A DESIGN FOR SATELLITE GROUND STATION RECEIVER AUTOCONFIGURATION

Phillip De Leon, Qingsong Wang, Steve Horan, Ray Lyman
New Mexico State University
Klipsch School of Electrical and Computer Engineering
Las Cruces, NM 88003
{pdeleon, qwang, shoran, rlyman}@nmsu.edu

ABSTRACT

In this paper, we propose a receiver design for satellite ground station use which can demodulate a waveform without specific knowledge of the data rate, convolutional code rate, or line code used. Several assumptions, consistent with the Space Network operating environment, are made including only certain data rates, convolutional code rates and generator polynomials, and types of line encoders. Despite the assumptions, a wide class of digital signaling (covering most of what might be seen at a ground station receiver) is captured. The approach uses standard signal processing techniques to identify data rate and line encoder class and a look up table with coded sync words (a standard feature of telemetry data frame header) in order to identify the key parameters. As our research has shown, the leading bits of the received coded frame can be used to uniquely identify the parameters. With proper identification, a basic receiver autoconfiguration sequence (date rate, line decoder, convolutional decoder) may be constructed.

KEY WORDS

Digital receiver, “smart” receiver, ground station receiver, autoconfiguration, self-configuration

1. INTRODUCTION

Users of NASA’s Space Network (SN) Multiple Access service may transmit their data at different data rates, i.e. 9600bps, 1Mbps, etc.; may convolutionally encode their data at different rates, i.e. rate 1/2, rate 1/3, etc.; and may line encode their data with one of several standards, i.e. Not Return to Zero (NRZ) or Biphase (BiΦ) also known as Manchester encoding [1], [2]. Normally, these communications parameters are known in advance so that the Ground Station Receiver (GSR) can be configured. The question we address in this paper is whether from the data signal itself, these parameters (data rate, convolutional encoder rate, line encoder) can be identified. Our motivation is to design a receiver which can configure itself automatically (autoconfigure).

In addition, a GSR autoconfiguration would be useful in the event of an emergency onboard a spacecraft. In such an event, the spacecraft might enter into alternate operating modes usually varying data communications parameters in an attempt reestablish communications. With such an autoconfiguration receiver, ground station controllers could be aided in fault recovery. As an example, with the Hubble Space Telescope (HST), both the data rate and data format change when it goes into a “safe-hold.” This event leads to a loss of data until controllers have realized HST is in the safe-hold condition [3]. Finally, such an autoconfiguration feature might be useful in forming clusters of autonomous, heterogeneous satellites where communications systems will need to adapt to different techniques in real-time.

As a first approach toward estimating the parameters, one might consider discriminating temporal, spectral, or statistical characteristics of the various codes (convolution and line) and make use of these characteristics to estimate the parameters. However, other than using the DC response to estimate the line code class (NRZ or BiΦ) but not the particular format (Line, Mark, or Space) or convolutional code rate, it is not clear this approach can yield information useful in parameter estimation. As a second approach, we consider the use of standard telemetry frame header bits used in synchronization tasks. We will generically call these header bits, sync words. The idea is that these sync words when coded, yield unique information sufficient to estimate the parameters. Of course this uniqueness depends on significant reduction of the number of sync words, code rates, and data formats under consideration from the state space of all possible combinations that can be envisioned.

In the next subsections, we give a brief review of the encoder structures found in the Space Network User’s Guide (SNUG). We consider only rate 1/2 and 1/3 convolutional coding as well as uncoded data and six standard line codes (NRZ-L, -M, -S; BiΦ-L, -M, -S) used in most telemetry applications. In the remainder of the paper, we describe the use of the sync words in estimating the key parameters. We also present results concerning the accuracy of these estimates.

1.1 Convolutional Coder

Fig. 1 shows a constraint length M+1, rate 1/2 convolutional coder . Here the generator polynomial for the ith path from input to output is

$$g^{(i)}(D) = g_0^{(i)} + g_1^{(i)}D + g_2^{(i)}D^2 + \cdots + g_M^{(i)}D^M \tag{1}$$

where D represents a unit delay [4]. A rate 1/3 coder would have an additional path from input to output. As described in the SNUG, only convolutional code rates 1/2 and 1/3 are used, each with constraint length 7. In addition, the generator polynomials under consideration as given in the SNUG are [1]:

$$\begin{aligned} g^{(1)}(D) &= g_0^{(1)} + g_1^{(1)}D + g_2^{(1)}D^2 + g_3^{(1)}D^3 + g_6^{(1)}D^6 \\ g^{(2)}(D) &= g_0^{(2)} + g_2^{(2)}D^2 + g_3^{(2)}D^3 + g_5^{(2)}D^5 + g_6^{(2)}D^6 \\ g^{(3)}(D) &= g_0^{(3)} + g_1^{(3)}D + g_2^{(3)}D^2 + g_4^{(3)}D^4 + g_5^{(3)}D^5 + g_6^{(3)}D^6 \end{aligned} \tag{2}$$

[The third polynomial, $g^{(3)}(D)$ is obviously not used in the rate 1/2 encoder]. Finally, uncoded (straight-through path from input to output) data may also be used [1].

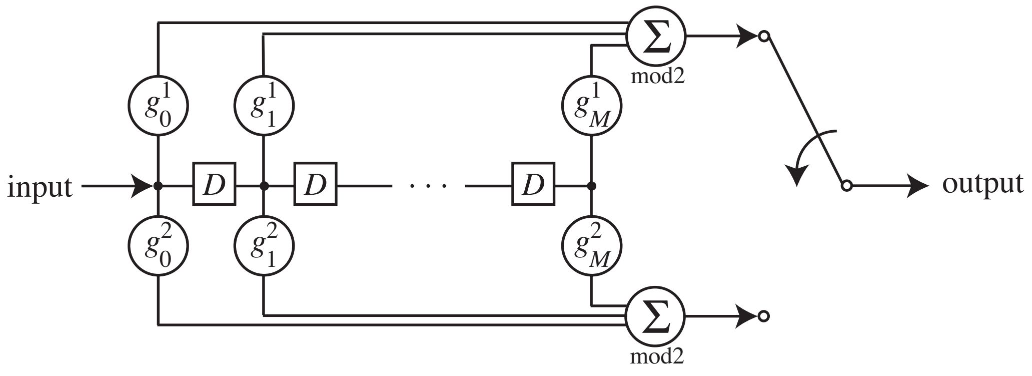

Figure 1: Constraint length M, rate 1/2 convolution coder

1.2 Line Coder

Prior to transmission, the data stream is converted to a waveform and the logical zeros and ones are encoded within this waveform as high (H) or low (L) voltage levels. The waveform or line encoding is used to assist in clock extraction at the receiver as well as to shape the spectrum of the transmitted signal [5], [6]. The usual desired spectral properties include a zero DC response and fast rolloff of the high-frequency spectrum. In this work, we consider the two line code classes which are standard for space telemetry: NRZ or BiΦ. For BiΦ, we consider the IRIG-106 definitions [4], [5]. Within each line code class are various formats: Line (L), Mark (M), and Space (S). The basic definitions are summarized in Table 1 below [2].

Table 1: Line Encoder Definitions

NRZ-L	“1”, “0” is represented by a high (H), low (L) level during the symbol period, respectively.
NRZ-M	“1” is represented by a change in level. “0” is represented by no change in level.
NRZ-S	“0” is represented by a change in level. “1” is represented by no change in level.
BiΦ-L	“1” is represented by a midperiod transition from H to L. “0” is represented by a midperiod transition from L to H.
BiΦ-M *IRIG-106 Definition*	“1” is represented by a midperiod transition. “0” is represented by no midperiod transition.
BiΦ-S *IRIG-106 Definition*	“0” is represented by a midperiod transition. “1” is represented by no midperiod transition.

1.3 Overall Encoding System

Fig. 2 illustrates the basic operation of the telemetry frame encoding process. A frame is assembled with a sync word in the header followed by data. This frame is then passed through the convolutional encoder and line coder resulting in a coded frame. The goal of the autoconfiguration receiver is to estimate the convolutional code rate and line code class parameters given only the coded frame. We will not assume any knowledge of the length of the sync word, thus we are not able to establish boundaries within the coded frame for where the coded sync word ends and where the coded data begins.

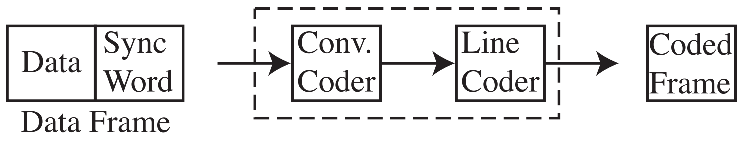

Figure 2: Overall coding procedure. Telemetry frame composed of a sync word (header) with a data payload. After convolutional and line coding, the result is a coded frame.

2. RECEIVER DESIGN APPROACH

The proposed approach toward a design for receiver autoconfiguration involves two stages as illustrated in Fig. 3. The first stage estimates data rate and the line code class (either NRZ or BiΦ). The second stage involves a look up table (LUT) whose entries or keys contain the coded sync words. These keys can be generated by passing all sync words under consideration through the structure in Fig. 2 for all combinations of convolutional encoder and line encoder that are to be considered. The LUT is constructed prior to the autoconfiguration and may be stored in a ROM.

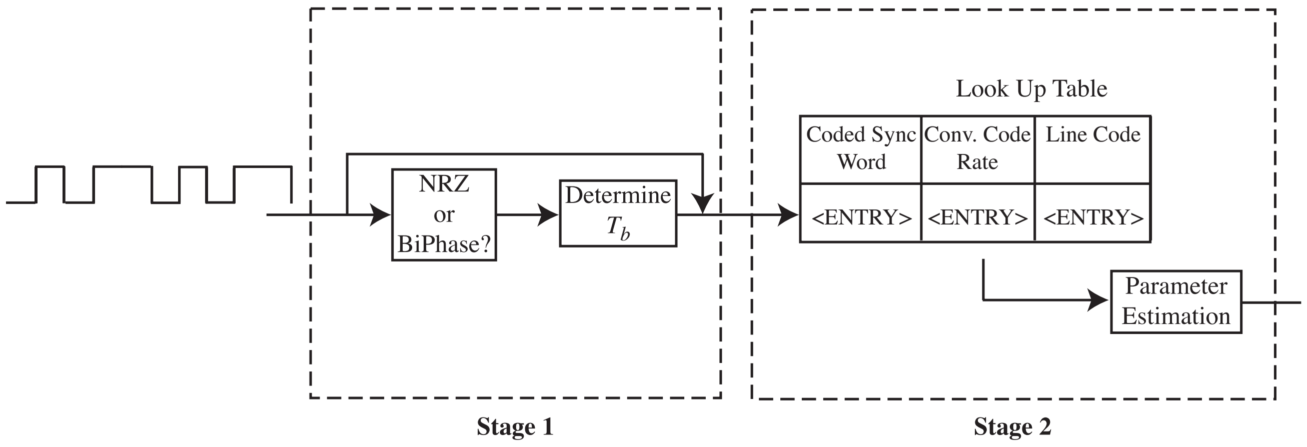

Figure 3: Proposed design for receiver autoconfiguration

In order to establish the data rate, we must know whether the line code is NRZ or BiΦ since the latter involves a voltage transition mid-period which would otherwise be interpreted as a doubling of the bit rate. In addition, once the line code class is established, the search space within the LUT can be reduced by half since only those keys with the line code need be searched.

3. STAGE 1: DATA RATE, LINE CODE CLASS ESTIMATION

We assume a spread spectrum system, knowledge of a user's Pseudo Noise (PN) code, and PN lock up to the user have already been accomplished by ground station operations. We assume at the input a despread, PCM signal (waveform) resembling Fig. 4 is available for analysis. For now, we do not assume any pulse shaping, e.g., raised cosine filtering, has been applied to the bits. If this has been applied, we assume that the bits have been processed to yield a waveform with rectangular pulses. In order to estimate the data rate, we propose to oversample the waveform for a short period of time and measure the smallest width pulse. Assuming a sample rate of f_s, and a shortest pulse width of L samples, then an estimate for the bit period is given by

$$\hat{T}_b = cLf_s \tag{1}$$

where c is 1 if NRZ line encoding is used and 2 if BiΦ is used [actual identification of the line code format (L, M, or S) is handled by Stage 2]. The estimate for the data rate is then

$$\hat{R}_b = \frac{1}{\hat{T}_b}. \tag{2}$$

Once c is known, this estimate can be compared to a list of "most likely" data rates with the closest match selected. This provides the data rate, R_b. It should be noted that in an effort to reduce cost, more and more space communication system designs use commercial off-the-shelf chip sets. These chips normally implement CCITT standards for which only certain data rates are allowed. As a start, the allowable date rates would form the list of "most likely" data rates which the proposed receiver design would compare against.

Figure 4: Example waveform.

From the waveform, we must now determine what line code format is used assuming either NRZ or BiΦ. There are two approaches to identifying the line code class: frequency-domain and time-domain based. The former is, in theory, more accurate but at a relatively high implementation cost while the latter's implementation is simple but the accuracy is a function of the length of the observed waveform.

3.1 Frequency-Domain Approach

By computing the power spectrum, $S(\omega)$, of the waveform one can differentiate between NRZ and BiΦ since the DC response of BiΦ is zero while that of NRZ is a peak. Therefore a simple procedure for identifying the line code (either NRZ or BiΦ) is to compute the power spectrum for the waveform and evaluate the derivative of the PSD at zero. If the derivative is negative (spectrum slopes down from 0Hz) the line code class is NRZ, otherwise the derivative is positive (spectrum slopes up from 0Hz) and the line code class is BiΦ. The derivative can be approximated by a simple backward difference equation

$$m = S(1) - S(0) \tag{3}$$

where $S(k)$ is the kth point in the spectrum. This procedure requires an FFT calculation and evaluation of (3).

This approach has the disadvantage of taking a relatively long period to compute not to mention the computational requirements. In the next subsection, we investigate a simpler approach that yields equivalent results.

3.2 Time-Domain Approach

The second approach to identify the line code class is strictly time-domain based. With this method we exploit the fact that with BiΦ, we cannot have a pulse width more than $1.5\hat{T}_b$ since each bit must have a transition. Therefore, we need only examine the waveform and check to see if there exists a pulse width with duration exceeding $1.5\hat{T}_b$. If such a pulse exists, the line code is NRZ otherwise it could be either NRZ or BiΦ. Since it is unreasonable to examine the entire waveform, we must determine how much of the waveform one would need to examine to be within some desired probability of correctly identifying the line code. Assuming +V and –V are equally likely, Fig. 5 illustrates how many bit periods one needs to examine to have a probability, P that no pulse of duration $1.5\hat{T}_b$ exists. We can show that to be 99.99% certain there is no pulse of width $1.5\hat{T}_b$, we must examine 48 bits. If during the examination, we encounter a pulse of width more than $1.5\hat{T}_b$ we are certain that NRZ is the line code and the procedure can stop. Otherwise we would default to BiΦ. The implementation for such an approach involves a simple digital filter (assuming ±1 levels) and is shown in Fig. 6.

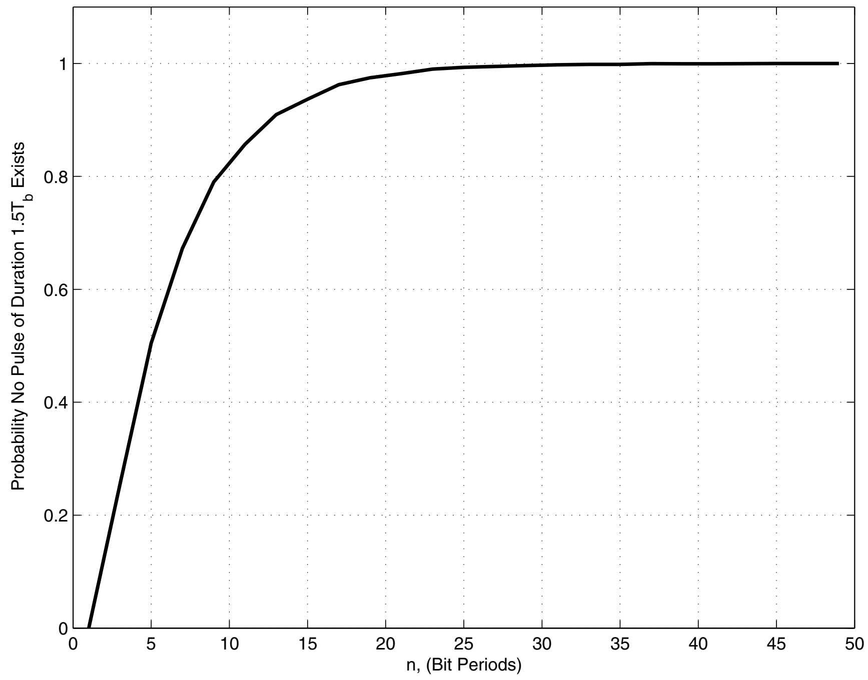

Figure 5: Probability of no pulse width more that $1.5\hat{T}_b$ as a function of the number of bits examined.

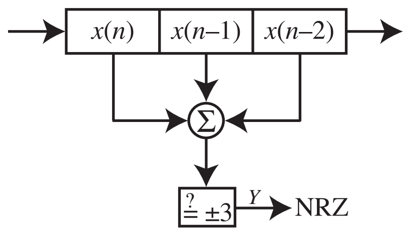

Figure 6: Time-domain approach to line code class identification.

4. STAGE 2: LOOK UP TABLE

In order to determine the particular line code (-L, -M, -S) and coding rate, we make use of what is known about the structure of the transmission frames. These are always preceded by a sequence of known symbols to aid the receiver in synchronization tasks. We use the generic term *sync word* for these known sequences.

The sync word used is specified in the data-link protocol. Users of the SN most often employ HDLC or one of the CCSDS recommendations. The most commonly used sync words from these are listed in Appendix A. Even if the protocol in use is not known in advance, if the set of possible sync words is small enough, then we may use a LUT approach to determine the code rate and line code.

The LUT is constructed by convolutionally encoding each of the sync words at each coding rate of interest. Then, each resulting sequence is line coded using each line code of interest. The resulting output sequences become keys in the LUT. When a frame is received, we check the first N binary symbols of the frame against the keys in the LUT, and when a match is found, we choose the line code and coding rate corresponding to that key. As in Fig. 7, we find that as long as we choose N greater than or equal to 13, there is no chance that identical keys will yield different values for the coding rate or line code. It is likely that when a frame is first received, we will not be examining the header of the frame. Therefore it is proposed that the above procedure is repeated, shifting in one new symbol until the match is detected thereby making a running correlator as typically found in frame synchronization. Work is in progress in quantifying the impact of this initialization.

Let us consider a couple of examples. Since the sixth and eighth sync words from Appendix A are identical in the first 96 bits, it is clear that, after convolutional encoding and line coding, the resulting output sequences will be identical in the first 13 binary symbols. But this causes no ambiguity because we find that any pair of identical output sequences always corresponds to the same transmission parameters; i.e., the same code rate and the same line code.

On the other hand, suppose we consider only the first $N = 12$ received binary symbols. We find that the fifth sync word from Appendix A produces output sequences which are identical in the first 12 binary symbols when it is convolutionally encoded at rates 1/2 and 1/3. Thus, if these 12 symbols are used as a key in the LUT, it will be impossible to determine which coding rate was applied.

Even with *N* sufficiently large, one possible difficulty with this approach is in the case of a key with length $K < N$. When comparing a received sequence to such a key, we use only the first *K* symbols. A problem occurs, though, when the *K* symbols of this sequence are identical to the first *K* symbols of another longer key somewhere else on the LUT. To see this, consider the sync word which results in the *K*-length output sequence. The user data that follows this sync word may cause the first *N* received symbols to be identical to the longer key. Fortunately, all the sync words of interest, except the HDLC "flag byte" 0 x 7E are longer than 13 bits, and none of the output sequences resulting from the HDLC flag byte exhibit the problem just described.

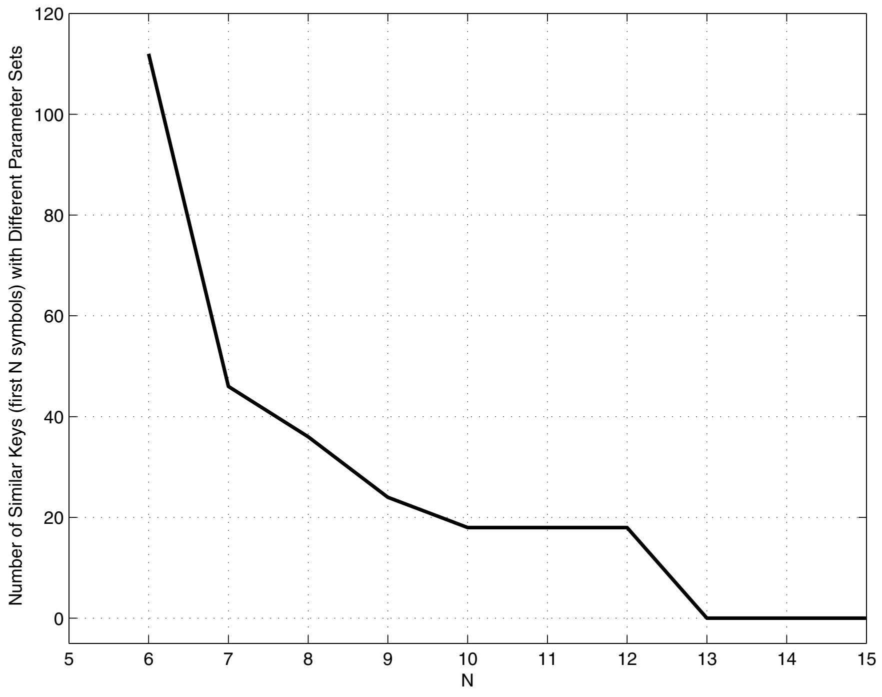

Figure 7: Number of key entries with the same *N* leading symbols but different parameters

If we consider the 8 sync words in Appendix A, two convolutional code rates (1/2 and 1/3) as well as uncoded data, and six line encoders, an 8K ROM will be sufficient to store the table. We note that both the line decoders and convolutional decoder are implemented with simple digital circuits. Together with the ROM, the system could easily be implemented in an FPGA and interfaced with a GSR.

5. FUTURE WORK

Regarding future work, our immediate task is relaxing the assumption that we know beforehand where the frame begins. Problems to be dealt with include the possibility that a sequence of symbols from mid frame falsely match one of the output sequences in the LUT. We also wish to investigate the impact of physical-layer effects, such as noise, pulse shaping and timing jitter, on our ability to identify the transmission parameters. More long term, we wish to find a more general characterization of the conditions under which this identification can be made. For example, what must be true about the set of sync words, or the parameters to be identified? Such knowledge would allow the techniques developed here to be applied more generally.

6. CONCLUSIONS

In this paper, we have proposed a ground station receiver which autoconfigures itself in the sense that the data rate, convolutional code rate, and line encoder type parameters are all estimated from the received waveform itself. The approach assumes the presence of a sync word in the frame header. After passing the sync word through the convolutional and line encoders, a key is produced and stored in a lookup table. When a frame is received at the receiver, we check the first N binary symbols of the frame against the keys in the table, and when a match is found, choose the coding rate and line code corresponding to that key. Our research has demonstrated that with $N = 13$, there is no chance that identical keys will yield different values for the coding rate or line code.

7. ACKNOWLEDGMENT

This research is supported by NASA Grant NAG5-13189. In particular, we wish to thank Mr. Dave Israel of the NASA Goddard Space Flight Center for discussions regarding our proposed approach.

REFERENCES

[1] "Data Encoders," *Space Network User's Guide, Revision 8*, June 2002, Appendix B, p. B-15.

[2] "Digital Signal Formats," *Space Network User's Guide, Revision 8*, June 2002, Appendix B, p. B-2.

[3] Dave Israel, NASA Goddard Space Flight Center, private communication, Feb. 2003.

[4] S. Haykin, *Communication Systems 4th Ed.*, John Wiley & Sons, Inc., New York, NY, 2001.

[5] S. Horan, *Introduction to PCM Telemetering Systems 2nd Ed.*, CRC Press, 2002.

[6] Telemetry Group, Range Commander's Council, Telemetry Standards, IRIG Standard 106-01, Part 1, WSMRi, NM, Feb. 2001, p. C-5.

Appendix A: Sync Words Under Consideration

The following table lists the sync words most commonly used on the Space Network.

Table A.1: Commonly Used Sync words, in Hex Notation.

Sync word	*Comments*
1: 7E	HDLC flag byte
2: EB90	CCSDS: TC
3: FAF320	CCSDS: Proximity-1
4: 1ACFFC1D	CCSDS: TM
5: 034776C7272895B0	CCSDS: TM rate-1/2 turbo
6: 25D5C0CE8990F6C9461BF79C	CCSDS: TM rate-1/3 turbo
7: 034776C7272895B0FCB88938D8D76A4F	CCSDS: TM rate-1/4 turbo
8: 25D5C0CE8990F6C9461BF79CDA2A3F31 766F0936B9E40863	CCSDS: TM rate-1/6 turbo

Appendix B: Sample Lookup Table Entry

The following is a sample entry for IRIG Code 0xFAF320 convolutionally encoded at rate 1/2 and line encoded with NRZ-L.

Coded Sync word	Code Rate	Line Code
. . . . . .		

1100 0000 1111 0000 0000 1100 0000 1100
1111 0011 0000 0011 0000 0000 1111 1100
1100 1100 0011 1100 0000 0011 0011 0000
	1 / 2	NRZ - L

. . . . . .		