Reed-Solomon Code Synchronization Revisited

L. J. Deutsch
Communications Systems Research Section

A concatenated coding scheme consisting of an inner (7, 1/2) convolutional code and an outer (255, 223) Reed-Solomon code has been recommended by the Consultative Committee for Space Data Systems for cross-supported space missions. The Reed-Solomon code that was chosen makes use of the Berlekamp encoding algorithm. This report examines some peculiarities of this code that could give rise to synchronization problems. Suggestions are given to alleviate these problems.

I. Introduction

Concatenated coding for deep space missions was developed in response to a need for a relatively error-free channel for high digital data rates. Such a channel is required, for example, when source data compression is to be used. There is a long history of work here at JPL on such concatenated systems both with and without data compression (Refs. 1-3).[1] In recent years it has become evident that such concatenated coding systems will be required by many future missions both by NASA and by foreign space agencies. For this reason, the Consultative Committee on Space Data Systems (CCSDS), which is formed of members from many space agencies throughout the world, has published a set of guidelines for concatenated coding systems. (These guidelines appear in the CCSDS publication *Telemetry Channel Coding*, a draft "blue book," February, 1984.)

[1]See also J. P. Odenwalder, *Concatenated Reed-Solomon/Viterbi Channel Coding for Advanced Planetary Missions: Analysis, Simulations, and Tests*, Submitted to the Jet Propulsion Laboratory by Linkabit Corporation, San Diego, Calif., Contract No. 953866, December, 1974.

The CCSDS-recommended system consists of an inner (7, 1/2) convolutional code and an outer (255, 233) Reed-Solomon code with 8-bit symbols. These are the same parameters as are used by the Voyager project. However, a new implementation of the Reed-Solomon code due to Berlekamp (Ref. 4) was chosen for the standards. This code has a symmetric code generator polynomial that can significantly reduce the amount of hardware needed to implement an encoder. Berlekamp also described a bit-serial algorithm for an encoder that further reduces its size and weight. Perlman and Lee (Ref. 5) worked with Berlekamp to produce a flight-qualifiable prototype for this encoder. As a result of their work, the CCSDS has adopted the Berlekamp Reed-Solomon code as a recommended standard. In Ref. 6 a very large scale integrated (VLSI) implementation of the Berlekamp encoder is described.

In order to reduce the error rate of the system still farther, Reed-Solomon codewords will be interleaved to a depth of I ($I = 5$ in most proposed applications). Interleaving occurs on the Reed-Solomon symbol level and is performed in such a way that the order of information symbols is preserved by the encoding process.[2] A set of I interleaved codewords is called a Reed-Solomon frame. Since the Reed-Solomon code is a block code, codeword synchronization is required before the decoding process can begin. This will be done by placing a fixed set of symbols, called a frame synchronization marker (or simply a frame marker), at the beginning of each Reed-Solomon frame. An analysis of this synchronization technique can be found in Ref. 7.

This report examines two phenomena that are peculiar to the Reed-Solomon code that could lead to code synchronization problems if they are not treated correctly. They are both fairly well known effects to those who work with Reed-Solomon codes but may be unknown to others who design systems that will use the codes.

The first effect arises from the fact that the recommended code generator polynomial is symmetric. Consider the case in which all the data symbols to be encoded are zero. Before encoding takes place, the synchronization marker is placed at the beginning of the frame. Suppose that there are n symbols in this marker (i.e., there are less than or equal to $8n$ bits in the marker) and that n is at most equal to I. Then the encoded frame has the form shown in Fig. 1. Notice that, since the code generator polynomial is symmetric, the 224th column of the frame is identical to the first. This can be easily seen by the fact that the Reed-Solomon code is cyclic and each row is just the cyclic rotation of the codeword formed by a constant multiplied by the generator polynomial. In particular, the synchronization marker appears in both the first and 224th column. This can result in false synchronization of the frame. In fact, this was observed in systems tests performed by the European Space Agency (ESA).

The second effect considered in this report is that repeated sequences of symbols of certain periods are Reed-Solomon codewords. This effect includes the special case of the constant codeword as well as (in the case of the CCSDS code) cycles of length 3, 5, and 15. This can produce only a very rare case of synchronization failure but it is a very useful fact in the testing of encoders and decoders. This effect was observed by C. Lahmeyer while implementing the decoders that will be eventually used by the Voyager project. R. L. Miller presented an explanation in an internal memorandum. This report presents a proof of the effect that can easily be generalized to other code sizes.

[2]See J. P. Odenwalder, op. cit.

II. False Synchronization of Constant Data Sequences

As mentioned in Section I, if the data source produces a constant symbol value, then there is a possibility of false frame synchronization in the 224th column. Since the (255, 223) Reed-Solomon code can correct only 16 symbol errors, a false synchronization in this case will cause the decoder to fail to decode – even in the case of a noiseless channel. This condition can be reported automatically by the decoder and, in fact, this is how ESA first noticed the problem. There are several possible solutions to this problem.

Since the false image of the marker is separated from the actual marker by exactly $32*I$ symbols, the frame syncronization system can keep track of both images and easily distinguish between them. Alternatively, the marker can be taken out of the frame and inserted, instead, between the frames. In this way, the marker is not encoded and the false image is never formed.

Another solution is to use "virtual zero fill." This is an algorithm by which a shortened Reed-Solomon code (Ref. 8) is used with the same hardware by assuming that all the missing symbols are zeros. For implementation reasons, the missing symbols are always assumed to be the first ones in each codeword. Code shortening has the effect of moving the marker from the first column of the frame to another one. This suppresses, in most cases, the formation of a false image. Table 1 shows the number of false images that are produced as a function of the information block length of the shortened code. The numbers in this table were produced by computer simulation of the Reed-Solomon encoding process. All the lengths not shown represent cases in which no false images are formed. In particular, shortening by only one symbol (information length 222) alleviates the false image problem and produces an undetectable degradation in performance in most cases.

It should be noted that the case of constant data is rather uninteresting. In fact, a good source encoding algorithm would probably never let this happen. However, the above solutions may prove useful in a case where there is no data compression.

III. Periodic Codewords

In this section, the case of Reed-Solomon codewords that are made up of periodic sequences of symbols is examined.

Consider a (255, 223) Reed-Solomon codeword (not necessarily the CCSDS code) that consists of a periodic sequence of symbols. Let the smallest period of the symbols be n. Then clearly $n | 225$ or $n \in \{1, 3, 5, 15, 17, 51, 85, 255\}$. It will be shown that, in fact, the only possible values are 1, 3, 5, 15, and 255. We will ignore the rather uninteresting (and trivial) case of $n = 255$.

Let $p(x)$ be a polynomial over GF(256) of the form

$$p(x) = \sum_{k=0}^{(255/n)-1} x^{nk}$$

where n is an integral divisor of 255. The polynomial $p(x)$ can be considered a codeword in a (255, 223) Reed-Solomon code if it is divisible by the code generator polynomial. Another way of saying this is that all the roots of the generator polynomial are also roots of $p(x)$. Notice that $p(x)$ may also be written as

$$p(x) = \frac{(x^{255} - 1)}{(x^n - 1)}.$$

Hence the roots of $p(x)$ are just those 255th roots of unity that are not also nth roots of unity. Let α be a primitive 255th root of unity. Then the roots of the CCSDS Reed-Solomon generator polynomial, $g(x)$, are

$$\alpha^{112}, \alpha^{113}, \alpha^{114}, \ldots, \alpha^{143}$$

and a simple check will show that none of these are either 1st, 3rd, 5th, or 15th roots of unity. This means that all the roots of $g(x)$ are also roots of $p(x)$ for $n \in \{1, 3, 5, 15\}$ and hence $p(x)$ is a multiple of $g(x)$. This is just another way of saying $p(x)$ is a Reed-Solomon codeword. However,

$$\alpha^{120}$$

is a 17th root of unity so n cannot be a multiple of 17 if $p(x)$ is to be a codeword. Clearly, all periodic sequences of these periods can be built linearly by combining polynomials of the form $p(x)$ and hence these will also be codewords. It should be noted that the Voyager Reed-Solomon code which has roots

$$\alpha^1, \alpha^2, \alpha^3, \ldots, \alpha^{32}$$

in its code generator polynomial also exhibits this property.

The phenomenon of periodic codewords does not present a problem in frame synchronization except in the unlikely case that the frame marker is similar to part of a periodic data sequence. If the frame contains some nonconstant data (such as a time tag or frame identification code) then the problem will never exist.

IV. Conclusions

The two effects that have been described in this report are well understood and well known by those who routinely work with Reed-Solomon codes. The issue of false frame marker images is, nonetheless, an important consideration in the design of an overall coded telemetry system. Any of the various solutions that are cited in Section II will alleviate that problem. The best solution, however, would probably be to remove the frame synchronization marker from within the Reed-Solomon code block and place it, instead, between adjacent code blocks. In this way, not only would the false image problem be solved, but the wasted overhead that is created by the Reed-Solomon encoding of the frame marker would be eliminated.

References

1. Miller, R. L., Deutsch, L. J., and Butman, S. A., *On the Error Statistics of Viterbi Decoding and the Performance of Concatenated Codes*, Publication 81-9, Jet Propulsion Laboratory, Pasadena, Calif., September, 1981.

2. Liu, K. Y., and Lee, J. J., *An Experimental Study of the Concatenated Reed-Solomon/Viterbi Channel Coding System Performance and Its Impact on Space Communications*, Publication 81-58, Jet Propulsion Laboratory, Pasadena, Calif., August, 1981.

3. Rice, R. F., *Channel Coding and Data Compression Systems Considerations for Efficient Communication of Planetary Imaging Data*, Technical Memorandum 33-695, Jet Propulsion Laboratory, Pasadena, Calif., June, 1974.

4. Berlekamp, E. Bit-serial Reed-Solomon encoders, *IEEE Trans. Information Theory, IT-28*, No. 6, November, 1982, pp. 869–874.

5. Perlman, M. and Lee, J. J., *Reed-Solomon Encoders – Conventional vs Berlekamp's Architecture*, Publication 82-71, Jet Propulsion Laboratory, Pasadena, Calif., December, 1982.

6. Truong, T. K., Deutsch, L. J., and Reed, I. S., *The VLSI Design of a Single Chip Reed-Solomon Encoder*, Publication 82-84, Jet Propulsion Laboratory, Pasadena, Calif., November, 1982.

7. Swanson, L., *A Comparison of Frame Synchronization Methods*, Publication 82-100, Jet Propulsion Laboratory, Pasadena, Calif., December, 1982.

8. Deutsch, L. J., The effects of Reed-Solomon code shortening on the performance of coded telemetry systems, *TDA Progress Report 42-75*, Jet Propulsion Laboratory, Pasadena, Calif., November, 1983, pp. 14–20.

Table 1. **Number of frame marker images as a function of information block length**

Information Block Length	Number of Marker Images	Information Block Length	Number of Marker Images
1 (shortest)	2	117	2
24	2	122	2
26	2	126	2
34	2	129	2
35	2	135	2
37	2	147	2
54	2	152	2
55	3	155	2
61	2	163	2
69	2	169	3
72	2	170	2
77	2	187	2
89	2	189	2
95	2	190	2
98	2	198	2
102	2	200	2
107	2	223 (full length)	2
112	3		

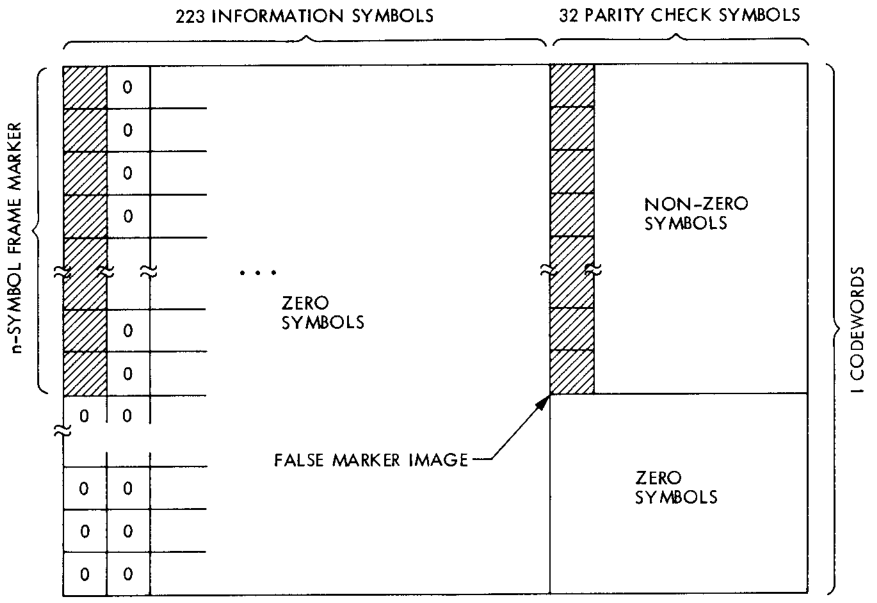

Fig. 1. Encoded Reed-Solomon frame with false frame marker image