



*Deep Space Network*

# 208

## Telemetry Data Decoding

---

Document Owner:

Approved by:

Signature Provided                      01/09/2013

Signature Provided                      01/09/2013

\_\_\_\_\_  
Andrew O'Dea                              Date  
DSN TT&C System Engineer

\_\_\_\_\_  
Timothy. T. Pham                              Date  
DSN Communications Systems Chief Engineer

Released by:

Signature Provided                      01/09/2013

\_\_\_\_\_  
DSN Document Release                      Date

DSN No. **810-005-208, Rev. B**  
Issue Date: January 10, 2013  
JPL D-19379

**Jet Propulsion Laboratory**  
California Institute of Technology

*Users must ensure that they are using the current version in DSN Telecommunications Link Design Handbook  
website: <http://eis.jpl.nasa.gov/deepspace/dsndocs/810-005/>*

© <2013> California Institute of Technology.  
U.S. Government sponsorship acknowledged.

## **Review Acknowledgment**

*By signing below, the signatories acknowledge that they have reviewed this document and provided comments, if any, to the signatories on the Cover Page.*

Signature Provided

12/06/2012

---

Jeff Berner  
DSN Project Chief Engineer

Date

## Document Change Log

Rev	Issue Date	Prepared By	Affected Paragraphs	Change Summary
Initial	11/30/2000	Robert Sniffin	All	Initial Release
Chg 1	03/31/2004	Robert Sniffin	Figures 3, 17	Corrects third generator polynomial hexadecimal representation in Figure 3. Removes implication in Figure 17 that Turbo code performance can be extrapolated below FER=10 <sup>-5</sup>
A	05/18/2009	Robert Sniffin	All	Deletes obsolete code types. Provides additional information on supported codes. Adds LDPC codes as a proposed capability
B	01/10/2013	Christine Chang Robert Sniffin	Table 1 Table 3 Table 5 Figure 21	<p>Corrected the specification of the connection vectors for the (7, 1/2) convolutional code in Table 1.</p> <p>Added decoder throughput for the 26 GHz signal path in Table 1.</p> <p>Changed codeblock length from 17949 (rate 1/2) to 17848 (per CCSDS 131.0-B-1 "CCSDS TM Sync and Channel Coding") in Table 3.</p> <p>Changed codeblock length from 76772 (rate 1/3) to 26772 (per CCSDS 131.0-B-1 "CCSDS TM Sync and Channel Coding") in Table 3.</p> <p>Removed distinction of Turbo decoding in Table 5.</p> <p>Added Turbo rate 1/2 codes in Figure 21.</p>

## *Contents*

<b><u>Paragraph</u></b>	<b><u>Page</u></b>
1 Introduction.....	6
1.1 Purpose.....	6
1.2 Scope.....	6
2 General Information.....	6
2.1 Telemetry Waveforms.....	6
2.2 Symbol Transition Density .....	8
2.3 BPSK, QPSK and SQPSK .....	9
2.4 Symbol Quantization .....	9
2.5 Forward Error Correcting Codes.....	10
2.5.1 Convolutional Codes.....	11
2.5.2 Frame Synchronization .....	15
2.5.2.1 Bit Domain Frame Synchronization .....	15
2.5.2.2 Symbol Domain Frame Synchronization .....	17
2.5.3 Randomization and De-randomization .....	19
2.5.4 Reed-Solomon Code .....	20
2.5.4.1 Reed-Solomon Encoder.....	20
2.5.4.2 Concatenated Convolutional and Reed–Solomon Code .....	20
2.5.4.3 Interleaving.....	20
2.5.5 Virtual Fill.....	23
2.5.6 Turbo Codes.....	25
2.5.6.1 Turbo Code Encoder.....	26
2.5.6.2 Turbo Code Decoder .....	28
2.5.7 Code Performance.....	29
2.6 Time Tagging.....	31
2.7 Data Formatting .....	32
2.8 Supported Telemetry Configurations.....	31
3 Proposed Capability .....	34
3.1 Low-Density Parity-check (LDPC) Codes.....	36
References.....	36

## *Illustrations*

<b><u>Figure</u></b>	<b><u>Page</u></b>
Figure 1. Telemetry Modulation Waveforms.....	8
Figure 2. Quantization Effects on Decoder Performance .....	10
Figure 3. $k=7$ , $r=1/2$ Convolutional Encoder Connection Vector Schematics.....	12
Figure 4. CCSDS Recommended 32-bit Attached Synchronization Marker.....	15
Figure 5. Attached Synchronization Markers for Turbo Codes .....	18
Figure 6. CCSDS Pseudo-Randomizer/De-randomizer.....	19
Figure 7. Berlekamp Architecture Reed-Solomon (255, 223) Encoder.....	21
Figure 8. Effect of Interleaving on RS Performance.....	22
Figure 9. Reed-Solomon Symbol Arrangement for Interleave Factor ( $I$ ) of 5 .....	24
Figure 10. Illustration of Virtual Fill .....	25
Figure 11. CCSDS Turbo Encoder .....	27
Figure 12. Turbo Code Structure in the Physical Channel.....	27
Figure 13. Cyclic Redundancy Check Generator.....	28
Figure 14. Relative Performance of Supported Codes.....	30
Figure 15. Measured Performance of DSN Turbo Decoder Showing Improvement with Code Rate and Error Floor Effects (Block Size = 8920 Bits).....	31
Figure 16. Example of Telemetry Data Flow Using Virtual Channels. ....	33
Figure 17. Spacecraft and Ground Configuration for BPSK Reed-Solomon, Convolutional, and Concatenated Coding. ....	34
Figure 18. Spacecraft and Ground Configuration for BPSK Turbo Coding.....	34
Figure 19. Spacecraft and Ground Configuration for QPSK/SQPSK Convolutional and Concatenated Coding. ....	35
Figure 20. Spacecraft and Ground Configuration for QPSK/SQPSK Turbo Coding.....	35
Figure 21. LDPC and Turbo Code Comparative Performance .....	37

## *Tables*

<b><u>Table</u></b>	<b><u>Page</u></b>
Table 1. Convolutional Decoder Characteristics .....	13
Table 2. Bit Domain Frame Synchronization Parameters.....	15
Table 3. Turbo Code Information Block and Codeblock Lengths .....	26
Table 4. DSN Turbo Decoder Characteristics .....	29
Table 5. DSN Time Tagging.....	32
Table 6. Codeblock Lengths for LDPC Code Rates .....	37

## ***1 Introduction***

### ***1.1 Purpose***

This module describes the capabilities and performance of the telemetry decoding and frame synchronization equipment used by the Deep Space Network (DSN) in order to assist the telecommunications engineer in designing compatible spacecraft equipment.

### ***1.2 Scope***

The detailed discussion in this module is limited to the performance of equipment that is currently installed at the Deep Space Communications Complexes (DSCCs) and performs data extraction in real time. Additional factors that affect telemetry performance such as imperfect residual or suppressed carrier synchronization (radio loss), imperfect subcarrier and symbol synchronization, and waveform distortion are discussed in module 207.

## ***2 General Information***

Extracting data from spacecraft return link telemetry includes those processes that convert radio frequency energy into one or more bit or symbol streams (discussed in module 207) and those processes that convert the received symbol stream to a replica of the data collected onboard the spacecraft that are discussed in this module. Throughout this module, the term *bit* is used to represent the smallest unit of user data and the term *symbol* is applied to what is transmitted through the communications channel.

### ***2.1 Telemetry Waveforms***

All modern spacecraft utilize pulse code modulation (PCM) to transfer binary data between the spacecraft and the mission operations. The data are phase-modulated onto an RF carrier (PCM/PM) or used to switch the phase of a subcarrier by plus or minus 90-degrees. The subcarrier is then phase modulated on the carrier for transmission via the space link. This modulation scheme is referred to as PCM/PSK/PM. Phase modulation is used because it has a constant envelope that enables non-linear amplifiers to be used. Non-linear amplifiers tend to be more efficient than the linear amplifiers that would be necessary if the envelope (amplitude) were used to carry information. Phase modulation is also immune to most interference that corrupts signal amplitude.

Although these techniques are referred to as pulse code modulation, they do not use pulses in the conventional sense. They use non-return to zero waveforms that can be envisioned as a pulse starting as a transition from a zero voltage to some other voltage and not returning to zero until something happens. That “something” determines the characteristics and

name of the waveform. The simplest case is referred to as non-return to zero-level (NRZ-L) where the cause of the waveform returning to zero is the bit stream level changing from a one to a zero. Thus, the modulation waveform matches the data waveform. This is the most common modulation waveform used but suffers from the problem that it is impossible to tell which of the two levels is a one and which is a zero. It also has the problem that a long string of zeros or ones will prevent phase transitions from occurring that are necessary to keep the receiver symbol synchronizer in-lock.

The first of these problems can be solved by a technique called differential encoding. There are two differential encoding waveforms referred to as non-return to zero-mark (NRZ-M) and non-return to zero-space (NRZ-S). In the first case, the modulating waveform changes whenever the input is a “one” bit (or “mark” from teletypewriter terminology) and remains the same whenever the input is a “zero” bit (or “space” again, from teletypewriter terminology). The second is the opposite. The waveform changes whenever a “zero” bit occurs and remains the same whenever a “one” bit occurs. These waveforms enable the data polarity to be determined even if the detected waveform is inverted. However, a failure to properly detect a bit will always result in second error as the waveform restores itself. This causes an increase in error rate and these waveforms are not normally used for deep space communication where there is another method of determining waveform polarity (see the discussion of synchronization markers) and anything that unnecessarily increases error rate is unacceptable. Similar to NRZ-L, a long string of zeros when using NRZ-M or ones when using NRZ-S will suppress the phase transitions necessary to keep the receiver symbol synchronizer in lock

There are three other PCM modulating waveforms that have been used for spacecraft communication and solve the phase transition problem. These are bi-phase waveforms where every bit interval contains at least one phase transition. There is one bi-phase waveform corresponding to each NRZ waveform and they are referred to as Bi- $\phi$ -L, Bi- $\phi$ -M, and Bi- $\phi$ -S. Bi- $\phi$ -L is also referred to as Manchester or Split-phase encoding. The result of including a transition in every bit interval is to convert the transmitted spectrum to double-sideband having a total bandwidth twice that of the NRZ waveforms with the peak of each sideband at the bit rate and no energy at the center frequency. This leaves room for a residual carrier that can be detected with minimum interference from the data it carries. However, the increased bandwidth requirements of bi-phase modulation has normally limited its application to forward links where bandwidth requirements are less, the frequency of phase transitions permit a very simple (and low mass) symbol synchronizer onboard the spacecraft, and a residual carrier is useful for metric data measurements.

Figure 1 depicts the six telemetry waveforms discussed above. An inverted NRZ-M waveform is also included to illustrate its immunity to inversion. It is important to remember that these waveforms are not part of the telemetry encoding and decoding schemes that are used to improve telemetry performance. As mentioned above, differential encoding somewhat reduces telemetry performance. The DSN can receive any of these waveforms but uses hardware algorithms to convert them to NRZ-L either as part of the decoding process or before delivery to the customer.

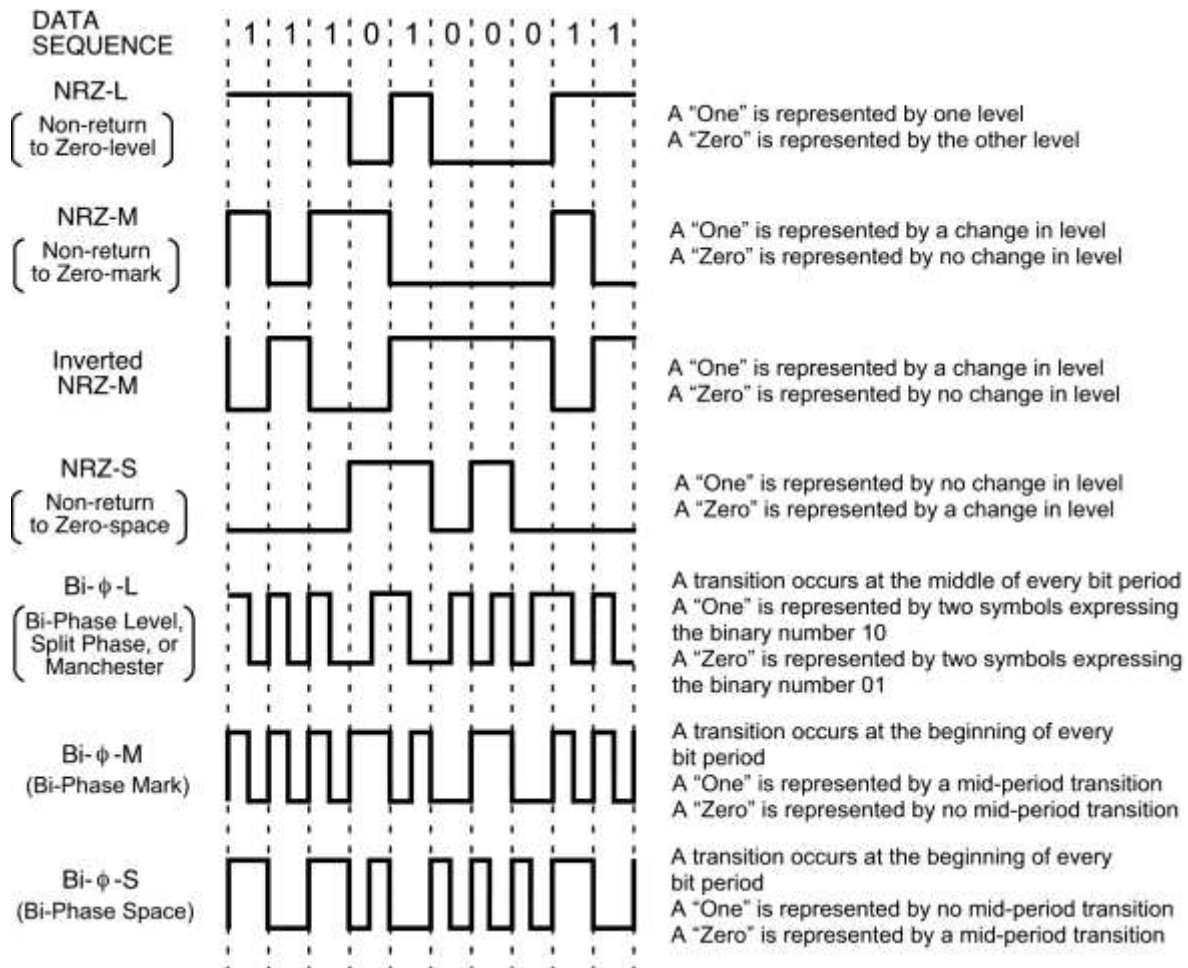


Figure 1. Telemetry Modulation Waveforms

## 2.2 Symbol Transition Density

The DSN derives symbol timing by observing successive phase transitions in the detected NRZ waveform and refining this estimate from additional phase transitions. This process requires that the received waveform have an adequate symbol transition density despite the nature of NRZ waveforms to produce long periods without transitions when delivering certain bit sequences. Transition density is defined as twice the probability that a symbol will be a one multiplied by the probability that it will be a zero. As, for truly random data both of these probabilities are 0.5, the transition density can have a value between 0 and 0.5. The DSN recommends that the transition density be between 0.25 and 0.5 (See Module 207) with the additional constraint that NRZ waveform has at least one phase transition every 64 symbol periods. It is the responsibility of the telecommunications designer to ensure that sufficient phase transitions are present in the transmitted data to maintain symbol synchronization. Several techniques for increasing the transition density are discussed below.



### 2.3 *BPSK, QPSK and SQPSK*

The binary NRZ waveform is used to shift the phase of the transmitted carrier or subcarrier in equal amounts from its rest phase. If the amount of phase shift is  $\pm 90$  degrees, the carrier or subcarrier is fully suppressed and the modulation is referred to Binary Phase Shift Keying (BPSK). BPSK results in a 180-degree phase reversal at each NRZ waveform transition. When PCM/PSK/PM is being used, the phase transitions are normally synchronized with subcarrier zero-crossings. No such synchronization is attempted between the carrier and the data for PCM/PM.

The capacity of the communications channel can be doubled by splitting the data stream into two parts consisting of alternate symbols from the input data stream. These two parts are used to BPSK modulate carriers that are in phase quadrature with each other and the two modulated carriers are summed for transmission. This technique is referred to as Quadrature Phase Shift Keying (QPSK). Like BPSK, QPSK can produce 180-degree phase reversals when the two modulated carriers change phase at the same time. A 90-degree phase change will occur when only one of the two carriers changes phase.

Phase reversals of 180 degrees can be completely eliminated by employing Staggered QPSK (SQPSK), also referred to as Offset QPSK (QPSK). In SQPSK, one of the two waveforms is delayed by  $1/2$  symbol period so that simultaneous phase transitions never occur and the greatest phase change in the transmitted waveform will be 90 degrees. SQPSK results in less degradation than QPSK in a bandwidth-limited channel.

The Consultative Committee for Space Data Systems (CCSDS) recommends that when a single data stream is being transferred, the stream be separated so that alternate symbols are transmitted on the two quadrature channels. The DSN supports this modulation format for all frequency bands and all supported data rates. The DSN also supports a modulation scheme for use in the near-Earth 26 GHz allocation at data rates in excess of 10 Mbps where the data stream is split into alternate bit streams, each bit stream is convolutionally coded, and the two symbol streams are delivered to the QPSK modulators. Upon reception, the two streams are separately decoded and then combined to recover the original data stream.

### 2.4 *Symbol Quantization*

Convolutional and Turbo codes, discussed below under Forward Error Correcting Codes, use decoding algorithms that are able to take into consideration not only that a symbol has been detected to be a one or a zero but also that a symbol is more likely to be a one than a zero. The DSN receivers produce symbol values (referred to as *soft symbols*) that are quantized as 8-bit values however the standard convolutional decoder only accepts 3-bit quantization. A mapping is provided at the input of the convolutional decoder to convert the 8-bit values to 3-bits. Figure 2 shows the effects of symbol quantization on convolutional decoder performance. This figure is included to illustrate the need to perform any conversion between NRZ-L and differential encoding prior to convolutional encoding and after convolutional decoding as the DSN does not include a decoder for differential NRZ waveforms nor a method of converting from differential

waveforms to NRZ-L without simultaneously converting them to one-bit quantization (*hard symbols*) which would result in a significant performance loss.

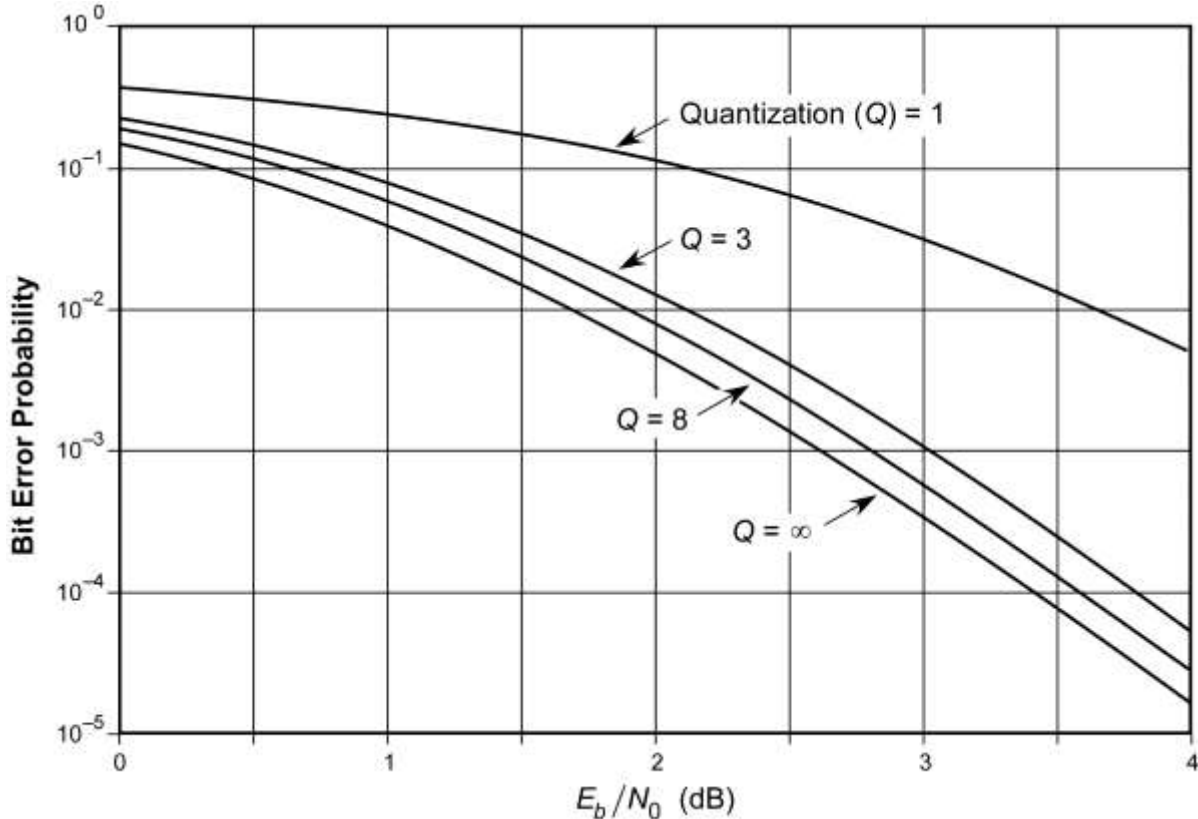


Figure 2. Quantization Effects on Decoder Performance

## 2.5 Forward Error Correcting Codes

Almost all spacecraft employ forward error correcting (FEC) codes to make more efficient use of the communications channel. Forward error correcting codes add additional symbols to the transmitted data stream that the decoder can use to improve its estimate of the encoded bit stream. The exceptions to FEC use would likely be extremely high data rate transmissions where adequate signal power is available to make the gain achieved by coding unnecessary and any bandwidth needed for the symbols added by coding is unavailable.

The DSN supports two convolutional codes, the Consultative Committee for Space Data Systems (CCSDS) standard Reed-Solomon code, and the CCSDS Turbo codes. Convolutional codes are used because they achieve significant coding gain with simple, highly reliable encoders and their decoders are of reasonable complexity. They also provide low latency and are useful when conditions may prevent a block of symbols from being received. The Reed-Solomon code provides excellent performance with minimum bandwidth expansion in a high signal-to-noise environment. It is most often used as an outer code in combination with a

convolutional inner code but may be used by itself under appropriate signal conditions. Turbo codes provide near-Shannon-limit error-correction performance with reasonable encoding and decoding complexity. The DSN presently includes an additional convolutional decoder that is used for the Cassini spacecraft support but it will be removed from service at the end of that mission.

### 2.5.1 Convolutional Codes

Convolutional codes are specified by their *constraint length* ( $K$ ) and *rate* ( $r$ ). Constraint length is the number of sequential input bits required to define the output symbols at any point in time. Rate is the number of data bits with respect to the number of coded symbols expressed as a fraction. In general, the performance of a convolutional code increases directly with  $k$  and inversely with  $r$ , but codes must be selected carefully because the channel bandwidth also varies inversely with  $r$  and decoder complexity increases exponentially with  $k$ .

The most common convolutional code is the CCSDS  $k=7, r=1/2$  ( $7, 1/2$ ) code. This code falls into a category referred to as *transparent* codes meaning that if the input to the encoder or decoder is inverted, the output will be inverted. Thus, the phase ambiguity associated with BPSK modulation does not need to be resolved until the coding gain is achieved. A convolutional encoder consists of a  $k$ -stage shift register with the outputs of selected stages connected by  $r$  exclusive-OR connection vectors. The  $r$  outputs (in this case 2) are transmitted alternately through the communications channel. The recommended code inverts the output of one of the two connection vectors which ensures that sufficient transitions will be available to keep the receiver symbol synchronizer in lock. A diagram of the CCSDS ( $7, 1/2$ ) code is shown in Figure 3. Figure 3 also shows a variation of this code used on some legacy deep space missions. The only difference between the two codes is the order in which symbols from the two connection vector outputs are transmitted. The DSN can decode either variation with or without the alternate symbols being inverted. The capabilities of the DSN convolutional decoder are summarized in Table 1.

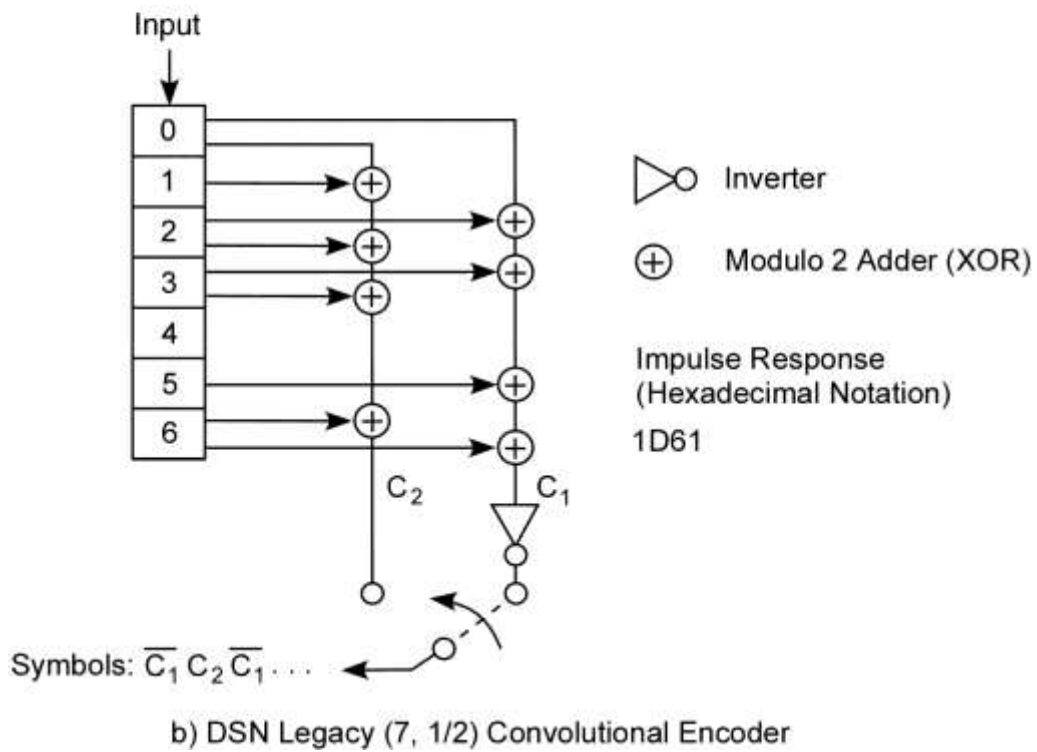
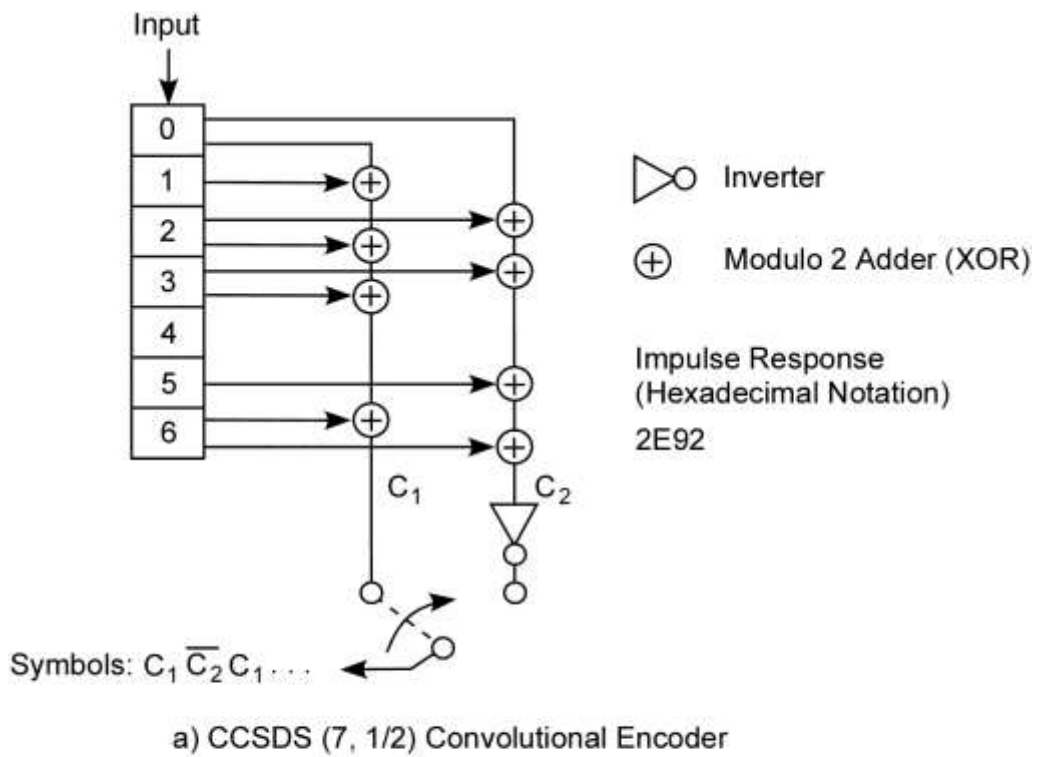


Figure 3.  $k=7, r=1/2$  Convolutional Encoder Connection Vector Schematics

Table 1. Convolutional Decoder Characteristics

Parameter	Value
Constraint length	7
Code rate	1/2
Connection vectors*	C1 = 1111001, C2 = 1011011 or C1= 1011011, C2 = 1111001
Alternate symbol inversion**	Selectable
Input quantization**	3 bits (8 levels)
S, X, and Ka 32 GHz Symbol (Input) rate	4 s/s to 13.2 Ms/s (max.)
S, X, and Ka 32 GHz Bit (Output) rate	6.6 Mb/s (max.)
Node synchronization**	Symbol Error Rate or Metric Normalization Rate
Node sync acquisition**	≤ 5000 bit times for $E_b/N_0 \geq 0.5$ dB (99% probability)
Performance vs. theoretical** (for 3-bit quantization)	≤ 0.05 dB
Ka 26 GHz Symbol (Input) Rate	The $(7, \frac{1}{2})$ Viterbi decoder input rate is limited to 240 Ms/s. Two Viterbi decoders can be used in parallel for QPSK/OQPSK to allow coded data up to 300 Ms/s (The 300 Ms/s limit comes from the demodulator, not the decoder.).***  Modulation is limited to 150 Ms/s (BPSK) or 300 Ms/s (QPSK/OQPSK).
Ka 26 GHz Bit (Output) Rate	Output bit rate is limited to: <ul style="list-style-type: none"> <li>• 75 Mb/s for Viterbi-encoded BPSK</li> <li>• 150 Mb/s for uncoded BPSK</li> <li>• 120 Mb/s for single Viterbi-encoded QPSK/OQPSK</li> <li>• 150 Mb/s for dual Viterbi-encoded QPSK/OQPSK</li> <li>• 300 Mb/s for uncoded QPSK/OQPSK</li> </ul>

Note:

\* DSN currently only supports the first configuration (CCSDS convolutional encoder) for Ka 26 GHz

\*\* Except for Ka 26 GHz reception.

\*\*\* The limitation of the upper range of the input rate is on how fast an instance of the convolutional decoder can decode. The implementation of the  $(7, \frac{1}{2})$  Viterbi decoder runs at a maximum symbol rate of 240 Ms/s. For higher symbol rates, the spacecraft needs to use QPSK/OQPSK and run two separate Viterbi encoders, one on "I" and one on "Q" channel. Each of these Viterbi decoders handles half of the symbols. For example at 300 Ms/s OQPSK, two Viterbi decoders would be running in parallel, with each decoder handling 150 Ms/s.

A convolutional decoder must establish node synchronization in order to correctly decode the incoming symbols. That is, which symbol of each received symbol pair represents the first symbol that was transmitted. For an  $r = \frac{1}{2}$ , transparent code, there are only two possibilities. The DSN decoder provides two methods for doing this, symbol error rate (SER) node synchronization and metric normalization rate (MNR) control.

The first method relies on the fact that when the decoder is operating properly the probability of the decoder falsely decoding a bit is at least two orders of magnitude less than the probability of a channel symbol error. The output can therefore be re-encoded and the resultant symbols compared with a delayed copy of the received symbols (to account for decoder delay). The number of differences between these two symbol streams will be an almost true count of the number of symbol errors received by the decoder. The maximum number of symbol errors and the interval over which these symbol errors are counted may be set over the range of 1 to 65535 at decoder initialization. Engineering research suggests that the decoder should obtain proper node sync alignment when the maximum number of symbol errors is set to 420 and the number of decoded bits in which this count is reached is set to 2000 provided the symbol SNR is greater than or equal to  $-2.5$  dB. This same technique of re-encoding the output bits and comparing them to a suitably delayed version of the input symbol stream is used to provide an estimate of the  $E_b/N_0$  with an accuracy of 0.1 dB provided that symbol errors are occurring. Under signal level conditions greater than  $E_b/N_0 = 12$  dB (where there are few symbol errors), the estimate of  $E_b/N_0$  becomes unreliable.

The second method relies on the fact that decoders based on the Viterbi algorithm maintain state metrics that need to be normalized periodically to prevent register overflow. If normalizations are occurring more frequently than a preset interval, the decoder will switch to the alternate node sync and attempt reacquisition. Both the permitted number of normalizations and the interval (as a number of decoded bits) over which this permitted number is accumulated may be set during encoder initialization. The maximum number of normalizations may be set over the range from 4 to 2036, modulo 8 (4, 12, 20, ..., 2036) and the interval used to detect this threshold may be set to the greater of 256 or 1 to 65535, modulo 256 bits. Engineering research suggests that the decoder should obtain proper node sync alignment with the maximum number of normalizations set to 180 and the interval set to 2048 bits provided the symbol SNR is greater than or equal to  $-2.5$  dB.

For extremely low signal-to-noise ratios or if the received symbol stream is invalid, there is a possibility that the decoder will choose the wrong node sync position. If this is detected, the decoder can be commanded to attempt resynchronization but there is no guarantee that the resynchronization will result in the alternate node sync being chosen.

The output stage of the convolutional decoder can be set to perform the conversion to NRZ-L should another telemetry waveform have been employed on the RF channel. The decoder can be operated in a pass-through mode (no decoding) so the waveform conversion capability can be used for data that are not convolutionally coded.

The convolutional decoder presently used for the Cassini spacecraft support is capable of decoding constraint lengths up to  $k=15$  and rates to  $r=1/6$ . As noted earlier, this decoder will be removed at the end of the Cassini project.

### 2.5.2 *Frame Synchronization*

Frame synchronization must be established before processing any block code such as Reed-Solomon or Turbo codes or before formatting the data for delivery. Synchronization is accomplished by preceding each codeblock or transfer frame with a fixed-length Attached Synchronization Marker (ASM). This known bit pattern can be recognized to determine the start of the codeblocks or transfer frames. It also can be used to resolve the phase ambiguity associated with BPSK or QPSK (SQPSK or OQPSK) modulation. The DSN contains two frame synchronizers. The first of these operates in the bit domain and is used with convolutionally coded, Reed-Solomon coded, or uncoded data. The second operates in the symbol domain and is used with Turbo coded data.

#### 2.5.2.1 *Bit Domain Frame Synchronization*

The Consultative Committee for Space Data Systems has adopted the 32-bit ASM shown in Figure 4 for synchronization in the bit domain. The pattern is represented in hexadecimal as 1ACFFC1D but any pattern having a length of 8 to 64 bits such as the Inter-range Instrumentation Group (IRIG) patterns can be accommodated.

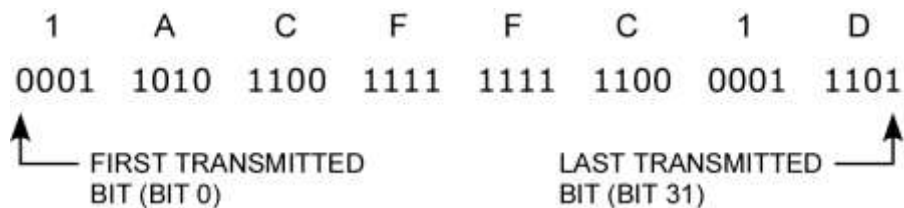


Figure 4. CCSDS Recommended 32-bit Attached Synchronization Marker

The DSN bit-domain frame synchronizer operation is defined by four operating modes: Search, Verify, Lock, and Flywheel. Parameters that affect the operation of the synchronizer are discussed in the following paragraphs and summarized in Table 2.

Table 2. Bit Domain Frame Synchronization Parameters

Parameter	Value
Frame length	8 – 65536 bits in multiples of 8
ASM length	8 – 64 bits
ASM search direction	Forward, Reverse, or Both
Bit-slip window	0 to 3 bits
In-lock bit error tolerance (permissible ASM bit errors while achieving lock)	0 to 31 bits
Number of verify frames	0 to 31
Automatic polarity correction	Enable or Disable
Out-of-lock bit error tolerance (permissible ASM bit errors while in-lock)	0 to 31 bits
Maximum flywheel frames	0 to 31
Maximum time to achieve lock	4 frames provided $BER \leq 10^{-2}$ (99.6% probability)

In the Search mode, the synchronizer assembles all received bits into blocks of the specified length while it attempts to find a pattern in the data that differs from the known ASM by less than a specified number of bit errors. The specified number of bit errors from the synchronization marker is called the In-lock Bit Error Tolerance (IL\_BET) and can have a value from 0 to 31. It does this for the ASM as specified, the inverse of the ASM as specified and, if requested, both the normal and inverse of the ASM with the bit order reversed. When a suitable pattern is found, the block being assembled is flagged as a short block ending with the assumed sync marker and the subsequent received bits are collected into a new data block of the specified length for delivery to the next step in the telemetry processing process. At this point, the synchronizer advances to the Verify mode. Should an inverse of the ASM have been detected, the polarity of all bits is inverted at they are assembled in the data block. Thus, the ambiguity associated with BPSK modulation is automatically resolved.

In the Verify mode, the synchronizer starts looking for an acceptable ASM a few bit periods (referred to as the bit-slip window) before the specified length of the data block. An “acceptable” marker is one that has no more than IL\_BET bit errors from the one previously detected. Should it find the pattern, it increments a counter towards declaring synchronization to be in-lock. Should it not find the pattern, it places the bits that it expected to be a sync maker at the front of the next data block and reverts to the Search mode until a suitable marker is found. The synchronizer remains in the Verify mode until the required number of sequential frames has been found at which time the synchronizer advances to the Lock mode. This number of frames that must be successfully detected before declaring lock can be set over the range of 0 to 31 with zero meaning that the Verify mode is skipped.



In the Lock mode, the synchronizer continues to examine the data stream for an acceptable ASM within the bit slip window using a bit error tolerance referred to as the Out-of-lock Bit Error Tolerance (OOL\_BET) that can be set independently of IL\_BET over the range of 0 to 31. The synchronizer remains in the Lock mode until no acceptable ASM is detected. Should this occur, the synchronizer places itself in the Flywheel mode.

In the Flywheel mode, the synchronizer discards the received bits that occurred where the ASM was anticipated and continues to place the remaining received bits into blocks of the specified frame size. It will continue this process until from 0 to 31 ASMs have been missed at which point it will switch to the Search mode. Should a frame with less than IL\_BET errors be recognized at the appropriate place and before the maximum number of flywheel frames has occurred, the synchronizer will return to the Lock mode.

### **2.5.2.2      *Symbol Domain Frame Synchronization***

The symbol domain bit synchronizer is part of the DSN Turbo decoder and includes automatic polarity correction to resolve the BPSK phase ambiguity. Although the operation is essentially similar to the bit domain frame synchronizer, the parameters have been optimized through simulations and are not available for user modification.

Synchronization in the symbol domain requires longer synchronization markers because the lack of coding gain before synchronization can result in enough symbol errors occurring during a 32-bit sequence to prevent reliable recognition. In addition, the performance gain that is achieved by increasing the code rate comes at the expense of a further reduction of symbol signal to noise ratio resulting in a further increase in symbol errors. To accommodate these factors, the CCSDS has recommended synchronization markers having a length of 32 symbols divided by the code rate,  $r$ . The recommended CCSDS synchronization markers for Turbo codes are illustrated in Figure 5.

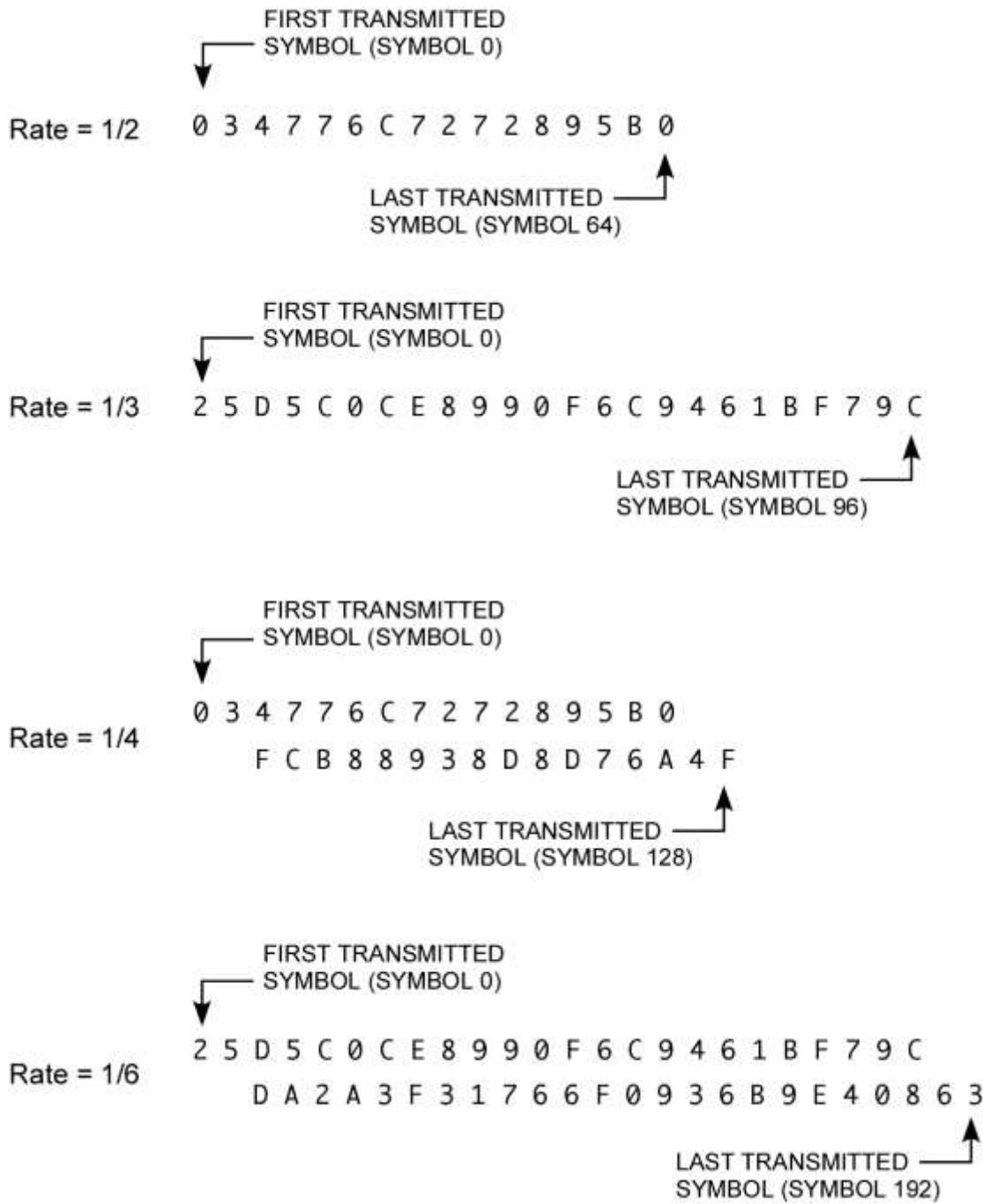
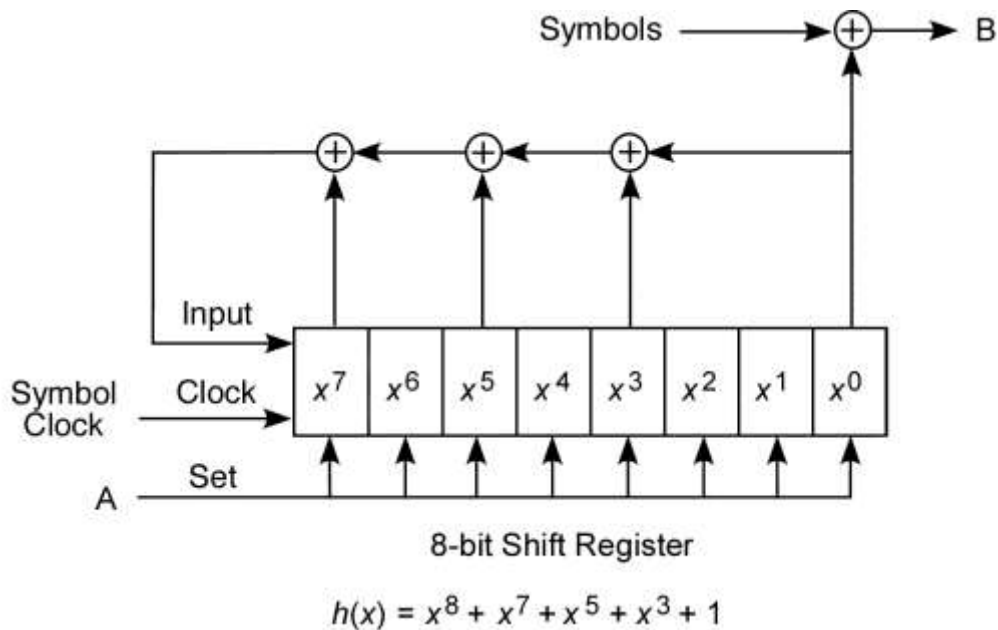


Figure 5. Attached Synchronization Markers for Turbo Codes

### 2.5.3 Randomization and De-randomization

The transition density of data may not be adequate for the receiver to maintain symbol synchronization if the data have not been convolutionally coded or when convolutional coding is used without alternate symbol inversion. This is especially true with NRZ-L uncoded data or when Reed-Solomon coding is used by itself as a sequence of consecutive ones or zeros for some period will provide no transitions.

The required transition density can be achieved for all data streams by modulo-two adding a standard pseudo-random, 255-bit sequence to the stream as it is formed into codeblocks or transfer frames for transmission and then modulo-two adding the same sequence to the received data in the received codeblocks. The code is arranged so that the first bit of the code is added to the first bit in the codeblock or transfer frame and the code is repeated as many times as necessary until the codeblock or transfer frame is completed. The DSN provides the capability to de-randomize uncoded, convolutionally coded, and Reed-Solomon coded data using the CCSDS pseudo-randomizer illustrated in Figure 6.



⊕ Modulo 2 Adder (XOR)

- A Reset each bit to 1 during ASM insertion (transmit) or detection (receive)
- B Randomized Symbols (transmit) or de-randomized symbols (receive)

Figure 6. CCSDS Pseudo-Randomizer/De-randomizer

#### **2.5.4 Reed-Solomon Code**

Reed-Solomon (RS) codes are linear block codes for hard-coded (one-bit digitized) data streams. They are often used in combination with a convolutional inner code that is applied between the point at which the RS coding is complete and the communications channel and then removed prior to RS decoding. In high signal-to-noise environments the RS and convolutional codes can be used independently. The code is *systematic*, meaning that the input bits appear, unchanged, in the output stream followed by parity information that is used by the decoder to correct errors. This property can be useful in forensic analysis of corrupted data. The codes, themselves, are also transparent however, the DSN implementation will always resolve the BPSK phase ambiguity prior to RS decoding. This is important because use of *virtual fill*, described below, renders the code non-transparent.

The RS code adopted by the DSN is one of the two RS codes recommended by the CCSDS and is referred to as the RS (255,223) code. The code divides the input bits into 8-bit sequences to form symbols that are concatenated into a 255 symbol codeword. The RS encoder creates parity symbols from these information symbols that enable the decoder to correct any combination of  $E$  or fewer symbol errors in each codeword. The value  $E$  is referred to as the code redundancy and, for the supported code, has a value of 16. The output of the encoder consists of the 255 information symbols followed by 32 ( $2E$ ) parity symbols. A complete description of this code is contained in references 3 and 4.

##### **2.5.4.1 Reed-Solomon Encoder**

The most common architecture for an RS encoder is named the Berlekamp Architecture, after its inventor. This architecture, in combination with appropriate selection of the RS code generator polynomial, enables parity symbols to be calculated using bit-serial multipliers constructed with a matrix of exclusive OR gates. Figure 7 shows the design of a Berlekamp encoder for producing the DSN/CCSDS standard RS code that includes support for interleaving and virtual fill as discussed below.

##### **2.5.4.2 Concatenated Convolutional and Reed-Solomon Code**

Errors in convolutionally coded channels tend to occur in bursts that result when noise causes the decoder to momentarily follow the wrong path through the decoding trellis. The combination of an outer Reed-Solomon (RS) code with an inner convolutional code provides good burst-error correction with minimal bandwidth expansion.

##### **2.5.4.3 Interleaving**

The burst errors associated with Viterbi decoding can be as long as several constraint lengths and equivalent to several consecutive RS symbols. Thus, several closely spaced error bursts can exceed an RS decoder's error correction capability. Interleaving is a technique that spreads the effects of burst errors across several RS codewords. The effect of interleaving RS coding performance is illustrated in Figure 8.

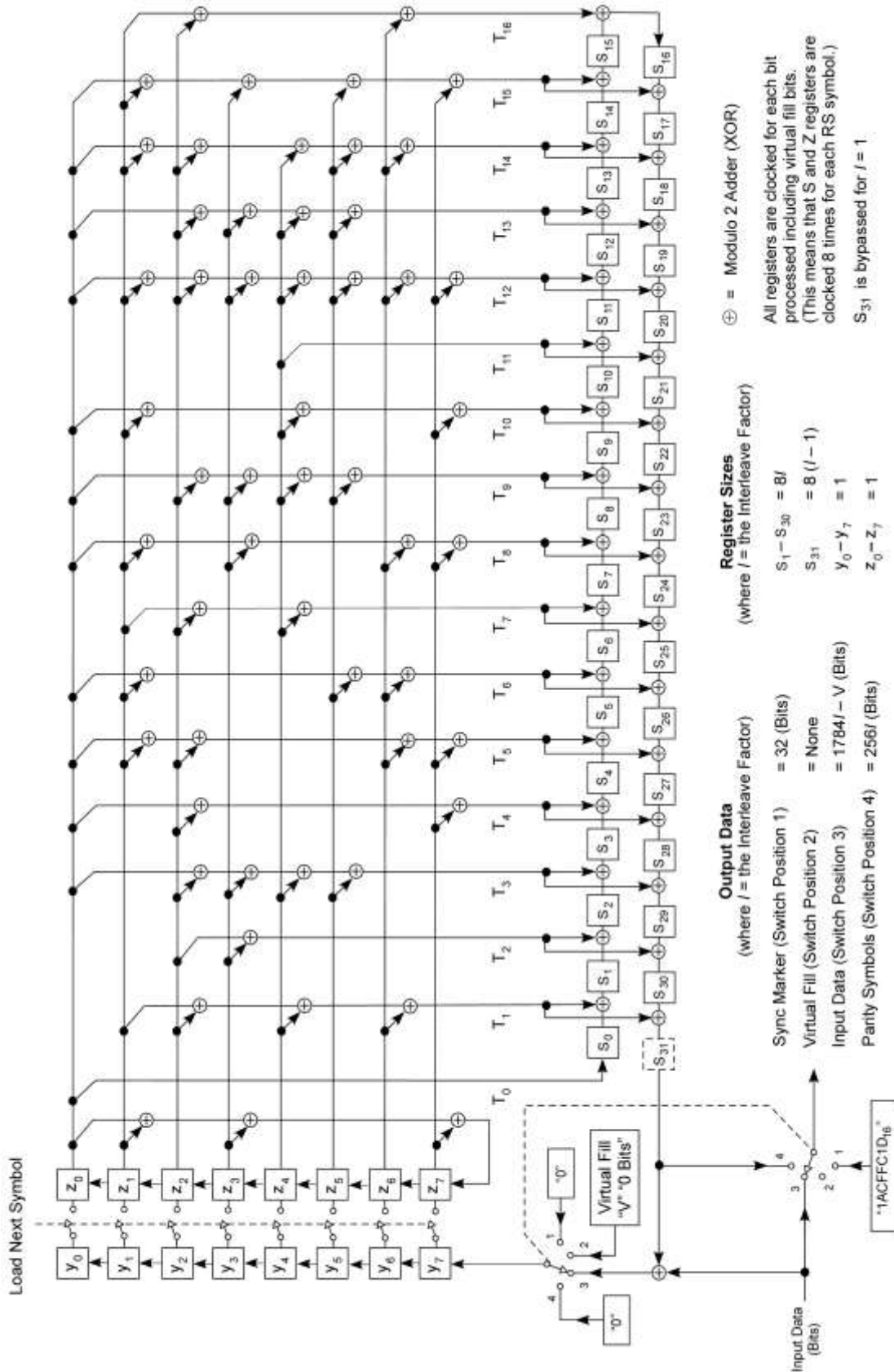


Figure 7. Reed-Solomon Encoder for RS (223, 255) Code

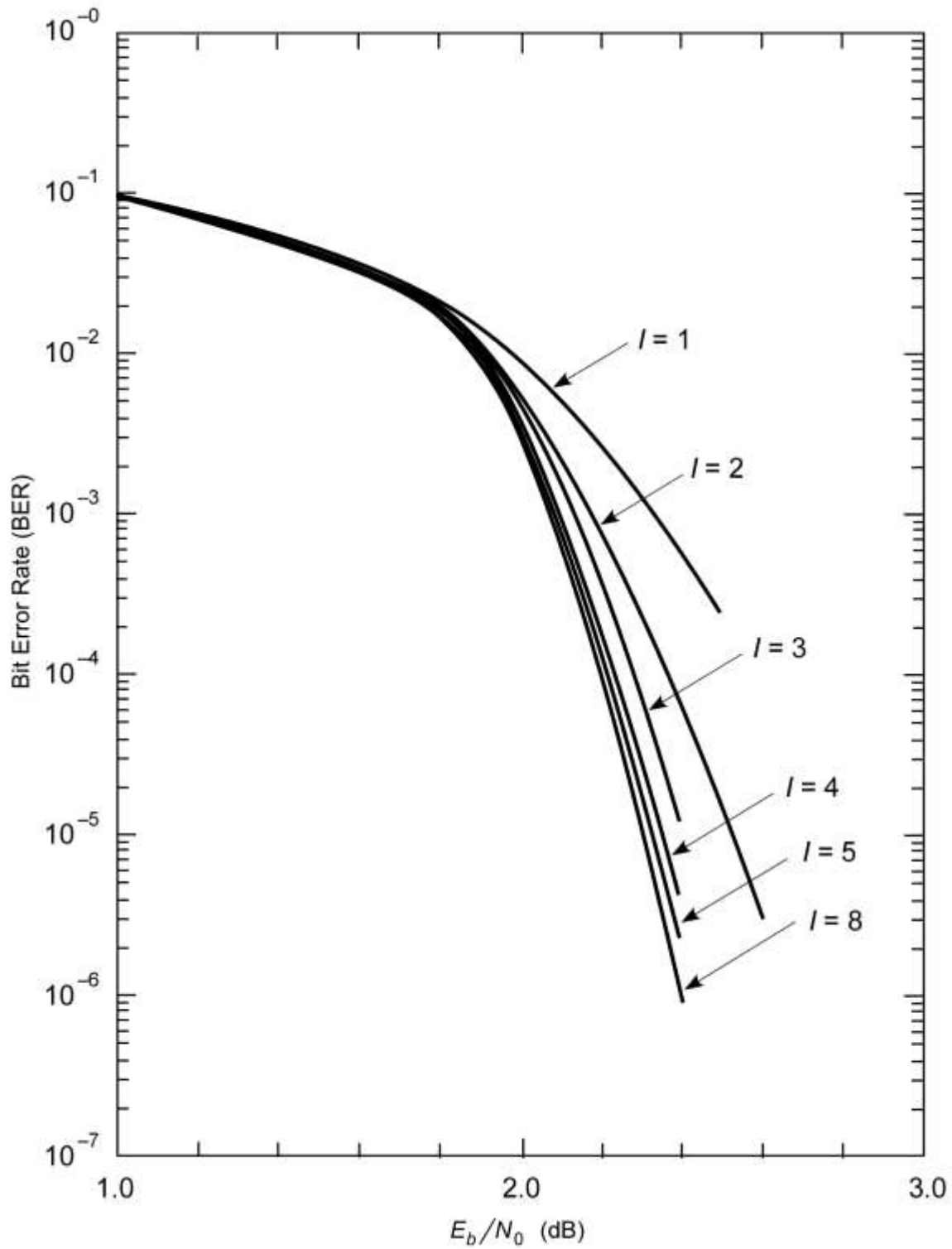


Figure 8. Effect of Interleaving on RS Performance

Interleaving is accomplished by storing partially completed parity symbols in  $31, 8I$ -bit shift registers for parity symbols  $(I - 1)$  through  $32I$  and one  $8(I - 1)$ -bit shift register, where  $I$  is the *interleave factor* so that the parity symbols from any codeblock are not transmitted consecutively. The first 8 bits of input data are collected to form an RS symbol as these bits are being delivered to the convolutional encoder or the information channel. When the symbol is complete, it is transferred into the parity computer that computes the first bit of partial parity “instantaneously” so an output of the parity registers is available for modulo two addition (XOR) with the first bit of the next input symbol. This output will either be the result of the parity calculation if  $I = 1$  or a zero if  $I > 1$ . As the remaining 7-bits of the second symbol are being collected, seven additional bits of partial parity are calculated from the first symbol and pushed into the parity registers resulting in additional bits being supplied for modulo two addition as the input bits are formed into symbols. This process continues until  $223I$  symbols have been processed. When  $I$  symbols have been processed, the output of the parity registers ceases to be the zeroes and each output bit includes the partial parity computed at all prior  $8I$  intervals.

When  $223I$  input symbols have been processed but before the last symbol is transferred to the parity calculation matrix, the input bit stream is set to all zeroes, guaranteeing that there will be no further changes to the collected parity symbols, and the output of the parity register array is connected to the convolutional encoder or the information channel. The last symbol is then processed resulting in the first parity symbol being delivered to the convolutional encoder or the information channel and the remaining symbols are clocked from the array while the array is filled with zeroes in preparation for processing the next codeblock.

Since the input data are passed directly to the convolutional encoder or information channel as the parity symbols are being calculated, the code remains systematic – independent of the interleave factor. The 32 parity symbols from the  $223I$  blocks of information symbols are dispersed across the entire  $32I$  parity symbol portion of the codeblock at  $I$ -symbol intervals. Figure 9 illustrates the symbol arrangement for an interleave factor of 5.

When the data are received, they are written into an array from which the parity symbols associated with each of the  $I$  RS codewords can be separated. DSN supports interleaving for values of  $I$  between 1 (no interleaving) and 8.

### **2.5.5 Virtual Fill**

The maximum amount of input data that can be transmitted in a codeblock varies from 1784 bits (with no interleaving) to 14,272 bits (with an interleaving depth of 8). If a transfer frame has less data than  $1784I$  bits (where  $I$  is the interleave factor), the codeblock can be completed by inserting virtual fill (all-zero RS symbols) between the ASM and the start of the input data. The amount of virtual fill (in units of 8-bits) must be fixed for a tracking pass and is inserted into the parity generator by the encoder and into the received symbol stream before it is decoded however these extra symbols are not transmitted. It is the fact that zeroes are inserted into the received data stream by the decoder that renders the code non-transparent because,

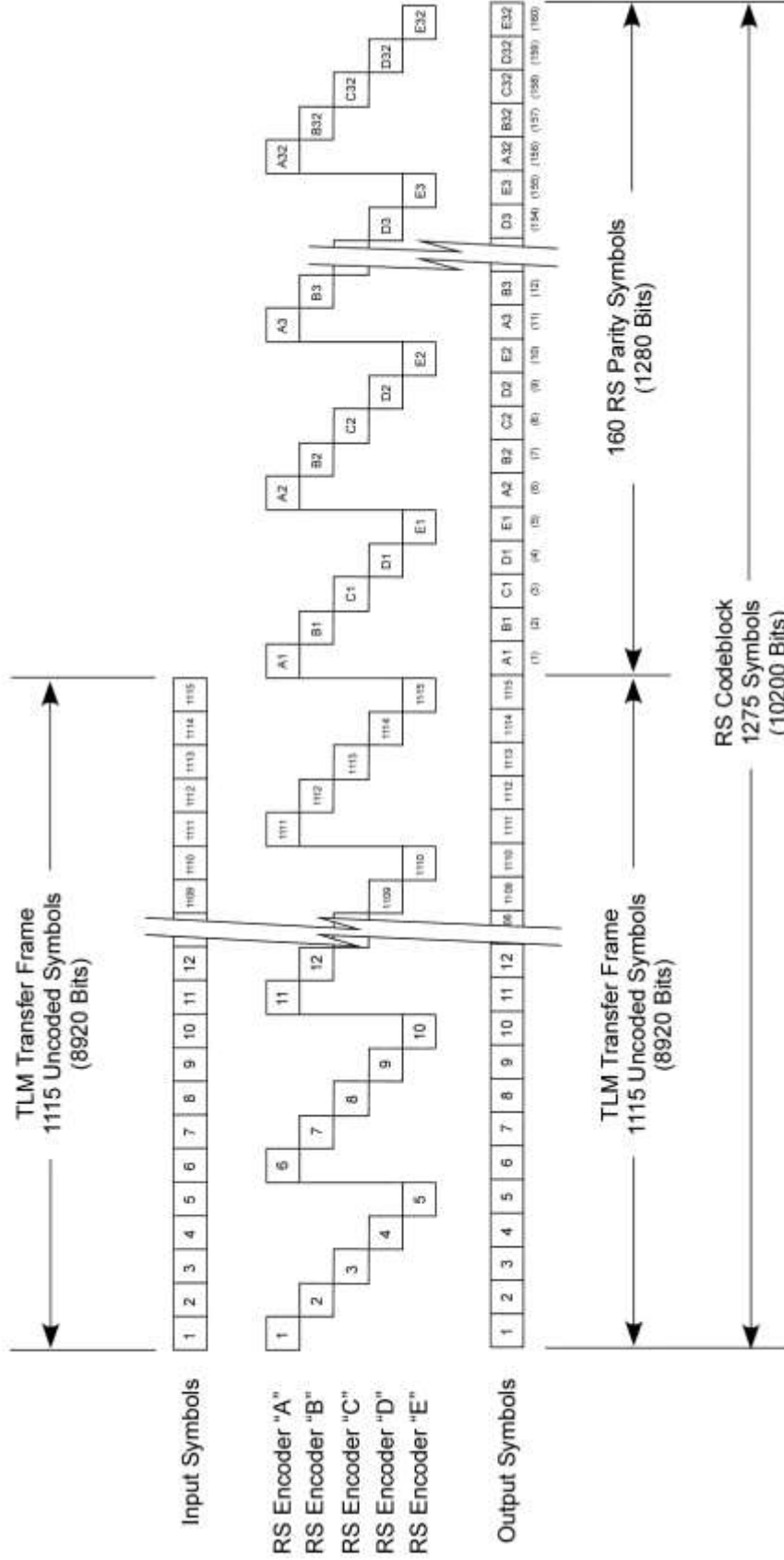


Figure 9. Reed-Solomon Symbol Arrangement for Interleave Factor (I) of 5



should an inversion have occurred, it would be necessary to insert ones instead of zeroes and this cannot be known. The efficiency of RS coding will decrease as the amount of virtual fill increases as the number of parity symbols remains fixed while the number of data symbols decreases. An illustration of virtual fill is shown in Figure 10.

RS Encoder

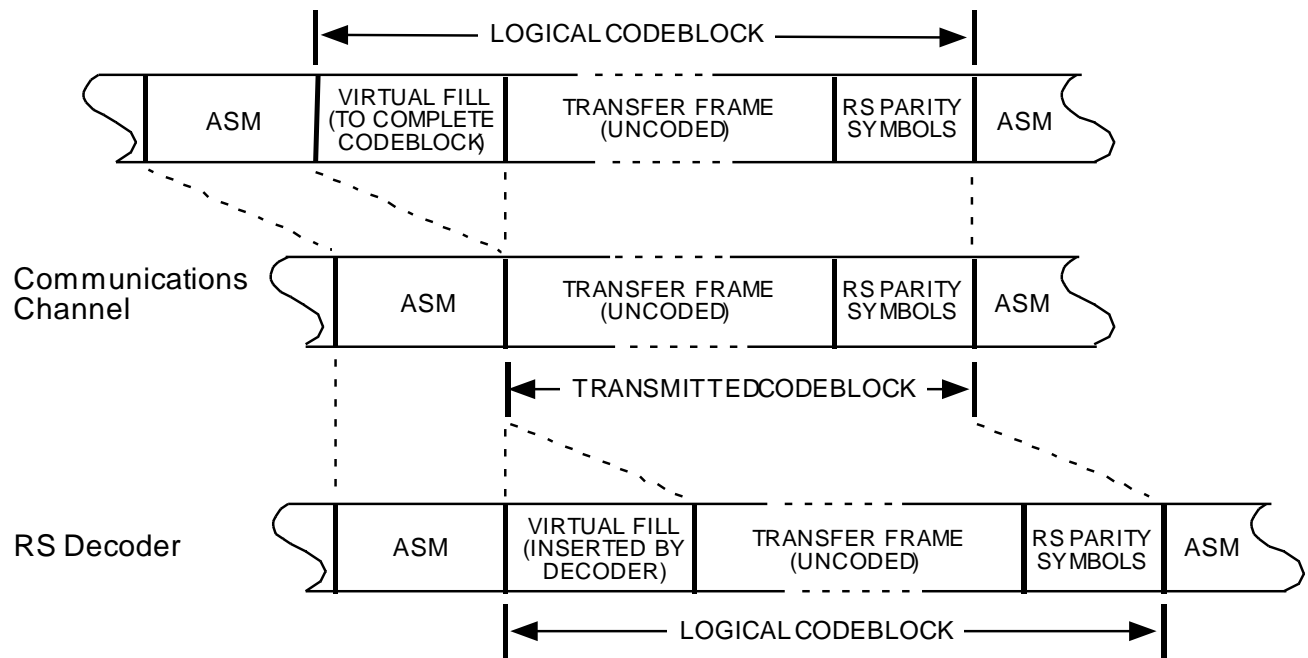


Figure 10. Illustration of Virtual Fill

**2.5.6 Turbo Codes**

Turbo codes provide error correction performance within approximately 0.8 dB of the theoretical limit at a BER of  $10^{-6}$ . This performance is achieved using encoders and decoders of reasonable complexity but at the expense of increased latency. A turbo code is a systematic block code where two sets of parity symbols from independent recursive convolutional encoders are provided. The encoders employ trellis termination so that the codeblock both begins and ends in a known state.

The use of recursive convolutional encoders is one feature of turbo codes. The second is the presence of an interleaver at the input of one of the convolutional encoders that changes the order of the information bits before they are encoded. It is the presence of the interleaver that establishes the minimum latency as equaling the block size as an entire block of data must be assembled before the parity generation process can begin. Although the information bits appear, unchanged, in the encoded output, they do not appear contiguously as is the case with Reed Solomon codes.

The DSN provides support for the turbo code specified in CCSDS Recommendation 131.0-B-1 for information block lengths ( $k$ ) of 1784, 3568, 7136, 8920 bits and nominal code rates ( $r$ ) of  $1/2$ ,  $1/3$ ,  $1/4$ , and  $1/6$ . The recommendation also permits an information block length of 16,384 bits however the encoder for this block length has not been completely specified and it is not supported by the DSN, The four supported block lengths are the same as would be required for Reed-Solomon encoding using an interleave factor ( $I$ ) of 1, 2, 4, or 5.

### 2.5.6.1 Turbo Code Encoder

Figure 11 illustrates the design of a CCSDS compliant turbo encoder. In actual practice, either the entire encoder or the information block buffer and interleaver (with appropriate changes to the input switching) must be duplicated to ensure a constant flow of symbols in the information channel. An actual implementation would also include the capability to preface each codeblock with the synchronization marker described above.

A block of information bits is entered into the information block buffer and the interleaver that stores them in accordance with the permutation algorithm defined by the recommendation. When the buffer and interleaver are full, the information is clocked into the encoders and the resultant symbols are transferred to the information channel in the order shown on the figure. When the last information bit has been transferred into each encoder, the switches at their inputs are placed in position 2 and the encoders permitted to run for four additional clock cycles. This causes four zeros to be entered into the encoders terminating the trellis. The encoder continues to output nonzero encoded symbols during trellis termination producing four extra symbols from the feedback line in addition to the  $k$  information bits.

The presence of the trellis termination symbols results in the channel code rates being slightly smaller than the nominal code rates. The information block and codeblock lengths for the 5 supported turbo codes are shown in Table 3. The structure of the turbo encoded data in the physical channel is illustrated in Figure 12.

Table 3. Turbo Code Information Block and Codeblock Lengths

Information block length, $k$ , bits	Corresponding Reed-Solomon Interleave depth, $I$	Codeblock length			
		Rate $1/2$	Rate $1/3$	Rate $1/4$	Rate $1/6$
1784	1	3576	5364	7152	10728
3568	2	7144	10716	14288	21432
7136	4	14280	21420	28560	42840
8920	5	17848	26772	35696	53544
16384*	N/A	32776	49164	65552	98328

\* Note: This information block length is not supported by the DSN.

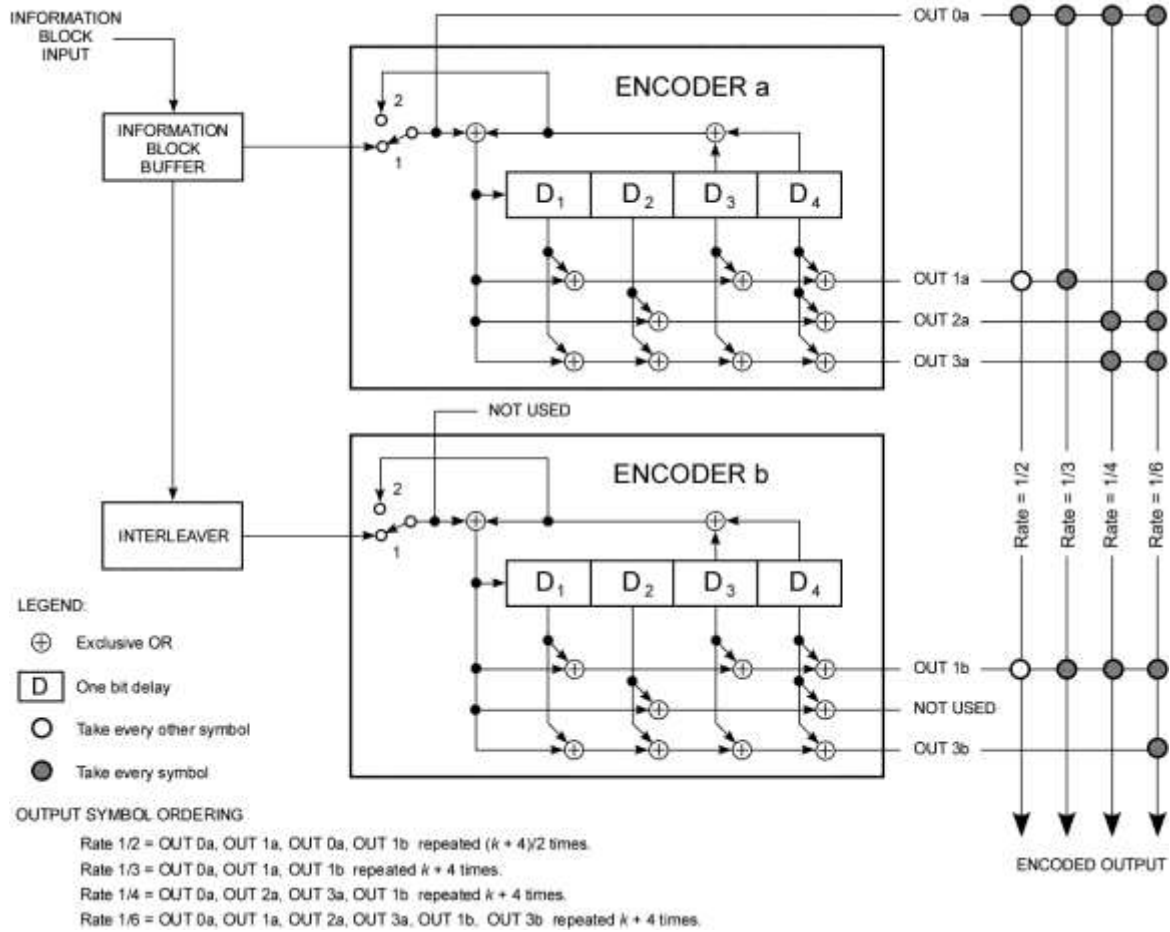
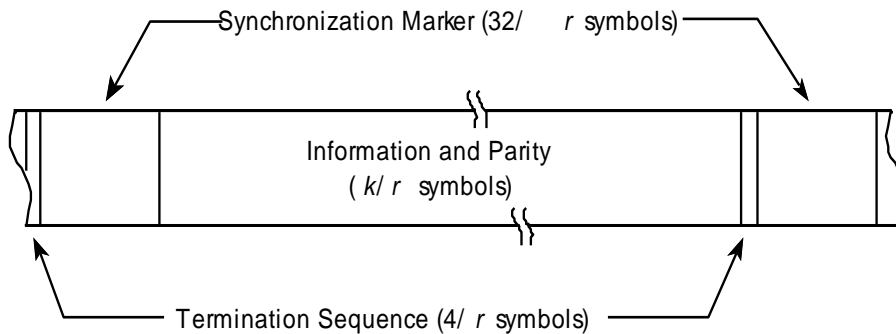


Figure 11. CCSDS Turbo Encoder



$k$  = information block size (1784, 3568, 7136, 8920, or 16384 bits)

$r$  = code rate (1/2, 1/3, 1/4, or 1/6)

Figure 12. Turbo Code Structure in the Physical Channel

### 2.5.6.2 Turbo Code Decoder

Upon recognizing the end of the synchronization marker, the turbo decoder uses a demultiplexer to separate the information symbols from the two sets of parity symbols and direct the information symbols and each of the parity streams into separate decoders. Each decoder makes a Maximum *A posteriori* Probability (MAP) estimate for each bit from the uncoded information symbols (in normal or permuted form, as appropriate) and the parity symbols generated by its corresponding encoder. The decoders exchange their MAP estimates via the appropriate permutation matrix to be used by the opposite decoder as *a priori* estimates for a second iteration. The exchange of MAP estimates continues for a specified number of times or until a satisfactory convergence is reached. Engineering research recommends 10 iterations and values as low as 6 have been successfully used in high data rate applications. The final output is a hard-quantized version of the likelihood estimates from either one of the decoders.

Unlike a Reed-Solomon decoder, there comes a point where a further increase in the  $E_b/N_0$  does not significantly increase a turbo decoder's performance. This region is referred to as the turbo decoder error floor and, for the recommended codes, occurs at a BER of less than  $10^{-7}$ . For operation near this region it is recommended that the data content of each information block be reduced to allow for a cyclic redundancy check (CRC) as an independent check on the decoding process to be inserted at the end of the codeblock. The DSN supports the 16-bit CRC specified in CCSDS Recommendation 132.0-B-1. A diagram of the CRC generator is shown in Figure 13. When CRC checking is enabled, the DSN decoder flags frames that are not successfully decoded but delivers all bits to the user.

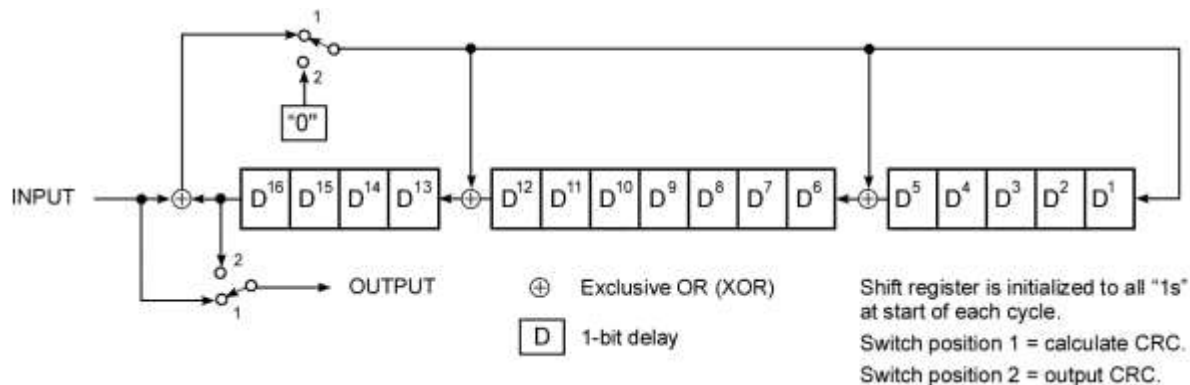


Figure 13. Cyclic Redundancy Check Generator

In addition to the latency required to create turbo-encoded data onboard the spacecraft, the DSN requires time to perform the iterative decoding process. The DSN turbo decoder is actually a set of parallel decoder modules where each module is filled with symbols while previously filled modules are either iterating or delivering their decoded results. The

decoder provides control over the number of iterations performed and a dimensionless convergence confidence threshold normally set at 100. The decoding process is considered complete if the confidence level at the end of an iteration exceeds the selected confidence threshold or if the specified maximum number of iterations is reached. The characteristics of the DSN Turbo Decoder are summarized in Table 4.

Table 4. DSN Turbo Decoder Characteristics

Parameter	Value
Code Supported	CCSDS
Information Block Lengths ( $K$ )	1784, 3568, 7136, 8920
Code Rates ( $r$ )	1/2, 1/3, 1/4, 1/6
ASM patterns	CCSDS compliant
Maximum Input Symbol Rates	Rate 1/2, 3.2 Msps Rate 1/3, 4.8 Msps Rate 1/4, 6.4 Msps Rate 1/6, 6.0 Msps
Number of Iterations	1 to 20 (nominal = 10)
Stopping Rule Threshold	0 (no confidence) to 32767 Nominal value = 100
Cyclic Redundancy Check	CCSDS 16-bit, Optional

### 2.5.7 Code Performance

The performance of a digital communications channel is expressed in the form of an error rate that is a function of the bit energy to noise spectral density ratio  $E_b/N_0$ . The two most common error rates used are the bit error rate (BER) and the frame error rate (FER). The FER, while being often the more significant of the two measures for judging performance, does not lend itself to comparison between code types because of its dependency on the code and the characteristics of the communications channel. On the other hand, BER is easily modeled for the additive white Gaussian noise (AWGN) channel which is a reasonable approximation for the deep space communications channel. Figure 14 provides a comparison of the BER performance for the codes supported by the DSN. Figure 15 shows the measured performance of the DSN Turbo Decoder for the same 8920 bit block size as Figure 14 but showing both the effects of increased code rate and the error floor.

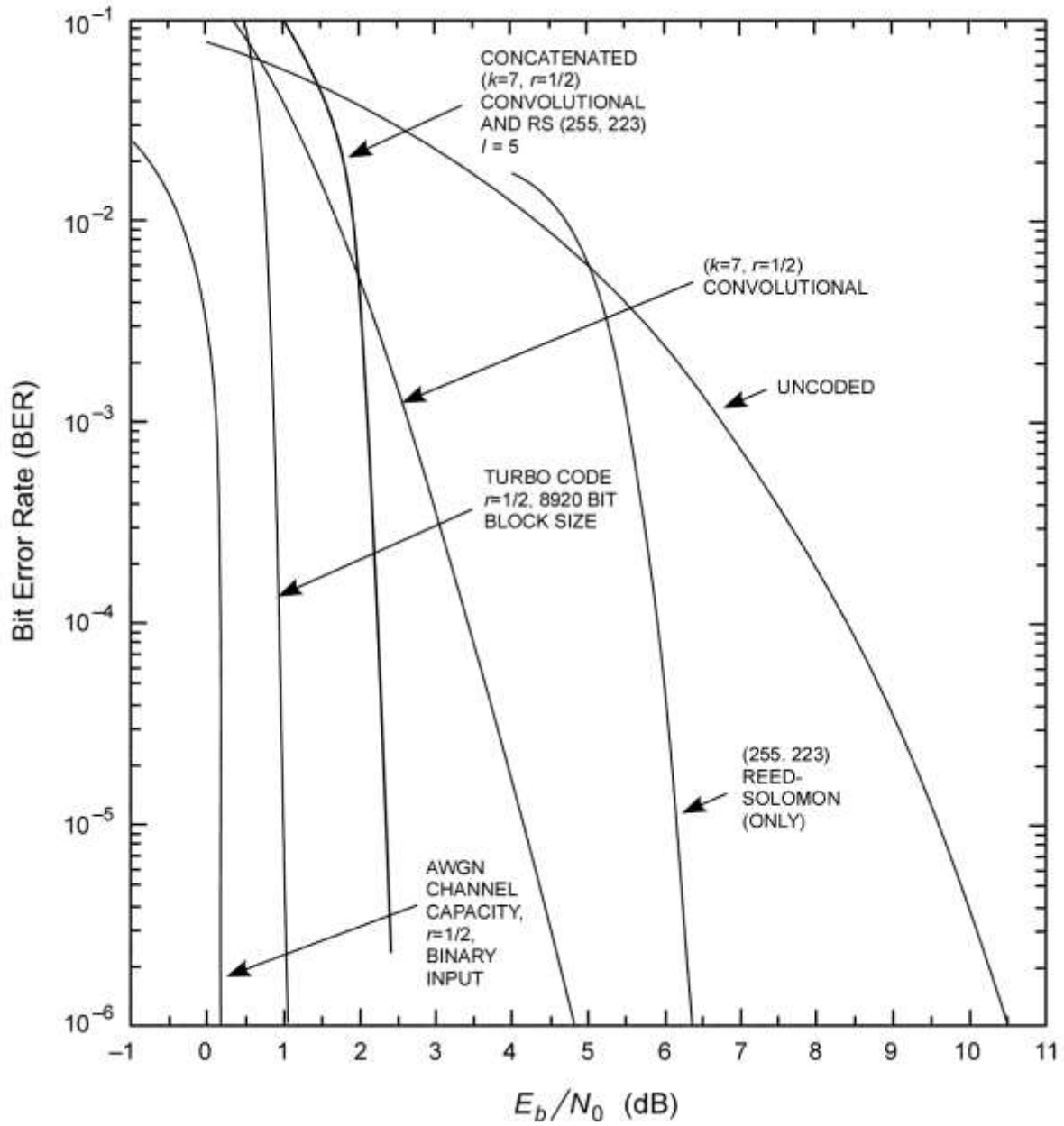


Figure 14. Relative Performance of Supported Codes

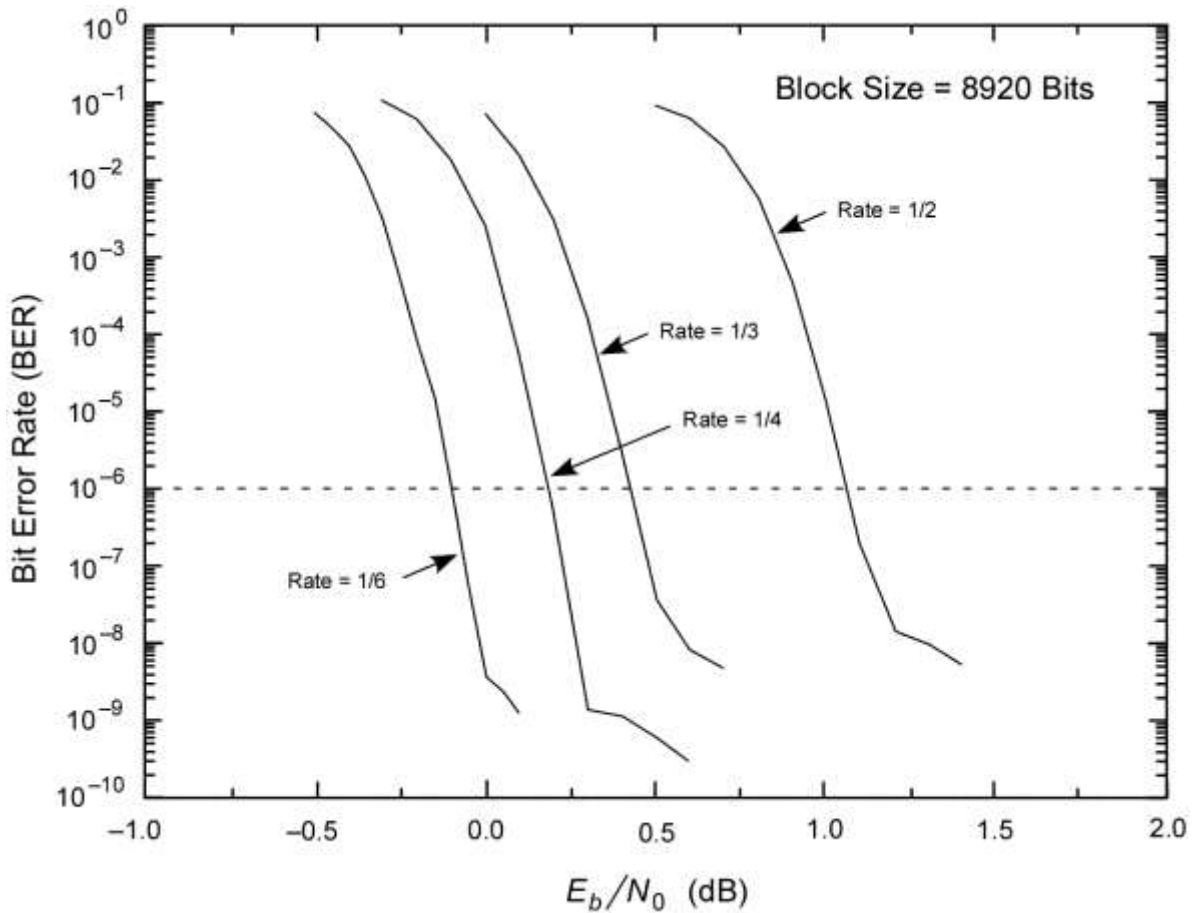


Figure 15. Measured Performance of DSN Turbo Decoder Showing Improvement with Code Rate and Error Floor Effects (Block Size = 8920 Bits)

## 2.6 Time Tagging

The DSN annotates every frame of data delivered to the user with its Earth-received time. The time may be specified as the beginning or end of each data frame depending on spacecraft data processing requirements. The time is calculated by determining the exact time the synchronization marker is recognized and adding a time delay measured when the equipment was installed to move the reference point to the input of the antenna's low noise amplifier. The normal precision of the time tag is 1 ms however additional precision can be provided by agreement between the DSN and users. Time tagging capability is summarized in Table 5.

Table 5. DSN Time Tagging

Parameter	Value
Normal Delivered Accuracy	Nearest ms
Station Reference	Test input port before LNA
Reference as Delivered	Leading edge of first bit of frame or trailing edge of last bit in frame
Accuracy	$\pm 5$ usec (for symbol rates > 2000 sps and carrier loop SNR $\geq$ 20 dB)

## 2.7 Data Formatting

The result of the previously described processing is a series of fixed-length frames of telemetry data. The content of these frames may represent a single stream of telemetry data or a portion of several streams of telemetry data referred to as virtual channels. Virtual channels allocate the physical channel on a frame by frame basis identified by a virtual channel identifier. The DSN separates the frames based on the virtual channel identifier and creates independent streams of telemetry data. The use of virtual channels enables portions of the data stream to be delivered to different locations or with different latencies. Two types of telemetry frames are supported. *Version I Frames*, originally specified in CCSDS Recommendation 102.0-B, have the capability to support up to eight virtual channels numbered from 0 to 7. *Version II Frames*, originally specified in CCSDS Recommendation 701.0-B, have the capability to support up to sixty-four virtual channels. The DSN can combine from 1 to 16 of these channels into *virtual data streams* and the same virtual channel may appear in multiple virtual data streams. The number of virtual data streams that can be created for any one project is limited to 16.

Figure 16 provides an example of telemetry data flow when virtual channels are used. As shown in the figure, the contents of a virtual channel may be created by combining packets from multiple sources. The packets from each source are identified by a header that contains an Application Process ID (APID) and a packet length. This enables the user to separate the packets from each source. Since the virtual channel identifier and packet header fields within the transfer frames are not protected from errors, it is recommended that virtual channels not be used unless frames are known to be decoded correctly as can be determined if Reed-Solomon coding or a CRC field is used.

The DSN annotates each frame delivered to a user with received time and accountability information for each channel being delivered as opposed to the physical channel. The structure and detailed content of the data blocks as delivered is beyond the scope of this document but several standard formats are available and deviations to these formats can be negotiated as part of the establishment of detailed mission requirements.



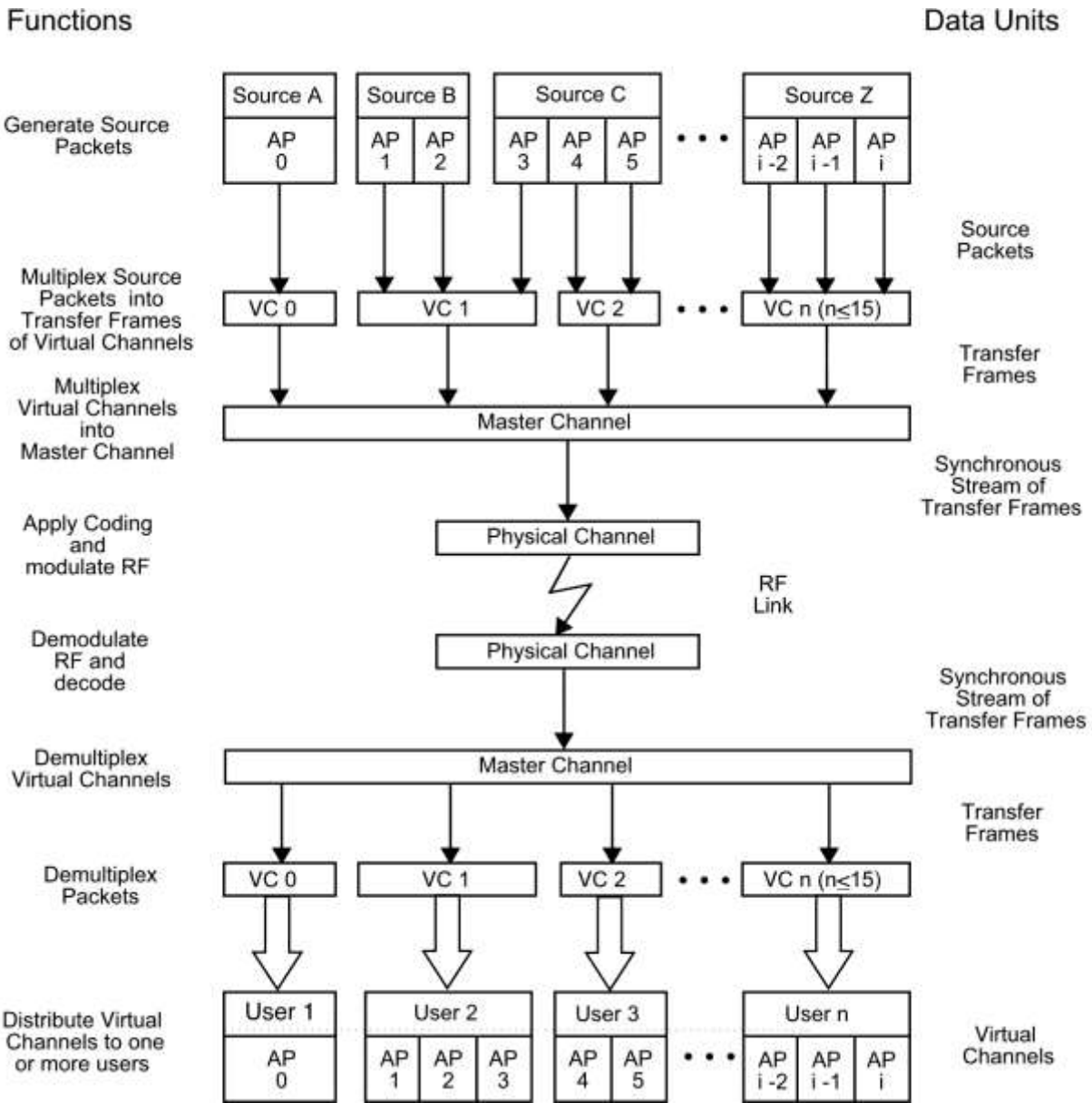


Figure 16. Example of Telemetry Data Flow Using Virtual Channels

## 2.8 Supported Telemetry Configurations

Figures 17 through 20 illustrate the telemetry coding configurations for spacecraft and ground equipment that are supported by the DSN. The order in which the steps in the coding and decoding process are performed are those recommended by the CCSDS and are fixed by hardware design.

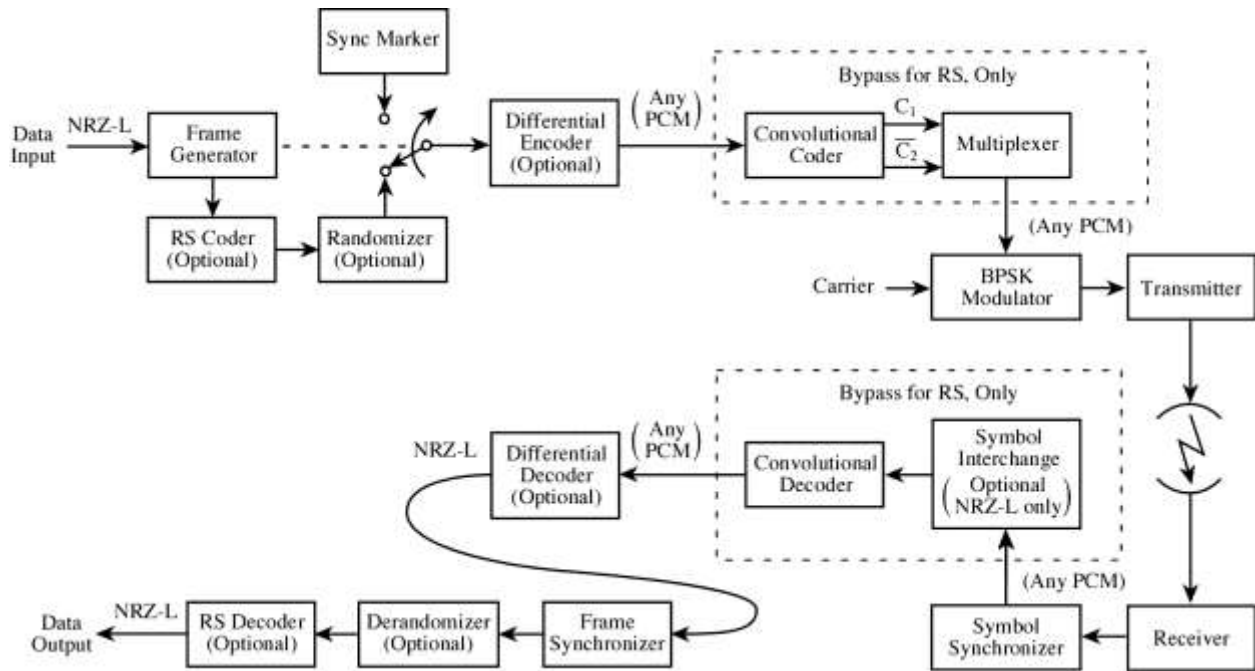


Figure 17. Spacecraft and Ground Configuration for BPSK Reed-Solomon, Convolutional, and Concatenated Coding

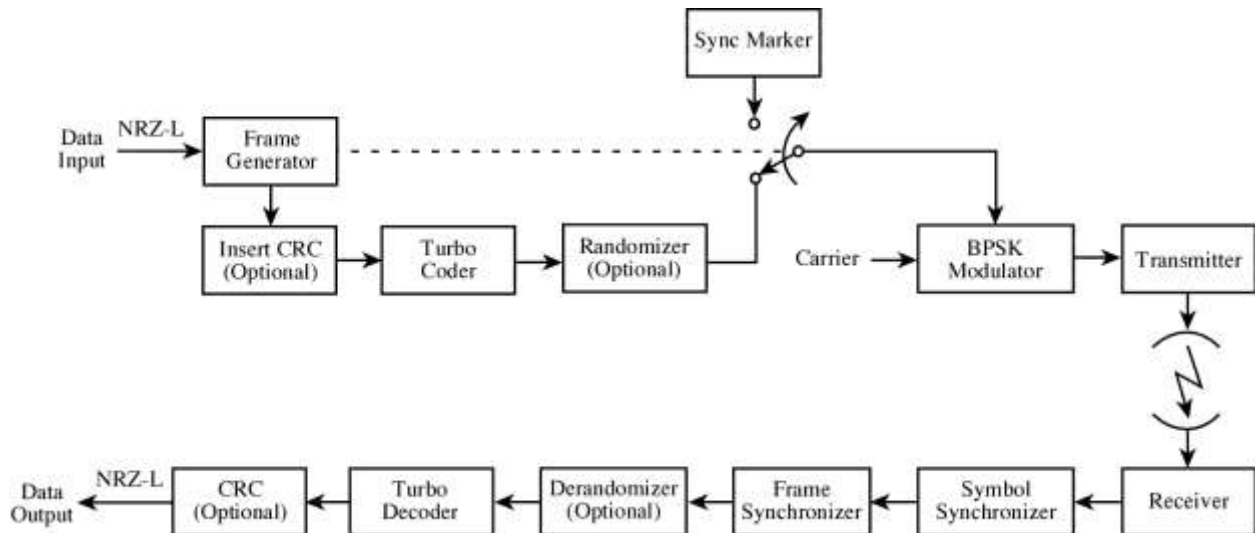


Figure 18. Spacecraft and Ground Configuration for BPSK Turbo Coding

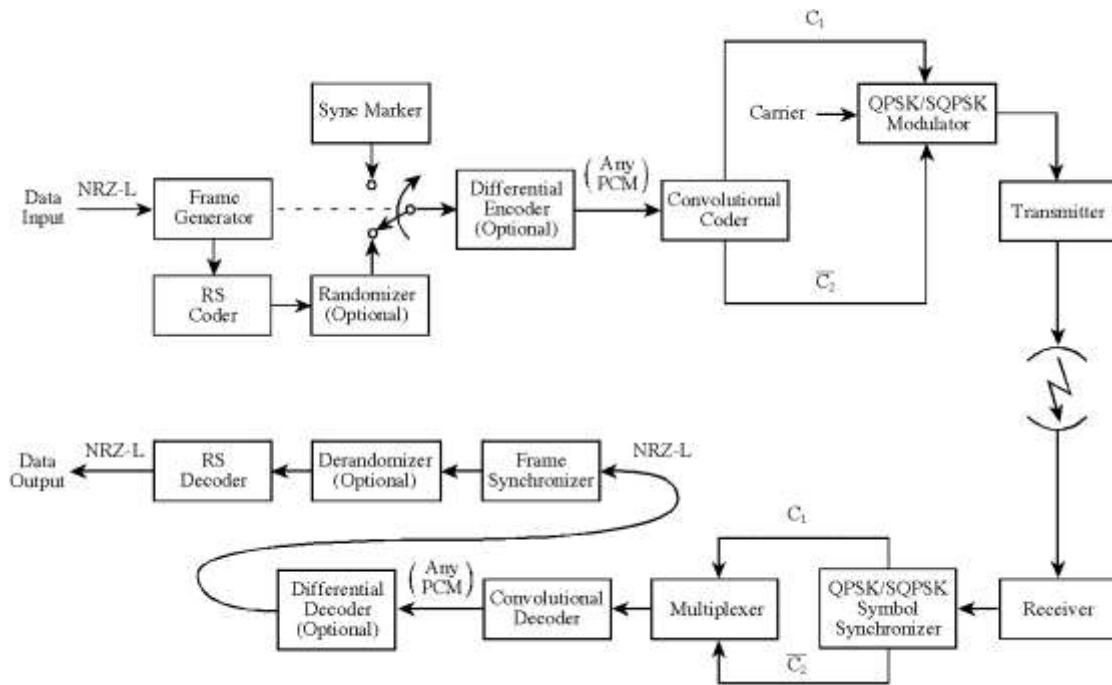


Figure 19. Spacecraft and Ground Configuration for QPSK/SQPSK Convolutional and Concatenated Coding

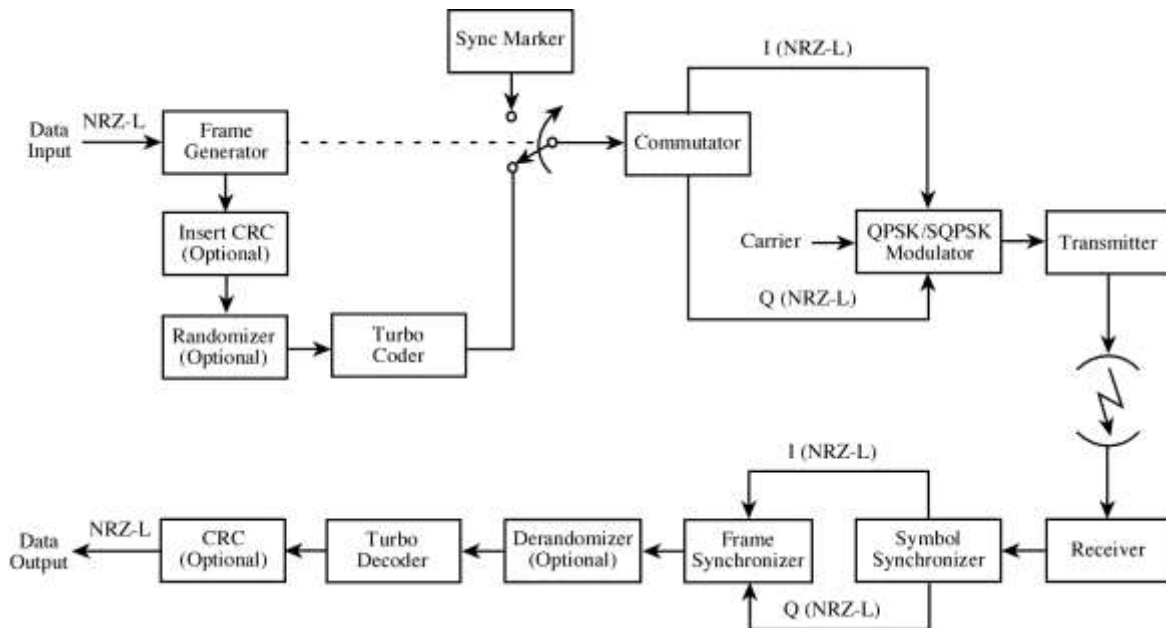


Figure 20. Spacecraft and Ground Configuration for QPSK/SQPSK Turbo Coding

### **3**            ***Proposed Capability***

The following paragraphs discuss capabilities that have not yet been implemented by the DSN but have adequate maturity to be considered for spacecraft mission and equipment design. Telecommunications engineers are advised that any capabilities discussed in this section cannot be committed to except by negotiation with the DSN System Engineering and Commitments Office.

#### **3.1**            ***Low-Density Parity-check (LDPC) Codes***

Low-Density Parity-Check (LDPC) codes have been developed that provide near-theoretical limit performance at high code rates to complement the similar performance provided by Turbo codes at low code rates. They promise to be especially useful in applications where the bandwidth required to use a Turbo code is not available or would complicate spacecraft equipment design. LDPC codes have an additional benefit that their decoder structure is more appropriate for high-speed hardware implementation and, on the average, requires fewer computations per decoded bit.

LDPC codes were originally invented by R. Gallager in 1961 but were largely forgotten for 30 years. The discovery of an iterative decoding algorithm, now referred to as Belief Propagation (BP) decoding, in the mid 1990s coupled with advances in digital processing technology revived interest in the coding technique. LDPC codes are similar to turbo codes in that they are binary block codes with large code blocks of hundreds to thousands of bits. The codes selected for deep space applications are members of a class of LDPC codes referred to as quasi-cyclic. This class of codes has an advantage that encoder implementation can be accomplished with shift registers.

The particular codes selected for deep space application are described in the CCSDS Experimental Specification 131.1-0-2. They are systematic and non transparent requiring that phase ambiguities be resolved using the frame markers that are required for codeblock synchronization. Although these codes theoretically have error floors, they are typically at least two decades below those of Turbo codes so the CRC that is recommended with Turbo codes is unnecessary with LDPC codes. The codes cannot guarantee sufficient bit transitions to keep receiver symbol synchronizers in lock so the pseudo-randomizer described in section 2.5.3 of this document is required unless the system designer verifies that sufficient symbol transition density is assured by other means. Codeblock lengths for the supported code rates are provided in Table 6.

Figure 21 is included to show how LDPC codes compliment Turbo codes. The figure is in the symbol domain to make the effects of code rate more apparent and to prevent the curves from over-writing each other if they were presented in the bit domain in a single figure.

Table 6. Codeblock Lengths for LDPC Code Rates

Information Block Length, $k$	Codeblock Length, $n$		
	Rate = 1/2	Rate = 2/3	Rate = 4/5
1024	2048	1536	1280
4096	8192	6144	5120
16384	32768	24576	20480

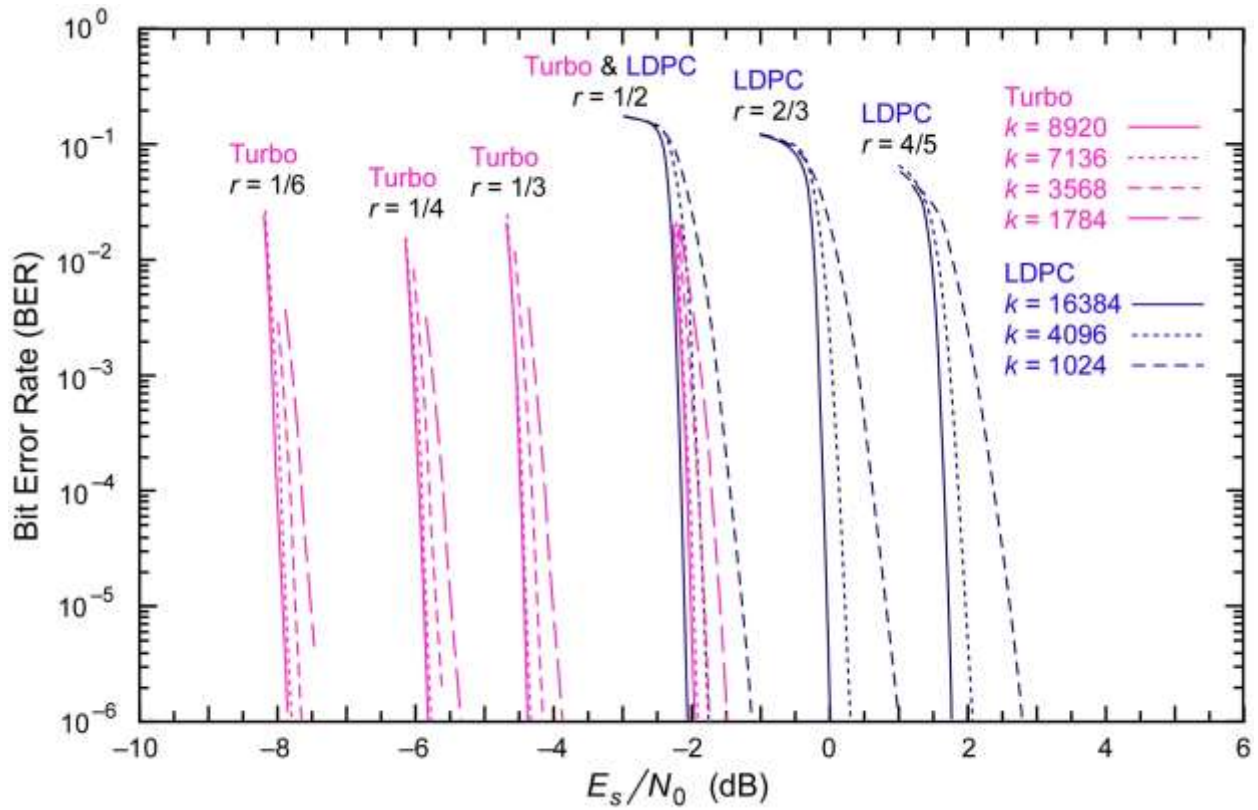


Figure 21. LDPC and Turbo Code Comparative Performance

## ***References***

- 1 FR. G. Gallager, “Low Density Parity Check Codes,” *IRE Transactions on Information Theory*, vol. IT-8, pp. 21–28, 1962.
- 2 CCSDS 102.0-B-5-S, Telemetry Channel Coding, Blue Book. Issue 5, November 2000.
- 3 CCSDS 130.0-G-1, Informational Report, TM Synchronization and Channel Coding – Summary of Concept and Rationale, June 2006
- 4 CCSDS 131.0-B-1, Recommendation, TM Synchronization and Channel Coding
- 5 CCSDS 131.1-O-2, Experimental Specification, Low Density Parity Check Codes for Use in Near-Earth and Deep Space Applications, September, 2007
- 6 CCSDS 701.0-B-2, Recommendation, Advanced Orbiting Systems, Networks and Data Links: Architectural Specification