# 4 A Protocol for Leibowitz; or, Booklegging by HF in the Age of Safe Æther

*by Travis Goodspeed and Muur P.*

Howdy y'all!

Today we'll discuss overloading of protocols for digital radio. These tricks can be used to hide data, exfiltrate it, watermark it, and so on. The nifty thing about these tricks is that they show how modulation and encoding of digital radio work, and how receivers for it are built, from really simple protocols like the amateur radio PSK31 and RTTY to complex ones like 802.11, 802.15.4, Bluetooth, etc.

We'll start with narrow-band protocols that you can play with at audio frequencies. So if you don't have an amateur license and a shortwave transceiver, you can use your sound card to do most of the work and run an audio cable between two laptops to send and receive it.[10]

– – – –   – – –   – – – –

Suppose that sometime in the future, our neighbor Alice lives in an America of modern–day Nehemiah Scudder,[11] whose Youtube preachers and Twitter lynch mobs have made the Internet into a Safe Zone for America's Youth, by disconnecting it from anything unsafe. So Alice's only option to get something unsafe to read is from Booklegger Bob in Canada, by shortwave radio.

But it ain't so easy. President Scudder has directed Eve at the Fair Communications Commission[12] to strictly monitor and brutally enforce radio regulations, defending the principles of Shortwave Neutrality and protecting the youth from microunsafeties.

So Alice and Bob need to make a shortwave radio polyglot, valid in more than one format. Intent on her mission, Eve is listening. So when Alice and Bob's transmissions are sniffed by Scudder's National Safety Agency or overheard by the general public, they must appear to be a popular approved plaintext protocol. It must appear the same on a spectrum waterfall, must decode to a valid message (`CQ CQ CQ de A1ICE A1ICE Pse k`), and nothing may draw undue attention to their communications. Bob, however, is able to find a secret, second meaning.

In this article, we'll introduce you to some of the steganographic tricks they could use, as well as some less stealthy—and more neighborly—ways to combine protocols. We'll start with PSK31 and RTTY, with a bit of CW for good measure. And just to show off, we'll also bring wired Ethernet into the mix, for an exfiltration trick worthy of being shared around campfires![13]

## 4.1 All You Need Is Sines

Well, not really. But it sure looks that way when you read about radio: sines are everywhere, and you build your signal out of them, using variations in their amplitude, frequency, phase to transmit information.[14] This stands to physical reason, since the sine wave is the basic kind of electromagnetic oscillation we can send through space. Of course, you can add them by putting them on the same wire, and multiply them by applying one signal to the base of a transistor through which the other one travels; you can also feed them through filters that suppress all but an interval of frequencies.

You can see these sines in the signal you receive on the waterfall display of Baudline or FLDigi, which show the incoming signal in the frequency domain by way of the Fourier transform. PSK31 transmissions, for example, will look like nice narrow bands on the waterfall view, which is the point of its design.

The waterfall view is close to how a mathematician would think about signals: all input whatsoever is a bunch of sine waves from all across the spectrum, even noise and all. A perfectly clean sine wave such as a carrier would make a single bright pixel

[10] You could also use loud speakers, but please don't. Pastor Laphroaig reminds us that there is a special level of hell for such people, who will spend Eternity next to those who scratch fingernails on chalk boards.

[11] `unzip pocorgtfo08.pdf ifthisgoeson.txt`

[12] Which some haters call Fundamentalist instead of Fair, but that's unsafe speech. Unsafe speech has consequences, neighbors. You don't want to find out about the consequences, so stay safe!

[13] Campfires are definitely not safe, so enjoy them while they last!

[14] Some combinations are useful, such as amplitude and phase, used, e.g., in DOCSIS; others aren't so useful, such as phase and frequency, because changes in one can't always be told from changes in the other.

in every line, a single bright 1-pixel stripe scrolling down. That line would expand to a multi-pixel band for a signal that is the carrier being modulated by changing its amplitude, frequency, or phase in any way, with the width of the band being the double of the highest frequency at which the changes are applied.[15]

Of course, the actual construction of digital radio receivers has very little to do with this mathematician's view of the signal. While a mix of ideal sines would neatly fall apart in a perfect Fourier transform, the real transform of sampled signal would have to be discrete, and would present all the interesting problems of aliasing, edge effects, leakage, scalloping, and so on. Thus the actual receiving circuits are specialized for their intended protocols particular kinds of modulation, designed to extract the intended signal's representation and ignore the rest—and therein lies Alice's and Bob's opportunity.

## 4.2 Related Work

In 2014, Paul Drapeau (KA1OVM) and Brent Dukes released `jt65stego`, a patched version of the JT65 mode that hides data in the error correcting bits.[16,17] The original JT65 by Joe Taylor (K1JT) features frames of 72 bits augmented by 306 error-correcting bits,[18] so Drapeau and Dukes were able to hide encrypted messages by flipping bits that normal radios will flip back. This reduces the odds of successfully decoding the cover message, but they do correct for some errors of the ciphertext.

Our concern in this article is not really stego, though that will be covered. Instead, we'll be looking at which protocols can be combined, embedded, emulated, and smuggled through other protocols. We'll play around with all sorts of crazy combinations, not because these combinations themselves are a secure means of communication, but because we'll be better at designing new means of communication for having thought about them.

## 4.3 Classic PSK31

PSK31 is best described in an article by Peter Martinez, G3PLX.[19] Here, we'll present a slightly simplified version, ignoring the QPSK extension and parts of the symbol set, so be sure to have a copy of Peter's article when implementing any of these techniques yourself.

This is a Binary Phase Shift Keyed protocol, with 31.25 symbols sent each second. It consumes just a bit more than 60 Hz, allowing for many PSK31 conversations to fit in the bandwidth of a single voice channel.

The PSK31 signal is commonly generated as audio then sent with Upper SideBand (USB) modulation, in which the audio frequency (1 kHz) is upshifted by an RF frequency (28.12 MHz) for transmission. For reception, the same thing happens in reverse, with a USB shortwave receiver downshifting the radio frequencies to the audio range. In older radios, this is performed by an audio cable. More modern radios, such as the Kenwood TS-590, implement a USB Audio Class device that can be run digitally to a nearby computer.

Because many different PSK31 transmissions can fit within the bandwidth of a single voice channel, modern PSK31 decoders such as FLDigi are capable of decoding multiple conversations at once, allowing an operator to monitor them in parallel. These parallel decodings are then contributed to aggregation websites such as PSKReporter that collect and map observations from many different receivers.

### 4.3.1 Varicode

Instead of ASCII, PSK31 uses a variable-length character encoding scheme called Varicode. This

[15] This is easy to see for frequency and phase, since these changes are added to the argument of the sine $A \cdot sin(\omega \cdot t + \theta)$, the frequency $\omega$ and the phase $\theta$. Seeing this for the amplitude $A$ is a bit trickier, but imagine $A$ to be another sine wave, modulating the carrier. Then we deal with the product of two sines, and this is, by the age-old trigonometric identities $sin(\alpha+\beta) = sin(\alpha)cos(\beta) + cos(\alpha)sin(\beta)$ and $sin(\alpha-\beta) = sin(\alpha)cos(\beta) - cos(\alpha)sin(\beta)$; hence adding these and remembering that the cosine is the sine shifted by $\pi/2$, $sin(\alpha)sin(\beta+\pi/2) = \frac{1}{2}(sin(\alpha+\beta) + sin(\alpha-\beta))$. That is, a product of sines is the arithmetic average of the sines of the sum and the difference of their arguments. If $\alpha$ is the carrier and $\beta$ is the change, the rainfall diagram will show the band from $\alpha - \beta$ to $\alpha + \beta$, that is $2\beta$-wide.

Seeing this sum and knowing the carrier frequency, one might wonder: can't we make do with just one term of the sum $\alpha+\beta$, and ignore $\alpha-\beta$? Indeed, if one applies a filter to cut the frequencies less than the carrier from the transmitted signal, one can save half the bandwidth and still recover the signal $\beta$. This trick is known as the Upper Side Band, and it used for the actual digital radio transmissions.

[16] `https://github.com/pdogg/jt65stego`

[17] Steganography in Commonly Used HF Protocols, Drapeau and Dukes, Defcon 22

[18] `unzip pocorgtfo08.pdf jt65.pdf`

[19] `unzip pocorgtfo08.pdf psk31.pdf`

character set features many of the familiar ASCII characters, but they are rearranged so that the most common characters require the fewest bits. For example, the letter e is encoded as 11, using two bits instead of the eight (or seven) that it would consume in ASCII. Lowercase letters are generally shorter than upper case letters, with uncommon control characters taking the most bits.

A partial Varicode alphabet is shown in Figure 2. Additionally, an idle of at least two 0 bits is required between Varicode characters. No character begins or ends with a 0, and for clock recovery reasons, there will never be a string of more than six 1 bits in a row.

#### 4.3.2 Encoding

To encode a message, letters are converted to bits through the Varicode table, delimited by 00 to keep them distinct. As PSK31 is designed for live use by a human operator in real time, any number of zeroes may be appended. That is, "e e" can be rendered to 110010011, 110000010011, or 1100100011; there is no difference in meaning, only transmission time.

PSK31 encodes the bit 1 as a continuous carrier and the bit 0 as a carrier phase reversal. So the sequence 11111111 is a boring old carrier wave, no different from holding a Morse key for a quarter-second, while 00000000 is a carrier that inverts its phase every 31.25 ms.

So what's a phase reversal? It just means that what used be the peak of the wave is now a trough, and what used to be the trough is now a peak.

#### 4.3.3 Decoding

As described in Martinez' PSK31 article, a receiver first uses a narrow bandpass filter to select just one PSK31 signal.

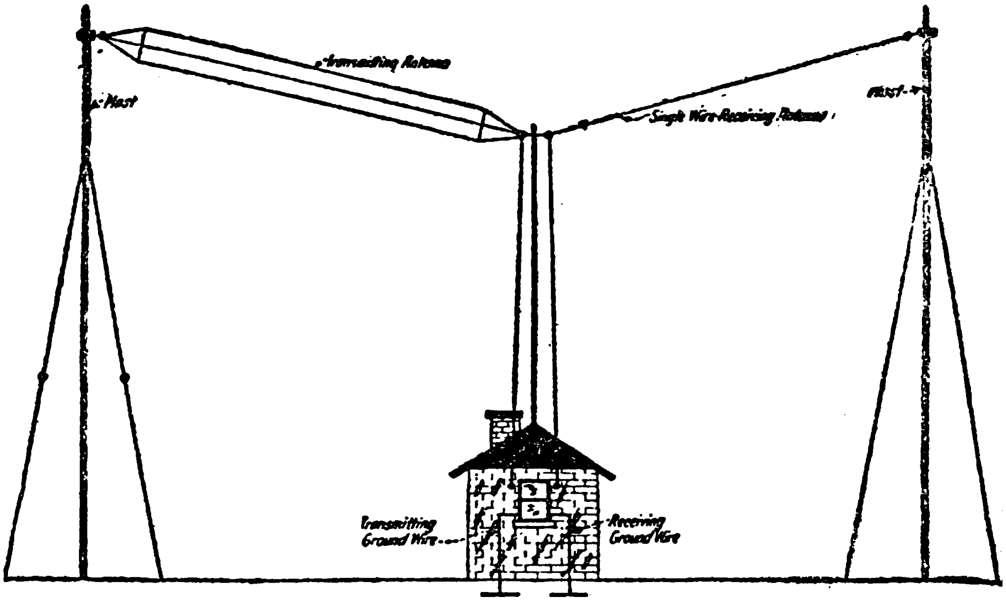
It then multiplies that signal with a time-delayed version of itself to extract the bits. The output will be negative when the signal reverses polarity, and positive when it does not.

Once the bits are in hand, the receiver splits them into Varicode characters. A character begins as the first 1 after at least two zeroes, and a character ends as the last 1 before two or more zeroes. After the characters are split apart, they are parsed by a lookup table to produce ASCII.

### 4.4 PSK31 Stego

#### 4.4.1 Extending the Varicode Character Set

G3PLX's original article contains a second part, in which he notes that his original protocol provides no support for extended characters, such as the British symbol for pounds sterling, £. Wishing to add such characters, but not to break compatibility, he noted that the longest legal Varicode character was ten



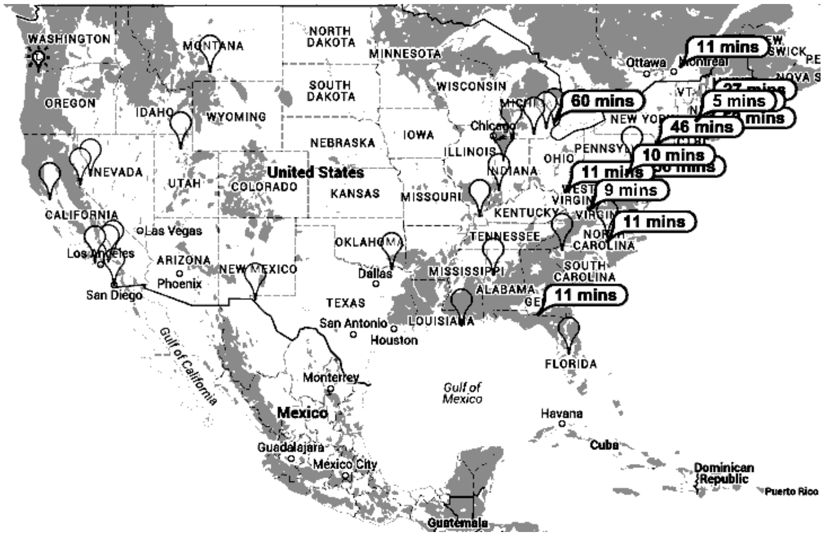


Figure 1: PSKReporter, a Service for Monitoring PSK31

bits long. Anything longer was ignored by the receiver as a damaged and unrecoverable character, so PSK31 uses those long sequences for extended characters.

Reviewing the source code of a few PSK31 decoders, we find that Varicode still has not defined anything with more than twelve bits. By prefixing the character Alice truly intends to send with a pattern such as `101101011011`, she can hide special characters within her message. To decode the hidden message, Bob will simply cut that sequence from any abnormally long character.

### 4.4.2 Hiding in Idle Lengths

PSK31 requires *at least* two `0` bits between characters, but it doesn't specify an exact limit. It's not terribly uncommon to see forgotten transmitters spewing limitless streams of zeroes into the ether as their operators sit idle, never typing a character that would result in a zero. Alice can abuse this to hide extra information by encoding data in the variable gap between characters.

For an example, Alice might place the minimal pair of zero bits (`00`) between characters to indicate a zero while a triplet (`000`) indicates a one.

### 4.4.3 Extending the Symbol Set

In its classic incarnation, PSK31 uses Binary Phase Shift Keying (BPSK), which means that the phase flips 180 degrees. This is sometimes called BPSK31, to distinguish it from a later variant, QPSK31, which uses Quadrature Phase Shift Keying (QPSK).

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| 11101 | LF | 1011 | a | 1111101 | A |
| 11111 | CR | 1011111 | b | 11101011 | B |
| 1 | SP | 101111 | c | 10101101 | C |
| 10110111 | 0 | 101101 | d | 10110101 | D |
| 10111101 | 1 | 11 | e | 1110111 | E |
| 11101101 | 2 | 111101 | f | 11011011 | F |
| 11111111 | 3 | 1011011 | g | 11111101 | G |
| 101110111 | 4 | 101011 | h | 101010101 | H |
| 101011011 | 5 | 1101 | i | 1111111 | I |
| 101101011 | 6 | 111101011 | j | 111111101 | J |
| 110101101 | 7 | 10111111 | k | 101111101 | K |
| 110101011 | 8 | 11011 | l | 11010111 | L |
| 110110111 | 9 | 111011 | m | 10111011 | M |
|  |  | 1111 | n | 11011101 | N |
|  |  | 111 | o | 10101011 | O |
|  |  | 111111 | p | 11010101 | P |
|  |  | 110111111 | q | 111011101 | Q |
|  |  | 10101 | r | 10101111 | R |
|  |  | 10111 | s | 1101111 | S |
|  |  | 101 | t | 1101101 | T |
|  |  | 110111 | u | 101010111 | U |
|  |  | 1111011 | v | 110110101 | V |
|  |  | 1101011 | w | 101011101 | W |
|  |  | 11011111 | x | 101110101 | X |
|  |  | 1011101 | y | 101111011 | Y |
|  |  | 111010101 | z | 1010101101 | Z |

Figure 2: Partial PSK31 Varicode Alphabet

QPSK performs phase changes in multiples of 90 degrees, providing G3PLX extra symbol space to perform error correction.

Alice can use the same trick to form a polyglot with BPSK31, but this presents a number of signal processing challenges. Simply using the 90-degree shifts of QPSK31 would be a bit of an indiscretion, as BPSK interpreters would have wildly varying interpretations of the message, often decoding the hidden bits to visible junk characters.

Using a terribly small shift is a tempting idea, as Alice's use of balanced 170 and 190 degree transitions might be rounded out to 180 degrees by the receiver. Unfortunately, this would require *extremely* stable and well tuned radio equipment, giving Bob as much trouble receiving the signal as Eve is supposed to have!

Instead of adding additional phases to BPSK31, we propose instead that the error correction of QPSK31 be abused to encode additional bits. Alice can encode data by *intentionally inserting errors* in a QPSK31 bitstream, relying upon Eve's receiver to remove them by error correction. Bob's receiver, by contrast, would know that the error bits are where the data really is.

## 4.5 Classic RTTY (ITA2)

RTTY—pronounced "Ritty"—is a radio extension of military teletypewriters that has been in use since the early thirties. It consists of five-bit letters, using shifts to implement uppercase letters and foreign alphabets. Although implementation details vary, most amateur stations use 45 baud, 170Hz shift, 1 start bit, 2 stop bits, and 5 character bits. The higher frequency is a mark (one), while the lower frequency is a space (zero).

As more digital protocols other than CW and RTTY weren't legalized until the eighties, all sorts of clever tricks were thought up. Figure 4 shows RTTY artwork from W2PSU's article in the September 1977 issue of 73 Magazine. Lacking computerized storage and cheap audio cassettes, it was the style at the time to store long stretches of paper tape as rolls in pie tins, with taped labels on the sides.

Figure 6 describes Western Union's ITA2 alphabet used by RTTY, which is often—if imprecisely—called Baudot Code. In that figure, `1` indicates a high-frequency mark while `2` indicates a low-frequency space. Note that these letters are sent almost like a UART, least-significant-bit first with one start bit and two stop bits.

## 4.6 Some Ditties in RTTY

### 4.6.1 Differing Diddles

Unlike a traditional UART, RTTY sends an idle character—colloquially known as a Diddle—of five marks when no data is available. This is done to prevent the receiver from becoming desynchronized, but it isn't strictly mandatory. By not sending the diddle character (`11111`) when idle, the mark bit's frequency can be left idle for a bit, encoding extra information.

Additionally, there are not one but *two* possible diddle characters! Traditionally the idle is filled with `11111`, which means `Shift to Letters`, so the transmitter is just repeatedly telling the receiver that the next character will be a letter. You could also send `11011`, which means `Shift to Figures`. Sending it repeatedly also has no effect, and jumping between these two diddle characters will give you a side-channel for communication which won't appear in normal RTTY receivers. As an added benefit, it is visually less conspicuous than causing the right channel of your RTTY broadcast to briefly disap-

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BPSK | 10101101 | 00 | 111011101 | 000 | 1 | 00 | 10101101 | 000 | 111011101 | 00 | 1 | 00 |
| PSK31 | C | | Q | | [SP] | | C | | Q | | [SP] | |
| Idle | | 0 | | 1 | | 0 | | 1 | | 0 | | |
| BPSK | 101101 | 00 | 11 | 000 | 1 | 00 | 1111101 | 000 | 10111101 | 00 | 1111111 | 00 |
| PSK31 | d | | e | | [SP] | | A | | 1 | | I | |
| Idle | | 0 | | 1 | | 0 | | 1 | | 0 | | 0 |
| BPSK | 10101101 | 00 | 1110111 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PSK31 | C | | E | | | | | | | | | |
| Idle | | 0 | | | | | | | | | | |

Figure 3: `010100101000` Hidden in PSK31 Idle Bits

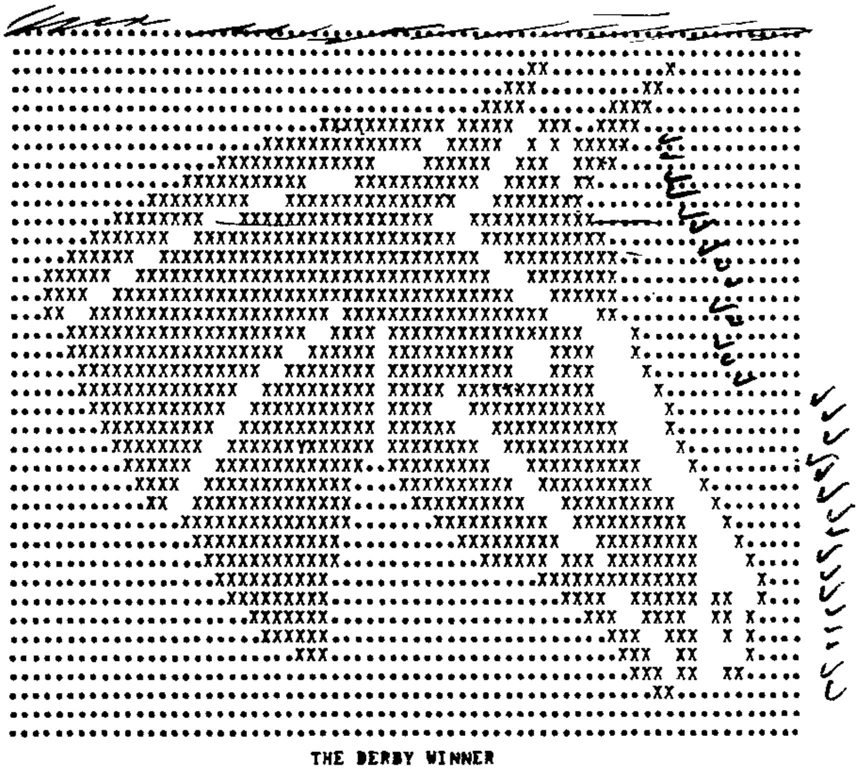


Figure 4: RTTY Art of Seattle Slew from the mid 1970's

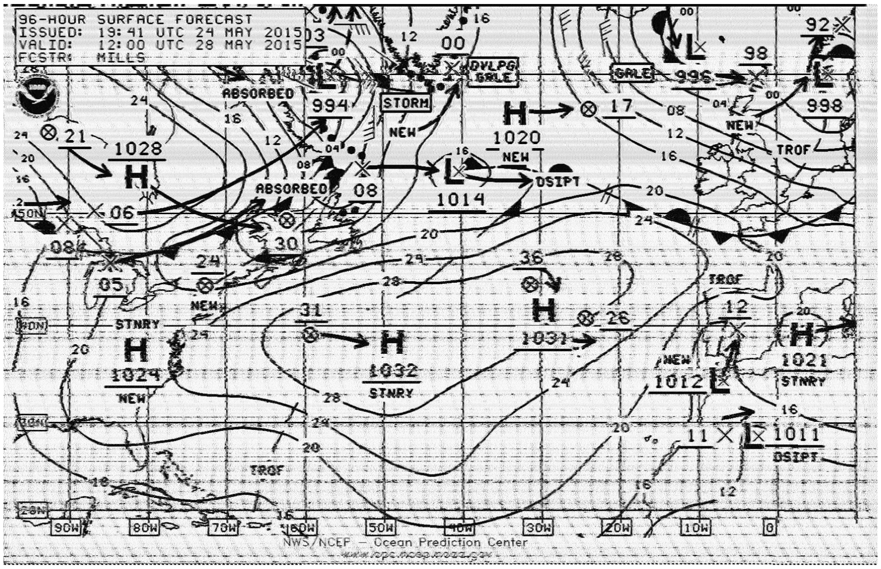


Figure 5: Weather Fax

|  | Letter | Figure |  | Letter | Figure |
|---|---|---|---|---|---|
| 00000 | Null | Null | 11010 | G | & |
| 00100 | Space | Space | 10100 | H | # |
| 10111 | Q | 1 | 01011 | J | ’ |
| 10011 | W | 2 | 01111 | K | ( |
| 00001 | E | 3 | 10010 | L | ) |
| 01010 | R | 4 | 10001 | Z | ” |
| 10000 | T | 5 | 11101 | X | / |
| 10101 | Y | 6 | 01110 | C | : |
| 00111 | U | 7 | 11110 | V | ; |
| 00110 | I | 8 | 11001 | B | ? |
| 11000 | O | 9 | 01100 | N | , |
| 10110 | P | 0 | 11100 | M | . |
| 00011 | A | – | 01000 | CR | CR |
| 00101 | S | Bell | 00010 | LF | LF |
| 01001 | D | WRU? | 11011 | FIGS |  |
| 01101 | F | ! | 11111 |  | LTRS |

Figure 6: RTTY’s ITA2 Alphabet

pear!

#### 4.6.2 Stop with the Stop Bits!

RTTY is described in the old UART tradition as `5/N/2`, meaning that it has `5` data bits, `N`o parity bits, and `2` stop bits. There’s a cool trick to UARTs that’s worth remembering: the transmitter can always have *more* stop bits than the receiver demands, and the receiver can always demand *fewer* stop bits than the transmitter sends.

### 4.7 Toe Tappin’ CW

Carrier Wave (CW) modulation—better known as Morse code—was the first widely deployed digital mode to replace spark-gap transmitters. Designed for a human operator to manually use, CW is a perfect choice for easy polyglots.

As a quick review, CW consists of dots and dashes. A dash is three times as long as a dot. The off-time between elements of a letter is as long as a dot, and the off-time between letters in a word is as long as a dash. The off-time between words is seven times as long as a dot, or a bit more than twice as long as a dash.

#### 4.7.1 QRSS

While other protocols have standard data rates, Morse relies on the recipient to adjust to the rate of the transmitter. Operators often find themselves unable to keep up with an expert or impatiently waiting on a station that transmits slowly, so shorthand was developed to ask the other side to change rate. `QRQ` requests that the other side transmit more quickly, and `QRS` requests that the other side slow down.

QRSS is a variant of CW in which the message is sent very, *very* slowly. Rather than a dot lasting a fraction of a second, it might last as long as a minute! A receiver can then take a recording of a very weak signal, slow down the recording, and visually observe the signal to determine its meaning.

While protocols such as RTTY and PSK31 don’t take kindly to the sorts of frequent interruptions that normal CW would impart, these protocols can easily produce QRSS transmissions that are legible by slowing down recordings. For example, Alice might send “`A1BOB A1BOB de A1ICE`” for a dot and “`A1BOB A1BOB de A1ICE. A1BOB A1BOB de A1ICE. A1BOB A1BOB de A1ICE.`” for a dash.

This is of course a bit easy to recognize from a waterfall, but it might be a fun way to meet your neighbors!

### 4.7.2 From Ethernet to Æther with Madeline

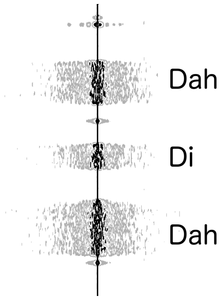
In a row house in Philly
that was covered with vines
Was an Ethernet network
in four twisted lines
In four twisted lines
they ran to the laundry
And to the satellite dish
and to the pantry
The twists ended too soon
and ceased to align
Interfering with 10 meters
all down the line
The protocol
was Madeline.

It's clear enough that you could transmit Morse code through Wifi by sending bursts of traffic, but what about wired Ethernet?

Some folks are very particular when wiring CAT5e cable, ensuring that the twisted pairs are untwisted at the last possible position before the connector. Other folks—such as your neighborly authors—are far less particular in their wiring, and when the wiring is performed poorly, interference is observed near 28.121 MHz!

Still better, the interference varies with traffic! When the network is idle, the interference appears as a nice thin carrier wave. When the network is busy, the interference grows to be nearly four hundred Hertz wide.

The following is a letter of Morse code transmitted from (poorly) wired Ethernet to the 10-meter band through what we are calling the Madeline protocol. This transmission isn't strong enough to carry very far, but the Baudline-generated waterfall in that figure was recorded from outside of a real house, with a signal generated by a real Ethernet network. The recording was made by an Upper SideBand receiver tuned to 28.120 MHz.[20] The narrow-band signal at 28.121 MHz becomes wide whenever lots of traffic goes across the wired network; in this case, from activity on a VNC session.



## 4.8 Patching FLDigi

All of this high-falutin' theorizin' don't do a lick of good without some software to back it up. Supposing that Alice is a modern unix programmer, but that Bob hasn't written code for anything more modern than a Commodore 64, Alice will need to provide him with a GUI application that easily interfaces with his radio.

The most direct route for this is to patch FLDigi, a popular open source application for digital communication over ham radio with a live operator. Internally, FLDigi implements softmodems for CW, PSK31, RTTY, WEFAX, and several other protocols.

## 4.9 Part 97; or, Don't be a Jerk!

Be aware that in general, it's both illegal and immoral to be a jerk on the amateur bands. Interference is forbidden in amateur radio, not because jamming research is bad, but because it's rude to stomp on someone else's transmission. Cryptography is forbidden in amateur radio, not because of any evil conspiracy to destroy privacy, but because cryptography makes a transmission opaque, preventing newcomers from joining the conversation.

So for those of you who do not live in Nehemiah Scudder's oppressive theocracy, please be so kind as to keep your polyglot messages unencrypted. Make a fox hunt of sorts out of your protocol experimentation, with the surface PSK31 message advertising your callsign along with the name and parameters of your real protocol.

– – – –   – – –   – – – –

We hope that this article has taught you a little about radio and signal processing. Get an amateur license, build a station, and start experimenting with new protocols on the friendly airwaves.

73's from Appalachia,
—Travis and Muur

[20] `unzip pocorgtfo08.pdf madelinek.wav`