NASA Technical Paper 1286

A Decoding Procedure for the Reed-Solomon Codes

Raymond S. Lim

AUGUST 1978

NASA

NASA Technical Paper 1286

A Decoding Procedure for the Reed-Solomon Codes

Raymond S. Lim
Ames Research Center
Moffett Field, California

National Aeronautics
and Space Administration

**Scientific and Technical
Information Office**

1978

SYMBOLS

d	minimum (Hamming) distance between code words in a code
E(x)	error polynomial
e	order of an element in $GF(2^m)$
e_j	error value at location X_j of received codeword
G	generator matrix of a code
$GF(2^m)$	Galois field of 2^m elements
g(x)	generator polynomial of a cyclic code
H	parity-check matrix of a code
k	number of information symbols in a code
M(x)	message polynomial
m	a positive integer
m(x)	minimum polynomial
n	code length (total number of symbols in a code)
(n,k)	linear block code with parameters n and k
p	number of errors in received codeword
p(x)	primitive polynomial
q	a positive integer defines the numbers of element in the Galois field
R(x)	received polynomial
r(x)	remainder polynomial
S	syndrome of a parity check
S_i	a syndrome component calculated by root substitution method
S_{i*}	a syndrome component calculated by R(x)/g(x)
t	error correcting capability of a code
V(x)	transmitted polynomial

X_j	error location at jth position of received codeword
α	primitive element of $GF(2^m)$
β	error location number
$\sigma(x)$	error location polynomial

A DECODING PROCEDURE FOR THE REED-SOLOMON CODES

Raymond S. Lim

Ames Research Center

SUMMARY

This paper describes a decoding procedure for the (n,k) t-error-correcting Reed-Solomon (RS) code, and an implementation of the (31,15) RS code for the I4-TENEX central system. This code can be used for error correction in large archival memory systems. The principal features of the decoder are (1) a Galois field arithmetic unit implemented by microprogramming a microprocessor, and (2) syndrome calculation by using the $g(x)$ encoding shift register. Complete decoding of the (31,15) code is expected to take less than 500 μsec. The syndrome calculation is performed by hardware using the encoding shift register and a modified Chien search. The error location polynomial is computed by using Lin's table, which is an interpretation of Berlekamp's iterative algorithm. The error location numbers are calculated by using the Chien search. Finally, the error values are computed by using Forney's method.

PROLOGUE

Historically, in the development of algebraic coding theory, the Reed-Solomon (RS) codes were recognized as the most powerful and elegant, and also the most complicated, block codes to decode. These codes were first described by I. S. Reed and G. Solomon in 1960. A systematic decoding algorithm was not discovered until 1968 by E. Berlekamp. Because of their complexity, the RS codes are not widely used except when no other codes are suitable.

In 1968, a (63,53) RS code with code symbols from $GF(2^6)$ was used to salvage the IBM Photo-Digital Storage System. In 1975, a (4095,4001) RS code with code symbols from $GF(2^{12})$ was used to make the magnetic tape archival data storage system viable at the Bureau of the Census, Department of Commerce. In 1975-1976, a (31,15) RS code with code symbols from $GF(2^5)$ was designed by E. Berlekamp for use in a classified defense communication system. Before proceeding with a description of decoding the RS codes, it is felt that the following poems selected to amplify the coding spirit are appropriate.

In Galois Fields*

In Galois fields, full of flowers
primitive elements dance for hours
climbing sequentially through the trees
and shouting occasional parities

The syndromes like ghosts in the misty damp
feed the smoldering fires of the Berlekamp
and high-flying exponents sometimes are downed
on the jagged peaks of the Gilbert bound.

Message and Clarity†

A message with content and clarity
Has gotten to be quite a rarity.
To combat the terror of serious error,
Use bits of appropriate parity.

INTRODUCTION

With present technology, very large memory systems ($\geq 10^{12}$ bits) designed for the archival storage of digital data are critically dependent on electronic error correction systems (EECS) for ensuring system viability and integrity (refs. 1, 2). In the IBM 3850 Mass Storage System, the EECS used is an Extended Group Coded Recording capable of correcting up to 32 8-bit bytes of data in a 208-byte data block. In the CDC 38500 Mass Memory System, the EECS used is a modified Group Coded Recording similar to that used in the IBM 2400 Series magnetic type systems.

The use of magnetic tape systems for archival storage of digital data depended even more critically on EECS to make them viable. The EECS devised by Brown and Sellers, which was used in the IBM 2400 series magnetic tape system, is not adequate for long-term archival storage of data (refs. 3-5).

At the Institute for Advanced Computation (IAC), archival storage systems such as the UNICON 690 (or UNICON 190), magnetic tape systems, and other mass memory systems are no exceptions. The viability of these systems depends critically on EECS. For the Institute, instead of designing a different EECS tailored to each particular archival system, it is advantageous to design just one EECS powerful enough to serve all systems within the Institute.

---

*By S. B. Weinstein of Bell Telephone Labs., IEEE, Trans. on Inf. Theory, March 1971, p. 220.

†From Error Correcting Codes, H. B. Mann, Ed., Wiley, N.Y., 1968.

This paper describes the (n,k) t-error-correcting Reed-Solomon (RS) codes and a decoding procedure suitable for implementation with the present technology. In particular, a (31,15) RS code is chosen as the EECS for the IAC I4-TENEX central system. This code is not a binary code, but a q-ary code with code symbols from $GF(2^5)$. It is believed that this code is powerful enough to meet all anticipated IAC requirements. Because of the PDP-10, the PDP-11, and the ILLIAC IV computers comprising the I4-TENEX system, this code is planned to have two modes of operation: 36-bit mode and 16/8-bit mode. The decoding of this code will be implemented by hardware and firmware, and consists of four steps: (1) the syndrome calculation is performed by hardware using the encoding shift register and a modified Chien search; (2) the error location polynomial computation is performed by firmware by microprogramming a 2900 series microprocessor to implement the Berlekamp iterative algorithm; (3) the error location numbers calculation is performed by hardware using the Chien search method; and (4) the error values computation is performed by firmware using a method suggested by Forney. Finally, this (31,15) RS code EECS is interfaced to the I4-TENEX system by means of a standard IAC 1011-Interface, like the Q1011 (ref. 6). With this interface, this EECS is just another processor in the I4-TENEX central system.

The author wishes to thank Professor Shu Lin of the University of Hawaii for his initial consultation and for reading this paper. He also thanks his colleagues D. K. Stevenson, G. F. Feierbach, and P. Hiss for reading and commenting on the work reported herein.

REED-SOLOMON CODES

The Reed-Solomon (RS) codes (refs. 7-9) are the most powerful of the known classes of block codes capable of correcting random errors and multiple-burst errors. The RS codes are nonbinary codes with code symbols from a Galois field of q elements $GF(q)$. From coding theory, if p is a prime number and q is any power of p, there are codes with code symbols from a q-symbol alphabet. These codes are called q-ary Bose-Chaudhuri-Hocquenghem (BCH) codes.

For engineering applications at the present, only binary codes derived from RS codes are of interest. For this reason, $GF(q)$ will be restricted to $GF(2^m)$, where m is a positive integer. The field $GF(2^m)$ is formed by a primitive polynomial of degree m with α as the primitive element of the field. In the algebra of a Galois field, α is also called the nth root of unity in $GF(2^m)$ since $\alpha^n = 1$ for $n = 2^m - 1$. With $q = 2^m$, the code symbols of an RS code are α^i, $i = \infty, 0, 1, 2, \ldots, 2^m - 2$, which are the 2^m distinct elements of $GF(2^m)$. The notation $\alpha^\infty = 0$ is used here.

Let m_0, d, s, and t be any positive integers, and α be an element of $GF(q^s)$. There exists a q-ary BCH code of length $n = q^s - 1$ symbols that corrects any combination of t or fewer errors and requires no more than $2st$ parity-check symbols. Let $g(X)$ be a polynomial of lowest degree over $GF(q)$ and select $q = 2^m$. The code generated by $g(X)$ is a cyclic BCH code and has

$$\alpha^{m_0}, \alpha^{m_0+1}, \ldots, \alpha^{m_0+d-2} \tag{1}$$

as roots. The special q-ary BCH code, for which $s = m_0 = 1$ and $d = 2t + 1$, is called the RS code. The roots of the RS code are

$$\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2t} \tag{2}$$

Since the minimum polynomial with root of α^j is simply $(x + \alpha^j)$, the generator polynomial $g(X)$ of a t-error-correcting RS code of length $2^m - 1$ is

$$g(X) = (X + \alpha)(X + \alpha^2) \ldots (X + \alpha^{2t}) \tag{3}$$

The codeword polynomials generated by $g(X)$ consist of the multiples of $g(X)$ modulo $X^n + 1$, and have $\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2t}$ as roots. Since $g(X)$ has degree $2t$, and α is a primitive nth root of unity in $GF(2^m)$, the RS code generated by $g(X)$ is a t-error-correcting cyclic code with the following parameters:

Code length (symbols): $n = 2^m - 1$

Number of parity check symbols: $n - k = 2t$

Minimum distance: $d = 2t + 1$

Number of information symbols: $k = 2^m - 1 - 2t$

Since a symbol in $GF(2^m)$ can be expressed as an m-tuple over $GF(2)$, the parameters of an RS code over $GF(2)$ are:

$$n = m(2^m - 1) \quad \text{bits}$$

$$n-k = 2mt \quad \text{bits}$$

$$d = 2t + 1$$

$$k = m(2^m - 1 - 2t) \quad \text{bits}$$

In coding theory, if $g(X)$ is a polynomial of degree $n-k$ and is a factor of $X^n + 1$, then $g(X)$ generates an (n,k) cyclic code (ref. 7, theorem 4.3; ref. 8, theorem 8.1). One way to show that an RS code generated by $g(X)$ is cyclic is to describe the code in terms of its roots of $g(X)$ in $GF(2^m)$. The order e of a field element α^i is the least positive integer for which $(\alpha^i)^e = 1$. Since $(\alpha^i)^n = 1$, e must be a factor of $n = 2^m - 1$. If e divides n, then $(X^e + 1)$ divides $(X^n + 1)$. Furthermore, an element α^i of order e must be a root of $(X^e + 1)$, then $(X + \alpha^i)$ divides $(X^e + 1)$, and hence it divides $(X^n + 1)$. Therefore, $(X^n + 1)$ has as roots all the $n = 2^m - 1$ nonzero elements of $GF(2^m)$. Since $g(X)$ has $\alpha^1, \alpha^2, \alpha^3, \ldots, \alpha^{2t}$ as roots, $g(X)$ is a factor of $(X^n + 1)$, and hence $g(X)$ generates a cyclic code.

ERROR CORRECTING CAPABILITY

The RS codes over $GF(2^m)$ are very effective for correcting random and burst errors. Since each code symbol is an m-tuple (or m-bit symbol) over $GF(2)$, a t-error-correcting RS code is capable of correcting any error pattern that affects t or fewer m-bit symbols. For example, since a burst of length $3m+1$ cannot affect more than four m-bit symbols, a four-symbols correcting code can correct any single burst of length $3m+1$ or less. It can also simultaneously correct any combination of two bursts of length $m+1$ or less because each such burst can affect no more than two symbols. At the same time, it can correct any combination of four or less random errors. In general, the RS code with error correcting capability t can be used to correct any of the following errors:

1. All single bursts of length b_1, no matter where it starts, if $b_1 \leq m(t-1) + 1$
2. Two bursts of length no longer than b_2 each, no matter where each burst starts, if $b_2 \leq m([t/2] - 1) + 1$, or any p bursts of length no longer than b_p each, no matter where each burst starts, if $b_p \leq m([t/p] - 1) + 1$

From the above discussion, it follows that the RS code can be used to correct random errors, single-burst errors, or multiple-burst errors.

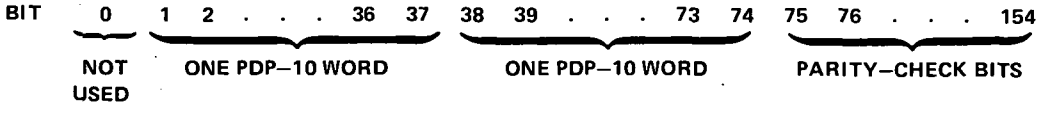
CODE SELECTION

The RS codes offer the designer a wide range of code parameters. In coding theory, a block code with parameters n and k is denoted as (n,k). In table 1, a list of RS codes is tabulated for m equal to 4, 5, and 6 with t ranging from 2 to 10. For the IAC I4-TENEX system consisting of computers

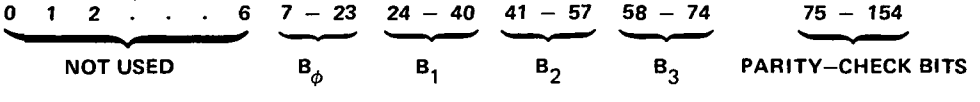
TABLE 1. - LIST OF REED-SOLOMON CODES

		$GF(2^m)$		GF(2)	
m	t	(n,k)	r = n-k = 2t	$n = m(2^m - 1)$	k = n-r
4	2	(15,11)	4	60	44
	3	(15,9)	6	60	36
	4	(15,7)	8	60	28
	5	(15,5)	10	60	20
	6	(15,3)	12	60	12
5	2	(31,27)	4	155	135
	3	(31,25)	6	155	125
	4	(31,23)	8	155	115
	5	(31,21)	10	155	105
	6	(31,19)	12	155	95
	7	(31,17)	14	155	85
	8	(31,15)	16	155	75
	9	(31,13)	18	155	65
6	4	(63,55)	8	378	330
	5	(63,53)	10	378	318
	6	(63,51)	12	378	306
	7	(63,49)	14	378	284
	8	(63,47)	16	378	282
	9	(63,45)	18	378	270
	10	(63,43)	20	378	258

with word lengths of 16 bits (PDP-11s), 36 bits (PDP-10s), and 32/64 bits (ILLIAC IV), the best choice for fitting these word lengths is the (31,15) code. The formats for the 36-bit and the 16/8-bit are shown in figure 1.



(a) Fitting of two PDP-10 words into 75 bits. Bit-Ø is not used (always equal to zero).



(b) Fitting of four PDP-11 words into 75 bits. B_i (i = Ø, 1, 2, and 3) is a 16-bit word plus parity.

Figure 1. - Data formats of the (31,15) RS code for 36-bit and 16/8-bit modes. Unused leading bits are always zero.

ENCODING

There are two methods for encoding linear cycle codes: the serial shift register method and the parallel matrix method. Let $M(X)$ be a message polynomial with k symbols encoded into a code polynomial (codeword) $V(X)$ with n symbols. In the serial shift register method, encoding in systematic form is accomplished by dividing $X^{n-k}M(X)$ by $g(X)$ and appending the remainder $r(X)$ to $X^{n-k}M(X)$. That is

$$V(X) = r(X) + X^{n-k}M(X) = q(X)\,g(X) \tag{4}$$

where $q(X)$ is the quotient. This indicates that $[r(X) + X^{n-k}M(X)]$ is a multiple of $g(X)$ and, therefore, is a code polynomial generated by $g(X)$. The codeword generated is

$$\underbrace{(r_0 r_1 r_2 \cdots r_{n-k-1}}_{\text{parity check bits}}\ \underbrace{m_0 m_1 \cdots m_{k-1})}_{\text{message bits}}$$

and the most significant symbol of the message, m_{k-1}, is sent first.

In the parallel matrix method, the generator matrix G has the form

$$G = [P_{k \times (n-k)},\ I_{k \times k}] \tag{5}$$

where P is a $k \times (n-k)$ matrix generated by retaining the remainder of

$$\frac{X^{n-k+i}}{g(X)},\ i = 0, 1, 2, \ldots, k-1 \tag{6}$$

and I is a $k \times k$ identity matrix. The encoding of the message vector M to a code vector V is

$$V = MG \tag{7}$$

The (31,15) RS code has $n = 31$, $k = 15$, $m = 5$, and $t = 8$. The primitive polynomial $p(X) = X^5 + X^2 + 1$ can be used to generate the 31 nonzero elements of $GF(2^5)$, as shown in table 2. The generator polynomial $g(X)$ from equation (3) is

$$g(X) = (X+\alpha)(X+\alpha^2)\ldots(X+\alpha^{15})(X+\alpha^{16}) \tag{8}$$

Multiplying out the terms of $g(X)$, the general form of $g(X)$ is

$$g(X) = X^{16} + \sum_{i=0}^{15} \alpha_i X^i = X^{16} + \alpha_{15} X^{15} + \alpha_{14} X^{14} + \ldots + \alpha_1 X + \alpha_0 \tag{9}$$

The evaluation of α_i in equation (9) is straightforward but extremely tedious. In order to avoid errors, a computer program should be used in conjunction with table 2 for such evaluation.

The implementation of equation (9) for encoding using the parallel matrix method is a matrix G as shown in figure 2. Each $\alpha_{i,j}$ in the matrix G is a field element in $GF(2^5)$. The encoding of $M(X)$ into $V(X)$ is

TABLE 2. - GALOIS FIELD OF 2^5.

	α^4	α^3	α^2	α^1	α^0		α^4	α^3	α^2	α^1	α^0
$0 =$	0	0	0	0	0	$\alpha^{15} =$	1	1	1	1	1
$1 =$	0	0	0	0	1	$\alpha^{16} =$	1	1	0	1	1
$\alpha =$	0	0	0	1	0	$\alpha^{17} =$	1	0	0	1	1
$\alpha^2 =$	0	0	1	0	0	$\alpha^{18} =$	0	0	0	1	1
$\alpha^3 =$	0	1	0	0	0	$\alpha^{19} =$	0	0	1	1	0
$\alpha^4 =$	1	0	0	0	0	$\alpha^{20} =$	0	1	1	0	0
$\alpha^5 =$	0	0	1	0	1	$\alpha^{21} =$	1	1	0	0	0
$\alpha^6 =$	0	1	0	1	0	$\alpha^{22} =$	1	0	1	0	1
$\alpha^7 =$	1	0	1	0	0	$\alpha^{23} =$	0	1	1	1	1
$\alpha^8 =$	0	1	1	0	1	$\alpha^{24} =$	1	1	1	1	0
$\alpha^9 =$	1	1	0	1	0	$\alpha^{25} =$	1	1	0	0	1
$\alpha^{10} =$	1	0	0	0	1	$\alpha^{26} =$	1	0	1	1	1
$\alpha^{11} =$	0	0	1	1	1	$\alpha^{27} =$	0	1	0	1	1
$\alpha^{12} =$	0	1	1	1	0	$\alpha^{28} =$	1	0	1	1	0
$\alpha^{13} =$	1	1	1	0	0	$\alpha^{29} =$	0	1	0	0	1
$\alpha^{14} =$	1	1	1	0	1	$\alpha^{30} =$	1	0	0	1	0

Note: Elements generated by $p(\alpha) = \alpha^5 + \alpha^2 + 1$.
For example, $\alpha^7 = (10100)$ means $\alpha^7 = \alpha^4 + \alpha^2$.

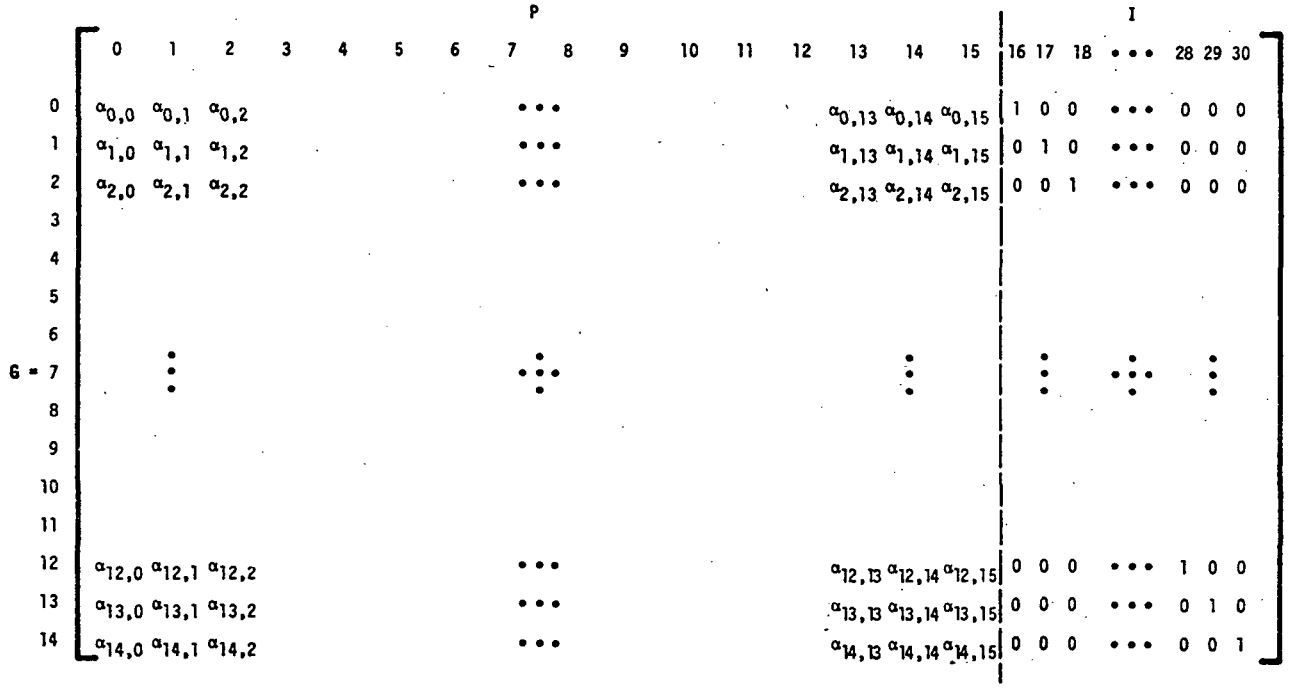


Figure 2. - Generator matrix G of (15,13) Reed-Solomon codes; $\alpha_{i,j}$ is a field element from $GF(2^5)$.

$$V(X) = (m_0 m_1 m_2 \cdots m_{k-2}\, m_{k-1})G \qquad (10)$$

where the message symbol m_{k-1} is the most significant digit. From figure 2 and equation (10), the parallel matrix encoding method requires about $15 \times 16 = 240$ Galois field multipliers. Because of the logic complexity, this method is generally not used except for very high-speed applications, or when G is very simple. For example, consider a 32-bit memory system built by using n words by 4-bit integrated-circuit, random-access memory (IC RAM) chips. In such a memory system, a single chip failure will result in a 4-bit error in the 32-bit word. A $t = 1$ (15,13) RS code with code symbols from $GF(2^4)$ can be used for error correction in such a memory system. The G matrix of this code requires about $13 \times 2 = 26$ Galois field multipliers. Using current technology, these multipliers can be implemented by table lookup using read-only memory (ROM) chips.

There are two shift register methods for encoding linear cyclic codes. One method uses a (n-k)-stage shift register, and the other uses a k-stage shift register (ref. 7). In practice, the (n-k)-stage shift register is most commonly used unless n-k is much greater than k. For encoding the (31,15) RS code, a (n-k) = 16 stages shift register (fig. 3) can be used to implement equation (4). The feedback multipliers $\alpha_0, \alpha_1, \ldots, \alpha_{15}$ are field elements of $GF(2^m)$. Each R_i register stage is a 5-tuple shift register. The operation of the encoder is as follows. With S1 set for feedback and S2 set to position 2, k information symbols are shifted into the encoder and simultaneously sent to the channel. Then S1 is turned to disable the feedback and S2 is turned to position 1; the 16 parity-check symbols stored in the encoder now are shifted out to the channel, clearing the shift registers.

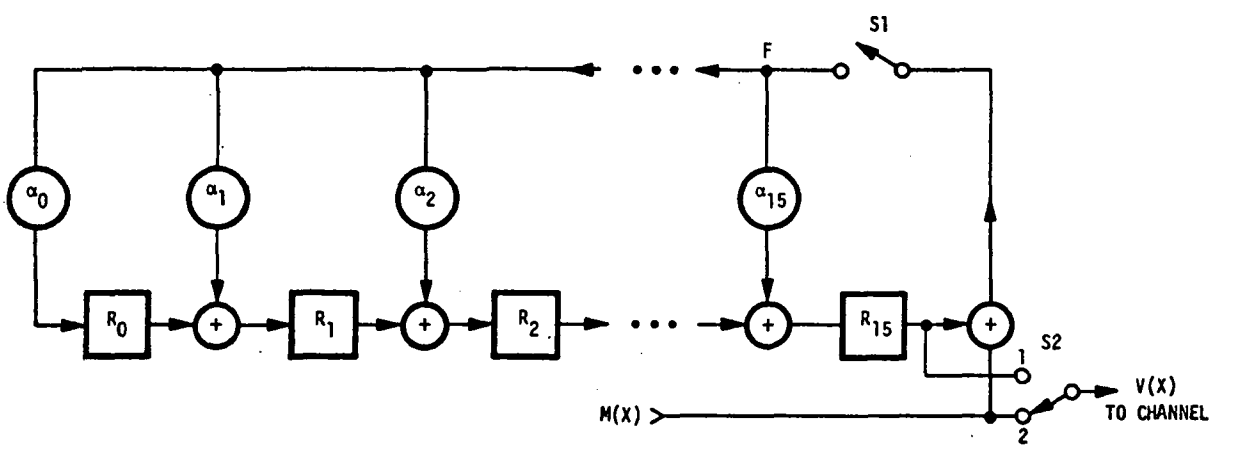


Figure 3. - Encoder for (31,15) code; α_i is a field element from $GF(2^5)$ and R_i is a 5-tuple shift register stage.

DECODING

In this section, an error correction procedure is described for the (n,k) t-error-correcting RS codes, and a design implementation is shown for the (31,15) $t = 8$ RS code. As described earlier in encoding, RS codes are non-binary codes. Therefore, the decoding procedure involves finding not only the error locations, but the error values as well. A t-error-correcting RS code generated by $g(X)$ is a cyclic code, and the codewords consist of all multiples of $g(X)$. The degree of $g(X)$ is 2t and its roots are $\alpha, \alpha^2, \ldots, \alpha^{2t}$. Since every codeword is some multiple of $g(X)$, $\alpha, \alpha^2, \ldots, \alpha^{2t}$ are also roots of every codeword.

Let $V(X)$ be the transmitted codeword, $E(X)$ be the channel noise error pattern, and $R(X)$ be the received codeword represented as follows:

$$V(X) = v_0 + v_1 x + v_2 x^2 + \ldots + v_{n-1} x^{n-1}$$

$$E(X) = e_0 + e_1 x + e_2 x^2 + \ldots + e_{n-1} x^{n-1}$$

$$R(X) = r_0 + r_1 x + r_2 x^2 + \ldots + r_{n-1} x^{n-1}$$

where v_i, e_i, and r_i, $i = 0, 1, 2, \ldots, n-1$, are elements of $GF(2^m)$. At the decoder,

$$R(X) = V(X) + E(X) \tag{11}$$

The error pattern $E(X)$ can be described by a list of values and locations of its nonzero components. For the decoding procedure to be described, the error location will be given in terms of an error-location number, which is simply α^j for the (n-j)th symbol. Let X_j be the error location number and e_j be the error value. Then for each nonzero component of $E(X)$, a pair of field elements (x_j, e_j) is required to describe that error. If $E(X)$ has p errors, then p pairs (x_j, e_j) are required to describe the errors. Any decoding procedure is a procedure for locating these p pairs of (x_j, e_j) if $p \leq t$.

Assume that $E(X)$ is an error pattern of p errors at locations $j_1, j_2, \ldots, j_p$. Then

$$E(X) = e_{j_1} x^{j_1} + e_{j_2} x^{j_2} + \ldots + e_{j_p} x^{j_p} \tag{12}$$

where $p \leq t$ and $0 \leq j_1 < j_2 < \ldots < j_p \leq n-1$. The first step in decoding is to check whether $V(X)$ is a codeword by calculating the syndrome. If the syndrome is zero, then either $V(X)$ actually has no errors or $V(X)$ has an undetectable error. In either case, $V(X)$ is accepted as no error. A nonzero syndrome indicates that an error has been detected; the error may or may not be correctable. For the RS codes, the syndrome is defined as a vector S with 2t components as follows (refs. 7, 8):

$$S_i = R(\alpha^i) = r_0 + r_1 \alpha^i + r_2 (\alpha^i)^2 + \ldots + r_{n-1} (\alpha^i)^{n-1} \tag{13}$$

for $i = 1, 2, 3, \ldots, 2t$. Combining equations (11) and (13) gives the result

$$S_i = V(\alpha^i) + E(\alpha^i) \tag{14}$$

Since $V(\alpha^i) = 0$,

$$S_i = E(\alpha^i) = \sum_{\ell=1}^{p} e_{j\ell} X_{j\ell}^i \tag{15}$$

Combining equations (12) and (15) gives the result

$$S_i = e_{j_1}(\alpha^{j_1})^i + e_{j_2}(\alpha^{j_2})^i + \ldots + e_{j_p}(\alpha^{j_p})^i \tag{16}$$

for $i = 1, 2, \ldots, 2t$. Expanding equation (16) gives the result

$$\left.\begin{aligned} S_1 &= e_{j_1}\alpha^{j_1} + e_{j_2}\alpha^{j_2} + \ldots + e_{j_p}\alpha^{j_p} \\ S_2 &= e_{j_1}(\alpha^{j_1})^2 + e_{j_2}(\alpha^{j_2})^2 + \ldots + e_{j_p}(\alpha^{j_p})^2 \\ &\vdots \\ S_{2t} &= e_{j_1}(\alpha^{j_1})^{2t} + e_{j_2}(\alpha^{j_2})^{2t} + \ldots + e_{j_p}(\alpha^{j_p})^{2t} \end{aligned}\right\} \tag{17}$$

Equation (17) is a set of $2t$ nonlinear equations which relates the $2t$ known quantities of S_i to $2p$ unknowns consisting of p unknown locations and p unknown error values. Any error correction procedure is a method of solving this set of equations for the p pairs of $(e_{j_\ell}, \alpha^{j_\ell})$, $\ell = 1, 2, \ldots, p$. Once $\alpha^{j_1}, \alpha^{j_2}, \ldots, \alpha^{j_p}$ are found, the powers $j_1, j_2, \ldots, j_p$ will indicate the error locations in $E(X)$. In general, there might be many error patterns that satisfy the $2t$ equations. If $p \leq t$, then the error pattern with the smallest p is the actual error pattern.

For notational convenience, let

$$\beta = \alpha^{j_\ell}, \quad \ell = 1, 2, \ldots, p \tag{18}$$

be the error location numbers (refs. 7, 8). Now equation (17) can be written as

$$\left.\begin{aligned} S_1 &= e_{j_1}\beta_1 + e_{j_2}\beta_2 + \ldots + e_{j_p}\beta_p \\ S_2 &= e_{j_1}\beta_1^2 + e_{j_2}\beta_2^2 + \ldots + e_{j_p}\beta_p^2 \\ &\vdots \\ S_{2t} &= e_{j_1}\beta_1^{2t} + e_{j_2}\beta_2^{2t} + \ldots + e_{j_p}\beta_p^{2t} \end{aligned}\right\} \tag{19}$$

These $2t$ syndrome components are symmetric functions in $\beta_1, \beta_2, \ldots, \beta_p$, which are known as power-sum symmetric functions. Next, let the error location polynomial be defined as follows:

$$\sigma(X) = (1+\beta_1 X)(1+\beta_2 X) \ldots (1+\beta_p X)$$
$$= \sigma_0 + \sigma_1 X + \sigma_2 X^2 + \ldots + \sigma_p X^p \qquad (20)$$

where

$$\left.\begin{aligned}
\sigma_0 &= 1 \\
\sigma_1 &= \beta_1 + \beta_2 + \ldots + \beta_p \\
\sigma_2 &= \beta_1\beta_2 + \beta_1\beta_3 + \ldots + \beta_{p-1}\beta_p \\
\sigma_3 &= \beta_1\beta_2\beta_3 + \beta_1\beta_2\beta_3 + \ldots + \beta_{p-2}\beta_{p-1}\beta_p \\
&\vdots \\
\sigma_p &= \beta_1\beta_2\beta_3 \ldots \beta_p
\end{aligned}\right\} \qquad (21)$$

The roots of $\sigma(X)$ are $\beta_1^{-1}, \beta_2^{-2}, \ldots, \beta_p^{-1}$, which are the inverses of the error location numbers. It can be seen from equations (19) and (21) that the coefficients of $\sigma(X)$ are related to the syndrome components S_i, $i = 1, 2, \ldots, 2t$. The coefficients $\sigma_1, \sigma_2, \ldots, \sigma_p$ are known as elementary symmetric functions of $\beta_1, \beta_2, \ldots, \beta_p$. Therefore, if it is possible to find $\sigma(X)$ from the syndrome components, the error location numbers can be found and the error pattern $E(X)$ can be determined. In the following, an effective decoding procedure is described and a design implementation for the (31,15) RS code is given. This procedure consists of four major steps as follows (refs. 7, 8).

1. Calculate the syndrome $S = (S_1, S_2, \ldots, S_{2t})$ from $R(X)$
2. Calculate the error location polynomial $\sigma(X)$ from S
3. Determine the error locations X_j by finding the roots of $\sigma(X)$
4. Calculate the error value e_j from X_j and S

STEP 1. SYNDROME CALCULATION

There are two methods for calculating the syndrome (S): the standard method, and the shift register method. The standard method uses equation (13) because this is the way the syndrome is defined. The shift register method uses the $g(X)$ encoding shift register.

In the standard method,, the syndrome calculation given in equation (13) is

$$\left.\begin{aligned}
S_1 &= r_0 + r_1\alpha^1 + r_2\alpha^2 + \ldots + r_{n-1}\alpha^{n-1} \\
S_2 &= r_0 + r_1\alpha^2 + r_2\alpha^4 + \ldots + r_{n-1}\alpha^{2(n-1)} \\
S_3 &= r_0 + r_1\alpha^3 + r_2\alpha^6 + \ldots + r_{n-1}\alpha^{3(n-1)} \\
&\vdots \\
S_{2t} &= r_0 + r_1\alpha^{2t} + r_2\alpha^{4t} + \ldots + r_{n-1}\alpha^{2t(n-1)}
\end{aligned}\right\} \qquad (22)$$

In matrix form, equation (22) can be written as

$$S = RH^T = (r_0\ r_1\ r_2\ \ldots\ r_{n-2}\ r_{n-1}) \left[\begin{array}{c} (\alpha^i)^j \\ i = 1, 2, 3, \ldots, 2t \\ j = 0, 1, 2, \ldots, n-1 \end{array} \right]_{n \times 2t} \tag{23}$$

where H^T is an n rows by 2t columns matrix. For the (31,15) t = 8 RS code, H^T is a 31×16 matrix shown in figure 4. An all-parallel syndrome calculation would require about 480 multipliers and 16 31-input modulo-2 adders in $GF(2^5)$, which is indeed very expensive. One compromise is a serial-parallel method using 16 circuits of the type shown in figure 5. These circuits required 31 clock cycles to calculate S, or about 1.55 μsec at the present STTL technology.

H^T =

	1	2	3	4	5	•••	15	16
0	1	1	1	1	1	•••	1	1
1	α	α^2	α^3	α^4	α^5	•••	α^{15}	α^{16}
2	α^2	α^4	α^6	α^8	α^{10}	•••	α^{30}	α
3	α^3	α^6	α^9	α^{12}	α^{15}	•••	α^{14}	α^{17}
4	α^4	α^8	•	•	•		•	•
5	α^5	α^{10}	•	•	•		•	•
6	α^6	α^{12}	•	•	•		•	•
7	α^7	α^{14}	•	•	•	•	•	•
8	α^8	α^{16}	•	•	•	•••	•	•
9	α^9	α^{18}	•	•	•	•	•	•
10	α^{10}	α^{20}	•	•	•		•	•
11	α^{11}	α^{22}	•	•	•		•	•
12	α^{12}	α^{24}	α^5	α^{17}	α^{29}	•••	α^{25}	α^6
•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•
29	α^{29}	α^{27}	α^{25}	α^{23}	α^{21}	•••	α	α^{30}
30	α^{30}	α^{29}	α^{28}	α^{27}	α^{26}	•••	α^{16}	α^{15}

Figure 4. - Parity-check matrix for the (31,15) t = 8 Reed-Solomon code.

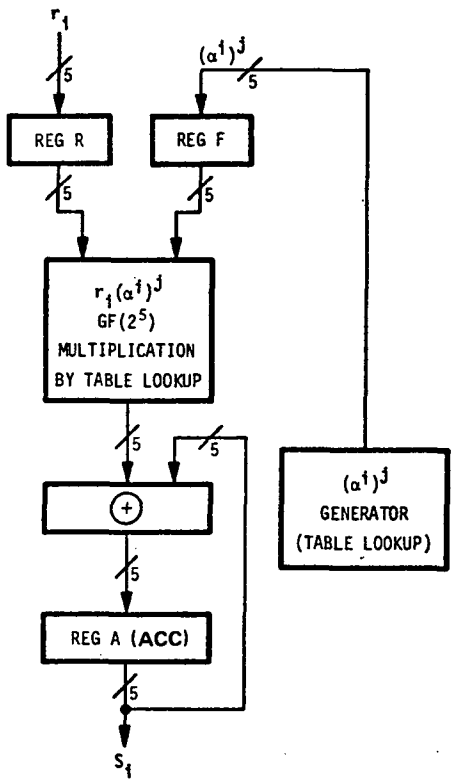


Figure 5. - Circuit for calculating the syndrome component S_i. Approximately six STLL MSI ICs are required.

For large $t(t \geq 4)$, the best way to calculate the syndrome is to use the g(X) shift register. This will result in some saving in logic because this shift register is already used for encoding. However, the S calculated by g(X) is not the same as the S calculated by $R(\alpha^i)$ of equation (13), but they are related. This relationship is described below.

From equation (13), let

$$S = (S_1, S_2, \ldots, S_{2t}) \tag{24}$$

be the syndrome calculated by $R(\alpha^i)$ with

$$S_i = R(\alpha^i) \tag{25}$$

for $i = 1, 2, \ldots, 2t$. Let

$$S^* = (S_1^*, S_2^*, \ldots, S_{2t}^*) \tag{26}$$

be the remainder calculated by dividing R(X) by g(X). The remainder S* is another form of the syndrome but $S^* \neq S$. Using the Euclidean division algorithm, the result of R(X)/g(X) is

$$R(X) = Q(X)\, g(X) + S^*(X) \tag{27}$$

where Q(X) is the quotient and

$$S^*(X) = S_1^* + S_2^* X + S_3^* X^2 + \ldots + S_{2t}^* X^{2t-1} \tag{28}$$

is the remainder. Substituting X by α^i in equation (27) gives the result

$$\begin{aligned} R(\alpha^i) &= Q(\alpha^i)\, g(\alpha^i) + S^*(\alpha^i) \\ &= S^*(\alpha^i) \end{aligned} \tag{29}$$

since $Q(\alpha^i)\, g(\alpha^i) = 0$. From equations (25), (28), and (29), the relationship between S and S^* is

$$\begin{aligned} S_i &= S^*(\alpha^i) \\ &= S_1^* + S_2^*(\alpha^i) + S_3^*(\alpha^i)^2 + \ldots + S_{2t}^*(\alpha^i)^{2t-1} \end{aligned} \tag{30}$$

for $i = 1, 2, \ldots, 2t$. The relationship of equation (30) can be implemented by the circuit shown in figure 6. This circuit is very similar to the Chien search circuit as described later in step 3. In figure 6, the $2t$ components of S can be obtained from S^* in $2t$ clock cycles.

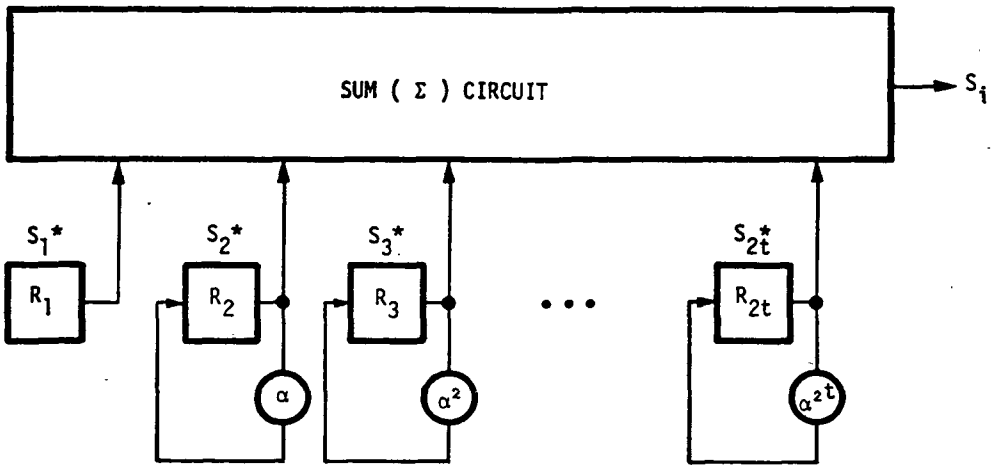


Figure 6. - Modified Chien search circuit for obtaining S_i from S_i^*

In the above discussions, two basic methods for syndrome calculations were presented. For large t, it is concluded that the best way to calculate the syndrome is to divide $R(X)$ by $g(X)$ to obtain $S^*(X)$, and then obtain S_i from S_i^* by a modified Chien search circuit. If $R(X)$ contains no error, $S^*(X) = 0$, and this calculation takes n clock cycles, which is the same speed as the serial-parallel method of figure 5. If $R(X)$ contains an error, S_i is obtained from S_i^*; then go to step 2, and so on. One significant point, which should be mentioned, is that this method of syndrome calculation requires practically no additional logics because both the $g(X)$ shift register and the Chien search circuit already exist.

STEP 2. $\sigma(X)$ CALCULATION

A method for calculating the error location polynomial $\sigma(X)$ is given below without proof. This method is the Lin's table (ref. 7), which is an interpretation of Berlekamp's iterative algorithm (ref. 9). The same method can also be found in reference 8 in a slightly different form.

To find $\sigma(X)$, start with the table

μ	$\sigma^{(\mu)}(X)$	d_μ	ℓ_μ	$\mu - \ell_\mu$
-1	1	1	0	-1
0	1	S_1	0	0
1				
2				
⋮				
2t				

and proceed to fill out the table. Assuming that the table has all rows filled out up to and including the μth row, then fill out the $(\mu+1)$th row as follows:

1. If $d_\mu = 0$, then $\sigma^{(\mu+1)}(X) = \sigma^{(\mu)}(X)$ and $\ell_{\mu+1} = \ell_\mu$

2. If $d_\mu \neq 0$, find another row preceding the μth row, say the ρth row, such that the number $\rho - \ell_\rho$ in the last column of the table has the largest value and $d_\rho \neq 0$. Then

$$\sigma^{(\mu+1)}(X) = \sigma^{(\mu)}(X) + d_\mu d_\rho^{-1} X^{(\mu-\rho)} \sigma^{(\rho)}(X) \tag{31}$$

and

$$\ell_{\mu+1} = \max[\ell_\mu, \ell_\rho + \mu - \rho] \tag{32}$$

In either case,

$$d_{\mu+1} = S_{\mu+2} + \sigma_1^{(\mu+1)} S_{\mu+1} + \ldots + \sigma_{\ell_{\mu+1}}^{(\mu+1)} S_{\mu+2-\ell_{\mu+1}} \tag{33}$$

where the $\sigma_i^{(\mu+1)}$ are the coefficients of $\sigma^{(\mu+1)}(X)$,

$$\sigma^{(\mu+1)}(X) = 1 + \sigma_1^{(\mu+1)} X + \sigma_2^{(\mu+1)} X^2 + \ldots + \sigma_{\ell_{\mu+1}}^{(\mu+1)} X^{\ell_{\mu+1}} \tag{34}$$

If $V(X)$ has exactly t errors, then the polynomial $\sigma^{(2t)}(X)$ in the last row is the required $\sigma(X)$. If $V(X)$ has more than t errors, then $\sigma^{(2t)}(X)$ has degree greater than t, and generally it is not possible to locate the errors. If $V(X)$ has fewer than t errors, the table terminates into a mode prior to step $2t$ where $d_\mu = d_{\mu+1} = 0$ and $\sigma^{(\mu)}(X) = \sigma^{(\mu+1)}(X)$.

The computation for $\sigma^{(\mu)}(X)$ and d_μ combined, on the average, requires about $2t$ additions and $2t$ multiplications for each step. Since there are $2t$ steps, the total is about $4t^2$ additions and $4t^2$ multiplications. For the (31,15) $t = 8$ RS code, at most 16 iterative steps are required to obtain $\sigma(X)$, and each step requires about 16 additions and 16 multiplications in $GF(2^5)$, plus inversions. At the present technology, an economic method for performing these computations is to have a Galois field arithmetic unit implemented by microprogramming a microprocessor like the 2900 series microprocessor family. With a few hardware augmentations for inversions and special instruction controls, these Galois field arithmetic computations can be executed at 100 nsec per instruction. At this rate and allowing a 300% program overhead, the $\sigma(X)$ calculation would take about 100 μsec.

STEP 3. DETERMINATION OF ERROR LOCATIONS

The error location numbers are the reciprocals of the roots of $\sigma(X)$. The roots of $\sigma(X)$ can be found simply by substituting $1, \alpha, \alpha^2, \ldots, \alpha^{n-1}$ into $\sigma(X)$. Since $\alpha^n = 1$, then $\alpha^{-\ell} = \alpha^{n-\ell}$. Therefore, if α^ℓ is a root of $\sigma(X)$, $\alpha^{n-\ell}$ is an error location number and the received digit $r_{n-\ell}$ is in error. If n is large, this substitution method is not desirable because the length of computation can be long.

If σ_i, $i = 1, 2, \ldots, p$, and $p \leq t$ are known from step 2, and using the fact that RS codes are cyclic, a procedure credited to Chien known as the Chien search (refs. 7-9) can be used to find the error location numbers. The received codeword $R(X)$ is tested on a digit-by-digit basis starting with the high-order digit r_{n-1} first. To decode r_{n-1}, the decoder tests whether α^{n-1} is an error location number. This is equivalent to testing whether α is a root of $\sigma(X)$. If α is a root, then from equations (20) and (21), the result is

$$\sigma_1 \alpha + \sigma_2 \alpha^2 + \ldots + \sigma_p \alpha^p = 1 \tag{35}$$

If equation (35) is satisfied, then α^{n-1} is an error location number and the digit r_{n-1} has an error; otherwise, r_{n-1} has no error. To decode $r_{n-\ell}$, the decoder tests

$$\sigma_1 \alpha^\ell + \sigma_2 \alpha^{2\ell} + \ldots + \sigma_p \alpha^{p\ell} = 1 \tag{36}$$

If equation (36) is satisfied, then α^ℓ is a root of $\sigma(X)$ and the digit $r_{n-\ell}$ has an error; otherwise, $r_{n-\ell}$ has no error.

The error location numbers testing procedure of the Chien search can be implemented in a straightforward manner by a circuit such as that shown in figure 7. The t σ-registers $R_1, R_2, \ldots, R_t$ are initially stored with $\sigma_1, \sigma_2, \ldots, \sigma_t$ calculated in step 2. It should be noted that if $p < t$, the register stages $R_{p+1}, R_{p+2}, \ldots, R_t$ are stored with zero since $\sigma_{p+1} = \sigma_{p+2} = \ldots = \sigma_t = 0$. To test r_{n-1}, the circuit is pulsed once. The multiplications are performed and $\sigma_1 \alpha, \sigma_2 \alpha^2, \ldots, \sigma_p \alpha^p$ are stored in the σ-registers. The output of the SUM circuit is 1 if and only if the sum

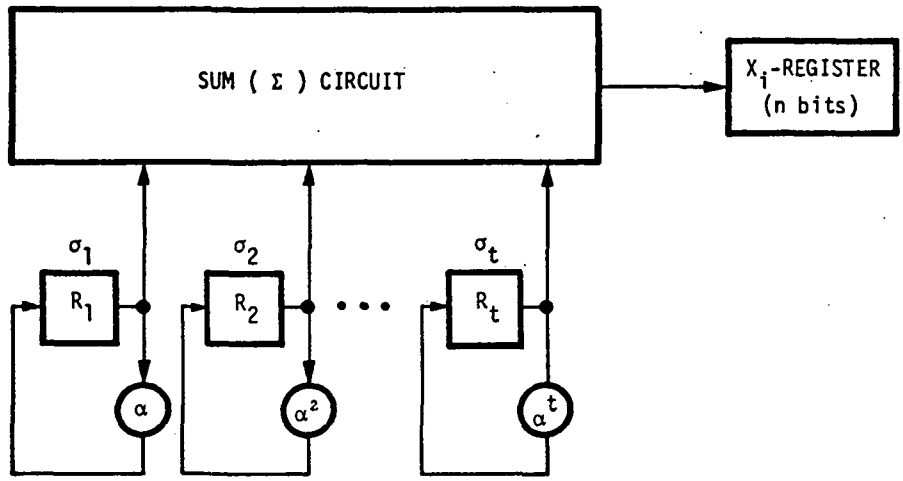


Figure 7. - Chien search cyclic error location circuit.

$$\sigma_1 \alpha + \sigma_2 \alpha^2 + \ldots + \sigma_p \alpha^p = 1$$

Otherwise, the output of SUM is 0 or α^i. Having tested r_{n-1}, the circuit is pulsed again. Now $\sigma_1 \alpha^2, \sigma_2 \alpha^4, \ldots, \sigma_p \alpha^{2p}$ are stored in the σ-registers. The output of the SUM circuit is 1 if and only if the sum

$$\sigma_1 \alpha^2 + \sigma_2 \alpha^4 + \ldots + \sigma_p \alpha^{2p} = 1$$

Otherwise, the output of SUM is 0 or α^i. This process continues until all n digits of R(X) are tested. The SUM output is shifted to an n-bit X_i-register. For example, for $n = 15$ and if the pattern in the X_i-register is

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	1	0	0	1	0	0	0	0	0	1	0	0

then the digits numbered 12, 6, and 3 of R(X) are in errors.

For the (31,15) $t = 8$ RS code, eight 5-tuple register stages are required. The multiplication circuits of $\alpha, \alpha^2, \ldots, \alpha^8$ can be implemented by table lookup using ROMs. The function of the SUM circuit can best be performed by the microprocessor since this function requires only eight XOR operations and a testing for "1" operation.

STEP 4. CALCULATION OF ERROR VALUES

The error values e_{j_ℓ}, $\ell = 1, 2, \ldots, p$, can be calculated from equation (19) once the syndrome components $S_1, S_2, \ldots, S_{2t}$ and the error locations numbers $\beta_1, \beta_2, \ldots, \beta_p$ are known. The set of equations in equation (19) is linear and one method of solving it for e_{j_ℓ} is credited to Forner (refs.

7-9). It should be noted that the error location number β_ℓ is equal to α^{j_ℓ}, and $\alpha^{j_\ell} = \alpha^{n-\ell} = \alpha^{-\ell}$, where α^ℓ is a root of $\sigma(X)$ as described in step 2. Once the p pairs (X_{j_ℓ}, e_{j_ℓ}) for $\ell = 1, 2, \ldots, p$ are known, E(X) in equation (12) can be determined.

To find e_{j_ℓ}, let

$$Z(X) = 1 + (S_1 + \sigma_1)X + (S_2 + \sigma_1 S_1 + \sigma_2)X^2 + \ldots + (S_p + \sigma_1 S_{p-1} + \sigma_2 S_{p-2} + \ldots + \sigma_p)X^p \quad (37)$$

then

$$e_{j_\ell} = \frac{Z(\beta_\ell^{-1})}{\prod_{\substack{i=1 \\ i \neq \ell}}^{p} (1 + \beta_i \beta_\ell^{-1})} \quad (38)$$

The computation of e_{j_ℓ} in equation (38) is relatively complex. For a t-error-correcting RS code, and assuming E(X) has $p \leq t$ digits in errors, the computation of Z(X) requires $p + p(p+1)/2$ additions and $p(p+1)/2$ multiplications for each digit in error. Let D be the denominator of equation (38). The interpretation of D is that D should consist of P - 1 terms. The missing term in D is the term where $\beta_i = \beta_\ell$. The computation of D requires p - 1 additions and (p-1) + (p-2) multiplications for each digit in error.

For the (31,15) t = 8 RS code, the computation of e_{j_ℓ} in equation (38) can be performed by the Galois field arithmetic unit implemented by microprograms in the 2900 series microprocessor. Assuming a 100-nsec instruction execution rate and a 100% program overhead, the computation for each error value will require about 20 μsec, or about 200 μsec for all eight error values.

CONCLUSION

A decoding procedure has been described for the (n,k) t-error-correcting Reed-Solomon (RS) codes. The code parameters are chosen such that the codes are cyclic so that encoding and some parts of decoding can be implemented by shift registers. In particular, a logic implementation scheme has been presented for the (31,15) t = 8 RS code. The principal features are (1) a Galois field arithmetic unit implemented by microprogramming a 2900 series microprocessor, and (2) syndrome calculation by using the g(X) encoding shift register. This arithmetic unit is capable of executing instructions at a 100-nsec clock cycle rate. Encoding the (31,15) code would be implemented by shift registers requiring 31 clock cycles (3.1 μsec). Decoding the (31,15) code requires four steps of computation with a total decoding time of about 500 μsec. With present technology, it is felt that the microprogrammable

microprocessor is the most feasible approach to decode this code except for the case where n and t are small, for example, $n \leq 15$ and $t \leq 2$. For these cases, an all-parallel method is probably more suitable.

Ames Research Center
National Aeronautics and Space Administration
Moffett Field, California 94035, March 8, 1978

REFERENCES

1. Introduction to the IBM 3850 Mass Storage System (MSS). IBM Document GA32-0028-2, July 1975, p. 57; IBM 3850 Mass Storage System Installation Guide. IBM Document GA32-0030-1, April 1976, p. 17.

2. 38500 Mass Storage System General Information Manual. Control Data Corporation Document 22294400, 1976; 38500 Mass Storage Reference Manual. Control Data Corporation Document 22083000, 1976.

3. Brown, D. T.; and Sellers, F. F., Jr.: Error Correction for IBM 800-bit-per-inch Magnetic Tape. IBM J. Res. Develop., July 1970, pp. 384-389.

4. Berlekamp, E. R.: Algebraic Codes for Improving the Reliability of Tape Storage. AFIPS Conf. Proc., vol. 44, May 19-22, 1975, pp. 497-499.

5. Poland, W. B.; Prine, G. E.; and Jones, T. L.: Archival Performance of NASA GFSC Digital Magnetic Tape. AFIPS Conf. Proc., vol. 42, June 4-8, 1973, Part II, pp. M68-M73.

6. Lim, R. S.: Technical Manual for Q1011, QM1011, and M1011: Virtual and DMA Central Memory Interface for PDP-11. IAC Report, Institute for Advanced Computation, NASA-Ames Research Center, NASA, July 1976.

7. Lin, S.: An Introduction to Error-Correcting Codes. Prentice-Hall, Englewood Cliffs, New Jersey, 1970.

8. Peterson, W. W.; and Weldon, E. J., Jr.: Error-Correcting Codes. 2nd Edition, MIT Press, Cambridge, Mass., 1972.

9. Berlekamp, E. R.: Algebraic Coding Theory. McGraw-Hill Book Co., Inc., New York, 1968.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TP - 1286		

4. Title and Subtitle	5. Report Date
A DECODING PROCEDURE FOR THE REED-SOLOMON CODES	August 1978
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Raymond S. Lim	A-7372
	10. Work Unit No. 366-18-50-00-00
9. Performing Organization Name and Address	11. Contract or Grant No.
NASA Ames Research Center Moffett Field, Calif. 94035	
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Technical Paper
National Aeronautics and Space Administration Washington, D. C. 20546	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

This paper describes a decoding procedure for the (n,k) t-error-correcting Reed-Solomon (RS) code, and an implementation of the (31,15) RS code for the I4-TENEX central system. This code can be used for error correction in large archival memory systems. The principal features of the decoder are (1) a Galois field arithmetic unit implemented by microprogramming a microprocessor, and (2) syndrome calculation by using the g(x) encoding shift register. Complete decoding of the (31,15) code is expected to take less than 500 μsec. The syndrome calculation is performed by hardware using the encoding shift register and a modified Chien search. The error location polynomial is computed by using Lin's table, which is an interpretation of Berlekamp's iterative algorithm. The error location numbers are calculated by using the Chien search. Finally, the error values are computed by using Forney's method.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Error-correcting codes Memory error correction Reliable memory systems	Unlimited STAR Category - 62

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	24	$3.50

*For sale by the National Technical Information Service, Springfield, Virginia 22161

NASA-Langley, 1978

National Aeronautics and
Space Administration

Washington, D.C.
20546

Official Business
Penalty for Private Use, $300

THIRD-CLASS BULK RATE

Postage and Fees Paid
National Aeronautics and
Space Administration
NASA-451

U.S.MAIL

NASA

POSTMASTER: If Undeliverable (Section 158
Postal Manual) Do Not Return