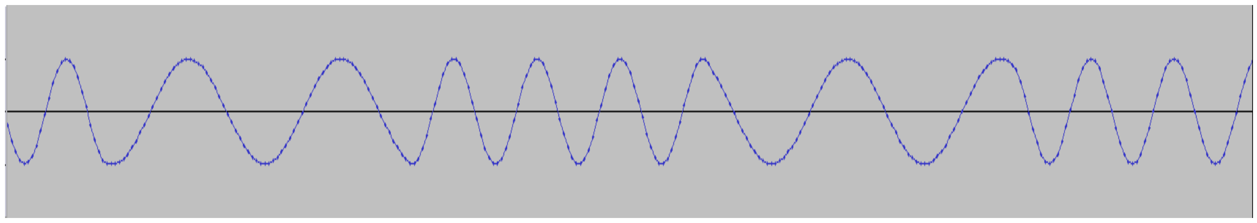
2400 & 4800 bps PSK for APRS / Packet Radio

WB2OSZ, April 2016

1200 BPS

Most APRS / amateur packet radio uses 1200 bit per second (bps) audio frequency shift keying (AFSK) for historical reasons. The binary 1 and 0 bits are conveyed by switching between tones of 1200 and 2200 Hz. It looks like this:



You can see how the audio frequency is changing.

It was cheap and easy to implement using 1970's (or earlier) technology. It has the advantage that it is not very fussy about the audio channel and works with the microphone and speaker connections of any transceiver designed for voice. However, it is not very efficient.

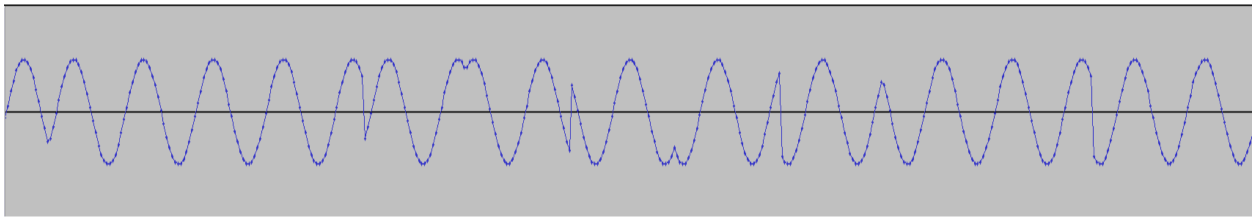
2400 BPS

We can get more bits per second through the same audio channel using more advanced techniques. This is nothing new. Back in the 1990's there were at least three commercial TNCs that allowed 2400 bits per second.

- MFJ-2400 which is an optional board for the MFJ-1270 or MFJ-1274.
- AEA PK232-2400.
- Kantronics KPC-2400.

Were they compatible with each other? I don't know. If not, that might help explain why 2400 bps never gained much popularity.

The first two listed above used the EXAR XR-2123 PSK modem chip which implements the V.26 / Bell 201 standard. Instead of different frequencies, there is a tone with a constant frequency. Information is conveyed by shifting the phase. The audio looks like this:



In this case, we have an audio carrier of 1800 Hz. 1200 times a second, it can switch to 1 of 4 different phases which represent 2 bits at the same time. (There are lots of PSK explanations out there. You know how to use Google.)

The terms "baud" and "bits per second" are often used interchangeably because they are the same number for the most common 1200 & 9600 speeds. Here we must be more careful about terminology.

Bits per second (bps) – obvious meaning.

Symbol – Time period when state of signal doesn't change. For AFSK, there are two states (one of two tones), conveying 1 bit per symbol. Phase shift keying, with 4 phases, can convey 2 bits at a time.

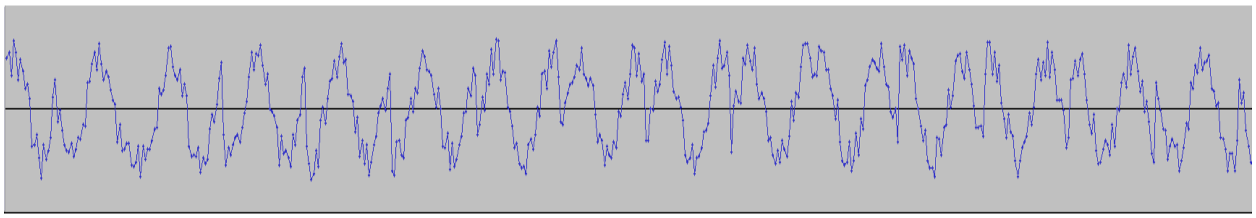
Baud – Number of symbols per second.

In this case, we have a 1200 "baud" signal because the phase can shift up to 1200 times per second. Each "symbol" can be 1 of 4 phases, representing 2 bits, so we end up with 1200 x 2 = 2400 bits per second.

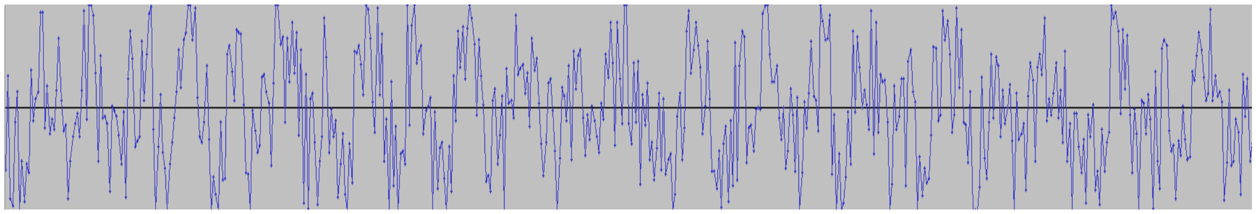
"gen_packets" is a utility to generate APRS frames and put the audio into a file rather than transmitting them over the radio. Here we generate 100 frames at 2400 bps.

```
$ gen_packets -B 2400 -n 100 -o test2400.wav
```

The beginning of the file looks like the waveform above. Increasing levels of random noise are added so we can see how well the demodulator works under adverse conditions. About ¼ from the beginning, we can see the increasing random noise level but the waveform is still recognizable.



Near the end it is unrecognizable:



Dire Wolf has 4 different implementations of V.26 style PSK demodulators using two different techniques, each with and without a bandpass filter in front.

Let's see how well each of the demodulator configurations can handle this.

```
$ atest -B 2400 -P P test2400.wav | grep decoded
59 packets decoded in 1 seconds.

$ atest -B 2400 -P Q test2400.wav | grep decoded
69 packets decoded in 1 seconds.

$ atest -B 2400 -P R test2400.wav | grep decoded
78 packets decoded in 1 seconds.

$ atest -B 2400 -P S test2400.wav | grep decoded
74 packets decoded in 1 seconds.
```

(This is still under development, and there could be more fine tuning, so the exact numbers will probably be different if you play along at home.)

If you had to pick only one, "R" is the best. In this test case. Under different conditions one of the others might do better.

It is also possible to run all of them in parallel and have the duplicates removed.

```
$ atest -B 2400 -P PQRS test2400.wav
…
…
DECODED[1] 0:00.2975 WB2OSZ-15 audio level = 49     ||||
[0.1] WB2OSZ-15>TEST:,The quick brown fox jumps over the lazy dog!
0001 of 0100

DECODED[2] 0:00.5968 WB2OSZ-15 audio level = 49     ||||
[0.1] WB2OSZ-15>TEST:,The quick brown fox jumps over the lazy dog!
0002 of 0100
…
…
DECODED[81] 0:26.8434 WB2OSZ-15 audio level = 93     __|_
[0.2] WB2OSZ-15>TEST:,The quick brown fox jumps over the lazy dog!
0089 of 0100

DECODED[82] 0:27.4490 WB2OSZ-15 audio level = 94     __||
[0.2] WB2OSZ-15>TEST:,The quick brown fox jumps over the lazy dog!
0091 of 0100
```

```
DECODED[83] 0:27.7536 WB2OSZ-15 audio level = 94      __||
[0.2] WB2OSZ-15>TEST:,The quick brown fox jumps over the lazy dog!
0092 of 0100

78 packets decoded in 2 seconds.
```

Notice the “_” and “|” characters at the ends of the lines. These indicate how well each of the demodulators did.

_	means no success
|	means no errors, i.e. correct CRC

Near the beginning, where the signal is clean, we see “||||” meaning all of PQRS decoded the frame. Toward the end, with a high noise level, “__||” indicates only the last two, R & S, were successful.

We have one more trick up our sleeves. There is an option to attempt fixing single bit errors.

```
$ atest -B 2400 -F 1 -P PQRS test2400.wav
…
…
DECODED[87] 0:26.8434 WB2OSZ-15 audio level = 93   [NONE]   __|:
[0.2] WB2OSZ-15>TEST:,The quick brown fox jumps over the lazy dog!
0089 of 0100

DECODED[88] 0:27.4490 WB2OSZ-15 audio level = 94   [NONE]   __||
[0.2] WB2OSZ-15>TEST:,The quick brown fox jumps over the lazy dog!
0091 of 0100

DECODED[89] 0:27.7536 WB2OSZ-15 audio level = 94   [NONE]   __||
[0.2] WB2OSZ-15>TEST:,The quick brown fox jumps over the lazy dog!
0092 of 0100

DECODED[90] 0:29.8678 WB2OSZ-15 audio level = 97   [SINGLE]   __:_
[0.2] WB2OSZ-15>TEST:,The quick brown fox jumps over the lazy dog!
0099 of 0100

86 packets decoded in 2 seconds.
```

Notice the new “:” character.

:	means it was possible to get a valid CRC by changing one bit.
__|:	means the R demodulator was error free. S was able to succeed by changing 1 bit. In this case we use the result from R.
__:_	means the R demodulator was able to get a good CRC by changing 1 bit.

By running multiple demodulators, tuned in different ways, we are able to get more error-free frames than by using any one of them individually.

Running all 4 at once is currently the default for all platforms.

On a Raspberry Pi, model 2, this takes about 35% of one core when decoding realtime from the soundcard input.

On a Raspberry Pi, model 1 B, this takes about 81% of the CPU.

If you are constrained by limited CPU power, you might want to select just one demodulator, probably R based on the observations here. There are two different ways to do this.

- On the command line:

```
direwolf -B 2400 -P R
```

- In the configuration file:

```
MODEM 2400 R
```

Compatibility?

Is the Dire Wolf implementation of 2400 bps PSK compatible with any other TNCs? I don't know. I'll be watching for a suitable old TNC with 2400 bps capability at the next ham radio flea market …

4800 BPS

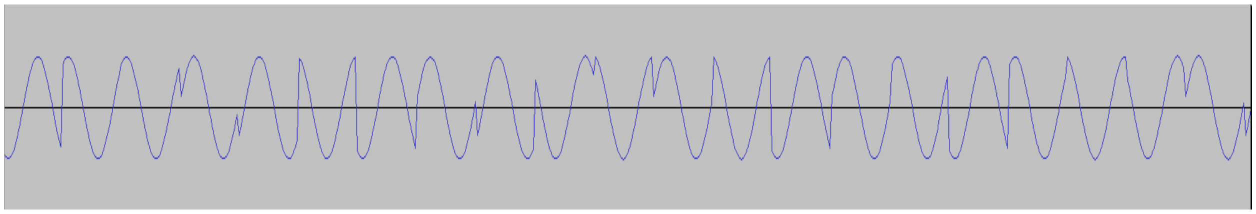
If we can distinguish between 4 different phases, why not go for 8? With a few minor modifications, a V.27 style demodulator was also implemented. This uses the same 1800 Hz audio carrier. It can change phase 1600 times per second so it is 1600 baud. The 8 phases can represent 3 bits at a time so we have 1600 x 3 = 4800 bits per second.

Four times faster than the old fashioned 1200 baud AFSK!

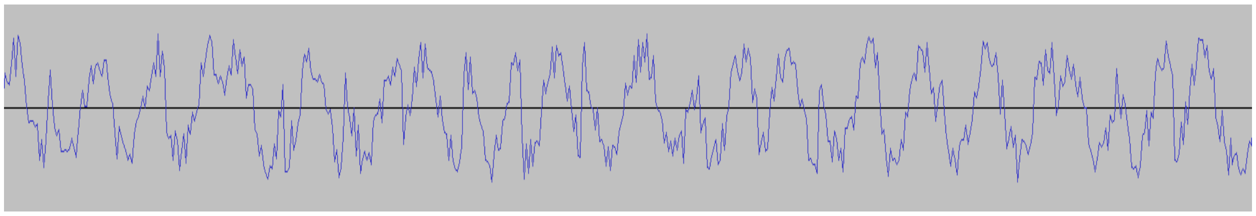
Once again, we will generate a test file with 100 frames and increasing noise.

```
$ gen_packets -B 4800 -n 100 -o test4800.wav
```

Here is a nice clean signal from near the beginning of the file. It's hard to visualize what is going on here but the computer has no problem figuring it out.



Remarkably, we can still decode en error-free frame with this level of noise about 80% from the beginning of the file.



In this case the demodulator configurations are called T, U, V, and W. Let's try them individually and all together.

```
John@zt ~/src/direwolf-dev
$ atest -B 4800 -P T test4800.wav | grep decoded
47 packets decoded in 1 seconds.

John@zt ~/src/direwolf-dev
$ atest -B 4800 -P U test4800.wav | grep decoded
51 packets decoded in 1 seconds.

John@zt ~/src/direwolf-dev
$ atest -B 4800 -P V test4800.wav | grep decoded
69 packets decoded in 0 seconds.

John@zt ~/src/direwolf-dev
$ atest -B 4800 -P W test4800.wav | grep decoded
72 packets decoded in 0 seconds.

John@zt ~/src/direwolf-dev
$ atest -B 4800 -P TUVW test4800.wav | grep decoded
72 packets decoded in 1 seconds.

John@zt ~/src/direwolf-dev
$ atest -B 4800 -F 1 -P TUVW test4800.wav | grep decoded
81 packets decoded in 1 seconds.
```

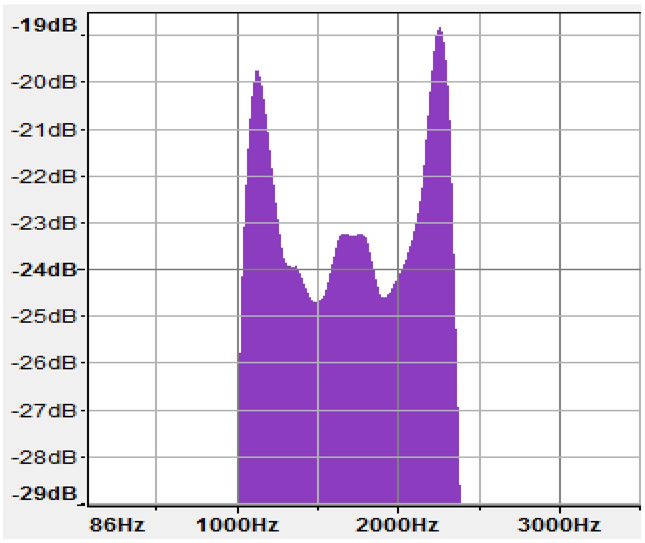
Once again we see a similar pattern. If you don't have much CPU power available, you might want to pick just the third one, V, instead of running them all in parallel.

How well does it work over the radio?

The problem with VHF/UHF FM voice transceivers is that they are designed for voice, not digital data. The transmitter uses pre-emphasis to boost the higher audio frequencies. The receiver uses de-emphasis to level it out again. A high pass filter removes the CTCSS tones before they can reach the speaker. A low pass filter takes out high frequencies not useful for voice.

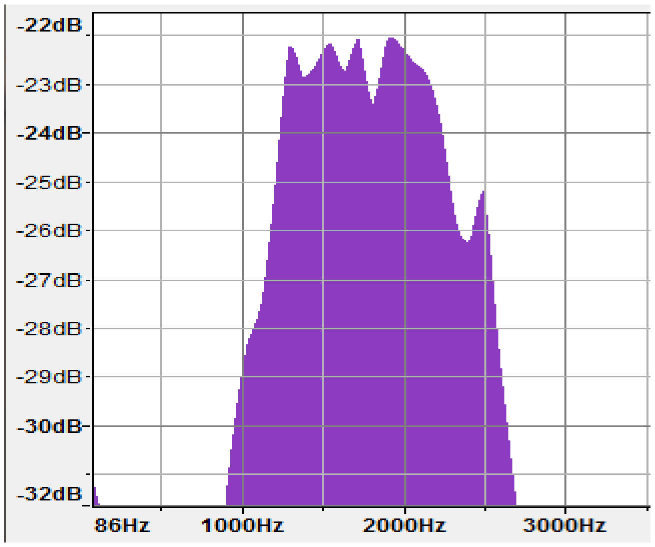
The related document, ***A-Better-APRS-Packet-Demodulator-Part-1-1200-baud.pdf*** , goes into detail about the problem this causes and one solution.

Most modern VHF/UHF mobile transceivers have a round 6 pin “data” connector. Remarkably the connector type and pin-out is standardized across the major manufacturers. “Data” is not a very good name because it is audio, not digital data. “External modem” would have been a much better name. This bypasses the normal audio stages and provides wider audio bandwidth suitable for use with modems for transmitting digital data.



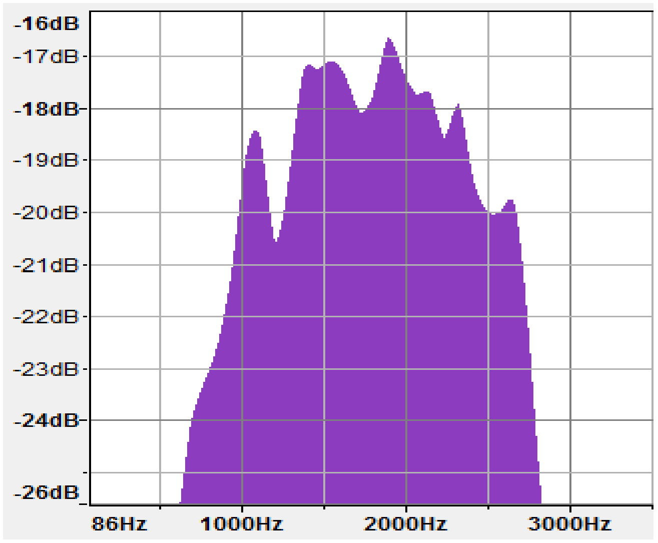
Here is the spectrum for a 1200 bps AFSK signal. As you would expect there are peaks around 1200 and 2200 Hz.

At any instant we only need to decide which one is stronger.



2400 bps 4 phase PSK.

There are 4 peaks. Is it related to using 4 phases?   I dunno. I don't have the theoretical background to explain it.



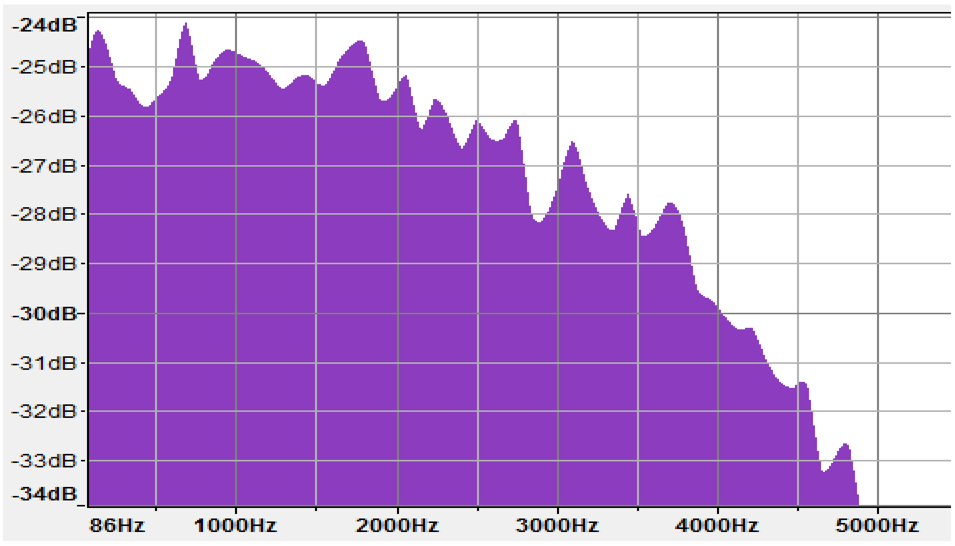
Finally 4800 bps 8 phase PSK.

It's wider and therefore more demanding of the radio channel.

It also needs to distinguish between 8 different signal states, not just 4, so it would be more susceptible to various types of distortion.

These all easily fit into the voice frequency range, generally considered to be 300 – 3000 Hz. However, the pre-emphasis, de-emphasis, and other filtering, found in most radios, all add distortion. You don't hear it with voice but it can cause deterioration of digital data making it less likely to decode properly.

Just for comparison, let's look at standard 9600 baud.   Keeping the scaling the same, for easy visual compartison, it looks like this:



Note the peak at 4800 Hz. This corresponds to the maximum frequency of alternating 0 and 1 bits.   The upper end of the audio passband needs to extend to at least 5000 Hz.

It's not as obvious how low we need to go. There is a peak around 200 so I suspect that a high pass filter, intended to keep CTCSS frequencies away from the speaker, would cause issues.

Results

Let’s try it with real radios and see what happens.

- 2400 bps PSK, over the radio, using the microphone and speaker connections, was a success.
- 4800 bps was not successful with the same particular mix of transmitter and receiver.   We already discussed how it is more demanding of the audio channel properties.
- 4800 bps was successful using the same transmitter (with Microphone input) and the PR9 pin of the “data” connector of a different receiver.

Your mileage may vary.