C. M. Melas

A New Group of Codes for Correction of Dependent Errors in Data Transmission

Abstract: Multiple related errors of any configuration can be automatically corrected by a class of codes having the property of using two groups of parity bits, one defining the error pattern, the other determining the location of the errors within the block.

In particular, error bursts can be corrected with a minimum amount of redundancy. Because each parity-bit group is derived by using maximum-length shift-register sequences, rather than by storing a decoding table, the implementation of these codes is relatively simple, as shown in an example of a three-bit-wide burst-correcting code. An example is given of an application of these codes in a data transmission system where only an even number of bits is likely to be corrupted by a noise burst.

1. Introduction

In transmitting binary data over channels where impulse noise is present, the errors encountered will generally not occur at random. Further, the type of modulation employed tends to produce groups of related errors. In certain phase-modulation systems, for example, a noise impulse of one-bit duration may produce errors in four successive information bits.

N. M. Abramson has described an efficient and easily implemented code for the correction of single and double adjacent errors.[1] Codes for the correction of error bursts have been proposed by Hagelbarger, Gilbert, Fire and others.[2, 3, 4]

The codes described here will efficiently correct any number of multiple-error patterns including those caused by noise bursts. The redundancy can be made as low as desired without greatly affecting the complexity of the encoding and decoding equipment. Because the correctors are derived by using two shift-register sequences, rather than by table lookup, the codes are relatively simple to implement.

We shall first show that a parity-bit group derived from the information digits with the use of a maximal-length shift-register sequence (or m-sequence) generator, as it is in the Abramson code, will locate almost all multiple-error patterns within the information block.[1] Then, a theorem will generalize this result to any m-sequence. To complete the code, a second group of parity bits, also derived with an m-sequence generator, is coupled to the locator group and used in conjunction with the latter to determine the patterns of the multiple errors to be corrected. An example of this code is described in detail and a simple implementation is suggested.

Specific applications of this class of codes are given for a modulation system where even errors only are expected, and codes correcting two and four different types of related errors are described.

2. The locator sequence

The sequence of parity bits used to determine the error location is a maximal-length shift-register sequence defined by Eq. (1) and generated as below.

Consider an R flip-flop shift register in which the state of the first flip-flop at time $t+1$ is dependent on the state of two or more of the other flip-flops at time t. If this dependence is chosen properly, the shift register will follow a sequence of 2^R-1 distinct binary states. Properties of these sequences have been described by several investigators.[5, 6] For example, if R=3, the configuration of Fig. 1 obtains.

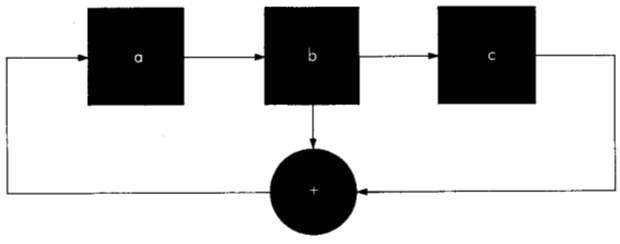


Figure 1 **Three-flip-flop *m*-sequence generator.**

Throughout the paper the symbol + will be used to indicate the exclusive OR *rather than the conventional* OR *function.*

An m-sequence can be generated by flip-flops a, b and c if

$$\begin{aligned} a_{t+1}&=b_t+c_t,\\ b_{t+1}&=a_t,\\ c_{t+1}&=b_t, \end{aligned} \tag{1}$$

and a, b, c are not all 0.

We can rewrite Eq. (1) as:

$$a_{t+3}=b_{t+2}+c_{t+2}, \tag{2}$$

$$b_{t+2}=a_{t+1}, \tag{3}$$

$$c_{t+2}=b_{t+1}=a_t. \tag{4}$$

Substitution of Eqs. (3) and (4) in Eq. (2) yields

$$a_{t+3}=a_{t+1}+a_t. \tag{5}$$

A similar relation holds for b and c flip-flops. Recurrence Eq. (5) is sufficient to define the m-sequence. In general, a sequence of this type of order R can be defined by the following recurrence formula, involving R consecutive terms:

$$a_{t+R}=a_t+k_1a_{t+1}+k_2a_{t+2}+\ldots k_{R-1}a_{t+R-1}, \tag{6}$$

where k_j is a coefficient of value 0 or 1, and a_t, a_{t+1}, a_{t+R} are terms of the sequence. In our example, $k_1=1$, $k_2=0$. To use the m-sequence in an error-correcting code, the parity-bit group, P_1-P_3, is derived from the information group, D_1-D_4, from the following equations:

$$a_1D_1+a_2D_2+\ldots a_5P_1+a_6P_2+a_7P_3=0, \tag{7}$$

$$b_1D_1+b_2D_2+\ldots b_5P_1+b_6P_2+b_7P_3=0, \tag{8}$$

$$c_1D_1+c_2D_2+\ldots c_5P_1+c_6P_2+c_7P_3=0, \tag{9}$$

where a_1-a_7, b_1-b_7, c_1-c_7 are the consecutive states of flip-flops a, b, and c (Table 1).

After transmission of both groups, the left side of Eqs. (7-9) is generated. If no error has occurred during transmission, the right side of all three equations will be 0, since they are all satisfied. A single error occurring in any bit B_t, including a parity bit, will cause the right side of the equations to assume the value of the coefficients a_t, b_t, and c_t, associated with that bit. For example, an error in D_2 will make the three equations respectively 0, 1, 0, as seen in Table 1. Each of seven possible single errors becomes associated with one distinct combination of a_t, b_t, and c_t, uniquely locating its position in the message.

Table 1

a	b	c
1	0	0
0	1	0
1	0	1
1	1	0
1	1	1
0	1	1
0	0	1

Assume now a multiple error has occurred; for example, three successive bits, B_t, B_{t+1}, B_{t+2} are altered by noise. The right side of each equation then assumes the value of the sum of the coefficients associated with those bits. The error locator is then $a_t+a_{t+1}+a_{t+2}$, $b_t+b_{t+1}+b_{t+2}$, $c_t+c_{t+1}+c_{t+2}$, and Eq. (5) can be used to derive a relation with the single-error locators of the sequence chosen. Adding a_{t+2} to both sides of the equation we have:

$$a_{t+3}+a_{t+2}=a_t+a_{t+1}+a_{t+2}. \tag{10}$$

Using Eq. (5) once more:

$$a_{t+3}+a_{t+2}=a_{t+5}. \tag{11}$$

It is apparent from Eqs. (10, 11) that the locator for triple adjacent errors is equal to the locator of a single error occurring 5 bits later. The locators for both types of errors follow the same m-sequence, but are shifted by five terms. It can be shown in this example that every type but one of four-bit-wide error-pattern locators follows the same sequence as the single-error locators, shifted by different amounts. The single exception in our example is a 1101 error pattern, defined as $a_t+a_{t+1}+a_{t+3}$, that always yields a 0 locator as shown by Eq. (5). These properties are generalized in Theorems 1 and 2.

Theorem 1 sets an upper limit to the locator sequence capabilities when used for error-burst correction. A locator sequence derived from an R flip-flop shift register can locate all errors caused by a noise burst R bits wide, and can locate all but one error if the noise burst is $R+1$ bits wide.

Theorem 2 defines a property of m-sequences that makes them very useful in nonindependent error-correcting codes. Once the coefficient $m_1 \ldots m_R$ defining the error type are known, the location of this error within 2^R-1 bits is determined by the same sequence for any $m_1 \ldots m_R$. The locator sequence defined in Table 1, for example, can be used as an error-correcting code for any one predetermined type of error (with the exception noted in Theorem 1) with no change in its structure. Its Hamming distance[7] is three.

• *Theorem 1*

Let

$$a_{t+R}=a_t+\sum_{1}^{R-1} c_ja_{t+j} \tag{12}$$

be a maximal-length shift-register sequence with terms $a_t \ldots a_{t+2^R-1}$. Consider any sequence

$$S_t=a_t+m_1a_{t+1} \ldots m_{R-1}a_{t+R-1}+a_{t+R}, \tag{13}$$

in which the m_i's are arbitrary but $m=0$ or $m=1$.

There is one and only one choice of m_i's for which $S_t\equiv 0$ for all values of t.

Proof

Adding Eqs. (12) and (13) we obtain

$$S_t+a_{t+R}=a_t+a_t+(c_1+m_1)a_{t+1}+\ldots +(c_{R-1}+m_{R-1})a_{t+R-1}+a_{t+R}, \quad (14)$$

$$S_t=(c_1+m_1)a_{t+1}+\ldots+(c_{R-1}+m_{R-1})a_{t+R-1}. \quad (15)$$

If $m_i=c_i$, then $S_t\equiv 0$. This is the only set of m_i's for which S_t vanishes for all values of t, since $a_{t+1}\ldots a_{t+R-1}$ are all independent and all their coefficients must be 0 if S_t is to vanish identically.

• *Theorem 2*

Let $a_{t+R}=a_t+\sum_1^{R-1} c_j a_{t+j}$ be a maximal-length shift-register sequence (m-sequence). Consider any sequence

$$S_t=a_t+m_1a_{t+1}+\ldots m_{R-1}a_{t+R-1}+m_ra_{t+R}, \quad (16)$$

in which all terms are defined as in Theorem 1.

Each sequence S_t is actually the m-sequence [Eq. (12)] shifted by an amount k with respect to the sequence of a_t, where k is different for each set of m_i's. Hence,

$$S_t=a_{t+k}.$$

There is a one-to-one correspondence between each set of m_i's and each k.

Proof

The proof will consist in showing that any sum S_t is a term a_{t+k} of the sequence, and that no two sums correspond to the same term a_{t+k}.

(**a**) Without loss of generality, Eq. (16) can be rewritten as

$$S_t=a_{t+R}+m_1a_{t+R+1}+m_2a_{t+R+2}+\ldots m_Ra_{t+2R}. \quad (17)$$

Each term of this sum belongs to the m-sequence defined by Eq. (12). We can write for each term:

$$a_{t+R}=a_t+c_1a_{t+1}+\ldots c_{R-1}a_{t+R-1}, \quad (18)$$

$$a_{t+R+1}=a_{t+1}+c_1a_{t+2}+\ldots c_{R-1}a_{t+R}, \quad (19)$$

$$a_{t+R+2}=a_{t+2}+c_1a_{t+3}+\ldots c_{R-1}a_{t+R+1}, \quad (20)$$

$$a_{t+R+3}=a_{t+3}+c_1a_{t+4}+\ldots c_{R-1}a_{t+R+2}, \quad (21)$$

$$a_{t+2R}=a_{t+R}+c_1a_{t+R+1}\ldots c_{R-1}a_{t+2R-1}. \quad (22)$$

Substituting Eqs. (18-22) in Eq. (17):

$$S_t=a_t+a_{t+1}(m_1+c_1)+a_{t+2}(c_1+m_1c_1+m_2)+a_{t+3}(c_3+m_1c_2+m_2c_1+m_3)+\ldots+a_{t+R}(m_1c_{R-1}+m_2c_{R-2}+\ldots m_R)+S_{R-1}m_Ra_{t+2R-1}. \quad (23)$$

For S_t to be a term of the same m-sequence as a_j, it is necessary and sufficient that it satisfy Eq. (12) for the same set of c_j coefficients.

Let S_t, $S_{t+1}\ldots S_{t+R}$ be the values taken by S_t for $R+1$ consecutive sums with fixed values of $m_1\ldots m_R$, defined in Eqs. (24-26). Thus

$$S_t=a_t+m_1a_{t+1}+m_2a_{t+2}+\ldots m_Ra_{t+R}, \quad (24)$$

$$S_{t+1}=a_{t+1}+m_1a_{t+2}+ma_{t+3}+\ldots m_Ra_{t+R+1}, \quad (25)$$

$$S_{t+R}=a_{t+R}+m_1a_{t+R+1}+\ldots m_Ra_{t+2R}. \quad (26)$$

Then S_t belongs to the m-sequence with coefficients $c_1\ldots c_{R-1}$, if and only if

$$S_{t+R}=S_t+c_1S_{t+1}+c_2S_{t+2}+\ldots c_{R-1}S_{t+R-1}. \quad (27)$$

Substitution of Eqs. (24-26) in Eq. (27) yields a polynomial identical to Eq. (23), proving that S_t is a member of the same m-sequence as its terms.

(**b**) No two sums S_t derived from different values of $m_1\ldots m_R$, but with a common first term a_t, map into the same term a_{t+k} of the m-sequence.

Let us assume the opposite is true. Then we can write, for two different sets m'_i and m''_i.

$$S_1=a_t+m'_1a_{t+1}+m'_2a_{t+2}\ldots m'_Ra_{t+R}=a_{t+k}, \quad (28)$$

$$S_2=a_t+m''_1a_{t+1}+m''_2a_{t+2}+\ldots m''_Ra_{t+R}=a_{t+k}. \quad (29)$$

Adding Eqs. (28) and (29):

$$a_{t+1}(m'_1+m''_1)+a_{t+2}(m'_2+m''_2)\ldots+\ldots a_{t+R}(m'_R+m''_R)=0. \quad (30)$$

The right-hand term of Eq. (30) is 0 for all values of t, and, since $a_{t+1}a\ldots a_{t+R}$ are independent, $m'_i=m''_i$.

2. Error pattern determination. Multiple error-correcting codes

Theorem 2 has shown how each multiple error shifted the m-sequence of locators with respect to the single error.

An additional parity-bit group derived from a different m-sequence can be used for error pattern determination. If every pattern to be corrected shifts the two sequences by a different amount, the relative shifts become associated with that pattern and are independent of the location of the error in the block.

An example of this code is given here using the sequence $a_t+a_{t+1}=a_{t+4}$ of period 15 as a locator, and the sequence $\alpha_t+\alpha_{t+1}=\alpha_{t+2}$ of period 3 for the correction of the four error patterns spanning a maximum of three bits. The corrector sequences are given in Table 2. The relative sequence shifts caused by the 11 and 101 errors can be visualized in that table. The corrector-word 1001 01 occurs for a single error in bit 1. Double-adjacent errors shift II sequences by 2 positions, and the locator 1001 now occurs two bit times earlier than the type-word 01. All 101 errors correspond to a relative shift of 1 position and a triple error results in a 00 type-word for all locators.

These relative shifts can be derived directly from the characteristic equations. For 11 errors

$$\alpha_t+\alpha_{t+1}=\alpha_{t+2} \quad (31)$$

$$a_t+a_{t+1}=a_{t+4}. \quad (32)$$

} Relative shift $a-\alpha=2$ positions

Table 2 **Corrector sequences for three-bit-wide error-correcting code.**

Error starting in bit	1						11						101						111					
	P_1	P_2	P_3	P_4	Π_1	Π_2	P_1	P_2	P_3	P_4	Π_1	Π_2	P_1	P_2	P_3	P_4	Π_1	Π_2	P_1	P_2	P_3	P_4	Π_1	Π_2
1	1	0	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	0	0	0	1	1	0	0
2	1	1	0	0	1	0	1	0	1	0	0	1	0	1	1	1	1	1	0	0	0	1	0	0
3	0	1	1	0	1	1	1	1	0	1	1	0	0	0	1	1	0	1	1	0	0	0	0	0
4	1	0	1	1	0	1	1	1	1	0	1	1	0	0	0	1	1	0	0	1	0	0	0	0
5	0	1	0	1	1	0	1	1	1	1	0	1	1	0	0	0	1	1	0	0	1	0	0	0
6	1	0	1	0	1	1	0	1	1	1	1	0	0	1	0	0	0	1	1	0	0	1	0	0
7	1	1	0	1	0	1	0	0	1	1	1	1	0	0	1	0	1	0	1	1	0	0	0	0
8	1	1	1	0	1	0	0	0	0	1	0	1	1	0	0	1	1	1	0	1	1	0	0	0
9	1	1	1	1	1	1	1	0	0	0	1	0	1	1	0	0	0	1	1	0	1	0	0	0
10	0	1	1	1	0	1	0	1	0	0	1	1	0	1	1	0	1	0	1	1	0	1	0	0
11	0	0	1	1	1	0	0	0	1	0	0	1	1	0	1	1	1	1	1	0	1	0	0	0
12	0	0	0	1	1	1	1	0	0	1	1	0	0	1	0	1	0	1	1	1	0	1	0	0
13	1	0	0	0	0	1	1	1	0	0	1	1	1	0	1	0	1	0	1	1	1	0	0	0
14	0	1	0	0	1	0	0	1	1	0	0	1	1	1	0	1	1	1	1	1	1	1	0	0
15	0	0	1	0	1	1	1	0	1	1	1	0	1	1	1	0	0	1	0	1	1	1	0	0

For 101 errors,

$$\alpha_t+\alpha_{t+2}=\alpha_{t+1}=\alpha_{t+7} \quad (33)$$

$$a_t+a_{t+2}=a_{t+8}. \quad (34)$$

} Relative shift $a-\alpha=1$ position

For 111 errors,

$$\alpha_t+\alpha_{t+1}+\alpha_{t+2}=0. \quad (35)$$

Another group of errors checked by this code is the 1-11-101-1001 groups, composed of one single, and three types of double errors.

Table 3 shows the parity check table for this code. Location of parity checks Π_1 and Π_2 was chosen to provide coherent cross-checking of parity bits.

Table 3 **Parity check table for three-bit-wide error-correcting code.**

	D_1	D_2	D_3	D_4	D_5	D_6	Π_2	D_7	D_8	D_9	Π_1	P_4	P_1	P_2	P_3
P_1	×	×		×		×	×	×	×				×		
P_2		×	×		×		×	×	×	×				×	
P_3			×	×		×		×	×	×	×				×
P_4				×	×		×		×	×	×	×			
Π_1		×	×		×	×		×	×		×	×		×	×
Π_2	×		×	×		×	×		×	×		×	×		×

3. Implementation of the codes

Figure 2 shows how the code can be encoded with relatively simple logical circuits. The state of the *m*-sequence flip-flops is shown at bit 1. As the incoming serial information is advanced into the shift register, the *m*-sequence flip-flops gate the proper information bits to the modulo 2 parity flip-flops. Between bits 6 and 7, an additional clock pulse is delivered to the locator *m*-sequence flip-flops to advance their state past the position allotted to the Π_2 bit. No information is accepted between bit times 9 and 15, while Π_1 and Π_2 are determined and their contents are added to the proper *p* flip-flops. Here Π_1 is added to P_3 and P_4, and Π_2 to P_1, P_2 and P_4.

The contents of parity flip-flops P_{1-4} and Π_{1-2} are shifted out with the nine information digits.

For the decoding operation, essentially the same elements are used (see Fig. 3). As the data are introduced in the data register, the parity checks are taken according to a procedure similar to the encoding. If all parity digits are 0, the data are correct and can be shifted directly out of the register. If an error exists, the following correction procedure is used.

(1) The initial conditions are those shown in Fig. 3, wherein the data are shifted in the data register, and both locator and error-type sequence generators are shifted synchronously until comparison is detected between the contents of the *P* flip-flops and the locator word.

(2) When comparison is indicated, the contents of the error-type sequence generator are compared with the Π flip-flops. If they match, a single error is detected, and the bit in error is in the last position of the data register. Correction is achieved by adding modulo 2, the contents of the corrector register to the data register.

(3) If the error-type sequence generator is 00, the corrector register is cleared to 111, and the locator generator to 0011. Step 1 is repeated, and the triple error corrected by adding the corrector register to the data register.

(4) If the error-type register does not match the contents of the II flip-flops, or is not 00, a 11 or 101 error has occurred. The sequence generator is then shifted (either once or twice) until it matches the II flip-flops. Simultaneously, the locator sequence generator is set to 1111 after one shift, or 0101 after two shifts, and the error-corrector register is set to 101 or 011. When the type has been determined, Step 1 is repeated and the error corrected as before. In all cases the bits in error will occupy the last positions of the data register.

The total correction process takes, at most, 30 bit times, and can easily be accomplished without additional buffering for data transmission at audio frequency rates, if a high-frequency clock is used for the process. Use of locator sequences of order higher than $R=4$ results in a corresponding decrease in redundancy with a negligible increase in the complexity of the instrumentation.

This implementation varies little as the message length increases. The only change would consist in increasing the data register and the locator sequence generator.

Table 4 lists the number of information and parity bits for some higher block lengths, as well as recurrence equations for the locator sequences to be used.

Table 4 **Some three-bit-wide error-correcting codes.**

Percent Redundancy	*Info. Bits*	*Parity Bits*	*Locator Sequence*
40%	9	6	$a_{t+4}=a_t+a_{t+1}$
12.7%	55	8	$a_{t+6}=a_t+a_{t+1}$
4%	245	10	$a_{t+8}=a_t+a_{t+1}+a_{t+6}+a_{t+7}$
1.2%	1011	12	$a_{t+10}=a_t+a_{t+3}+a_{t+8}$

4. Code parity-bit efficiency

The parity-bit efficiency can be defined as the ratio of the number of combinations of parity bits used for error correction, to the total number of possible combinations. Six parity bits or 64 combinations are available in the code just described. Of these combinations, 60 are used in 15 bits, one to indicate an error-free message.

The efficiency is therefore $\frac{60+1}{64}=0.95$.

In general, for a code $2^{k_1}-1$ composed of two groups of K_1 and K_2 parity bits, where the period $2^{K_1}-1$ of one group is a multiple of the period $2^{K_2}-1$ of the

Figure 2 **Basic elements of encoder.**

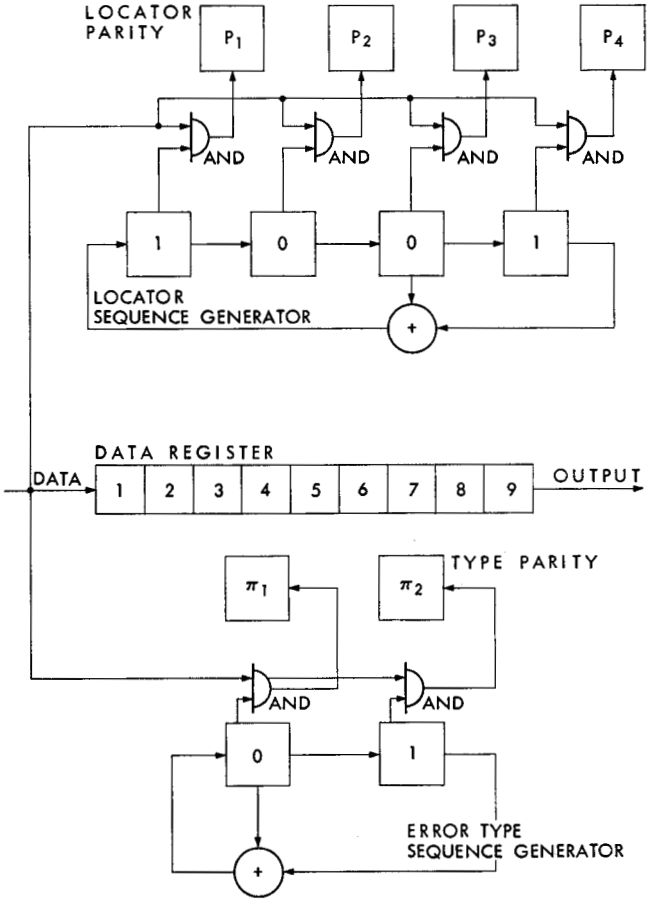
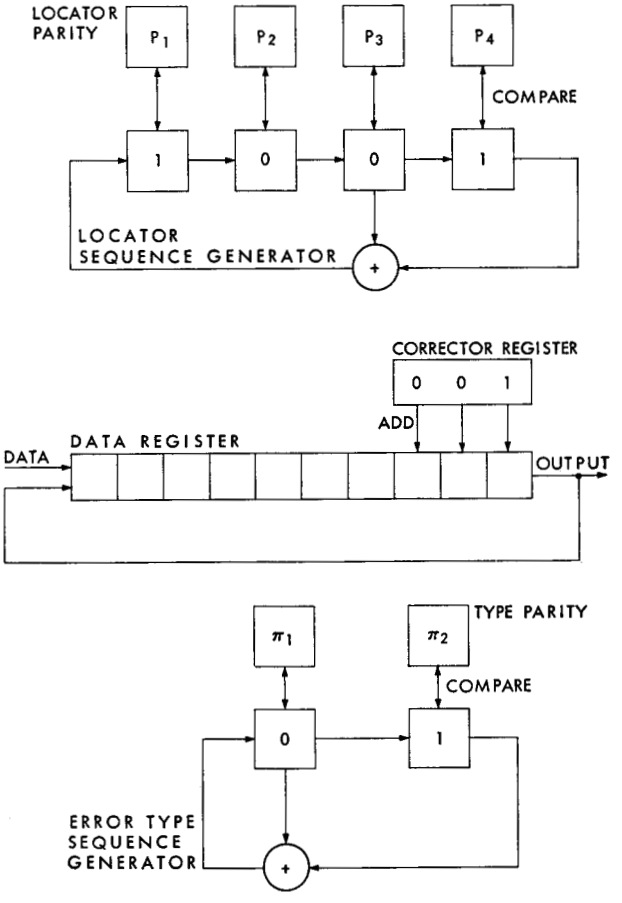


Figure 3 **Basic elements of decoder.**



other, the efficiency can be expressed as

$$E=\frac{2^{K_2}(2^{K_1}-1)+1}{2^{K_1+K_2}}=1+\frac{1}{2^{K_1+K_2}}-\frac{1}{2^{K_1}}. \qquad (36)$$

If $K_1 \gg K_2$, the efficiency approaches unity. It is better than 0.95 for all practical codes.

Let us now examine the case where the two sequences do not have periods in a submultiple relation. To each multiple error must now be associated two distinct relative shifts, depending upon the location of the error in the block.

The efficiency is thus reduced by half, since one more parity bit is required for the error-type determination. It approaches 0.5.

5. Codes for correcting error bursts greater than three bits in duration

An error burst covering K bits can result in $2^K/2$ different error patterns, requiring a minimum of $K-1$ bits in the error-type group.

While no specific codes for the correction of error bursts of more than three bits in length are given here, they can be readily constructed by determining the two m-sequences to be matched for the desired block length and error-burst width.

6. Nonindependent errors in binary phase modulation systems

The interference in many signaling media does not always take the form of random disturbance (white noise), but may consist of noise bursts which can last for one or more bit times, depending on the speed of transmission. The types of errors caused by these bursts depend on the modulation method used to convert binary data to a form suitable for transmission. In amplitude modulation, two discrete levels of the carrier are produced, one for 0's and the other for 1's. Similarly, a frequency-modulated carrier will consist of one discrete frequency for 0's, and another for 1's. In phase modulation, however, where the carrier is reversed by 180° at each binary transition, it is difficult to identify one phase of the carrier with a 0 and the other with a 1, since no absolute phase reference exists at the receiver. Rather, a phase reversal of the carrier can be defined as a 1, and the absence of phase reversal as a 0. Since a phase reversal corresponds to a binary transition of the transmitted data, the phase-modulated carrier can be said to transmit binary data coded in NRZI form, whereas amplitude or frequency modulation handles data in NRZ form. Figure 4a shows an example of NRZ and NRZI coding.

Figure 4 **Single-bit-wide error in NRZ and NRZI codes.**

A noise pulse that would cause a single error in NRZ data causes a double adjacent error if NRZI code is used, as seen in Fig. 4b. The NRZI code can be derived from the NRZ code from the consideration that an NRZI bit is 1 if the corresponding NRZ bit and the preceding NRZ bit are different, and 0 if they are the same. Let D_t and B_t be the NRZ and NRZI bits corresponding to the same waveform; then

$$B_t=D_t+D_{t-1} \qquad (37)$$

$$B_{t+1}=D_t+D_{t+1}. \qquad (38)$$

An error in D_t will cause an error in both B_t and B_{t+1}. Similarly adjacent errors in D_t and D_{t+1} will affect B_t and B_{t+2}.

Table 5
Parity check table for 11-101 error-correcting code.

	D_1	D_2	D_3	P_1	P_2	P_3	Π_1
P_1	×	×		×			
P_2		×	×		×		
P_3			×	×		×	
Π_1	×		×		×		×

A code that corrects double adjacent and 101 errors is therefore the equivalent in NRZI transmission, to a code that corrects single and double adjacent errors, as proposed by Abramson[1] or described in Appendix I. The locator sequence used in this code can be any m-sequence according to the desired block length. The error-type check bit is taken across every other digit, and is always 1 for 11 errors and 0 for 101 errors.

Table 5 shows an example of this code for a 7-bit block. Parity check Π_1 is always 1 for double adjacent errors, and always 0 for 101 errors. The corrector words are listed in Table 6.

Table 6 **Corrector sequence for 11-101 error-correcting code.**

Error in	P_1	P_2	P_3	Π_1	*Error in*	P_1	P_2	P_3	Π_1
D_1 D_2	0	1	0	1	D_1 D_3	1	1	1	0
D_2 D_3	1	0	1	1	D_2 D_4	0	1	1	0
D_3 P_1	1	1	0	1	D_3 P_2	0	0	1	0
P_1 P_2	1	1	1	1	P_1 P_3	1	0	0	0
P_2 P_3	0	1	1	1	P_2 Π_1	0	1	0	0
P_3 Π_1	0	0	1	1					

Note that this code is cyclical, i.e., an error occurring in the first and last bits of the word is not corrected as a double adjacent error. However, a double error occurring in the last bit of one block and the first bit of the next can be corrected since the corresponding corrector words will take the values 0001 and 1001, not encountered in the previous table. In general, a parity check across the complete block will detect the occurrence of any odd-order error.

If the noise burst covers three bits, the following error patterns are expected in a phase-modulation system: 11, 101, 1111 and 1001. That error group can also be corrected by the code described earlier (Table 2). The correctors for 1111 errors are identical to the single error correctors, and the 1001 pattern generates the same correctors as 111 errors.

Error detection

From Theorem 1 it is apparent that R parity bits derived from an m-sequence will detect all error bursts up to R bits in length, since the first multiple error to set all parity bits to 0 is of order $R+1$. For example, a six-bit code with two parity bits will detect single, double adjacent, and 101 errors, or all errors caused by an error burst two-bits wide, whether NRZ or NRZI coding is used.

The simplicity with which the parity bits can be generated makes this method attractive where efficient error detection for transmission systems is required.

Although no exhaustive study of the error-correcting capabilities of codes involving m-sequence correctors has been made in this article, it is believed that the versatility of these codes in handling different types of error patterns, as well as the possibility of their simple implementation has been demonstrated.

Appendix: A code for the correction of single and double adjacent errors.

An inverse sequence of correctors generated from $R+1$ flip-flops and of period 2^R-1 can replace the combination of a maximal-length inverse-sequence of correctors generated from R flip-flops and one all-check parity bit, for correction of single and double adjacent errors. Consider a shift-register sequence defined in terms of the states of $R+1$ flip-flops $p_1 \ldots p_{R+1}$ at times t and $t+1$:

$$(p_1)_{t+1}=c_1(p_2)_t+c_2(p_3)_t+\ldots c_{R-1}(p_R)_t+1, \quad (39)$$

$$(p_2)_{t+1}=(p_1)_t, \quad (40)$$

$$(p_3)_{t+1}=(p_2)_t, \quad (41)$$

$$(p_{R+1})_{t+1}=(p_R)_t. \quad (42)$$

The coefficients $c_1 \ldots c_{R-1}$ determine an inverse maximal length sequence of length 2^{R-1}, generated by flip-flops $p_1 \ldots p_R$; flip-flop p_{R+1} is defined by Eq. (42). Substitution of Eqs. (40-42) in Eq. (39) yields

$$(p_1)_{t+1}=c_1(p_3)_{t+1}+c_2(p_4)_{t+1} +c_3(p_5)_{t+1}+\ldots c_{R-1}(p_{R+1})_{t+1}+1. \quad (43)$$

Equation (43) is equivalent to

$$(p_1)_t=c_1(p_3)_t+c_2(p_4)_t+c_3(p_5)_t +\ldots c_{R-1}(p_{R+1})_t+1, \quad (44)$$

which expresses the relation between the flip-flop states at time t.

Since $(p_1)_t$, $(p_2)_t$, $(p_3)_t \ldots (p_{R+1})_t$ is the corrector word for a single error occurring at time t, Eq. (44) defines the following relation between the bits of this word for which $c_j=1$:

$$(p_1)_t+c_1(p_3)_t+c_2(p_4)_t+\ldots c_{R-1}(p_{R+1})_t=1. \quad (45)$$

The corrector word for a double adjacent error is obtained by adding modulo 2, the corrector word for single errors occurring at times t and $t+1$.

$$\{(p_1)_t+(p_1)_{t+1}\} \quad \{(p_2)_t+(p_2)_{t+1}\} \quad \{(p_3)_t+(p_3)_{t+1}\} \ldots \quad \{(p_{R+1})_t+(p_{R+1})_{t+1}\}. \quad (46)$$

Modulo 2 addition of Eqs. (43) and (44) defines the following relation between the bits of this corrector for which $c_j=1$.

$$\{(p_1)_t+(p_1)_{t+1}\}+c_1\{(p_3)_t+(p_3)_{t+1}\} +c_2\{(p_4)_t+(p_4)_{t+1}\}+\ldots+ c_{R-1}\{(p_{t+1})_t+(p_{R+1})_{t+1}\}=0. \quad (47)$$

Comparison of Eqs. (45) and (47) indicates that the corrector words for single errors will always be different from those for double adjacent errors.

Example

In Table 1,

$$(p_1)_{t+1}=(p_2)_t+(p_3)_t.$$

Where p_1, p_2, p_3 are the correctors derived from P_1, P_2, P_3, the Π_1 all-check bit can be replaced by a bit P_4 such that the corrector derived from (P_4) is

$$(p_4)_{t+1}=(p_3)_t. \quad (48)$$

The parity check table is shown in Table 7, and the corrector sequence is given in Table 8.

For all single errors,

$$p_1+p_3+p_4=1. \quad (49)$$

For all double adjacent errors,

$$p_1+p_3+p_4=0. \quad (50)$$

Table 7
Parity check table for new SEC-DAEC code.

	D_1	D_2	D_3	P_1	P_2	P_3	P_4
P_1	×	×		×			
P_2		×	×		×		
P_3			×	×		×	
P_4				×	×		×

Table 8 **Single and double adjacent corrector sequences.**

Single Errors	p_1	p_2	p_3	p_4	*Double adjacent Errors*	p_1	p_2	p_3	p_4
D_1	1	0	0	0	D_1D_2	0	1	0	0
D_2	1	1	0	0	D_2D_3	1	0	1	0
D_3	0	1	1	0	D_3P_1	1	1	0	1
P_1	1	0	1	1	P_1P_2	1	1	0	0
P_2	0	1	0	1	P_2P_3	0	1	1	1
P_3	0	0	1	0	P_3P_4	0	0	1	1
P_4	0	0	0	1	P_4D_1	1	0	0	1

Revised manuscript received October 28, 1959

References

1. N. M. Abramson, "A Class of Systematic Codes for Non-Independent Errors," Technical Report No. 51, Dec. 30, 1958, Stanford Electronics Laboratories, Stanford, Calif.
2. D. W. Hagelbarger, "Recurrent Codes: Easily Mechanized Burst Correcting Binary Codes," *Bell System Tech. J.*, 69 (July 1959).
3. E. N. Gilbert, "A Problem in Binary Encoding," paper presented at AMS Meeting, Apr. 29, 1958.
4. P. Fire, "A Class of Multiple-Error-Correcting Binary Codes for Non-Independent Errors," RSL-E-2, March 1959, Sylvania Reconnaissance Systems Laboratory, Mountain View, Calif.
5. N. Zierler, "Several Binary-Sequence Generators," Technical Report No. 95, Lincoln Laboratory, M.I.T., Cambridge, Mass.
6. B. Elspas, "Theory of Autonomous Linear Sequential Networks," IRE PGCT, March 1959.
7. R. W. Hamming, "Error Correcting and Error Detecting Codes," *Bell System Tech. J.*, **29**, 147, 1950.