Consultative Committee for Space Data Systems

RECOMMENDATION FOR SPACE DATA SYSTEM STANDARDS

TM SYNCHRONIZATION AND CHANNEL CODING

CCSDS 131.0-B-1

BLUE BOOK

September 2003

AUTHORITY

Issue:	Blue Book, Issue 1
Date:	September 2003
Location:	Not Applicable

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 Recommendations is detailed in the *Procedures Manual for the Consultative Committee for Space Data Systems (*reference [B1]), and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the address below.

This Recommendation is published and maintained by:

CCSDS Secretariat
Office of Space Communication (Code M-3)
National Aeronautics and Space Administration
Washington, DC 20546, USA

STATEMENT OF INTENT

The Consultative Committee for Space Data Systems (CCSDS) is an organization officially established by the management of member space Agencies. 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 **Recommendations** and are not considered binding on any Agency.

This **Recommendation** is issued by, and represents the consensus of, the CCSDS Plenary body. Agency endorsement of this **Recommendation** is entirely voluntary. Endorsement, however, indicates the following understandings:

– Whenever an Agency establishes a CCSDS-related **standard**, this **standard** will be in accord with the relevant **Recommendation**. Establishing such a **standard** does not preclude other provisions which an Agency may develop.

– Whenever an Agency establishes a CCSDS-related standard, the Agency will provide other CCSDS member Agencies with the following information:

  - The **standard** itself.
  - The anticipated date of initial operational capability.
  - The anticipated duration of operational service.

– Specific service arrangements are made via memoranda of agreement. Neither this Recommendation nor any ensuing standard is a substitute for a memorandum of agreement.

No later than five years from its date of issuance, this **Recommendation** 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 **Recommendation** is issued, existing CCSDS-related Agency standards and implementations are not negated or deemed to be non-CCSDS compatible. It is the responsibility of each Agency to determine when such standards or implementations are to be modified. Each Agency is, however, strongly encouraged to direct planning for its new standards and implementations towards the later version of the Recommendation.

FOREWORD

This document is a technical Recommendation 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.

This Recommendation establishes a common framework and provides a common basis for the synchronization and channel coding schemes to be used by space missions with the TM or AOS Space Data Link Protocol (references [1] or [2]) over space-to-ground and space-to-space communications links. This Recommendation was developed by consolidating the specifications regarding synchronization and channel coding in older CCSDS Recommendations [B2] and [B3].

This Recommendation does not change the major technical contents defined in [B2] and [B3], but the presentation of the specification has been changed so that:

a) these schemes can be used to transfer any data over any space link in either direction;

b) all CCSDS space link protocols are specified in a unified manner;

c) the layered model matches the Open Systems Interconnection (OSI) Basic Reference Model (reference [3]).

Together with the change in presentation, a few technical specifications in [B2] and [B3] have been changed in order to define all Space Data Link Protocols in a unified way. Also, some technical terms in references [B2] and [B3] have been changed in order to unify the terminology used in all the CCSDS Recommendations that define space link protocols and to define these schemes as general communications schemes. These changes are listed in annex E of this Recommendation.

Through the process of normal evolution, it is expected that expansion, deletion or modification to this document may occur. This Recommendation is therefore subject to CCSDS document management and change control procedures, as defined in the *Procedures Manual for the Consultative Committee for Space Data Systems*. 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 addressed to the CCSDS Secretariat at the address indicated on page i.

DOCUMENT CONTROL

Document	**Title and Issue**	**Date**	**Status**
CCSDS 131.0-B-1	TM Synchronization and Channel Coding, Issue 1	September 2003	Original Issue

CONTENTS

CONTENTS (continued)

CONTENTS (continued)

1 INTRODUCTION

1.1 PURPOSE

The purpose of this Recommendation is to specify synchronization and channel coding schemes used with the TM Space Data Link Protocol (reference [1]) or the AOS Space Data Link Protocol (reference [2]). These schemes are to be used over space-to-ground or space-to-space communications links by space missions.

1.2 SCOPE

This Recommendation 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 Recommendation applies to the creation of Agency standards and to the future data communications over space links between CCSDS Agencies in cross-support situations. This Recommendation 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 Recommendation 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 Recommendation is anticipated. Where mandatory capabilities are clearly indicated in sections of this Recommendation, 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 nine numbered sections and six annexes:

a) section 1 presents the purpose, scope, applicability and rationale of this Recommendation 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 specifies the convolutional coding;

d) section 4 specifies the Reed-Solomon coding;

e) section 5 specifies the turbo coding;

f) section 6 specifies the frame synchronization scheme;

g) section 7 specifies the Pseudo-Randomizer;

h) section 8 specifies the allowed lengths of Transfer Frames;

i) section 9 lists the managed parameters associated with synchronization and channel coding;

j) annex A lists acronyms and terms used within this document;

k) annex B provides a list of informative references;

l) annex C defines the service provided to the users;

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

n) annex E provides information on Reed-Solomon coefficients;

o) annex F lists the changes from relevant previously published CCSDS Recommendations [B2] and [B3].

1.6 CONVENTIONS AND DEFINITIONS

1.6.1 DEFINITIONS

1.6.1.1 Definitions from the Open System Interconnection (OSI) Basic Reference Model

This Recommendation makes use of a number of terms defined in reference [3]. The use of those terms in this Recommendation 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 Recommendation makes use of a number of terms defined in reference [4]. The use of those terms in this Recommendation 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 Terms Defined in This Recommendation

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

asynchronous: not *synchronous*.

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.

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

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.

synchronous: of or pertaining to a sequence of events occurring in a fixed time relationship (within specified tolerance) to another sequence of events.

1.6.2 NOMENCLATURE

The following conventions apply throughout this Recommendation:

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.

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).

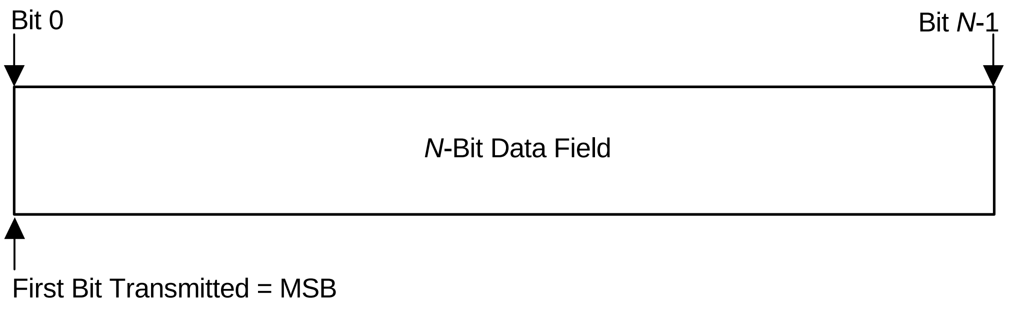


Figure 1-1: Bit Numbering Convention

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

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

1.7 REFERENCES

The following documents contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All documents are subject to revision, and users of this Recommendation 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 Recommendations.

[1] *TM Space Data Link Protocol.* Recommendation for Space Data Systems Standards, CCSDS 132.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, September 2003.

[2] *AOS Space Data Link Protocol.* Recommendation for Space Data Systems Standards, CCSDS 732.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, September 2003.

[3] *Information Technology—Open Systems Interconnection—Basic Reference Model: The Basic Model*. International Standard, ISO/IEC 7498-1. 2nd ed.. 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.* Recommendation for Space Data Systems Standards, CCSDS 401.0-B. Blue Book. Washington, D.C.: CCSDS, March 2003.

NOTE – Informative references are listed in annex B.

2 OVERVIEW

2.1 ARCHITECTURE

Figure 2-1 illustrates the relationship of this Recommendation to the Open Systems Interconnection reference model (reference [3]). Two sublayers of the Data Link Layer are defined for CCSDS space link protocols. The TM and AOS Space Data Link Protocols specified in references [1] and [2], respectively, correspond to the Data Link Protocol Sublayer, and provide functions for transferring data using the protocol data unit called the Transfer Frame. 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.

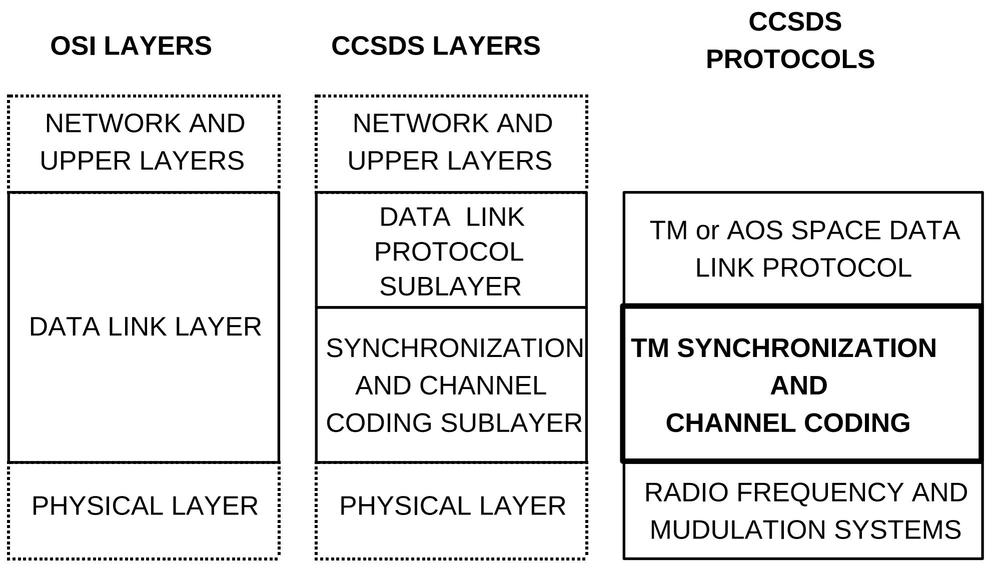

OSI LAYERS
CCSDS LAYERS
CCSDS PROTOCOLS
NETWORK AND UPPER LAYERS
NETWORK AND UPPER LAYERS
DATA LINK LAYER
DATA LINK PROTOCOL SUBLAYER
SYNCHRONIZATION AND CHANNEL CODING SUBLAYER
TM or AOS SPACE DATA LINK PROTOCOL
TM SYNCHRONIZATION AND CHANNEL CODING
PHYSICAL LAYER
PHYSICAL LAYER
RADIO FREQUENCY AND MUDULATION SYSTEMS

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 three functions for transferring Transfer Frames over a space link:

a) error-control coding, including frame validation (optional);

b) synchronization; and

c) pseudo-randomizing (optional).

2.2.2 ERROR-CONTROL CODING

This Recommendation specifies the following three types of error-control codes:

a) convolutional codes (section 3);

b) Reed-Solomon codes (section 4);

c) turbo codes (section 5).

One of the convolutional codes described in section 3 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 3.1, the punctured convolutional code specified in 3.2 has the advantage of smaller bandwidth expansion.

For Physical Channels which are bandwidth-constrained and cannot tolerate the increase in bandwidth required by the convolutional codes, the Reed-Solomon codes specified in section 4 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 Reed-Solomon code alone, a concatenation of a convolutional code as the inner code with a Reed-Solomon code as the outer code may be used for improved performance.

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

NOTE – In this Recommendation, 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.

Some codes are also used to check whether or not each decoded Transfer Frame can be used as a valid data unit by the upper layers at the receiving end. This function is called Frame Validation. The Reed-Solomon decoder can determine, with a very high probability, whether or not it can correctly decode a Transfer Frame. Therefore, the Reed-Solomon code is also used for Frame Validation. When the Reed-Solomon code is not used, the Frame Error Control Field defined in references [1] or [2] shall be used for Frame Validation.

NOTE – Frame Validation explained above presupposes correct frame synchronization. That is, a Transfer Frame declared valid by the Reed-Solomon decoder is valid only if the Transfer Frame is correctly synchronized. If a Transfer Frame is not correctly synchronized, then the Reed-Solomon decoder may perform meaningless error correction against the Transfer Frame and declare it valid.

2.2.3 SYNCHRONIZATION

This Recommendation specifies a method for synchronizing Transfer Frames using an Attached Sync Marker (ASM) (see section 6).

The ASM 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.

2.2.4 PSEUDO-RANDOMIZING

This Recommendation specifies an optional pseudo-randomizer to improve symbol transition density as an aid to bit synchronization (see section 7).

2.3 INTERNAL ORGANIZATION OF SUBLAYER

2.3.1 SENDING END

Figure 2-2 shows the internal organization of the Synchronization and Channel Coding Sublayer of the sending 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. Depending on the options actually 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, and delivers a continuous and contiguous stream of channel symbols to the Physical Layer.

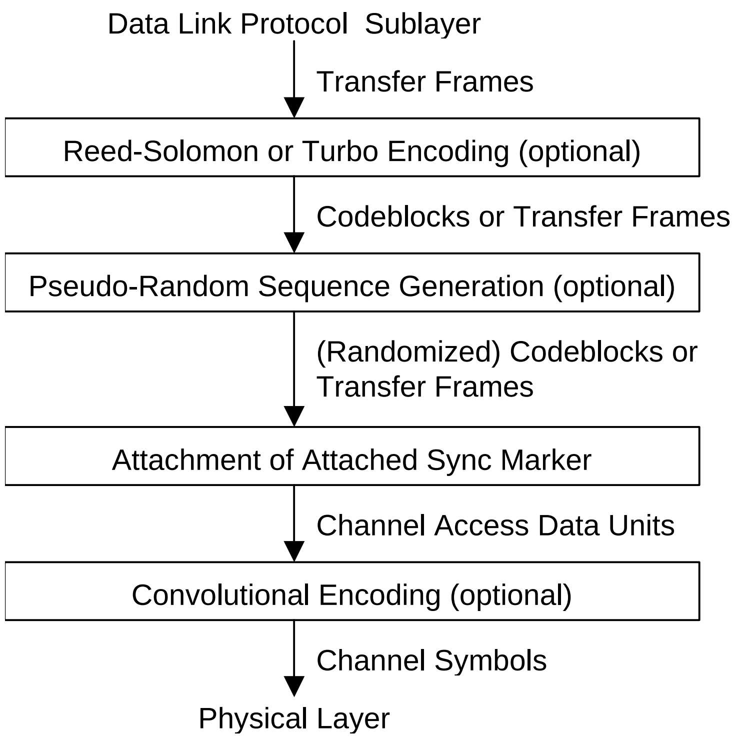

Data Link Protocol Sublayer
Transfer Frames
Reed-Solomon or Turbo Encoding (optional)
Codeblocks or Transfer Frames
Pseudo-Random Sequence Generation (optional)
(Randomized) Codeblocks or Transfer Frames
Attachment of Attached Sync Marker
Channel Access Data Units
Convolutional Encoding (optional)
Channel Symbols
Physical Layer

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

2.3.2 RECEIVING END

Figure 2-3 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. Depending on the options actually 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 channel symbols from the Physical Layer, performs functions selected for the mission, and delivers Transfer Frames to the Data Link Protocol Sublayer.

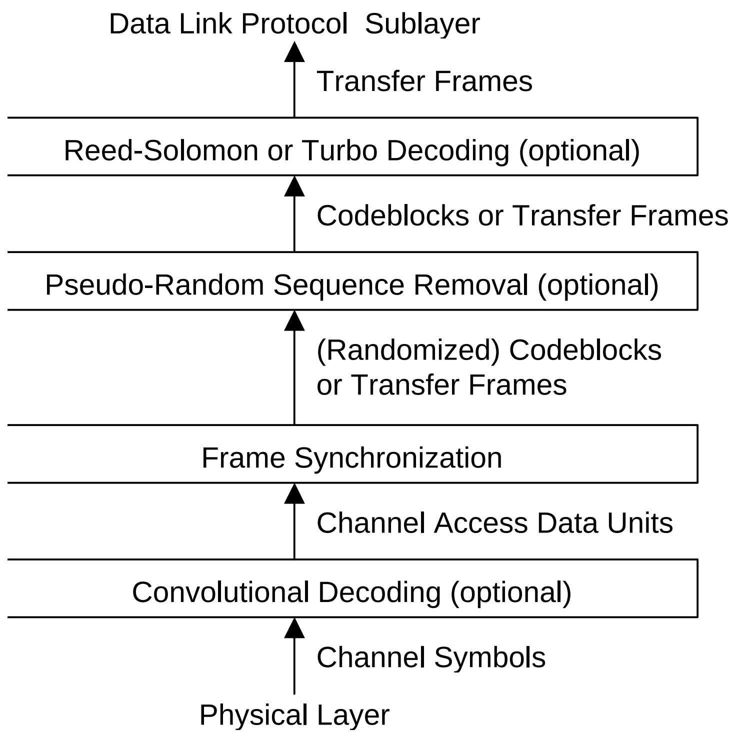

Data Link Protocol Sublayer
Transfer Frames
Reed-Solomon or Turbo Decoding (optional)
Codeblocks or Transfer Frames
Pseudo-Random Sequence Removal (optional)
(Randomized) Codeblocks or Transfer Frames
Frame Synchronization
Channel Access Data Units
Convolutional Decoding (optional)
Channel Symbols
Physical Layer

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

3 CONVOLUTIONAL CODING

3.1 BASIC CONVOLUTIONAL CODE

3.1.1 BASIC CONVOLUTIONAL CODE DESCRIPTION

3.1.1.1 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 3.1.2. When this code is punctured according to 3.2, higher code rates (lower overhead) may be achieved, although with somewhat lower error correcting performance.

3.1.1.2 The convolutional decoder is a maximum-likelihood (Viterbi) decoder.

NOTES

1 Basic convolutional code, by itself, cannot guarantee sufficient symbol transitions when multiplexing schemes are used, e.g., those implemented in Quadrature Phase Shift Keying (QPSK). Therefore, the Pseudo-Randomizer defined in section 7 is required unless the system designer verifies that sufficient symbol transition density is assured by other means when the Randomizer is not used.

2 If the decoder's correction capability is exceeded, undetected burst errors may appear in the output. For this reason, when TM or AOS Transfer Frames are used, the Frame Error Control Field (FECF) specified in references [1] and [2] is required to validate the Transfer Frame unless the Reed-Solomon code is used (see section 4).

3.1.1.3 It is recommended that soft bit decisions with at least 3-bit quantization be used whenever constraints (such as location of decoder) permit.

3.1.2 BASIC CONVOLUTIONAL CODE SPECIFICATION

3.1.2.1 This recommended basic convolutional code is a non-systematic code and a specific decoding procedure, with the following characteristics:

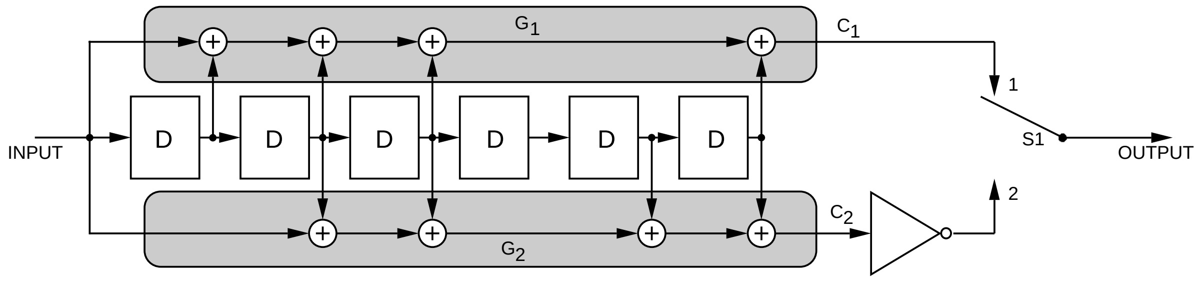
(1) Nomenclature:	Convolutional code with maximum-likelihood (Viterbi) decoding.
(2) Code rate (*r*):	1/2 bit per symbol.
(3) Constraint length (*K*):	7 bits.
(4) Connection vectors:	G1 = 1111001 (171 octal); G2= 1011011(133 octal).
(5) Symbol inversion:	On output path of G2.

NOTE – An encoder block diagram is shown in figure 3-1.

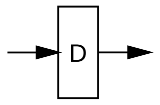
3.1.2.2 The output symbol sequence is: $C_1(1), \overline{C_2(1)}, C_1(2), \overline{C_2(2)}. . .$

3.1.2.3 When suppressed-carrier modulation systems are used, Non-Return-to-Zero-Mark (NRZ-M) or Non-Return-to-Zero-Level (NRZ-L) may be used as a modulating waveform. If the user contemplates conversion of his modulating waveform from NRZ-L to NRZ-M, such conversion should be performed on-board at the input to the convolutional encoder. Correspondingly, the conversion on the ground from NRZ-M to NRZ-L should be performed at the output of the convolutional decoder. This avoids unnecessary link performance loss.

3.1.2.4 When a fixed pattern (the fixed part of the convolutionally encoded Attached Sync Marker) in the symbol stream is used to provide node synchronization for the Viterbi decoder, care must be taken to account for any modification of the pattern resulting from the modulating waveform conversion.



NOTES:

1.  = 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.

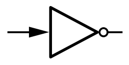
5.  = INVERTER.

Figure 3-1: Basic Convolutional Encoder Block Diagram

3.2 PUNCTURED CONVOLUTIONAL CODES

3.2.1 GENERAL

The code rate (*r*=1/2), constraint length (*k*=7) convolutional code can be modified to achieve an increase in bandwidth efficiency. This modification is achieved by using a puncture pattern P(*r*). Puncturing removes some of the symbols before transmission, providing lower overhead and lower bandwidth expansion than the original code, but with slightly reduced error correcting performance.

3.2.2 PUNCTURED CONVOLUTIONAL CODES DESCRIPTION

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. Figure 3-2 depicts the punctured encoding scheme.

NOTE – The symbol inverter associated with G2 in the rate 1/2 code (defined in 3.1.2) is omitted here. Therefore, the Pseudo-Randomizer defined in section 7 is required unless the system designer verifies that sufficient symbol transition density is assured by other means when the Randomizer is not used.

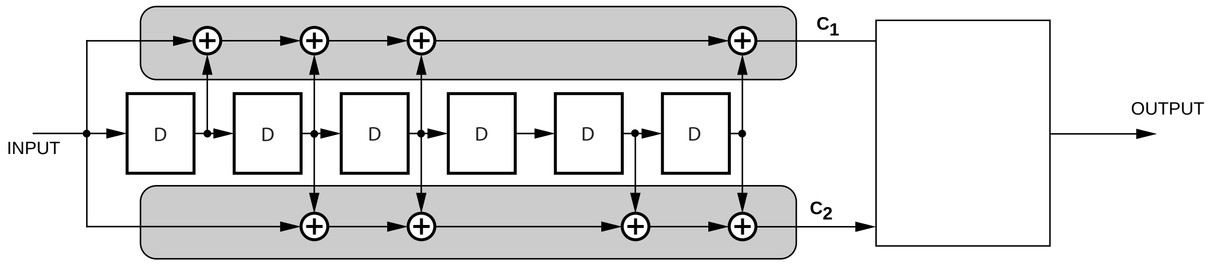


Figure 3-2: Punctured Encoder Block Diagram

3.2.3 PUNCTURED CONVOLUTIONAL CODES SPECIFICATION

3.2.3.1 The punctured convolutional code has the following characteristics:

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

3.2.3.2 The puncturing patterns for each of the punctured convolutional code rates are defined by table 3-1.

Table 3-1: 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	$C_1(1)$ $C_2(1)$ $C_2(2)$ $C_2(3)$ $C_2(4)$ $C_1(5)$ $C_2(6)$ $C_1(7)$ ...

4 REED-SOLOMON CODING

4.1 INTRODUCTION

4.1.1 The Reed-Solomon (R-S) code defined in this section is a powerful burst error correcting code. In addition, the code chosen has an extremely low undetected error rate. This means that the decoder can reliably indicate whether or not it can make the proper corrections. To achieve this reliability, proper codeblock synchronization is mandatory.

4.1.2 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 two options shall not be mixed in a single Physical Channel.

NOTES

1 The extremely low undetected error rate of this code means that the R-S decoder can, with a high degree of certainty, validate the decoded codeblock and consequently the contained TM Transfer Frame (reference [1]) or AOS Transfer Frame (reference [2]). For this reason, the Frame Error Control Field (FECF) specified in references [1] and [2] is not required when this Reed-Solomon Code is used (see section 1).

2 The Reed-Solomon coding, by itself, cannot guarantee sufficient channel symbol transitions to keep receiver symbol synchronizers in lock. Therefore, the Pseudo-Randomizer defined in section 7 is required unless the system designer verifies that sufficient symbol transition density is assured by other means when the Randomizer is not used.

4.1.3 The Reed-Solomon code may be used alone, and as such it provides an excellent forward error correction capability in a burst-noise channel. However, should the Reed-Solomon code alone not provide sufficient coding gain, it may be concatenated with the convolutional code defined in section 3. Used this way, the Reed-Solomon code is the *outer code*, while the convolutional code is the *inner code*.

4.2 SPECIFICATION

The parameters of the selected Reed-Solomon (R-S) code are as follows:

a) J = 8 bits per R-S symbol.

b) E = Reed-Solomon error correction capability, in symbols, within a R-S codeword. E may be selected to be 16 or 8 R-S symbols.

c) General characteristics of Reed-Solomon codes:

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

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

3) $2E$ is the number of R-S symbols among n symbols of an R-S codeword representing parity checks.

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

d) Field generator polynomial:

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

over GF(2).

e) Code generator polynomial:

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

over GF(2^8), where F(α) = 0.

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) Reed-Solomon code when E = 16 and a (255,239) Reed-Solomon code when E = 8.

f) The selected code is a systematic code. This results in a systematic codeblock.

g) Symbol interleaving:

The allowable values of interleaving depth are I=1, 2, 3, 4, 5, and 8. I=1 is equivalent to the absence of interleaving. The interleaving depth shall normally be fixed on a Physical Channel for a Mission Phase. Symbol interleaving is accomplished in a manner functionally described with the aid of figure 4-1. (It should be noted that this functional description does not necessarily correspond to the physical implementation of an encoder.)

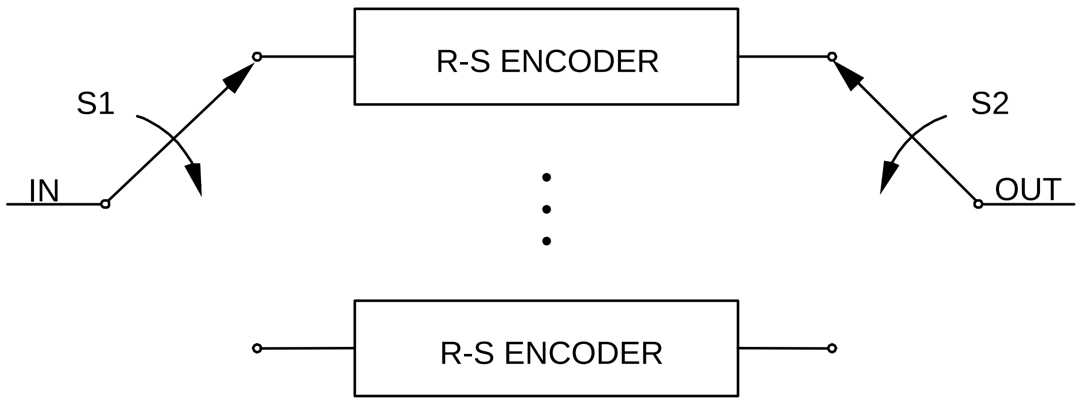


Figure 4-1: Functional Representation of R-S Interleaving

Data bits to be encoded into a single Reed-Solomon 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.

h) Maximum codeblock length:

The maximum codeblock length, in R-S symbols, is given by:

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

i) Shortened codeblock length:

A shortened codeblock length may be used to accommodate frame lengths smaller than the maximum. However, since the Reed-Solomon code is a block code, the decoder must always operate on a full block basis. To achieve a full codeblock, ‘virtual fill’ must be added to make up the difference between the shortened block and the maximum codeblock length. The characteristics and limitations of virtual fill are covered in subsection 6.2(j). Since the virtual fill is not transmitted, both encoder and decoder must be set to insert it with the proper length for the encoding and decoding processes to be carried out properly.

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*), 2*EI* check symbols are computed over *kI* symbols, of which the leading *Q* symbols are treated as all-zero symbols. A (*nI*–*Q*, *kI*–*Q*) shortened codeblock results where the leading *Q* symbols (all zeros) are neither entered into the encoder nor transmitted.

NOTE – It should be noted that 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.

j) Reed-Solomon codeblock partitioning and virtual fill:

The R-S codeblock is partitioned as shown in figure 4-2.

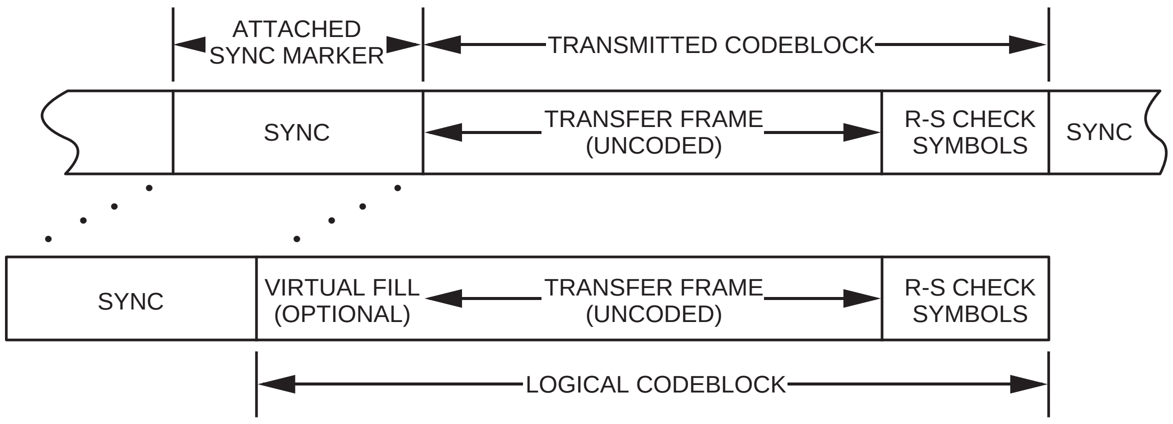


Figure 4-2: Reed-Solomon Codeblock Partitioning

The **Reed-Solomon Check Symbols** consist of the trailing 2*EI* symbols (2*EIJ* bits) of the codeblock. (As an example, when *E* = 16 and *k* = 223, for *I*=5 this is always 1280 bits.)

The **Transfer Frame** is defined by the TM Space Data Link Protocol (reference [1]) or the AOS Space Data Link Protocol (reference [2]). For constraints on the length of the Transfer Frame, see section 8.

The **Attached Sync Marker** used with R-S coding is a 32-bit pattern specified in section 6 as an aid to synchronization. It precedes the Transmitted Codeblock. Frame synchronizers should, therefore, be set to expect a marker at every Transmitted Codeblock + 32 bits.

The **Transmitted Codeblock** consists of the Transfer Frame (without the 32-bit sync marker) and R-S check symbols. It 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.)

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.)

Virtual fill is used to logically complete the codeblock and is not transmitted. If used, virtual fill shall:

1) consist of all zeros;
2) not be transmitted;
3) not change in length for a Mission Phase on a particular Physical Channel;
4) be inserted only at the beginning of the codeblock (i.e., after the attached sync marker but before the beginning of the transmitted codeblock);
5) be inserted only in integer multiples of 8*I* bits.

k) Dual basis symbol representation and ordering for transmission:

Each 8-bit Reed-Solomon 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, α^1, α^2, . . ., α^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, β^1, β^2, . . ., β^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 Reed-Solomon 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.

The representation used in this Recommendation is the dual basis eight-bit string $z_0, z_1, \ldots, z_7$, transmitted in that order (i.e., with z_0 first). The relationship between the two representations is given by the 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

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

Further information relating the dual basis (Berlekamp) and conventional representations is given in annex C. 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.

l) Synchronization:

Codeblock synchronization of the Reed-Solomon decoder is achieved by synchronization of the Attached Sync Marker associated with each codeblock. (See section 6.)

m) Ambiguity resolution:

The ambiguity between true and complemented data must be resolved so that only true data is provided to the Reed-Solomon decoder. Data in NRZ-L form is normally resolved using the 32-bit Attached Sync Marker, while NRZ-M data is self-resolving.

5 TURBO CODING

5.1 INTRODUCTION

5.1.1 Turbo codes are binary block codes with large code blocks (hundreds or thousands of bits). They are systematic and inherently non-transparent. Phase ambiguities are resolved using Attached Sync Markers (ASMs), which are required for Codeblock synchronization.

5.1.2 Turbo codes may be used to obtain even greater coding gain than those provided by concatenated coding systems.

NOTES

1 Turbo coding, by itself, cannot guarantee sufficient bit transitions to keep receiver symbol synchronizers in lock. Therefore, the Pseudo-Randomizer defined in section 7 is required unless the system designer verifies that sufficient symbol transition density is assured by other means when the Randomizer is not used.

2 While providing outstanding coding gain, turbo codes may still leave some residual errors in the decoded output. For this reason, when TM or AOS Transfer Frames are used, the Frame Error Control Field (FECF) specified in references [1] or [2], respectively, is required to validate the Transfer Frame unless the Reed-Solomon code is used. (See section 1.)

3 Differential encoding (i.e., NRZ-M signaling) after the turbo encoder is not recommended since soft decoding would require the use of differential detection with considerable loss of performance. Differential encoding before the turbo encoder cannot be used because the turbo codes recommended in this document are non-transparent. This implies that phase ambiguities have to be detected and resolved by the frame synchronizer.

4 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. 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

5.2 SPECIFICATION

5.2.1 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.

5.2.2 The recommended turbo code is a systematic code with the following specifications:

a)	Code type:	Systematic parallel concatenated turbo code.
b)	Number of component codes:	2 (plus an uncoded component to make the code systematic).
c)	Type of component codes:	Recursive convolutional codes.
d)	Number of states of each convolutional component code:	16.
e)	Nominal code rates:	r = 1/2, 1/3, 1/4, or 1/6 (selectable).

NOTE – Because of 'trellis termination' symbols (see subsection 5.2(j)), the true code rates (defined as the ratios of the information block lengths to the codeblock lengths in table 5-2) are slightly smaller than the nominal code rates. In this Recommendation, the term 'code rate' always refers to the nominal code rates, r = 1/2, 1/3, 1/4, or 1/6.

f) The specified information block lengths *k* are shown in table 5-1. They are chosen for compatibility with the corresponding Reed-Solomon interleaving depths, also shown in table 7-1.

The corresponding codeblock lengths in bits, $n=(k+4)/r$, for the specified code rates are shown in table 5-2.

Table 5-1: 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	
16384	Not Applicable	For highest coding gain

Table 5-2: Codeblock Lengths for Supported Code Rates (Measured in Bits)

Information block length *k*	Codeblock 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
16384	32776	49164	65552	98328

g) Turbo code permutation:

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 Reed-Solomon codes, the turbo code permutation scrambles individual bits and resembles a randomly selected permutation in its lack of apparent orderliness.

The recommended permutation for each specified block length k is given by a particular reordering of the integers 1, 2, . . ., k as generated by the following algorithm.

First express k as $k=k_1k_2$. The parameters k_1 and k_2 for the specified block sizes are given in table 5-3.

Next do the following operations for s=1 to s=k to obtain permutation numbers $\pi(s)$. 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:

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

Table 5-3: 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
16384	(NOTE)	(NOTE)

NOTE – These parameters are currently under study and will be incorporated in a later revision.

$$m = (s - 1) \bmod 2$$

$$i = \left\lfloor \frac{s-1}{2\,k_2} \right\rfloor$$

$$j = \left\lfloor \frac{s-1}{2} \right\rfloor - i\,k_2$$

$$t = (19i + 1) \bmod \frac{k_1}{2}$$

$$q = t \bmod 8 + 1$$

$$c = (p_q\,j + 21m) \bmod k_2$$

$$\pi(s) = 2\left(t + c\,\frac{k_1}{2} + 1\right) - m$$

The interpretation of the permutation numbers is such that the *s*th bit read out on line ‘in b’ in figure 5-2 is the $\pi(s)$th bit of the input information block, as shown in figure 5-1.

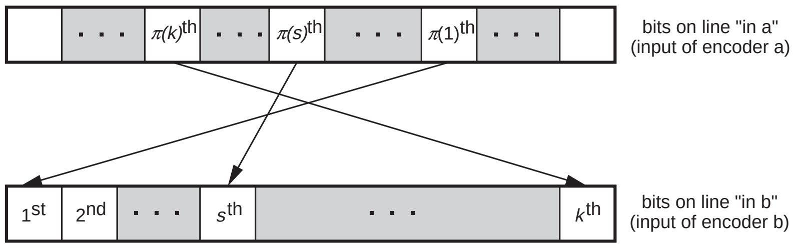


Figure 5-1: Interpretation of Permutation

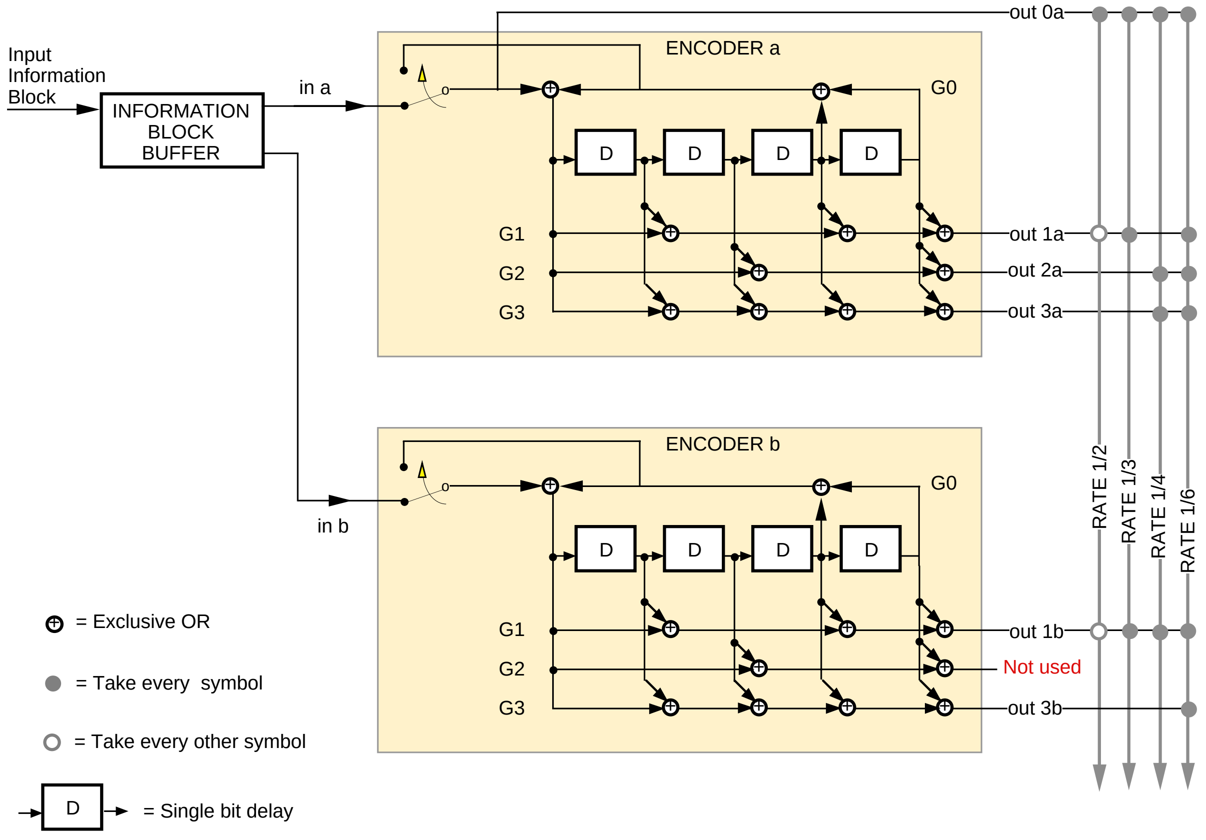


Figure 5-2: Turbo Encoder Block Diagram

h) Backward and forward connection vectors (see figure 5-2):

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

2) Forward connection vector for both component codes and rates 1/2 and 1/3: G1 = 11011. Puncturing of every other symbol from each component code is necessary for rate 1/2. No puncturing is done for rate 1/3.

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

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

i) Turbo encoder block diagram:

The recommended encoder block diagram is shown in figure 7-2. Each input frame of *k* information bits is held in a frame buffer, and the bits in the buffer are read out in two different orders for the two component encoders. The first component encoder (a) operates on the bits in unpermuted order (‘in a’), while the second component encoder (b) receives the same bits permuted by the interleaver (‘in b’). 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 subsection 7.2(g).

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 subsection 7.2(h). A key difference between these convolutional component encoders and the standalone convolutional encoder recommended in section 3 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.

j) Turbo codeblock specification:

Both component encoders in figure 5-2 are initialized with 0s in all registers, and both are run for a total of k+4 bit times, producing an output Codeblock of (k+4)/r encoded symbols, where r is the nominal code rate. For the first k bit times, the input switches are in the lower position (as indicated in the figure) to receive input data. For the final 4 bit times, these switches move to the upper position to receive feedback from the shift registers. 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 4 bit times. Filling the registers with zeros is called terminating the trellis. During trellis termination the encoder continues to output nonzero encoded symbols. In particular, the ‘systematic uncoded’ output (line ‘out 0a’ in the figure) includes an extra 4 bits from the feedback line in addition to the k information bits.

In figure 5-2, the encoded symbols are multiplexed from top-to-bottom along the output line for the selected code rate to form the Turbo Codeblock. 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. Note that this pattern implies that out 1b is the first to be punctured, out 1a is the second, and so forth. The Turbo Codeblocks constructed from these output sequences are depicted in figure 5-3 for the four nominal code rates.

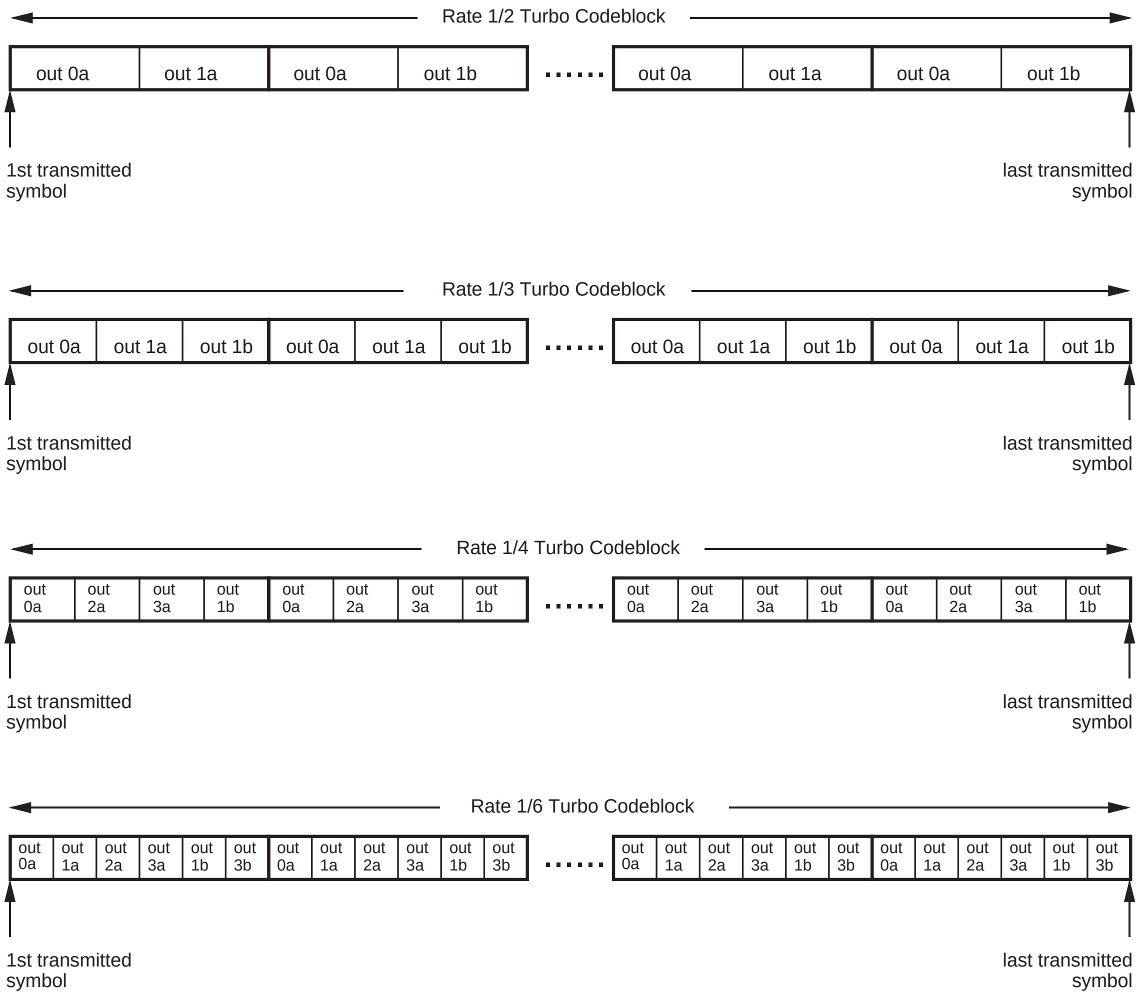


Figure 5-3: Turbo Codeblocks for Different Code Rates

k) Turbo codeblock synchronization:

Codeblock synchronization of the turbo decoder is achieved by synchronization of an Attached Sync Marker (ASM) associated with each Turbo Codeblock. The ASM is a bit pattern specified in section 6 as an aid to synchronization, and it precedes the Turbo Codeblock. Frame synchronizers should be set to expect a marker at a recurrence interval equal to the length of the ASM plus that of the Turbo Codeblock.

A diagram of a Turbo Codeblock with ASM is shown in figure 5-4. Note that the length of the Turbo Codeblock is inversely proportional to the nominal code rate *r*.

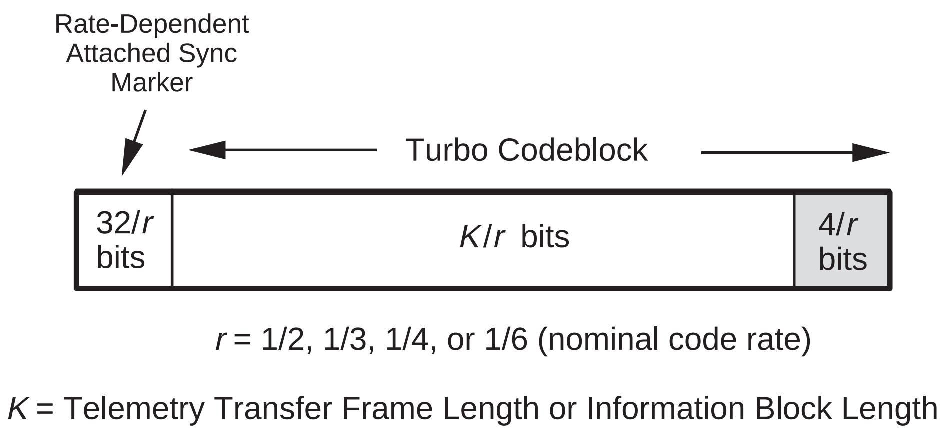

Rate-Dependent Attached Sync Marker
Turbo Codeblock
32/r bits
K/r bits
4/r bits
r = 1/2, 1/3, 1/4, or 1/6 (nominal code rate)
K = Telemetry Transfer Frame Length or Information Block Length

Figure 5-4: Turbo Codeblock with Attached Sync Marker

6 FRAME SYNCHRONIZATION

6.1 INTRODUCTION

Frame or Codeblock synchronization is necessary for proper decoding of Reed-Solomon Codeblocks and Turbo Codeblocks, and subsequent processing of the Transfer Frames. Furthermore, it is necessary for synchronization of the pseudo-random generator, if used (see section 7). It is also useful in assisting the node synchronization process of the Viterbi decoder for the convolutional code.

6.2 THE ATTACHED SYNC MARKER (ASM)

6.2.1 GENERAL

Synchronization of the Reed-Solomon or Turbo Codeblock (or Transfer Frame, if the Physical Channel is not Reed-Solomon coded or turbo coded) is achieved by using a stream of fixed-length Codeblocks (or Transfer Frames) with an Attached Sync Marker (ASM) between them. Synchronization is acquired on the receiving end by recognizing the specific bit pattern of the ASM in the Physical Channel data stream; synchronization is then customarily confirmed by making further checks.

6.2.2 ENCODER SIDE

6.2.2.1 If the Physical Channel is not Reed-Solomon coded or turbo coded, the code symbols composing the ASM are attached directly to the Transfer Frame.

6.2.2.2 If the Physical Channel is Reed-Solomon coded or turbo coded, the code symbols composing the ASM are attached directly to the encoder output without being encoded by the Reed-Solomon or turbo code. If an inner convolutional code is used in conjunction with an outer Reed-Solomon code, the ASM is encoded by the inner code but not by the outer code. (See section 3.)

6.2.2.3 The data unit that consists of the ASM and the Transfer Frame (if the Physical Channel is not Reed-Solomon coded or turbo coded) or the Reed-Solomon or Turbo Codeblock (if the Physical Channel is Reed-Solomon coded or turbo coded) is called the Channel Access Data Unit (CADU).

6.2.3 DECODER SIDE

For a concatenated Reed-Solomon and convolutional coding system, the ASM may be acquired either in the channel symbol domain (i.e., before any decoding) or in the domain of bits decoded by the inner code (i.e., the code symbol domain of the Reed-Solomon code). For a turbo coding system, the ASM must be acquired in the channel symbol domain (i.e., the code symbol domain of the turbo code).

6.3 ASM BIT PATTERNS

The ASM for data that is not turbo coded shall consist of a 32-bit (4-octet) marker with a pattern shown in figure 6-1. The ASM 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 6-2 through 6-5. The ASM bit patterns are represented in hexadecimal notation as:

ASM for non-turbo-coded data:	1ACFFC1D
ASM for rate-1/2 turbo coded data:	034776C7272895B0
ASM for rate-1/3 turbo coded data:	25D5C0CE8990F6C9461BF79C
ASM for rate-1/4 turbo coded data:	034776C7272895B0 FCB88938D8D76A4F
ASM for rate-1/6 turbo coded data:	25D5C0CE8990F6C9461BF79C DA2A3F31766F0936B9E40863

```
FIRST TRANSMITTED BIT
(Bit 0)
↓
0001   1010   1100   1111   1111   1100   0001   1101
                                                    ↑
                               LAST TRANSMITTED BIT
                                            (Bit 31)
```

Figure 6-1: ASM Bit Pattern for Non-Turbo-Coded Data

```
FIRST TRANSMITTED BIT
(Bit 0)
↓
0000001101000111011101101100011100100111001010001001010110110000
                                                               ↑
                                           LAST TRANSMITTED BIT
                                                       (Bit 63)
```

Figure 6-2: ASM Bit Pattern for Turbo-Coded Data (for Rate 1/2 Turbo Code)

```
FIRST TRANSMITTED BIT
(Bit 0)
↓
001001011101010111000000110011101000100110010000111101101100100101000110000110111111011110011100
                                                                                               ↑
                                                                           LAST TRANSMITTED BIT
                                                                                       (Bit 95)
```

Figure 6-3: ASM Bit Pattern for Turbo-Coded Data (for Rate 1/3 Turbo Code)

```
FIRST TRANSMITTED BIT
(Bit 0)
↓
 0000001101000111011101101100011100100111001010001001010110110000
 1111110010111000100010010011100011011000110101110110101001001111
                                                                 ↑
                                             LAST TRANSMITTED BIT
                                                        (Bit 127)
```

Figure 6-4: ASM Bit Pattern for Turbo-Coded Data (for Rate 1/4 Turbo Code)

```
FIRST TRANSMITTED BIT
(Bit 0)
↓
 001001011101010111000000110011101000100110010000111101101100100101000110000110111111011110011100
 110110100010101000111111001100010111011001101111000010010011011010111001111001000000100001100011
                                                                                                 ↑
                                                                             LAST TRANSMITTED BIT
                                                                                        (Bit 191)
```

Figure 6-5: ASM Bit Pattern for Turbo-Coded Data (for Rate 1/6 Turbo Code)

6.4 LOCATION OF ASM

6.4.1 The ASM is attached to (i.e., shall immediately precede) the Reed-Solomon or Turbo Codeblock, or the Transfer Frame if the Physical Channel is not Reed-Solomon or turbo coded.

6.4.2 The ASM for one Codeblock (or Transfer Frame) shall immediately follow the end of the preceding Codeblock (or Transfer Frame); i.e., there shall be no intervening bits (data or fill) preceding the ASM.

6.5 RELATIONSHIP OF ASM TO REED-SOLOMON AND TURBO CODEBLOCKS

6.5.1 The ASM is NOT a part of the encoded data space of the Reed-Solomon Codeblock, and it is not presented to the input of the Reed-Solomon encoder or decoder. 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 ASM, Reed-Solomon Codeblock, and Transfer Frame is illustrated in figure 4-2.

6.5.2 Similarly, the ASM is not presented to the input of the turbo encoder or decoder. It is directly attached to the Turbo Codeblock, as shown in figure 5-4.

6.6 ASM FOR EMBEDDED DATA STREAM

6.6.1 A different ASM pattern (see figure 6-6) may be required where another data stream (e.g., a stream of Transfer Frames played back from a tape recorder in the forward direction) is inserted into the data field of the Transfer Frame of the main stream appearing on the communications channel. The ASM for the embedded data stream, to differentiate it from the main stream marker, shall consist of a 32-bit (4-octet) marker with a pattern as follows:

```
FIRST TRANSMITTED BIT
(Bit 0)
↓
0011  0101  0010  1110  1111  1000  0101  0011
                                             ↑
                            LAST TRANSMITTED BIT
                                        (Bit 31)
```

Figure 6-6: Embedded ASM Bit Pattern

6.6.2 This pattern is represented in hexadecimal notation as:

352EF853

7 PSEUDO-RANDOMIZER

7.1 INTRODUCTION

7.1.1 In order to maintain bit (or symbol) synchronization with the received communications signal, every data capture system at the receiving end requires that the incoming signal have a minimum bit transition density (see recommendation 2.4.9 in reference [5]).

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

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

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

7.2 PSEUDO-RANDOMIZER DESCRIPTION

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

7.2.2 If the pseudo-randomizer is used, on the sending end it is applied to the Codeblock or Transfer Frame after turbo encoding or R-S encoding (if either is used), but before convolutional encoding (if used). On the receiving end, it is applied to derandomize the data after convolutional decoding (if used) and codeblock synchronization but before Reed-Solomon decoding or turbo decoding (if either is used).

NOTES

1 ‘Derandomization’ consists of either: a) exclusive OR-ing the pseudo-random sequence with the received bits of a transfer frame or a Reed-Solomon codeblock, *or* b) inverting (or not inverting), according to the pseudo-randomizer bit pattern, the demodulator output of a Turbo Codeblock.

2 The configuration at the sending end is shown in figure 7-1.

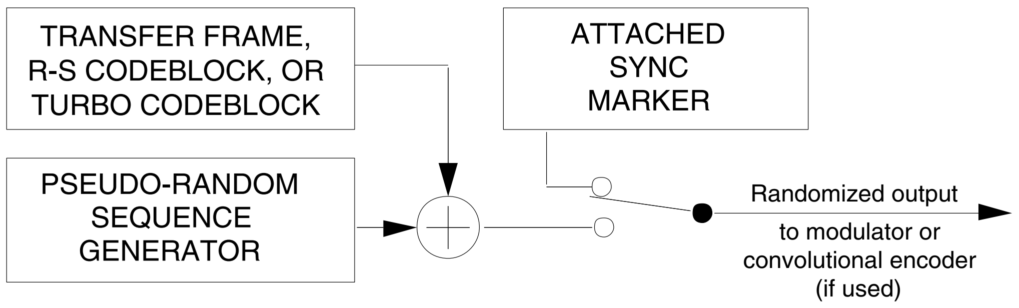

TRANSFER FRAME, R-S CODEBLOCK, OR TURBO CODEBLOCK
ATTACHED SYNC MARKER
PSEUDO-RANDOM SEQUENCE GENERATOR
Randomized output
to modulator or convolutional encoder (if used)

Figure 7-1: Pseudo-Randomizer Configuration

7.3 SYNCHRONIZATION AND APPLICATION OF PSEUDO-RANDOMIZER

7.3.1 The Attached Sync Marker (ASM) is already optimally configured for synchronization purposes and it is therefore used for synchronizing the pseudo-randomizer.

7.3.2 The pseudo-random sequence is applied starting with the first bit of the Codeblock or Transfer Frame. On the sending end, the Codeblock or Transfer Frame is randomized by exclusive-ORing the first bit of the Codeblock or Transfer Frame with the first bit of the pseudo-random sequence, followed by the second bit of the Codeblock or Transfer Frame with the second bit of the pseudo-random sequence, and so on.

7.3.3 On the receiving end, the original Codeblock or Transfer Frame is reconstructed using the same pseudo-random sequence. After locating the ASM in the received data stream, the pseudo-random sequence is exclusive-ORed with the data bits immediately following the ASM. The pseudo-random sequence is applied by exclusive-ORing the first bit following the ASM with the first bit of the pseudo-random sequence, followed by the second bit of the data stream with the second bit of the pseudo-random sequence, and so on.

7.3.4 The pseudo-random sequence shall NOT be exclusive-ORed with the ASM.

7.4 SEQUENCE SPECIFICATION

7.4.1 The pseudo-random sequence shall be generated using the following polynomial:

$$\mathrm{h}(x) = x^8 + x^7 + x^5 + x^3 + 1$$

7.4.2 This sequence begins at the first bit of the Codeblock or Transfer Frame and repeats after 255 bits, continuing repeatedly until the end of the Codeblock or Transfer Frame. The sequence generator is initialized to the all-ones state at the start of each Codeblock or Transfer Frame.

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

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

7.5 LOGIC DIAGRAM

Figure 7-2 represents a possible generator for the specified sequence.

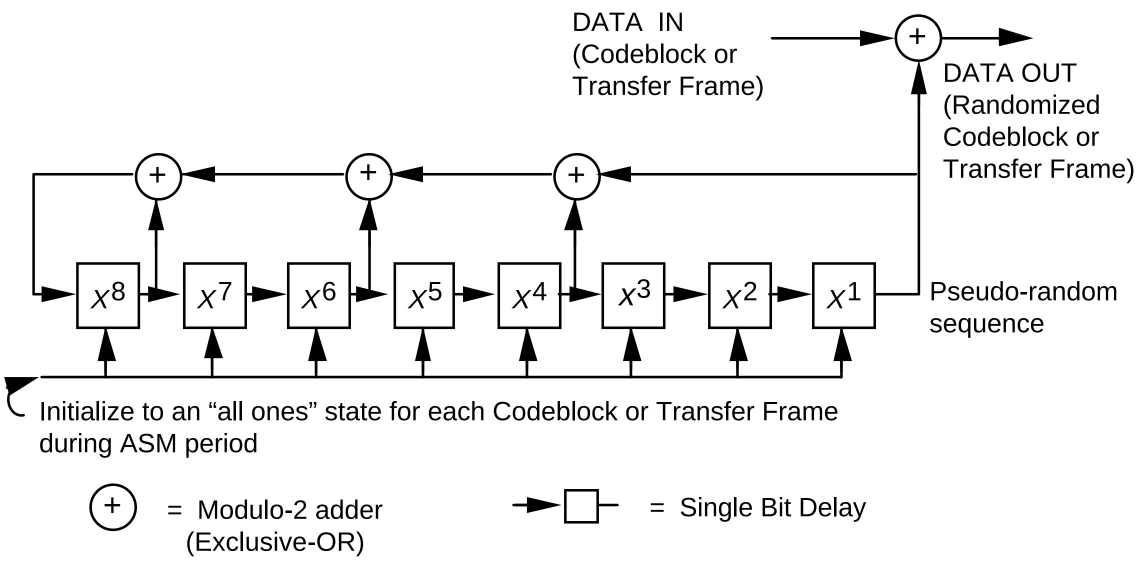


Figure 7-2: Pseudo-Randomizer Logic Diagram

8 TRANSFER FRAME LENGTHS

8.1 GENERAL

8.1.1 Neither the TM Space Data Link Protocol (reference [1]) nor the AOS Space Data Link Protocol (reference [2]) specifies the length of Transfer Frames because there are constraints on the Transfer Frame length depending on the selected coding options.

8.1.2 The constraints on Transfer Frame lengths specified in this section apply to both TM Transfer Frames and the AOS Transfer Frames.

8.1.3 Once selected, the Transfer Frame length must 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 Attached Sync Marker (ASM) specified in section 6.

8.2 CASE 1: UNCODED

The length of the Transfer Frames may be any integer number of octets, as required by the using project, with a maximum of 2048 octets.

8.3 CASE 2: CONVOLUTIONAL ONLY

The length of the Transfer Frames may be any integer number of octets, as required by the using project, with a maximum of 2048 octets.

8.4 CASE 3: REED-SOLOMON ONLY

8.4.1 With the Reed-Solomon Codes specified in section 4, only certain specific lengths of Transfer Frames may be contained within the codeblock’s data space. In some cases these lengths may be shortened in discrete steps by using virtual fill at a small sacrifice in coding gain.

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

8.4.3 The following equation specifies allowed lengths for Transfer Frames (L in octets) when octet compatibility is sufficient:

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

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.

8.4.4 When 32-bit compatibility is required, the Transfer Frame length must be chosen so that it shall be expressed by the above equation and the codeblock length (255-q)I (in octets) shall be a multiple of 4.

8.5 CASE 4: CONCATENATED CODING

The allowable lengths of Transfer Frames when the concatenated (Reed-Solomon and convolutional) coding is used are the same as those for the Reed-Solomon only case (Case 3) shown in 10.4.

8.6 CASE 5: TURBO CODING

8.6.1 The Turbo Codes specified in section 5 of this Recommendation are block codes. Therefore, the Transfer Frame lengths must match the information block lengths for the selected turbo code.

8.6.2 Performance for only the following information block lengths (i.e., Transfer Frame lengths) has been validated by CCSDS and approved for use (values are in octets):

a) 223;

b) 446;

c) 892;

d) 1115;

e) 2048.

NOTE – Interleaver parameters for the length 2048 octets are under study by the CCSDS. Until finalized, use of this option is not recommended.

9 MANAGED PARAMETERS

9.1 OVERVIEW OF MANAGED PARAMETERS

9.1.1 In order 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. Through the use of a management system, management conveys the required information to the synchronization and channel coding systems.

9.1.2 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.

9.2 MANAGED PARAMETERS FOR SELECTED OPTIONS

Table 9-1 lists the managed parameters and shows the selected options for a particular Physical Channel.

Table 9-1: Managed Parameters for Selected Options

Managed Parameter	Allowed Values
Reed-Solomon Coding	Used, Not used
Turbo Coding	Used, Not used
Pseudo-Randomizer	Used, Not used
Convolutional Coding	Used, Not used

9.3 MANAGED PARAMETERS FOR CONVOLUTIONAL CODING

Table 9-2 lists the managed parameters for convolutional coding.

Table 9-2: Managed Parameters for Convolutional Coding

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

9.4 MANAGED PARAMETERS FOR REED-SOLOMON CODING

Table 9-3 lists the managed parameters for Reed-Solomon coding.

Table 9-3: Managed Parameters for Reed-Solomon Coding

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

9.5 MANAGED PARAMETERS FOR TURBO CODING

Table 9-4 lists the managed parameters for turbo coding.

Table 9-4: Managed Parameters for Turbo Coding

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, 16384

9.6 MANAGED PARAMETERS FOR FRAME SYNCHRONIZATION

Table 9-5 lists the managed parameters for frame synchronization.

Table 9-5: Managed Parameters for Frame Synchronization

Managed Parameter	Allowed Values
Transfer Frame Length (bits)	Integer
ASM Length (bits)	32, 64, 96, 128, 192

NOTE – The ASM length is determined by the selected coding schemes.

ANNEX A

ACRONYMS AND TERMS

(This annex **is not** part of the Recommendation)

A1 INTRODUCTION

This annex lists key acronyms and terms that are used throughout this Recommendation to describe synchronization and channel coding.

A2 ACRONYMS

AOS	Advanced Orbiting Systems
ASM	Attached Sync Marker
CADU	Channel Access Data Unit
CCSDS	Consultative Committee For Space Data Systems
FECF	Frame Error Control Field
GF	Galois Field
MSB	Most Significant Bit
NRZ-L	Non-Return-to-Zero-Level
NRZ-M	Non-Return-to-Zero-Mark
OSI	Open Systems Interconnection
QPSK	Quardrature Phase Shift Keying
R-S	Reed-Solomon
TC	Telecommand
TCM	Trellis Coded Modulation
TM	Telemetry
VCDU	Virtual Channel Data Unit

A3 TERMS

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, $n>k$.

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.

Channel Symbol: The unit of output of the innermost encoder.

Codeblock: A codeblock of an (n,k) block code is a sequence of *n* channel symbols which were produced as a unit by encoding a sequence of *k* information symbols, and will be decoded as a unit.

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.

Codeword: In a block code, one of the sequences in the range of the one-to-one transformation (see **Block Encoding**).

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 (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 *i*th stage of the shift register is to be used in computing that parity check.

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 *i*th stage of the shift register is to be used in computing the feedback value, except for the leftmost bit which is ignored.

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.

GF(*n*): The Galois Field consisting of exactly '*n*' elements.

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.

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

NRZ-L: A modulating waveform in which a data 'one' is represented by one of two levels, and a data 'zero' is represented by the other level.

NRZ-M: A modulating waveform 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.

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.

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: 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.

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.

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 N symbols long (a symbol is defined here to be an element of the code's alphabet) and that the parity part is M symbols long. A 'shortened' code is created by taking only S ($S < N$) information symbols as input, appending a fixed string of length N-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'.

ANNEX B

INFORMATIVE REFERENCES

(This annex **is not** part of the Recommendation)

[B1] *Procedures Manual for the Consultative Committee for Space Data Systems*. CCSDS A00.0-Y-8. Yellow Book. Issue 8. Washington, D.C.: CCSDS, July 2002.

[B2] *Telemetry Channel Coding.* Recommendation for Space Data Systems Standards, CCSDS 101.0-B-6. Blue Book. Issue 6. Washington, D.C.: CCSDS, October 2002.

[B3] *Advanced Orbiting Systems, Networks and Data Links: Architectural Specification*. Recommendation for Space Data System Standards, CCSDS 701.0-B-3. Blue Book. Issue 3. Washington, D.C.: CCSDS, June 2001.

[B4] M. Perlman and J. Lee. *Reed-Solomon Encoders—Conventional vs. Berlekamp's Architecture*. JPL Publication 82-71. Pasadena, California: NASA-Jet Propulsion Laboratory, December 1982.

NOTE – Normative references are listed in 1.7.

ANNEX C

SERVICE DEFINITION

(This annex **is** part of the Recommendation.)

C1 GENERAL

C1.1 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.

C1.2 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 may provide other parameters to the service user (e.g., parameters for controlling the service, monitoring performance, facilitating diagnosis, and so on).

C2 OVERVIEW OF THE SERVICE

C2.1 The TM Synchronization and Channel Coding provides unidirectional (one way) transfer of a sequence of fixed-length TM or AOS Transfer Frames at a constant rate over a Physical Channel across a space link, with optional error detection/correction.

C2.2 Only one user can use this service on a Physical Channel, and Transfer Frames from different users are not multiplexed together within one Physical Channel.

C3 SERVICE PARAMETERS

C3.1 FRAME

The Frame parameter is the service data unit of this service and shall be either a TM Transfer Frame defined in reference [1] or an AOS Transfer Frame defined in reference [2]. The length of any Transfer Frame transferred on a Physical Channel must be the same, and is established by management.

C3.2 QUALITY INDICATOR

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

C3.3 SEQUENCE INDICATOR

The Sequence Indicator is a parameter that is 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.

C4 SERVICE PRIMITIVES

C4.1 GENERAL

C4.1.1 The service primitives associated with this service are:

a) ChannelAccess.request;

b) ChannelAccess.indication.

C4.1.2 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.

C4.1.3 The ChannelAccess.indication is passed from the service provider to the service user at the receiving end to deliver a Frame.

C4.2 ChannelAccess.request

C4.2.1 Function

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

C4.2.2 Semantics

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

ChannelAccess.request (Frame)

C4.2.3 When Generated

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

C4.2.4 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 channel symbols.

C4.2.5 Additional Comments

The ChannelAccess.request primitive is used to perform the functions described in 2.3.1 and to transfer the resulting channel symbols over a Physical Channel across the space link.

C4.3 ChannelAccess.indication

C4.3.1 Function

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

C4.3.2 Semantics

The ChannelAccess.indication primitive shall provide parameters as follows:

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

C4.3.3 When Generated

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

C4.3.4 Effect On Receipt

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

C4.3.5 Additional Comments

The ChannelAccess.indication primitive is used to perform the functions described in 2.3.2 and to deliver Transfer Frames of a Physical Channel to the service user.

ANNEX D

TRANSFORMATION BETWEEN BERLEKAMP AND CONVENTIONAL REPRESENTATIONS

(This annex **is not** part of the Recommendation)

D1 PURPOSE

This annex provides information to assist users of the Reed-Solomon code in this Recommendation 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 Recommendation is based.

D2 TRANSFORMATION

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

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

where the components z_i are coefficients of ℓ_i, respectively:

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

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

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

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

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

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

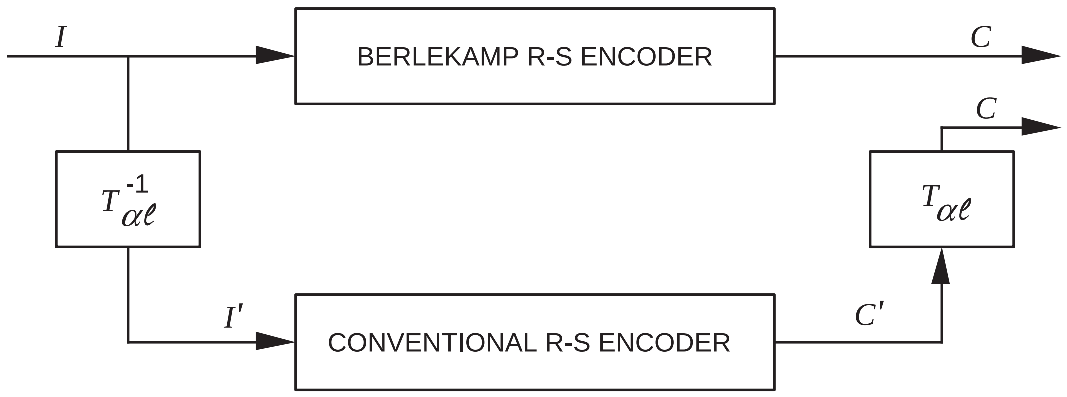


Figure D-1: Transformational Equivalence

Conventional and Berlekamp types of (255,k) Reed-Solomon encoders are assumed to have the same self-reciprocal generator polynomial whose coefficients appear in paragraph 6.2(d) and (e). The representation of symbols associated with the conventional encoder is the polynomials in 'α' appearing in table D-1. 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

$$[1\ 0\ 1\ 1\ 1\ 0\ 0\ 1] \overset{T_{\alpha\ell}^{-1}}{\begin{bmatrix} 1&1&0&0&0&1&0&1 \\ 0&1&0&0&0&0&1&0 \\ 0&0&1&0&1&1&1&0 \\ 1&1&1&1&1&1&0&1 \\ 1&1&1&1&0&0&0&0 \\ 0&1&1&1&1&0&0&1 \\ 1&0&1&0&1&1&0&0 \\ 1&1&0&0&1&1&0&0 \end{bmatrix}} = [u_7, u_6, \ldots, u_0] = 00101010 = I'$$

Note that 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 D-1.

Example 2:

Given check symbol C',

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

Then,

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

Table D-1: Equivalence of Representations[1]

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

[1] From table 4 of reference [B5]. Note: Coefficients of the 'Polynomial in Alpha' column are listed in descending powers of α, starting with α^7.

Table D-1: Equivalence of Representations (continued)

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

Table D-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 D-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 E

EXPANSION OF REED-SOLOMON COEFFICIENTS

(This annex **is not** part of the Recommendation)

Purpose:

While the equations given in the Reed-Solomon Coding section of this Recommendation 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 that $G_3 = G_{29} = G_{13} = G_{19}$.

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

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	$= G_{16}$	$= \alpha^0$	0	0	0	0	0	0	0	1
G_1	$= G_{15}$	$= \alpha^{30}$	1	0	1	0	0	1	0	1
G_2	$= G_{14}$	$= \alpha^{230}$	0	1	1	0	1	0	0	1
G_3	$= G_{13}$	$= \alpha^{49}$	0	0	0	1	1	0	1	1
G_4	$= G_{12}$	$= \alpha^{235}$	1	0	0	1	1	1	1	1
G_5	$= G_{11}$	$= \alpha^{129}$	0	1	1	0	1	0	0	0
G_6	$= G_{10}$	$= \alpha^{81}$	1	0	0	1	1	0	0	0
G_7	$= G_9$	$= \alpha^{76}$	0	1	1	0	0	1	0	1
	G_8	$= \alpha^{173}$	0	1	0	0	1	0	1	0

ANNEX F

CHANGES FROM REFERENCES [B2] AND [B3]

(This annex **is not** part of the Recommendation)

F1 GENERAL

This Recommendation was developed from the specifications regarding synchronization and channel coding in older CCSDS Recommendations [B2] and [B3], but a few technical specifications in [B2] and [B3] have been changed in order to define all Space Data Link Protocols in a unified way. These technical changes are described in subsection F1. Also, some technical terms in references [B2] and [B3] have been changed in order to unify the terminology used in all the CCSDS Recommendations that define space link protocols, as well as to define these schemes as general communications schemes. These terminology changes are listed in F2.

F2 TECHNICAL CHANGES

F2.1 PARAMETERS ASSOCIATED WITH TRANSFER FRAME LENGTH

In [B2] and [B3], the periods during which the value of parameters shall be fixed are not consistently specified. In this Recommendation, any parameter associated with the length of the Transfer Frame shall be fixed for a Mission Phase on a particular Physical Channel.

F2.2 TRANSFER FRAME LENGTHS

The constraints on Virtual Channel Data Unit (VCDU) (AOS Transfer Frame) lengths specified in [B3] were different from those on TM Transfer Frame lengths specified in [B2]. In this Recommendation, the same constraints are applied to both types of Transfer Frames.

F3 TERMINOLOGY CHANGES

Tables F-1 and F-2 list the terms that have been changed from references [B2] and [B3], respectively.

Table F-1: Terms That Have Been Changed from Reference [B2]

Terms Used in Reference [B2]	Terms Used in This Recommendation
Telemetry Transfer Frame	TM Transfer Frame

Table F-2: Terms That Have Been Changed from Reference [B3]

Terms Used in Reference [B3]	**Terms Used in This Recommendation**
PCA_PDU	Channel Symbols
Virtual Channel Data Unit (VCDU)	AOS Transfer Frame