The Consultative Committee for Space Data Systems

Draft Recommendation for
Space Data System Standards

TM SYNCHRONIZATION AND CHANNEL CODING

DRAFT RECOMMENDED STANDARD

CCSDS 131.0-P-4.1

PINK SHEETS

February 2023

The Consultative Committee for Space Data Systems

Draft Recommendation for
Space Data System Standards

TM SYNCHRONIZATION AND CHANNEL CODING

DRAFT RECOMMENDED STANDARD

CCSDS 131.0-P-4.1

PINK SHEETS

February 2023

10 PSEUDO-RANDOMIZER

10.1 OVERVIEW

In order for the receiver system to work properly, every data capture system at the receiving end requires that the incoming signal have sufficient bit transition density (see recommendation 2.4.9 in reference [5]), and allow proper synchronization of the decoder. The incoming signal must also be free of significant spectral lines, and be free of patterns that interfere with codeword synchronization and validation (see 2.2.2).

~~NOTE — Designers should note that the length-255 pseudo-randomizer may introduce spectral lines at 1/255 of the symbol rate, and these may be significant in some systems.~~

In order to ensure proper receiver operation, the data stream must be sufficiently random. The Pseudo-Randomizer defined in this section is the preferred method to ensure sufficient randomness for all combinations of CCSDS-recommended modulation and coding schemes. The Pseudo-Randomizer defined in this section is required unless the system designer verifies proper operation of the system if this Randomizer is not used.

NOTE – Problems with telemetry links have been encountered because this Pseudo-Randomizer was not used, and sufficient randomness was not ensured by other means and properly verified.

The presence or absence of pseudo-randomization is fixed for a Physical Channel and is *managed* (i.e., its presence or absence is not signaled in the transmitted data stream but must be known a priori) by the receiver.

10.2 PSEUDO-RANDOMIZER DESCRIPTION

NOTE – This subsection (including figure 10-1) does not apply to the case of LDPC encoding of a stream of SMTFs, where randomization is applied according to 8.3 and figure 8-3.

10.2.1 The method for ensuring sufficient transitions is to exclusive-OR each bit of the codeblock, codeword, or Transfer Frame with a standard pseudo-random sequence.

10.2.2 If the pseudo-randomizer is used, on the sending end it shall be applied to the codeblock, codeword, or Transfer Frame after Reed-Solomon, Turbo, or LDPC encoding (if any of these are used), but before convolutional encoding (if used). On the receiving end, it shall be applied to derandomize the data after convolutional decoding (if used) and codeblock or codeword synchronization but before Reed-Solomon, Turbo, or LDPC decoding (if any of these are used).

NOTE – The configuration at the sending end is shown in figure 10-1.

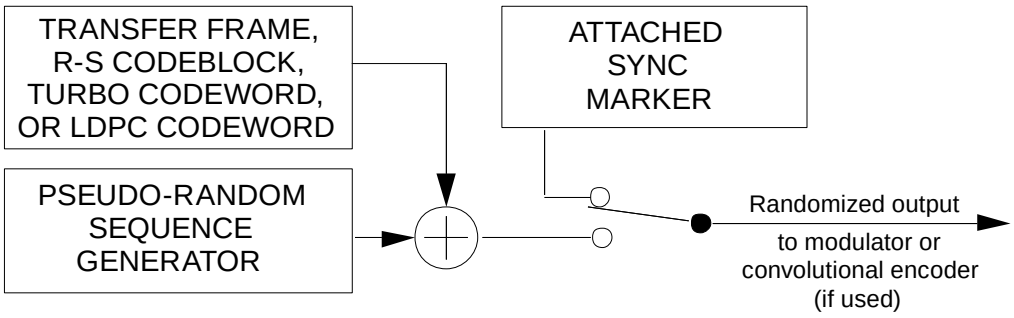


Figure 10-1: Pseudo-Randomizer Configuration

10.3 SYNCHRONIZATION AND APPLICATION OF PSEUDO-RANDOMIZER

10.3.1 The Attached Sync Marker (ASM) shall be used for synchronizing the pseudo-randomizer.

NOTE – The Attached Sync Marker (ASM) is already optimally configured for synchronization purposes.

10.3.2 The pseudo-random sequence shall be applied starting with the first bit of the codeblock, codeword, or Transfer Frame. On the sending end, the codeblock, codeword, or Transfer Frame shall be randomized by exclusive-ORing the first bit of the codeblock, codeword, or Transfer Frame with the first bit of the pseudo-random sequence, followed by the second bit of the codeblock, codeword, or Transfer Frame with the second bit of the pseudo-random sequence, and so on.

10.3.3 On the receiving end, the original codeblock, codeword, or Transfer Frame shall be reconstructed (i.e., derandomized) using the same pseudo-random sequence.

10.3.4 After locating the ASM in the received data stream, the data immediately following the ASM shall be derandomized.

NOTES

1 The ASM was not randomized and is not derandomized.

2 Derandomization can be accomplished by performing exclusive-OR with hard bits or inversion with soft bits.

10.4 SEQUENCE SPECIFICATION

10.4.1 The pseudo-random sequence shall be 131071 bits in length and shall be generated using the following seventeen-degree polynomial:

$$h(x) = x^{17} + x^{14} + 1$$

10.4.2 For backward compatibility with legacy systems, a 255-bit pseudo-random sequence may be generated instead, by using the following eight-degree polynomial:

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

NOTE – Designers should note that this 255-bit pseudo-randomizer may introduce spectral lines at 1/255 of the symbol rate, and these may be significant in some systems. In addition, it could not guarantee full compliance with ITU power flux density limits.

10.4.3 This sequence shall begin at the first bit of the codeblock, codeword, or Transfer Frame and shall repeat after ~~255 bits~~131071 bits or 255 bits, depending on the selected polynomial, continuing repeatedly until the end of the codeblock, codeword, or Transfer Frame. The sequence generator shall be initialized to ~~the all-ones~~'11000111000111000' in case of the 17-degree polynomial, or to the 'all-ones' state, in case of the 8-degree polynomial, state at the start of each codeblock, codeword, or Transfer Frame.

~~NOTE – The first 40 bits of the pseudo-random sequence from the generator are shown below. The leftmost bit is the first bit of the sequence to be exclusive-ORed with the first bit of the codeblock, codeword, or Transfer Frame; the second bit of the sequence is exclusive-ORed with the second bit of the codeblock, codeword, or Transfer Frame, and so on.~~

~~1111 1111 0100 1000 0000 1110 1100 0000 1001 1010 . . . .~~

NOTES

1 The initial seed for the 131071-bit pseudo-randomizer has been optimized for performance and to avoid a long sequence of ones at the beginning of the generated sequence.

2 The first 40 bits of the pseudo-random sequences of both polynomials are shown below. The leftmost bit is the first bit of the sequence to be exclusive-ORed with the first bit of the codeblock, codeword, or Transfer Frame; the second bit of the sequence is exclusive-ORed with the second bit of the codeblock, codeword, or Transfer Frame, and so on.

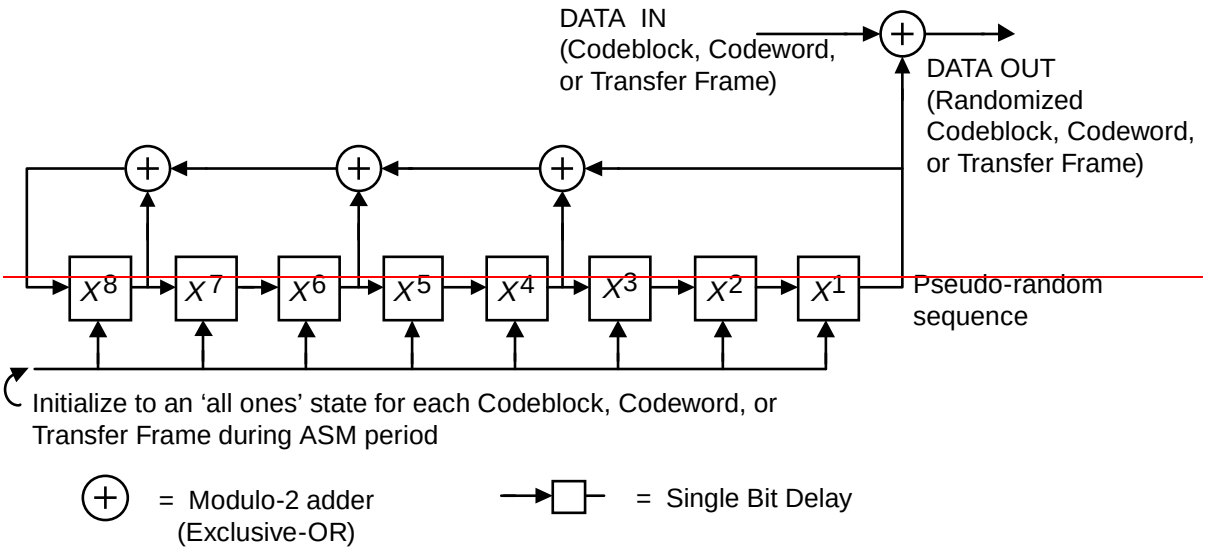
0001 1100 0111 0001 1011 1001 0001 1011 1010 1001. . . . (131071-bit randomizer)

1111 1111 0100 1000 0000 1110 1100 0000 1001 1010 . . . . (255-bit randomizer)

10.5 LOGIC DIAGRAM

~~NOTE – Figure 10-2 represents a possible generator for the specified sequence.~~

NOTE – Figure 10-2 and 10-3 represent two possible generators for the 131071-bit and 255-bit sequences, respectively.



~~**Figure 10-2: Pseudo-Randomizer Logic Diagram**~~

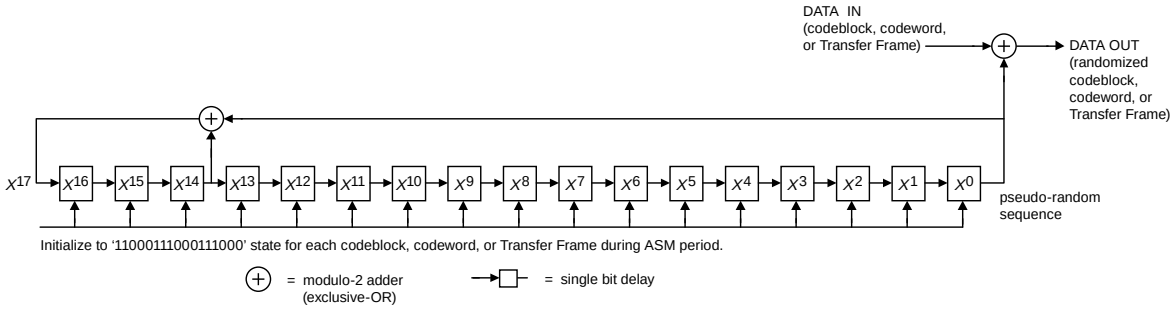


Figure 10-2: Pseudo-Randomizer Logic Diagram for the 131071-Bit Sequence

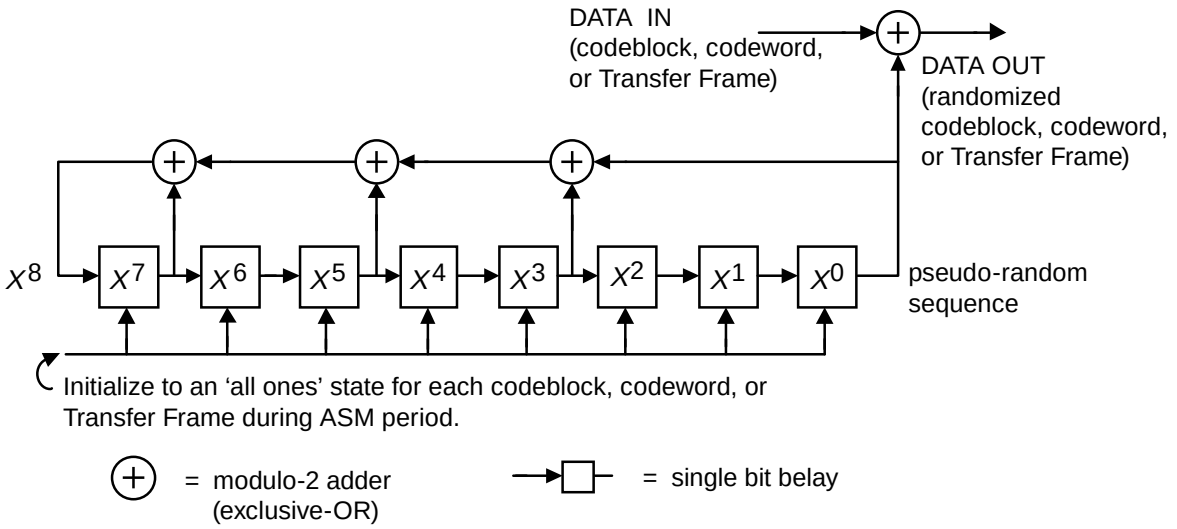


Figure 10-3: Pseudo-Randomizer Logic Diagram for the 255-Bit Sequence (Only for Backward Compatibility with Legacy Systems)

Table 12-1: Managed Parameters for Selected Options

Managed Parameter	Allowed Values
Randomizer	~~Present/~~Long (131071 bits) Short (255 bits) Absent
Coding Method	None Convolutional Reed-Solomon Concatenated Code Turbo LDPC coding (of a Transfer Frame) LDPC coding (of a stream of SMTFs)

12.4 MANAGED PARAMETERS FOR CONVOLUTIONAL CODE

The managed parameters for convolutional code shall be those specified in table 12-2.

Table 12-2: Managed Parameters for Convolutional Code

Managed Parameter	Allowed Values
Code Rate (*r*)	1/2, 2/3, 3/4, 5/6, 7/8

12.5 MANAGED PARAMETERS FOR REED-SOLOMON CODE

The managed parameters for Reed-Solomon code shall be those specified in table 12-3.

Table 12-3: Managed Parameters for Reed-Solomon Code

Managed Parameter	Allowed Values
Error Correction Capability (*E*, symbols)	8, 16
Interleaving Depth (*I*)	1, 2, 3, 4, 5, 8
Virtual Fill Length (*Q*, symbols)	Integer