FOX-1 Telemetry Coding And Modulation Design

Phil Karn, KA9Q
Paul Williamson, KB5MU
Michelle Thompson, W5NYV

Introduction

The AMSAT FOX-1 satellite is scheduled for launch in November 2014. FOX-1 will carry a conventional FM repeater with an unconventional telemetry design. There will be two modes:

1. A "Full-Speed Data" mode that will run at about 10 kb/s when the repeater isn't operational. This mode is intended for receiving camera images from the spacecraft.

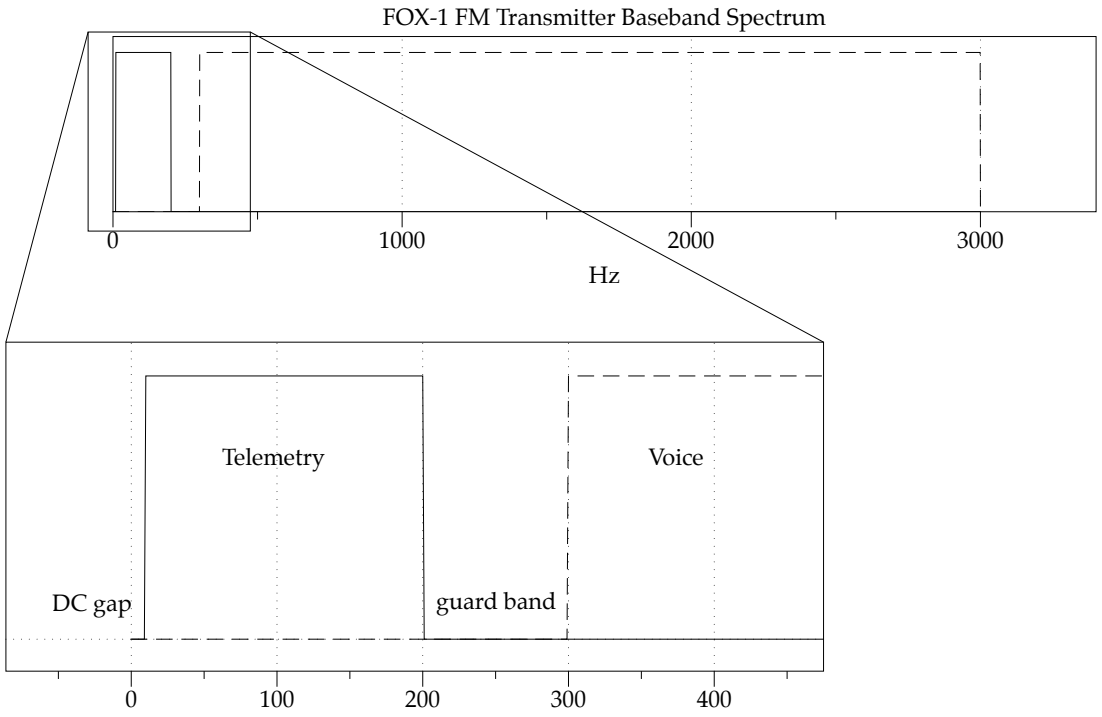
2. A "Data Under Voice" mode that will run continuously during repeater operation in the sub-audible range below 200 Hz.

Because FOX-1 carries only a FM transmitter, all data will be sent as noncoherent FSK.

Data Under Voice Mode

During normal FM repeater operations, FOX-1 will transmit a continuous telemetry stream at a data rate of 140 bps. This very low rate is necessary because the signal must fit below the normal speech audio band that begins at about 300 Hz. To provide for a margin, the channel signaling rate will be 200 bps, which results in a spectral null at 200 Hz. A relatively simple lowpass filter removes the spectral components above this frequency.

The 60 bps difference between the data and channel bit rates is taken up by forward error correction coding (a Reed-Solomon block code) and line coding for spectral shaping.



Forward Error Correction

AMSAT's extensive experience with LEO satellites suggests that the primary impairment of the FOX-1 telemetry channel will be slow fading due to a non-uniform antenna pattern, slow spacecraft rotation and a changing view angle as seen by the ground station. Because of the generous link margins afforded by the relatively short slant range, the signal should be very strong except when in a fade.

AMSAT FOX-1 Downlink Budget (baseline)
KA9Q, Fall 2013

400		mW	TX power
2000		km	Range
145		Mhz	Frequency
0		dB	Tx losses
0		dBi	Tx gain (average)
0		dB	Rx losses
0		dBi	Rx gain
2000		K	Noise T
2.07	=B28/(B6*1000000)	m	wavelength
-3.98	=10*LOG10(B4)-30	dBW	Tx Power
-3.98	=B16+B8-ABS(B7)	dBW	EIRP
141.7	=20*LOG10(4*PI()*B5*1000/B15)	dB	Path Loss
-145.68	=B17-ABS(B18)	dBW	Rx dens
-145.68	=B19+B11-ABS(B10)	dBW	Rx Power
-195.59	=10*LOG10(B27*B12)	dBW/Hz	Rx N0
49.92	=B20-B21	dB	P/N0
15000	15000	Hz	Bandwidth
8.15	=B22-10*LOG10(B23)	dB	IF SNR
1.38E-23		J/K	Boltzmann k
299792458		m/s	Speed of light c

FOX-1 downlink spreadsheet summary

Random bit errors are characteristic of weak signal systems, such as a high-altitude satellite where the primary impairment is thermal noise. These sorts of systems are modeled as channels with additive white gaussian noise (AWGN). The errors are distributed randomly with respect to time. Convolutional encoders are customarily used in such applications because they are a good solution for the detection and correction of random errors. The conventional choice of forward error correction (FEC) for the AWGN satellite channel is convolutional coding with Viterbi decoding.

For a channel that has slow deep fades, but is otherwise strong, the conventional choice is Reed-Solomon block coding. Reed-Solomon codes don't perform as well as convolutional codes when dealing with random bit error patterns, and convolutional codes don't handle deep fades as well as Reed-Solomon.

Sometimes the two can be combined as on AMSAT-OSCAR-40 where both thermal noise and fading due to an asymmetric antenna pattern were serious concerns. That design included two layers of interleaving to mitigate the otherwise harmful effects of fading on convolutional coding.

The error correction code proposed for the FOX-1 low speed telemetry mode is the venerable (255,223) Reed-Solomon (RS) block code over GF(256).[1] Each RS symbol is an 8-bit data byte for which the natural block size is 255 bytes (2040 bits) containing 32 bytes of parity and 223 bytes of data.

RS codes are well suited to fading channels because they can correct a burst of errors up to one half the number of parity symbols long. For FOX-1, this equals 16 bytes in the data-under-voice channel.

A single (255,223) RS frame can tolerate a fade up to 800 milliseconds long in each 12.8 second frame.

A special synchronization byte is sent between RS blocks not only to help the receiver find block boundaries, but also to help avoid the false frame sync that can theoretically occur if the receiver were to rely only on successful RS decoding. This can happen because RS codes are cyclic codes. Cyclic codes have the property that any shift of a valid codeword always forms another valid codeword. Should two consecutive frames contain the same user data, the end of one frame followed immediately by the beginning of the next would produce a valid but spurious RS codeword even with incorrect frame alignment. The synchronization byte alleviates this risk.

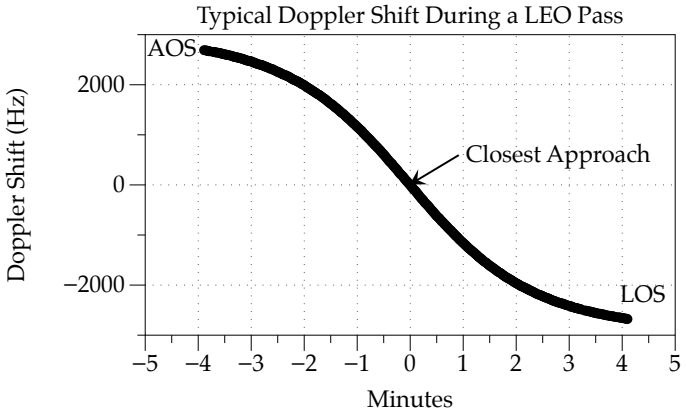
The false sync problem can also be avoided by "shortening" the RS codeword where the beginning of the data field is padded with zeroes

that are not actually transmitted. Shortened RS codes are not cyclic except in the special case where the data is all zeroes. The FOX-1 software team will choose the final RS frame size to fit the "natural" size of a telemetry data frame.

Spectral Shaping

A common problem in many digital transmission channels and storage systems is that they cannot pass or store signals with very low frequency (or DC) components. Typical FM modulators generally cannot create signals with DC or low-frequency components, and typical FM demodulators cannot recreate DC or low-frequency components. These physical limitations are assumed to exist for FOX-1 hardware.

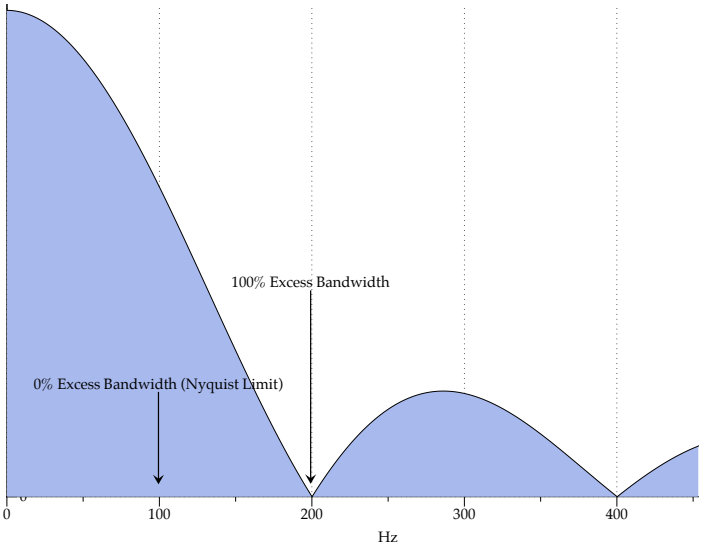
An additional related problem occurs with Doppler shift. When FM is received from a low-altitude satellite like FOX-1, the Doppler shift appears as a low-frequency component on the modulation, almost as if there was a DC offset. Information in low-frequency components will be lost. Our decoder cannot rely upon low-frequency or DC components. Therefore, the encoder should eliminate low-frequency or DC components.



The simplest and most common baseband encoding of a binary digital data signal is the so-called non-return-to-zero (NRZ) format where a logic '1' is encoded as a positive signal and a logic '0' is encoded as a negative signal of the same magnitude (or vice versa). A closely related encoding, familiar to amateurs from its use in the HDLC framing of the AX.25 packet radio standard, is NRZI: a '0' is encoded as a signal change (from - to + or from + to -) and a '1' is encoded as no change.

For both NRZ and NRZI, random data produces a sin(x)/x (or sinc) shaped spectrum with nulls at multiples of the data rate and a maximum spectral amplitude at DC (0 Hz). This is clearly undesirable.

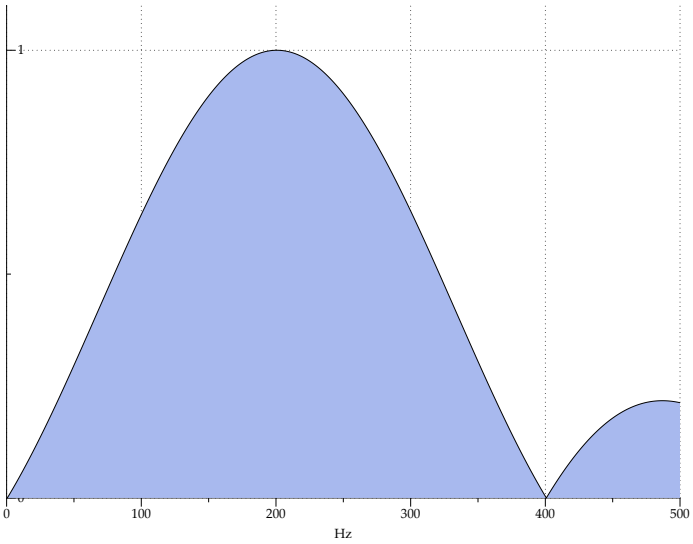
The medium speed (9600 bps) FM/FSK modems pioneered by K9NG and G3RUH address this problem with a self-synchronizing scrambler, but it is important to understand that it doesn't really fix the problem. The scrambler will eliminate the even more troublesome spectral line at DC that occurs when the data stream contains long constant data strings (NRZ) or long strings of 1's (NRZI), or when the 0's and 1's are otherwise not balanced. The scrambler "whitens" the data, i.e., it looks random, but a random NRZ stream still has a spectral maximum at DC.



Unfiltered spectrum of 200 baud NRZ and NRZI codes

Sending a signal with a spectral maximum at DC over a non-DC-coupled channel, like FM, produces a waveform that, in the short term, is no longer symmetric around the time axis. This makes detection less reliable.

But, there are digital encodings with no DC component regardless of the data sequence, such



Unfiltered spectrum of 200 baud Manchester code

as the several flavors of Manchester coding. The best known encodes a logic '0' as one cycle of a square wave at the bit rate and a logic '1' as one cycle of the same square wave with opposite polarity. That is, a '0' might become '01' while a '1' would become '10'. This completely eliminates DC and greatly reduces low frequency components at the cost of doubling the total signal bandwidth. Peak spectral density now occurs at the data rate (where NRZ/I has a null) with the first null occurring at twice the data rate. This method was used in the AMSAT Phase III 400 bps telemetry system.

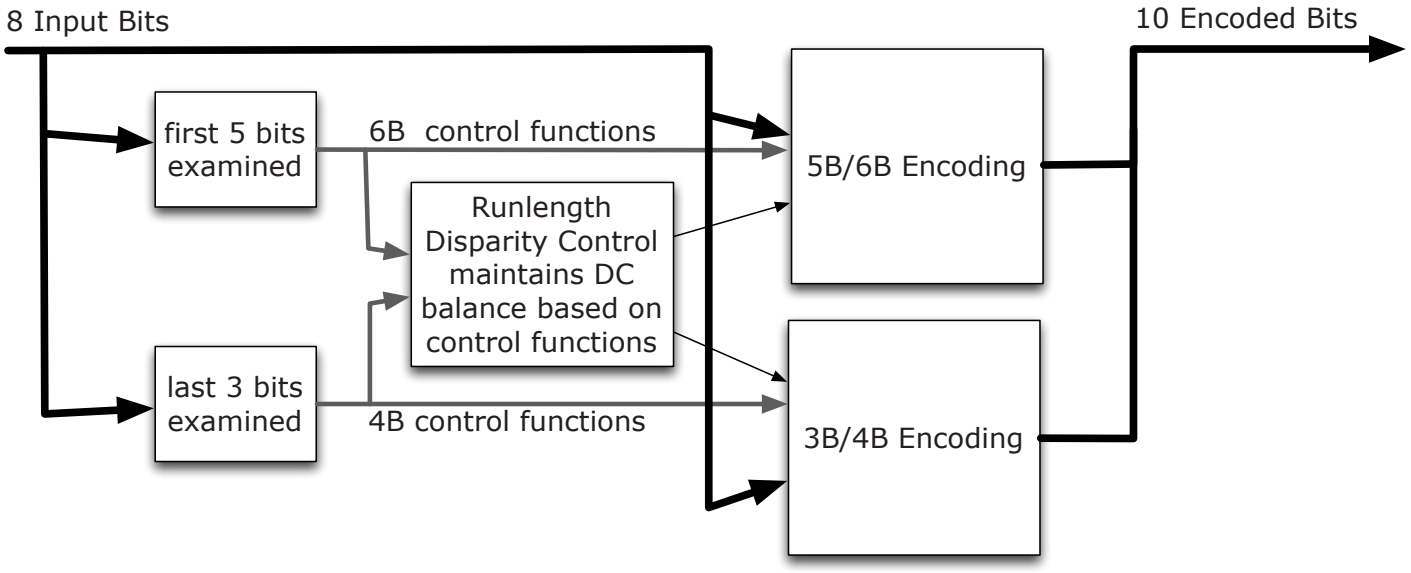
It seems like there should be a better way, and indeed there is. The specific method chosen for FOX-1 was suggested by Tony Monteiro, AA2TX. It is called 8b10b coding.

8b10b Line Coding

In 8b10b coding, 8-bit groups of data bits are turned into 10-bit groups of channel bits with a lookup table and a 1-bit memory. The tables and rules were chosen so that regardless of the data the channel always contains an equal number of 0's and 1's over the long term. In other words, it is DC-free. No 20-bit channel sequence contains more than two extra 1's or 0's, minimizing low frequency components. And there are never more than five 0's or 1's in a row so that transitions are frequent enough to maintain receiver timing synchronization -- a function otherwise provided by scrambling or bit-stuffing. Though more complex than Manchester, the spectral overhead of 8b10b is only 20% vs 50% for Manchester.

The 8b10b code sees widespread use in modern digital interfaces and storage from the old DAT (Digital Audio Tape) through Serial ATA, Firewire 800, USB 3.0, PCI Express 2.0 and the HDMI/DVI digital video interface and many more.

In 8b10b coding, each 8-bit data byte is divided



8b10b coding block diagram

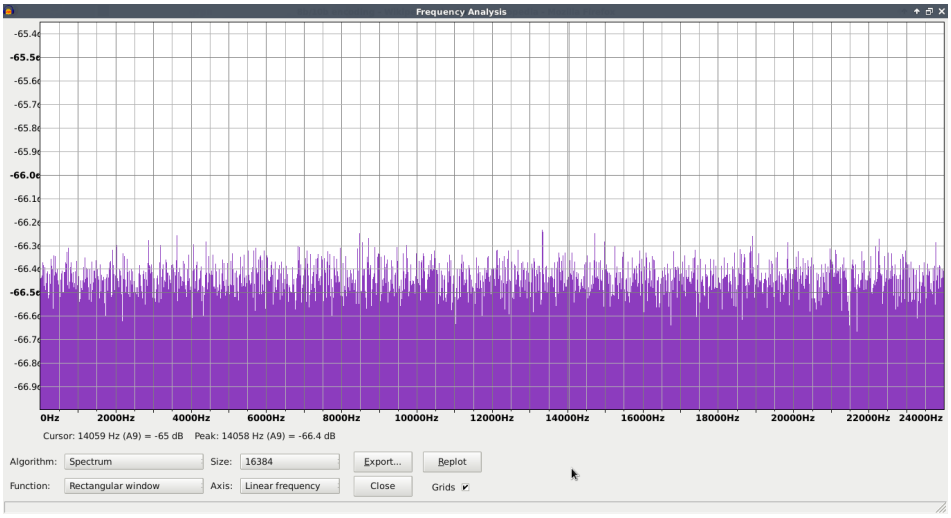
into 5- and 3-bit components that become 4-bit and 6-bit channel sequences. About half of the component values are assigned a single channel sequence while the rest are assigned two, with the one chosen to meet the overall balance and run-length rules. This is done with the help of a 1-bit Runlength Disparity (RD) memory bit.

The effect of 8b10b coding on the spectrum of a random bit stream can be seen in the following figures. Produced with Audacity, the first is of a random bit stream. The second is of an 8b10b encoded random bit stream. The sample rate is defined at 48kHz. Therefore, the Nyquist rate is 24kHz. Because the data is taken at the sampling rate for the random bit stream, the spectrum appears flat. Had it been stretched to some number of samples per data bit to produce an NRZ waveform, then a sinc-shaped spectrum would emerge.

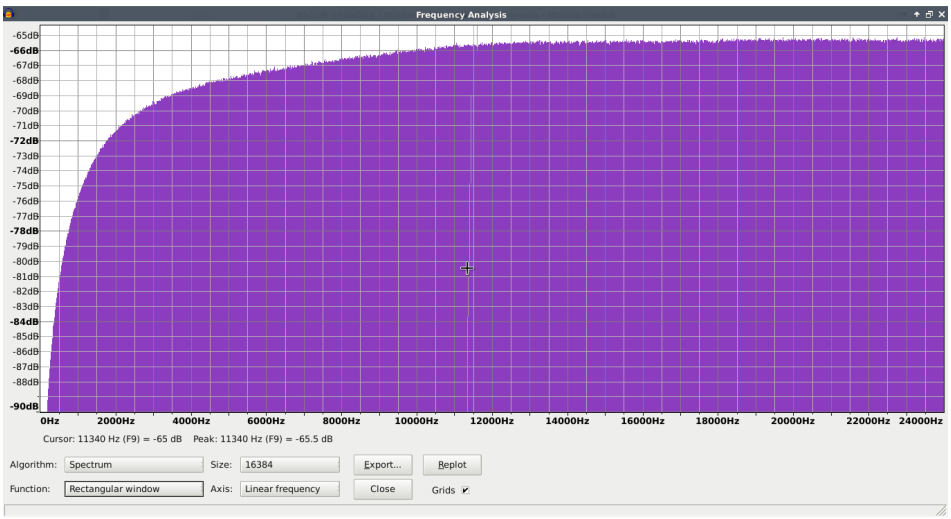
Testing the All-Zeroes Case

It's generally a good idea to check the spectrum of the signal when all-zeroes are transmitted in order to test whether or not there are spectral problems. All-zeroes are often sent when there is no data to transmit. Since this is a data pattern that an operator is likely to encounter, there was interest in whether or not an operator could hear or be bothered by the sound of the telemetry signal.

Because the codeword repeats every 10 samples, the spectral peaks repeat every Fs/10. In this simulation, Fs = 48kHz. As expected, the peaks are every 4.8kHz with no DC component.



unfiltered NRZ spectrum, random data



unfiltered 8b10b spectrum, random data

Decoding 8b10b

Encoding 8b10b is straightforward, as it is accomplished with lookup tables. Decoding it is somewhat trickier, and there doesn't seem to be much in the literature about how to do it well.

One way is to build an inverse lookup table to map each 10-bit channel sequence back to a 8-bit data sequence. Invalid 10-bit sequences are flagged as 'erasures' to assist the Reed-Solomon decoder. Helping the Reed-Solomon decoder in this way, by giving it the location of the error, grants us a significant advantage.

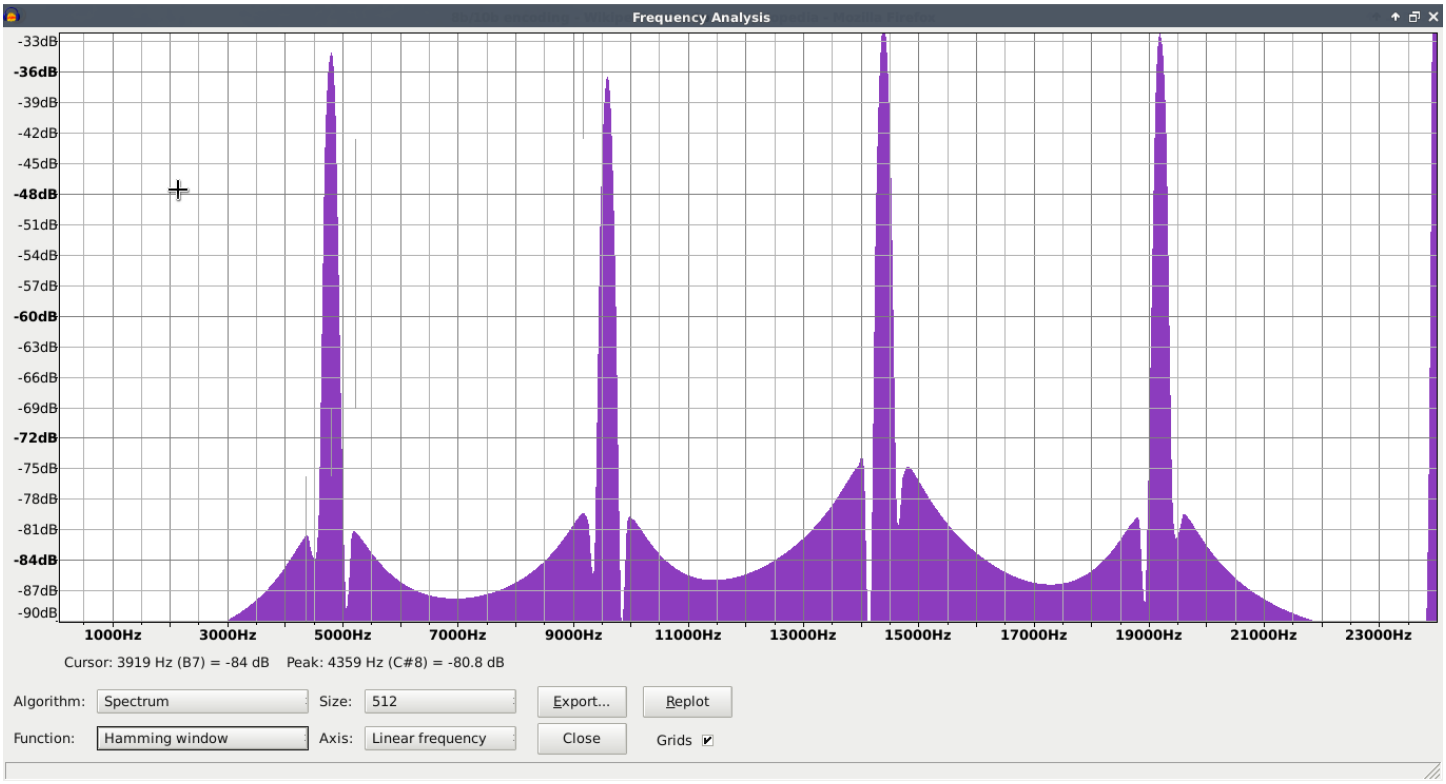
When the error locations in a Reed-Solomon codeword are unknown, the decoder can correct up to half as many symbols in error as there are parity symbols. For our (255,223) code with 32 parity bytes, we can correct up to 16 bytes in error anywhere

in the codeword. But we can do better when at least some of the error locations are known. Up to twice as better, in fact. If we know where every error is located, we can correct as many errors as we have parity symbols. But, that leaves no additional detection margin. We are at the limit of detection, and have to be certain that none remain.

This works, but we can do even better. Recall that some 8-bit data values have more than one 10-bit channel sequence. The encoder can't just pick at random, though, it must use the one that meets the bit-balance rules. That's the reason for the runlength disparity control block in the encoder. It has one bit of memory that keeps track of which way the balance needs to be guided in order to balance the number of zeroes and ones. The receiver can detect more errors, and better help the RS decoder, by rejecting those 10-bit sequences that cannot legally follow the previous 10-bit sequence because they would violate the bit-balance rules.

Since we don't know in advance which codeword the runlength disparity control block chose during the encoding, the best method is to try both branches and see which decoded stream complies with bit-balancing. In other words, we test both codewords and go with the branch that has better statistics.

This is essentially what the Viterbi decoding algorithm does. Although best known for convolutional decoding, the Viterbi algorithm can be used to deduce the most likely hidden internal state of a system from its externally visible clues, even when noise corrupts some of those clues. Because the 8b10b encoder has only one bit of hidden state (the RD bit), compared with 7 or more in a convolutional encoder, this is actually pretty easy. Tentatively decode a series of 10-bit channel words, first by assuming the RD bit starts at +1 and again assuming it starts at -1. See which assumption produces a greater number of successful decodings. With that assumption, commit the oldest word in the sequence, slide the observation window one word forward, and repeat the process.



unfiltered 8b10b spectrum, all 0's data

Control words and synchronization

The 8b10b code provides another useful feature: control words. Because 10 bits are used to encode only 8 data bits, usable 10-bit patterns are left over that can be used as out-of-band control words. Up to 12 of these control words can be used, but by carefully restricting their use some very desirable properties can be had. One is the “comma symbol”, a unique sequence that cannot be found in any stream of encoded bits, even when word framing is unknown, except when a certain control word is sent. This word is used after each Reed-Solomon frame to provide a way for the receiver to synchronize to frame boundaries.

It can also be used to synchronize the receiver to 8b10b boundaries (since frame sync implies word sync) but an easier way is to examine a sliding window of channel bits looking for the phasing that results in the fewest number of invalid (erased) data bytes.

[1] GF stands for Galois Field, a finite collection of mathematical symbols or objects that is closed under addition and multiplication. These fields enable computations in many different applications. Named for Évariste Galois, a mathematician 1811-1832. He died in a duel.

Contact infomation:

Phil Karn KA9Q	karn@ka9q.net
Paul Williamson KB5MU	kb5mu@amsat.org
Michelle Thompson W5NYV	w5nyv@yahoo.com