NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Technical Report 32-1275

Error Correction for Deep Space Network Teletype Circuits

Harold Fredricksen

JET PROPULSION LABORATORY

CALIFORNIA INSTITUTE OF TECHNOLOGY

PASADENA, CALIFORNIA

June 1, 1968

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Technical Report 32-1275

Error Correction for Deep Space Network Teletype Circuits

Harold Fredricksen

Approved by:

R M Goldstein

R. M. Goldstein, Manager
Communications Systems Research Section

JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA

June 1, 1968

Contents

Abstract

The design of a system for controlling errors in teletype transmission is described in this report. A code with high error-correction capability was developed following an analysis of the error statistics of the channel. That code is detailed, the decoding algorithm is given, a communications processor for directing communications within the station is described, and the performance of the code and operation of the communications processor on a set of coded messages—transmitted from an overseas site—is presented.

Error Correction for Deep Space Network Teletype Circuits

I. Introduction

This report presents a system for controlling errors on Deep Space Network (DSN) interconnecting teletype links. It will be demonstrated that, by using very simple codes and very straightforward ideas for decoding, extensive classes of teletype errors can be corrected and the time-expensive necessity for repeat transmissions can be largely eliminated, while at the same time, a very high degree of reliability can be maintained. To minimize repetitive transmissions is especially desirable for the command links *outbound* from the Space Flight Operations Facility (SFOF) at the Jet Propulsion Laboratory in Pasadena, California, to the Deep Space Stations. An additional advantage in the use of ground-line coding is for *inbound* telemetry links to JPL. Because it is very costly for a tracking station to back up and repeat information, and an error in transmission of telemetry from the tracking station to the SFOF has to be tolerated, the error-correction process in this instance is doubly beneficial.

The need for more reliable ground communication is evident. By encoding data for ground transmission, very reliable information can be transmitted in conditions of noise observed on the communication channel used by the DSN. That these results could be achieved was demonstrated by observing transmissions on the channel and obtaining error-rate statistics. The means for developing the program and the equipment used are described in the following sections. The demonstrated system works over normal, existing NASA Communications System (NASCOM) teletype links.

II. Design of the Error-Rate Experiments

Two different sets of experiments were carried out to measure channel statistics. For the second set, which was patterned after the first, however, some of the data gathered for the first test was found to be unimportant and not included in the later test. Also, a slightly different format was selected for the second experiment. This new format was, in part, dictated by the results of the first experiment.

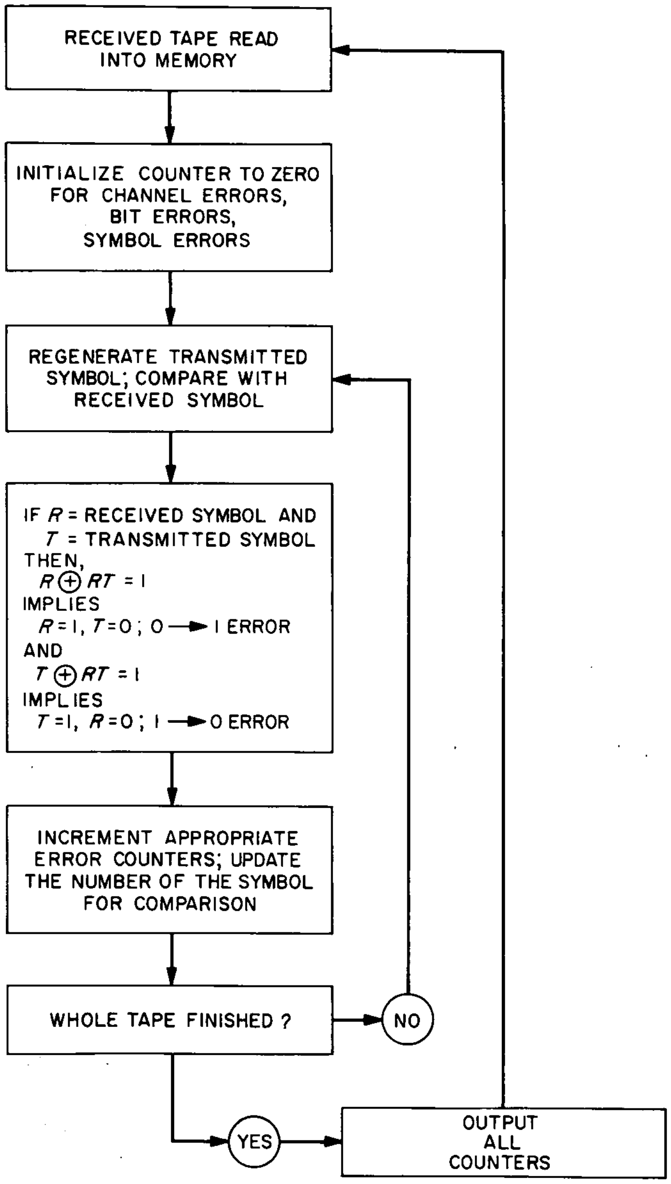
The tests consisted of the formation of a sample message on a tape; this message was then transmitted as a regular message over the NASCOM links. These tapes were sent on a twice daily basis to all DSN stations.

In the first test, the receiving station would, upon receiving the message tape, transmit the message back to the originating station (for all tests, the originating station was the Venus Deep Space Station [DSS 13] at Goldstone) and would then return its received message to JPL via air freight. The message received at the overseas station would, in most cases, contain errors that arose in the transmission. These errors would be observed in the version transmitted from the overseas station to Goldstone, in addition to any errors in the return path.

In fact, where tapes could be paired up as being the one-way and two-way versions of the same message, it was observed that, almost without exception, the errors introduced into the message had been detected in both tapes; this fact indicated that the message from Goldstone to the tracking station had traveled through a noisier environment. While the reason for this phenomenon was never determined, it was decided to run messages on a one-way basis only—i.e., on the outward link—for the second set of experiments. It is over this most critical channel that command messages—which demand greater reliability than usual telemetry—travel; therefore, it was most protected by the coding.

The encoding of the test messages was accomplished on an SDS-910 computer[1] at the Venus Deep Space Station at Goldstone; the later computer evaluation of the received tapes was also done there. The message was formed from the output of the maximal-length degree-23 shift register whose output satisfies the recursion $a_{n+23} + a_{n+5} + a_n = 0$. As the bits were produced by the shift register, they were encoded in blocks of 5 for transmission as teletype characters, since the teletype code is a *five-level* code. The first sample message was 3200 teletype characters long, or 640 s long at the standard teletype rate of 25 bits/s (60 words/min in teletype codes), which was then standard speed. The message was stored on Mylar tape and was used for each transmission. The second master tape was prepared with a slightly different format. To each 4 bits produced by the shift register there was adjoined a 5th bit to serve as an *odd parity* check on the first 4 bits. (The reason for this will be explained later.) This tape was 3184 characters long.

Because of the random character of the message, a printed version had a distinctive appearance and the transmission came to be known as the *garbage message.* Regular punctuation that occurs in most messages at the end of sentences, lines, etc., appeared randomly throughout the message, so yielded no clear sentence or line structure. At stations all over the world, operators monitoring traffic over these lines had to give up any attempt to observe transmissions and had to fight a tendency to *clear* the line. Nonetheless, it was essential to retain the pseudorandom character of the message to be able to perform the analysis on the received tapes in the most reliable manner. By term-by-term mod-2 addition of the received message to the transmitted message, the errors were tallied by counting the number of *ones* in the result. In the first experiment, a *one* in the mod-2 sum of the two words appeared if either a *one* changed into a *zero* or vice versa. A distinction was made between these two types of errors in the second experiment.



RECEIVED TAPE READ INTO MEMORY
INITIALIZE COUNTER TO ZERO FOR CHANNEL ERRORS, BIT ERRORS, SYMBOL ERRORS
REGENERATE TRANSMITTED SYMBOL; COMPARE WITH RECEIVED SYMBOL
IF R = RECEIVED SYMBOL AND T = TRANSMITTED SYMBOL THEN, $R \oplus RT = 1$ IMPLIES $R = 1, T = 0$; $0 \rightarrow 1$ ERROR AND $T \oplus RT = 1$ IMPLIES $T = 1, R = 0$; $1 \rightarrow 0$ ERROR
INCREMENT APPROPRIATE ERROR COUNTERS; UPDATE THE NUMBER OF THE SYMBOL FOR COMPARISON
WHOLE TAPE FINISHED ?
NO
YES
OUTPUT ALL COUNTERS

Fig. 1. Error-counting program

[1]Scientific Data Systems.

The errors were tabulated in a number of ways. The number of (five-level) characters, as well as the number of bits in error, was counted. A further breakdown by levels gave the number of bit errors in each of the five levels. Also counted was the number of times various patterns of errors occurred; and to check for *burstiness* of the channel, the number of characters in error that occurred successively was counted. The channel was, in general, nonbursty at the character level, so in the second experiment, this burst statistic was not gathered. In the second experiment, the number of bit errors per character was counted and the pattern error count was eliminated. Figure 1 is a flow chart of the error-counting program. The Appendix contains some sample data sheets.

III. Analysis of the Test and Evaluation of the Channel

Estimates of the error rates for the channel solicited in advance from various sources ranged from 0.01 to 0.05 for a worst-case character-error rate. We found that character-error rate was about 10^{-3}. For the purpose of code design, this worst-case error rate of 0.01 was used in all calculations.

Table 1 gives a breakdown of errors for the second set of tapes. Error rates and patterns were fairly evenly distributed over the various stations. The Johannesburg channel showed the worst performance, although the error rate was only 1.1×10^{-3}, as compared with an overall rate of 8×10^{-4} for all stations. In general, a higher error rate is expected on the Johannesburg channel because of the high-frequency radio link between London and the Pretoria, South Africa, station.

The distribution of errors in Table 1 in each channel of the teletype characters is reasonably uniform. That this is not always the case is exhibited in Table 2, which gives the distribution of errors in each level for the first set of tapes. A decided predominance toward error is seen in levels 4 and 5, which indicates a kind of machine failure. Dirty contacts tend to disturb the punching mechanism so that errors are observed. Since the errors were observed on tapes to all stations with approximately the same distribution, it may have been the sending machine that had the dirty contacts. This type of error is impossible to code for, since a dirty contact is not a random event. The best solution for this problem is to

Table 2. Data from first experiment

Error	Number of errors					
	Level 1	Level 2	Level 3	Level 4	Level 5	Total
0→1	135	121	84	603	634	1577
1→0	29	22	32	88	94	265
Combined	164	143	116	691	728	1842

Table 1. Data from second experiment

Station	Number of tapes	Number of errors																				
		Sync errors	1→0 channel errors, channel No.					0→1 channel errors, channel No.					Combined channel errors, channel No.					Pattern errors				
			1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	Single	Double	Triple	Quad	Quint
Woomera AOMJ	45	11	8	7	5	7	4	19	5	9	4	4	27	12	14	11	8	27	7	2	0	0
Canberra ANBE	40	10	6	7	2	9	9	6	5	8	3	4	12	12	10	12	13	37	7	3	3	0
Johannesburg LJOB	42	14	6	14	7	12	6	14	7	16	4	10	20	21	23	16	16	21	12	5	4	0
Madrid LRID	3	2	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0
All stations	130	37	20	28	14	29	19	39	17	33	11	18	59	45	47	40	37	86	26	10	7	0

keep the equipment in good repair. This is apparently a nontrivial problem for teletype machines operating in overseas locations.

The pattern errors that were mentioned in Section II and noted in Table 1 are very interesting. The single errors are those where a *one* is changed to a *zero*, or the reverse. The multiple errors break down into two really different kinds. The first kind seems to be just more pronounced occurrences of errors like the single errors.

However, there is another kind of multiple error that causes a symbol to change drastically, in the sense that the received symbol has no obvious relation to the transmitted symbol. This difference can best be explained by observing the path a message travels. While messages are traveling within the United States, there is no coding on the characters of the message. But at the stations in New York and San Francisco, where messages are prepared for transmission overseas, the teletype characters are encoded into 7-tuples, each containing four *ones*. These 7-tuples are then transmitted with the transmission monitored by the Van Duuren Automatic Repeat Query (ARQ) System. The ARQ system ensures that, of every 7 bits transmitted, there are exactly 4 *ones* present. If 4 *ones* are not present, the transmission is halted, and an automatic request for retransmission of the incorrect word is initiated, halting traffic in both directions. Then the last three characters held in a buffer are retransmitted. The ARQ remains in this mode until the offending symbol is transmitted with 4 *ones* in 7 bits. During noisy times, it is conjectured that both a *zero* → *one* and a *one* → *zero* change occur in some symbols. This leaves the symbol meeting the 4-in-7 criterion; however, the symbols defined by the two ARQ characters have no relation to one another and, in general, a multiple error in the teletype symbol can occur with higher probability than would be the case if errors were occurring independently. However, single errors in the ARQ will be detected and the automatic request for retransmission activated.

If a single error is made before the ARQ encoding, the new symbol will be encoded in the ARQ code and then decoded back to what was sent over the ARQ so that the single error is preserved. Errors of this type, be they single errors or multiple errors, follow a more reasonable probability distribution. It is interesting to note the distinction between these two classes of errors, but no simple coding scheme can take advantage of this knowledge. The ARQ system is a fixed block in the channel over which JPL has no control. A reevaluation of the mapping that carries teletype symbols into ARQ code symbols is indicated so that distances would be preserved and code words that are close in one metric would be close in the other.

Another result of the experiments worth noting is the difference between *zero* → *one* transitions and *one* → *zero* transitions. A large discrepancy between these types of errors is noted in Table 2. According to one theory, the difference is due to a mechanical functioning of the teletype machine. In the machine a *zero* is *space*, and a *one* is denoted by a *mark*. It turns out that channel noise occurring on the lines is more like a mark than a space so that noise has a tendency to change *zeroes* into *ones*. Of 110 tapes in the first set of experiments, on 38 occasions, only *zero* → *one* errors were observed, while 3 tapes have only *one* → *zero* errors. The second data set does not exhibit the imbalance to the same degree as the first experiment. No attempt was made to take advantage of this imbalance in the coding scheme, partly because no simple coding scheme can take good advantage of this fact.

The most important result of the testing was the discovery of synchronization errors. These errors occur when a group of symbols is inserted into, or deleted from, the body of the message by the ARQ system or by the NASCOM communications processor at Goddard Space Flight Center (GSFC), Greenbelt, Maryland. The effect of an error of this type on the data from these experiments is disastrous, because the correlation is essentially zero between a sequence of this type and itself shifted by some amount. So even though these errors were not frequent (they occur with probability $< 10^{-4}$), they can be very damaging to the communication. Therefore, it was necessary to provide some coding to detect errors of this type. The actual method used will be discussed in the section dealing with the choice of the code used for error correction.

IV. Introduction to Coding

Solomon (Ref. 1) gives an introduction to coding over finite fields slanted toward the applications in this report. Enough of that theory will be given here to make this report self-contained.

Teletype transmission of English text is often corrupted by errors, as anyone who has ever received a telegram or TWX will attest. Since teletype characters are really 5-tuples of *zeros* and *ones*, errors in reception of a teletype character are actually errors in the binary bits. For the most part, in an English message errors can be detected and, indeed, corrected, due to the high redundancy of English. However, if the message to be transmitted is just a sequence of *zeros* and *ones,* then an error sending one bit into another or an inserted or deleted symbol will not be noticeable by any context.

By coding the information and adding redundancy to the message, certain patterns of errors in transmission can be corrected. These redundant symbols can be added as *parity checks* on the information symbols. The parity checks can be chosen so that each code word is at least a certain distance (in some metric) from each of the other code words. The usual metric chosen in coding applications is the *Hamming metric*: two words are a distance d apart in the Hamming metric if they differ (have different symbols) in exactly d places. If the minimum distance between code words is d, then the code, taken as an error-correcting code, will correct up to $(d-1)/2$ errors. As an error-detecting code, it will detect up to $d-1$ errors.

A. Encoding

Codes over alphabets with more than two symbols can be obtained as follows: Group k bits together to form code symbols; the symbols are then considered to be elements of a field F with 2^k elements. *Addition* of elements is bit-by-bit mod-2 addition. To define multiplication takes more work.

If we choose a so-called *primitive element* β in the finite field, the nonzero elements are powers of β and multiplication in $F - \{0\}$ (the nonzero elements of F) is accomplished by adding the respective powers of β modulo $2^k - 1$. We used one of these codes, a Reed–Solomon code (Ref. 1), to solve the problems of errors occurring in the NASCOM lines discussed in previous sections. The Reed–Solomon (R–S) 15,9 code over the field of 2^4 elements was chosen. The 15 nonzero elements of that field are the powers of β where β is the solution to the irreducible equation $x^4 + x + 1 = 0$ over the field of two elements (consult Table 3). Each code word consists of 15 symbols from the field, 9 of which are information symbols, and the remaining 6 are check symbols. Thus, the code has rate $9/15 = 0.6$. The reason for using 4-tuples instead of 5-tuples will become apparent later.

The check symbols are computed by using the polynomial

$$F(x) = \prod_{i=0}^{8} (x + \beta^i) = x^9 + r_1 x^8 + \cdots + r_8 x^0$$

Table 3. The 16-element field

Field element	Beta polynomial	4-tuple	SDS code	Teletype code
0		(0,0,0,0)	0	T
1	1	(0,0,0,1)	1	Z
β	β	(0,0,1,0)	2	L
β^2	β^2	(0,1,0,0)	4	H
β^3	β^3	(1,0,0,0)	8	O
β^4	$\beta + 1$	(0,0,1,1)	3	A
β^5	$\beta^2 + \beta$	(0,1,1,0)	6	I
β^6	$\beta^3 + \beta^2$	(1,1,0,0)	/ (12)	N
β^7	$\beta^3 + \beta + 1$	(1,0,1,1)	= (11)	M
β^8	$\beta^2 + 1$	(0,1,0,1)	5	Y
β^9	$\beta^3 + \beta$	(1,0,1,0)	Space (10)	R
β^{10}	$\beta^2 + \beta + 1$	(0,1,1,1)	7	Q
β^{11}	$\beta^3 + \beta^2 + \beta$	(1,1,1,0)	> (14)	V
β^{12}	$\beta^3 + \beta^2 + \beta + 1$	(1,1,1,1)	$\vee$ (15)	F
β^{13}	$\beta^3 + \beta^2 + 1$	(1,1,0,1)	: (13)	X
β^{14}	$\beta^3 + 1$	(1,0,0,1)	9	D

Addition: Modulo-2 addition of 4-tuples.
Multiplication: $\beta^i \cdot \beta^j = \beta^{i+j}$;
$i+j$ is reduced modulo 15, if necessary.

so that

$$r_1 = \sum_{i=0}^{8} \beta^i,\ r_2 = \sum_{\substack{i \neq j \\ i,j=0}}^{8} \beta^i \beta^j,\ \cdots,\ r_8 = \prod_{i=0}^{8} \beta^i$$

If $x_0, x_1, \cdots, x_8$ are the nine information symbols, then the six check symbols are computed in sequence by multiplying the ninth oldest symbol by r_8, the eight oldest by r_7, etc., and adding the powers of β so computed, i.e.,

$$x_i = \sum_{j=1}^{9} r_j x_{i-j},\ i = 9, 10, 11, \cdots, 14$$

The polynomial

$$F(x) = \prod_{i=0}^{8} (x + \beta^i)$$

is called the *generator polynomial* of the code.

If $a_0, \cdots, a_8$ are the information symbols of a code word a, we associate with $F(x)$ the Mattson–Solomon polynomial (Ref. 1)

$$g_a(x) = \sum_{i=0}^{8} c_i x^i$$

where we find the c_i by solving the equations

$$g_a(\beta^j) = a_j, \qquad 0 \leq j \leq 8$$

Solution of those nine equations in nine unknowns is possible because of properties of determinants of powers of β.

It is then easy to see that $g_a(\beta^j) = a_j$, also, for $9 \leq j \leq 14$. Now $g_a(x)$ is of degree 8 or less (unless a is 0) and, therefore, has at most 8 *zeros*. Since the code word has 15 positions, any nonzero code word has at least 7 nonzero positions, and the minimum distance in the code is 7 or more; code words with exactly 7 nonzero positions are readily exhibited. We can, thus, correct all error patterns of 3 errors or fewer, since an error on 3 or fewer positions is closest to the original code word. We shall only correct double errors and we will then be able to detect all patterns of 4 errors (i.e., detect them and not try to correct them unless we can do so correctly).

To detect errors resulting from insertion or deletion of symbols, the following synchronization error-detection scheme has been devised: consider the code defined by

$$H(x) = (x + \beta^{-1})\, F(x)$$

The minimum distance between words in this code is 6, since the Mattson–Solomon polynomials have degrees 9 at most. Furthermore, any code word in the code F generated by $F(x)$ is in the code H generated by $H(x)$. We pick a vector of length 15 in H which is not in F, in particular, $\delta = (\beta^{14}, \beta^{13}, \beta^{12}, \cdots, \beta, 1)$. This vector is, therefore, at distance 6 or more from every vector in F. More importantly, since H is cyclic, any shift of this vector δ added to δ is in the code and, hence, has distance 6 or more from any vector in F.

For sync protection, we add δ to every code word before sending and subtract it off after receiving, just as is done in the *Mariner* 1969 high-rate telemetry system. If any sync errors have occurred, the received vector will be distance 6 or more from any code vector. Hence, the error will be detected as a sync error and will cause a request for repeat transmission.

For ease of programming, the vector δ' of F is added to δ, where

$$\delta' = (\beta^{14},\beta^{13},\beta^{12},\beta^{11},\beta^{10},\beta^{9},\beta^{8},\beta^{7},\beta^{6},\beta^{7},\beta^{11},\beta^{9},\beta^{7},0,\beta^{2})$$

then

$$\delta'' = \delta + \delta' = (0,0,\ 0,\ 0,0,0,0,0,0,\beta^{13},\beta^{13},\beta,\beta^{12},\beta,\beta^{8})$$

is added to the code words for transmission. Of course δ'' works as well as δ.

It has been stated that the code symbols were chosen to be elements from the field of 2^4 elements. This means they are represented as binary 4-tuples. But the teletype format allows for sending binary 5-tuples. This gives some latitude on what teletype symbols are used to represent code symbols. One could send one level of the teletype symbol as a constant *zero* or *one*, which seems a waste, or one could choose the 5th level in such a way that all symbols be even (odd) parity. We could then use the appearance of an odd (even) parity symbol as an indicator of error in transmission. Physical considerations dictate that neither choice of parity is practical. If we choose odd parity, operators monitoring traffic at teletype printers are frustrated due to the fact that the two punctuation symbols *carriage return* and *line feed* are both odd parity symbols: if these characters were to appear in the body of a message, the result would be impossible to decipher from the printed version.

There are objections to using even parity exclusively also, although the problem in doing so is more subtle than in the odd parity case. The *blank* symbol (00000) is, of course, an even parity symbol. However, the communications processor at Goddard regards the blank symbol as conveying no information; therefore, it is programmed to delete all blanks from the message. It is just such deletions as these that can cause synchronization errors. Another reason for avoiding the even parity code is the nature of the message sequence *figures–H–letters.* When the communications processor receives this code, it automatically turns itself off. Both *figures* and *H* are even parity symbols. If they were to appear naturally in a code message and an error were to occur to change some other even parity symbol to *letters* creating the triple *figures–H–letters,* the rest of the message would be disregarded.

Therefore, a hybrid system of even and odd parity words should be used and the teletype symbols corresponding to *figures, H, letters, blank, carriage return, line feed,* should be avoided. This allows 26 characters to send, and the 16 elements of the field must be mapped into this set of 26 elements in a way that maximizes the distance between any message symbol used and the six to be avoided. Of course, the inverse mapping of teletype symbols back to 4-tuples will require mapping the unused teletype symbols into some 4-tuple. Whenever possible, an unused teletype symbol was mapped into the 4-tuple, which has the uniquely closest image under the original mapping—since one can expect single teletype errors to be more probable than double errors. The mapping chosen appears in Table 4.

Table 4. Mapping of teletype symbols

Teletype symbol	Code symbols		
	5-tuple	4-tuple	Power of β
Blank	00000	0000	0
E	00001	0000	0
Line feed	00010	0001	β^0
A	00011	0011	β^4
Space	00100	0010	β
S	00101	0010	β
I	00110	0110	β^5
V	00111	0011	β^4
Carriage return	01000	0100	β^2
D	01001	1001	β^{14}
R	01010	1010	β^9
J	01011	0101	β^8
N	01100	1100	β^6
F	01101	1111	β^{12}
C	01110	0111	β^{10}
K	01111	0111	β^{10}
T	10000	0000	0
Z	10001	0001	β^0
L	10010	0010	β
W	10011	1001	β^{14}
H	10100	0100	β^2
Y	10101	0101	β^8
P	10110	1011	β^7
Q	10111	0111	β^{10}
O	11000	1000	β^3
B	11001	1100	β^6
G	11010	1101	β^{13}
Figures	11011	1101	β^{13}
M	11100	1011	β^7
X	11101	1101	β^{13}
V	11110	1110	β^{11}
Letters	11111	1111	β^{12}

B. Decoding

Decoding is much more difficult than encoding and takes more machine time: 1/20 s for decoding vs milliseconds for encoding. However, the decoding is much faster than real time so that total bit rates of up to 1500 bits/s (900 information bits/s) can be accommodated, allowing decoding before the next coded word is fully received.

In Ref. 2, the decoding procedure for Reed–Solomon codes is given, with an example of how the correction procedure works when two errors have been made. In Fig. 2, a flow chart for a computer program is given which corrects double errors in the R–S code. This program runs on an SDS-910 computer and completes its corrections in < 0.04 s. The time for transmission of one teletype word (15 symbols) is 2½ s so that real-time decoding can be easily achieved. A further simplification is added to the decoding, which does not appear in the literature (Ref. 2). In the demonstrated program, instead of checking to see if 3 or more errors have been made after a check has shown that 2 or more errors have been made, a simple test discovered by Solomon (Ref. 3) is applied. One forms the equation $\sigma^2 + \alpha\sigma + \beta = 0$ by assuming two errors have been made and exhaustively searching for the positions of the errors α and β. Then the above equation has a solution for the errors in these positions if, and only if,

$$tr\,\frac{\beta}{\alpha^2} = 0$$

where the trace function *tr* is defined

$$tr\,x = x + x^2 + x^4 + \cdots + x^{2^{k-1}}$$

If

$$tr\,\frac{\beta}{\alpha^2} \neq 0$$

then more than 2 errors have been made and an automatic request for retransmission of the message is initiated. We give an example to indicate the decoding algorithm.

Example

Assume that an error equal to β^7 occurs in the position corresponding to the location of β^3 (i.e., the 4th position of the word) and an error equal to β^{11} occurs in the position corresponding to the location β^{10}. That is, the elements β^7 and β^{11} have been added to the values found in the positions β^3 and β^{10}, respectively. Assume that the synchronization vector δ has been removed by adding it back. If the received vector is a_i, from the definition

$$S_j = \sum_{i=0}^{14} a_i \beta^{ij}$$

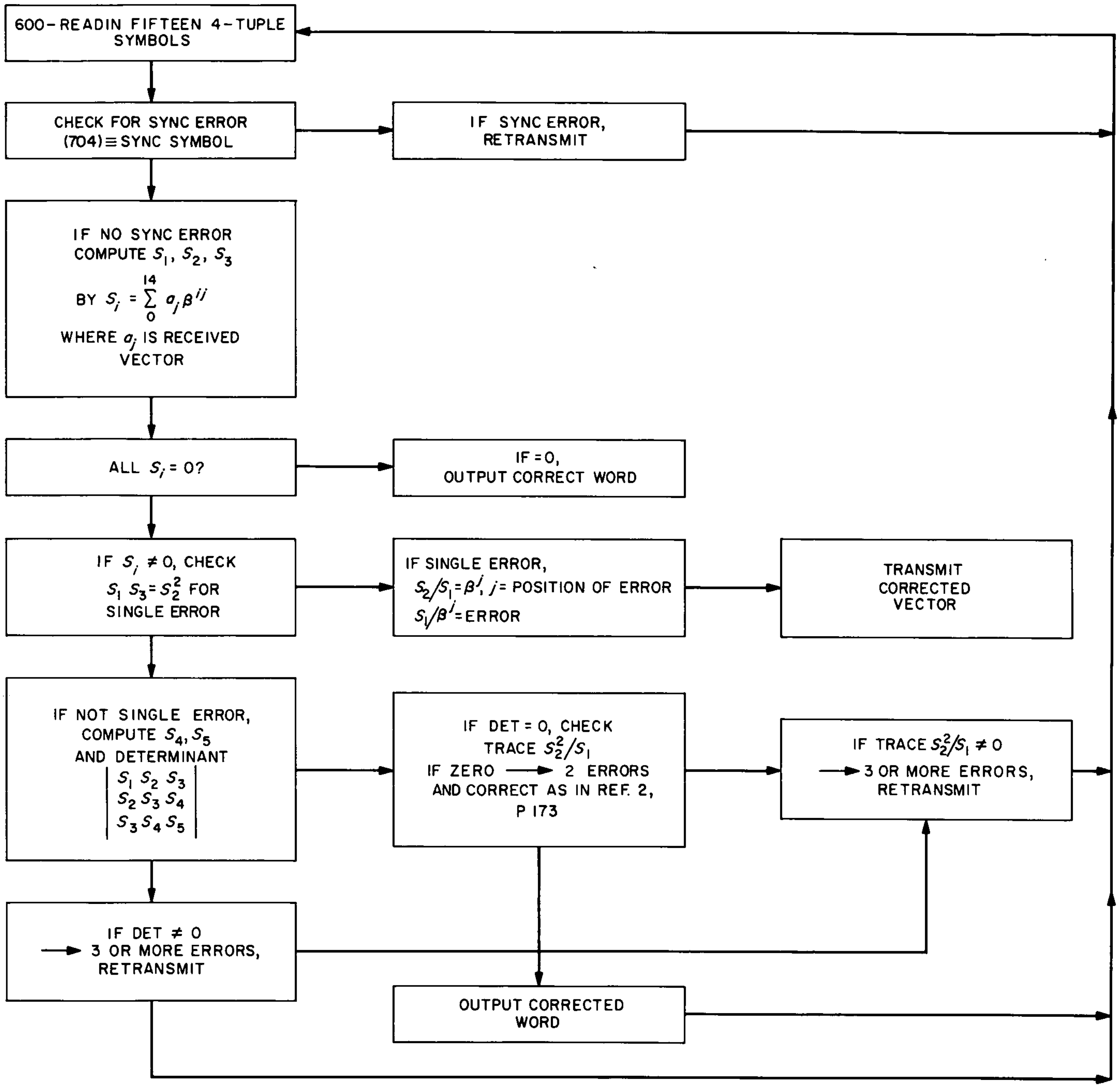

600-READIN FIFTEEN 4-TUPLE SYMBOLS
CHECK FOR SYNC ERROR (704) ≡ SYNC SYMBOL
IF SYNC ERROR, RETRANSMIT
IF NO SYNC ERROR COMPUTE S_1, S_2, S_3 BY $S_i = \sum_0^{14} a_j \beta^{ij}$ WHERE a_j IS RECEIVED VECTOR
ALL $S_i = 0$?
IF = 0, OUTPUT CORRECT WORD
IF $S_i \neq 0$, CHECK $S_1 S_3 = S_2^2$ FOR SINGLE ERROR
IF SINGLE ERROR, $S_2/S_1 = \beta^j$, j = POSITION OF ERROR S_1/β^j = ERROR
TRANSMIT CORRECTED VECTOR
IF NOT SINGLE ERROR, COMPUTE S_4, S_5 AND DETERMINANT $\begin{vmatrix} S_1 & S_2 & S_3 \\ S_2 & S_3 & S_4 \\ S_3 & S_4 & S_5 \end{vmatrix}$
IF DET = 0, CHECK TRACE S_2^2/S_1 IF ZERO → 2 ERRORS AND CORRECT AS IN REF. 2, P 173
IF TRACE $S_2^2/S_1 \neq 0$ → 3 OR MORE ERRORS, RETRANSMIT
IF DET ≠ 0 → 3 OR MORE ERRORS, RETRANSMIT
OUTPUT CORRECTED WORD

Fig. 2. Double error-counting program

one finds

$$\begin{aligned}
S_1 &= \beta^7\beta^3 + \beta^{11}\beta^{10} = \beta^{10} + \beta^{21} = \beta^{10} + \beta^6 = \beta^7 \\
S_2 &= \beta^7\beta^6 + \beta^{11}\beta^{20} = \beta^{13} + \beta^{31} = \beta^{13} + \beta = \beta^{12} \\
S_3 &= \beta^7\beta^9 + \beta^{11}\beta^{30} = \beta^{16} + \beta^{41} = \beta + \beta^{11} = \beta^6 \\
S_4 &= \beta^7\beta^{12} + \beta^{11}\beta^{40} = \beta^{19} + \beta^{51} = \beta^4 + \beta^6 = \beta^{12} \\
S_5 &= \beta^7\beta^{15} + \beta^{11}\beta^{50} = \beta^{22} + \beta^{61} = \beta^7 + \beta = \beta^{14} \\
S_6 &= \beta^7\beta^{18} + \beta^{11}\beta^{60} = \beta^{25} + \beta^{71} = \beta^{10} + \beta^{11} = \beta^{14}
\end{aligned}$$

If all the S_i are $= 0$, one checks for whether or not the received vector satisfies the generating polynomial recursion. If it does, then no errors have occurred, and the received vector was the transmitted vector. If the received vector does not satisfy the recursion, then an uncorrectable error has occurred.

If some $S_i \neq 0$, one tries to determine if a single error has occurred. The occurrence of a single error can be noted because $S_3S_1 = S_1^2$. Then the error position is equal to S_2/S_1 and the error itself is equal to S_1^2/S_2. This, of course, is subject to the check that the corrected vector satisfies the code recursion.

If $S_3S_1 \neq S_2^2$, then 2 or more errors have occurred. This is the case being considered in more detail in this example. From the system of equations

$$\begin{aligned}
S_1\sigma_3 + S_2\sigma_2 + S_3\sigma_1 &= S_4 \\
S_2\sigma_3 + S_3\sigma_2 + S_4\sigma_1 &= S_5 \\
S_3\sigma_3 + S_4\sigma_2 + S_5\sigma_1 &= S_6
\end{aligned}$$

If the determinant

$$D = \begin{vmatrix} S_1 & S_2 & S_3 \\ S_2 & S_3 & S_4 \\ S_3 & S_4 & S_5 \end{vmatrix}$$

is nonzero, then more than 2 errors have occurred. If the determinant D is *zero,* assume that 2 errors have occurred and find σ_1 and σ_2. Now

$$\begin{vmatrix} \beta^7 & \beta^{12} & \beta^6 \\ \beta^{12} & \beta^6 & \beta^{12} \\ \beta^6 & \beta^{12} & \beta^{14} \end{vmatrix} = 0$$

Assume $\sigma_3 = 0$; the system

$$\begin{aligned}
\beta^{12}\sigma_2 + \beta^6\sigma_1 &= B^{12} \\
\beta^6\sigma_2 + B^{12}\sigma_1 &= \beta^{14}
\end{aligned}$$

has the solution $\sigma_2 = \beta^{13}$, $\sigma_1 = \beta^{12}$. The error locations are the solutions of the equation

$$x^2 + \sigma_1 x + \sigma_2 = 0$$

assuming that the equation has exactly 2 solutions, there may not be 2 solutions. We use the test of Solomon (Ref. 3):

$$\sigma_2/\sigma_1^2 = \frac{\beta^{12}}{\beta^{26}} = \frac{\beta^{12}}{\beta^{11}} = \beta$$

and

$$tr\,\frac{\sigma_2}{\sigma_1^2} = tr\,\beta = \beta + \beta^2 + \beta^4 + \beta^8 = 0$$

So there are 2 solutions. They are $x_1 = \beta^3$ and $x_2 = \beta^{10}$.

The errors y_i are computed from the equations

$$\begin{cases} y_1x_1 + y_2x_2 = S_1 \\ y_1x_1^2 + y_2x_2^2 = S_2 \end{cases}, \text{ which is } \begin{cases} y_1\beta^3 + y_2\beta^{10} = \beta^7 \\ y_1\beta^6 + y_2\beta^{20} = \beta^{12} \end{cases}$$

and the solutions are $y_1 = \beta^7$, $y_2 = \beta^{10}$. The corrected word is indeed found to satisfy the code recursion, and has been corrected.

V. Communications Processor for DSN Teletype Correction

To avoid unnecessary delay and reduce errors, the processing of the teletype traffic must be automated. This requires a format restriction for all messages. Such a format restriction is necessary on all traffic through the NASCOM communications processor at Goddard. A total of about 60 symbols is required, including such items as addressee, location, time, sender, and priority. This information is sufficient to pass messages through the communication processor at Goddard to the DSN site specified in the NASCOM format statement. When the message has arrived at the intended destination, it will be the job of the DSN communications processor to do any further routing necessary within the station, after first stripping off the NASCOM header. The NASCOM header in the demonstration system is followed by the DSN header, which will include the legend, "JGUS TTY MSG," followed by three teletype characters which signify the status of the message to follow and what handling it should receive (JGUS is the address given the experimental teletype terminal by NASCOM).

It is envisaged that there would be four different kinds of messages that might be sent to a station in an operational version of the demonstrated system. They are: (1) station-status information (up-dates, ephemeris, etc.) and telemetry data transmission, (2) spacecraft command messages, (3) English text (no coding), (4) *request for retransmission* and general channel status. Messages of type 3 would require no coding and could be directly printed on a teletype receiver. The computer will automatically route coded messages to the decoding program (Fig. 2). Messages of type 1 would be coded also, but since errors are not as serious here, a less powerful, higher rate code can be used (the 15,11 version of the code used here). These messages would be routed to a different decoder. Since only single errors would be corrected by the 15,11 code, a less sophisticated program is necessary, essentially a part of the present program.

Messages of type 4 are what make the whole operation work. A full duplex capability would be available at all sites for sending and receiving, and a separate class of messages will be used to coordinate the two. Messages of this type would generally require no coding, but some human intervention may be desired to make decisions or to monitor the computer operation.

In the demonstration system, after the status of the message has been decided, the body of the message follows. After 4 code words of length 15 have been sent, *carriage return* and *line feed* instructions will be sent so that a general message will consist of blocks of 60 teletype characters/line. The number of lines will vary with the message, but a notification of the length of any message precedes the message. In an operational system, messages of type 3 could be made to fit this format, as well. Messages of types 1, 2, and 3 would be stored before sending, either on tape or in the computer itself.

Figure 3 shows a typical outgoing message from the demonstration system, together with the various stages in the encoding (Fig. 4). Thus, there are 72 information bits, two 9-tuples of field elements, two 15-tuples of code words, two 15-tuples in standard teletype correspondence, and the actual transmitted message including NASCOM

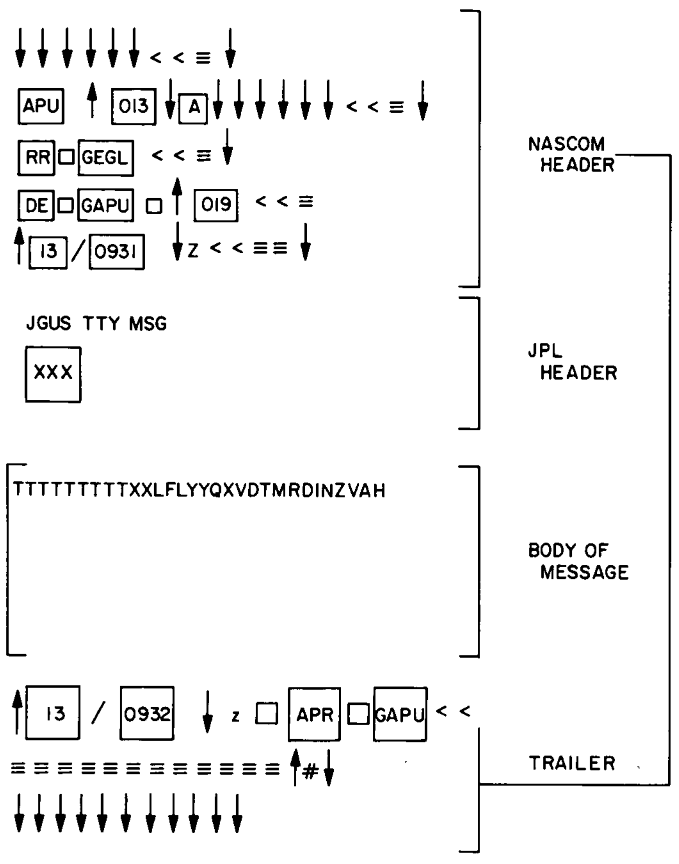


Fig. 3. Typical DSN message [Entries in blocks (e.g., $\boxed{\text{DE}}$) are to be inserted before each transmission]

header and tailer. A symbol, in this case teletype *line feed* (denoting a binary 2 = 00010), appears before the MMM coded message triple (this triple is explained below). This character is put in by the machine and denotes that it sent two coded words.

A certain amount of bookkeeping must be done by the communications processor before the decoding of the message takes place. The communications processor must be programmed to route messages and to strip off header and other extraneous information. The message is received in the communications buffer, which alerts the communications processor via an interrupt. The processor first looks for *spaces* in both the 83rd and 84th characters. If it finds them there, as they should be, the header is stripped and algebraic decoding begins. If either a synchronization or character error has occurred in, or before, these two positions, so that these two *spaces* are not in their correct position, then the processor searches for *carriage return* and *line feed* until it finds them (normally the 92nd and 93rd characters). Decoding then

				WORD 1										WORD 2					
0	0	0	0	0	0	0	0	0		0	0	1	1	1	0	1	1	1	72 INFORMATION BITS (SCAN TOP TO BOTTOM, LEFT TO RIGHT)
0	0	0	0	0	0	0	0	0		1	1	1	1	0	0	0	0	0	
0	0	0	0	0	0	0	0	0		0	1	0	1	0	0	1	1	0	
0	0	0	0	0	0	0	0	0		1	1	1	0	1	0	1	0	1	
0	0	0	0	0	0	0	0	0		β^8	β^{10}	β^{13}	β^{11}	β^{14}	0	β^7	β^9	β^{14}	TWO 9 - TUPLES OF FIELD ELEMENTS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					TWO CODE WORDS
β^8	β^{10}	β^{13}	β^{11}	β^{14}	0	β^7	β^9	β^{14}	β^7	1	β^4	1	1	1					
0	0	0	0	0	0	0	0	0	β^{13}	β^{13}	β	β^{12}	β	β^8					TWO CODES WORDS WITH SYNC WORD ADDED
β^8	β^{10}	β^{13}	β^{11}	β^{14}	0	β^7	β^9	β^{14}	β^5	β^6	1	β^{11}	β^4	β^2					
T	T	T	T	T	T	T	T	T	X	X	L	F	L	Y					TWO TELETYPE WORDS
Y	Q	X	V	D	T	M	R	D	I	N	Z	V	A	H					

Fig. 4. Encoding process

begins. If the processor does not synchronize, it would ask for a repeat of the entire message.

With the assumption that a point is reached at which the decoder detects that it has correct synchronization, algebraic decoding begins. If two or fewer errors have been made, they are corrected and the information symbols are outputted. If the result is not a code word, an uncorrectable error has occurred. Or if the received word fails the three tests for no error, one error, or two errors, an uncorrectable error has also occurred. In either case, a repeat transmission is requested.

After all the code words have been decoded and outputted, the NASCOM tailer is erased and the machine reacts to the *ready for new message* condition.

The coded message triple MMM mentioned above is included in the header of the message to inform the DSN communications processor that the message that follows is to be interpreted as a coded message. If the message were uncoded, a different triple, AAA, would signal the communications processor to handle the message as an English text message requiring no decoding. The message would then be displayed on a teletype monitor. In the description of the program used at the JGUS station for receiving teletype traffic given below, the difference in handling of the two types of messages is explained further.

The AAA and MMM message triples were chosen to represent the two types of message, since the teletype code of A differs in all 5 places from that of M. The three symbols of the triple are considered together, representing 15 binary bits. An error in any symbol will result in errors in some set of the bits representing that symbol. To decode the message triple, the distance between the received triple and MMM is noted. If AAA is received, this difference is 15. If the received message triple is MMM then the difference is 0 and the message is assumed to be coded. In the event of a very noisy channel and the received message triple lies between AAA and MMM, in that the bit pattern differs from MMM in from 0 to 11 places, we assume the message is coded and instruct the decoder to decode the received message. If the triple differs from MMM in 12 to 15 places, we assume an uncoded message was transmitted. The computer is then free to consider other problems, perhaps a separate side calculation, while it is waiting for the end-of-message notification.

Various other triples lying between AAA and MMM can be established as points for other types of message if more than 2 types of message are available to be transmitted.

VI. Performance

Under the assumptions of Section I, the theoretical performance of this error correction and detection system can be computed. We are assuming a channel in which there is a symbol error probability of 0.01 and, independently of this, an insertion-deletion probability of 10^{-4}. We know that the code corrects up to double errors, and detects up to four-fold errors. Furthermore, even if

Table 5. Characteristics of error-correcting system

Machine	Cost (1967), $1,000	Program storage, 1,000 words	Running time	Information rate	Information bits/code word	Word-error probability, worst case	Word repeat probability, worst case
SDS 910 with communications buffer	~100	3.5	Encoding, ms Decoding, 1/20 s/3-s word	0.6	36	10^{-8}	1/200

a five-fold error is made, the probability of not detecting such errors is still only 0.4×10^{-4}.

Using these facts, we conclude that the probability of an undetected error due to symbol errors only, without sync errors included, is about $10^{-8.2}$. The effect of sync errors, together with symbol errors, causing undetected errors, is so small that the exponent is not affected.

The probability of needing a repeat is also an important parameter of the system. The probability of a detected, but uncorrectable, error is about 3.4×10^{-3}, due almost entirely to the occurrence of triple errors. The probability of a sync error in a code word is $15 \times 10^{-4} = 1.5 \times 10^{-3}$, and the probability of detecting the error is so close to 1 that the value of 1.5×10^{-3} is not changed.

All told then, the probability of needing a repeat of a given code word is about 4.9×10^{-3}, or 1 in 200 code words will need to be repeated in the noisiest case. This repeat probability, while not negligibly small, is the price for the extremely low error probability of 10^{-8}.

The properties of the demonstrated error correction and detection system are summarized in Table 5, together with operating parameters. In view of the extreme reliability attainable, the system appears attractive for spacecraft command use. When data compression is used between overseas DSS sites and the SFOF at JPL, such coding will also be necessary on the inbound telemetry links, since because of the data compression, the incoming data will have less redundancy.

VII. Operation of the DSN Communications Processor

The prototype communications processor was programmed to monitor communications between stations. The stations of Johannesburg (LJOB) and the Information Processing Laboratory (JGUS) were chosen since these two stations were outfitted with SDS-910 computers and the necessary associated buffers to make computer interaction feasible.

Sample messages were sent from LJOB to JGUS and decoded at JGUS, when received. Time was at a premium at the LJOB site because the *Mariner V* mission was going on simultaneously. The test involved sending a prescribed 10-min message a total of 8 times in a 2-h period. A 5-min waiting period was observed between transmissions of the message to allow time for operators at LJOB to type necessary header information into the computer and to allow the computer at JGUS to branch out of the decoding mode to its *acquiescent* state where it waits for another message. The computer (communications processor) can also receive other messages while it is in its acquiescent state, but it then assumes that it is receiving a coded message unless it has been explicitly told that an uncoded message is being received (the triple AAA). The decoding of an uncoded message presents no difficulty, since the decoding can be disregarded.

The experiment was conducted on a one-way basis, only, because LJOB was on mission status and could allow only small amounts of time to the experiment. In particular, LJOB could only allow 2 h between 08:00 and 10:00 GMT. Corresponding starting time was 1:00 a.m. Pasadena time, at which time no monitor was on duty at JGUS to handle arriving messages; the ability of the communications processor at JGUS to handle communications without an operator present was demonstrated.

The test of the decoder was not conclusive, however. Since the LJOB station was on mission status, all lines were tuned to maximum reliability. In the first 8 tests of 8 messages each—a total of 195,840 characters—there were 3 single errors, one double error, and five synchronization errors. The additive errors were corrected, and the synchronization errors were noted; but the error probability of 5/196,840 is much smaller than the worst case 1% error probability previously observed on messages from LJOB. The error probabilities have dropped for two reasons: (1) the equipment had been tuned up for *Mariner*, and (2) the transmission at LJOB and JGUS did not go through a paper punch. The paper punch was thought to be a large producer of error in the original experiment. After the *Mariner* mission, the experiment can be repeated. The channel may have degraded sufficiently by that time that some more interesting error pattern may occur.

Following are descriptions of the programs used at the LJOB and JGUS stations.

LJOB to JGUS, Send Only

Mode 0: Initialization

Setting up lines	TTY lines — LJOB to JGUS plugboard connection from SDS 910 to communications buffer plug in lines to monitor
Computer setup	load tapes — standard fill
Program setup	input header information

The typing of the *date-time group* and the transmission of the first character of the header takes the computer from mode 0 to mode 1.

Mode 1: Send

Characters are sent out of memory to the TTY lines under interrupts. While waiting for the interrupts to come *true,* the computer sits in an *idle* configuration counting the number of characters transmitted.

When the last character is sent (header + message + tailer), the interrupt is disabled, and the computer halts. A manual transfer to mode 0 of *send only* or to mode 0 of some other program is then enacted.

JGUS, Receive Only

Mode 0. Initialization

Set up lines	TTY lines from SFOF plugboard connection from communications buffer to SDS-910 plug in lines to monitor
Computer setup	load tapes
Program setup	branch to idle—wait for transmission configuration

Characters are counted under interrupts as they arrive. When 82 characters have been received, automatically branch to mode 1.

Mode 1. Strip header

If no errors have occurred in the header, the computer determines from information contained in the header the information about the message to follow. If message is uncoded, branch to mode 2*; if message is coded, branch to mode 2.

Mode 2. Count 15 characters of message

In this mode, 15 characters at a time are counted off; when 15 arrive, branch to mode 3.

Mode 3. Decode

Interrupts are operating, but the 15-character word can be decoded in $< 1/20$ s. Since TTY character rate is 5/s, no character interrupt will pull the program out of the decoder—(no problem even if one is pulled back to mode 2 by interrupts). Normal operation will be to branch to mode 4 for output when decoding is complete.

Mode 4. Output

Type out status of decoded word. Normal operation will cause a pullback by interrupts to mode 2 to receive characters. If all output is complete, branch to mode 2. If all characters have been received and decoded, branch to the waiting configuration which precedes mode 1.

Mode 2*. Wait in this mode until all traffic stops. The computer will automatically branch back to the waiting mode by counting machine cycles: 7×10^6 cycles are counted. At SDS machine cycle speed of 8 μs, there will be 56 s of *dead time* before branching back occurs.

Appendix

Sample Analyses of Data Tapes

Case 1

Bit errors		Pattern errors	
Channel 1	407	0	2331
2	429	1	165
3	429	2	286
4	459	3	272
5	426	4	127
Total	2150	5	19

This is a sample output from the first experiment. Approximately ¾ of the way through the experiment, there was a synchronization error causing approximately half the bits from that point on to be in error. The number of errors in each channel is approximately constant, because of the pseudorandom character of the construction. The pattern-error count indicates the number of times there were exactly 0, 1, $\cdots$, 5 errors in a character. Of the 2331 characters that contained no errors, almost all of them occur before the sync error. In fact, the number that occurs after the sync error is about the same as the number of characters containing 5 errors. Since the probability of a bit error after the sync error is ½, the greatest number of pattern errors should be about half of the 5 bits in each character, and one observes this in the sample case.

Case 2

Bit errors		Pattern errors		$0\to1$ Bit errors		$0\to1$ Pattern errors		$1\to0$ Bit errors		$1\to0$ Pattern errors	
Channel 1	0	p=0	3177	Channel 1	0	p=0	3177	Channel 1	0	p=0	3184
2	2	1	7	2	2	1	7	2	0	1	0
3	0	2	0	3	0	2	0	3	0	2	0
4	0	3	0	4	0	3	0	4	0	3	0
5	5	4	0	5	5	4	0	5	0	4	0
Total	7	5	0	Total	7	5	0	Total	0	5	0

This shows results from typical tape from the second experiment. There was a total of 7 errors in the tape of 15,920 bits for a rate of about 5×10^{-4}. All the errors were of $0\to1$ type, and each occurred in a different character.

Case 3

Bit errors		Pattern errors		$0\to1$ Bit errors		$0\to1$ Pattern errors		$1\to0$ Bit errors		$1\to0$ Pattern errors	
Channel 1	1574	p=0	233	Channel 1	795	p=0	767	Channel 1	779	p=0	836
2	1347	1	1	2	682	1	1289	2	665	1	1259
3	1579	2	2060	3	797	2	855	3	782	2	787
4	1562	3	5	4	789	3	214	4	773	3	242
5	1614	4	885	5	814	4	59	5	800	4	60
Total	7676	5	0	Total	3877	5	0	Total	3799	5	0

This case is similar to case 1. A sync error occurs near the beginning of the tape. The first column of pattern errors is very interesting. Since all characters have odd parity, comparison of two of them shows they have an even number of differences. The number of odd errors is presumably close to the correct number of single errors occurring in the tape. Note the number of double errors is about twice the number of quadruple errors, as can be expected. Little can be concluded from the remaining columns.

Case 4

				0→1		0→1		1→0		1→0	
Bit errors		Pattern errors		Bit errors		Pattern errors		Bit errors		Pattern errors	
Channel 1	0	p=0	3176	Channel 1	0	p=0	3176	Channel 1	0	p=0	3183
2	3	1	6	2	3	1	6	2	0	1	1
3	1	2	1	3	0	2	2	3	1	2	0
4	1	3	1	4	1	3	0	4	0	3	0
5	6	4	0	5	6	4	0	5	0	4	0
Total	11	5	0	Total	10	5	0	Total	1	5	0

In this case, the three columns of pattern errors are most important. We see from the first column there were 6 single, 1 double, and 1 triple error patterns. From the second column we see that there were 6 single errors of *zero→one* and 2 double errors of *zero→one*. The third column indicates a single error of *one→zero*. But the three columns together show that the *one→zero* errors must have occurred at the same time as one of the double *zero→one* errors. This error may have been one that occurred during the ARQ portion of the channel.

References

1. Solomon, G., *Introduction to Algebraic Coding Theory,* McGraw-Hill Book Co., Inc., New York (in preparation).
2. Peterson, W. W., *Error Correcting Codes,* John Wiley & Sons, Inc., New York, 1961.
3. Solomon, G., "Solution of Algebraic Equations Over Fields of Characteristic 2," *Information and Control* (to be published).