JS8Call de KN4CRD

2018-10-08 - v0.7 - Pre-Release

FT8 has taken over the airwaves as *the* digital communication mode for making QSOs over HF/VHF/UHF. The mode has been widely popular as the latest offering in K1JT's WSJT-X application. FT8 is based on the same foundation as JT65, JT9, and WSPR modes for weak signal communication, but transmits faster with only slightly reduced sensitivity.

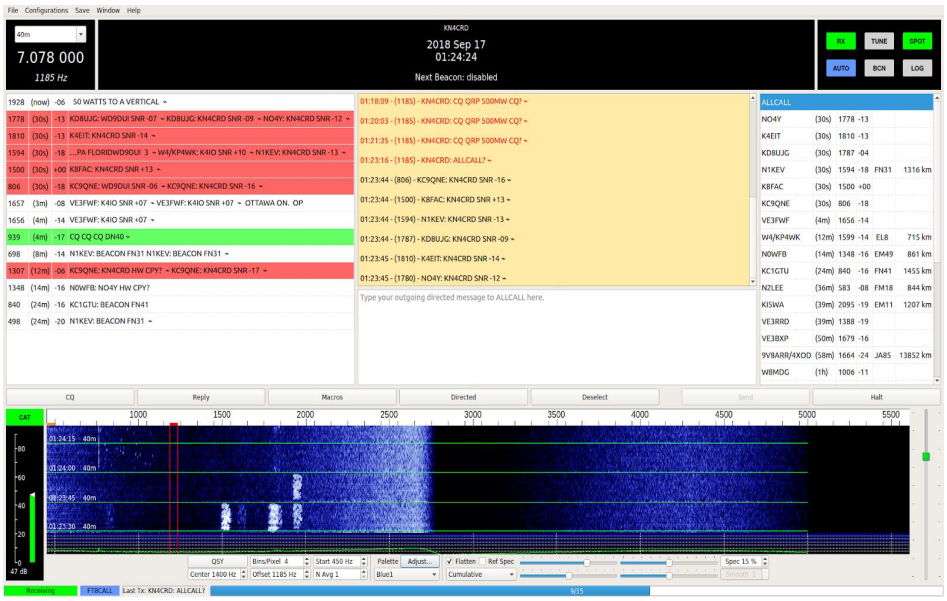
While FT8 is an incredibly robust weak signal mode, it is designed heavily to take advantage of short band openings on HF/VHF/UHF and only offers a minimal QSO framework. However, many operators are using these weak signal qualities to make successful QSOs on the HF bands where other modes fail.

JS8Call is an ***experiment*** to test the feasibility of a digital mode with the robustness of FT8, combined with a messaging and network protocol layer for weak signal *communication* on HF, using keyboard-to-keyboard style interface. JS8Call is heavily inspired by WSJT-X, Fldigi, and FSQCall and would not exist without the hard work and dedication of the many developers in the amateur radio community.

JS8Call stands on the shoulder of giants...the takeoff angle is better up there.

Read more on the original design inspiration here.

For release announcements and discussion, join the JS8Call mailing list here:
https://groups.io/g/js8Call



History

- **July 6, 2017** - The initial idea of using a modification to the FT8 protocol to support long-form QSOs was developed by Jordan, KN4CRD, and submitted to the WSJT-X mailing list: https://sourceforge.net/p/wsjt/mailman/message/35931540/
- **August 31, 2017** - Jordan, KN4CRD, did a little development and modified WSJT-X to support long-form QSOs using the existing FT8 protocol: https://sourceforge.net/p/wsjt/mailman/message/36020051/ He sent a video example to the WSJT-X group: https://widefido.wistia.com/medias/7bb1uq62ga
- **January 8, 2018** - Jordan, KN4CRD, started working on the design of a long-form QSO application built on top of FT8 with a redesigned interface.
- **February 9, 2018** - Jordan, KN4CRD, submitted question to the WSJT-X group to see if there was any interest in pursuing the idea: https://sourceforge.net/p/wsjt/mailman/message/36221549/
- **February 10, 2018** - Jordan KN4CRD, Julian OH8STN, John N0JDS, and the Portable Digital QRP group did an experiment using FSQ. The idea of FT8Call, combining FT8, long-form QSOs, and FSQCall like features was born.
- **February 11, 2018** - Jordan, KN4CRD, inquired about the idea of integrating long-form messages into WSJT-X: https://sourceforge.net/p/wsjt/mailman/message/36223372/
- **February 12, 2018** - Joe Taylor, K1JT, wrote back: https://sourceforge.net/p/wsjt/mailman/message/36224507/ saying that “Please don't let my comment discourage you from proceeding as you wish, toward something new.”
- **March 4, 2018** - Jordan, KN4CRD, published a design document for FT8Call: https://github.com/jsherer/ft8call
- **July 6, 2018** - Version 0.0.1 of FT8Call released to the development group
- **July 15, 2018** - Version 0.1 released - a dozen testers
- **July 21, 2018** - Version 0.2 released - 75 testers
- **July 27, 2018** - Version 0.3 released - 150 testers
- **August 12, 2018** - Version 0.4 released - (“leaked” on QRZ) - 500 testers
- **September 2, 2018** - Version 0.5 released - 3000 testers
- **September 14, 2018** - Version 0.6 released - 5000 testers
- **October 8, 2018** - Version 0.7 released - 6000 testers, name changed to JS8 & JS8Call

Notice

JS8Call is a **derivative** of the WSJT-X application, restructured and redesigned for message passing using a custom FT8 modulation called JS8. It is not supported by nor endorsed by the WSJT-X development group. While the WSJT-X group maintains copyright over the original work and code, JS8Call is a derivative work licensed under and in accordance with the terms of the GPLv3 license. The source code modifications are public and can be found in this repository: https://bitbucket.org/widefido/wsjtx/

JS8Call is and will always be **free** software (free as in beer and free as in speech).

You might be asking...why is this named JS8Call? Why *was* it renamed from FT8Call? Why not something else, like BACON or HF Messenger? Good question! It is named this way as an homage to its heritage:

- JS8Call was previously named FT8Call.
- JS8Call uses a custom FT8 modulation called JS8 (Jordan Sherer designed 8-FSK modulation). This is the base RF transport.
- JS8Call has a directed “calling” protocol laid over top the base RF transport to support free-form and directed message passing.

Hence JS8 + Directed Calling = JS8Call.

Download & Install

JS8Call currently comes in a variety of builds.

- Desktop Linux (64-bit x86_64, Ubuntu AppImage)
- Desktop Linux (64-bit x86_64, deb)
- Desktop Linux (32-bit i386, deb)
- Raspbian Stretch (armv7, AppImage)
- Raspbian Stretch (armv7, deb)
- Windows 10 (win32_64)
    - Windows 10 is the only officially supported Windows build at this time, even though the application works all the way back to Windows XP.
- Mac OSX 10.10+ (x86_64)

For the most up-to-date download links, check:

- JS8Call Release Announcements
- JS8Call Release Download Links

Of course, you are always free to take a look at the source code as well!

The application is distributed using the WSJT-X installer on Windows. For Linux, it is distributed as an AppImage, a single file executable that can be run portably on your Linux desktop or

RaspberryPi, as well as a .deb. AppImage is an untraditional approach to distributing Linux software so if you've never dealt with an AppImage before, all you need to do is:

1. Download the AppImage for your platform
2. Set the AppImage file to be executable
3. Run the AppImage file

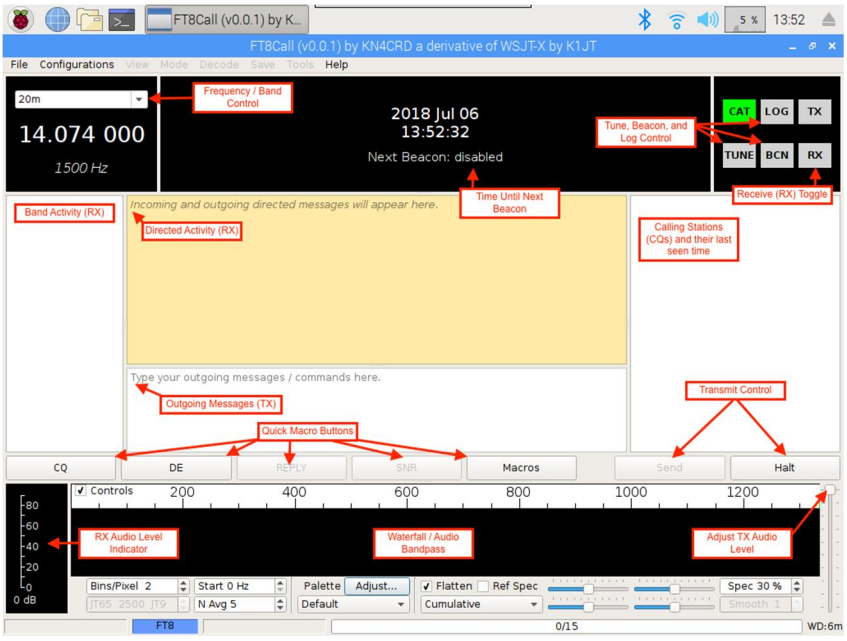
For more information on running AppImages, please check out: https://docs.appimage.org/user-guide/run-appimages.html

NOTE: Until the general release of JS8Call, development versions will only carry a 14 day lifespan. After expiration, you'll be required to upgrade to the latest version of the application. This expiration will go away after a 1.0 general release.

Operating JS8Call

If you've used FSQ, Fldigi or WSJT-X before, you'll feel right at home with JS8Call. The premise is that JS8Call changes the encoding structure of FT8 modulated messages, breaking up long freetext messages into multiple **back to back** transmission frames / cycles.

Here's what you'll see when you start up the application:



Clock Sync / Timing

In the application you can see the current time reported by your PC in UTC format. An accurate clock is important with JS8Call, as the decoder operates within a 15-second window of transmission (frames). Your clock being off greater than 2 seconds from UTC can cause messages to not decode at your station. It is best to use an Internet, NTP or GPS time source for synchronizing your clock as accurately as possible.

NOTE: You don’t actually have to have the exact time synchronized...just the start of every 15-second window. Many operators can manually synchronize their system clock based on signals in the waterfall.

Starting with version 0.7 of JS8Call, there’s a new manual clock sync tool that you can use to drift your curent time to match signals you see / hear (or to an external time source like a Timex watch, a GPS handheld, WWV, or the rooster crowing). This is intended to be used as a fail-safe for when your synchronized time source is not available (like if you were out portable, away from cellular or GPS reception).

USB

Make sure your rig is set to USB mode. If you are running LSB, you’ll likely see reversed signals you cannot decode.

Band / Call Activity

Band activity is displayed on the left. Callsigns you've heard CQing are on the right. Right clicking will show a menu with an option to move your RX/TX offset to that audio frequency (QSY).

Waterfall

There is a waterfall at the bottom of the screen to show you the signals in your audio passband. You can click on the waterfall to set your audio frequency offset.

There is also an option to QSY to that frequency by centering your selected audio offset to the rig passband center. This allows you to use narrow filters easily.

Messages

The top yellow text box shows you messages that are either on the frequency offset you're on or who have directed a message to you (they sent a message that included your callsign).

You type into the white box on the bottom to prepare a message for transmission.

Normal FT8 character restrictions **do not** apply! The extended character set includes all printable uppercase ASCII (A-Z 0-9 Space ./?+-`~!@#$%^&*()_=[]\{}|;’:”,<>). The message structure is variable encoded, so the most common characters take the least amount of space, and special characters take longer to send.

As you type your message you’ll see the send button display the number of frames (15 second transmit cycles) it’ll take to send your complete message. All you have to do is click send (or hit enter) to start transmitting on the next interval. As each frame is transmitted one after the other, the button will update with the number of frames left.

Because of this special variable encoding, messages in JS8Call cannot be decoded by WSJT-X. The same is also true, WSJT-X messages will not be shown in JS8Call.

Messages come in two forms:

1. standard JS8Call free text messages
2. directed JS8Call messages

Standard Messages

Standard messages are freetext messages that do not start with a callsign or a directed command. These messages will only print at another's station location if they align their receive offset within 10Hz of your transmit offset. This is operation similar to other keyboard-to-keyboard digital modes, like Olivia, RTTY, and PSK.

Directed Messages

Directed messages are special JS8Call transmissions that automatically prefix your message with your callsign, similar to how FSQCall operates. Directed messages are useful for communicating in that you do not have to include your callsign in your message, allowing you to use more of the transmission frame(s) for actual message text, as well as alerting the recipient that a message was sent to them. As long as you are in the same passband, you do not have to be on the same frequency offset to receive a directed message.

To send a directed message, all you need to do is include the callsign of the receiving station as the first word in the message (or select a callsign in your heard list).
There is a special "ALLCALL" callsign that you can use to send the message to anybody who is able to receive your message. Some examples:

- DR4CNK HELLO HOW ARE YOU JIM?
  - Will be sent as: **KN4CRD: DR4CNK HELLO HOW ARE YOU JIM? ⌁**
- ALLCALL HELLO NET PSE QSY 14300
  - Will be sent as: **KN4CRD: ALLCALL HELLO NET PSE QSY 14300 ⌁**

You'll notice a special character at the end of the message, ala ⌁. This is called an Electric Arrow, and is a symbol to indicate the End of Transmission. JS8Call displays this as after the last frame of the message has been transmitted with nothing else to follow. This means you get a visual indicator that the transmission is done and you can begin transmitting a reply. It's also the base character in JS8Call's icon.

There are special directed messages that you can send to stations to have them automatically reply if they have AUTO enabled. They are comprised in the form of [CALLSIGN][COMMAND].

Available commands:

- `? - What is my SNR?`
- `@ - What is your QTH (station location)?`
- `& - What is your QTC (station message)?`
- `$ - What stations are you hearing? (Will transmit the top 4 ranked by SNR)`
- `QSO [CALLSIGN]? - Can you communicate directly with CALLSIGN?`

    - If the station ACKs, they will send back the SNR and the last time the callsign was heard at their station
- >message - Please (optionally) relay this message and display it in a reply dialog.
    - The receiving station will display the message in a dialog prompting the receiver to reply with a message.
    - Optionally, this can be used to relay messages between stations by prefixing another callsign:
        - KN4CRD>HELLO!
          (will send the message to KN4CRD)
        - KN4CRD>DR4CNK>HELLO!
          (will send the message to DR4CNK through KN4CRD)
- #message - Please ACK if you receive this message in its entirety
- AGN? - Have the station automatically retransmit their last message
- ---
- QTC - Send station message
- QTH - Send location message
- GRID - Send a long-form grid locator (to be spotted on a map via PSKReporter & APRS-IS)
- ---
- QSL? - Did you receive my last transmission?
- QSL - I received your last transmission
- YES - I confirm your last inquiry
- NO - I negative confirm your last inquiry
- HW CPY? - How do you copy?
- RR - Roger. Received. I copy.
- FB - Fine Business
- QRZ? - Who is calling me?
- 73 - I send my Best Regards / End of Contact

Examples:

If we wanted to ask DR4CNK what his QTH was, we'd send:

- DR4CNK@
    - And he would respond with a directed message back to me: "DR4CNK: KN4CRD MY QTH IS SOUTH OF FRANCE", automatically if AUTO reply is enabled.

You can also use "ALLCALL" with the "?" command and all stations that receive your message will respond on a random frequency offset, example:

- ALLCALL?
    - And all stations who heard your message who are not engaged in a QSO and who have not responded to your ALLCALL during their BEACON interval, would respond with your signal strength:
      "DR4CNK: KN4CRD SNR +21"

If we wanted to transmit a message to OH8STN through DR4CNK, we could use the relay command and send:

- DR4CNK>OH8STN HELLO JULIAN!
    - During relay, at each hop the originating sender's call is appended to the message.
    - The command above would be received by OH8STN, they would send an ACK back, then retransmit the message, like so:
        - KN4 station sends:
            KN4CRD: DR4CNK>OH8STN HELLO JULIAN!
        - DR4 station relays:
            DR4CNK: OH8STN HELLO JULIAN! DE KN4CRD

You can also mix and match standard and free text messages, but most of the time you won't need to.

BEACON - Beacons

There is an automated beacon that transmits on an interval once turned on (BCN button on the top right). This interval can be changed in the settings. There's no protection against the beacon transmitting over a message you're receiving, so you'll have to keep an eye on it. All beacons are transmitted on a random (unused) frequency offset between 500Hz-1000Hz to help prevent QRM.

The intent of beaconing is not to report on propagation...it's to help populate your heard list (on the right) so you know who's likely to be reachable so you can try to make contact. You can't work them if you can't "hear" them (or if they cannot hear you).

When you have BEACON enabled and you first receive a BEACON from a station, your station will try to send a BEACON ACK reply to signal to the other operator that you can hear them. If they receive your response, an indicator will be displayed next to your callsign in their heard list. This helps you find, at a glance, other operators that are confirmed to be able to hear you.

Also, keep in mind that unattended beaconing may be against the rules of your jurisdiction. To be most safe, beacons should only be sent while you're at the control point of your station.

AUTO - Automatic Replies

While AUTO is enabled, the software will automatically respond to directed queries, like "?", "@", and "&". When AUTO is turned off, JS8Call will buffer responses to directed queries in the send message textbox until you are ready to send the replies manually.

If you would like to participate in AUTO, but would not like to be responsible for message relays, you can disable relays while AUTO is enabled in the settings.

LOG - Station Log

There's a log item in the main menu of the application. You can also press F5 to start a log entry. The software will do its best effort to pre-populate log fields. However, you'll likely have to fill out some missing information manually since the QSO is freetext and not automated.

The log is stored in JS8Call.log & JS8Call.adif in the log directory (which you can find by clicking “File -> Open log directory” in the main menu).

Currently, the logging function in JS8Call will log each contact under the JS8 mode. We have not petitioned ADIF yet for inclusion in the mode tables, but will plan to do so before the general release. You will likely not be able to log to LoTW or eQSL until ADIF acceptance.

In the meantime, you can submit the mode as DATA. There is an option in the Logging settings to log the mode as DATA instead of JS8.

Once logged, the selected directed callsign is automatically deselected

SELCALL - Selective Calling

When enabled, SELCALL acts like a filter, preventing ALLCALL and BEACONs from being displayed in your RX text area. When enabled, only directed messages to you will be displayed. While in SELCALL mode, a new GROUPCALL callsign is available to transmit to. It works like ALLCALL, except it will only be printed at stations that are within 100Hz of you. So, if you want to create a round table discussion with a few of your friends, QSY to within 100Hz from each other and send: "GROUPCALL HEY FRIENDS!"

SPOT - Callsign Spotting

When SPOT is enabled, JS8Call will report callsigns you hear (or your callsign if heard by other stations) to PSKReporter under the “JS8Call” mode.

JS8Call will also spot GRID commands with 6 or more characters to APRS-IS / aprs.fi. Make sure to set your grid locator to 6-12 characters for the most accurate spot. You can drill down with this map to your location if you’re unsure of your grid: http://k7fry.com/grid/. If you have a lat/lon, you can also use the lonlat2maiden script here: http://www.jidanni.org/geo/maidenhead/

CQ - Calling CQ

The default way to call cq is with the “CQCQCQ” message. This is configured by default. What’s notable, though, is that you can configure this message in the settings. These are the messages supported to be sent in one 15-second transmission:

- CQCQCQ
- CQ TEST
- CQ QRP
- CQ DX

Each of these messages can also include your 4 digit grid, ala:

- CQCQCQ EM73
- CQ TEST FN04
- CQ QRP JO42
- CQ DX GC28

You can start your CQ message with one of these formats and it will be sent directed, meaning your callsign will automatically be included. You can add to the messages without issue:

- CQ QRP 500MW CQ?

If you deviate from these formats, you will not be sending a directed message and must include your callsign in your message.

REPLY - Replying to a CQ

The default way to reply to a cq is with “HW CPY?”. This allows the caller to choose who to connect with and send a signal report to. You can customize this message with a reply, but keep in mind that most stations will be replying with something that can be send in one 15-second transmission. Here’s an example exchange:

- →**KN4CRD: CQ QRP EM73** ⌁
- ←DR4CNK: KN4CRD HW CPY? ⌁
- →**KN4CRD: DR4CNK SNR -12 TU 4 CALL QSL?** ⌁
- ←*DR4CNK: KN4CRD RR -22 FB INTO GO28 GUD QRP DX!* ⌁

SAVED - Saved Messages

There are a few quick saved message buttons for transmitting common messages. You can edit these in the settings window. Just be mindful that long messages will take a while to send.

Frequencies

Most operators testing the application can be found +/- 4-8kHz from the standard FT8 frequencies. It is essential to avoid the main FT8 frequencies, as that will cause confusion among WSJT-X operators. Here are some suggested frequencies to use:

- `160m:  1.842 MHz // 2kHz above FT8`
- `80m:  3.578 MHz // 5kHz above FT8`
- `40m:  7.078 MHz // 4kHz above FT8`
- `30m: 10.130 MHz // 6kHz below FT8`
- `20m: 14.078 MHz // 4kHz above FT8`
- `17m: 18.104 MHz // 4kHz above FT8`
- `15m: 21.078 MHz // 4kHz above FT8`
- `12m: 24.922 MHz // 9kHz above FT8`
- `10m: 28.078 MHz // 4kHz above FT8`
- `6m: 50.318 MHz // 5kHz above FT8`

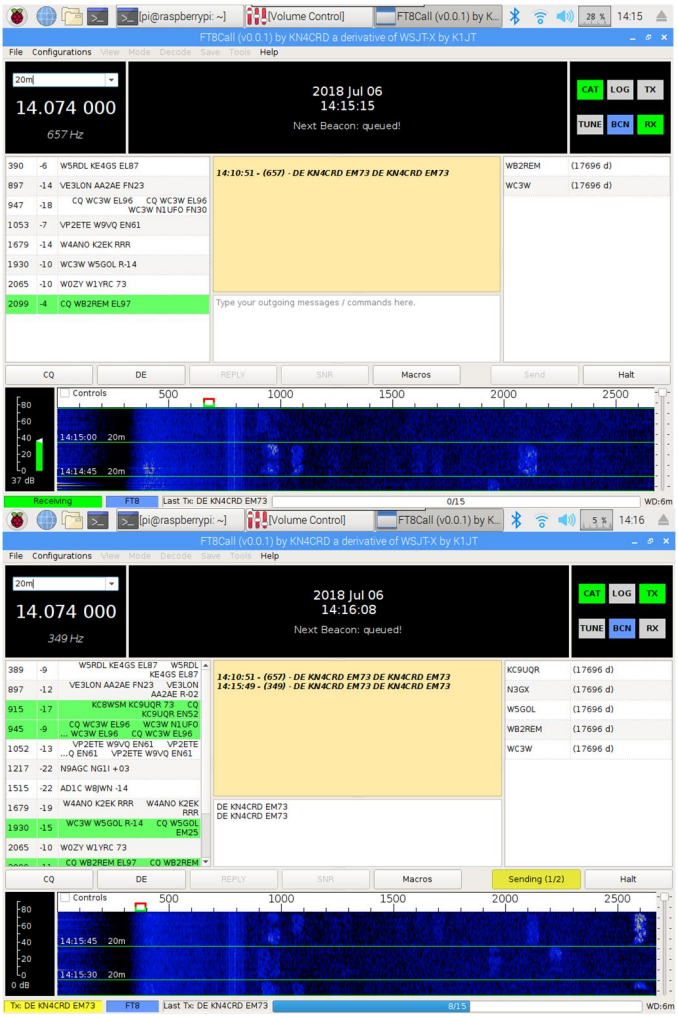
You might notice a few of these being close to the JT9 frequencies. Don’t grab your pitchforks! JS8Call blocks out the lower 500Hz of the passband, which is enough room for 25 simultaneous JT9 signals.

But also, please keep in mind these are only suggested frequencies and **are subject to change while in development**. We all have VFOs, so please use them. Just remember to be good operators and prevent from interfering with other signals on our shared bands.

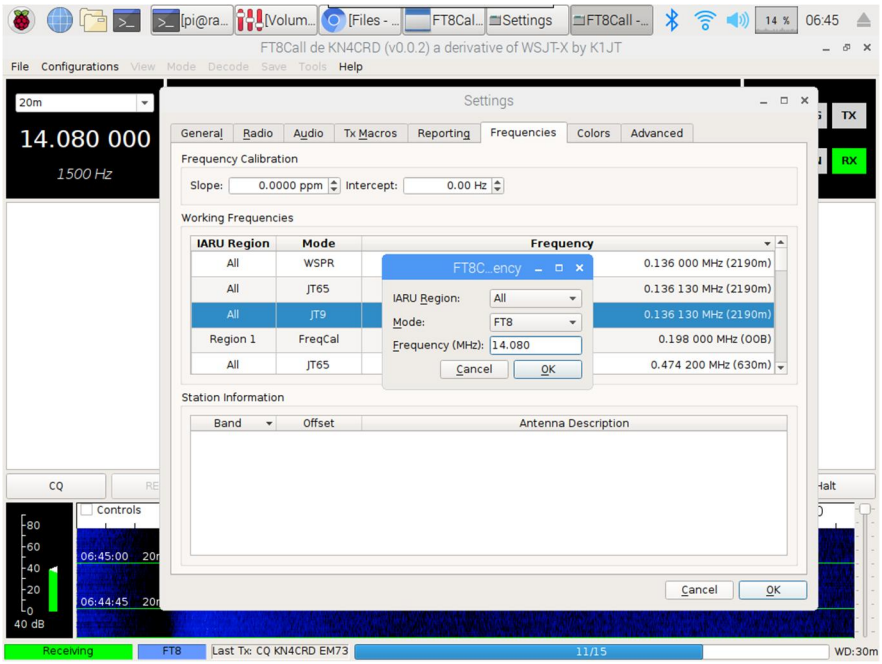
You **CAN** type into the frequency dropdown list to change to a different frequency. JS8Call will not limit which frequencies you can transmit on.

You can use the mailing list Sked Chat or the Facebook group to schedule on other frequencies with test operators.

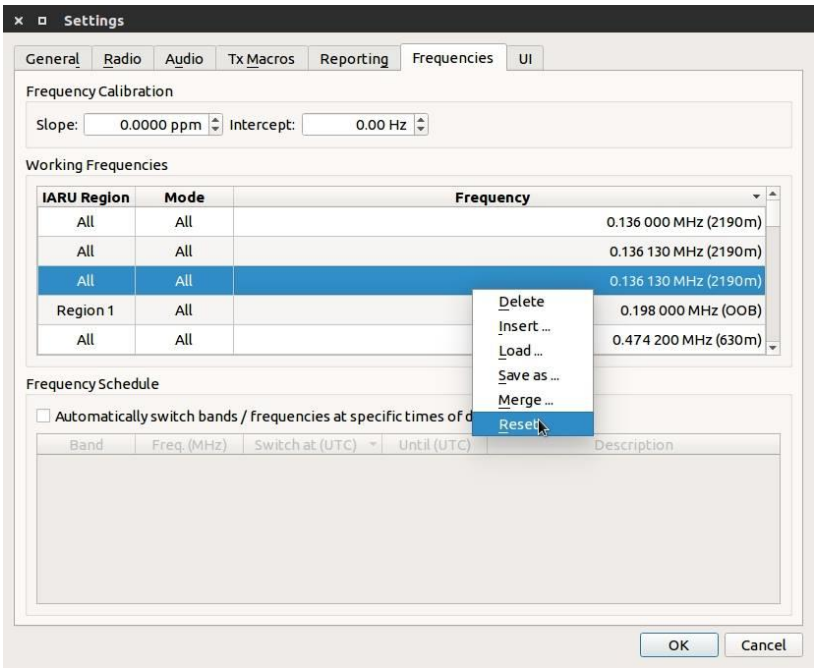
If you want to transmit on a non-standard frequency (recommended) you can either modify the frequencies list in the settings, or you can type directly into the band dropdown box in the top left of the screen.



If you'd like to add custom frequencies for JS8Call, you can do so in the settings:



If you'd like to reset to the suggested frequencies, right click the frequencies box and click Reset.



Tips & Tricks

- An example QSO:
  - →**KN4CRD: CQCQCQ EM73** ⌁
  - ←DR4CNK: KN4CRD SNR +01 GOOD SIGNAL ⌁

    - →**KN4CRD: DR4CNK SNR -12 TU 4 CALL RIG IS KX2 5W DIPOLE** ⌁
    - ←*DR4CNK: KN4CRD RR -22 FB KX3 100W VERT* ⌁
    - →**KN4CRD: DR4CNK RR FB REALLY ENJOYING THE CHAT MODE WITH LONG MESSAGES. BUT HEY LET'S TRY A RELAY** ⌁
    - *… (and on, and on, if you want)*
    - →**KN4CRD: DR4CNK 73** ⌁
    - →**KN4CRD: CQCQCQ EM73** ⌁
- You do not need to include your callsign when initiating your directed replies. They will be prefixed to your message automatically when you have a callsign selected in your heard list.
- You do not have to reply on the same frequency offset as the caller. But, if you're calling another station off their frequency, you need to include their callsign at the beginning of the the message so it is directed to them and will show up in their yellow directed activity window.
- Directed messages pack as much data as standard FT8 frames. The following examples are all 1 transmit cycle long
    - Example:
        - KN4CRD/P: CQCQCQ EM73 (1 transmit frame)
        - VE3/KN4CRD: CQ QRP EM73 (1 transmit frame)
        - KN4CRD: ALLCALL? (1 transmit frame)
        - DR4CNK: KN4CRD SNR +15 (1 transmit frame)
        - DR4CNK: KN4CRD AGN? (1 transmit frame)
- For replying to a station's CQ, double click their call in the call activity window, then either choose a directed command or type a message to them:
    - DR4CNK: KN4CRD HW CPY?
    - DR4CNK: KN4CRD SNR +12
    - DR4CNK: KN4CRD YES
    - DR4CNK: KN4CRD NO
    - DR4CNK: KN4CRD RR
    - DR4CNK: KN4CRD 73
    - DR4CNK: KN4CRD HELLO MY FRIEND GREAT TO HEAR YOU!
- You can send freetext at any time! That's what JS8Call was designed for:
    - HI JIM TU 4 CALL UR -12 INTO ATLANTA BTU DE KN4CRD (4 transmit frames)
- It might be helpful to learn some of the morse code prosign/abbreviations and psk31 abbreviations:
    - https://en.wikipedia.org/wiki/Prosigns_for_Morse_code
    - http://www.hamblog.co.uk/common-psk31-abbreviations/
    - Examples:
        - K - over
        - BTU - back to you
        - FB - fine business
        - HW? - how do you copy?

Frequently Asked Questions

- What are the random three (or six) characters at the end of relay and acked message commands?

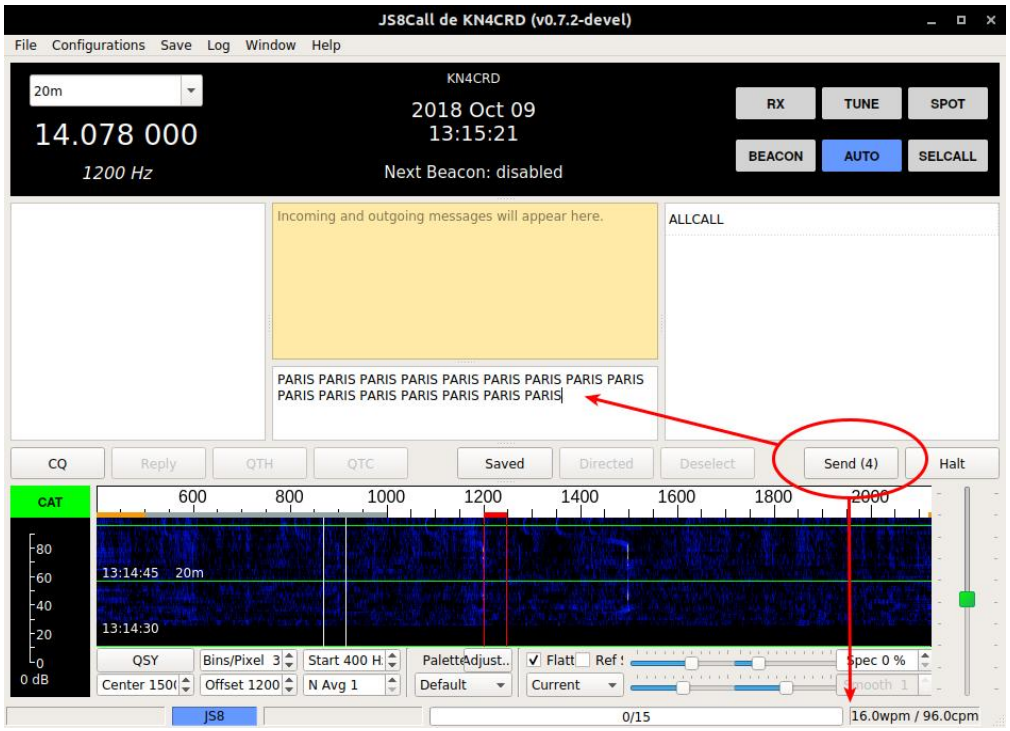
    - These are a checksum for the message added to ensure all of the message frames were delivered correctly before retransmitting / alerting. If received in its entirety by the receiving station, these checksums will not be displayed to them.

- Beacons transmit back to back (30 seconds). Why? Is this normal?
    - Yes. They currently transmit for 30 seconds to compensate for band fading, qrn, or stations transmitting over each other.
    - In the future, beacons that transmit for 30 seconds may also be able to offer extra information during the beacon (things like compound callsigns, extra long grid locators for telemetry, station power, etc)

- You said that all printable uppercase ASCII characters can be used. Do some take more time to send than others?
    - Yes. The characters that are sent in the messages are variable encoded, ranging from 3 to 19 bits in length based on their probability of being used in a sentence. The most common characters take the least amount of space, allowing us to send more than 13 characters per transmission cycle on average.
    - Example: Space and E are only 3 bits in length. You could send about 23 (!!) of them in a single transmission. Whereas a character like { is 14 bits in length, you could only send 4 of those. (But really, how frequently do you use that character?)
    - Here are some examples of phrases that could be sent in one 15 second transmit cycle:
        - EEEEEEEEEEEEEEEEEEEEE (22 characters)
        - I HAVE EATEN A SHOE (20 words per minute)
        - WHICH WAY TO OHIO (16 words per minute)
        - NEVER HAVE I EVER (16 words per minute)
        - TU UR 599 4A GA (20 words per minute)
        - Etc

- How fast does JS8Call transmit?
    - JS8Call uses the same 15-second transmission cycle as FT8. What is different is that due to the variable encoding of the characters, JS8Call can transmit up to 23 characters per transmission frame. For average sentences, JS8Call can pack words very tightly, at around 10-15WPM.
    - Example:
        - "WE HOLD THESE TRUTHS TO BE SELF-EVIDENT THAT ALL MEN ARE CREATED EQUAL THAT THEY ARE ENDOWED BY THEIR CREATOR WITH CERTAIN UNALIENABLE RIGHTS THAT AMONG THESE ARE LIFE LIBERTY AND THE PURSUIT OF HAPPINESS."
            - This phrase is 35 words. It would take 11 transmission cycles to send (2 minutes 45 seconds). That is just under 13 WPM.
        - “A SUCCESSFUL MAN IS ONE WHO CAN LAY A FIRM FOUNDATION WITH THE BRICKS OTHERS HAVE THROWN AT HIM”
            - This phrase is 19 words. It would take 5 transmission cycles to send (1 minute 15 seconds). That is just over 15WPM.
        - “TEST THIS IS A TEST”
            - This phrase is 5 words and takes 1 transmission cycle to send (15 seconds). That is 20 WPM

- Morse code has a neat way of calculating WPM, timing how long it takes to transmit the word PARIS. In JS8Call, PARIS is encoded into 17 bits (3.4 bits/character). Each transmission cycle can pack up to 69 character bits. That equates to about 16 WPM. (69/17=4.05 words / (15 seconds * 4))
- The app shows this in the status bar:



- Isn't 10-20 WPM too slow to have a conversation?
  - If propagation is good enough for a faster mode, you should be using it instead! But, with poor conditions like we have experienced at solar minimum, JS8Call might just be the best balance.
  - It may seem really slow (and it is, relatively speaking). However, FT8 modulation is able to decode (theoretically) down to -24dB below the noise floor. Not many modes can say this, especially those which transmit at faster speeds. What does this mean? JS8Call may work when other modes cannot.
  - **We believe that communicating slowly is better than not communicating at all.**

- What is the JS8Call Relay Challenge?
  - This is a friendly competition to maximize the number of continents one can pass a message back and forth to using the retransmit command.
  - We'll be giving away an award (and prize) to the first team of operators to successfully relay a message from one continent across two other continents (NA, SA, EU, AF, AS, OC, AN) and relay an ACK back to the original station using JS8Call. All you need to do is submit your logs from each station and optionally photographic/video documentation of your effort.
  - For example, this is what the outgoing and incoming messages could be:
    - LB9YH>KN4CRD>VK1MIC QSL?

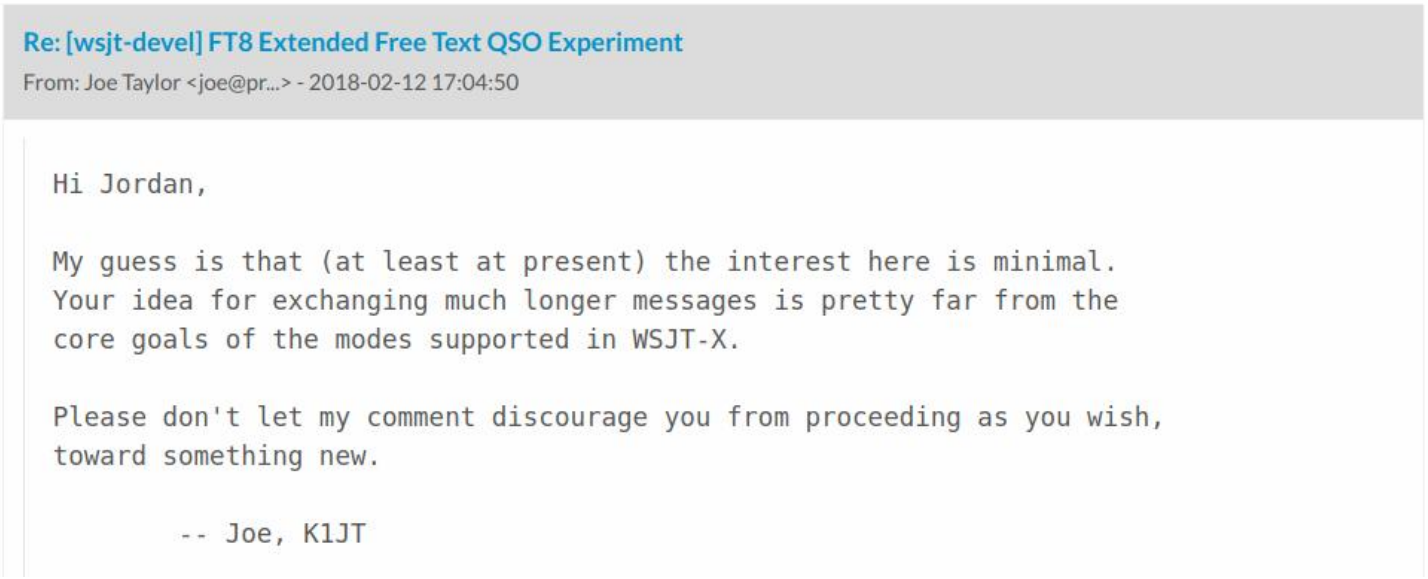
        - VK1MIC>KN4CRD>LB9YH QSL

- Does BEACON mode violate FCC 97.221 Automatically Controlled Digital Station rules in the United States?
    - For operators in the United States, here's a reference to the rules: http://www.arrl.org/part-97-text
    - With this, keep in mind:
        - 1) The control operator is responsible for the station operation. The software makes a best effort to require a human to be present during operation (beacon off by default, a watchdog timer feature built-in, etc). It is up to the operator to make sure they are in compliance with the rules of their jurisdiction.
        - 2) Responses to directed queries by non-automatic stations fall under §97.221.C.1 exemption.
    - So, my recommendation to operators is to turn off BEACON when not at the station control point, but, they can feel comfortable leaving AUTO on while they are away since their station would only be responding to queries initiated by a non-automatic station.

- Why isn't my station responding to ALLCALL?
    - Previous versions of JS8Call (FT8Call) had a directed message of "ALLCALL?" that had stations return SNR reports automatically. This has been replaced, starting in version 0.7 of JS8Call, with BEACON and BEACON ACKs. Stations will no longer respond to the "ALLCALL?" query.

- I love what you're doing. Do you have a PayPal or Patreon where I can I send you a donation as a "Thank you?"
    - I appreciate the gesture! I continue to work on this project as a donation of my time to the Amateur Radio ecosystem. I'm not looking for payment of any kind. If you feel so obliged, however, I'd appreciate if instead you sent along any donation you'd like to make to a local charity of your choosing. Something like the American Red Cross, Salvation Army, or even a local Amateur Radio club. They'd put that money to far better use!

- What does Joe Taylor, K1JT (or the WSJT-X development team) think of JS8Call?
    - We have not heard anything from him/them, so you'll have to reach out and ask!
    - However, as you can see in the History section at the start of the document, I did receive acknowledgement from Joe before pursuing the JS8Call project back in February 2018:



Re: [wsjt-devel] FT8 Extended Free Text QSO Experiment

From: Joe Taylor <joe@pr...> - 2018-02-12 17:04:50

Hi Jordan,

My guess is that (at least at present) the interest here is minimal.
Your idea for exchanging much longer messages is pretty far from the
core goals of the modes supported in WSJT-X.

Please don't let my comment discourage you from proceeding as you wish,
toward something new.

-- Joe, K1JT

via https://sourceforge.net/p/wsjt/mailman/message/36224507/

Troubleshooting

If you're having trouble, head over to the troubleshooting chatroom for help: https://groups.io/g/JS8Call/chat/1423 or email Jordan directly: kn4crd@gmail.com

Common Problems & Solutions

JS8Call will not run on my system
Make sure you are running a supported operating system, that you have disabled any programs that may be using your audio device, or preventing JS8Call from using the audio device...like an aggressive antivirus. If you're running Windows, and have a Windows Defender running, you'll need to either whitelist JS8Call or turn off the defender.

I see signals on the waterfall but I cannot decode them
Make sure the signals you are seeing are actually JS8Call signals and not FT8 signals (they are incompatible) by ensuring you're on one of the JS8Call frequencies. Make sure you are in Upper Sideband (USB) mode. Make sure you have synchronized your clock to within 2 seconds of UTC. Make sure you're not running WSJT-X at the same time.

I do not see any signals on the waterfall
Check your incoming audio from your rig. Make sure JS8Call audio is configured correctly. Check to make sure you're on one of the JS8Call frequencies. Keep in mind that JS8Call is still in development and has more than an order of magnitude fewer operators on the air. There may actually be nobody on within your reception range. Check PSKReporter to see if there are others on the band. If you still cannot see any signals, either:

1. You have an RX problem with your station
2. None of the operators are operating on the band you are on

3. Or propagation isn't being friendly to you

NOTE: Keep in mind that JS8Call isn't magic...we're still at the mercy of the ionosphere.

My rig won't transmit
Check your outgoing audio to your rig. Make sure JS8Call audio is configured correctly. Unplug the rig from the computer and hook up the output to a set of headphones or speakers. Try to transmit, maybe with the TUNE button in the app. Can you hear the tones? If not, then you have an audio problem, if so then you have a transceiver problem. Make sure your PTT is configured correctly for your rig or use VOX. You can test this in the settings. The PTT button will turn green if it can key your transmitter. If you have audio into the rig, but still have no RF out, make sure your rig is configured correctly by checking your digital gain / tx gain / mic levels. After that, make sure your rig works...switch over to FM or CW and send a carrier to make sure the rig can actually transmit at all.

Bug Reports

You can send bug reports to Jordan Sherer (KN4CRD) at kn4crd@gmail.com or in the troubleshooting chatroom in the groups.io page: https://groups.io/g/JS8Call/chat/1423

Inside Jokes

Many of the testers have come to use the term "Baconing" instead of "Beaconing".

API Definition

JS8Call uses a JSONRPC API offered through UDP. It is currently highly experimental and subject to drastic change in the future (like, for instance, moving to a HTTP or XMLRPC implementation instead).

Once released, the API will allow you to:

- `RIG.GET_FREQ - Get the current Frequency`
- `RIG.SET_FREQ - Set the current Frequency`

- `STATION.GET_CALLSIGN - Get the current callsign`
- `STATION.GET_GRID - Get the current grid locator`
- `STATION.SET_GRID - Set the current grid locator`
- `STATION.GET_QTC - Get the current station message`
- `STATION.SET_QTC - Set the current station message`

- `RX.GET_CALL_ACTIVITY - Get the current heard list`
- `RX.GET_BAND_ACTIVITY - Get the current activity in the band activity list`

- `RX.GET_TEXT - Get the text from the yellow rx box`

- `TX.GET_TEXT - Get the text in the tx box`
- `TX.SET_TEXT - Set the text in the tx box`
- `TX.SEND_MESSAGE - Send a message`

- `LOG.QSO - QSO has been added to the JS8Call log`

- `WINDOW.RAISE - Bring the window to the foreground`

Technical Implementation

JS8Call is under active development and details about the technical implementation are subject to change. Detail will be added here as the implementation stabilizes. Until then, the code is the source of truth for the implementation.

Modulation

JS8Call uses FT8 modulation as the base transport for data. Being a derivative of WSJT-X, JS8Call heavily leverages the work by the WSJT-X Development Group on the FT8 mode. Very few modifications have been made (see source code for the exhaustive list) to the base FT8 modulation, with the exception of two important items:

1. Modifying the CRC algorithm to prevent JS8Call from interfering with FT8 signals
2. And allowing all 75 bits to be utilized for data transport

Protocol

The JS8Call protocol sits at a layer above the base transport. Much of the implementation is inspired by the design document: https://github.com/jsherer/JS8Call with a few deviations from the original proposal.

Messages in JS8Call are transmitted in 15-second intervals (frames), with each frame being classified as one of 6 types:

1. Beacon
2. Compound Callsign Partial
3. Compound Callsign Directed Command
4. Directed Command
5. Data Padded
6. Data Unpadded

Further, each frame includes a flag identifying it as:

1. Default Frame (any frame)
2. First Frame (first frame of the transmission)
3. Last Frame (last frame of the transmission)
4. Reserved (for future use)

And finally, there are special encodings for:

1. Callsigns
2. Callsign Prefix/Suffixes
3. Signal Reports
4. Power Levels
5. Grids

Beacon

Beacon frames are comprised of:

- Beacon Type (BEACON or CQ)
- Compound Callsign
- Grid

Compound Callsign Partial

Compound callsign partials are used as one-half of a 2-frame compound transmission when one of the stations includes a compound callsign. Compound callsign partials are always the 1st frame in a 2-frame compound transmission, encoding the “from” portion of a directed command with compound callsigns.

The frame includes:

- Callsign
- 4 character alphanumeric prefix or suffix (A-Z 0-9)
- Grid or Numeric Value (SNR or PWR)

Compound Callsign Directed Command

Compound callsign directed commands are a special case for compound callsign partials where the numeric value encodes a directed command to be used with a compound directed message. It is one-half of a 2-frame compound transmission. Compound callsign directed commands are always the 2nd frame in a 2-frame compound transmission, encoding the “to” portion of a directed command with compound callsigns.

The frame includes:

- Callsign
- 4 character alphanumeric prefix or suffix (A-Z 0-9)
- Directed Command

Directed Command

Standard callsigns can send a directed command in one frame.

The frame includes:

- From Callsign
- To Callsign
- Directed Command
- Numeric Value

Data

Data frames are the backbone for long-form messages in JS8Call. They are 75-bit frames that use a variable encoding to pack character data into the smallest transmission possible.

Data frames come in two flavors:

- Unpadded: All bits are used for character data, with no pad bits at the end.
- Padded: The character data is less than the frame size, so pad bits are added. The pad bits are a series of 1-bits up until the first 0-bit. The rest of the bits from the beginning of the frame to that point are data.

Data frames may need to include pad bits because of the variable encoding that character data uses for packing. The variable encoding used is a modified Huffman code that represents the most common characters (based on their frequency of observation in most texts) in fewer bits than less common characters, with the option to shift in alternate alphabets.

The complete modified Huffman code is located in Appendix A.

Callsigns

Callsigns are encoded in 28-bits as described in: EME 2000 - http://www.ka9q.net/papers/eme-2000.ps.gz

Callsign Prefix / Suffix

Prefixes and suffixes are 4 character alphanumeric encoded in 21-bits with a 1-bit flag to indicate whether or not it is a prefix or suffix. Alphanumeric digits can each be encoded in 5.25 bits (there are only 1,874,161 combinations of 4 character alphanumeric prefix/suffix, which is less than can be represented in a 21-bit number 2^{21} = 2,097,152)

Grids

Grids are encoded in 15-bits as described in: http://physics.princeton.edu/pulsar/k1jt/wsjtx-doc/wsjtx-main-1.7.0.html#PROTOCOL_OVERVIEW

Appendix A: Code Table

Huffman Code:

Character code weighted by frequency

" "	"01"
"E"	"100"
"T"	"1101"
"A"	"0011"
"O"	"11111"
"I"	"11100"
"N"	"10111"
"S"	"10100"
"H"	"00011"
"R"	"00000"
"D"	"111011"
"L"	"110011"
"C"	"110001"
"U"	"101101"
"M"	"101011"
"W"	"001011"
"F"	"001001"
"G"	"000101"
"Y"	"000011"
"P"	"1111011"
"B"	"1111001"
"."	"1110100"
"V"	"1100101"
"K"	"1100100"
"-"	"1100001"
"+"	"1100000"
"?"	"1011001"
"!"	"1011000"
"\""	"1010101"
"X"	"1010100"
"0"	"0010101"
"J"	"0010100"
"1"	"0010001"
"Q"	"0010000"
"2"	"0001001"
"Z"	"0001000"
"3"	"0000101"
"5"	"0000100"
"4"	"11110101"
"9"	"11110100"
"8"	"11110001"
"6"	"11110000"
"7"	"11101011"
"/"	"11101010"

(s,c)-Dense Code:

See jsc.h, jsc.cpp, & jsc_map.cpp in the source repository for the complete dense code table.