VHF Radio Beacon For CubeSAT

By

Neal Makela

Russell Jones

Jeffery Tlusty

Final Report for ECE 445, Senior Design, Spring 2013

TA: Kevin Basset

1 May 2013

Project No. 10

Abstract

A radio transmitter beacon is designed for use on a CubeSAT style satellite. This transmitter enables the location of the satellite and serves as a backup communication link should the main radio malfunction. Stringent power and size requirements were met to ensure compatibility with other systems onboard the satellite. The transmitter is a Frequency Modulation (FM) type radio which operates in the 144-148 MHz frequency band. A microcontroller interprets data from the CubeSAT main computer via a serial interface. An Audio Frequency-Shift Keying (AFSK) modulation scheme is then used to transmit data to a terrestrial system operated by the CubeSAT organization at the University of Illinois. This project was successful in communication and meeting physical size requirements, but suffered problems such as high power consumption and low transmit power. Future revisions will require a more efficient amplifier and more strategic Printed Circuit Board (PCB) layout.

Contents

1 Introduction 1

1.1 Purpose 1

1.2 Specifications 1

1.3 Modules 1

1.3.1 AFSK Signal Generator 2

1.3.2 Low Pass Filter 2

1.3.3 VCO 2

1.3.4 Channel Filter 2

1.3.5 Power Amplifier 3

1.3.6 Output Filter 3

1.3.7 Voltage Regulator 3

2 Design 4

2.1 AFSK Signal Generator 4

2.1.1 Design Procedure 4

2.1.2 Design Details 4

2.2 Low Pass Filter 6

2.2.1 Design Procedure 6

2.2.2 Design Details 6

2.3 Voltage Controlled Oscillator 6

2.3.1 Design Procedure 6

2.3.2 Design Details 7

2.4 Channel Filter 7
2.4.1 Design Procedure 8
2.4.2 Design Detail 9
2.5 Power Amplifier 10
2.5.1 Design Procedure 10
2.5.2 Design Details 10
2.6 Output Filter 11
2.6.1 Design Procedure 11
2.6.2 Design Details 12
2.7 Voltage Regulator 12
2.7.1 Design Procedure 12
2.7.2 Design Details 13
3 Verification 14
3.1 PIC Microcontroller 14
3.1.1 Testing Procedure 14
3.1.2 Results 14
3.2 VCO 15
3.2.1 Testing Procedure 15
3.2.2 Results 15
3.3 Channel Filter 15
3.3.1 Testing Procedure 15
3.3.2 Results 15

3.4 Amplifier . . . 16
3.4.1 Testing Procedure . . . 16
3.4.2 Results . . . 16
3.5 Output Filter . . . 17
3.5.1 Testing Procedure . . . 17
3.5.2 Results . . . 17
3.6 Voltage Regulator . . . 18
3.6.1 Testing Procedure . . . 18
3.6.2 Results . . . 18
4 Costs . . . 18
4.1 List of parts and equipment needed . . . 18
5 Conclusions . . . 19
5.1 Accomplishments . . . 19
5.2 Uncertainties . . . 19
5.3 Ethical considerations . . . 19
5.4 Future work / Alternatives . . . 19
Appendix . . . 21
A Final Revision Schematics . . . 22
B Requirements and Verifications for Final Revision . . . 25
C PIC Code . . . 29
C.1 LUT Generating Code (Python) . . . 29
C.2 Main Program . . . 30
C.3 User Functions Program . . . 32

D Final Revision PCB layout 41

E Past Revision Block Diagrams 42

F Parts 42

G Future PCB Layout 44

1 Introduction

1.1 Purpose

This project entailed designing a backup communication system for a CubeSAT style satellite. The CubeSAT organization has a need for such a transmitter to use for an upcoming launch. The beacon, consisting of an FM transmitter circuit, allows a terrestrial antenna to locate the satellite as it orbits around the earth. In addition to enabling location of the satellite, the beacon transmits status data encoded in AFSK tones generated by an onboard microcontroller.

In general, size, power, and environmental factors (vibrational tolerance and thermal management) play a crucial role in determining the future success of this project. If future revisions are designed correctly, this project has the potential to be flown into space and operate for an indefinite period of time.

1.2 Specifications

- 1257 mW total power consumption
- 100 mW transmit power
- 80 mm x 25 mm PCB footprint
- 144-148 MHz transmit frequency tunable before launch
- AFSK tone conversion and transmission at 1200 baud

1.3 Modules

The final revision of this project was constructed of the modules depicted in Fig. 1 below. This revision is the product of two additional failed versions of the project that were attempted earlier in the semester. The initial design utilized two mixers to take the baseband signal to the transmit frequency as seen in Fig. 16 (Appendix E). It was found that, even though the initial design successfully passed the design review, the double-mixing scheme was fundamentally flawed and would make data transmission impossible.

The second revision as seen in Fig. 17 (Appendix E) used a single mixer with a Voltage-Controlled Oscillator (VCO). It was seen that while a VCO made data transmission possible, the mixer caused signal power to drop to unacceptably low levels.

The final revision uses a single VCO for transmission. This scheme, although technically not a mainstream

design, proved to work much better than predicted. Component counts were drastically reduced as well as the PCB footprint of the entire circuit.

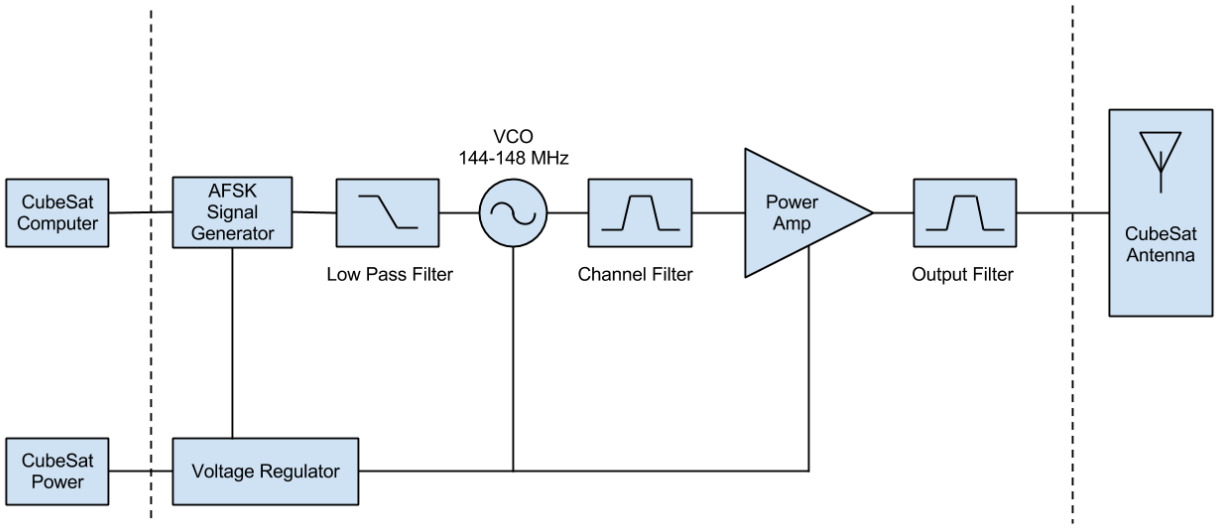

Figure 1: Final Revision System Block Diagram

1.3.1 AFSK Signal Generator

An AFSK signal is generated by a PIC16 Microcontroller. This microcontroller also communicates with the CubeSAT main computer to determine when to transmit, as well as the specified data for transmission.

1.3.2 Low Pass Filter

Next, the signal is fed into a low pass filtering stage. In this design, the low pass filter was deemed somewhat redundant and ended up being used as an impedance matching attenuator stage.

1.3.3 VCO

To perform the actual frequency modulation, a VCO takes the baseband signal as a control voltage input and generates the FM signal at a fixed frequency between 144-148 MHz.

1.3.4 Channel Filter

Originally used to filter out signal images generated by mixers in previous designs, the channel filter serves to attenuate out-of-band noise before the FM signal is amplified. This filter also suppresses square wave harmonics generated by the VCO.

1.3.5 Power Amplifier

The Power amplifier is used to raise signal power to at least 20 dBm. The actual power needed is dependent on the insertion losses from the filters. This amplifier is adjustable before launch to achieve ensure the FM signal has a power of 20 dBm at the antenna.

1.3.6 Output Filter

This filter provides a final attenuation of noise before the signal reaches the antenna. The satellites mission could be jeopardized should too much noise be illegally emitted on frequency bands outside of 144-148 MHz.

1.3.7 Voltage Regulator

The voltage regulator simply serves to provide clean 5 V power independent from any distortion caused by other CubeSAT systems. The most important feature of the voltage regulator is that it can be turned on or off by a digital input from the PIC16 Microcontroller, saving power while the radio is not transmitting.

2 Design

2.1 AFSK Signal Generator

2.1.1 Design Procedure

A microcontroller was chosen to fulfill this role as well as provide system control and power management. Initially, a competing idea was to build a system entirely out of discrete logic. Some kind of VCO with a few logic gates could, in theory, provide the same functionality as a microcontroller. However, due to the extremely rapid advancement in microcontroller technology, it is quite typical for low-power microcontrollers to consume less than a milliwatt of power during operation, and virtually negligible amounts during a standby mode. Using discrete logic gates would drastically add to the circuits power consumption and physical footprint, as well as reduce the design flexibility given by programmability.

The CubeSAT team chooses PIC microcontrollers for use in all of their subsystems. Using microcontrollers of the same type increases the flexibility and reduces code development time. Maintaining a codebase that includes several different types of processors and microcontrollers vastly increases the complexity of the CubeSAT system as a whole.

It was decided that a low power variant of the PIC would be chosen. In addition to power consumption, a critical feature was that the PIC needed to have a Digital to Analog Converter (DAC). The DAC is necessary to generate an AFSK waveform based on the incoming serial data from the CubeSAT main computer. The chip that met these requirements was the PIC16LF1508.

2.1.2 Design Details

The microcontrollers DAC has 5-bit precision. To fully utilize all of the DACs 32 voltage levels, the DACs level needs to change by only one level at a time. At the default clock rate the instruction cycle duration on the PIC is 8 μs [10]. So, suppose a line of C code that pushes a new value to the DAC takes 4 instructions to execute. Only 31 values can be pushed to the DAC in one period of a 1 kHz sine wave.

$$\frac{1 \text{ sample}}{4 \text{ instructions}} \frac{1 \text{ instruction}}{8 \ \mu\text{s}} \frac{1000 \ \mu\text{s}}{1 \text{ period}} = 31.25 \ \frac{\text{samples}}{\text{period}} \tag{1}$$

Pushing the values of a sampled sine wave 31 times per period as fast as possible in C approximated a 1 kHz sine wave. Thus, it can be assumed that it turned a simple DAC assignment in C translates into 4 instructions. The amount of change from one sample to the next is approximated by the derivative of the

sine wave:

$$\frac{d}{dt}32\sin(\frac{2\pi}{31.25}t) = \frac{64\pi}{31.25}\cos(\frac{2\pi}{31.25}t) \approx 6.43 \tag{2}$$

The maximum of this function is 6.43, which means the maximum amount the DAC will change from one sample to the next will be 6 or 7. This means that 5 or 6 values are skipped, reducing the precision of the sine wave by quite a bit when compared to the theoretical maximum.

Next, a Python program (Appendix C.2) was written that generates assembly code which pushes a new value to the DAC with one instruction. The DAC is controlled by the value at a memory location. There is no instruction that can directly move a literal word into a memory location in the PICs instruction set. However, in one instruction the PIC can increment or decrement the DACs value, or take the DACs value and add or subtract it to a general purpose register (set to two). A no-operation instruction also exists. The Python program calculates a sine wave with a peak-to-peak of 32 (the DAC has 5-bit precision), then it decides which of $x-2, x-1, x, x+1, x+2$ is the closest to the actual value of the sine wave, where x is the previous value of the DAC. This calculation generates the best possible approximation as long as the sine waves value never changes by more than 2 in a sample.

$$1\frac{\text{sample}}{\text{instruction}}\frac{1\text{ instruction}}{8\ \mu\text{s}}\frac{1000\ \mu\text{s}}{1\text{ period}} = 125\ \frac{\text{samples}}{\text{period}} \tag{3}$$

The above result is a reasonable amount of samples. However, it is also possible to change the frequency of the PICs internal clock. The clock frequency can go up to 16 MHz without an external oscillator [10]. At this clock speed, it is possible to generate 4000 samples per period, which is more than is needed for the purposes of this project. Running the PIC at such a clock speed, however, would make it possible to use less efficient, but more readable C code.

In the final revision, the python program was still used to generate the array from which the DACs final output values are read. This array was populated to generate a 100 Hz sine wave. Then, by reading every 13th sample, a 1300 Hz wave can be generated, and by reading every 21st sample, a 2100Hz wave can be generated. Using the same lookup table allowed, unlike the other methods of generating a sine wave, switching frequencies at any phase.

A schematic of the PIC wiring is visible in Fig. 10.

2.2 Low Pass Filter

2.2.1 Design Procedure

The AFSK waveform from the PIC was initially to be used directly as the data signal in the first revision. To ensure proper transmission and elimination of signal noise, the DAC waveform was fed through a low pass filter to decrease the chance of miscommunication.

However, the second and third revisions of the project utilized a VCO to perform the frequency modulation. The AFSK signal was now utilized as the control voltage for the VCO. When used in this manner, a noisy AFSK signal poses much less of a problem as the signal is not directly being mixed to higher frequencies, but is used in relative isolation as a control input. With respect to its original purpose, the low pass filter is obsolete. In the second and final revisions, the low pass filter is instead used to change the control voltage scaling and offset. For example, if the control voltage needed to be in a smaller range or offset by a certain value, the low pass filter components can be used to accomplish this task.

2.2.2 Design Details

In the final revision, the low pass filter is used as a simple resistor divider network at the VCO input. Using Ohms law, the transfer function of the low pass filter is a simple voltage divider:

$$V_{\text{out}} = V_{\text{in}}(R_1/(R_1 + R_2)) \tag{4}$$

It was discovered through a series of emails with Fox Electronics that the input impedance of the VCO control voltage pin is very high ($\sim$ 10 MΩ), and thus can be neglected for simple analysis. Using this relationship it is possible to scale the control voltage should the frequency deviation of the VCO output be too high.

2.3 Voltage Controlled Oscillator

2.3.1 Design Procedure

The VCO is the most critical component in the final revision. This module performs the frequency modulation of the baseband control voltage signal and enables the satellite to transmit data to earth.

In the first revision, a double mixing scheme was attempted. This scheme had a fundamental flaw that would make it impossible to receive data from the satellite on earth given the available receiver equipment. It is unfortunate that this mistake slipped past the design review and had to be corrected more than halfway

through the semester. A mixing stage results in the input signal being shifted up. The image (negative frequency) of the signal also appears with mixing, but is filtered out. So, if a "1" is represented by a single frequency coming from the microcontroller, it will still be a single frequency after mixing. This behavior shown in Fig. 2.

While this type of modulation is a valid scheme, what is expected from the modem on the base station is that the carrier will be frequency modulated based on the microcontrollers output. That is, a "1" would modulate the carrier at 1300 Hz and a "0" would modulate it at 2100 Hz. This type of modulation is shown in Fig. 3.

The second revision recognized the above mistake and incorporated a VCO into the design. A VCO circuit enables the carrier freuqency to be modulated directly, which is what the receiver equipment is designed to receive. The VCO was designed using one of the mixers from the first revision, so the actual circuit suffered minimal changes. It was seen that this VCO worked at low frequencies in testing and could have been used effectively to perform the proper frequency modulation. The VCO operated at 10.7 MHz and used a mixer operating at 155 MHz to produce a signal tunable around 144.3 MHz. However, when the second revision PCB was designed, it was discovered that Voltage Controlled Crystal Oscillators (VCXO) are not only produced in single-chip packages, but are available in virtually any frequency and are extremely stable due to their crystal-based nature.

Thus, it was decided that the final revision was to completely eliminate the need for mixers by utilizing an extremely stable VCXO to perform the frequency modulation. The FVXO-HC53 chip from Fox Electronics was chosen to fulfill this role [1]. Available in any transmit frequency of interest, these chips provide extremely large real estate savings while drastically increasing frequency stability and reliability of the system.

2.3.2 Design Details

The VCXO from Fox Electronics was remarkably simple to set up. For the purposes of this project, a frequency in the 144-148 MHz frequency band must be chosen when ordering parts. To integrate the VCXO into the circuit, all that was required for correct operation was to wire the control voltage and output signal to their respective blocks. A schematic of how the VCXO is integrated into the project is visible in Fig. 11. This component allowed the tunability requirement to be met. Due to the simplicity of the final design, all that is needed to change the transmit frequency is to replace the VCXO with another chip operating at the frequency of interest.

2.4 Channel Filter

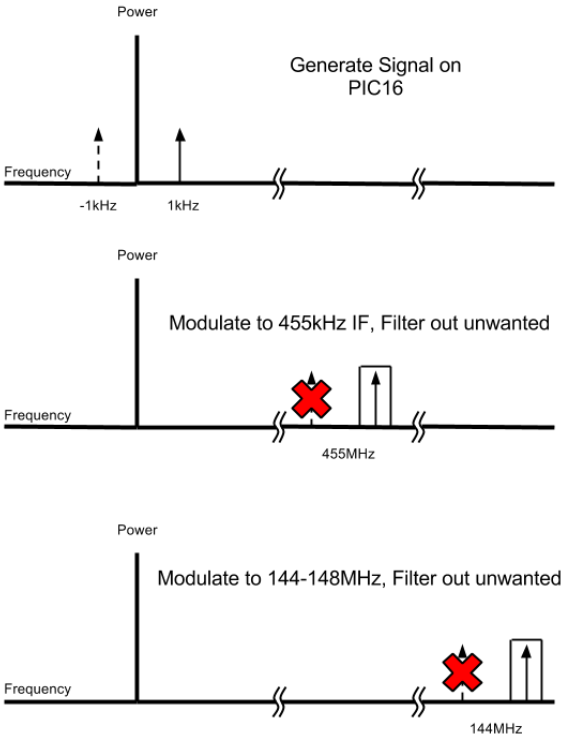

Figure 2: FSK followed by Frequency Mixing Stages

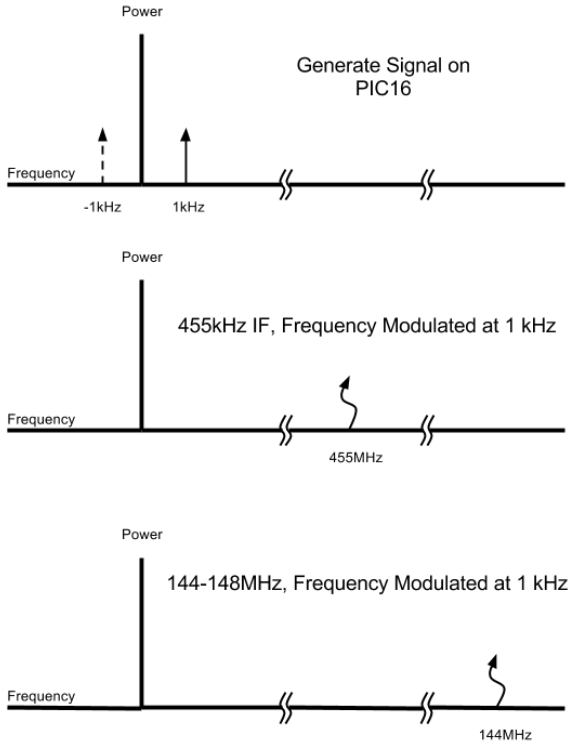

Figure 3: FSK followed by Frequency Modulating and Mixing

2.4.1 Design Procedure

In the first revision, the channel filter was necessary to filter out signal images produced by the mixing stages. A simple Pi/T filter was used to accomplish this task. However, what was difficult about implementing this module was the necessary steepness of the -3 dB rolloff. Since an intermediate frequency (IF) of 455 kHz was chosen in the first revision, it was proposed that this filter would need to attenuate undesired signals by at least -30 dB. Since the signal image would appear approximately 910 kHz away from the desired signal, this meant the filter would have to achieve the -30 dB attenuation in less than 1 MHz, which is an extremely aggressive requirement.

Due to the tradeoff of filter quality and increasing PCB footprint, it was decided that this filter would contain only three orders of LC pairs.

The second revision saw a much more relaxed version of this filter produced. In this revision, since an IF of 10.7 MHz was used, the filter would need a much less aggressive rolloff as the signal image would be

generated more than 20 MHz away from the desired signal.

2.4.2 Design Detail

The channel filter was designed using a Chebyshev bandpass topology. This type of filter uses Chebyshev polynomials to approximate the frequency characteristic of the filter. The critical tradeoff in a Chebyshev filter is increasing sharpness in cutoff for an increase in ripple size. The larger the ripple size, the sharper the cutoff. The Chebyshev polynomials are shown in Eq. 5 [2].

$$C_n(\omega) = 2^{n-1}\left[\omega^n - \frac{n}{1!2^2}\omega^{n-2} + \frac{n(n-3)}{2!2^4}\omega^{n-4} - \frac{n(n-4)(n-5)}{3!2^6}\omega^{n-6} + \frac{n(n-5)(n-6)(n-7)}{4!2^8}\omega^{n-8}...\right] \tag{5}$$

From the filter topology, S_{21} can be found, which is in terms of s, L, and C. which is in terms of s, L and C. These polynomials can then be compared to Eq. 6, in which the polynomial coefficients of Eq. 5, effectively control the transfer function of the filter based on the various L and C values.

$$|S_{21}(j\omega)|^2 = \frac{1}{1+\epsilon^2 C^2 n^{(\omega/\omega_C)}} \tag{6}$$

The filter would then have a frequency characteristic mimicking the function of the Chebyshev Polynomial. ϵ refers to the ripple parameter, which controls the allowed size of the ripple in the filter. For the channel filter, a 2 dB ripple limit was chosen. Since the ultimate goal of this filter was to attenuate the mixer image frequency, the center frequency of this filter was chosen to be 140 MHz. A realistic bandwidth for the filter using the parts available could then be reached. By choosing a 140 MHz center frequency, the ripple parameter was not as crucial of a parameter. The cutoff frequency on the high end was around 150 MHz to allow for some decay by the time a signal reaches 165 MHz. Past this point, the image frequency is introduced and necessarily attenuated by the channel filter. A schematic of this filter is visible in Fig. 11. A fifth order regime was chosen for the channel filter to ensure cutoff at the desired frequencies. The filter also includes an impedance matching network at the input to match the coupling capacitor and impedance from the mixer to 50 Ω. A perfectly matched two-port network will suffer minimal losses at the frequencies of interest. A simulation of the filter with this network included is shown in Fig. 4 and was performed in ADS (Advanced Design Systems).

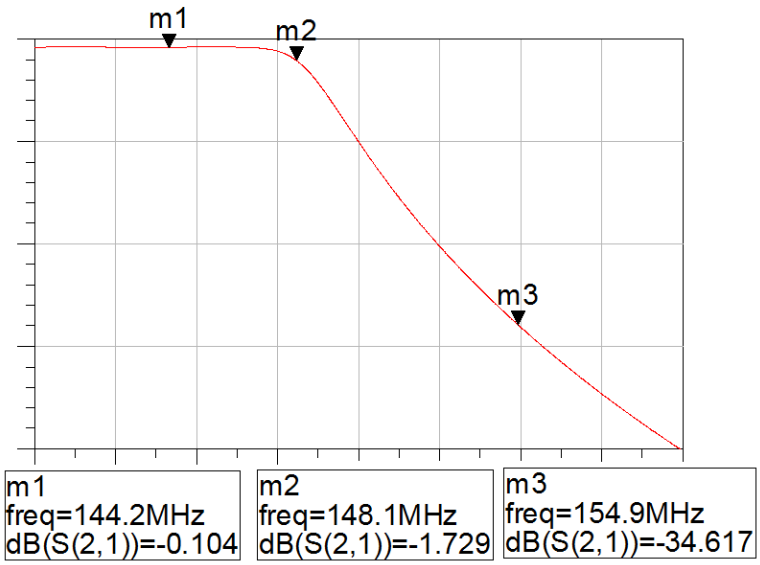

Figure 4: Channel Filter Simulated Output

2.5 Power Amplifier

2.5.1 Design Procedure

In order to actually receive the signal on earth, it was determined by the CubeSAT team that an output power of 20 dBm would be sufficient to communicate with a terrestrial receiver [9]. The challenge with constructing a proper power amplifier is that the power output would necessarily need to be higher than 20 dBm due to insertion losses from previous and subsequent stages. The first revision and second revisions saw major failures with this module. Due to the extremely low powers at which gain compression occurred on the mixers, the power amplifier would have to amplify the signal by unreasonable levels given the space available on the PCB. In the final revision, due to the much higher power output levels from the Fox VCXO, less amplification was needed to meet the 20 dBm output power requirement.

2.5.2 Design Details

The final revision utilizes a cascade of two non-inverting op-amps visible in Fig. 12.. It was decided that this type of amplifier would offer high gain and could be adjusted via replacing resistors should the gain need to be lowered. A reference design from Texas Instruments was used [3]. The gain of an ideal, non-inverting op-amp as seen in fig 5 is given by Eq. 7 [5]:

$$A_v = 1 + R_f/R_1 \tag{7}$$

Choosing $R_f = 200\ \Omega$ and $R_1 = 10.5\ \Omega$, a single amplifier configuration gives a voltage gain of 20.05 dB. Cascading two of these amplifiers in series yields a final theoretical gain of 40.1 dB.

This configuration utilizes a design by Texas Instruments . Due to the high frequency and high power output required (20 dBm), a high slew rate amplifier needed to be chosen. The slew rate of an amplifier is essentially a measure of how fast the output voltage can change. The relationship between slew rate and frequency is given by Eq. 8 [4]:

$$\text{slew rate} = 2\pi f V_{pp} \tag{8}$$

Choosing an output frequency of 148 MHz at the top of the band, and a peak-to-peak voltage of 5 V, the minimum slew rate is calculated to be 4524 $\frac{\text{V}}{\mu\text{s}}$. A high-speed current-feedback amplifier, the Intersil EL5167, was chosen in an attempt to meet this requirement.

The power lines for the two op-amps are tied to the 5 V voltage regulator. When the radio is transmitting, the voltage regulator turns on and thus the op-amps become powered.

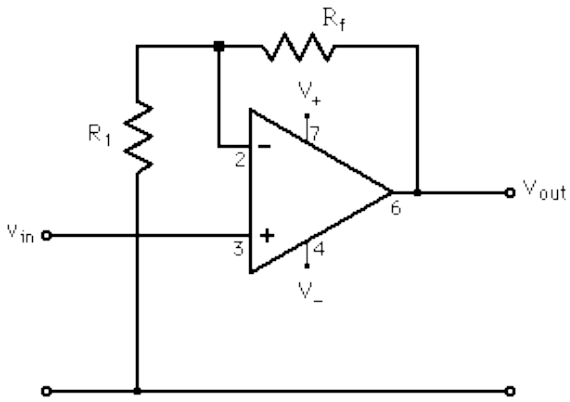

Figure 5: Non-Inverting Amplifier [5]

2.6 Output Filter

2.6.1 Design Procedure

From revision to revision, the output filter maintained its purpose of attenuating noise outside the 2 m band. In addition, the output filter was required to have insertion losses low enough to pass the signal through with minimal attenuation. For all revisions, it remained important to refrain from emitting signals above -30dBc outside the allowed frequency band. The FCC regulates which bands are available under specific licenses, and the amateur band (144-148 MHz, or 2 m) is the only allowed range in which the CubeSAT can transmit

[7]. To meet these requirements, it was decided that the output filter would be another bandpass filter with a bandwidth of about 4 MHz. Due to size constraints, and therefore constraints on the number of orders, the bandwidth was more realistically designed to be around 12 MHz.

2.6.2 Design Details

The output filter was also designed using a Chebyshev Filter. As seen in Fig. 12, this filter contains three orders of L-C stages in a Pi topology. The input impedance was assumed to be 50 Ω from the amplifier stage, and the output impedance to be 50 Ω from the antenna. The ripple size was limited to 2 dB for this filter. The -3 dB bandwidth would ideally be exactly 4 MHz, (148-144). A more realistic bandwidth using the limited Q value parts would be around 12 MHz as shown in Fig. 6 below. Signals at the image frequency and square wave harmonics generated by the Fox VCXO would still be attenuated using the non ideal bandwidth.

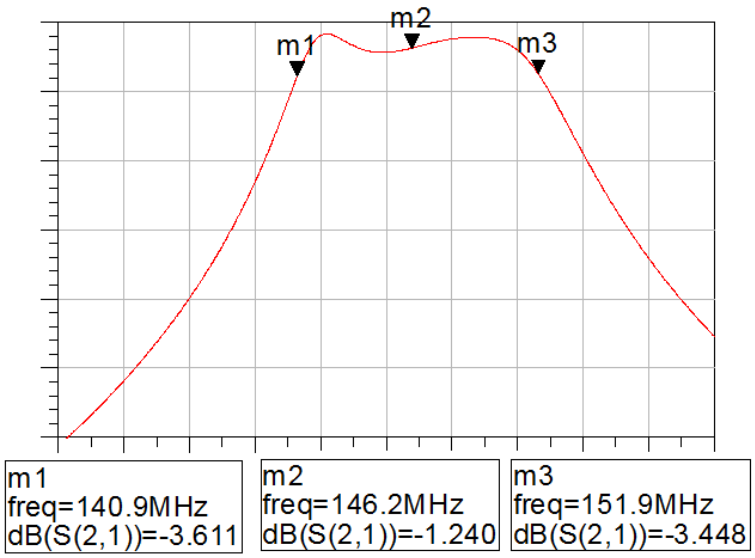

Figure 6: Output Filter Simulated Characteristic

2.7 Voltage Regulator

2.7.1 Design Procedure

The voltage regulator was one of the simpler modules to design. This module serves to regulate power to the amplifiers and turn off when the radio is not transmitting. The type of regulator chosen, the Texas Instruments LP38692 features a digital enable pin which is controlled by the PIC microcontroller [6]. It can supply 1 A of current which is much more than the radio would ever need. This regulator remained

unchanged through all design revisions.

2.7.2 Design Details

The voltage regulator can supply a maximum of 1 A @ 5 V output, which translates to 5 W of output power. The PIC directly interfaces with the control voltage PIN to enable the amplifiers when transmitting, which is visible in the schematic of Fig. 10.

3 Verification

3.1 PIC Microcontroller

The microcontroller does three things: communicate with the satellites CPU, generate AFSK tones, and control the power to the rest of the circuit. Each of these tasks can be tested and verified separately.

3.1.1 Testing Procedure

Communication with CubeSAT CPU uses the SPI protocol. This functionality was tested by loading a test program that repeats the message back to the CPU. If the messages match, then communication is working.

The AFSK can be tested by loading a program that continuously outputs a sequence of bits. Then, the output is connected to an oscilloscope. It should show a 476 μs period sine wave for the zeros in the bit sequence and a 796 μs period sine wave for the ones.

The power control can be tested with an LED. There is an LED attached to the voltage control pin. Loading a program that turns the voltage control on and off should also toggle the LED.

3.1.2 Results

The PIC functions exactly as intended. Serial communications were successfully transmitted to and from the beacon. Power control functionality was seen to work correctly via the LED. As shown in Fig. 7, the AFSK signal was correctly generated.

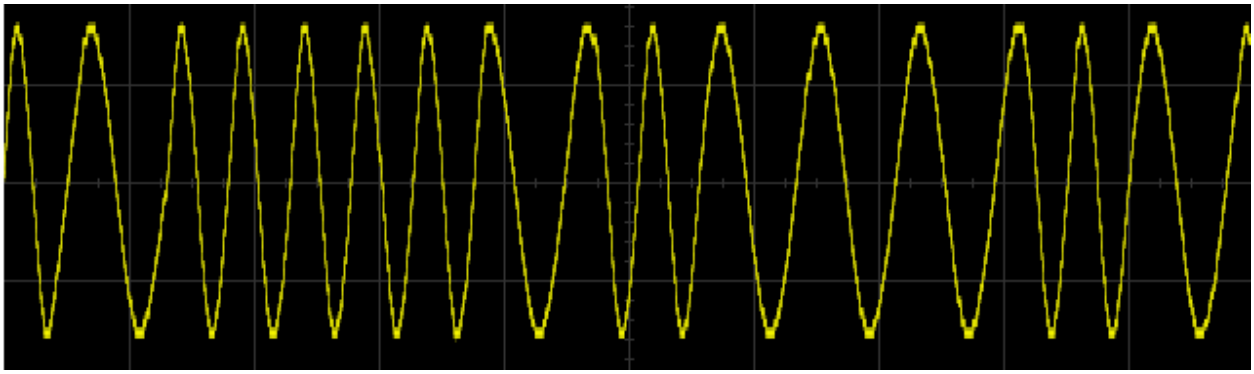

Figure 7: PIC FSK output

3.2 VCO

3.2.1 Testing Procedure

The final testing on this module was essentially two tests, the first of which was to simply power the chip and check that it is actually generating a frequency in the range declared by the datasheet. The next test was to input a control voltage with a signal generator and view the output frequency being modulated on an oscilloscope. To test compatibility with the PIC, a test jig was constructed that fed the AFSK signal to the Fox VCXO chip on a vectorboard.

3.2.2 Results

The VCO functioned exactly as expected on the datasheet. A control voltage input can effectively modulate the carrier frequency with a frequency deviation of about 20 kHz over the full 3.3 V range.

3.3 Channel Filter

3.3.1 Testing Procedure

The channel filter was tested by connecting a signal generator to the input and a spectrum analyzer to the output. By sweeping a known signal into the input and viewing the output at the spectrum analyzer, the attenuation at key frequencies can be found. For the channel filter, the important frequencies of interest are 144-148 MHz, 155 and 165 MHz. A leaky oscillator may pose for leakage around 155 MHz and the image frequency from the second revision was expected to lie at 165 MHz. The filter is designed to be a bandpass filter, so as a sanity check a frequency of 130 MHz was also measured. The testing procedure above also reveals the insertion loss of the filter. Also, a sweep was performed from 100 MHz to 200 MHz to confirm the general shape of the filter.

3.3.2 Results

The channel filter results are shown in Fig. 8. Looking at the plot below, it can be seen that around 144 and 148 MHz an insertion loss of around -12 dB was measured. At 155 MHz, about -15 dB of loss was measured and at 165 MHz about -58 dB was measured. At 130 MHz, about -10 dB of loss was measured. Referring to the Appendix B, both requirements had failed.

The insertion loss was greater than the simulated -10 dB. The biggest contributor to this increase was an impedance mismatch. The impedance of the signal generator is 50 Ω, whereas the filter was designed to have the mixer of the second revision as an input. Thus, the filter was designed for an input impedance

1500 Ω. However, In the final revision of the transmitter, a VCXO chip was changed from the mixer at the last minute, resulting in a large impedance mismatch and thus increased insertion losses. The bandwidth of the filter is closer to 50 MHz rather than the desired 20 MHz. This error is most likely due to the chosen parts having non-exact values and lower Quality factor (Q) than initially planned. The filter Q has almost a direct correlation to the bandwidth of the filter and is itself dependent on the individual Q values of the various components. In addition, it was not considered that each part would, in reality, be of non-ideal value due to manufacturing tolerances. Neglecting to take these two considerations into account is the reason the bandwidth of the real filter is much larger than the simulated filter. However, frequency sweep did at least mimic the general shape of the simulated filter result shown in the Fig. 4.

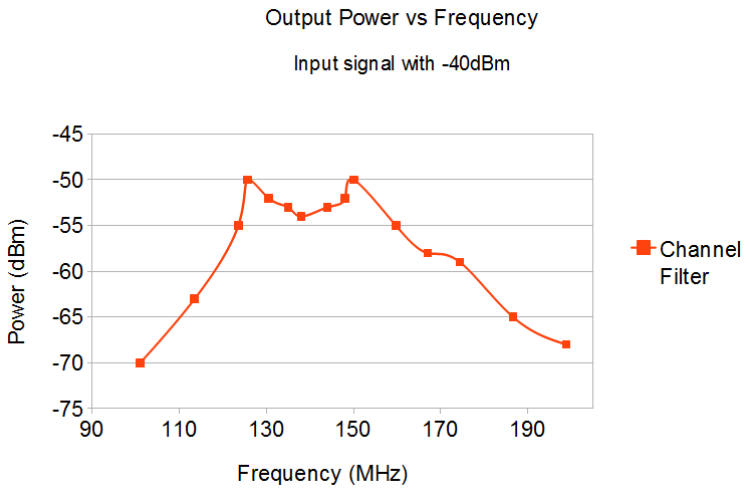

Figure 8: Measured Response of the Channel Filter

3.4 Amplifier

3.4.1 Testing Procedure

Testing the amplifier entailed first checking whether or not the amplifiers were powered and had correct feedback resistor settings. A signal generator was hooked to the input, while an oscilloscope was inserted at the output. A 150 MHz signal was generated, and it was expected to see an output waveform with an amplitude of 20 dBV greater than the input.

3.4.2 Results

Initially, nothing worked. The output did not respond to the input, even when the opamp was drawing current and the feedback resistors were confirmed to be correct. During an attempt to resolder all parts in

the op-amp circuit, it was realized that the land pattern positioned under the op-amps was much too large. Due to the small size of the op-amps, it was deemed near impossible to solder them correctly after many attempts. By the time it was realized that a PCB error had been made, it was too late to obtain op-amps of similar land pattern. The only data collected from this type of amplifier was that the on power was about 800 mW, which was much higher than our initial goal of 250 mW. The requirements and verifications table used to test this module is in Appendix B.

3.5 Output Filter

3.5.1 Testing Procedure

The output filter was tested in a similar fashion to the channel filter. An input signal was applied to the input of the filter and the output was connected to a spectrum analyzer. The filter was tested at points including 144,148, 118, 130 and 165 MHz. In the ideal case, all but the 144 and 148 MHz frequencies would be attenuated by more than 30 dB.

3.5.2 Results

Referring to the plot in Fig. 9, the output filter also failed both of its requirements outlined in Appendix B. The insertion loss at the desired frequencies was found to be around -17 dB and the bandwidth looks to be extended to around 40 MHz. The desired insertion loss was less than -10 dB and the bandwidth was simulated to be around 12 MHz. This filter failed the same verifications as the channel filter for nearly identical reasons. The parts of the the filter had lower Q values than what would be required for a sharp bandwidth cutoff and also had tolerances. Through simulation, even a 0.1 pF change in the capacitors is seen to have a significant change in the frequency characteristic. This filter could be improved by including higher orders, higher Q parts, and by purchasing multiple parts at or near the desired value. Also, a more careful analysis of the input and output impedances could improve the insertion loss.

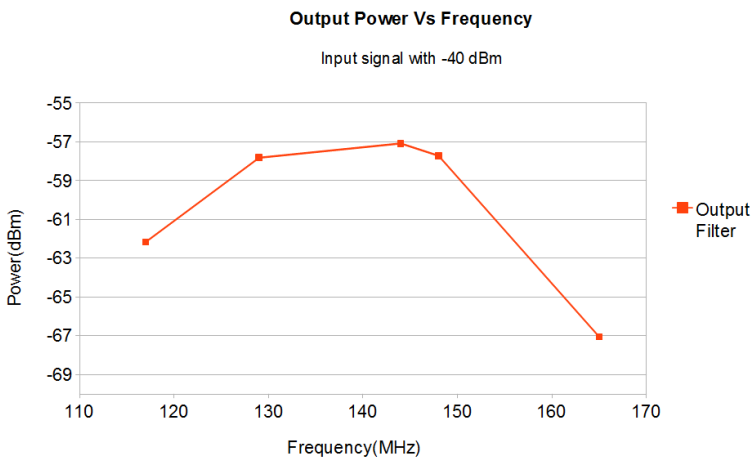

Figure 9: Measured Response of the Output Filter

3.6 Voltage Regulator

3.6.1 Testing Procedure

On each respective PCB, the voltage regulator was soldered and directly connected to a 7.4 V voltage source. When a digital input greater than 3 V was emitted from the PIC, the voltage regulator was expected to turn on and output 5 V. Next, to ensure stability, the input voltage to the regulator was varied over its full operating range of 5-10 V [6].

3.6.2 Results

The voltage regulator indeed regulated voltage sufficiently. The device operated with an input voltage ranging from 5 V to 10 V, with the only visible ripple and instability introduced from the power supply itself. The PIC was also successfully able to activate the regulator during transmission, even with an enable voltage of less than 2 V.

4 Costs

4.1 List of parts and equipment needed

The parts used to determine the cost of constructing a complete radio beacon amount to one complete unit. This device is not intended for volume production, so low order quantity pricing is used. A complete parts table is in Appendix F. The development cost of the overall system is estimated at $62687.95, with the materials costs amounting to $187.95.

5 Conclusions

5.1 Accomplishments

The radio beacon was indeed seen to transmit across the lab, even with the onboard amplifier malfunctioning. One computer was configured to mimic the satellite and fed data to the beacon via a serial interface. A radio receiver was connected to a modem which fed data into a different computer. When the system was activated, data typed on the first computer was directly displayed on the second. While certain aspects such as filtering and power amplification did not satisfactorily meet the design requirements, this project is considered successful.

5.2 Uncertainties

It is not known whether or not that both the channel filter and output filter are necessary for a solid design; only one of these filters is theoretically necessary. Additional testing is required to determine which filter is appropriate to eliminate. Another uncertainty is whether or not all the parts chosen will work in space. To address this, appropriate environmental testing needs to be performed to verify the frequency stability and durability of the radio.

5.3 Ethical considerations

It would be of poor ethics to design a radio that would generate too much out-of-band-noise. Designing with disregard or concealing that our radio would transmit with wide of a bandwidth or on illegal frequencies could cause disastrous financial consequences to the CubeSAT organization. According to the Code of Federal Regulations, §97.113, No amateur station shall...engage in any form of broadcasting [7]. Should the satellite launch with a revision of the radio that does not sufficiently attenuate out-of-band noise (VCO square wave harmonics, Local Oscillators, etc.), there is a very real possibility that a fine could be levied against CubeSAT on behalf of the FCC. Stringent final testing would necessarily be performed to ensure that noise is sufficiently suppressed.

5.4 Future work / Alternatives

Although requirements for this project were not met, there is no denying that the CubeSAT team still has a need for this project to be continued. Without a backup transmitter, the satellite will have no way of relaying status data should the main radio circuit encounter any problems. Even though the final revision that was produced is far from an ideal transmitter, it would only take one or two more moderate revisions

to construct a truly spaceworthy design. It has been shown that given a team with more time and adequate RF knowledge, this project will provide as a working base design for the final transmitter.

In the immediate future, a new revision of the PCB needs to be constructed. Due to the vast decrease in circuit footprint, this revision will feature a much more streamlined layout. The amplifiers may need to be changed as they have not been tested fully. It is recommended that any new amplifiers or circuits be completely characterized before they are placed on a new PCB.

A few options could be implemented to adjust the filter performance. The most immediate would be to purchase higher Q components, which may require larger package sizes and to implement them in place of the current filter. When designing these filters, higher orders, impedance matching and tolerances of parts must be taken into consideration to meet design requirements. An alternative to building LC filters is to switch to a single-chip solution such as a Surface Acoustic Wave (SAW) filter. The typical insertion loss of a SAW filter is between 9 to 15 dB [8]. Although the insertion loss is undesirable, the cutoffs are very sharp and they are fairly common filters in RF design.

Power regulation and management can be vastly simplified as well. Instead of using a 5 V regulator, the 3.3 V and 7.4 V sources can be used directly from the CubeSAT connection. All that would be added to reduce power consumption would be one or two load switches to turn off each respective supply voltage. Alternately, powered components could be chosen to have enable pins directly controlled by the PIC. A sample future PCB part placement is visible in Appendix 18.

References

[1] Fox Electronics. (2008). *FVXO HC53* [Online]. Available: http://www.foxonline.com/pdfs/FVXO_HC53.pdf

[2] S. J. Franke, "ECE 435: Wireless Communication Systems Course Notes," Dept. Elect. and Comput. Eng., Univ. of Illinois, Urbana-Champaign, 2012.

[3] B. Carter (2003). *RF and IF amplifiers with op amps* [Online]. Available: http://www.ti.com/lit/an/slyt102/slyt102.pdf

[4] *Op-amp Slew Rate* [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/electronic/a741p3.html

[5] *Non-inverting Amplifier* [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/electronic/opampvar.htm#c3

[6] Texas Instruments (2013). *LP38690* [Online]. Available: http://www.ti.com/lit/ds/snvs322j/snvs322j.pdf

[7] *Prohibited Transmissions*, Code of Federal Regulations 47 CFR 97.113, 2010 Available:

[8] Sawtek Inc (1999). *Std Low-Loss 140 MHz Bandpass Filter* [Online]. Available: www.triquint.com/products/d/DOC-A-00000445

[9] K. P. Basset, "Path Calcs Spreadsheet," unpublished.

[10] Microchip Technology, Inc. (2011, Oct.). *PIC16(L)F1508 Data Sheet* [Online]. Available: http://datasheet.octopart.com/PIC16F1508-I/SS-Microchip-datasheet-10922349.pdf

A Final Revision Schematics

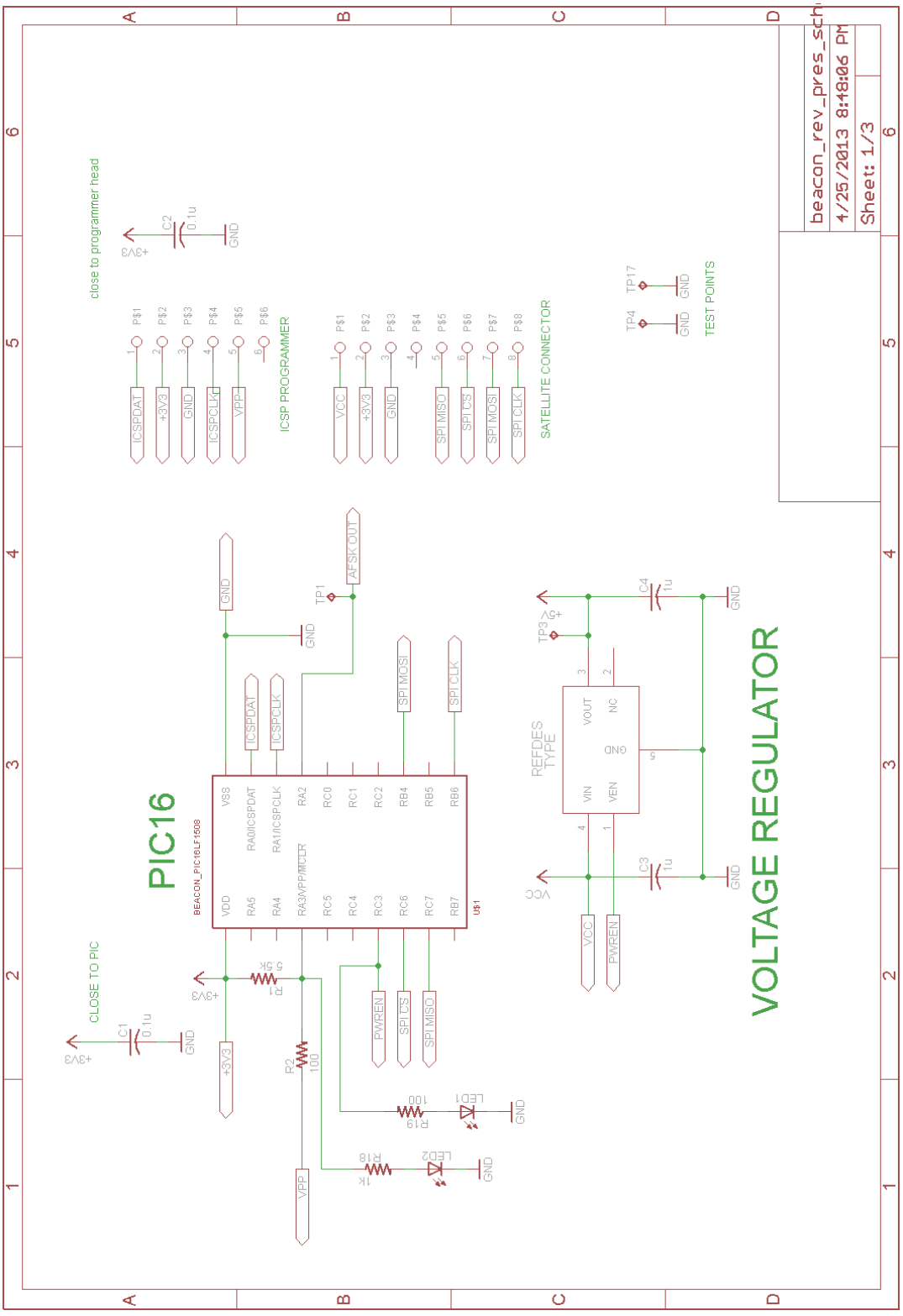

Figure 10: Final Revision Schematic (page 1). This page details the wiring of the PIC and the AFSK signal origin. Debugging LEDs are included on the PIC, as well as a connection to the voltage regulator. In addition, the satellite and programming connectors are shown.

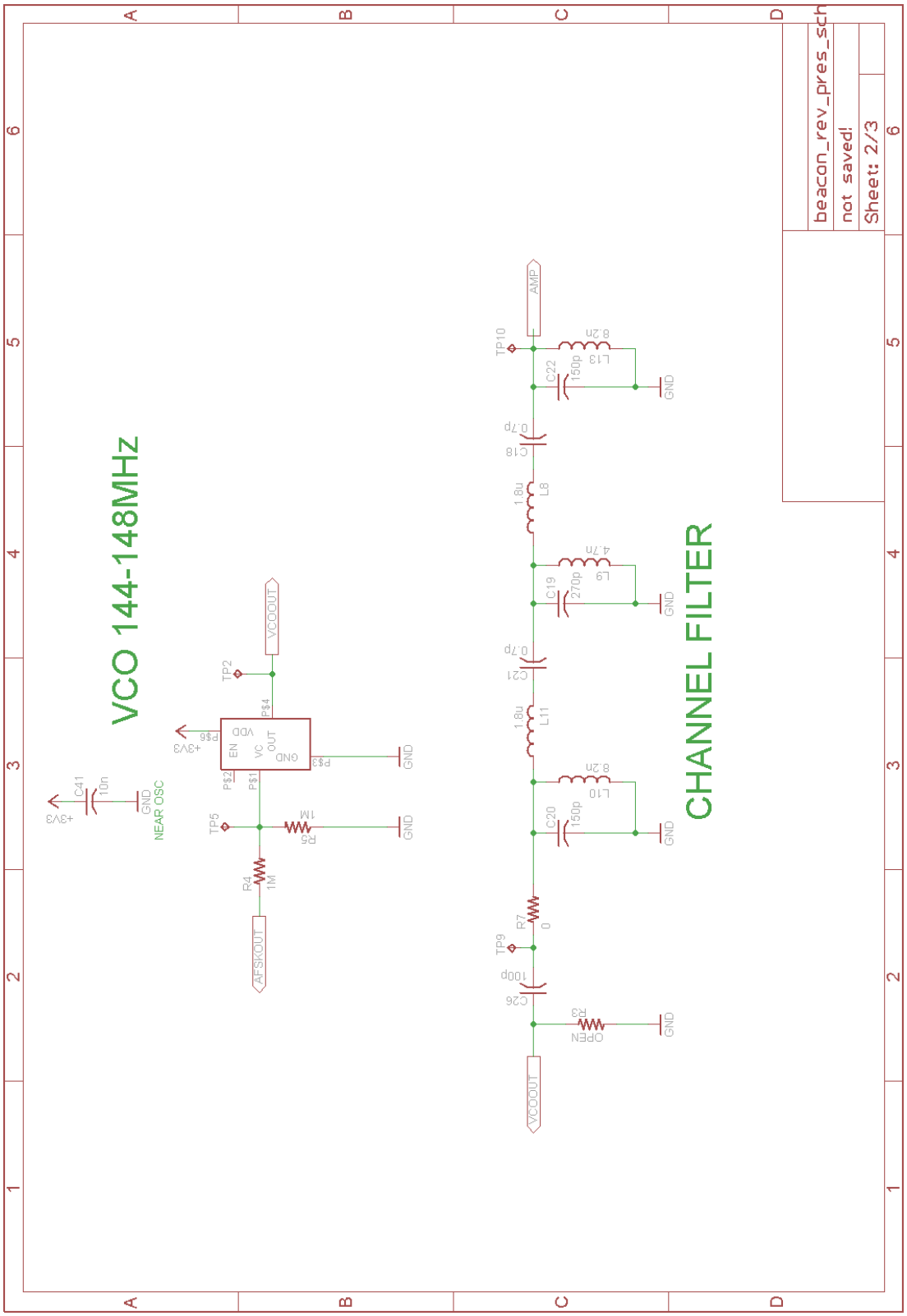

Figure 11: Final Revision Schematic (page 2). The VCO from Fox Electronics and the channel filter are depicted. The remnants of the low pass filter have been configured into a voltage divider at the control voltage input.

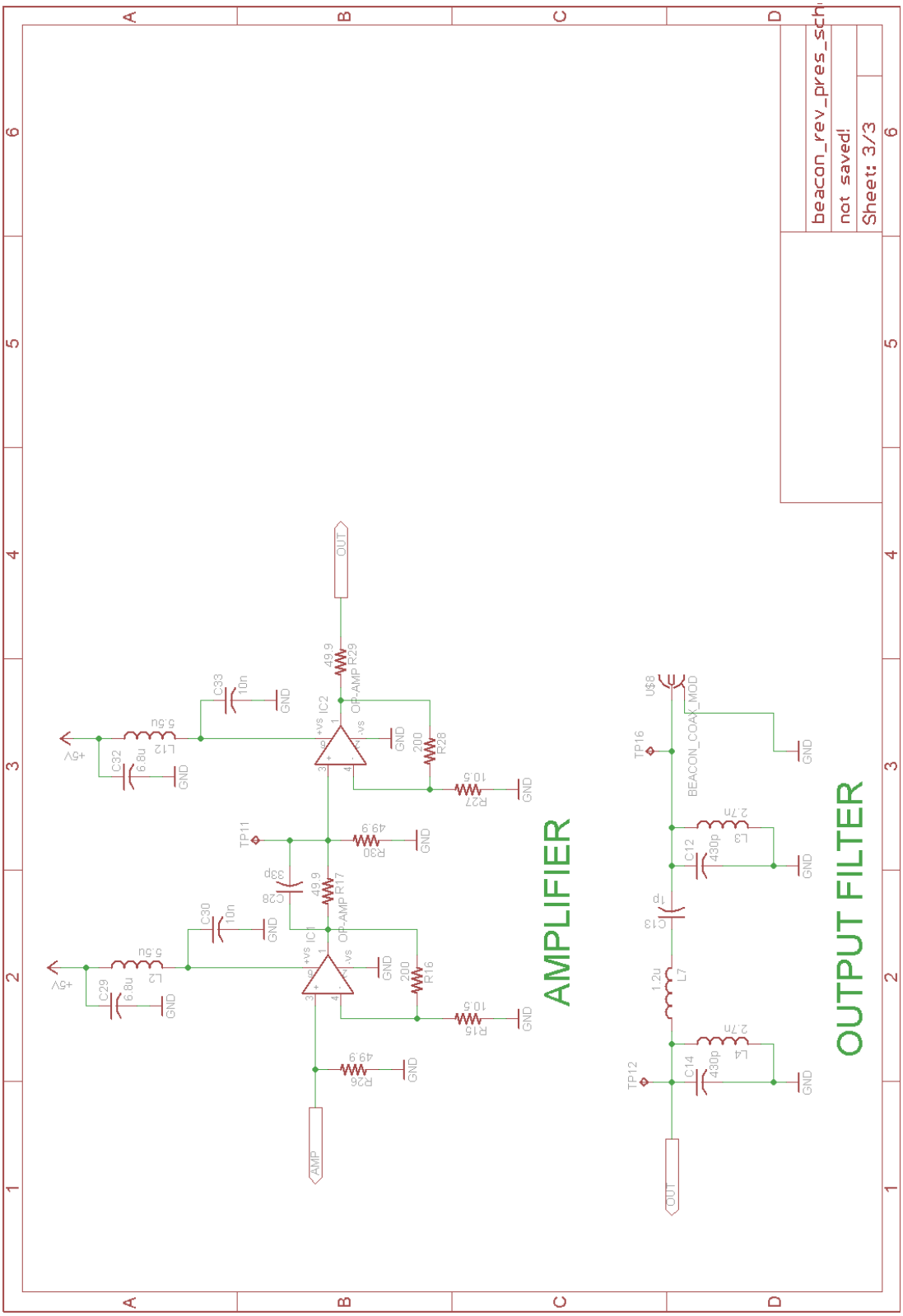

Figure 12: Final Revision Schematic (page 3). The Amplifier stage and output filter are detailed. Heavy decoupling is employed on the +5V rail into the op-amps.

B Requirements and Verifications for Final Revision

Microcontroller Requirements	Verifications
1. Microcontroller receives 3.2-3.6 Volts	1. Look at blue LED. If it is lit, the PIC is powered.
2. The PIC is programmed with the latest version of the code a. The ICD is connected to the computer b. The ICD is connected to the circuit c. Latest program is loaded onto the PIC	2. Open the latest version of the code in MPLAB X IDE. Then, press Program. a. MPLAB does not ask you what device you want to use to program the PIC. b. The device ID read by MPLAB is not 0x0. c. MPLAB outputs successfully programmed.
3. Two-Way communication with cubeSAT a. Receives data over the SPI data bus b. Sends an Acknowledge byte back to the cubeSAT CPU after finishing sending its data	3. a. Attach the SPI data bus to a Bus Pirate. Attach the Bus Pirate to a computer. Open a serial communications program and connect to Bus Pirate. Send m to open the menu. Select 5, which is SPI. Then, send a string in quotes. The PIC should reply with the index of the byte being read. b. Connect to the SPI to a Bus Pirate as described above. Send a string of at least 80 bytes. Then send r to read the reply, it should be 0xAA.
4. Generate AFSK signal a. Generates a 1.3 ± 0.01 kHz signal for zero and a 2.1 ± 0.01 kHz signal for one for every bit in the 80 bytes in memory corresponding to the data packet	4. a. Load the test program that already has a packet in memory which is "start—alphanumabcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUV0123456789end. Connect pin 17 to an oscilloscope. The output should be 2.1 ± 0.01 kHz for 16 cycles, 1.3 ± 0.01 kHz for 4 cycles, 2.1 ± 0.01 kHz for 4 cycles, and 1.3 ± 0.01 kHz for 2 cycles, repeated continuously.
5. Voltage Regulator is controllable	5.

a. Voltage Regulator is off when not transmitting.	a. Do not send any communication over the SPI bus. Look at red LED. If it is off, the voltage regulator is turned off.
b. Voltage Regulator turns on when transmitting.	b. Look at red LED. If it is lit, the voltage regulator is turned on.
6. PIC and LEDs consume less than 10 mW of power.	6. Read the current and voltage from the power supply when operating. Verify that the $I \times V <$ 10 mW.
VCO Requirements	**Verifications**
1. VCO generates a frequency modulated waveform centered around F_0, where F_0 is the center frequency of the VCO	1. Connect signal generator at input of filter, and spectrum analyzer at output of filter. Set signal generator to F_0 at 0 dBm. Verify output signal is a $F_0 \pm 0.05$ MHz with magnitude between 0 and -10 dBm.
2. Signal outside of band are attenuated by 30 dB.	2. Connect signal generator at input of filter and spectrum analyzer to the output.
a. Signals below 130 MHz.	a. Set signal generator to 130 MHz at 0 dBm. Verify output signal is a 130 ± 0.05 MHz with magnitude less than -30 dBm.
b. Signals above 155 MHz	b. Set signal generator to 155 MHz at 0 dBm. Verify output signal is a 155 ± 0.05 MHz with magnitude less than -30 dBm.
Amplifier Requirements	**Verification**
1. Amplifies the input signal by more than 40 dB.	1. Connect a function generator between the input of the amplifier and GND. Generate a 146 MHz sine wave with peak-to-peak voltage of -40 dBm. Connect an oscilloscope with an attenuator between the output of the amplifier and GND.Verify the signal output is at least 20 dBm.
a. Amplifier is powered	a. With a voltmeter, verify that the amplifier is receiving 5.0 ± 0.1 V between V_{cc} and GND.

b. Input and output impedance are 50 Ω. c. Op-amp is configured to provide a gain of at least 20.	b. With an ohmmeter, verify that the output impedance of the previous stage is 50 ± 10 Ω. c. With a multimeter, measure the feedback and inverting-input resistors. Each resistor should measure 200 and 10.5 Ω, respectively.
2. The power consumption of the amplifier is 190 mW when amplifying a -20 dBm waveform.	2. Connect a function generator to the input of the amplifier. Generate a 144 MHz signal with a power of -20 dBm. Read the current and voltage from the power supply when powering the amplifier with 5 Volts. Verify that the $I \times V < 190$ mW.
Output Filter Requirements	**Verification**
1. Successfully passes desired signal F_0 with less than 10 dB of insertion loss.	1. Connect signal generator at input of filter, and spectrum analyzer at output of filter. Set signal generator to F_0 at 0!dBm. Verify output signal is a $F_0 \pm 0.05$ MHz with magnitude between 0 and -10 dBm.
2. Attenuates signals outside of 144-148 ± 4 MHz by 20 dB a. Frequency of 140 MHz b. Frequency of 152 MHz	2. a. Connect signal generator at input of filter, and spectrum analyzer at output of filter. Set signal generator to 140 MHz at 0 dBm. Verify output signal is a 140 ± 0.05 MHz with magnitude less than -20 dBm. b. Connect signal generator at input of filter, and spectrum analyzer at output of filter. Set signal generator to 152 MHz at 0 dBm. Verify output signal is a 152 ± 0.05 MHz with magnitude less than -20 dBm.
Voltage Regulator Requirements	**Verification**
1. The voltage regulator is enabled by the enable pin.	1. Power the regulator with 6-9 V

a. The voltage regulator turns off when enable is low b. The voltage regulator turns on when enable is high	a. Ground the enable pin, it should output less than 0.1 V. b. Use a power supply to send 3.3 V to the enable pin of the regulator. It should output 5 ± 0.1 V.
2. Outputs a 5 ± 0.1 V when a 6-9 V voltage is applied.	2. Attach the input of the regulator to a power supply. Attach the output to a voltmeter. Move the power supplys voltage from 6 volts to 9 volts and back again. The output should stay within 0.1 volts of 5 V.
3. Voltage regulator consumes less than 1 mW.	3. Read the current and voltage from the power supply when powered with 7.4 V. Verify that the $I \times V < 1$ mW.
Physical Constraints for Satellite	**Verification**
1. The board should be 80 mm $\times$ 25 mm at most.	1. Measure with a caliper, it should measure at most 80 mm in one dimension, and at most 25 mm the other.
2. There should be 6 M2.5 mounting screw holes at (from the bottom left): -x=4mm, y=4mm -x=4mm, y=21mm -x=40mm, y=4mm -x=40mm, y=21mm -x=76mm, y=4mm -x=76mm, y=21mm	2. Place an M2.5 screw in each hole, verify that they fit in the hole. Use a caliper to measure their position from the bottom left corner of the board.

C PIC Code

C.1 LUT Generating Code (Python)

```
prefix = """



;/*Generated by code_gen.py ((C) 2013 Russell Jones)
;  For the University of Illinois cubeSAT project */



"""



from math import *



FREQUENCY = 100
CLK = 16000000



f = open("generated.arr","w")
print(prefix+"""
/* This is the slowest method, reading an array and putting it into the DAC takes
 * 42 instructions, which is 168 clock cycles
 * This is with the free XC8 compiler, another compiler might do better.
 */
""",file=f)
period = CLK/(168*FREQUENCY) #assuming 168 clock cycles per lookup
print("sine[" + str(int(period)) + "] = {",end="",file=f)
for i in range(0,int(period)):
    if (i != 0):
            print(", ",end="",file=f)
```

```
    dacout = int(round(15.5*(sin(2*pi/period*i)+1)))
    print(hex(dacout),end="",file=f)
print("};",file=f)
f.close()
```

C.2 Main Program

```
/*****************************************************************************/
/* Files to Include                                                          */
/*****************************************************************************/


#if defined(__XC)
        #include <xc.h>             /* XC8 General Include File */
#elif defined(HI_TECH_C)
        #include <htc.h>            /* HiTech General Include File */
#endif


#include <stdint.h>            /* For uint8_t definition */
#include <stdbool.h>           /* For true/false definition */


#include "system.h"            /* System funct/params, like osc/peripheral config */
#include "user.h"              /* User funct/params, such as InitApp */


/*****************************************************************************/
/* User Global Variable Declaration                                          */
/*****************************************************************************/
/* Test packet is just 0x47 repeated 80 times*/
char packet[80] = "start|alphanum:abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789end";
        char rep1k, rep2k;
```

```
/************************************************************************/
/* Main Program                                                         */
/************************************************************************/
void main(void)
{
        /* Configure the oscillator for the device */
        ConfigureOscillator();


        /* Initialize I/O and Peripherals for application */
        InitPorts();


        InitDAC();


        InitSPI();


        while(1) {
            get_packet(packet);
            power_on();
            transmit(packet);
            power_off();
            spi_rw(0xAA);
            spi_reset();
        }
```

```
}
```

C.3 User Functions Program

```
/******************************************************************************/
/* Files to Include                                                           */
/******************************************************************************/


#if defined(__XC)
        #include <xc.h>             /* XC8 General Include File */
#elif defined(HI_TECH_C)
        #include <htc.h>            /* HiTech General Include File */
#endif


#include <stdint.h>             /* For uint8_t definition */
#include <stdbool.h>            /* For true/false definition */


#include "user.h"


/******************************************************************************/
/* User Functions                                                             */
/******************************************************************************/


/* <Initialize variables in user.h and insert code for user algorithms.> */


void InitPorts(void)
{
```

```
        /* All ports are initialized to be output ports outputting 0*/
        /*Clear the port outputs*/
        PORTA = PORTB = PORTC = 0;



        /*Clear the port latches*/
        LATA = LATB = LATC = 0;


        /* No analog input */
        ANSELA = ANSELB = ANSELC = 0;


        /* Make all pins outputs except MCLR */
        TRISA = TRISB = TRISC = 0;


        #if (POWER_ALWAYS_ON)
            PORTC |= 0x08;
        #endif
}


void InitDAC(void)
{
        /* Turn on DAC on RA2 */
        DACCON0 = 0b10010000;
}


void InitSPI(void)
{
```

```
        /* Make SDI/SPI_MOSI (RB4) and SCK/SPI_CLK (RB6) inputs */
        TRISB |= 0b01010000;


        /* Make SS (RC6) an input pin*/
        TRISC |= 0b01000000;


        /* Make SDO/SPI_MISO (RC7) an output pin*/
        TRISC &= 0b01111111;


        /* SSPSTAT<6> = CPHA; 1 is active to idle */
        SSPSTAT = 0b01000000;


        /* SSPCON1<3:0> = 0100 for SPI slave mode SS controlled */
        /* SSPCON1<4> = CPOL; 0 is clock idle low */
        SSPCON1 = 0b00000101;


        /* Disable Buffer Overwrite */
        /* SSPCON3<4> = 0 */
        SSPCON3 = 0b00000000;


        /* SSPCON1<5> = 1 to enable synchronous serial communication */
        SSPCON1 |= 0b00100000;
}


void SPIEnableInterrupts(void)
{
```

```
        /* Bit 7 is interrupts enabled (general) */
        /* Bit 6 is peripheral interrupts enabled */
        INTCON = 0b11000000;


        /* Enable SSP interrupts */
        PIE1 = 0b00001000;
}


int spi_rw(int wdata)
{
        SSPBUF = wdata;
        while (!(SSPSTAT&0x01));
        int rdata = SSPBUF;
        return rdata;
}


void spi_reset(void)
{
        SSPCON1 &= 0b11011111;
        SSPCON1 |= 0b00100000;
}


void get_packet(char *packet)
{
        char last = 0;
        unsigned char i = 0;
        while (i < 80) {
            while (PORTC&0x40) {
                i = 0;
```

```
            }
            last = packet[i++] = spi_rw(i);
        }
}



void power_on(void)
{
        PORTC |= 0x08;
}



void power_off(void)
{
        #if (!POWER_ALWAYS_ON)
            PORTC &= 0xF7;
        #endif
}




unsigned lut_index = 0;
#define REPS 79
#define SAMPLES 952
/* 100 Hz LUT*/
const unsigned char sine[SAMPLES] = {0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x11, 0
void sine1_3(void)
{
        for (int ii = 0; ii < REPS; ii++) {
            DACCON1 = sine[lut_index];
            lut_index += 13; /*13 hundred Hz*/
            if (lut_index > SAMPLES) {
```

```
            lut_index -= SAMPLES;
        }
    }
}


void sine2_1(void)
{
    for (int ii = 0; ii < REPS; ii++) {
        DACCON1 = sine[lut_index];
        lut_index += 21; /*21 hundred Hz*/
        if (lut_index > SAMPLES) {
            lut_index -= SAMPLES;
        }
    }
}


void transmit_byte(char byte)
{
    for (char i = 0x01; i != 0; i <<= 1) {
        if (i&byte) {
            sine1_3();
        } else {
            sine2_1();
        }
    }
}


void transmit_serial(char byte)
{
    sine2_1();
```

```
        transmit_byte(byte);
        sine1_3();
}
#if AX25
unsigned char ax25_ones = 0;


void transmit_ax25(char byte)
{
        for (char i = 0x01; i != 0; i <<= 1) {
            if (i&byte) {
                sine1_3();
                if (++ax25_ones == 5) {
                    sine2_1();
                    ax25_ones = 0;
                }
            } else {
                sine2_1();
            }
        }
}
#endif //AX25


void zeros(char n)
{
        for (char i = 0; i < n; i++) {
            sine2_1();
        }
}


void ones(char n)
```

```
{
        for (char i = 0; i < n; i++) {
            sine1_3();
        }
}


unsigned char packet_header[16];


void transmit(char *packet)
{
#if AX25
        for (unsigned char i = 0; i < 64; i++) {
            packet[i] = packet[i+16];
        }
        packet[0] = 'B';
        packet[1] = 'a';
        packet[2] = 's';
        packet[3] = 'e';
        packet[4] = ' ';
        packet[5] = ' ';
        packet[6] = 0xe0;
        packet[7] = 'C';
        packet[8] = 'u';
        packet[9] = 'b';
        packet[10] = 'S';
        packet[11] = 'A';
        packet[12] = 'T';
        packet[13] = 0x61;
        packet[14] = 0x3e;
        packet[15] = 0xf0;
        unsigned short crc_res = CRCCCITT(packet,80,0xffff,0x0000);
```

```
        /*AX.25 Header*/
        transmit_byte(0x7d);          //Flag


        for (char i = 0; i < 80; i++)
            transmit_ax25(packet[i]);




        /*AX.25 Footer*/
        transmit_ax25((unsigned char)(crc_res << 8)); //FCS 1
        transmit_ax25((unsigned char) crc_res);           //FCS 2
        transmit_byte(0x7d);                              //Flag
#elif SERIAL
        ones(10);
        for (char i = 0; i < 80; i++)
            transmit_serial(packet[i]);
#else
   for (char i = 0; i < 80; i++)
            transmit_byte(packet[i]);
#endif
}
```

D Final Revision PCB layout

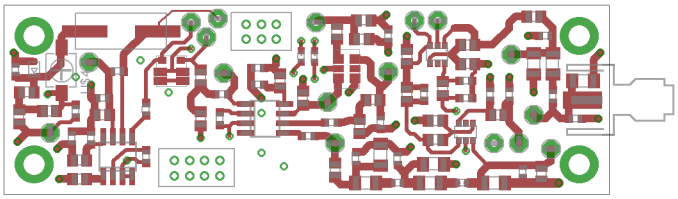

Figure 13: Top Layer of the Final Revision PCB.

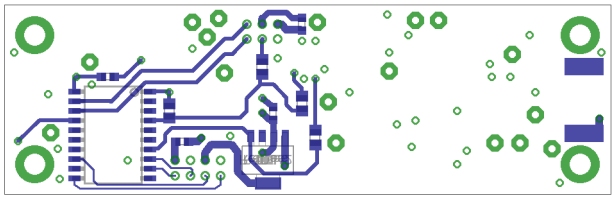

Figure 14: Bottom Layer of the Final Revision PCB.

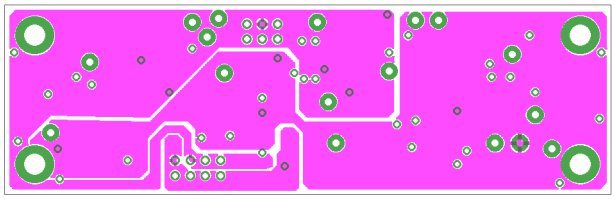

Figure 15: Power Layer of the Final PCB. This layer has been split into a 5V plane (right section), 7.4V plane at the power entry (the U shaped section around the 8 pin connector) and a 3V3 section filling the remainder of the plane. Continuous RF ground plane is not shown.

E Past Revision Block Diagrams

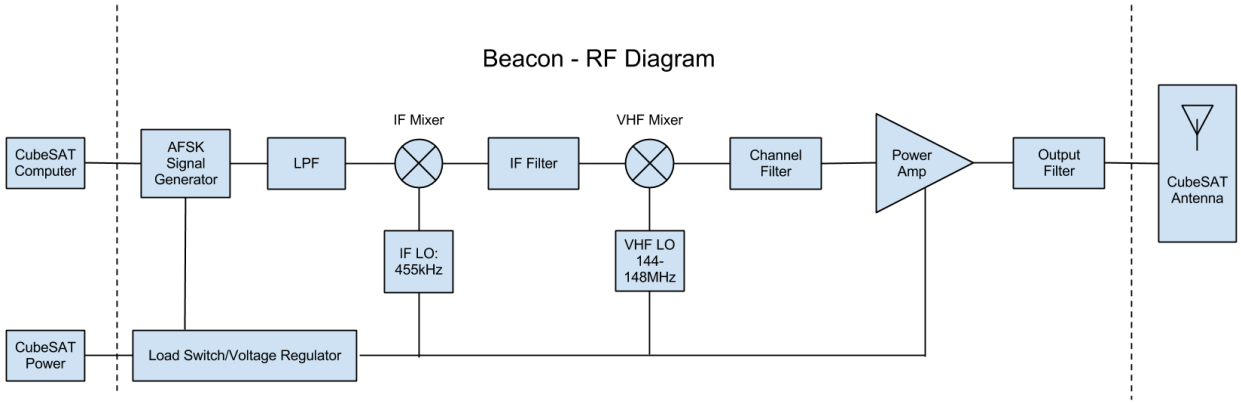

Figure 16: First Revision Block Diagram. This scheme was fundamentally flawed. Given the receiver equipment available by the CubeSAT team, there was no realistic way of successfully transmitting data. No frequency modulation was actually being performed by this topology.

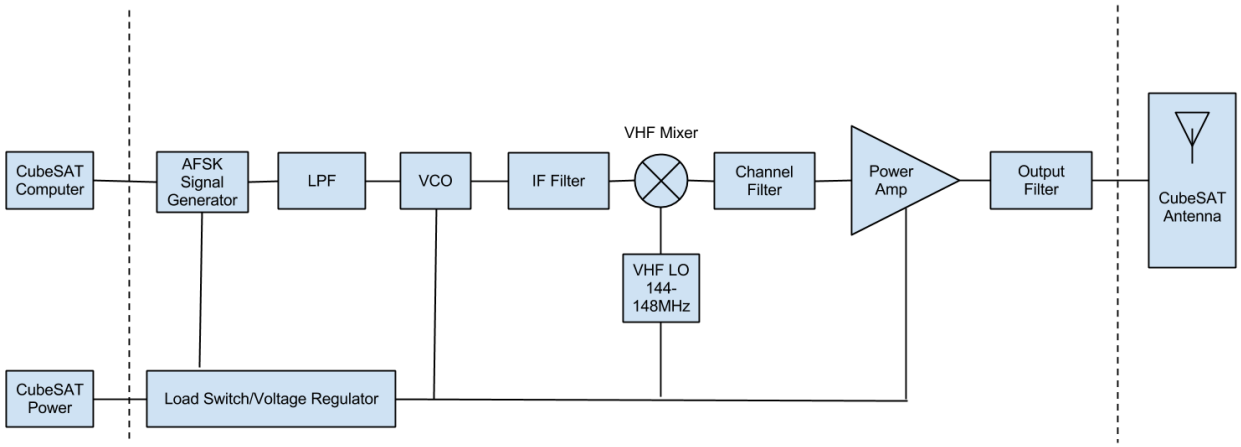

Figure 17: Second Revision Block Diagram. In this revision, frequency modulation was correctly implemented with a VCO. However, the mixer used (and VCO constructed from a mixer) proved to provide disastrous gain compression problems. The PCB made for this revision was never fully constructed, and instead was used to retrofit and demonstrate the working Final Revision.

F Parts

Part	**Quantity**	**Part Number**	**Cost**
Resistor - 49.9 Ω	4	292-49.9-RC	$0.16

Resistor - 200 Ω	2	RR1220P-201-D	$0.30
Resistor - 10.5 Ω	2	292-10.5-RC	$0.08
Resistor - 100 Ω	1	CR0805-FX-1000ELF	$0.05
Resistor - 5.49 kΩ	1	292-5.49K-RC	$0.04
Resistor - 1 kΩ	1	292-1.0K-RC	$0.04
Resistor - 1 MΩ	2	292-1.0M-RC	$0.08
Resistor - 0 Ω	1	CR0201-J/-000GLF	$0.06
Capacitor - 0.1 μF	2	GRM188L81H104KA93D	$0.20
Capacitor - 1 μF	2	VJ0603V105MXQCW1BC	$0.06
Capacitor - 4.7 μF	2	CC0603KRX5R6BB475	$0.20
Capacitor - 0.01 μF	3	VJ0603Y103JXQCW1BC	$0.24
Capacitor - 33 pF	1	GRM1885C1H330FA01J	$0.10
Capacitor - 430 pF	2	06035A431JAT2A	$0.88
Capacitor - 1 pF	2	C0603C109B5GACTU	$1.04
Capacitor - 0.7 pF	2	600S0R7BT250XT	$2.14
Capacitor - 100 pF	2	600F101GT250XT	$4.10
Capacitor - 150 pF	2	C0603C151F5GACTU	$2.02
Inductor - 1.8 μH	2	IMC1210ER1R8J	$1.14
Inductor - 5.6 μH	2	ILSB0805ER5R6K	$0.32
Inductor - 8.2 nH	2	581-HLQ028R2BTTR	$1.14
Inductor - 2.7nH	2	LQP10A2N7C00T1	$0.66
Inductor - 1.2uH	1	IMC1008ER1R2J	$0.16
PIC Microcontroller	1	PIC16LF1508-E/SO	$1.65
Voltage Regulator	1	LP38692MP-5.0	$1.19
OpAmp	2	EL5167	$7.62
VCXO	1	FVXO-HC53BR-144.015	$51.91
Test Points	10	5002	$5.80
DF-11 Connector (8pin)	1	DF11-8DP-2DSA(08)	$0.51
DF-11 Connector (6pin)	1	DF11-6DP-2DSA(08)	$0.43
SMA Connector	1	142-0701-801	$3.89
PCB	1	N/A	$100

Total			$187.95
Laborer	**Rate ($/hour)**	**Hours**	**Total (Rate*2.5*hours)**
Russell	$50.00	150	$18750
Neal	$50.00	200	$25000
Jeff	$50.00	150	$18750
Labor Totals	**$50.00**	**500**	**$62500**
Grand Total			**$62687.95**

G Future PCB Layout

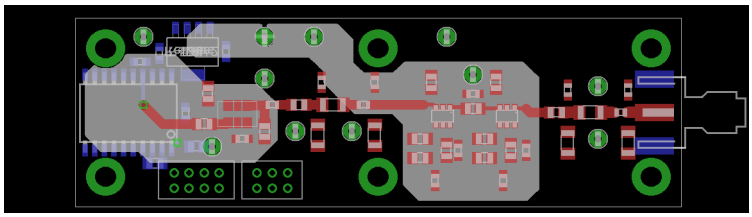

Figure 18: The next revision PCB layout may look something like this. The grey area is a split 3V3 and 7.4 V power plane, which is much less chaotic than the second revision PCB. The RF signal path has been routed to show the simplicity of the next revision when compared to previous PCB layouts.