QRSS and You

Using absurdly low-speed CW for "communications"

(As well as other ultra-narrowband modes)

What Is QRSS?

The term **QRSS** is derived from **QRS** - a cw ("Morse code") abbreviation that means "You are sending too fast" or "Slow down." By extension, then, QRSS would imply ***very*** slow sending speed.

While it is technically possible to copy CW sent at such a slow speed - provided, of course, that the signal/noise ratio was good enough to hear it - doing so would be very tedious as "copying" code at very slow CW speeds is *not* the same as copying CW at "normal" speeds. At "normal" speeds (that is, anything above 5 words per minute) where one hears the "sound" of the letters and words, QRSS operation becomes a matter of timing the lengths of the elements received and manually assembling them into letters and words. While I have done this in the past using a pen, paper and a watch, I prefer to leave such tedious tasks to computers!

The big question: Why?

Why send CW at such a slow speed? It all comes down to communications theory. The faster the signaling rate, the more bandwidth you need. The more bandwidth you have, the more energy (i.e. transmitter power) you need to keep your signal above the noise. In a nutshell this means that the faster you go, the more power you need, all other things being equal.

Let's take a look at a 12 WPM CW signal for a moment. The "dit" length at this speed is approximately 1/10th of a second, so we can say that we could sent 5 dits (and 5 "inter-dit" spaces) in one second. As a general rule-of-thumb, to keep the dits from running into each other, we'd need to make sure that our "receive" system was capable of responding at about 3 times the frequency of the dit's 1/10th of a second period, or about 1/30th of a second - which is to say, about 30 Hertz of bandwidth.

So, this means that if we want to receive a CW signal that is running at 12 WPM, we'd have to use a CW filter that was no narrower than 30 Hz: Going narrower than this will cause the "dits" to run together and we'll have difficulty copying the signal.

Let's compare this to a standard SSB filter of, say, 2.4 kHz bandwidth. This 2400 Hz wide filter is 80 times wider than the 30 Hz filter (for an "ideal" filter) and

How "wide" is your brain?

You may be reading some of this and say to yourself, "*Self, this doesn't make sense - I can hear a weak signal as well with the SSB filter as I can with the CW filter. What's the deal here?*"

We are getting into a field referred to as ***Psychoacoustics***, or "How the brain perceives sounds." As it turns out, the "trained ear" can fairly easily resolve a bandwidth of less than 30 Hz - ***assuming the presence of random noise*** in the background. This means that if you have a single CW signal amongst 2.4 Hz of white noise - or even other CW signals that are at roughly the same strength but at a different pitch, then your brain/ear is perfectly capable of picking it out, being able to "ignore" the 2.4 Hz of noise and the other "dissimilar" signals: Under these

will let 80 times as much energy through it. Since our desired CW signal only takes 30 Hz of the 2400 Hz, most of our receive energy is where our signal is NOT (assuming a signal that is near the noise level, of course) and we are at a 19db disadvantage (ignoring Psychoacoustics, for the moment.)

From this example we learn two things:

1. For best performance, we receive with a filter that is no wider than the signal we are trying to receive
2. If you have a wider filter than the signal you are trying to receive, you are receiving "extra" energy from the noise that is "diluting" the desired signal - making the effective "signal-to-noise" ratio worse.

What #2 means that if, for example, you are ***just*** at the threshold of copiability with that 30 Hz filter mentioned above, if you used a 60 Hz wide filter, the signal that you were trying to receive would have to be ***doubled*** in power to get back to that "threshold of copiability." (Again, we aren't taking into account things like operator skill or psychoacoustics.)

conditions, ***narrower filters won't always improve the "copy" for a skilled operator.***

Put this same CW signal in amongst other similar signals - some of which are much stronger and very close to the same frequency- and even the trained ear is hard-pressed to make out the "buried" signal. Under these conditions, the brain isn't able to reject those "other" signals and a our hypothetical 30 Hz filter may make a world of difference.

A simple detection/indication circuit - like an audio level meter - does ***not*** have any means to be able to pick out that 30 Hz of signal among the 2.4 Hz of noise, so if you were to look at the meter alone, you would never be able to see the meter deflect in sync with the keying of the signal. Take the output of a 30 Hz bandpass filter - one that passes only that range of frequencies occupied by the CW signal- and feed it into the same meter and then you will likely see the meter deflect with the CW signal. Why? Instead of 80 parts noise and 1 part CW signal bandwidth, we are now feeding it only that part of the bandwidth containing the CW signal.

Becoming Narrow-Minded:

For a given modulation type (in this case, on-off keyed CW) the lower the "data rate", the narrower your transmitted signal, the narrower the required receive bandwidth, and the lower the lower the transmitted power required to maintain that "threshold of copiability." Communications theory (Shannon's Law) states that if you were willing to transmit your data *infinitely slowly* you could communicate with *infinitely narrow detection bandwidth* and *infinitely low* (not zero) power. It should go without saying that there are practical limits to how slow you would go to convey useful information in a reasonable amount of time.

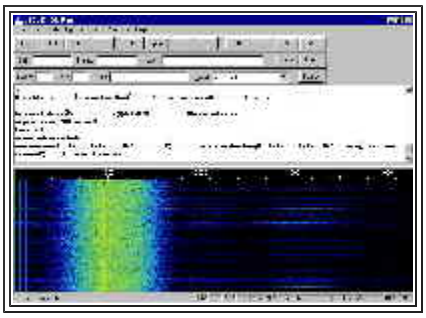
QRSS generally means that the CW sending speed is below 2-3 WPM - usually ***much*** slower than that. Let's take as a rather extreme example, the VA3LK beacon on 137.79 Hz. This experimental beacon has operated at a "dit" rate of one dit every ***ninety*** seconds - that's about .0133 words-per-minute, or about ***0.8 words-per-hour***. This also implies that the detection bandwidth for such a signal (***see the equations below)*** should be at least 0.033 Hz - that's 33 Millihertz (mHz - notice the small "m"!) Going much much narrower than this and the "dits" will start to run together and sacrifice "intelligibility."

Compare this to the 2.4 Hz bandwidth for a moment: 33 mHz (notice the small "m") is 72000 times ***narrower*** than an SSB bandwidth - a difference of over 48 db. A better example would be to compare this with a 30 Hz CW filter - a "reasonable" value for a well-designed CW filter. This would still be 900 times narrower than that - over 29 db difference in the receive signal/noise ratio. (Once again, we are ignoring psychoacoustics.)

Putting it into practice:

What this means is that going from a 12 WPM signal received in a 30 Hz filter to a 0.0133 WPM (that's 0.8 WPH - Words Per Hour) signal received in a 0.033 Hz filter (which would be about as narrow as you'd want to go to copy a signal as "fast" as that...) would theoretically be equivalent of getting almost a 1000-fold increase in transmitter power. The obvious tradeoff is that the communications rate is ***very low***. This rate makes it tedious - almost impossible - to copy the signal by ear. Additionally, keeping a 0.033 Hz wide filter centered on the signal would be a feat in itself. Fortunately, computers have afforded us a solution.

"Fourier is our friend..."



An example of a "Waterfall" display on the Digipan program.
Frequency is on the horizontal axis, time is on the vertical axis, and the "brightness" of the color indicates relative signal strength.
Displays where the horizontal and vertical axis are swapped (and move right to left) are sometimes called "curtain" displays.

Through the "magic" of computers, we can ***simultaneously*** see many little "slices" (often referred to as "bins") of spectrum simultaneously visually on what is called a *Waterfall Display*. This means that even though the signal may drift from one 0.033 Hz "slice" to another, we will still be able to see the signal as it moves around.

Various programs display these "slices" on a graphical display: In the example, the X (horizontal) axis shows frequency, the Y (vertical) axis shows time (with the most recent being at the top) and the Z (brightness/color) indicates relative signal strength. Each "pixel" on the X axis represents a certain range of frequencies - and the more energy in that range of frequencies, the "brighter" that pixel will become. (The "brightness" could be represented by an increase in actual brightness, or a change in color - depending on the way the program is designed.)

Receiving the signals

We must make sure, however, that our receive system doesn't drift too fast: If the receive system drifts, say, 0.033 Hz every 90 seconds, then that signal has spent equal time in each "slice" of spectrum, "lighting it up" only half as much as it would if it were to have stayed put, effectively reducing our "sensitivity" by 3db. If it were to drift, say, 0.33 Hz over that 90 second period, then we just slid through 10 "slices" - lighting each one up 1/10th as much as it would have been had it remained stable - possibly making the signal undetectable. As it turns out, the visual medium allows our brain to do a remarkably good job in "integrating" various bits of the signal so even if our signal has drifted a bit, it is often possible to pick out evidence of "coherence" amongst the chaos of the random noise. That is, ***the background noise is random whereas our desired signal is not***.

In general (and there ***are*** exceptions, of course) VFO-type radios (i.e. non-synthesized) are ***not*** good candidates for use for QRSS reception. Given the typical drift rate of 100 Hz/hour for these radios, this would imply that they move about 1.6 Hz/minute, limiting the effective minimum useable bandwidth to 0.05 Hz or more. There is also the problem that when using such narrow bandwidths, it the displayed frequency range is very small (under 100 Hz, usually) and it may be difficult to keep a VFO radio within that range, let alone figure out exactly which frequency you are on in the first place.

There is another factor that should be considered: Propagational phase shifts.

In effect, changing the phase of a signal on a particular frequency is the ***same thing*** as changing its frequency during the change. Let's suppose that we have a 1 Hz tone. If we were to retard the phase of it by 360 degrees every second, then we are "eating" one cycle every second, resulting in a 999 Hz tone.

Propagation is a tricky thing: A very distant signal will likely arrive at the receive point via a skywave. Over that distance the effective path length could change slightly. At our example frequency of 137.79, the wavelength is approximately 2 kilometers: The total distance the signal travels to get to the receive site 2000+ km away could easily change by 1 km in the course of a minute or so, resulting in a 180 degree phase change. Since a phase change amounts to a frequency change, that signal may have actually moved while you were trying to "copy" that dit. It is for this reason that one may want to avoid running narrower filters than you absolutely have to. Fortunately, LF propagation is quite stable and these sorts of effects are much less prevalent than on HF.

"Seeing" the dits and dahs

"Reading" the dits and dahs from a waterfall display takes a bit of getting used to, but it is really quite effective in digging the signals out of the "noise." Even the slightest trace of signal on the display can be perceived by the eyes: The brain is very good at picking bits of order out of visual chaos...

One suggestion that I would make: When you have, for certain, determined the length of the dits and dahs, it helps to mark their "length" on a piece of paper. When QSB or QRM occurs, holding that piece of paper up the the screen can help make the decision whether or not what is on the screen is "too long for a dit" or "too short for a dah." This method depends on the "scroll" speed of the waterfall display being constant - something that may not be true - especially on computers slower than a 200 MHz pentium.

Using the Software

I have had good luck using **Spectran** (see the link below) for QRSS. **Argo** is a program very similar to Spectran except that also has screen-capture capabilities but no real-time audio filtering. Both of these programs will display a "waterfall" at various speeds.

For best results, do not use a filter ***narrower*** than the following equations show:

CW Speed (in WPM) * 2.5 = Minimum filter bandwidth in Hz

Or, putting it another way:

3/(Dit length in seconds) = Minimum filter bandwidth in Hz

Keep in mind that these are the ***minimum*** filter bandwidths that aren't likely to cause the dits and spaces to be smeared together excessively. If conditions permit, I usually run the filters slightly ***wider*** than this to help "sharpen" the dits and dahs as they appear on the screen. The tradeoff here is that the signal/noise ratio is worsened with the wider signal.

Spectran vs Argo:

Up to this point, I have mentioned only Spectran, but there is also the **Argo** program written by the same authors. These two programs operate very similarly - except that Argo is more oriented toward visual representation of the recieved signal (having better facilities of displaying various bandwidths,

automated screen-capture, etc.) while Spectran has a stronger emphasis toward being a general-purpose utility program for analyzing and filtering the audio spectrum (as it has built-in bandpass, notch, and noise reduction) - in addition to visually displaying it. Argo also provides the facility of inputting corrections for errors in the effective rate of a sound card.

If QRSS (or a similar "visual" slow mode) is your forte, Argo may be the better choice - but try them both and see which one is your favorite: ***They are both free!***

Spectrum Lab:

Another useful program for QRSS work is **Spectrum Lab** by DL4YHF (see the link below). This program is actually a suite of tools that includes, among many other things, a spectral display that can be configured to convey spectral information visually - including a number of "canned" presets for QRRSS modes.

While very powerful, Spectrum Lab is also a bit complicated and has a fairly steep "learning curve" but there are a number of resources on the web that give some details on how this may be done.

Beware incorrect sample rates!

As with Argo, Spectrum Lab also provides a means for inputting corrections for the effective sample rates of sound cards. Why is this important? As it turns out when you run your sound card at a "11.025kHz" sample rate, chances are that it is ***not*** at that rate! Why? Newer versions of the Windows (tm) operating system actually run the sound card hardware at just a single sample rate - usually 48kHz - ***no matter what the program calls for!*** Why do this? Since it is possible for more than one program to access the sound card's input and output streams the operating system can only run it at a single rate for all programs. If the program calls for a sample rate of something *other* than the "native" sample rate a conversion is done in software - but this conversion isn't usually very precise!

Another source of error can be in the sound card hardware itself: Some lower-cost chipsets derive their sound card clock from sources that don't yield a precise sample rate of, say, 48 kHz. Not unexpectedly, this, too, can result in an error.

So, if the computer "thinks" that the sample rate is "exactly" right - but it isn't - it will display frequencies incorrectly: If the sample rate is higher than nominal, it will display frequencies that are too low and vice-versa.

How far off can a sample rate be, you may ask? While differences of 1-2% aren't uncommon, I have seen a netbook with a sampe rate that was nearly 9% high when running at a sample rate of 11025 kHz!

A program like Spectrum Lab allows one to take a very precise audio frequency - such as the 500Hz and 600Hz tones broadcast by WWV/H (and received in AM - NOT SSB!) and use those to calculate and input correction factors for the varied sample rates.

An actual "listening" session

Using Windows' (tm) built-in "Screen Capture"

On 24 January, 2001 at midnight, I decided to try to listen for the beacon operated by Larry Keyser, VA3LK - ***now SK***. After spending a few minutes "listening" (looking at the waterfall display, actually) I decided that it was getting too late to stay up. I set the computer to "record" at a 6000 samples/sec rate with 16 bits, mono, and went to bed. The next morning, I had a file that was about 250 megabytes that would take about 7 hours to play.

Or would it?

"Listening faster"

Since I have **two** computers in my shack (the one is the "ham" computer, and the other is the faster computer used for audio/video editing, etc.) I strung an audio cable between the two and "played" the file back at 44100 samples/sec instead of the original 6000 samples/sec. This has the effect of "time compressing" 7 hours of "listening" into less than an hour.

What is also required is that one calculates what the "original" frequency versus the "faster" frequency is going to be. Since the "Ham" computer that I used at the time had a ***really cheap*** ($8.00) sound card, I knew already that it was slightly off-frequency (especially with the "non-standard" 6000 ksps rate that the recording program wanted to use) so I recorded a precise tone from WWV (the frequency of which I knew) at the 6000 Hz sample rate, and then played it back at 44100 samples/sec and measured the tone frequency using Spectran on the ***other*** computer. I divided the new frequency by the original and now had a ratio that I could use to calculate the new "receive" frequency, the bandwidth, and "time compression" factor.

With my sound card combination, I ended up with a 810 Hz tone (from VA3LK) being translated to 6006 Hz - a factor of 7.41. To look at 6006 Hz I had to change Spectran's sample rate to 22050 samples/sec. I also had to multiply the "bandwidth" on spectran by the same amount - so I used a bandwidth of 1.3 Hz or so. Also, I calculated that VA3LK's 90 second long "dits" were now going to be about 12.1 seconds long, and set the "scroll" rate on the waterfall appropriately. The results are below.

Going faster still...

If this isn't fast enough for you, then you can force the playback program to play the mono (single audio channel) file as a ***stereo*** file. This will effectively double the playback sample rate because ***two*** samples are being played back *simultaneously.*

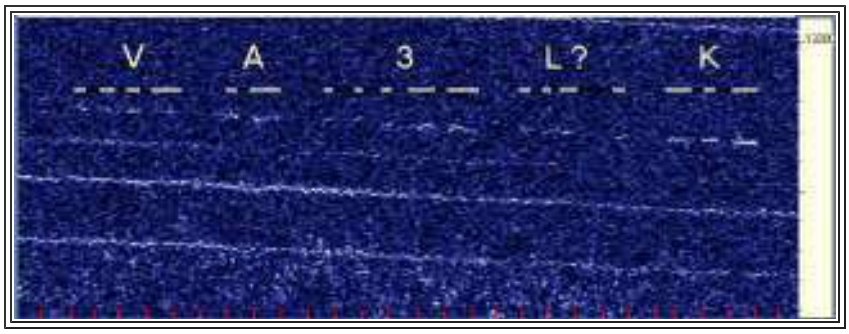
There is a "gotcha" with this method, however: Since we are effectively "throwing away" every-other sample, we are also ***halving*** the time resolution of the original. What this means is that we cannot play back ***or*** record a signal higher in frequency than ***one fourth*** of the original sampling rate without alisasing effects - which in this case amount to added noise and distortion. This means that for, say, a 6000 Hz sample rate, the ***original recorded*** audio cannot contain (un-aliased) frequencies higher than 1500 Hz.

You can use the operating system to capture what is on the screen ***for*** you without using a program that has screen-capture capabilities. To do this simply make sure the program you want to capture is in the active window and hit and hold **ALT** and then hit **Print Screen**. This will copy the active window to the clipboard.

At this point, you may go into a drawing program (like Paint) and use the ***edit - paste*** function. You may then edit/crop the image and save it to a file.

The programs ARGO and Spectrum lab also have built-in screen-capture capability that can automatically save a file at defined intervals to allow you to visually review the results at a later date.

I was able to easily avoid this potential problem by recording the original audio through a 300 Hz CW filter centered at 800 Hz - passing only the range of frequencies from 650 to 950 Hz. Even if I had used the SSB filter (which would have passed audio up to 2.4 Hz) I could have run the audio file through a low-pass filter program to remove the "offending" higher frequencies prior to "playing" them back.



The above image is two "screens" from Spectran "stitched" together showing the received callsign from the late VA3LK. The original 6 ksps file was played back at an effective sample rate of 96ksps as described above. The "tics" at the bottom represent approxmately 162 seconds.

(Click on the picture for a larger version, or click here for a 175k .GIF version)

Update:

I replayed the original audio file and "tweaked" the settings on Spectran and was able to improve the "copy" a bit, the result of which is displayed above. This time, I played the original sample back at 96 ksps (by forcing the playback software to 48 Hz in *stereo* mode - so it processed two samples at once) and used an effective bandwidth of approximately 0.030 Hz - taking into account the multiplication factor. I sent the results to Larry, VA3LK and he replied with the following:

```
Clint:
Answer to your question, YES.  You have heard the message from VA3LK on
137.7894 kHz.  Congratulations my friend!
Thank You very much for your effort. I really appreciate the time it
took, now we can continue the march westward,
Hello CQ CQ KH6, ZL, VK LowFer people, ANYONE HOME?
```

If you have any comments about this page, let me know what ***you*** think by sending an email!

Modes other than CW:

Using on-off keyed CW is attractive because of its simplicity. One can simply see the dots and dashes and "decode" the received message. Additionally, it does not need any synchronization (i.e. there is no "start bit" to try to find.) Its utter simplicity also belies its disadvantages, however:

Receive system information:

Receiver: Modified Drake TR-7 with outboard homebrew DDS VFO (based on an AD9835). Nowadays I also use an RFSpace SDR-14 receiver.

- The on-off keying has a penalty in terms of communications power: It may not be possible to tell (except by occasional educated guesses) whether the signal is unkeyed or simply faded out.
- The variable element length (dits, dahs, interelement spaces, intercharacter spaces, etc.) are easy targets for QSB and QRM.
- The coding is somewhat inefficient: It takes twice as long to send a dah than a dit. (We are counting interelement spaces, remember...)

One possibility that reduces these difficulties is ***FSCW***. In this mode, "key up" is one frequency, and the "key down" is another. If you are able to receive both of these frequencies, then you can have positive verification between these two states and may be able to "fill in" some gaps that would otherwise be left by the "***Did it unkey, or QSB?"*** uncertainty. This would also allow the *possibility* of copying a signal if one of the two frequencies were being blocked by a carrier or QRM.

Antenna: LF Engineering LF-400B atop the metal roof of my house - near the MedFER antenna shown ***here***. *(No, the MedFER transmitter wasn't on when I was listening...)*
Location: West Jordan, UT, grid DN40ao. (About 15 miles southwest of downtown Salt Lake City, Utah.)

The analog receive bandwidth was 300 Hz. RF noise blanking was done using the Line-Synchronous Noise Blanker and the noise blanker built into the TR-7 (an NB-7 rev. 2)

The display above was done at an effective bandwidth of approximately 0.17 Hz (taking into account "time compression") using Spectran.

There is one mode that reduces some of these difficulties: ***DFCW*** or **D**ual **F**requency **CW**. Simply put, one frequency is a dit, another is a dah, and unkeying represents a space. This provides two advantages: By representing a dit and a dah on different ***frequencies*** rather than different lengths of elements, the "dah" need only as long as the dit. Furthermore, since we don't need to space dits and dahs apart from each other, we can (if we choose) eliminate the space between a dit and dah or a dah and a dit - but not between dits and dits and dahs and dahs (or else we couldn't tell when one dit ended and another began.) "Encoding" the signal in this way as the obvious advantage of more than doubling the speed at which messages may be sent.

The obvious disadvantage of this method is that ***both*** frequencies must be clear of QRM. It also takes some "getting used to" when decoding the message.

Purely digital modes:

There has been some discussion about using a purely digital mode. The most frequently mentioned of these is BPSK. BPSK stands for **B**i-**P**hase **S**hift **K**eying. What this means is that a "0" might be sent with one phase of the carrier, and a "1" would be sent with carrier 180 degrees different. This method brings to bear several complications right off the bat that need to be taken into account:

- To know the phase of the carrier at any given time, one would need to know what the phase ***had been*** in the past and what the two possibilities might be in the future. In a nutshell, one needs to have a local "copy" of what the carrier looks like as a basis of comparison.
- Character/bit framing needs to be accomplished. This is less of a challenge because one could simply decree "the start bit is always coincident with the beginning of the first second of the minute" and as long as everyone's clocks are accurate (not so difficult to do these days) then things are fine.

- There is a possible problem with phase ambiguity. Unless one is already knows exactly what is being transmitted, the signal looks the same whether it is phase reversed or not. How can we tell if what is being sent right now is a 0 or a 1?

The most daunting problem is the first one. To be sure, we could conceivably lock the transmitter and receiver to some global standard (such as GPS) but assuming that you have done that, propagational changes may "dilute" the bit during the time that it is being received because of phase shifts. While this isn't as much of problem on LF frequencies, it does effectively limit the minimum bandwidth attainable.

A common solution is to recover the carrier and the data separately. This is frequently done at higher bit rates because one can simply look for a carrier with a narrow filter and then, once it is acquired, use that carrier to demodulate the data itself. In the case of very narrowband communications the bandwidth of your carrier recovery system may be *about* the same as your "data" filter. In other words, it may take every bit as long to acquire the carrier as it did to demodulate the data it contained.

> ***Another way to "Listen" Faster...***
>
> Bill DeCarle, VE2IQ, has written a program called "CRUNCH" that not only speeds up playback of a .WAV file, but will convert the "new" high frequency back down to a more user-friendly low frequency.
>
> To get this program, it's probably best to do an internet search to find its current home on the web.

Fortunately, we *can* look into the future. Sort of. If we can live with a delay in our communications system (which could be ***hours*** in some cases) we can "cheat" a bit. If we "record" the signal, we can recover the carrier and figure out what its doing and then "go back" and using the information that we gained from the first "carrier recovery" pass, go back and figure out what the data ***was***. This scheme is commonly used in high-speed TDMA networks where we can't afford to waste "airtime" just to lock onto our signal. The obvious disadvantage is that we lose ability for "real time" (assuming that it was practical if you were sending just one bit per minute, for example) communications.

The phase ambiguity problem could be resolved in several ways: One could simply receive the signal and then try "assembling" it both ways - only one of which will make sense. We could also use a *differential* coding scheme where we might send a "0" with a phase shift and a "1" with no phase shift - but this scheme has its own problems such as knowing what the *previous* bit was.

To be sure, BPSK has turned out to be quite robust: Bill DeCarle's ***Coherent*** program takes advantages of several of the predictable traits of a BPSK signal (such as the location of the start and stop bits by previous agreement) and, knowing the message length beforehand, one can effectively "average" numerous repetitions together and reconstruct the original message.

Another mode (somewhat similar to **Coherent***)* is called **WOLF** (**W**eak-signal **O**peration on **L**ow **F**requency.) It is essentially a more specialized implementation of **Coherent**. This mode uses a fixed-length 15 character message, repeated at a precise rate. For a brief description on how to use this program, see Lyle Koehler's Wolf For Dummies page.

Because links change over time, it is probably best to do your own internet search for the programs mentioned above.

Return the KA7OEI main page, or look at these other pages:

DSP Software for digging out the weak/buried signals:

Spectran Beta 4 *and* ***ARGO*** *- Spectran is the successor to the well-known Hamview program. This windows version can use any standard (full-duplex) sound card. Like "Spectrogram" this program produces a graphical "waterfall" display and has a real-time audio bandpass filter. This program is more specifically suited for amateur/radio use as it's display algorithm can more distinctly show "buried" signals than Spectrogram. The ARGO program is specifically tailored for QRSS - extremely slow speed CW. It's frequency range is limited to audio, however. (By I2PHD and IK2CZL. Freeware - non-commercial use.) You can download both of these from the Weak Signals web site.*

Spectrum Lab software by DL4YHF - This is a general-purpose program that, among other things, can be used to copy and display QRSS signals. Note that this program can be rather complicated and intimidating to use so read the documentation - such as it is - very carefully!

Any comments or questions? Send an email!

This page copyright 1999-2012 and maintained by Clint Turner, KA7OEI
This page last updated on 20120228

Since 12/2010: 13,215