An Introduction to Multi-Frequency Shift Keying (MFSK)

20.11.2017

Introduction

Radio Amateurs wishing to transmit data or text (digital modes) rather than voice, are often interested in communicating over very long distances, such as from one side of the world to the other. To do this effectively, the sensitivity of the chosen mode, and its ability to handle propagation problems are important factors. Because the bands are a shared resource, bandwidth needs to be kept to a minimum, and ideally power requirements should be modest. Fortunately, when considering conversational operation (chat modes), transmission speeds can often also be modest, which is helpful since speed can be traded off for improved reliability, lower power, or narrower bandwidth.

DX Conversation Modes
The drive to improve the performance of low power long distance links has led to significant improvements in chat modes over the last 10 years or so. We have long since moved on from Radio Teletype! Hellschreiber has been revived, and has proved to be simple and effective. Newer techniques such as PSK31, designed by Peter G3PLX, have extended the performance of narrow-band links considerably. PSK31 has deservedly gained a strong following.

PSK31 use differential binary PSK (DBPSK) modulation, since DBPSK offers very high sensitivity and rejection of noise, and the bandwidth requirement is small. PSK31 is therefore ideal for low power. However, the problems facing very long distance (DX) communication on HF include selective fading and ionospheric modulation of the signal, and in general PSK modes do not handle these problems very well.

On lower frequencies another challenge predominates, a combination of high levels of impulse noise and strong multi-path (NVIS) propagation. PSK modes use a symbol rate the same as the bit rate, and high baud rates lead to sensitivity to multi-path timing errors. MFSK modes, with much lower symbol rate are, if designed well, a perfect medium for conquering these problems.

The MFSK Option
It is hard to know why MFSK was not considered for Amateur use earlier than it was. Perhaps Amateurs associated MFSK with old-fashioned but complex diplomatic services, and inexpensive Amateur MFSK certainly wasn't practical before the arrival of the PC sound card. MFSK was used successfully by the British Foreign Office, the Belgian and French military and others, using such systems as Piccolo and Coquelet. However, as has been clearly demonstrated by other recent revivals (Hellschreiber is a good example), old ideas combined with modern techniques such as DSP can prove to be very effective.

Early MFSK systems were designed for high communications reliability in the days of electromechanical equipment, at a time when digital signal processing and forward error correction, as we know them today, were not practical. These expensive MFSK systems provided very good performance for the time - robust, sensitive and reliable, with good results in fading and poor ionospheric conditions, without requiring error correction. There are some military MFSK systems of a broadly similar nature still used for these reasons (MIL STD 188-141A ALE for example).

With the widespread availability of fast computers and sound cards, the opportunity arose in the late 1990's to revisit and modernise the MFSK technique, to create new high performance modes with the advantages of MFSK, the ease of use of the PC and sound card, and the advantages of sophisticated DSP techniques.

MFSK16
The first of these modes was **MFSK16**, designed by Murray ZL1BPU, first coded by Nino IZ8BLY, and released in December 1999 ('Stream' software). Many ground-breaking techniques first appeared in this software. MFSK16 has full-time error correction, was designed for long-path DX, and is still arguably the best mode for this role today.

No mode is perfect, and some of the limitations of MFSK16 led to further study, and the development of a related but much improved mode, specifically for the lower bands. The new mode deployed Offset Incremental FSK coding (IFK+) for the first time in a chat mode, and the improved robustness achieved allowed the mode to operate successfully without error correction. It is also drift-proof and much simpler to use than MFSK16.

DominoEX
This new mode, christened **DominoEX**, was designed by Murray ZL1BPU and first coded by Con ZL2AFP in 2004 ('ZL2AFP DominoEX' software). In addition to the IFK+ modulation, DominoEX also introduced many other new techniques. One of these is an unusual 'Nibble Varicode' alphabet, even more efficient, which allows the most commonly used characters to be sent in just one symbol. The Nibble Varicode also allows for two Extended ASCII character sets, one for normal typing, the other for transmitting ID messages while the keyboard is idle.

The IFK+ modulation allows the receiver to be very tolerant of tuning, and will easily handle drift and poor tuning which would make reception difficult in other narrow-band modes. Sensitivity, typing speed and robustness on the low bands are all markedly improved.

Other MFSK Modes
More recently, three further MFSK modes of note have been developed. **Olivia** by Pawel SP9VRC uses a Hadamard transform for text coding (like MT63) and then uses a Walsh function to map the data to MFSK tones (again similar to MT63, but MFSK instead of OFDM). There are typically 32 or more tones, and there are many confusing options. Olivia is notable for extreme typing delays and rather slow typing speed for a given bandwidth, but it is quite robust.

THOR, by Dave W1HKJ (in 'FLDIGI' software), is an FEC adaptation of DominoEX. It uses the DominoEX IFK+ modulation system, but employs the MFSK16 binary varicode and FEC coding. The use of punctured codes based on noise-burst overload, detector anomalies and code violations, and associated soft decision decoding, make this a very effective mode in noisy situations.

EXChat is a version of DominoEX that operates in 'Sentence Mode', which makes operating quick and conversations easy. You simply type a sentence, then press ENTER and it is transmitted, much like phone texting.

FSQ (Fast Simple QSO). FSQCall is a new Chat-mode design which is very robust and very efficient. It uses 33 tones and a special alphabet which allows most characters to be sent as just one tone. It operates in sentence mode, and can transmit at up to 60 WPM, yet occupies only 5Hz/WPM. FSQ also carries a special Selective Calling and Net Management application called FSQCALL.

WSQ (Weak Signal QSO). WSQCall is a variation of FSQCall, designed specifically for LF/MF weak signal operation. Like FSQ, it uses 33 tones and a special alphabet which allows most characters to be sent as just one tone. However it uses much less bandwidth (50 Hz) which gives it reliable reception at only -25 dB SNR. It operates in sentence mode, transmits at 5 WPM, yet occupies only 50 Hz bandwidth. WSQCall also carries the same Selective Calling and Net Management features as FSQCALL.

The final MFSK mode to be mentioned is **MEPT_JT**, an idea proposed by Murray ZL1BPU and developed by Joe K1JT (his 'WSPR' software). MEPT_JT is not primarily a QSO mode - it is intended for propagation monitoring. The simple fixed message (call, location and power) has error correction added, and is convolved with a PN sequence. A very slow 4-tone MFSK modulation scheme is used, each transmission taking two minutes. The receiver can decode several stations at once, and the sensitivity of this system is remarkable: almost -30dB S/N! The receiver software automatically posts 'spots' on a world-wide database via the internet.

MFSK Overview for Beginners

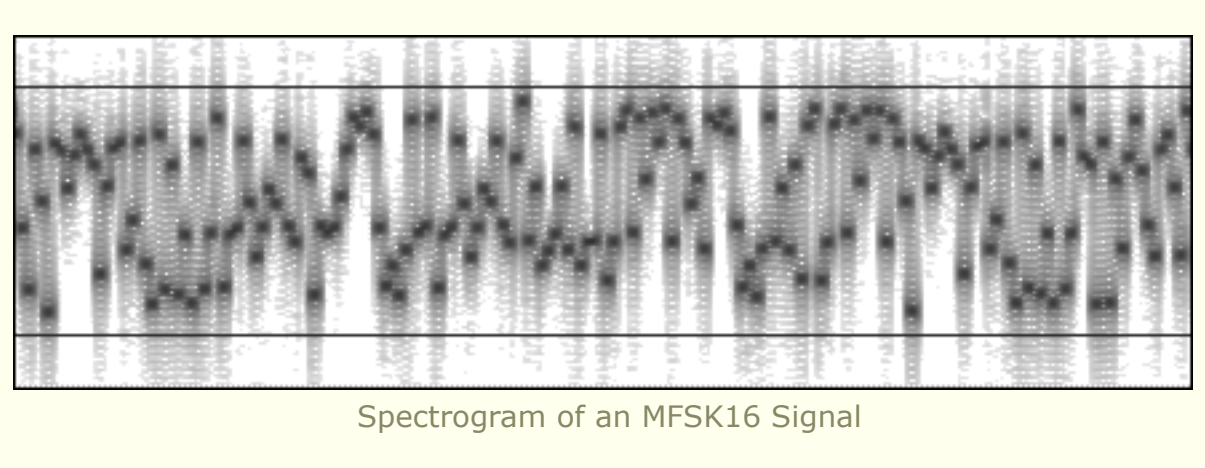
What follows is an introduction to MFSK modes and how they work. MFSK is a technique for transmitting digital data using multiple tones, extending the RTTY two-tone technique to many tones, usually, but not always, one tone at a time.

MFSK means **Multi - Frequency Shift Keying**, and should not be confused with MSK (Minimum Shift Keying). There are a number of different techniques, but we will consider only systems which send a single tone at a time, i.e. a sequential system. Piccolo and Coquelet, although they use tone pairs, are still sequential.

> MFSK transmissions have a unique sound, almost musical, which is maybe why Piccolo was given its name. You can play this sample of an MFSK16 transmission (183kB) by clicking on the speaker logo.

The transmission changes from one tone to the next smoothly, with no sudden change in phase, and no change in amplitude. MFSK also uses relatively narrow tone spacings, and each tone carries multiple data bits, so remarkable data rates are achieved for a given bandwidth. The following picture is a spectrogram of an MFSK16 signal (16 carriers) with a spacing of 15.625 Hz and operating at 15.625 baud (symbols per second). Each tone represents four bits, so transmission operates at 62.5 bps and occupies about 316 Hz bandwidth.

The two thin horizontal lines in the picture are added 1000 Hz and 1300 Hz markers, and the dots are the transmitted tones. The picture width represents about 7 seconds. This short transmission contains about 120 letters.



Spectrogram of an MFSK16 Signal

Advantages of MFSK

- High rejection of pulse and broadband noise due to narrow receiver bandwidth per tone
- Low baud rate for sensitivity and multi-path rejection - data bit rate higher than symbol baud rate
- Tolerance of ionospheric effects such as Doppler, fading and multi-path
- Constant transmitter power - phase continuous for minimum keying noise
- No need for a linear transmitter

Most important of all, with an MFSK system, *the error rate reduces as the number of tones is increased*. With PSK systems the exact opposite is true.

Disadvantages of MFSK

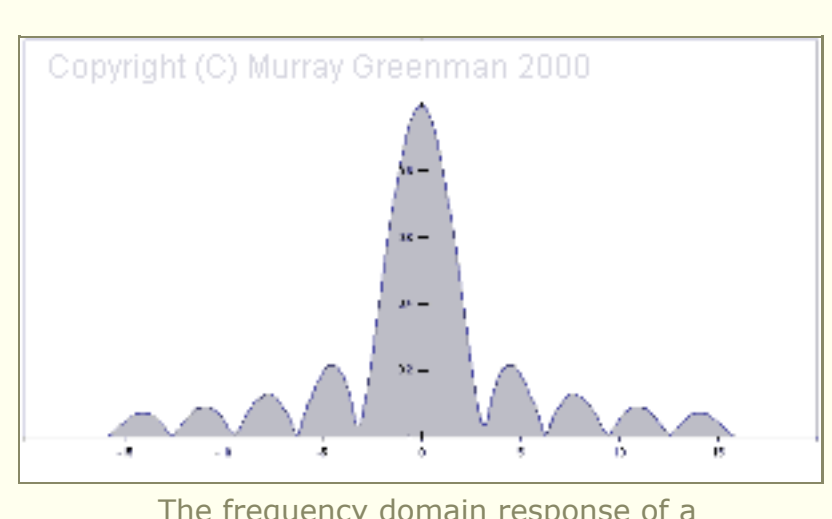
- Narrow spacing and narrow bandwidth of the individual tone detectors can make tuning difficult
- Good transceiver stability and low TX/RX offset are important (problem is solved in DominoEX)
- MFSK also uses more bandwidth for a given text speed than a PSK system, but is more robust.

How MFSK Works

MFSK is a system where individual pulses of different radio frequencies carry information, and the frequency of the pulse defines the data carried. This is the same as FSK, for example radio teletype (RTTY), but rather than two different frequencies, many more are used - from four to 64, for example. MFSK16 uses 16 tones, DominoEX uses 18. Unlike Morse code dots, the tones follow each other without pause, on slightly different frequencies.

Transmission
MFSK systems generally space the tones as closely as possible, to restrict the transmitted bandwidth. The tones must be spaced at a separation equivalent to the baud rate, or a multiple of the baud rate - the rate at which the 'dots' are sent - otherwise it is difficult to separate one tone from another at the receiver. This allows the signalling to be *orthogonal*, (each tone can be separately detected without being influenced by another tone, or affecting the result of any other detector) as will be explained over the next few paragraphs.

If an FSK or MFSK transmission is 'hard keyed', i.e. each different tone starts and stops suddenly, it causes undesirable keying sidebands. A second tone coming on when the first goes off, with no thought to the phase relationship, is just like two separate on-off keyed signals, and wide keying sidebands are generated by both tones. Any train of tone bursts gives the signal its characteristic frequency domain sin(x)/x shape (shown below for one tone):



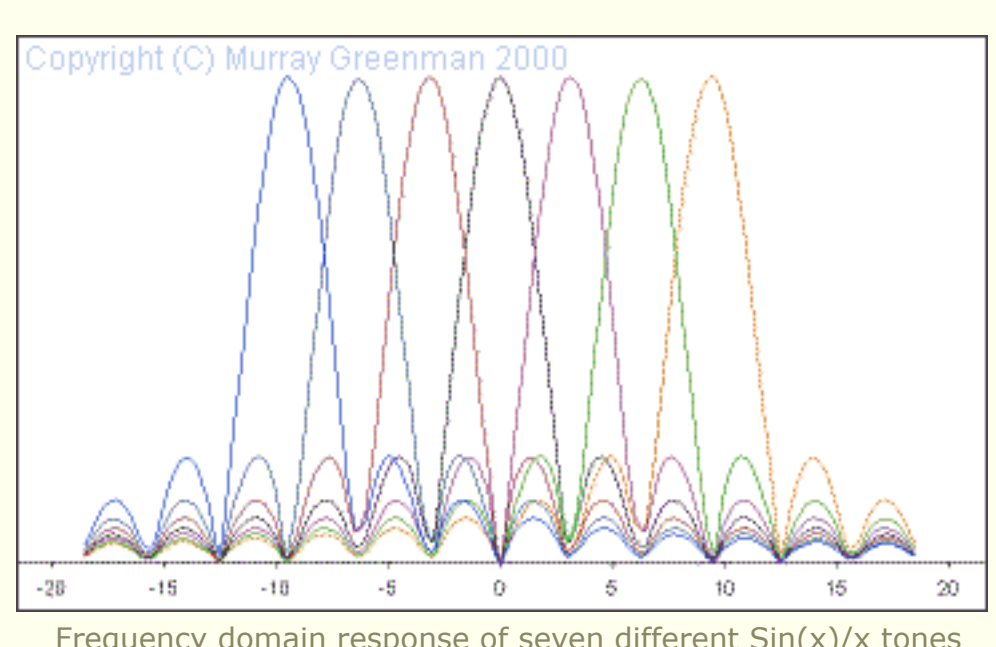
The frequency domain response of a
hard keyed single tone.

It is important therefore to manage the change in frequency in an MFSK system *without* instantaneously changing the phase. Fortunately with sound card software this is easy to do.

The shape of the transmitted signal has a main peak, with nulls spaced either side of the carrier frequency. The first nulls occur at the carrier frequency ± the baud rate. The humps and nulls are clearly visible on either side in the picture above. The big hump in the centre is the wanted signal, and it is these that cause the black dots in the spectrogram further up the page. If you look carefully at the spectrogram, you will also see the side lobes as grey streaks above and below the individual dots.

Of course the dots or tone bursts are not isolated, but preceded and followed immediately by other dots at the same or different close frequencies. We can superimpose the sin(x)/x shape of each one to see what happens, and then arrange the spacing of the tones to achieve the best results.

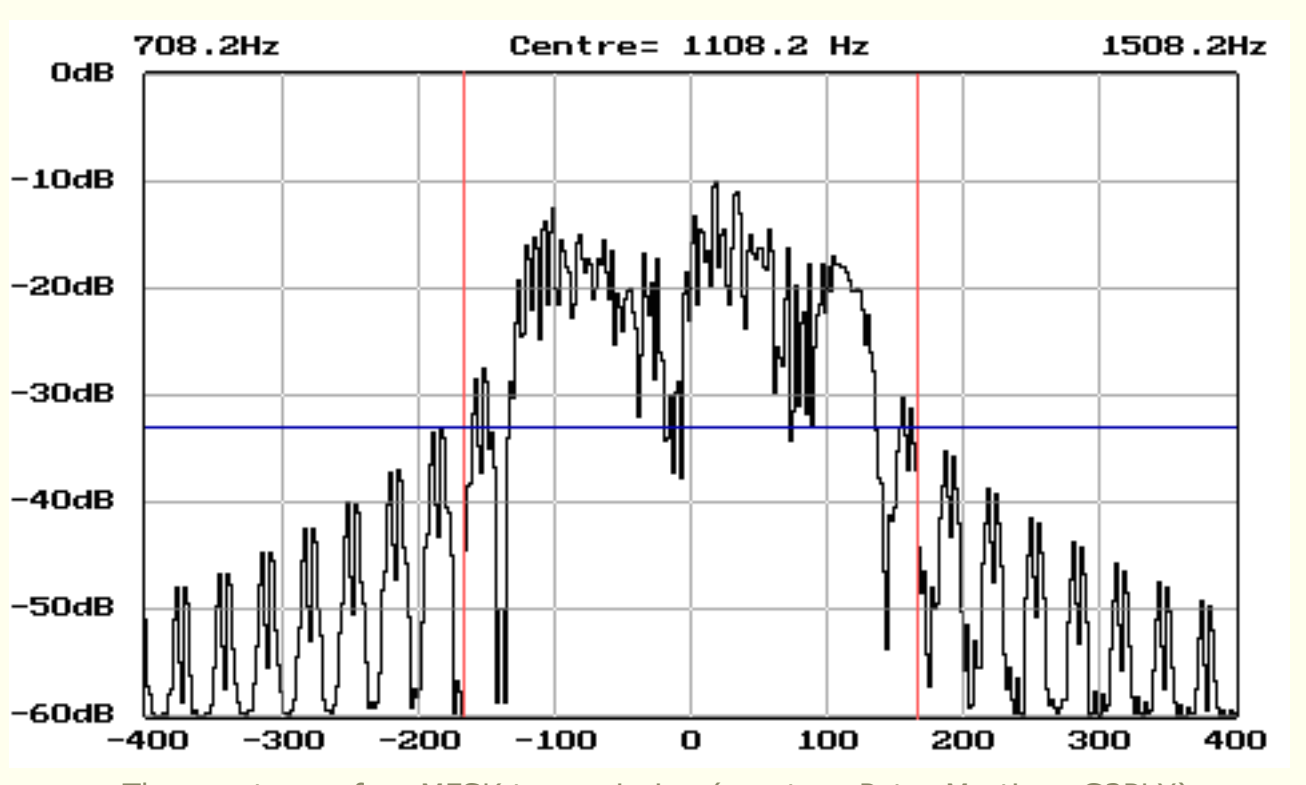
In the next illustration, seven of these tone spectra are superimposed, so that the nulls of each carrier coincide with the peak of the next, to minimise cross-talk in the receiver (energy intended for one receiver channel appearing in the next channel), and therefore allow orthogonal signalling. This occurs *when the symbol period (1/baud rate) and tone spacing are reciprocal*, i.e. the spacing is 1/symbol period or n/symbol period, where n is a small integer.



Frequency domain response of seven different Sin(x)/x tones

When the transmission consists of multiple tones spaced as described, the long-term average signal broadens out across the peak, but retains the characteristic shape, as illustrated above.

The following image shows the spectrum of a real experimental 8FSK signal, transmitting random data at 31.25 baud with a tone spacing of 31.25 Hz. The vertical axis in this image is logarithmic, so the side lobes are more obvious than in the simulation above. Note that the sidelobes are spaced 31.25 Hz because of the 31.25 Hz baud rate. The broad centre peak is a little ragged, because the data does not average out in the time of the spectrogram recording.

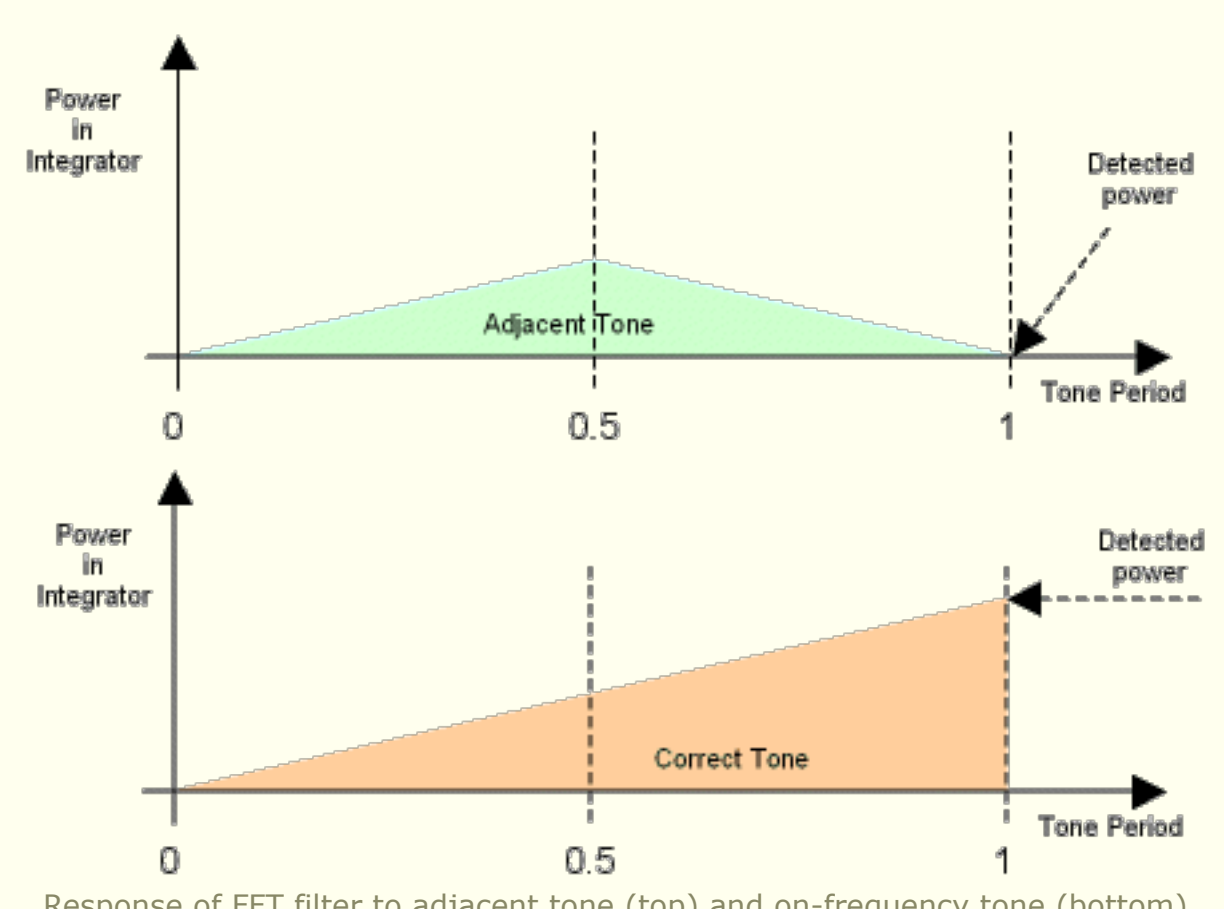


The spectrum of an MFSK transmission (courtesy Peter Martinez G3PLX)

The spectrogram was taken with 0dB set at the level of a single constant tone. A standard method of calculating the necessary bandwidth of radio transmissions (defined by mode, number of tones and symbol rate) is laid out by the CCIR, and for the above transmission is 331.25 Hz (±166 Hz, the red lines). Looking at the spectrum, the signal is well below -30dB from the single tone carrier at this bandwidth (indicated by the blue line), easily exceeding the CCIR definition of <0.5% of the total transmitter power (about -20dB). The performance is this good because the signal is transmitted using phase-synchronous tones (CPFSK).

Reception
Because the tone spacing and baud rate match, the receiver detector filters (consisting of overlapping Discrete Fourier Transforms operated at an exact multiple of the baud rate) are able to independently decode each tone without energy from one tone appearing in the detector intended for the next tone (a property called *orthogonality*). The narrowest spacing at which this independence can occur is when the symbol rate and tone spacing are the equivalent, and the spacing must in any case be an integer multiple of the symbol rate (with DominoEX8 for example, the spacing is 15.625Hz, twice the 7.8125 baud rate).

This behaviour happens because at the filter frequency the adjacent tones have an uneven number of tone half-cycles in the detection period, so the phase changes continually during the tone integration period, and the adjacent tone energy sums to zero at the exact duration of the tone. The correct tone, however, has an even number of half-cycles, no change of phase, and the energy accumulates to a maximum over the tone detector duration (the detector uses a technique called 'integrate and dump', and operates synchronously with the received symbols, using sync recovered from the symbol-rate energy properties of the signal).



Response of FFT filter to adjacent tone (top) and on-frequency tone (bottom)

Essentially the filter for each tone has a bandwidth of less than half the tone spacing, and in addition to rejecting adjacent tone energy, is able thereby to achieve excellent signal to noise ratio and good rejection of multi-path Doppler energy. The overlapping DFT technique with symbol-synchronous 'integrate and dump' closely mimics the behaviour of the complex discrete component filters used by Piccolo, or electromechanical reed filters of Coquelet, and provides dozens of drift-free correctly spaced filters in just one mathematical step.

Forward Error Correction

Forward Error Correction (FEC) can be added to many digital modes, in order to reduce the number of errors in text. This is achieved by sending extra coded data which helps the receiver to correctly decode what was sent, despite receiving errors. Since MFSK is inherently reliable, the addition of FEC makes the system extraordinarily robust. It also transpires that there are very elegant FEC solutions ideally suited to the features of MFSK.

Convolutional Coding
A popular method of coding for FEC is to multiply the data in a binary bit-stream with a special polynomial algorithm (a process called *convolutional coding*), which typically gives two coded bits to be transmitted for every incoming data bit. This of course doubles the required data rate and bandwidth, or halves the typing speed.

However, the decoding process is not so simple, as it requires some clever logic, the Viterbi decoder. Since this is much more than a beginner can be expected to understand, it won't be explained here. Chip Fleming has an interesting Tutorial on Convolutional Coding if you would like to learn more.

Interleaving
One of the problems with FEC coding is that it works best if all the errors are spread out evenly. Unfortunately interference (especially static and splatter) is burst-type noise, and tends to take out data several bits at a time, which makes life very difficult for the decoder. To avoid this problem, we muddle up the order of the transmitted bits using an Interleaver. Then, when bursts destroy adjacent bits, the process of de-interleaving spreads the errors out, easing the job of the decoder.

One really cool thing about MFSK16 is that the order of the FEC data bits and the placement of bits in the interleaved data is always known, since the bits are received in groups of four (from one symbol), always with the same position in the group (weight). This means that the FEC decoder bit assignment and Interleaver un-muddling become virtually automatic. The transmitter does not need to waste bandwidth transmitting Interleaver synchronizing information, and in addition the receiver does not have to work out how to arrange the incoming bits, as the bit weights infer the positions.

Avoiding FEC
The cost of using an FEC system is that you have only half the potential text throughput, and in addition you have to wait around for the Interleaver and De-interleaver to do their thing (we call this latency). Latency is a nuisance, as it damages the 'slickness' of the mode (waiting time between overs). However, if the MFSK mode is designed very carefully, it is quite possible to operate MFSK *without* FEC. The design must minimize the effects of ionospheric propagation timing and frequency changes, which cause one received tone element to interfere with another, or arrive at the wrong frequency or time and thus be misinterpreted (we call this Inter-Symbol Interference, or ISI).

This problem was avoided with Coquelet, at the cost of extra bandwidth, by using two completely separate tone sets alternately. This level of robustness is achieved more efficiently in DominoEX using just 18 tones and IFK+ tone management. The benefits of faster typing speed and low latency are achieved, but the user has to accept a few errors in the received text.

Bits and Bauds

One of the most confusing things about MFSK is that the signalling rate is not the same as the data rate, because each tone carries more than one data bit. To explain this, we'll define all the terms used and show how they inter-relate.

Symbol Rate
The basic element of transmission in any data mode is the *Symbol*. In most modes, each symbol implies a '0' or '1', but in MFSK systems, each symbol carries information according to how many tones there are - three bits of information for 8 tones, four bits for 16 tones, and so on. Each MFSK tone burst is one symbol. The symbol rate is always measured in *baud* (symbols/second), the reciprocal of the duration of the symbol.

Channel Data Rate
The data carried by the MFSK tones is inevitably coded in some way so that the 'raw data' rate may not be the same as the user input or output data rate. However, the Channel Data Rate is always the number of bits per symbol x the Symbol Rate. The channel data rate is measured in bits/second (bps). For example, for a 10 baud 8FSK mode (8 tone FSK) there are three data bits per symbol, so the raw Channel Data Rate is 3 bits x 10 baud = 30 bps.

User Data Rate
Very often data is coded using an FEC system designed to reduce errors that occur due to the transmission path. For MFSK systems the most appropriate type of FEC is the rate $^1/_2$ sequential convolutional type, where every user data bit is represented in the transmission by two (sometimes more) coded data bits. This ratio is the *Coding Rate* of the coder. For example, if there are two coded bits for every one data bit, the Coding Rate = $^1/_2$. Sometimes there might be four bits, but only three are sent, when Coding Rate would be = $^2/_3$. Thus the User Data Rate is the Channel Data Rate x Coding Rate.

Alphabet Coding
There are many ways to encode the alphabet from the keyboard for transmission. Perhaps the most common now is ASCII (ITA-5), but ITA-2 (as used by teleprinters) is also widely used. MFSK16 and THOR, like Morse and PSK31, are based on a binary Varicode, which, unlike most alphabets, assigns a different number of bits to different characters, so that more frequently used characters have fewer bits and are therefore sent faster.

DominoEX is most unusual, in that although it uses a Varicode, the design is based on four-bit groups (a 'nibble'). Of course four bit groups are easy to transmit with an MFSK system. Maximum efficiency is achieved by assigning the nibbles so that the most common characters can be sent in just one nibble, and it turns out that most of the alphabet can be sent in just two nibbles. The design allows for three or more nibbles per character, and so allows a huge character set.

The number of bits per alphabet character therefore depends on the character frequency, just like Morse. For example:

Character	MFSK16	DominoEX	FSQ	Morse
space	100	0	0	
a	101100	4	1	·–
e	1100	1	5	·
E	111011100	3,8	5,29	·
Z	101010110100	7,9	26,29	––··

Thus, the alphabet coding performance depends on the chosen code, and when using a Varicode, the performance also depends on the text sent (so you can't give a precise number of bits per character, just an approximation):

Alphabet	Bits/Char
ITA-5 ASCII	10
ITA-2	7.5
Varicode	~ 7-8

In addition to being efficient for sending text, a Varicoded alphabet is essentially infinitely expandable. For example, in MFSK16 all the European accented characters are defined, and others have been added for control purposes, that are outside the character set. The MFSK16 varicode is not the same as the PSK31 varicode, although the technique is similar. For DominoEX the varicode is expanded to provide two complete characters sets, one for keyboard typing, the other for automatic messages, such as ID.

Another important advantage of using a varicode is that the stream of data can be much more quickly resynchronised in case of errors, than is possible with other systems (such as ASCII), and so a minimum of data is lost.

Text Throughput
The user is most interested in the actual usable text throughput, which is specified in characters per second (CPS) or words per minute (WPM). Both depend on the alphabet used, and the number of words per minute depends on the average word size. In English this is taken for convenience to be five letters plus a letter space. So we can say that:

Text Throughput (CPS) = User Data Rate / Alphabet Bits per Character

Text Throughput (WPM) = CPS x 60 / letters per word

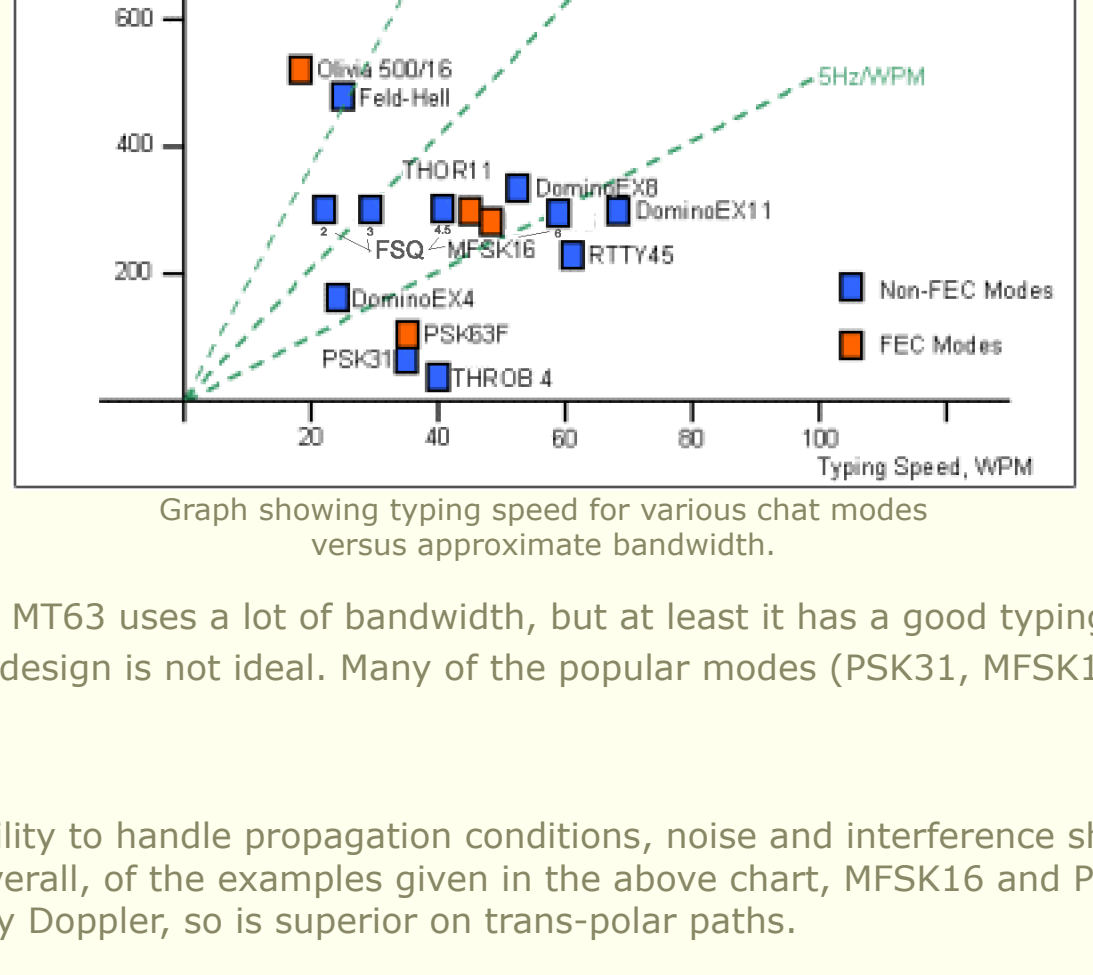
Worked Example
Say we are using an MFSK system with 16 tones (16FSK), operating at 15.625 baud with FEC Rate = 1/2, and an ASCII alphabet using 10 bits/character. Then:

```
Symbol Rate           = 15.625 baud
Channel Data Rate     = 15.625 x log216 = 15.625 x 4        = 62.5   bps
User Data Rate        = 62.5 x 1/2 (FEC RATE)               = 31.25  bps
Text Throughput (CPS) = 31.25 / 10 CPS                      =  3.125 CPS
Text Throughput (WPM) = 31.25 x 60 / (10 x 6)               = 31.25  WPM
```

This will take place in a bandwidth little more than 16 x 15.625 = 250 Hz.

Comparisons

Bandwidth
One useful way to assess digital modes is to consider the bandwidth efficiency, i.e. how much bandwidth is required for a given typing speed. A well designed MFSK mode should have reasonably good bandwidth efficiency. If this is not the case, then the mode has clearly been engineered for some other advantage at the expense of bandwidth or speed. In general those mode with FEC are less bandwidth efficient.



Graph showing typing speed for various chat modes
versus approximate bandwidth.

From the graph (which shows most of the popular chat modes), it is clear that MT63 uses a lot of bandwidth, but at least it has a good typing speed. Olivia has high bandwidth and very slow typing. This is the price paid for strong error correction, which is in turn required because the modem design is not ideal. Many of the popular modes (PSK31, MFSK16, DominoEX, PSK63F) achieve quite good bandwidth efficiency, 5 $^{Hz}/_{WPM}$ or better.

Sensitivity and Robustness
Of course, bandwidth is just one way to assess modes. The sensitivity, and ability to handle propagation conditions, noise and interference should also be considered. This was done as part of the design and verification for MFSK16 and DominoEX (for other modes this has not always been so!). Overall, of the examples given in the above chart, MFSK16 and PSK31 are considered the best for DX QSOs. PSK31 often performs poorly on long paths. MFSK16 is slightly more sensitive than PSK31, and is unaffected by Doppler, so is superior on trans-polar paths.

On the lower bands, (160/80/40m) DominoEX excels, being best equipped to handle NVIS propagation. It is also very effective on LF and VHF, because of higher sensitivity at low baud rate and (important on VHF) high drift tolerance. When lightning or severe interference is the problem, THOR is a good choice. MFSK modes use 100% transmitter duty cycle, so if battery power is limited, use Morse or Hellschreiber rather than an MFSK mode, as both have low duty cycle. These suggestions are supported by practical exerience and ionospheric simulation tests.

Summary - Amateur MFSK Modes

The first serious Amateur Radio MFSK modes appeared in 2000 - MFSK16 (and slower but more robust MFSK8). Using test software by Nino IZ8BLY, initial QSOs were held 18 June 2000. The first QSO was from ZL1BPU - IZ8BLY on 18.105 MHz, and the second ZL1AN - ZL1BPU on 3.560 MHz. DominoEX was released in 2004 and made a dramatic difference to ease of operation. While slightly inferior to MFSK16 in sensitivity and accuracy on the higher bands, the ease of tuning, low band performance and slicker operation are worthwhile advantages.

Many other MFSK modes have appeared since MFSK16 (listed in approximate chronological order):

- JASON by Alberto I2PHD. Designed for LF QSO, but very slow. 17 tones, ASCII coding.
- THROB by Lionel G3PPT (which uses concurrent tone pairs and no FEC).
- DominoEX, designed by Murray ZL1BPU and first coded by Con ZL2AFP (18-FSK with IFK+ coding).
- Olivia by Pawel SP9VRC (wideband, 16 to 64 tones and Walsh-Hadamard FEC).
- A full implementation of ALE (MIL STD 188-141A, wide-spaced 8-FSK) by Charles G4GUO.
- MEPT_JT, an extremely sensitive narrow-band propagation test mode by Joe K1JT, 4-FSK with FEC and PN sequence.
- THOR, 18-FSK, a DominoEX extension with FEC, by Dave W1HKJ.
- WSQ (Weak Signal QSO) 33-FSK, IFK+, extremely sensitive weak signal mode for LF/MF QSOs. Designed by Murray ZL1BPU and Con ZL2AFP; coded by Con ZL2AFP.
- EXChat, a text-message (chat) version of DominoEX, by Con ZL2AFP.
- FSQ (Fast Simple QSO) 33-FSK, IFK+, operating at 2 - 6 baud (20 - 60 WPM). Very efficient alphabet. This mode carries a very capable Selective Calling application. Designed by Murray ZL1BPU and Con ZL2AFP; coded by Con ZL2AFP.

Software for all these modes is readily available. Many different programs include MFSK16 and DominoEX.