JASON Version 0.99

Introduction

First of all, why the name Jason ? Well, you all know the program Argo...

Argo was the name of the mythological ship that brought the Argonauts in Colchis, searching for the Golden Fleece. The coxswain of the ship Argo was the Greek hero Jason. So this program, which loosely relies on the technology of Argo, has been named Jason.

And, if you don't like mythology, you can always read Jason as an acronym : **J**ust **A**nother **S**hip-**O**wner **N**ame :-)

The JASON Coding Scheme

The coding scheme of Jason is based on the ideas about IFK (Incremental Frequency Keying) initially proposed by Steve Olney, VK2ZTO.

Basically, the information is coded in the absolute value of the difference between two frequencies sent sequentially. This has the advantage of not needing a precise initial tuning. A tuning error of a few Hertz is perfectly acceptable.

Another characteristic is that, being the frequencies sent one at a time, you don't need a linear amplifier. A class-D Mosfet TX will do nicely. The frequency deltas can assume one of 16 different values. After sending one tone, the next one is shifted by the appropriate amount, up or down depending on the setting of the USB/LSB switch. With 16 deltas we need 17 slots (tones), and any overflow causes a wraparound.

With 16 possible deltas, each baud (a baud is a change in the signal transmitted, in this case a change in frequency) encodes 4 bits (called a nibble), which are not enough for a reasonable alphabet. So we need two nibbles for our alphabet.

But now a problem arises: how can we get character synchronization? In other words, which is the high-order and which is the low-order nibble? I solved the problem in the simplest way, which may be not the optimal one. I used the high order bit of each nibble to encode this info. So the high-order nibble is of the form '1xxx'b, while '0xxx'b is the low-order one. Here xxx stands for the actual information transmitted.

So now there are 6 bits available to encode our alphabet, enough for 64 symbols. I decided that my alphabet would be that that in ASCII code goes from x'20' (the blank) to x'5f'. This allows the transmission of all the letters (upper case only...), the ten digits, and practically all the punctuation symbols normally used.

JASON Signals

Ok, this explains the encoding. What about the signaling? For this version of Jason (but it may change in future versions), I chose the following parameters for the standard speed setting:

- The default Rx and Tx center frequency is 800 Hz.
- Each one of the 17 slots is separated from the adjacent by 3 FFT bins, so to guarantee some orthogonality.

- Each FFT bin is roughly 84 millihertz for the standard ***Normal*** speed setting, so the slots are at a distance of roughly 252 millihertz, for a total band occupancy of 4.038 Hz for the standard ***Normal*** speed setting.
- Each baud (each tone sent) lasts for roughly 11.89 seconds for the standard ***Normal*** speed setting (the inverse of the FFT bin width)
- The throughput hence is about 2.5 characters per minute for the standard ***Normal*** speed setting, hardly a speed champion, but it compares favorably with QRSS.

The following table summarises the parameters for the various speed and turbo on/off settings.

Speed	**Tone Duration(S)**	**Tone Spacing(Hz)**	**Bandwidth (Hz)**	**Characters/Minute (Turbo Off)**	**Characters/Minute (Turbo On)**
Slow	95.2	0.03	0.5	0.3	0.6
Normal	11.9	0.25	4	2.5	5
Fast	1.5	2	32	20	40

Options

The ***Options*** menu allows settings for a number of parameters.

===

- **Select Audio Input…:** Displays a Volume Control Dialog for your Soundcard Device. Here you can select the audio source input.
- **Read from Wav file…:** Displays an open file dialog to allow audio data to be read from a .WAV file (NOTE: Must have a sampling rate of 11025Hz).
- **Read from JAS file…:** Displays an open file dialog to allow data to be read from a .JAS file. A JAS file contains the down-sampled I and Q components of the audio input.

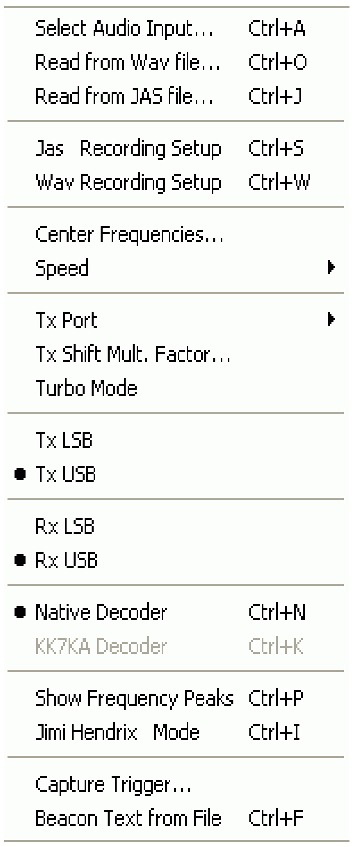
===

- **Jas Recording Setup:** Displays a save file dialog allowing the specification of the .JAS filename and save location.
- **Wav Recording Setup:** Displays a save file dialog allowing the specification of the .WAV filename and save location.

===

- **Center Frequencies…:** Displays a dialog that allows setting of the audio TX and RX frequencies (can be set independently) in the range of 50Hz – 5000Hz (the upper limit is set by the fixed 11025Hz sampling rate – Nyquist). Should initially be set to the default of 800Hz to allow easy netting.
- **Speed ->:** Displays a sub-menu to allow selection of three different speeds as shown in the table above (Slow/Normal/Fast). This setting must be the same at each end of the communication path.

===


Select Audio Input... Ctrl+A
Read from Wav file... Ctrl+O
Read from JAS file... Ctrl+J
Jas Recording Setup Ctrl+S
Wav Recording Setup Ctrl+W
Center Frequencies...
Speed
Tx Port
Tx Shift Mult. Factor...
Turbo Mode
Tx LSB
● Tx USB
Rx LSB
● Rx USB
● Native Decoder Ctrl+N
KK7KA Decoder Ctrl+K
Show Frequency Peaks Ctrl+P
Jimi Hendrix Mode Ctrl+I
Capture Trigger...
Beacon Text from File Ctrl+F

- **Tx Port ->:** displays a sub-menu allowing the selection of the three output options as outlined in the ***Interfacing JASON*** section below.
- **Tx Shift Mult. Factor…:** In the case where the output audio is fed to a frequency multiplying circuit (e.g., PLL) you need to enter the frequency multiplying factor here to allow JASON to reduce the frequency shift

deltas to ensure that the correct shifts are present at the multiplied frequency.

- **Turbo Mode:** An ON/OFF selection to enable/disable the TURBO mode. Selecting Turbo mode causes the tone time to be half as long while still maintaining the same tone shifts. This should only be done when the S/N is good. This setting must be the same at each end of the communication path.
  ===
- **Tx LSB / Tx USB:** Selection must match the correct sideband setting for the Tx at your end.
- **Rx LSB / Rx USB:** Selection must match the correct sideband setting for the Rx at your end.
  ===
- **Native Decoder:** Selects original decoder.
- KK7KA Decoder: Selects KK7KA decoder. Not selectable (greyed out) in v0.99 as decoder has not been upgraded by KK7KA to accommodate speed settings other than Normal.
  ===
- **Show Frequency Peaks:** Enables/disables a real-time readout of the detected peak frequency between the yellow lines on receive.
- **Jimi Hendrix Mode:** Amplitude clipper intended to be used in the presence of impulsive noise. For those not atune to Mr. Hendrix – he was famous for a guitar sound produced by heavy clipping.
  ===
- **Capture Trigger…:** Displays a dialog that allows entry of a piece of text that will trigger the beginning of text capture. Useful for leaving the receiver running waiting for a signal to begin (or rise out of the noise). The received screen text is saved in a file called "Jason.log" in the directory where Jason.exe is located.
- **Beacon Text from File:** Type different beacon messages into separate text files (.txt) and select a message from this open file dialog.
  ===

Capture Range

The program has a capture range of 1.5 times the bandwidth (my arbitrary choice), i.e. slightly more than 6 Hz for the standard ***Normal*** speed setting. The capture range can be easily positioned with the mouse. To do this, left-click on the approximate center of the white lines that represent the received signal. The capture window (the two yellow lines)will then positioned so that its center will coincide with the frequency you have clicked on. Be warned that any signal that falls, even slightly, outside the capture window will be ignored by the decoding algorithm.

Frequency Stability Requirements

Using the standard ***Normal*** speed setting as an example, it is evident that the combination of the Tx and Rx drifts must be such that, in each 11.89 sec. interval, the frequency must stay in one single FFT bin, i.e. 84 millihertz. Actually, a drift from -1 to +1 bin is tolerated, and accounted for, by the program. But let's remain on the conservative side. If we translate this into short term stability, where short term means a period of 10 minutes, we compute that our oscillator must not drift, in each 10-minute interval, more than 0.084 * 600 / 11.89 = 4.24 Hz (roughly). This means, at 136kHz, a stability of 31 ppm is required, which doesn't look like a difficult figure to achieve.

Interfacing JASON

How does Jason interface with the radio?

For reception, it's quite easy. Just bring the Rx audio into the sound card, just as you do now with Argo and Spectran. Keep the audio level low. Jason works best when the level bar on the left side of the panel is below 50 %.

You will see in the waterfall window two yellow lines. Tuning must be done so that to ensure that the white lines of the received signal are always inside the two yellow lines. The program discards all that falls outside. Fine-tuning is possible with the mouse. Click on a signal line on the waterfall window, and its frequency will be brought at the center of the yellow lines (the receiving window). The Rx center frequency can also be adjusted with a menu choice.

For transmission, I have envisaged three different modes, to make interfacing the easiest possible.

1. **Parallel Port Output** : via the Options menu, you can choose between LPT1 or LPT2. The code (ranging from 0 to 16) for the tone to be sent is output, with the ***Strobe*** pin pulsed high for 100ms. The Tx condition is indicated by the ***SelectInput*** pin being high.

2. **Serial Port Output** : via the Options menu, you can choose between COM1, COM2 or COM3. The serial format is 9600 - 8N1. The code (ranging from 0 to 16) for the tone to be sent is output. The Tx condition is indicated by the RTS being active. Via the Option menu, you can also choose the ZL1BPU format, which is what is needed by Murray's AVR Atmel board (http://www.qsl.net/zl1bpu/MICRO/EXCITER) Basically, an ASCII 'T' is sent at the beginning of transmission and an 'X' at the end. Moreover, the tones to be sent are identified by the three character sequence 'A00', 'A01'...'A09', 'A0A', A0B'....'A0F', 'A10' . No CR/LF used.

3. **Audio Output** : selectable via the Options menu.
   If you choose this method, a note is generated using the sound card, with a software implementation of a DDS, with a very long sine table (262144 entries), which ensures low distortion. The default center frequency generated is 800 Hz, but can be selected via the ***Options*** menu in the range 50Hz to 5000Hz.

   When using audio output, there is the possibility to specify a shift multiplier factor (default = 1), which can be handy when generating the RF using a PLL process that involves divisions. The output is in stereo with left and right channels outputting the audio in quadrature (I and Q) allowing feeding to a phasing-type SSB transmitter set-up. To maximise sideband rejection of a phasing-type SSB transmitter a dialog for adjusting the relative amplitude and phase of the two stereo channels (I and Q) is available in the ***Options | Tx Port*** menu.

With these interfacing possibilities, it should be easy to hook-up a Tx to Jason, both if you have the capability to up-convert the audio tone to the working frequency, or if you use a DDS board, either with a parallel or serial interface.

If you use a DDS board, you will need the following table to set up the frequency to generate for the standard speed setting, depending on the code output from Jason. Add the value in the table to the nominal value of the carrier generated by the DDS.

Tone #	Frequency (Hz)
0	797.981
1	798.234
2	798.486
3	798.738
4	798.991
5	799.243
6	799.496
7	799.748
8	800.000
9	800.252
10	800.505
11	800.757
12	801.009
13	801.262
14	801.514
15	801.766
16	802.019

These are also the default frequencies for the standard ***Normal*** speed setting generated when using the audio output, provided that your sound card has an exact sampling rate.

The encoding is taken care of by Jason; the DDS board task is only to generate the appropriate frequency, according to the code received via the serial or parallel port.

Frequency Accuracy for Using Audio Output Interface

A common method for interfacing is using a soundcard that is normally installed in a PC as standard. It is useful to analyse the frequency accuracy and stability requirements for JASON. JASON assumes an exact 11025Hz sample rate. The initial frequency accuracy determines the ability to place the desired signal in the input frequency range of JASON. The table below outlines requirements for different speeds.

	Bin Width (Hz)	**Display Range (Hz)**	**Capture Range (Hz)**	**Required Capture Window Accuracy (Hz)**	**Required Display Window Accuracy (Hz)**
Slow	0.0105	2.69	0.76	**±0.1**	**±1**
Normal	0.0841	21.5	6.06	**±1**	**±9**
Fast	0.673	172.3	48.45	**±8**	**±70**

The final frequency accuracy is the sum of the receiver frequency error and the soundcard error. Sampling frequency accuracies of soundcard are usually much poorer in terms of ppm than modern receivers. Typically soundcards sampling frequencies are derived from a standard ±100ppm crystal oscillator while receivers are typically ±5ppm. Working in favour of the soundcard is that it is dealing with audio frequencies; therefore 100ppm @ 800Hz = 0.08Hz error. For 5ppm accuracy receiver working at 134kHz the error is about 0.7Hz. So worse error is 0.78Hz (say 0.8Hz). Short-term drift of both the soundcard and receiver will be much less than the initial accuracy error. Therefore with this combination it is possible to start JASON in the ***Fast*** mode without having to manually adjust the signal to be within the capture window (assuming both ends of the communication path are using gear of the same specifications). If the operators are prepared to manually adjust the capture window then the signal need only be within the display window requiring 8 times less initial accuracy and so the ***Normal*** mode is also possible without further calibration for the equipment accuracy described above.

Important Disclaimer: Older soundcards usually had ±100ppm accuracy for the standard sampling rates of 44100, 22050, 11025 samples/second. However, the newer soundcards that have 96kHz and 48kHz sampling rates will likely have ±100ppm for these two rates, but the standard rates of 44100, 22050, 11025 might not have the same accuracy. For example a newer model external soundcard might be off by about 64ppm for the 48kHz and 96kHz sampling rates, while the sampling rates for 44100, 22050 and 11025 might be off by many ppm!!! Because of this it is probably necessary to calibrate the soundcard if it is one of the newer cards.

Methods for calibrating soundcards can be found on the Internet. As JASON has no provision to enter the real sampling rate (it assumes exactly 11025Hz) you can compensate by changing the ***Rx*** centre frequency. For example, if you find your soundcard sampling frequency is high by 100ppm (i.e., 11026.103Hz), you should enter a ***Rx*** centre frequency 100ppm low (e.g., 799.92001Hz for a nominal 800Hz). Of course, this does not account for any receiver errors.

WAV and JAS Recording

Incoming signal data can be saved for reading back at a later date. Two formats are available - WAV and JAS. The WAV format is Mono, 11025Hz sample rate, 16-bits per sample. The JAS format brings out the internal down converted and down sampled I/Q data (much lower effective sampling rate) that is normally fed to the internal JASON complex FFT algorithm before decoding.

Starting and stopping WAV and JAS recording is done via the ‘W’ and ‘J’ buttons respectively that appear just to the right of the ***Options*** and ***About*** buttons when JASON is in Rx mode. Note that if you haven’t setup the filenames for saving to, you will be presented with a file save dialog when you first click these buttons. So if you want to suddenly capture some interesting signals make sure you have setup the filenames beforehand (See WAV and JAS recording setup in the ***Options*** menu).

After recording a WAV or JAS file you can feed it back later to JASON via the WAV and JAS reading modes in the ***Options*** menu.

The purpose of recording to a JAS file is that, compared to a WAV file, the size of a JAS file at ***Slow, Normal*** and ***Fast*** speeds is 0.4%, 3% and 25% respectively of a WAV file for the same record time. Handy for recording overnight sessions without filling up your hard drive!!! Of course there is a catch. The JAS file only records the signal passing into the internal complex FFT and decoder - therefore you cannot change the centre frequency or speed if you discover that the signal is outside the capture range or at the wrong speed when you play the JAS file back later. So if you see a nice signal but some of it or all of it falls outside the capture and/or if it is the wrong speed then too bad. Any changes you make are ignored during playback. In comparison the WAV file records the whole audio signal and when played back allows adjustment of centre frequency and speed. So if you see a nice signal that is wholly or partly outside the capture window and/or the wrong speed, you can make changes via the ***Options*** menu and run the WAV again. The playback of the WAV file is just like a speeded up version of real-time operation (so you have to be quick if you want to click in the display window to centre the signal unless the file is long :-).

JASON Beaconing

For weak signal work it is useful to be able to repeat a message continuously to allow other stations to wait for propagation conditions to be suitable for reception.   This can be done by typing in the text message to be repeated in the typing input window enclosed in curly braces like this { *text to beacon* }. For example { I2PHD IN JASON MODE }. Alternatively, a text file can be created containing the beacon text (say in Notepad) and the file selected from the ***Options*** menu.

I do hope this notes are sufficient to make you understand how Jason works, and how to interface it with a Tx for LF work.

Should you have any questions, feel free to contact me at dibene@usa.net.

Enjoy,

Alberto I2PHD