Some communication and data processing systems use codes to detect and correct errors that occur during transmission or processing of information. The codes introduce redundancy; however, this is offset by increased reliability in recovering information. Fire codes, which provide protection against errors in bursts, are specifically discussed in this paper. Emphasis is placed on generating Fire codes using a computer program, selecting a code to satisfy a given system requirement, and implementing codes using existing techniques and hardware.

ERROR DETECTION AND CORRECTION CODES

P. T. Komiske

The advent of digital transmission systems which transfer quantized information rather than information in analog form has enhanced the importance of error-detecting and error-correcting codes. Major advances in the construction of these codes have been made in the last several years, with the objective of providing greater reliability in the recovery of intelligence in digital transmission systems.

The design of many of these codes is based on a mathematical model which essentially establishes rules for the incorporation of redundant information. This redundancy can range from the minimum necessary to provide error detection only, to a maximum that can yield almost any desired level of error correction.

Error-detection and -correction codes have wide application in the fields of information-processing and communications. In the first of these, special error circuits are being designed into information-processing systems. Although basic error detection circuits predominate, error correction is also provided in special military and commercial applications. In the field of communications, a variety of schemes for error detection and correction have been explored. These range from (1) feedback systems that re-transmit a complete set of original information, either automatically or on demand, to (2) codes that provide enough redundancy to detect and correct any errors introduced during transmission, but without feedback in the system. A typical system might be the communications link between a satellite and a ground station, where detection and correction are not possible unless a code is used.

To illustrate the feedback system, we might consider three sequential transmissions of the following information: "Start work at eight-thirty." A majority decision, e.g. two out of three, could be used to determine the correct letter. Instead of transmitting the information in the above sequence, and then repeating if necessary, the information could also be sent as follows: "Start Start Start work work work at at at eight-thirty eight-thirty eight-thirty." A decision to repeat each word would then be based on whether the word was understood at the time of receipt. This can be considered to be a way of coding information if the option of retransmission is omitted; the information is simply sent automatically a number of times. However, the information coding in this case does not permit reconstruction of the information under conditions of unusual interference.

Redundant codes have been explored for possible application in communication systems subjected to environments that cause sequential errors (bursts). Since most present communication and information-processing systems operate in the binary number system, error bursts consist of sets of consecutive symbols (ones or zeros) that are the inverses of the original values. Mr. P. Fire of Stanford Electronics Laboratories developed a type of code to cope with errors that occur in bursts.[1] The attractive feature of the Fire code is that it is easily implemented with known shift-register generator theory and techniques.

In this paper we consider three aspects—generation, selection, and implementation—of the burst-error-detecting and -correcting Fire code. Based on the mathematical structure and constraints governing the generation of Fire codes, a set of tables has been computed to permit convenient selection of the appropriate code for any system application within the limits of the tables (detection of up to 25 symbol error bursts). Selection of the best code, normally a difficult procedure, is much simplified with these tables.

Generation of a Table of Fire Codes

The Fire codes are mathematically grounded in ring theory. Peterson gives the mathematical development in detail.[2]

It can be shown that information can be represented by the coefficients of a polynomial. In the binary system, the coefficients are either ones or zeros. Information may then be considered to be the components of a vector in an n-dimensional space, where n is the length of a code vector in bits.

The Fire codes are best described by a generating polynomial

$$g(X) = p(X)(X^c - 1),$$

in which the two factors are relatively prime and $p(X)$ is of degree m and irreducible, i.e. not divisible by any polynomial of degree greater than zero but less than m.

In operation, the codes are capable of *detecting* any combination of two error bursts in which the length of the shorter burst is not greater than m bits and the sum of the burst lengths is no greater than $c + 1$ bits, or a single burst of length d bits which is not greater than $c + m$ (the number of check symbols). A single error burst of length b can be *corrected*, providing that $c \geq b + d - 1$ and $m \geq b$.

The length of the code vector n is given by the least common multiple of e and c, where e is the order of the roots of the irreducible polynomial $p(X)$. Further, c must not be divisible by e or the length n will not be maximal and the efficiency of the code will be reduced.

To shorten the code length n it is only necessary to set the first symbols transmitted equal to zero (by the appropriate feedback connections in the shift-register). This gives a *shortened* Fire code that is easily implemented.

The governing mathematical constraints are:

$$d \geq b, \tag{1}$$

$$m \geq b, \tag{2}$$

$$c \geq b + d - 1, \tag{3}$$

$$\frac{c}{e} \neq I, \text{ where } I \text{ is an integer,} \tag{4}$$

and

$$c + m = \text{degree of the generator polynomial } g(X). \tag{5}$$

To completely define the code, the following quantities must be computed for each code falling within the constraints defined above.

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

$$n = e \times c, \tag{7}$$

and

$$k = n - (c + m), \text{ where } k = \text{number of information symbols.} \tag{8}$$

An additional quantity of interest is the transmission efficiency of the code, defined here as k/n.

A computer program has been developed to generate all codes defined by the boundary conditions imposed by Eqs. (1) through (5), and to compute the quantities defined by Eqs. (6) through (8).

Examples of printouts of the codes generated through generator degree 25, with a minimum burst correction capability of 2, are shown in Tables I and II.[3]

Selection Criteria and Rules

The tables facilitate either selection of a particular code or comparative evaluation of several codes, since they are listed in increasing order of the irreducible polynomial exponent m and increasing numbers of check symbols, respectively. (Check symbols are the additional ones and zeros needed to provide the detection and correction desired; in other words, they represent redundancy.)

[1] P. Fire, "A Class of Multiple-Error-Correcting Binary Codes for Non-Independent Errors," Stanford Electronics Laboratories Technical Report No. 55, April 1959, Stanford University, Stanford, California.

[2] W. Wesley Peterson, *Error Correcting Codes*, M.I.T. Press and John Wiley and Sons, Inc., New York, 1961.

[3] For the complete set of codes see, P. T. Komiske, *Report on the Generation, Selection, and Implementation of Burst Error Correction and Detection Fire Codes*, The Johns Hopkins University, Applied Physics Laboratory, TG-666, March 1965.

The power (exponent) of the error burst detector, c, the number of check symbols, $c + m$, the number of information symbols, k, the code length, n, the transmission efficiency, k/n, and the order of the roots, e, of the irreducible polynomial $p(X)$ are also listed. An asterisk in the n column indicates that the computed length is not the code's actual length, i.e., it is not the least common multiple of e and c. In these cases, it is necessary to hand-compute the code length by determining the least common multiple of e and c.

The organization of information in the tables has been developed to expedite the selection of a particular code. Suggested selection rules are:

1. Select from one of the two tables a basic value of b.

2. From column m in that same table select the exponent of the irreducible polynomial. To facilitate this choice, it has been noted in several cases (m = 8, 9, and 10) that selection of the minimum value of m also minimizes hardware mechanization. This, however, requires verification for other values of m.

3. From column $c + m$ select the number of check symbols in accordance with requirements. (These requirements may call for (1) minimizing the number of check symbols, or (2) using the maximum number of check symbols to increase the detection capability.) Note that the tables provide for the use of 11 to 25 check symbols. System requirements, or the desired detection capability, determine which value of $c + m$ should be selected.

For example, to correct a burst equal to 4, select the table for b = 4, which contains 120 codes. Next, enter column m and select the minimum m (m = 4 in this case). This value should be selected to minimize hardware.

TABLE I

LIST OF CODES: BURST CORRECTION B = 3

M	C	C+M	K	N	K/N	E
3	5	8.	27.	35.	0.7714	7.
3	6	9.	33.	42.	0.7857	7.
3	7	10.	39.	49.*	0.7959	7.
3	8	11.	45.	56.	0.8036	7.
3	9	12.	51.	63.	0.8095	7.
4	18	22.	248.	270.	0.9185	15.
4	19	23.	262.	285.	0.9193	15.
4	20	24.	276.	300.*	0.9200	15.
4	21	25.	290.	315.*	0.9206	15.
5	5	10.	145.	155.	0.9355	31.

TABLE II

LIST OF CODES: BURST CORRECTION B = 4

M	C	C+M	K	N	K/N	E
4	7	11.	94.	105.	0.8952	15.
4	8	12.	108.	120.	0.9000	15.
4	9	13.	122.	135.*	0.9037	15.
4	10	14.	136.	150.*	0.9067	15.
4	11	15.	150.	165.	0.9091	15.
6	9	15.	552.	567.*	0.9735	63.
6	10	16.	614.	630.	0.9746	63.
6	11	17.	676.	693.	0.9755	63.
6	12	18.	738.	756.*	0.9762	63.
6	13	19.	800.	819.	0.9768	63.

Implementation Using Shift Register Techniques and Binary Circuit Elements

Encoding and decoding a Fire code involves the operations of addition, subtraction, multiplication, division, and recognition of predetermined patterns (such as "all zeros" or "all ones"). For transmission systems using a binary information channel, these calculations are carried out in a mathematical structure in which the rules are determined by the coding requirements. Addition and subtraction operations are performed in modulo two; the two operations are indistinguishable for the binary case, i.e., only two symbols, 1 and 0, are required. Rules of addition and subtraction are $1 + 1 = 0 = 1 - 1$. Similarly, $x + x = x - x = 0$ and $x = x$, $2x = 0$, $3x = x$, etc. Multiplication and division are treated as operations involving two polynomials; the calculations can be mechanized by shift register techniques.[1,2,4]

Shift Register Encoding Process

We have stated that the generalized Fire code generator polynomial $g(X)$ is given by:

$$g(X) = p(X)(X^c - 1),$$

where $p(X)$ is an irreducible polynomial of degree m and $c + m$ is the number of check symbols (bits in the binary case). Thus, $g(X)$ is a polynomial of degree $c + m$.

Now let the information to be transmitted be represented as a sequence of k bits. This sequence may be described by a polynomial q of degree $k - 1$,

[4] M. B. Greenlee, "Multiplication and Division of a Binary Number Polynomial by a Fixed Polynomial," APL/JHU Internal Memo S6BV-SGS-007, January 1963.

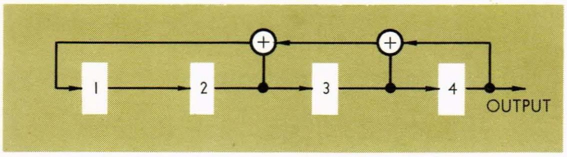


Fig. 1—Shift register for encoding a (7, 4) cyclic binary code.

$$q(X) = \sum_{i=0}^{k-1} a_i X^i,$$

with a_i being zero or one according to whether the ith bit in the sequence is a zero or a one.

This information can be encoded by multiplying it by the generator polynomial $g(X)$, which yields a polynomial $S(X)$ of degree $c + m + k - 1$. Thus, $S(X) = g(X) \cdot q(X)$. The sets of $S(X)$ are called "code vectors." Shift-register multiplication circuits can be utilized.[2,4]

The hardware required to perform this multiplication is a k-bit shift register and some number, v, of modulo-two adders. However, there is a problem in transmitting data from this shift register since the information symbols in the receiver do not appear in sequence until after decoding (division by $g(X)$).

Encoding in this manner is not desirable since a shift register with k stages is required and transmission of long information sequences, e.g. k very large, would require too much hardware. An example of this encoding method is Peterson's cyclic (7, 4) binary code

$$g(X) = 1 + X + X^3 \text{ and } h(X) = (X^7 - 1)/g(X)$$
$$= 1 + X + X^2 + X^4$$

in view of the interchangeability of + and − and the use of modulo-two operations . As a check, $[(1 + X + X^2 + X^4)(X^3 + X + 1) = X^7 + 2X^5 + 2X^4 + 2X^3 + 2X^2 + 2X + 1 = X^7 + 1 = X^7 - 1]$. The shift register shown in Fig. 1 can be used for encoding where ⊕ represents a modulo-two adder and □ represents a binary element.

A mechanization technique for generating the code vectors with less hardware is desirable.

Let $f(X)$ be the polynomial of degree $(n - 1)$, which represents the coded information received. If we use the well-known division algorithm $f(X) = g(X) \cdot q(X) + r(X)$, where $g(X)$ is the coding polynomial of degree $(n - k)$ used at the transmitter, we obtain the divided polynomial $q(X)$ and the remainder $r(X)$, where the degree of $r(X)$ is obviously less than $(n - k)$. We note that since $f(X)$ is a polynomial of degree $(n - 1)$, we can have $(n - k)$ new coefficients although only k bits of information are to be transmitted. We choose to have the first $(n - k)$ coefficients be zero, and write

$$f(X) = \sum_{i=n-k}^{n-1} a_i X^i.$$

We also write

$$r(X) = \sum_{i=0}^{n-k-1} r_i X^i \quad = 1$$

Then,

$$q(X) \cdot g(X) = (f(X) - r(X))$$
$$= \sum_{i=n-k}^{n-1} a_i X^i - \sum_{i=0}^{n-k-1} r_i X^i.$$

The coefficients of the polynomial obtained by differencing these two series may be considered to be the components of a vector $[f(x) - r(X)]$ in a space of dimension n. The set of such vectors may be thought of as code vectors, with $f(X)$ representing the information bits and $r(X)$ (low order terms) symbolizing the check bits. The relationship is shown below:

Low Order Bits High Order Bits Time Order of Transmission

$$\underbrace{(1\text{---------}1)}_{n\,-\,k\ \textit{Check Bits}} \quad \underbrace{(1\text{---------------}1)}_{k\ \textit{Information Bits.}} \quad \longrightarrow$$

To encode a sequence of information bits, the remainder, $r(X)$, must be calculated by dividing $f(X)$ by $g(X)$. *Only the remainder need be retained.*

This calculation can be mechanized by a shift-register division circuit.[2,3] Note that division is the conventional division process but that addition and subtraction are modulo two. Simultaneously, $r(X)$ is computed as the k information bits are shifted into the channel. At the end of the k shifts, $r(X)$ remains in the register. The feedback connections corresponding to the divisor, $g(X)$, are then disabled and the $(n - k)$ check bits, $r(X)$, are shifted into the channel, completing transmission of the code vector. This method is termed the $(n - k)$ shift-register mechanization.

For example, the (7, 4) cyclic binary code shown for k-bit register encoding is simplified as:

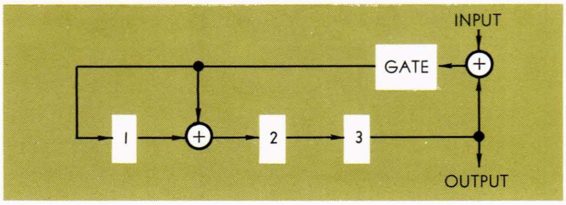


Fig. 2—An $(n - k)$ stage shift register encoder.

$$g(X) = 1 + X + X^3.$$

A block diagram of the shift register is shown in Fig. 2.

The hardware reduction in this case is one binary stage. For longer codes the reduction becomes greater, i.e., the shift-register length is $c + m$ instead of k. Remembering that $c + m = (n - k)$, the improvement is

$$\frac{k}{c+m} = \frac{k}{n-k} = \frac{1}{\frac{n}{k} - 1}.$$

Shift-Register Decoding Process

The receiver of the code vector (n bits) is assumed to have a-priori knowledge of the generator function $g(X)$, the code length n, and the number of information bits k.

The decoding process, which is not necessarily optimum for mechanization, is summarized as follows:

1. The k information bits received are stored in a k-bit buffer and, simultaneously, the received vector is divided by the generator polynomial. The remainder left in the register is $r(X)$, the check symbol results.

2. If $r(X) = 0$ at this time there were no errors in the received message. If $r(X) \neq 0$, the division process is continued with no input to the division circuit. For each shift in the decoder, one symbol is shifted out of the buffer, and after each shift the register contents are checked for a detectable error pattern ("all zeros" in the leftmost bits of the register).

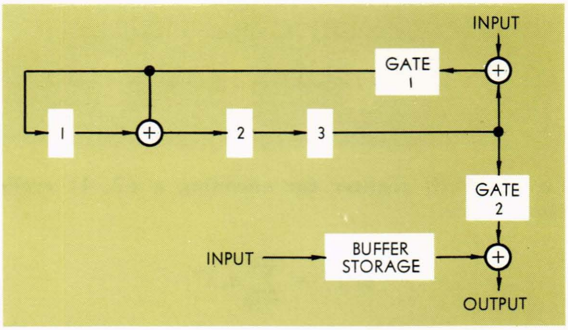


Fig. 3—(7, 4), ($n - k$) shift register decoder.

3. If a correctable error is detected, the feedback paths are disabled and the register output is added to the information symbols, one symbol at a time, as the symbols are shifted out of the buffer and the register.

An example of the decoder for the (7, 4) code using an ($n - k$) shift-register encoder, where $g(X) = 1 + X + X^3$, is shown in Fig. 3.

It is desirable to have the *correction* symbols appear at the output of the check-symbol calculation device at the same time that the information sym-

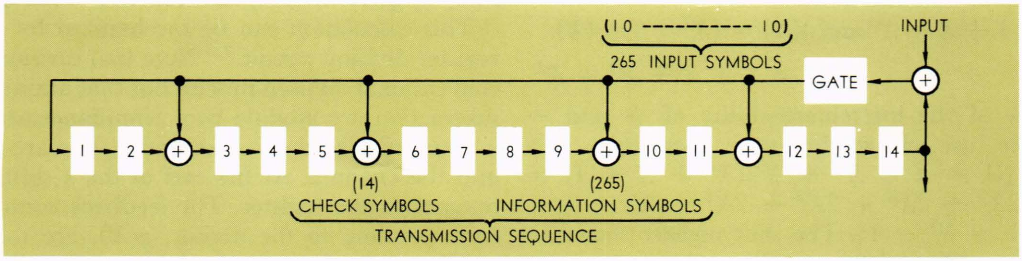


Fig. 4—(279, 265), ($n - k$) shift register encoder.

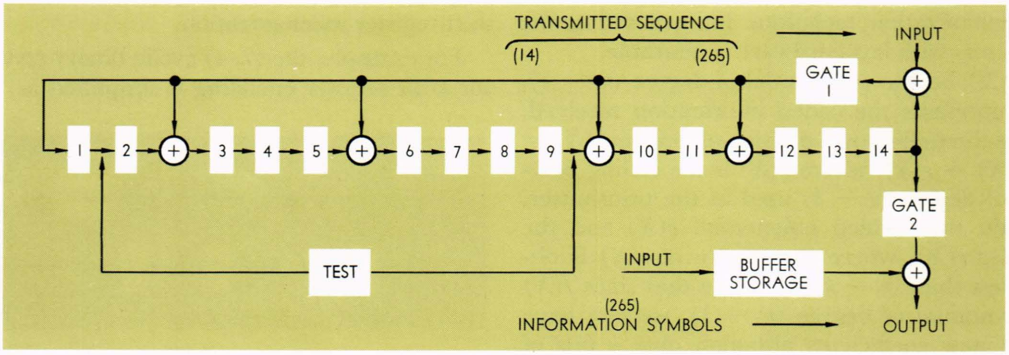


Fig. 5—(279, 265), ($n - k$) shift register decoder.

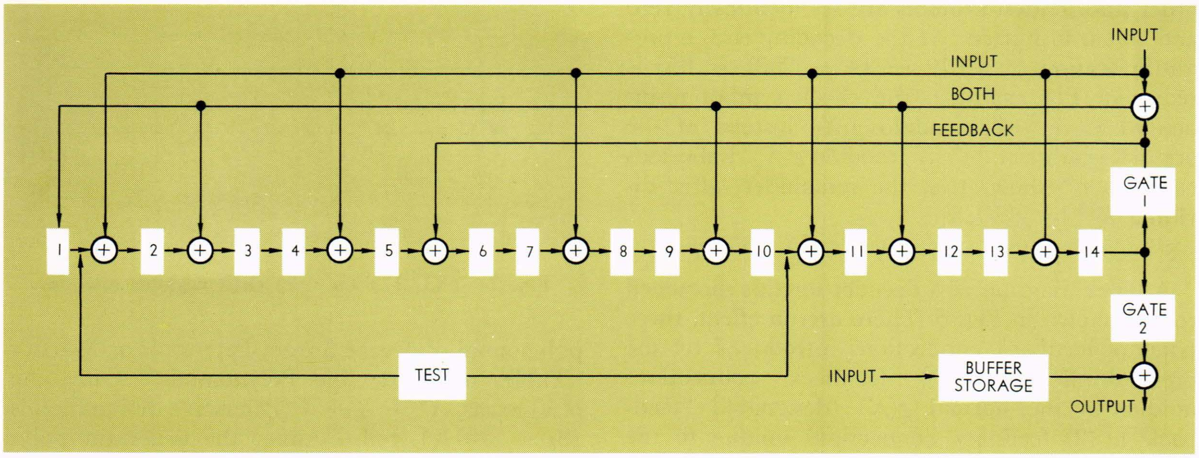


Fig. 6—(219, 200), $(n - k)$ shift register decoder.

bol to be corrected is available. To allow time for the division process required to obtain $r(X)$, the k information bits must be stored in a buffer while the vector $S(X)$ is divided by $g(X)$.

From the practical standpoint the $(n - k)$ shift-register decoder is considered to be the only feasible decoder as opposed to the shift register of length k. The same reasoning applies as in the case of the encoder in that the timing circuits are essentially eliminated, and the shift register length is kept at a minimum, i.e. equal to the number of check symbols.

Encoder-Decoder Mechanization

Complete mechanization of encoding and decoding will now be shown for several codes. First, consider the generator polynomial,

$$g(X) = p(X)(X^c - 1),$$

where

$$p(X) = X^5 + X^2 + 1, \qquad (m = 5),$$

and

$$(X^c - 1) = X^9 + 1, \qquad (c = 9),$$

and where $[g(X) = (X^5 + X^2 + 1)(X^9 + 1) = X^{14} + X^{11} + X^9 + X^5 + X^2 + 1]$ generates a binary Fire code of length $n = (2^5 - 1) \times 9 = 279$. This code corrects any single error burst of length 5 or less. It has 14 check symbols and $279 - 14 = 265$ information symbols. Encoding can be done with the $(n - k)$ shift register shown in Fig. 4. The decoder circuit is shown in Fig. 5. Note that the decoder and encoder shift registers are identical.

The operation of the decoder follows the format outlined previously. Specifically, for this mechanization:

1. The received code vector, i.e. 279 bits, is shifted into the buffer and shift register simultaneously. Gate 1 is open and allows the input to pass to the output. Gate 2 is closed until the entire code vector has been shifted into the register (the information bits are stored simultaneously in the buffer).
2. The received code vector is shifted out of the buffer one symbol at a time, and the shift register is shifted once for each symbol, with no input.
3. The error pattern must be in the last five stages when all zeros appear in the first nine stages and the error pattern is about to come out of the buffer. Gate 1 is closed, Gate 2 is opened, and the symbols will be corrected. If the first nine stages never contain all zeros, an uncorrectable error pattern has been detected. If all stages contain zeros, no errors have occurred.

Mechanization of a Shortened Code

It was stated earlier that a shortened code is required for some systems. This must be done if it is required to correct longer bursts, with the hardware and/or system requirements held within practical limits. Note the code-burst-correcting capability versus the length of the code in Tables I and II.

A code can be shortened by setting some of the high-order information symbols identically zero and omitting them. While the encoding and check-symbol calculations are not affected by the leading zeros, decoding is affected. Assume that a code has 200 information symbols and is required to correct any burst of length 5 or less. The code already described can be used. To use the generator with the shortened system (214, 200), the high-order information symbols are set identically zero and not transmitted. At the decoding end, a pre-multiplication by $[(279 - 14 - 200) = 65]$ is required. For example, the check symbol result desired is $X^{79}r(X)$ modulo $g(X)$ instead of the original residue of $X^{14}r(X)$ modulo $g(X)$. Laborious calculation* shows that the remainder, after dividing X^{79} by $g(X)$, is

$$X^{13} + X^{11} + X^{10} + X^9 + X^7 + X^4 + X^2 + X + 1.$$

A block diagram of a decoder for this shortened code is shown in Fig. 6. There are, in effect, three types of feedback connections introduced by the pre-multiplication; (1) feedback connections unique to the normal $g(X)$ (designated "feedback"), (2) feedback connections unique to the shortened $g(X)$ (designated ("input"), and, (3) feedback connections common to both generator polynomials (designated "both"). The two applicable generator polynomials are shown below:

Normal

$$g(X) = X^{14} + X^{11} + X^9 + X^5 + X^2 + 1$$

Common Common

Shortened

$$g(X) = X^{13} + X^{11} + X^{10} + X^9 + X^7 + X^4 + X^2 + X + 1.$$

Encoder-Decoder Mechanization of Codes Generated

Complete encoding-decoding mechanization for several codes from Tables I and II is given below to illustrate the procedure.

1. *(35, 27) Code Mechanization*

Consider the first code with a burst-correcting capability of 3 found in the tables. The characteristics of this code are:

b	$= 3$
m	$= 3$
c	$= 5$
$c + m$	$= 8$
k	$= 27$
n	$= 35$
e	$= 7.$

The generator polynomial is:

$$g(X) = p(X)(X^c - 1) = p(X)(X^5 - 1),$$

where $p(X)$ is an irreducible polynomial[2,5] of degree m. It is now necessary to select an irreducible polynomial of degree 3 from Peterson's or Marsh's "Tables of Irreducible Polynomials." Only one $p(X)$ exists, $X^3 + X + 1$ (Peterson's designation is $13_8 = 1011_2$). Substituting, the generator polynomial becomes

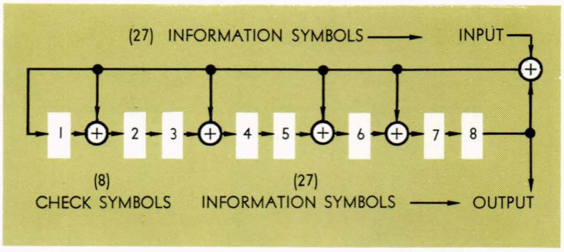


Fig. 7—(35, 27), (*n* – *k*) shift register encoder.

$$g(X) = (X^3 + X + 1)(X^5 - 1),$$
$$= X^8 + X^6 + X^5 - X^3 - X - 1$$

or

$$= X^8 + X^6 + X^5 + X^3 + X + 1 \text{ (since + and – are equivalent in this algebra).}$$

The $(n - k)$ shift-register encoder and decoder block diagrams are shown in Figs. 7 and 8, respectively.

2. *(23, 15) Shortened Code Mechanization*

Now consider shortening this code, say to a total of 23 symbols from its original length of 35 symbols, i.e. (23, 15). The code's capability remains fixed—it is in fact more powerful, but more redundant—and now has these capabilities:

Shortened		versus	*Normal*	
b	$= 3$		b	$= 3$
m	$= 3$		m	$= 3$
c	$= 5$		c	$= 5$
$c + m$	$= 8$		$c + m$	$= 8$
k	$= 15$		k	$= 27$
n	$= 23$		n	$= 35$
e	$= 7$		e	$= 7$

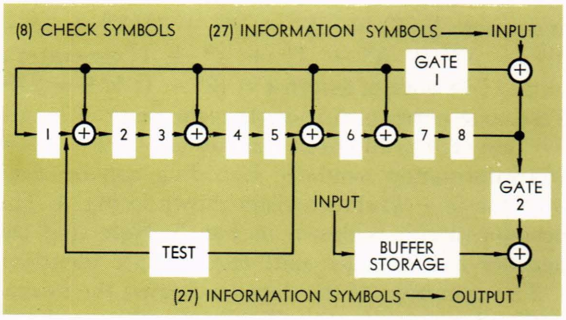


Fig. 8—(35, 27), (*n* – *k*) shift register decoder.

* A multiplier-divider circuit of the type described in Ref. 4 can be used for this calculation.

[5] R. W. Marsh, "Table of Irreducible Polynomials Over GF(2) Through Degree 19," NSA Report, Washington, D.C., 1957.

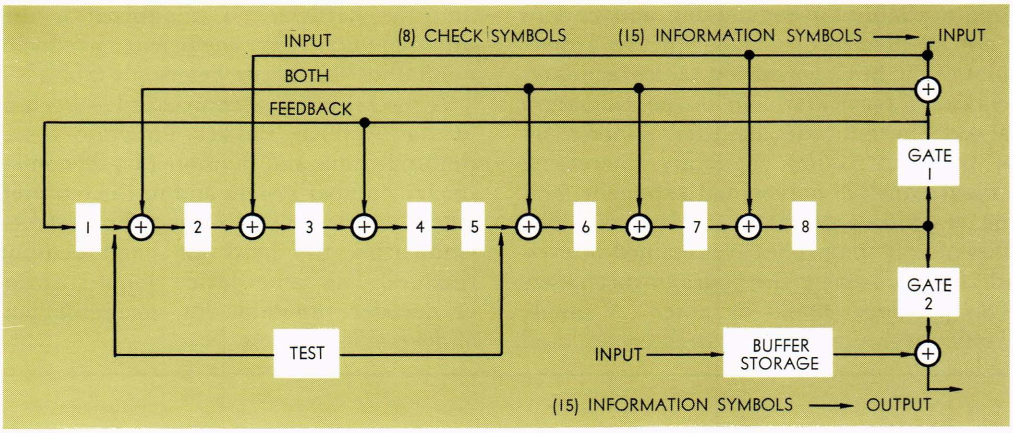


Fig. 9—Shortened code (23, 15), (n – k) shift register decoder.

The generator polynomial for the shortened code (23, 15) must be computed. Assume that the 12 high-order information symbols are identically zero and are not transmitted. A pre-multiplication by [(35 – 8 – 15) = 12] is required. This is equivalent to shifting 12 times in the shift register, which takes care of the 12 symbols that are set to zero. The check symbol result desired is $X^{20}r(X)$ modulo $g(X)$ rather than $X^8r(X)$ modulo $g(X)$. By performing the division of X^{20} by $g(X)$ the required remainder is found to be $X^7 + X^6 + X^5 + X^2 + X$.

The new $g(X)$ consists of three sets of feedback terms as shown previously. These are:

Normal or original:

$$g(X) = X^8 + X^6 + X^5 + X^3 + X + 1$$

↑ ↑ ↑

Common Common

Shortened: ↓ ↓ ↓

$$g(X) = X^7 + X^6 + X^5 + X^2 + X$$

There are feedback connections unique to the normal $g(X)$, feedback connections unique to the shortened $g(X)$, and feedback connections common to both generator polynomials. The generator polynomial becomes $g(X) = X^8 + X^7 + X^6 + X^5 + X^3 + X^2 + X + 1$ and is mechanized as shown in Fig. 9. Note the three types of feedback connections.

This example clearly shows the increased decoder complexity resulting from shortening the code, but the complexity is represented only by an increased number of modulo-two adders or "exclusive or's," and not by a change of shift-register length.

The result is that a very long code can retain all of its capability with the shortened length, i.e., pre-multiplication by the number of shifts results *only* in a change of feedback connections in the decoder. The worst case is a modulo-two adder between each decoder stage.

Additional items of interest have been investigated. For example, the second code with a burst-correction capability of 4 has a greater single burst-detection capability, $c + 1 = 9$, than does the first code, $c + 1 = 8$, but has the same correction capability of 4. This is a feature that may be desirable in some systems in that the detection capability of the code has been increased. This, however, is at the expense of hardware; the code requires 12 shift-register stages, where the first required only 11.

The first irreducible polynomial for a given degree in Peterson's or Marsh's tables has a *minimum number of non-zero coefficients*. Investigation has shown that in many cases this irreducible polynomial requires the *least number of feedback connections in the decoder*.

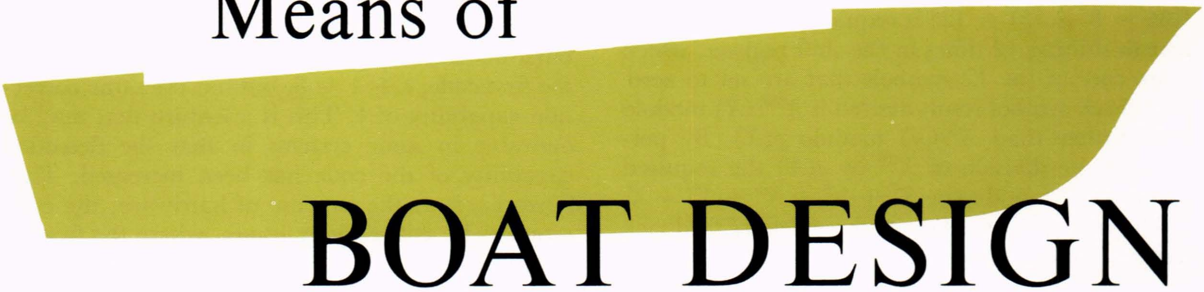
Comparison of the complexity of the encoder-decoder, using the two fourth-degree irreducible polynomials (*minimum* non-zero coefficient irreducible polynomial and *non-minimum* non-zero coefficient irreducible polynomial) to implement the first code of Table II discussed above, shows a significant increase in hardware. The *minimum* non-zero-coefficient polynomial requires five modulo-two adders in the encoder and six modulo-two adders in the decoder. The *non-minimum*, non-zero-coefficient polynomial requires nine modulo-two adders in the encoder and ten modulo-two adders in the decoder. Both methods do exactly the same job. Similar results are evident in the shortened code mechanization.

Conclusions

Fire codes, through generator degree 33, with a minimum burst correction capability of 2, have been made available for engineering applications but are not all described in detail in this article. The format of the burst correction tables facilitates the selection of a particular code or the evaluation of several codes in that codes are listed in increasing order of burst correction capability, increasing order of the irreducible polynomial exponent, and increasing magnitude of the check symbols. The capabilities of very long codes are retained or even increased after shortening the codes to mechanize them. This property should be noted. A simple mechanization is available if shift registers are used and the hardware is minimized if the first or minimal non-zero coefficient irreducible polynomial in Marsh's or Peterson's tables is used.

A special-purpose computer has been developed at the Applied Physics Laboratory to perform multiplication and division of polynomials in the binary number system and in the required algebra when it is desirable to use a shortened code, thus eliminating the need for hand computation of residues. The same device doubles as an encoder or decoder simulator for generator polynomials of degree 36 and less.

Ways and Means of BOAT DESIGN



J. R. Apel

The two most beautiful forms in creation belong to a well-designed sailboat and a well-shaped woman. A categorical statement such as this would ordinarily bring a flood of abuse upon the person who made it, but among judges of boats and women, the statement goes virtually uncontested. Though both subjects would make an interesting discussion, this paper will concern itself only with boats.

To those who know and understand them, there is real beauty in the lines of a hull or the set of the rigging of a boat, be it an 8-foot dinghy or an 80-foot diesel yacht. To those who are untrained in things nautical but who respond keenly to the visual arts, it is apparent that ships and boats have a high degree of functional styling about them. It takes only a bit of study and observation by the novice to become educated to some of the niceties of the business and to become convinced that the old saw about boats being called "she" has more truth to it than would appear at first glance.

This article will discuss the design of pleasure boats and the designers; it will also discuss the considerations that enter into designing different types of boats. Examples of a few designs will be shown with the emphasis on speed boats and hydroplanes; and an examination, in a bit of detail, will be made of a 150-mph Gold Cup hydroplane. The presentation of this article will be given from the standpoint of one whose vocation (in more fortunate times, perhaps) encompassed much of this subject matter, but who is now reduced to boating and boat design as an avocation only. So this will be a "hobby" article.

Few of the technicalities of the trade will be presented and virtually no connection will be made with its mother-science, fluid dynamics. This is because first, the technicalities are not too interesting and second, yacht design is much more nearly an art than a science. With the exception of a few craft such as the 12-meter racing sailboats of recent years, orderly research effort in pleasure

J. R. Apel, a physicist in the Plasma Dynamics Group, co-authored a paper entitled "Beam-Plasma Interactions" in the May-June 1964 *Digest*. Coming from a boatbuilding family, Mr. Apel studied boat design at the Westlawn School of Yacht Design and practiced this profession for several years.