|  |  |
| --- | --- |
| **Radiocommunication Study Groups** |  |
|  |  |
|  |  |
| Source: Document 5A/TEMP/319 | **Annex 4 to**  **Document 5A/788-E** |
| **17 November 2011** |
| **English only** |
| Annex 4 to Working Party 5A Chairman’s Report | |
| Working document towards a Preliminary Draft new RECOMMENDATION ITU-R M.[VARICODE] | |
| Telegraphic alphabet for data communication by phase shift keying at 31 baud | |

Summary

Phase shift keying at 31 baud (PSK‑31) has become the most widely used data communication mode in the amateur radio service since its introduction in 1998. PSK‑31 utilizes a binary telegraphic alphabet, commonly called “Varicode,” which accommodates all ASCII characters, and uses shorter sequences for more commonly used characters. A Recommendation formally documenting the Varicode character set and the PSK‑31 transmission protocol is contemplated, and this document sets forward this description of both as a first input to a working document towards such a Recommendation.

Working document towards a Preliminary Draft  
new RECOMMENDATION ITU-R M.[VARICODE]

Telegraphic alphabet for data communication by phase shift keying at 31 baud

Scope

*[To be developed]*

The ITU Radiocommunication Assembly,

considering

a) that phase shift keying at a data rate of 31 baud has become a predominant transmission mode in the amateur and amateur-satellite services;

b) that phase shift keying at 31 baud utilizes a telegraphic alphabet, commonly called “Varicode,” optimized for the English language, in which more frequently used characters occupy fewer bits;

c) that telegraphic alphabets should be documented and updated from time to time to meet the needs of radiocommunication services,

recommends

**1** that Annex 1 should be used to define Varicode characters and their applications in radiocommunication services.

Annex 1

# 1 Introduction

PSK‑31 is a digital communication mode which is intended for live keyboard-to-keyboard conversations, similar to radioteletype. Its data rate is 31.25 bauds (about 50 words per minute), and its narrow bandwidth (approximately 60 Hz at −26 dB) reduces its susceptibility to noise. PSK‑31’s ITU emission designator is 60H0J2B. It uses BPSK modulation without error correction or QPSK modulation with error correction (convolutional encoding and Viterbi decoding). In order to minimize occupied bandwidth, the output is cosine-filtered before reaching the transmitter audio input. PSK‑31 is readily monitored and the most popular implementation uses DSP software running on a computer soundcard inside a computer.

Each transmission has a preamble, an idle signal of continuous zeroes corresponding to continuous phase reversals at the symbol rate of 31.25 reversals/second, and a postamble, continuous unmodulated carrier representing a series of logical ones. The absence of phase reversals squelches the decoder.

While a symbol rate of 31.25 bauds is typical of most amateur service use, the symbol rate can be varied in direct proportion to the frequency of phase reversals. Transmissions at symbol rates as high as 125 bauds have been achieved.

# 2 Varicode characters

Different characters are represented by a variable-length combination of bits called Varicode. Because shorter bit-lengths are used for the more common letters in the English language, Varicode improves efficiency in terms of the average character duration. Varicode is also self-synchronizing. No separate process is needed to define where one character ends and the next begins, since the pattern used to represent a gap between two characters (at least two consecutive zeroes) never occurs in a character. Because no Varicode characters can begin or end with a zero, the shortest character is a single one by itself. The next is 11, then 101, 111, 1011, and 1101, but not 10, 100, or 1000 (because they end with zeroes), and not 1001 (since it contains two consecutive zeros). This scheme generates the 128-character ASCII set with ten bits. [*Ed. Note* – Consider addressing accommodation of other character sets].

The Varicode character set is shown below, starting with NUL and ending with DEL. The codes are transmitted left bit first, with 0 representing a phase reversal on BPSK and 1 representing a steady carrier. A minimum of two zeros is inserted between characters. Some implementations may not handle all the codes below 32. Note that the lower case letters have the shortest patterns and so are the fastest to transmit.

The varicode character set control characters

| Varicode | Abbreviation | Description |
| --- | --- | --- |
| 1010101011 | NUL | Null character |
| 1011011011 | SOH | Start of Header |
| 1011101101 | STX | Start of Text |
| 1101110111 | ETX | End of Text |
| 1011101011 | EOT | End of Transmission |
| 1101011111 | ENQ | Enquiry |
| 1011101111 | ACK | Acknowledgment |
| 1011111101 | BEL | Bell |
| 1011111111 | BS | Backspace |
| 11101111 | HT | Horizontal Tab |
| 11101 | LF | Line feed |
| 1101101111 | VT | Vertical Tab |
| 1011011101 | FF | Form feed |
| 11111 | CR | Carriage return |
| 1101110101 | SO | Shift Out |
| 1110101011 | SI | Shift In |
| 1011110111 | DLE | Data Link Escape |
| 1011110101 | DC1 | Device Control 1 (XON) |
| 1110101101 | DC2 | Device Control 2 |
| 1110101111 | DC3 | Device Control 3 (XOFF) |
| 1101011011 | DC4 | Device Control 4 |
| 1101101011 | NAK | Negative Acknowledgement |
| 1101101101 | SYN | Synchronous Idle |
| 1101010111 | ETB | End of Trans. Block |
| 1101111011 | CAN | Cancel |
| 1101111101 | EM | End of Medium |
| 1110110111 | SUB | Substitute |
| 1101010101 | ESC | Escape |
| 1101011101 | FS | File Separator |
| 1110111011 | GS | Group Separator |
| 1011111011 | RS | Record Separator |
| 1101111111 | US | Unit Separator |
| 1110110101 | DEL | Delete |

Printable characters

| Varicode | Glyph |  | Varicode | Glyph |  | Varicode | Glyph |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | SP |  | 1010111101 | @ |  | 1011011111 | ` |
| 111111111 | ! |  | 1111101 | A |  | 1011 | a |
| 101011111 | " |  | 11101011 | B |  | 1011111 | b |
| 111110101 | # |  | 10101101 | C |  | 101111 | c |
| 111011011 | $ |  | 10110101 | D |  | 101101 | d |
| 1011010101 | % |  | 1110111 | E |  | 11 | e |
| 1010111011 | & |  | 11011011 | F |  | 111101 | f |
| 101111111 | ' |  | 11111101 | G |  | 1011011 | g |
| 11111011 | ( |  | 101010101 | H |  | 101011 | h |
| 11110111 | ) |  | 1111111 | I |  | 1101 | i |
| 101101111 | \* |  | 111111101 | J |  | 111101011 | j |
| 111011111 | + |  | 101111101 | K |  | 10111111 | k |
| 1110101 | , |  | 11010111 | L |  | 11011 | l |
| 110101 | - |  | 10111011 | M |  | 111011 | m |
| 1010111 | . |  | 11011101 | N |  | 1111 | n |
| 110101111 | / |  | 10101011 | O |  | 111 | o |
| 10110111 | 0 |  | 11010101 | P |  | 111111 | p |
| 10111101 | 1 |  | 111011101 | Q |  | 110111111 | q |
| 11101101 | 2 |  | 10101111 | R |  | 10101 | r |
| 11111111 | 3 |  | 1101111 | S |  | 10111 | s |
| 101110111 | 4 |  | 1101101 | T |  | 101 | t |
| 101011011 | 5 |  | 101010111 | U |  | 110111 | u |
| 101101011 | 6 |  | 110110101 | V |  | 1111011 | v |
| 110101101 | 7 |  | 101011101 | W |  | 1101011 | w |
| 110101011 | 8 |  | 101110101 | X |  | 11011111 | x |
| 110110111 | 9 |  | 101111011 | Y |  | 1011101 | y |
| 11110101 | : |  | 1010101101 | Z |  | 111010101 | z |
| 110111101 | ; |  | 111110111 | [ |  | 1010110111 | { |
| 111101101 | < |  | 111101111 | \ |  | 110111011 | | |
| 1010101 | = |  | 111111011 | ] |  | 1010110101 | } |
| 111010111 | > |  | 1010111111 | ^ |  | 1011010111 | ~ |
| 1010101111 | ? |  | 101101101 | \_ |  |  |  |

# 3 QPSK mode

[*Ed. Note −* Reference to other Recommendations is likely appropriate in lieu of this section.]

The QPSK mode reduces the error rate while keeping the bandwidth and the traffic speed the same. There is a 3 dB signal to noise degradation with QPSK, because the same transmitter power is being shared by twice the signals. Therefore, the error-correction scheme has to be at least good enough to correct the extra errors which result from the 3 dB SNR penalty, and preferably a lot more, or it will not be worth doing. By doing simulations in a computer, and tests on the bench with a noise generator, it has been found that when the bit error-rate is less than 1% with BPSK, it is much better than 1% with QPSK and error-reduction, but when the BER is worse than 1% on BPSK, the QPSK mode is actually worse than BPSK. Therefore, if we are dealing with radio paths where the signal is just simply very noisy, there is actually no advantage to QPSK at all.

On-the-air testing shows that QPSK with the convolutional coding for error-reduction is usually better than BPSK, except where the signal was deliberately attenuated to make it artificially weak. Typical radio circuits are far from being non-fading with white noise. Typical radio paths have errors in bursts rather than randomly spread, and error-reduction schemes can give useful benefits in this situation in a way that cannot be achieved by anything which can be done in the linear part of the signal path. With the convolutional coding used in PSK‑31, a 5:1 improvement is typical, but it does depend on the kind of path being used. There may be times when one mode works better than the other, and other times when the reverse will be the case. The switch between straight BPSK and error-corrected QPSK modes in PSK‑31 is done with both the bandwidth and the data-rate remaining the same. Contacts tend to start on BPSK and change to QPSK if both stations agree. Although both stations have to be using the same sideband in QPSK, it does not matter for BPSK.

# 4 Convolutional coding

[*Ed. Note* − Reference to other Recommendations is likely appropriate in lieu of this section.]

Convolutional coding is used to reduce errors in the QPSK mode. In a convolutional code, the characters are converted to a bitstream and then this bitstream is itself processed to add the error-reduction qualities. Since the convolutional code used in PSK‑31 doubles the number of data bits, it is a natural choice for the QPSK mode which provides double the bit-rate available with BPSK. The convolutional encoder generates one of the four phase-shifts, not from each data bit to be sent, but from a sequence of them. This means that each bit is effectively spread out in time, intertwined with earlier and later bits in a precise way. The more spread out, the better will be the ability of the code to correct bursts of noise, but too great a spread would introduce an excessive transmission delay. A time spread of 5 bits was chosen.

It is not quite correct to refer to the convolutional code system as error-correcting since the raw data is not actually transmitted in its original form and therefore it makes no sense to talk about it being corrupted by the link and corrected in the decoder. In PSK‑31, the raw data is transformed from binary (1 of 2) to quaternary (1 of 4) in such a way that there is a precisely known pattern in the sequence of quaternary symbols. In the code used in PSK‑31, each quaternary symbol transmitted is derived from a run of 5 consecutive data bits. This means that each binary bit to be transmitted generates a 5-symbol sequence, overlapping with the sequences from adjacent bits, in a predictable way which the receiver can use to estimate the correct sequence even in the presence of corruptions in parts of the sequence.

The convolutional code

The left columns in the following table contain the 32 combinations of a run of five Varicode bits, transmitted left bit first. The right columns are the corresponding phase shifts to be applied to the carrier, in degrees. A continuous phase advance is the same as an HF frequency shift.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 00000 | 180 |  | 01000 | 0 |  | 10000 | +90 |  | 11000 | −90 |
| 00001 | +90 |  | 01001 | −90 |  | 10001 | 180 |  | 11001 | 0 |
| 00010 | −90 |  | 01010 | +90 |  | 10010 | 0 |  | 11010 | 180 |
| 00011 | 0 |  | 01011 | 180 |  | 10011 | −90 |  | 11011 | +90 |
| 00100 | −90 |  | 01100 | +90 |  | 10100 | 0 |  | 11100 | 180 |
| 00101 | 0 |  | 01101 | 180 |  | 10101 | −90 |  | 11101 | +90 |
| 00110 | 180 |  | 01110 | 0 |  | 10110 | +90 |  | 11110 | −90 |
| 00111 | +90 |  | 01111 | −90 |  | 10111 | 180 |  | 11111 | 0 |

For example, consider the space symbol – a single 1 preceded and followed by character gaps of five zeroes each: 00000 100000. Overlaying a five-bit-wide window on 00000100000, and sliding it from left to right (one bit at-a-time) is illustrated in the following table.

0 0 0 0 0 1 0 0 0 0 0 Phase°

0 0 0 0 0 180

0 0 0 0 1 +90

0 0 0 1 0 -90

0 0 1 0 0 -90

0 1 0 0 0 0

1 0 0 0 0 +90

0 0 0 0 0 180

Representing 00000100000 would be the successive run-of-five groups 00000, 00001, 00010, 00100, 01000, 10000, 00000. This results in the transmitter sending the QPSK pattern 180, +90, −90, −90, 0, +90, 180. Note that a continuous sequence of zeros (the idle sequence) gives continuous reversals, the same as BPSK.

# 5 VITERBI decoding

[*Ed. Note −* Reference to other Recommendations may be appropriate in lieu of this section.]

Viterbi decoding is used on the receiving side. It consists of a whole bank of parallel encoders, each fed with one possible guess at the transmitted data sequence. The outputs of these parallel encoders are all compared with the received symbol stream. Each time a new symbol is received, the encoders need to add an extra bit to their sequence guesses and consider that the new bit might be a 0 or a 1. This doubles the number of sequence guesses, but a clever technique allows half of all the guessed sequences to be discarded as being less likely than the other half, and this means that the number of guesses being tracked stays constant. After a large number of symbols have been received, the chances of a wrong guess at the first symbol tends to zero, so the decoder can be pretty sure that the first bit was right and it can be fed to the output. In practice this means that the decoder always outputs decoded data bits some time after they have been received. The one-way delay in PSK‑31 is 25 bits (800 ms) which is long enough to make sure that the decoder has done a good job, but not so long that it introduces an unacceptable delay in displaying the received text.

[REFERENCES

Steve Ford, WB8IMY, PSK‑31—Has RTTYs Replacement Arrived? QST, May 1999, pp 41-41.

Peter Martinez, G3PLX, PSK‑31: A New Radio-Teletype Mode, Radio Communication, December 1998 and January 1999.

PSK‑31 Gets Raves, The ARRL Letter, Vol. 18, No. 7, February 12, 1999.

Steve Karty, N5SK, PSK‑31 Spec, available at [http://www.arrl.org/PSK‑31-spec](http://www.arrl.org/psk31-spec).]