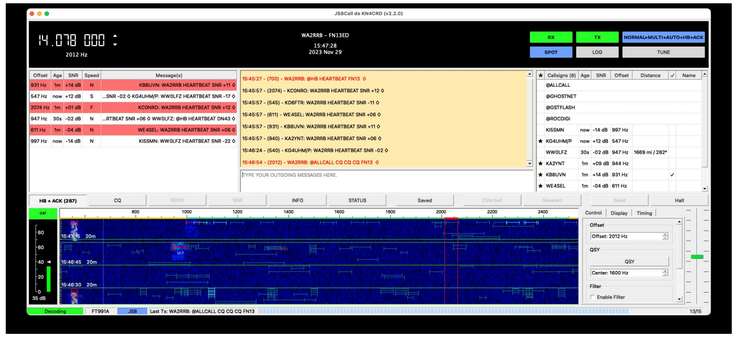


JS8 & JS8Call Practicalities

installation and basic use of JS8Call & supporting software packages

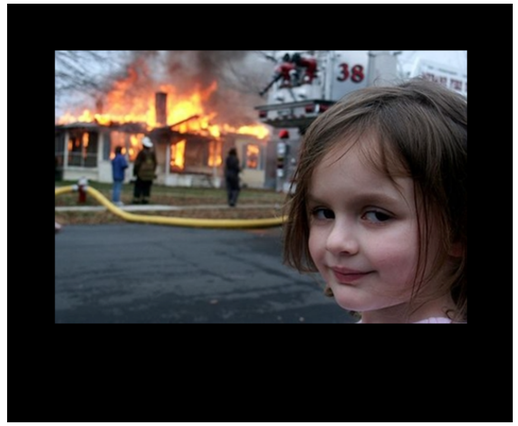
Jeffrey Sonstein - WA2RRB
(presentation for JAN 2024 Rochester Amateur Radio Association meeting)

Don’t Worry, Be Happy

and please hold your questions to the end

- The last section of this live presentation will be a Q&A session

- And yes, there will be a live demo



JS8, the Protocol

“Just what we need, Yet Another Digital Protocol”

- Where does JS8 come from, and why?



JS8, the Protocol

“Just what we need, Yet Another Digital Protocol”

- Where does JS8 come from?
    - “JS8Call is an experiment in combining the robustness of FT8 (a weak-signal mode by K1JT) with a messaging and network protocol layer for weak signal communication. The open source software is designed for connecting amateur radio operators who are operating under weak signal conditions and offers real-time keyboard-to-keyboard messaging, store-and-forward messaging, and automatic station announcements. 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 shoulders of giants...the takeoff angle is better up there.” - *https://www.delta25.de/JS8-2021-11/JS8Call_Guide.pdf*

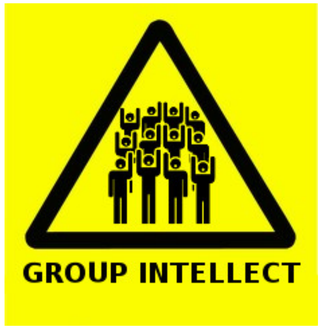
JS8, the Protocol

"Just what we need, Yet Another Digital Protocol"

- Why Yet Another Protocol?
  - "JS8Call is a derivative of the WSJT-X application, restructured and redesigned for message passing". Note the key objectives:
    - real-time keyboard-to-keyboard messaging (*conversational* QSO)
    - store-and-forward messaging (think *EmComm*)
    - automatic station announcements/requests/responses (*ad hoc* networks)

JS8, the Protocol

"Just what we need, Yet Another Digital Protocol"



- Why do I like it?

    1. It is *conversational*

In some ways, it reminds me of using Morse to communicate. JS8 encourages the use of short back-and-forth sentence-like transmissions, with no support for the **LineFeed** or **CarriageReturn** characters.

JS8 is not RTTY, nor is it PSK31.

JS8, the Protocol

“Just what we need, Yet Another Digital Protocol”



- Why do I like it?

    2. store-and-forward messaging (think *EmComm*)

Post-emergency writeup after writeup shows us that depending upon the FM repeater network continuing to operate is an error. When weather is sufficiently bad that electrical towers go down, so do repeater towers, for example. And yet, messages must flow between stations which may not be able to directly copy one another.

Ask any former military operator with experience operating tactical/portable stations, and they will confirm: ***depending upon humans to accurately relay messages from one station to another in an emergency is problematic at best***.

JS8, the Protocol

“Just what we need, Yet Another Digital Protocol”



- Why do I like it?

    3. automatic station announcements/requests/responses

By emitting a *HeartBeat*, a station can get an automated signal-to-noise-ratio (SNR) report as a response from whoever hears that *HeartBeat* signal (and has their station configured to respond automatically, which is the norm)

By seeing who responds to a *HEARING?* message, a station can tell who might be able to relay to another station they cannot directly contact… etc. There are a number of other built-in short-packet-size requests a station can use for, amongst other things, building *ad hoc* networks.

JS8, the Protocol

“Just what we need, Yet Another Digital Protocol”



- Why do I like it? (Misc page)

4. JS8 is perfectly happy with low-power/weak-signal and very noisy conditions

5. JS8 allows you to receive/decode from many stations simultaneously

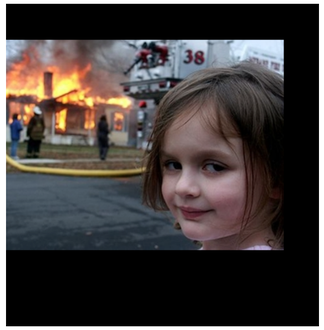
6. JS8 has pre-determined & well-documented (but non-mandatory) “meeting frequencies”, one per band

JS8, the Protocol

“Just what we need, Yet Another Digital Protocol”

- Remember:

  there shalt be demo



JS8Call, the Software Package

Basic installation & configuration issues



- Background:
  - *separation of concerns* as problem solving approach
  - The *layers model* (aka *“divide and conquer”*) turns a big monolithic problem into a bunch of smaller & more manageable problems
  - IMHO good engineers are lazy engineers

JS8Call, the Software Package

Basic installation & configuration issues

- ***Religious Materials Warning***
  - My preference is to run free and open-source software (aka FOSS) whenever I can. Examples of FOSS which I use:
    - flrig (for rig control)
    - fldigi (for PSK31 and auto-generated Morse)
    - JS8Call (for JS8)

JS8Call, the Software Package

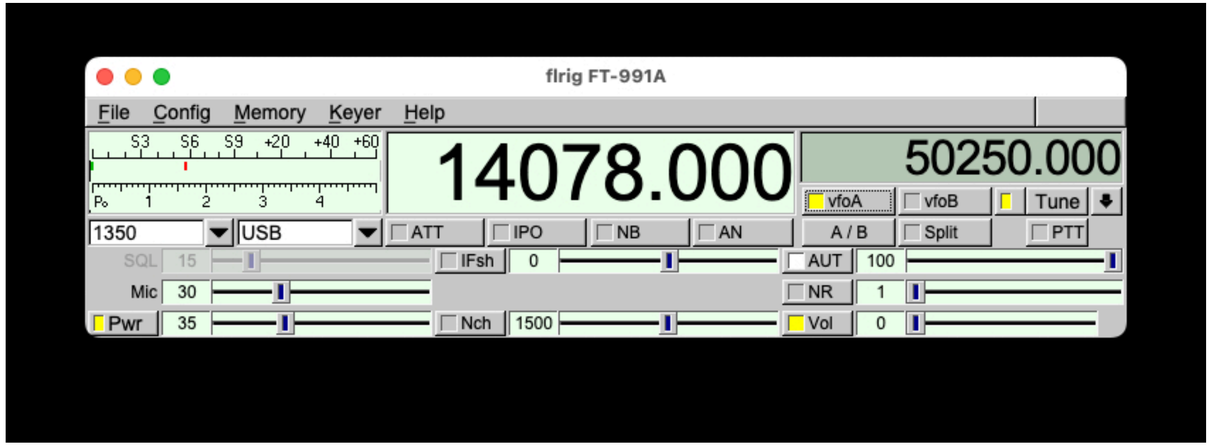
Basic installation & configuration issues

- ***Religious Materials Warning***
  - I prefer delegating rig control to flrig & thus configuring that “layer” once, vs a rig-config process within each s/w package
  - I think about *hamlib* as the *hidden config* alternative
  - I try to avoid the dreaded *Not Invented Here Syndrome*

flrig for rig control

What it should look like, if installed properly and working right

- Down-Arrow revealing *power* & *volume* buttons
- Overdriving warning
- *AGC* off
- *RF* Gain
- *“Noise Reduction”* off
- *“Vol”* aka *AF* Gain aka “*avoid annoying my spouse*”

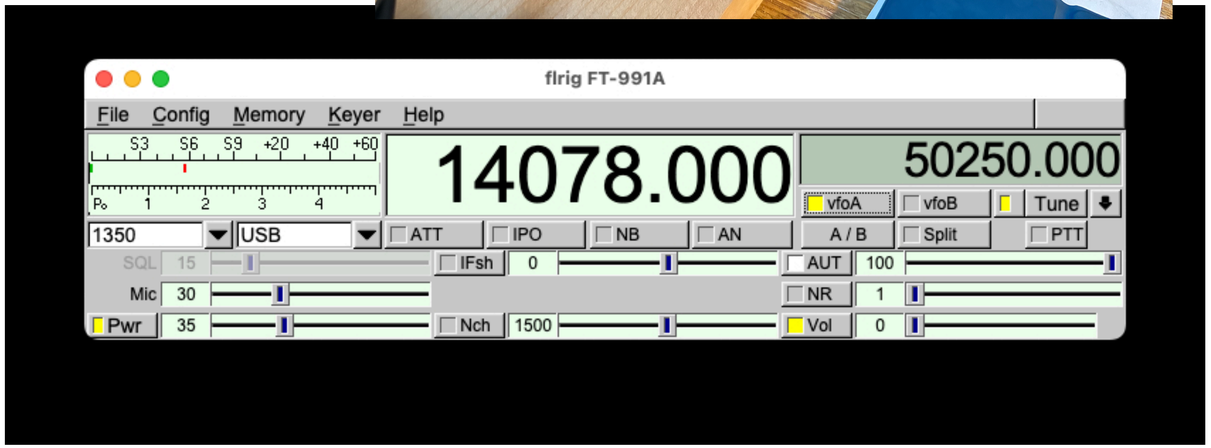


flrig for rig control

What it should look like, if installed properly and working right

- Why do I like to move so much stuff off the actual rig's front panel?
  - *Reduces* hand/arm/eyes/ *attention swings* between screen/mouse/keyboard & rig front panel
  - *Avoids tight* FT-991A *physical controls* (adjusting RF gain, a common task, I keep hitting the frequency control knob)

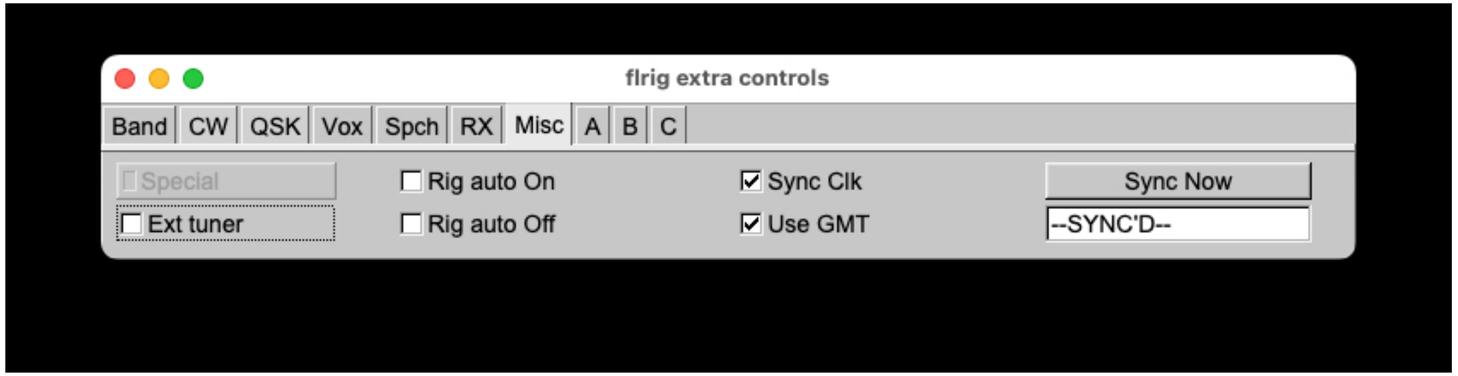




flrig for rig control

What it should look like, if installed properly and working right

- “additional controls”
  - Sync your clock
  - Use GMT
  - Auto on/off… your mileage may vary, depending upon rig and security needs
- Remember: you can *leave this on between stopping one app & starting another*, like from JS8Call to fldigi

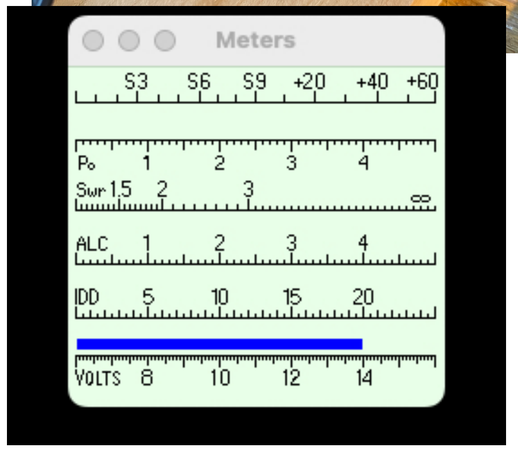


flrig for rig control

What it should look like, if installed properly and working right

- “Meters” display
  - Power out, SWR, and ALC all wrapped up in one display (Yeasu requires cycling through one meter displaying at a time)
  - Again, *reduces* hand/arm/eyes/*attention swings* between screen/mouse/keyboard & rig front panel





JS8Call

Basics

- What does JS8 (the protocol) combined with JS8Call (the application) do that enables success in the face of
  - low power
  - high noise
  - and long distances?

JS8Call

Basics: Forward Error Correction



- What does JS8 (the protocol) combined with JS8Call (the application) do that enables success in the face of low power, high noise, and long distances?
  - Uses *Forward Error Correction*

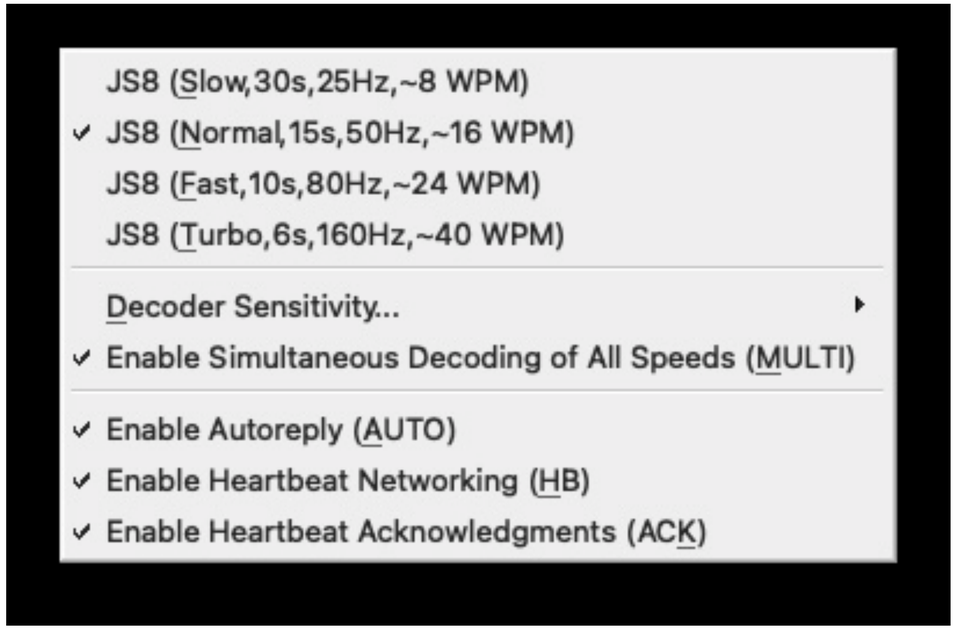
"In computing, telecommunication, information theory, and coding theory, forward error correction (FEC) or channel coding is a technique used for controlling errors in data transmission over unreliable or noisy communication channels... the sender encodes the message in a redundant way, most often by using an error correction code or error correcting code... allows the receiver not only to detect errors... but often to correct... a reverse channel to request re-transmission may not be needed. The cost is a fixed, higher forward channel bandwidth."

(from *Wikipedia*)

JS8Call

Configuration Basics: Modes

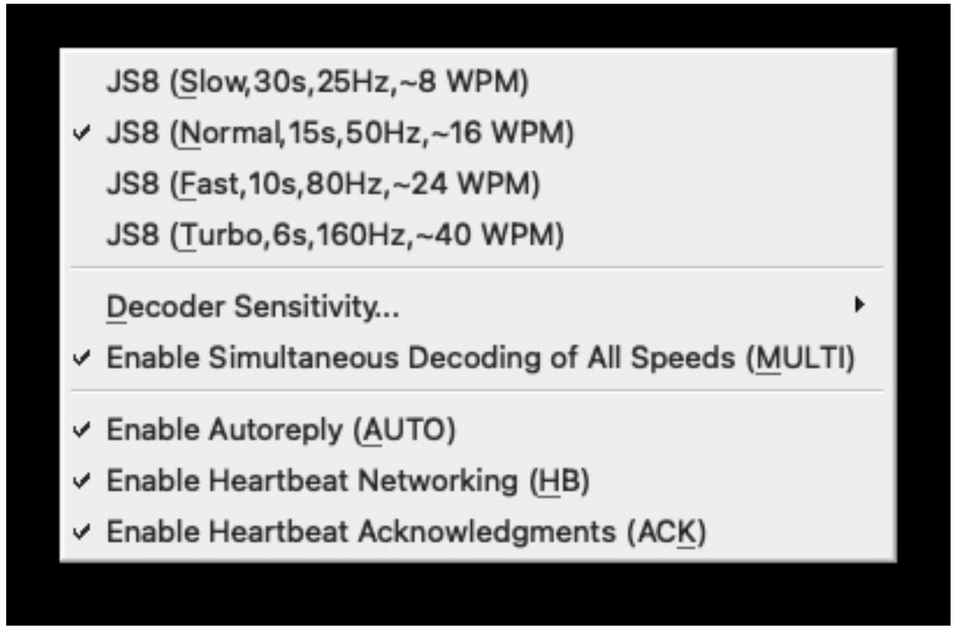
- What does JS8 (the protocol) combined with JS8Call (the application) do that enables success in the face of low power, high noise, and long distances?
  - Transmits in 6-30 second cycles, depending upon "Mode"
    - **Slow** 30s, 25HZ, ~8wpm
    - **Normal** 15s, 50Hz, ~16wpm
    - **Fast** 10s, 80Hz, ~24wpm
    - **Turbo** 6s, 160Hz, ~40wpm



JS8Call

Configuration Basics: Decoding & Automation

- What does JS8 (the protocol) combined with JS8Call (the application) do that enables success in the face of low power, high noise, and long distances?
  - Allows various sensitivity levels (think “false positive decode attempts”)
  - Enables simultaneous decoding of multiple channels
  - Provides for automated queries/replies



JS8Call

Configuration Basics: Time

- Note that reasonably accurate and non-drifting timekeeping by your computer is critical for JS8/JS8Call to operate correctly (talk about Windows issues, Unix-derived systems like MacOS, and software like ***nntp***)



JS8Call

Operations Basics: Typeahead Buffer

- Talk about the "typeahead buffer”

Note that the text in the white area (where it says “type your outgoing messages here”) which has not yet been transmitted *may be edited* “live”



JS8Call

Operations Basics: Messages Sent/Received

- Talk about the “messages sent/received buffer”

red text in the ***yellow*** area displays what you have sent

black text in the ***yellow*** displays what you want to receive (heartbeats, CQs, @GROUP messages, etc)



JS8Call

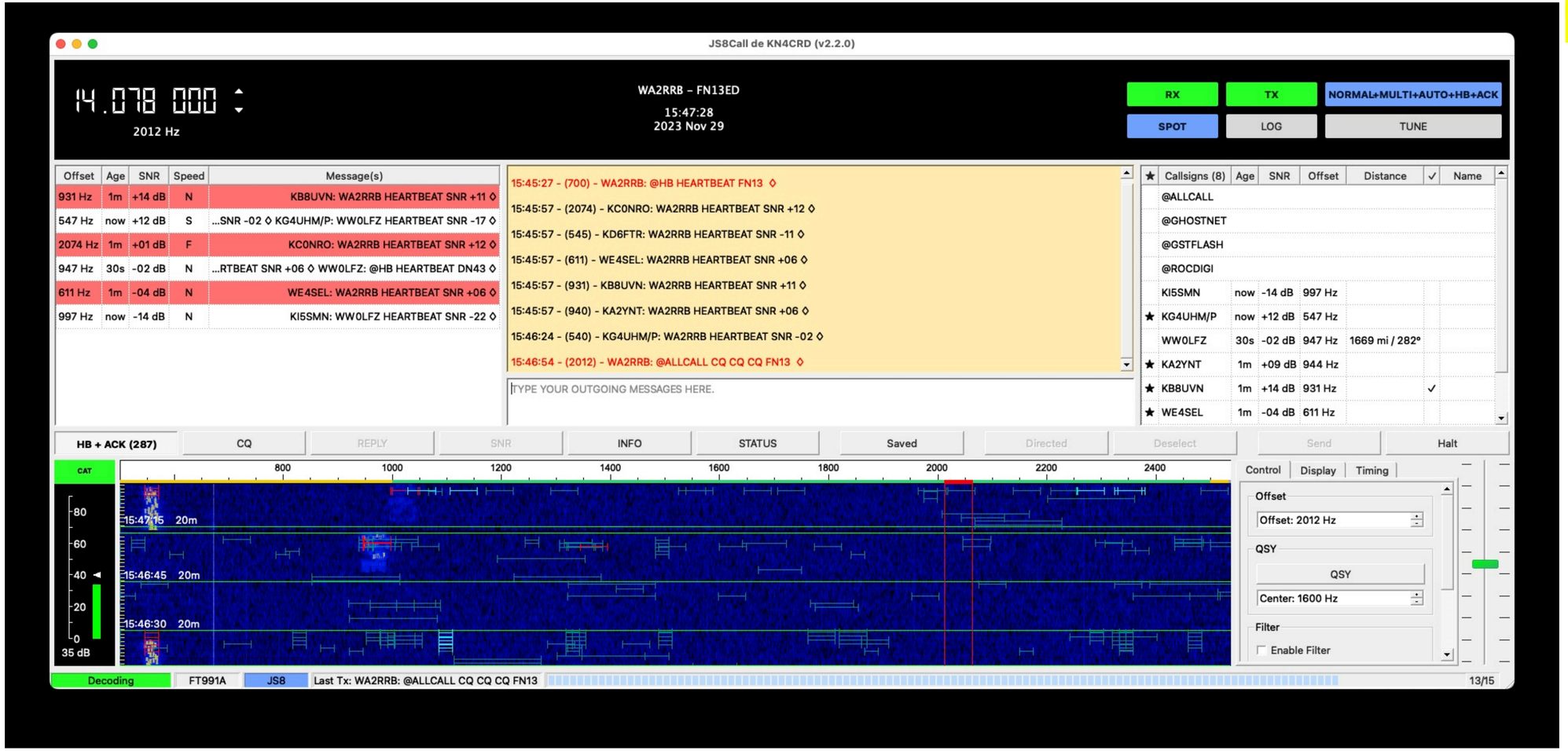
Operations Basics: Tips & Tricks



1. Set AGC & NoiseReduction to Zero/Off
2. Use abbreviations/prosigns - but not so as to hide the meaning of the msg
3. This protocol is 100% Duty Cycle - 25 watts is fine, no need to cook the finals
4. Use HB functionality “pings” to check propagation
5. Be patient when calling CQ as JS8 is not yet as popular as FT8
6. Keep messages to 3 or 4 cycles, to reduce transmission times

JS8Call

Operations Basics: Understanding the Display



Resources

(will be displayed again at end of presentation)



- this slideset:

  https://www.well.com/~jeffs/RARA/

- the JS8 manual:

  https://www.delta25.de/JS8-2021-11/JS8Call_Guide.pdf

- an older presentation:

  https://www.nm5hd.com/documents/PRESENTATIONS/20191213_JS8_Presentation_updated.pdf

- video of JS8 author presentation:

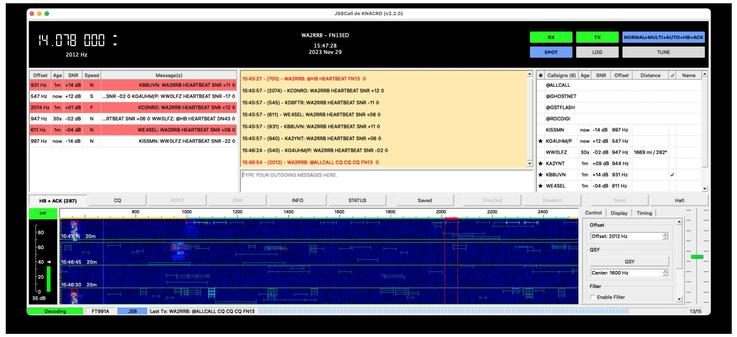
  https://www.youtube.com/watch?v=mZKhVcFOljY

Live Demo and Q&A Period

AMA (ask me anything) time



- Enough of me talking, let us shift into **demo mode** now
  - We will walk through the *flrig* & *JS8Call* **settings** which I use (your mileage may vary)
  - I will take **questions** and try to show you the **answers**, **live**



JS8 & JS8Call Practicalities

installation and basic use of JS8Call & supporting software packages

Jeffrey Sonstein - WA2RRB
(presentation for JAN 2024 Rochester Amateur Radio Association meeting)