

CubeSat FSK Modem Implemented on a DSP

Joe Doughney, Darren Pocci, Chris Williams

1 Overview of project goals and practical applications

To build a frequency-shift-keying (FSK) modem, which is a device that can demodulate an analog signal to digital information and modulate digital information to an analog signal, using a digital signal processor (DSP) for the CubeSat satellite. This modem will be able to receive and transmit signals in a real world environment, where noise is an issue, to and from a host computer via the KISS protocol. Additionally, the modem was originally intended to be able to switch between a 1200 and 9600 baud rate. However, only the 1200 baud rate was implemented for reasons described later.

Practical applications for modems are any application that requires analog data to be translated into digital data over data receiving and transmission. Applications of modems include radios, internet connection over a Plain Old Telephone System (POTS), WiFi, and satellite communications [1]. For this project, satellite communication was the application of significance as this modem is designed to be apart of the CubeSat project where data is collected about the Earth in a small satellite and transmitted to a base station on Earth [2]. A DSP was chosen to implement an FSK modem as it is a low power consuming device compared to a commercial modem.

2 Relevant Theory

Frequency-shift-keying involves shifting a carrier frequency to two different frequencies called mark and space that correspond to a binary one and binary zero, respectively. Figure 1 gives a good overview of an interpreted FSK signal. For the modem, quadrature demodulation is implemented. In quadrature demodulation, the received signal, $s(n)$, is multiplied by a delayed signal, $s(n-k)$. The delayed signal is the received signal delayed by a number of samples, k [3]. A pictorial representation is given in Figure 2. Since the modem will employ binary FSK, the received signal, $s(n)$, will be either Eq. (1) or Eq. (2) where f_1 is the mark frequency, f_0 is the space frequency, and f_s is the sampling frequency:

$$s(n) = A \sin(2\pi f_0 n / f_s) \tag{1}$$

$$s(n) = A \sin(2\pi f_1 n / f_s) \tag{2}$$

Analyzing $s(n)*s(n-k)$ yields four possible results, two of which are significant and are shown in Table 1. The second term of each result is irrelevant and will be omitted as it can and will be low pass filtered out. The first term of each result is only a function of k , therefore this value will be constant when $s(n)$ and $s(n-k)$ have the same frequency component. When the frequency components are different, a sample varying signal results but only serves as a transition between the two constant values. As a result, the output of the low pass filtered demodulated signal will be a varying function similar to the original binary wave [3]. Figure 3 gives an overview on the entire process. To maximize the separation between the two constant values, and thus the efficiency, k must be chosen to maximize:

$$d(k) = |\cos(2\pi f_1 n / f_s) - \cos(2\pi f_0 n / f_s)| \quad (3)$$

To implement a continuous phase FSK transmitter, the transmitter needs to keep track of the phase of the sine wave during transitions between symbols [4]. For example, say the transmitter starts to transmit the space sine wave:

$$s(n) = A \sin(2\pi f_0 n / f_s + \phi) \quad (4)$$

After N samples are outputted, and using the same general formula as Eq. (4), ϕ needs to be updated such that:

$$\phi_{new} = 2\pi f_0 N / f_s + \phi_{old} \quad (5)$$

The next sine wave can then be transmitted according to Eq. (4) using Eq. (5), changing for the appropriate space or mark frequency.

To communicate between the modem and host computer, a protocol called KISS was implemented. KISS protocol is used such that frames of data may be sent to and received from the computer properly. KISS protocol implements special framing characters, shown in Table 2, to perform specific functions. Basically, every frame sent between the host and DSP needs to begin and end with a FEND character to specify the beginning and end of the frame. The FESC character is used such that the receiving unit does not improperly interpret a framing character in the actual data. This error avoidance is accomplished by the transmitting unit translating each

FEND or FESC character in the data into FESC TFEND and FESC TFESC, respectively. The FESC character is used to put the receiving unit in escape mode such that the FESC and the following framing character are translated into the original character used in the data. As a result, the receiving unit can append a FEND or a FESC to the data without perceiving it as the end of a frame or going into escaped mode [5].

3 Original Objectives

The original idea for implementation is summarized in the receiver and transmitter block diagrams in Figure 4 and Figure 5, respectively. For the receiver, the incoming analog signal would be sampled and sent to two separate bandpass filters that would only pass either signals at the mark or space frequencies. The power of these signals would then be calculated via the sum of squares of the samples and compared in the discriminator [6]. If the mark filter output had higher power than the space filter output, a binary one would be appended to the output and vice versa. Synchronization would be performed off of the DSP's clock and triggered once a specific start sequence was detected. For the transmitter, a simple discriminator would determine if the binary input was a zero or one and use a look up table to output the sine wave corresponding to binary one (mark) or binary zero (space) frequency. Computation of the sine wave was also a possibility, but needed to be tested to see if the computation could be performed under the time of the sampling period. Data transmission to and from the host computer was not initially planned for, but ideas to use the DSP's Real Time Data Exchange (RTDX) protocol was brought to our attention four weeks into the project.

4 Implementation Details

Additional features included modem operation at ITU standards of 1.3 kHz for the mark frequency and 2.1 kHz for the space frequency. The modem data rate was also set to 1200 baud.

4.1 Receiver

The receiver takes the input as a sinusoid and creates a buffer of binary data from that input signal as shown in Figure 6. The number of samples to delay the signal was chosen to maximize Eq.

(3), which resulted in the choice of 35. Figure 7 gives the numerical values for Eq. (3). The first part of the receiver is a band pass filter with cut off frequencies of 700 Hz and 2600 Hz. The filter is an 8th order Chebyshev Type II filter with magnitude response shown in Figure 8. These cut off frequencies were set to make sure that our signals at 1300Hz and 2100Hz were not attenuated at all but also to eliminate out of band noise from contributing to the signal being interpreted. After this process is completed, the signal is demodulated using quadrature demodulation, which was described in the theory section. The demodulated signal is then low pass filtered with a cutoff frequency of 1200 Hz. This filter is a 4th order Elliptical filter with magnitude response shown in Figure 9. The cutoff of 1200 Hz was chosen since the baud rate is 1200 symbols per second and, therefore, is the fastest that the signal can be switching back and forth from a one to a zero. Once the signal has been low pass filtered it is squared and compared to a threshold value. This threshold value is used to determine when the signal has arrived. It is expected to wait in the receive function for some time before data is started to be received. The signal is squared to get a larger difference between random noise and the desired signal; also, because the first binary data could be a zero, which is a negative number after being demodulated. Once the threshold value has been triggered, the incoming bits are appended to a buffer every 40 samples. The demodulated signal is sampled every 40 samples because this is the number of samples received when sampling a 1200 baud signal at 48 kHz, the sampling frequency of the DSP.

In addition, a synchronization word is also being detected to synchronize which sample represents the data being sent. The first byte received is always going to be 01111110. Thus, when the last zero crossing (from 1 to 0) is detected, the center of the signal is 20 samples away. Therefore, the value of the demodulated signal 20 samples after the zero crossing of the synchronization word and every 40 samples after that is when the receiver discriminates the data. This method insures that the sample is most representative of the binary bit currently being received. From that point, the incoming bits are appended to a buffer every 40 samples until the signal being transmitted has stopped (channel clears), or the buffer has been filled. To determine if the signal is no longer present, the signal is squared and compared to a threshold value. Once the process of receiving the signal is complete, the index to the buffer, or the variable that keeps track of how many samples have been received, is saved. Therefore, when the received bits are sent to the computer we know the length of the buffer that we are outputting.

4.2 Transmitter

For the transmitter, a sine look-up table was used instead of calculating the sine wave directly as shown in Figure 10. Calculating the sine value would allow for a pure continuous phase modulated sine signal. However, constraints on computing power and the excess time it took to compute the value of sine each sample eliminated this option. Fortunately, a close to continuous phase output was achieved by using an over sampled sine look-up table. It is important to get as close to continuous phase as possible because without continuous phase there would be a lot of high frequency noise in the signal. While in the implementation of the receiver the noise is filtered out, it is desired such that the transmitter is able to communicate with all kinds of receivers, some of which may be affected by this high frequency noise. In order to get a continuous phase modulated signal using a sine look-up table, the sine look-up table must be oversampled by a factor of 13. However, in an effort to save on space in memory it was found that to get very few glitches, or high frequency components, the look -up table only needed to be over sampled by a factor of 4. This claim was confirmed by looking at the FFT of the signal on the oscilloscope. Therefore, this oversampling factor was used for the transmitter.

4.3 KISS Protocol

Aside from the receiver and transmitter, the decoded bits need to be sent to the host and bits need to be received from the host. This action, in theory, would be done across the serial line via the KISS protocol. The DSP used with the project did not have a serial port, but data can be sent to and from the computer via Real Time Data Exchange (RTDX) over the USB port. Unfortunately, implementing RTDX was unsuccessful for reasons unknown. However, the data that would be sent to and received from the host was implemented via KISS protocol using memory. It is hoped that data can be sent across the serial line using a pointer to the appropriate memory location. To implement KISS protocol, on the DSP, a FEND is appended to the beginning and end of the data that is being sent from the DSP to the host, while replacing FEND and FESC in the data with FESC TFEND and FESC TFESC, respectively, before sending it to the host. Also, the data received from the host needs to have the FEND stripped off from the beginning and end while translating the FESC TFEND and FESC TFESC into FEND and FESC, respectively, before

transmitting over the air.

5 Accomplishments & Failures

Originally, the filter method of detecting the mark and space frequencies was used. It was implemented by first low passing and downsampling the signal by a factor of 5 to save on computations. Then, the signal was passed through the two band pass filters. Over the 8 samples per symbol, the the sum of squares of the filtered mark and space signal was calculated. However, it was found that the impulse responses of the filters did not attenuate fast enough. In early testing the filter method worked fine when a pure sinusoid was input, but when a modulated signal was input the filters would still be resonating when they should have had an output of zero. This error caused false positives in detecting the signal. When impulse responses of the filtered were made to be quicker by lowering the filter's order, in band interference occurred. Therefore, the filter method was abandoned. To replace the filter method, quadrature demodulation, described in the theory section, was used. Quadrature demodulation gave a clear response to what the incoming bit was and consistently gave the correct response to our transmitted signal.

To communicate to the host computer, RTDX was the hopeful implementation. However, because RTDX must be used with blocks of samples at a time, the original working code needed to be adjusted from the sample by sample method. When the adjustment was made, the original code would work without implementing RTDX. When RTDX was added, the code would run but then get lost in disassembly without ever returning. Therefore, it was decided to just display the data in a memory window since the functionality of our project did not depend on RTDX.

While the modem ran at 1200 baud, 9600 baud operation was unsuccessful. At 9600 baud, there were only 5 samples per symbol are available. These samples would be unable to complete a full period when transmitting the signal at 1.3 kHz and 2.1k Hz, therefore rendering it impossible to demodulate it. It was later found out that Cubesat does not implement the 9600 baud transmitter the same way as the 1200 baud transmitter. Therefore, the 9600 baud rate feature was abandoned.

KISS protocol was implemented effectively as described in implementation details. Also, the transmitter was successfully implemented by using a lookup table described in the implementation details. Originally, a true continuous phase transmitter was attempted but the sine computations

took too long for our sample by sample method and caused glitches in the output. Thus, an oversampled sine lookup table was used and was able to get an output that was close enough to continuous phase.

6 Description of Code

The code is broken up into several different blocks so that it would be easier to debug, it is more clear to someone viewing the code for the first time, and also to make it easy to change things quickly. The major groupings of the code are input, receiver, bit output, bit input, transmitter, and output. Each of these are inside of an "if" statement so that each section can be conditionally executed during different stages of the modem's process.

The first of these, input, simply just took the raw data samples from the serial port and loaded them into a location in memory. While this could have been done inside of another part of the code, such as the receiver portion, by having a function that does only the input it would be really easy to change where the input was coming from. In other words, if this code was to be put on another DSP (like the DSP Cubesat is using for example) and it had different inputs then this function would have to be slightly altered and the rest of the code could remain the same. For this same reason the output function is separated from the computational functions so that the code will still work and any other output method that may be used will be able to take the output from a global buffer.

After the data has been input, the receiver portion of the code is the first to be executed. This block of code took the input from a microphone during demonstration, but could also be taken from a wire or antenna, and converted the sinusoid it received into binary data. This section of the code continued to receive data until the signal that was being received was lost or the buffer it was writing too had been filled completely. Once this part of the code has ended by either of the two previous methods a variable is set that tells the code where to go the next time around. From receive the next function is bit output.

In bit output the binary data that was just received is sent to the computer. Before the data is sent out it needs to be changed into KISS format for the computer to understand. To do this, a FEND is appended to the beginning of the data, which denotes the start of data transmission, a

FEND is also sent once the length of data that has just been received is finished being transmitted. While the data is being sent out any FEND or FESC words that are found in the data are replaced with FESC TFEND or FESC TFESC. This is described in more detail in the methods section. This must be done so that there is no ambiguity of when the end of the data stream has been reached. Once all of the data has been sent that was just received through the air, the necessary variables are set so that the next section of code that is executed is bit input.

Bit input is the part of the code where the DSP waits for the computer to send a string of data in K.I.S.S format. Once this string of binary numbers has been received the KISS formatting is stripped off so that the original binary string is ready to be sent out via the transmitter. Once either the buffer has been completely filled or the second FEND has been received delimiting the end of the data being transferred, the necessary variables are set to go into the transmitter portion of the code.

The transmitter takes a string of ones and zeros that have been stored into a buffer and outputs the corresponding frequency, 1.3kHz for a 1 and 2.1kHz for a 0, for each bit in that buffer. However at the beginning of transmission, a synchronization word is first sent out. Once that has been sent out the DSP begins transmitting the data that has just been received from bit input. This data is then transmitted until the end of the buffer is reached or the DSP has sent the length of the signal that the CPU sent it. Once this is done the necessary variables are set and the process is restarted.

7 Testing Methodology & System Performance

7.1 Testing Methodology

To test the modem a random generated bit string was sent from one DSP and a second DSP was used to receive the information. More specifically, data was transmitted from one DSP and out through a speaker. A microphone then picked up the signal and sent it to a second DSP where the data was stored in a buffer. This showed that the program could convert from digital to analog and back to digital without losing any data while in an open air environment. To verify that it worked correctly the received data was compared to the sent data. The data was also displayed on the oscilloscopes to prove that the data was sent using FSK and received using quadrature demodulation.

Also, the modem simulated the KISS protocol (described above), which is a protocol for communication between the DSP and computer. To simulate this, the received data was sent to a second buffer where the appropriate framing bits were added (simulating the DSP talking to the computer). This data was then sent back to the original buffer (simulating the computer talking to the DSP) and the framing bits were stripped off leaving us with our original data string. The data that was just placed in the buffer should now match the data that we received from the transmitter earlier only without the synchronization sequence on the beginning. In reality the computer would not send back the same bit string it received. This was simulated to show that the KISS protocol worked properly, and that the modem will be able to communicate with the computer on the satellite. Finally, a demonstration was made to show that the modem could then transmit the data the computer sent to it by showing the FSK of the data on the oscilloscope.

7.2 System Performance

To test the modem, ten trials were conducted where a random number sequence of 2000 bits long was generated and outputted via the transmitter to a acoustic speaker. A microphone received the acoustic signal and decoded the bits while running through each state of the modem. The transmitter output was confirmed by visualization on an oscilloscope. The receiver and its states were verified by observing memory. Each of the ten trials resulted in the correct 2000 bit sequence received, outputted to the host, and received from the host. Therefore, the modem was deemed a well performing modem.

8 Pitfalls

Before starting this project it would have helped to have considered the problems that the impulse delay times of the filters would cause. A lot of time was spent trying to get this method to work, and because of the reasons mentioned earlier it was decided to implement the program with a different method. The RTDX feature in MATLAB is another area that would have helped to know more about before this project started. This was a very unfamiliar feature, which meant the program was unable to work properly with RTDX. If it was implemented correctly it would have made for a better presentation. Lastly, it would have been useful to know that we did not

have the computational power to calculate the value of sine each iteration of the code. This led to trying multiple methods to see what would work the best, and finally a over-sampled look up tables were encoded for the two frequencies, but a fair amount of time was lost trying all of these methods out. If these things were known before the start of this project a great deal of time would have been saved, and with this time it would have been possible to implement some of the things mentioned above that there was not enough time to complete.

9 Extensions

Given more time an attempt would have been made to make the program work at 1200 and 9600 baud, instead of just at 1200 baud. The problem with this is that it would have been necessary to change a lot of things in the code to get it to work, and time just didn't permit this to happen. Also, finding a way to use RTDX with the program would have been another thing to try given more time. This would have allowed the modem to talk to the DSP and tell it what to send through MATLAB GUI. Implementing the program in order to test it with the CubeSat equipment would have been another good idea to try out if there was more time. This would have made the testing more realistic, and it would have made for a better presentation because it would provide a better simulation of how the modem would actually work on the satellite. Lastly, due to the lack of time the program did not include detection of the carrier frequency, which would have been the next thing to include if there was more time.

References

- [1] Wikipedia, “Modem,” Wikipedia, the free encyclopedia, <http://en.wikipedia.org/wiki/Modem>.
- [2] Illinois Tiny Satellite Initiative, “CubeSat @ UofI,” University of Illinois at Urbana-Champaign, <http://cubesat.ece.uiuc.edu>.
- [3] G. Baudoin, F. Virolleau, O. Venard, and P. Jardin, “Teaching DSP through the Practical Case Study of an FSK Modem,” Texas Instruments, Tech. Rep., 1996.
- [4] Matthew Berry, Robert Morrison, “Digital Transmitter: Introduction to Frequency Shift Keying,” Connexions Module, <http://cnx.org/content/m10659/latest>.
- [5] Mike Chepponis, Phil Karn, “The KISS TNC: A simple Host-to-TNC communications protocol,” Computer Networking Conference, <http://www.ax25.net/kiss.htm>.
- [6] Watkins-Johnson Company, “FSK: Signals and Demodulation,” *Tech-Notes*, vol. 7, no. 5, 1980.

Appendix

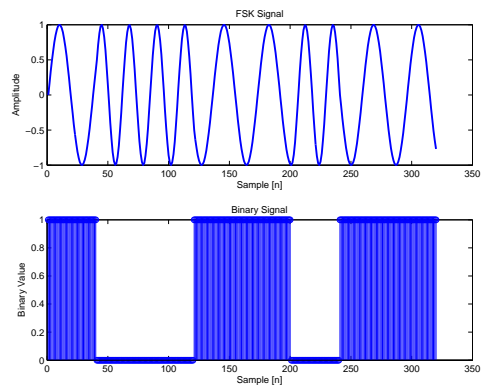


Figure 1: Above: FSK Signal. Below: Corresponding Binary Signal

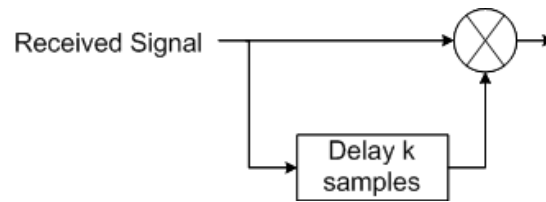


Figure 2: Quadrature Demodulator

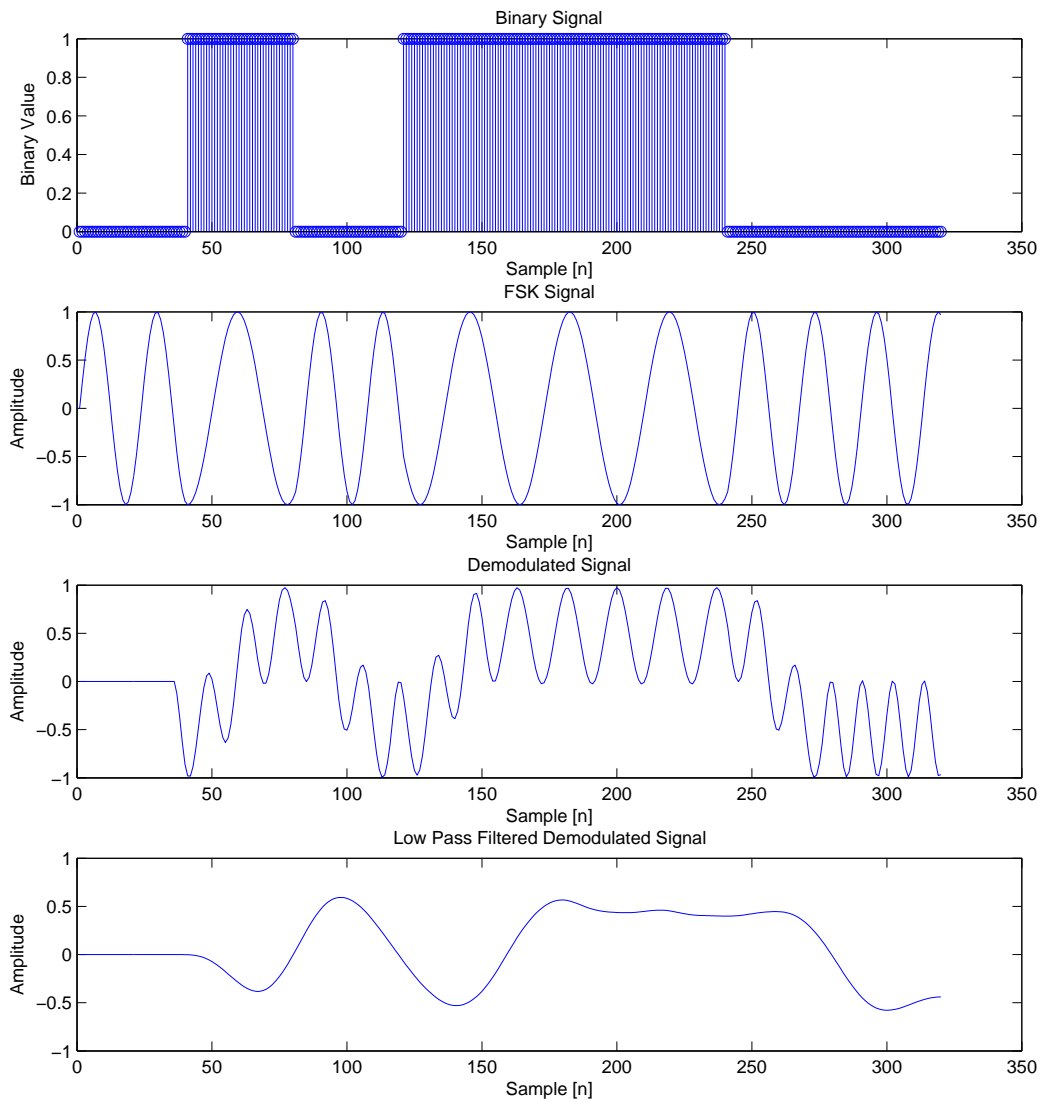


Figure 3: Demodulation Process

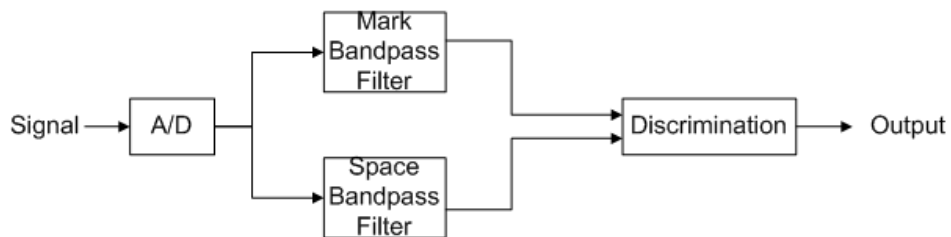


Figure 4: Original Receiver

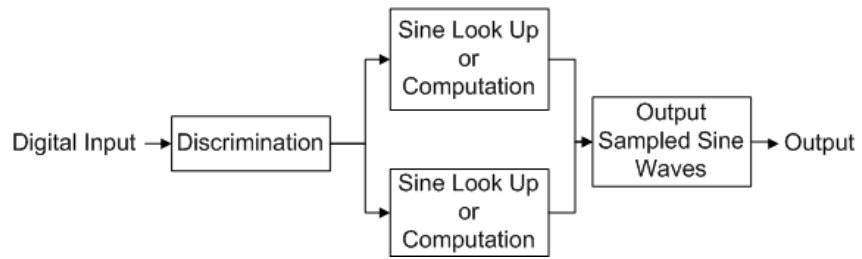


Figure 5: Original Transmitter

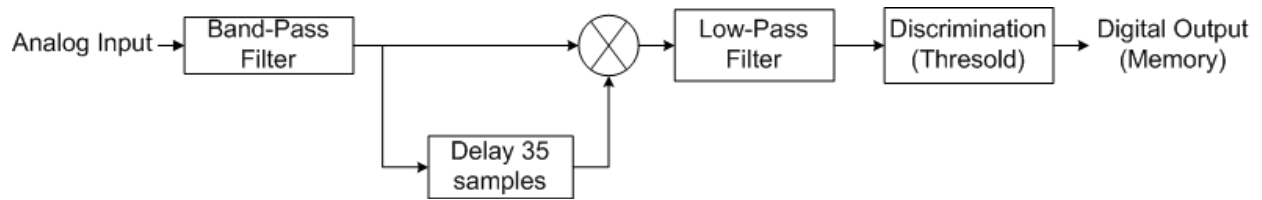


Figure 6: Receiver

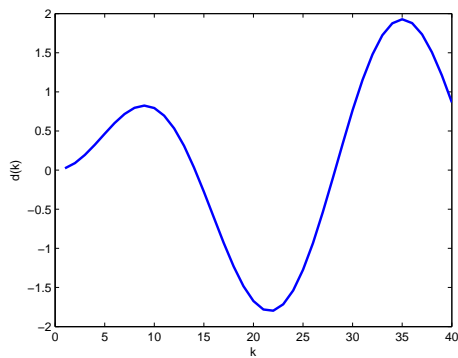


Figure 7: Effectiveness of Number of Samples to Delay Signal

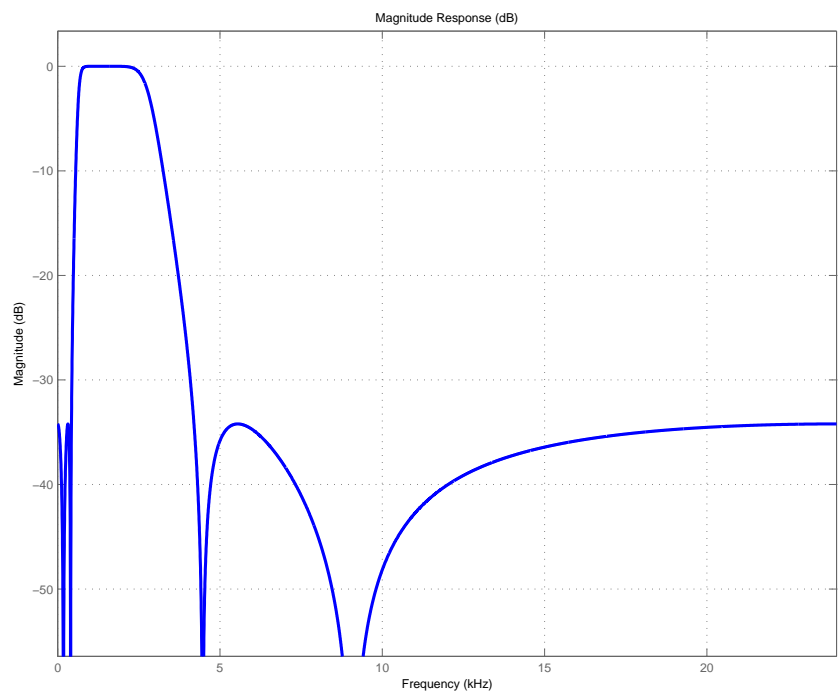


Figure 8: Magnitude Response of 8th-order Chebyshev Type II Band Pass Filter

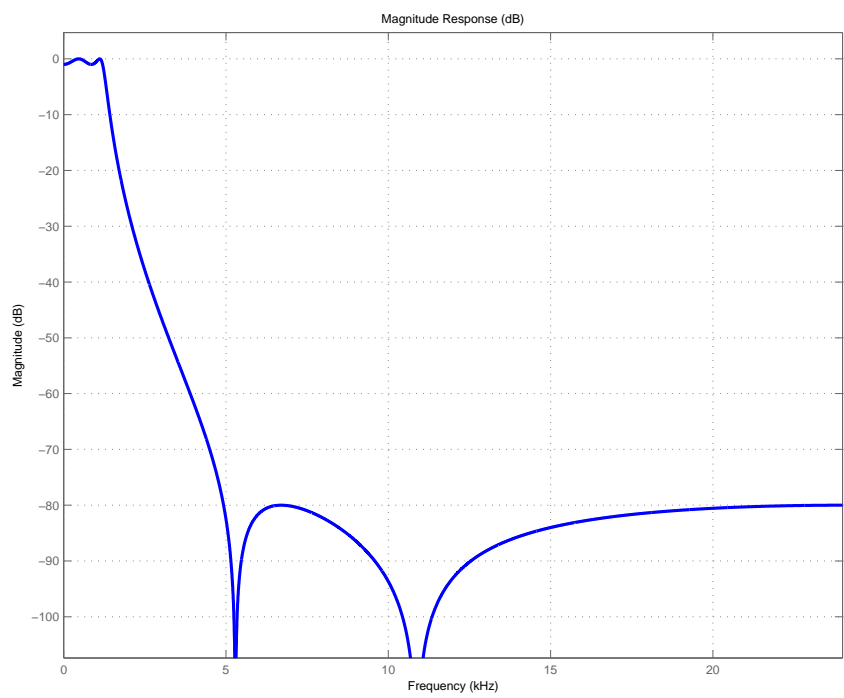


Figure 9: Magnitude Response of 4th-order Elliptical Low Pass Filter

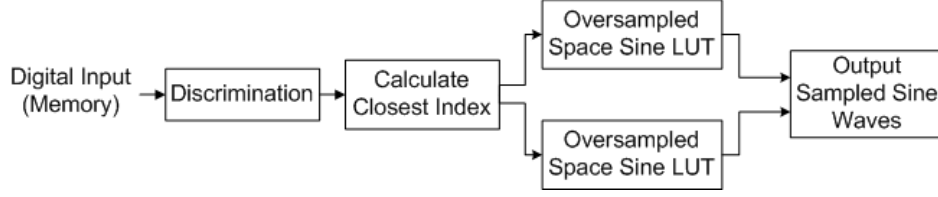


Figure 10: Transmitter

Table 1: Quadrature Demodulator Output

$s(n)$	$s(n-k)$	$s(n)*s(n-k)$
$A \sin(2\pi f_0 n / f_s)$	$A \sin(2\pi f_0 (n-k) / f_s)$	$\frac{A^2}{2} [\cos(2\pi f_0 k / f_s) - \cos(4\pi f_0 n / f_s - 2\pi f_0 k / f_s)]$
$A \sin(2\pi f_1 n / f_s)$	$A \sin(2\pi f_1 (n-k) / f_s)$	$\frac{A^2}{2} [\cos(2\pi f_1 k / f_s) - \cos(4\pi f_1 n / f_s - 2\pi f_1 k / f_s)]$

Table 2: KISS Framing Characters

<i>Abbreviation</i>	<i>Description</i>	<i>Binary Value</i>
FEND	Frame End	11000000
FESC	Frame Escape	11011011
TFEND	Transposed Frame End	11011100
TFESC	Transposed Frame Escape	11011101