TDA Progress Report 42-94
April–June 1988

Phobos Lander Coding System: Software and Analysis

K.-M. Cheung and F. Pollara
Communications Systems Research Section

This article describes the software developed for the decoding system used in the telemetry link of the Phobos Lander mission. Encoders and decoders are provided to cover the three possible telemetry configurations. The software can be used to decode actual data or to simulate the performance of the telemetry system. The theoretical properties of the codes chosen for this mission are analyzed and discussed.

I. Introduction

The Soviets plan to launch a mission to Mars in mid-July 1988. This mission consists of two identical spacecraft, each carrying a lander to be placed on Phobos, a moon of Mars. The insertion into Mars orbit will occur in mid-February 1989. The landing on Phobos will take place toward the end of April or early May 1989. This mission is designed to last one year with a possible extension to two years. NASA has agreed to provide some data acquisition support.

NASA/JPL's direct involvement in the Phobos project centers on the landers. These are complex spacecraft that are capable of receiving commands, transmitting telemetry, providing two-way range and doppler, and supporting Very Long Baseline Interferometry (VLBI) measurements.

The Phobos Lander coding system block diagram is shown in Fig. 1. The telemetry data are transmitted in frames of 2048 bits during a Phobos–Earth view period, which occurs approximately every 7 hours, 39 minutes (the Phobos orbital period) for a duration of approximately 20 to 40 minutes. The Phobos Lander telemetry system can be configured to operate in three different modes, as shown in Fig. 1. The first mode involves the use of a $K = 6, r = 1/2$ convolutional code, where K is the constraint length and r the rate of the code. The second mode involves the use of a $K = 9, r = 1/2$ time-varying convolutional code with period 4. The third mode involves the use of a concatenation of a Bose–Chaudhuri–Hocquenghem (BCH) block code and a $K = 9, r = 1/2$ time-varying convolutional code.

Our work is concerned primarily with the analysis and implementation of the following software modules to support the Phobos Lander telemetry processing requirement:

(1) A convolutional encoder ($K = 9, r = 1/2$, time-varying).

(2) A BCH encoder (128,113).

(3) A Viterbi decoder ($K = 9, r = 1/2$, time-varying).

(4) A BCH decoder (128,113).

(5) Convolutional code ($K = 9, r = 1/2$, time-varying) and BCH (128,113) test and simulation software.

In addition to the above software modules, an encoder module, a decoder module, and a simulation software module

for the $K = 6, r = 1/2$ convolutional code have been developed. The $K = 6, r = 1/2$ modules are used only to generate performance comparisons with the other operational modes.

All the software modules are written in FORTRAN-77 on an IBM-AT-compatible computer. They are contained in three programs:

(1) *k6.f*. This program contains the encoder and decoder modules for the $K = 6, r = 1/2$ convolutional code. It determines the bit-error-rate performance of this convolutional code by simulation.

(2) *k9.f*. This program contains the encoder and decoder modules for the $K = 9, r = 1/2$ time-varying convolutional code. It determines the bit-error-rate performance of this convolutional code by simulation.

(3) *bchk9.f*. This program contains the encoder and decoder modules for the $K = 9, r = 1/2$ time-varying convolutional code as well as the encoder and decoder modules for the (128,113) extended BCH code. It determines the bit-error-rate performance of the concatenated system consisting of the extended BCH code as the outer code and the time-varying convolutional code as the inner code, with interleaving.

The software will be transported to a Modcomp 7845 computer and integrated with the Phobos Lander software package being developed by ICI of Spain.

In this article, theoretical and simulation results on the Phobos Lander coding system are presented. The $K = 6, r = 1/2$ convolutional code is discussed in Section II. The $K = 9$, $r = 1/2$ time-varying convolutional code is delineated in Section III. The concatenated system consisting of the extended BCH code and the time-varying convolutional code is outlined in Section IV. Finally, the detailed decoding algorithm of the extended BCH code and the weight enumerator and the decoder error probability of the extended BCH code are presented in Appendix A and Appendix B, respectively.

II. $K = 6, r = 1/2$ Convolutional Code

When operating in the first mode, the Phobos Lander spacecraft will convolutionally encode the telemetry data using a constraint length 6, rate 1/2 convolutional encoder as shown in Fig. 2. The Phobos Lander telemetry processor software will provide the capability to decode convolutionally encoded telemetry data ($K = 6, r = 1/2$) using a maximum-likelihood decoding algorithm (Viterbi algorithm). This is an existing DSN capability which will be integrated into the Phobos Lander processor.

The (6,1/2) convolutional code used has connection vectors $g_1 = 73$ and $g_2 = 61$ (in octal), is transparent, and has a free distance d_f equal to 8. The truncation length of the Viterbi decoder is a variable parameter and was set to 49 in our simulations. The choice of truncation length of the decoder is somewhat arbitrary as long as it is larger than approximately 5K, which guarantees a small performance degradation due to path truncation. By assuming perfect synchronization on the biphase modulated symbol stream on an additive white Gaussian noise (AWGN) channel, the bit-error-rate (BER) performance of the code is simulated as a function of the bit signal-to-noise ratio (E_b/N_0), and the result is given in Fig. 3. All the simulation results shown in this article assume that the received symbols are unquantized.

III. $K = 9, r = 1/2$ Time-Varying Convolutional Code

The second operational mode uses a constraint length 9, rate 1/2, time-varying convolutional code. The connection vectors are $g_1 = 557, g_2 = 663, g_3 = 711, g_4 = 745$ (in octal) and are used in pairs according to the periodic sequence (g_1, g_2), (g_2, g_3), (g_3, g_4), (g_4, g_1), ..., with period 4, as shown in Fig. 4. This code has $d_f = 10$, which is far from optimal, since the Plotkin upper bound applied to convolutional codes gives $d_f \leqslant 12$. For this code the minimum distance error events are always at distance 10 independent of their starting point within the period (phase), but the number of such events depends on the phase.

In fact, the best-known fixed convolutional code with $K = 9, r = 1/2$ has connection vectors 561,753 (in octal) and $d_f = 12$, which achieves the bound. Figure 5 shows the performance of this code compared to the code chosen for Phobos Lander on an AWGN channel, with perfect synchronization and a truncation length of 65 bits for both codes. The same two codes concatenated with the BCH code are compared in Fig. 6. It is well known that every periodic convolutional code with period T and rate k/n has an equivalent fixed code of rate Tk/Tn. In our case, the fixed code has rate 4/8 and $d_f = 10$, but now error events can start only every four bits, since the encoder inputs four bits at a time.

IV. Concatenated Coding System

When operating in the third mode, the Phobos Lander spacecraft will first encode the telemetry data using a (128, 113) extended BCH code. Each data frame contains 19 BCH code words ($19 \times 113 = 2147$ bits) and is padded with 99 zeros to obtain the desired frame length of 2048 bits. The

actual frame structure is still undefined, but it is natural to envision that the 99 additional bits could be used as a frame marker. The BCH code words are then interleaved to depth 16 before they are convolutionally encoded by a (9,1/2) time-varying convolutional code. The DSN Phobos Lander telemetry processor software will provide the capability to first decode the received data stream (convolutionally encoded) using a maximum-likelihood decoding algorithm. The output data of the Viterbi decoder will then be deinterleaved before they are block decoded using a simple but efficient BCH algebraic decoding algorithm. The detailed decoding algorithm for the extended BCH code is given in Appendix A, and the weight enumerator and the decoder error probability of the extended BCH code are given in Appendix B.

The (128,113) extended BCH code consists of a (127,113) BCH code plus an overall parity bit. The (127,113) BCH code has a generator polynomial $g(x) = 1 + x + x^2 + x^4 + x^5 + x^6 + x^8 + x^9 + x^{14}$ and is capable of correcting two errors. The schematic diagram of the encoder of the (127,113) BCH code is given in Fig. 7. With the addition of a parity check bit to the BCH code word, the code is also capable of detecting three errors. The (9,1/2) time-varying convolutional code is the same as that discussed in Section IV. Again, by assuming perfect synchronization on the symbol stream in a AWGN channel, the BER performance of the code is simulated and the result is given in Fig. 8. Figure 8 also shows a comparison of the three operating modes of Phobos Lander.

V. Conclusion

The results obtained in this article give a first performance evaluation of the Phobos Lander telemetry system. The software modules will be integrated in the Phobos Lander telemetry processor and used at the DSN stations to support this mission.

Acknowledgment

The authors would like to thank Dr. R. J. McEliece of Caltech's Electrical Engineering Department for his helpful suggestions on the decoding algorithms.

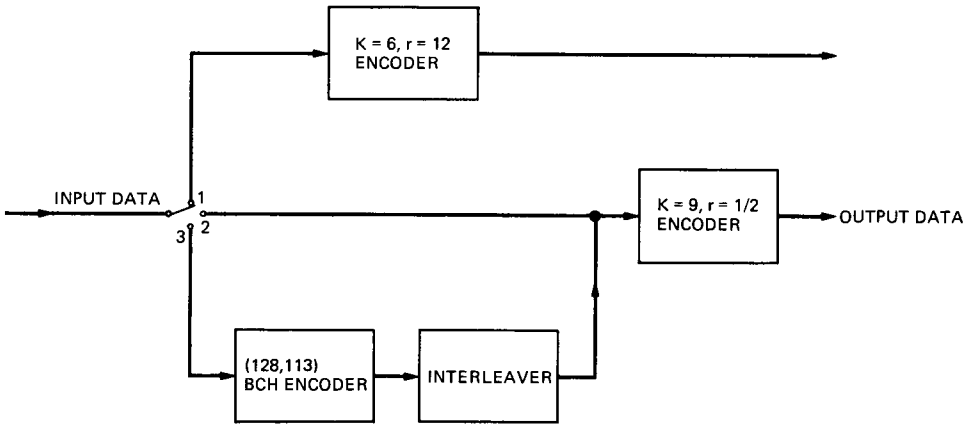


Fig. 1. Phobos Lander coding system

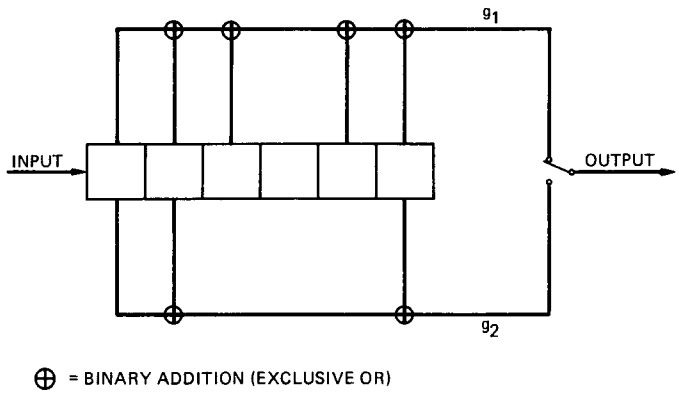


Fig. 2. The $K = 6$, $r = 1/2$ encoder

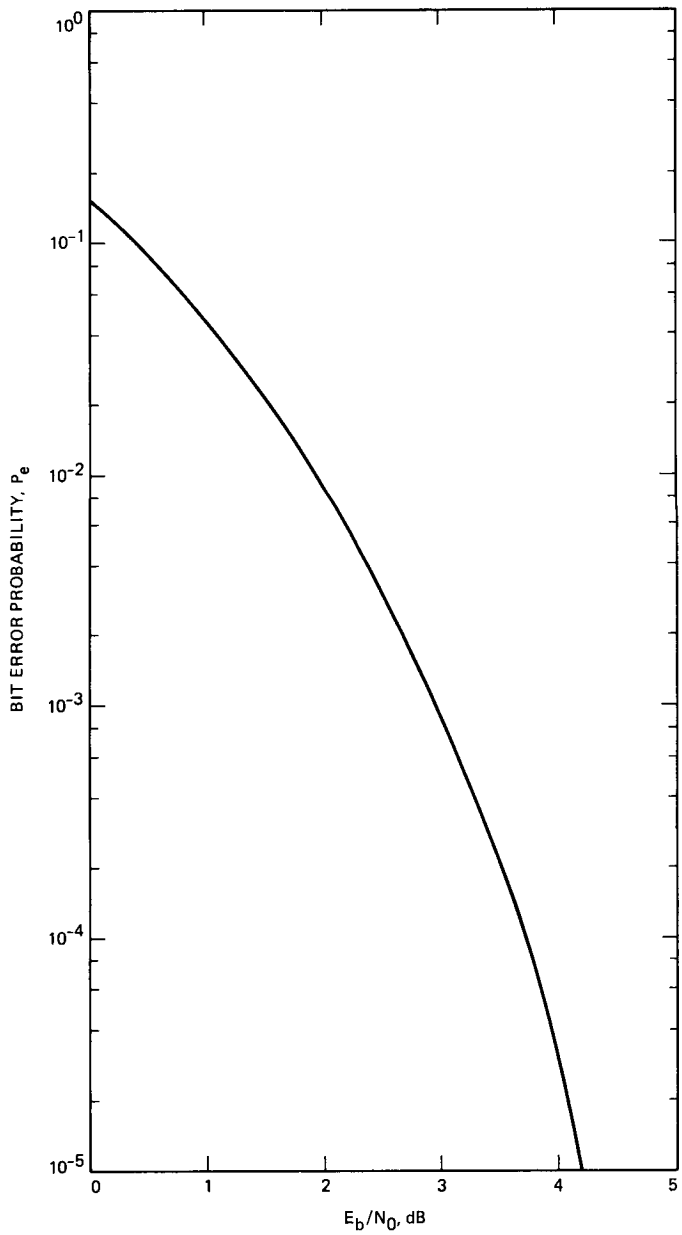
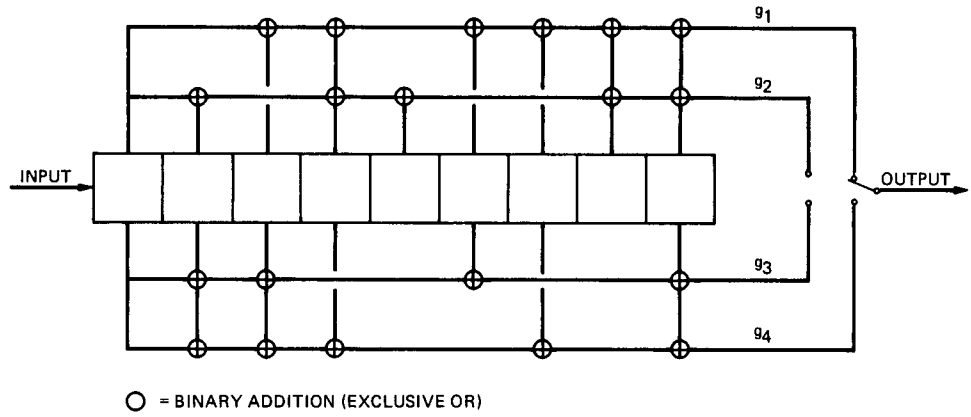


Fig. 3. Performance of the $K = 6$, $r = 1/2$ code



INPUT BITS	1		2		3		4	
OUTPUT SYMBOLS	g_1	g_2	g_2	g_3	g_3	g_4	g_4	g_1

Fig. 4. The $K = 9$, $r = 1/2$ encoder

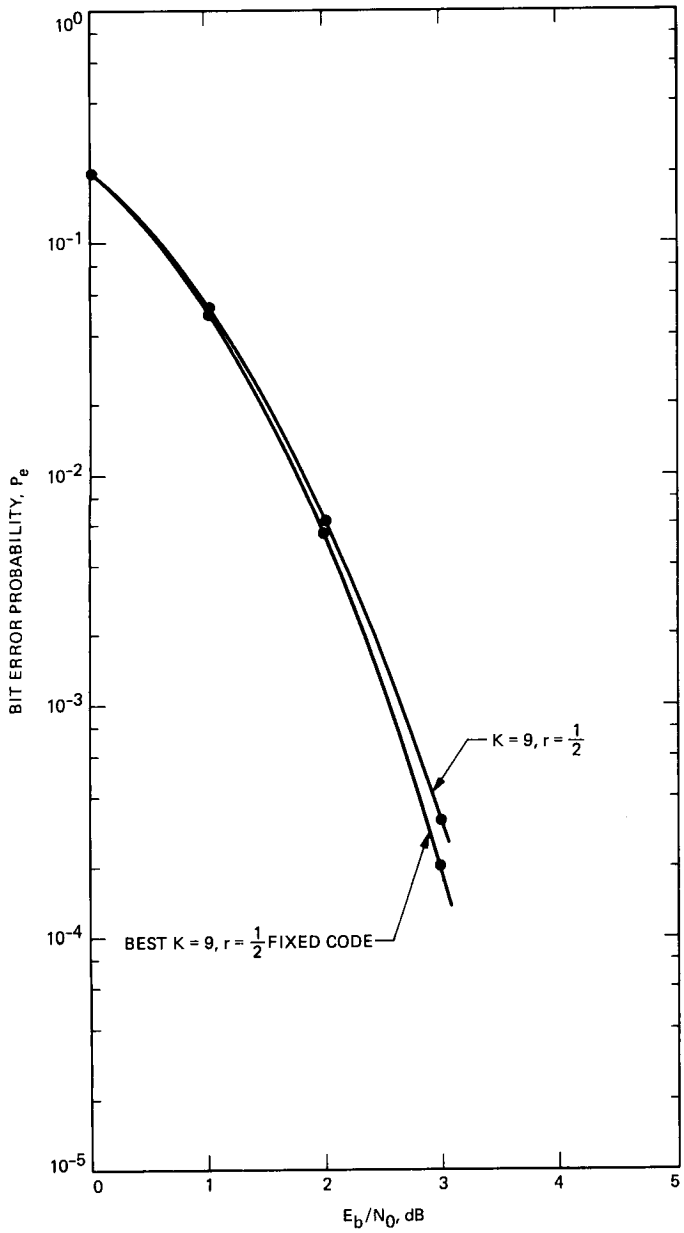


Fig. 5. Performance of two $K = 9$, $r = 1/2$ codes

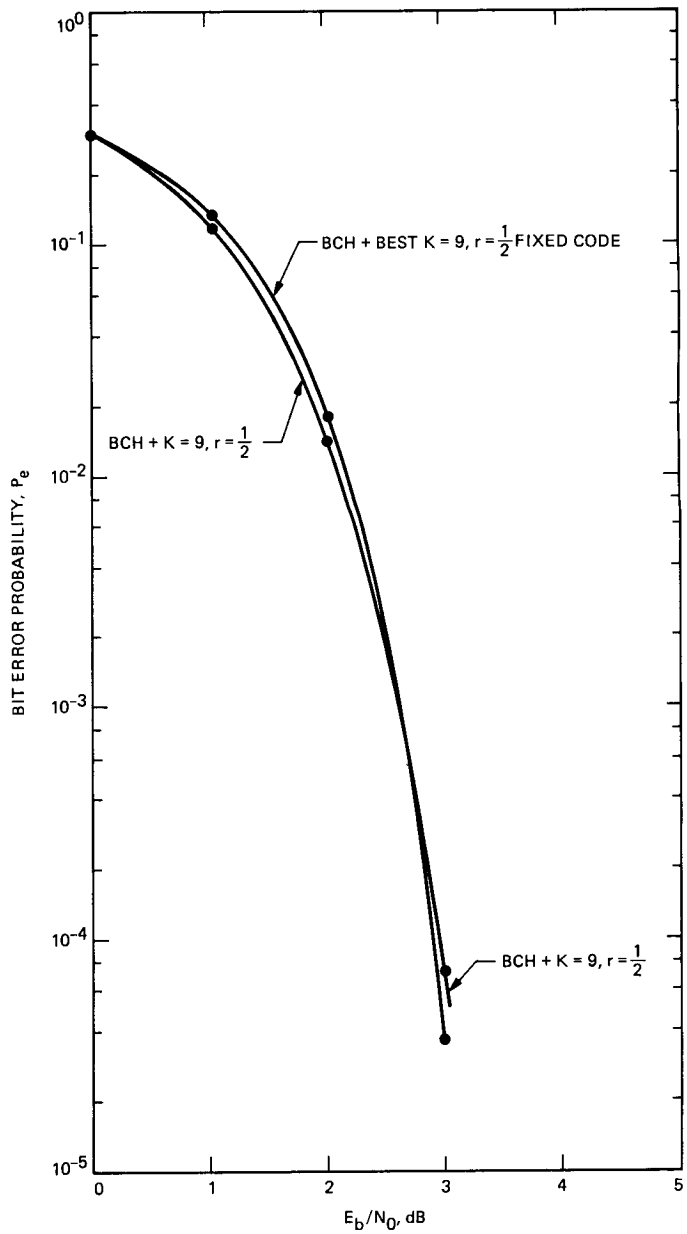


Fig. 6. Performance of two $K = 9$, $r = 1/2$ codes, concatenated with the BCH code

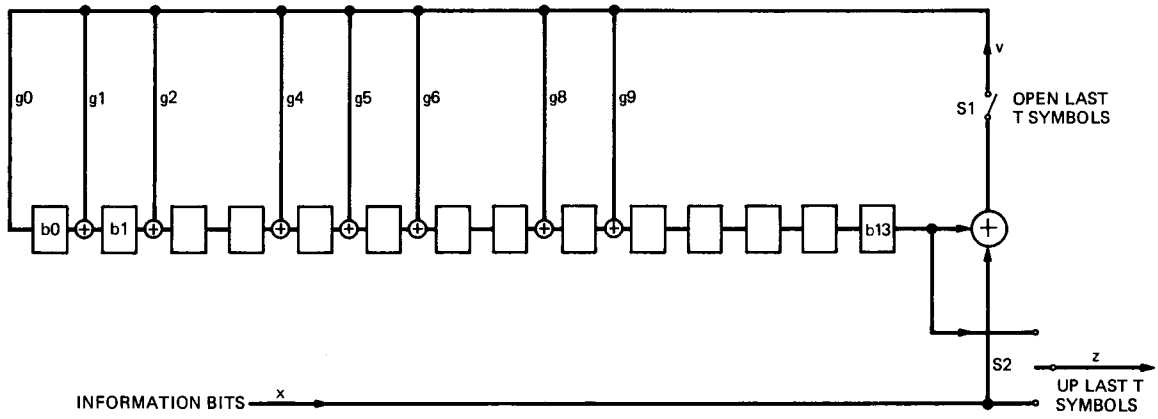


Fig. 7. Encoder for the (127,113) BCH code

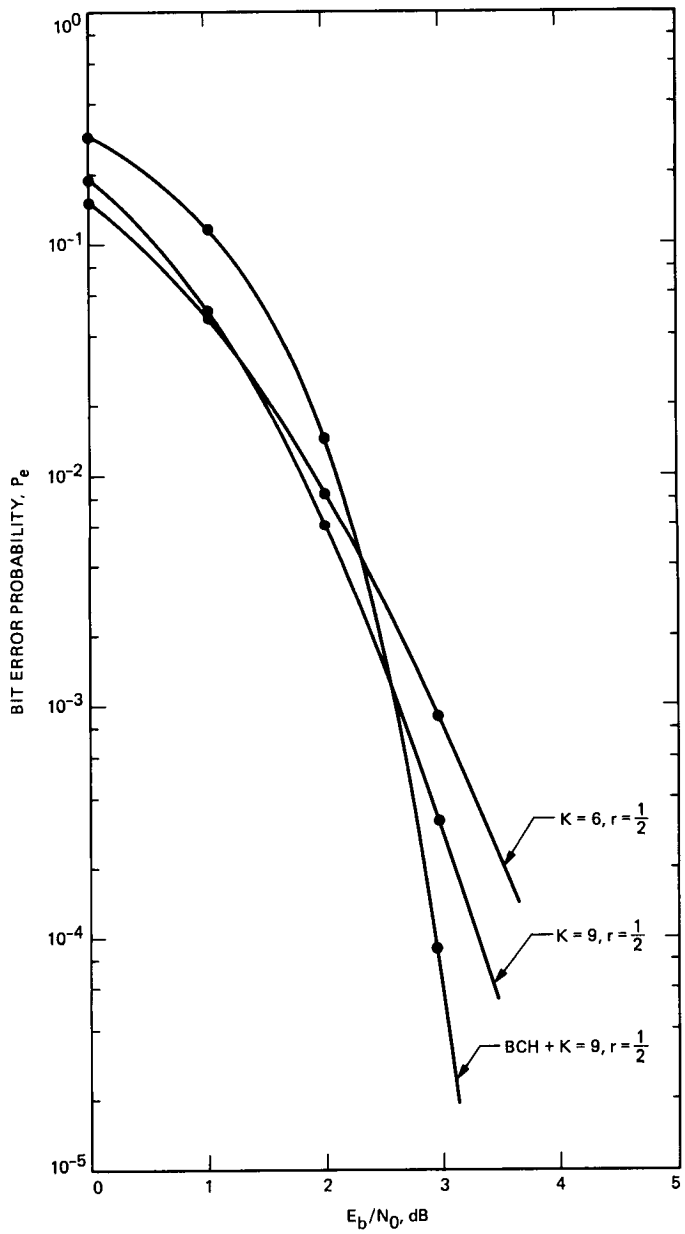


Fig. 8. Performance of a concatenated system

Appendix A
Decoding the (128,113) Extended BCH Code

The (128,113) extended BCH code consists of a (127,113) BCH code plus an overall parity bit. The generator polynomial for the (127,113) BCH code is $g(x) = 1 + x + x^2 + x^4 + x^5 + x^6 + x^8 + x^9 + x^{14}$. The (127,113) BCH code can correct two errors. With the introduction of an overall parity bit, the extended code can also detect three errors.

The decoding of the (127,113) BCH code requires finite field manipulations. A primitive irreducible polynomial $f(x) = x^7 + x^3 + 1$ of order 7 over $GF(2)$ is used to construct $GF(2^7)$. Let α be a root of $f(x)$ (i.e., $f(\alpha) = 0$). Then for all $\lambda \in GF(2^7)$, λ can be represented as a linear combination of $1, \alpha, \ldots, \alpha^6$ over $GF(2^7)$.

Let $g_1(x)$ and $g_3(x)$ denote the minimal polynomials of α and α^3, respectively (i.e., $g_1[\alpha] = 0$ and $g_3[\alpha^3] = 0$). It can be observed that $g(x) = g_1(x)g_3(x)$. Let $\underline{C} = [C_0, C_1, \ldots, C_{126}]$ be a BCH code word, where $C_i \in GF(2)$. Let $C(x)$ denote the polynomial $C_0 + C_1x + \ldots + C_{126}x^{126}$. The following equations are then obtained [A-1]:

$$C(\alpha) = 0 \tag{A-1}$$

and

$$C(\alpha^3) = 0 \tag{A-2}$$

Let $\underline{C'}$ be the received pattern and let $\underline{E}$ be the error pattern. Then $\underline{C'} = \underline{C} + \underline{E}$. Let

$$C'(\alpha) = S_1 \tag{A-3}$$

and

$$C'(\alpha^3) = S_3 \tag{A-4}$$

where S_1, S_3 are known as the syndromes and $S_1, S_3 \in GF(2^7)$. It is shown in [A-1] that this code can correct two or fewer errors. The correction of a single error [A-1] is trivial. The correction of two errors, however, involves solving the following quadratic equation:

$$x^2 + S_1x + \left(\frac{S_2}{S_1} + S_1^2\right) = 0 \qquad (S_1 \neq 0) \tag{A-5}$$

This would normally require an exhaustive search for elements in $GF(2^7)$ which satisfy Eq. (A-5). This exhaustive search can be bypassed by transforming Eq. (A-5) to the form $x^2 + x = \beta$, where $\beta \in GF(2^7)$. By using the concept of the trace of an element in a finite field [A-2], the solvability of Eq. (A-5) can be determined by testing if $Tr(\beta) = 0$. If Eq. (A-5) is solvable, the two roots of that equation, which indicate the two error locations in the code word, can be found readily with simple algebra. The details of this algorithm are discussed in [A-1] and [A-2].

The detection of three errors is done by considering the overall parity of the 128 received symbols as well as the syndromes S_1 and S_3. The complete decoding algorithm for the (128,113) extended BCH code is given in Fig. A-1.

References

[A-1] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, Amsterdam: North-Holland, 1977.

[A-2] R. J. McEliece, *Finite Fields for Computer Scientists and Engineers*, Norwell, Massachusetts: Kluwer Academic Publishers, 1987.

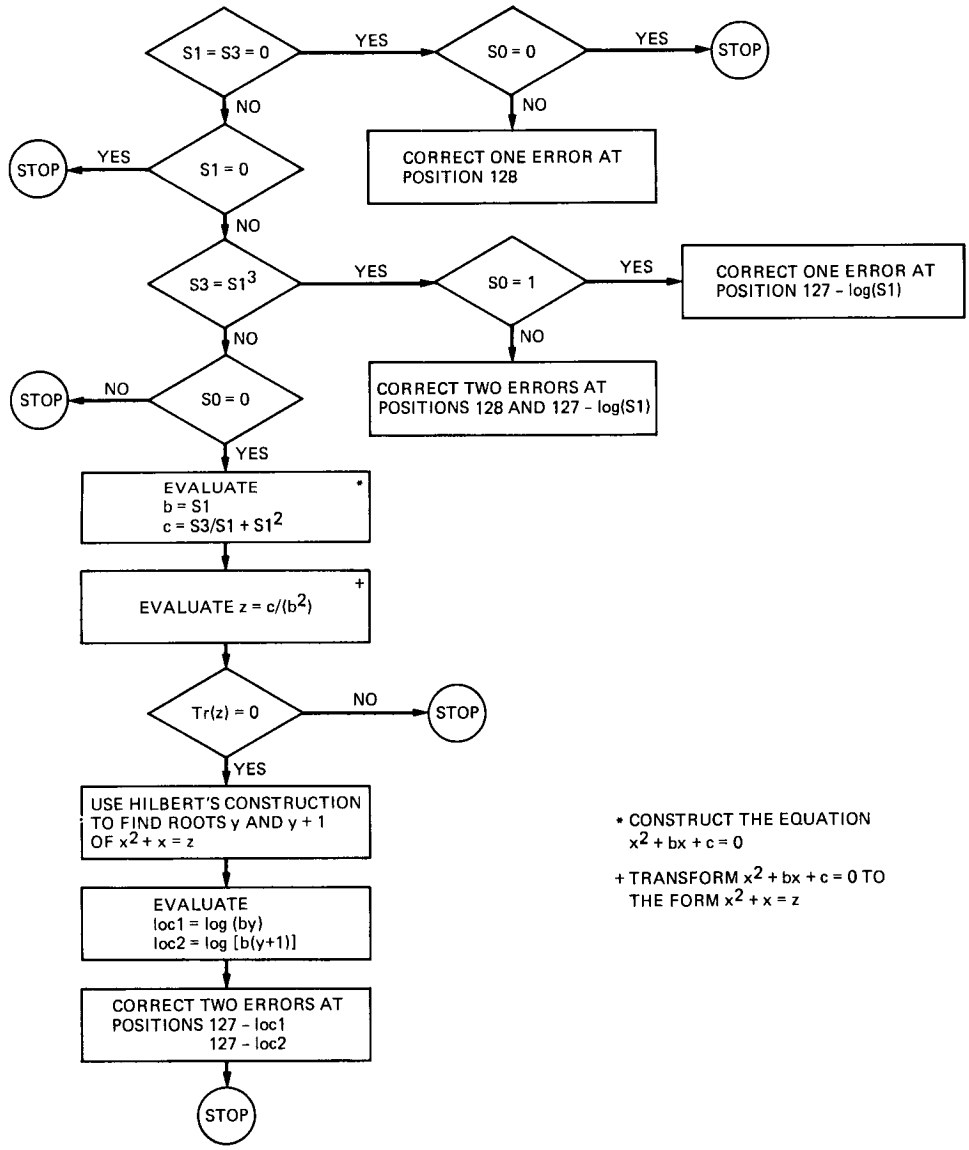


Fig. A-1. Decoding algorithm for the (128,113) BCH code

Appendix B

The Weight Enumerator and the Decoder Error Probability of the (128,113) Extended BCH Code

The decoder error probability $P_E(u)$ of a code is defined to be the conditional probability that a code word is being misdecoded to another code word given that an error pattern of weight u occurs. If all error patterns of weight u are equally probable, then $P_E(u)$ is given by the following expression [B-1], [B-2] (assuming the code is binary):

$$P_E(u) = \frac{D_u}{\binom{n}{u}} \tag{B-1}$$

where D_u denotes the number of decodable words of weight u.

In order to find D_u, the weight enumerator A_u of the code (the number of code words of weight u) must first be evaluated. The weight enumerator of the dual of the (128,113) extended BCH code, which is denoted by B_u, is given in [B-3]. A_u is then obtained by using MacWilliams's identity on binary codes [B-4]:

$$A(z) = 2^{-(n-k)}(1+z)^n B\left(\frac{1-z}{1+z}\right) \tag{B-2}$$

where $A(z) = A_0 + A_1 z + \ldots + A_n z^n$ and $B(z) = B_0 + B_1 + \ldots + B_n z^n$. For the dual of the (128,113) extended BCH code, $B_0 = 1$, $B_{56} = 8128$, $B_{64} = 16{,}510$, $B_{72} = 8128$, and $B_{128} = 1$. The A_i's are then found exactly by using MacWilliams's identity, and their values are tabulated in Fig. B-1 for even i, since $A_i = 0$ for odd i. The number of decodable words of weight u, D_u, is then calculated as follows [B-1]:

$$D_4 = \binom{6}{2} A_6$$

$$D_5 = \binom{6}{1} A_6$$

$$D_6 = A_6 + \binom{6}{1}\binom{122}{1} A_6 + \binom{8}{2} A_8$$

$$D_{2n-1} = \binom{130-2n}{1} A_{2n-2} + \binom{2n}{1} A_{2n}$$

$$D_{2n} = \binom{130-2n}{2} A_{2n-2} + A_{2n} + \binom{128-2n}{1}\binom{2n}{1} A_{2n} + \binom{2n+2}{2} A_{2n+2}$$

where $4 \leqslant n \leqslant 64$. The decoder error probability $P_E(u)$ is calculated using Eq. (B-1). The values of $P_E(u)$ are tabulated in Fig. B-2. It is observed that $P_E(u)$ "oscillates" between two values: 0.496 and 0.00781, depending on whether u is even or odd. This observation will be discussed in more detail in a forthcoming progress report.

References

[B-1] R. J. McEliece and L. Swanson, "On the Decoder Error Probability for Reed-Solomon Codes," *IEEE Tran. Inform. Theory*, vol. IT-32, pp. 701–703, September 1986.

[B-2] K.-M. Cheung, "More on the Decoder Error Probability for Reed-Solomon Codes," *TDA Progress Report 42-91*, vol. July–September 1986, Jet Propulsion Laboratory, Pasadena, California, pp. 213–223, November 15, 1987.

[B-3] E. R. Berlekamp, *Algebraic Coding Theory*, Laguna Hills, California: Aegean Park Press, 1984.

[B-4] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications*, Englewood Cliffs, New Jersey: Prentice-Hall, 1983.

```
A[  0] = 1.000e+000
A[  2] = 0.000e+000
A[  4] = 0.000e+000
A[  6] = 3.414e+005
A[  8] = 8.729e+007
A[ 10] = 1.384e+010
A[ 12] = 1.448e+012
A[ 14] = 1.061e+014
A[ 16] = 5.697e+015
A[ 18] = 2.315e+017
A[ 20] = 7.303e+018
A[ 22] = 1.827e+020
A[ 24] = 3.683e+021
A[ 26] = 6.070e+022
A[ 28] = 8.272e+023
A[ 30] = 9.413e+024
A[ 32] = 9.020e+025
A[ 34] = 7.332e+026
A[ 36] = 5.087e+027
A[ 38] = 3.029e+028
A[ 40] = 1.555e+029
A[ 42] = 6.914e+029
A[ 44] = 2.672e+030
A[ 46] = 8.998e+030
A[ 48] = 2.649e+031
A[ 50] = 6.834e+031
A[ 52] = 1.548e+032
A[ 54] = 3.082e+032
A[ 56] = 5.406e+032
A[ 58] = 8.359e+032
A[ 60] = 1.141e+033
A[ 62] = 1.374e+033
A[ 64] = 1.462e+033
A[ 66] = 1.374e+033
A[ 68] = 1.141e+033
A[ 70] = 8.359e+032
A[ 72] = 5.406e+032
A[ 74] = 3.082e+032
A[ 76] = 1.548e+032
A[ 78] = 6.834e+031
A[ 80] = 2.649e+031
A[ 82] = 8.998e+030
A[ 84] = 2.672e+030
A[ 86] = 6.914e+029
A[ 88] = 1.555e+029
A[ 90] = 3.029e+028
A[ 92] = 5.087e+027
A[ 94] = 7.332e+026
A[ 96] = 9.020e+025
A[ 98] = 9.413e+024
A[100] = 8.272e+023
A[102] = 6.070e+022
A[104] = 3.683e+021
A[106] = 1.827e+020
A[108] = 7.303e+018
A[110] = 2.315e+017
A[112] = 5.697e+015
A[114] = 1.061e+014
A[116] = 1.448e+012
A[118] = 1.384e+010
A[120] = 8.729e+007
A[122] = 3.414e+005
A[124] = 0.000e+000
A[126] = 0.000e+000
A[128] = 1.000e+000
```

Fig. B-1. Weight enumerator of the (128,113) BCH code

```
Pe(  0) = 0.0000000000
Pe(  1) = 0.0000000000
Pe(  2) = 0.0000000000
Pe(  3) = 0.0000000000
Pe(  4) = 0.4800000000
Pe(  5) = 0.0077419355
Pe(  6) = 0.4967812973
Pe(  7) = 0.0078282110
Pe(  8) = 0.4960509691
Pe(  9) = 0.0078097428
Pe( 10) = 0.4960908022
Pe( 11) = 0.0078116229
Pe( 12) = 0.4959824226
Pe( 13) = 0.0078096133
Pe( 14) = 0.4961009553
Pe( 15) = 0.0078118830
Pe( 16) = 0.4962086132
Pe( 17) = 0.0078135469
Pe( 18) = 0.4961605096
Pe( 19) = 0.0078124440
Pe( 20) = 0.4961972575
Pe( 21) = 0.0078132943
Pe( 22) = 0.4961436700
Pe( 23) = 0.0078121348
Pe( 24) = 0.4961385234
Pe( 25) = 0.0078122653
Pe( 26) = 0.4961611465
Pe( 27) = 0.0078126833
Pe( 28) = 0.4961716826
Pe( 29) = 0.0078127893
Pe( 30) = 0.4961683532
Pe( 31) = 0.0078126913
Pe( 32) = 0.4961742825
Pe( 33) = 0.0078128440
Pe( 34) = 0.4961790413
Pe( 35) = 0.0078128959
Pe( 36) = 0.4961759676
Pe( 37) = 0.0078128097
Pe( 38) = 0.4961297598
Pe( 39) = 0.0078118161
Pe( 40) = 0.4961092886
Pe( 41) = 0.0078117645
Pe( 42) = 0.4961724062
Pe( 43) = 0.0078132626
Pe( 44) = 0.4961930461
Pe( 45) = 0.0078130283
Pe( 46) = 0.4961573712
Pe( 47) = 0.0078122672
Pe( 48) = 0.4961524476
Pe( 49) = 0.0078125765
Pe( 50) = 0.4962063144
Pe( 51) = 0.0078137769
Pe( 52) = 0.4962010077
Pe( 53) = 0.0078128671
Pe( 54) = 0.4961540703
Pe( 55) = 0.0078122035
Pe( 56) = 0.4961433166
Pe( 57) = 0.0078124122
Pe( 58) = 0.4962101574
Pe( 59) = 0.0078141525
Pe( 60) = 0.4962602342
Pe( 61) = 0.0078141898
Pe( 62) = 0.4962243897
Pe( 63) = 0.0078130264
Pe( 64) = 0.4961881147
Pe( 65) = 0.0078130264
Pe( 66) = 0.4962243897
Pe( 67) = 0.0078141898
Pe( 68) = 0.4962602342
Pe( 69) = 0.0078141525
Pe( 70) = 0.4962101574
Pe( 71) = 0.0078124122
Pe( 72) = 0.4961433166
Pe( 73) = 0.0078122035
Pe( 74) = 0.4961540703
Pe( 75) = 0.0078128671
Pe( 76) = 0.4962010077
Pe( 77) = 0.0078137769
Pe( 78) = 0.4962063144
Pe( 79) = 0.0078125765
Pe( 80) = 0.4961524476
Pe( 81) = 0.0078122672
Pe( 82) = 0.4961573712
Pe( 83) = 0.0078130283
Pe( 84) = 0.4961930461
Pe( 85) = 0.0078132626
Pe( 86) = 0.4961724062
Pe( 87) = 0.0078117645
Pe( 88) = 0.4961092886
Pe( 89) = 0.0078118161
Pe( 90) = 0.4961297598
Pe( 91) = 0.0078128097
Pe( 92) = 0.4961759676
Pe( 93) = 0.0078128959
Pe( 94) = 0.4961790413
Pe( 95) = 0.0078128440
Pe( 96) = 0.4961742825
Pe( 97) = 0.0078126913
Pe( 98) = 0.4961683532
Pe( 99) = 0.0078127893
Pe(100) = 0.4961716826
Pe(101) = 0.0078126833
Pe(102) = 0.4961611465
Pe(103) = 0.0078122653
Pe(104) = 0.4961385234
Pe(105) = 0.0078121348
Pe(106) = 0.4961436700
Pe(107) = 0.0078132943
Pe(108) = 0.4961972575
Pe(109) = 0.0078124440
Pe(110) = 0.4961605096
Pe(111) = 0.0078135469
Pe(112) = 0.4962086132
Pe(113) = 0.0078118830
Pe(114) = 0.4961009553
Pe(115) = 0.0078096133
Pe(116) = 0.4959824226
Pe(117) = 0.0078116229
Pe(118) = 0.4960908022
Pe(119) = 0.0078097428
Pe(120) = 0.4960509691
Pe(121) = 0.0078282110
Pe(122) = 0.4967812973
Pe(123) = 0.0077419355
Pe(124) = 0.4800000000
Pe(125) = 0.0000000000
Pe(126) = 1.0000000000
Pe(127) = 1.0000000000
Pe(128) = 1.0000000000
```

Fig. B-2. Decoder error probability of the (128,113) BCH code