The Consultative Committee for Space Data Systems

Report Concerning Space Data System Standards

TC SYNCHRONIZATION AND CHANNEL CODING—SUMMARY OF CONCEPT AND RATIONALE

INFORMATIONAL REPORT

CCSDS 230.1-G-3

GREEN BOOK

October 2021

The Consultative Committee for Space Data Systems

Report Concerning Space Data System Standards

TC SYNCHRONIZATION AND CHANNEL CODING—SUMMARY OF CONCEPT AND RATIONALE

INFORMATIONAL REPORT

CCSDS 230.1-G-3

GREEN BOOK

October 2021

AUTHORITY

Issue:	Informational Report, Issue 3
Date:	October 2021
Location:	Washington, DC, USA

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and reflects the consensus of technical panel experts from CCSDS Member Agencies. The procedure for review and authorization of CCSDS Reports is detailed in *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-4).

This document is published and maintained by:

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

FOREWORD

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

http://www.ccsds.org/

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

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

Member Agencies

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

Observer Agencies

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

DOCUMENT CONTROL

Document	Title	Date	Status
CCSDS 230.1-G-1	TC Synchronization and Channel Coding—Summary of Concept and Rationale, Informational Report, Issue 1	June 2006	Original issue, superseded
CCSDS 230.1-G-2	TC Synchronization and Channel Coding—Summary of Concept and Rationale, Informational Report, Issue 2	November 2012	Issue 2, superseded
CCSDS 230.1-G-3	TC Synchronization and Channel Coding—Summary of Concept and Rationale, Informational Report, Issue 3	October 2021	Current issue
CCSDS 230.1-G-3 EC 1	Editorial Change 1	January 2023	Restores missing table.

CONTENTS

CONTENTS (continued)

CONTENTS (continued)

Section Page

Figure

CONTENTS (continued)

Figure Page

CONTENTS (continued)

CONTENTS (continued)

1 INTRODUCTION

1.1 PURPOSE

This Informational Report contains the concept and supporting rationale for the TC Synchronization and Channel Coding Recommended Standard developed by the Consultative Committee for Space Data Systems (CCSDS). It has been prepared to serve two major purposes:

- to provide an introduction and overview for the concept upon which the detailed CCSDS TC Synchronization and Channel Coding Recommended Standard (reference [1]) are based;
- to describe and explain the procedures employed and to supply the supporting rationale, including performance data and historical background information on their selection.

This document is a CCSDS Informational Report and is therefore not to be taken as a CCSDS Recommended Standard. The actual Recommended Standard is in reference [1].

1.2 SCOPE

The procedures and data formats developed for the TC Synchronization and Channel Coding are designed for space communications links. Typically, the links are used to carry telecommand data from ground elements to spacecraft, and the 'TC' in the name originates from this use. However, the TC Synchronization and Channel Coding can also be applied to links between spacecraft and to links carrying non-telecommand data.

The procedures and data formats of the TC Synchronization and Channel Coding are designed with reliability and efficiency as primary considerations. The results reflect the consensus of experts from many space agencies.

This document provides supporting and descriptive material only; it is not part of the Recommended Standard. In the event of any conflict between the TC Synchronization and Channel Coding Recommended Standard (reference [1]) and the material presented in this report, the Recommended Standard (reference [1]) shall prevail.

1.3 ORGANIZATION

An overview of TC Synchronization and Channel Coding is presented in section 2. Sections 3 through 9 describe the channel coding elements, section 10 discusses options for repeated transmissions, and section 11 presents performance data. Annex A is a glossary, and annex B is a list of acronyms and abbreviations. Annexes C and D provide theoretical background. Annex E gives performance data for obsolete features. Annex F shows examples of data structures before and after the channel coding and synchronization process. Annex G considers a go-back-*n* scheme, and annex H addresses the undetected error rate of Bose-Chaudhuri-Hocquenghem (BCH) codes.

1.4 REFERENCES

The following publications are referenced in this document. At the time of publication, the editions indicated were valid. All publications are subject to revision, and users of this document are encouraged to investigate the possibility of applying the most recent editions of the publications indicated below. The CCSDS Secretariat maintains a register of currently valid CCSDS publications.

[1] *TC Synchronization and Channel Coding*. Issue 4. Recommendation for Space Data System Standards (Blue Book), CCSDS 231.0-B-4. Washington, D.C.: CCSDS, July 2021.

[2] *Overview of Space Communications Protocols*. Issue 3. Report Concerning Space Data System Standards (Green Book), CCSDS 130.0-G-3. Washington, D.C.: CCSDS, July 2014.

[3] *TC Space Data Link Protocol*. Issue 4. Recommendation for Space Data System Standards (Blue Book), CCSDS 232.0-B-4. Washington, D.C.: CCSDS, October 2021.

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

[5] *TM Synchronization and Channel Coding*. Issue 3. Recommendation for Space Data System Standards (Blue Book), CCSDS 131.0-B-3. Washington, D.C.: CCSDS, September 2017.

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

[7] *Space Link Extension—Forward CLTU Service Specification*. Issue 4. Recommendation for Space Data System Standards (Blue Book), CCSDS 912.1-B-4. Washington, D.C.: CCSDS, August 2016.

[8] F. Baudin, et al. *Design and Development of a Very High Speed Reed-Solomon Encoder/Decoder Chip Set: Architectural Design Report*. ESA/ESOC Project no. 13.501. CSEM Technical Report 496.I.. Neuchâtel, Switzerland: CSEM, 1992.

[9] J.C. Morakis. *Analysis of a Proposed TCM Coding System*. N.p., n.d. [Not publicly available.]

[10] R. J. McEliece. *Finite Field for Scientists and Engineers*. Kluwer International Series in Engineering and Computer Science. Norwell, Massachusetts: Kluwer, 1987.

[11] *Communications Operation Procedure-1*. Issue 2. Recommendation for Space Data System Standards (Blue Book), CCSDS 232.1-B-2. Washington, D.C.: CCSDS, September 2010.

[12] *TM Synchronization and Channel Coding—Summary of Concept and Rationale*. Issue 3. Report Concerning Space Data System Standards (Green Book), CCSDS 130.1-G-3. Washington, D.C.: CCSDS, June 2020.

[13] H. Bruneel and M. Moeneclaey. "On the Throughput Performance of Some Continuous ARQ Strategies with Repeated Transmissions." *IEEE Transactions on Communications* 34, no. 3 (March 1986): 244-249.

[14] Erik Mose Sørensen and Paolo Ferri. "Use of Packet Telemetry and telecommand Standards for a Deep-Space Mission: the Rosetta Case." In *Proceedings of the Fifth International Conference on Space Operations (SpaceOps 1998) (1-5 June 1998, Tokyo, Japan)*. 5e006. N.p.: SpaceOps, 1998.

[15] A. Sastry. "Improving Automatic Repeat-Request (ARQ) Performance on Satellite Channels Under High Error Rate Conditions." *IEEE Transactions on Communications* 23, no. 4 (April 1975): 436-439.

[16] E. Paolini and M. Chiani. "Input to the Ongoing CCSDS Books Revision." Presented at CCSDS Coding and Synchronization Working Group meeting (October 2010, London).

[17] Min-Goo Kim and Jae Hong Lee. "Undetected Error Probabilities of Binary Primitive BCH Codes for Both Error Correction and Detection." *IEEE Transactions on Communications* 44, no. 5 (May 1996): 575-580.

[18] S. Dolinar, et al. "Bounded Angle Iterative Decoding of LDPC Codes." In *Proceedings of MILCOM 2008 (16–19 Nov. 2008, San Diego, CA)*, 1–6. New York: IEEE, 2008.

[19] S. Dolinar, et al. "Bounds on Error Probability of Block Codes with Bounded-Angle Maximum-Likelihood Incomplete Decoding." In *Proceedings of ISITA 2008 (7–10 Dec. 2008, Auckland, New Zealand)*, 1–6. New York: IEEE, 2008.

[20] J. Thorpe. "Low-Density Parity-Check (LDPC) Codes Constructed from Protographs." *IPN Progress Report* 42-154 (August 15, 2003).

[21] T. Richardson. "Error Floors of LDPC Codes." In *Proceedings of the 41st Annual Allerton Conference on Communication, Control, and Computing (1–3 Oct. 2003, Monticello, Illinois)*, 1425–1435. Urbana-Champaign: University of Illinois, 2003.

[22] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold. "Irregular Progressive Edge-Growth (PEG) Tanner Graphs." In *Proceedings of 2002 IEEE International Symposium on Information Theory*, 480. New York: IEEE, 2002.

[23] G. Liva, et al. "Short Turbo Codes over High Order Fields." *IEEE Transactions on Communications* 61, no. 6 (June 2013): 2201–2211.

[24] L. Dolecek, et al. "Non-Binary Protograph-Based LDPC Codes: Enumerators, Analysis, and Designs." *IEEE Transactions on Information Theory* 60, no. 7 (July 2014): 3913–3941.

[25] K. S. Andrews, et al. "The Development of Turbo and LDPC Codes for Deep-Space Applications." *Proceedings of the IEEE* 95, no. 11 (November 2007): 2142–2156.

[26] J. Chen and M.P.C. Fossorier. "Near Optimum Universal Belief Propagation Based Decoding of Low-Density Parity Check Codes." *IEEE Transactions on Communications* 50, no. 3 (March 2002): 406–414.

[27] M. Baldi, et al. "On the Use of Ordered Statistics Decoders for Low-Density Parity-Check Codes in Space Telecommand Links." *EURASIP Journal on Wireless Communications and Networking* 2016, no. 1 (December 2016).

[28] M. Baldi, et al. "On the Applicability of the Most Reliable Basis Algorithm for LDPC Decoding in Telecommand Links." In *6th International Conference on Information and Communication Systems (ICICS) (7–9 April 2015, Amman, Jordan)*, 1-6. Piscataway, New Jersey: IEEE, 2015.

[29] M. Baldi, et al. "State-Of-The-Art Space Mission Telecommand Receivers." *IEEE Aerospace and Electronic Systems Magazine* 32, no. 6 (June 2017): 4–15.

[30] J. Massey. "Optimum Frame Synchronization." *IEEE Transactions on Communications* 20, no. 2 (April 1972): 115–119.

2 OVERVIEW

2.1 CONTEXT

TC Synchronization and Channel Coding forms part of the CCSDS Space Link Protocols. The report in reference [2] provides an overview of the Space Link Protocols and shows the relationship of the TC Synchronization and Channel Coding to the other elements. It also shows how the Space Link Protocols relate to the equivalent layers of the OSI model.

The TC Synchronization and Channel Coding Recommended Standard (reference [1]) defines the behavior of the Synchronization and Channel Coding Sublayer. The interfaces between the sublayer and its immediate neighbors are shown in figure 2-1.

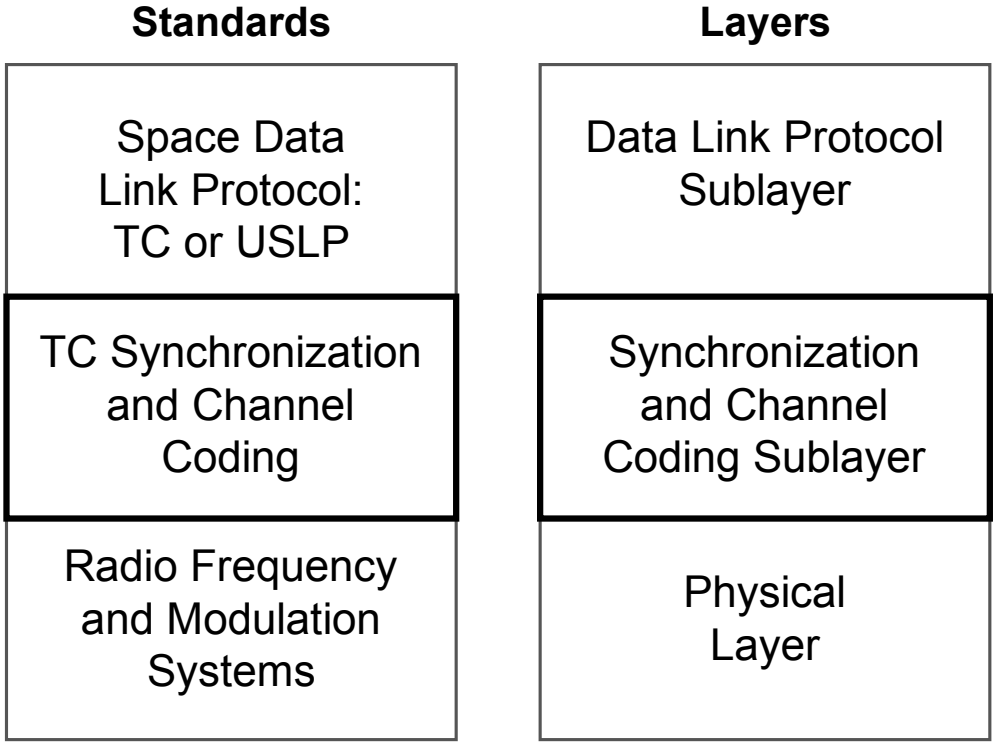

Figure 2-1: Related Standards and Layers

The neighbor above is the Data Link Protocol Sublayer, which in this case is defined by either the TC Space Data Link Protocol (reference [3]) or the Unified Space Data Link Protocol (reference [4]). The Data Link Protocol Sublayer generates variable-length TC Transfer Frames and passes them to the Synchronization and Channel Coding Sublayer.

NOTE – TC Synchronization and Channel Coding is used with the variable-length frames. TM Synchronization and Channel Coding (reference [5]) is used with the fixed-length frames.

The neighbor below is the Physical Layer. The TC Synchronization and Channel Coding Recommended Standard (reference [1]) includes procedures, called Physical Layer Operations Procedures (PLOPs), for controlling the behavior of the Physical Layer. The Radio Frequency and Modulation Systems Recommended Standard (reference [6]) provides recommendations for RF modulation and the properties of the generated RF signal.

2.2 PURPOSE

The purpose of the Synchronization and Channel Coding Sublayer and the associated Physical Layer Operations Procedures at the sending end is to:

- encode the data units received from the sublayer above to provide a forward error detection and correction capability, which gives a high degree of protection against errors or corruptions that occur during transmission through the space link;
- encapsulate the data units so that the start and end can be detected by the receiving end;
- enable the receiving end to resolve the data ambiguity (sense of '1' and '0') in the received symbol stream;
- control the transmission of an acquisition bit pattern, which enables the receiver to acquire bit synchronization;
- ensure there are sufficient bit transitions in the transmitted bit stream so that the receiver can maintain bit synchronization during the reception of a data unit;
- control the optional transmission of an idle bit pattern, which enables the receiver to maintain bit synchronization between data units.

The purpose of the Synchronization and Channel Coding Sublayer at the receiving end is to:

- detect the start and end of each data unit in the received symbol stream;
- resolve, if necessary, the data ambiguity (sense of '1' and '0') in the received symbol stream;
- decode each data unit and optionally correct the errors detected.

One instance of the Synchronization and Channel Coding Sublayer processes the data stream for a single Physical Channel. A Physical Channel is a stream of bits transferred over a space link in a single direction.

NOTE – Layers above the Synchronization and Channel Coding Sublayer may be using timers that include the time for a data unit to pass through the Synchronization and Channel Coding Sublayer. For example, the Sequence-Controlled Service Type of the TC Space Data Link Protocol has such timers. It can therefore be helpful if queuing and buffering mechanisms in the Synchronization and Channel Coding Sublayer are designed to limit unpredictable delays.

2.3 PROCEDURES AT THE SENDING END

TC Synchronization and Channel Coding defines procedures for the sending end, which are applied in the order shown in figure 2-2 when BCH coding is used, and in figure 2-2 when Low-Density Parity-Check (LDPC) coding is used. An optional systematic retransmission is also defined, but its position relative to the other procedures is implementation dependent; therefore it is not shown in the figure.

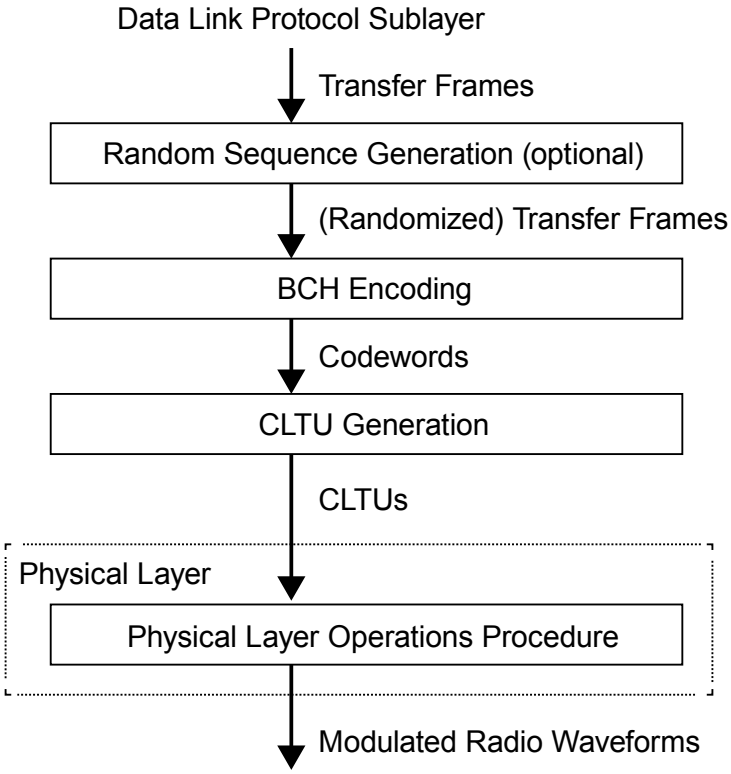

Figure 2-2: Procedures at the Sending End When BCH Coding Is Used

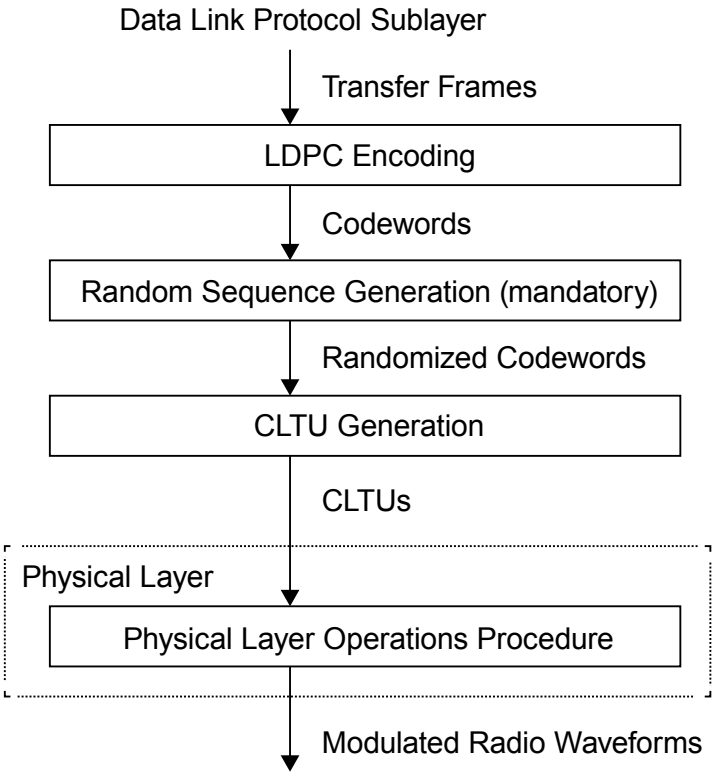

Figure 2-3: Procedures at the Sending End When LDPC Coding Is Used

Random sequence generation improves bit transition density in order to help maintain bit synchronization of the RF equipment at the receiving end. While randomization is optional with BCH coding, it should be used unless it can be shown that a sufficient bit transition density will be generated by other means. Randomization is discussed in section 9.

NOTE – For brevity, the word 'random' is used in place of 'pseudo-random' in this report.

BCH coding provides protection against errors that occur during RF transmission. At the sending end, the data is encoded with a BCH block code; the encoding procedure is discussed in section 3. The encoding generates a set of BCH codewords.

LDPC coding provides greater protection against errors that occur during RF transmission than BCH coding does, at the cost of requiring greater complexity at both the transmitter and receiver. At the sending end, the data is encoded with an LDPC block code; the encoding procedure is discussed in section 4. The encoding generates a set of LDPC codewords.

A Communications Link Transmission Unit (CLTU) is generated from each set of BCH codewords or randomized LDPC codewords. The CLTU consists of a Start Sequence, a set of BCH codewords or randomized LDPC Codewords, and a Tail Sequence when applicable. The Start Sequence provides frame synchronization for the Transfer Frames. CLTU generation is discussed in sections 4 and 5.

NOTE – In Recommended Standards published in September 2003, the name 'Communications Link Transmission Unit' replaced the earlier name 'Command Link Transmission Unit'.

The Physical Layer Operations Procedures define how CLTUs and additional data structures, the Acquisition Sequence and the Idle Sequence, are used with different states of channel modulation. The Recommended Standard (reference [1]) includes two procedures, called PLOP-1 and PLOP-2, which are discussed in section 7.

The optional systematic retransmission at the sending end can improve performance for missions with a long light time delay (deep space missions). It is discussed in section 10.

2.4 PROCEDURES AT THE RECEIVING END

TC Synchronization and Channel Coding defines procedures for the receiving end, which are applied in the order shown in figure 2-4 when BCH coding is used and figure 2-5 when LDPC coding is used.

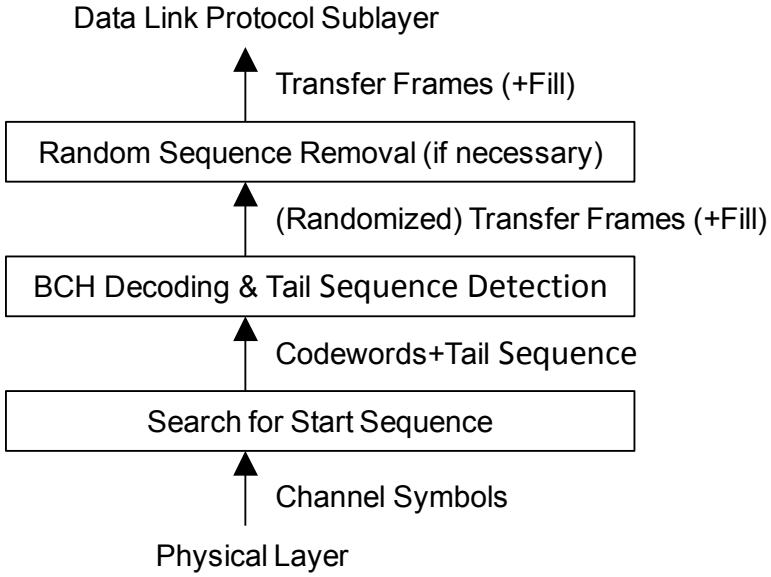

Figure 2-4: Procedures at the Receiving End When BCH Coding Is Used

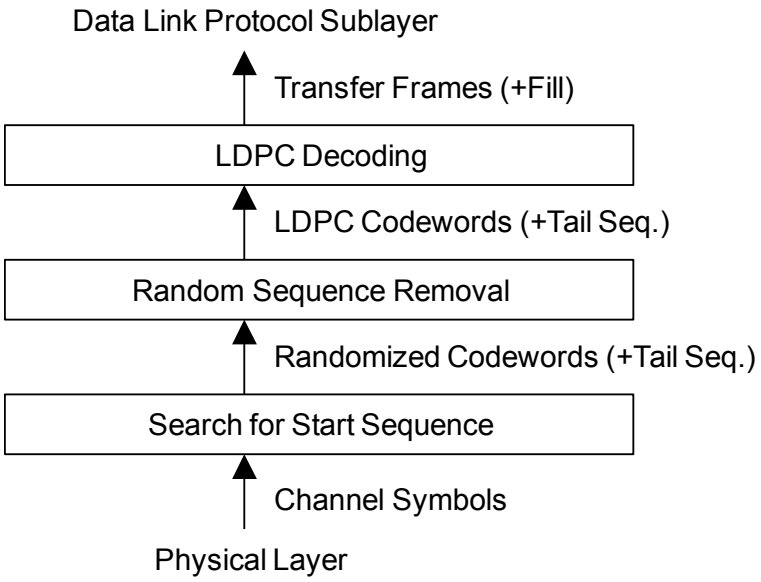

Figure 2-5: Procedures at the Receiving End When LDPC Coding Is Used

The CLTU Reception Procedure processes a stream of bits. It searches the input stream, looking for the Start Sequence, which identifies the start of a CLTU. The CLTU Reception Procedure is discussed in section 8

BCH decoding decodes BCH codewords in the decoding mode set for the Physical Channel and delivers data extracted from the successfully decoded codewords. Two decoding modes are available: an error detecting mode and an error correcting mode. The procedure continues processing the codewords until it detects a decoding failure or identifies the tail sequence by pattern matching.

If Pseudo-randomization is used with BCH decoding, the data extracted from the decoded BCH codewords is derandomized before it is passed to the Data Link Protocol Sublayer. Randomization is discussed in section 9.

LDPC decoding decodes LDPC codewords and delivers data extracted from the successfully decoded codewords. Codewords must be derandomized prior to decoding. The derandomization and decoding procedures continue until the decoder detects a decoding failure. An optional tail sequence may be used with the (128,64) LDPC code; in this case, it may be identified either by a decoding failure or by pattern-matching prior to random sequence removal.

3 BCH ENCODING

3.1 INTRODUCTION

At the sending end, the Data Link Protocol Sublayer generates variable-length Transfer Frames and passes them to the Synchronization and Channel Coding Sublayer. The format of a Transfer Frame is defined in references [3] and [4]. The Synchronization and Channel Coding Sublayer encodes the Frames with a BCH block code and generates a set of BCH codewords.

The BCH encoding is the subject of this section. It provides a forward error detection and correction capability that gives a high degree of protection against errors or corruptions that occur during transmission through the space link.

If randomization is in use for the Physical Channel, the Frames are randomized before the BCH encoding. Section 9 discusses randomization at the sending and receiving ends.

3.2 FORMAL INTERFACE TO DATA LINK PROTOCOL SUBLAYER

The Recommended Standard (reference [1]) includes a formal definition of the service interface between the Data Link Protocol Sublayer and the Synchronization and Channel Coding Sublayer at the sending end. The Data Link Protocol Sublayer uses the following service primitive:

ChannelAccess.request (Frames)

The Frames parameter consists of one or more Transfer Frames to be transferred to the receiving end through the Physical Channel.

From the point of view of the Synchronization and Channel Coding Sublayer, the content of the Frames parameter is a single block of data. The Synchronization and Channel Coding Sublayer does not use the length fields or other values in the Transfer Frames and it does not need to know the positions of any Frame boundaries within the block of data.

For a single ChannelAccess.request, the Synchronization and Channel Coding Sublayer generates a set of BCH codewords, and that set of BCH codewords is placed in a single CLTU. One of the managed parameters for the Physical Channel is the maximum length of a CLTU; this parameter places a constraint on the maximum length of the Frames parameter for the service request. The length of the CLTU can be calculated from the total length of the Frames in the Frames parameter as follows:

Length of the CLTU = 10 + ((Total length of the Frames + 6) / 7) * 8,

where the lengths are expressed in octets, and the division represented by the '/' character is integer division, which truncates any fractional part of the result. The result of the division corresponds to the number of BCH codewords. So, for example, if the total length of the Frames is 1189 octets, then:

- 1189 + 6 = 1195;
- integer division of 1195 by 7 gives 170 (the result of the division is 170.71 and the fractional 0.71 is truncated), so there will be 170 BCH codewords;
- 170 * 8 = 1360; and
- addition of 10 gives a CLTU length of 1370 octets.

Also, some of the standards derived from the Recommended Standard (reference [1]) place a limit of one Transfer Frame per CLTU. A CLTU carrying a single maximum-length Transfer Frame (1024 octets) has 147 BCH codewords and a length of 1186 octets.

3.3 FORMAT OF A BCH CODEWORD

3.3.1 THE 64-BIT BCH CODEWORD

The format of a BCH codeword is shown in figure 3-1.

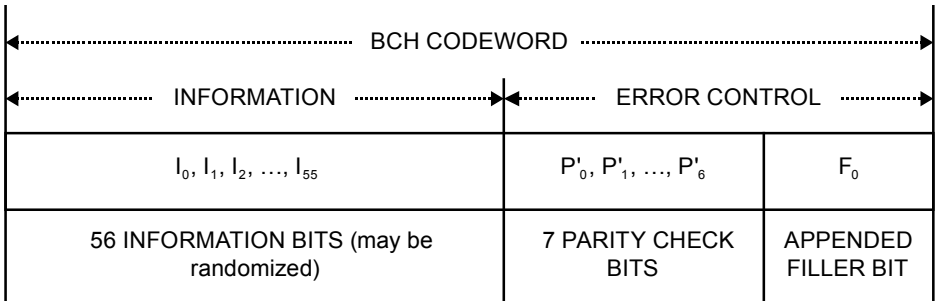

Figure 3-1: Format of a BCH Codeword

Information bits: Subsection 3.4 below discusses how the variable-length input from the Frames parameter is handled to provide 56 Information bits for each BCH codeword.

Parity bits: The Information bits are used as input to the encoding process described in 3.6 below. The encoding process delivers seven parity bits, P_0 to P_6. It is the complements of these bits, P'_0 to P'_6, which are placed in the Error Control field of the BCH codeword.

Filler bit: The Filler Bit is always zero.

When a BCH codeword is transmitted as part of a CLTU, the first bit of the codeword to be transmitted is I_0.

3.3.2 OBSOLETE CODEWORD LENGTHS

The Recommended Standard (reference [1]) includes only the BCH codeword with a length of 64 bits. Earlier CCSDS recommendations for Telecommand Channel Coding included further codeword length options of 40 bits, 48 bits and 56 bits. For the other codeword lengths, the Error Control field remained at eight bits, and the Information field was shortened, so the codeword carried less data. The selected codeword length option was fixed for the mission.

The further length options were eliminated in favor of a single standard length of 64 bits, in recommendations published in June 2000.

The 64-bit codeword is the most efficient of the options. Table 3-1 shows the bandwidth overhead of the BCH coding for the 64-bit codeword and for the obsolete shorter codeword lengths.

Table 3-1: Bandwidth Overhead for Different Codeword Lengths

Codeword length	Data length	Bandwidth overhead
64 bits	56 bits	14%
56 bits (obsolete)	48 bits	17%
48 bits (obsolete)	40 bits	20%
40 bits (obsolete)	32 bits	25%

3.4 BREAKING THE FRAMES INTO FIXED-LENGTH PIECES

The data from the Transfer Frames are placed, seven octets (56 bits) at a time, into a set of BCH codewords.

It should be noted that if randomization is in use for the Physical Channel, then the Frame data are randomized before being placed in the BCH codewords. Section 9 discusses randomization.

The first seven octets from the start of ‘Frames’ are placed in the Information field of the first codeword and the next seven octets in the next codeword and so on. Frame boundaries are ignored during this process. As mentioned in 3.2 above, the Synchronization and Channel Coding Sublayer does not need to know the positions of any Frame boundaries within the ‘Frames’ data.

At the end of ‘Frames’, there may not be seven octets left for the last BCH codeword. In this case, the available octets are placed in the first octets of the Information field and the rest of the Information field is filled with a fill pattern:

– the fill pattern consists of alternating ‘0’ and ‘1’ bits, starting with a ‘0’, so an octet of fill bits has the hexadecimal value 55;

– only the last BCH codeword of a CLTU can have fill bits;

– the last BCH codeword can have up to six octets of fill bits;

– if randomization is in use for the Physical Channel, then randomization of the fill bits is optional.

Figure 3-2 shows a simple example, where a Frames parameter with a total length of 25 octets results in four BCH codewords. In the example, the last BCH codeword has three octets of fill bits.

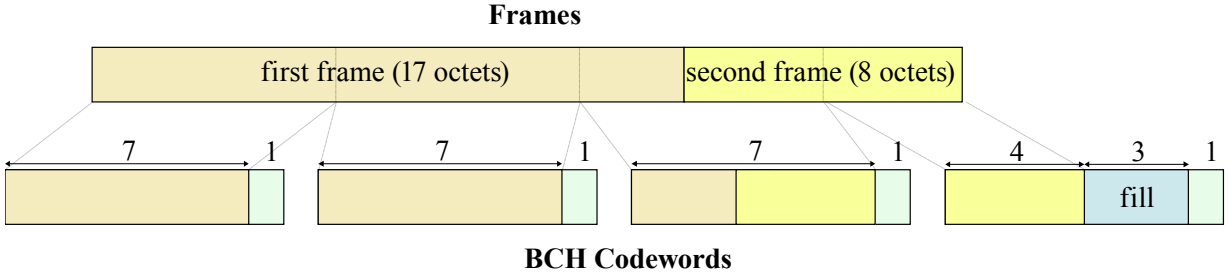

Figure 3-2: Example of Breaking ‘Frames’ into BCH Codewords

3.5 THE BCH CODE

3.5.1 GENERAL

The BCH code specified in the Recommended Standard (reference [1]) is a systematic (63,56) block code. A 56-bit information word is encoded into a 63-bit codeword. Because the code is systematic, the information word is unchanged by the encoding, which generates seven check bits.

The code is an expurgated Hamming code, derived from a basic (63,57) Hamming code.

3.5.2 BASIC (63,57) HAMMING CODE

The basic Hamming codes have a codeword length of 2^n–1 bits, with n check bits. The codes have minimum distance of three; that is, each codeword differs from every other codeword in at least three bit positions. A basic Hamming code can either correct a single error or detect two errors, but not both simultaneously.

When n=6, the basic Hamming code has six check bits, a codeword length of 63 bits and an information word of 57 bits. Therefore the code has 2^{57} codewords corresponding to the 2^{57} possible input information words.

The generator polynomial, g1(x), for the basic (63,57) Hamming code is:

$$g1(x) = x^6 + x + 1$$

The same polynomial can be applied to the received 63-bit codeword to give the 6-bit Syndrome Value (SYND). Considering the received 63-bit codeword as a polynomial whose coefficients are the bit values, the syndrome is the remainder when it is divided (with polynomial division and arithmetic in the binary field) by the generator polynomial.

If the syndrome is zero, then the received codeword is one of the codewords and is assumed to be error-free. If a single error has occurred, the syndrome value indicates the position of the error.

3.5.3 EXPURGATED HAMMING CODE (63,56)

The basic (63,57) Hamming code can be improved by limiting it to the even-parity codewords. The resulting code has only 2^{56} codewords so it has an information word of 56 bits and 7 check bits. The code now includes a parity check, and the generator polynomial, g(x), for this expurgated Hamming code (63,56) is:

$$g(x) = g1(x)\,(x + 1)$$

$$g(x) = (x^6 + x + 1)\,(x + 1)$$

$$g(x) = x^7 + x^6 + x^2 + 1.$$

The code has a minimum distance of four. If it is used in error detecting mode, it can detect up to three errors, and it can also detect any odd number of errors. If it is used in error-correcting mode, it can detect up to two errors and correct a single error.

3.6 BCH ENCODING

The Recommended Standard (reference [1]) includes a schematic diagram for the implementation of the BCH encoding using a linear feedback shift register.

4 SHORT BLOCKLENGTH LDPC CODES

4.1 INTRODUCTION

Traditionally, uplink communication has been used primarily for telecommand, where short command sequences were transmitted to an unmanned spacecraft a few times per day to a few times per month. Current and future spacecraft use uplink communication for a much wider variety of uses. While telecommand continues to be an essential application, both in normal and emergency situations, there is increasing demand for transmitting larger volumes of data to spacecraft. Flight equipment can be reprogrammed, both with software for microprocessors and firmware for Field Programmable Gate Arrays (FPGAs). Manned missions benefit from live uplink video and Internet access.

One view of this variety of uplink needs is summarized in table 4-1, where they have been categorized into four different applications. The different applications place different demands on the telecommunications link, and hence the error correcting codes chosen for each application may also be different.

Table 4-1: Four Modes of Operation for Uplink Codes

Mode	Purpose	Blocklength (kb)	Throughput (kb/s)
A	Emergency	0.1	0.01
B	Command/ARQ	0.1–1	1–4
C	File Upload	1–4	1000
D	Human Support	>4	20000

The TC Synchronization and Channel Coding Recommended Standard (reference [1]) defines codes that are primarily intended for Modes A and B in this table. While not standardized by CCSDS, some missions have selected LDPC codes from the TM Synchronization and Channel Coding Recommended Standard (reference [5]) for Modes C and D, and the codes are well suited to these uses, for these applications are similar to telemetry transfer.

For Modes A and B, considerable improvement became possible over the legacy BCH code with the advent of modern coding techniques in the 1990s. Two binary LDPC codes were standardized, and the codes may be substituted in place of the BCH code, with little impact to other aspects of the telecommand system. Subsection 4.2 briefly describes how these codes were selected.

4.2 CODE DESIGN AND SELECTION

4.2.1 INTRODUCTION

Initial trade studies promptly determined that codes of rate 1/2 provided an attractive balance between encoding and decoding complexity, coding gain, and bandwidth expansion, not so much to meet spectral requirements, but to limit the number of parity bits to be stored and

transmitted. As noted in table 4-1, blocklengths on the order of 100 bits were of interest. LDPC codes are well suited to such a code rate and blocklength, and in contrast to turbo codes, LDPC codes probably provide a wider range of implementation options to trade performance against speed and complexity.

The vast majority of LDPC coding research has studied binary codes, though some results (references [23] and [24]) show that short nonbinary LDPC codes, defined over the finite field GF(256), can offer performance improvements of 1.0 to 1.3 dB over corresponding binary LDPC codes. However, decoders for these nonbinary codes are far more complex than decoders for the binary codes. For telecommand, the decoder is onboard the spacecraft, and so low decoding complexity is a particularly important metric. Hence, binary LDPC codes were selected over the nonbinary alternatives, despite the loss in performance. A protograph plus circulant construction (reference [20]) was selected because it permits fast, simple encoders, and is amenable to fast, well-structured decoders.

For long blocklength codes, research shows that good designs typically use about 3.5 edges per variable node (reference [25]); for short codes, this number must be increased to eliminate small trapping sets (reference [21]) and low weight codewords. A wide variety of protographs were studied, and that shown in figure 4-1 yields excellent performance for short block lengths.

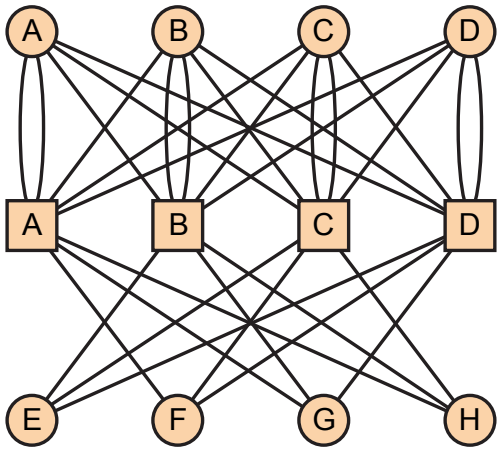

Figure 4-1: Protograph for Short Blocklength LDPC Codes

In this figure, circles represent variable nodes, and squares represent check nodes, or constraint equations. To construct a full-size LDPC code, one replicates the protograph M times, in this case yielding $8M$ variable nodes (one per transmitted channel symbol), and $4M$ constraints on those channel symbols. Each edge in the protograph becomes a bundle of M edges; these are cut, cyclically permuted, and reconnected. Such a cyclic permutation is called a circulant, and is represented as an $M \times M$ identity matrix that has been cyclically shifted to the right N places, or as a notational shorthand, simply by the integer N. A circulant must be chosen for each edge in the protograph; this is done using a variant of Progressive Edge Growth (PEG) (reference [22]), a greedy algorithm that intends to maximize loop length among other metrics. It should be noted that, while the protograph has parallel edges, they do not remain so when the full graph is constructed.

This process was performed twice to generate a pair of codes with blocklengths: (n=128, k=64) and (n=512, k=256), where k is the number of information bits encoded, and n is the number of transmitted code symbols. The resulting circulant choices are listed in table 4-2 and table 4-3, and the full parity check matrix, $H_{128\times64}$, for the (n=128, k=64) code is shown in figure 4-2. Here, dots are used to indicate 1s in the parity check matrix, and light blue lines are added to clarify the block-circulant structure.

Table 4-2: Circulant Selections for (128,64) LDPC Code

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
A	0,7	2	14	6		0	13	0
B	6	0,15	0	1	0		0	7
C	4	1	0,15	14	11	0		3
D	0	1	9	0,13	14	1	0	

Table 4-3: Circulant Selections for (512,256) LDPC Code

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
A	0,63	30	50	25		43	62	0
B	56	0,61	50	23	0		37	26
C	16	0	0,55	27	56	0		43
D	35	56	62	0,11	58	3	0	

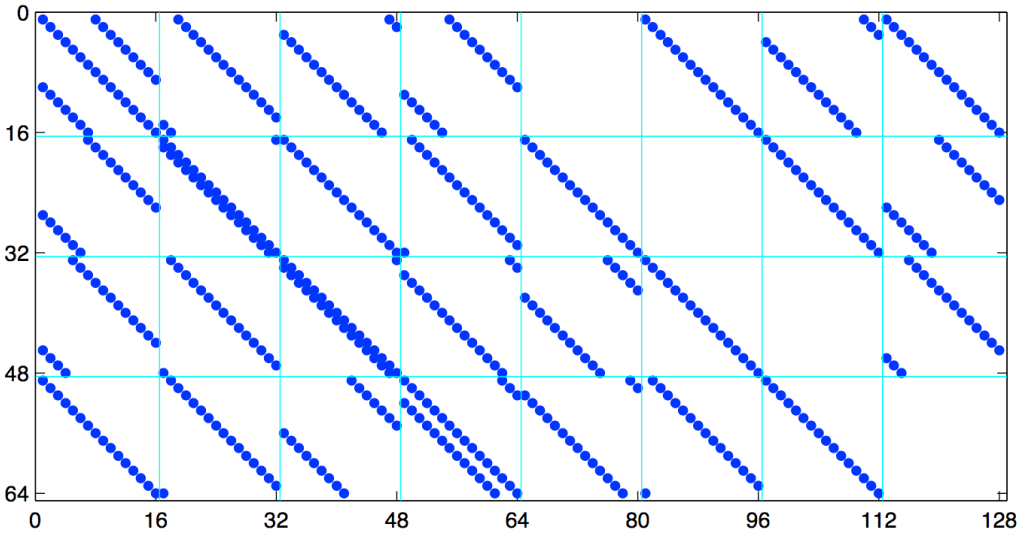

Figure 4-2: Parity Check Matrix for (128,64) LDPC Code

Some of the circulants selected are free parameters, because variable nodes and check nodes can be relabeled without changing the performance of the code. This freedom was used to put identity circulants on the main diagonal of the left half of the parity check matrix, and on the first sub-diagonal of the right half of *H* (considered as a block-matrix). This may simplify memory addressing in a hardware decoder implementation.

By computer search, the first terms of the weight distribution for the (128,64) code are:

Weight	Number of codewords
0	1
14	16
16	492
18	5424
20	81760
22	1127152
24	14869768

For the (512,256) code, the first terms of the weight distribution are approximately:

Weight	Number of codewords
0	1
40	64
42	704
44	6336
46	44736

Because these are quasi-cyclic codes, codewords appear in sets. Any quasi-cyclic shift of a codeword will be a codeword with the same weight; however, it may not be distinct if the codeword is a cyclic shift of itself. For example, the hexadecimal value 88882222 00000000 11114444 00000000 describes a 128-symbol codeword of the (128,64) LDPC code with weight 16, and this codeword has only four distinct cyclic shifts.

4.3 FILL DATA AND THE PSEUDORANDOMIZER

The TC standard was developed with the expectation that the transfer frames from the data link protocol sublayer would fill one or a few BCH codewords, and modestly larger data volumes were accommodated with the addition of PLOP-2 to the protocol. This is the same application profile for which the short-blocklength LDPC codes are designed (low data-rate command and emergency communications), and so the same design remains appropriate.

The coding and synchronization sublayer of the telecommand protocol consists of four operations, as shown in figure 4-3, from the perspective of the transmitter. The LDPC codes are a direct substitution for the BCH code, but with the additional changes that the randomizer is mandatory and is placed after the encoder. The remainder of this section discusses the implications more carefully.

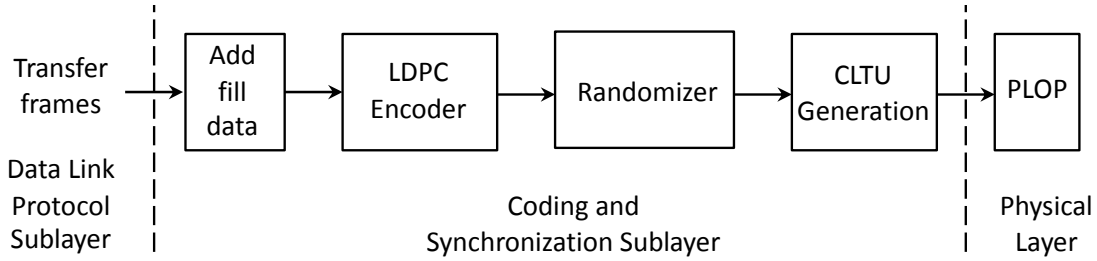

Figure 4-3: Components of the Coding and Synchronization Sublayer

With either the BCH code or the LDPC codes, fill data is added if necessary to complete the last codeword.

The randomizer is mandatory with the LDPC codes, because the codewords may be longer, and longer transition-free runs are possible in the unrandomized symbol stream. This is in contrast to BCH codes, where the parity symbols are inverted to prevent long transition-free runs.

With LDPC codes, the randomizer is placed after the encoder to improve the decoder's ability to detect synchronization errors. Because the LDPC codes are quasi-cyclic, a received codeword that is shifted by one symbol (perhaps caused by a 'slip' of the receiver's symbol tracking loop) may appear very similar to a different codeword, and be mis-corrected in this way. Placing the randomizer after the encoder prevents this problem.

CLTU generation (the addition of the start sequence and tail sequence) is performed with LDPC codewords in the same way as with BCH codewords, though with a longer start sequence for better detectability at low Signal-to-Noise Ratio (SNR). It should be noted that there are no synchronization markers between LDPC codewords, as there are in the TM standard. This is acceptable because codeword synchronization is established at the beginning of each CLTU. Markerless codeword synchronization may be used if desired for marginally improved performance.

4.4 IMPACT ON THE RADIO FREQUENCY AND MODULATION LAYER

The LDPC codes can operate at a lower SNR than the (63,56) BCH code for several reasons: the code rate is reduced to 1/2 from 0.89, they are typically decoded with a soft-decision decoder which saves about 2 dB over a hard-decision decoder as a rule of thumb, and both LDPC codes have longer codeword lengths than the BCH code. These changes need not have any significant impact on the transmit side of an uplink communications link, but there are two notable impacts on the receive side.

a) It will be advantageous for the spacecraft's radio receiver to work at a lower symbol SNR than with the current BCH code. By analogy, a communications system is only as capable as its weakest link in terms of coding gain. The BCH code is currently a weak link, and by replacing it with the stronger LDPC code, the system becomes

more capable. The next-weakest link may be the SNR at which the receiver can maintain symbol synchronization, and in this case, further performance gains may be won by improving the receiver's tracking loops. If one aims to achieve a particular Codeword Error Rate (CER), the potential gains are shown in figure 11-8. If the receiver can maintain synchronization at an SNR below that required by an LDPC code, then the full power of the code can be used; otherwise the benefits are smaller. Similarly, if one has a constraint on the Undetected Codeword Error Rate (UER), the potential gains are shown in figure 11-9.

b) An LDPC code is generally decoded using 'soft symbols', rather than the binary 'hard symbols' typically used for a BCH code. This provides a performance improvement of about 2 dB, but depends on a receiver that can produce soft outputs. This modification is not mandatory, however, for a belief propagation decoder can also operate on the hard symbols if necessary. Conversely, BCH codes are typically decoded with an algebraic decoder that operates on binary inputs, but there are soft-decision BCH decoding algorithms that can provide a performance improvement at a substantial cost in complexity.

5 CLTU GENERATION WITH BCH CODING

5.1 INTRODUCTION

At the sending end, the set of BCH codewords resulting from the BCH encoding are placed in a CLTU.

The CLTU includes fixed data patterns at the start and end. The Start Sequence provides the synchronization pattern and delimits the beginning of the first BCH codeword. The Tail Sequence marks the end of the CLTU.

5.2 LENGTH OF A CLTU

For a single ChannelAccess.request from the Data Link Protocol Sublayer, the Synchronization and Channel Coding Sublayer generates a set of BCH codewords, and that set of BCH codewords is placed in a single CLTU.

If N is the number of BCH codewords in the CLTU, then:

$$\text{Length of the CLTU} = 2 + (N + 1) * 8,$$

where the lengths are expressed in octets. The minimum length of a CLTU is therefore the case N=1, giving a CLTU of 18 octets.

The Recommended Standard (reference [1]) does not specify an upper limit for the length of a CLTU. However, it specifies a managed parameter for the maximum length of a CLTU for a Physical Channel. Also, the Space Link Extension Forward CLTU Service (reference [7]) has a management parameter that sets the maximum length of a CLTU for a service instance.

Section 11 contains a detailed analysis of the performance of the link based on the CLTU length.

5.3 FORMAT OF A CLTU

Figure 5-1 shows the format of a CLTU.

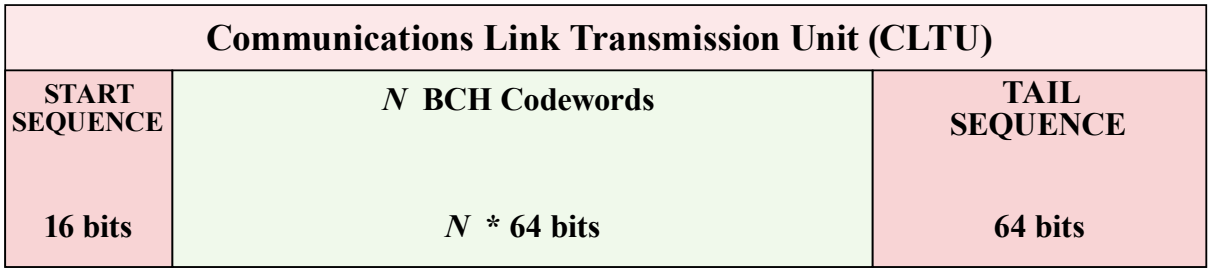

Figure 5-1: Format of a CLTU

Figure 5-2 extends the example from figure 3-2 by showing the resulting CLTU.

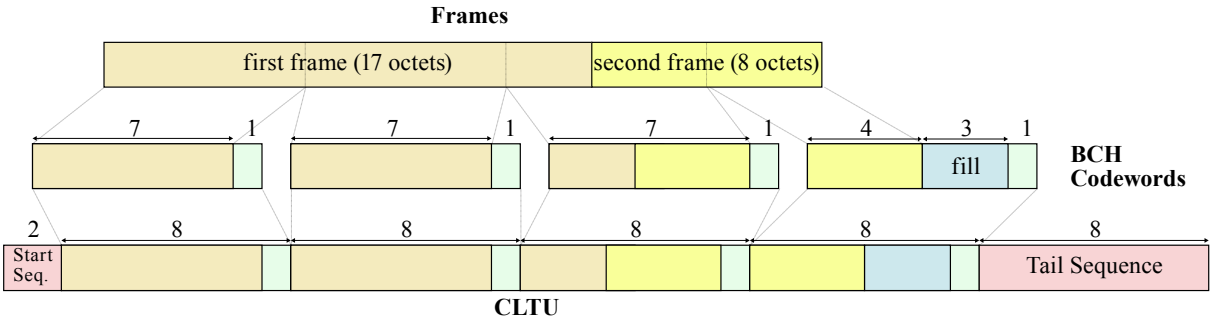

Figure 5-2: Example of BCH Codewords in a CLTU

5.4 START SEQUENCE

The Start Sequence is a 16-bit (two-octet) field containing the synchronization pattern shown in figure 5-3. The pattern has the hexadecimal value EB90.

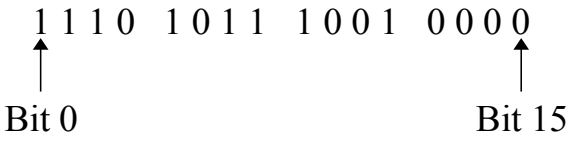

Figure 5-3: Start Sequence with BCH Coding

When the CLTU is transmitted, bit 0 of the Start Sequence is the first bit to be transmitted. The Start Sequence is immediately followed by the first BCH codeword of the CLTU.

The Start Sequence provides a synchronization pattern with low autocorrelation side lobes, thereby reducing the probability that the receiving end will mistakenly identify the pattern. It is clearly distinguishable from the Idle Sequence or Acquisition Sequence. The Idle Sequence and the Acquisition Sequence are discussed in section 7.

5.5 TAIL SEQUENCE

5.5.1 TAIL SEQUENCE FORMAT

The Tail Sequence is a 64-bit (8-octet) field containing the following bit pattern:

11000101 11000101 11000101 11000101 11000101 11000101 11000101 01111001.

The first seven octets of the Tail Sequence each contain the bit pattern 11000101 (hexadecimal C5), and the last octet has the bit pattern 01111001 (hexadecimal 79). So the Tail Sequence is hexadecimal C5 C5 C5 C5 C5 C5 C5 79.

The Tail Sequence immediately follows the last BCH codeword of the CLTU.

The Tail Sequence field has the same length as a BCH codeword, but its contents differ from a valid or correctable BCH codeword. As discussed in section 8, when the decoding process at the receiving end encounters an invalid or noncorrectable BCH codeword, it stops processing the CLTU and starts looking for the Start Sequence of the next CLTU.

In recommendations published in November 1995, the current Tail Sequence format replaced the earlier pattern described in 5.5.3 below. Annex D contains the theoretical background behind the choice of the current Tail Sequence pattern.

5.5.2 EARLIER TAIL SEQUENCE LENGTHS

The Tail Sequence field has the same length as a BCH codeword. This was also the case when one of the obsolete codeword length options was used.

As discussed in 3.3.2, earlier CCSDS recommendations for Telecommand Channel Coding included further codeword length options of 40 bits, 48 bits, and 56 bits. For a Tail Sequence with one of these other lengths, the last octet contained hexadecimal 79, and the other octets contained hexadecimal C5.

The shorter Tail Sequence lengths were eliminated with the associated BCH codeword length options, in recommendations published in June 2000.

5.5.3 EARLIER TAIL SEQUENCE PATTERN

Earlier CCSDS recommendations for Telecommand Channel Coding defined a different bit pattern for the Tail Sequence. The pattern consisted of alternating ‘0’ and ‘1’ bits, starting with a ‘0’ bit, so that each octet in the Tail Sequence contained hexadecimal 55.

In recommendations published in November 1995, the old Tail Sequence pattern was replaced by the current pattern described in 5.5.1 above. The current Tail Sequence provides a more distinctive pattern, which is more reliably detected as the end of a CLTU (see performance discussion in section 11 and annex E).

6 CLTU GENERATION WITH LDPC CODING

6.1 INTRODUCTION

At the sending end, the set of randomized LDPC codewords resulting from the LDPC encoding and pseudo-randomization are placed in a CLTU.

The CLTU includes a fixed data pattern at the start, and optionally also at the end when the (128,64) LDPC code is used. The Start Sequence provides the synchronization pattern and delimits the beginning of the first LDPC codeword. When included, the Tail Sequence marks the end of the CLTU.

6.2 LENGTH OF A CLTU

For a single ChannelAccess.request from the Data Link Protocol Sublayer, the Synchronization and Channel Coding Sublayer generates a set of LDPC codewords, and that set of LDPC codewords is placed in a single CLTU.

The length of a CLTU is determined by the LDPC code selected, the number of codewords, N, and if the optional Tail Sequence is used with the (128,64) code. There are three cases:

- (128,64) LDPC code without Tail Sequence: Length of CLTU = N*16 + 8
- (128,64) LDPC code with Tail Sequence: Length of CLTU = N*16 + 24
- (512,256) LDPC code without Tail Sequence: Length of CLTU = N*64 + 8

where the lengths are expressed in octets. The minimum length of a CLTU occurs with the first case when N=1, giving a CLTU of 24 octets.

The Recommended Standard (reference [1]) does not specify an upper limit for the length of a CLTU. However, it specifies a managed parameter for the maximum length of a CLTU for a Physical Channel. Also, the Space Link Extension Forward CLTU Service (reference [7]) has a management parameter which sets the maximum length of a CLTU for a service instance.

6.3 FORMAT OF A CLTU

Figure 6-1 shows the format of a CLTU. When the (128,64) LDPC code is used, the tail sequence is optional; when the (512,256) code is used, there is no tail sequence.

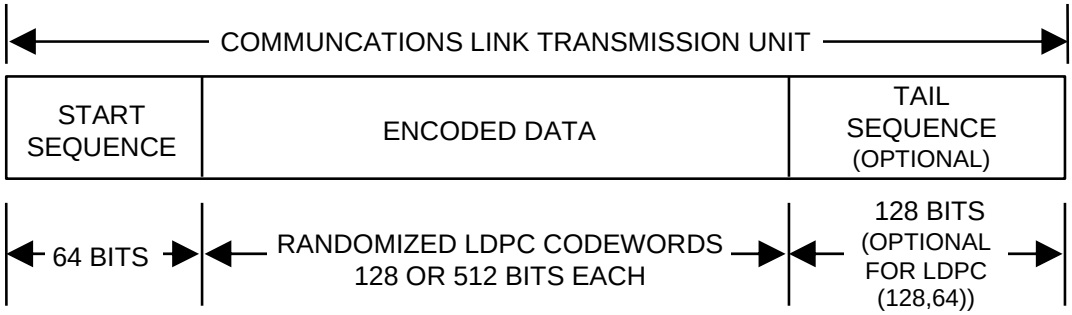

Figure 6-1: Format of a CLTU with LDPC Coding

6.4 START SEQUENCE

The Start Sequence is a 64-bit (eight-octet) field containing the synchronization pattern shown in figure 6-2. The pattern has the hexadecimal value 0347 76C7 2728 95B0.

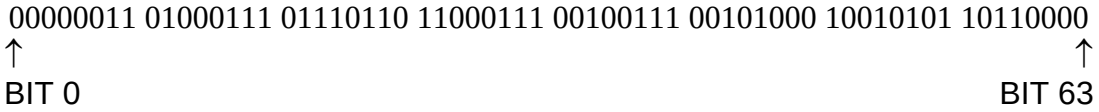

Figure 6-2: Start Sequence with LDPC Coding

When the CLTU is transmitted, bit 0 of the Start Sequence is the first bit to be transmitted. The Start Sequence is immediately followed by the first randomized LDPC codeword of the CLTU.

The Start Sequence provides a synchronization pattern with low autocorrelation side lobes, thereby reducing the probability that the receiving end will mistakenly identify the pattern. It is clearly distinguishable from the Idle Sequence or Acquisition Sequence. The Idle Sequence and the Acquisition Sequence are discussed in section 7.

6.5 TAIL SEQUENCE

6.5.1 TAIL SEQUENCE FORMAT

The Tail Sequence is optional when the (128,64) LDPC code is used. It is a 128-bit (16-octet) field containing the following bit pattern:

```
01010101 01010101 01010101 01010110 10101010 10101010 10101010 10101010
         01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101
↑                                                                               ↑
BIT 0                                                                       BIT 127
```

In hexadecimal, the tail sequence is 5555 5556 AAAA AAAA 5555 5555 5555 5555. The Tail Sequence immediately follows the last randomized LDPC codeword of the CLTU.

The Tail Sequence field has the same length as an LDPC codeword, has a high bit transition density, and if derandomized, it is reasonably distinct from any LDPC codeword.

If the tail sequence is used, then it may be identified either by pattern matching, or by using an LDPC decoder that will reliably report a decoding error when presented with a noisy version of this binary sequence. If the tail sequence is not used, the end of a CLTU can only be found by using an LDPC decoder that reliably identifies binary sequences that are not codewords.

7 PHYSICAL LAYER OPERATIONS PROCEDURES

7.1 GENERAL

The Physical Layer interfaces directly with the transmission medium, typically a radio path through space. As such, it activates and deactivates the physical connection between the transmitter and the receiving spacecraft.

As the Physical Layer is mainly concerned with providing the physical connection between the transmitter and the receiving spacecraft, the major specification of this layer is found in reference [6]. However, a small part is specified by the TC Synchronization and Channel Coding Recommended Standard (reference [1]).

The Physical Layer modulates the CLTUs received from the Synchronization and Channel Coding Sublayer onto the RF carrier. Reference [1] defines two PLOPs:

PLOP-1. The Physical Channel is deactivated between each CLTU, so (at least) bit synchronization is lost. In this case, the Physical Channel is reactivated with each CLTU.

PLOP-2. The Physical Channel is not deactivated after each CLTU, and an idle sequence ensures that the bit synchronization is maintained at the receiving end.

Reference [1] specifies that PLOP-2 be used for missions whose planning begins after September 2010. PLOP-1 may still be used in ground equipment for the support of legacy missions.

7.2 ELEMENTS

7.2.1 COMMUNICATIONS SESSION

The PLOPs specify the sequence of operations performed during a communications session. A communications session is defined as a continuous period of time during which the signal path is established for the communications channel.

A session begins when the Physical Layer establishes the physical connection from the transmitter to the receiver, by the provision of the RF carrier. The session ends with the removal of the carrier.

7.2.2 ACQUISITION SEQUENCE

The purpose of the Acquisition Sequence is to enable the receiving end to acquire bit synchronization. It is a bit pattern with a high transition density, consisting of alternating ‘1’ and ‘0’ bits, starting with either a ‘1’ or a ‘0’.

The length of the Acquisition Sequence is selected for a mission according to the communications link performance properties of the mission. The preferred minimum length is 16 octets (128 bits). The length does not have to be an integral number of octets.

7.2.3 IDLE SEQUENCE

The purpose of the Idle Sequence is to enable the receiving end to maintain bit synchronization when no CLTUs are being transmitted. It is a bit pattern with a high transition density, consisting of alternating '1' and '0' bits, starting with either a '1' or a '0'.

An Idle Sequence can have any length and does not have to be an integral number of octets.

NOTE – Earlier CCSDS recommendations for Telecommand Channel Coding included the requirement that the bit pattern of an Idle Sequence should start with a '0'. This requirement was related to the earlier Tail Sequence pattern (see 5.5.3), which also consists of alternating '1' and '0', starting with a '0'. With the improved performance of the current Tail Sequence pattern (see 5.5.1), the requirement was removed.

7.2.4 CARRIER MODULATION MODES

7.2.4.1 General

The Carrier Modulation Modes (CMMs) consist of different states of data modulation upon the RF carrier that creates the physical telecommand channel. Table 7-1 shows the four basic CMMs which are combined to perform the PLOPs.

Table 7-1: Carrier Modulation Modes

Mode	Description
CMM-1	Unmodulated carrier only
CMM-2	Carrier Modulated with Acquisition Sequence
CMM-3	Carrier Modulated with CLTUs
CMM-4	Carrier Modulated with Idle Sequence

7.2.4.2 CMM-1

CMM-1 is concerned with the establishment and maintenance of the radio-frequency part of the physical connection. The unmodulated carrier mode is defined as the state in which no telecommand modulation (i.e., data modulation) is present.

In establishing and maintaining the connection, the sending end uses information about the status of the receiving spacecraft transponders. This information is carried by the No RF Available Flag in the Communications Link Control Word (CLCW), as specified in reference [3].

7.2.4.3 CMM-2

CMM-2 is concerned with the establishment of the modulation part of the physical connection. The sending end transmits an Acquisition Sequence with the length selected for the mission, to enable the receiving end to acquire bit synchronization.

The sending end may use information about the quality of the received bit stream, carried by the No Bit Lock Flag in the CLCW, as specified in reference [3]. The use of the No Bit Lock Flag is optional and mission-specific. If used, the flag provides the same service throughout CMM-3 and CMM-4.

7.2.4.4 CMM-3

CMM-3 is concerned with the transmission of one CLTU on the physical connection.

7.2.4.5 CMM-4

CMM-4 is concerned with the maintenance of the modulation part of the physical connection. The Idle Sequence is transmitted to enable the receiving end to maintain bit synchronization.

The maximum length of the Idle Sequence is unconstrained and is dictated by the timing of the telecommand operations (e.g., when no CLTU is available).

7.3 PLOP-1

The sequence of CMMs for PLOP-1 is shown in figure 7-1, reproduced from Recommended Standard (reference [1]).

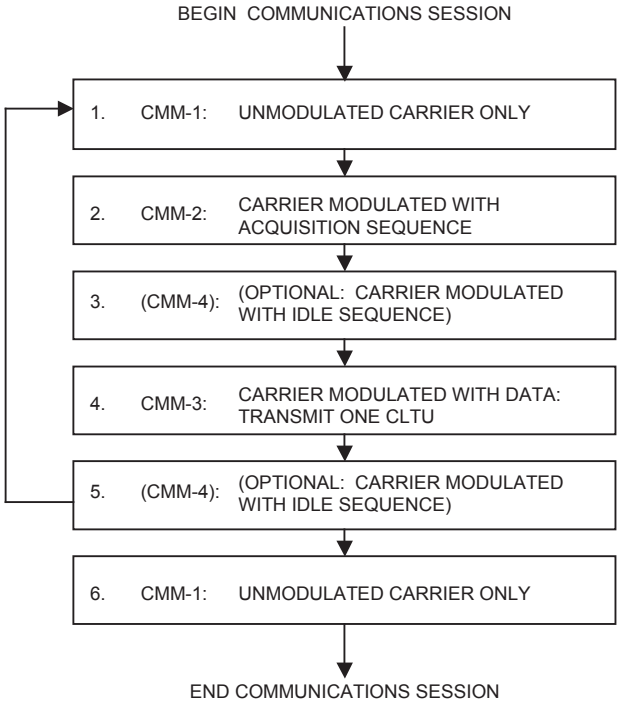

Figure 7-1: PLOP-1 Carrier Modulation Modes

In PLOP-1, the Physical Channel returns to CMM-1 (unmodulated carrier only) after each CLTU and, as a result, bit synchronization is lost at the receiving end. Therefore the Acquisition Sequence (CMM-2) is transmitted to enable the receiving end to regain bit synchronization, before transmitting the next CLTU (CMM-3). Optional Idle Sequences of any desired lengths can be transmitted (CMM-4) before and after a CLTU, to suit the timing conditions.

7.4 PLOP-2

The sequence of CMMs for PLOP-2 is shown in figure 7-2, reproduced from Recommended Standard (reference [1]).

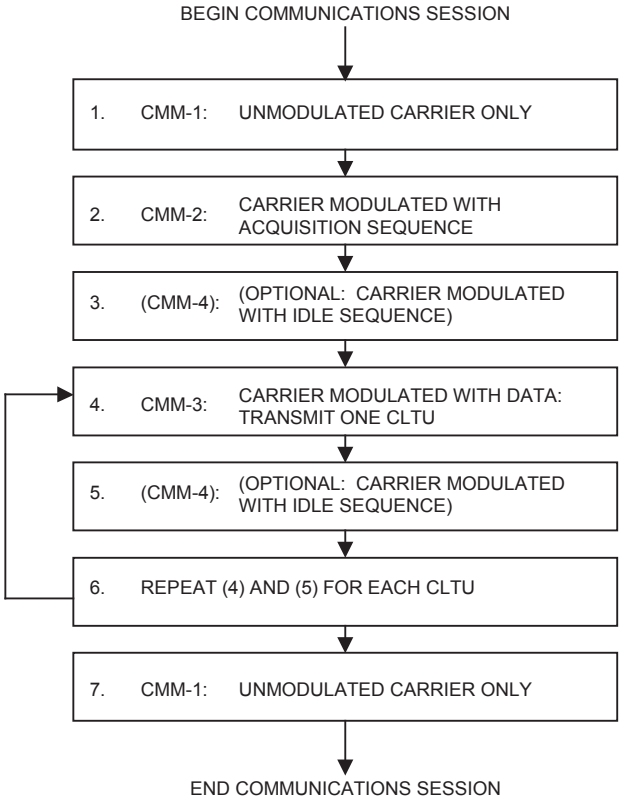

Figure 7-2: PLOP-2 Carrier Modulation Modes

In PLOP-2, CMM-1 is only used at the start and end of a communications session. The Acquisition Sequence (CMM-2) is transmitted only once, when it is used at the start of a session to enable the receiving end to acquire bit synchronization.

Successive CLTU transmissions (CMM-3) are separated only by an optional Idle Sequence (CMM-4) of any desired length. For example, if no CLTU is available for transmission, the Idle Sequence continues until a CLTU becomes available or the end of the session is reached. The Idle Sequence enables bit synchronization to be maintained at the receiving end.

The Idle Sequence between CLTUs is optional, and it can be omitted if the next CLTU is available. However, reference [1] recommends that an Idle Sequence of at least one octet should be transmitted between two CLTUs (see 8.7 below).

7.5 SELECTION OF PLOP-1 OR PLOP-2

Reference [1] specifies that PLOP-2 be used for missions whose planning begins after September 2010. PLOP-1 may still be used in ground equipment for the support of legacy missions.

With PLOP-1, the Acquisition Sequence is transmitted before every CLTU. The maximum throughput of the channel is therefore reduced compared to a similar channel operated with PLOP-2. The magnitude of the reduction depends on factors such as:

- the average CLTU length;
- the lengths of any optional Idle Sequences in PLOP-1 and in PLOP-2;
- the duration of the CMM-1 transmission between CLTUs in PLOP-1.

However, PLOP-1 can provide improved reliability, thereby reducing the need for retransmissions by the higher layers. With PLOP-2, there is a small risk that the receiving end fails to detect the boundary between successive CLTUs. With PLOP-1, this risk is avoided. The difference depends on the behavior of the CLTU reception procedures at the receiving end and is discussed in 8.7 below.

7.6 FORMAL INTERFACE TO SUBLAYER ABOVE

The Recommended Standard (reference [1]) does not define the service provided by the PLOP and the Physical Layer to the Synchronization and Channel Coding Sublayer. Clearly, the PLOP accepts CLTUs from the Synchronization and Channel Coding Sublayer, but there is no formal definition of the service interface. The Recommended Standard makes no mention of any information passing from the PLOP and the Physical Layer to the Synchronization and Channel Coding Sublayer.

The Space Link Extension Forward CLTU Service (reference [7]) defines a service for transmitting CLTUs. The underlying function is the service provided by the PLOP as defined in reference [1]. The Forward CLTU Service includes operations for sending a CLTU and for reporting events in the performance of the service. Parameters include, for example, the minimum delay between transmitting one CLTU and the next. Implementers may find it helpful to consider the operations and parameters of the Forward CLTU Service when designing the interfaces presented by the PLOP and the Physical Layer.

8 CLTU RECEPTION PROCEDURES

8.1 INTRODUCTION

Sections 3, 4, and 7 discuss the procedures at the sending end of the space link. This section looks at the procedures for reception of a CLTU at the receiving end.

At the receiving end, the Physical Layer delivers a stream of channel bits to the Synchronization and Channel Coding Sublayer, together with information on the state of the physical communications channel. The Synchronization and Channel Coding Sublayer:

- scans the input stream of bits, looking for the Start Sequence of a CLTU;
- if necessary, uses the Start Sequence to resolve the data ambiguity (sense of '1' and '0') in the input stream of bits;
- decodes and derandomizes the following BCH codewords or LDPC codewords until a decoding failure is detected, or a CLTU tail sequence is detected by pattern matching;
- delivers the data, which consists of candidate Transfer Frames, to the Data Link Protocol Sublayer;
- scans the input stream of bits, looking for the Start Sequence of the next CLTU, and so on.

Decoding and derandomization are performed differently, depending on whether BCH or LDPC coding is in use, as described in section 9.

The following subsections discuss the CLTU reception procedures in more detail.

8.2 CLTU RECEPTION LOGIC

The definition of the CLTU reception logic in the Recommended Standard (reference [1]) uses the state diagram shown in figure 8-1. There are three states:

– S1, INACTIVE, when no bits are being received from the Physical Layer, typically because no bit modulation is detected or because bit lock is not achieved;

– S2, SEARCH, when the bits received from the Physical Layer are being searched for a CLTU Start Sequence;

– S3, DECODE, when BCH codewords or LDPC codewords are being decoded.

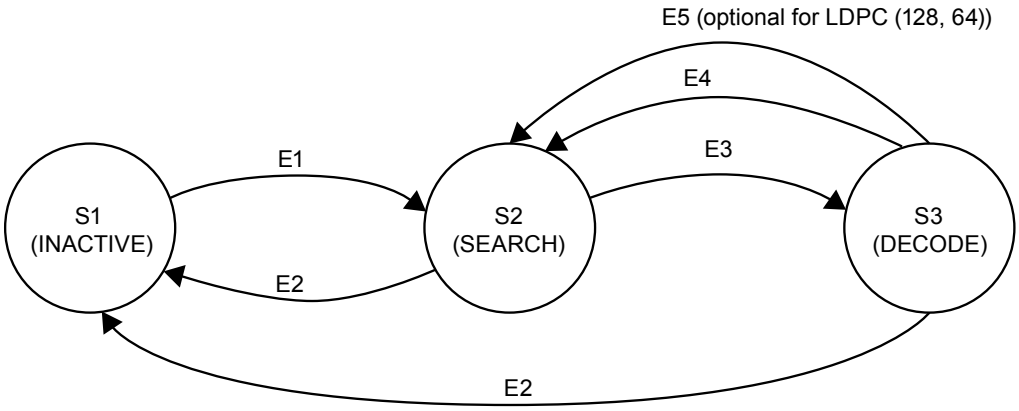

Figure 8-1: State Diagram for CLTU Reception Logic

The CLTU reception logic can also be summarized by the state table in table 8-1.

Table 8-1: State Table for CLTU Reception Logic

	INACTIVE State S1	**SEARCH State S2**	**DECODE State S3**
CHANNEL ACTIVATION Event E1	Go to SEARCH	-	-
CHANNEL DEACTIVATION Event E2	-	Go to INACTIVE	End processing of current CLTU; Go to INACTIVE
START SEQUENCE FOUND Event E3	-	Go to DECODE	-
CODEWORD REJECTION Event E4	-	-	End processing of current CLTU; Go to SEARCH
TAIL SEQUENCE FOUND Event E5	-	-	End processing of current CLTU; Go to SEARCH

8.3 INACTIVE STATE

The initial state for the CLTU reception procedure is the INACTIVE state (S1).

In this state, the Physical Layer is not sending any bits to the Synchronization and Channel Coding Sublayer. The Physical Layer does not have bit synchronization of an RF signal.

When the Physical Layer achieves bit synchronization, it signals CHANNEL ACTIVATION (Event E1) and starts sending the demodulated bits to the Synchronization and Channel Coding Sublayer. The CLTU reception procedure goes to SEARCH state (S2) and starts processing the received bits.

If the RF signal is lost or there is a loss of bit synchronization, the Physical Layer signals CHANNEL DEACTIVATION (Event E2), and the CLTU reception procedure returns to the INACTIVE state (S1).

8.4 SEARCH STATE

8.4.1 GENERAL

In the SEARCH state (S2) the CLTU reception procedure searches the received stream of bits, looking for the Start Sequence. As long as the Start Sequence is not found, the incoming bits are discarded.

Figure 5-3 and figure 6-2 show the bit pattern of the Start Sequence when BCH and LDPC coding are used, respectively.

Once the Start Sequence is found, this is START SEQUENCE FOUND (Event E3) and it marks the start of a CLTU. The Start Sequence itself is discarded and the CLTU reception procedure goes to DECODE state (S3). The first bit following the Start Sequence is the first bit of the first BCH codeword or LDPC codeword of the CLTU.

8.4.2 RESOLVING THE SENSE OF ‘0’ AND ‘1’

The ability of the Physical Layer to resolve the data ambiguity (sense of ‘1’ and ‘0’) in the received bit stream depends on the modulation scheme in use. If the Physical Layer is unable to resolve the data ambiguity, then the CLTU reception procedure uses the Start Sequence for the purpose. In this case, the CLTU reception procedure searches the received stream of bits, looking for the Start Sequence or its inverse.

Figure 8-2 shows the inverse bit pattern of the Start Sequence, which has the hexadecimal value 146F, when BCH coding is used. Figure 8-3 shows the inverse bit pattern of the Start Sequence, which has the hexadecimal value FCB8 8938 D8D7 6A4F, when LDPC coding is used.

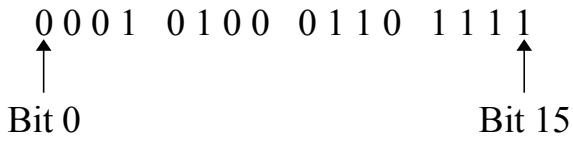

Figure 8-2: Inverse of the Start Sequence Bit Pattern with BCH Coding

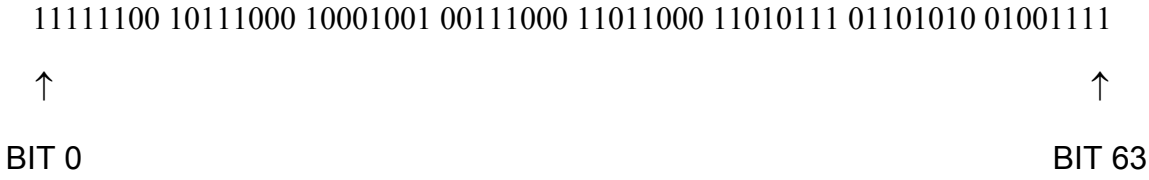

Figure 8-3: Inverse of the Start Sequence Bit Pattern with LDPC Coding

If the inverse of the Start Sequence is found, then this also triggers the START SEQUENCE FOUND (Event E3). The CLTU reception procedure inverts all the following bits in the received stream of bits until the next time the SEARCH state is entered, when it again searches for the Start Sequence or its inverse.

8.4.3 ERRORS IN THE START SEQUENCE

With BCH coding, the Recommended Standard (reference [1]) specifies two options in the CLTU reception procedure for the handling of errors in the Start Sequence: either no errors are accepted, or a single error is accepted.

If no errors are accepted, then the received bit pattern is accepted as a valid Start Sequence only if it exactly matches the expected bit pattern. This option is used with the error-detecting mode for the BCH decoding.

If a single error is accepted, then the received bit pattern is accepted as a valid Start Sequence either if it exactly matches the expected bit pattern or if it differs by one bit from the expected bit pattern. This option is used with the error-correcting mode for the BCH decoding.

NOTE – If the option for accepting a single error applies, and the CLTU reception procedure is searching for the Start Sequence or its inverse, then a received bit pattern that differs by one bit from the Start Sequence inverse is also accepted.

With LDPC coding, improved performance can be achieved by using a soft-symbol receiver. Identifying a valid Start Sequence from soft symbols is a more complex problem, as addressed in 11.9.1.

8.5 DECODE STATE

8.5.1 GENERAL

In the DECODE state (S3), the CLTU reception procedure decodes the BCH codewords or LDPC codewords in the received stream of bits. The DECODE state is entered from the SEARCH state when the Start Sequence is found. The first bit following the Start Sequence is the first bit of the first BCH codeword or LDPC codewords of the CLTU.

The CLTU reception procedure partitions the incoming bit stream into codewords of the anticipated size (64 bits for the BCH code, and 128 or 512 bits for the two LDPC codes). When LDPC coding is used, the codewords are derandomized. The CLTU reception procedure attempts to decode each codeword in turn, until one of three events occurs: CHANNEL DEACTIVATION (Event E2), CODEWORD REJECTION (Event E4), or TAIL SEQUENCE FOUND (Event E5).

The CLTU reception procedure stops when the decoder encounters a codeword that does not decode successfully. It rejects and discards the failed codeword, and this is codeword REJECTION (Event E4) which marks the end of a CLTU. When used, the CLTU Tail Sequence is designed to trigger a CODEWORD REJECTION event. Following a CODEWORD REJECTION event, the CLTU reception procedure stops decoding and returns to SEARCH state. When no Tail Sequence is used, the search for the Start Sequence must resume at the beginning of the uncorrected codeword. When a Tail Sequence is used, the search may resume at the end of the uncorrected codeword.

When the CLTU Tail Sequence is used, the DECODE procedure may also be stopped when the Tail Sequence is recognized by pattern-matching. The Tail Sequence is discarded, and TAIL SEQUENCE FOUND (Event E5) marks the end of a CLTU.

Occurrence of a CHANNEL DEACTIVATION (Event E2) also stops the decoding. The CLTU reception procedure discards any partially received codeword and returns to INACTIVE state.

The data contents of successfully decoded codewords are transferred to the Data Link Protocol Sublayer. When BCH coding is used, then if randomization is in use for the Physical Channel, the data are derandomized before transfer (see section 9). The CLTU reception procedure also delivers an indication of the start and end of the data extracted from the CLTU.

The CLTU reception procedure may use a variety of algorithms for decoding codewords. BCH codes are normally paired with a Physical Layer that produces a hard decision output, as a sequence of binary channel symbols which are either ‘0’ or ‘1’. In this case, one of two decoding algorithms is expected: either Error Detection as described in 8.5.2 or Error Correction as described in 8.5.3.

If the Physical Layer produces a soft decision output with more quantization levels, the CLTU reception procedure receives a sequence of channel symbols with a larger range of possible values. In the soft decision case, the principle of searching for the codeword with the minimum distance from the received sequence still applies. The discussion of such algorithms for BCH codes is outside the scope of this report. LDPC codes are normally used with a Physical Layer that produces soft decisions, and three possible algorithms are described briefly in 11.6, 11.7, and 11.8. The typical iterative decoder in 11.6 is perhaps the most common, and is widely used as a benchmark for both error correction performance and decoder complexity. The layered belief propagation decoder in 11.7 may offer advantages in performance or complexity or both, but there is no clear consensus. The Most Reliable Basis decoder in 11.8 unquestionably offers considerable performance improvement at the cost of complexity.

8.5.2 ERROR DETECTION MODE FOR BCH CODING

8.5.2.1 Overview

One of the modes of operation for BCH decoding is the error detection mode, which is also called Triple Error Detection (TED) because it is capable of detecting up to three errors in a BCH codeword.

In TED mode, no attempt is made to correct any errors. The decoding of a BCH codeword is considered successful only if no errors are found. If the decoding process detects any errors, then the codeword is rejected.

In TED mode, the value of the received Filler Bit at the end of a BCH codeword is ignored.

The Recommended Standard (reference [1]) states that, when TED mode is in use, no error is allowed in the Start Sequence (see 8.4.3).

Because the received codeword can have more than three errors, there is a small risk that the TED mode decoding fails to detect errors. There is also a small risk that transmission errors can transform the Tail Sequence so that it appears to be an error-free codeword. Section 11 includes information on the performance of the BCH code in TED mode, including the probabilities of failing to detect the Tail Sequence.

8.5.2.2 An Implementation for Error Detection Mode

The Recommended Standard (reference [1]) includes a schematic diagram for the implementation of the BCH encoding using a linear feedback shift register. A similar method can be used by a BCH decoder in TED mode to calculate the seven parity bits.

A BCH decoder for TED mode could be implemented as follows:

– the first 56 bits from the received BCH codeword are used to calculate seven parity bits;

- the next seven bits from the received BCH codeword are inverted and compared with the calculated parity bits;
- if any difference is found between the received parity bits and the calculated parity bits, the BCH codeword is rejected.

The value of the Filler Bit, which is the last bit of the received BCH codeword, is not used by the decoder.

8.5.3 ERROR CORRECTION MODE FOR BCH CODING

8.5.3.1 Overview

One of the modes of operation for BCH decoding is the error correction mode, which has the full name Single Error Correction and Double Error Detection mode because it is capable of correcting a single error and detecting two errors in a BCH codeword. For convenience it is generally called Single Error Correction (SEC) mode.

In SEC mode, the decoding of a BCH codeword is considered successful if no errors are found or if a single error is found and corrected. If the decoding process detects uncorrectable errors, then the codeword is rejected.

In SEC mode, the value of the received Filler Bit at the end of a BCH codeword is ignored unless the obsolete Filler Bit Augmentation Algorithm described in 8.5.3.3 is in use.

Because the received codeword can have more than two errors, there is a small risk that the SEC mode decoding fails to detect errors or that it makes a faulty correction. There is also a risk that transmission errors can transform the Tail Sequence so that it appears to be an error-free or correctable codeword. Section 11 includes information on the performance of the BCH code in SEC mode, including the probabilities of failing to detect the Tail Sequence.

8.5.3.2 An Implementation for Error Correction Mode

Figure 8-4 shows a schematic diagram for a decoder for SEC mode, using shift registers. It consists of the following devices:

- an inverter, which inverts the Parity Bits of the incoming BCH codeword;
- a 63-stage Buffer Register (BR);
- a six-stage upper Syndrome Register (SR), which calculates the syndrome;
- a six-stage lower Position Location Register (PLR);
- a seven-input AND gate for error correction at the appropriate location;
- a one-stage Even/Odd Detector (EOD) which calculates the parity;

– a Hold device, which applies the EOD output to the AND gate;

– a six-input OR gate, with inputs tied to the SR, which outputs Syndrome Binary (SB);

– an XOR gate (Codeword Rejection [CR]), which indicates an uncorrectable error, as described below;

– an error correction XOR gate at the output.

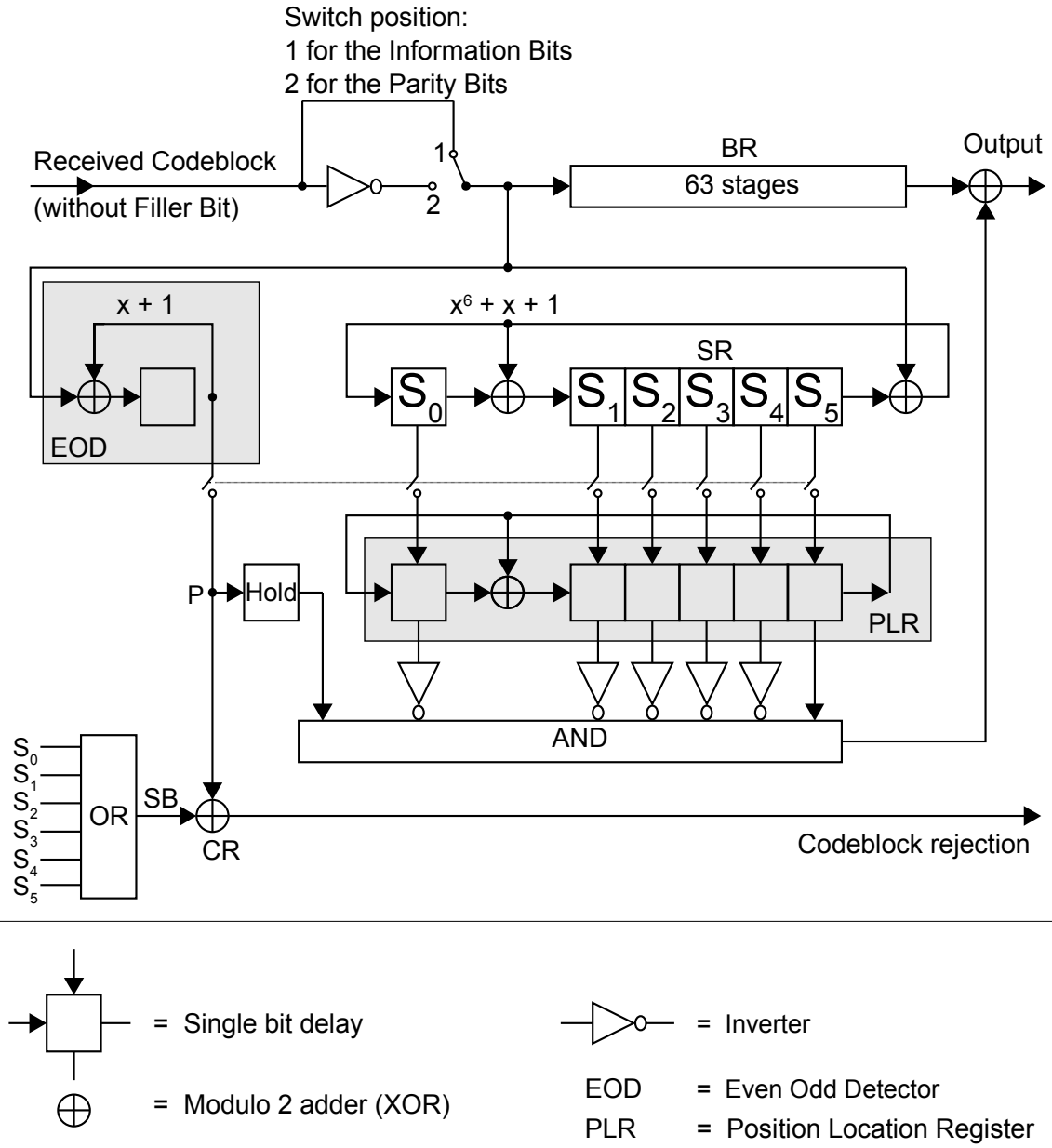

Figure 8-4: An Implementation for the Error Correction Mode Decoder

The decoder operates as follows.

The EOD and the SR are all set to zeros before receiving a new BCH codeword. The first 63 bits of a BCH codeword are input to the decoder.

NOTE – The Filler Bit at the end of a BCH codeword is ignored.

The switch at the input is in position 1 during the first 56 bits of each BCH codeword and in position 2 (which uses the inverter) for the next seven bits. The resulting 63 bits are input to the BR, the SR, and the EOD.

After the 63rd bit is received:

- the BR is full;
- the SR contains the error syndrome (S = S0, S1, S2, S3, S4, S5);
- the EOD outputs a bit P, which is '1' if the parity is odd and '0' if it is even.

The set of seven switches below SR closes momentarily to:

- transfer the contents of the SR to the PLR;
- transfer the output of the EOD to the XOR gate CR and to the Hold device.

At the same time, the contents of the SR are input to the OR gate (the transfer path is not shown in the figure). The output of the OR gate is SB, and:

- SB = '0' if the error syndrome S (S = S0, S1, S2, S3, S4, S5) is 'all 0';
- SB = '1' if S>0.

Then the SR and the EOD are all set to zeros to prepare for the next BCH codeword. The seven switches open before the 63 bits of the next BCH codeword are clocked in, and the process repeats.

While the SR is solving for the syndromes of the next BCH codeword, the PLR is solving for the location of any single error in the current one. The Hold applies the output of the EOD to the AND gate for the next 63 shifts of the PLR. In the meantime, as the next BCH codeword is shifted into the buffer BR, the current one is shifted out, and if there is an error, it is corrected at the output XOR gate. This correction occurs when the contents of the PLR register are '000001' and the Hold output is '1'.

The XOR gate CR determines whether the codeword is rejected, as summarized in table 8-2.

Table 8-2: Codeword Decisions for Error Correction Mode Decoder

EOD OUTPUT (P)	BINARY SYNDROME VALUE (SB)	DECISION
'0'	**'0'**	No Errors Detected The output of the AND gate is '0', so no correction is made. The output of the XOR gate CR is '0', so the codeword is accepted.
'0'	**'1'**	CODEWORD REJECTION: Even Number of Errors Detected The output of the AND gate is '0', so no correction is made. The output of the XOR gate CR is '1', so the codeword is rejected.
'1'	**'0'**	CODEWORD REJECTION: Odd Number of Errors Detected in an Apparently Correct Codeword The output of the AND gate is '0', so no correction is made. The output of the XOR gate CR is '1', so the codeword is rejected.
'1'	**'1'**	Correction of Single Error: Odd Number of Errors Detected in an Incorrect Codeword The output of the AND gate is '1' at the position where the contents of PLR are '000001', correcting the single error. The output of the XOR gate CR is '0', so the codeword is accepted.

8.5.3.3 The Obsolete Filler Bit Augmentation Algorithm

With the earlier Tail Sequence pattern (see 5.5.3), there is a higher risk of missing (failing to recognize) the Tail Sequence. If a single bit of this Tail Sequence becomes altered in the channel, the received sequence may no longer be recognized in SEC mode as uncorrectable, and an improper correction may take place. Because the expected CODEWORD REJECTION does not occur, the Tail Sequence is missed. Missing a Tail Sequence has far-reaching effects, since it is possible that the CLTU reception procedure will not be in SEARCH state for the next CLTU and therefore the entire subsequent CLTU may be missed.

The Filler Bit Augmentation algorithm was designed to reduce the problem significantly and improve the reliability of identifying a Tail Sequence. The algorithm uses the Filler Bit to selectively inhibit the SEC mode. Each valid BCH codeword terminates on a Filler Bit which is always set to '0' but the Tail Sequence terminates in a '1'. This fact can be exploited at the receiving end to inhibit the error correction function of the SEC mode whenever the received

Filler Bit is a '1'. When the algorithm is used, the decision in the last row of table 8-2 is expanded as shown in table 8-3. This algorithm substantially improves the ability to identify a Tail Sequence with the earlier Tail Sequence pattern.

NOTE – The earlier Tail Sequence pattern consists of an alternating pattern of 'ones' and 'zeros' starting with a 'zero', and so ends with a 'one'. The current Tail Sequence pattern (see 5.5.1) also ends with a 'one' and is therefore compatible with a receiving end that uses the Filler Bit Augmentation algorithm.

Table 8-3: Decoding Strategy Modified for Filler Bit Augmentation

FILLER BIT VALUE	DECISION WHEN FILLER BIT AUGMENTATION ALGORITHM IN USE
'0'	Correction of Single Error: Odd Number of Errors Detected in an Incorrect Codeword The output of the AND gate is '1' at the position where the contents of PLR are '000001', correcting the single error. The output of the XOR gate is '0', and the Filler Bit is '0', so the codeword is accepted.
'1'	CODEWORD REJECTION: Odd Number of Errors Detected and Filler Bit is '1' The output of the AND gate is '1' at the position where the contents of PLR are '000001', correcting the single error. The output of the XOR gate is '0', but the Filler Bit is '1', so the codeword is rejected.
The table replaces the DECISION box in the last row of table 8-2.	

While the algorithm substantially reduces the likelihood of missing a Tail Sequence, there is a penalty in the form of a slight increase in frame rejection rate. A BCH codeword containing one error which would otherwise be corrected and accepted may also have an error in its Filler Bit. If this algorithm is not used, the erroneous Filler Bit is not tested and the codeword is corrected and accepted. Under this algorithm, such a codeword is rejected, even though it is correctable.

The current Tail Sequence pattern (see 5.5.1) has an improved performance compared to the earlier pattern. The distance to the nearest codewords is increased by one. That is, it differs in more bits from the nearest codewords, so it takes more channel errors to corrupt it into a correctable codeword. Therefore the Filler Bit Augmentation algorithm is not included in the current issue of the Recommended Standard (reference [1]).

Section 11 includes information on the performance of the current Tail Sequence pattern and annex E gives statistics on the performance of obsolete features including the Filler Bit Augmentation algorithm.

8.6 INTERFACE TO DATA LINK PROTOCOL SUBLAYER

8.6.1 DEFINITION OF THE INTERFACE

8.6.1.1 Formal Definition

The Recommended Standard (reference [1]) includes a formal definition of the service interface between the Synchronization and Channel Coding Sublayer and the Data Link Protocol Sublayer at the receiving end. The Synchronization and Channel Coding Sublayer uses the service primitive:

ChannelAccess.indication (Frames).

to deliver data to the Data Link Protocol Sublayer. One instance of the ChannelAccess.indication corresponds to the processing of one CLTU by the CLTU reception procedure.

The Synchronization and Channel Coding Sublayer does not need to know the positions of any Frame boundaries within the data it extracts from a CLTU, nor does it use the length fields or other values in the Transfer Frames.

8.6.1.2 Alternative Description

In the formal definition described in Recommended Standard (reference [1]), the interface consists of a single interaction in which the Synchronization and Channel Coding Sublayer delivers all the data extracted from a CLTU. However, this definition does not exclude implementations in which the data from a CLTU is delivered piece-by-piece, provided the Synchronization and Channel Coding Sublayer also delivers indications of the start and end of the CLTU. In this respect, it may be helpful to consider the following alternative description of the interface.

The Synchronization and Channel Coding Sublayer provides a ‘Data Start’ signal to the Data Link Protocol Sublayer, indicating that data are being transferred. The Data Start signal is set to ‘true’ while the Channel Coding Sublayer is in the process of actively transferring data octets. The first octet transferred after Data Start goes ‘true’ corresponds to the first octet of the first Transfer Frame. The Data Start signal is set to ‘false’ (indicating ‘Data Stop’) when the Synchronization and Channel Coding Sublayer stops transferring octets because of a decoder failure or channel deactivation.

8.6.2 DELIVERED FRAMES

8.6.2.1 Overview

The Frames parameter of the ChannelAccess.indication consists of one or more Transfer Frames, but:

- any fill bits added at the sending end to complete the last BCH codeword of the CLTU have not yet been removed;
- the last Frame may be incomplete;
- the Frames have not yet been validated;
- the Spacecraft Identifier fields of the Frames have not yet been checked to verify that the Frames are addressed to the receiving spacecraft.

The Data Link Protocol Sublayer therefore has to perform various procedures on the received data to check that it is handling valid Transfer Frames.

8.6.2.2 Length Field

For each incoming CLTU, the Data Link Protocol Sublayer receives a block of data that is frame-synchronized; that is, it begins on a frame boundary. The Frame Delimiting and Fill Removal Procedure in the Data Link Protocol Sublayer can therefore locate the Frame Length field of the first Frame. The procedure locates the positions of any following Frames in the block, discards truncated Frames, and removes any fill bits. (See references [3] and [4] for definitions of the Frame Delimiting and Fill Removal Procedures.)

8.6.2.3 Frame Error Control Field

References [3] and [4] define an optional Frame Error Control Field in a Transfer Frame. The presence or absence of the field is a managed parameter for a Physical Channel.

The purpose of the field is to detect errors that have not been detected in the Frame by the Synchronization and Channel Coding Sublayer. If the field is present, it is checked by the Frame Validation Check Procedure in the Data Link Protocol Sublayer.

The field uses a Cyclic Redundancy Check (CRC) code. The same CRC code is used in other CCSDS frames, and its performance is discussed in reference [12].

The Frame Error Control Field adds significant extra protection against errors, and its use is particularly advisable when the BCH decoding uses SEC mode. For example, if a received BCH codeword has more than two errors, the BCH decoding in SEC mode may detect a single error and make an erroneous correction. (See the performance discussion in section 11.)

8.6.2.4 Spacecraft Identifier

Spacecraft that have identical RF characteristics may both detect an RF signal carrying TC data and start the decoding and frame reconstruction activities. When the Frame reaches the Data Link Protocol Sublayer, the Frame Validation Check Procedure checks the Spacecraft Identifier field to determine if the Transfer Frame is meant for the receiver. If it is not, the Frame is ignored.

If the Frame contains errors, there is a risk that the Spacecraft Identifier field is corrupt. Therefore the undetected error rate for the Spacecraft Identifier field must be good enough to protect the spacecraft against accepting foreign data.

8.7 RELATIONSHIP TO THE PLOPS

8.7.1 PLOP-1

When the sending end is operating PLOP-1, there is a period of CMM-1 (unmodulated carrier only) between CLTUs. At the receiving end, the CMM-1 causes a loss of bit synchronization in the Physical Layer, which signals CHANNEL DEACTIVATION (Event E2). Therefore the CLTU reception procedure is forced back to the INACTIVE state (S1). So even if the CLTU reception procedure fails to detect the CLTU Tail Sequence, it is forced out of the DECODE state (S3).

During the following Acquisition Sequence (CMM-2), the Physical Layer achieves bit synchronization and signals CHANNEL ACTIVATION (Event E1), sending the CLTU reception procedure to the SEARCH state (S2), where it is ready to detect the Start Sequence of the next CLTU.

8.7.2 PLOP-2

8.7.2.1 Failing to Detect a Tail Sequence

When the sending end is operating PLOP-2, a CLTU is followed by an optional amount of Idle Sequence (CMM-4) and then the next CLTU. The Physical Layer at the receiving end maintains bit synchronization and does not signal an event to the CLTU reception procedure. In this case, the successful operation of the procedure depends on the Tail Sequence causing a CODEWORD REJECTION (Event E4) or a TAIL SEQUENCE FOUND (Event E5), to put the procedure to SEARCH state (S2) ready for the next CLTU.

When relying on decoder failure to detect the tail sequence, there is a small risk that the Tail Sequence will not cause a CODEWORD REJECTION. If this happens, the CLTU reception procedure stays in DECODE state (S3) and tries to decode the data following the Tail Sequence as a BCH CODEWORD. It can therefore miss the beginning of the next CLTU. Section 11 includes performance data on the risk of failing to detect a Tail Sequence.

8.7.2.2 Synchronization Lockout with BCH Coding

The Recommended Standard (reference [1]) also mentions another small risk with operating PLOP-2, which can be avoided by inserting a minimum Idle Sequence of one octet between CLTUs. The risk is synchronization lockout, in which a failure during the reception of one CLTU can cause the loss of the following CLTU.

Synchronization lockout is caused by a combination of two events. The first event is the reception procedure staying in or entering SEARCH state (S2) following an error, such as:

- the Start Sequence of a CLTU is not detected;
- a faulty BCH codeword (not the Tail Sequence) fails to decode.

The second event is the detection of a chance Start Sequence pattern over the last bits of the last BCH codeword of the CLTU and the first bits of its Tail Sequence, causing an erroneous change to DECODE state (S3). The CLTU reception procedure attempts to process the next 64 bits as a BCH codeword. The 64 bits are the remainder of the Tail Sequence and the following bits. If there is no Idle Sequence present, the following bits include the beginning of the Start Sequence of the following CLTU, causing the following CLTU to be lost.

NOTE – The missed Start Sequence on the following CLTU is caused by the reception procedure being in the wrong state, rather than by any properties of the received bits in the Start Sequence.

The risk of synchronization lockout is higher in systems that accept a Start Sequence with a single error. It is also increased in systems in which the CLTU reception procedure is searching for the Start Sequence or its inverse.

The chance Start Sequence that triggers the second event is most likely to include only a few bits of the Tail Sequence. For example, the inverse Start Sequence ends with four ‘1’s, and the Tail Sequence starts with two ‘1’s, so a Start Sequence might occur by chance (without corruption) across the last 14 bits of the last BCH codeword and the first two bits of the Tail Sequence. With some corruption, five or six bits of the Tail Sequence could be in the chance Start Sequence. The 8-bit inserted Idle Sequence should be sufficient to provide protection, unless bit errors are so frequent that CLTU reception is unlikely to be working anyway.

9 PSEUDO-RANDOMIZATION

9.1 OVERVIEW

The purpose of randomization is to enable the receiving end to maintain bit synchronization with the received signal.

NOTE – For brevity, the word ‘random’ is used in place of ‘pseudo-random’ in this report.

To ensure proper receiver operation, the data stream must have a minimum bit transition density. That is, it must be sufficiently random. The randomizer specified in reference [1] is the preferred method to ensure sufficient randomness for all combinations of CCSDS-recommended modulation and coding schemes.

Reference [1] specifies that the randomizer is mandatory with LDPC coding, and should be used with BCH coding unless the system designer verifies that the system will operate properly without it.

With BCH coding, the presence or absence of randomization is fixed for a Physical Channel and is a managed parameter.

NOTE – It may be helpful to have a way to temporarily disable randomization during early pre-launch testing (e.g., board-level checkout, spacecraft integration, environmental testing) so that integration and testing personnel can read the binary data that they are familiar with. This can make data validation and troubleshooting easier.

The randomization procedure consists of exclusive-ORing a random bit pattern with the data. When the procedure is repeated, the result is the original data. Therefore the derandomization procedure at the receiving end is the same as the randomization procedure at the sending end.

9.2 RELATIONSHIP TO OTHER PROCEDURES

9.2.1 AT THE SENDING END, WHEN BCH CODING IS USED

At the sending end, the Transfer Frames received from the Data Link Protocol Sublayer are randomized before the BCH encoding.

As described in 3.4, the Frame data are placed, seven octets (56 bits) at a time, into a set of BCH codewords. If randomization is in use for the Physical Channel, then the data octets from the Frames are randomized before being placed in the BCH codewords.

At the end of the Frames, there may not be exactly seven octets left, so in this case the Information field of the last BCH codeword in a CLTU is filled with a pattern of fill bits. If randomization is in use for the Physical Channel, then the randomization of the fill bits is optional. Whichever option is chosen, the 56 bits in the Information field of a BCH codeword must be the same as the 56 bits used to generate the parity bits for the codeword.

The Error Control Field of a BCH codeword is not randomized.

Randomization is applied during the generation of BCH codewords and does not affect the generation of CLTUs or the PLOPs, so the following fields are not randomized:

- the Start Sequence of a CLTU;
- the Tail Sequence of a CLTU;
- the Acquisition Sequence transmitted by a PLOP;
- the Idle Sequence transmitted by a PLOP.

9.2.2 AT THE SENDING END, WHEN LDPC CODING IS USED

At the sending end, the codewords are randomized after LDPC encoding.

As described in section 4, the Transfer Frame(s) to be transmitted in a CLTU are partitioned into segments of 64 bits or 256 bits, depending on the LDPC code selected, and each is encoded. If the last segment is not complete, fill bits are added as needed before LDPC encoding. Each LDPC codeword is then randomized.

The randomizer is reset to the all-ones pattern at the beginning of each codeword.

Randomization is applied to each LDPC codeword, and does not affect the generation of CLTUs or the PLOPs, so the following fields are not randomized:

- the Start Sequence of a CLTU;
- the Tail Sequence of a CLTU, if used with the (n=128, k=64) LDPC code;
- the Acquisition Sequence transmitted by a PLOP;
- the Idle Sequence transmitted by a PLOP.

9.2.3 AT THE RECEIVING END, WHEN BCH CODING IS USED

At the receiving end, derandomization takes place after the BCH decoding. The CLTU reception procedure derandomizes the data extracted from successfully decoded BCH codewords, before passing the data to the Data Link Protocol Sublayer.

The derandomization is applied to the 56-bit Information field of a BCH codeword. Therefore any fill bits in the Information field of the last BCH codeword of a CLTU are derandomized, whether they were randomized or not. One of the tasks of the Data Link Protocol Sublayer is to discard any fill bits following the end of the last Frame, as described in 8.6.2.

NOTE – An octet of fill bits has the hexadecimal value 55, as described in 3.4. In a system that uses randomization but does not randomize the fill bits, the values of the fill bits are changed by the derandomization process. In this case, the octets of fill bits that are delivered to the Data Link Protocol Sublayer do not have the hexadecimal value 55. As a CLTU can have up to six octets of fill bits, and the minimum length of a Frame is six octets, there is a small risk that the Data Link Protocol Sublayer could mistake the fill bits for a Frame. The risk only applies in systems that use multiple Frames per CLTU. The validation by the Data Link Protocol Sublayer of the Frame Length field and other fields is likely to detect that the fill bits are not a valid Frame.

9.2.4 AT THE RECEIVING END, WHEN LDPC CODING IS USED

At the receiving end, derandomization takes place immediately prior to LDPC decoding. The derandomization is applied to each 128-symbol or 512-symbol codeword, depending on which LDPC code is selected. Because derandomization is always applied to entire codewords, any fill data is always derandomized to its original alternating zero-one pattern.

NOTE – The Synchronization and Channel Coding Sublayer does not remove any fill data that may have been added, and it is the obligation of the Data Link Protocol Sublayer to do so. There may be up to 7 octets or 15 octets of fill data, depending on which LDPC code is used. Every octet of fill data has the hexadecimal value 55, and so should be readily identifiable.

9.3 RANDOMIZATION PROCEDURE

9.3.1 THE RANDOM SEQUENCE

The random sequence is generated by the Bit Transition Generator (BTG), which uses a linear feedback shift register with eight bits. The bits of the shift register are all initialized to '1'.

Figure 9-1, which is reproduced from reference [1], shows the logic diagram for the BTG. The logic can be represented by the polynomial:

$$h(x) = x^8 + x^6 + x^4 + x^3 + x^2 + x + 1.$$

The first 40 bits of the generated sequence are:

1111 1111 0011 1001 1001 1110 0101 1010 0110 1000.

The generated sequence repeats after 255 bits.

NOTE – The maximum repetition period of a sequence generated by an m-bit linear feedback shift register is $2^m - 1$. In this case $m = 8$.

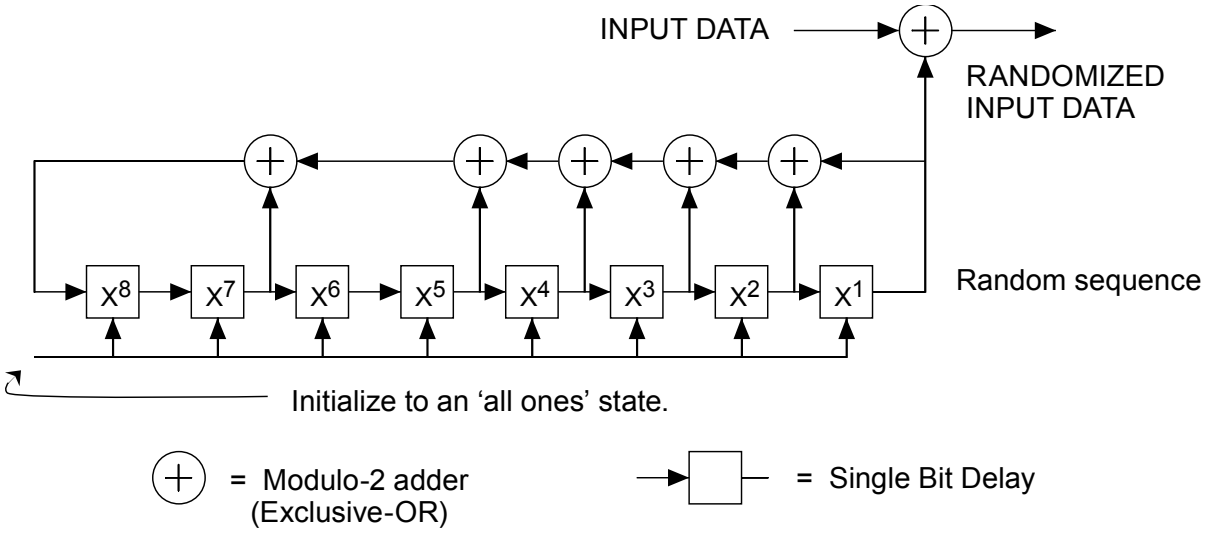

Figure 9-1: Logic Diagram for the Bit Transition Generator

9.3.2 APPLYING THE RANDOM SEQUENCE AT THE SENDING END, WHEN BCH CODING IS USED

The random sequence is restarted for each CLTU. Therefore, at the sending end, the shift registers of the BTG are all reinitialized to '1' at the start of processing for each new ChannelAccess.request from the Data Link Protocol Sublayer.

The first bit of the first Transfer Frame in the Frames parameter of the request is exclusive-ORed with the first bit of the random sequence. Then the second bit of the first Transfer Frame is exclusive-ORed with the second bit of the random sequence. And so on until the end of the Frames and (optionally) the fill bits are reached.

The boundaries between Frames are ignored. The BTG is not reinitialized until the start of processing for the next ChannelAccess.request.

9.3.3 APPLYING THE RANDOM SEQUENCE AT THE SENDING END, WHEN LDPC CODING IS USED

The random sequence is restarted for each LDPC codeword. The first symbol of the first codeword is exclusive-ORed with the first bit of the random sequence. Then the second symbol of the codeword is exclusive-ORed with the second bit of the random sequence, and so on, until the end of the codeword is reached. Then the shift registers of the BTG are all reinitialized to '1', and the next codeword is randomized similarly.

9.3.4 APPLYING THE RANDOM SEQUENCE AT THE RECEIVING END, WHEN BCH CODING IS USED

The random sequence is restarted for each CLTU. Therefore, at the receiving end, the shift registers of the BTG are all reinitialized to '1' whenever the CLTU reception procedure (see section 8) finds a CLTU Start Sequence (Event E3, START SEQUENCE FOUND).

The 56 Information bits from the first decoded BCH codeword are exclusive-ORed with the first 56 bits of the random sequence. Then the 56 Information bits from the next decoded BCH codeword are exclusive-ORed with the next 56 bits of the random sequence. And so on until the end of the last successfully decoded BCH codeword in the CLTU. The processing for the CLTU ends with a CODEWORD REJECTION (Event E4) or a CHANNEL DEACTIVATION (Event E2).

9.3.5 APPLYING THE RANDOM SEQUENCE AT THE RECEIVING END, WHEN LDPC CODING IS USED

The random sequence is restarted for each LDPC codeword. Therefore, at the receiving end, the shift registers of the BTG are all initialized to '1' whenever the CLTU reception procedure (see section 8) finds a CLTU Start Sequence (Event E3, START SEQUENCE FOUND). The received noisy codeword is derandomized using the first 128 bits or 512 bits of the random sequence, depending on the LDPC code selected. Then the BTG is reinitialized to the all-ones pattern, and the next LDPC codeword is derandomized similarly. This is repeated until an LDPC decoder failure occurs, or a CLTU tail sequence is detected by pattern matching.

The method for derandomizing a codeword depends on the implementation of the decoder. For a soft-symbol decoder that associates positive numbers with a '0' code symbol and negative numbers with a '1' code symbol, derandomization is achieved by negating each code symbol that is paired with a '1' from the BTG. Soft-symbol decoders can be built with the opposite sign convention, in which case derandomization is achieved by negating each code symbol that is paired with a '0' from the BTG. Hard-symbol decoders may also be used; in this case, each binary code symbol is derandomized by exclusive-ORing it with the corresponding bit from the BTG.

9.4 BACKGROUND

9.4.1 ADDING RANDOMIZATION TO THE RECOMMENDATIONS

Early CCSDS recommendations for Telecommand Channel Coding did not include randomization. Telecommand data tended to consist of short, discrete commands with frequent transitions. Then there was a shift towards larger volumes of telecommand data, including lengthy transfers of large areas of computer memory, parts of which have few transitions. This led to problems in systems that rely on frequent bit transitions in the data to maintain bit synchronization.

Randomization was added to the CCSDS recommendations for Telecommand Channel Coding published in November 1995.

9.4.2 RELATIONSHIP TO TM RANDOMIZATION

The randomization in reference [1] is not the same as the randomization for TM Synchronization and Channel Coding in reference [5]. The basic principle is the same, but the pseudo-random bit pattern and its underlying generator polynomial are different.

The TM randomizer was found to have some minor interactive properties with the Reed-Solomon code used in TM for forward error correction. The randomizer chosen for telecommand avoids this interaction. Annex C includes the theoretical background to the choice of the generator polynomial for the randomizer specified in reference [1].

There are also differences in the position of the randomization relative to the other procedures. In TC (reference [1]), the randomization is before the BCH encoding, or after the LDPC encoding, because of the quasi-cyclic structure of the LDPC codes as described in section 4. In TM Synchronization and Channel Coding (reference [5]), the randomization is after the Reed-Solomon, Turbo, or LDPC encoding and before any Convolutional encoding.

10 OPTIONS FOR REPEATED TRANSMISSIONS

10.1 INTRODUCTION

The TC Synchronization and Channel Coding Sublayer has an optional systematic retransmission. It is an optional feature of the sending end of the sublayer: the receiving end of the sublayer is not affected. The systematic retransmission was added to the CCSDS recommended standard (reference [1]) in 2010.

The systematic retransmission can improve performance for missions with a long light time delay (deep space missions).

10.2 PARAMETERS FOR SYSTEMATIC RETRANSMISSION

10.2.1 SERVICE INTERFACE

The recommended standard (reference [1]) includes a formal definition of the service interface provided by the TC Synchronization and Channel Coding Sublayer. It defines the primitive that is used to request the transfer of one or more frames:

ChannelAccess.request (Frames, Repetitions).

The Repetitions parameter controls the systematic retransmission.

10.2.2 REPETITIONS PARAMETER

The ChannelAccess.request causes the sublayer to process the data indicated by the Frames parameter and to transfer the resulting CLTU. The CLTU is transferred the number of times specified by the Repetitions parameter.

The value of Repetitions specifies the number of times the CLTU is transferred, not the additional transfers after the first one. For example, if the value of Repetitions is 3, then the sublayer transfers the CLTU three times. The name ‘Repetitions’ was chosen to avoid confusion with terms used in the specification of the Communication Operations Procedure 1 (COP-1) of the TC Space Data Link Protocol (reference [3]).

The recommended standard (reference [1]) does not specify how the systematic retransmission is implemented. In particular, it does not specify how the retransmission is positioned relative to the other procedures of the sublayer. So, for example, an implementation could:

- generate the CLTU once and then transfer it as many times as indicated by the Repetitions parameter; or
- generate the CLTU each time it is transferred.

Support for the Repetitions parameter is optional. In a system that does not support systematic retransmission, the parameter is absent or ignored. In this case, the system behaves as if the value of Repetitions is 1.

10.2.3 FRAMES PARAMETER

The data indicated by the Frames parameter consist of one or more Transfer Frames. However, the use of multiple frames increases the risk of loss:

– in a CLTU with multiple frames, the loss of a frame causes the loss of all subsequent frames in the same CLTU; and

– the probability of loss increases with the length of the CLTU.

Therefore, when using the systematic retransmission, further performance improvement can be obtained by restricting the Frames parameter to consist of a single Transfer Frame.

10.3 USING THE SYSTEMATIC RETRANSMISSION

10.3.1 OVERVIEW

Transfer Frames are generated by the Space Data Link Protocol and are handled by one of the COP-1 services specified in reference [11]. COP-1 has two services: the Sequence-Controlled Service and the Expedited Service.

The systematic retransmission in the TC Synchronization and Channel Coding Sublayer is intended for use with the Transfer Frames of the Sequence-Controlled Service. The discussion here applies to these frames.

10.3.2 COP-1 AUTOMATIC RETRANSMISSION

The COP-1 Sequence-Controlled Service includes an automatic retransmission, which is a classic Go-Back-*N* (GBN) scheme. The retransmission is part of a closed loop: the sending end sends Transfer Frames, and the receiving end returns protocol data via telemetry, to acknowledge the receipt of frames and to report missing frames. The retransmission can result in multiple copies of the same frame arriving at the receiving end, but COP-1 automatically discards any duplicate frames.

NOTE – ‘Automatic retransmission’ refers to the COP-1 retransmission. ‘Systematic retransmission’ refers to the retransmission in the TC Synchronization and Channel Coding Sublayer.

COP-1 retransmission occurs when:

- the sending end receives a report of one or more missing frames; or
- the sending end timer expires and one or more transmitted frames have not been acknowledged.

The missing frame and all the frames that follow it are retransmitted. This is the GBN retransmission, illustrated at the beginning of annex G.

The automatic retransmission works well for near-Earth missions: the round-trip time delay is short, so COP-1 can respond efficiently to the need for retransmission of a lost frame. Typically, the timer value is set to the total length of the COP-1 closed-loop path, including processing and serialization times at both ends, and the light-time delays on the uplink and downlink.

10.3.3 RETRANSMISSION FOR DEEP SPACE MISSIONS

10.3.3.1 Long Wait

For deep space missions, the COP-1 closed-loop path is long, and the response is less efficient than it is for the near-Earth case. The sending end transmits a sequence of frames and then waits. Any COP-1 automatic retransmission happens only after a long waiting time.

During the long waiting time, the uplink is unused. Performance and efficiency can be improved by using retransmissions, without waiting for the response from the receiving end.

10.3.3.2 Option for Systematic Retransmission

The systematic retransmission is one option for the retransmissions. For example, for a sequence of frames ABCDEF, and a repetition factor of 3, the systematic retransmission gives a transmission pattern of the form:

AAABBBCCCDDDEEEFFF.

When frame loss errors are sporadic, this pattern is an advantage. Because the errors are sporadic, they are less likely to hit all the repeats of a frame. The advantage increases as the number of frames in the sequence increases.

When frame loss errors come in bursts, the systematic retransmission is vulnerable to an error burst that hits all the repeats of a frame.

(See annex G for a numerical evaluation of this GBN scheme with multiple copies.)

10.3.3.3 Option for Early Automatic Retransmission

Early automatic retransmission is another option for the retransmissions. A deep space mission can set a short timer value for COP-1, in order to force the early automatic retransmission of a sequence of frames. For a sequence of frames ABCDEF, this gives a cyclic transmission pattern of the form:

ABCDEF ABCDEF ABCDEF...

When frame loss errors come in bursts, the cyclic transmission pattern is less vulnerable than the systematic retransmission. When frame loss errors are sporadic, successful reception of a sequence of frames depends on where the errors fall within a cycle. The frame loss of an early frame in a cycle causes the remaining frames in that cycle to be rejected.

10.3.3.4 Choosing an Option

Neither of these retransmission options is a universal solution. They provide more tools for solving the problems of delivering a complete sequence of frames.

In practice, the best solution for a mission can consist of a combination of both options. Mission designers can use numerical methods and simulations to optimize the transmission patterns and repetition factors to suit the link time delays and other link conditions.

10.3.4 SCENARIOS USING SYSTEMATIC RETRANSMISSION

10.3.4.1 Scenario with Sporadic Frame Loss

The typical scenario for the application of systematic retransmission is a deep space mission with long light-time delay and with link conditions for which errors are sporadic, so that sometimes one frame, or perhaps two frames, are lost from a sequence of frames.

When the systematic transmission option is used in this scenario, it can increase the probability that the full sequence is received during a limited transmission session.

10.3.4.2 Scenarios with High Probability of Frame Loss

There are deep space scenarios in which the probability of frame loss is exceptionally high:

- **Spacecraft on the Far Side of Earth's orbit**. In this scenario, the RF link to the spacecraft can suffer severe solar interference, leading to extreme frame loss rates. The telemetry link is also likely to be affected by the solar interference.
- **Spacecraft Unable to Maintain Correct Attitude**. In this scenario, the low-gain antenna at the receiving end of the telecommand link is intermittently pointing in the wrong direction. If the spacecraft is spinning, there is a periodic element to the high frame loss rate. When the spacecraft enters safe mode, the telecommand data rate is very low and telemetry transmissions have very low power and a low data rate.

For these deep-space scenarios, the systematic retransmission provides another means to attempt to deliver a sequence of telecommand frames, despite the very difficult conditions.

10.4 INDEPENDENCE OF THE SYSTEMATIC RETRANSMISSION

The systematic retransmission of the TC Synchronization and Channel Coding Sublayer is independent of the automatic retransmission of COP-1. The two retransmission mechanisms are in different sublayers of the protocol stack and are controlled by different parameters.

If systematic retransmission is in use, then its behavior does not affect the behavior of COP-1. However, users can choose to set different parameter values (such as the COP-1 timer value) to take account of the combined effect of the retransmissions on the output to the uplink.

Automatic retransmission operates within a TC virtual channel and systematic retransmission operates within a Physical Channel.

11 PERFORMANCE DATA

11.1 INTRODUCTION

The objective of the TC Synchronization and Channel Coding Sublayer is to provide a reliable service for the delivery of Transfer Frames.

This section defines the performance criteria for a reliable service. It provides performance data on the coding options in the Recommended Standard (reference [1]) to enable the system engineer to select the coding parameters and other features so that the performance criteria are satisfied. Because the criteria are measured in frames, the features considered here extend into the Data Link Protocol Sublayer.

Subsections 11.2 and 11.3 specify performance criteria appropriate for a telecommand link using either BCH or LDPC codes. Subsections 11.4 and 11.5 consider the BCH code in particular, and assume a binary symmetric channel with additive white Gaussian noise. Performance statistics are shown for three different channel bit error rates: 10^{-4}, 10^{-5}, and 10^{-6}. In equations, the channel bit error rate is represented by the letter p. Subsections 11.6 through 11.9 address performance with the LDPC codes, and assume an Additive White Gaussian Noise (AWGN) channel. An LDPC code is most commonly decoded using an iterative Belief Propagation algorithm that passes messages between computational nodes. Performance results for the basic 'flooding' message passing schedule are given in 11.6, and 11.7 considers a modified message-passing schedule known as layered decoding. As an alternative to the BP algorithm, 11.8 presents results for a Most Reliable Basis (MRB) decoder.

Annex F provides performance data for obsolete synchronization and coding features not included in the current issue of the Recommended Standard (reference [1]).

11.2 PERFORMANCE CRITERIA

11.2.1 GENERAL

The unit for measuring the performance is the Transfer Frame. A reliable service for the delivery of Transfer Frames is considered to be achieved when the following two performance criteria are simultaneously met.

11.2.2 TRANSFER FRAME REJECTION RATE

A maximum of one Transfer Frame is deleted (rejected) for every 10^3 frames transmitted. This operating point is defined to be the Command Threshold.

11.2.3 TRANSFER FRAME UNDETECTED ERROR RATE

A maximum of one Transfer Frame for every 10^9 frames transmitted is erroneously accepted (that is, contains one or more undetected bit errors).

11.3 PERFORMANCE COMPONENTS

To support the performance criteria, the Recommended Standards (references [1] and [3]) specify procedures for handling the Transfer Frame and related structures:

- PLOP-1 or PLOP-2 and the Acquisition Sequence to enable the receiving end to acquire bit synchronization;
- randomization (optional for BCH, mandatory for LDPC) to ensure that the receiving end can maintain bit synchronization and codeword synchronization;
- placing a Start Sequence at the start of each CLTU so that the CLTU reception procedure can delimit the beginning of the first codeword (and therefore the beginning of the first frame) in the CLTU;
- encoding/decoding of the codewords to provide error detection and/or correction;
- placing a Tail Sequence at the end of each CLTU (when applicable) so that the CLTU reception procedure can detect the end of the CLTU and so start searching for the Start Sequence of the next CLTU;
- Frame Delimiting and Fill Removal Procedure (in the Data Link Protocol Sublayer at the receiving end) to delimit the end of the first Transfer Frame in the data extracted from the CLTU and delimit the beginning and end of each subsequent frame in the data;
- Frame Validation Check Procedure (in the Data Link Protocol Sublayer at the receiving end) to check the mandatory fields in the header of each Transfer Frame, and to check the optional Frame Error Control Field, which provides additional protection against undetected errors.

11.4 FACTORS AFFECTING FRAME REJECTION RATE

11.4.1 GENERAL

The following factors affect the frame rejection rate, in which a frame is lost or deleted because of errors on the channel.

11.4.2 BIT SYNCHRONIZATION FACTOR

As described in section 7, the PLOP in the Physical Layer at the sending end transmits CMM-1 (unmodulated carrier mode) at the start of a communications session. The transponder in the Physical Layer at the receiving end locks onto the signal.

Then the PLOP transmits the Acquisition Sequence (CMM-2) with the length selected for the mission. The Acquisition Sequence, consisting of alternating ‘1’ and ‘0’ bits, has maximum transition density, to provide the fastest lock-up time for the onboard bit synchronizer.

The preferred minimum length of 128 bits was chosen to provide 0.9999 probability of acquisition of bit synchronization, based on experience with a number of hardware implementations operating at the command threshold level. This length may be modified as needed to suit different hardware characteristics or channel bit error rates.

Because the probability of achieving bit synchronization is primarily hardware dependent, it is not considered further in the performance analysis in this section. The remainder of this section assumes that bit synchronization is achieved whenever the Acquisition Sequence is transmitted. As described in section 8, whenever bit synchronization is achieved, the Physical Layer signals CHANNEL ACTIVATION (Event E1), and the CLTU reception procedure goes to SEARCH state (S2).

11.4.3 CLTU START SEQUENCE FACTORS

11.4.3.1 General

There are two CLTU Start Sequence factors:

- the probability that the CLTU reception procedure misses a Start Sequence because it is not in SEARCH state (S2) when the Start Sequence is received;
- the probability that the Start Sequence is not recognized by the CLTU reception procedure in SEARCH state (S2).

11.4.3.2 CLTU Reception Procedure Not in SEARCH State

If the Start Sequence follows an Acquisition Sequence (plus an optional length of Idle Sequence), then the SEARCH state of the CLTU reception procedure depends on successfully achieving bit synchronization. As described in 11.4.2, this performance analysis assumes that bit synchronization is achieved.

If the sending end is operating the PLOP-1 procedure, then the Start Sequence always follows an Acquisition Sequence, so this performance analysis assumes that the CLTU reception procedure is in SEARCH state when receiving the Start Sequence.

If the sending end is operating the PLOP-2 procedure, then the Start Sequence follows the Tail Sequence of the previous CLTU (with an optional length of Idle Sequence between the Tail Sequence and the Start Sequence). The synchronization lockout described in 8.7.2.2 can cause a Start Sequence to be missed, but the probability is not considered here. The probability that the CLTU reception procedure is not in SEARCH state when receiving the Start Sequence therefore depends on the probability of failing to recognize the Tail Sequence (see 11.4.5).

NOTE – For PLOP-1 and PLOP-2, there is the remote possibility that the proper Start Sequence is missed because of a false (premature) start when the Start Sequence is erroneously recognized in the Acquisition Sequence/Idle Sequence pattern. The Start Sequence has a distance of six bits from the Acquisition Sequence/Idle Sequence, so it would require six changed bits to convert a section of the Acquisition Sequence/Idle Sequence into a Start Sequence. If the option to accept the Start Sequence with one error is in use, then a change of five bits could be enough to cause a premature start.

11.4.3.3 Start Sequence Not Recognized

The Start Sequence is a 16-bit synchronization pattern. The CLTU reception procedure in SEARCH state searches the received stream of bits, looking for the Start Sequence pattern.

As described in 8.4.3, there are two options in the CLTU reception procedure for the handling of errors in the Start Sequence. Either:

a) no errors are accepted, or

b) a single error is accepted.

The options are indicated by subscripts X and Y in the following discussion.

If no errors are accepted, then the received bit pattern is accepted as a valid Start Sequence only if it exactly matches the expected bit pattern. In this case, the probability P_{SX} of failing to recognize the Start Sequence is therefore the probability that one or more bit errors fall on any of the 16 bits of the Start Sequence:

$$P_{SX} = 1 - (1-p)^{16}. \quad (1)$$

If a single error is accepted, then the received bit pattern is accepted as a valid Start Sequence either if it exactly matches the expected bit pattern or if it differs by one bit from the expected bit pattern. In this case, the probability P_{SY} of failing to recognize the Start Sequence is therefore the probability that two or more bit errors appear in the 16 bits of the Start Sequence:

$$P_{SY} = 1 - [(1-p)^{16}+16\mathrm{p}(1-p)^{15}]. \quad (2)$$

Table 11-1 shows the values of the probabilities P_{SX} and P_{SY} for the three channel bit error rates.

Table 11-1: Probability of Not Recognizing the Start Sequence

	Channel Bit Error Rate		
Case	**10^{-4}**	**10^{-5}**	**10^{-6}**
P_{SX} no errors accepted	1.60×10^{-3}	1.60×10^{-4}	1.60×10^{-5}
P_{SY} single error accepted	1.20×10^{-6}	1.20×10^{-8}	1.20×10^{-10}

11.4.4 BCH CODEWORD FACTOR

11.4.4.1 Decoding the BCH Codewords

The BCH codewords of a CLTU are decoded one by one, as long as each codeword is successfully decoded. If a BCH codeword causes a codeword rejection, the remaining codewords of the CLTU are discarded.

The probability of a codeword rejection in a CLTU increases with the number of BCH codewords in the CLTU.

As described in 8.5, there are two options in the CLTU reception procedure for the decoding of BCH codewords. Either

a) TED mode is used, or

b) SEC mode is used.

When TED mode is in use, then no error is allowed in the Start Sequence, as in option X in 11.4.3.3 above. Similarly, when SEC mode is used, a single error is accepted in the Start Sequence. The options are indicated by subscripts X and Y in the following discussion.

11.4.4.2 Codeword Rejection in TED Mode

In TED mode, the probability P_{CX} that one of the BCH codewords contains errors causing rejection of the complete CLTU is:

$$P_{CX} \leq 1 - [(1 - p)^{63}]^N, \tag{3}$$

where N is the number of codewords in the CLTU. The inequality sign indicates that not all possible error patterns are indeed detected by the BCH code in TED mode. However, the effect of undetected errors can be considered sufficiently small to justify the use of (3) as a reliable approximation. Details are reported in annex H.

Table 11-2 shows the values of the probability P_{CX} for the three channel bit error rates.

Table 11-2: Probability P_{CX} of Codeword Rejection in TED Mode

Number of Codewords *N*	Channel Bit Error Rate		
	10^{-4}	10^{-5}	10^{-6}
1	6.28×10^{-3}	6.30×10^{-4}	6.30×10^{-5}
2	1.25×10^{-2}	1.26×10^{-3}	1.26×10^{-4}
4	2.49×10^{-2}	2.52×10^{-3}	2.52×10^{-4}
6	3.71×10^{-2}	3.77×10^{-3}	3.78×10^{-4}
9	5.51×10^{-2}	5.65×10^{-3}	5.67×10^{-4}
12	7.28×10^{-2}	7.53×10^{-3}	7.56×10^{-4}
16	9.59×10^{-2}	1.00×10^{-2}	1.01×10^{-3}
20	1.18×10^{-1}	1.25×10^{-2}	1.26×10^{-3}
24	1.40×10^{-1}	1.50×10^{-2}	1.51×10^{-3}
28	1.62×10^{-1}	1.75×10^{-2}	1.76×10^{-3}
32	1.83×10^{-1}	2.00×10^{-2}	2.01×10^{-3}
36	2.03×10^{-1}	2.24×10^{-2}	2.27×10^{-3}
37	2.08×10^{-1}	2.30×10^{-2}	2.33×10^{-3}
74	3.73×10^{-1}	4.56×10^{-2}	4.65×10^{-3}
110	5.00×10^{-1}	6.70×10^{-2}	6.91×10^{-3}
147	6.04×10^{-1}	8.85×10^{-2}	9.22×10^{-3}
293	8.42×10^{-1}	1.69×10^{-1}	1.83×10^{-2}
439	9.37×10^{-1}	2.42×10^{-1}	2.73×10^{-2}
586	9.75×10^{-1}	3.09×10^{-1}	3.62×10^{-2}

11.4.4.3 Codeword Rejection in SEC Mode

Two of the values used in the decoding process are the PARity indicator (PAR) and the SYND, shown in table 11-3. In SEC mode, the values are used to determine the actions of the decoder, and they lead to six possible cases, as shown in table 11-4.

Table 11-3: Meaning of Decoding Values

Error parity check	PAR='0'	The number of errors is even, 0, 2, 4, 6, …
	PAR='1'	The number of errors is odd, 1, 3, 5, 7, …
Syndrome	SYND=0	The received codeword is a valid codeword, though not necessarily the one that was transmitted.
	SYND>0	The received codeword is not valid, so errors are present.

Table 11-4: Decoding Cases in SEC Mode

PAR	SYND	Action	Case	
'0'	0	accept codeword	A1	The received codeword has no errors.
			A2	The received codeword has an even number of errors (at least 4).
'0'	>0	codeword rejection	B	The received codeword has an even number of errors (at least 2).
'1'	0	codeword rejection	C	The received codeword has an odd number of errors (at least 3).
'1'	>0	correct single error and accept codeword	D1	The corrected codeword has no errors.
			D2	The corrected codeword has an even number of errors.

In cases A1 and D1, the SEC decoding is successful and delivers an error-free codeword. In cases A2 and D2, the SEC decoding delivers a codeword containing undetected errors; these cases are considered further in 11.5 below. Cases B and C lead to codeword rejection and are considered here.

The probabilities P_B and P_C for cases B and C are given by:

$$P_B = P2 + P4\,(1-R4) + P6\,(1-R6) + \ldots$$

$$P_C = P3\,(1-R3) + P5\,(1-R5) + \ldots,$$

where Pi is the probability of i errors and Ri is the portion of these errors that are not detected. Adding the two equations gives:

$$P_B + P_C = P2 + P3\ (1-R3) + P4\ (1-R4) + P5\ (1-R5) + P6\ (1-R6) + \ldots$$

To three significant figures, this gives similar results to the general expression for the probability of two or more errors in a codeword. So the probability P_{CY} that one of the BCH codewords contains errors causing rejection of the complete CLTU can be written as:

$$P_{CY} \leq 1 - [(1-p)^{63} + 63p(1-p)^{62}]^N, \tag{4}$$

where N is the number of codewords in the CLTU. The inequality sign indicates that not all possible error patterns of two or more errors are indeed corrected or detected by the BCH code in SEC mode. However, the effect of undetected errors can be considered sufficiently small to justify the use of (4) as a reliable approximation. Details are reported in annex H.

Table 11-5 shows the values of the probability P_{CY} for the three channel bit error rates.

Table 11-5: Probability P_{CY} of Codeword Rejection in SEC Mode

Number of Codewords N	Channel Bit Error Rate		
	10^{-4}	10^{-5}	10^{-6}
1	1.95×10^{-5}	1.95×10^{-7}	1.95×10^{-9}
2	3.89×10^{-5}	3.90×10^{-7}	3.91×10^{-9}
4	7.78×10^{-5}	7.81×10^{-7}	7.81×10^{-9}
6	1.17×10^{-4}	1.17×10^{-6}	1.17×10^{-8}
9	1.75×10^{-4}	1.76×10^{-6}	1.76×10^{-8}
12	2.33×10^{-4}	2.34×10^{-6}	2.34×10^{-8}
16	3.11×10^{-4}	3.12×10^{-6}	3.12×10^{-8}
20	3.89×10^{-4}	3.90×10^{-6}	3.91×10^{-8}
24	4.67×10^{-4}	4.69×10^{-6}	4.69×10^{-8}
28	5.44×10^{-4}	5.47×10^{-6}	5.47×10^{-8}
32	6.22×10^{-4}	6.25×10^{-6}	6.25×10^{-8}
36	7.00×10^{-4}	7.03×10^{-6}	7.03×10^{-8}
37	7.19×10^{-4}	7.22×10^{-6}	7.23×10^{-8}
74	1.44×10^{-3}	1.44×10^{-5}	1.45×10^{-7}
110	2.14×10^{-3}	2.15×10^{-5}	2.15×10^{-7}
147	2.86×10^{-3}	2.87×10^{-5}	2.87×10^{-7}
293	5.68×10^{-3}	5.72×10^{-5}	5.72×10^{-7}
439	8.50×10^{-3}	8.57×10^{-5}	8.57×10^{-7}
586	1.13×10^{-2}	1.14×10^{-4}	1.14×10^{-6}

11.4.5 TAIL SEQUENCE FACTOR

There is a risk of missing (failing to recognize) the Tail Sequence. If errors in the channel alter the Tail Sequence, the received sequence may be mistakenly accepted as a valid or correctable codeword. Because the expected CODEWORD REJECTION does not occur, the Tail Sequence is missed. In this case, it is possible that the CLTU reception procedure will not be in SEARCH state for the next CLTU, and therefore the subsequent CLTU may be missed.

NOTE – The description here of the probability of missing a Tail Sequence applies to the Tail Sequence pattern defined in the current issue of the Recommended Standard (reference [1]). (See annex E for the probability for the earlier Tail Sequence pattern.)

The two options in the CLTU reception procedure for the decoding of BCH codewords have different probabilities of missing a Tail Sequence. Either

a) TED mode is used; or

b) SEC mode is used.

Two of the values in decoding the BCH code are PAR and SYND, shown in table 11-3. When decoding in TED mode, the codeword is accepted only if PAR='0' and SYND=0. When decoding in SEC mode:

– the codeword is accepted if PAR='0' and SYND=0; or

– one error is corrected and the codeword is accepted if PAR='1' and SYND>0.

The PAR and SYND values for the Tail Sequence pattern with different numbers of errors, e = 0, 1, 2 or 3, are shown in table 11-6. If a value, t, in the table can lead to the mistaken acceptance of the Tail Sequence as a codeword, then its contribution to the probability of a missed Tail Sequence is given by the expression:

$$t\ \mathrm{p}^{e}\ (1-\mathrm{p})^{63-e}\ . \tag{5}$$

Numbers of errors greater than three are not shown, because they do not affect the three significant figures for the probabilities in table 11-7.

Table 11-6: Parity and Syndrome When Tail Sequence Has Errors

	Number of errors, *e*			
	***e* = 0**	***e* = 1**	***e* = 2**	***e* = 3**
PAR='0', SYND=0	-	-	-	651
PAR='0', SYND>0	-	63	-	39060
PAR='1', SYND=0	1	-	-	-
PAR='1', SYND>0	-	-	1953	-
Total combinations	1	63	1953	39711

When there are no errors, the Tail Sequence gives PAR='1' and SYND=0, so it is rejected in TED and SEC modes. With all 63 possible cases of a single error, the Tail Sequence gives PAR='0' and SYND>0 and is rejected in TED and SEC modes.

There are 1953 combinations of two errors, all of which give PAR='1' and SYND>0. These are rejected in TED mode but are erroneously corrected and accepted in SEC mode, causing a missed Tail Sequence.

Most of the 39711 combinations of three errors cause the Tail Sequence to be rejected in TED and SEC modes, but there are 651 combinations that give PAR='0' and SYND=0 and are accepted in both modes, causing a missed Tail Sequence.

Using expression 5 above, the probability P_{TX} of missing the Tail Sequence in TED mode is given by:

$$P_{TX} = 651\ \mathrm{p}^3\ (1 - \mathrm{p})^{60}. \tag{6}$$

The probability P_{TY} of missing the Tail Sequence in SEC mode is given by:

$$P_{TY} = 1953\ \mathrm{p}^2\ (1 - \mathrm{p})^{61} + 651\ \mathrm{p}^3\ (1 - \mathrm{p})^{60}. \tag{7}$$

Table 11-7 shows the values of the probabilities P_{TX} and P_{TY} for the three channel bit error rates.

Table 11-7: Probability of Missing the Tail Sequence

	Channel Bit Error Rate		
Case	**10^{-4}**	**10^{-5}**	**10^{-6}**
P_{TX} error detection mode	6.47×10^{-10}	6.51×10^{-13}	6.51×10^{-16}
P_{TY} error correction mode	1.94×10^{-5}	1.95×10^{-7}	1.95×10^{-9}

11.4.6 FRAMES AND CLTUs

11.4.6.1 General

The probabilities discussed in 11.4.2 to 11.4.5 affect CLTU rejection. However, in the performance criteria in 11.2, the Transfer Frame is the accounting entity.

If there is only one Transfer Frame in a CLTU, the probability of frame rejection is the same as the probability of rejection of the CLTU.

If there are multiple Transfer Frames in a CLTU, the acceptance of a frame depends on the successful acceptance of the codewords that precede it in the CLTU. If the frame ends in codeword *M*, then the probability of frame rejection is the same as the probability of rejection of a CLTU with *M* codewords. For the last frame in a CLTU, the probability of frame rejection is the same as the probability of rejection of the CLTU. As this is the worst case for the frames in the CLTU, it appears prudent to base performance comparisons on this value.

CLTUs may be sent either as independent entities (PLOP-1, in which modulation is dropped between CLTUs) or as a sequence (PLOP-2, in which bit synchronization is maintained on the channel). The choice of PLOP-1 or PLOP-2 contributes to the CLTU rejection probability.

11.4.6.2 Independent CLTUs (PLOP-1)

In calculating the rejection probability with independent CLTUs, two events are considered:

– the CLTU Start Sequence is not recognized;

– the CLTU Start Sequence is recognized, but one of the codewords is rejected.

The probability of CLTU rejection depends on which options are in use in the CLTU reception procedure:

a) no errors accepted in the Start Sequence and codewords decoded in TED mode; or

b) a single error accepted in the Start Sequence and codewords decoded in SEC mode.

For a frame that is the last or only frame in a CLTU, the corresponding frame rejection probabilities, P_{FX} and P_{FY}, are given by:

$$P_{FX} = P_{SX} + (1 - P_{SX})\, P_{CX}$$

$$P_{FY} = P_{SY} + (1 - P_{SY})\, P_{CY}.$$

Table 11-8 shows the values of the probabilities P_{FX} and P_{FY} for the three channel bit error rates. The table includes values for long CLTUs carrying multiple maximum-length Transfer Frames. For example, a CLTU with 293 codewords can carry up to 2051 octets, which accommodates two 1024-octet frames.

Table 11-8: Frame Rejection Probabilities, P_{FX} and P_{FY}, for the Last or Only Frame in an Independent CLTU (PLOP-1)

Number of codewords in the CLTU	Total frame lengths up to (octets)	P_{FX} P_{FY}	Channel Bit Error Rate		
			10^{-4}	10^{-5}	10^{-6}
1	7	P_{FX} (TED)	7.87×10^{-3}	7.90×10^{-4}	7.90×10^{-5}
		P_{FY} (SEC)	2.06×10^{-5}	2.07×10^{-7}	2.07×10^{-9}
16	112	P_{FX} (TED)	9.73×10^{-2}	1.02×10^{-2}	1.02×10^{-3}
		P_{FY} (SEC)	3.12×10^{-4}	3.14×10^{-6}	3.14×10^{-8}
37	259	P_{FX} (TED)	2.09×10^{-1}	2.32×10^{-2}	2.34×10^{-3}
		P_{FY} (SEC)	7.21×10^{-4}	7.24×10^{-6}	7.24×10^{-8}
74	518	P_{FX} (TED)	3.74×10^{-1}	4.57×10^{-2}	4.67×10^{-3}
		P_{FY} (SEC)	1.44×10^{-3}	1.45×10^{-5}	1.45×10^{-7}
110	770	P_{FX} (TED)	5.01×10^{-1}	6.71×10^{-2}	6.92×10^{-3}
		P_{FY} (SEC)	2.14×10^{-3}	2.15×10^{-5}	2.15×10^{-7}
147	1029	P_{FX} (TED)	6.05×10^{-1}	8.86×10^{-2}	9.23×10^{-3}
		P_{FY} (SEC)	2.86×10^{-3}	2.87×10^{-5}	2.87×10^{-7}
293	2051	P_{FX} (TED)	8.42×10^{-1}	1.69×10^{-1}	1.83×10^{-2}
		P_{FY} (SEC)	5.68×10^{-3}	5.72×10^{-5}	5.72×10^{-7}
439	3073	P_{FX} (TED)	9.37×10^{-1}	2.42×10^{-1}	2.73×10^{-2}
		P_{FY} (SEC)	8.50×10^{-3}	8.57×10^{-5}	8.57×10^{-7}
586	4102	P_{FX} (TED)	9.75×10^{-1}	3.09×10^{-1}	3.63×10^{-2}
		P_{FY} (SEC)	1.13×10^{-2}	1.14×10^{-4}	1.14×10^{-6}

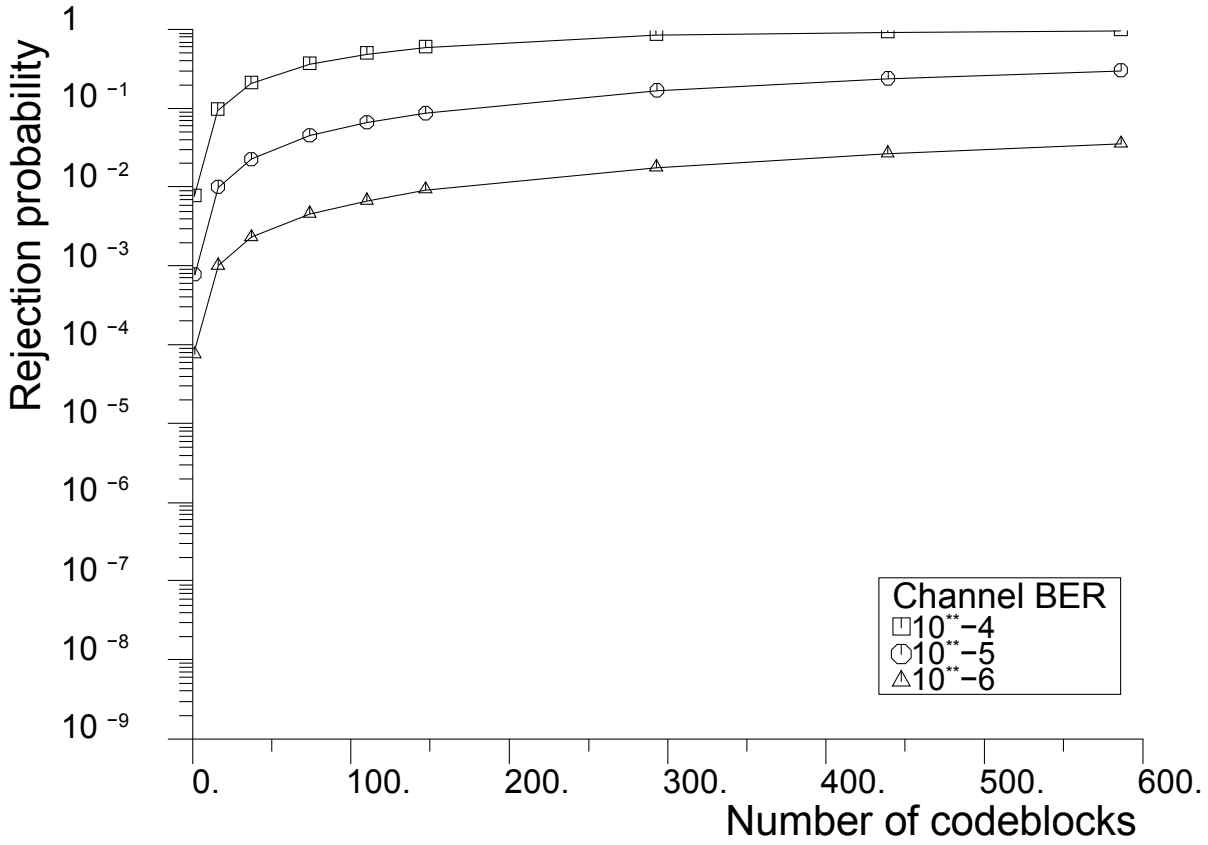

Figure 11-1: Frame Rejection Probability in TED Mode, P_{FX} or P_{F2X}, for the Last or Only Frame in a CLTU (PLOP-1 or PLOP-2)

Figure 11-1 shows the probability P_{FX} of frame rejection in TED mode. The values apply for the last or only frame in an independent CLTU (PLOP-1, table 11-8). Within the accuracy of the tables and figures, the values in TED mode are the same as the probability P_{F2X} when the CLTU is in a sequence of CLTUs (PLOP-2, table 11-9).

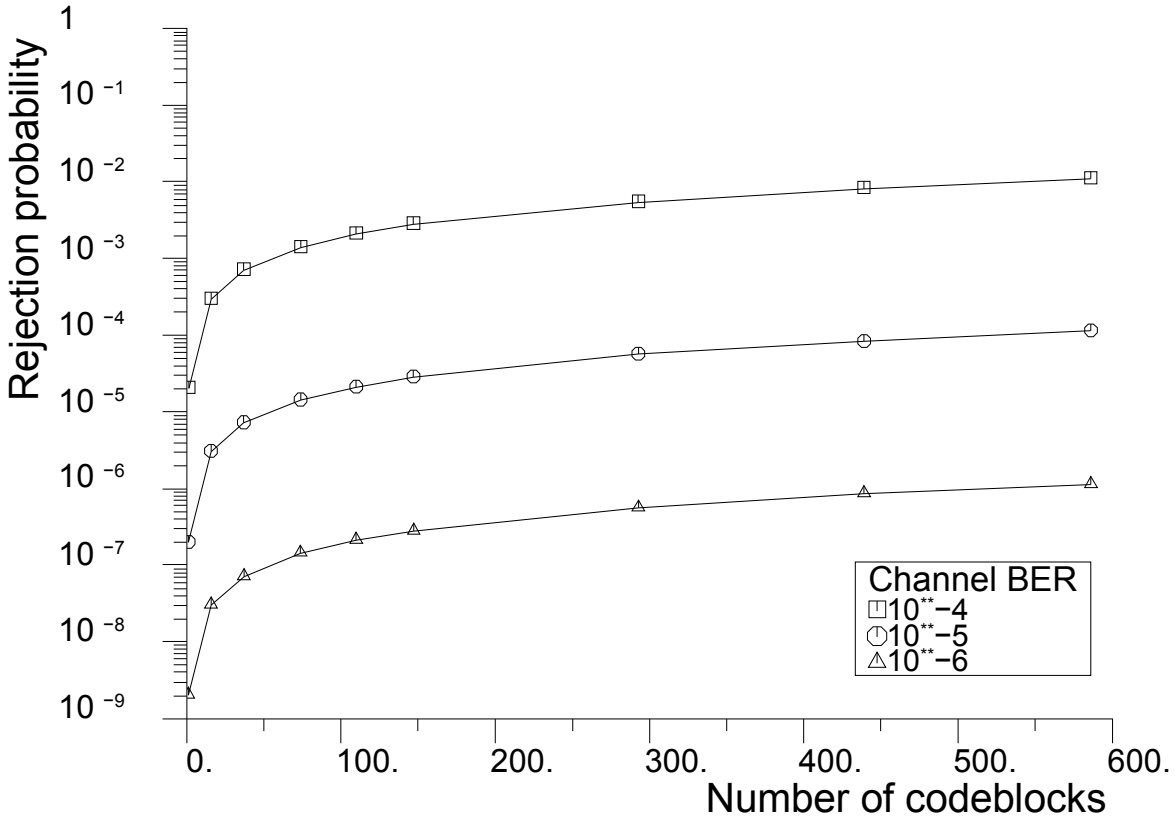

Figure 11-2: Frame Rejection Probability in SEC Mode, P_{FY}, for the Last or Only Frame in an Independent CLTU (PLOP-1)

Figure 11-2 shows the probability P_{FY} of frame rejection in SEC mode. The values apply for the last or only frame in an independent CLTU (PLOP-1, table 11-8). Figure 11-3 below shows the probability in SEC mode when the CLTU is in a sequence of CLTUs (PLOP-2).

11.4.6.3 Sequence of CLTUs (PLOP-2)

If the sending end is operating the PLOP-2 procedure, then the Start Sequence follows the Tail Sequence of the previous CLTU (with an optional length of Idle Sequence between the Tail Sequence and the Start Sequence). Failure of the CLTU reception procedure to recognize the Tail Sequence of a CLTU can cause the following CLTU to be missed.

In calculating the rejection probability with a sequence of CLTUs, three events are considered:

– the CLTU reception procedure is not in SEARCH state when the Start Sequence arrives, because the previous Tail Sequence was missed;

– the CLTU Start Sequence is not recognized;

– the CLTU Start Sequence is recognized, but one of the codewords is rejected.

The probability of CLTU rejection depends on which options are in use in the CLTU reception procedure:

a) no errors accepted in the Start Sequence and codewords decoded in TED mode; or

b) a single error accepted in the Start Sequence and codewords decoded in SEC mode.

For a frame that is the last or only frame in a CLTU, the corresponding frame rejection probabilities, P_{F2X} and P_{F2Y}, are given by:

$$P_{F2X} = P_{TX} + (1 - P_{TX}) (P_{SX} + (1 - P_{SX}) P_{CX})$$

$$P_{F2Y} = P_{TY} + (1 - P_{TY}) (P_{SY} + (1 - P_{SY}) P_{CY}).$$

Table 11-9 shows the values of the probabilities P_{F2X} and P_{F2Y} for the three channel bit error rates. Because the values of P_{TX} are small compared to P_{SX} and P_{CX}, the values of P_{F2X} in the table are the same as the values of P_{FX} in table 11-8. Figure 11-3 shows a graph of the P_{F2Y} values. (See figure 11-1 above for a graph of the P_{F2X} values.)

Table 11-9: Frame Rejection Probabilities, P_{F2X} and P_{F2Y}, for the Last or Only Frame in a CLTU in a Sequence of CLTUs (PLOP-2)

Number of codewords in the CLTU	Total frame lengths up to (octets)	P_{F2X} P_{F2Y}	Channel Bit Error Rate		
			10^{-4}	10^{-5}	10^{-6}
1	7	P_{F2X} (TED)	7.87×10^{-3}	7.90×10^{-4}	7.90×10^{-5}
		P_{F2Y} (SEC)	4.01×10^{-5}	4.02×10^{-7}	4.03×10^{-9}
16	112	P_{F2X} (TED)	9.73×10^{-2}	1.02×10^{-2}	1.02×10^{-3}
		P_{F2Y} (SEC)	3.32×10^{-4}	3.33×10^{-6}	3.33×10^{-8}
37	259	P_{F2X} (TED)	2.09×10^{-1}	2.32×10^{-2}	2.34×10^{-3}
		P_{F2Y} (SEC)	7.40×10^{-4}	7.43×10^{-6}	7.43×10^{-8}
74	518	P_{F2X} (TED)	3.74×10^{-1}	4.57×10^{-2}	4.67×10^{-3}
		P_{F2Y} (SEC)	1.46×10^{-3}	1.47×10^{-5}	1.47×10^{-7}
110	770	P_{F2X} (TED)	5.01×10^{-1}	6.71×10^{-2}	6.92×10^{-3}
		P_{F2Y} (SEC)	2.16×10^{-3}	2.17×10^{-5}	2.17×10^{-7}
147	1029	P_{F2X} (TED)	6.05×10^{-1}	8.86×10^{-2}	9.23×10^{-3}
		P_{F2Y} (SEC)	2.88×10^{-3}	2.89×10^{-5}	2.89×10^{-7}
293	2051	P_{F2X} (TED)	8.42×10^{-1}	1.69×10^{-1}	1.83×10^{-2}
		P_{F2Y} (SEC)	5.70×10^{-3}	5.74×10^{-5}	5.74×10^{-7}
439	3073	P_{F2X} (TED)	9.37×10^{-1}	2.42×10^{-1}	2.73×10^{-2}
		P_{F2Y} (SEC)	8.52×10^{-3}	8.59×10^{-5}	8.59×10^{-7}
586	4102	P_{F2X} (TED)	9.75×10^{-1}	3.09×10^{-1}	3.63×10^{-2}
		P_{F2Y} (SEC)	1.14×10^{-2}	1.15×10^{-4}	1.15×10^{-6}

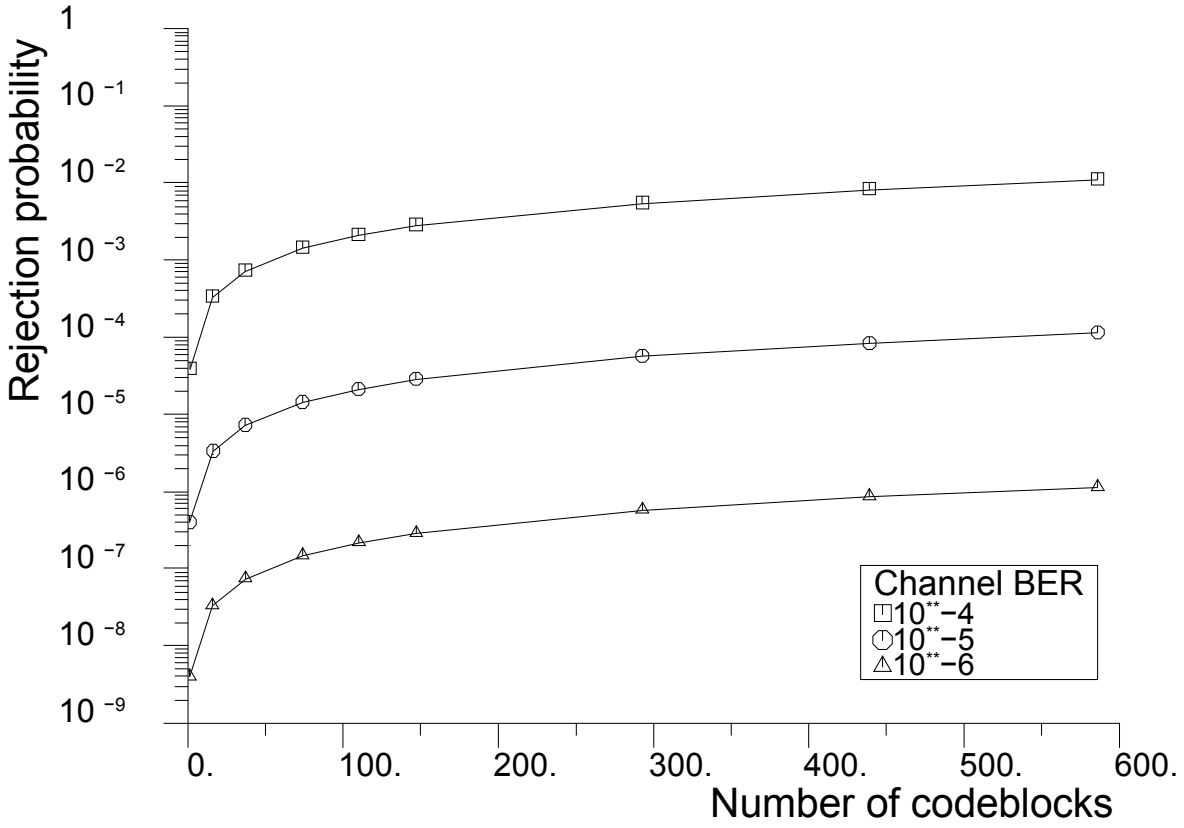

Figure 11-3: Frame Rejection Probability in SEC Mode, P_{F2Y}, for the Last or Only Frame in a CLTU in a Sequence of CLTUs (PLOP-2)

Figure 11-3 shows the probability P_{F2Y} of frame rejection in SEC mode. The values apply for the last or only frame in a CLTU in a sequence of CLTUs (PLOP-2, table 11-9). Figure 11-2 above shows the probability in SEC mode in an independent CLTU.

11.5 FACTORS AFFECTING FRAME UNDETECTED ERROR RATE

11.5.1 GENERAL

The second performance requirement deals with undetected error; it requires that the probability of accepting a frame with an undetected error is smaller than 10^{-9}. Again, the unit of accountability is the frame.

The optional Frame Error Control Field specified in references [3] and [4] can give additional protection against undetected errors. It is a 16-bit field providing a CRC over the contents of the frame. At the receiving end, the Frame Error Control Field is validated by the Frame Validation Check Procedure in the Data Link Protocol Sublayer (see 8.6.2.3).

11.5.2 SOURCES OF UNDETECTED ERRORS

The use of SEC mode for codeword decoding reduces the frame rejection rate, but as the following discussion shows, it increases the probability of an undetected error in the frame.

Subsection 11.4.4.3 above describes the different cases that arise during the decoding of BCH codewords. Table 11-10 shows the cases that lead to undetected errors in the decoding process. In SEC mode, cases A2 and D2 can both occur. In TED mode, only case A2 can occur.

Table 11-10: Sources of Undetected Errors

Case	Description	Modes
A2	The received codeword has an even number of errors (at least 4), and the errors are not detected.	TED & SEC
D2	The received codeword has an odd number of errors (at least 3), and the SEC mode correction delivers a codeword with an even number of errors (at least 2).	SEC

An upper limit for A2 is given by adding the probabilities of 4, 6, 8, … errors occurring. Similarly, an upper limit for D2 is given by adding the probabilities of 3, 5, 7, … errors occurring. As table 11-11 shows, the probabilities decrease rapidly as the number of errors increases, so the upper limit for A2 can be considered as the probability of four errors occurring in the codeword, and the upper limit for D2 can be considered as the probability of three errors occurring in the codeword.

Table 11-11: Probability of *n* Errors Occurring in a Codeword

Number of errors, *n*	Channel Bit Error Rate		
	10^{-4}	10^{-5}	10^{-6}
1	6.26×10^{-3}	6.30×10^{-4}	6.30×10^{-5}
2	1.94×10^{-5}	1.95×10^{-7}	1.95×10^{-9}
3	3.95×10^{-8}	3.97×10^{-11}	3.97×10^{-14}
4	5.92×10^{-11}	5.95×10^{-15}	5.96×10^{-19}
5	6.99×10^{-14}	7.02×10^{-19}	7.03×10^{-24}
6	6.76×10^{-17}	6.79×10^{-23}	6.79×10^{-29}
7	5.50×10^{-20}	5.53×10^{-27}	5.53×10^{-34}
8	3.85×10^{-23}	3.87×10^{-31}	3.87×10^{-39}
9	2.35×10^{-26}	2.37×10^{-35}	2.37×10^{-44}

In a simulation of the decoder in figure 8-4, all combinations of single, double, triple, and quadruple errors were tested (reference [9]) in SEC mode. The tests demonstrated that the decoder has an error-detection performance that is better than the upper limits. Extending the results to include TED mode, the performance is shown in table 11-12.

Table 11-12: Error Detection Performance for a Codeword in SEC and TED Modes

Number of errors	Combinations	Performance in SEC	Performance in TED
1	63	All corrected	All detected
2	1953	All detected	All detected
3	39711	651 detected	All detected
		39060 undetected*	
4	595665	585900 detected	585900 detected
		9765 undetected	9765 undetected
*In all these cases, the SEC correction delivers a codeword with four errors.			

11.5.3 UNDETECTED ERROR PROBABILITIES WITHOUT CRC

Tables 11-11 and 11-12 show error probabilities and decoding performance for a codeword. The following two tables show the probabilities of an undetected error in a frame for the case that the optional CRC is not used. Table 11-13 shows the probabilities for TED mode and table 11-14 shows the probabilities for SEC mode.

Table 11-13: Probability of Undetected Error in a Frame, TED Mode, No CRC

Number of Codewords *N*	Channel Bit Error Rate		
	10^{-4}	10^{-5}	10^{-6}
2	1.89×10^{-12}	1.91×10^{-16}	1.91×10^{-20}
10	9.47×10^{-12}	9.53×10^{-16}	9.53×10^{-20}
20	1.89×10^{-11}	1.91×10^{-15}	1.91×10^{-19}
30	2.84×10^{-11}	2.86×10^{-15}	2.86×10^{-19}
40	3.79×10^{-11}	3.81×10^{-15}	3.81×10^{-19}
50	4.74×10^{-11}	4.76×10^{-15}	4.77×10^{-19}
60	5.68×10^{-11}	5.72×10^{-15}	5.72×10^{-19}
70	6.63×10^{-11}	6.67×10^{-15}	6.67×10^{-19}
80	7.58×10^{-11}	7.62×10^{-15}	7.62×10^{-19}
90	8.53×10^{-11}	8.57×10^{-15}	8.58×10^{-19}
100	9.47×10^{-11}	9.53×10^{-15}	9.53×10^{-19}
110	1.04×10^{-10}	1.05×10^{-14}	1.05×10^{-18}
120	1.14×10^{-10}	1.14×10^{-14}	1.14×10^{-18}
130	1.23×10^{-10}	1.24×10^{-14}	1.24×10^{-18}

Table 11-14: Probability of Undetected Error in a Frame, SEC Mode, No CRC

Number of Codewords *N*	Channel Bit Error Rate		
	10^{-4}	10^{-5}	10^{-6}
2	7.75×10^{-8}	7.79×10^{-11}	7.80×10^{-14}
10	3.88×10^{-7}	3.90×10^{-10}	3.90×10^{-13}
20	7.75×10^{-7}	7.79×10^{-10}	7.80×10^{-13}
30	1.16×10^{-6}	1.17×10^{-9}	1.17×10^{-12}
40	1.55×10^{-6}	1.56×10^{-9}	1.56×10^{-12}
50	1.94×10^{-6}	1.95×10^{-9}	1.95×10^{-12}
60	2.33×10^{-6}	2.34×10^{-9}	2.34×10^{-12}
70	2.71×10^{-6}	2.73×10^{-9}	2.73×10^{-12}
80	3.10×10^{-6}	3.12×10^{-9}	3.12×10^{-12}
90	3.49×10^{-6}	3.51×10^{-9}	3.51×10^{-12}
100	3.88×10^{-6}	3.90×10^{-9}	3.90×10^{-12}
110	4.26×10^{-6}	4.29×10^{-9}	4.29×10^{-12}
120	4.65×10^{-6}	4.68×10^{-9}	4.68×10^{-12}
130	5.04×10^{-6}	5.07×10^{-9}	5.07×10^{-12}

11.5.4 UNDETECTED ERROR PROBABILITIES WITH CRC

The CRC gives additional protection against undetected errors in a frame. In the tests (reference [9]), the performance of the CRC was analyzed for the cases in which the codeword decoding delivers a codeword with errors. The CRC performance for a frame with errors in more than one codeword was also considered.

Tables 11-15 and 11-16 show the probabilities of an undetected error in a frame when the CRC is used.

Table 11-15: Probability of Undetected Error in a Frame, TED Mode, with CRC

Number of Codewords *N*	Channel Bit Error Rate		
	10^{-4}	10^{-5}	10^{-6}
2	8.77×10^{-26}	8.86×10^{-34}	8.87×10^{-42}
10	3.94×10^{-24}	3.99×10^{-32}	3.99×10^{-40}
20	1.67×10^{-23}	1.68×10^{-31}	1.69×10^{-39}
30	3.81×10^{-23}	3.85×10^{-31}	3.86×10^{-39}
40	6.84×10^{-23}	6.91×10^{-31}	6.92×10^{-39}
50	1.07×10^{-22}	1.09×10^{-30}	1.09×10^{-38}
60	1.55×10^{-22}	1.57×10^{-30}	1.57×10^{-38}
70	2.12×10^{-22}	2.14×10^{-30}	2.14×10^{-38}
80	2.77×10^{-22}	2.80×10^{-30}	2.80×10^{-38}
90	3.51×10^{-22}	3.55×10^{-30}	3.55×10^{-38}
100	4.34×10^{-22}	4.39×10^{-30}	4.39×10^{-38}
110	5.26×10^{-22}	5.31×10^{-30}	5.32×10^{-38}
120	6.26×10^{-22}	6.33×10^{-30}	6.33×10^{-38}
130	7.35×10^{-22}	7.43×10^{-30}	7.44×10^{-38}

Table 11-16: Probability of Undetected Error in a Frame, SEC Mode, with CRC

Number of Codewords *N*	Channel Bit Error Rate		
	10^{-4}	10^{-5}	10^{-6}
2	2.55×10^{-20}	2.58×10^{-26}	2.58×10^{-32}
10	1.15×10^{-18}	1.16×10^{-24}	1.16×10^{-30}
20	4.85×10^{-18}	4.91×10^{-24}	4.91×10^{-30}
30	1.11×10^{-17}	1.12×10^{-23}	1.12×10^{-29}
40	1.99×10^{-17}	2.01×10^{-23}	2.02×10^{-29}
50	3.13×10^{-17}	3.16×10^{-23}	3.17×10^{-29}
60	4.52×10^{-17}	4.57×10^{-23}	4.58×10^{-29}
70	6.17×10^{-17}	6.24×10^{-23}	6.24×10^{-29}
80	8.07×10^{-17}	8.16×10^{-23}	8.17×10^{-29}
90	1.02×10^{-16}	1.03×10^{-22}	1.04×10^{-28}
100	1.26×10^{-16}	1.28×10^{-22}	1.28×10^{-28}
110	1.53×10^{-16}	1.55×10^{-22}	1.55×10^{-28}
120	1.82×10^{-16}	1.84×10^{-22}	1.85×10^{-28}
130	2.14×10^{-16}	2.17×10^{-22}	2.17×10^{-28}

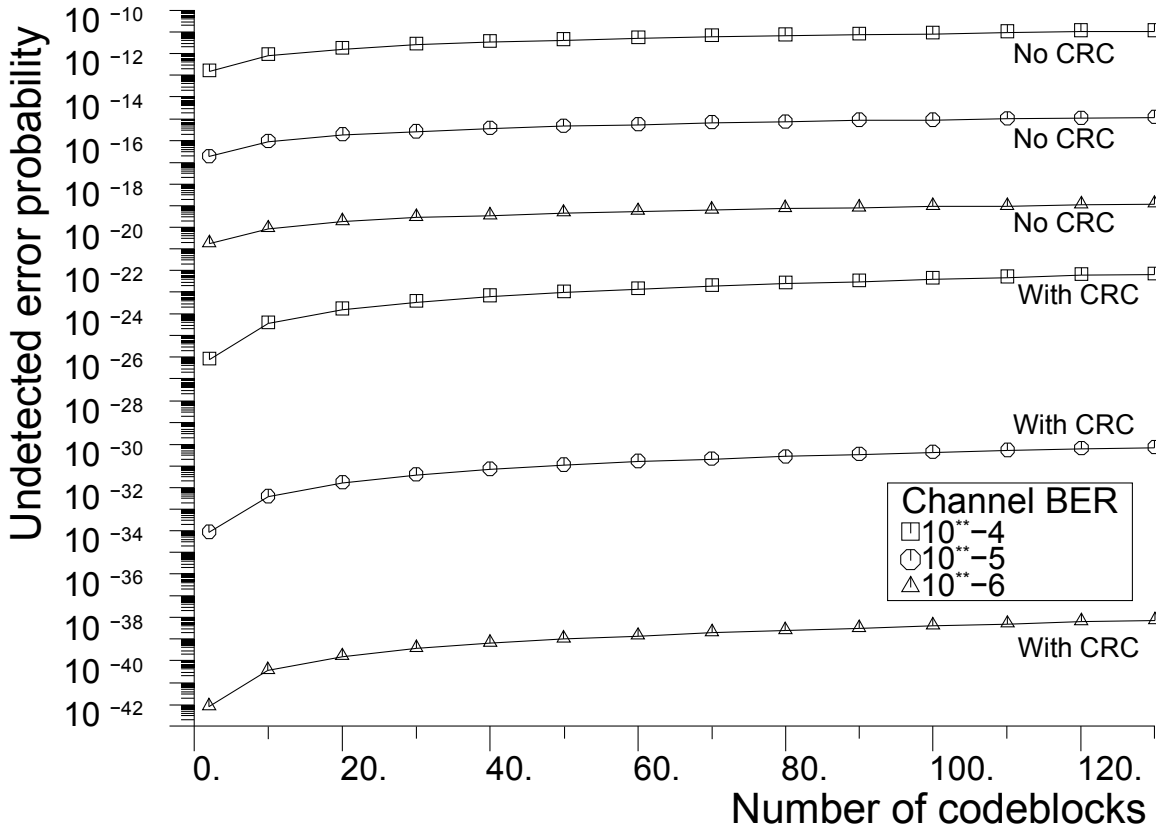

Figure 11-4: Probability of Undetected Error in a Frame in TED Mode

Figure 11-4 brings together the results of table 11-13 and table 11-15 to show the probabilities of undetected error in a frame in TED mode, with and without the CRC.

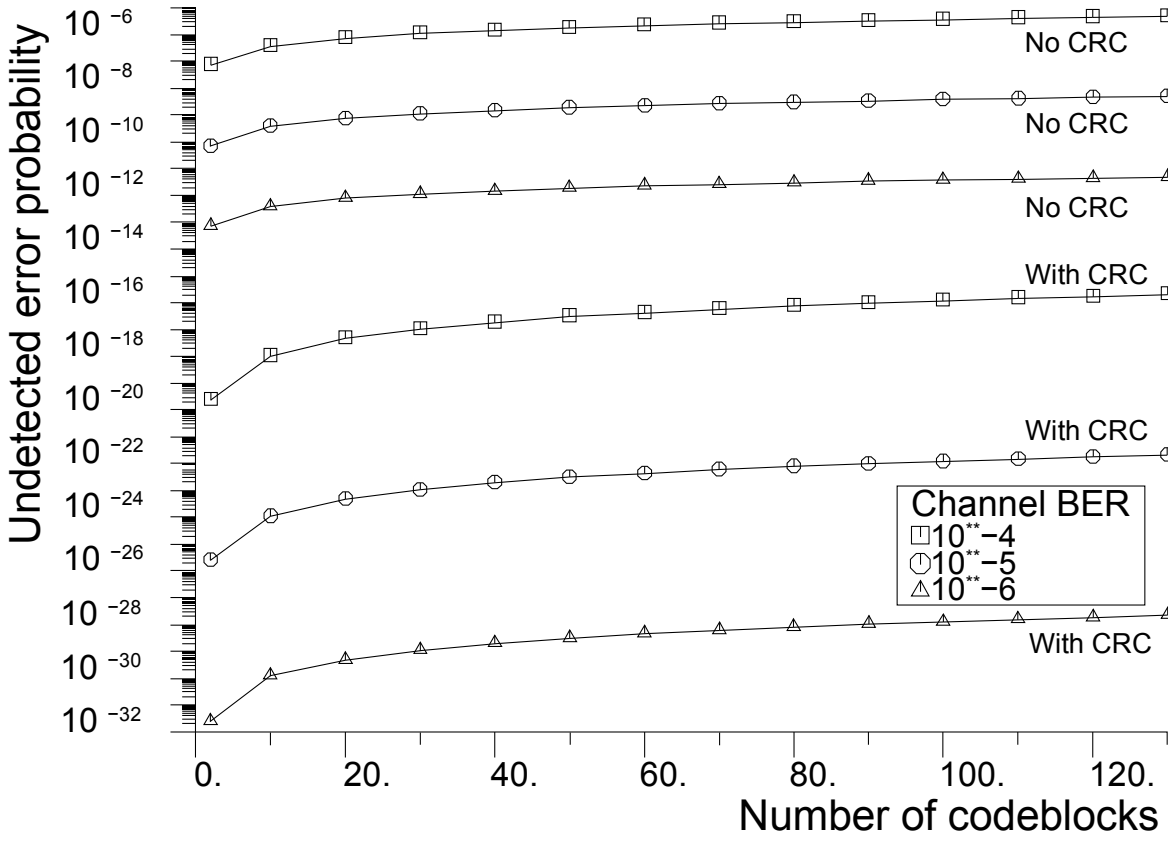

Figure 11-5: Probability of Undetected Error in a Frame in SEC Mode

Figure 11-5 brings together the results of table 11-14 and table 11-16 to show the probabilities of undetected error in a frame in SEC mode, with and without the CRC.

11.6 LDPC CODE PERFORMANCE WITH TYPICAL ITERATIVE DECODERS

The Belief Propagation (BP) decoder operates by iteratively passing messages, representing 'beliefs' about the transmitted code symbols, between computational nodes. Different message-passing schedules are possible: the 'flooding' schedule exchanges all messages within a half-iteration simultaneously and is a well-understood reference case; layered schedules are addressed in 11.7. The computational nodes may also either perform the optimal Sum-Product Algorithm (SPA), or a lower-complexity Normalized Min-Sum (NMS) algorithm. Performance improves as more iterations are performed, but with diminishing returns as that number gets large. Unless otherwise noted, performance results reported in this section were determined with 100 iterations, which is well beyond the point where performance improvements become negligible.

Figure 11-6 reports the performance (CER vs. E_b/N_0) obtained for the (128,64) and (512,256) codes with SPA and NMS (reference [26]) algorithms. In both cases, the performance of the optimal and the sub-optimal algorithms are virtually the same.

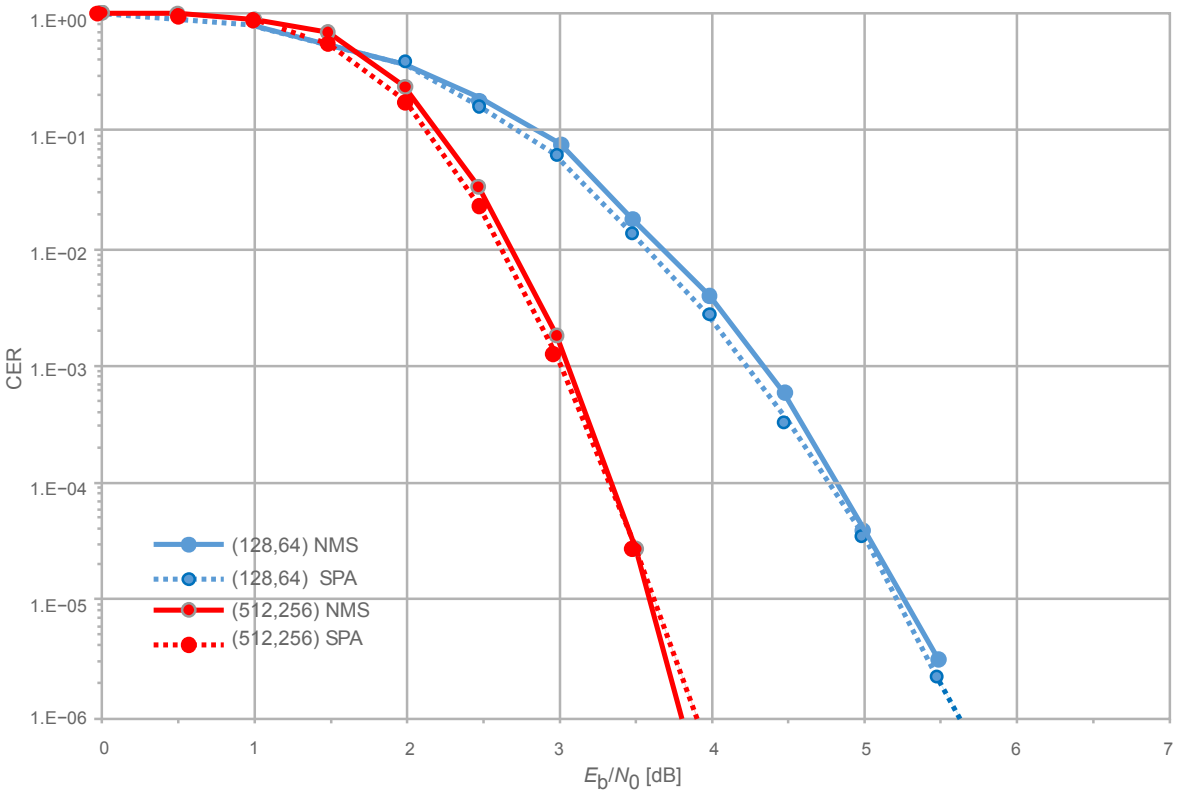

Figure 11-6: Codeword Error Rates for Two Iterative Decoders

Bit error rates for the LDPC codes with a 'flooding' BP decoder are shown in figure 11-7, in comparison to several other codes. Bit Error Rate (BER) curves for the short (n=128, k=64) code are shown in green and for the longer (n=512, k=256) code in red. For comparison, performance of the BCH code is shown in both its TED and SEC/Double Error Detection (DED) modes. While the BCH code in TED mode has negative coding gain, it has error detection capability that is not shown in this graph. Also shown are the rate-1/2 capacity and uncoded curves.

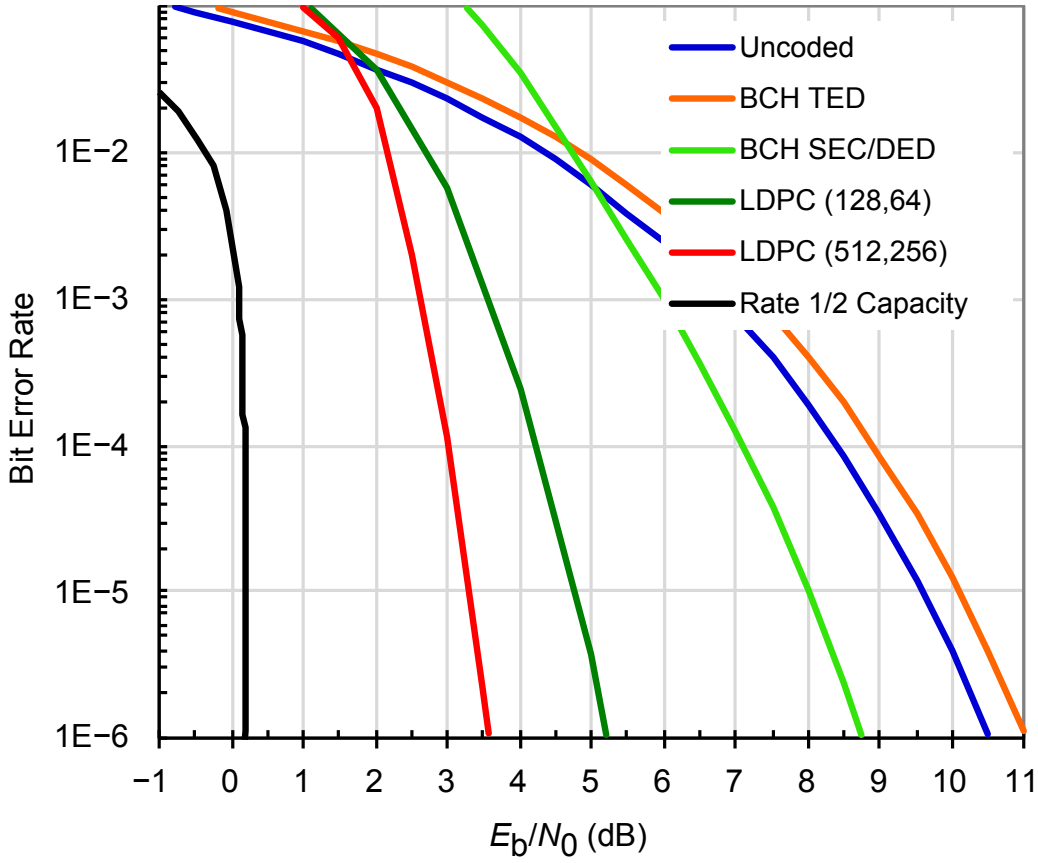

Figure 11-7: Bit Error Rates of Several Codes

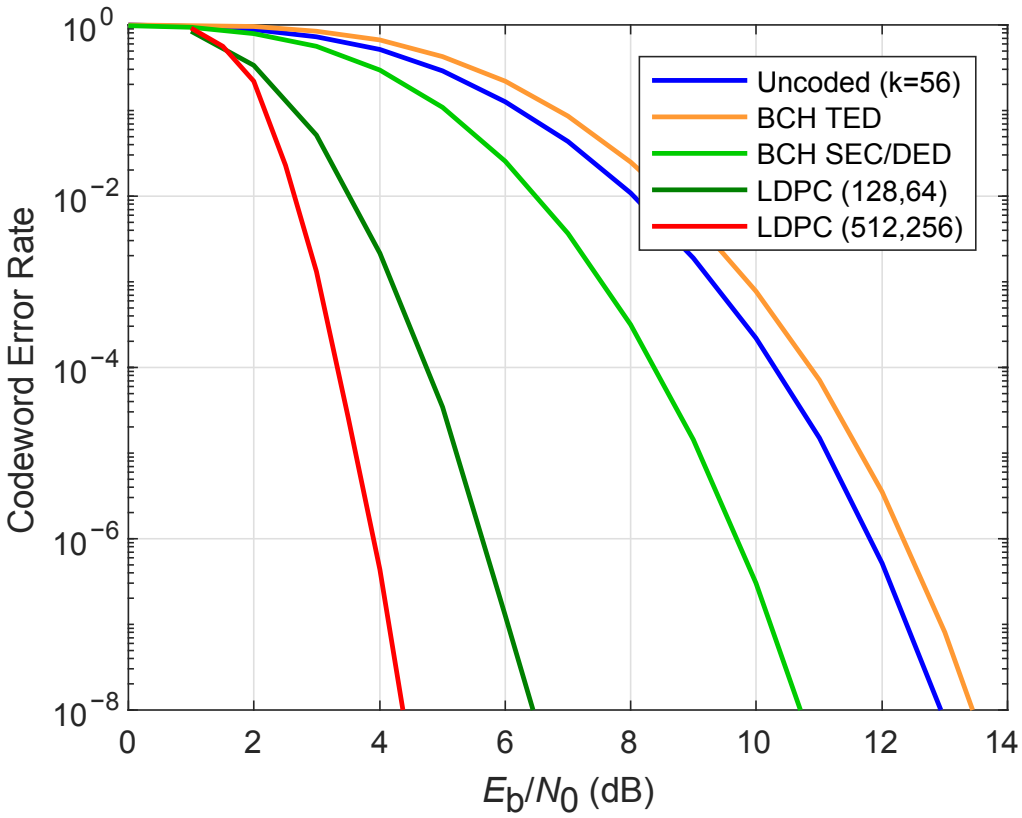

Figure 11-8: Codeword Error Rates of Several Codes

Codeword error rates are shown for the same set of codes in figure 11-8. Some decoders can detect most of their codeword errors, and report a decoding failure rather than an incorrect result. This is beneficial, particularly in a spacecraft telecommand setting where incorrect commands may incur a large cost. There is a trade-off between the undetected error rate and the codeword error rate determined by the 'completeness' of the decoder. Undetected error rates are shown for some codes in figure 11-9, where an SPA decoder was used for the LDPC codes. Performance curves for the two BCH decoders appear in reverse order in the two figures. A typical LDPC decoder detects most of its errors, so the undetected error rate is low for the (128,64) code, and unknown for the (512,256) code. Methods for varying the 'completeness' of an LDPC decoder are described in references [18] and [19].

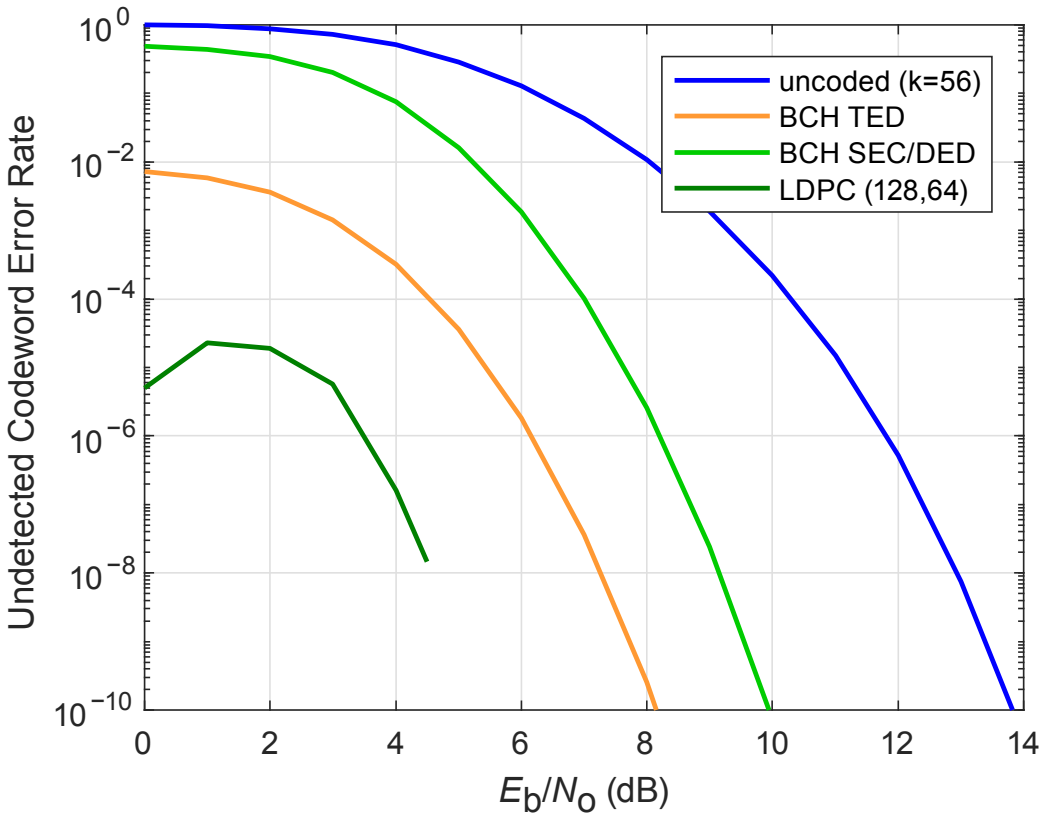

Figure 11-9: Undetected Codeword Error Rates of Several Codes, Using an SPA LDPC Decoder

11.7 LAYERED-BP DECODING

The standard BP decoder uses the flooding schedule where variable node processing begins after all the check nodes have been updated, and vice versa. In contrast, layered-BP decoding divides the parity check matrix into sub-matrices, called layers. The layers may be either horizontal, so that each layer consists of a group of check nodes; or vertical, where each layer consists of a group of variable nodes. In either case, the nodes within a layer are updated simultaneously, and the layers are updated sequentially within an iteration. In this manner, the updated extrinsic information in one layer can be used in the same iteration when updating the next layer, thus accelerating the convergence speed. The number of nodes in each layer determines the degree of parallelism, and may vary between layers.

Horizontal layered BP

In the processing of each horizontal layer, firstly, the old check node messages generated by this layer in the previous iteration (l – 1) are subtracted from the a-posteriori values of the corresponding variable nodes. That is,

$$L(q_{ji}) = L(q_i) - L^{(l-1)}(r_{ji})$$

where $L(q_i)$ is the a-posteriori Log-Likelihood Ratio (LLR) associated with the ith variable node, $L^{(l-1)}(r_{ji})$ is the incoming message from check node j in the previous iteration, and $L(q_{ji})$ is the outgoing message to check node j. Then the check nodes in this layer perform their computations to get new check node messages,

$$L^{(l)}(r_{ji}) = 2\tanh^{-1}\left\{\prod_{i'\in N(j)\backslash i}\tanh\left(\frac{L(q_{ji'})}{2}\right)\right\}$$

where \ indicates exclusion and $N(j)$ is the set of variable nodes connected to check node j, for example, $N(j)\backslash i$ is the set of variable nodes connected to check node j, except for variable node i. Next, the a-posteriori values of the variable nodes are updated by adding the newly generated check node messages,

$$L(q_i) = L(q_{ji}) + L^{(l)}(r_{ji})$$

Finally, the updated a-posteriori values are passed to the next layer as an input, and this process is repeated for each subsequent layer. When all layers have been updated, an iteration of the horizontal layered-BP decoding algorithm is complete.

Vertical layered BP

The processing schedule for vertical layered-BP decoding is similar with the one used in horizontal layered-BP decoding, but with a different processing direction. In log-likelihood ratio form, the equations use the operator $[\pm]$, defined as

$$L_1[\pm]L_2 = \ln\left(\frac{1 \pm e^{L_1}e^{L_2}}{e^{L_1} \pm e^{L_2}}\right)$$

First, the old variable node messages are removed from the a-posteriori LLRs for the check nodes:

$$L(r_{ji}) = L(r_j)[-]L^{(l-1)}(q_{ji})$$

This produces extrinsic messages, $L(r_{ji})$, for the variable nodes in the layer. Then the standard variable node update equation is applied,

$$L^{(l)}(q_{ji}) = L_{\mathrm{ch},i} + \sum_{j'\in M(i)\backslash j} L(r_{j'i})$$

Where $L_{\mathrm{ch},i}$ is the channel information about variable node i, and $M(i)$ is the set of check nodes connected to variable node i, for example, $M(i)\backslash j$ is the set of check nodes connected to variable node i, except for check node j. Finally, the resulting variable node messages are incorporated into the check node extrinsics,

$$L(r_j) = L(r_{ji})[+]L^{(l)}(q_{ji})$$

for use in updating the next layer.

Implementation and performance of layered BP

Because of practical limitations on hardware, decoders are usually implemented in a semi-parallel manner, so that only some variable and check node processing is performed simultaneously. This makes layered-BP decoding attractive; in a fully parallel implementation, layered-BP with a single layer is equivalent to standard BP.

The schematic diagram for a layered-BP decoder is shown in figure 11-10. The figure gives a universal architecture for either a horizontal or vertical layer processor. In this figure, CNU and VNU stand for check node processing unit, and variable node processing unit, respectively. Since the updating procedure is identical between different layers, only one physical layer processor is instantiated, and is used by each layer sequentially.

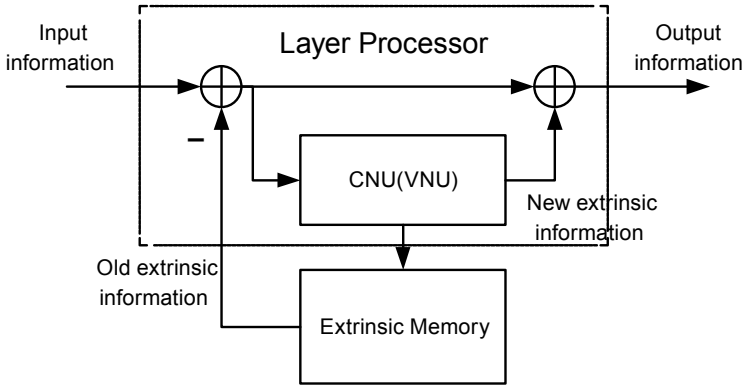

Figure 11-10: A Universal Architecture for Layered-BP Decoder

Bit error rate curves for the two uplink LDPC codes under layered-BP decoding were determined by software simulation with floating-point operation, and the results are shown in figure 11-11. Each curve uses a maximum of 20 iterations; there are 16 nodes per layer for the (128,64) code, and 64 nodes per layer for the (512,256) code. The parity check matrix was partitioned into layers following the circulant structure, so that each row of circulants was taken as a layer for horizontal layered-BP decoding, and each column for vertical layered-BP decoding. For comparison, the BER performance of the standard BP algorithm is also shown. Both vertical and horizontal layered-BP decoding have a slightly better performance than that of standard BP decoding.

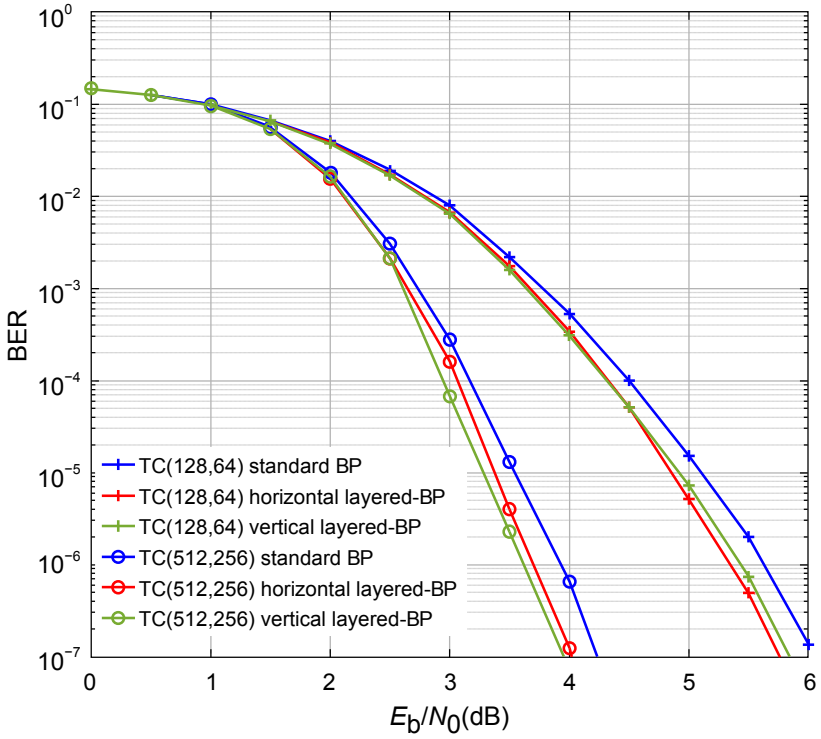

Figure 11-11: BER Curves of the Two Uplink LDPC Codes under Layered-BP Decoding

The average number of iterations for the (128,64) LDPC code under different decoding methods are shown on the left side of figure 11-12, and the results for the (512,256) LDPC code are shown on the right side. Again, the maximum number of iterations in each case is 20. The average number of iterations used by layered-BP is less than that of standard BP, indicating a faster convergence speed.

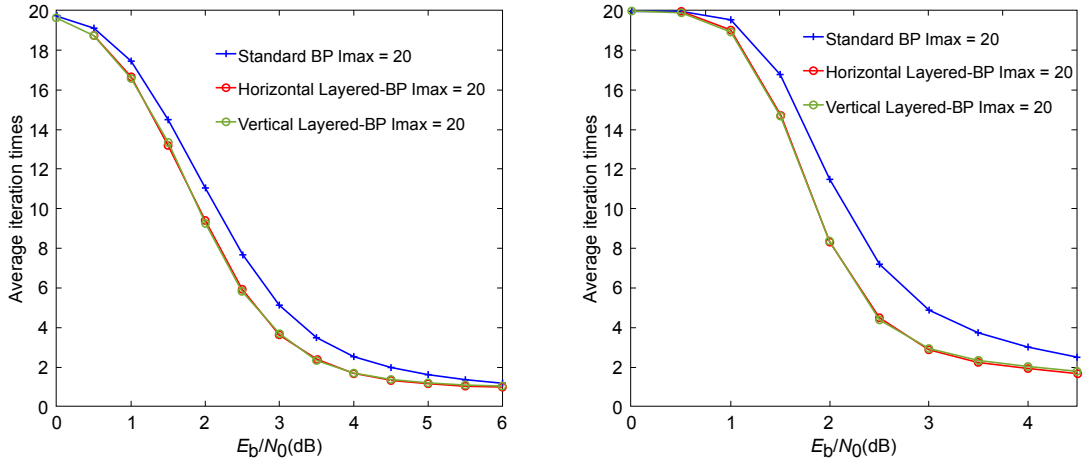

Figure 11-12: The Average Iteration Times of Layered-BP

11.8 MOST RELIABLE BASIS DECODING

The decoding performance achievable with the LDPC codes described in reference [1] can be significantly improved by the use of a MRB algorithm (references [27]–[28]), or by a combination, that can be indicated as hybrid decoding, of such an algorithm with an iterative one [29].

The MRB is a non-iterative soft-decoding procedure, potentially able to achieve performance very close to that of a maximum likelihood (ML) decoder. While a detailed description of the MRB algorithm can be found in the literature (e.g., references [27]–[28]), a summary description is provided in the following.

The rationale of the MRB algorithm is a transformation from the original code generator matrix, **G**, to a new matrix, **G***, based on the k most reliable bits[1] among the received ones. Once the k most reliable bits are identified and collected in a vector **v***, the following steps are performed:

- perform Gauss-Jordan elimination on matrix **G**, with respect to the positions of the k bits, in such a way as to obtain a systematic generator matrix **G*** corresponding to such bits;
- encode **v*** by **G*** to obtain a candidate codeword **c*** = **v*G***;
- select an integer i, that is called the order of the algorithm (denoting by MRB(i) an instance of the algorithm with order i);
- consider all (or an appropriate subset of) test error patterns (TEPs) of length k and Hamming weight $w \leq i$;
- for each pattern: add it to **v***, encode by **G***, verify if the Euclidean distance from the received vector is smaller than that of the previous candidate codeword and, if this is true, update the candidate.

At the end of the process, the algorithm always provides a codeword, at minimum distance from the received vector within the considered set. If the order i of the MRB was equal to k, then this would correspond to the ML decoding. While this is hardly feasible, it is possible to select a practical order value (at least for the short (128,64) code) still sufficient to achieve remarking performance.

The main drawback of MRB is its complexity, limiting its practical application to the (128,64) code and for relatively low data rates (up to few kb/s). Besides, a decoder based on MRB is complete, that is, it always returns a codeword, requiring then external means to detect a failure, for example, use of a Cyclic Redundancy Check (CRC), or the end of a CLTU, for example, a CLTU Tail Sequence.

[1] The k bits, out of the received n bits, whose Log-Likelihood Ratioss (LLR) given the channel observation, are the largest ones.

A potential improvement over the MRB is its use together with an iterative decoding algorithm, such as the Sum-Product (SPA) or the Normalized Min-Sum (NMS) algorithms, in a so-called hybrid decoder. In such case, the iterative decoder (e.g., SPA or NMS) would be activated first. Only if an error is detected after the iterative decoding procedure (i.e., at the end of the maximum number of iterations), then the original received symbols are passed to the MRB decoder. While the overall complexity (in terms of HW or SW) is actually higher than for the MRB decoder (or the iterative decoder) alone, the MRB portion is activated only for a fraction of time, and still it provides the expected performance gain over the iterative decoder. A comparison, in term of CER, between the iterative NMS and the hybrid NMS+MRB algorithms for the (128,64) code is reported in figure 11-13. In this case, MRB(4) (that is, order 4) was assumed. As it can be seen, the hybrid NMS+MRB algorithm provides a significant advantage (up to 1.5 dB) over the NMS alone, with performance in line with those achievable with the longer (512,256) code (and NMS decoder).

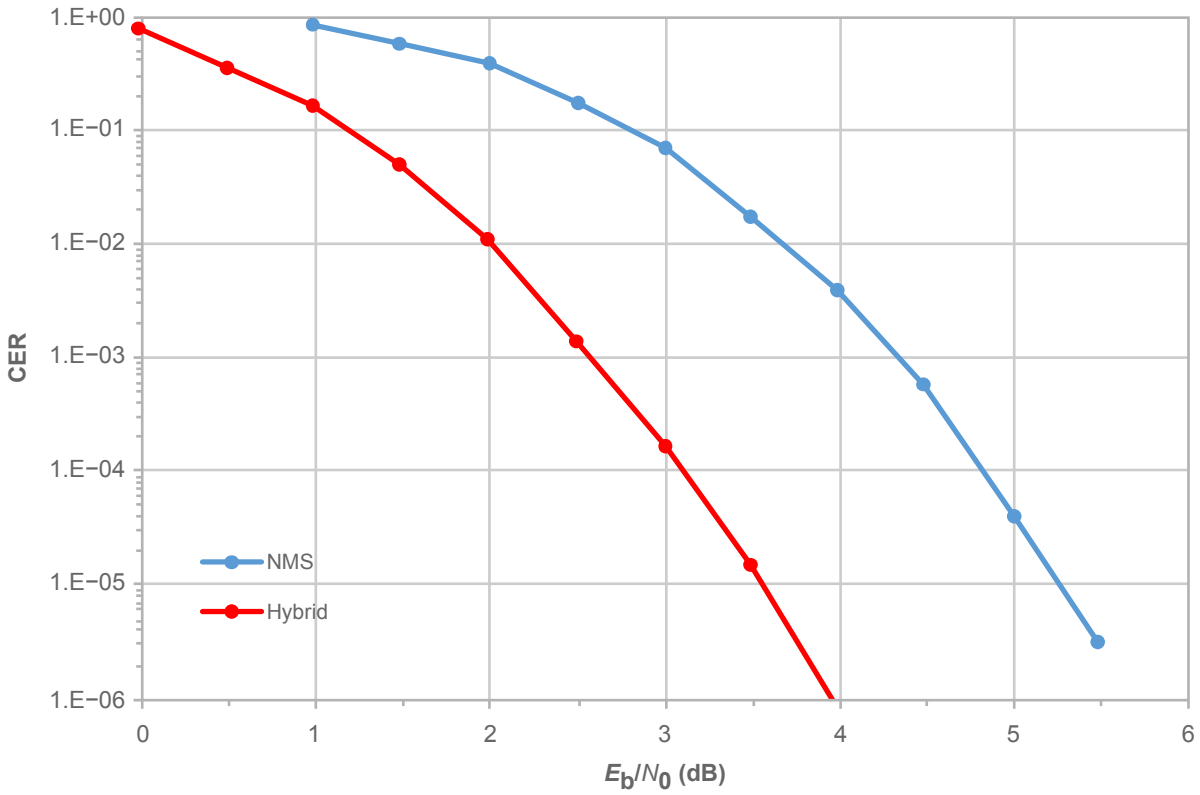

Figure 11-13: Comparison between NMS and Hybrid Decoding Algorithms, (128,64) Code

11.9 SOFT-SYMBOL DETECTION OF THE START AND TAIL SEQUENCES

11.9.1 START SEQUENCE

Because LDPC codes operate at a lower symbol SNR than BCH codes do, a longer start sequence is required for reliable detection. The natural procedure is to scan through a received bitstream with a correlator or some variant thereof, until a match exceeding some threshold is found. Rather than a correlator, the optimum algorithm was derived by Massey (reference [30]), and he noted that an approximate version of that algorithm gave similar performance with dramatically reduced complexity. Decreasing the threshold decreases the probability of missing a start sequence, but also increases the probability of a false alarm. With P(miss) being the probability of missing the start sequence, and P(FA) being the probability of false alarm in detecting the start sequence, such Receiver Operating Characteristics (ROCs) are plotted at a range of SNRs in figure 11-14. Because the LDPC codes have a threshold of E_b/N_0 around 3 to 4 dB, the start sequence detector should have a point on the ROC curve that allows reliable operation at this level. That is, P(miss) should be smaller than the codeword error rate of the decoder, and P(FA), times the expected number of symbols to be tested, should also be smaller than the decoder's codeword error rate. Figure 11-14 shows that a 64-symbol start sequence with the Approximate Massey algorithm achieves this requirement; similar figures would show that 16- and 32-symbol start sequences are not sufficient.

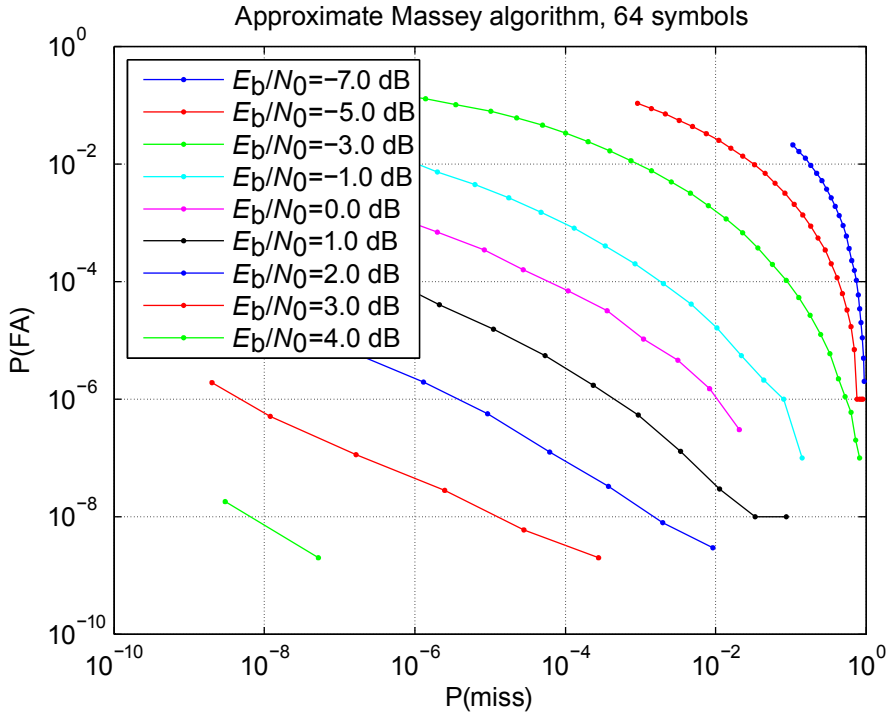

Figure 11-14: False-Alarm vs. Miss Rate for Start Sequence Detection

The standard 64-symbol start sequence has the autocorrelation function shown on the left side of figure 11-15, and cross correlation with the alternating …0101… sequence on the right side. While these may not be optimum, there are no substantial defects that would cause concern for false acquisition.

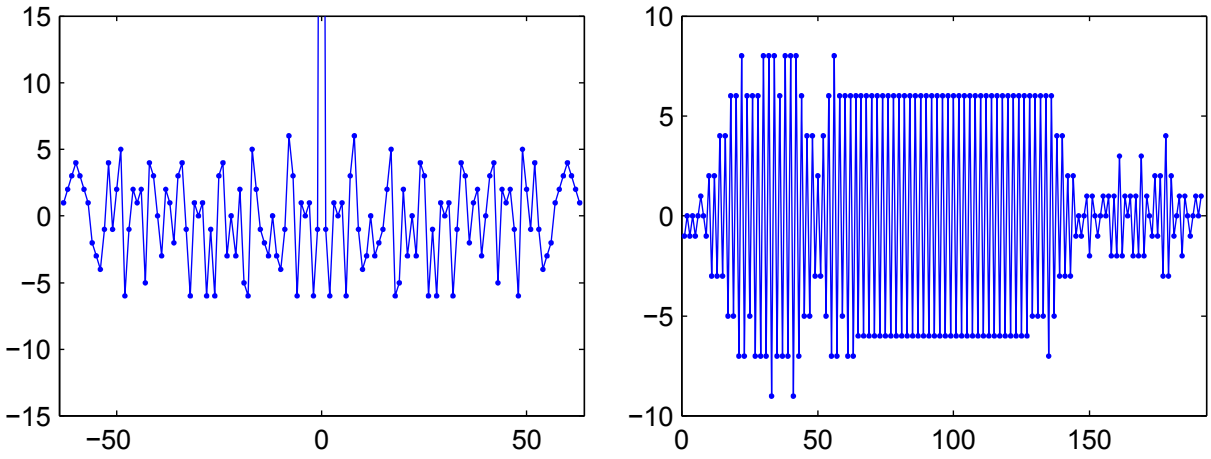

Figure 11-15: The 64-Symbol Start Sequence's Autocorrelation, and Cross Correlation with the Alternating …0101… Idle Sequence

11.9.2 TAIL SEQUENCE

With the BCH code, the tail sequence was chosen to be an uncorrectable vector, with a Hamming distance at least two from each of the BCH codewords. This requires no detection circuitry at the receiver, for the expected response is identical to that when an uncorrectably noisy codeword is received. The possible drawback is that such a receiver does not distinguish between a properly terminated CLTU and an invalid CLTU containing a corrupted codeword; separating these is left as a task for the upper protocol layers.

With LDPC codes, one can again mark the end of a CLTU with an uncorrectable codeword that is 128 or 512 bits long, though the overhead incurred is somewhat unattractive. Another alternative is to omit the tail sequence altogether, in which case the decoder will almost certainly fail to decode whatever symbols follow. Both alternatives are permitted by the standard for the (128,64) code, and only the latter for the (512,256) code. They have modestly different characteristics. The receiver implementation differs between the two cases by the presence of Event E5 in the state diagram shown in figure 8-1 and table 4-1, and they have modestly different performance characteristics.

When the tail sequence is included, the receiver must check for its presence at the end of each codeword. If found, it is discarded, and the receiver returns to its search state. This allows the receiver to report when a CLTU is properly terminated. Because the tail sequence is explicitly identified, it frees the decoder from any obligation to fail when the tail sequence is encountered. Hence, a complete decoder could be used, that decodes any sequence of n symbols to a codeword. While complete decoders are generally computationally complex, they may provide improvements in decoding performance of 1.5 dB or more.

Alternatively, the tail sequence may be omitted. This alternative is attractive for it saves the overhead of the tail sequence entirely, and offers minor simplifications to the implementation of both the transmit and receive sides. However, it imposes several restrictions on the receiver. The demodulator is obliged to produce enough symbols for the LDPC decoder to attempt, and fail, to decode a subsequent codeword. As shown in figure 11-9, the undetected error rate of an SPA LDPC decoder is low enough at any SNR that the probability of erroneously accepting a codeword beyond the end of a CLTU is small, but another LDPC decoder may not have this property. As with the BCH case, the receiver is also unable to distinguish between an intentional decoder failure at the end of a CLTU and an unintentional failure due to channel noise. Aside from the loss of potentially valuable statistics, it places the burden of validating a CLTU on upper protocol layers.

For the (128,64) code, the chosen tail sequence has hexadecimal representation:

55555556 AAAAAAAA 55555555 55555555

This sequence has a high bit transition density, and if derandomized, it is reasonably distinct from any LDPC codeword. If derandomized, the distribution of nearby codewords is:

Hamming Distance	Number of codewords
15	3
16	0
17	67
18	0
19	Many

The tail sequence was designed for an earlier version of this standard in which there was no randomizer after the encoder; in that architecture, its Hamming distance of 18 from the nearest LDPC codeword was a notable attribute.

ANNEX A

GLOSSARY

Acquisition Sequence: A specific high transition density bit pattern transmitted to permit the receiving end to acquire symbol synchronization. (See 7.2.2.)

Bit Transition Generator: A generator that produces the random sequence of 255 bits, used in the optional randomization procedure at the sending and receiving ends. (See 9.3.)

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

Carrier Modulation Modes: Different states of data modulation upon the RF carrier that creates the Physical Channel. (See 7.2.4.)

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

Communications Link Transmission Unit, CLTU: A Synchronization and Channel Coding Sublayer data structure consisting of a Start Sequence, a set of BCH codewords, and a Tail Sequence. (See 5.3.)

Communications Session: A continuous period of time during which the signal path is established for the communications channel. (See 7.2.1.)

Cyclic Redundancy Check, CRC: The error detection method employed in the optional Frame Error Control Field of a Transfer Frame. The CRC provides a means to detect errors introduced into the Frame during transmission. At the sending end, the CRC value is calculated from the contents of the Frame and placed in the Frame Error Control Field. At the receiving end, the CRC value is calculated and compared with the value in the Frame Error Control Field. If the values are different, the Frame contains errors. The CRC is used only for error detection, not for error correction.

Data Link Protocol Sublayer: The sublayer above the Synchronization and Channel Coding Sublayer. At the sending end, it delivers Transfer Frames to the Synchronization and Channel Coding Sublayer. At the receiving end, the Synchronization and Channel Coding Sublayer delivers candidate Transfer Frames to the Data Link Protocol Sublayer.

Decoding: The processing of a received noisy codeword, through which the information and parity bits of the codeword are used to detect or correct errors.

Encoding: The processing of the information bits in a codeword at the sending end to generate the parity bits.

Event: In this report, an occurrence that causes the CLTU Reception Procedure to perform actions and change states. (See 8.2.)

Fill bits: Bits used to complete an integral number of codewords. Because the data received from the Data Link Protocol Sublayer may not fit exactly in an integral number of codewords, the Information field of the last codeword in a CLTU may contain fill bits. An octet of fill bits has the hexadecimal value 55. The fill bits are transmitted as part of the CLTU. At the receiving end, the fill bits are detected and removed by the Data Link Protocol Sublayer. (See 3.4 and 8.6.2.2.)

Filler Bit: The last bit of the last octet of a BCH codeword. The Filler Bit is always zero. (See 3.3.)

Frame: In this document, a TC Transfer Frame or a USLP Transfer Frame.

Idle Sequence: A specific high transition density bit pattern transmitted during a communications session in the absence of a CLTU, to enable the receiver to maintain bit synchronization. (See 7.2.3.)

Octet: A contiguous string of eight bits; an eight-bit word.

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

Physical Layer: The layer below the Synchronization and Channel Coding Sublayer, which provides the RF channel. At the sending end, it provides the radio frequency and modulation techniques required to create and operate the channel. At the receiving end, it provides the reception, demodulation, and symbol synchronization for the channel.

Physical Layer Operations Procedure, PLOP: A specific procedure of the Physical Layer designed to activate and deactivate the physical communications channel by invoking RF carrier and modulation techniques. (See section 7.)

Pseudo-randomization: A bandwidth-efficient technique of algorithmically translating the data bits to ensure frequent bit transitions in the communications channel, also called randomization in this report. No additional bits are added by this process.

Randomization: Pseudo-randomization. For brevity, ‘randomization’ is used in place of ‘pseudo-randomization’ in this report.

Reliable: Meets the quality, quantity, continuity, and completeness criteria specified by the communications system.

Start Sequence: A specific bit pattern that delimits the start of the first BCH codeword and, if necessary, resolves the sense of a ‘1’ and ‘0’ in the CLTU. (See 5.4.)

Symbol: A bit in an encoded data stream.

Synchronization and Channel Coding Sublayer: The sublayer responsible for 1) the provision of data structures to enable bit synchronization to be achieved and maintained and 2) the coding of Frames to provide a forward error detection and correction capability. The sublayer is the subject of this report and of the Recommended Standard (reference [1]).

TC Transfer Frame: The data unit generated by the TC Space Data Link Protocol. The TC Transfer Frame is specified in reference [3].

Tail Sequence: A specific data pattern that delimits the end of a CLTU. The Tail Sequence field has the same length as a codeword, but its contents differ from a valid or correctable codeword.

USLP Transfer Frame: The data unit generated by the Unified Space Data Link Protocol. The USLP Transfer Frame is specified in reference [4].

ANNEX B

ACRONYMS AND ABBREVIATIONS

ACK	positive acknowledgement
ARQ	automatic repeat queuing
BCH	Bose-Chaudhuri-Hocquenghem
BER	bit error rate
BP	belief propagation
BR	buffer register
BTG	bit transition generator
CCSDS	Consultative Committee for Space Data Systems
CER	codeword error rate
CLCW	communications link control word
CLTU	communications link transmission unit
CMM	carrier modulation mode
CNU	check node processing unit
CR	codeword rejection
CRC	cyclic redundancy check
DED	double error detection
EOD	even/odd detector
ESA	European Space Agency
FBA	filler bit augmentation
GBN	go-back-*n* (scheme)
HW	hardware

LDPC	low-density parity-check
LLR	log-likelihood ratio
MGBN	go-back-*n* (scheme) with multiple copies
ML	maximum likelihood
MRB	most reliable basis
MSB	most significant bit
NAK	negative acknowledgement
NMS	normalized min-sum
OSI	Open Systems Interconnection
PAR	error parity check
PLOP	physical layer operations procedure
PLR	position location register
RF	radio frequency
ROC	receiver operating characteristic
R-S	Reed-Solomon
SB	syndrome binary
SEC	single error correction
SNR	signal-to-noise ratio
SPA	sum-product algorithm
SR	syndrome register
SW	software
SYND	syndrome value
TC	telecommand

TED	triple error detection
TEP	test error pattern
TM	telemetry
USLP	Unified Space Data Link Protocol
VNU	variable node processing unit

ANNEX C

THEORETICAL BACKGROUND OF THE RANDOMIZATION SEQUENCE

NOTE – This annex contains an edited version of *Selection of New Randomizer* by Larsen, Justesen, and Paaske, January 1994. The report was produced for ESA and was an input for the CCSDS Panel 1A meeting in July 1994.

C1 INTRODUCTION

The generator polynomial for the randomizer specified in reference [1] for TC channel coding is the one recommended in the conclusion (C4) of this annex. Because a Reed-Solomon (R-S) code may be chosen in the future for TC channel coding, it was decided to choose a randomizer sequence that avoids interaction with the R-S codes.

The problem addressed in this annex comes from the discovery that the randomizer sequence specified for TM channel coding in reference [5] is also a codeword of the standard (255,223) Reed-Solomon code in reference [5]. In C2, the reasons for this are explained. Since this is an unfortunate choice of the randomizer sequence, C3 describes a search for randomizer sequences that:

- are not codewords in the standard R-S (255,223) code in reference [5] at any interleaving depth;
- are not codewords in the R-S (255,239) code in reference [8] at any interleaving depth;
- have large distances from the two synchronization markers in reference [5] and their inverses.

In C4, a randomizer sequence fulfilling all these requirements is given. The proposed sequence has the property that it is marked as not decodable by a proper decoder, for example, the decoder in reference [8], used at any interleaving depth in the range one to eight.

C2 THEORY OF RANDOM SEQUENCES AND CODES

The theory of this section is mostly found in reference [10] and is applied here to the case of binary pseudo-random sequences as used in reference [5]. A pseudo-random binary sequence:

$$x_0x_1x_2\ldots.x_t\ldots.$$

is generated by the linear recursion:

$$x_t = r_{m-1}x_{t-1} + r_{m-2}x_{t-2} + \ldots\ r_1x_{t-m+1} + r_0x_{t-m},$$

where r_j are binary coefficients. The properties of the sequence depend on the characteristic polynomial (or randomizer polynomial):

$$R(z) = z^m + r_{m-1}z^{m-1} + \ldots.+ r_1z + r_0.$$

The sequence has period 2^m-1 if and only if R(z) has degree m and is irreducible and primitive when calculations are done in the binary field GF(2). The sequences generated by such a polynomial of degree m all have the same distribution of runs of '0's and '1's. The run distribution for sequences of period 2^m-1 is described in table C-1.

Table C-1: Run Distribution for Sequences of Period 2^m-1

Length	0-runs	1-runs
1	2^{m-3}	2^{m-3}
2	2^{m-4}	2^{m-4}
r	2^{m-r-2}	2^{m-r-2}
m-2	1	1
m-1	1	0
m	0	1

A consequence of the characteristic 2 of this field is that the recursion with the characteristic polynomial:

$$R_p(z) = z^{m2^p} + r_{m-1}z^{(m-1)2^p} + \ldots.+ r_1z^{2^p} + r_0.$$

is also fulfilled by the sequence $x_0x_1\ldots..$ If the sequence is divided into groups of 2^p bits, and these groups are considered as elements of the field GF(2^p) represented in some basis, usage of $R_p(z)$ on the corresponding bits in each group gives a recursion on the elements in GF(2^p):

$$R(z) = z^m + r_{m-1}z^{m-1} + \ldots.+ r_1z + r_0.$$

Calculations are now done in GF(2^p), and the sequence has the period 2^m-1 if and only if R(z) is irreducible over GF(2) and is a primitive polynomial. If the sequence is interpreted as a polynomial:

$$X(z) = x_0 + x_1 z + x_2 z^2 + ... + x_{2^{m-2}} z^{2^{m-2}},$$

the implication of the recursion is that R(z) is a parity check polynomial for a cyclic code containing X(z). The length of this code is 2^m-1, and the generator polynomial is $(z^{2^{m-1}}-1)/R(z)$. When X(z) is evaluated at a root of the generator polynomial $z = \sigma$, it gives $X(\sigma) = 0$ (a syndrome for the code). The roots of the generator polynomial are all roots of $z^{2^{m-1}}-1$ that are not roots in R(z).

The relations between a Reed-Solomon code over GF(2^p) with generator G(z) and length 2^p-1 and random sequences X(z) with period 2^p-1 can now be considered. If the roots of G(z) are denoted σ_{RS}, then $X(\sigma_{RS}) = 0$ if and only if all these roots σ_{RS} are distinct from the roots of R(z), since in that case, all roots of G(z) are roots of the generator polynomial for the code that X(z) belongs to.

Example: Among the 16 different polynomials with degree 8 that are irreducible and primitive is $z^8 + z^7 + z^5 + z^3 + 1$, which is used in reference [5] as randomizer polynomial for TM channel coding. The field GF(2^8) is defined by the polynomial $z^8 + z^6 + z^4 + z^3 + z^2 + z + 1$, and b is the primitive root of this polynomial. The roots of R(z) are b^{37}, b^{74}, b^{148}, b^{41}, b^{82}, b^{164}, b^{73} and b^{146}, and they are all outside the roots of the R-S (255,223) code: b^{112}, ... b^{143}. Thus the random sequence is a codeword in the R-S code.

If c, c^2, c^4, c^8, ... are the roots of R(z), the sequence is a codeword of the R-S code that is interleaved with interleaving depth *I*, if and only if c^I, c^{2I}, c^{4I}, c^{8I}, ... are not roots of the generator polynomial for the R-S code. Since the characteristic of the field is two, the properties for the sequence are identical for all interleaving depths $I=2^p$, that is, 1, 2, 4, 8, 16. The other values of *I* must be checked separately.

C3 SEARCH FOR RANDOMIZER POLYNOMIALS

The search described in this section aims at fulfilling the following properties for the pseudo-random sequence with period 255 generated by the randomizer polynomial R(z) of degree 8:

- The sequence is not a codeword in the standard (255,223) code at any interleaving depth, as explained in C2 above. The distance to codewords should be large and preferably should be greater than 16.
- The sequence is not a codeword (at any interleaving depth) in the (255,239) code defined in reference [8], where the generator polynomial has roots b^{120}, b^{113}, ... b^{135}, and b is defined in the example in C2 above.
- The sequence has a large minimum distance from the two synchronization markers, Syncl: 1A CF FC 1D (hex), Sync2: 35 2E F8 53 (hex), and their inverses.

The results for all 16 possible randomizer polynomials giving period 255 are shown in table C-2.

Table C-2: Results for the 16 Randomizer Polynomials

R(z) (hex)	Interleaving depths for which the sequence is a codeword		Minimum distance to			
	in code (255,223)	in code (255,239)	Sync1	Sync1 inverse	Sync2	Sync2 inverse
171	None	3,5,6,7	9	7	7	7
1A9	1,2,4,8	1,2,4,5,8	8	9	8	8
169	None	1,2,3,4,6,8	8	6	7	9
165	7	5,7	7	9	6	9
1F5	None	None	7	8	9	8
18D	7	3,6,7	8	5	9	6
1C3	None	1,2,3,4,5,6,8	9	8	9	8
1E7	1,2,4,8	1,2,4,7,8	8	8	8	8
11D	None	3,5,6,7	9	8	8	9
12B	1,2,4,8	1,2,4,5,8	8	8	6	10
12D	None	1,2,3,4,6,8	8	8	9	9
14D	7	5,7	9	9	9	8
15F	None	None	8	8	8	9
163	7	3,6,7	8	8	9	7
187	None	1,2,3,4,5,6,8	8	9	9	8
1CF	1,2,4,5	1,2,4,7,8	8	8	10	7

The polynomial 1A9 (hex) is the one used in reference [5] as a randomizer polynomial for TM channel coding. The table shows that 15F (hex) is the better choice of the two polynomials that do not generate codewords in any of the two codes at any interleaving depth, since it has the best distance properties with respect to the synchronization markers. The sequence generated has distances to codewords that are so large it is not decoded at any interleaving depth in the range 1 to 8.

C4 CONCLUSION

The randomizer polynomial should be selected as:

$$z^8 + z^6 + z^4 + z^3 + z^2 + z + 1,$$

which is identical to the polynomial used for generating the field.

The first 40 binary symbols generated by this polynomial are:

1111 1111 0011 1001 1001 1110 0101 1010 0110 1000,

or in hexadecimal notation,

FF 39 9E 5A 68.

The sequence generated has a minimum distance to the synchronization markers and their inverses of at least 8. The sequence thus generated gives non-zero syndromes for both R-S codes for all interleaving depths in the range 1 to 8, and the sequence is not decoded by a proper working decoder at any of these interleaving depths.

ANNEX D

THEORETICAL BACKGROUND OF THE CLTU TAIL SEQUENCE

NOTE – This annex contains an edited version of *Uncorrectable Sequences and Telecommand* by Ekroot, McEliece, Dolinar, and Swanson, May 15, 1993. The original report was an input for the CCSDS Panel 1A meeting in September 1993 and contributed to the selection of the current CLTU Tail Sequence described in 5.5.1. At the time the report was written, the CCSDS-recommended CLTU Tail Sequence was the one described in 5.5.3, and it had multiple lengths as described in 5.5.2.

D1 ABSTRACT

The purpose of a tail sequence for command link transmission units is to fail to decode, so that the command decoder will begin searching for the start of the next unit. A tail sequence used by several missions and recommended for this purpose by CCSDS is analyzed. A single channel error can cause the sequence to decode. An alternative sequence, requiring at least two channel errors before it can possibly decode, is presented. (No sequence requiring more than two channel errors before it can possibly decode exists for this code.)

D2 INTRODUCTION

When a CLTU consisting of many codewords is received by a spacecraft, the command decoder verifies either that each codeword is a valid codeword and accepts it, or that it is a slightly corrupted codeword and corrects it, or that it is too far from a valid codeword and rejects it. Rejecting a codeword causes the receiver to give up on the unit and begin searching for the start of the next unit. At the end of the CLTU, there is a tail sequence designed to be rejected as a codeword, sending the decoder into a 'search mode'. This report analyzes the performance of the tail sequence recommended by CCSDS and used by several missions. So instead of the usual question about a code, that is, how many errors can the code correct or detect, the question here is how many errors can occur before an uncorrectable sequence becomes correctable.

D3 ANALYSIS OF UNCORRECTABLE SEQUENCES

D3.1 GENERAL

In order for a sequence to be uncorrectable, it must be far enough from a codeword to cause the decoder not to decode. At the very least, it must differ from the nearest codeword in more positions than the decoder is able to correct. However, channel errors can make such a sequence decodable. The more errors that must occur before the sequence becomes correctable, the less likely it is that the sequence will accidentally decode. In order to

maximize the number of channel errors before the sequence will decode, it is necessary to characterize and find sequences that are as far away from codewords as possible.

A code is designed to have codewords that are maximally far away from each other. If, as a simple example, a new code is created by using a subset of the codewords, then the unused codewords are still far away from the codewords of the new code. Intuitively, these unused codewords are candidates for uncorrectable sequences.

The codes discussed in this article include the perfect (63,57) Hamming code, the (63,56) expurgated Hamming code, and shortened versions of the expurgated code. The codewords for the shortened codes correspond to subsets of the codewords of the (63,56) code, which are themselves the even codewords from the (63,57) code. By making use of the larger code's properties and keeping track of what happens as the codeword sets get smaller, sequences that are uncorrectable in each of the smaller codes can be found.

D3.2 THE PERFECT (63,57) HAMMING CODE

The generator polynomial for the (63,57) code is the sixth-degree primitive polynomial:

$$g_p(x) = x^6 + x + 1. \tag{8}$$

The (63,57) code is perfect and has minimum distance 3; that is, every binary sequence of length 63 either is a codeword or is Hamming distance one away from exactly one codeword. For a perfect code, there are no holes left when the space of binary sequences is filled with spheres of radius one centered at the codewords.

The even-weight words are a promising subset to use for a new code, leaving the odd-weight words as candidates for uncorrectable sequences. In fact, the (63,56) code described in the next section uses the even-weight words.

D3.3 THE (63,56) EXPURGATED HAMMING CODE

The generator polynomial for the (63,56) code is given by:

$$g(x) = x^7 + x^6 + x^2 + 1 = (x + 1)(x^6 + x + 1). \tag{9}$$

Since the generator polynomial is the product of $x + 1$ and the generator polynomial $g_p(x)$ of the perfect (63,57) code, the (63,56) code consists of only the even-weight codewords from the perfect (63,57) code. This code has minimum distance four and can correct at most one error.

It should be noted that any odd-weight codeword in the (63,57) code is exactly Hamming distance three away from the nearest even-weight codewords, and the even-weight codewords are the codewords of the (63,56) code. Any odd-weight binary sequence of length 63 differs from a nearest codeword by either three bits or one bit. Similarly, any even-weight sequence either is a codeword or is Hamming distance two from the nearest codewords.

The following example illustrates how the concepts of distance relate to a sequence that is not a codeword in the perfect (63,57) code. The sequence selected for the example plays a role in the CCSDS Recommended Standards for telecommand; the description of that role is deferred to annex subsection D4.

Example 1: The length 63 sequence:

01010101010101010101010101010101010101010101010101010101010101 1010101

has even weight, and is therefore either a codeword in the (63,56) code or two away from a codeword. The parts of the two-part *syndrome* $(s_1(\mathbf{r}),s_2(\mathbf{r}))$ of a sequence $\mathbf{r} = r_{N-1}r_{N-2}\ldots r_1r_0$ are given by:

$$s_1(\mathbf{r}) = \sum_{i=0}^{N-1} r_i\alpha^j \pmod{\alpha^6 + \alpha + 1} \tag{10}$$

$$s_2(\mathbf{r}) = \sum_{i=0}^{N-1} r_i\alpha^j \pmod{\alpha + 1}\,, \tag{11}$$

where α is a root of the generator polynomial $g(x)$, and N is the length of the codewords. The two-part syndrome tells either if the sequence is a codeword, how to correct it if it is distance one from a codeword, or that it is not near enough to a single codeword. Specifically, if the syndrome is (0,0), then the sequence is a codeword; if the syndrome is (0,1), then the sequence has odd weight and differs from the nearest codeword in three positions; if the syndrome is $(\alpha^j,1)$, then the sequence has odd weight and differs from the nearest codeword in the jth position; if the syndrome is $(\alpha^j,0)$, then the weight is even, and the sequence differs from the nearest codewords in two positions.

For this example sequence, the first part of the syndrome (modulo $\alpha^6 + \alpha + 1$) is:

$$s_1(\mathbf{r}) = \sum_{i=0}^{N-1} r_i\alpha^j \pmod{\alpha^6 + \alpha + 1} \tag{12}$$

$$= 1 + \alpha^2 + \alpha^4 + \alpha^6 + \alpha^7 + \alpha^9 + \alpha^{11} + \ldots + \alpha^{61} \tag{13}$$

$$= 1 + \alpha^2 + \alpha^4 + (\alpha + 1) + \frac{\alpha^7 + \alpha^{63}}{1 + \alpha^2} \tag{14}$$

$$= \alpha + \alpha^2 + \alpha^4 + (\alpha + 1) + \frac{\alpha\alpha^6 + 1}{1 + \alpha^2} \tag{15}$$

$$= \frac{(\alpha + \alpha^2 + \alpha^4)(1 + \alpha^2)}{1 + \alpha^2} + \frac{\alpha(\alpha + 1) + 1}{1 + \alpha^2} \tag{16}$$

$$= \frac{\alpha + \alpha^2 + \alpha^4 + \alpha^3 + \alpha^4 + \alpha^6 + \alpha^2 + \alpha + 1}{1 + \alpha^2} \tag{17}$$

$$= \frac{\alpha^3 + \alpha^6 + 1}{1 + \alpha^2} \tag{18}$$

$$= \frac{\alpha^3 + (1 + \alpha) + 1}{1 + \alpha^2} \tag{19}$$

$$= \frac{\alpha^3 + \alpha}{1 + \alpha^2} \tag{20}$$

$$= \alpha, \tag{21}$$

where equations 14, 16, and 19 follow from equivalence modulo $\alpha^6 + \alpha + 1$. The second part of the syndrome (modulo $\alpha + 1$) is:

$$s_2(\mathbf{r}) = \sum_{i=0}^{N-1} r_i \alpha^j \pmod{\alpha + 1} \tag{22}$$

$$= \text{weight } (\mathbf{r}) \pmod 2 \tag{23}$$

$$= 32 \pmod 2 \qquad = 0. \tag{24}$$

The syndrome (α,0) indicates that the sequence is not a codeword in the (63,56) code, and is two away from the nearest codewords. A single error in any bit except r_1 will make the sequence differ from a codeword by one and be thus decodable. (If the error is in r_1, the syndrome becomes (0,1), indicating that it is three away from a codeword.)

If instead a sequence is considered that is an odd-weight word in the perfect (63,57) code, it is three away from the codewords of the (63,56) code. This means that two errors must occur before it becomes decodable by a single-error-correcting (63,56) decoder. Such a sequence would be a better choice for a tail sequence because it is more resistant to accidental decoding in the presence of errors.

In selecting a particular sequence for the command coding application, the effects on the distance and uncorrectability properties as the code is shortened must be taken into account.

D3.4 SHORTENING THE (63,56) CODE

Select a subset of a code, where all the codewords in the subset have zeros in some specified positions. Since all of the codewords in the subset have zeros in the specified positions, those positions carry no information and can be ignored. The resulting set of codewords forms a *shortened code*. Shortening cannot decrease the minimum distance, and will only increase the minimum distance if the code is shortened severely. For the (63,56) code, it can be shown that as long as the shortened length is greater than 32, the minimum distance will remain 4, and the maximum distance of any sequence to the nearest codewords will remain 3.

For this application, shortening will be done by taking only the codewords that have zeros in the leftmost or first *m* positions. It should be noted that for each shortened word, there is a corresponding full-length word that has zeros in the first *m* positions.

Example 2: The sequence of length 55:

0101010101010101010101010101010101010101010101010101 1010101

corresponds to the full-length sequence:

0000000001010101010101010101010101010101010101010101 1010101

with weight 28.

The first part of the syndrome for the full-length sequence is:

$$s_1(\mathbf{r}) = \sum_{i=0}^{N-1} r_i \alpha^j \pmod{\alpha^6 + \alpha + 1} \tag{25}$$

$$= 1 + \alpha^2 + \alpha^4 + \alpha^6 + \alpha^7 + \alpha^9 + \alpha^{11} + \ldots + \alpha^{53} \tag{26}$$

$$= 1 + \alpha^2 + \alpha^4 + (1 + \alpha) + (\alpha^7 + \alpha^9 + \alpha^{11} + \ldots + \alpha^{53}) \tag{27}$$

$$= \alpha + \alpha^2 + \alpha^4 + \frac{\alpha^7 + \alpha^{55}}{1 + \alpha^2} \tag{28}$$

$$= \alpha + \alpha^2 + \alpha^4 + \frac{\alpha(\alpha + 1) + \alpha(\alpha + 1)^9}{(1 + \alpha)^2} \tag{29}$$

$$= \alpha + \alpha^2 + \alpha^4 + \frac{\alpha(1 + (\alpha + 1)^8)}{1 + \alpha} \tag{30}$$

$$= \alpha + \alpha^2 + \alpha^4 + \frac{\alpha(1 + \alpha^8 + 1)}{1 + \alpha} \tag{31}$$

$$= \alpha + \alpha^2 + \alpha^4 + \frac{\alpha\alpha^2(\alpha + 1)}{1 + \alpha} \tag{32}$$

$$= \alpha + \alpha^2 + \alpha^4 + \alpha^3 \tag{33}$$

$$= \alpha\,(\alpha + 1)^3 \tag{34}$$

$$= \alpha^{19}. \tag{35}$$

The second part of the syndrome is zero since the weight is even. The two-part syndrome $(\alpha^{19},0)$ indicates that the sequence is not a codeword, and that it is two away from the nearest codewords. A single error can make the sequence differ from a codeword by one and be thus decodable.

D3.5 FINDING A GOOD UNCORRECTABLE SEQUENCE

The concepts in D3.3 and D3.4 lead to the definition of a *good uncorrectable sequence* as one for which it and all the desired truncations of it are maximally distant from the codewords in the corresponding code, that is, an odd-weight codeword in the perfect (63,57) code. The syndrome for good uncorrectable sequences is (0,1). The analysis below shows that an uncorrectable sequence can be chosen so that, when it is truncated by octets, its syndrome does not change, and therefore it does not become correctable.

If a given sequence is truncated by m bits, and if it is desired that the syndrome not be changed by the truncation, then the smallest possible m is eight. This is because in order not to change either part of the syndrome, the truncated bits must correspond to a polynomial that is zero modulo both $\alpha + 1$ and $\alpha^6 + \alpha + 1$. The lowest order nonzero polynomial satisfying that requirement is $g(\alpha) = \alpha^7 + \alpha^6 + \alpha^2 + 1$. Therefore the shortest such nonzero sequence is 11000101, which has length eight.

There are engineering reasons for truncating by octets in the application considered in annex subsection D4. Also, bit synchronization requirements often make it preferable to have many transitions in the sequence, and thus a mostly zeros sequence is undesirable. For the remainder of the report, it is assumed that shortening will be done only by multiples of eight bits, and that octets of zeros are not of interest.

Since the second part of the syndrome must be 1 for the sequence and all of its truncations, and the truncated bits 11000101 have even weight, the nontruncated part of the sequence must have odd weight. Since the first part of the syndrome must be 0 for the sequence and all of its truncations, the nontruncated part of the sequence must correspond to a polynomial that is zero modulo $\alpha^6 + \alpha + 1$. The shortest such sequence is 1000011.

A simple construction of a good uncorrectable sequence is a concatenation of octets of the form 11000101 with the seven bits 1000011 in the rightmost positions. This is not the only

good uncorrectable sequence, but it does have good distance and bit synchronization properties. The syndrome for this sequence is confirmed in the next example.

Example 3: The sequence:

11000101 11000101 11000101 11000101 11000101 11000101 11000101 1000011

has weight 31, so the second part of the syndrome $s_2(\mathbf{r})$ is 1. The first part of the syndrome (modulo $\alpha^6 + \alpha + 1$) for the full-length sequence is:

$$s_1(\mathbf{r}) = \sum_{i=0}^{N-1} r_i \alpha^j \pmod{\alpha^6 + \alpha + 1} \tag{36}$$

$$= 1 + \alpha^2 + \alpha^6 + \sum_{k=0}^{6} \alpha^{7+8k} (1 + \alpha^2 + \alpha^6 + \alpha^7) \tag{37}$$

$$= 1 + \alpha^2 + \alpha^6 + \sum_{k=0}^{6} \alpha^{7+8k} (1 + \alpha + \alpha^6)(1 + \alpha) \tag{38}$$

$$= 0. \tag{39}$$

Thus the sequence is three away from the nearest codewords. A single error cannot make this sequence decodable.

D4 COMMAND LINK CODING

D4.1 GENERAL

The CCSDS Recommended Standard uses a tail sequence that is specially constructed not to decode. By not decoding, it causes the receiver to begin searching for the next CLTU. It will be shown that the sequence that has been recommended is not the best sequence in terms of distance, and a better one will be given.

In order to apply the results of annex subsection D3 to the command coding problem, the operations of the CLTU must be detailed.

D4.2 BCH CODEWORD

The BCH codeword has K information bits, 7 *inverted* parity check bits, and a fill bit for a total length of L. Because of the code selected and the desire to shorten in units of 8 bits, the number of information bits and block lengths considered are K = 32, 40, 48, and 56, and L = 40, 48, 56, and 64, respectively.

D4.3 COMMAND LINK TRANSMISSION UNIT

The CLTU consists of

- a 16-bit start sequence, namely 1110101110010000;
- a number of BCH codewords (the information may be padded with fill to make the information into a multiple of K bits);
- a tail sequence that is a sequence of bits the same length as a codeword and designed to be uncorrectable. The idea is to cause the receiver to stop decoding and begin looking for the start sequence of the next CLTU.

D4.4 TAIL SEQUENCES

The CCSDS Recommended Standard specifies that the tail sequence **t** be a sequence of alternating zeros and ones beginning with a zero. Ignoring the fill bit, and noting that the parity bits are inverted, it can be seen that for a block length L of 64, this corresponds to the sequence in Example 1. As illustrated in the example, this sequence has distance properties such that a single channel error can, and almost certainly will, make the sequence decodable.

For the shorter block lengths L = 40, 48, and 56, the sequences of alternating zeros and ones beginning with a zero have corresponding full-length sequences with zeros filled in the first positions. All three of these corresponding sequences have even weights and nonzero syndromes. The calculation for L = 56 is done in Example 2.

If instead, the sequence:

11000101 11000101 11000101 11000101 11000101 11000101 11000101 0111100 0

is used as the tail sequence, it corresponds (when the fill bit is removed and the parity bits are inverted) to the sequence in Example 3. If one channel error occurs, then this sequence will still be uncorrectable.

D4.5 AUGMENTATION USING THE FILL BIT

The CCSDS Recommended Standard proposes to use the fill bit as a flag to tell the decoder to operate in error-detect mode only. This improves the probability of spotting the tail sequence by not allowing the decoder to correct any errors when the fill bit is 1. In this mode, two errors are sufficient to make the CCSDS tail sequence (of alternating zeros and ones) decodable, while a minimum of three errors is required to make the sequence presented here decodable.

D5 CONCLUSIONS

The analysis in this report shows that there are uncorrectable sequences that can tolerate one more channel error than the CCSDS tail sequence before becoming decodable. It is also shown that this property may be preserved for the shortened as well as the full-length codes recommended by CCSDS. A sequence satisfying these requirements should be considered for the role of the CCSDS tail sequence since it is more resistant to channel errors than the proposed tail sequence, and still has many transitions to aid bit synchronization.

ANNEX E

PERFORMANCE OF OBSOLETE FEATURES

E1 INTRODUCTION

This annex contains performance data for obsolete features of TC channel coding. The features were included in earlier versions of CCSDS Recommendations or were covered in CCSDS Reports. These features are not included in the current issue of the Recommended Standard (reference [1]), but their performance data are preserved here for comparative and historical reference. The features covered by this annex are:

– shorter lengths for BCH codeword (see 3.3.2);

– earlier pattern for Tail Sequence (see 5.5.3);

– use of two Tail Sequences with the earlier pattern;

– Filler Bit Augmentation algorithm (see 8.5.3.3).

E2 CODEWORD REJECTION

In TED mode, the probability P_{CX} of one of the BCH codewords in a CLTU causing a codeword rejection is:

$$P_{CX} = 1 - [(1 - p)^n]^N,$$

where n is one less than the codeword length, and N is the number of codewords in the CLTU. For a given frame length, the number of codewords required to hold the frame increases as the codeword size decreases. Tables E-1 to E-4 show the values of the probability P_{CX} for the three channel bit error rates for different frame lengths.

Table E-1: Probability P_{CX} of Codeword Rejection in TED Mode, Frame Length 20 Octets

Codeword Length	Number of Codewords *N*	Channel Bit Error Rate		
		10^{-4}	10^{-5}	10^{-6}
64	**3**	1.87×10^{-2}	1.89×10^{-3}	1.89×10^{-4}
56	**4**	2.18×10^{-2}	2.20×10^{-3}	2.20×10^{-4}
48	**4**	1.86×10^{-2}	1.88×10^{-3}	1.88×10^{-4}
40	**5**	1.93×10^{-2}	1.95×10^{-3}	1.95×10^{-4}

Table E-2: Probability P_{CX} of Codeword Rejection in TED Mode, Frame Length 50 Octets

Codeword Length	Number of Codewords *N*	Channel Bit Error Rate		
		10^{-4}	10^{-5}	10^{-6}
64	**8**	4.92×10^{-2}	5.03×10^{-3}	5.04×10^{-4}
56	**9**	4.83×10^{-2}	4.94×10^{-3}	4.95×10^{-4}
48	**10**	4.59×10^{-2}	4.69×10^{-3}	4.70×10^{-4}
40	**13**	4.94×10^{-2}	5.06×10^{-3}	5.07×10^{-4}

Table E-3: Probability P_{CX} of Codeword Rejection in TED Mode, Frame Length 120 Octets

Codeword Length	Number of Codewords *N*	Channel Bit Error Rate		
		10^{-4}	10^{-5}	10^{-6}
64	**18**	1.07×10^{-1}	1.13×10^{-2}	1.13×10^{-3}
56	**20**	1.04×10^{-1}	1.09×10^{-2}	1.10×10^{-3}
48	**24**	1.07×10^{-1}	1.12×10^{-2}	1.13×10^{-3}
40	**30**	1.10×10^{-1}	1.16×10^{-2}	1.17×10^{-3}

Table E-4: Probability P_{CX} of Codeword Rejection in TED Mode, Frame Length 256 Octets

Codeword length	Number of Codewords *N*	Channel Bit Error Rate		
		10^{-4}	10^{-5}	10^{-6}
64	**37**	2.08×10^{-1}	2.30×10^{-2}	2.33×10^{-3}
56	**43**	2.11×10^{-1}	2.34×10^{-2}	2.36×10^{-3}
48	**52**	2.17×10^{-1}	2.41×10^{-2}	2.44×10^{-3}
40	**64**	2.21×10^{-1}	2.47×10^{-2}	2.49×10^{-3}

In SEC mode, the probability P_{CY} of one of the BCH codewords in a CLTU causing a codeword rejection is:

$$P_{CY} = 1 - [(1-p)^n + np(1-p)^{(n-1)}]^N,$$

where n is one less than the codeword length, and N is the number of codewords in the CLTU. Tables E-5 to E-8 show the values of the probability P_{CY} for the three channel bit error rates for different frame lengths.

Table E-5: Probability P_{CY} of Codeword Rejection in SEC Mode, Frame Length 20 Octets

Codeword Length	Number of Codewords *N*	Channel Bit Error Rate		
		10^{-4}	10^{-5}	10^{-6}
64	**3**	5.84×10^{-5}	5.86×10^{-7}	5.86×10^{-9}
56	**4**	5.92×10^{-5}	5.94×10^{-7}	5.94×10^{-9}
48	**4**	4.31×10^{-5}	4.32×10^{-7}	4.32×10^{-9}
40	**5**	3.70×10^{-5}	3.70×10^{-7}	3.70×10^{-9}

Table E-6: Probability P_{CY} of Codeword Rejection in SEC Mode, Frame Length 50 Octets

Codeword Length	Number of Codewords *N*	Channel Bit Error Rate		
		10^{-4}	10^{-5}	10^{-6}
64	**8**	1.56×10^{-4}	1.56×10^{-6}	1.56×10^{-8}
56	**9**	1.33×10^{-4}	1.34×10^{-6}	1.34×10^{-8}
48	**10**	1.08×10^{-4}	1.08×10^{-6}	1.08×10^{-8}
40	**13**	9.61×10^{-5}	9.63×10^{-7}	9.63×10^{-9}

Table E-7: Probability P_{CY} of Codeword Rejection in SEC Mode, Frame Length 120 Octets

Codeword Length	Number of Codewords *N*	Channel Bit Error Rate		
		10^{-4}	10^{-5}	10^{-6}
64	**18**	3.50×10^{-4}	3.51×10^{-6}	3.52×10^{-8}
56	**20**	2.96×10^{-4}	2.97×10^{-6}	2.97×10^{-8}
48	**24**	2.59×10^{-4}	2.59×10^{-6}	2.59×10^{-8}
40	**30**	2.22×10^{-4}	2.22×10^{-6}	2.22×10^{-8}

Table E-8: Probability P_{CY} of Codeword Rejection in SEC Mode, Frame Length 256 Octets

Codeword Length	Number of Codewords *N*	Channel Bit Error Rate		
		10^{-4}	10^{-5}	10^{-6}
64	**37**	7.19×10^{-4}	7.22×10^{-6}	7.23×10^{-8}
56	**43**	6.36×10^{-4}	6.38×10^{-6}	6.39×10^{-8}
48	**52**	5.60×10^{-4}	5.62×10^{-6}	5.62×10^{-8}
40	**64**	4.73×10^{-4}	4.74×10^{-6}	4.74×10^{-8}

E3 TAIL SEQUENCE FACTOR

When operating with PLOP-2, the Tail Sequence factor is relevant to the performance. As described in 11.4.5, if the Tail Sequence is not recognized, then the following CLTU may be missed.

Earlier CCSDS Recommendations had a different pattern for the CLTU Tail Sequence (see 5.5.3). With the earlier pattern, a single error could cause the Tail Sequence to be missed in SEC mode, and two errors could cause it to be missed in TED mode.

Table E-9 shows the probabilities of missing the Tail Sequence for the earlier Tail Sequence pattern. The probability P_{TX} is for decoding in TED mode and P_{TY} is for decoding in SEC mode. The table shows the probabilities for different Tail Sequence lengths because the length of the Tail Sequence is the same as the length of a BCH codeword.

The values in the table are based on the probabilities of one or two errors in the Tail Sequence. However, errors in some positions do not cause the Tail Sequence to be missed. For example, only 31 of the 1953 possible double errors in a 63-bit codeword cause the Tail Sequence to become a valid codeword. These factors have been taken into account in the table.

Table E-9: Probabilities P_{TX} and P_{TY} of Missing a Tail Sequence with the Earlier Pattern

Tail Sequence Length (bits)	P_{TX} P_{TY}	Channel Bit Error Rate		
		10^{-4}	10^{-5}	10^{-6}
64	P_{TX} (TED)	3.08×10^{-7}	3.10×10^{-9}	3.10×10^{-11}
	P_{TY} (SEC)	6.16×10^{-3}	6.20×10^{-4}	6.20×10^{-5}
56	P_{TX} (TED)	2.39×10^{-7}	2.40×10^{-9}	2.40×10^{-11}
	P_{TY} (SEC)	4.77×10^{-3}	4.80×10^{-4}	4.80×10^{-5}
48	P_{TX} (TED)	1.69×10^{-7}	1.70×10^{-9}	1.70×10^{-11}
	P_{TY} (SEC)	3.38×10^{-3}	3.40×10^{-4}	3.40×10^{-5}
40	P_{TX} (TED)	1.20×10^{-7}	1.20×10^{-9}	1.20×10^{-11}
	P_{TY} (SEC)	2.39×10^{-3}	2.40×10^{-4}	2.40×10^{-5}

Various techniques were introduced to reduce the risk of missing a CLTU because of failing to recognize the Tail Sequence. One of these involved inserting an extra Tail Sequence at the end of each CLTU so that each CLTU had a double Tail Sequence.

If the first Tail Sequence was not recognized, there was a high probability that the second Tail Sequence would be recognized, and so the next CLTU would not be missed. Table E-10 shows the probabilities of missing both Tail Sequences (earlier pattern) when the double Tail Sequence technique is used.

Table E-10: Probabilities P_{TXD} and P_{TYD} of Missing Both Tail Sequences (Earlier Pattern) When a Double Tail Sequence Is Used

Tail Sequence Length (bits)	P_{TXD} P_{TYD}	Channel Bit Error Rate		
		10^{-4}	10^{-5}	10^{-6}
64	P_{TXD} (TED)	9.5×10^{-14}	9.6×10^{-18}	9.6×10^{-22}
	P_{TYD} (SEC)	3.8×10^{-5}	3.8×10^{-7}	3.8×10^{-9}
56	P_{TXD} (TED)	5.7×10^{-14}	5.8×10^{-18}	5.8×10^{-22}
	P_{TYD} (SEC)	2.3×10^{-5}	2.3×10^{-7}	2.3×10^{-9}
48	P_{TXD} (TED)	2.9×10^{-14}	2.9×10^{-18}	2.9×10^{-22}
	P_{TYD} (SEC)	1.1×10^{-5}	1.2×10^{-7}	1.2×10^{-9}
40	P_{TXD} (TED)	1.4×10^{-14}	1.4×10^{-18}	1.4×10^{-22}
	P_{TYD} (SEC)	5.7×10^{-6}	5.8×10^{-8}	5.8×10^{-10}

The Filler Bit Augmentation (FBA) algorithm described in 8.5.3.3 was another technique to reduce the risk of failing to recognize the Tail Sequence with the earlier pattern. The algorithm applies only to SEC mode. Table E-11 shows the probability P_{TYF} of missing the Tail Sequence when the algorithm is used.

Table E-11: Probability P_{TYF} of Missing a Tail Sequence (Earlier Pattern) When Filler Bit Augmentation Is Used in SEC Mode

Tail Sequence Length (bits)	Channel Bit Error Rate		
	10^{-4}	10^{-5}	10^{-6}
64	9.24×10^{-7}	9.29×10^{-9}	9.30×10^{-11}
56	7.16×10^{-7}	7.20×10^{-9}	7.20×10^{-11}
48	5.08×10^{-7}	5.10×10^{-9}	5.10×10^{-11}
40	3.59×10^{-7}	3.60×10^{-9}	3.60×10^{-11}

For comparison, table E-12 shows the probabilities of missing the current Tail Sequence (see 5.5.1) for all lengths in TED mode, in SEC mode, and in SEC mode with FBA.

Table E-12: Probabilities of Missing a Tail Sequence (Current Pattern) for Different Modes and Lengths

Tail Sequence Length (bits)	Mode	Channel Bit Error Rate		
		10^{-4}	10^{-5}	10^{-6}
64	TED	6.47×10^{-10}	6.51×10^{-13}	6.51×10^{-16}
	SEC	1.94×10^{-5}	1.95×10^{-7}	1.95×10^{-9}
	SEC + FBA	2.59×10^{-9}	2.60×10^{-12}	2.60×10^{-15}
56	TED	4.27×10^{-10}	4.29×10^{-13}	4.29×10^{-16}
	SEC	1.28×10^{-5}	1.29×10^{-7}	1.29×10^{-9}
	SEC + FBA	1.71×10^{-9}	1.72×10^{-12}	1.72×10^{-15}
48	TED	2.60×10^{-10}	2.61×10^{-13}	2.61×10^{-16}
	SEC	7.80×10^{-6}	7.83×10^{-8}	7.83×10^{-10}
	SEC + FBA	1.04×10^{-9}	1.04×10^{-12}	1.04×10^{-15}
40	TED	1.51×10^{-10}	1.52×10^{-13}	1.52×10^{-16}
	SEC	4.54×10^{-6}	4.56×10^{-8}	4.56×10^{-10}
	SEC + FBA	6.06×10^{-10}	6.08×10^{-13}	6.08×10^{-16}

ANNEX F

PRACTICAL EXAMPLES OF FRAMES AND CLTUS

F1 BCH CODED FRAMES AND CLTUS

This annex contains a set of practical examples of TC Transfer Frames and CLTUs.

The example TC Transfer Frames show the construction of the frames, including the calculated Frame Error Control Fields that contain the CRC for error detection. The TC Transfer Frame is specified in reference [3].

The example CLTUs show the CLTU components. The Error Control Field can be seen in the last octet of each BCH codeword. The CLTUs are shown with and without randomization. As described in 9.2.1, the randomization of any fill bits that may be present in the last BCH codeword of a CLTU is optional. In the CLTUs in this annex, the fill bits are not randomized.

The TC Transfer Frames and CLTUs are shown in hexadecimal notation. For the TC Transfer Frame and CLTU, the left-most octet is transmitted first, and the most significant bit of each octet is the first bit of the octet to be transmitted.

NOTE – The Acquisition Sequences and Idle Sequences before, between, and after the CLTUs are not shown in the examples here. These sequences depend on the Physical Layer Operations Procedure in use, as described in section 7.

Example 1: Type BC Frame with Control Command 'Unlock'

Spacecraft Identifier	1B
Virtual Channel Identifier	0
Frame Type	BC (shows that the frame carries a control command)
Frame Sequence Number	0 (all type-B frames have sequence number 0)
Frame Data Field length	1 octet
Frame Data Field content	00 (control command 'Unlock')
Frame length	8 octets
Frame Error Control Field	4C A9
TC Transfer Frame	30 1B 00 07 00 00 4C A9
CLTU (unrandomized)	
Start Sequence	EB 90
1st BCH codeword	30 1B 00 07 00 00 4C A4
2nd BCH codeword	A9 55 55 55 55 55 55 7A
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79
CLTU (randomized)	
Start Sequence	EB 90
1st BCH codeword	CF 22 9E 5D 68 E9 4A FC
2nd BCH codeword	5C 55 55 55 55 55 55 F4
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79

Example 2: Type BC Frame with Control Command 'Set V(R)'

Spacecraft Identifier	1B
Virtual Channel Identifier	0
Frame Type	BC (shows that the frame carries a control command)
Frame Sequence Number	0 (all type-B frames have sequence number 0)
Frame Data Field length	3 octets
Frame Data Field content	82 00 00 (control command 'Set V(R)', V(R)= 0)
Frame length	10 octets
Frame Error Control Field	F6 F0
TC Transfer Frame	30 1B 00 09 00 82 00 00 F6 F0
CLTU (unrandomized)	
Start Sequence	EB 90
1st BCH codeword	30 1B 00 09 00 82 00 54
2nd BCH codeword	00 F6 F0 55 55 55 55 D6
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79
CLTU (randomized)	
Start Sequence	EB 90
1st BCH codeword	CF 22 9E 53 68 6B 06 0C
2nd BCH codeword	F5 9A 79 55 55 55 55 06
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79

Example 3: Type BC Frame with Control Command 'Set V(R)'

Spacecraft Identifier	1B
Virtual Channel Identifier	0
Frame Type	BC (shows that the frame carries a control command)
Frame Sequence Number	0 (all type-B frames have sequence number 0)
Frame Data Field length	3 octets
Frame Data Field content	82 00 10 (control command 'Set V(R)', V(R)= 16)
Frame length	10 octets
Frame Error Control Field	E4 C1
TC Transfer Frame	30 1B 00 09 00 82 00 10 E4 C1
CLTU (unrandomized)	
Start Sequence	EB 90
1st BCH codeword	30 1B 00 09 00 82 00 54
2nd BCH codeword	10 E4 C1 55 55 55 55 3E
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79
CLTU (randomized)	
Start Sequence	EB 90
1st BCH codeword	CF 22 9E 53 68 6B 06 0C
2nd BCH codeword	E5 88 48 55 55 55 55 EE
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79

Example 4: Type BC Frame with Control Command 'Set V(R)'

Spacecraft Identifier	1B
Virtual Channel Identifier	1
Frame Type	BC (shows that the frame carries a control command)
Frame Sequence Number	0 (all type-B frames have sequence number 0)
Frame Data Field length	3 octets
Frame Data Field content	82 00 00 (control command 'Set V(R)', V(R)= 0)
Frame length	10 octets
Frame Error Control Field	F0 51
TC Transfer Frame	30 1B 04 09 00 82 00 00 F0 51
CLTU (unrandomized)	
Start Sequence	EB 90
1st BCH codeword	30 1B 04 09 00 82 00 E8
2nd BCH codeword	00 F0 51 55 55 55 55 AA
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79
CLTU (randomized)	
Start Sequence	EB 90
1st BCH codeword	CF 22 9A 53 68 6B 06 B0
2nd BCH codeword	F5 9C D8 55 55 55 55 7A
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79

Example 5: Type BC Frame with Control Command 'Set V(R)'

Spacecraft Identifier	1B
Virtual Channel Identifier	1
Frame Type	BC (shows that the frame carries a control command)
Frame Sequence Number	0 (all type-B frames have sequence number 0)
Frame Data Field length	3 octets
Frame Data Field content	82 00 10 (control command 'Set V(R)', V(R)= 16)
Frame length	10 octets
Frame Error Control Field	E2 60
TC Transfer Frame	30 1B 04 09 00 82 00 10 E2 60
CLTU (unrandomized)	
Start Sequence	EB 90
1st BCH codeword	30 1B 04 09 00 82 00 E8
2nd BCH codeword	10 E2 60 55 55 55 55 42
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79
CLTU (randomized)	
Start Sequence	EB 90
1st BCH codeword	CF 22 9A 53 68 6B 06 B0
2nd BCH codeword	E5 8E E9 55 55 55 55 92
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79

Example 6: Type AD Frame for Data on the Sequence-Controlled Service

Spacecraft Identifier	1B
Virtual Channel Identifier	0
Frame Type	AD (the frame carries data on Sequence-Controlled service)
Frame Sequence Number	255
Frame Data Field length	1 octet
Frame Data Field content	01
Frame length	8 octets
Frame Error Control Field	70 FB
TC Transfer Frame	00 1B 00 07 FF 01 70 FB
CLTU (unrandomized)	
Start Sequence	EB 90
1st BCH codeword	00 1B 00 07 FF 01 70 E8
2nd BCH codeword	FB 55 55 55 55 55 55 5C
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79
CLTU (randomized)	
Start Sequence	EB 90
1st BCH codeword	FF 22 9E 5D 97 E8 76 B0
2nd BCH codeword	0E 55 55 55 55 55 55 D2
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79

Example 7: Type AD Frame for Data on the Sequence-Controlled Service

Spacecraft Identifier	1B
Virtual Channel Identifier	0
Frame Type	AD
Frame Sequence Number	0
Frame Data Field length	2 octets
Frame Data Field content	01 02
Frame length	9 octets
Frame Error Control Field	BE 58
TC Transfer Frame	00 1B 00 08 00 01 02 BE 58
CLTU (unrandomized)	
Start Sequence	EB 90
1st BCH codeword	00 1B 00 08 00 01 02 12
2nd BCH codeword	BE 58 55 55 55 55 55 0C
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79
CLTU (randomized)	
Start Sequence	EB 90
1st BCH codeword	FF 22 9E 52 68 E8 04 4A
2nd BCH codeword	4B 34 55 55 55 55 55 A0
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79

Example 8: Type AD Frame for Data on the Sequence-Controlled Service

Spacecraft Identifier	1B
Virtual Channel Identifier	0
Frame Type	AD
Frame Sequence Number	1
Frame Data Field length	3 octets
Frame Data Field content	01 02 03
Frame length	10 octets
Frame Error Control Field	F2 93
TC Transfer Frame	00 1B 00 09 01 01 02 03 F2 93
CLTU (unrandomized)	
Start Sequence	EB 90
1st BCH codeword	00 1B 00 09 01 01 02 0A
2nd BCH codeword	03 F2 93 55 55 55 55 5C
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79
CLTU (randomized)	
Start Sequence	EB 90
1st BCH codeword	FF 22 9E 53 69 E8 04 52
2nd BCH codeword	F6 9E 1A 55 55 55 55 8C
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79

Example 9: Type AD Frame for Data on the Sequence-Controlled Service

Spacecraft Identifier	1B
Virtual Channel Identifier	0
Frame Type	AD
Frame Sequence Number	2
Frame Data Field length	4 octets
Frame Data Field content	01 02 03 04
Frame length	11 octets
Frame Error Control Field	3C EB
TC Transfer Frame	00 1B 00 0A 02 01 02 03 04 3C EB
CLTU (unrandomized)	
Start Sequence	EB 90
1st BCH codeword	00 1B 00 0A 02 01 02 22
2nd BCH codeword	03 04 3C EB 55 55 55 44
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79
CLTU (randomized)	
Start Sequence	EB 90
1st BCH codeword	FF 22 9E 50 6A E8 04 7A
2nd BCH codeword	F6 68 B5 C4 55 55 55 0A
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79

Example 10: Type AD Frame for Data on the Sequence-Controlled Service

Spacecraft Identifier	1B
Virtual Channel Identifier	0
Frame Type	AD
Frame Sequence Number	6
Frame Data Field length	8 octets
Frame Data Field content	01 02 03 04 05 06 07 08
Frame length	15 octets
Frame Error Control Field	14 BB
TC Transfer Frame	00 1B 00 0E 06 01 02 03 04 05 06 07 08 14 BB
CLTU (unrandomized)	
Start Sequence	EB 90
1st BCH codeword	00 1B 00 0E 06 01 02 42
2nd BCH codeword	03 04 05 06 07 08 14 12
3rd BCH codeword	BB 55 55 55 55 55 55 3E
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79
CLTU (randomized)	
Start Sequence	EB 90
1st BCH codeword	FF 22 9E 54 6E E8 04 1A
2nd BCH codeword	F6 68 8C 29 A6 39 4A 5E
3rd BCH codeword	B3 55 55 55 55 55 55 C6
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79

Example 11: Type AD Frame for Data on the Sequence-Controlled Service

Spacecraft Identifier	1B
Virtual Channel Identifier	0
Frame Type	AD
Frame Sequence Number	7
Frame Data Field length	9 octets
Frame Data Field content	01 02 03 04 05 06 07 08 09
Frame length	16 octets
Frame Error Control Field	CF 90
TC Transfer Frame	00 1B 00 0F 07 01 02 03 04 05 06 07 08 09 CF 90
CLTU (unrandomized)	
Start Sequence	EB 90
1st BCH codeword	00 1B 00 0F 07 01 02 5A
2nd BCH codeword	03 04 05 06 07 08 09 8E
3rd BCH codeword	CF 90 55 55 55 55 55 AE
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79
CLTU (randomized)	
Start Sequence	EB 90
1st BCH codeword	FF 22 9E 55 6F E8 04 02
2nd BCH codeword	F6 68 8C 29 A6 39 57 C2
3rd BCH codeword	C7 50 55 55 55 55 55 AA
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79

Example 12: Type AD Frame for Data on the Sequence-Controlled Service

Spacecraft Identifier	1B
Virtual Channel Identifier	0
Frame Type	AD
Frame Sequence Number	0
Frame Data Field length	11 octets
Frame Data Field content	C0 10 00 C0 00 00 03 2E AF 8A 06
Frame length	18 octets
Frame Error Control Field	9F 71
TC Transfer Frame	00 1B 00 11 00 C0 10 00 C0 00 00 03 2E AF 8A 06 9F 71
CLTU (unrandomized)	
Start Sequence	EB 90
1st BCH codeword	00 1B 00 11 00 C0 10 14
2nd BCH codeword	00 C0 00 00 03 2E AF 9E
3rd BCH codeword	8A 06 9F 71 55 55 55 48
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79
CLTU (randomized)	
Start Sequence	EB 90
1st BCH codeword	FF 22 9E 4B 68 29 16 4C
2nd BCH codeword	F5 AC 89 2F A2 1F F1 D2
3rd BCH codeword	82 C6 CD D9 55 55 55 1C
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79

Example 13: Type AD Frame for Data on the Sequence-Controlled Service

Spacecraft Identifier	1B
Virtual Channel Identifier	0
Frame Type	AD
Frame Sequence Number	0
Frame Data Field length	12 octets
Frame Data Field content	C1 10 00 FF FF 00 04 01 02 03 11 82
Frame length	19 octets
Frame Error Control Field	8D 80
TC Transfer Frame	00 1B 00 12 00 C1 10 00 FF FF 00 04 01 02 03 11 82 8D 80
CLTU (unrandomized)	
Start Sequence	EB 90
1st BCH codeword	00 1B 00 12 00 C1 10 1C
2nd BCH codeword	00 FF FF 00 04 01 02 E2
3rd BCH codeword	03 11 82 8D 80 55 55 F0
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79
CLTU (randomized)	
Start Sequence	EB 90
1st BCH codeword	FF 22 9E 48 68 28 16 44
2nd BCH codeword	F5 93 76 2F A5 30 5C AE
3rd BCH codeword	0B D1 D0 25 3B 55 55 98
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79

Example 14: Type AD Frame for Data on the Sequence-Controlled Service

Spacecraft Identifier	1B
Virtual Channel Identifier	0
Frame Type	AD
Frame Sequence Number	0
Frame Data Field length	12 octets
Frame Data Field content	C1 11 04 C0 00 00 04 01 02 03 72 17
Frame length	19 octets
Frame Error Control Field	8D 80
TC Transfer Frame	00 1B 00 12 00 C1 11 04 C0 00 00 04 01 02 03 72 17 8D 80
CLTU (unrandomized)	
Start Sequence	EB 90
1st BCH codeword	00 1B 00 12 00 C1 11 96
2nd BCH codeword	04 C0 00 00 04 01 02 F4
3rd BCH codeword	03 72 17 8D 80 55 55 00
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79
CLTU (randomized)	
Start Sequence	EB 90
1st BCH codeword	FF 22 9E 48 68 28 17 CE
2nd BCH codeword	F1 AC 89 2F A5 30 5C B8
3rd BCH codeword	0B B2 45 25 3B 55 55 68
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79

Example 15: Type BD Frame for Data on the Expedited Service

Spacecraft Identifier	1B
Virtual Channel Identifier	0
Frame Type	BD
Frame Sequence Number	0 (all type-B frames have sequence number 0)
Frame Data Field length	1 octet
Frame Data Field content	E1
Frame length	8 octets
Frame Error Control Field	BB 22
TC Transfer Frame	20 1B 00 07 00 E1 BB 22
CLTU (unrandomized)	
Start Sequence	EB 90
1st BCH codeword	20 1B 00 07 00 E1 BB 38
2nd BCH codeword	22 55 55 55 55 55 55 A2
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79
CLTU (randomized)	
Start Sequence	EB 90
1st BCH codeword	DF 22 9E 5D 68 08 BD 60
2nd BCH codeword	D7 55 55 55 55 55 55 2C
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79

Example 16: Type BD Frame for Data on the Expedited Service

Spacecraft Identifier	1B
Virtual Channel Identifier	0
Frame Type	BD
Frame Sequence Number	0 (all type-B frames have sequence number 0)
Frame Data Field length	11 octets
Frame Data Field content	C0 10 00 C0 00 00 03 2E AF 8A 06
Frame length	18 octets
Frame Error Control Field	D9 65
TC Transfer Frame	20 1B 00 11 00 C0 10 00 C0 00 00 03 2E AF 8A 06 D9 65
CLTU (unrandomized)	
Start Sequence	EB 90
1st BCH codeword	20 1B 00 11 00 C0 10 E0
2nd BCH codeword	00 C0 00 00 03 2E AF 9E
3rd BCH codeword	8A 06 D9 65 55 55 55 2C
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79
CLTU (randomized)	
Start Sequence	EB 90
1st BCH codeword	DF 22 9E 4B 68 29 16 B8
2nd BCH codeword	F5 AC 89 2F A2 1F F1 D2
3rd BCH codeword	82 C6 8B CD 55 55 55 78
Tail Sequence	C5 C5 C5 C5 C5 C5 C5 79

F2 LDPC CODED FRAMES AND CLTUS

Just as an error correcting code will correct errors introduced by channel noise, it will also correct some implementation errors. Hence, interoperability is best tested by providing identical inputs to two implementations of the sending end and verifying that their outputs exactly match, symbol by symbol. Moreover, only the sending end is standardized. Implementers are free to build their receiving equipment in any way they choose, particularly with respect to LDPC decoding algorithms, and detection algorithms for the start and tail sequences. 'Good' implementations will all have very similar performance in a statistical sense, but individual results may vary.

Example 1: Text string with the (128,64) Code

At the sending end, the Synchronization and Channel Coding Sublayer accepts Transfer Frames as input. While not consistent with what might be produced by the Data Link Protocol Sublayer, the 70-character ASCII text string, 'Short Blocklength LDPC Codes for TC Synchronization and Channel Coding', is used here as an example. This sample transfer frame is listed in hexadecimal in table F-1; it is converted to a binary string by taking the hexadecimal octets in order, Most Significant Bit (MSB) first within each octet.

Table F-1: Example Transfer Frame

53	68	6f	72	74	20	42	6c	6f	63	6b	6c	65	6e	67	74
68	20	4c	44	50	43	20	43	6f	64	65	73	20	66	6f	72
20	54	43	20	53	79	6e	63	68	72	6f	6e	69	7a	61	74
69	6f	6e	20	61	6e	64	20	43	68	61	6e	6e	65	6c	20
43	6f	64	69	6e	67										

At the sending end, the following steps are performed:

- fill data is added to form an integer number of codewords;
- the data is partitioned into codewords and encoded;
- the codewords are pseudo-randomized;
- a start sequence and optional tail sequence are added.

For the example transfer frame above, two octets of fill data are added, the result is encoded to form nine 128-bit codewords, and each codeword is pseudo-randomized. The randomizer is set to the all-ones state at the beginning of each codeword. When the 64-bit start sequence and optional 128-bit tail sequence are added, the 1344-bit result is shown in table F-2 (again displayed as octets, MSB first).

Table F-2: Example CLTU Encoded with the (128,64) LDPC Code and with the Optional Tail Sequence Included

03	47	76	c7	27	28	95	b0	ac	51	f1	28	1c	c9	44	99
8b	ac	73	70	f6	70	50	64	90	5a	f5	36	0d	87	61	81
b0	ed	79	e1	24	2c	2b	09	97	19	d2	1e	38	aa	26	b6
05	34	d6	14	aa	19	8f	c0	90	5d	fb	29	48	8f	69	87
ff	fc	68	28	91	cf	67	47	df	6d	dd	7a	3b	90	68	96
e8	fe	1b	f0	ef	fc	ad	85	97	4b	f1	34	01	93	67	81
f4	57	4c	82	fe	09	b9	b0	96	56	f0	7a	09	87	62	d5
46	fa	a1	f8	e6	38	aa	6d	bc	51	ff	34	06	8c	6a	d5
ba	9a	7e	65	4d	9e	77	ee	bc	56	fa	33	06	8e	53	a0
61	67	ea	9a	82	cc	70	2e	55	55	55	56	aa	aa	aa	aa
55	55	55	55	55	55	55	55								

Any sublayer implementation that produces this output from the same input is interoperable at the sending side. While the receiving side is not standardized, an implementation should be able to locate this CLTU when embedded in random binary data and combined with Additive White Gaussian Noise (AWGN) with a Signal to Noise Ratio (SNR) above about E_b/N_0 = 4 dB. The result will be a Transfer Frame that appears in ASCII as 'Short Blocklength LDPC Codes for TC Synchronization and Channel CodingUU', because this sublayer cannot remove the two octets of fill data that have binary value 01010101, or 'U' in ASCII.

Example 2: Text String with the (512, 256) LDPC Code

Here, the same example from table F-1 is repeated using the longer (512,256) LDPC code. In this case, 26 octets of fill data are added, and three 512-bit codewords are formed and pseudo-randomized. Again, the randomizer is set to the all-ones state at the beginning of each codeword. When the 64-bit start sequence is added, the 1600-bit result is shown in table F-3 (again displayed as octets, MSB first).

Table F-3: Example CLTU Encoded with the (512,256) LDPC Code

03	47	76	c7	27	28	95	b0	ac	51	f1	28	1c	c9	44	99
03	ea	44	cd	54	30	6f	b4	3a	88	f7	ea	1e	81	e7	ae
09	b8	5d	a7	d8	e0	3f	4f	6b	45	d6	b5	12	1c	dc	eb
52	f4	2a	39	30	3b	48	ef	bf	1c	98	46	fe	9b	5e	0f
b5	2a	db	79	da	47	fe	dd	df	6d	dd	7a	3b	90	68	96
04	fb	40	cf	58	24	69	b4	3b	c7	d5	8e	2f	ac	a3	cd
25	b4	59	ba	96	e3	3c	1d	6c	0f	32	09	22	dd	4c	b7
5b	73	64	07	5e	da	f3	3a	35	8a	95	ac	b8	ea	33	5b
f8	e6	c8	8e	69	53	71	4a	bc	56	fa	33	06	8e	53	a0
39	dc	7a	f4	64	0b	5d	95	07	fd	ee	fb	1b	97	92	b8
33	89	6d	81	ad	d3	05	68	c6	94	37	c1	89	82	35	6f
9f	4b	0f	40	a6	95	f3	3d	0f	bd	3e	32	8d	2f	06	66
e0	64	f5	37	cc	58	41	60								

Any sublayer implementation that produces this output from the same input is interoperable at the sending side. While the receiving side is not standardized, an implementation should be able to locate this CLTU when embedded in random binary data and combined with AWGN with a SNR above about E_b/N_0 = 3 dB. The result will be a Transfer Frame that appears in ASCII as 'Short Blocklength LDPC Codes for TC Synchronization and Channel CodingUUUUUUUUUUUUUUUUUUUUUUUUUU', because this sublayer cannot remove the 26 octets of fill data.

Example 3: Repeated Transfer Frame with the (128,64) LDPC Code

As one more case, Example 1 from F-1 is considered, the transfer frame is an eight-octet 'hardware command' shown in table F-4, and it is consistent with the transfer frames that might be generated by the TC Data Link Protocol [3].

Table F-4: Example Transfer Frame with Hardware Command

30	1B	00	07	00	00	4C	A9

With the (128,64) code, no fill data are required. One 128-bit codeword is formed and pseudo-randomized. With the 64-bit start sequence, and without the optional tail sequence, a 192-bit CLTU results. If a repetition factor of 3 is specified, the CLTU is repeated three times, and the result is given in table F-5.

Table F-5: Example CLTUs Generated from a Single (128,64) LDPC Codeword, with No Tail Sequence, and Repetition Factor of 3

03	47	76	c7	27	28	95	b0	cf	22	9e	5d	68	e9	4a	5c
2e	98	ea	12	bf	94	8b	66	03	47	76	c7	27	28	95	b0
cf	22	9e	5d	68	e9	4a	5c	2e	98	ea	12	bf	94	8b	66
03	47	76	c7	27	28	95	b0	cf	22	9e	5d	68	e9	4a	5c
2e	98	ea	12	bf	94	8b	66								

Performance will vary, but when embedded in random binary data and combined with AWGN with an SNR above about $E_b/N_0 = 2$ dB ($E_s/N_0 = -1$ dB, where E_s is the energy per transmitted BPSK symbol), a good implementation should be able to locate and recover at least one of the three copies of this CLTU with a probability of about 95 percent.

ANNEX G

PERFORMANCE EVALUATION OF A GO-BACK-*N* SCHEME WITH MULTIPLE COPIES

NOTE – This annex contains an edited version of a report by Università Politecnica delle Marche, Ancona, Italy, in 2010. The report was produced for ESA and was an input for the CCSDS working groups that considered the addition of the systematic retransmission described in section 10.

G1 INTRODUCTION

A classic GBN scheme is shown, for an example, in figure G-1. When a frame is not successfully received (#3 in the figure), notification is sent through a 'Negative Acknowledgement' (NAK) to the transmitter, which sends the lost frame again, together with all the following ones. For the example in the figure, it is assumed that the receiver is able to send to the transmitter a NAK as the response to an incorrect reception. As an alternative, the NAK acquisition can be replaced by a counter timeout.

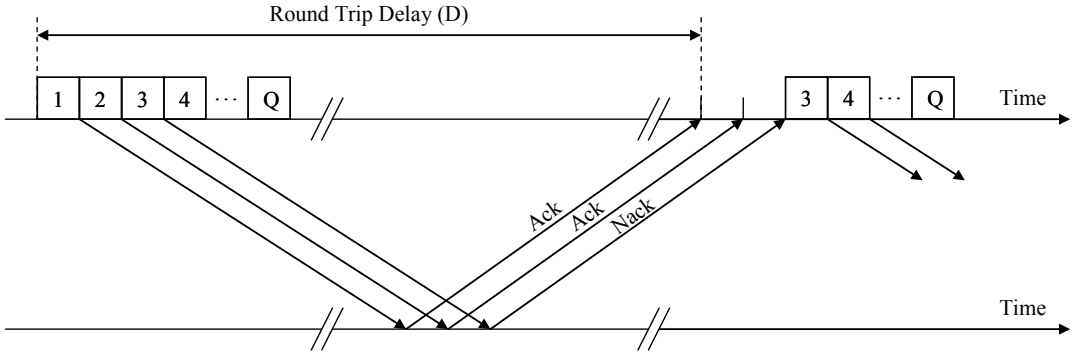

Figure G-1: Basic Go-Back-*N* Scheme

In this annex, a different scheme, Go-Back-*N* with Multiple Copies (MGBN), is also considered. It is shown, for an example, in figure G-2. This is a particular case of a more general class of Automatic Repeat Queuing (ARQ) strategies, described in reference [13]. In the considered scheme, each frame is transmitted M times (M = 3 in the example), and retransmission occurs only when the receiver rejects all repetitions.

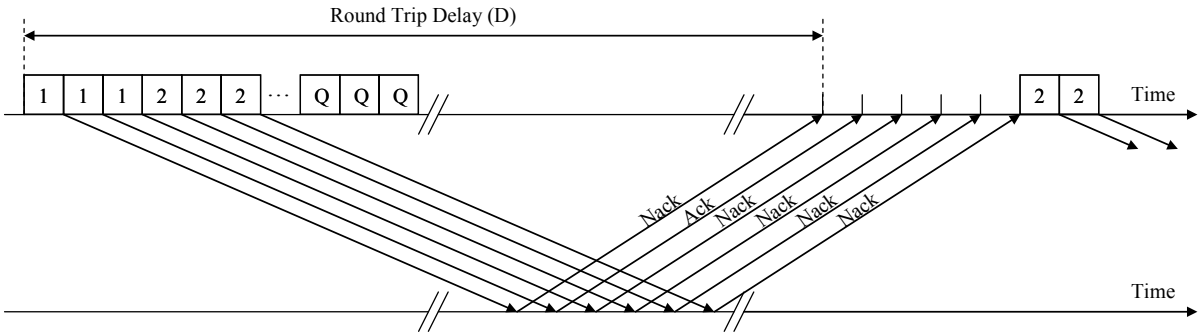

Figure G-2: Go-Back-*N* Scheme with Multiple Copies

The GBN scheme is usually effective for missions with relatively small propagation delays, but it can encounter problems in the case of moderate/high round-trip delay (reference [14]) and/or high error rate. Under such conditions, the MGBN scheme can have better performance (reference [13]). On the other hand, in the case of the presence of error bursts, when the burst length is on the order of the chosen M, the MGBN scheme does not yield similar benefits, and it can be preferable to adopt further variants, such as that presented in reference [15].

G2 PERFORMANCE ANALYSIS

The following notation is used in this annex:

p = transition error probability (from 1 to 0 and vice versa);

P_{ed} = frame rejection probability;

M = number of repetitions of each frame within a retransmission;

I = number of retransmissions;

P_{nsr} = probability of non-successful reception;

$\left(P_{nsr}\right)^{*}_{MGBN}$ = designed threshold for $\left(P_{nsr}\right)_{MGBN}$;

Q = number of frames in a transmitted sequence;

D = round-trip delay;

T_f = frame duration;

T_{ave} = average time required for the successful transmission of a frame;

ρ = average throughput;

$i_{\max}$ = maximum number of scheduled retransmissions.

In order to discuss when the MGBN scheme can behave better than the GBN one, some 'performance figures' can be considered.

The first figure is the *average number of retransmissions*. It is expected that the MGBN scheme allows reducing the average number of retransmissions, at least under the assumption of uniform error distribution. A formal demonstration is provided below.

The value of P_{ed} depends on the quality of the channel. More precisely, the lower the value of p, the lower the value of P_{ed}. Explicit analytical formulas can be given to describe such a behavior.

Using the random variable I, representing the number of retransmissions, the probability of sending a number of retransmissions $I = i$ (with $i = 0, 1, 2, \ldots$) is:

$$\Pr\{I = i\} = P_{ed}^{iM}\left(1 - P_{ed}^{M}\right). \tag{40}$$

In this annex, the meaning of *retransmission* is sending of a new sequence of frames, where each frame is repeated M times. Consequently, with I retransmissions, each frame is sent $M \cdot I$ times.

The average value of I can be easily computed; it results in:

$$\langle I \rangle_{MGBN} = \sum_{i=1}^{\infty} i \Pr\{I = i\} = \sum_{i=1}^{\infty} i P_{ed}^{iM}\left(1 - P_{ed}^{M}\right) = \frac{P_{ed}\left(1 - P_{ed}^{M}\right)}{M} \sum_{i=0}^{\infty} i M P_{ed}^{iM-1} = \frac{P_{ed}^{M}}{1 - P_{ed}^{M}}. \tag{41}$$

Equation 41 provides the average value for the GBN scheme too, simply setting $M = 1$. So:

$$\langle I \rangle_{GBN} = \frac{P_{ed}}{1 - P_{ed}}. \tag{42}$$

At this point, the following ratio can be considered:

$$\frac{\langle I \rangle_{MGBN}}{\langle I \rangle_{GBN}} = \frac{P_{ed}^{M}}{1 - P_{ed}^{M}} \cdot \frac{1 - P_{ed}}{P_{ed}} = \frac{P_{ed}^{M-1}}{1 + P_{ed} + P_{ed}^{2} + \cdots + P_{ed}^{M-1}}. \tag{43}$$

The value assumed by this ratio is shown in figure G-3, as a function of P_{ed} and for different values of M. As expected (and also evident from equation 43), it is always smaller than 1, thus proving that having multiple frame transmissions allows reducing the number of retransmissions (on average).

It is necessary to observe that the developed analysis does not consider possible problems due to the presence of buffers. Similarly, in the computation, it is assumed that no limit is imposed on the number of retransmissions. The latter assumption, rather common in the literature, is an approximation, whose impact, however, is always negligible since the probability of having a high number of retransmissions rapidly approaches zero.

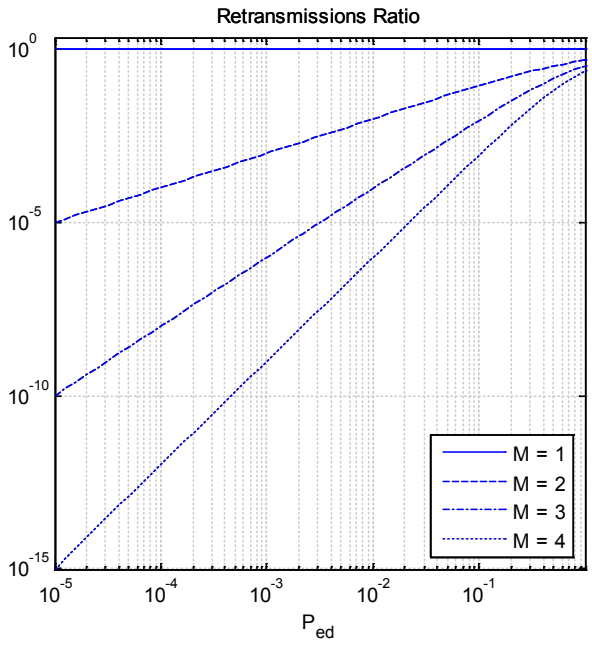

Figure G-3: Retransmissions Ratio for Different Values of *M*

A more direct way for evaluating the efficiency of the MGBN scheme consists in calculating the probability P_{nsr} that a frame (and, by extension, a sequence of frames) is not successfully received within a maximum number of retransmissions $i_{\max}$. It is easy to find:

$$\begin{aligned} \left(P_{nsr}\right)_{GBN} &= P_{ed}^{i_{\max}+1}, \\ \left(P_{nsr}\right)_{MGBN} &= P_{ed}^{(i_{\max}+1)M}. \end{aligned} \tag{44}$$

So:

$$\left(P_{nsr}\right)_{MGBN} = \left[\left(P_{nsr}\right)_{GBN}\right]^{M}. \tag{45}$$

This rule holds in general. In the particular case of $i_{\max} = 0$ (no retransmission allowed), it is:

$$\begin{aligned} \left(P_{nsr}\right)_{GBN} &= P_{ed}, \\ \left(P_{nsr}\right)_{MGBN} &= P_{ed}^{M}. \end{aligned}$$

For a given value of P_{ed}, it is therefore simple to determine the smallest value of M that permits having $\left(P_{nsr}\right)_{MGBN}$ smaller than a prefixed threshold, the latter being denoted by $\left(P_{nsr}\right)_{MGBN}^{*}$. In fact, for $i_{\max} = 0$, this value results in (the logarithm base is unimportant):

$$M = \left\lceil \frac{\log\left[\left(P_{nsr}\right)^{*}_{MGBN} \right]}{\log\left[P_{ed}\right]} \right\rceil . \tag{46}$$

This special case ($i_{max} = 0$) also models the systematic retransmission procedure (described in 10.3.3.2).

G3 NUMERICAL EXAMPLES

Example 1

For the case $P_{ed} = 0.1$ and $i_{max} = 0$, table G-1 shows the values of M that are required for smaller and smaller $\left(P_{nsr}\right)^{*}_{MGBN}$ (one order of magnitude from one value to another).

Table G-1: Minimum M to Have the Probability of Nonsuccessful Reception Smaller Than the Specified Threshold for $P_{ed} = 0.1$

$\left(P_{nsr}\right)^{*}_{MGBN}$	M
0.01	2
0.001	3
10^{-4}	4
10^{-5}	5
…	…

The 'price' to pay for the P_{nsr} reduction is the increase in the ACK waiting time.

Example 2

The round-trip delay is $D = 100$ min = 6000 s.

The data rate is 4 kbps (for deep-space missions, the data rate can be lower: for example, 1 kb/s).

The frame length is 300 octets.

$P_{ed} = 0.1$.

Using the data above, each frame lasts $T_f = 0.6$ sec.

By employing the GBN scheme, if reception is successful without retransmissions, the ACK is received after $D + T_f$ sec. (this time is measured from the start of the transmission); by employing the MGBN scheme, instead, this time grows up to $D + MT_f$ sec. If D is large (that

is certainly true for the considered space applications, particularly in the deep space scenario) the increase of $(M-1)T_f$ is negligible. As an example, with the above numerical values and setting $M = 5$, the extra-delay should be 2.4 sec.

However, the frames are not transmitted individually; on the contrary, they are organized in sequences, each sequence including Q frames (a typical value is $Q = 100$); so the delay becomes:

– $D + MQT_f$ sec. in the worst case (the ACK is sent after the last repetition of the last frame in the sequence);

– $D + MQT_f - (M-1)T_f$ sec. in the best case (the ACK is sent after the first repetition of the last frame in the sequence).

By using the numerical values above and setting $M = 5$ and $Q = 100$, it is possible to find:

GBN: $D + QT_f = 101$ min.

MGBN: $D + MQT_f = 105$ min.

$D + MQT_f - (M-1)T_f = 104$ min, 57 sec, 60 cent.

So the waiting time in the MGBN scheme increases, with the specified values, by about 4 min. As a positive counterpart, however, it is possible to observe that, while the probability of having an ACK acquisition in 101 min., for the GBN scheme, is equal to $1 - P_{ed} = 0.9$, the probability of having an ACK acquisition in 105 min. (or even earlier), for the MGBN scheme, is $1 - P_{ed}^5 = 0.99999$, which is much higher. So it is possible to conclude that the advantage offered by the MGBN scheme seems more significant than the disadvantage. In the same perspective, it is important to observe that a probability 0.99999 is achieved, by using the GBN scheme, only after 404 min; that means a delay of 404 min.

G4 AVERAGE TIME FOR SUCCESSFUL TRANSMISSION

A further way to evaluate the efficiency of the system consists in computing the average time, T_{ave}, that is required for successfully transferring a frame. The inverse of T_{ave}, suitably normalized, gives the throughput, and it is particularly significant in the case of continuous transmissions. In any event, it provides a useful tool for comparing the different ARQ schemes.

As usual in the literature, it is convenient to express the round-trip delay as a multiple (n) of T_f (frame transmission time), that is, putting $D = nT_f$. With the numerical data in Example 2 of G3, $n = 10000$.

By generalizing the well-known procedure for the T_{ave} computation in the GBN scheme, it is easy to obtain:

$$\left(T_{ave}\right)_{MGBN} = MT_f\left[1+\frac{nP_{ed}^M}{M\left(1-P_{ed}^M\right)}\right]. \tag{47}$$

The value of T_{ave} for the GBN scheme can be computed from equation 47 as a particular case (M=1), and results in:

$$\left(T_{ave}\right)_{GBN} = T_f\left[1+\frac{nP_{ed}}{1-P_{ed}}\right]. \tag{48}$$

For the sake of comparison, it is useful to consider the following ratio:

$$\frac{\left(T_{ave}\right)_{GBN}}{\left(T_{ave}\right)_{MGBN}} = \frac{T_f\left[1+\frac{nP_{ed}}{1-P_{ed}}\right]}{MT_f\left[1+\frac{nP_{ed}^M}{M\left(1-P_{ed}^M\right)}\right]} = \frac{1+(n-1)P_{ed}}{M+(n-M)P_{ed}^M}\cdot\frac{1-P_{ed}^M}{1-P_{ed}}, \tag{49}$$

which expresses the inverse of the ratio between the average throughputs (ρ) resulting from the considered schemes; that is:

$$\frac{\left(T_{ave}\right)_{GBN}}{\left(T_{ave}\right)_{MGBN}} = \frac{\rho_{MGBN}}{\rho_{GBN}}. \tag{50}$$

The MGBN scheme is more favorable than the GBN scheme, from the throughput viewpoint, when the ratio equation 50 is greater than 1 (more and more for higher values of the ratio).

On the other hand, the ratio is determined by the values of M, P_{ed} and n. Some examples are shown in figures G-4–G-9, as a function of n, by considering different P_{ed} and using M as a parameter.

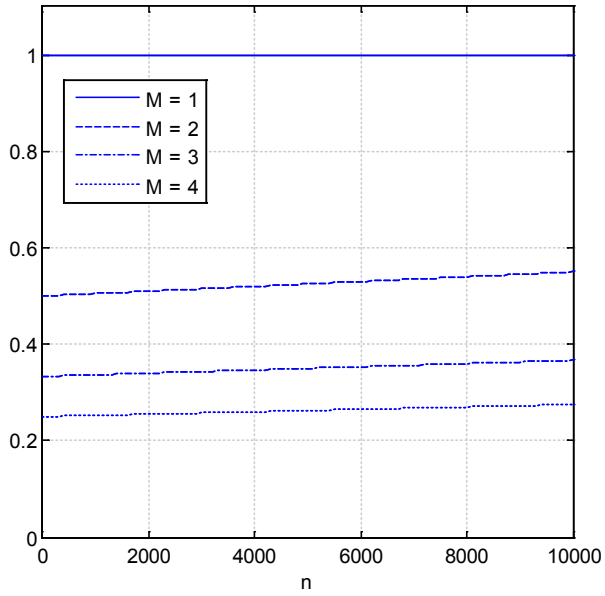

Figure G-4: Throughput Ratio for Different Values of M When $P_{ed} = 10^{-5}$

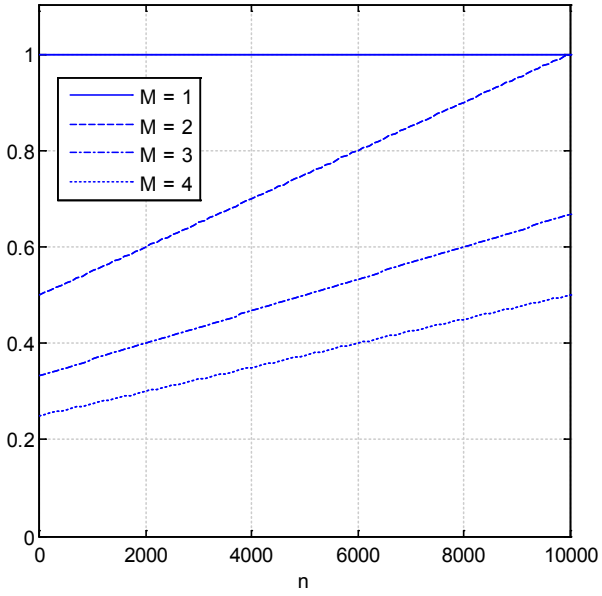

Figure G-5: Throughput Ratio for Different Values of M When $P_{ed} = 10^{-4}$

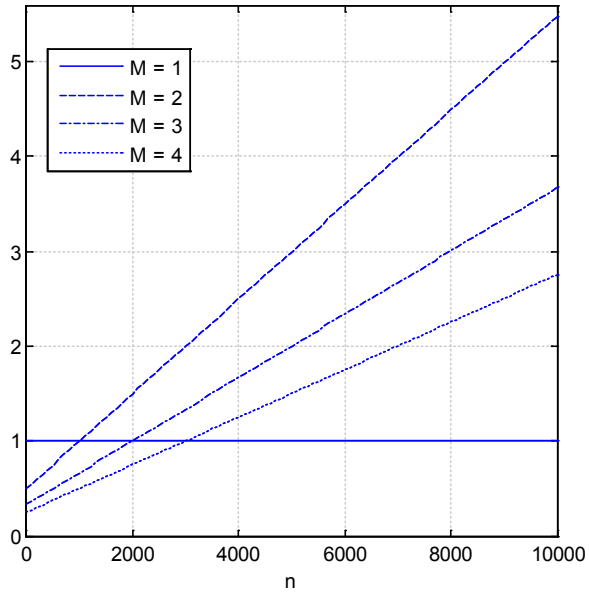

Figure G-6: Throughput Ratio for Different Values of M When $P_{ed} = 10^{-3}$

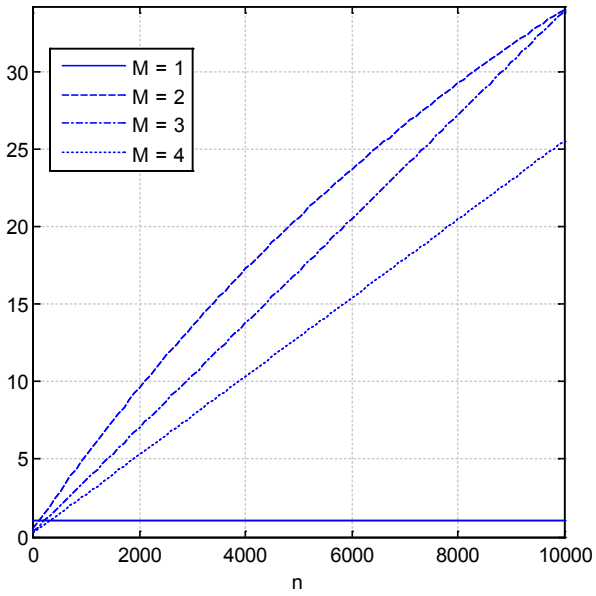

Figure G-7: Throughput Ratio for Different Values of M When $P_{ed} = 10^{-2}$

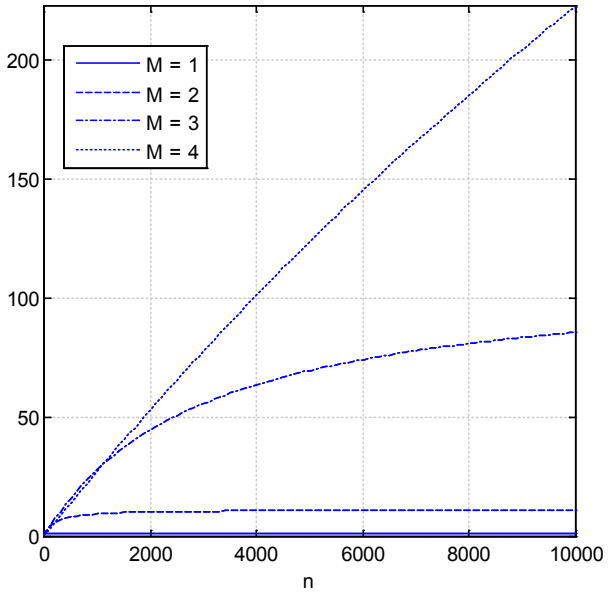

Figure G-8: Throughput Ratio for Different Values of *M* When $P_{ed} = 10^{-1}$

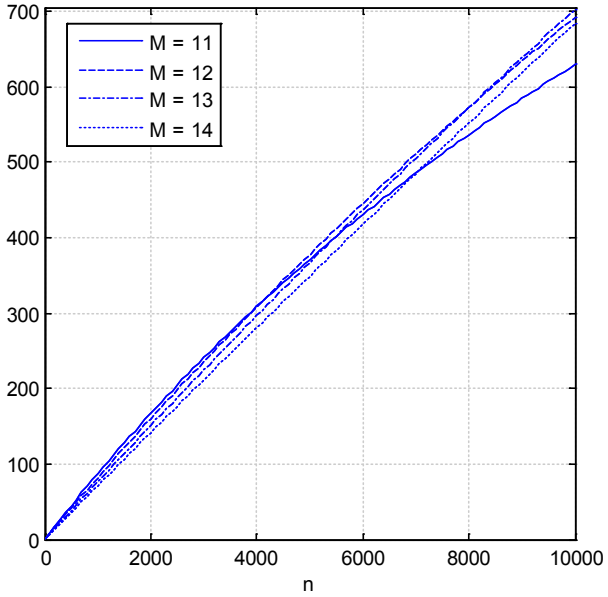

Figure G-9: Throughput Ratio for Different Values of *M* When $P_{ed} = 0.5$

From the previous figures, it is possible to see that, for very small values of P_{ed}, the performance parameter considered here (i.e., $(T_{ave})_{GBN} / (T_{ave})_{MGBN}$) is not improved by the adoption of the MGBN scheme: in fact, the ratio is always smaller than 1 for all the considered values of n (for n = 10000, in particular, it is the value resulting from the round-trip delay computed in the Example 2). On the contrary, for $P_{ed} \geq 0.001$ (which, actually, are realistic values in many operating conditions), the MGBN scheme becomes more and more convenient, and it is even possible to optimize M. Focusing attention on the case n = 10000, table G-2 shows the value of M that maximizes the throughput ratio, for different values of P_{ed}.

Table G-2: Values of *M* That Maximize the Throughput Ratio for Different Values of P_{ed}

P_{ed}	M_{opt}
10^{-5}	1
10^{-4}	1
0.001	2
0.01	2 (o 3)
0.1	4
0.5	13

ANNEX H

EFFECT OF UNDETECTED ERRORS IN CALCULATING THE PROBABILITY OF CODEWORD REJECTION

NOTE – This annex contains an edited version of the report 'ESA-Univpm.BCH_Codeblock_Rejection.v0.2' by Università Politecnica delle Marche, Ancona, Italy, October 2011. The report was produced for ESA and was an input for the CCSDS working groups at the Boulder meeting in 2011.

It was first pointed out in reference [16] that the expressions used for the probability of CLTU rejection in the previous version of the TC Green Book did not take into account the possible occurrence of undetected errors.

In the considered settings, a (n = 63, k = 56) BCH code is used, which is obtained from the (63, 57) Hamming code by expurgation. The (63, 56) BCH code has minimum distance 4; thus it can be used in TED mode or, alternatively, in SEC mode (which means it is also able to detect double errors).

Each CLTU is formed by a number N of consecutive BCH codewords, and it is rejected when at least one of such codewords is revealed to be in error by the BCH decoder. A codeword error event is not detected by the BCH code when the error vector coincides with a valid codeword having Hamming weight greater than 0. A_i denotes the number of weight-i codewords in the BCH code.

H1 TED MODE

When the BCH code is used in TED mode, the occurrence of an error vector with weight $i \geq 1$ causes a codeword rejection, on condition that it is detected. So, the probability of codeword rejection is:

$$\begin{aligned} P_{RX} &= \sum_{i=1}^{n}\left[\binom{n}{i} - A_i\right] p^i (1-p)^{n-i} = \\ &= 1-(1-p)^n - \sum_{i=1}^{n} A_i p^i (1-p)^{n-i}, \\ &= 1-(1-p)^n - P_{UE}, \end{aligned}$$

where p denotes the channel bit error probability, and $P_{UE} = \sum_{i=1}^{n} A_i p^i (1-p)^{n-i}$ is the undetected error probability. The probability of CLTU rejection hence follows as:

$$P_{CX} = 1-\left[1-P_{RX}\right]^N.$$

Upper bounds on P_{RX} and P_{CX} can be obtained by neglecting P_{UE} in their expressions; that is:

$$P_{RX} \le \tilde{P}_{RX} = 1-(1-p)^n,$$
$$P_{CX} \le \tilde{P}_{CX} = 1-\left[(1-p)^n\right]^N.$$

Taking into account that P_{UE} is usually very small, $\tilde{P}_{RX}$ and $\tilde{P}_{CX}$ can be considered as approximate estimations of P_{RX} and P_{CX}. Figure H-1 reports a comparison between the exact and approximate values of P_{RX}, so obtained, for the case of the (63, 56) BCH code, having the weight spectrum reported in H3. As can be seen in figure H-1, neglecting the term P_{UE} results in a very good approximation.

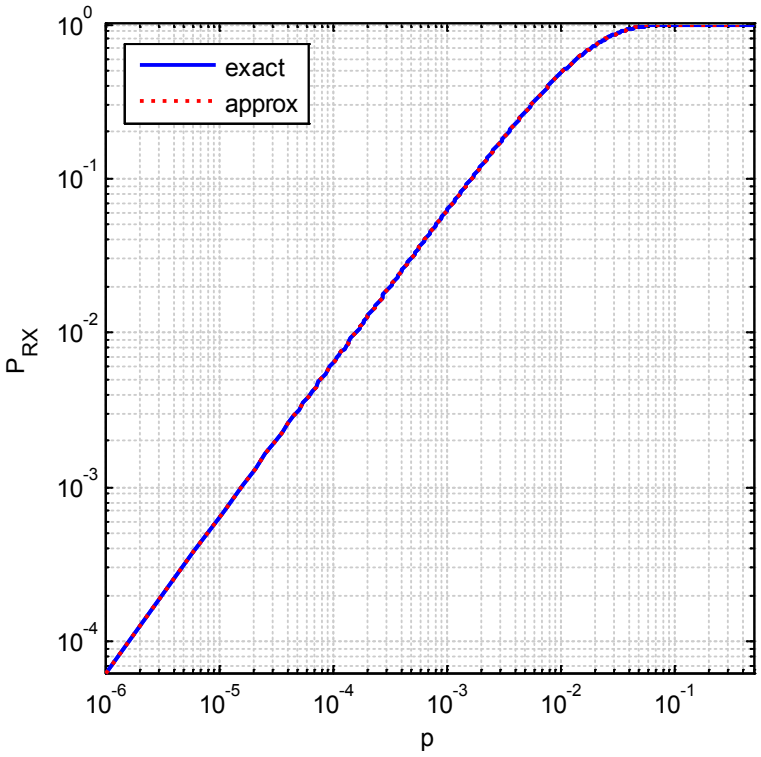

Figure H-1: Exact and Approximate Values of P_{RX} for the (63, 56) BCH Code

This situation is even more evident if the exact and approximate values of P_{CX} are compared. This has been done in figure H-2, where the case in which $N = 20$ is considered.

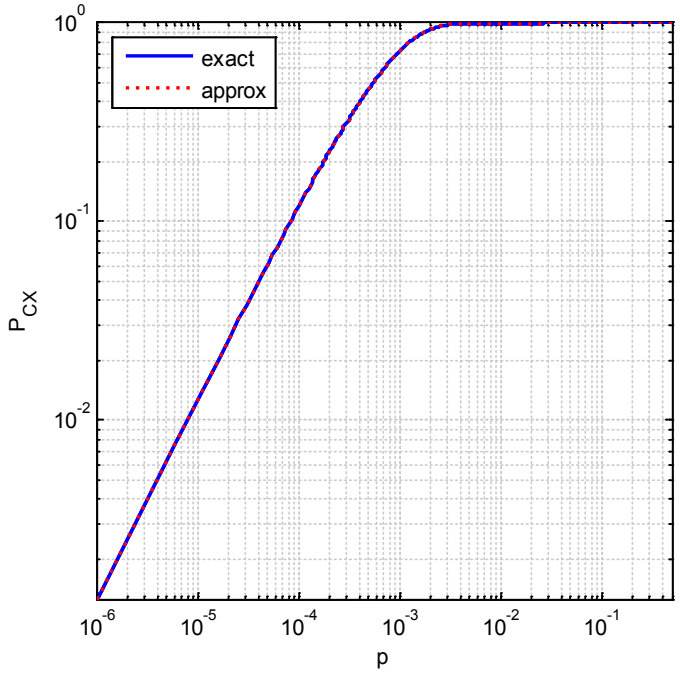

Figure H-2: Exact and Approximate Values of P_{CX} for the (63, 56) BCH Code and $N = 20$

For better clarity, figure H-3 reports the percentage error between the approximate and the exact values of P_{CX}, for $N = 20$. From the figure, it can be observed that the error is always very small ($< 4 \cdot 10^{-5}$ %), at least for the considered choice of the parameters.

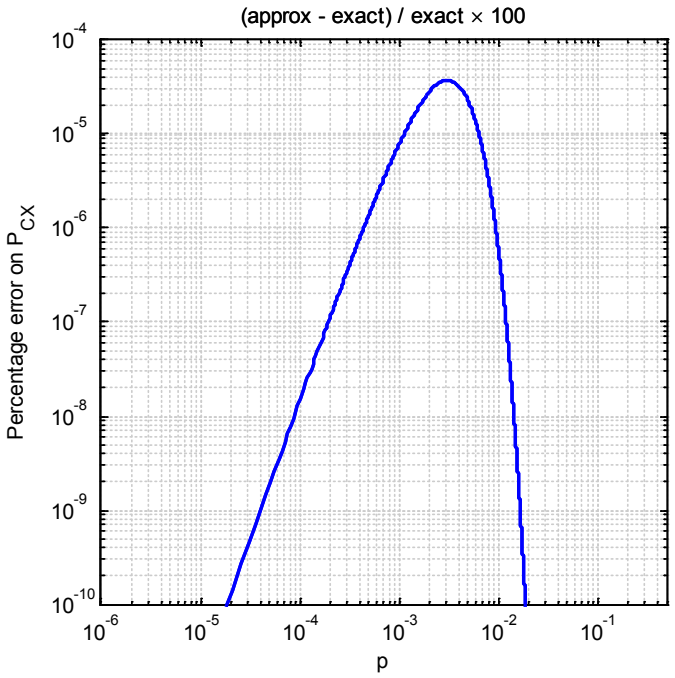

Figure H-3: Percentage Error between the Approximate and the Exact Values of P_{CX} for the (63, 56) BCH Code and $N = 20$

H2 SEC MODE

When the BCH code is used in SEC mode, the occurrence of an error vector with weight $i \geq 2$ causes a codeword rejection, on condition that it is detected. A first kind of undetected error occurs when a weight-i error pattern causes transition to one of the A_i near codewords, as in TED mode. This event has probability:

$$P_1(i) = A_i p^i (1-p)^{n-i}.$$

In addition, in SEC mode, an undetected error also occurs when the error pattern causes transition to a vector that is at Hamming distance 1 from another codeword, in which decoding results. There are n vectors at distance 1 from each codeword. If, starting from a codeword, a weight-i error pattern exists that causes transition to another codeword, then i error patterns with weight $i - 1$ and $n - i$ error patterns with weight $i + 1$ exist, causing transition to a vector at Hamming distance 1 from the same codeword. Therefore the event of a transition to a vector at Hamming distance 1 from a codeword has probability:

$$P_2(i) = A_i \left[i p^{i-1} (1-p)^{n-i+1} + (n-i) p^{i+1} (1-p)^{n-i-1} \right].$$

So, the probability of codeword rejection is:

$$P_{RY} = \sum_{i=2}^{n} \left[\binom{n}{i} p^i (1-p)^{n-i} - P_1(i) - P_2(i) \right] =$$

$$= 1 - (1-p)^n - np(1-p)^{n-1} - \sum_{i=2}^{n} \left[P_1(i) + P_2(i) \right].$$

The same result can be obtained by following more general approaches (reference [17]). In this case, the probability of CLTU rejection results in:

$$P_{CY} = 1 - \left[1 - P_{RY} \right]^N.$$

Also in this case, upper bounds on the values of P_{RY} and P_{CY} can be obtained by neglecting $P_1(i)$ and $P_2(i)$, that is:

$$P_{RY} \leq \tilde{P}_{RY} = 1 - (1-p)^n - np(1-p)^{n-1},$$

$$P_{CY} \leq \tilde{P}_{CY} = 1 - \left[(1-p)^n + np(1-p)^{n-1} \right]^N.$$

If it is taken into account that $P_1(i)$ and $P_2(i)$ are usually very small, it is possible to consider $\tilde{P}_{RY}$ and $\tilde{P}_{CY}$ as approximate estimations of P_{RY} and P_{CY}. Figure H-4 reports a comparison between the exact and approximate values of P_{RY} for the case of the (63, 56) BCH code, having the weight spectrum reported in H3. As shown in figure H-4, the effect of the approximation is now more evident with respect to the case in which the code is used in TED mode.

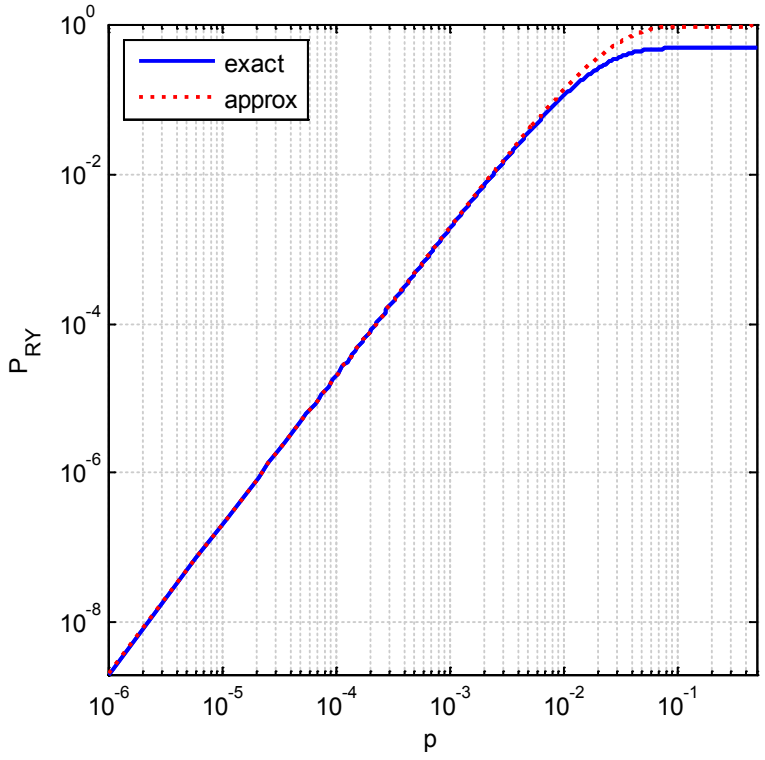

Figure H-4: Exact and Approximate Values of P_{RY} for the (63, 56) BCH Code

Also in this case, it is useful to compare the exact and approximate values of P_{CY}. This has been done in figure H-5, where the case with $N = 20$ is considered.

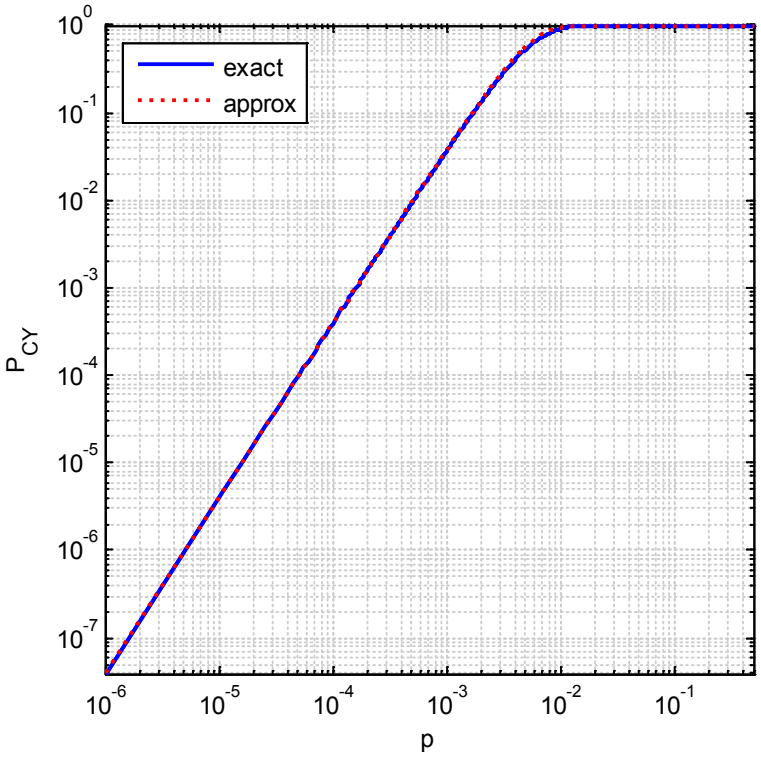

Figure H-5: Exact and Approximate Values of P_{CY} for the (63, 56) BCH Code and $N = 20$

For better clarity, figure H-6 reports the percentage error between the approximate and the exact values of P_{CY}, for $N = 20$. From figure H-6, it can be observed that, in this case, the percentage error is more significant with respect to the TED mode case, especially for not-too-low channel error probabilities. In particular, the error reaches seven percent for p around $5 \cdot 10^{-3}$.

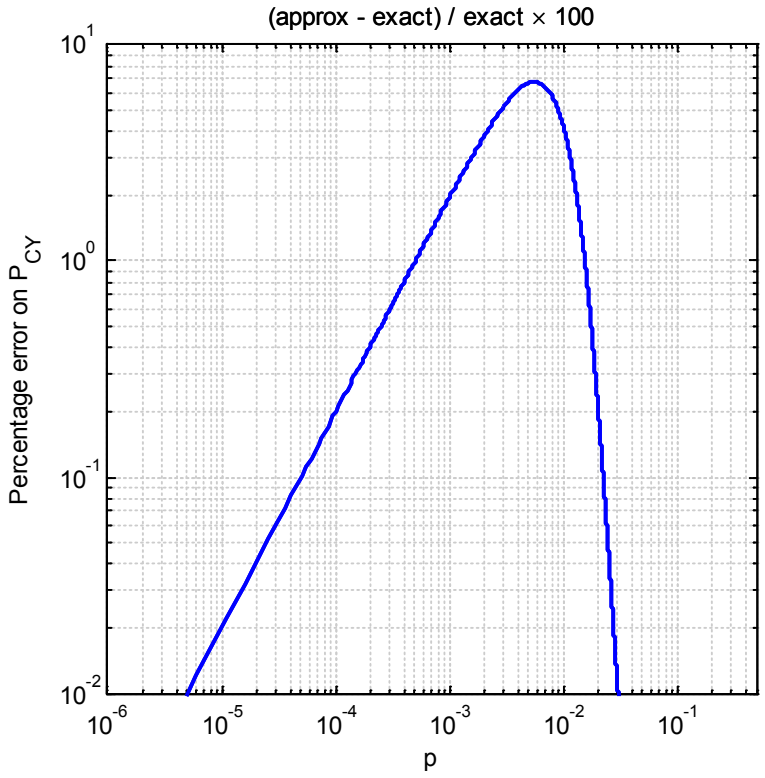

Figure H-6: Percentage Error between the Approximate and the Exact Values of P_{CY} for the (63, 56) BCH Code and $N = 20$

H3 WEIGHTS DISTRIBUTION FOR THE (63, 56) BCH CODE

The weights distribution for the (63, 56) BCH code, obtained from the (63, 57) Hamming code by expurgation, is as follows:

i	A_i
0	1
4	9765
6	1057224
8	60544953
10	1996794072
12	41694856749
14	584173436400
16	5724932809365
18	40448633569680
20	210758816714985
22	823875120414360
24	2447745309517725
26	5580858785942664
28	9832942289229633
30	13449656041565856
32	14317376396958243
34	11867343566087520
36	7647844002734159
38	3818482327223928
40	1468647185710635
42	431553634502760
44	95799462143175
46	15827726179440
48	1908310936455
50	163568562192
52	9621890019
54	369776680
56	8649279
58	109368
60	651