Unified SPUTNIX protocol (USP)

Protocol description

Document revision 1.04

The protocol is intended for use on space-to-Earth and Earth-to-space links in telemetry and telecommand transmission systems (TM/TC).
The protocol describes the physical and data link layers.

The protocol is primarily intended for use on relatively low-speed (1200-115200 bps) half-duplex channels, taking into account the needs of small spacecraft in low Earth orbit. Its implementation on devices based on microcontrollers and general use integrated transceivers is taken into account.

General agreements

Except where otherwise noted, the following agreements apply:

- All fields are MSB first.
- For multi-byte fields bytes order is always noted in the relevant section of the document
- All bit sequences mentioned in the document are transmitted from left to right.

Radio signal parameters

The protocol does not restrict use to any specific frequencies. However, recommended configurations are available to allow transmission lines to comply with the amateur radio regulations for use on amateur radio links.

At this point, the use of GMSK (Gaussian Frequency Shift Keying with Minimum Shift) is determined and recommended. However, there are no fundamental restrictions that prevent the use of other types of manipulation. In the case of using FSK, a lower frequency value corresponds to zero, and a higher value to one.

General frame structure

The general frame structure is shown in the figure below

Preamble	Sync	PLS-code	Type	Data packet
>=32 bits	64 bits	64 bits	16 bits*	

Shaded part of the frame is coded by convolutional code, scrambled, and coded with RS-code,

* The length of the type field is specified before the above codes are superimposed.

Preamble

It is recommended to use a 32-bit preamble with alternating zeros and ones 55555555h.

Sync

64-bit sync sequence 5072F64B2D90B1F5h is used.
On the receiving side, it is recommended to consider the correct sequence, which differs from the indicated one by 13 bits or less. The probability of a false synchronization per bit is 9.4 * 10-7, which at a rate of 9600 bps gives an average of one false synchronization in 109 seconds.

As a rule, hardware transceivers support automatic detection of a 32-bit sequence only . It is recommended in this case to set the threshold to 7 permissible errors, and check the second part of the sequence programmatically with the same threshold. The probability of a false sync per bit is 1.1 * 10-6, which at 9600 bps gives an average of one false sync every 94 seconds.

The Eb / N0 curve for each option is given in the section "Energy Capabilities of the Protocol".

PLS-code

The physical layer signaling (PLS) code is transmitted directly behind the sync sequence. The code is 7 bits long, encoded with 64-7 code, and the encoded field is 64 characters long. The Hamming distance of the code is 32 bits.

The code is a linear block code with the following generating matrix:

$$G=\begin{vmatrix}
0011001100110011001100110011001100110011001100110011001100110011\\
0000111100001111000011110000111100001111000011110000111100001111\\
0000000011111111000000001111111100000000111111110000000011111111\\
0000000000000000111111111111111100000000000000001111111111111111\\
0000000000000000000000000000000011111111111111111111111111111111\\
1111111111111111111111111111111111111111111111111111111111111111\\
0101010101010101010101010101010101010101010101010101010101010101
\end{vmatrix}$$

The most significant bit of the original value is multiplied by the first row of the matrix, the least significant bit is multiplied by the last one, the resulting values are summed modulo 2, or, in other words, an exclusive operation is performed or between all rows of the table, opposite to which there is one in the encoded value. The resulting value is scrambled by modulo 2 addition (XOR) with the bit sequence
0111000110011101100000111100100101010011010000100010110111111010

The code used is fully equivalent to that used in the standards DVB-S2 (part 5.5.2) and CCSDS 131.2-B-1 (part 5.3.3), despite a different way of defining.

PLS-code carries information about the type and parameters of the encoding, the size of the FEC codeblock.

At the moment, only one type of data encoding has been implemented with two possible sizes of the encoded data block :

PLS-code	FEC	Data length with the header, bytes
0	Convolutional code (rate ½ K=7), Reed-Solomon code (255,223)	223
1	Convolutional code (rate ½ K=7), Reed-Solomon code (255,223)	48

* When using a data block shorter than the length of the data block of the Reed-Solomon code, virtual padding is performed (see below).

All other values are currently reserved.

Data frame

Next, a data frame (frame) with a header is transmitted, on the transmitting side it is sequentially subjected to the following transformations corresponding to separate sections of CCSDS 131.0-B-3:

- Coding by Reed-Solomon code;
- Scrambling;
- Convolutional coding.

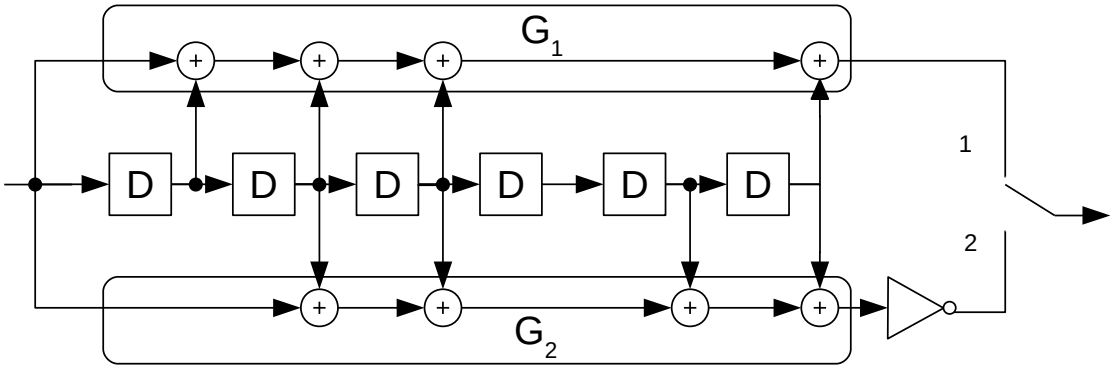
FEC coding

Convolutional coding

The convolutional code used is identical to that recommended in CCSDS Book 131/0-B-3, clause 3.3.1. It should be noted, however, that the sync sequence, like the PLS code, is not convolutional encoded, as recommended by the referenced document.

Code parameters:
type: convolutional code with maximum likelihood decoding;
relative code rate r = ½;
constraint length K = 7 bits;
connection vectors: G1 = 1111001, G2 = 1011011;
inversion: G2 output is inverted.

he encoder circuit is shown in the figure below. (CCSDS 131.0-B-3 Figure 3-1):



The squares with the D symbol denote a 1-bit delay, the adders are modulo 2. The first symbol is transmitted at switch position 1.

Scrambler

The scrambler is used in accordance with CCSDS 131/0-B-3 section 10.4.1.

Scrambling is performed by an exclusive operation or (addition modulo 2) with a sequence generated by a polynomial:

$h(x) = x^8 + x^7 + x^5 + x^3 + 1$

This is an M-sequence of length 255. The first 40 bits of the sequence are shown below :
1111 1111 0100 1000 0000 1110 1100 0000 1001 1010 . . . .

Reed-Solomon code

The Reed-Solomon code (255,223) is used in the version described in CCSDS 131/0-B-3 Section 4.

Shortening the Reed-Solomon code

If the length of the data block (frame with header), determined by the PLS code, is less than the length of the code data block, the code is shortened by padding the data block with zeros in front to the size of the data block of the Reed-Solomon code, encoding and removing padded zeros before scrambling and convolutional encoding. On the receiving side, zeros are added again before decoding. This procedure is called virtual filling and is based on the fact that the Reed-Solomon code is a systematic code and does not change the data block during encoding, but only supplements it with control characters. The codeblock shortening is done in accordance with clause 4.3.7 of CCSDS 131/0-B-3.

175 bytes	48 bytes	32 bytes
Virtual fill with zeros	Data block	RS check symbols

The part of the coded block that is discarded during transmission before scrambling and restored during reception after descrambling is highlighted is shaded .

If a data frame with a header is shorter than a data block, the block is also padded with zeros on the right, but they are transmitted in the usual way and this is not a code shortening.

Data frame structure

The data frame is generally preceded by a two-byte header consisting of a type field, which is IEEE 802.3 EtherType.

The length field is generally not provided. When adding encapsulation to the USP (with assigning new EtherType) of protocols that rely on external framing, that is, they do not include a length field, it may be necessary to add a header with such a field. It is recommended to use two bytes following the type field in little-endian format. The length must include all data encapsulated by the protocol, but not include any USP headers, in other words, it counts data starting with the byte following the length field. The length field encodes the length of the payload in bytes, including all subsequent headers, but not including the frame header itself.

In total, the data block has the following form :

EtherType	Packet data
16 bits	0...221 bytes
Big endian	–

In the case of encapsulation of a protocol that requires transmission of length, it is recommended to use this structure :

EtherType	Length field	Packet data
16 бит	16 бит	0...219 байт
Big endian	Little endian	–

Data integrity check

The USP does not use a separate checksum for data integrity checking. It relies on the control of the Reed-Solomon code, which, given the parameters of the code, provides a sufficient degree of verification.

AX.25 packet transmission using USP

Encapsulation of the AX.25 protocol is performed in a similar way, but not completely identical to the AX.25 BPQ protocol created for the same purpose.
In this case, the EtherType field uses the value 08FFh (big-endian, FF08h in little-endian, unofficially assigned, but de facto used in existing implementations). Before the beginning of the AX.25 header, there is a length value that carries the length of the AX.25 packet, including the AX.25 header. Note that this length is not the same as AX.25 BPQ implementations for EtherNet that add an extra 4 bytes to avoid confusion and problems caused by length mismatch.
In this case, HDLC framing is not used, that is, flags and checksum are not transmitted, and bit stuffing is not used. The task of determining the packet length is played by the length from the frame header, and the integrity control is carried out by means of the USP.

In total, the data block has the following form :

EtherType = 08FFh*	Length	Загловок AX.25	AX.25 packet data
16 bits	16 bits	15-31 bytes**	0...203 bytes
Big endian	Little endian	–	–

*FF08h в little-endian.
**in a typical space telemetry case unnumbered frames are used, so the header length is 16 bytes. The specified maximum number of data bytes is specified for this particular case. For different header lengths, the maximum data size must be adjusted.

To verify a HEX dump of a packet, you can use the free program Wireshark in dummy header mode or add an arbitrary 12 bytes of MAC addresses at the beginning of the dump.

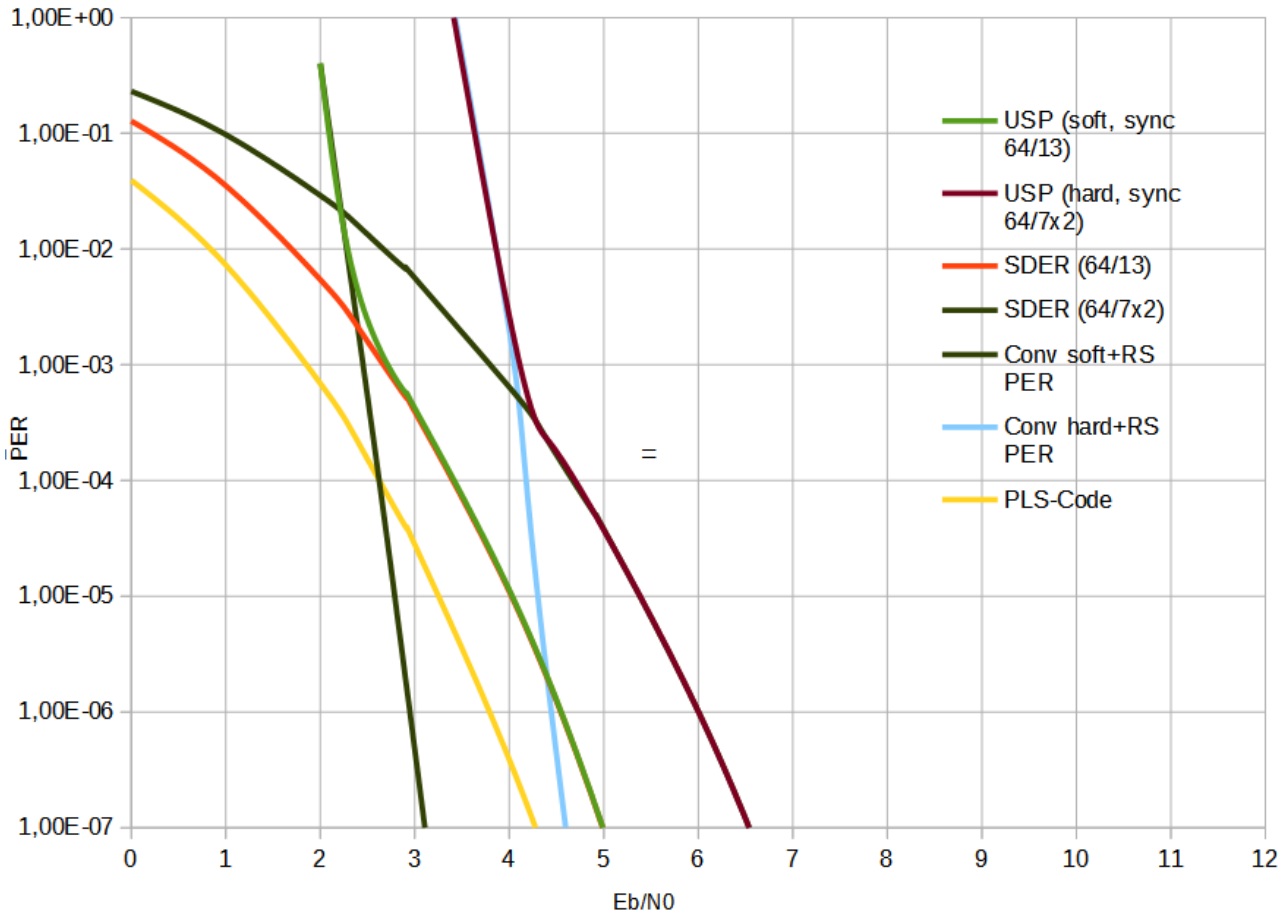
Appendix 1. Protocol energy capabilities

The energy capabilities of the protocol in the form of an Eb / N0 graph are shown in the graph below. The graph is shown in two versions:

- the soft Viterbi algorithm and detecting a sync with an allowance of 13 errors
- hard decoding with an allowance of 7 errors in each half of the sync .

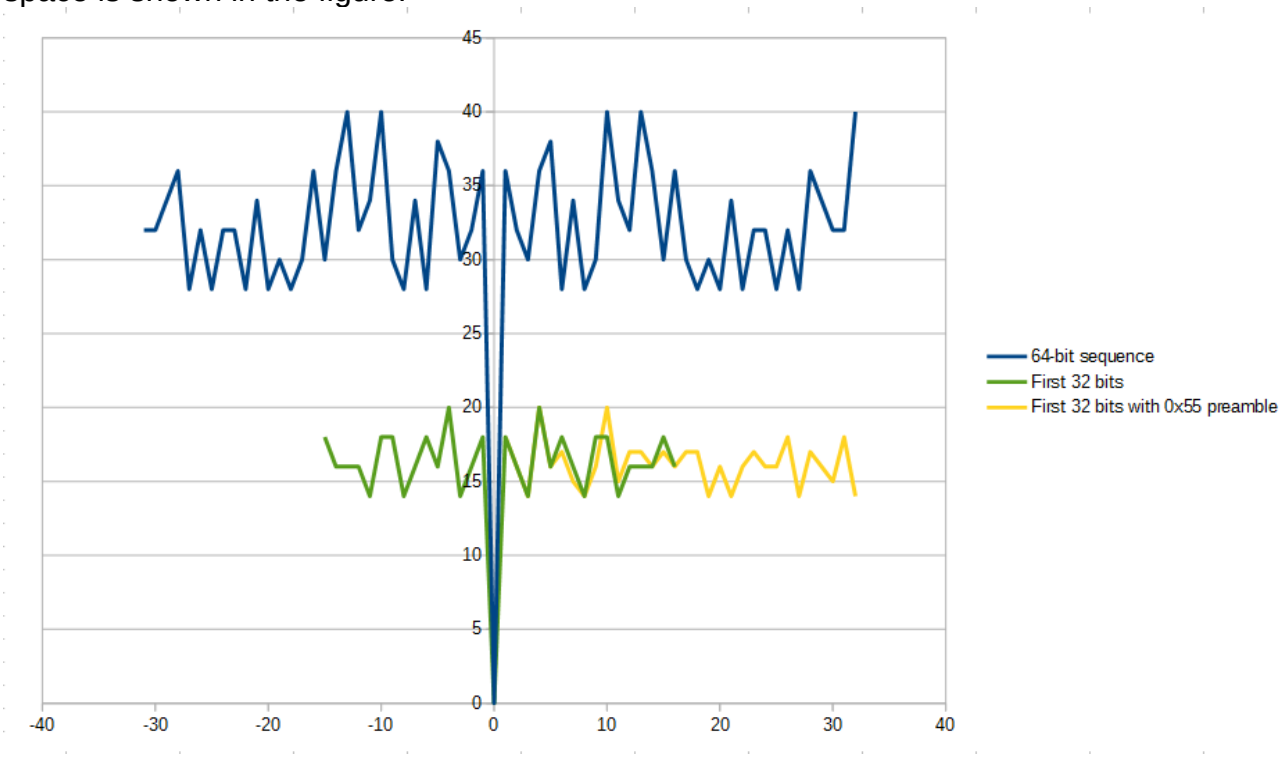
It also shows the contribution of individual components: the probability of synchronization error, the probability of incorrect reception of the PLS code and the probability of error when decoding the FEC. The graph shows that with soft decoding at Eb / N0≈2.8, the probability of successful frame reception is 99.9% (i.e., PER≤0.001) for the additive Gaussian noise model (AGWN).

With hard decoding, the same probability of successful reception is ensured at Eb / N0≈4.1, i.e., the parameters are degraded by about 1.5 dB.



Appendix 2. Sync

The sequence is balanced in terms of the number of zeros and ones, and has a maximum of 5 consecutive zeros and 5 consecutive ones.
The sequence is optimized to have both itself and its first half autocorrelation acceptable - by itself and with the addition of preamble.
The graph of the autocorrelation function for different use cases in the Hamming distance space is shown in the figure.



Appendix 3. Justification of the adopted technical solutions

Overview of existing protocols
Before creating our own protocol, the existing ones were analyzed. In particular, the following were considered:
• CCSDS 131/0-B-3 TM synchronization and channel coding
• Protocol of VHF transceiver GOMspace NanoCom U482 / AX100 in ASM and ASM + Golay mode
• Protocol of the device AAUSAT-4
• Family of FEC protocols of the AO-40 apparatus and its modifications

All mentioned protocols use FEC. Below is a brief description and reasons why the protocol was not used without modification.

CCSDS 131/0-B-3 TM synchronization and channel coding
https://public.ccsds.org/Pubs/131x0b2ec1s.pdf
USP is based on this protocol. The main problem when used in half-duplex low-speed lines is the inability to dynamically change the frame length, which leads to inefficient channel use, a large round trip, and, as a consequence, to a slowdown in transmission.

GOMspace NanoCom U482 / AX100 VHF transceiver protocol in ASM or ASM + Golay mode
This protocol has a length field that allows short packets to be used when required. However, it also has disadvantages. The protocol uses a sync word of short length, as well as an uncoded length field, which limits the energy capabilities of the radio channel to values that are far from the capabilities of the used FEC. ASM+Golay is much better, but anyway is limits energy capabilities far from achievable with the used FEC.

AO-40 FEC family of protocols and its modifications
https://www.amsat.org/articles/g3ruh/125.html
The protocol is well optimized for operation in fading conditions and has good energy capabilities. The problem is the inability to dynamically change the frame length.

So there are our design goals:
– use full capabilities off selected FEC code;
– have an ability to send long packets and short frames (for acknowledges and so on) without significant time losses.
– have an ability to extend the protocol with new FECs, frame sizes, keeping backward compatibility.

It was decided to use CCSDS 131/0-B-3, where it’s possible, but:
– add long 64-bit sync, optimized to have both itself and its first half autocorrelation acceptable - by itself and with the addition of preamble. Leave sync outside any FEC codes;
– add PLS-code, like used in DVB-S2, to select FEC and codeblock and frame size. Have most of the values reserved for future use;
– recommend a way to use AX.25 as internal frame.

Document revision history

Rev. 1.01 First internal revision
Rev. 1.02 Sync is changed to improve first-half self-corellation, for adopting to hardware transcievers.
Rev. 1.03 More sync improvement, to optimize full autocorellation and two half autocorellations – with preamble and without it.
Rev. 1.04 Appendixes renumbering. Justification of the adopted technical solutions clarified.