The FT8 Revolution

Mike Hasselbeck
WB2FKO

SARA Hamfest
20 October 2018

NMVHF

FT8 is revolutionizing how amateur radio operators work DX

HF DX is possible even at the solar minimum

WHAT THIS PRESENTATION IS ABOUT:

How and why FT8 was designed

Why FT8 works so well for weak signal communications

WHAT THIS PRESENTATION IS *NOT* ABOUT:

How to setup a station for digital operation

How to operate effectively with FT8 (yes, there is a learning curve)

Related software: FT8 Call, JTDX, JTHF, JT Alert, etc

Picking a fight with ops who prefer traditional modes

Wide variety of digital modes in amateur radio

PSK31	RTTY	CLOVER
QPSK31	PACKET	MFSK16
AMTOR	THOR	OLIVIA
PACTOR	HELLSCHREIBER	WSJT-X

Choice strongly depends on what we are trying to do

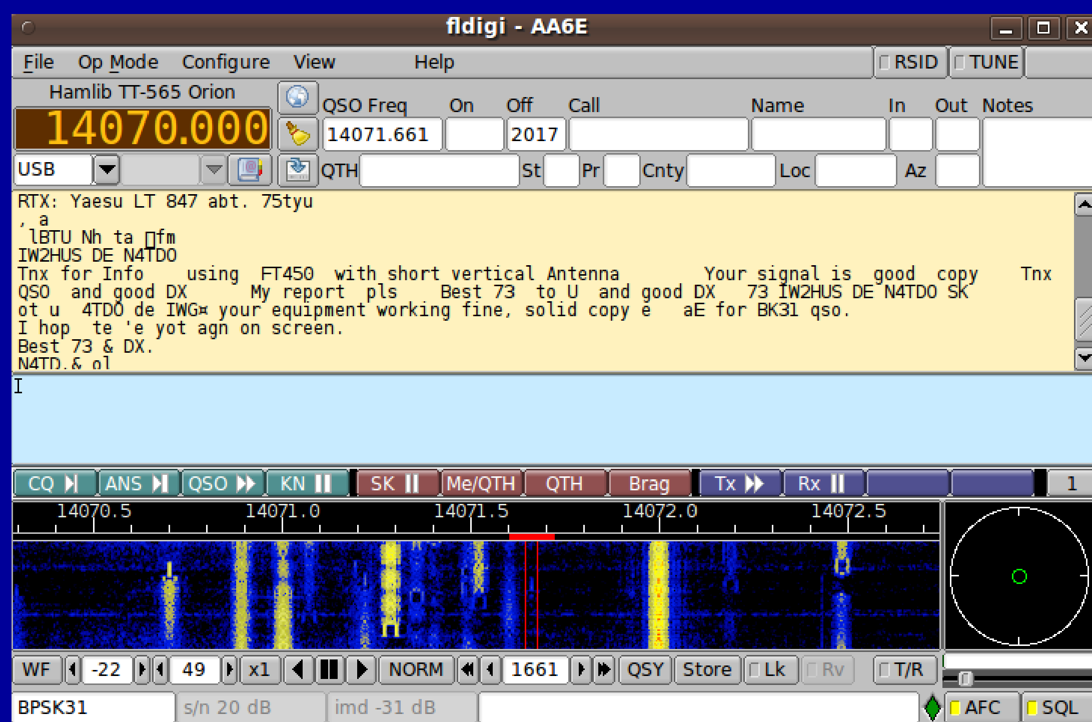
PSK31 is popular:
Keyboard-to-Keyboard Communication in Real-Time

Phase-shift keying at 31 baud (bits per second)

PC + sound card

Five bit code (baudot). Equivalent to ~ 60 wpm cw

Primarily on HF



Working DX: Want that new grid or country in the log!

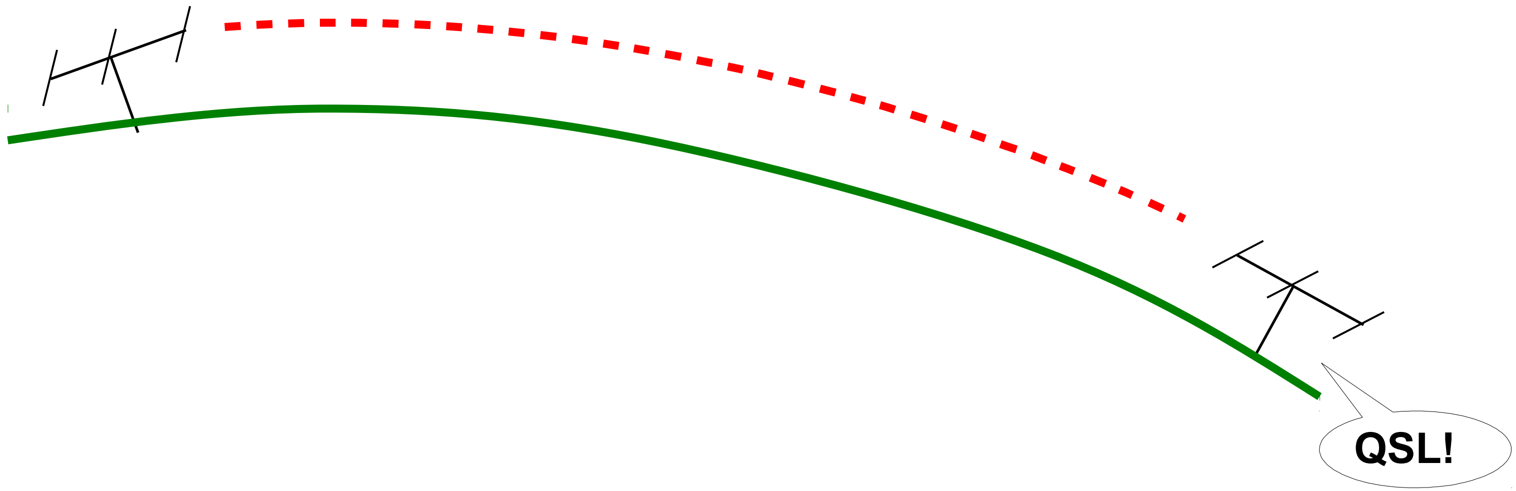
Usually don’t have the luxury or even desire to chat

Success if we just exchange callsigns and a report

WSJT-X: Digital protocol for minimum communication on marginal paths

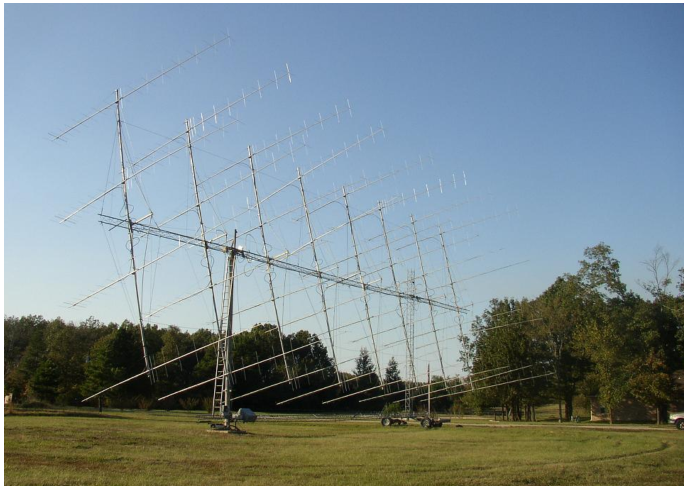
Fundamental Design Premise:

Exchange only enough information for a *minimum* QSO



FT8 is a sub-mode of WSJT-X Derived from JT65






FT8: Franke-Taylor Design, 8-tone FSK

Introduced for Alpha-testing 30 June 2017

Design motivated by 6m Es:

Short duration, weak but steady openings

15 second sequences ⇒ 4x faster QSOs than JT65

4–6 dB less sensitive than JT65

Steve Franke, K9AN

Joe Taylor, K1JT

Foundational work for JT65 & FT8:
Compact messages

QSO messages can be very efficiently coded

CALLSIGN1 CALLSIGN2 GRID

Tom Clark, K3IO

Phil Karn, KA9Q

Reference: Clark and Karn, Proc. CSVHF, 1996

JT65/FT8 messages are generally not free-form

Greatly reduces the amount of data required

Defines a 72/75 bit message protocol

JT65/FT8 messages are generally not free-form

Greatly reduces the amount of data required

Defines a 72/75 bit message protocol

KG5FHU WB2FKO DM65

This message can be coded into 71 digital bits

Compare: 209 bits in Morse Code (1 dit = 1 bit)

Why are compact messages useful?

1) **Make data packets very small**

– OR –

2) **Add other, very useful data to message**

FORWARD ERROR CORRECTION:

The crucial enhancement CW does not have

Modern technology:

- Modems
- Hard drives
- CDs
- DVDs
- Blue-Ray
- Digital TV
- Satellites
- Deep-space probes
- QR codes for phones

Amateur radio:

- D-Star
- DMR
- Fusion
- QPSK31
- MFSK16
- Olivia
- WSJT

What is Forward Error Correction?

EXAMPLE:

Simple 4 character alphabet without FEC

A:	00
B:	01
C:	10
D:	11

Each character represented by 2 data bits

A:	00
B:	01
C:	10
D:	11

Transmit an 8 character string:

C B D A B C A C

Requires 16 data bits:

10 01 11 00 01 10 00 10

A:	00
B:	01
C:	10
D:	11

Transmit an 8 character string:

C B D A B C A C

Requires 16 data bits:

10 01 11 00 01 10 00 10

Communication channels often have noise

A:	00
B:	01
C:	10
D:	11

Transmit an 8 character string:

C B D A B C A C

Requires 16 data bits:

10 01 11 00 01 10 00 10

Some bits may be incorrectly received

2 of 16 bits incorrect:

10 01 11 00 00 00 00 10

Decoded message:

C B D A A A A C

Correct reception requires
100% transmission throughput

C B D A B C A C

87.5% transmission throughput garbles
some of the message

C B D A A A A C

Can message be correctly received with < 100% throughput?

Solution 1: Send message multiple times

Three transmissions: 48 bits with 87.5% throughput
2 of 16 bits in each TX incorrect

TX1: **10 01 11 00 00 00 00 10**
Decoded message:
C B D A A A A C

TX2: **10 01 10 00 11 10 00 10**
Decoded message:
C B C A D C A C

TX3: **00 01 11 00 01 10 00 11**
Decoded message:
A B D A B C A D

Solution 1: Send message multiple times

Three transmissions: 48 bits with 87.5% throughput
2 of 16 bits in each TX incorrect

TX1: **10 01 11 00 00 00 00 00 10**

Decoded message:

C B D A A A A C

TX2: **10 01 10 00 11 10 00 10**

Decoded message:

C B C A D C A C

TX3: **00 01 11 00 01 10 00 11**

Decoded message:

A B D A B C A D

5th character ambiguity: **A, D, B ?**

Solution 2: Send bits multiple times

1 data bit, 2 redundancy bits: 48 bits total:

111 000 000 111 111 111 000 000 000 111 111 000 000 000 111 000

Interpretation of received data

000 = **0**; No error

111 = **1**; No error

011 = **1;** Corrected

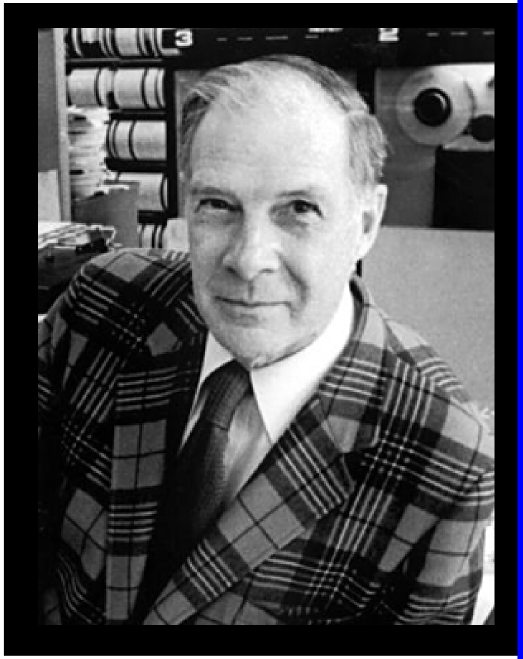
101 = **1;** Corrected

001 = **0**; Corrected

Triple modular redundancy

Solution 3: Hamming code words

Invented in 1950 at Bell Labs



Richard Hamming

Simple example: 2 data bits, 3 parity bits

A: 00010

B: 11001

C: 10110

D: 11101

Each character represented by **5 bits** instead of 2 bits

Transmit same 8 character string *only once*:

C B D A B C A C

Requires **40 data bits** instead of 48 (or 16):

10110 11001 11101 00010 11001 10110 00010 10110

Assume 87.5% throughput: 5 bits are incorrect

Transmitted signal:

10110 11001 11101 00010 11001 10110 00010 10110

Received signal:

10110 11000 01101 00010 11011 10110 00000 10111

What happens at the receiver?

10110 11000 01101 00010 11011 10110 00000 10111

C ? ? A ? C ? ?

Five characters are not recognized
Not in the codeword dictionary!

What happens at the receiver?

10110 11000 01101 00010 11011 10110 00000 10111

C ? ? A ? C ? ?

That's worse than with no FEC !

What happens at the receiver?

10110 11000 01101 00010 11011 10110 00000 10111

C ? ? A ? C ? ?

Not really worse. Now we know there are errors and could ask transmitter to re-send.

What happens at the receiver?

10110 11000 01101 00010 11011 10110 00000 10111

C ? ? A ? C ? ?

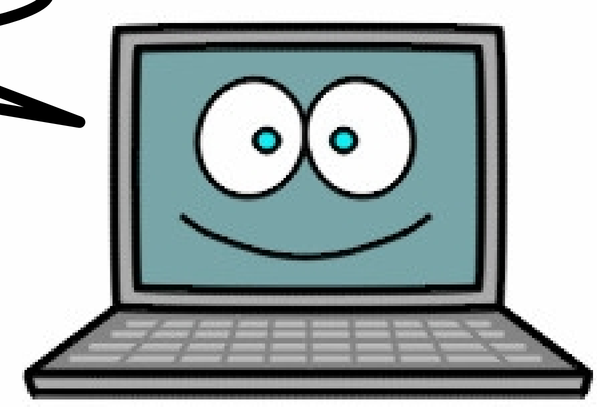
The 16-bit message gave only 2 false decodes *BUT* the receiver didn't know it!

What happens at the receiver?

10110 11000 01101 00010 11011 10110 00000 10111

C ? ? A ? C ? ?

Let the FEC algorithm try to figure out what isn't initially recognized

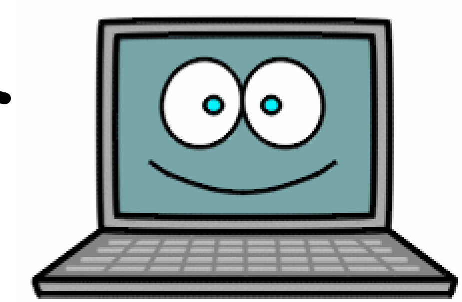


2nd received character: 11000

Compare it to our dictionary or code book

A:	00010	XX0X0	3 bits wrong
B:	11001	1100X	**1 bit wrong**
C:	10110	1XXX0	3 bits wrong
D:	11101	11X0X	2 bits wrong

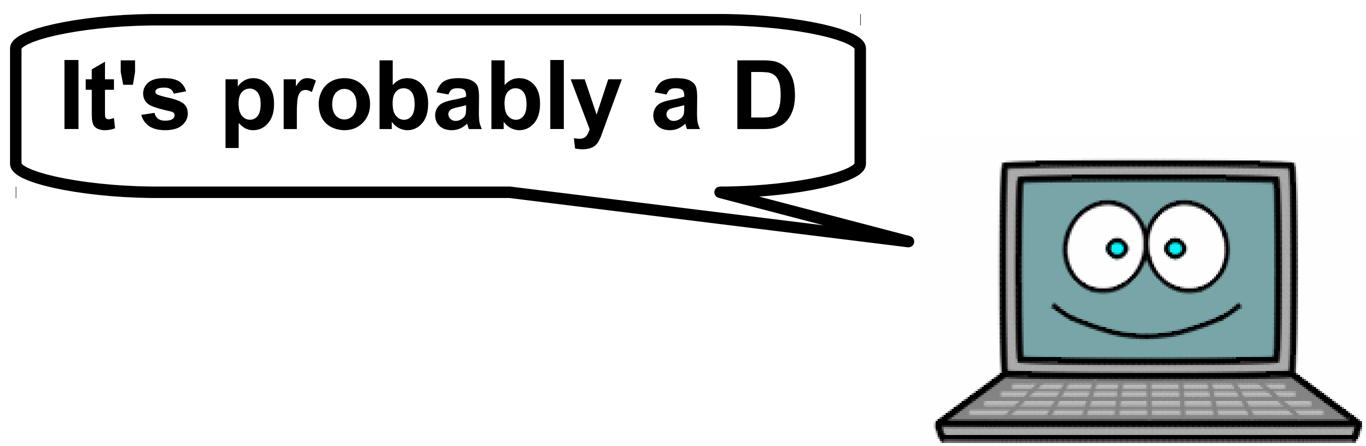
It's probably a B



3rd received character: 01101

Compare it to our dictionary

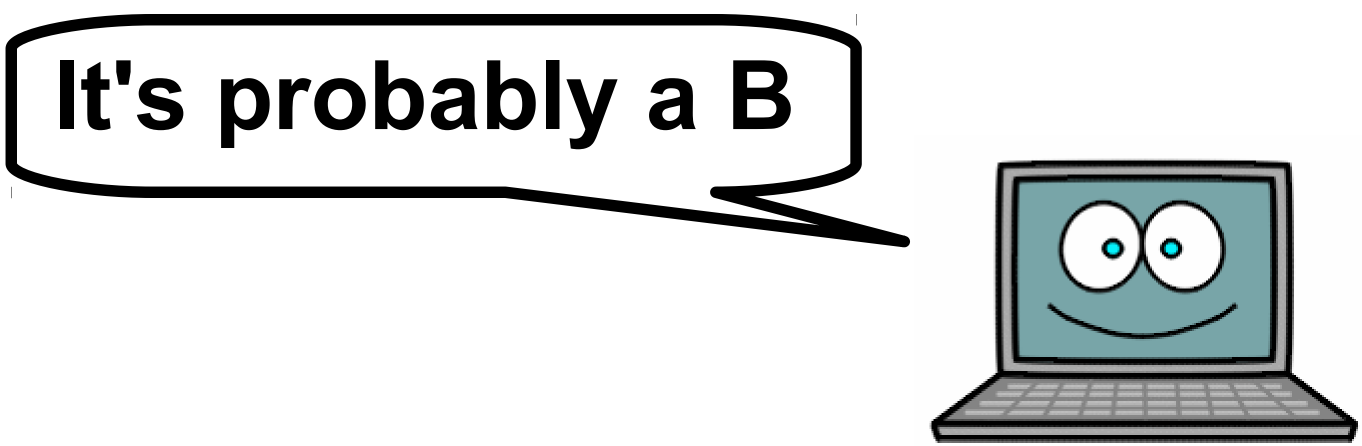
A:	00010	0XXXX	4 bits wrong
B:	11001	X1X01	2 bits wrong
C:	10110	XX1XX	4 bits wrong
D:	11101	X1101	**1 bit wrong**



5th received character: 11011

Compare it to our dictionary

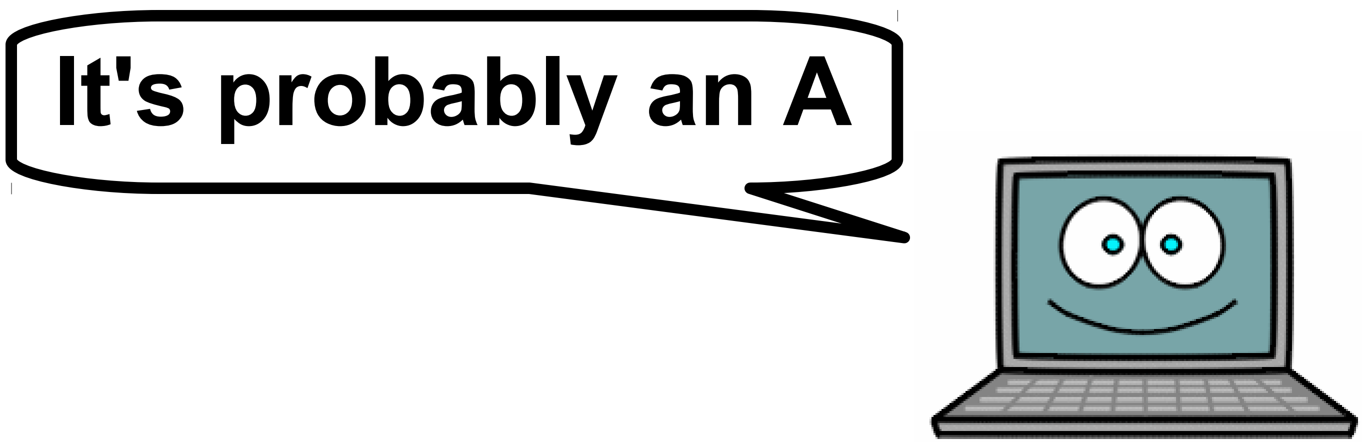
A:	00010	XX01X	3 bits wrong
B:	11001	110X1	**1 bit wrong**
C:	10110	1XX1X	3 bits wrong
D:	11101	11XX1	2 bits wrong



7th received character: 00000

Compare it to our dictionary

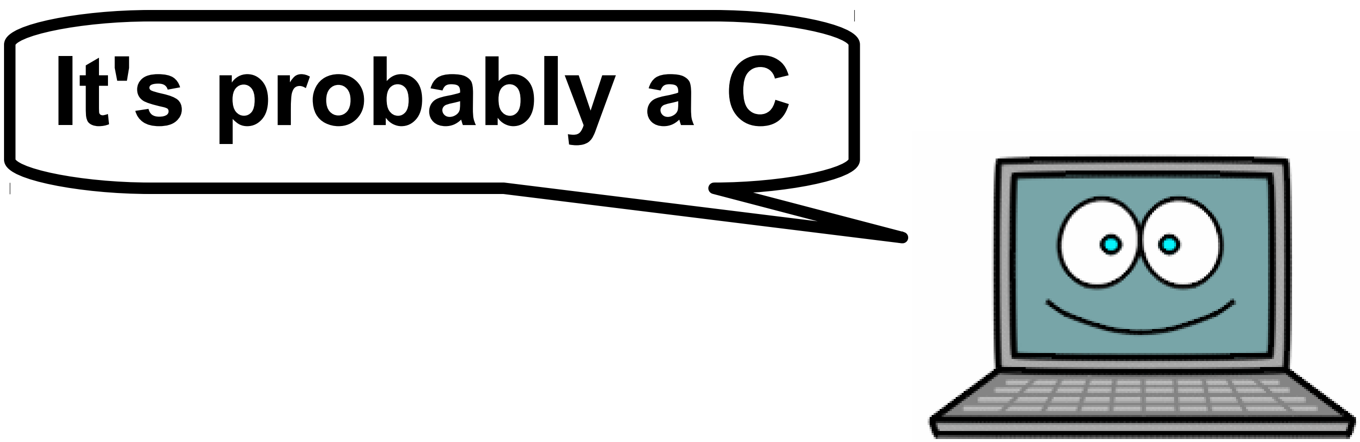
A:	00010	000X0	1 bit wrong
B:	11001	XX00X	3 bits wrong
C:	10110	X0XX0	3 bits wrong
D:	11101	XXX0X	4 bits wrong



8th received character: 10111

Compare it to our dictionary

A:	00010	X0X1X	3 bits wrong
B:	11001	1XXX1	3 bits wrong
C:	10110	1011X	**1 bit wrong**
D:	11101	1X1X1	2 bits wrong



Transmitted message:

10110 11001 11101 00010 11001 10110 00010 10110

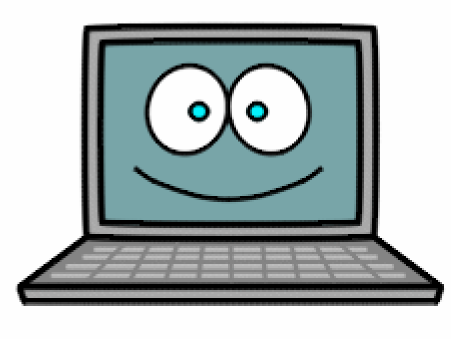
C B D A B C A C:

Received message, no FEC

C ? ? A ? C ? ?

After FEC decoding

C B D A B C A C



Perfect decode **even with 12.5% data loss**

This FEC scheme won't work if we lose more than 1 bit per character

EXAMPLE: First two characters won't decode

10000 00110 11101 00010 11001 10110 00010 10110

Solution 1: Ask transmitter to re-send

Solution 2: Add more parity bits*

* as determined by the Shannon Limit

Solution 3: Design decoder to arrange data to be read as columns instead of rows

10000 00110 11101 00010 11001 10110 00010 10110

Write/Transmit direction →

Read direction ↓

1	**0**	**1**	**0**	**1**	**1**	**0**	**1**
0	**0**	**1**	**0**	**1**	**0**	**0**	**0**
0	**1**	**1**	**0**	**0**	**1**	**0**	**1**
0	**1**	**0**	**1**	**0**	**1**	**1**	**1**
0	**0**	**1**	**0**	**1**	**0**	**0**	**0**

FORWARD ERROR CORRECTION in FT8

Each 75 bit message is augmented with
+ 87 Forward Error Correction bits (Low Density Parity Check)
+ 12 Cyclical Redundancy Check bits
= 174 bits total

50% of the message length is for FEC using LDPC

Dr Robert Gallagher
Inventor of the LDPC
in 1960

HOW TONES GENERATE A DIGITAL MESSAGE

Alphabet

A: 00
B: 01
C: 10
D: 11

8 FSK TONES

TONE 0: 000
TONE 1: 001
TONE 2: 010
TONE 3: 011
TONE 4: 100
TONE 5: 101
TONE 6: 110
TONE 7: 111

Example: 12-bit message

C B D A B C

1 0 0 1 1 1 0 0 0 1 1 0

TONE 4 TONE 7 TONE 0 TONE 6

CBDABC is sent as a
TONE SEQUENCE: 4,7,0,6

FT8 TONES GENERATE A DIGITAL MESSAGE

174 bits per FT8 message*

÷ 3 bits per TONE

= 58 TONES SENT per FT8 message

*174 bits = 75 data + 87 FEC + 12 CRC

TIME SYNCHRONIZATION

PSK31 and other digital modes can be sent and decoded randomly

WSJT modes gain additional sensitivity by requiring tight time-synch of the stations

Lock the 2 computers to a reference clock

TIME SYNCHRONIZATION

Computer Internet synch only gets in the ballpark

JT65 decoder requires an accuracy ≤ 0.03 seconds

FT8 decoder requires an accuracy ≤ 0.02 seconds

The message supplies its own synch signal

TIME SYNCHRONIZATION IN FT8

An FT8 message has 79 time intervals

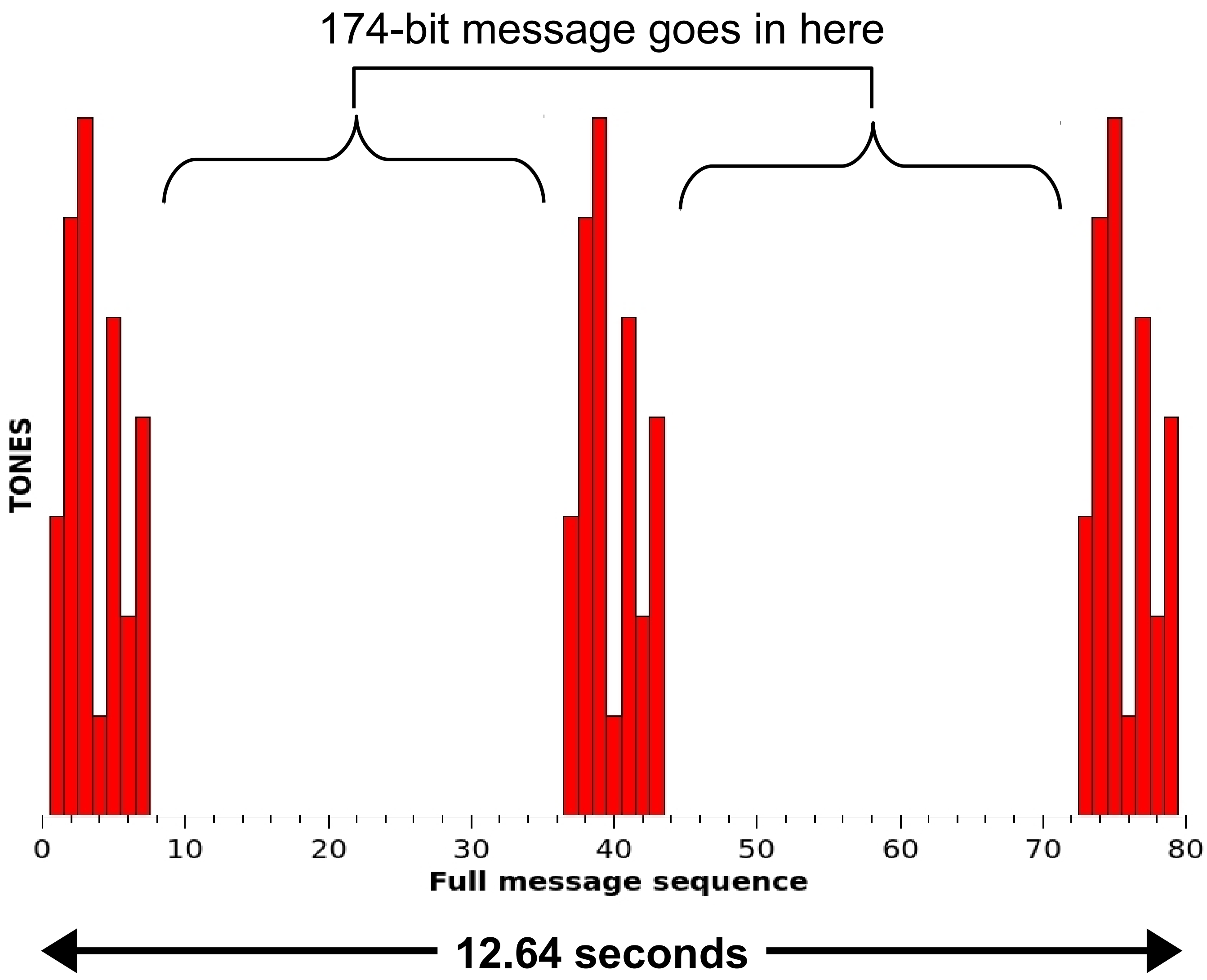
Each interval is 0.16 seconds

Total message duration: 12.64 seconds

58 intervals allotted for the message + FEC + CRC

21 intervals allotted for SYNCH TONES

7 tone sequence* at start, middle, and end of transmission



* 7x7 Costas Array: 7 frequencies x 7 time steps

WHAT THE HECK IS A *COSTAS ARRAY* ???

AND WHY SHOULD I CARE ABOUT IT?

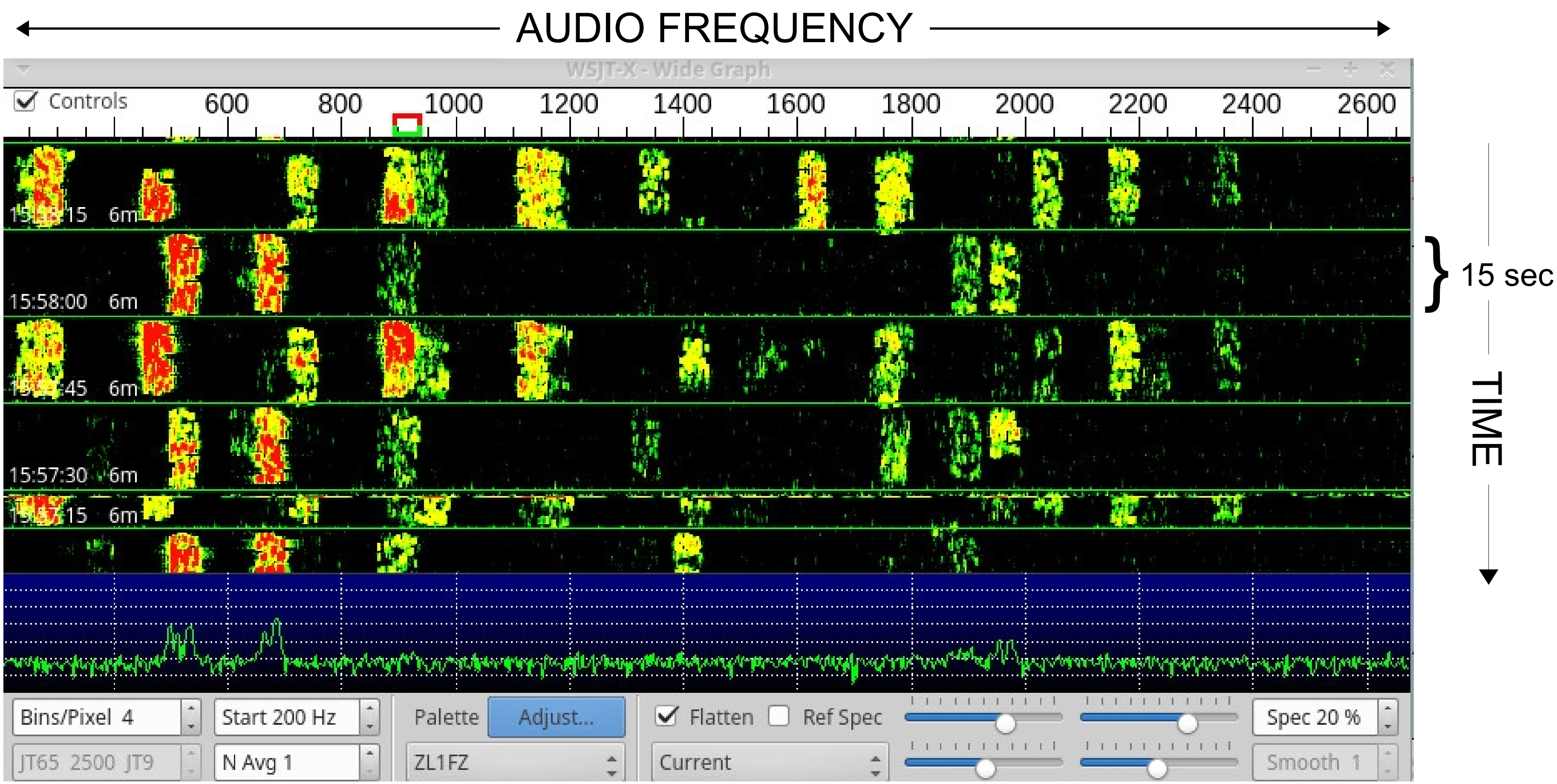
Dr John P Costas

A talk is in preparation for:

NEW MEXICO TECHFEST

February 2019, Albuquerque

FT8 WATERFALL DISPLAY



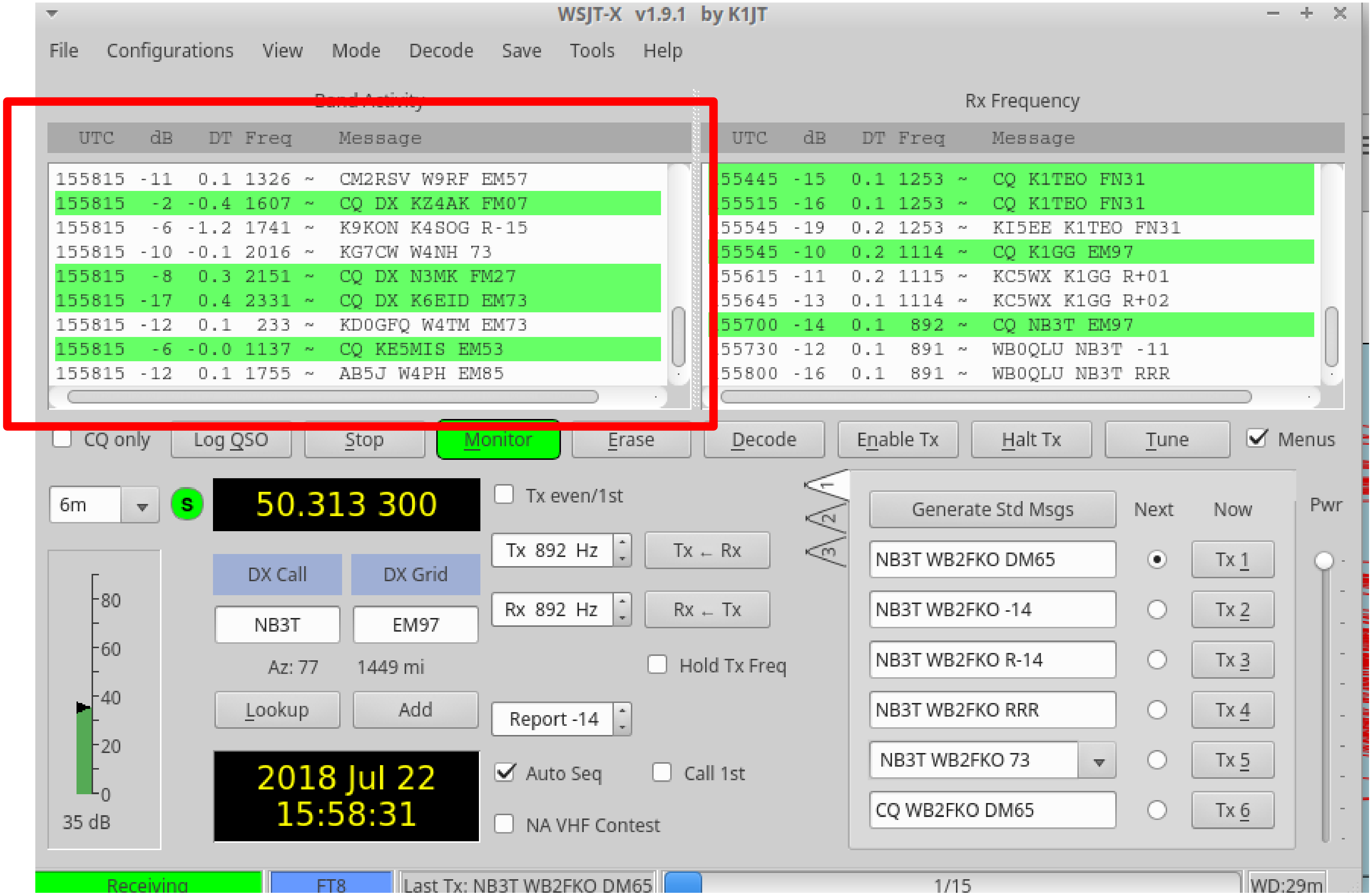
World Wide VHF Contest July 2018

VFO: 50.313 MHz

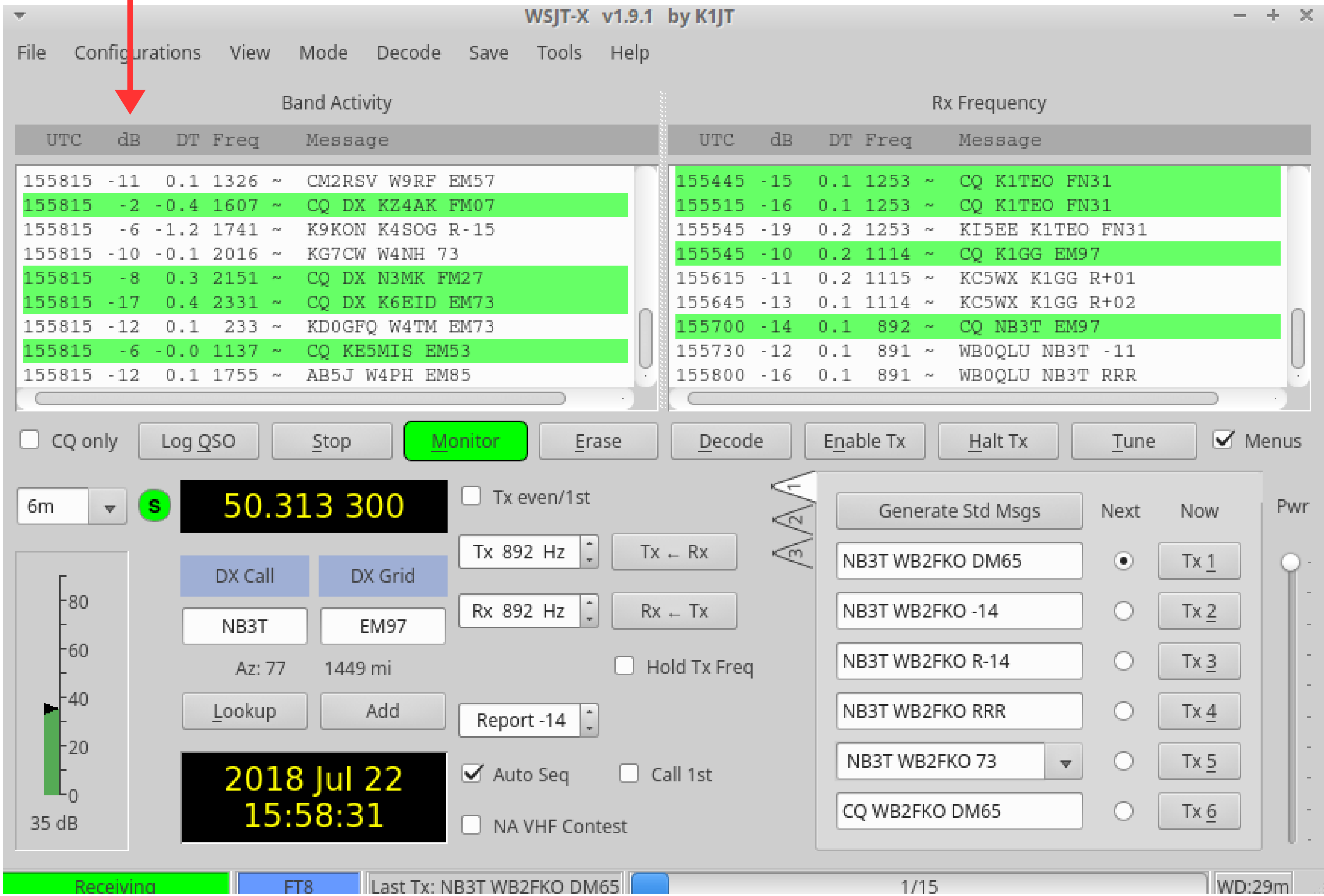
Upper sideband

50 Hz signals in ~2200 Hz receiver bandwidth

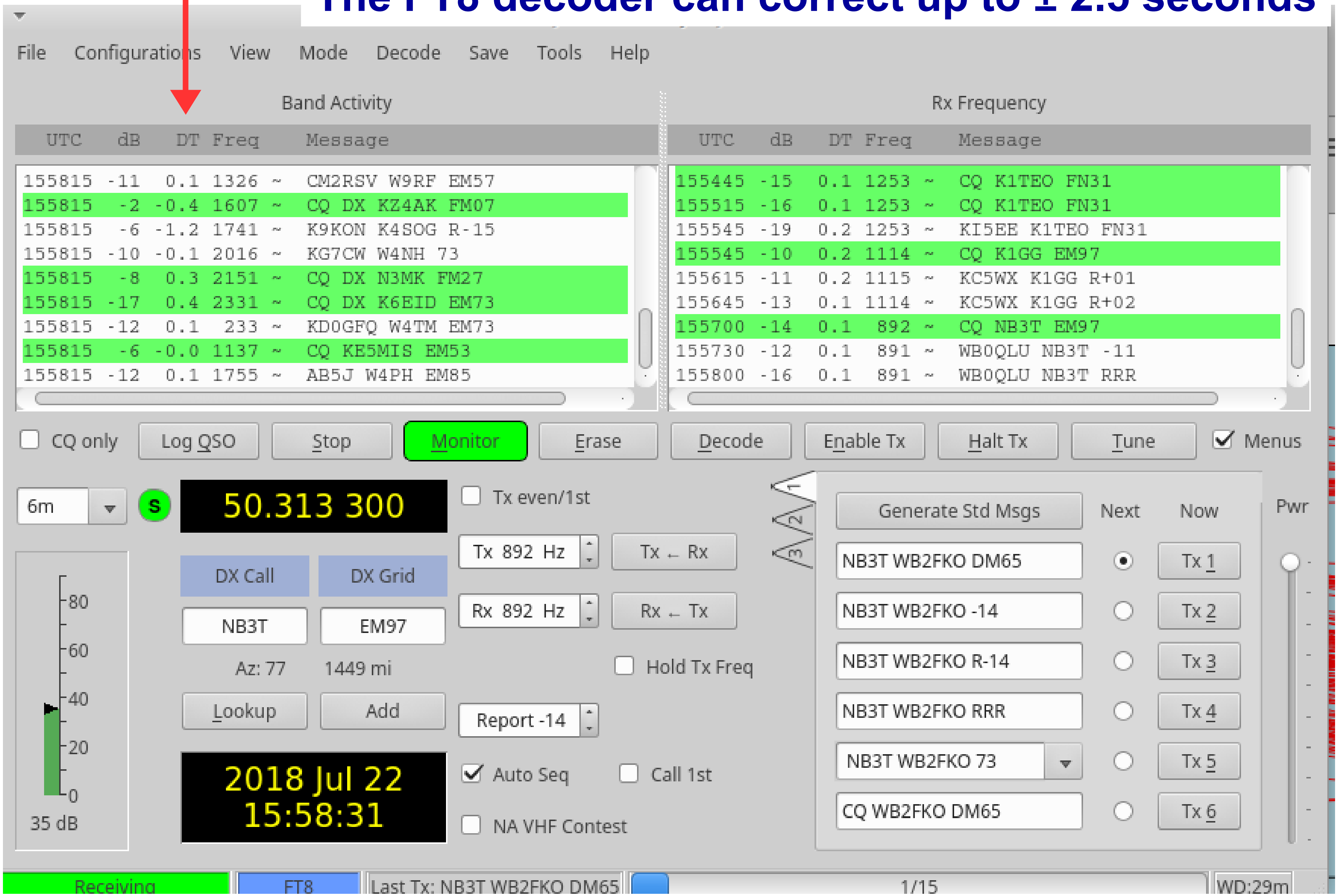
Simultaneous decodes of multiple signals in waterfall display



Relative strength of decoded signals

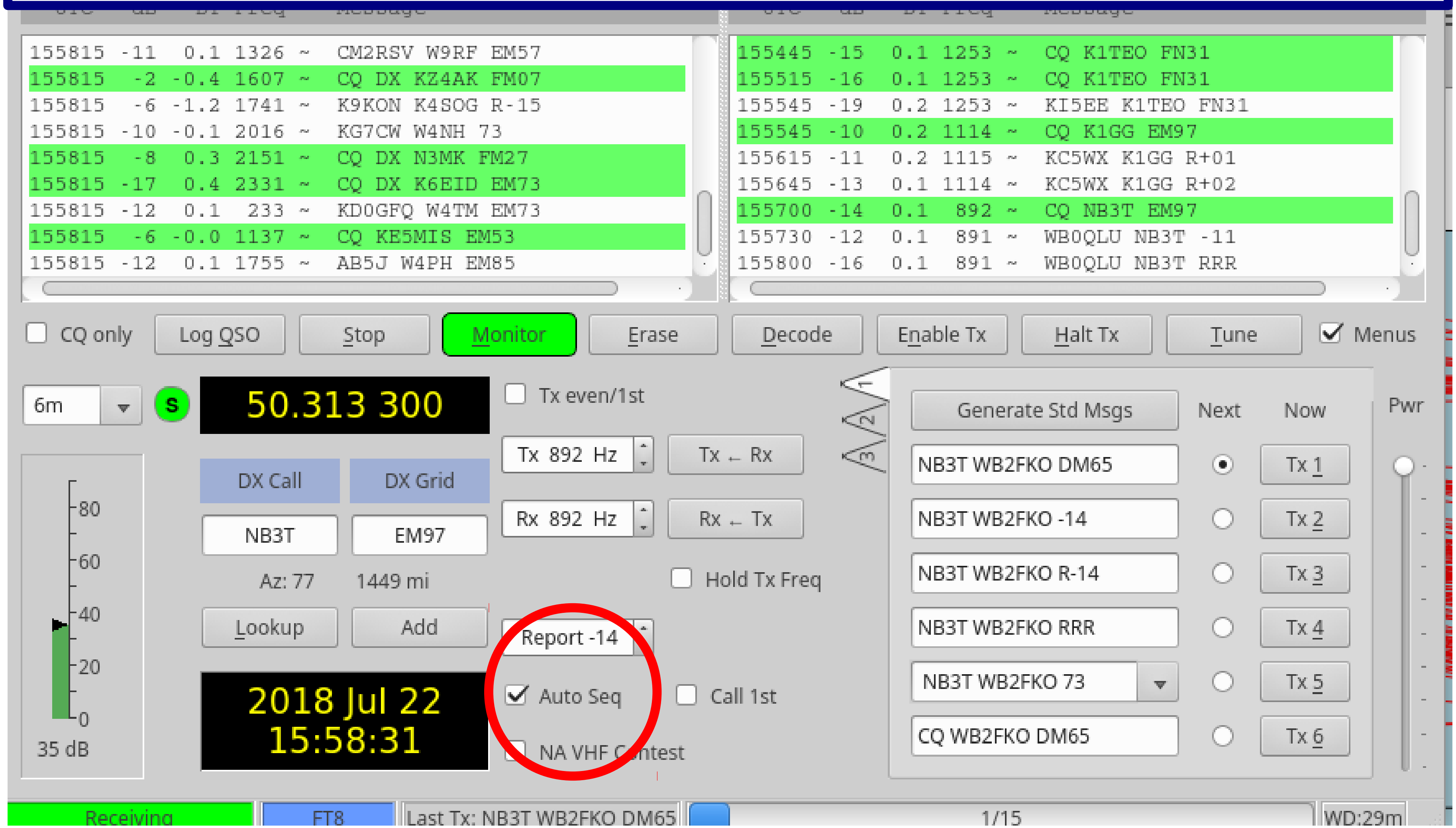


Time offset between stations
The FT8 decoder can correct up to ± 2.5 seconds



Audio baseline frequency
Each signal requires 50 Hz bandwidth

Time between FT8 transmissions is ~ 2 seconds
Faster than most ops can react!
Most use the Auto Sequence feature to send next message



No need to constantly fiddle with VFO

30+ different stations can be simultaneously decoded in the receiver bandwidth

Decodes possible even when signals partially overlap

FT8 Frequencies

1.840	18.100
3.573	21.074
7.074	24.915
10.136	28.074
14.074	50.313

Hardware cost to get on the digital modes

$ 

< $ 

[I assume everyone has a computer]

Why FT8 is better than CW for working DX

REASON 1:

FT8 is ~ 8 dB more sensitive than CW at 12 wpm

Communication possible with signals that are inaudible

REASON 2:

False character decodes extremely unlikely:
FT8 is All-or-Nothing

CW ops adept at filling-in missing characters

REASON 3:

Multiple streams of real-time decoded signals

Most CW ops can only reliably deal with one signal

REASON 4:

FT8 learning curve not as steep as CW

Many ops have poor or no CW skills

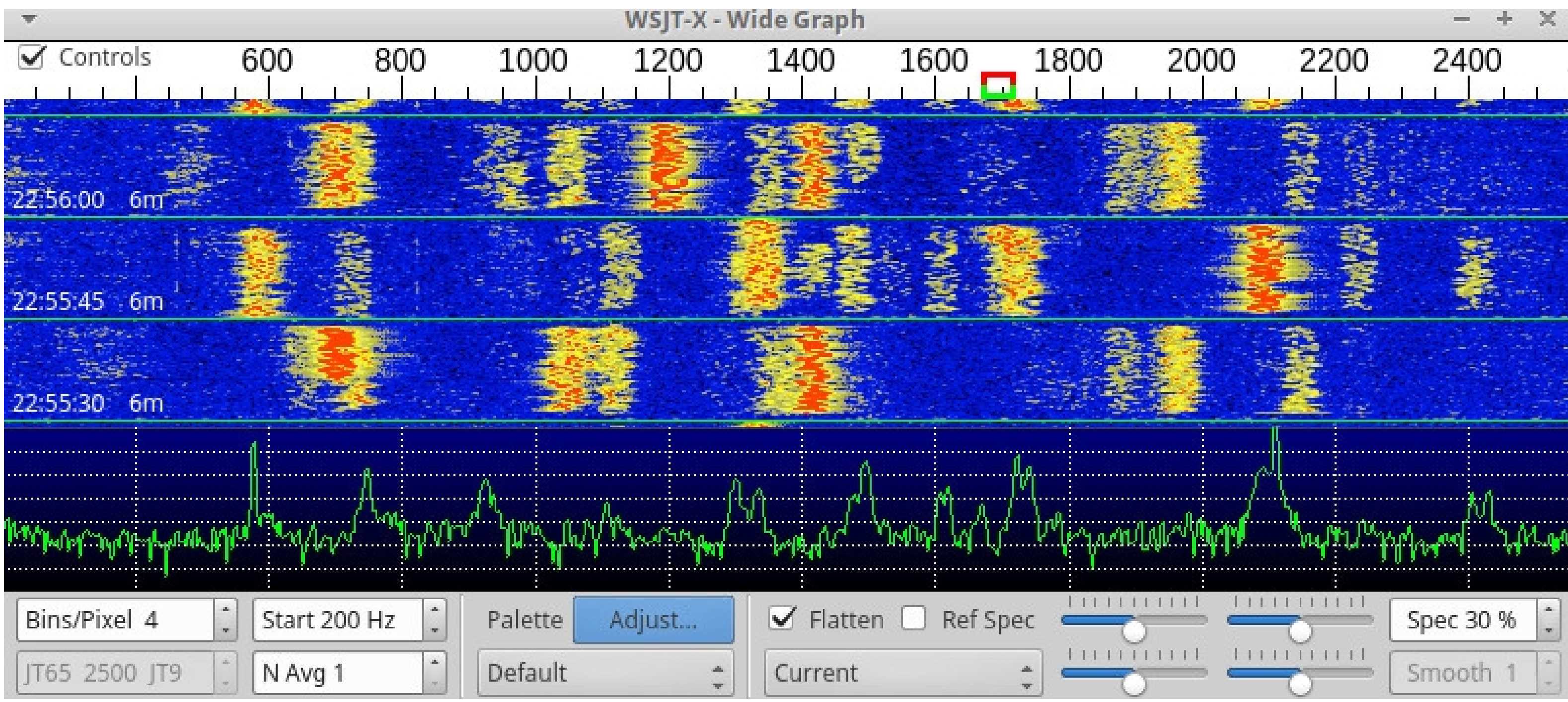
Chasing DX available to the hearing impaired

REASON 5:

QRM and congestion dramatically reduced

Many stations comfortably spread out over ~2500 Hz

A station picks out callers and works at discretion



REASON 5 (continued):

Two good CW ops can complete a QSO far faster than the minimum time for FT8: 45 seconds

...but they first have to find each other

...and they likely have to fight through QRM

Is FT8 more time-efficient in the aggregate?

REASON 5 (continued):

FT8 DXpedition Mode: Fox and Hounds

Many stations (Hounds) calling single, rare DX (Fox)

Pre-arranged frequency

Fox can work multiple hounds simultaneously

Run rates approaching 1000/hour on multiple bands

REASON 5 (continued):

AA7A and other experienced DXers provided guidance for Fox-Hound development

1) FT8 likely to displace RTTY in HF digital contests

2) FT8 will be preferred mode on future DXpeditions

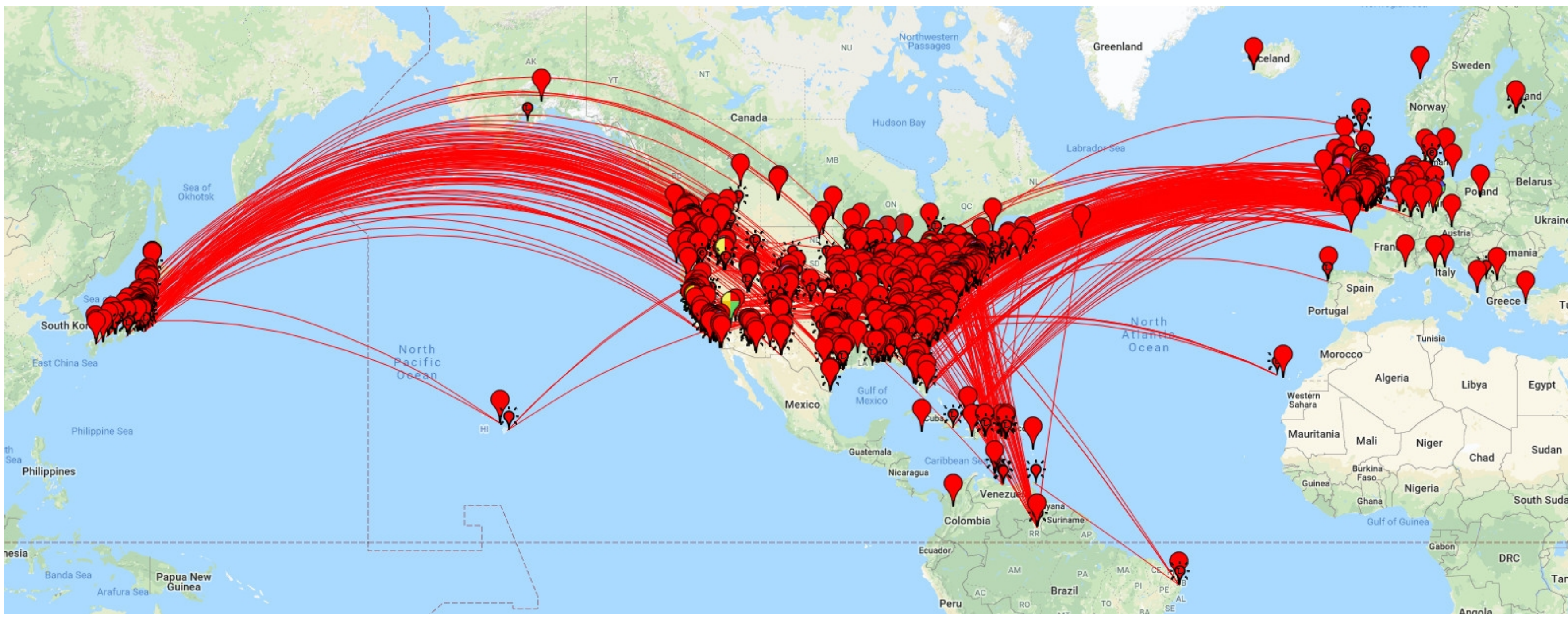
Ned Stearns, AA7A

REASON 6:

There is widespread adoption and momentum

The empirical evidence is mounting: FT8 makes working DX easier!

WSJT-X can automatically upload FT8 spots to pskreporter.info



15 minute window snapshot of 6-meters. 8 July 2018

Major update of WSJT-X was released on 17 October 2018

Data payload size increase from 75 to 77 bits in FT8 and MSK144

Accommodates VHF contest rover suffix: /R

Auto-sequencing between contest and non-contest stations

Field Day contest exchanges (example: 2B NM)

ARRL RTTY Roundup contest exchanges (599)

75 and 77 bit format are not compatible

Alerts for new DXCC, new grid, etc

~ 1 dB more sensitivity

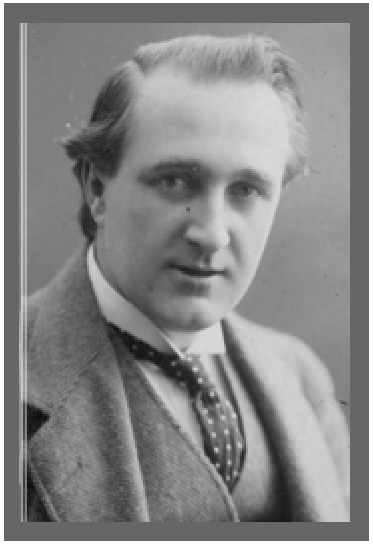
“If a computer decodes it, it's not real ham radio!”

“If a computer decodes it, it's not real ham radio!”

To each his own.

Considerable skill required to use FT8 effectively.

Get better with practice.



“It's difficult to make predictions, especially about the future.”

– Robert S. Petersen

FT8 or something like it is here to stay

DXers and Contesters will do one of the following:

1) **Stick with traditional modes**

2) **Move entirely to digital**

3) **Use a combination of analog and digital to best advantage**