

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-B-5.1

PINK BOOK

August 2024

The Consultative Committee for Space Data Systems

Recommendation for Space Data System Standards

TM SYNCHRONIZATION AND CHANNEL CODING

RECOMMENDED STANDARD

CCSDS 131.0-B-6

BLUE BOOK

June 2024

AUTHORITY

Issue:	Recommended Standard, Issue 5
Date:	September 2023
Location:	Washington, DC, USA

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and represents the consensus technical agreement of the participating CCSDS Member Agencies. The procedure for review and authorization of CCSDS documents is detailed in *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-4), and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the email address below.

This document is published and maintained by:

CCSDS Secretariat
National Aeronautics and Space Administration
Washington, DC, USA
Email: secretariat@mailman.ccsds.org

STATEMENT OF INTENT

The Consultative Committee for Space Data Systems (CCSDS) is an organization officially established by the management of its members. The Committee meets periodically to address data systems problems that are common to all participants, and to formulate sound technical solutions to these problems. Inasmuch as participation in the CCSDS is completely voluntary, the results of Committee actions are termed **Recommended Standards** and are not considered binding on any Agency.

This **Recommended Standard** is issued by, and represents the consensus of, the CCSDS members. Endorsement of this **Recommendation** is entirely voluntary. Endorsement, however, indicates the following understandings:

- Whenever a member establishes a CCSDS-related **standard**, this **standard** will be in accord with the relevant **Recommended Standard**. Establishing such a **standard** does not preclude other provisions which a member may develop.
- Whenever a member establishes a CCSDS-related **standard**, that member will provide other CCSDS members with the following information:
  - -- The **standard** itself.
  - -- The anticipated date of initial operational capability.
  - -- The anticipated duration of operational service.
- Specific service arrangements shall be made via memoranda of agreement. Neither this **Recommended Standard** nor any ensuing **standard** is a substitute for a memorandum of agreement.

No later than five years from its date of issuance, this **Recommended Standard** will be reviewed by the CCSDS to determine whether it should: (1) remain in effect without change; (2) be changed to reflect the impact of new technologies, new requirements, or new directions; or (3) be retired or canceled.

In those instances when a new version of a **Recommended Standard** is issued, existing CCSDS-related member standards and implementations are not negated or deemed to be non-CCSDS compatible. It is the responsibility of each member to determine when such standards or implementations are to be modified. Each member is, however, strongly encouraged to direct planning for its new standards and implementations towards the later version of the Recommended Standard.

FOREWORD

This document is a technical Recommended Standard for use in developing synchronization and channel coding systems and has been prepared by the Consultative Committee for Space Data Systems (CCSDS). The synchronization and channel coding concept described herein is intended for missions that are cross-supported between Agencies of the CCSDS.

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This Recommended Standard is therefore subject to CCSDS document management and change control procedures, which are defined in the *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-4). Current versions of CCSDS documents are maintained at the CCSDS Web site:

http://www.ccsds.org/

Questions relating to the contents or status of this document should be sent to the CCSDS Secretariat at the email address indicated on page i.

At time of publication, the active Member and Observer Agencies of the CCSDS were:

Member Agencies

- Agenzia Spaziale Italiana (ASI)/Italy.
- Canadian Space Agency (CSA)/Canada.
- Centre National d'Etudes Spatiales (CNES)/France.
- China National Space Administration (CNSA)/People's Republic of China.
- Deutsches Zentrum für Luft- und Raumfahrt (DLR)/Germany.
- European Space Agency (ESA)/Europe.
- Federal Space Agency (FSA)/Russian Federation.
- Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil.
- Japan Aerospace Exploration Agency (JAXA)/Japan.
- National Aeronautics and Space Administration (NASA)/USA.
- UK Space Agency/United Kingdom.

Observer Agencies

- Austrian Space Agency (ASA)/Austria.
- Belgian Science Policy Office (BELSPO)/Belgium.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- China Satellite Launch and Tracking Control General, Beijing Institute of Tracking and Telecommunications Technology (CLTC/BITTT)/China.
- Chinese Academy of Sciences (CAS)/China.
- China Academy of Space Technology (CAST)/China.
- Commonwealth Scientific and Industrial Research Organization (CSIRO)/Australia.
- Danish National Space Center (DNSC)/Denmark.
- Departamento de Ciência e Tecnologia Aeroespacial (DCTA)/Brazil.
- Electronics and Telecommunications Research Institute (ETRI)/Korea.
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Geo-Informatics and Space Technology Development Agency (GISTDA)/Thailand.
- Hellenic National Space Committee (HNSC)/Greece.
- Hellenic Space Agency (HSA)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Institute of Space Research (IKI)/Russian Federation.
- Korea Aerospace Research Institute (KARI)/Korea.
- Ministry of Communications (MOC)/Israel.
- Mohammed Bin Rashid Space Centre (MBRSC)/United Arab Emirates.
- National Institute of Information and Communications Technology (NICT)/Japan.
- National Oceanic and Atmospheric Administration (NOAA)/USA.
- National Space Agency of the Republic of Kazakhstan (NSARK)/Kazakhstan.
- National Space Organization (NSPO)/Chinese Taipei.
- Naval Center for Space Technology (NCST)/USA.
- Netherlands Space Office (NSO)/The Netherlands.
- Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- Scientific and Technological Research Council of Turkey (TUBITAK)/Turkey.
- South African National Space Agency (SANSA)/Republic of South Africa.
- Space and Upper Atmosphere Research Commission (SUPARCO)/Pakistan.
- Swedish Space Corporation (SSC)/Sweden.
- Swiss Space Office (SSO)/Switzerland.
- United States Geological Survey (USGS)/USA.

DOCUMENT CONTROL

Document	Title	Date	Status
CCSDS 131.0-B-1	TM Synchronization and Channel Coding, Issue 1	September 2003	Original issue, superseded
CCSDS 131.0-B-2	TM Synchronization and Channel Coding, Recommended Standard, Issue 2	August 2011	Issue 2, superseded
CCSDS 131.0-B-3	TM Synchronization and Channel Coding, Recommended Standard, Issue 3	September 2017	Issue 3, superseded
CCSDS 131.0-B-4	TM Synchronization and Channel Coding, Recommended Standard, Issue 4	April 2022	Issue 4, superseded
CCSDS 131.0-B-5	TM Synchronization and Channel Coding, Recommended Standard, Issue 5	September 2023	Issue 5, superseded
CCSDS 131.0-B-6	TM Synchronization and Channel Coding, Draft	August 2025	Adds turbo code channel interleaver, and moves slicing to its own section, applicable to all block codes

NOTE – Substantive changes from the previous issue are marked with change bars in the inside margin.

CONTENTS

Section Page

1 INTRODUCTION 1-1
1.1 PURPOSE 1-1
1.2 SCOPE 1-1
1.3 APPLICABILITY 1-1
1.4 RATIONALE 1-2
1.5 DOCUMENT STRUCTURE 1-2
1.6 CONVENTIONS AND DEFINITIONS 1-3
1.7 PATENTED TECHNOLOGIES 1-7
1.8 REFERENCES 1-8
2 OVERVIEW 2-1
2.1 ARCHITECTURE 2-1
2.2 SUMMARY OF FUNCTIONS 2-1
2.3 INTERNAL ORGANIZATION OF SUBLAYER 2-5
3 TRANSFER FRAME SYNCHRONIZATION, SLICING, AND VALIDATION ... 3-8
3.1 OVERVIEW 3-8
3.2 SLICING 3-9
3.3 SYNCHRONIZATION WITH SLICING 3-9
3.4 SYNCHRONIZATION WITHOUT SLICING 3-10
3.5 TRANSFER FRAME VALIDATION 3-11
4 CONVOLUTIONAL CODING 4-1
4.1 OVERVIEW 4-1
4.2 GENERAL 4-1
4.3 BASIC CONVOLUTIONAL CODE SPECIFICATION 4-2
4.4 PUNCTURED CONVOLUTIONAL CODES 4-4
5 REED-SOLOMON CODING 5-1
5.1 OVERVIEW 5-1
5.2 GENERAL 5-1
5.3 SPECIFICATION 5-1
5.4 DISCUSSION 5-6
6 CONCATENATED CODING 6-1
7 TURBO CODING 7-1
7.1 OVERVIEW 7-1
7.2 GENERAL 7-1
7.3 SPECIFICATION 7-1
8 LOW-DENSITY PARITY-CHECK CODING 8-11
8.1 OVERVIEW 8-11
8.2 GENERAL 8-11
8.3 LOW-DENSITY PARITY-CHECK CODE WITH RATE 223/255 8-12
8.4 LOW-DENSITY PARITY-CHECK CODE FAMILY WITH RATES 1/2, 2/3, AND 4/5 8-17
9 CODEWORD SYNCHRONIZATION 9-1
9.1 OVERVIEW 9-1

9.2 THE CODE SYNCHRONIZATION MARKER 9-2
9.3 CSM BIT PATTERNS 9-2
9.4 LOCATION OF CSM 9-4
9.5 RELATIONSHIP OF CSM TO REED-SOLOMON, TURBO, AND LDPC CODEBLOCKS AND CODEWORDS 9-4
10 PSEUDO-RANDOMIZER 10-6
10.1 OVERVIEW 10-6
10.2 PSEUDO-RANDOMIZER DESCRIPTION 10-6
10.3 SYNCHRONIZATION AND APPLICATION OF PSEUDO-RANDOMIZER 10-7
10.4 SEQUENCE SPECIFICATION 10-7
10.5 LOGIC DIAGRAM 10-8
11 TRANSFER FRAME LENGTHS 11-1
11.1 OVERVIEW 11-1
11.2 GENERAL 11-1
11.3 CASE 1: UNCODED 11-1
11.4 CASE 2: CONVOLUTIONAL ONLY 11-1
11.5 CASE 3: WHEN TRANSFER FRAME SLICING IS USED 11-2
11.6 CASE 4: REED-SOLOMON ONLY, WITHOUT TRANSFER FRAME SLICING 11-2
11.7 CASE 5: CONCATENATED, WITHOUT TRANSFER FRAME SLICING 11-3
11.8 CASE 6: TURBO, WITHOUT TRANSFER FRAME SLICING 11-3
11.9 CASE 7: LDPC, WITHOUT TRANSFER FRAME SLICING 11-3
12 MANAGED PARAMETERS 12-1
12.1 OVERVIEW 12-1
12.2 GENERAL 12-1
12.3 MANAGED PARAMETERS FOR SELECTED OPTIONS 12-1
12.4 MANAGED PARAMETERS FOR CONVOLUTIONAL CODE 12-2
12.5 MANAGED PARAMETERS FOR REED-SOLOMON CODE 12-2
12.6 MANAGED PARAMETERS FOR TURBO CODE 12-3
12.7 MANAGED PARAMETERS FOR LOW-DENSITY PARITY-CHECK CODING 12-3
13 USE OF TELECOMMUNICATIONS CHANNEL CODES FOR GROUND-TO-SPACE AND SPACE-TO-SPACE LINKS 13-1
13.1 OVERVIEW 13-1
13.2 TURBO CODES 13-1
13.3 LOW-DENSITY PARITY-CHECK CODES 13-2

ANNEX A SERVICE (NORMATIVE) A-1
ANNEX B SECURITY, SANA, AND PATENT CONSIDERATIONS (INFORMATIVE) B-1
ANNEX C ANNEX TO SUBSECTION 7.3, LOW-DENSITY PARITY-CHECK CODE WITH RATE 223/255 (INFORMATIVE) C-1
ANNEX D ABBREVIATIONS AND ACRONYMS (INFORMATIVE) D-1
ANNEX E INFORMATIVE REFERENCES (INFORMATIVE) E-2
ANNEX F TRANSFORMATION BETWEEN BERLEKAMP AND CONVENTIONAL REPRESENTATIONS (INFORMATIVE) F-1

ANNEX G EXPANSION OF REED-SOLOMON COEFFICIENTS (INFORMATIVE).....**G-1**

CONTENTS (continued)

Figure Page

1-1 Bit Numbering Convention 1-7
2-1 Relationship with OSI Layers 2-1
2-2 Internal Organization of the Sublayer at the Sending End 2-6
2-3 Internal Organization of the Sublayer at the Receiving End 2-7
3-1 Transfer Frames Sliced to Form LDPC Codeblocks 3-8
3-2 Transfer Frames Sliced to Form LDPC Codeblocks 3-9
3-3 Transfer Frames Sliced to Form LDPC Codeblocks 3-11
4-1 Basic Convolutional Encoder Block Diagram 4-3
4-2 Punctured Encoder Block Diagram 4-4
5-1 Reed-Solomon Codeblock Partitioning 5-5
5-2 Functional Representation of R-S Interleaving 5-7
7-1 Interpretation of Permutation 7-4
7-2 Turbo Encoder Block Diagram 7-4
7-3 Turbo Codewords for Different Code Rates 7-7
7-4 Turbo Codeword with Attached Sync Marker 7-8
8-1 Base Parity Check Matrix of the Basic (8176,7156) LDPC Code 8-13
8-2 Systematic Circulant Generator Matrix 8-16
8-3 Shortened Codeword 8-17
9-1 ASM Bit Pattern for Uncoded, Convolutional, Reed-Solomon, Concatenated, Rate 7/8 LDPC (Applied to a Transfer Frame), and all LDPC (Applied to a Stream of SMTFs) Coded Data 9-3
9-2 ASM Bit Pattern for Rate 1/2 Turbo and Rates 1/2, 2/3, and 4/5 LDPC (Applied to a Transfer Frame) Coded Data 9-3
9-3 ASM Bit Pattern for Rate 1/3 Turbo Coded Data 9-3
9-4 ASM Bit Pattern for Rate 1/4 Turbo Coded Data 9-4
9-5 ASM Bit Pattern for Rate 1/6 Turbo Coded Data 9-4
10-1 Pseudo-Randomizer Configuration 10-7
10-2 Pseudo-Randomizer Logic Diagram for the 131071-Bit Sequence 10-8
10-3 Pseudo-Randomizer Logic Diagram for the 255-Bit Sequence (Only for Backward Compatibility with Legacy Systems) 10-9
F-1 Transformational Equivalence F-2

Table

4-1 PUNCTURE CODE PATTERNS FOR CONVOLUTIONAL CODE RATES 4-4
7-1 SPECIFIED INFORMATION BLOCK LENGTHS 7-2
7-2 CODEWORD LENGTHS FOR SUPPORTED CODE RATES (MEASURED IN BITS) 7-2
7-3 PARAMETERS K_1 AND K_2 FOR SPECIFIED INFORMATION BLOCK LENGTHS.....7-3
8-1 SPECIFICATION OF CIRCULANTS 8-14
8-2 VALUES OF SUBMATRIX SIZE M FOR SUPPORTED CODES.....8-17
8-3 DESCRIPTION OF $\Phi_K(0,M)$ AND $\Phi_K(1,M)$.....8-19
8-4 DESCRIPTION OF $\Phi_K(2,M)$ AND $\Phi_K(3,M)$.....8-20
8-5 CODEWORD LENGTHS FOR SUPPORTED CODE RATES (MEASURED IN BITS) 8-21
12-1 MANAGED PARAMETERS FOR SELECTED OPTIONS.....12-2
12-2 MANAGED PARAMETERS FOR CONVOLUTIONAL CODE 12-2
12-3 MANAGED PARAMETERS FOR REED-SOLOMON CODE 12-2
12-4 MANAGED PARAMETERS FOR TURBO CODE 12-3
12-5 MANAGED PARAMETERS FOR LOW-DENSITY PARITY-CHECK CODE APPLIED TO A TRANSFER FRAME 12-3
C-1 TABLE OF CIRCULANTS FOR THE GENERATOR MATRIX.....C-1
F-1 EQUIVALENCE OF REPRESENTATIONS F-5

1 INTRODUCTION

1.1 PURPOSE

The purpose of this Recommended Standard is to specify synchronization and channel coding schemes used with the TM Space Data Link Protocol (reference [1]), the Advanced Orbiting Systems (AOS) Space Data Link Protocol (reference [2]), or the Unified Space Data Link Protocol (USLP) (reference [6]). These schemes are to be used over space-to-ground, space-to-space, or ground-to-space communications links by space missions over the RF Physical Layer defined in reference [5].

1.2 SCOPE

This Recommended Standard defines synchronization and channel coding schemes in terms of:

a) the services provided to the users of this specification;

b) data formats; and

c) the procedures performed to generate and process the data formats.

It does not specify:

a) individual implementations or products;

b) the methods or technologies required to perform the procedures; or

c) the management activities required to configure and control the system.

1.3 APPLICABILITY

This Recommended Standard applies to the creation of Agency standards and to the future data communications over space links between CCSDS Agencies in cross-support situations. This Recommended Standard includes comprehensive specification of the data formats and procedures for inter-Agency cross support. It is neither a specification of, nor a design for, real systems that may be implemented for existing or future missions.

The Recommended Standard specified in this document is to be invoked through the normal standards programs of each CCSDS Agency and is applicable to those missions for which cross-support based on capabilities described in this Recommended Standard is anticipated. Where mandatory capabilities are clearly indicated in sections of this Recommended Standard, they must be implemented when this document is used as a basis for cross support. Where options are allowed or implied, implementation of these options is subject to specific bilateral cross support agreements between the Agencies involved.

1.4 RATIONALE

The CCSDS believes it is important to document the rationale underlying the recommendations chosen, so that future evaluations of proposed changes or improvements will not lose sight of previous decisions.

1.5 DOCUMENT STRUCTURE

This document is divided into thirteen numbered sections and seven annexes:

a) section 1 presents the purpose, scope, applicability and rationale of this Recommended Standard and lists the conventions, definitions, and references used throughout the document;

b) section 2 provides an overview of synchronization and channel coding;

c) section 3 describes an optional procedure for slicing Transfer Frames into Information Blocks of a different length;

d) section 4 specifies convolutional coding;

e) section 5 specifies Reed-Solomon coding;

f) section 6 specifies concatenated coding;

g) section 7 specifies Turbo coding;

h) section 8 specifies low-density parity-check coding;

i) section 9 specifies the codeword synchronization scheme;

j) section 10 specifies the Pseudo-Randomizer;

k) section 11 specifies the allowed lengths of Transfer Frames;

l) section 12 lists the managed parameters associated with synchronization and channel coding;

m) section 13 specifies use of these codes for ground-to-space and space-to-space links;

n) annex A defines the service provided to the users;

o) annex B discusses security issues related to TM Channel Coding;

p) annex C provides the generator matrix circulant table applicable to rate-223/255 LDPC coding (see 8.3);

q) annex D lists acronyms and terms used within this document;

r) annex E provides a list of informative references;

s) annex F provides information on transformation between the Berlekamp (dual basis) and Conventional representations;

t) annex G provides information on Reed-Solomon coefficients.

1.6 CONVENTIONS AND DEFINITIONS

1.6.1 DEFINITIONS

1.6.1.1 Definitions from the Open System Interconnection Basic Reference Model

This Recommended Standard makes use of a number of terms defined in reference [3]. The use of those terms in this Recommended Standard shall be understood in a generic sense; i.e., in the sense that those terms are generally applicable to any of a variety of technologies that provide for the exchange of information between real systems. Those terms are:

a) Data Link Layer;

b) Physical Layer;

c) service;

d) service data unit.

1.6.1.2 Definitions from OSI Service Definition Conventions

This Recommended Standard makes use of a number of terms defined in reference [4]. The use of those terms in this Recommended Standard shall be understood in a generic sense: i.e., in the sense that those terms are generally applicable to any of a variety of technologies that provide for the exchange of information between real systems. Those terms are:

a) indication;

b) primitive;

c) request;

d) service provider;

e) service user.

1.6.1.3 Definition of Terms Used in This Recommended Standard

For the purposes of this Recommended Standard, the following definitions apply. Many other terms that pertain to specific items are defined in the appropriate sections.

block encoding: A one-to-one transformation of sequences of length k of elements of a source alphabet to sequences of length n of elements of a code alphabet, $n>k$.

Channel Access Data Unit (CADU): The Code Synchronization Marker (CSM) and randomized codeword following block-encoding if used; or the CSM and randomized Transfer Frame if block coding is not used. CADUs are sent to the convolutional encoder (if used), or to the physical layer if convolutional coding is not used. See paragraph 9.1.2 for details.

circulant: In low-density parity-check coding, a square matrix with binary entries, where each row is a one-entry right cyclic shift of the preceding row.

code rate: The average ratio of the number of binary digits at the input of an encoder to the number of binary digits at its output.

codeblock: The aggregation of *one or more* codewords. In this document, the term codeblock is used only for R-S coding. An R-S codeblock is the aggregation of I codewords, where I is the interleaving depth. If I=1, the terms codeblock and codeword are used interchangeably.

coded symbol: The unit of output from an encoder. A block encoder generates a codeword composed of coded symbols, and a convolutional encoder generates a stream of coded symbols. A decoder takes noisy coded symbols as input.

codeword: In a block code, one of the sequences in the range of the one-to-one transformation (see **block encoding**). A codeword of an (n,k) block code is a sequence of n coded symbols which are produced by encoding a sequence of k information symbols.

concatenation: The use of two or more codes to process data sequentially with the output of one encoder used as the input of the next.

connection vector (backward): In Turbo coding, a vector used to specify the feedback to the shift registers in the encoder. For a shift register with s stages, a backward connection vector is an s-bit binary number. A bit equal to 'one' in position i (counted from the left) indicates that the output of the ith stage of the shift register is to be used in computing the feedback value, except for the leftmost bit which is ignored.

connection vector (forward): In convolutional and Turbo coding, a vector used to specify one of the parity checks to be computed by the shift register(s) in the encoder. For a shift register with s stages, a connection vector is an s-bit binary number. A bit equal to 'one' in position i (counted from the left) indicates that the output of the ith stage of the shift register is to be used in computing that parity check.

constraint length: In convolutional coding, the number of consecutive input bits that are needed to determine the value of the output symbols at any time.

convolutional code: As used in this document, a code in which a number of output symbols are produced for each input information bit. Each output symbol is a linear combination of the current input bit as well as some or all of the previous k-1 bits where k is the constraint length of the code.

differential encoding: a signaling technique used to resolve phase ambiguities with certain modulations, equivalent to NRZ-M signaling.

expurgated code: A subcode of lower rate obtained by deleting some of the codewords from another code. This may be accomplished by removing rows from a generator matrix or adding rows to a parity-check matrix.

GF(2^J): The Galois Field consisting of exactly '2^J' elements.

information block: The input to a block encoder.

inner code: In a concatenated coding system, the last encoding algorithm that is applied to the data stream. The data stream here consists of the codewords generated by the outer decoder.

Low-Density Parity-Check (LDPC) code: An error correcting code with codewords taken as the set of vectors in the nullspace of a sparse matrix, called a low-density parity-check matrix. In this document, modifications of such codes through puncturing, expurgation, etc., are also considered LDPC codes.

Mission Phase: a period of a mission during which specified communications characteristics are fixed. The transition between two consecutive mission phases may cause an interruption of the communications services.

modulating waveform: A way of representing data bits ('1' and '0') by a particular waveform.

Non-Return-to-Zero-Level (NRZ-L): A data format in which a data 'one' is represented by one of two levels, and a data 'zero' is represented by the other level.

Non-Return-to-Zero-Mark (NRZ-M): A data format in which a data 'one' is represented by a change in level and a data 'zero' is represented by no change in level.

outer code: In a concatenated coding system, the first encoding algorithm that is applied to the data stream.

Physical Channel: a stream of bits transferred over a space link in a single direction.

punctured code: As used in this document, a code obtained by deleting some of the parity symbols generated by the convolutional encoder before transmission. The bandwidth efficiency obtained by puncturing is increased compared to the original code, although the minimum weight (and therefore its error-correcting performance) will be less than that of the original code.

Reed-Solomon (R-S) symbol: A set of J bits that represents an element in GF(2^J), the code alphabet of a J-bit Reed-Solomon code.

space link: a communications link between a spacecraft and its associated ground system or between two spacecraft. A space link consists of one or more Physical Channels in one or both directions.

Synchronization and Channel Coding Sublayer: That sublayer of the Data Link Layer used by CCSDS space link protocols which uses a prescribed coding technique to reliably transfer Transfer Frames through the potentially noisy Physical Layer.

Synchronization-Marked Transfer Frame (SMTF): A transfer frame and its preceding ASM.

systematic code: A code in which the input information sequence appears in unaltered form as part of the output codeword.

transparent code: A code that has the property that complementing the input of the encoder or decoder results in complementing the output.

trellis termination: The operation of filling with zeros the s stages of each shift register used in the Turbo encoder, after the end of the information block. During trellis termination the encoders continue to output encoded symbols for s-1 additional clock cycles.

Turbo code permutation: A fixed bit-by-bit permutation of the entire input block of information bits performed by an interleaver, used in Turbo codes.

Turbo code: As used in this document, a block code formed by combining two component recursive convolutional codes. A Turbo code takes as input a block of k information bits. The input block is sent unchanged to the first component code and bit-wise interleaved (see **Turbo code permutation**) to the second component code. The output is formed by the parity symbols contributed by each component code plus a replica of the information bits.

virtual fill: In a systematic block code, a codeword can be divided into an information part and a parity (check) part. Suppose that the information part is k symbols long (a symbol is defined here to be an element of the code's alphabet) and that the parity part is n symbols long. A 'shortened' code is created by taking only S ($S < k$) information symbols as input, appending a fixed string of length k-S and then encoding in the normal way. This fixed string is called 'fill'. Since the fill is a predetermined sequence of symbols, it need not be transmitted over the channel. Instead, the decoder appends the same fill sequence before decoding. In this case, the fill is called 'virtual fill'.

1.6.2 NOMENCLATURE

The following conventions apply for the normative specifications in this Recommended Standard:

a) the words 'shall' and 'must' imply a binding and verifiable specification;

b) the word 'should' implies an optional, but desirable, specification;

c) the word ‘may’ implies an optional specification;

d) the words ‘is’, ‘are’, and ‘will’ imply statements of fact.

NOTE – These conventions do not imply constraints on diction in text that is clearly informative in nature.

1.6.3 CONVENTIONS

In this document, the following convention is used to identify each bit in an *N*-bit field. The first bit in the field to be transmitted (i.e., the most left justified when drawing a figure) is defined to be ‘Bit 0’, the following bit is defined to be ‘Bit 1’, and so on up to ‘Bit *N*-1’. When the field is used to express a binary value (such as a counter), the Most Significant Bit (MSB) shall be the first transmitted bit of the field, i.e., ‘Bit 0’ (see figure 1-1).

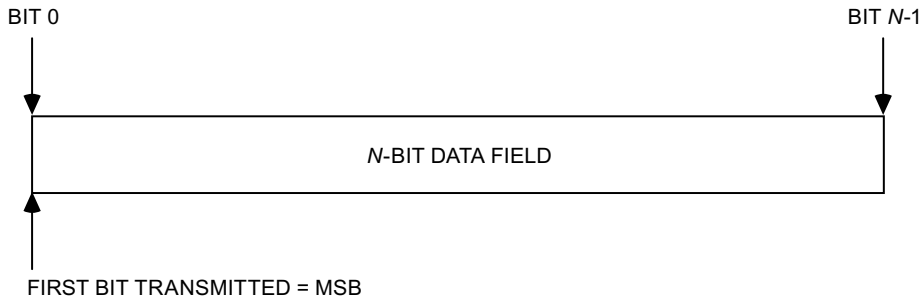


Figure 1-1: Bit Numbering Convention

In accordance with standard data-communications practice, data fields are often grouped into 8-bit ‘words’ which conform to the above convention. Throughout this Recommended Standard, such an 8-bit word is called an ‘octet’.

The numbering for octets within a data structure starts with ‘0’.

1.7 PATENTED TECHNOLOGIES

The CCSDS draws attention to the fact that it is claimed that compliance with this document may involve the use of patents concerning Turbo Coding (section 7) and LDPC Coding (section 8).

The CCSDS takes no position concerning the evidence, validity, and scope of these patent rights.

The holders of these patent rights have assured the CCSDS that they are willing to negotiate licenses under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of these patent rights are registered with CCSDS. Information can be obtained from the CCSDS Secretariat at the

address indicated on page i. Contact information for the holders of these patent rights is provided in annex B.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. The CCSDS shall not be held responsible for identifying any or all such patent rights.

1.8 REFERENCES

The following documents contain provisions which, through reference in this text, constitute provisions of this Recommended Standard. At the time of publication, the editions indicated were valid. All documents are subject to revision, and users of this Recommended Standard are encouraged to investigate the possibility of applying the most recent editions of the documents indicated below. The CCSDS Secretariat maintains a register of currently valid CCSDS Recommended Standards.

[1] *TM Space Data Link Protocol*. Issue 3. Recommendation for Space Data System Standards (Blue Book), CCSDS 132.0-B-3. Washington, D.C.: CCSDS, October 2021.

[2] *AOS Space Data Link Protocol*. Issue 4. Recommendation for Space Data System Standards (Blue Book), CCSDS 732.0-B-4. Washington, D.C.: CCSDS, October 2021.

[3] *Information Technology—Open Systems Interconnection—Basic Reference Model: The Basic Model*. 2nd ed. International Standard, ISO/IEC 7498-1:1994. Geneva: ISO, 1994.

[4] *Information Technology—Open Systems Interconnection—Basic Reference Model—Conventions for the Definition of OSI Services*. International Standard, ISO/IEC 10731:1994. Geneva: ISO, 1994.

[5] *Radio Frequency and Modulation Systems—Part 1: Earth Stations and Spacecraft*. Issue 32. Recommendations for Space Data System Standards (Blue Book), CCSDS 401.0-B-32. Washington, D.C.: CCSDS, October 2021.

[6] *Unified Space Data Link Protocol*. Issue 3. Recommendation for Space Data System Standards (Blue Book), CCSDS 732.1-B-2. Washington, D.C.: CCSDS, October 2021.

NOTE – Informative references are listed in annex E.

2 OVERVIEW

2.1 ARCHITECTURE

Figure 2-1 illustrates the relationship of this Recommended Standard to the Open Systems Interconnection (OSI) reference model (reference [3]). Two sublayers of the Data Link Layer are defined for CCSDS space link protocols. The telemetry (TM), AOS, and USLP protocols specified in references [1], [2], and [6], respectively, correspond to the Data Link Protocol Sublayer and provide functions for transferring data using the protocol data unit called the Transfer Frame. The Radio Frequency and Modulation Systems specified in reference [5] corresponds to the Physical Layer. The Synchronization and Channel Coding Sublayer provides additional functions necessary for transferring Transfer Frames over a space link. These functions are error-control coding/decoding, Transfer Frame delimiting/synchronizing, and bit transition generation/removal.

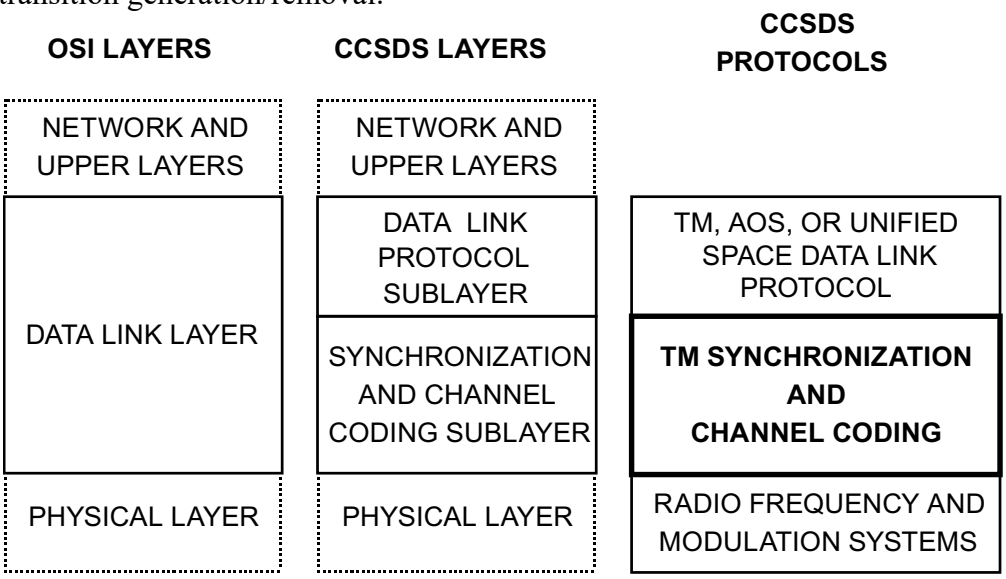


Figure 2-1: Relationship with OSI Layers

2.2 SUMMARY OF FUNCTIONS

2.2.1 GENERAL

The Synchronization and Channel Coding Sublayer provides the following four functions for transferring Transfer Frames over a space link:

a) error-control coding, including frame validation;

b) transfer frame slicing;

c) synchronization; and

d) pseudo-randomizing.

2.2.2 ERROR-CONTROL CODING

This Recommended Standard specifies the following five types of error-control coding:

a) convolutional coding (section 4);

b) R-S coding (section 5);

c) concatenated R-S and convolutional coding (section 6);

d) Turbo coding (section 7);

e) LDPC coding (section 8).

One of the convolutional codes described in section 4 alone may be satisfactory depending on performance requirements.

For Physical Channels, which are bandwidth-constrained and cannot tolerate the increase in bandwidth required by the basic convolutional code specified in 4.3, the punctured convolutional codes specified in 4.4 have the advantage of smaller bandwidth expansion.

Alternatively, the R-S codes and the high rate LDPC code specified in sections 5 and 8 also have the advantage of smaller bandwidth expansion and have the capability to indicate the presence of uncorrectable errors. Where a greater coding gain is needed than can be provided by a convolutional code or R-S code alone, a concatenation of a convolutional code as the inner code with a R-S code as the outer code may be used for improved performance (see section 6).

The Turbo codes specified in section 7 or the LDPC codes specified in section 8 may be used to obtain even greater coding gain where the environment permits.

AOS and USLP are symmetrical and may be used over space-to-ground, ground-to-space, or space-to-space (in both directions).

NOTE

1 In this Recommended Standard, the characteristics of the codes are specified only to the extent necessary to ensure interoperability and cross-support. The specification does not attempt to quantify the relative coding gain or the merits of each approach discussed, nor does it specify the design requirements for encoders or decoders.

2.2.3 FRAME VALIDATION

After decoding is performed, the upper layers at the receiving end also need to know whether or not each decoded Transfer Frame can be used as a valid data unit, i.e., an indication of the quality of the received frame is needed. This function is called Frame Validation. The R-S and LDPC decoders can determine, with a very high probability, whether or not they can correctly decode a codeword. Therefore, the R-S and LDPC codes are also used for Frame Validation.

When the R-S or LDPC codes are not used, the Frame Error Control Field (FECF) defined in references [1] or [2] is used for Frame Validation.

2.2.4 TRANSFER FRAME SLICING

Transfer frames may optionally be partitioned into Information Blocks of different size, both to eliminate constraints on the transfer frame length and to reduce the amount of overhead when long transfer frames are used.

As such, when a block code is used, this recommended standard specifies an optional method for slicing Transfer Frames into Information Blocks appropriate for the codeword message size.

When slicing is not used, each codeword contains exactly one transfer frame. When slicing is used, Transfer Frames and codewords may have different lengths, but this necessitates a second level of synchronization.

2.2.5 SYNCHRONIZATION

When slicing is used, the standard specifies a method for synchronizing Transfer Frames using an Attached Sync Marker (ASM) (see section 3). In this case, the standard also specifies a second method for synchronizing codewords using a CSM (see section 9).

When slicing is not used, each codeword contains exactly one Transfer Frame, and only one set of synchronization markers is needed. It is referred to interchangeably as the ASM or the CSM (see Figure 2-2).

The CSM may also be used for resolution of data ambiguity (sense of ‘1’ and ‘0’) if data ambiguity is not resolved by the modulation method used in the Physical Layer.

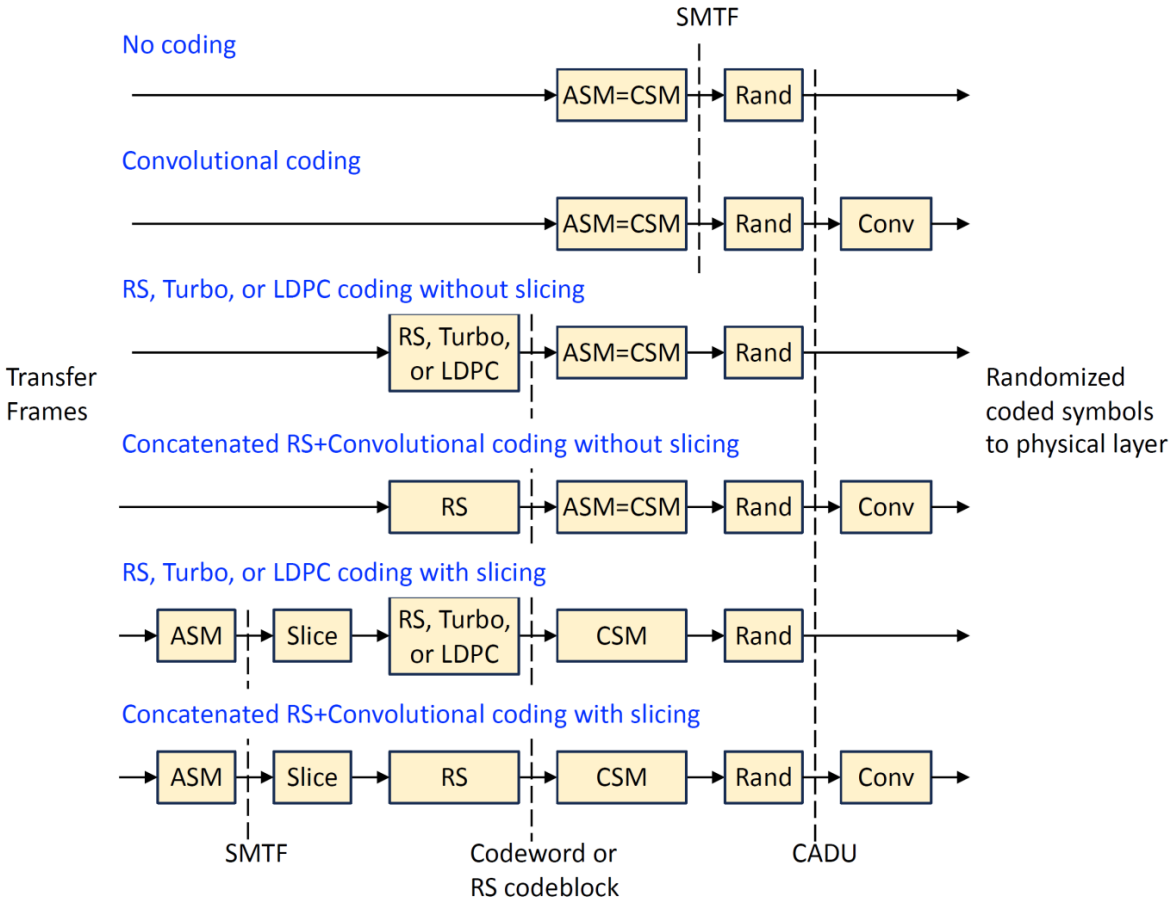


Figure 2-2: supported combinations of coding and slicing.

2.2.6 PSEUDO-RANDOMIZING

This Recommended Standard specifies a pseudo-randomizer to improve several aspects of the telemetry link that aid receiver acquisition, bit synchronization (see section 10), convolutional code synchronization, and proper Frame Validation (see 2.2.3).

2.3 INTERNAL ORGANIZATION OF SUBLAYER

2.3.1 SENDING END

Figure 2-3 shows the internal organization of the Synchronization and Channel Coding Sublayer of the sending end. This figure identifies functions performed by the sublayer[1] and shows logical relationships among these functions. The figure is not intended to imply any hardware or software configuration in a real system. Depending on the options used for a mission, not all of the functions may be present in the sublayer.

At the sending end, the Synchronization and Channel Coding Sublayer accepts Transfer Frames of fixed length from the Data Link Protocol Sublayer (see figure 2-1), performs functions selected for the mission, attaches CSMs, and delivers a continuous and contiguous stream of coded symbols to the Physical Layer.

[1] Figures 2-2 and 2-3 are limited to the contents of this Recommended Standard and do not cover, e.g., SCCC (reference [E5]) and DVB-S2 (reference [E6]).

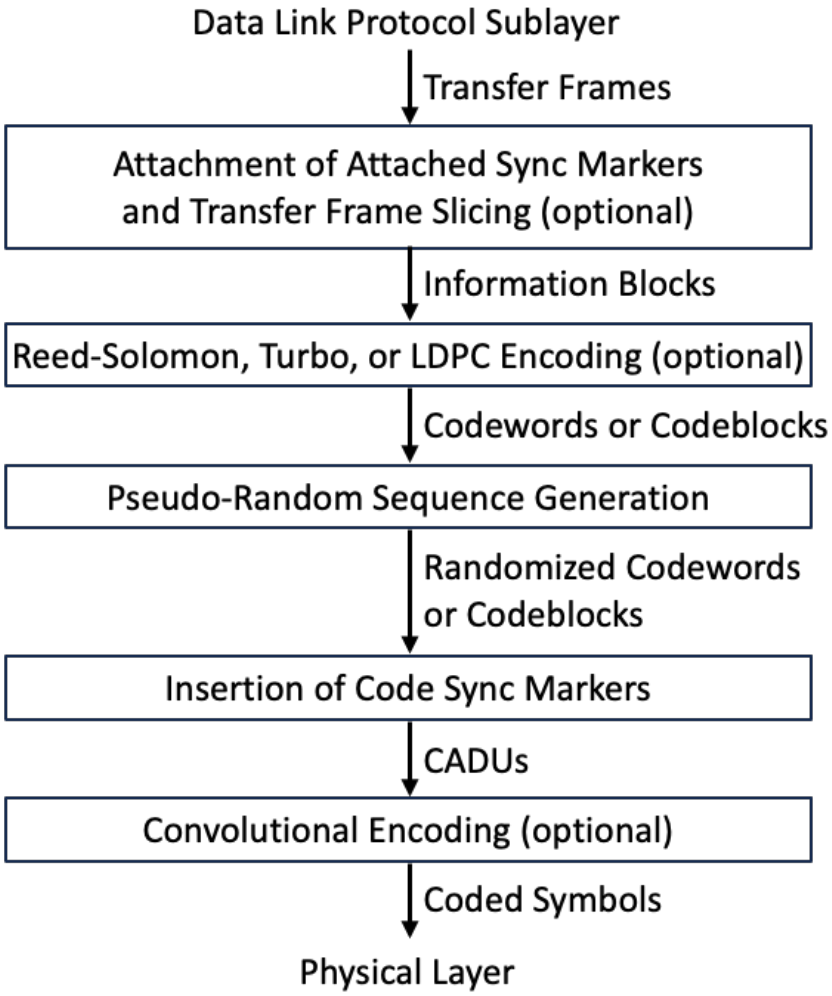


Figure 2-3: Internal Organization of the Sublayer at the Sending End

2.3.2 RECEIVING END

Figure 2-4 shows the internal organization of the Synchronization and Channel Coding Sublayer of the receiving end. This figure identifies functions performed by the sublayer and shows logical relationships among these functions. The figure is not intended to imply any hardware or software configuration in a real system (e.g., some implementations perform frame synchronization before convolutional decoding when convolutional code rate 1/2 is used). Depending on the options used for a mission, not all of the functions may be present in the sublayer.

At the receiving end, the Synchronization and Channel Coding Sublayer accepts a continuous and contiguous stream of coded symbols from the Physical Layer, performs functions selected for the mission, and delivers Transfer Frames to the Data Link Protocol Sublayer.

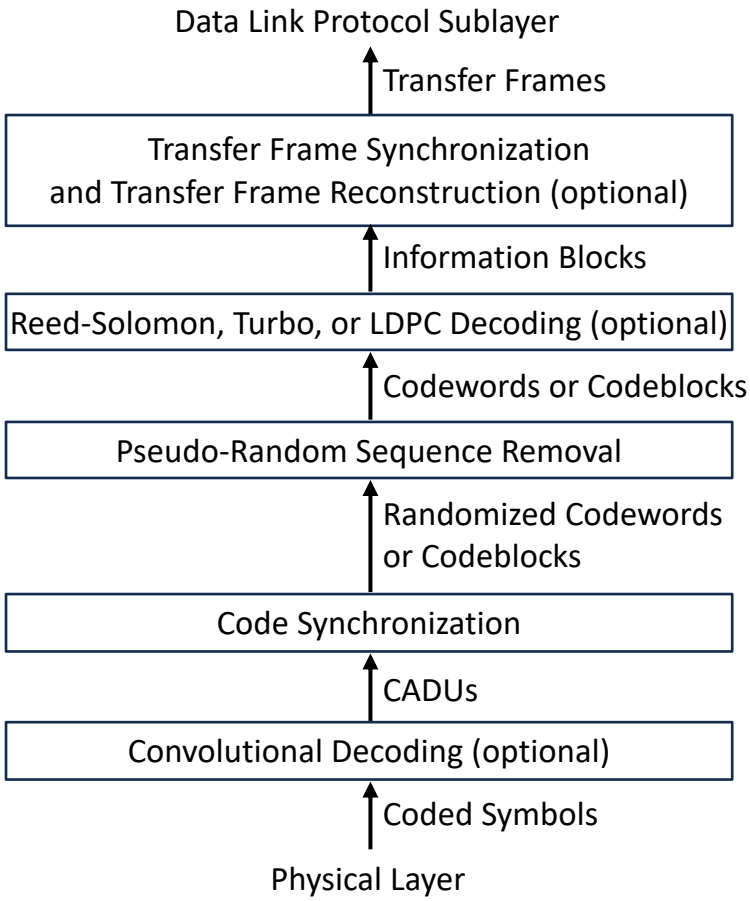


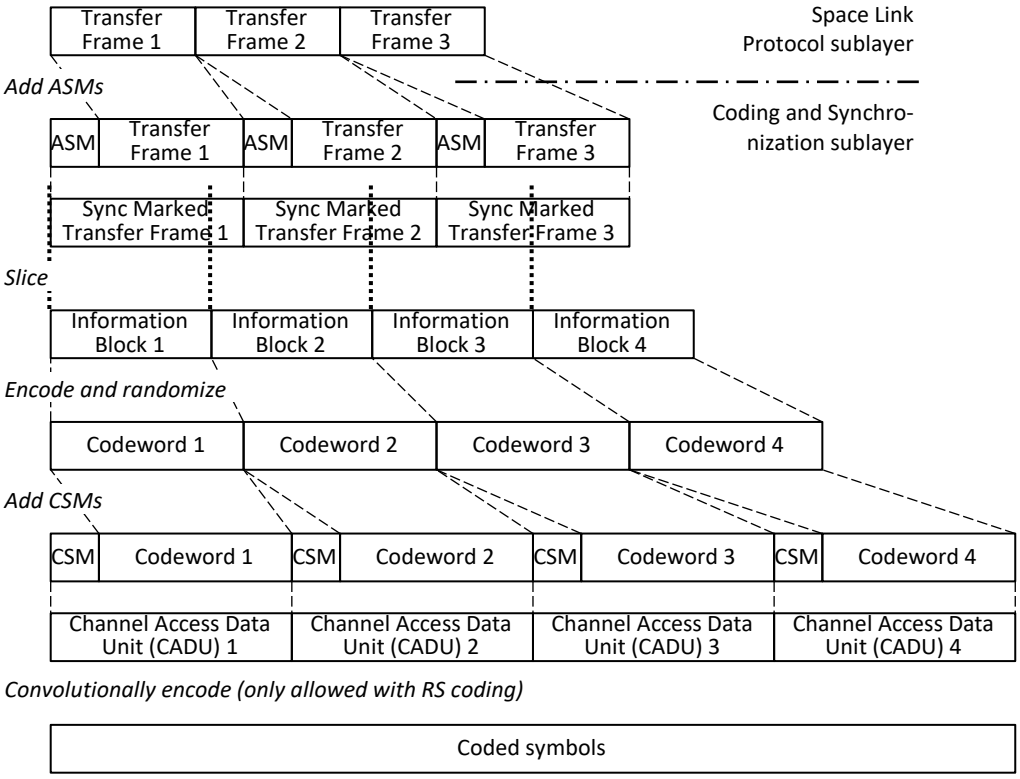
Figure 2-4: Internal Organization of the Sublayer at the Receiving End

3 TRANSFER FRAME SYNCHRONIZATION, SLICING, AND VALIDATION

3.1 OVERVIEW

Most error correcting codes in this specification have constraints on the length of the information blocks that they can encode. Sometimes it is straightforward to generate Transfer Frames that satisfy these length constraints, and sometimes this imposes undesirable constraints on the rest of a communications system design. To meet this need, a stream of Synchronization Marked Transfer Frames may be sliced into information blocks of a different length to equal the message size, *k*, of the block code. This slicing procedure is optional, and its use is managed and fixed for a given mission phase.

Transfer Frame slicing allows for the use of Transfer Frames that are longer than the maximum information block length, and may offer practical advantages in implementation, though it generally results in information block boundaries that do not coincide with Transfer Frame boundaries. Slicing is not used if there is no coding, or with convolutional coding (Section 4), because these alternatives do not constrain the information block length.



Note: When Reed-Solomon coding is used and *I* > 1, the encoder generates codeblocks rather than codewords.

Figure 3-1: Procedure at the sending end when slicing is used

3.2 SLICING

3.2.1 Transfer Frame slicing is optional with R-S, Turbo, or LDPC coding. Its use is managed and fixed for a given mission phase.

3.2.2 Transfer Frame slicing shall not be used with uncoded data, or if convolutional coding is used alone.

3.3 SYNCHRONIZATION WITH SLICING

3.3.1 When used, synchronization with slicing is performed at the sending end as follows, and as shown in Figure 3-1.

a) Each *M*-bit Transfer Frame is prepended with a 32-bit ASM, to yield a stream of *M*+32 bit SMTFs. The ASM bit pattern shall be as shown in Figure 3-2, and represented in hexadecimal as 1ACF FC1D.

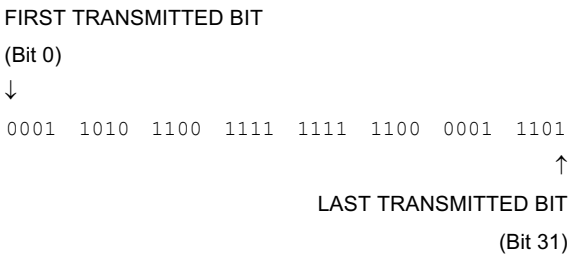


Figure 3-2: ASM Bit Pattern When Slicing is Used

NOTE – The ASM bit pattern matches the 32-bit CSM pattern specified in Section 9. They cannot be confused by the receiver, because the ASM is randomized.

b) The stream of SMTFs is divided into *k*-bit Information Blocks for encoding.

c) If Turbo or LDPC coding is used, then each *k*-bit Information Block is encoded into an *n*-bit codeword, as described in Section 7 or Section 8, respectively. If R-S coding is used, then sets of *I k*-bit Information Blocks are encoded and interleaved into an *nI*-bit codeblock, where *I* is the interleaving depth, as described in Section 5.

d) Each codeword or codeblock is randomized, as described in Section 10. Randomization is mandatory.

e) CSMs are prepended to each randomized codeword or codeblock, as described in Section 9.

3.3.2 When synchronization with slicing is used, the reverse process is followed at the receiving end, as follows.

a) Code synchronization is performed to identify the CSMs and extract the codewords or codeblocks.

b) The codewords or codeblocks are de-randomized.

c) Each codeword is decoded (or set of *I* interleaved codewords, in the case of R-S codes).

d) The decoded Information Blocks are concatenated to form a stream of SMTFs.

e) Transfer Frame synchronization is performed to identify the ASMs and extract the Transfer Frames.

3.4 SYNCHRONIZATION WITHOUT SLICING

3.4.1 When synchronization without slicing is used, then the sending end acts in a simpler way, as follows, and as shown in Figure 3-3.

a) Each Transfer Frame is an Information Block.

b) If Turbo or LDPC coding is used, then each *k*-bit Information Block is encoded into an *n*-bit codeword, as described in Section 7 or Section 8, respectively. If R-S coding is used, then sets of *I* *k*-bit Information Blocks are encoded and interleaved into an *nI*-bit codeblock, where *I* is the interleaving depth, as described in Section 5.

c) Each codeword or codeblock is randomized, as described in Section 10. Randomization is mandatory.

d) CSMs are prepended to each (randomized) codeword or codeblock, as described in Section 9.

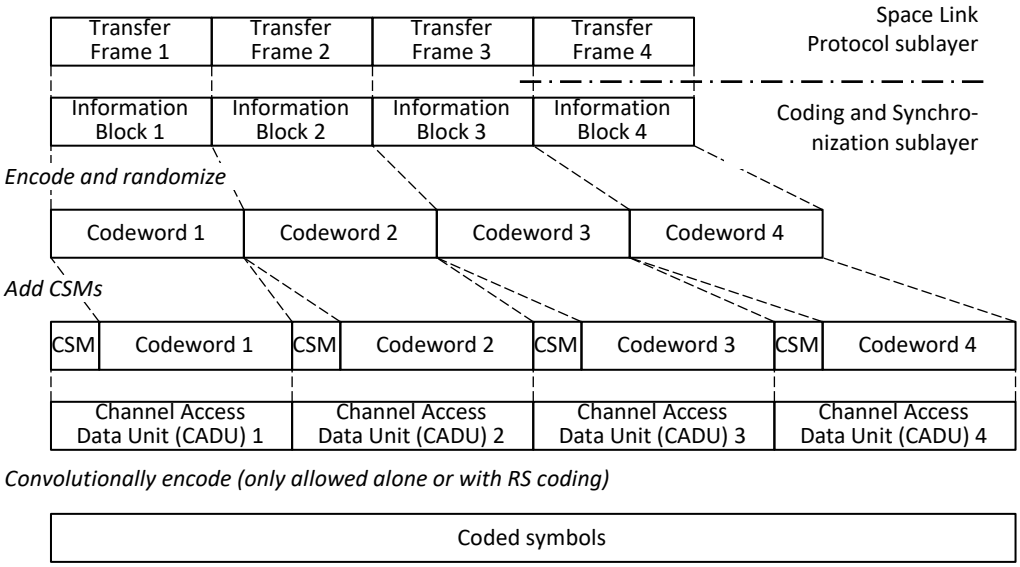
3.4.2 When synchronization without slicing is used, the reverse process at the receiving end is simplified, as follows.

a) Code synchronization is performed to identify the CSMs and extract the codewords or codeblocks. Because there is only one level of synchronization, CSMs also serve the function of ASMs, and the two terms may be used interchangeably.

b) The codewords or codeblocks are de-randomized.

c) Each codeword is decoded (or set of *I* interleaved codewords, in the case of R-S codes).

d) The decoded Information Blocks are Transfer Frames.



Note: When Reed-Solomon coding is used and *I* > 1, the encoder generates codeblocks rather than codewords.

Figure 3-3: Procedure at the sending end when slicing is not used

3.5 TRANSFER FRAME VALIDATION

3.5.1 With no coding, convolutional coding, or turbo coding, typical decoders are unable to detect decoding errors. When these codes are used with TM, AOS, or USLP Transfer Frames, the FECF specified in references [1], [2], or [6] is mandatory, and shall be used for Transfer Frame validation.

3.5.2 With R-S, Concatenated, or LDPC coding, the decoder may be used to validate Information Blocks.

a) When slicing is used, a Transfer Frame shall be marked invalid if any part of it is contained in an invalid Information Block.

b) When slicing is not used, Information Blocks and Transfer Frames are equivalent, and the decoder may directly validate Transfer Frames.

c) When these codes are used with TM, AOS, or USLP Transfer Frames, the FECF specified in references [1], [2], or [6] is optional. If present, the FECF may also be used for Transfer Frame validation.

4 CONVOLUTIONAL CODING

4.1 OVERVIEW

The basic convolutional code is a rate (r) 1/2, constraint-length (K) 7 transparent code which is well suited for channels with predominantly Gaussian noise. This code is defined in 4.3. When this code is punctured according to 4.4, higher code rates may be achieved although with lower error correcting performance.

Puncturing allows a single code rate of either 2/3, 3/4, 5/6, or 7/8 to be selected. The four different puncturing schemes allow selection of the most appropriate level of error correction and symbol rate for a given service or data rate.

4.2 GENERAL

4.2.1 CODE SYNCHRONIZATION MARKER

The CSM used with convolutional coding shall be the 32-bit pattern specified in 9.2, and it shall always be inserted before performing convolutional encoding.

4.2.2 DATA RANDOMIZATION

The pseudo-randomizer defined in section 10 shall be used.

NOTE – Some legacy systems do not include the randomizer. An inverter is specified with the basic convolutional code to assure sufficient bit transitions to keep receiver symbol synchronizers in lock, when used with Binary Phase Shift Keying (BPSK) modulation. However, sufficient bit transitions cannot be guaranteed by the inverter alone if some multiplexing schemes are used, e.g., with Quadrature Phase Shift Keying (QPSK) modulation, or if a punctured convolutional code is used. There are also data patterns for which convolutional code synchronization cannot be determined. The pseudo-randomizer is also used to aid signal acquisition and to mitigate spectral lines in the transmitted signal.

4.2.3 FRAME VALIDATION

When TM, AOS, or USLP Transfer Frames are used, the FECF specified in references [1], [2], or [6] shall be used to validate the Transfer Frame, unless the convolutional code is concatenated with an outer R-S code (see section 6).

NOTE – If the decoder's correction capability is exceeded, undetected bursts of errors may appear in the output.

4.2.4 QUANTIZATION

Soft bit decisions with at least three-bit quantization should be used whenever constraints (such as complexity of decoder) permit.

4.3 BASIC CONVOLUTIONAL CODE SPECIFICATION

4.3.1 The basic convolutional code shall be the non-systematic code with the following characteristics:

(1) Nomenclature: Convolutional code with maximum-likelihood decoding.

(2) Code rate (r): 1/2 bit per symbol.

(3) Constraint length (K): 7 bits.

(4) Connection vectors: G1 = 1111001 (171 octal); G2= 1011011(133 octal).

(5) Symbol inversion: On output path of G2.

NOTE – An encoder block diagram is shown in figure 4-1. When a single encoder is used, G2 inversion provides no benefit to data randomization when even-order modulations higher than BPSK are used. G2 inversion does provide value when coding is done after channel splitting and with separate encoders on each channel.

4.3.2 The output symbol sequence shall be: $C_1(1)$, $\overline{C_2(1)}$, $C_1(2)$, $\overline{C_2(2)}$. . . .

4.3.3 When suppressed-carrier modulation systems are used:

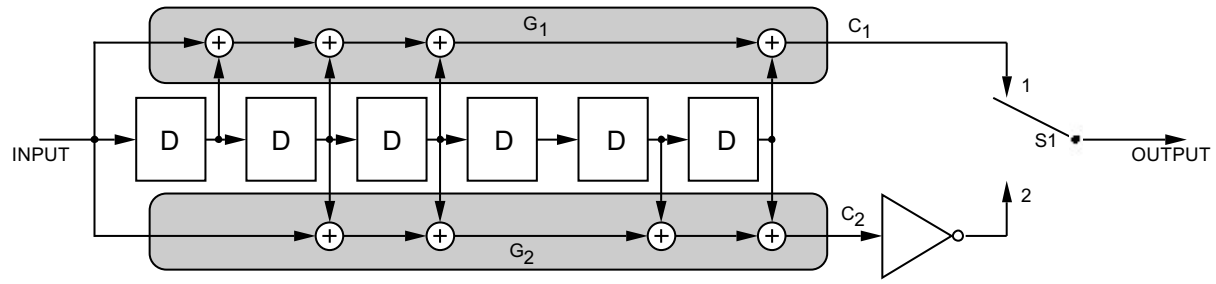
a) NRZ-M or NRZ-L may be used as a modulating waveform.

b) If the user contemplates differential encoding, i.e., conversion of his modulating waveform from NRZ-L to NRZ-M, such conversion should be performed at the input to the convolutional encoder.

NOTES

1 Since the convolutional codes are transparent, differential encoding can be used before the convolutional encoder to help phase ambiguity resolution and, correspondingly, the conversion at the receiving end from NRZ-M to NRZ-L should be performed at the output of the convolutional decoder. Differential encoding after the convolutional encoder is not advised because it introduces considerable loss of performance. It also would require differential detection, which is more complex with soft symbols.

2 When a fixed pattern (the fixed part of the convolutionally encoded CSM) in the symbol stream is used to provide node synchronization for the convolutional decoder, any modification of the pattern resulting from the modulating waveform conversion needs to be accounted for.



NOTES:

1. → D → = SINGLE BIT DELAY.
2. FOR EVERY INPUT BIT, TWO SYMBOLS ARE GENERATED BY COMPLETION OF A CYCLE FOR S1: POSITION 1, POSITION 2.
3. S1 IS IN THE POSITION SHOWN (1) FOR THE FIRST SYMBOL ASSOCIATED WITH AN INCOMING BIT.
4. ⊕ = MODULO-2 ADDER.
5. → ▷○ = INVERTER.

Figure 4-1: Basic Convolutional Encoder Block Diagram

4.4 PUNCTURED CONVOLUTIONAL CODES

4.4.1 The punctured convolutional code shall have the following characteristics:

(1) Nomenclature: Punctured convolutional code with maximum-likelihood decoding.

(2) Code rate (r): 1/2, punctured to 2/3, 3/4, 5/6 or 7/8.

(3) Constraint length (K): 7 bits.

(4) Connection vectors: G1 = 1111001 (171 octal); G2 = 1011011 (133 octal).

(5) Symbol inversion: None.

4.4.2 The puncturing patterns for each of the punctured convolutional code rates shall be as specified in table 4-1.

NOTE – Figure 4-2 depicts the punctured encoding scheme.

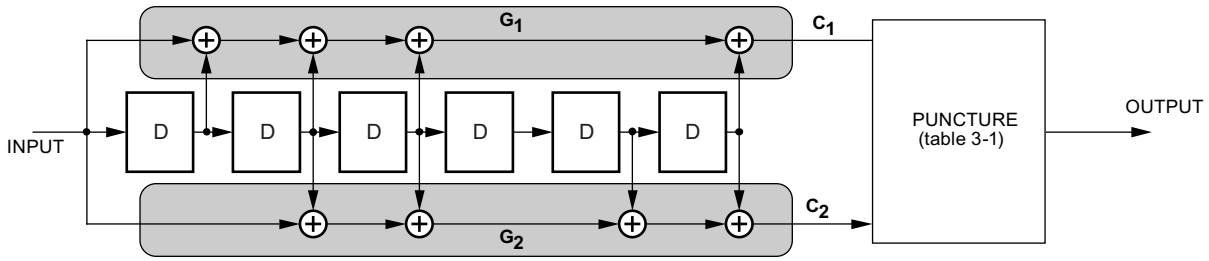


Figure 4-2: Punctured Encoder Block Diagram

Table 4-1: Puncture Code Patterns for Convolutional Code Rates

Puncturing Pattern 1 = transmitted symbol 0 = non-transmitted symbol	**Code Rate**	**Output Sequence** $C_1(t)$, $C_2(t)$ denote values at bit time t
C_1: 1 0 C_2: 1 1	2/3	$C_1(1)\ C_2(1)\ C_2(2)$ ...
C_1: 1 0 1 C_2: 1 1 0	3/4	$C_1(1)\ C_2(1)\ C_2(2)\ C_1(3)$ ...
C_1: 1 0 1 0 1 C_2: 1 1 0 1 0	5/6	$C_1(1)\ C_2(1)\ C_2(2)\ C_1(3)\ C_2(4)\ C_1(5)$ ...
C_1: 1 0 0 0 1 0 1 C_2: 1 1 1 1 0 1 0	7/8	$C_1(1)\ C_2(1)\ C_2(2)\ C_2(3)\ C_2(4)\ C_1(5)\ C_2(6)\ C_1(7)$ ...

5 REED-SOLOMON CODING

5.1 OVERVIEW

The R-S codes defined in this section are powerful burst error correcting codes. One of two different error-correcting options may be chosen. For maximum performance (at the expense of accompanying overhead) the *E*=16 option can correct 16 R-S symbols in error per codeword. For lower overhead (with reduced performance) the *E*=8 option can correct 8 R-S symbols per codeword. The R-S code may be used alone, and as such it provides an excellent forward error correction capability in a burst-noise channel. However, should the R-S code alone not provide sufficient coding gain, it may be concatenated with the convolutional code defined in section 4. Used this way, the R-S code is the *outer code*, while the convolutional code is the *inner code*.

5.2 GENERAL

5.2.1 DATA RANDOMIZATION

The pseudo-randomizer defined in section 10 shall be used.

NOTE – Some legacy systems do not include the randomizer. The recommended R-S codes, by themselves, cannot guarantee sufficient bit transitions to keep receiver symbol synchronizers in lock. Because of the quasi-cyclic nature of these codes, undetected decoding errors may result from incorrect codeblock synchronization. The pseudo-randomizer is also used to aid signal acquisition and to mitigate spectral lines in the transmitted signal.

5.2.2 FRAME VALIDATION

The FECF specified in references [1], [2], and [6] is optional. The system designer may choose to use it for additional Transfer Frame validation, particularly with the *E*=8 code.

NOTE – The R-S code with *E*=16 has an extremely low undetected error rate, and that with *E*=8 has an undetected error rate low enough for some applications [E7]. Therefore, the R-S decoder may be used alone to validate the Information Block, and consequently the contained Transfer Frame(s).

5.3 SPECIFICATION

5.3.1 PARAMETERS

The parameters of the selected R-S code are as follows:

a) *J* shall be 8 bits per R-S symbol.

b) *E* shall be 16 or 8 R-S symbols.

NOTE – E is the R-S error correction capability, in symbols, within a R-S codeword.

5.3.2 GENERAL CHARACTERISTICS

The code shall conform to the following general characteristics:

a) J, E, and I (the depth of interleaving) are independent parameters.

b) $n = 2^J - 1 = 255$ symbols per R-S codeword.

c) $2E$ is the number of R-S symbols among n symbols of an R-S codeword representing parity checks.

d) $k = n - 2E$ is the number of R-S symbols among n R-S symbols of an R-S codeword representing information.

5.3.3 FIELD GENERATOR POLYNOMIAL

The field generator polynomial shall be:

$$F(x) = x^8 + x^7 + x^2 + x + 1$$

over GF(2).

5.3.4 CODE GENERATOR POLYNOMIAL

The code generator polynomial shall be:

$$g(x) = \prod_{j=128-E}^{127+E} (x - \alpha^{11j}) \qquad = \sum_{i=0}^{2E} G_i x^i$$

over GF(2^8), where $F(\alpha) = 0$.

NOTES

1 It should be recognized that α^{11} is a primitive element in GF(2^8) and that $F(x)$ and $g(x)$ characterize a (255,223) R-S code when $E = 16$ and a (255,239) R-S code when $E = 8$.

2 The selected code is a systematic code. This results in a systematic codeblock.

5.3.5 SYMBOL INTERLEAVING

5.3.5.1 The allowable values of interleaving depth are I=1, 2, 3, 4, 5, and 8.

NOTE – I=1 is equivalent to the absence of interleaving.

5.3.5.2 The interleaving depth shall normally be fixed on a Physical Channel for a Mission Phase.

NOTE – Discussion of symbol interleaving is contained in 5.4.1.

5.3.6 MAXIMUM CODEBLOCK LENGTH

The maximum codeblock length, in R-S symbols, shall be determined by the following equation:

$$L_{\max} = nI = (2^J - 1)I = 255I$$

5.3.7 SHORTENED CODEBLOCK LENGTH

5.3.7.1 A shortened codeblock length may be used to accommodate frame lengths smaller than the maximum.

NOTE – However, since the R-S code is a block code, the decoder must always operate on a full block basis.

5.3.7.2 To achieve a full codeblock, 'virtual fill' shall be added to make up the difference between the shortened block and the maximum codeblock length.

NOTES

1 The characteristics and limitations of virtual fill are covered in 5.3.8.2.

2 Since the virtual fill is not transmitted, both encoder and decoder need to be set to insert it with the proper length for the encoding and decoding processes to be carried out properly.

5.3.7.3 When an encoder (initially cleared at the start of a block) receives $kI–Q$ symbols representing information (where Q, representing fill, is a multiple of I, and is less than kI), $2EI$ check symbols shall be computed over kI symbols, of which the leading Q symbols shall be treated as all-zero symbols.

NOTE – A ($nI–Q$, $kI–Q$) shortened codeblock results.

5.3.7.4 The leading Q symbols (all zeros) of the resulting shortened codeblock shall be neither entered into the encoder nor transmitted.

NOTE – Shortening the transmitted codeblock length in this way changes the overall performance to a degree dependent on the amount of virtual fill used. Since it incorporates no virtual fill, the maximum codeblock length allows full performance. In addition, as virtual fill in a codeblock is increased (at a specific bit rate), the number of codeblocks per unit time that the decoder must handle increases. Therefore, care should be taken so that the maximum operating speed of the decoder (codeblocks per unit time) is not exceeded.

5.3.8 REED-SOLOMON CODEBLOCK PARTITIONING AND VIRTUAL FILL

5.3.8.1 Parts of the partitioned R-S codeblock (see figure 5-1) are defined as follows:

a) The **Reed-Solomon Check Symbols** shall consist of the trailing $2EI$ symbols ($2EIJ$ bits) of the codeblock.

NOTES

1 As an example, when $E = 16$ and $k = 223$, for I=5 this is always 1280 bits.

2 If Transfer Frame slicing is used, the **Information Block** is constructed according to Section 3. Otherwise, the Information Block is the same as a Transfer Frame, defined by the TM Space Data Link Protocol (reference [1]), the AOS Space Data Link Protocol (reference [2]), or the Unified Space Data Link Protocol (reference [6]). For constraints on the length of the Transfer Frame, see section 11.

b) The **CSM** used with R-S code

1) shall be the 32-bit pattern specified in section 9;

2) shall precede the transmitted codeblock.

NOTE – Frame synchronizers should therefore be set to expect a marker at every transmitted codeblock + 32 bits.

c) The **Reed-Solomon codeblock** shall consist of the Information Block (without the 32-bit sync marker) and R-S check symbols.

NOTES

1 The R-S codeblock is the received data entity physically fed into the R-S decoder. (As an example, when $E = 16$ and $k = 223$, using I=5 and no virtual fill, the length of the transmitted codeblock will be 10,200 bits; if virtual fill is used, it will be incrementally shorter, depending on the amount used.)

2 The **logical codeblock** is the logical data entity operated upon by the R-S decoder. It can have a different length than the transmitted codeblock because it accounts for the amount of virtual fill that was introduced. (As an example, when $E = 16$ and k

= 223, for *I*=5 the logical codeblock always appears to have exactly 10,200 bits in length.)

3 The R-S codeblock is partitioned as shown in figure 5-1.

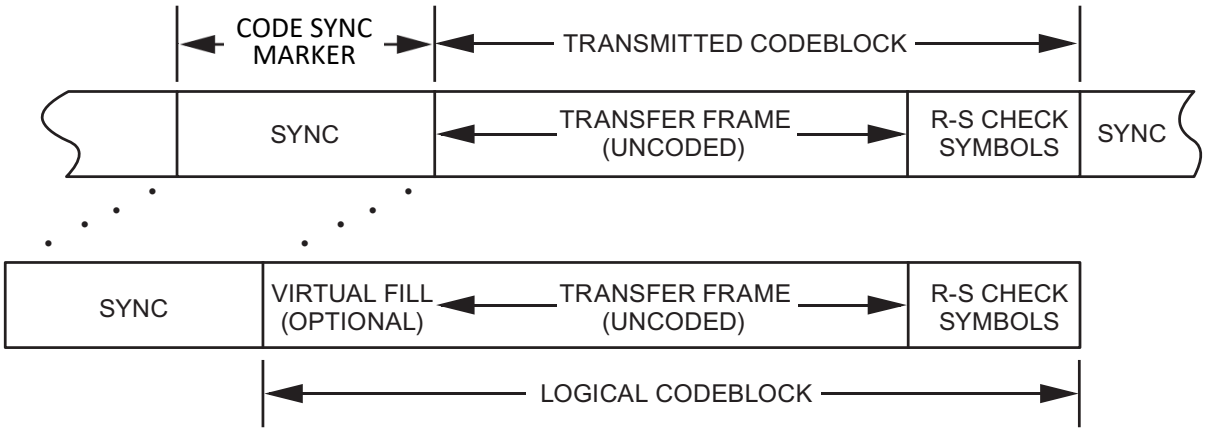


Figure 5-1: Reed-Solomon Codeblock Partitioning

5.3.8.2 **Virtual fill** shall be used to logically complete the codeblock. If used, virtual fill shall:

a) consist of all zeros;

b) not be transmitted;

c) not change in length for a Mission Phase on a particular Physical Channel;

d) be inserted only at the beginning of the codeblock (i.e., after the CSM but before the beginning of the transmitted codeblock);

e) be inserted only in integer multiples of 8*I* bits.

5.3.9 DUAL BASIS REPRESENTATION

5.3.9.1 Dual basis representation shall be used.

5.3.9.2 The order of transmission shall be **dual basis** eight-bit string $z_0, z_1, \ldots, z_7$ (i.e., with z_0 transmitted first).

5.3.9.3 The relationship between the two representations shall conform to the following two equations:

$$[z_0, \ldots, z_7] = [u_7, \ldots, u_0] \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

and

$$[u_7, \ldots, u_0] = [z_0, \ldots, z_7] \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

NOTES

1 Discussion of dual basis representation is contained in 5.4.2.

2 Further information relating the dual basis (Berlekamp) and conventional representations is given in annex F. Also included is a recommended scheme for permitting the symbols generated in a conventional encoder to be transformed to meet the symbol representation required by this document.

5.3.10 CODEBLOCK SYNCHRONIZATION

Codeblock synchronization of the R-S decoder shall be achieved by synchronization of the CSM associated with each codeblock. (See section 9.)

NOTE – At the receiving end, the ambiguity between true and complemented data must be resolved so that only true data is provided to the R-S decoder. Data in NRZ-L form is normally resolved using the 32-bit CSM, while NRZ-M data is self-resolving.

5.4 DISCUSSION

5.4.1 SYMBOL INTERLEAVING

Symbol interleaving is accomplished in a manner functionally described with the aid of figure 5-2. (It should be noted that this functional description does not necessarily correspond to the physical implementation of an encoder.)

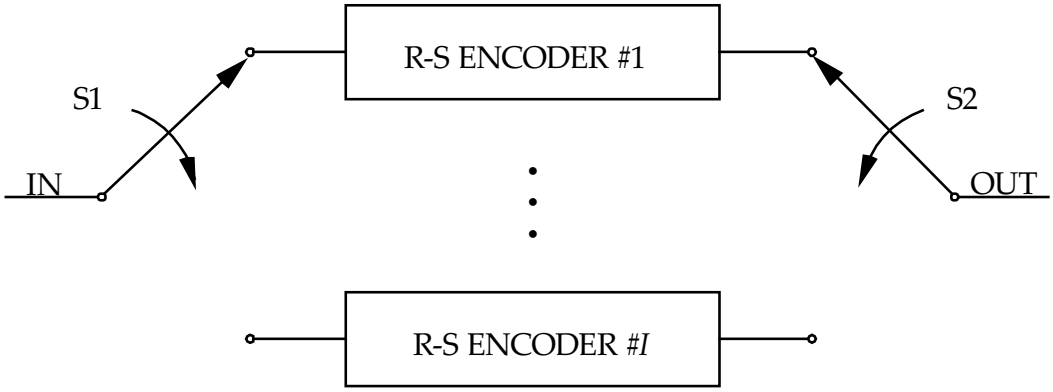


Figure 5-2: Functional Representation of R-S Interleaving

Data bits to be encoded into a single R-S codeblock enter at the port labeled 'IN'. Switches S1 and S2 are synchronized together and advance from encoder to encoder in the sequence 1,2, . . ., *I*, 1,2, . . ., *I*, . . ., spending one R-S symbol time (8 bits) in each position.

One codeblock will be formed from *kI* R-S symbols entering 'IN'. In this functional representation, a space of 2*EI* R-S symbols in duration is required between each entering set of *kI* R-S information symbols.

Because of the action of S1, each encoder accepts *k* of these symbols, with each symbol spaced *I* symbols apart (in the original stream). These *k* symbols are passed directly to the output of each encoder. The synchronized action of S2 reassembles the symbols at the port labeled 'OUT' in the same way as they entered at 'IN'.

Following this, each encoder outputs its 2*E* check symbols, one symbol at a time, as it is sampled in sequence by S2.

If, for *I*=5, the original symbol stream is

$$d_1^1 \ldots d_1^5 \, d_2^1 \ldots d_2^5 \ldots d_k^1 \ldots d_k^5 \quad [2E \times 5 \text{ spaces}]$$

then the output is the same sequence with the [2*E* × 5 spaces] filled by the [2*E*× 5] check symbols as shown below:

$$p_1^1 \ldots p_1^5 \ldots p_{2E}^1 \ldots p_{2E}^5$$

where

$$d_1^i \, d_2^i \ldots d_K^i \; p_1^i \ldots p_{2E}^i$$

is the R-S codeword produced by the *i*th encoder. If *q* virtual fill symbols are used in each codeword, then replace *k* by (*k* – *q*) in the above discussion.

With this method of interleaving, the original *kI* consecutive information symbols that entered the encoder appear unchanged at the output of the encoder with 2*EI* R-S check symbols appended.

5.4.2 DUAL BASIS SYMBOL REPRESENTATION AND ORDERING FOR TRANSMISSION

Each 8-bit R-S symbol is an element of the finite field GF(256). Since GF(256) is a vector space of dimension 8 over the binary field GF(2), the actual 8-bit representation of a symbol is a function of the particular basis that is chosen.

One basis for GF(256) over GF(2) is the set $(1, \alpha^1, \alpha^2, \ldots, \alpha^7)$. This means that any element of GF(256) has a representation of the form

$$u_7\alpha^7 + u_6\alpha^6 + \ldots + u_1\alpha^1 + u_0\alpha^0$$

where each u_i is either a zero or a one.

Another basis over GF(2) is the set $(1, \beta^1, \beta^2, \ldots, \beta^7)$ where $\beta = \alpha^{117}$. To this basis there exists a so-called 'dual basis' $(\ell_0, \ell_1, \ldots, \ell_7)$. It has the property that

$$\mathrm{Tr}(\ell_i\beta^j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

for each $j = 0, 1, \ldots, 7$. The function Tr(z), called the 'trace', is defined by

$$\mathrm{Tr}(z) = \sum_{K=0}^{7} z^{2^k}$$

for each element z of GF(256). Each R-S symbol can also be represented as

$$z_0\ell_0 + z_1\ell_1 + \ldots + z_7\ell_7$$

where each z_i is either a zero or a one.

6 CONCATENATED CODING

6.1 Concatenated codes shall consist of a combination of a R-S code defined in section 5 with one of the convolutional codes defined in section 4.

6.2 The R-S code shall be the outer code and the convolutional code shall be the inner code.

7 TURBO CODING

7.1 OVERVIEW

Turbo codes are binary block codes with large codewords (hundreds or thousands of bits). Turbo codes may be used to obtain even greater coding gains than those provided by concatenated coding systems. They are systematic and inherently non-transparent.

7.2 GENERAL

7.2.1 DATA RANDOMIZATION

The pseudo-randomizer defined in section 10 shall be used.

NOTE – Some legacy systems do not include the randomizer. The recommended Turbo codes, by themselves, cannot guarantee sufficient bit transitions to keep receiver symbol synchronizers in lock. The pseudo-randomizer is also used to aid signal acquisition and to mitigate spectral lines in the transmitted signal.

7.2.2 FRAME VALIDATION

When Turbo codes are used with TM, AOS, or USLP Transfer Frames, the FECF specified in references [1], [2], or [6], respectively, shall be used to validate the Transfer Frame.

NOTE – While providing outstanding coding gain, Turbo codes may still leave some undetected errors in the decoded output.

7.3 SPECIFICATION

NOTE – A Turbo encoder is a combination of two simple encoders. The input is a frame of k information bits. The two component encoders generate parity symbols from two simple recursive convolutional codes, each with a small number of states. The information bits are also sent uncoded. A key feature of Turbo codes is an interleaver, which permutes bit-wise the original k information bits before input to the second encoder.

7.3.1 Code type shall be systematic parallel concatenated Turbo code.

7.3.2 Number of component codes shall be two (plus an uncoded component to make the code systematic).

7.3.3 Type of component codes shall be recursive convolutional codes.

7.3.4 Number of states of each convolutional component code shall be 16.

7.3.5 Nominal code rates shall be r = 1/2, 1/3, 1/4, or 1/6 (selectable).

NOTE – Because of 'trellis termination' symbols (see 7.3.10), the true code rates (defined as the ratios of the information block lengths to the codeword lengths in table 7-2) are slightly smaller than the nominal code rates. In this Recommended Standard, the term 'code rate' always refers to the nominal code rates, r = 1/2, 1/3, 1/4, or 1/6.

7.3.6 The specified information block lengths k shall be those specified in table 7-1. The corresponding codeword lengths in bits, $n=(k+4)/r$, for the specified code rates shall be those specified in table 7-2.

NOTE – Information block lengths are chosen for compatibility with the corresponding R-S interleaving depths, also shown in table 7-1.

Table 7-1: Specified Information Block Lengths

Information block length k, bits	Corresponding Reed-Solomon interleaving depth I	Notes
1784 (=223 × 1 octets)	1	For very low data rates or low latency
3568 (=223 × 2 octets)	2	
7136 (=223 × 4 octets)	4	
8920 (=223 × 5 octets)	5	For highest coding gain

Table 7-2: Codeword Lengths for Supported Code Rates (Measured in Bits)

Information block length k	Codeword length n			
	rate 1/2	rate 1/3	rate 1/4	rate 1/6
1784	3576	5364	7152	10728
3568	7144	10716	14288	21432
7136	14280	21420	28560	42840
8920	17848	26772	35696	53544

7.3.7 Turbo code permutation for each specified block length k shall conform to a particular reordering of the integers 1, 2, . . ., k as generated by the following algorithm.

NOTE – The interleaver is a fundamental component of the Turbo encoding and decoding process. The interleaver for Turbo codes is a fixed bit-by-bit permutation of the entire block of data. Unlike the symbol-by-symbol rectangular interleaver used with R-S codes, the Turbo code permutation scrambles individual bits and resembles a randomly selected permutation in its lack of apparent orderliness.

1) k shall be expressed as $k=k_1k_2$; the parameters k_1 and k_2 for the specified block sizes shall be selected from table 7-3.

Table 7-3: Parameters k_1 and k_2 for Specified Information Block Lengths

Information block length	k_1	k_2
1784	8	223
3568	8	223 × 2
7136	8	223 × 4
8920	8	223 × 5

2) The following operations shall be performed for s=1 to s=k to obtain permutation numbers $\pi(s)$:

$p_1 = 31$; $p_2 = 37$; $p_3 = 43$; $p_4 = 47$; $p_5 = 53$; $p_6 = 59$; $p_7 = 61$; $p_8 = 67$

NOTE – In the equation below, $\lfloor x \rfloor$ denotes the largest integer less than or equal to x, and p_q denotes one of the following eight prime integers:

$$
\begin{aligned}
m &= (s-1) \bmod 2 \\
i &= \left\lfloor \frac{s-1}{2\,k_2} \right\rfloor \\
j &= \left\lfloor \frac{s-1}{2} \right\rfloor - i\,k_2 \\
t &= (19i+1) \bmod \frac{k_1}{2} \\
q &= t \bmod 8 + 1 \\
c &= (p_q\,j + 21m) \bmod k_2 \\
\pi(s) &= 2\left(t + c\,\frac{k_1}{2} + 1\right) - m
\end{aligned}
$$

The permutation numbers shall be interpreted such that the *s*th bit read out on line 'in b' in figure 7-2 is the $\pi(s)$th bit of the input information block, as shown in figure 7-1.

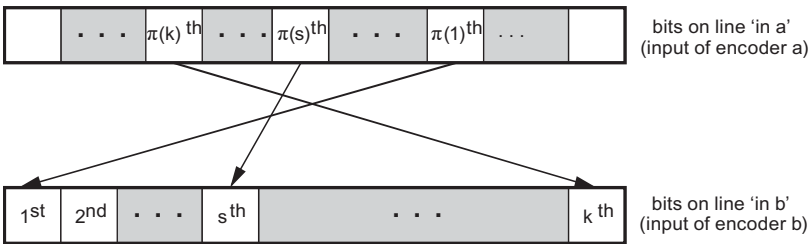


Figure 7-1: Interpretation of Permutation

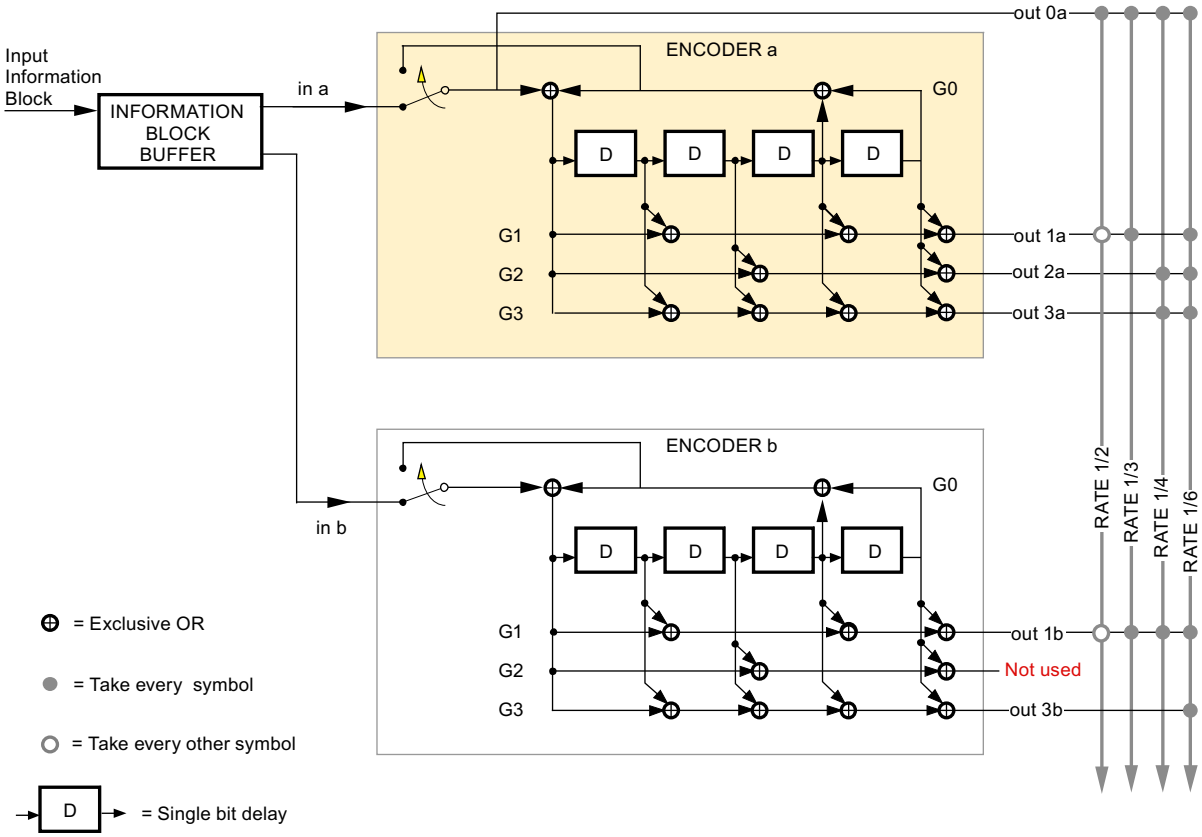


Figure 7-2: Turbo Encoder Block Diagram

7.3.8 Backward and forward connection vectors (see figure 7-2) shall be as follows:

1) Backward connection vector for both component codes and all code rates shall be G0 = 10011.

2) Forward connection vector for both component codes and rates 1/2 and 1/3 shall be G1 = 11011.

i) Puncturing of every other symbol from each component code shall be done for rate 1/2.

ii) No puncturing shall be done for rate 1/3.

3) Forward connection vectors for rate 1/4 shall be G2 = 10101, G3 = 11111 (1st component code); G1 = 11011 (2nd component code). No puncturing shall be done for rate 1/4.

4) Forward connection vectors for rate 1/6 shall be G1 = 11011, G2 = 10101, G3 = 11111 (1st component code); G1 = 11011, G3 = 11111 (2nd component code). No puncturing shall be done for rate 1/6.

7.3.9 Turbo encoder operation:

1) Each input frame of *k* information bits shall be held in a frame buffer, and the bits in the buffer shall be read out in two different orders for the two component encoders.

2) The first component encoder (a in figure 7-2) shall operate on the bits in unpermuted order ('in a'), while the second component encoder (b figure 7-2) shall receive the same bits permuted by the interleaver ('in b').

NOTES

1 The read-out addressing for 'in a' is a simple counter, while the addressing for 'in b' is specified by the Turbo code permutation described in 7.3.7.

2 The recommended encoder block diagram is shown in figure 7-2.

3 The component encoders are recursive convolutional encoders realized by feedback shift registers as shown in figure 7-2. The circuits shown in this figure implement the backward connection vector, G0, and the forward connection vectors, G1, G2, G3, specified in 7.3.8. A key difference between these convolutional component encoders and the standalone convolutional encoder recommended in section 4 is their recursiveness. In the figure this is indicated by the signal (corresponding to the backward connection vector G0) fed back into the leftmost adder of each component encoder.

7.3.10 Turbo codeword specification:

1) Both component encoders (see figure 7-2) shall be initialized with '0's in all registers, and both shall run for a total of k+4 bit times, producing an output codeword of (k+4)/r encoded symbols, where r is the nominal code rate.

2) For the first k bit times, the input switches shall be in the lower position (as indicated in figure 7-2) to receive input data.

3) For the final four bit times, these switches shall move to the upper position to receive feedback from the shift registers.

NOTE – This feedback cancels the same feedback sent (unswitched) to the leftmost adder and causes all four registers to become filled with zeros after the final four bit times. Filling the registers with zeros is called terminating the trellis.

4) During trellis termination the encoder shall continue to output nonzero encoded symbols. In particular, the 'systematic uncoded' output (line 'out 0a' in the figure) shall include an extra four bits from the feedback line in addition to the k information bits.

NOTE – In figure 7-2, the encoded symbols are multiplexed from top-to-bottom along the output line for the selected code rate to form the Turbo codeword. For the rate 1/3 code, the output sequence is (out 0a, out 1a, out 1b); for rate 1/4, the sequence is (out 0a, out 2a, out 3a, out 1b); for rate 1/6, the sequence is (out 0a, out 1a, out 2a, out 3a, out 1b, out 3b). These sequences are repeated for (k+4) bit times. For the rate 1/2 code, the output sequence is (out 0a, out 1a, out 0a, out 1b), repeated (k+4)/2 times. This pattern implies that out 1b is the first to be punctured, out 1a is the second, and so forth. The Turbo codewords constructed from these output sequences are depicted in figure 7-3 for the four nominal code rates.

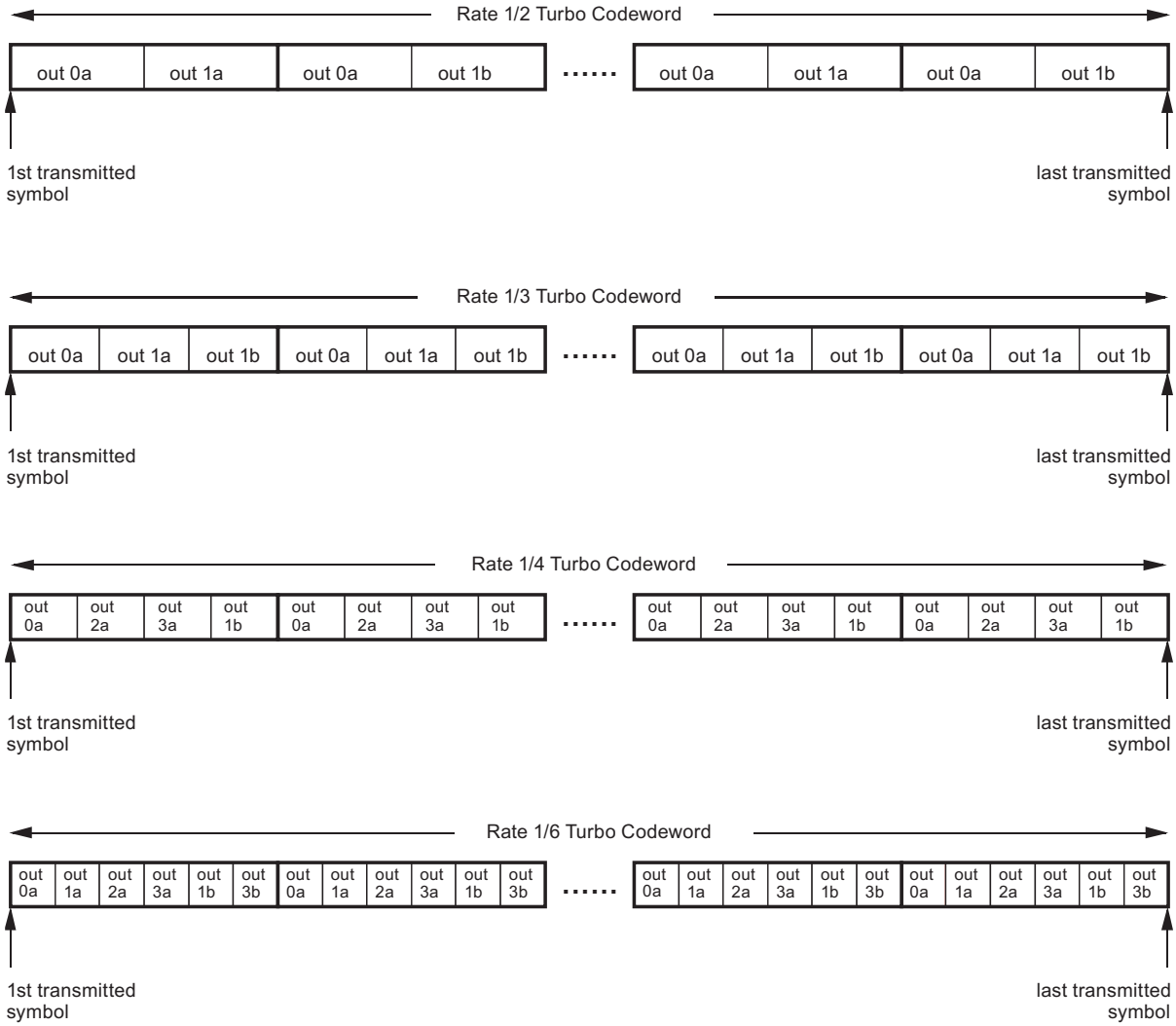


Figure 7-3: Turbo Codewords for Different Code Rates

7.3.11 Turbo codeword synchronization:

A CSM shall be associated with each Turbo codeword (see section 9), and it shall precede the Turbo codeword.

NOTES

1 The CSM is used to achieve codeword synchronization of the Turbo decoder (i.e., frame synchronizers are normally set to expect a marker at a recurrence interval equal to the length of the CSM plus that of the Turbo codeword).

2 Differential encoding does not provide benefits with Turbo Codes, and the CSM can also be used to resolve phase ambiguities. In fact, differential encoding before the Turbo encoder cannot be used because the Turbo codes recommended in this document are non-transparent, and differential encoding after the Turbo encoder is not advised because it introduces considerable loss of performance. It also would

require differential detection, which is more complex with soft symbols. This implies that phase ambiguities have to be detected and resolved before decoding.

3 A diagram of a Turbo codeword with CSM is shown in figure 7-4. The length of the Turbo codeword is inversely proportional to the nominal code rate *r*. Each codeword is placed between CSMs.

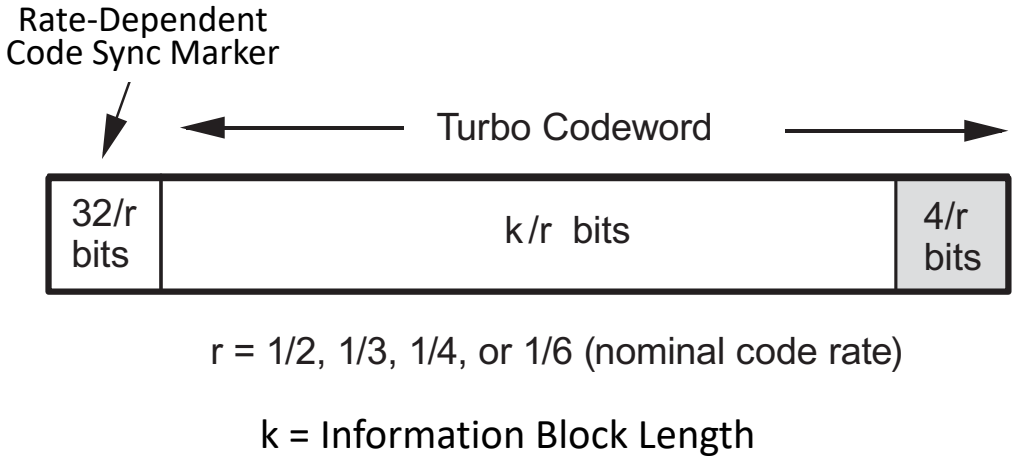


Figure 7-4: Turbo Codeword with Code Sync Marker

7.3.12 Channel interleaver:

A channel interleaver should be used to improve the communication channel against burst errors, unless the system designer verifies that the concerns identified in the note below are resolved by other means (e.g. improved fading margin in the link budget).

The use of the channel interleaver is a managed parameter.

NOTES

1 The turbo codes, by themselves, are vulnerable to burst errors. A channel interleaver is highly recommended for turbo codes in case of tumbling spacecraft or solar scintillation.

2 This channel interleaver is external to the Turbo encoding process and must not be confused with the intrinsic interleaver (bit permutation) between the two constituent convolutional codes. While the intrinsic interleaver changes the order of information bits, the channel interleaver changes the order of coded symbols.

The channel interleaver is a row-column interleaver with a depth of 4, constructed as follows:

1) At the sending end (encoder side), each of the $n=(k+4)/r$ coded symbols produced by the turbo encoder of rate $r=1/2$, 1/3, 1/4, or 1/6, are written on the channel

interleaver rows of size N_c=4/r =8, 12, 16, or 24. Each row thus contains four consecutive outputs of the two constituent encoders of the turbo encoder, see Figure 7-5.

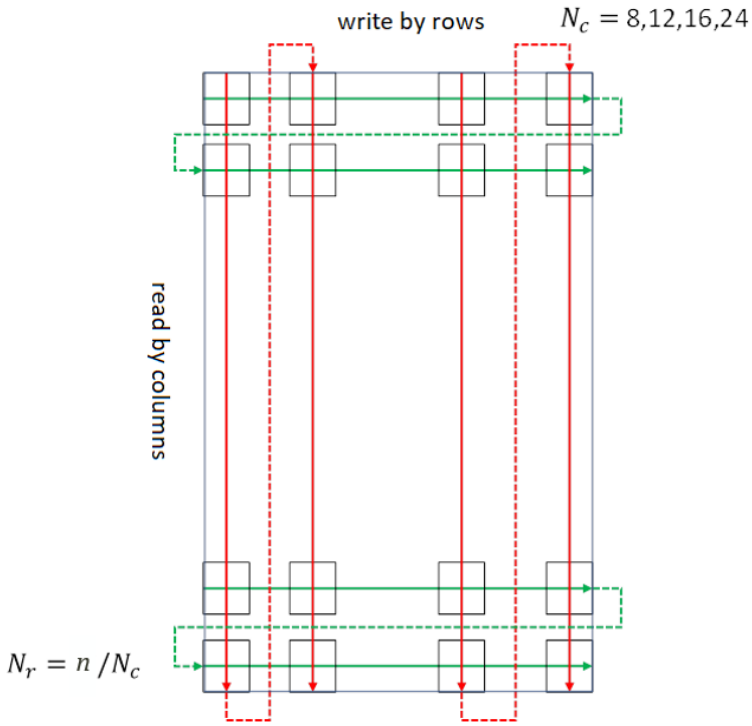


Figure 7-5: Writing and Reading Operations in the Row-Column Channel Interleaver

2) The symbols are then read out from channel interleaver column by column. The column length N_r depends on the codeword length as N_r= n/N_c.

NOTE – The resulting values of parameters n, N_r, and N_c for the 16 available code configurations are reported in Table 7-4. Notice that the number of columns depends only on the code rate r and the number of rows only on the block size k.

Table 7-4: Number of Rows and Columns of Row-Column Channel Interleavers

r	**1/2**		**1/3**		**1/4**		**1/6**	
N_c	**8**		**12**		**16**		**24**	
k	***n***	N_r	***n***	N_r	***n***	N_r	***n***	N_r
1784	3576	447	5364	447	7152	447	10728	447
3568	7144	893	10716	893	14288	893	21432	893
7136	14280	1785	21420	1785	28560	1785	42840	1785
8920	17848	2231	26772	2231	35696	2231	53544	2231

3) At the receiving end (decoder side), the channel de-interleaver performs the complementary function. Hence, the received symbols with the interleaved codewords are written in columns and read out in rows, using the same size for N_c and N_r defined in Table 7-4.

8 LOW-DENSITY PARITY-CHECK CODING

8.1 OVERVIEW

Low-Density Parity-Check (LDPC) codes are binary block codes with large codewords (hundreds or thousands of bits). They may be used to obtain greater coding gains than those provided by concatenated coding systems.

An LDPC code is specified indirectly by a *v*-by-*w* parity check matrix H consisting of *v* linearly independent rows. Parity check matrices may include additional linearly dependent rows without changing the code. A coded sequence of *w* bits must lie in the *w*–*v* dimensional dual space of H; that is, it must satisfy all *v* parity check equations corresponding to the *v* rows of H. Alternatively, the code can be described through a (*w*–*v*)-by-*w* generator matrix G, such that a coded sequence lies in the *w*–*v* dimensional space of G. An encoder maps an input frame of *k* information bits uniquely into a codeword of *n* bits. LDPC codes may be shortened or expurgated so that $k<w–v$, and the remaining dimensions of the code remain unused. LDPC codes may also be extended or punctured to make *n* greater or less than *w*.

The distinguishing feature of LDPC codes is a low density of ones in the matrix H. Conversely, the generator matrix G is usually dense; that is, its density of ones is in the same order of that of zeros, at least for the non-systematic part of G.

Subsection 8.3 describes a code with a rate of 223/255 (approximately 7/8), and 8.4 describes a set of nine codes with rates 1/2, 2/3, and 4/5. These codes are systematic and non-transparent.

8.2 GENERAL

8.2.1 SYNCHRONIZATION

8.2.1.1 The (8160,7136) code defined in 8.3 shall be used with the 32-bit CSM shown in figure 9-1.

8.2.1.2 All of the nine codes with rates 1/2, 2/3, and 4/5, defined in 8.4, shall be used with the 64-bit CSM shown in figure 9-2.

NOTE – Differential encoding does not provide benefits with LDPC codes, and the CSM can also be used to resolve phase ambiguities. In fact, differential encoding before the LDPC encoder cannot be used because the LDPC codes recommended in this document are non-transparent, and differential encoding after the LDPC encoder is not advised because it introduces considerable loss of performance. It also would require differential detection, which is more complex with soft symbols. This implies that phase ambiguities have to be detected and resolved before decoding.

8.2.2 DATA RANDOMIZATION

The pseudo-randomizer defined in section 10 shall be used.

NOTE – Some legacy systems do not include the randomizer. The recommended LDPC codes, by themselves, cannot guarantee sufficient bit transitions to keep receiver symbol synchronizers in lock. Because of the quasi-cyclic nature of these codes, undetected decoding errors may result from incorrect codeword synchronization. The pseudo-randomizer is also used to aid signal acquisition and to mitigate spectral lines in the transmitted signal.

8.2.3 FRAME VALIDATION

8.2.3.1 The LDPC decoder may be used alone to validate the Information Block, and consequently the contained Transfer Frame(s).

8.2.3.2 The FECF specified in references [1], [2], and [6] is optional, and the system designer may choose to use it for additional Transfer Frame validation.

NOTE – The undetected frame and bit error rates of these LDPC codes lie several orders of magnitude below the corresponding detected error rates for any given operating signal-to-noise ratio.

8.3 LOW-DENSITY PARITY-CHECK CODE WITH RATE 223/255

8.3.1 OVERVIEW

The (8160,7136) recommended code is an expurgated, shortened, and extended version of a basic (8176,7156) LDPC code.

The recommended code has rate 223/255 and matches the length and dimension of the (255,223) I=4 R-S code.

The basic code is transparent, although the modified version of this code is not, because of the sense of the fill bits.

Construction of the initial code is described in 8.3.2, expurgation and encoding are described in 8.3.4, and the shortening and extension that yield the recommended code are described in 8.3.5.

8.3.2 BASIC (8176,7156) LDPC CODE USED IN CONSTRUCTION

8.3.2.1 The parity check matrix for the (8176,7156) LDPC code shall be formed by using a 2×16 array of 511×511 square circulants.

NOTE – This creates a parity check matrix of size 1022×8176 and rank 1020.

8.3.2.2 The structure of the parity check base matrix shall be as shown in figure 8-1.

Figure 8-1: Base Parity Check Matrix of the Basic (8176,7156) LDPC Code

8.3.3 DISCUSSION

Each $A_{i,j}$ is a 511 × 511 circulant. The row weight of each of the 32 circulants is two, i.e., there are two '1's in each row. The total row weight of each row in the parity check matrix is 2 × 16 = 32. The position of the '1's in the first row of each circulant is defined in the second column of table 8-1; each subsequent row is given by a one-bit right cyclic shift of the preceding row. There are 511 possible positions, with position numbers between 0 and 510. The third column represents the absolute position of the '1's in the parity check matrix. There are 8176 possible positions, therefore these numbers are between 0 and 8175.

Table 8-1: Specification of Circulants

Circulant	'1's position in 1st row of circulant	Absolute '1's position in 1st row of Parity Check Matrix
$A_{1,1}$	0, 176	0, 176
$A_{1,2}$	12, 239	523, 750
$A_{1,3}$	0, 352	1022, 1374
$A_{1,4}$	24, 431	1557, 1964
$A_{1,5}$	0, 392	2044, 2436
$A_{1,6}$	151, 409	2706, 2964
$A_{1,7}$	0, 351	3066, 3417
$A_{1,8}$	9, 359	3586, 3936
$A_{1,9}$	0, 307	4088, 4395
$A_{1,10}$	53, 329	4652, 4928
$A_{1,11}$	0, 207	5110, 5317
$A_{1,12}$	18, 281	5639, 5902
$A_{1,13}$	0, 399	6132, 6531
$A_{1,14}$	202, 457	6845, 7100
$A_{1,15}$	0, 247	7154, 7401
$A_{1,16}$	36, 261	7701, 7926
$A_{2,1}$	99, 471	99, 471
$A_{2,2}$	130, 473	641, 984
$A_{2,3}$	198, 435	1220, 1457
$A_{2,4}$	260, 478	1793, 2011
$A_{2,5}$	215, 420	2259, 2464
$A_{2,6}$	282, 481	2837, 3036
$A_{2,7}$	48, 396	3114, 3462
$A_{2,8}$	193, 445	3770, 4022
$A_{2,9}$	273, 430	4361, 4518
$A_{2,10}$	302, 451	4901, 5050
$A_{2,11}$	96, 379	5206, 5489
$A_{2,12}$	191, 386	5812, 6007
$A_{2,13}$	244, 467	6376, 6599
$A_{2,14}$	364, 470	7007, 7113
$A_{2,15}$	51, 382	7205, 7536
$A_{2,16}$	192, 414	7857, 8079

8.3.4 ENCODING

NOTE – Two bits of information lie outside the structure of the quasi-cyclic encoder and increase complexity of the generator matrix for the basic (8176,7156) LDPC code. They are not included in this specification, which results in a generator matrix for a systematic (8176,7154) subcode that can be constructed entirely of circulants as shown in figure 8-2.

8.3.4.1 The generator matrix for the systematic (8176,7154) subcode shall be that illustrated in figure 8-2.

8.3.4.2 The left portion of the matrix shall be a 7154 × 7154 identity matrix, shown here as a block matrix, where I denotes the identity matrix of size 511 × 511, and 0 denotes the all-zero matrix of size 511 × 511.

8.3.4.3 The right portion of the matrix shall contain two columns of 511 × 511 circulants, denoted $B_{i,j}$, and constructed as follows:

1) $D = \begin{bmatrix} A_{1,15} & A_{1,16} \\ A_{2,15} & A_{2,16} \end{bmatrix}$ shall be defined from figure 8-1 and table 8-1.

NOTE – This equation describes a 1022 × 1022 matrix.

2) u = (1 0 0 0 …. 0) shall be the unit 511 tuple, i.e., a vector quantity of length 511 with a '1' at the leftmost position and '0's in the rest.

3) $z_i = (b_{i,1} \quad b_{i,2})$ shall be defined, where i = 1, 2, ..., 14 and the $b_{i,j}$s are the first rows of the $B_{i,j}$ circulants.

NOTE – For i = 1, 2, ..., 14, z_i is a vector with 1020 elements.

4) $M_i = \begin{bmatrix} A_{1,i} \\ A_{2,i} \end{bmatrix}$ shall be defined, where i = 1, 2, …, 14.

NOTE – The parity check matrix can now be represented as: $[M_1 \; M_2 \ldots M_{14} \; D]$.

5) The 511th and 1022nd elements of z_i shall be set to zero and shall be solved for z_i, where i = 1, 2, …, 14 and T superscript represents matrix transpose.

NOTE – Since the rank of D is 1020 and not 1022, there are two linearly dependent columns. These columns can be taken to be the 511th and 1022nd.

6) The $b_{i,j}$s shall be extracted from the z_is.

NOTE – The $b_{i,j}$s are numerically tabulated in annex C.

$$\begin{bmatrix}
I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & B_{1,1} & B_{1,2} \\
0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & B_{2,1} & B_{2,2} \\
0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & B_{3,1} & B_{3,2} \\
0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & B_{4,1} & B_{4,2} \\
0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & B_{5,1} & B_{5,2} \\
0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & B_{6,1} & B_{6,2} \\
0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & B_{7,1} & B_{7,2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & B_{8,1} & B_{8,2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & B_{9,1} & B_{9,2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & B_{10,1} & B_{10,2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & B_{11,1} & B_{11,2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & B_{12,1} & B_{12,2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & B_{13,1} & B_{13,2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & B_{14,1} & B_{14,2}
\end{bmatrix}$$

Figure 8-2: Systematic Circulant Generator Matrix

8.3.5 RECOMMENDED (8160,7136) CODE

NOTE – The generator matrix in 8.3.4 describes an (8176,7154) subcode of the (8176,7156) code defined by the parity check matrix in 8.3.2.2.

The (8176,7154) subcode shall be shortened and extended as follows to form an (8160,7136) code with parameters that are multiples of 32.

1) The encoder shall accept as input an Information Block of 7136 bits (i.e. 892 octets matching the length and dimension of (255,223) I=4 R-S),

2) 18 zeros shall be prefixed to the 7136-bit message to be encoded, yielding a 7154-element row vector.

3) This vector shall be multiplied by the generator matrix of section 6.2.3, yielding an 8176-element vector consisting of 18 zeros, 7136 systematic message symbols, and 1022 parity symbols.

4) From this vector, the 18 leading zeros shall be discarded, and two zeros shall be appended, yielding a codeword of 8160 symbols.

NOTE – The arrangement of these 18 virtual fill bits, information bits, parity bits, and two zero-fill bits is shown in figure 8-3.

Figure 8-3: Shortened Codeword

8.4 LOW-DENSITY PARITY-CHECK CODE FAMILY WITH RATES 1/2, 2/3, AND 4/5

8.4.1 OVERVIEW

Nine punctured LDPC codes are specified, with information block lengths k=1024 bits, 4096 bits, and 16384 bits, and code rates r=1/2, 2/3, and 4/5. These parameters and the corresponding codeword lengths, $n=k/r$, are shown in table 8-5.

8.4.2 PARITY CHECK MATRICES

8.4.2.1 The parity check matrices shall be constructed from $M \times M$ submatrices, where the submatrix size is listed in table 8-2.

Table 8-2: Values of Submatrix Size *M* for Supported Codes

Information block length k	Submatrix size M		
	rate 1/2	rate 2/3	rate 4/5
1024	512	256	128
4096	2048	1024	512
16384	8192	4096	2048

8.4.2.2 The parity check matrices for the rate-1/2 codes shall satisfy the following equation:

$$H_{1/2} = \begin{bmatrix} 0_M & 0_M & I_M & 0_M & I_M \oplus \Pi_1 \\ I_M & I_M & 0_M & I_M & \Pi_2 \oplus \Pi_3 \oplus \Pi_4 \\ I_M & \Pi_5 \oplus \Pi_6 & 0_M & \Pi_7 \oplus \Pi_8 & I_M \end{bmatrix}$$

where I_M and 0_M are the $M \times M$ identity and zero matrices, respectively, and Π_1 through Π_8 are permutation matrices.

8.4.2.3 The parity check matrices for the rate-2/3 and rate-4/5 codes are specified with additional columns and permutation matrices and shall satisfy the following equations:

$$H_{2/3} = \left[\begin{array}{cc|c} 0_M & 0_M & \\ \Pi_9 \oplus \Pi_{10} \oplus \Pi_{11} & I_M & H_{1/2} \\ I_M & \Pi_{12} \oplus \Pi_{13} \oplus \Pi_{14} & \end{array}\right]$$

$$H_{4/5} = \left[\begin{array}{cccc|c} 0_M & 0_M & 0_M & 0_M & \\ \Pi_{21} \oplus \Pi_{22} \oplus \Pi_{23} & I_M & \Pi_{15} \oplus \Pi_{16} \oplus \Pi_{17} & I_M & H_{2/3} \\ I_M & \Pi_{24} \oplus \Pi_{25} \oplus \Pi_{26} & I_M & \Pi_{18} \oplus \Pi_{19} \oplus \Pi_{20} & \end{array}\right]$$

8.4.2.4 The permutation matrix Π_K shall have non-zero entries in row i and column $\pi_K(i)$ for $i \in \{0,..., M\text{-}1\}$ and

$$\pi_k(i) = \frac{M}{4}\left(\left(\theta_k + \lfloor 4i / M \rfloor\right) \bmod 4\right) + \left(\phi_k\left(\lfloor 4i / M \rfloor, M\right) + i\right) \bmod \frac{M}{4}$$

where the functions θ_k and $\phi_k(j,M)$ are defined in table 8-3 and table 8-4.

NOTE – Values defined in tables 8-3 and 8-4 describe $\phi_k(j,M)$s using 7-tuples where consecutive positions in a tuple correspond to submatrix sizes from the set M={128, 256, 512, 1024, 2048, 4096, 8192}.

8.4.2.5 For each of the parity check matrices, the coded symbols corresponding to the last M columns shall be punctured (not transmitted).

Table 8-3: Description of $\phi_k(0,M)$ and $\phi_k(1,M)$

k	θ_k	$\phi_k(0,M)$ $M = 2^7 \ldots 2^{13}$							$\phi_k(1,M)$ $M = 2^7 \ldots 2^{13}$						
1	3	1	59	16	160	108	226	1148	0	0	0	0	0	0	0
2	0	22	18	103	241	126	618	2032	27	32	53	182	375	767	1822
3	1	0	52	105	185	238	404	249	30	21	74	249	436	227	203
4	2	26	23	0	251	481	32	1807	28	36	45	65	350	247	882
5	2	0	11	50	209	96	912	485	7	30	47	70	260	284	1989
6	3	10	7	29	103	28	950	1044	1	29	0	141	84	370	957
7	0	5	22	115	90	59	534	717	8	44	59	237	318	482	1705
8	1	18	25	30	184	225	63	873	20	29	102	77	382	273	1083
9	0	3	27	92	248	323	971	364	26	39	25	55	169	886	1072
10	1	22	30	78	12	28	304	1926	24	14	3	12	213	634	354
11	2	3	43	70	111	386	409	1241	4	22	88	227	67	762	1942
12	0	8	14	66	66	305	708	1769	12	15	65	42	313	184	446
13	2	25	46	39	173	34	719	532	23	48	62	52	242	696	1456
14	3	25	62	84	42	510	176	768	15	55	68	243	188	413	1940
15	0	2	44	79	157	147	743	1138	15	39	91	179	1	854	1660
16	1	27	12	70	174	199	759	965	22	11	70	250	306	544	1661
17	2	7	38	29	104	347	674	141	31	1	115	247	397	864	587
18	0	7	47	32	144	391	958	1527	3	50	31	164	80	82	708
19	1	15	1	45	43	165	984	505	29	40	121	17	33	1009	1466
20	2	10	52	113	181	414	11	1312	21	62	45	31	7	437	433
21	0	4	61	86	250	97	413	1840	2	27	56	149	447	36	1345
22	1	19	10	1	202	158	925	709	5	38	54	105	336	562	867
23	2	7	55	42	68	86	687	1427	11	40	108	183	424	816	1551
24	1	9	7	118	177	168	752	989	26	15	14	153	134	452	2041
25	2	26	12	33	170	506	867	1925	9	11	30	177	152	290	1383
26	3	17	2	126	89	489	323	270	17	18	116	19	492	778	1790

Table 8-4: Description of $\phi_k(2,M)$ and $\phi_k(3,M)$

k	θ_k	$\phi_k(2,M)$ $M = 2^7 \ldots 2^{13}$							$\phi_k(3,M)$ $M = 2^7 \ldots 2^{13}$						
1	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	12	46	8	35	219	254	318	13	44	35	162	312	285	1189
3	1	30	45	119	167	16	790	494	19	51	97	7	503	554	458
4	2	18	27	89	214	263	642	1467	14	12	112	31	388	809	460
5	2	10	48	31	84	415	248	757	15	15	64	164	48	185	1039
6	3	16	37	122	206	403	899	1085	20	12	93	11	7	49	1000
7	0	13	41	1	122	184	328	1630	17	4	99	237	185	101	1265
8	1	9	13	69	67	279	518	64	4	7	94	125	328	82	1223
9	0	7	9	92	147	198	477	689	4	2	103	133	254	898	874
10	1	15	49	47	54	307	404	1300	11	30	91	99	202	627	1292
11	2	16	36	11	23	432	698	148	17	53	3	105	285	154	1491
12	0	18	10	31	93	240	160	777	20	23	6	17	11	65	631
13	2	4	11	19	20	454	497	1431	8	29	39	97	168	81	464
14	3	23	18	66	197	294	100	659	22	37	113	91	127	823	461
15	0	5	54	49	46	479	518	352	19	42	92	211	8	50	844
16	1	3	40	81	162	289	92	1177	15	48	119	128	437	413	392
17	2	29	27	96	101	373	464	836	5	4	74	82	475	462	922
18	0	11	35	38	76	104	592	1572	21	10	73	115	85	175	256
19	1	4	25	83	78	141	198	348	17	18	116	248	419	715	1986
20	2	8	46	42	253	270	856	1040	9	56	31	62	459	537	19
21	0	2	24	58	124	439	235	779	20	9	127	26	468	722	266
22	1	11	33	24	143	333	134	476	18	11	98	140	209	37	471
23	2	11	18	25	63	399	542	191	31	23	23	121	311	488	1166
24	1	3	37	92	41	14	545	1393	13	8	38	12	211	179	1300
25	2	15	35	38	214	277	777	1752	2	7	18	41	510	430	1033
26	3	13	21	120	70	412	483	1627	18	24	62	249	320	264	1606

8.4.3 ENCODING

8.4.3.1 The encoder shall accept as input an information block of length k as per table 8-5.

8.4.3.2 Codewords consistent with the parity-check matrices in 8.4.2 shall be produced by performing matrix multiplication by block-circulant generator matrices.

NOTE – This family of codes supports rates $K/(K+2)$, where K=2 for a rate 1/2 code, K=4 for rate 2/3, and K=8 for rate 4/5. Corresponding generator matrices, G, have size $MK \times M(K+3)$ if punctured columns are described in the encoding, or $MK \times M(K+2)$ if punctured columns are omitted.

Table 8-5: Codeword Lengths for Supported Code Rates (Measured in Bits)

Information block length k	Codeword length n		
	rate 1/2	rate 2/3	rate 4/5
1024	2048	1536	1280
4096	8192	6144	5120
16384	32768	24576	20480

8.4.3.3 The generator matrices shall be constructed as follows:

1) P shall be the $3M \times 3M$ submatrix of H consisting of the last $3M$ columns. Q shall be the $3M \times MK$ submatrix of H consisting of the first MK columns.

2) $W = (P^{-1}Q)^T$ shall be computed, where the arithmetic is performed modulo-2.

8.4.3.4 The matrix $G = \left[I_{MK} \quad W \right]$ shall be constructed, where I_{MK} is the $MK \times MK$ identity matrix, and W is a dense matrix of circulants of size $MK \times 3M$.

8.4.3.5 The last M columns of G shall be punctured.

9 CODEWORD SYNCHRONIZATION

9.1 OVERVIEW

9.1.1 SYNCHRONIZATION

Codeword synchronization is necessary for proper decoding of R-S, Turbo, and LDPC codewords, and it is necessary for synchronization of the pseudo-random generator (see section 10). It is also useful in assisting the node synchronization process of the decoder for the convolutional code. Frame synchronization is necessary for subsequent processing of the Transfer Frames, as specified in Section 3.

For a coding system using the basic convolutional code specified in 4.3, the CSM can be acquired either in the coded symbol domain (i.e., before any decoding) or in the domain of bits decoded by the convolutional decoder.

For a concatenated R-S and convolutional coding system using the basic convolutional code specified in 4.3, the CSM can be acquired either in the coded symbol domain (i.e., before any decoding) or in the domain of bits decoded by the inner code (i.e., the coded symbol domain of the R-S code).

For a coding system using the punctured convolutional codes specified in 4.4, the CSM can only be acquired in the domain of bits decoded by the convolutional decoder. It cannot be acquired in the coded symbol domain (i.e., before any decoding).

For a concatenated R-S and convolutional coding system using the punctured convolutional codes specified in 4.4, the CSM can only be acquired in the domain of bits decoded by the inner code (i.e., the coded symbol domain of the R-S code); i.e.; it cannot be acquired in the coded symbol domain (i.e., before any decoding).

For a Turbo or LDPC coding system, the CSM can only be acquired in the coded symbol domain (i.e., before any decoding in the coded symbol domain of the Turbo or LDPC code).

9.1.2 CHANNEL ACCESS DATA UNIT

This Recommended Standard defines a data unit called the Channel Access Data Unit (CADU) whose contents are as per Table 9-1. The CADU is the protocol unit after block coding (if used), randomization, and CSM attachment, but prior to convolutional encoding (if used).

Table 9-1: Channel Access Data Unit Content with Different Coding Schemes

Applied Coding Scheme	CADU
Convolutional Coding	CSM and randomized Transfer Frame
R-S Coding	CSM and randomized R-S codeblock
Concatenated Coding	CSM and randomized R-S codeblock

Turbo Coding	CSM and randomized Turbo codeword
LDPC Coding	CSM and randomized LDPC codeword

9.2 THE CODE SYNCHRONIZATION MARKER

9.2.1.1 The CSM shall immediately follow the end of the preceding codeword or codeblock; i.e., there shall be no intervening bits preceding the CSM.

9.2.1.2 When slicing is not used, CSMs shall be placed between the Transfer Frames. In these cases, the CSMs also serve the function of ASMs, and the terms may be used interchangeably.

9.2.1.3 When R-S coding (section 5) or Concatenated coding (section 6) is used, CSMs shall be placed between the R-S codeblocks.

9.2.1.4 When Turbo coding (section 7) or LDPC coding of Transfer Frames (section 8) is used, CSMs shall be placed between the codewords.

NOTE – Synchronization is acquired on the receiving end by recognizing the specific bit pattern of the regularly spaced CSMs; synchronization may be verified by making further checks.

9.2.1.5 If convolutional coding is used, the CSM shall be convolutionally encoded.

9.2.1.6 If an inner convolutional code is used in conjunction with an outer R-S code, the CSM shall be encoded by the inner code but not by the outer code. (See section 6.)

9.3 CSM BIT PATTERNS

9.3.1 The CSM for data that is not Turbo or LDPC coded shall consist of a 32-bit (4-octet) marker with a pattern shown in figure 9-1.

9.3.2 The CSM for data that is Turbo coded with nominal code rate r = 1/2, 1/3, 1/4, or 1/6 shall consist of a 32/r-bit (4/r-octet) marker with bit patterns shown in figures 9-2 through 9-5.

9.3.3 The CSM for data that is LDPC coded with nominal code rate r = 7/8 shall consist of a 32-bit marker with bit pattern shown in figure 9-1.

9.3.4 The CSM for data that is LDPC coded with code rate r = 1/2, 2/3, or 4/5 shall consist of a 64-bit marker with bit pattern shown in figure 9-2.

NOTE – The CSM bit patterns are represented in hexadecimal notation as:

CSM for uncoded data, convolutional, Reed-Solomon, concatenated, and rate-7/8 LDPC coded data:	`1ACFFC1D`
CSM for rate-1/2 Turbo and rates 1/2, 2/3, and 4/5 LDPC coded data:	`034776C7272895B0`
CSM for rate-1/3 Turbo coded data:	`25D5C0CE8990F6C9461BF79C`
CSM for rate-1/4 Turbo coded data:	`034776C7272895B0 FCB88938D8D76A4F`
CSM for rate-1/6 Turbo coded data:	`25D5C0CE8990F6C9461BF79C DA2A3F31766F0936B9E40863`

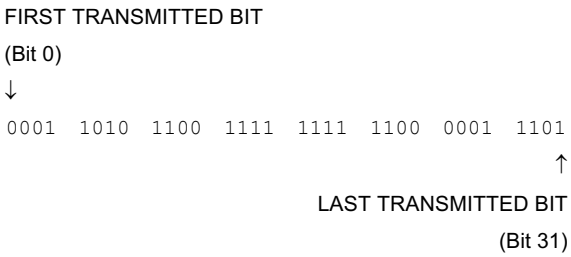


Figure 9-1: CSM Bit Pattern for Uncoded, Convolutional, Reed-Solomon, Concatenated, and Rate 7/8 LDPC Coded Data

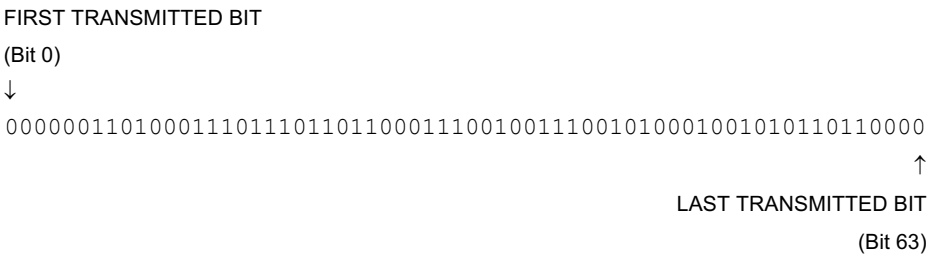


Figure 9-2: CSM Bit Pattern for Rate 1/2 Turbo and Rates 1/2, 2/3, and 4/5 LDPC Coded Data

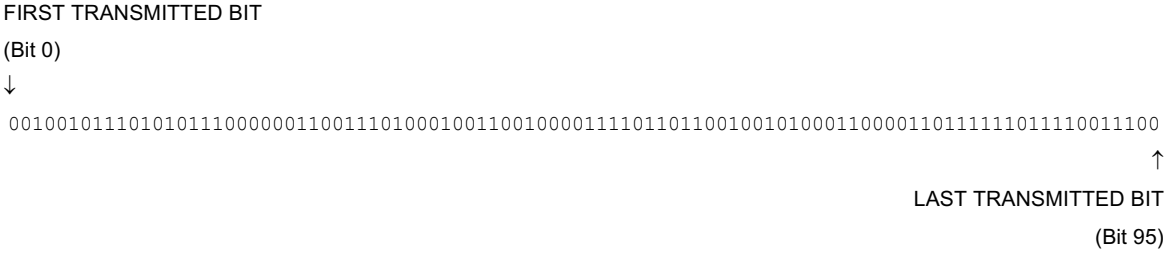


Figure 9-3: CSM Bit Pattern for Rate 1/3 Turbo Coded Data

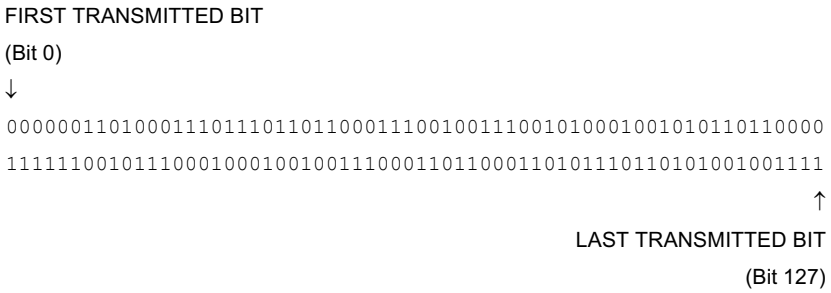


Figure 9-4: CSM Bit Pattern for Rate 1/4 Turbo Coded Data

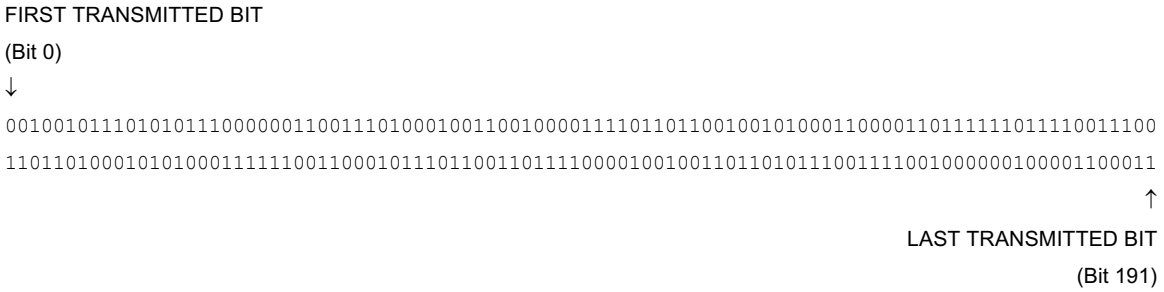


Figure 9-5: CSM Bit Pattern for Rate 1/6 Turbo Coded Data

9.4 LOCATION OF CSM

9.4.1 The CSM shall be attached to (i.e., shall immediately precede) the codeblock (if R-S encoded), the codeword (if Turbo or LDPC encoded), or the Transfer Frame (if convolutionally encoded only or uncoded).

9.4.2 The CSM shall immediately follow the end of the preceding codeblock (if R-S encoded), the codeword (if Turbo or LDPC encoded), or the Transfer Frame (if convolutionally encoded only or uncoded); i.e., there shall be no intervening bits (data or fill) preceding the CSM.

9.5 RELATIONSHIP OF CSM TO REED-SOLOMON, TURBO, AND LDPC CODEBLOCKS AND CODEWORDS

9.5.1 The CSM shall NOT be a part of the encoded data space of the R-S codeblock, and it shall not be presented to the input of the R-S encoder or decoder.

NOTE – This prevents the encoder from routinely regenerating a second, identical marker in the check symbol field under certain repeating data-dependent conditions (e.g., a test pattern of 01010101010 ... among others), which could cause synchronization difficulties at the receiving end. The relationship among the CSM, R-S codeblock, and Information Block is illustrated in figure 5-1.

9.5.2 Similarly, the CSM shall not be presented to the input of the Turbo encoder or decoder. It shall be directly attached to the Turbo codeword (see figure 7-4).

9.5.3 Similarly, the CSM shall not be presented to the input of the LDPC encoder or decoder. It shall be directly attached to the LDPC codeword.

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 data stream be sufficiently random (have little periodicity) and 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).

The Pseudo-Randomizer defined in this section is mandatory to ensure sufficient randomness for all combinations of CCSDS-recommended modulation and coding schemes.

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.

10.2 PSEUDO-RANDOMIZER DESCRIPTION

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

10.2.2 With no coding, at the sending end, the pseudo-randomizer shall be applied to the Transfer Frame. On the receiving end, it shall be applied to derandomize the data.

10.2.3 With convolutional coding alone, at the sending end, the pseudo-randomizer shall be applied to the Transfer Frame prior to encoding. On the receiving end, it shall be applied to derandomize the data after decoding.

10.2.4 With concatenated R-S and Convolutional coding, at the sending end, the pseudo-randomizer shall be applied to the codeblock after R-S encoding, but before convolutional encoding. On the receiving end, it shall be applied to derandomize the data after convolutional decoding but before R-S decoding.

10.2.5 With Turbo or LDPC coding, at the sending end, the pseudo-randomizer shall be applied to the codeword after encoding. At the receiving end, it shall be applied to the codeword to derandomize the data after decoding.

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

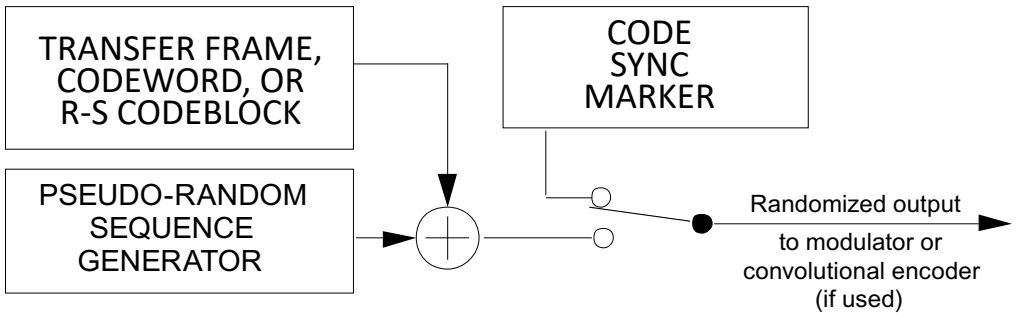


Figure 10-1: Pseudo-Randomizer Configuration

10.3 SYNCHRONIZATION AND APPLICATION OF PSEUDO-RANDOMIZER

10.3.1 The CSM shall be used for synchronizing the pseudo-randomizer.

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 CSM in the received data stream, the data immediately following the CSM shall be derandomized.

NOTES

1 The CSM was not randomized and is not derandomized.

2 Derandomization can be accomplished by performing exclusive-OR with hard bits or by inverting the soft bit values corresponding to the ‘ones’ in the pseudo-random sequence.

10.4 SEQUENCE SPECIFICATION

10.4.1 The pseudo-random sequence shall be 131071 bits in length and shall be generated using the following polynomial of degree seventeen, as shown in Figure 10-2:

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

10.4.2 For backward compatibility with legacy systems, a 255-bit pseudo-random sequence may be generated instead, by using the following polynomial of degree eight, as shown in Figure 10-3:

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

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 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 '11000111000111000' in case of the polynomial of degree 17, or to the 'all-ones' state, in case of the polynomial of degree 8, at the start of each codeblock, codeword, or Transfer Frame.

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

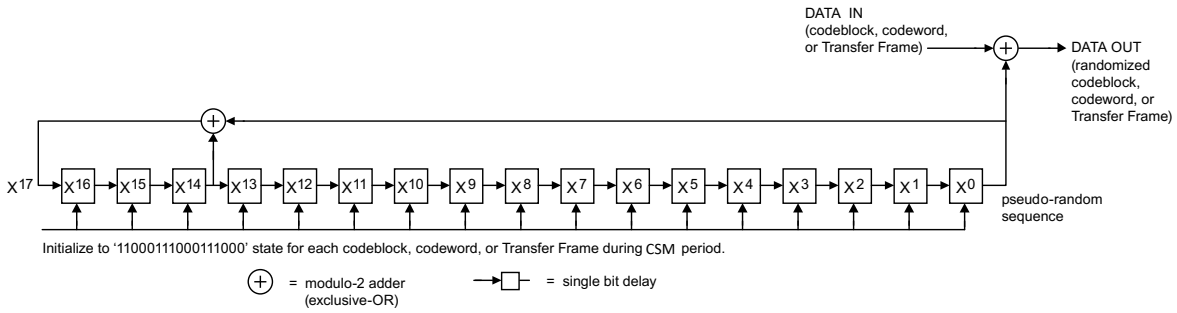


Figure 10-2: Pseudo-Randomizer Logic Diagram for the 131071-Bit Sequence

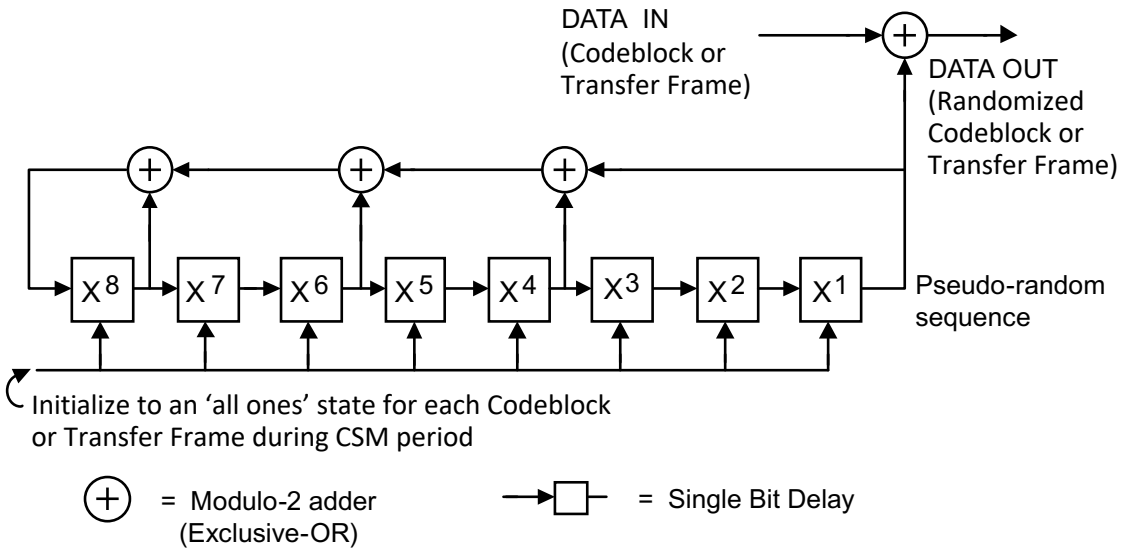


Figure 10-3: Pseudo-Randomizer Logic Diagram for the 255-Bit Sequence (Only for Backward Compatibility with Legacy Systems)

11 TRANSFER FRAME LENGTHS

11.1 OVERVIEW

The TM Space Data Link Protocol (SDLP, reference [1]), the AOS Space Data Link Protocol (reference [2]), and USLP (reference [6]) only specify the maximum length of Transfer Frames because there may be constraints on the Transfer Frame length depending on the selected coding options[2].

NOTE – The maximum length of a Transfer Frame is constrained by the number of bits allocated to the First Header Pointer in the Space Data Link Protocol. The 11-bit field in the TM protocol supports 2048 octet Transfer Frames, while the 16bit fields in AOS and USLP support 65536 octet Transfer Frames.

This Recommended Standard places no constraints on the Transfer Frame length if slicing is used, or if there is no coding, or if convolutional coding alone is used. Without slicing, when R-S (with or without concatenation), Turbo, or LDPC codes are used, there are additional constraints on the Transfer Frame length. The various cases are enumerated below.

11.2 GENERAL

11.2.1 Once selected, the Transfer Frame length shall be fixed for a Mission Phase on a particular Physical Channel.

NOTE – The Transfer Frame lengths shown here do not include the length of the CSM specified in section 9.

11.3 CASE 1: UNCODED

The length of the Transfer Frames shall be any integer number of octets, as required by the using project, with a maximum of 2048 octets for TM SDLP, and 65536 octets for AOS SDLP and USLP.

11.4 CASE 2: CONVOLUTIONAL ONLY

The length of the Transfer Frames shall be any integer number of octets, as required by the using project, with a maximum of 2048 octets for TM SDLP, and 65536 octets for AOS SDLP and USLP.

---

[2] USLP contains a frame length field that is specified by the sender for both fixed- or variable-length transfer frames. However, that field is not used by the procedures defined in this Recommended Standard.

11.5 CASE 3: WHEN TRANSFER FRAME SLICING IS USED

The length of the Transfer Frames shall be any integer number of octets, as required by the using project, with a maximum of 2048 octets for TM SDLP, and 65536 octets for AOS SDLP and USLP.

11.6 CASE 4: REED-SOLOMON ONLY, WITHOUT TRANSFER FRAME SLICING

NOTES

1 With the R-S Codes specified in section 5, only certain specific lengths of Transfer Frames may be contained within the codeblock's data space. In some cases, these lengths can be shortened at a small sacrifice in coding gain.

2 Since these R-S codes have a symbol length of 8 bits, the length of the codeblock (in octets) is a multiple of the interleaving depth, which provides 'octet compatibility'.

11.6.1 If necessary, Transfer Frame lengths shall be shortened in discrete steps by using virtual fill.

11.6.2 If high-speed efficiency is needed for '32-bit compatibility' (with 32-bit processors, for example), then the length of the codeblock shall be a combined multiple of 4 and the interleaving depth.

11.6.3 When only octet compatibility is required, lengths for Transfer Frames (L in octets) shall be determined using the following equation:

$$L = (255\text{-}2E\text{-}q)\, I$$

such that L is a positive integer,

where E = error correction capability,

q = number of virtual fill symbols per R-S codeword, and

I = interleaving depth.

11.6.4 When 32-bit compatibility is required, the Transfer Frame length must be chosen so that

a) it is expressed by the equation in 11.6.3; and

b) the codeblock length $(255\text{-}q)I$ (in octets) is a multiple of 4.

11.7 CASE 5: CONCATENATED, WITHOUT TRANSFER FRAME SLICING

The allowable lengths of Transfer Frames when the concatenated (R-S and convolutional) coding is used are the same as those for the R-S–only case (Case 4) shown in 11.6.

11.8 CASE 6: TURBO, WITHOUT TRANSFER FRAME SLICING

11.8.1 The Transfer Frame lengths shall be selected to match the information block lengths for the selected Turbo code.

NOTE – The Turbo Codes specified in section 7 of this Recommended Standard are block codes.

11.8.2 Only the following information block lengths shall be used (values are in octets):

a) 223;

b) 446;

c) 892;

d) 1115.

NOTE – Performance for only the above block lengths (i.e., Transfer Frame lengths) has been validated by CCSDS and approved for use (values are in octets).

11.9 CASE 7: LDPC, WITHOUT TRANSFER FRAME SLICING

11.9.1 The Transfer Frame lengths must match the information block lengths for the selected LDPC code.

NOTE – The LDPC Codes specified in section 8 of this Recommended Standard are block codes.

11.9.2 When the rate-7/8 LDPC code is used, the only allowable Transfer Frame length is 892 octets.

11.9.3 When the 1/2-, 2/3-, and 4/5-rate LDPC codes are used, the allowable Transfer Frame lengths are 128 octets, 512 octets, or 2048 octets.

12 MANAGED PARAMETERS

12.1 OVERVIEW

To conserve bandwidth on the space link, some parameters associated with synchronization and channel coding are handled by management rather than by inline communications protocol. The managed parameters are those which tend to be static for long periods of time, and whose change generally signifies a major reconfiguration of the synchronization and channel coding systems associated with a particular mission. Management conveys the required information to the synchronization and channel coding systems through a management system.

In this section, the managed parameters used by synchronization and channel coding systems are listed. These parameters are defined in an abstract sense and are not intended to imply any particular implementation of a management system.

12.2 GENERAL

12.2.1 All the managed parameters specified in this section shall be fixed for all Transfer Frames on a Physical Channel during a given Mission Phase.

12.2.2 When the R-S or LDPC codes are not used, the FECF defined in references [1] or [2] shall be present.

NOTE – The presence or absence of the FECF is established by the management of the relevant Data Link Protocol. When the R-S or LDPC codes are used, the FECF can still be present, but no check is required by the decoding system.

12.2.3 If present, the FECF shall occur within every Transfer Frame transmitted within the same Physical Channel throughout a Mission Phase.

12.3 MANAGED PARAMETERS FOR SELECTED OPTIONS

The managed parameters for a particular Physical Channel shall be those specified in table 12-1.

Table 12-1: Managed Parameters for Selected Options

Managed Parameter	Allowed Values
Randomizer	Long (131071 bits) Short (255 bits)
Coding Method	None Convolutional Reed-Solomon Concatenated Code Turbo LDPC

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
Slicing	Absent
Transfer Frame length (*L*, octets)	≤2048 with TM SDLP, ≤65536 with AOS SDLP or USLP

12.5 MANAGED PARAMETERS FOR REED-SOLOMON CODE

The managed parameters for R-S 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
Slicing	Present, Absent

Transfer Frame length (*L*, octets)	Slicing present: ≤2048 with TM SDLP, ≤65536 with AOS SDLP or USLP Slicing absent: (255-2*E*-*q*)*I*

12.6 MANAGED PARAMETERS FOR TURBO CODE

The managed parameters for Turbo code shall be those specified in table 12-4.

Table 12-4: Managed Parameters for Turbo Code

Managed Parameter	Allowed Values
Nominal Code Rate (*r*)	1/2, 1/3, 1/4, 1/6
Information Block Length (*k,* bits)	1784, 3568, 7136, 8920
Slicing	Present, Absent
Transfer Frame length (*L*, octets)	Slicing present: ≤2048 with TM SDLP, ≤65536 with AOS SDLP or USLP Slicing absent: *k*/8
Channel Interleaver	Present, Absent

12.7 MANAGED PARAMETERS FOR LOW-DENSITY PARITY-CHECK CODING

The managed parameters for LDPC coding shall be those specified in table 12-5.

Table 12-5: Managed Parameters for Low-Density Parity-Check Code

Managed Parameter	Allowed Values
Code Rate (*r*)	1/2, 2/3, 4/5, 7/8
Information Block Length (*k*, bits)	1024, 4096, 16384 (if *r*=1/2, 2/3, or 4/5), 7136 (if *r*=7/8)
Slicing	Present, Absent
Transfer Frame length (*L*, octets)	Slicing present: ≤2048 with TM SDLP, ≤65536 with AOS SDLP or USLP Slicing absent: *k*/8

13 USE OF TELECOMMUNICATIONS CHANNEL CODES FOR GROUND-TO-SPACE AND SPACE-TO-SPACE LINKS

13.1 OVERVIEW

The error control codes specified in this document are designed for use with fixed-length Transfer Frames as defined in the TM Space Data Link Protocol (reference [1]), AOS Space Data Link Protocol (reference [2]), or USLP (reference [6]). The AOS and USLP protocols are defined for Telemetry (downlink) use, as is TM, but AOS and USLP are also designed for use for ground-to-space and space-to-space communications. This bidirectional use will typically be adopted for high-rate missions, missions for which the space-to-ground and ground-to-space links are symmetric, or for missions that are adopting upper-layer networking protocols like delay/disruption-tolerant networking or Internet Protocol.

NOTE – In some cases, it is most practical to use Transfer Frames with a fixed length that does not need to match the information block length of the error correcting code. To support this application, synchronization markers may be prepended to the Transfer Frames, the resulting SMTFs concatenated into a stream, and then 'sliced' according to the information block length of the code. This slicing is described in Section 3.

13.2 TURBO CODES

13.2.1 DISCUSSION

Turbo codes are best suited to power-constrained links, for which the Signal-to-Noise Ratio (SNR), E_b/N_0, is a dominant concern. Their code rates of $r \leq 1/2$ provide greater coding gain than LDPC codes, at a cost of greater bandwidth expansion and increased demodulator and decoding complexity. They are best suited to links beyond low-Earth orbit.

13.2.2 SPECIFICATION

For AOS or USLP ground-to-space links, any of the turbo codes in section 7 shall be selected.

NOTES

1 The turbo codes in section 7 offer code rates of r =1/2, 1/3, 1/4, and 1/6, and block lengths of 1784, 3568, 7136, and 8920 information bits.

2 When a low-rate turbo code (particularly 1/4 or 1/6) is used near its decoding threshold, the symbol SNR may be below −5 dB. The radio receiver's symbol tracking loop may require an uncommonly narrow loop bandwidth, as well as an external means for Doppler compensation.

13.3 LOW-DENSITY PARITY-CHECK CODES

13.3.1 DISCUSSION

LDPC codes are best suited to high-data-rate links because of their code rates of $r \geq 1/2$ and the potentially parallel implementation architecture for the decoder. They are best suited to links on which bandwidth is limited, onboard computational resources are available to support an iterative decoder, and a Physical Layer modulation that supports at least two coded symbols per modulation symbol is available (e.g., QPSK/OQPSK and above—see reference [5]).

13.3.2 SPECIFICATION

For AOS or USLP ground-to-space links, any of the LDPC codes in section 8 shall be selected.

NOTE – The LDPC codes in section 8 offer code rates of r=1/2, 2/3, 4/5, and approximately 7/8. Block lengths of 1024, 4096, and 16384 information bits are available with the first three code rates, and 7136 information bits in the last case.

ANNEX A

SERVICE

(NORMATIVE)

A1 OVERVIEW

A1.1 BACKGROUND

This annex provides service definition in the form of primitives, which present an abstract model of the logical exchange of data and control information between the service provider and the service user. The definitions of primitives are independent of specific implementation approaches.

The parameters of the primitives are specified in an abstract sense and specify the information to be made available to the user of the primitives. The way in which a specific implementation makes this information available is not constrained by this specification. In addition to the parameters specified in this annex, an implementation can provide other parameters to the service user (e.g., parameters for controlling the service, monitoring performance, facilitating diagnosis, and so on).

A2 OVERVIEW OF THE SERVICE

The TM Synchronization and Channel Coding provides unidirectional (one way) transfer of a sequence of fixed-length TM, AOS, or USLP Transfer Frames at a constant frame rate over a Physical Channel across a space link, with optional error detection/correction.

Only one user can use this service on a Physical Channel.

A3 SERVICE PARAMETERS

A3.1 FRAME

A3.1.1 The Frame parameter is the service data unit of this service and shall be either a TM Transfer Frame defined in reference [1], an AOS Transfer Frame defined in reference [2], or a USLP Transfer Frame defined in reference [6].

A3.1.2 The length of any Transfer Frame transferred on a Physical Channel must be the same and is established by management.

A3.2 QUALITY INDICATOR

The Quality Indicator parameter shall be used to notify the user at the receiving end of the service that there is an uncorrectable error in the received Transfer Frame.

A3.3 SEQUENCE INDICATOR

The Sequence Indicator parameter shall be used to notify the user at the receiving end of the service that one or more Transfer Frames of the Physical Channel have been lost as the result of a loss of frame synchronization.

A4 SERVICE PRIMITIVES

A4.1 GENERAL

A4.1.1 The service primitives associated with this service are:

a) ChannelAccess.request;

b) ChannelAccess.indication.

A4.1.2 The ChannelAccess.request primitive shall be passed from the service user at the sending end to the service provider to request that a Frame be transferred through the Physical Channel to the user at the receiving end.

A4.1.3 The ChannelAccess.indication shall be passed from the service provider to the service user at the receiving end to deliver a Frame.

A4.2 ChannelAccess.request

A4.2.1 Function

The ChannelAccess.request primitive is the service request primitive for this service.

A4.2.2 Semantics

The ChannelAccess.request primitive shall provide a parameter as follows:

ChannelAccess.request (Frame)

A4.2.3 When Generated

The ChannelAccess.request primitive is passed to the service provider to request it to process and send the Frame.

A4.2.4 Effect On Receipt

Receipt of the ChannelAccess.request primitive causes the service provider to perform the functions described in 2.3.1 and to transfer the resulting coded symbols.

A4.3 ChannelAccess.indication

A4.3.1 Function

The ChannelAccess.indication primitive is the service indication primitive for this service.

A4.3.2 Semantics

The ChannelAccess.indication primitive shall provide parameters as follows:

ChannelAccess.indication	(Frame, Quality Indicator, Sequence Indicator)

A4.3.3 When Generated

The ChannelAccess.indication primitive is passed from the service provider to the service user at the receiving end to deliver a Frame.

A4.3.4 Effect On Receipt

The effect of receipt of the ChannelAccess.indication primitive by the service user is undefined.

ANNEX B

SECURITY, SANA, AND PATENT CONSIDERATIONS

(INFORMATIVE)

B1 SECURITY CONSIDERATIONS

B1.1 SECURITY BACKGROUND

It is assumed that security is provided by encryption, authentication methods, and access control to be performed at higher layers (application and/or transport layers). Mission and service providers are expected to select from recommended security methods, suitable to the specific application profile. Specification of these security methods and other security provisions is outside the scope of this Recommended Standard. The coding layer has the objective of delivering data with the minimum possible number of residual errors. An LDPC, R-S, or other code with FECF must be used to ensure that residual errors are detected and the frame flagged. There is an extremely low probability of additional undetected errors that may escape this scrutiny. These errors may affect the encryption process in unpredictable ways, possibly affecting the decryption stage and producing data loss, but will not compromise the security of the data.

B1.2 SECURITY CONCERNS

Security concerns in the areas of data privacy, authentication, access control, availability of resources, and auditing are to be addressed in higher layers and are not related to this Recommended Standard. The coding layer does not affect the proper functioning of methods used to achieve such protection at higher layers, except for undetected errors, as explained above.

The physical integrity of data bits is protected from channel errors by the coding systems specified in this Recommended Standard. In case of congestion or disruption of the link, the coding layer provides methods for frame re-synchronization.

B1.3 POTENTIAL THREATS AND ATTACK SCENARIOS

An eavesdropper can receive and decode the codewords but will not be able to get to the user data if proper encryption is performed at a higher layer. An interferer could affect the performance of the decoder by congesting it with unwanted data, but such data would be rejected by the authentication process. Such interference or jamming must be dealt with at the Physical Layer and through proper spectrum regulatory entities.

B1.4 CONSEQUENCES OF NOT APPLYING SECURITY

There are no specific security measures prescribed for the coding layer. Therefore, consequences of not applying security are only imputable to the lack of proper security measures in other layers. Residual undetected errors may produce additional data loss when the link carries encrypted data.

B2 SANA CONSIDERATIONS

The recommendations of this document do not require any action from Space Assigned Numbers Authority.

B3 PATENT CONSIDERATIONS

B3.1 TURBO CODING

Implementers should be aware that a wide class of Turbo codes is covered by a patent by France Télécom and Télédiffusion de France under US Patent 5,446,747 and its counterparts in other countries.[3]

Potential user agencies should direct their requests for licenses to:

Mr. Christian Hamon
CCETT GIE/CVP
4 rue du Clos Courtel
BP59, 35512 CESSON SEVIGNE Cedex, France
Tel: +33 2 99 12 48 05, Fax: +33 2 99 12 40 98
E-mail: christian.hamon@cnet.francetelecom.fr

B3.2 LDPC CODING

Implementers should be aware that the 1/2-, 2/3-, and 4/5-rate codes are covered by US Patent 7,343,539. Potential user agencies should direct their requests for licenses to:

Office of Technology Transfer
California Institute of Technology
1200 E. California Blvd., Mail Code 210-85
Pasadena, CA 91125
Tel: +1 626 395 3822, +1 626 577 2528
E-mail: Vieregg@Caltech.edu

[3] US Patent 5446747 expired in 2012.

ANNEX C

ANNEX TO SUBSECTION 8.3, LOW-DENSITY PARITY-CHECK CODE WITH RATE 223/255

(INFORMATIVE)

C1 GENERATOR MATRIX CIRCULANT TABLE

Table 13-1: Table of Circulants for the Generator Matrix

Circulant	1st row of circulant
$b_{1,1}$	55BF56CC55283DFEEFEA8C8CFF04E1EBD9067710988E25048D67525426939E206 8D2DC6FCD2F822BEB6BD96C8A76F4932AAE9BC53AD20A2A9C86BB461E43759C
$b_{1,2}$	6855AE08698A50AA3051768793DC238544AF3FE987391021AAF6383A6503409C3 CE971A80B3ECE12363EE809A01D91204F1811123EAB867D3E40E8C652585D28
$b_{2,1}$	62B21CF0AEE0649FA67B7D0EA6551C1CD194CA77501E0FCF8C85867B9CF679C18B CF7939E10F8550661848A4E0A9E9EDB7DAB9EDABA18C168C8E28AACDDEAB1E
$b_{2,2}$	64B71F486AD57125660C4512247B229F0017BA649C6C11148FB00B70808286F1A 9790748D296A593FA4FD2C6D7AAF7750F0C71B31AEE5B400C7F5D73AAF00710
$b_{3,1}$	681A8E51420BD8294ECE13E491D618083FFBBA830DB5FAF330209877D801F92B5 E07117C57E75F6F0D873B3E520F21EAFD78C1612C6228111A369D5790F5929A
$b_{3,2}$	04DF1DD77F1C20C1FB570D7DD7A1219EAECEA4B2877282651B0FFE713DF338A63 263BC0E324A87E2DC1AD64C9F10AAA585ED6905946EE167A73CF04AD2AF9218
$b_{4,1}$	35951FEE6F20C902296C9488003345E6C5526C5519230454C556B8A04FC0DC642 D682D94B4594B5197037DF15B5817B26F16D0A3302C09383412822F6D2B234E
$b_{4,2}$	7681CF7F278380E28F1262B22F40BF3405BFB92311A8A34D084C086464777431D BFDDD2E82A2E6742BAD6533B51B2BDEE0377E9F6E63DCA0B0F1DF97E73D5CD8
$b_{5,1}$	188157AE41830744BAE0ADA6295E08B79A44081E111F69BBE7831D07BEEBF7623 2E065F752D4F218D39B6C5BF20AE5B8FF172A7F1F680E6BF5AAC3C4343736C2
$b_{5,2}$	5D80A6007C175B5C0DD88A442440E2C29C6A136BBCE0D95A58A83B48CA0E7474E 9476C92E33D164BFF943A61CE1031DFF441B0B175209B498394F4794644392E
$b_{6,1}$	60CD1F1C282A1612657E8C7C1420332CA245C0756F78744C807966C3E13264388 78BD2CCC83388415A612705AB192B3512EEF0D95248F7B73E5B0F412BF76DB4
$b_{6,2}$	434B697B98C9F3E48502C8DBD891D0A0386996146DEBEF11D4B833033E05EDC28 F808F25E8F314135E6675B7608B66F7FF3392308242930025DDC4BB65CD7B6E
$b_{7,1}$	766855125CFDC804DAF8DBE3660E8686420230ED4E049DF11D82E357C54FE256E A01F5681D95544C7A1E32B7C30A8E6CF5D0869E754FFDE6AEFA6D7BE8F1B148
$b_{7,2}$	222975D325A487FE560A6D146311578D9C5501D28BC0A1FB48C9BDA173E869133 A3AA9506C42AE9F466E85611FC5F8F74E439638D66D2F00C682987A96D8887C
$b_{8,1}$	14B5F98E8D55FC8E9B4EE453C6963E052147A857AC1E08675D99A308E7269FAC5 600D7B155DE8CB1BAC786F45B46B523073692DE745FDF10724DDA38FD093B1C

Circulant	1st row of circulant
$b_{8,2}$	1B71AFFB8117BCF8B5D002A99FEEA49503C0359B056963FE5271140E626F6F8FC E9F29B37047F9CA89EBCE760405C6277F329065DF21AB3B779AB3E8C8955400
$b_{9,1}$	0008B4E899E5F7E692BDCE69CE3FAD997183CFAEB2785D0C3D9CAE510316D4BD6 5A2A06CBA7F4E4C4A80839ACA81012343648EEA8DBBA2464A68E115AB3F4034
$b_{9,2}$	5B7FE6808A10EA42FEF0ED9B41920F82023085C106FBBC1F56B567A14257021BC 5FDA60CBA05B08FAD6DC3B0410295884C7CCDE0E56347D649DE6DDCEEB0C95E
$b_{10,1}$	5E9B2B33EF82D0E64AA2226D6A0ADCD179D5932EE1CF401B336449D0FF775754C A56650716E61A43F963D59865C7F017F53830514306649822CAA72C152F6EB2
$b_{10,2}$	2CD8140C8A37DE0D0261259F63AA2A420A8F81FECB661DBA5C62DF6C817B4A61D 2BC1F068A50DFD0EA8FE1BD387601062E2276A4987A19A70B460C54F215E184
$b_{11,1}$	06F1FF249192F2EAF063488E267EEE994E7760995C4FA6FFA0E4241825A7F5B65 C74FB16AC4C891BC008D33AD4FF97523EE5BD14126916E0502FF2F8E4A07FC2
$b_{11,2}$	65287840D00243278F41CE1156D1868F24E02F91D3A1886ACE906CE741662B40B 4EFDFB90F76C1ADD884D920AFA8B3427EEB84A759FA02E00635743F50B942F0
$b_{12,1}$	4109DA2A24E41B1F375645229981D4B7E88C36A12DAB64E91C764CC43CCEC188E C8C5855C8FF488BB91003602BEF43DBEC4A621048906A2CDC5DBD4103431DB8
$b_{12,2}$	2185E3BC7076BA51AAD6B199C8C60BCD70E8245B874927136E6D8DD527DF0693D C10A1C8E51B5BE93FF7538FA138B335738F4315361ABF8C73BF40593AE22BE4
$b_{13,1}$	228845775A262505B47288E065B23B4A6D78AFBDDB2356B392C692EF56A35AB4A A27767DE72F058C6484457C95A8CCDD0EF225ABA56B7657B7F0E947DC17F972
$b_{13,2}$	2630C6F79878E50CF5ABD353A6ED80BEACC7169179EA57435E44411BC7D566136 DFA983019F3443DE8E4C60940BC4E31DCEAD514D755AF95A622585D69572692
$b_{14,1}$	7273E8342918E097B1C1F5FEF32A150AEF5E11184782B5BD5A1D8071E94578B0A C722D7BF49E8C78D391294371FFBA7B88FABF8CC03A62B940CE60D669DFB7B6
$b_{14,2}$	087EA12042793307045B283D7305E93D8F74725034E77D25D3FF043ADC5F8B5B1 86DB70A968A816835EFB575952EAE7EA4E76DF0D5F097590E1A2A978025573E

The numbers in the second column represent the hexadecimal representation of the first row of each circulant. Since there are only 511 possible positions, the leftmost bit is padded with a zero to allow a 128-digit hexadecimal number, i.e., the leftmost number in the specification of each circulant shall be interpreted as octal (=3 bits), as opposed to the others which shall be interpreted as hexadecimal (=4 bits). The generator matrix circulants do not have a low density of '1's as do the parity-check matrix circulants in table 8-1. All $B_{i,j}$ circulant shifting is done by a single-bit right shift of the previous row.

ANNEX D

ABBREVIATIONS AND ACRONYMS

(INFORMATIVE)

D1 INTRODUCTION

This annex lists key abbreviations and acronyms that are used throughout this Recommended Standard.

D2 ACRONYMS

AOS	Advanced Orbiting Systems
ASM	Attached Sync Marker
CADU	Channel Access Data Unit
CCSDS	Consultative Committee for Space Data Systems
CSM	Code Sync Marker
FECF	Frame Error Control Field
GF	Galois Field
LDPC	Low-Density Parity-Check
MSB	Most Significant Bit
NRZ-L	Non-Return-to-Zero-Level
NRZ-M	Non-Return-to-Zero-Mark
OQPSK	Offset Quadrature Phase Shift Keying
OSI	Open Systems Interconnection
QPSK	Quadrature Phase Shift Keying
R-S	Reed-Solomon
SANA	Space Assigned Numbers Authority
SDLP	Space Data Link Protocol
SMTF	Sync-Marked Transfer Frame
TM	Telemetry
USLP	Unified Space Link Protocol

ANNEX E

INFORMATIVE REFERENCES

(INFORMATIVE)

[E1] *Organization and Processes for the Consultative Committee for Space Data Systems*. Issue 4. CCSDS Record (Yellow Book), CCSDS A02.1-Y-4. Washington, D.C.: CCSDS, April 2014.

[E2] *Telemetry Channel Coding*. Issue 6-S. Recommendation for Space Data System Standards (Historical), CCSDS 101.0-B-6-S. Washington, D.C.: CCSDS, (October 2002) August 2005.

[E3] *Advanced Orbiting Systems, Networks and Data Links: Architectural Specification*. Issue 3-S. Recommendation for Space Data System Standards (Historical), CCSDS 701.0-B-3-S. Washington, D.C.: CCSDS, (June 2001) August 2005.

[E4] M. Perlman and J. J. Lee. *Reed-Solomon Encoders—Conventional vs. Berlekamp's Architecture*. JPL Publication 82-71. Pasadena, California: JPL, December 1, 1982.

[E5] *Flexible Advanced Coding and Modulation Scheme for High Rate Telemetry Applications*. Issue 2. Recommendation for Space Data System Standards (Blue Book), CCSDS 131.2-B-2. Washington, D.C.: CCSDS, February 2023.

[E6] *CCSDS Space Link Protocols over ETSI DVB-S2 Standard*. Issue 2. Recommendation for Space Data System Standards (Blue Book), CCSDS 131.3-B-2. Washington, D.C.: CCSDS, April 2022.

[E7] *TM Synchronization and Channel Coding—Summary of Concept and Rationale*. Issue 3. Recommendation for Space Data System Standards, CCSDS 131.1-G-3. Washington, D.C.: CCSDS, June 2020.

NOTE – Normative references are listed in 1.7.

ANNEX F

TRANSFORMATION BETWEEN BERLEKAMP AND CONVENTIONAL REPRESENTATIONS

(INFORMATIVE)

F1 PURPOSE

This annex provides information to assist users of the R-S code in this Recommended Standard to transform between the Berlekamp (dual basis) and Conventional representations. In addition, it shows where transformations are made to allow a conventional encoder to produce the dual basis representation on which this Recommended Standard is based.

F2 TRANSFORMATION

Referring to figure 13-1, it can be seen that information symbols I entering and check symbols C emanating from the Berlekamp R-S encoder are interpreted as

$$[z_0, z_1, \ldots, z_7]$$

where the components z_i are coefficients of ℓ_i, respectively:

$$z_0\ell_0 + z_1\ell_1 + \ldots + z_7\ell_7$$

Information symbols I' entering and check symbols C' emanating from the conventional R-S encoder are interpreted as

$$[u_7, u_6, \ldots, u_0]$$

where the components u_j are coefficients of α^j, respectively:

$$u_7\alpha^7 + u_6\alpha^6 + \ldots + u_0$$

A pre- and post-transformation is required when employing a conventional R-S encoder.

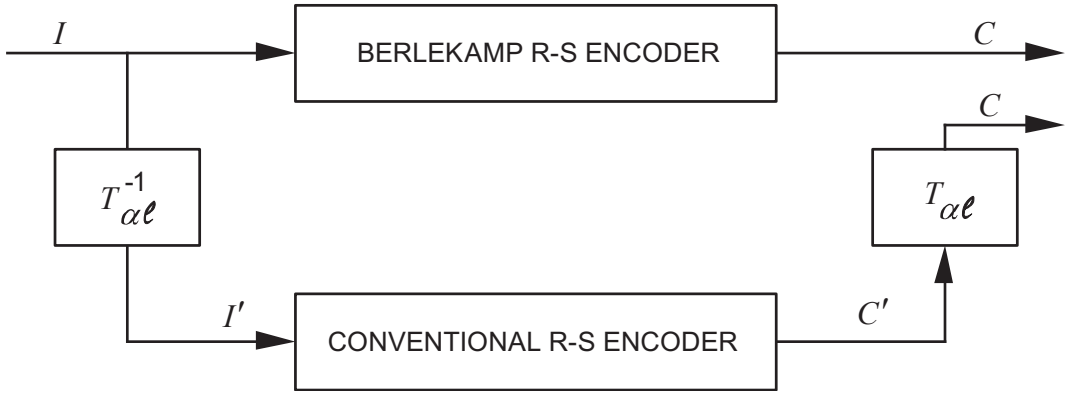


Figure 13-1: Transformational Equivalence

Conventional and Berlekamp types of (255,*k*) R-S encoders are assumed to have the same self-reciprocal generator polynomial whose coefficients appear in 5.3.3 and 5.3.4. The representation of symbols associated with the conventional encoder is the polynomials in 'α' appearing in table F-. Corresponding to each polynomial in 'α' is the representation in the dual basis of symbols associated with the Berlekamp type encoder.

Given

$$\alpha^i = u_7\alpha^7 + u_6\alpha^6 + ... + u_0$$

where $0 \le i < 255$ (and α^* denotes the zero polynomial, $u_7, u_6, ... = 0, 0, ...$),

the corresponding element is

$$z = z_0\ell_0 + z_1\ell_1 + ... + z_7\ell_7$$

where

$$[z_0, z_1, ..., z_7] = [u_7, u_6, ..., u_0]\, T_{\alpha\ell}$$

and

$$T_{\alpha\ell} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Row 1, row 2, ... , and row 8 in $T_{\alpha\ell}$ are representations in the dual basis of α^7 (10 ... 0), α^6 (010 ... 0), ... , and α^0 (00 ... 01), respectively.

The inverse of $T_{\alpha\ell}$ is

$$T_{\alpha\ell}^{-1} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Row 1, row 2, ... , and row 8 in $T_{\alpha\ell}^{-1}$ are polynomials in 'α' corresponding to ℓ_0 (10 ... 0), ℓ_1 (010 ... 0), ... , and ℓ_7 (00, ... 01), respectively. Thus,

$$[z_0, z_1, \ldots, z_7]\, T_{\alpha\ell}^{-1} = [u_7, u_6, \ldots, u_0]$$

Example 1:

Given information symbol *I*,

$$[z_0, z_1, \ldots, z_7] = 10111001$$

then

$$[1\ 0\ 1\ 1\ 1\ 0\ 0\ 1] \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} = [u_7, u_6, ..., u_0] = 00101010 = I'$$

($T_{\alpha\ell}^{-1}$)

The arithmetic operations are reduced modulo 2. Also,

$$[z_0, z_1, \ldots, z_7] = 10111001$$

and

$$[u_7, u_6, \ldots, u_0] = 00101010\ (\alpha^{213})$$

are corresponding entries in table F-.

Example 2:

Given check symbol C',

$$[\alpha_7, \alpha_6, \ldots, \alpha_0] = 01011001\ (\alpha^{152})$$

Then,

$$[0\ 1\ 0\ 1\ 1\ 0\ 0\ 1] \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} = [z_0, z_1, \ldots, z_7] = 11101000 = C$$

Table F-1: Equivalence of Representations[4,5]

POWER	POLY IN ALPHA	$\ell_{01234567}$	POWER	POLY IN ALPHA	$\ell_{01234567}$
*	00000000	00000000	31	11001101	01111010
0	00000001	01111011	32	00011101	10011110
1	00000010	10101111	33	00111010	00111111
2	00000100	10011001	34	01110100	00011100
3	00001000	11111010	35	11101000	01110100
4	00010000	10000110	36	01010111	00100100
5	00100000	11101100	37	10101110	10101101
6	01000000	11101111	38	11011011	11001010
7	10000000	10001101	39	00110001	00010001
8	10000111	11000000	40	01100010	10101100
9	10001001	00001100	41	11000100	11111011
10	10010101	11101001	42	00001111	10110111
11	10101101	01111001	43	00011110	01001010
12	11011101	11111100	44	00111100	00001001
13	00111101	01110010	45	01111000	01111111
14	01111010	11010000	46	11110000	00001000
15	11110100	10010001	47	01100111	01001110
16	01101111	10110100	48	11001110	10101110
17	11011110	00101000	49	00011011	10101000
18	00111011	01000100	50	00110110	01011100
19	01110110	10110011	51	01101100	01100000
20	11101100	11101101	52	11011000	00011110
21	01011111	11011110	53	00110111	00100111
22	10111110	00101011	54	01101110	11001111
23	11111011	00100110	55	11011100	10000111
24	01110001	11111110	56	00111111	11011101
25	11100010	00100001	57	01111110	01001001
26	01000011	00111011	58	11111100	01101011
27	10000110	10111011	59	01111111	00110010
28	10001011	10100011	60	11111110	11000100
29	10010001	01110000	61	01111011	10101011
30	10100101	10000011	62	11110110	00111110

[4] From table 4 of reference [E4]. Note: Coefficients of the 'Polynomial in Alpha' column are listed in descending powers of α, starting with α^7.

[5] The underlined entries correspond to values with exactly one non-zero element and match a row in the matrix.

Table F-1: Equivalence of Representations (continued)

POWER	POLY IN ALPHA	$\ell_{01234567}$	POWER	POLY IN ALPHA	$\ell_{01234567}$
63	01101011	00101101	95	10111010	10110010
64	11010110	11010010	96	11110011	11011100
65	00101011	11000010	97	01100001	01111000
66	01010110	01011111	98	11000010	11001101
67	10101100	00000010	99	00000011	11010100
68	11011111	01010011	100	00000110	00110110
69	00111001	11101011	101	00001100	01100011
70	01110010	00101010	102	00011000	01111100
71	11100100	00010111	103	00110000	01101010
72	01001111	01011000	104	01100000	00000011
73	10011110	11000111	105	11000000	01100010
74	10111011	11001001	106	00000111	01001101
75	11110001	01110011	107	00001110	11001100
76	01100101	11100001	108	00011100	11100101
77	11001010	00110111	109	00111000	10010000
78	00010011	01010010	110	01110000	10000101
79	00100110	11011010	111	11100000	10001110
80	01001100	10001100	112	01000111	10100010
81	10011000	11110001	113	10001110	01000001
82	10110111	10101010	114	10011011	00100101
83	11101001	00001111	115	10110001	10011100
84	01010101	10001011	116	11100101	01101100
85	10101010	00110100	117	01001101	11110111
86	11010011	00110000	118	10011010	01011110
87	00100001	10010111	119	10110011	00110011
88	01000010	01000000	120	11100001	11110101
89	10000100	00010100	121	01000101	00001101
90	10001111	00111010	122	10001010	11011000
91	10011001	10001010	123	10010011	11011111
92	10110101	00000101	124	10100001	00011010
93	11101101	10010110	125	11000101	10000000
94	01011101	01110001	126	00001101	00011000

Table F-1: Equivalence of Representations (continued)

POWER	POLY IN ALPHA	$\ell_{01234567}$	POWER	POLY IN ALPHA	$\ell_{01234567}$
127	00011010	11010011	159	10000101	01101111
128	00110100	11110011	160	10001101	10010101
129	01101000	11111001	161	10011101	00010011
130	11010000	11100100	162	10111101	11111111
131	00100111	10100001	163	11111101	00010000
132	01001110	00100011	164	01111101	10011101
133	10011100	01101000	165	11111010	01011101
134	10111111	01010000	166	01110011	01010001
135	11111001	10001001	167	11100110	10111000
136	01110101	01100111	168	01001011	11000001
137	11101010	11011011	169	10010110	00111101
138	01010011	10111101	170	10101011	01001111
139	10100110	01010111	171	11010001	10011111
140	11001011	01001100	172	00100101	00001110
141	00010001	11111101	173	01001010	10111010
142	00100010	01000011	174	10010100	10010010
143	01000100	01110110	175	10101111	11010110
144	10001000	01110111	176	11011001	01100101
145	10010111	01000110	177	00110101	10001000
146	10101001	11100000	178	01101010	01010110
147	11010101	00000110	179	11010100	01111101
148	00101101	11110100	180	00101111	01011011
149	01011010	00111100	181	01011110	10100101
150	10110100	01111110	182	10111100	10000100
151	11101111	00111001	183	11111111	10111111
152	01011001	11101000	184	01111001	00000100
153	10110010	01001000	185	11110010	10100111
154	11100011	01011010	186	01100011	11010111
155	01000001	10010100	187	11000110	01010100
156	10000010	00100010	188	00001011	00101110
157	10000011	01011001	189	00010110	10110000
158	10000001	11110110	190	00101100	10001111

Table F-1: Equivalence of Representations (continued)

POWER	POLY IN ALPHA	$\ell_{01234567}$	POWER	POLY IN ALPHA	$\ell_{01234567}$
191	01011000	10010011	223	01100100	10011010
192	10110000	11100111	224	11001000	10011000
193	11100111	11000011	225	00010111	11001011
194	01001001	01101110	226	00101110	00100000
195	10010010	10100100	227	01011100	00001010
196	10100011	10110101	228	10111000	00011101
197	11000001	00011001	229	11110111	01000101
198	00000101	11100010	230	01101001	10000010
199	00001010	01010101	231	11010010	01001011
200	00010100	00011111	232	00100011	00111000
201	00101000	00010110	233	01000110	11011001
202	01010000	01101001	234	10001100	11101110
203	10100000	01100001	235	10011111	10111100
204	11000111	00101111	236	10111001	01100110
205	00001001	10000001	237	11110101	11101010
206	00010010	00101001	238	01101101	00011011
207	00100100	01110101	239	11011010	10110001
208	01001000	00010101	240	00110011	10111110
209	10010000	00001011	241	01100110	00110101
210	10100111	00101100	242	11001100	00000001
211	11001001	11100011	243	00011111	00110001
212	00010101	01100100	244	00111110	10100110
213	00101010	10111001	245	01111100	11100110
214	01010100	11110000	246	11111000	11110010
215	10101000	10011011	247	01110111	11001000
216	11010111	10101001	248	11101110	01000010
217	00101001	01101101	249	01011011	01000111
218	01010010	11000110	250	10110110	11010001
219	10100100	11111000	251	11101011	10100000
220	11001111	11010101	252	01010001	00010010
221	00011001	00000111	253	10100010	11001110
222	00110010	11000101	254	11000011	10110110

ANNEX G

EXPANSION OF REED-SOLOMON COEFFICIENTS

(INFORMATIVE)

Purpose:

While the equations given in the R-S Coding section of this Recommended Standard are fully specifying, this annex provides additional assistance for those implementing either the $E = 16$ or the $E = 8$ code.

For $E = 16$:

COEFFICIENTS OF $g(x)$					**POLYNOMIAL IN α**							
					α^7	α^6	α^5	α^4	α^3	α^2	α^1	α^0
G_0	=	G_{32}	=	α^0	0	0	0	0	0	0	0	1
G_1	=	G_{31}	=	α^{249}	0	1	0	1	1	0	1	1
G_2	=	G_{30}	=	α^{59}	0	1	1	1	1	1	1	1
G_3	=	G_{29}	=	α^{66}	0	1	0	1	0	1	1	0
G_4	=	G_{28}	=	α^4	0	0	0	1	0	0	0	0
G_5	=	G_{27}	=	α^{43}	0	0	0	1	1	1	1	0
G_6	=	G_{26}	=	α^{126}	0	0	0	0	1	1	0	1
G_7	=	G_{25}	=	α^{251}	1	1	1	0	1	0	1	1
G_8	=	G_{24}	=	α^{97}	0	1	1	0	0	0	0	1
G_9	=	G_{23}	=	α^{30}	1	0	1	0	0	1	0	1
G_{10}	=	G_{22}	=	α^3	0	0	0	0	1	0	0	0
G_{11}	=	G_{21}	=	α^{213}	0	0	1	0	1	0	1	0
G_{12}	=	G_{20}	=	α^{50}	0	0	1	1	0	1	1	0
G_{13}	=	G_{19}	=	α^{66}	0	1	0	1	0	1	1	0
G_{14}	=	G_{18}	=	α^{170}	1	0	1	0	1	0	1	1
G_{15}	=	G_{17}	=	α^5	0	0	1	0	0	0	0	0
		G_{16}	=	α^{24}	0	1	1	1	0	0	0	1

NOTE – $G_3 = G_{29} = G_{13} = G_{19}$.

Further information, including encoder block diagrams, is provided in reference [E4].

For $E = 8$:

COEFFICIENTS OF *g*(*x*)					**POLYNOMIAL IN α**							
					α^7	α^6	α^5	α^4	α^3	α^2	α^1	α^0
G_0	=	G_{16}	=	α^0	0	0	0	0	0	0	0	1
G_1	=	G_{15}	=	α^{30}	1	0	1	0	0	1	0	1
G_2	=	G_{14}	=	α^{230}	0	1	1	0	1	0	0	1
G_3	=	G_{13}	=	α^{49}	0	0	0	1	1	0	1	1
G_4	=	G_{12}	=	α^{235}	1	0	0	1	1	1	1	1
G_5	=	G_{11}	=	α^{129}	0	1	1	0	1	0	0	0
G_6	=	G_{10}	=	α^{81}	1	0	0	1	1	0	0	0
G_7	=	G_9	=	α^{76}	0	1	1	0	0	1	0	1
		G_8	=	α^{173}	0	1	0	0	1	0	1	0