphone connector typically has an audio component and a push-to-talk (PTT) line. Shorting the PTT line to ground activates the radio's 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 and/or damage it could cause if someone accidentally plugged it into a telephone line.

I decided to use the DIN5's fifth pin as a TX enable pin. This can be used to prevent or allow the TNC to transmit based on some external factor. The most common use case for this would be someone using two transmitters in relatively close proximity, one for data and another for voice. When the voice radio is transmitting, the TNC can be signaled to avoid transmitting until the voice transmission is complete.

Receiving position information from the GPSr

Instead of designing a GPS receiver from scratch or designing the supporting circuitry around a specialized IC, I propose using a self-contained GPSr module. The microcontroller will communicate with the GPSr module using UART at 4800 baud. The ATMega1284p has multiple UART controllers, so the module doesn't require any additional parts. The data will be in NMEA 0183 format which is fairly informative and easy to parse.

Interfacing with the Computer

Current TNC designs typically use DB9 connectors for RS232 serial connections to interface the TNC and the computer. However, fewer and fewer new computers are being built with serial ports. Most computer peripherals now use the Universal Serial Bus (USB). USB-to-Serial adapters are available to connect legacy equipment to modern PCs, but they cost around $15. Developing a USB interface

seems like the obvious answer, but it adds complexity. It also adds costs if you go through the procedure to get a valid USB vendor ID and product ID. Creating a USB interface also closes the door to compatibility with existing KISS mode TNC 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.

Power Section

The device will be powered through an external DC power source. As 12VDC is a common voltage for amateur radio gear, the input voltage range should accept 12V +/- a few volts. This will likely be supplied by a 12V "wall wart" style power supply connected with a standard barrel connector or, for the portable version, an external battery. There will be an on/off switch. The positive side of the power input will go through a protection diode 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.

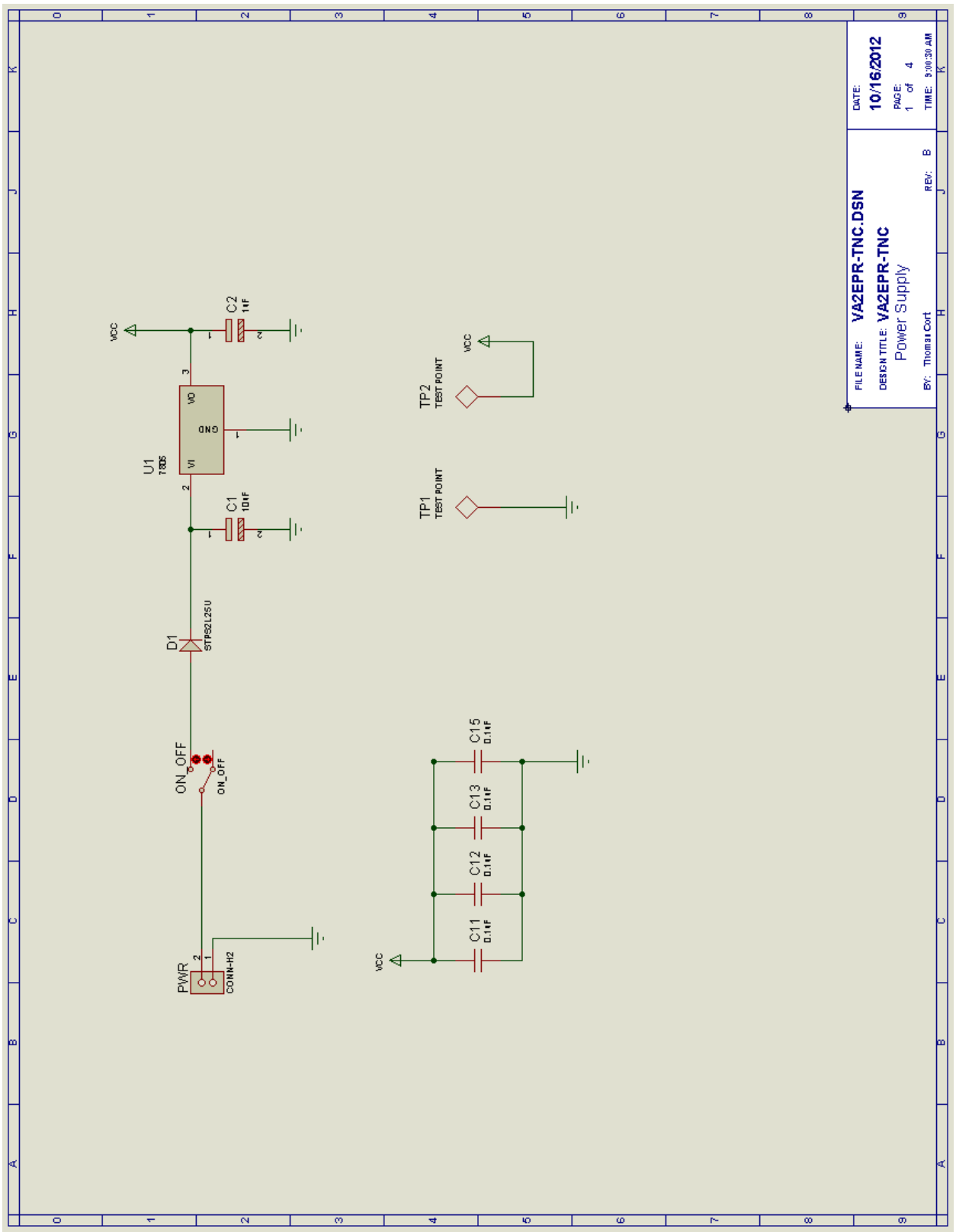
Miscellaneous Hardware

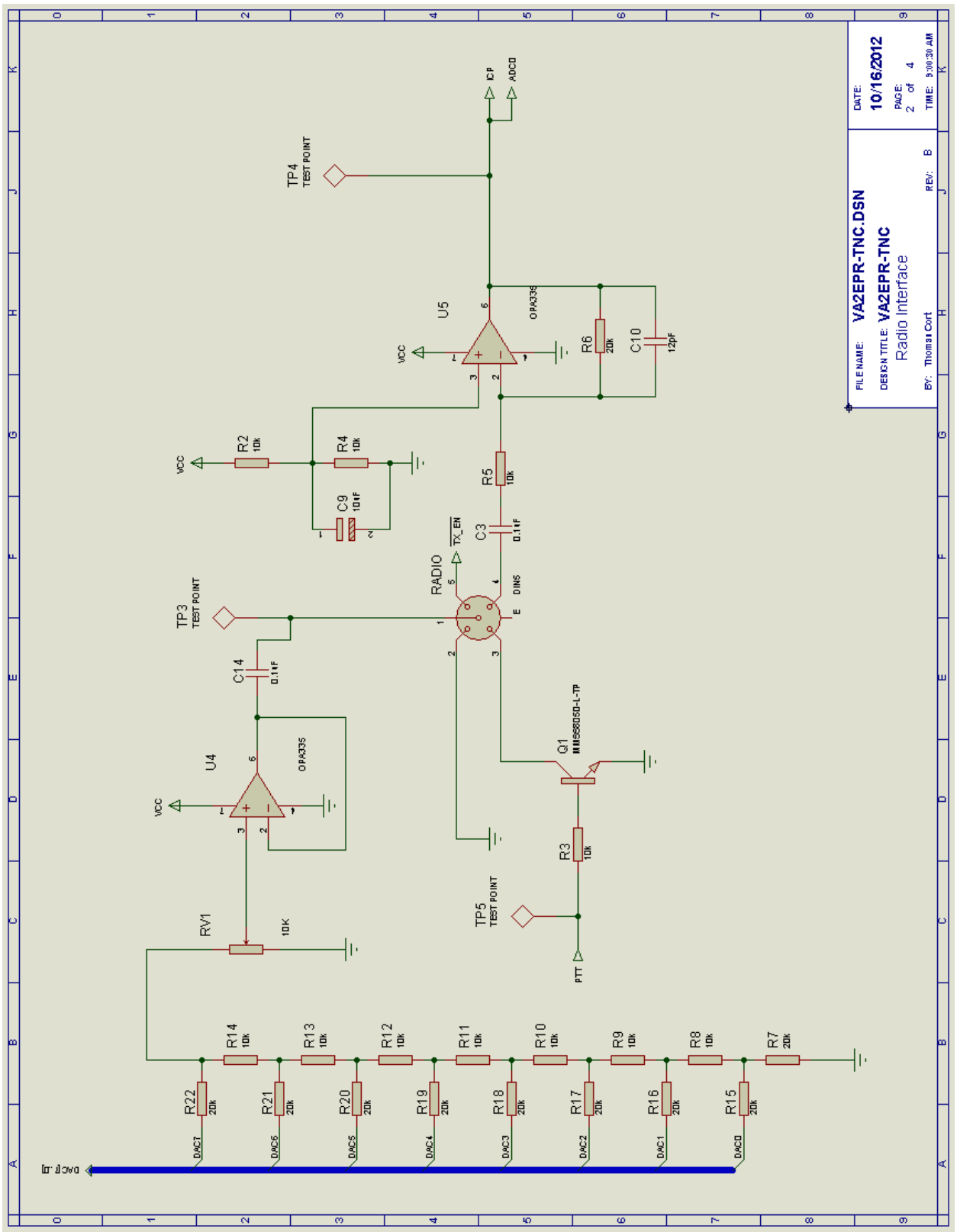
A clock frequency of 14.7456 MHz will be used for the microcontroller as it will give the lowest error rate for both AFSK and USART. The 14.7456 MHz clock can be divided to 1200.00 Hz, 9600.00 Hz, and 2199.52 Hz (very close to 2200 Hz).

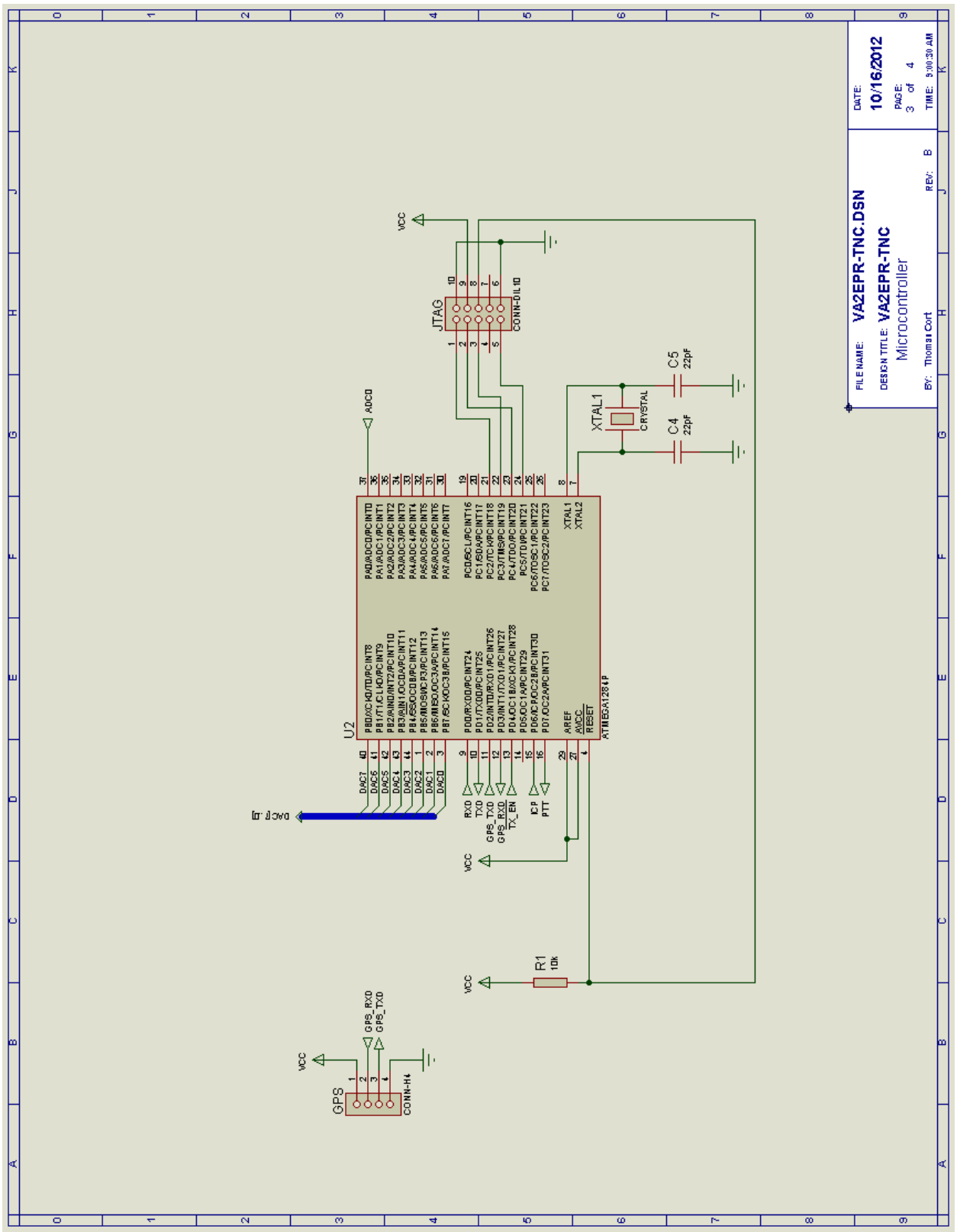
A JTAG debug / In-circuit programming port will be present as well. This is necessary as the microcontroller doesn’t come preprogrammed. The connector will only be accessible when the enclosure is opened. The JTAG port should only be needed for programming the chip and debugging. It isn’t pertinent to everyday use.

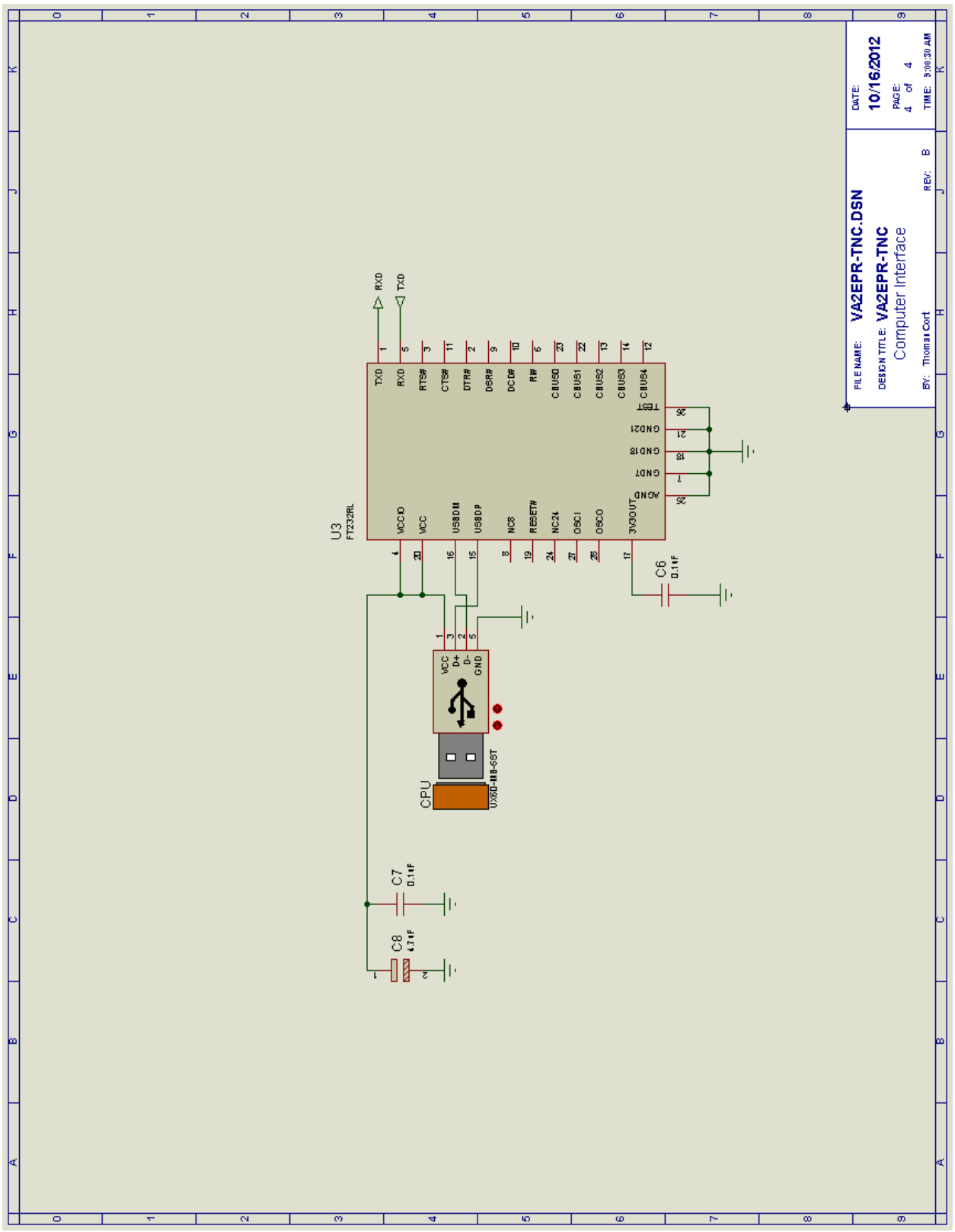
To summarize, there will be the following external connectors: Computer Port (USB Mini B), Radio Port (DIN5 female), JTAG/ICP (CONN-2x5), +9V (Barrel Connector). All of the connectors will mount along the same edge of the printed circuit board (PCB) to allow for the greatest flexibility in enclosures. 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




DATE: 10/16/2012
PAGE: 2 of 4
TIME: 9:00:30 AM
FILE NAME: VA2EPR-TNC.DSN
DESIGN TITLE: VA2EPR-TNC
Radio Interface
REV: B
BY: Thomas Cort
ICP
ADCO
TP4
TEST POINT
U5
OPA335
VCC
R6
20k
C10
12pF
R2
10k
R4
10k
C9
10nF
R5
10k
C3
0.1nF
TX_EN
RADIO
DIN5
TP3
TEST POINT
C14
0.1nF
U4
OPA335
VCC
Q1
MMBS8050-L-TP
R3
10k
TP5
TEST POINT
PTT
RV1
10K
R14
10k
R13
10k
R12
10k
R11
10k
R10
10k
R9
10k
R8
10k
R7
20k
R22
20k
R21
20k
R20
20k
R19
20k
R18
20k
R17
20k
R16
20k
R15
20k
DAC7
DAC6
DAC5
DAC4
DAC3
DAC2
DAC1
DAC0
DACBUS


FILE NAME: VA2EPR-TNC.DSN
DESIGN TITLE: VA2EPR-TNC
Microcontroller
BY: Thomas Cort
REV: B
DATE: 10/16/2012
PAGE: 3 of 4
TIME: 9:00:30 AM
U2
ATMEGA1284P
JTAG
CONN-DIL10
GPS
CONN-H4
XTAL1
CRYSTAL
C4 22pF
C5 22pF
R1 10k
VCC
ADC0
DAC0..DAC7
RXD
TXD
GPS_TXD
GPS_RXD
TX_EN
ICP
PTT
GPS_RXD
GPS_TXD


RXD
TXD
U3
FT232RL
TXD
RXD
RTS#
CTS#
DTR#
DSR#
DCD#
RI#
CBUS0
CBUS1
CBUS2
CBUS3
CBUS4
TEST
GND21
GND18
GND7
AGND
VCCIO
VCC
USBDM
USBDP
NC8
RESET#
NC24
OSCI
OSCO
3V3OUT
C6
0.1uF
VCC
D+
D-
GND
CPU
UX60-MB-5ST
C7
0.1uF
C8
4.7uF
FILE NAME: VA2EPR-TNC.DSN
DESIGN TITLE: VA2EPR-TNC
Computer Interface
BY: Thomas Cort
REV: B
DATE: 10/16/2012
PAGE: 4 of 4
TIME: 9:00:30 AM

Proposed Bill of Materials

Design Title : VA2EPR-TNC
Author : Thomas Cort
Revision : B
Design Created : Thursday, August 23, 2012
Design Last Modified : Tuesday, October 16, 2012
Total Parts In Design : 57

22 Resistors

Quantity:	References	Value	Order Code	Cost
12	R1-R5, R8-R14	10k	P10.0KFCT-ND	$0.11
10	R6, R7, R15-R22	20k	P20.0KFCT-ND	$0.11
Sub-totals:				$2.42

15 Capacitors

Quantity:	References	Value	Order Code	Cost
2	C1, C9	10uF	493-4165-1-ND	$0.78
1	C2	1uF	399-8265-1-ND	$0.43
8	C3, C6, C7, C11-C15	0.1uF	445-1349-1-ND	$0.11
2	C4, C5	22pF	399-1113-1-ND	$0.08
1	C8	4.7uF	399-3696-1-ND	$0.36
1	C10	12pF	709-1169-1-ND	$0.17
Sub-totals:				$3.56

5 Integrated Circuits

Quantity:	References	Value	Order Code	Cost
1	U1	7805	L7805CDT-TR	$0.77
1	U2	ATMEGA1284P	ATMEGA1284P-AU-ND	$8.84
1	U3	FT232RL	768-1007-1-ND	$4.83
2	U4, U5	OPA335	296-13467-5-ND	$3.42
Sub-totals:				$21.28

1 Transistors

Quantity:	References	Value	Order Code	Cost
1	Q1	MMSS8050-L-TP	MMSS8050-L-TPMSCT-ND	$0.48
Sub-totals:				$0.48

1 Diodes

Quantity:	References	Value	Order Code	Cost
1	D1	STPS2L25U	497-5573-1-ND	$0.83
Sub-totals:				$0.83

13 Miscellaneous

Quantity:	References	Value	Order Code	Cost
1	CPU	UX60-MB-5ST	H2959CT-ND	$1.34
1	GPS	CONN-H4	28500-ND	$42.88
1	JTAG	CONN-DIL10	S9169-ND	$0.65
1	ON_OFF	ON_OFF	EG4313-ND	$3.65
1	PWR	CONN-H2	CP-037A-ND	$0.98
1	RADIO	DIN5	CP-2350-ND	$1.13
1	RV1	10K	Digikey 3361P-103GLFDKR-ND	$1.47
5	TP1-TP5	TEST POINT	5007K-ND	$0.41
1	XTAL1	CRYSTAL	631-1106-ND	$0.48
Sub-totals:				$54.63

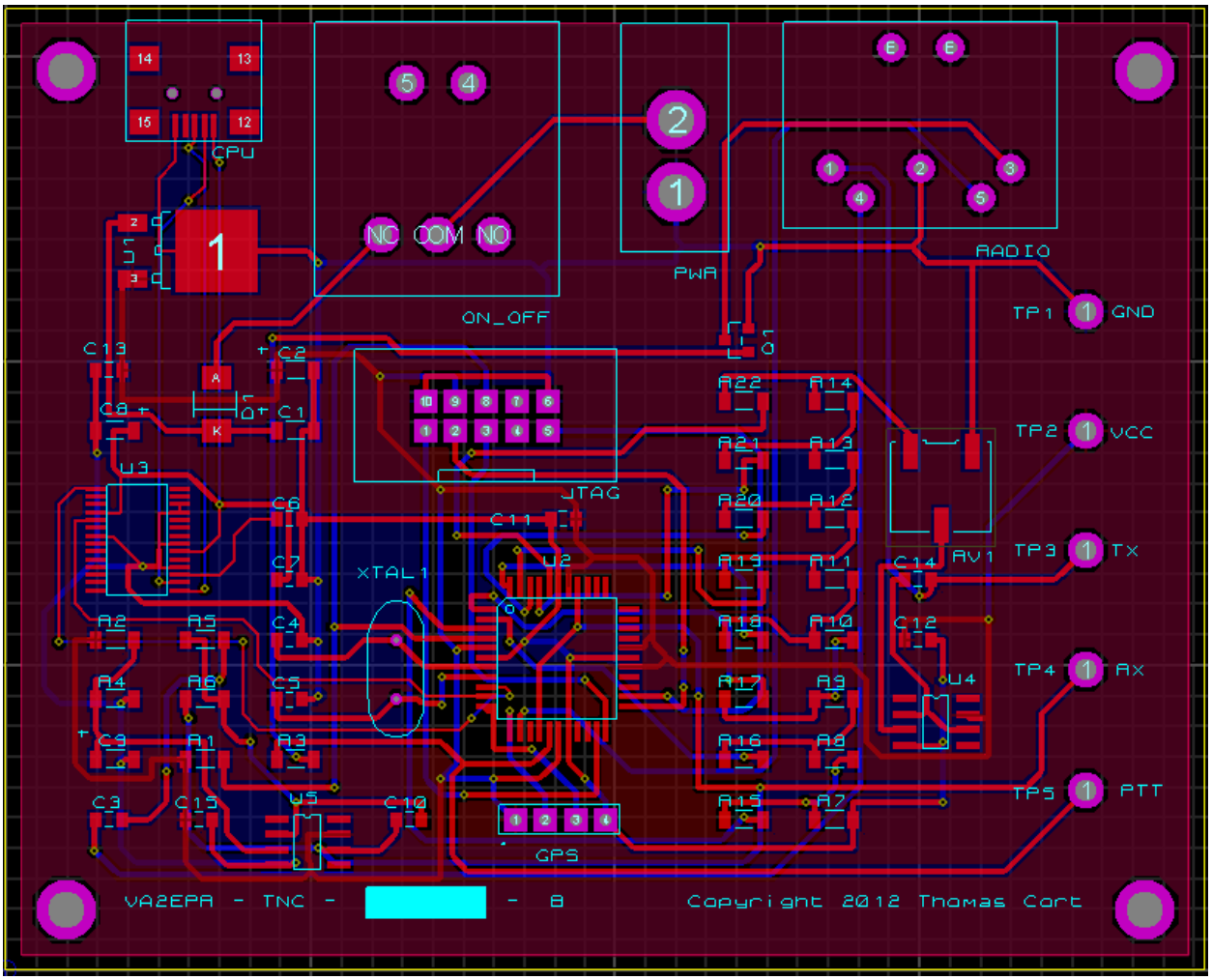
Totals: $83.20

Tuesday, October 16, 2012 9:12:16 AM

Additional Materials

1	2Layer Green PCB 10cm x 10cm Max	$2.49
1	TP-41 5.370 x 2.059 x 4.144 Aluminum Enclosure	$7.38
1	CABLE ASSY STR 2.5MM MONO 6'	$3.01
1	CABLE ASSY STR 3.5MM STEREO 6'	$3.08
1	CONN CIRCULAR DIN 5 PIN MALE	$1.97
1	CBL USB A-MNI B CON 3' 28/28 AWG	$2.07
1	POWER SUPPLY 100-240V 1000MA	$9.18
1	4 x 10mm M3 Round Standoff & 8 Screws	$0.63

Printed Circuit Board Layout



Approximate Timeline

	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		Brainstorming	2 days	Wed 8/22/12	Thu 8/23/12	
2		Preliminary Schematic	6 days	Wed 8/22/12	Wed 8/29/12	1
3		Bill of Materials	7 days	Wed 8/22/12	Thu 8/30/12	2
4		Order Parts	2 days	Thu 8/30/12	Fri 8/31/12	3
5		Breadboard Proof of Concept	10 days	Fri 8/31/12	Thu 9/13/12	4,2
6		Develop Product Specification	4 days	Mon 9/10/12	Thu 9/13/12	3
7		Gant Chart	5 days	Thu 9/6/12	Wed 9/12/12	
8		Project Proposal	4.8 wks	Thu 9/13/12	Tue 10/16/12	7
9		Project Plan Presentation	2 days	Wed 10/17/12	Thu 10/18/12	8
10		Create PCB Packages	5 days	Thu 9/6/12	Wed 9/12/12	3
11		Place Parts on PCB	5 days	Thu 9/13/12	Wed 9/19/12	2
12		Route Parts on PCB	1 day	Wed 9/19/12	Wed 9/19/12	11
13		Order PCBs - Wait for Arrival	3 wks	Thu 9/20/12	Wed 10/10/12	10,11,12
14		Develop Firmware	12 wks	Fri 8/31/12	Thu 11/22/12	2
15		Make Radio Interface Cables	18 days	Thu 9/27/12	Mon 10/22/12	
16		Populate PCB	4 days	Wed 10/10/12	Mon 10/15/12	13
17		Buy Enclosures	1 day	Fri 9/21/12	Fri 9/21/12	11
18		Schematic - Rev B	5 days	Wed 10/10/12	Tue 10/16/12	
19		Layout - Rev B	5 days	Wed 10/10/12	Tue 10/16/12	
20		Bill of Materials - Rev B	1 day	Tue 10/16/12	Tue 10/16/12	18
21		Order Parts - Rev B	2 days	Thu 10/18/12	Fri 10/19/12	20
22		Order PCBs & Wait for Arrival - Rev B	11 days	Thu 10/18/12	Thu 11/1/12	19
23		Populate Boards - Rev B	3 days	Fri 11/2/12	Tue 11/6/12	21,22
24		Assemble Enclosure	1 wk	Wed 11/7/12	Tue 11/13/12	17,23
25		Write Report	13 days	Wed 11/14/12	Fri 11/30/12	24
26		Test/Debug/Improve/Fix	11 days	Fri 11/23/12	Fri 12/7/12	

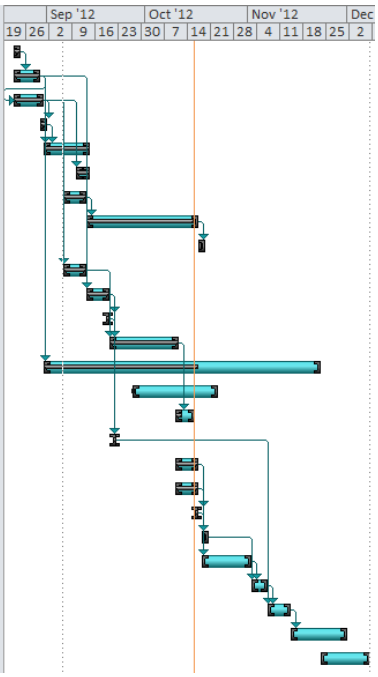


Figure 8

In planning the project, I expect that the firmware will be much more challenging, and thus take more time, than the hardware. The hardware is made up of several standard textbook circuit blocks while the firmware will need to implement custom signal processing and resource management. As a result, I arranged the project timeline such that the hardware will be done early, leaving plenty of time for firmware development and a second hardware revision. As it turns out, I will be making a revision B board, so the scheduling choice turned out to be a good one. I feel that it is important to get a large portion of the firmware written in the first semester as the PC software will be a large undertaking in the second semester.

Draft Product Specification

Table 1

Dimensions	5.500” W x 2.250” H x 4.500” D
Weight	TBD
Input Power	DC 7.5V to 16V, 75 mA Typical (1.5A Max)
RX Audio	0.1Vpp Min, 1Vpp Ideal, 5.5Vpp Max
TX Audio	Up to 5Vpp
Ports	2.1mm Barrel (Power Input), DIN5 (Radio), USB Mini B (Computer), JTAG (Debug/Program)
Modulation	1200 Baud AFSK NRZI with 1200 Hz & 2200 Hz tones (half-duplex).
Maximum Transmission Unit (MTU)	8KB (TX), Unlimited (RX)
Processor	ATMega1284p running at 14.7456 MHz
Host Operating System Requirements	Windows 98 or greater, Linux, Mac OS 8 or greater, Android, or Windows CE.NET 4.2 or greater.
Recommended Host Software	Any packet radio software supporting KISS mode TNCs (e.g. AGWPE (Windows), Xastir (Linux), and many more).
Operating Temperature Range	0C to 70C
Standards Implemented in Firmware	Bell 202, KISS, NMEA, CSMA/CA
Compliance	This device is a prototype and hasn’t been formally tested for compliance with any standards (e.g. FCC Class B, UL, CE, etc.).

Conclusion

In conclusion, this will be an exciting and challenging project. I believe that it is feasible to meet my 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. I expect the hardware to be fairly straightforward. The firmware will require time to develop, but it is feasible. I do expect there to be room for future work as it isn't practical to implement every possible feature. I will focus on getting the most important features working well in the first revision. At a minimum, the project will meet the functional requirement of being able to remotely tracking a hiker and connecting to other AX.25 network nodes.

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. Additional challenges will be waiting when the PC software is developed. The AX.25 protocol specification is over 140 pages.

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 radio systems). 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

- http://en.wikipedia.org/wiki/Bell_202_modem
- http://en.wikipedia.org/wiki/Frequency-shift_keying
- http://en.wikipedia.org/wiki/Non-return-to-zero#Non-Return-to-Zero_Inverted_.28NRZI.29
- http://en.wikipedia.org/wiki/KISS_%28TNC%29
- http://en.wikipedia.org/wiki/AX.25
- http://en.wikipedia.org/wiki/Automatic_Packet_Reporting_System