NASA TECHNICAL MEMORANDUM

NASA TM X-62,442

NASA TM X-62,442

AUGMENTED BURST-ERROR CORRECTION FOR UNICON LASER MEMORY

Raymond S. Lim

Ames Research Center
Moffett Field, Calif. 94035

June 1974

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM X-62,442		

4. Title and Subtitle	5. Report Date
AUGMENTED BURST-ERROR CORRECTION FOR UNICON LASER MEMORY	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Raymond S. Lim	A-6090
	10. Work Unit No. 997-36-60-09-08
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 Technical Memorandum
12. Sponsoring Agency Name and Address	14. Sponsoring Agency Code
National Aeronautics and Space Administration Washington, D. C. 20546	

15. Supplementary Notes

16. Abstract

This paper describes a proposed augmented single-burst-error correction system for data stored in the UNICON Laser Memory. In the proposed system, a long Fire Code with code length $n > 16,768$ bits is used as an outer code to augment an existing inner shorter Fire Code for burst error corrections. The inner Fire Code is a (80,64) code shortened from the (630,614) code, and it is used to correct a single-burst-error on a per-word basis with burst length $b \leq 6$. The outer code, with $b \leq 12$, would be used to correct a single-burst-error on a per-page basis, where a page consists of 512 32-bit words. In the proposed system, the encoding and error detection processes would be implemented by hardware. A minicomputer, currently used as a UNICON memory management processor, would be used on a time-demanding basis for error correction. Based upon existing error statistics, this combination of an inner code and an outer code would enable the UNICON system to obtain a very low error rate in spite of flaws affecting the recorded data. The approach of the long Fire Code described here is also applicable to other mass memory systems (such as rotating disks) where single-burst-error correction can be obtained at very low redundancy.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Error Correction Mass Memory	Unclassified-Unlimited STAR Category 62

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	26	$3.75

*For sale by the National Technical Information Service, Springfield, Virginia 22151

AUGMENTED BURST-ERROR CORRECTION FOR UNICON LASER MEMORY

Raymond S. Lim

Institute for Advanced Computation

Ames Research Center, NASA, Moffett Field, Calif. 94035

SUMMARY

This paper describes a proposed augmented single-burst-error correction system for data stored in the UNICON Laser Memory. In the proposed system, a long Fire Code with code length $n > 16{,}768$ bits is used as an outer code to augment an existing inner shorter Fire Code for burst-error corrections. The inner Fire Code is a (80,64) code shortened from the (630,614) code, and it is used to correct a single-burst error on a per-word basis with burst length $b \leq 6$. The outer code, with $b \leq 12$, would be used to correct a single-burst error on a per-page basis, where a page consists of 512 32-bit words. In the proposed system, the encoding and error detection processes would be implemented by hardware. A minicomputer, currently used as a UNICON memory management processor, would be used on a time-demanding basis for error correction. Based upon existing error statistics, this combination of an inner code and an outer code would enable the UNICON system to obtain a very low error rate in spite of flaws affecting the recorded data. The approach of the long Fire Code described here is also applicable to other mass memory systems (such as rotating disks) where single-burst-error correction can be obtained at very low redundancy.

INTRODUCTION

The UNICON laser memory, or UNICON, is a digital memory manufactured by Precision Instrument Co. for the ILLIAC-IV computer as part of the mass I/O storage. The UNICON has an on-line storage capacity of about 10^{12} bits. In such a high-density data storage system, contamination and other defects can easily obliterate a group of data bits. In order to operate successfully in spite of this problem in the UNICON, an elaborate error-correction system is used.

Error statistics obtained on the operation and checkout of the UNICON indicate that the desired net error rate of one bad word of data per 2×10^{11} bits read might be met if two Fire Codes were used for single-burst-error correction. These two Fire Codes would be arranged as a system consisting of an inner code and an outer code. The inner code would be a short (80,64) Fire Code used to correct an error burst of length $b \leq 6$ on a per-word basis. The outer code would be a long (16803,16768) Fire Code used to correct an error burst of length $b \leq 12$ on a per-page basis. It should be noted that this system of an inner code and an outer code would not function in the same sense as the concepts of concatenated codes (ref. 1). Furthermore, the Reed-Solomon code from $GF(2^m)$ successfully used in the IBM Photo-Digital memory (ref. 2) is not feasible for the UNICON because of the long codeword length of n=16803.

The inner (80,64) Fire Code currently in use is implemented all in hardware by shift registers and error correction is done on-line as data are read off the strip. The (16803,16768) outer Fire Code would

be decoded in a manner suggested by Chien (ref. 3). A similar scheme for burst error correction with $b \leq 11$ was implemented successfully in the IBM model 3830 controller for the Model 3330 disk. The problem of effectively implementing the (16803,16768) outer code has been solved by a number of innovations and in a hybrid manner. Most important is the use of hardware for encoding, while using a minicomputer on a time-demanding basis for error correction. The minicomputer performs a calculation to find the burst error location i by applying two well-known theorems from Number Theory: the Euclidean algorithm for division and the Chinese remainder theorem for simultaneous congruences, sometimes thought to be merely of academic interest only to Number Theorists.

CODING REQUIREMENT FOR UNICON

The basic storage element of the UNICON is a metal coated plastic strip about 4.75 x 31 in. The recording density across the strip is 11805 tracks over 3.8 in., or about 3000 tracks per in. Each track can store up to eight records of data, with each record containing 16,768 data bits plus other record identification bits. The method for recording data on the strip is to use the laser to either burn a hole for a 1-bit, or not to burn a hole for a 0-bit.

Data entering and leaving the UNICON are handled by a UNICON Controller (ref. 4). The Controller has two interfaces: a central memory (CM) interface for data, and a minicomputer interface for control. The minicomputer (a PDP-11) functions as the UNICON memory management processor (UMP). The UMP obtains its control programs

through its own virtual hardware interface into the CM. By means of this virtual address space, UMP can address up to 4096 pages in CM. Data transfer to the UNICON is always one page of data at a time from CM, where a page of data consists of 512WX32B, which is 16,384 bits. For each page of data, the controller attaches 16WX16B of header information and 8WX16B of checksum parity bits to form a data record of 16,768 bits. The checksum is simply a single-parity check with check symbols from $GF(2^{128})$.

The basic word length in the UNICON is 64 bits. As previously mentioned, the UNICON currently has a built-in (80,64) Fire Code which is used to correct a single-burst error with burst length $b \leq 6$ in a per-word-basis. Because of optic imperfections and other mechanical flaws, data read back from the strip always contain errors. When the UNICON is properly aligned, most errors are correctable by the (80,64) Fire Code. Preliminary observations on the error statistic indicate that the probability of getting an error not correctable by this Fire Code is about one for every 10^{11} bits of data transferred, which is about one error per strip. The observed uncorrectable errors, for those that are understood and the cause can be explained, are typically single-burst errors with $b \leq 11$. Thus if a long Fire Code with $b \leq 12$ is used to augment the existing (80,64) Fire Code to correct a single-burst-error on a per-page basis, then it is possible that the error rate can be reduced to a very low level. The long Fire Code chosen is a (16803,16758) code. In this arrangement, the (16803,16768) code is called the outer code, while the (80,64) code is called the inner code.

DESCRIPTION OF THE FIRE CODE

An error burst of length b is defined as a vector whose non-zero components are confined to b consecutive bit positions, the first and last of which are non-zero. It is known that cyclic codes for single-burst-error correction can be systematically constructed (refs. 5 and 6). These codes are rather efficient and they can be implemented with feedback shift registers. An important class of these codes is known as the Fire Codes (ref. 7). In the discussion which follows, let

n = length of codeword in bits.

k = number of message bits in a codeword.

$n-k$ = number of redundant bits in a codeword.

b = maximum length of single error burst which can be corrected, in bits.

$m(x)$ = message word.

$v(x)$ = transmitted codeword.

$E(x)$ = error burst.

$R(x) = v(x) + E(x)$ = received codeword.

It is clear that for a given k and b, the objective is to construct an (n,k) code with as small a redundancy $n-k$ as possible.

It is also known from the Reiger bound (ref. 8) that the number of parity check bits of a b-burst-error correcting code is

$$n-k \geq 2b \tag{1}$$

This is an upper bound, and codes which meet the Reiger bound are said to be optimal. The ratio

$$Z = \frac{2b}{n-k} \tag{2}$$

is a measure of the burst-correcting efficiency of the code. An optimal code has $Z=1$. The fire code at best has

$$Z = \frac{2b}{3b-1} \simeq \frac{2}{3} \tag{3}$$

which is not very optimal with respect to the Reiger bound.

The Fire Code is a linear cyclic code which can be systematically constructed for correcting a single burst of error in a codeword of n bits. The constructed codeword is in systematic form; that is, the $n-k$ parity check bits are simply concatenated after the message bits. A Fire Code with code symbols from $GF(2)$ which is capable of correcting any burst of length b or less and of simultaneously detecting any burst of length $d \geq b$ is best described by its generator polynomial

$$g(x) = p(x)\ (x^c + 1) \tag{4}$$

where

$p(x)$ = an irreducible polynomial of degree m whose roots have order e; that is, the period of $p(x)$ is e.

$c \geq b + d - 1$

$d \geq b$

$m \geq b$

$(c,e) = 1$, that is, c and e are relatively prime.

For pure error correction purpose, the best choice is to set $m=b$ and $d=b$. This results in a Fire Code with the following parameters:

$$n = \text{LCM}\ (e,c) = ec$$

$$n-k = c + m = 3b - 1$$

$$k = ec - (3b - 1)$$

Another version of the Fire Code, which is the half-length version, has the following code paraments:

$$n = (\frac{c}{2})\ e, \text{ where } c \text{ is an even integer.}$$

$$n-k = 3b - 2$$

$$k = (\frac{c}{2})\ e - (3b - 2)$$

In the conventional decoder implemented by shift registers, decoding the Fire Code requires two n shifts, n shifts for parity checking and another n shifts for error correction. As the first n shifts attributed to parity checking are concurrent with the reading operation, no time delay occurs in this operation. The second n shifts needed to locate the error burst and correct it is a time delay due to error correction. If n is very large, like 16768 bits, then this long time delay may be intolerable in real life operation. However, if there exists within easy reach some power for general computation, resided either within the host computer or within the controller of the storage system, then the second n shifts can be reduced to the minimum of $(e+c-2)$ plus computation time.

THE (80,64) FIRE CODE AND ITS IMPLEMENTATION

The generator polynomial for the (80,64) Fire Code is

$$g(x) = (x^6 + x + 1)\ (x^{10} + 1) \tag{5}$$

In this case, $p(x) = (x^6 + x + 1)$ is primitive so the order of its roots is

$$e = 2^6 - 1 = 63 \tag{6}$$

The code parameters for this code are as follows:

$(63,10) = 1$, that is, 63 and 10 are relatively prime.

$$n = ec = 63 \times 10 = 630$$

$$n-k = c + m = 10 + 6 = 16$$

$$k = 630 - 16 - 614$$

$m = 6$ so that $b = 6$

$$c = 10 = b + d - 1$$

$$= 6 + d - 1 \text{ so that } d = 5$$

Note that the condition $d \geq b$ is not met. This means that this code probably will not be able to correct all possible single-burst errors of length $b \leq 6$.

From the above calculations, the code generated by $g(x)$ in equation (5) is a (630,614) code. Since the word length of the UNICON is 64 bits, the code must be shortened to a (80,64) code by making the $(614 - 64) = 550$ most significant bits of the message equal to zero and omitting them. This process will not affect the encoding and parity checking calculation since leading zeros will not affect them. It does, however, affect the error correction procedure since the unaltered procedure would require 550 shifts corresponding to the 550 omitted zeros before reading the actual received codeword out of the buffer. Instead of making the buffer wait for 550 shifts, the standard procedure is to premultiply the decoding shift register by X^{550} modulo $g(x)$.

The parity-check calculation, as it stands, is the residue of $X^{16} R(x)$ modulo $g(x)$. An additional automatic multiplication by

X^{550} is desired; that is, the residue X^{566} is desired. With the help of a program written in basic, the remainder $r(x)$ after dividing X^{566} by $g(x)$ is found to be

$$r(x) = (x^{13} + x^{12} + x^{11} + x^6 + x^3 + x^2 + x)$$

This basic program is shown in figure 1. The encoding shift register is shown in figure 2, and the decoding shift register is shown in figure 3.

THE (16803,16768) FIRE CODE AND ITS IMPLEMENTATION

The objective of this code is to have $k = 16768$ and $b = 12$. The procedure to find a Fire Code that meets this requirement is as follows:

1. Let $m = d = b = 12$ and $c = 2b - 1 = 23$

   so that $c + m = 23 + 12 = 35$

2. $g(x) = p(x)\ (x^{23} + 1)$

3. $e = \frac{n}{c} = \frac{k + (n-k)}{c} = \frac{16803}{23} = 513+$

   Thus $p(x)$ has degree $m = 12$, and a period $e \geq 514$ is desired. With some looking, the trinomial $(X^{12} + X^5 + 1)$ is not primitive but has a period $e = 819$ is acceptable. Thus $g(x) = (X^{12} + X^5 + 1)(X^{23} + 1)$.

Now that $g(x)$ is found, the code parameters are:

$$n = e_{\cdot} = 819 \times 23 = 18837$$

$$n-k = c + m = 23 + 12 = 35$$

$$k = (18837 - 35) = 18802$$

The result is a (18837,18802) code, and this code can be shortened to (16803,16768) by omitting 2034 leading zeros.

The error correction part of the decoding requires $n = 16803$ shifts, which is too long a time delay for practical application. However, there exists a high-speed decoder for this code suggested by Chien (ref. 3). This high-speed decoder consists of two feedback shift registers, one based on the factor (X^c+1) and the other based on the factor $p(x)$. The two registers are run in synchronism for parity checking and error detection. A general diagram of this decoder is shown in figure 4. In this decoder, the syndrome is presented by the remainder of (X^c+1) and $p(x)$, and the error pattern is identified when it has the same representation in both registers. The operation of this decoder is as follows:

1. Shift both registers n times to enter the received codeword $R(x)$.
2. If $R(x)$ has no error, the syndrome in both registers is zero.
3. If the syndrome is not zero, shift only the (X^c+1) register until its $(b-1)$ high-order bit positions equal to zero. The error pattern is contained in the b low-order bit positions. The maximum number of shifts is $(c-1)$. Freeze this register.
4. Next shift the $p(x)$ register until its contain is matched with the error pattern in the (X^c+1) register. The maximum number of shifts is $(e-1)$. If a correctable error occurs, this match condition will occur, otherwise an uncorrectable error is detected.

From the above description, it is noted that the maximum decoding time is equal to the minimum of (e+c-2), as compared to n for the conventional decoder.

Suppose a correctable burst of length $\leq b$ at location i, $o \leq i < n$, has occurred, then the error can be written as

$$E(x) = x^i B(x) \tag{8}$$

The remainders calculated in the registers are

$$S_p(x) = x^i B(x) \qquad \text{modulo } p(x) \tag{9a}$$

$$S_c(x) = x^i B(x) \qquad \text{modulo } (x^c+1) \tag{9b}$$

It is noted that the syndrome $S_c(x)$ is capable of determining the burst pattern and its location up to a multiple of c. Therefore, by shifting the syndrome $S_c(x)$ in its register and testing for zero at the (b-1) high-order bit positions, the error pattern and its location i can be determined quickly up to multiple of c. That is,

$$i \equiv r_c \qquad \text{modulo } c \tag{10a}$$

where $o \leq r_c < c$. Now that the error pattern is determined, the p(x) register can be shifted until a match is obtained. This gives

$$i \equiv r_p \qquad \text{modulo } e \tag{10b}$$

If i can be solved in equation (10), then the decoding is complete. The problem in equation (10) is a classical problem of simultaneous congruence in Number Theory. If e and c are relatively prime, then the Chinese Remainder Theorem can be used to solve for i.

The Chinese Remainder Theorem simply states that

"Numbers which satisfy the simultaneous congruence $i \equiv a_1 \pmod{\mu_1}, \ldots, i \equiv a_j \pmod{\mu_j}$ exist if the μ_j are relatively

prime in pairs, and such numbers constitute a single number

class modulo $\prod_{1}^{j} \mu_k$."

Now to solve for i. According to the Euclidean division algorithm, if (e,c) = 1, there exists intergers A_c and A_p such that

$$A_c\, e + A_p\, c = 1 \qquad \text{modulo } (n = ec) \tag{11}$$

Multiply both sides by i, the result is

$$i = (A_c\, e)\, i + (A_p\, c)\, i \qquad \text{modulo } n \tag{12}$$

From the shift registers, note that

$$i = q_c\, c + r_c \qquad \text{for } (x^c + 1) \text{ register} \tag{13a}$$

$$i = q_p\, e + r_p \qquad \text{for } p(x) \text{ register} \tag{13b}$$

By substituting equation (13) into equation (12), the result is

$$i = (A_c\, e)\, r_c + (A_p\, c)\, r_p \qquad \text{modulo } n \tag{14}$$

Since

$$A_c\, q_c\, e\, c \equiv 0 \qquad \text{modulo } n \tag{15a}$$

$$A_p\, q_p\, e\, c \equiv 0 \qquad \text{modulo } n \tag{15b}$$

The key equation is equation (14). The computation of A_c and A_p can be done easily off line by the UMP with the numbers $A_c e$ and $A_p c$ stored in its memory. Once the error-pattern-match condition is found in the decoder, r_c and r_p are available and the UMP can compute i by equation (14). Once i is solved, UMP will fetch the appropriate words from CM and perform the error correction with the error pattern stored in the p(x) register.

It is not convenient to describe the detailed implementation of the (16803,16768) Fire Code since its code length is too long, 16803 bits. However, in order to illustrate the principle of this high-speed decoding

scheme, a shorter (35,27) Fire Code with $b \leq 3$ shortened to a (24,16) code is fully illustrated in the Appendix.

CONCLUSION

This paper has presented an augmented scheme for single-burst-error correction for the UNICON laser memory. The augmentation is the addition of an outer code, a (16803,16768) Fire Code, to correct for a single error burst with length $b \leq 12$ on a per-page basis. This (16803,16768) Fire Code is an addition to the existing (80,64) Fire Code which has already been implemented in the UNICON to correct a single error burst with length $b \leq 6$ in a per-word-basis. Based upon observed error statistics, this augmented scheme would reduce the error rate to about one error per 2×10^{11} bits of data transfer, which is about one error per two strips read. The (16803,16768) code would be implemented by a hybrid method using a high-speed hardware decoder for syndrome calculation and a minicomputer for error location computation and error correction. For the case where the periods of the two polynomial factors of $g(x)$ are relatively prime, a simple application of the Chinese Remainder Theorem would determine the burst location.

ACKNOWLEDGEMENT

The author wishes to thank Dr. Mel Pirtle for his initial suggestion for augmenting the burst-error-correction for the UNICON.

REFERENCES

1. Forney, G. D., Jr.: Concatenated Codes. MIT Research Monograph No. 37, The MIT Press, Cambridge, Mass., 1966.

2. Oldham, I. B.; Chien, R. T.; and Tang, D. T.: Error Detection and Correction in a Photo-Digital Storage System. IBM J. Res. Dev., Nov. 1968, pp. 422-430.

3. Chien, R. T.: Burst-Correcting Codes with High-Speed Decoding. IEEE Trans. on Information Theory, Jan. 1969, pp. 109-113.

4. Lim, R. S.: A Channel Controller for the Unicon Laser Memory (to be submitted for publication as a TM).

5. Peterson, W. W.: Error-Correcting Codes. MIT Press and John Wiley & Sons, 1961.

6. Peterson, W. W. and Weldon, E. J.: Error-Correcting Codes, 2nd Ed., MIT Press, Cambridge, Mass., 1972.

7. Fire, P.: A Class of Multiple-Error-Correcting Binary Codes for Non-Independent Errors. Sylvania Rec. Sys. Lab., Mountain View, Calif., Sylvania Report RSL-E-2, 1959.

8. Reiger, S. H.: Codes for the Correction of Clustered Errors. IRE Trans. on Information Theory, Mar., 1960, pp. 16-21.

APPENDIX

The purpose of this appendix is to illustrate the principle of the high-speed decoder by implementing a (24,16) Fire Code shortened from the (35,27) code. The generator polynomial for this code is:

$$g(x) = (x^3 + x + 1)\ (x^5 + 1)$$

where $p(x) = (x^3 + x = 1)$ is primitive and hence its order $e = 7$. The parameters of the code are:

$$m = b = 3 \quad , \quad d = b = 3$$

$$c = 2b - 1 = 5$$

$$n = ec = 7 \times 5 = 35$$

$$n - k = c + m = 3b - 1 = 8$$

$$k = n - (n-k) = 27$$

The computation of $A_c e$ and $A_p c$ are as follows:

$$A_c\ e + A_p\ c = 1 \qquad \text{mod } 35$$

$$7A_c + 5A_p = 1 \qquad \text{mod } 35$$

$$A_c = 3 \text{ and } A_p = 3$$

Thus

$$i = (21\ r_c + 15\ r_p) \qquad \text{mod } 35$$

Lets shorten the (35,27) code to a (24,16) code by making the leading 11 message bits equal to zero. Encoding is not affected. However, decoding must account for the missing 11 leading zeros. Any multiple of $C = 5$ shifts of the $(x^c + 1)$ register has no effect. Any multiple of $e = 7$ shifts of the $p(x)$ register has no effect. For the $(x^c + 1)$ register,

$$(3 + 11) - (5 \times 2) = 4 \text{ shifts}$$

meaning input data should be connected to the input of the 5th stage of the $(x^c + 1)$ register. For the p(x) register,

$$(3 + 11) - (7 \times 2) = 0 \text{ shifts}$$

meaning input data should be connected to the input of the first stage of the shift register. The decoder is shown in figure A-1.

For a message m(x) of

```
                0  123  456  789  10 11 12   13 14 15
  m(x) =        1  011  000  101   1  0  1    0  0  1
```

the encoded V(x) is

```
                0  123  456  789  10 11 12   13 14 15  |  01  234  567
               |1  011  000  101   1  0  1    0  0  1  |  |11  011  101|
                ------------------------------------    |  ----------
                              m(x)                      |     n-k
```

In these examples, bit-0 is the MSB and it is sent first. To illustrate the principle of error correction, lets introduce two error patterns as follows:

Error Pattern No. 1

```
                0  123  ....... 9  10  11  12  13 .... 15  |  01....7
  E1(x) =       0  000  ....... 0 |1   1   1 | 0 ....  0  |  00....0
                                   ----------
                                     i = 10
```

Error Pattern No. 2

```
                0 123  4  56  7  89 ..................  15  |  01....7
  E2(x) =       0 000  0 |11  1 |00 ..................  0   |  00....0
                          -------
                            i = 5
```

The corrections for these two error patterns are shown in figures A-2 and A-3, respectively.

```
03001   REM--THIS PROGRAM COMPUTES X(EXPN) MOD G(X) BY LINEAR SHIFT
03002   REM--REGISTER SEQUENCES.
03010   REM--INITIALIZATION
03020   PRINT "N";
03030   INPUT N
03040   LET I=1
03045   DIM X(50)
03050   FOR J=0 TO 16
03051   X(J)=0
03052   NEXT J
03060   REM--SHIFT RIGHT SUBT.
03065   X(16)=X(15)
03066   X(15)=X(14)
03067   X(14)=X(13)
03068   X(13)=X(12)
03069   X(12)=X(11)
03080   IF X(16)=X(10) GOTO 83
03081   X(11)=1
03082   GOTO 90
03083   X(11)=0
03090   IF X(16)=X(9) GOTO 93
03091   X(10)=1
03092   GOTO 100
03093   X(10)=0
03100   X(9)=X(8)
03110   X(8)=X(7)
03120   X(7)=X(6)
03130   IF X(16)=X(5) GOTO 133
03131   X(6)=1
03132   GOTO 140
03133   X(6)=0
03140   X(5)=X(4)
03150   X(4)=X(3)
03160   X(3)=X(2)
03170   X(2)=X(1)
03180   IF X(16)=X(0) GOTO 183
03181   X(1)=1
03182   GOTO 190
03183   X(1)=0
03190   IF X(16)=I GOTO 193
03191   X(0)=1
03192   GOTO 210
03193   X(0)=0
03200   REM--END OF SHIFT RIGHT SUBT.
03205   LET X(16)=0
03210   LET I=0
03220   LET N=N-1
03230   IF N>=0 GOTO 65
03240   PRINT "X(0), X(1), ....., X(15)"
03241 PRINT X(0);X(1);X(2);X(3);X(4);X(5);X(6);X(7)
03242 PRINT X(8);X(9);X(10);X(11);X(12);X(13);X(14);X(15)
03250   PRINT
03260   GOTO 20
03999   END
```


Figure 1.- A basic program to calculate X^{566} modulo

$$g(x)=X^{16}+X^{11}+X^{10}+X^{6}+X+1.$$

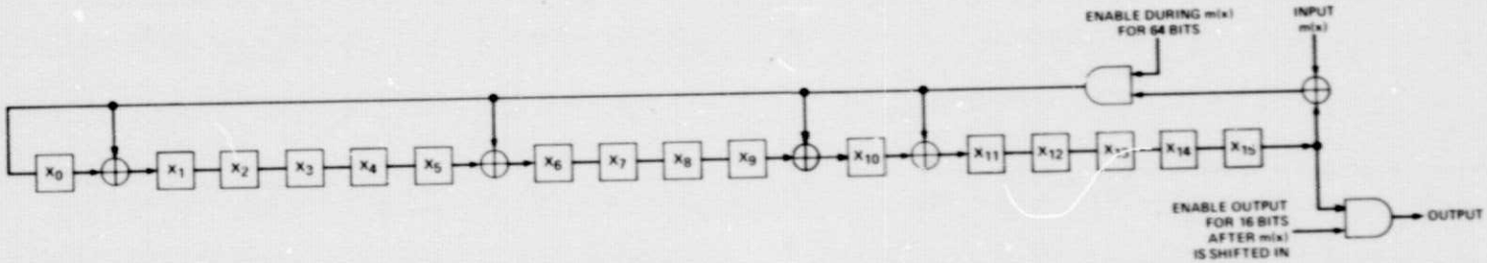


Figure 2.- Encoder for the (80,64) fire code with $g(x)=(X^{16}+X^{11}+X^{10}+X^{6}+X+1)$.

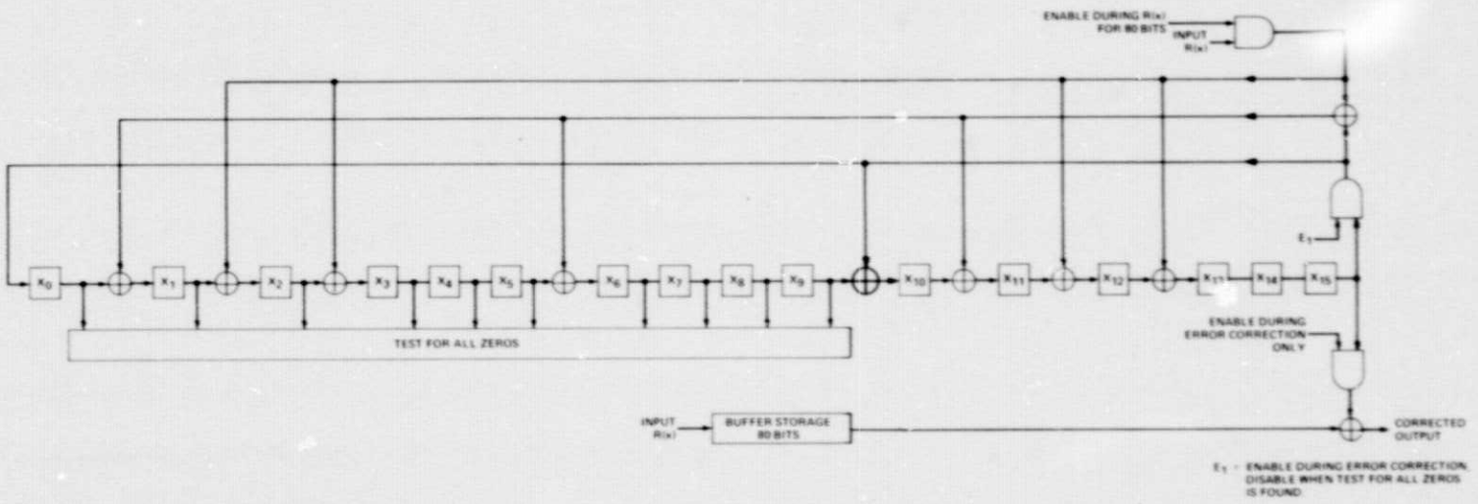


Figure 3.- Decoder for the (80,64) fire code with $g(x)=(X^{16}+X^{11}+X^{10}+X^{6}+X+1)$ and premultiplication $r(y)=(X^{13}+X^{12}+X^{11}+X^{6}+X^{3}+X^{2}+X)$.

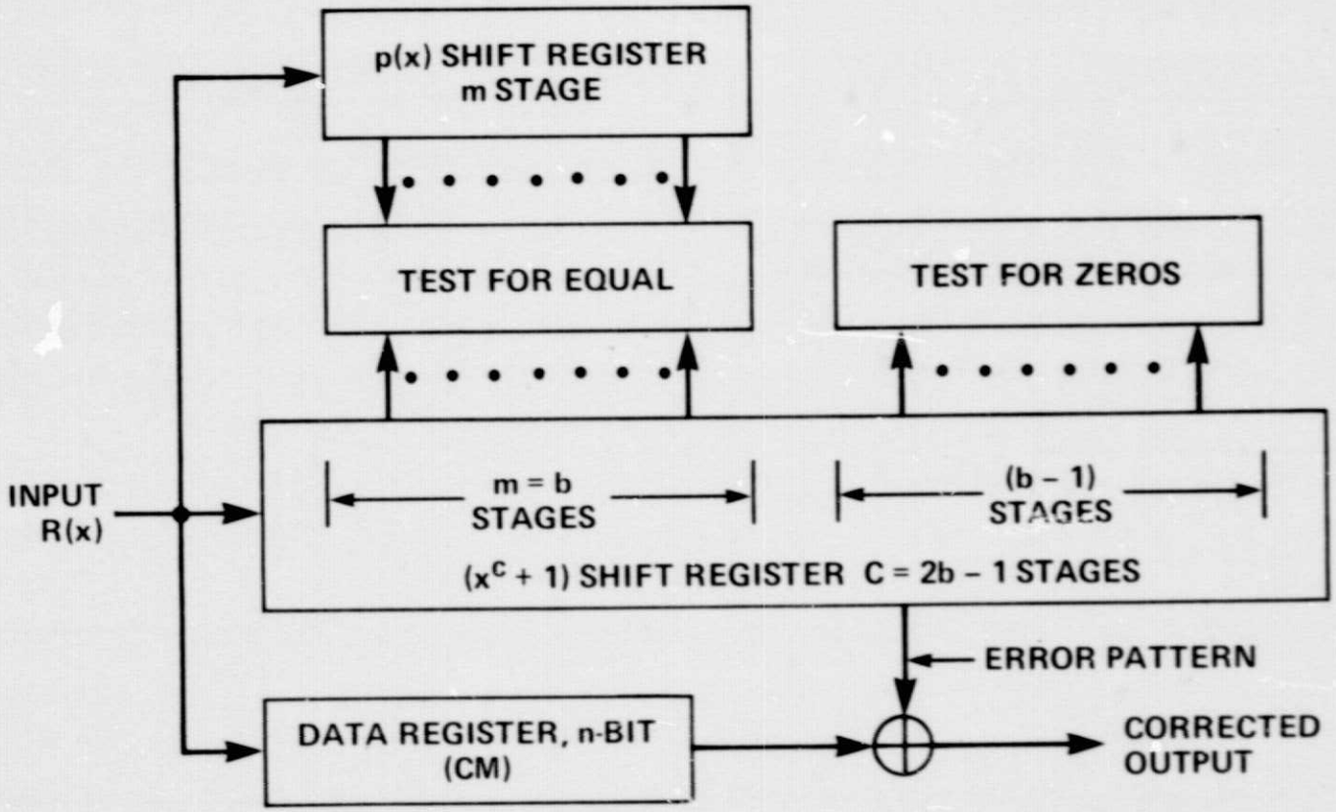


Figure 4.- High-speed decoder for the fire code.

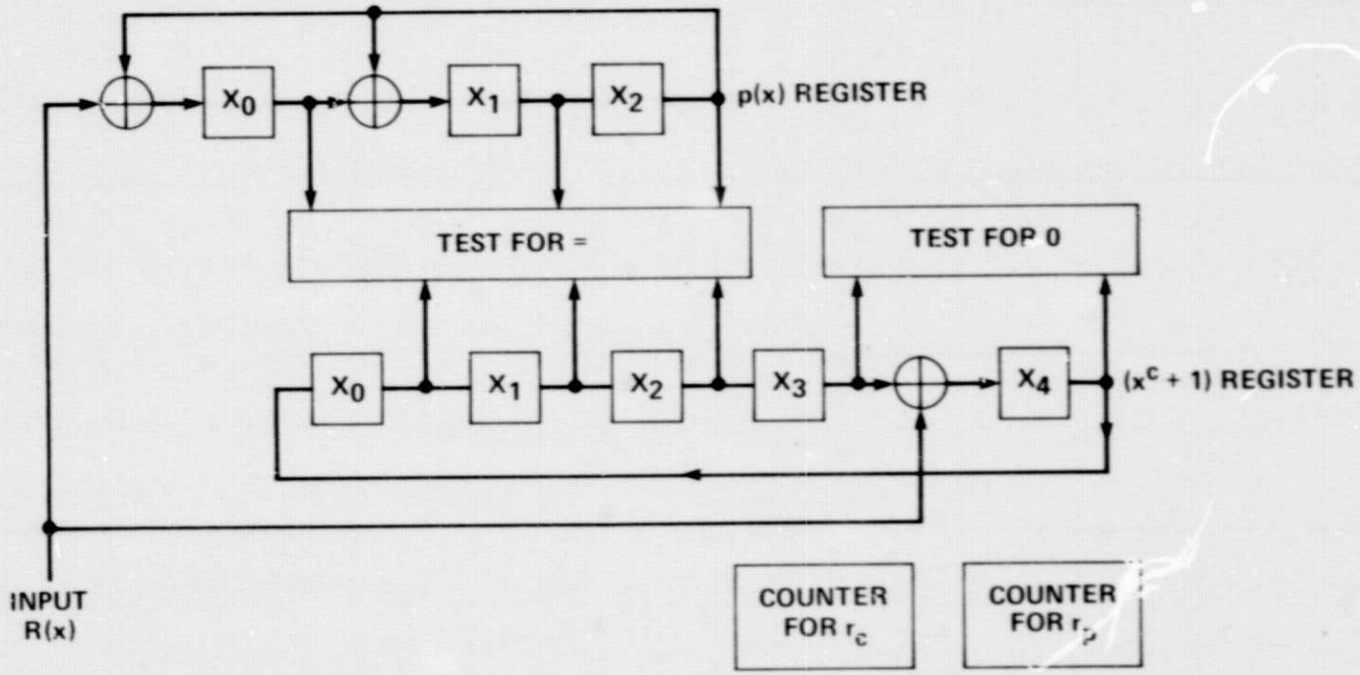
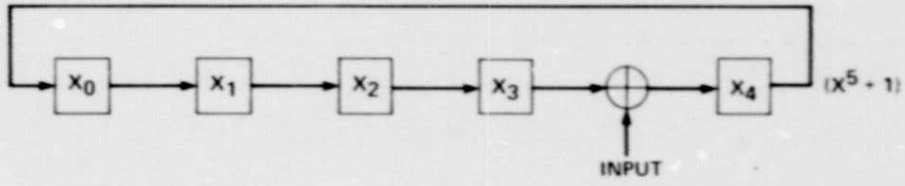


Figure A-1.- High-speed decoder for the (24,16) fire code.

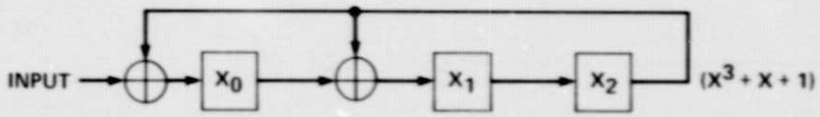


INITIAL		0	0	0	0	0
INPUT	BITS					
0	1	0	0	0	0	1
1	0	1	0	0	0	0
2	1	0	1	0	0	1
3	1	1	0	1	0	1
4	0	1	1	0	1	0
5	0	0	1	1	0	1
6	0	1	0	1	1	0
7	1	0	1	0	1	0
8	0	0	0	1	0	1
9	1	1	0	0	1	1
10	0	1	1	0	0	1
11	1	1	1	1	0	1
12	0	1	1	1	1	0
13	0	0	1	1	1	1
14	0	1	0	1	1	1
15	1	1	1	0	1	0
0	1	0	1	1	0	0
1	1	0	0	1	1	1
2	0	1	0	0	1	1
3	1	1	1	0	0	0
4	1	0	1	1	0	1
5	1	1	0	1	1	1
6	0	1	1	0	1	1
7	1	1	1	1	0	0

ERROR PATTERN

r_c = 0 SHIFT TO OBTAIN ZERO HERE

Figure A-2.1.- (X^C+1) decoding for error pattern No. 1.

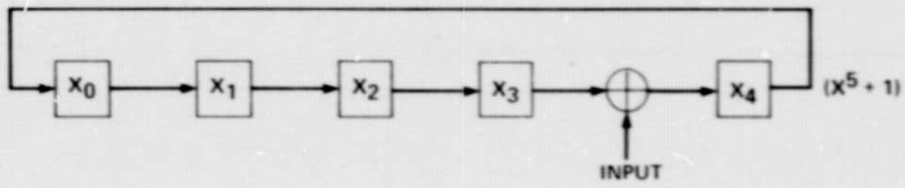


INPUT	BITS				
INITIAL			0	0	0
0	1		1	0	0
1	0		0	1	0
2	1		1	0	1
3	1		0	0	0
4	0		0	0	0
5	0		0	0	0
6	0		0	0	0
7	1		1	0	0
8	0		0	1	0
9	1		1	0	1
10	0		1	0	0
11	1		1	1	0
12	0		0	1	1
13	0		1	1	1
14	0		1	0	1
15	1		0	0	0
0	1		1	0	0
1	1		1	1	0
2	0		0	1	1
3	1		0	1	1
4	1		0	1	1
5	1		0	1	1
6	0		1	1	1
7	1		0	0	1
		r_p SHIFTS			
		1	1	1	0
		2	0	1	1
		3	1	1	1

$r_p = 3$

$i = 21 \times 0 + 15 \times 3 \quad \text{MOD } 35$
$= 10 \quad \text{MOD } 35$

Figure A-2.2.- p(x) decoding for error pattern No. 1.

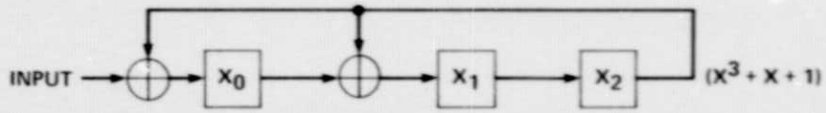


INITIAL INPUT	BITS					
		0	0	0	0	0
0	-	0	0	0	0	1
1	0	1	0	0	0	0
2	1	0	1	0	0	1
3	1	1	0	1	0	1
4	0	1	1	0	1	0
5	1	0	1	1	0	0
6	1	0	0	1	1	1
7	0	1	0	0	1	1
8	0	1	1	0	0	1
9	1	1	1	1	0	1
10	1	1	1	1	1	1
11	0	1	1	1	1	1
12	1	1	1	1	1	0
13	0	0	1	1	1	1
14	0	1	0	1	1	1
15	1	1	1	0	1	0
0	1	0	1	1	0	0
1	1	0	0	1	1	1
2	0	1	0	0	1	1
3	1	1	1	0	0	0
4	1	0	1	1	0	1
5	1	1	0	1	1	1
6	0	1	1	0	1	1
7	1	1	1	1	0	0

ERROR PATTERN

r_c = 0 SHIFT TO OBTAIN ZERO HERE

Figure A-3.1.- (X^c+1) decoding for error pattern No. 2.



INITIAL INPUT	BITS				
		0	0	0	
0	1	1	0	0	
1	0	0	1	0	
2	1	1	0	1	
3	1	0	0	0	
4	0	0	0	0	
5	1	1	0	0	
6	1	1	1	0	
7	0	0	1	1	
8	0	1	1	1	
9	1	0	0	1	
10	1	0	1	0	
11	0	0	0	1	
12	1	0	1	0	
13	0	0	0	1	
14	0	1	1	0	
15	1	1	1	1	
0	1	0	0	1	
1	1	0	1	0	
2	0	0	0	1	
3	1	0	1	0	
4	1	1	0	1	
5	1	0	0	0	
6	0	0	0	0	
7	1	1	0	0	
		r_p SHIFTS			
		1	0	1	0
		2	0	0	1
		3	1	1	0
		4	0	1	1
		5	1	1	1

$r_p = 5$

$i = 21 \times 0 + 15 \times 5 \quad \text{MOD } 35$
$= 5 \quad \text{MOD } 35$

Figure A-3.2.- p(x) decoding for error pattern No. 2.