SOME EXAMPLES OF SINGLE AND DOUBLE
ADJACENT ERROR CORRECTION

F. B. Wood

and

W. J. Johnson, Jr.

SOME EXAMPLES OF SINGLE AND DOUBLE
ADJACENT ERROR CORRECTION

By

F. B. WOOD

and

W. J. JOHNSON, JR.

ABSTRACT

N. M. Abramson has developed a class of single-error-correcting, double-adjacent-error-correcting codes (SEC - DAEC). The theoretical analysis has been documented through a report of the Stanford Electronics Laboratories. Disclosures on encoders and decoders for SEC - DAEC codes have been submitted by N. M. Abramson to IBM. The function of this report is to give a digest of the SEC - DAEC codes and to illustrate one example of single-error-correction and one example of double-error-correction to assist IBM engineers in evaluating the potential application of this class of codes. References are made to other codes and analyses of same which should be considered in determining what code IBM should adopt for data transmission.

NOTICE

TABLE OF CONTENTS

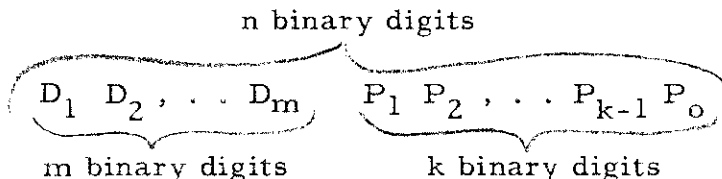*These two sections are a condensation of N. Abramson's report (Ref. 3).

Introduction

The occurrence of independent errors in a binary communication system may be combated quite effectively through the use of single error-correcting block codes.[1],[2] In many cases, however, (e.g., telephone lines with impulse noise) errors do not occur independently and the probability that the n'th binary digit will be in error will depend upon whether the (n-1)th binary digit was in error. That is, in many situations, it is quite probable that if two errors occur in one word the errors are in adjacent binary digits. Using a double error correcting code[1],[2] in such situations will increase the reliability of transmission at the cost of a large increase in the number of binary digits necessary to transmit each word. Such a code clearly does not make use of the fact that double errors are likely to be adjacent errors.

N. M. Abramson, Stanford University, has developed a class of systematic codes for correcting double adjacent errors (3). He has also prepared IBM patent disclosures on some hardware to utilize this class of codes (4, 5). The objective of this report is to illustrate the functioning of two sample cases in the SAC - DAEC code. The class of codes developed by Abramson are more efficient than simply using a Hamming double error correcting code [1]. A comparison of the upper bound on the number of possible information bits when k parity bits are used is shown in Table I for SEC - DAEC codes and for ordinary double-error-correcting codes.

From Table I it can be seen that if the multiple errors in a character or symbol are at most double adjacent errors, that the SEC - DAEC code is more efficient than the Hamming code. Potential application of this code should be compared with the "burst correcting codes" developed at Bell Telephone Laboratories.[6],[7],[8],[9]

Rules for Generating SEC-DAEC Codes*

Because of the fact that we restrict our attention to systematic block codes we may write any possible word of our code as

n binary digits

$$D_1 \quad D_2 , \ldots D_m \qquad P_1 \quad P_2 , \ldots P_{k-1} \quad P_o$$

m binary digits   k binary digits

That is, we shall assume that each word is n binary digits long. M of these n digits may be used to convey the information; we denote these by D_1, D_2, . . . D_m . The remaining k digits, denoted by P_1 , P_2, . . . P_{k-1} , P_o , are determined by suitable parity checks over the information digits.**

The first property of the codes which must be determined is the number of parity digits necessary for any given number of information digits. Stated another way, we might ask for the largest number of information digits we may use for a specified number of parity digits, k. In Table I we have listed for k = 4, 5, ... 8, the maximum number of information digits possible for the class of SEC-DAEC codes obtained. This upper bound is denoted by m* - 1. For purposes of comparison, we have also given in the same table m**, an upper bound on the number of information digits possible for codes which correct all double errors.

We shall present a set of rules for constructing SEC-DAEC codes where for any given number of parity digits, k, the number of information digits is m* -1, the maximum possible as indicated in Table I. In order to completely specify a systematic code, it is only necessary to specify the digits checked by each of the parity digits. A table for determining the first parity bit is given in Table II. An example of how this is used in constructing a code is given in Table III for the case m = 10, k = 5.

The rules, then, for filling out a parity check table of k rows (corresponding to the k parity digits) and m* - 1 + k = n columns (corresponding to the m* - 1 information digits and the k parity digits) are as follows:

---

* See footnote on p. 1
**Even parity is assumed throughout this paper.

REFERENCE 3, TABLE I

Table I - Comparison of Codes		
Redundant Bits k	Information Bits	
	$m^* - 1$ SEC - DAEC Abramson	m^{**} SEC - DEC Hamming
4	3	1
5	10	2
6	25	4
7	56	8
8	119	14

k = number of parity digits

$m^* - 1$ = upper bound on the number of possible information digits for the class of SEC-DAEC codes obtained.

m^{**} = upper bound on the number of possible information digits for ordinary double error correcting codes.

1. Number the rows P_1, P_2, . . . P_{k-1}, and P_o as indicated in the example in Table III.

2. Let P_o be a parity check over all the digits.

3. The parity checks for P_1 may be determined from Table II.

For the derivation of this table refer to N. M. Abramson's report.[3]

For any given k we use the sequence of zeros and ones in Table II to obtain the parity checks on P_1 as follows:

For any k the corresponding sequence in Table II is m* -1 + k digits long. If the j th digit in this sequence is a zero P_1 will check the j th digit of the code words; if the j th digit in this sequence is a one P_1 will not check the j th digit of the code words. That is, we need only write the sequence obtained from Table II in the first row of the parity check table we are constructing, using a cross for a zero and a blank space for a one. For example:

for k = 5, Table II gives the sequence:

0 1 0 1 1 0 0 1 0 0 0 1 1 1 1 .

The row P_1 in Table III was derived by placing a cross (x) in each position having a zero in the above sequence. The notation of Table III means that for this code parity bits:

P_1 checks information bits D_1, D_3, D_6, D_7, D_9, and D_{10}.

4. The parity checks in each succeeding row of the parity check table (except the last, P_o) are then the same as the row directly above, except that they are shifted to the right by one digit. (See Table III).

These four rules together with Table II allow one to construct SEC-DAEC codes.

Generation of Parity Checks by Binary Shift Registers

The fact that the codes obtained in this paper may be simply instrumented depends directly upon the properties of the sequence of zeros and ones giving the parity checks for P_1. (See Rule 3). Each of the sequences

TABLE II

Parity Checks for
P_1 in SEC-DAEC Codes
(m – sequences)

Digit number k	4	5
1	0	0
2	0	1
3	1	0
4	0	1
5	1	1
6	1	0
7	1	0
8		1
9		0
10		0
11		0
12		1
13		1
14		1
15		1

(For an extended table see Ref. 3, Table II)

TABLE III

Parity Table for $m = 10$, $k = 5$.

	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	P_1	P_2	P_3	P_4	P_0
P_1	x		x			x	x		x	x	x				
P_2		x		x			x	x		x	x	x			
P_3			x		x			x	x			x	x	x	
P_4				x		x			x	x			x	x	x
P_0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

(Ref. 3, Fig. 3)

of length $2^R - 1$, listed in Table II may be derived from a R stage binary shift register. A block diagram of a four-stage register is given in the upper part of Fig. 1 consisting of flip-flop F_1, F_2, F_3, F_4 with four switches, and three mod-2 adders. In Fig. 1 the switches are set in the pattern 1001 corresponding to the values found in Table IV for R = 4. The procedure is to insert an arbitrary binary number (except 0000) into the flip-flops. This binary number is then shifted to the right every T seconds while simultaneously we feed into F_1 as indicated in the diagram. The successive entries of zeros and ones into F_1, then form a linear binary shift register sequence. (T is the period or bit time).

For certain settings of the switches the successive entries in the flip flops of a R-stage register will be all the R digit binary numbers except the all zero number. In this case, the sequence of zeros and ones in F_i is periodic of period $2^R - 1$ and is called a maximal length linear binary shift register sequence -- or m-sequence. The sequence of zeros and ones used in Rule 3 can always be taken to be an m-sequence out of k - 1 stage shift register ending in k - 1 ones. Putting all ones into the four flip flops F_1, F_2, F_3, F_4 of figure 1 with the switches as indicated will cause the sequence of length $2^4 - 1 = 15$ given in Columns F_i in Fig. 2 to appear the F-register starting with the first shift.

The encoding operation for these SEC-DAEC codes then can make use of the fact that the parity checks are derived from m-sequences. For example, the timing signals for P_1 may be obtained directly from the first flip flop of a k - 1 stage shift register which is started with ones in all k - 1 flip flops. The timing signals for P_2 to P_{k-1} are the same except shifted in time.

The decoding operation is almost as simple as encoding. The first step of course, is to form the checking number C_1 C_2 . . . C_{k-1} C_0 as in an ordinary Hamming code. That is, we set $C_1 = 0$ if P_1 of the received word satisfies its parity check; we set $C_1 = 1$ if P_1 does not satisfy its parity check. The timing signals for this operation are, of course, derived in the same manner as in the encoding operation.

There are then four possibilities:

(1)   All the C_1 are zero:

No errors occurred.

(2) C_0 is one; C_1 is zero for some i:

A single error occurred.

TABLE IV

SWITCH SETTINGS

(VALUES OF S_1 FOR R-STAGE MAXIMAL LENGTH LINEAR BINARY SHIFT REGISTERS)

R	3	4	5	6	7	8	9	10
S_1	0	1	0	0	0	0	0	0
S_2	1	0	0	0	0	0	0	0
S_3	1	0	1	0	0	0	0	0
S_4		1	0	0	0	1	0	0
S_5			1	1	0	1	1	0
S_6				1	1	1	0	0
S_7					1	0	0	1
S_8						1	0	0
S_9							1	0
S_{10}								1

(For an extended Table see Ref. 3, Table 3).

We take a $k - 1$ stage shift register with the switches (S_i) given by Table IV. This shift register is also started with all ones in the flip flops. We start shifting the contents of the flip flops in the usual manner. After some number of shifts, say N, the mod 2 sum of the number in the flip flops and the first $k - 1$ digits of the checking number will be all ones.

The error occurred in the N th digit of the word.

(3) C_o is zero; C_i is one for some i:
A double adjacent error occurred*.

We take a $k - 1$ stage shift register with the S_i given by Table 3. This shift register (which can be the same register used for possibility 2) is started with a one in F_i and zeros in all the other flip flops. After N shifts the mod 2 sum of the number in the flip flops and the first $k - 1$ digits of the checking number will be all zeros.

The double adjacent error occurred in the N th and N + 1 th (mod n) digits of the word.

(4)   All the C_i are one:

An odd number of errors greater than one occurred. **

Sample Cases

To illustrate the code let us consider a six bit alphanumeric code having 64 possible symbols with a four bit station identification symbol representing a distinction between sixteen local stations. This makes ten bits of information per character sent into an intermediate buffer from a remote keyboard. Looking at Table I, we see five parity bits are needed for the SEC-DAEC code. Note that eight parity bits would be needed in a Hamming SEC-DEC code, instead of the five needed in this code.

To illustrate single-error-correction we shall use the examples of Table V in which the RAMAC character code is used. (10).

---

*-------These codes consider the case where both the first and last digits of a word are in error as a double adjacent error and automatically correct the word if this occurs. Examples are shown in Table IX in the Appendix.
** See Table X in the Appendix for an illustration of this.

TABLE V

SAMPLE MESSAGES
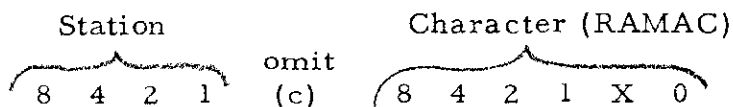
	Station				omit (c)		Character (RAMAC)					
	8	4	2	1			8	4	2	1	X	0
Digit in SEC-DAEC Example	10	9	8	7			6	5	4	3	2	1

First Example:

	8	4	2	1	(c)		8	4	2	1	X	0
Letter "R" from station "10"	1	0	1	0			[1]	0	0	1	1	0

Single Error in Position 6:

	8	4	2	1	(c)		8	4	2	1	X	0
Received as "J" instead of "R".	1	0	1	0			[0]	0	0	1	1	0

Second Example:

	8	4	2	1	(c)		8	4	2	1	X	0
Letter "J" from Station "14".	1	1	[1	0]			0	0	0	1	1	0

Double Error in Position 7, 8:

	8	4	2	1	(c)		8	4	2	1	X	0
Received as Letter "J" from Station "13" instead of "14"	1	1	[0	1]			0	0	0	1	1	0

Encoding

The encoder is illustrated by Figure 1. The blocks marked "+" are mod 2 adders, and the blocks marked "A" are AND circuits. Only fifteen steps of the clock are used in encoding. The additional ten are sometimes used in decoding for the error correction cycle. The Four Stage Shift Register F_1, F_2, F_3, F_4 enclosed in the dotted lines produces a maximal length linear binary shift register sequence. In this example the switches are set $S_1 = S_4 = 1$ and $S_2 = S_3 = 0$ so that the sequences of Ref. 3 (Fig. 4) are produced.

The rules (Ref. 4) for encoding are as follows:

(a)   Cycles 1 through 10:

   (1)   Start $F_i = 1$, $P_i = 0$

   (2)   The content of D is shifted out and first digit of message is entered in D

   (3)   If D = 1 and $F_i = 0$, P_i changes; (i = 1, 2, 3, 4) otherwise P_i remains the same.

   (4)   If D = 1, P_0 will change; if D = 0, then P_0 remains the same.

   (5)   F shifts

(b)   Steps 11-15 (i. e. , steps 1k for k = 1, 2, 3, 4, 5)

   Omit step (1).

   (2)   The content of D is shifted out and P_k is entered in D. (k = 1 - 4, 0) (On step 15 use P_0).

   (3)   (4), (5) the same as in (a).

The sample message used in Fig. 2 is "R from station 10" or 1010100110. The process illustrated in Fig. 2 adds the digits 01101 to make the encoded symbol: 011011010100110. This can also be
$$P_0 ---P_1 \; D_{10} ----D_1$$
derived from Table III by adding up the digits appearing in the "x" squares except the P_i P_i square and making P_i "0" or "1" to obtain even parity.
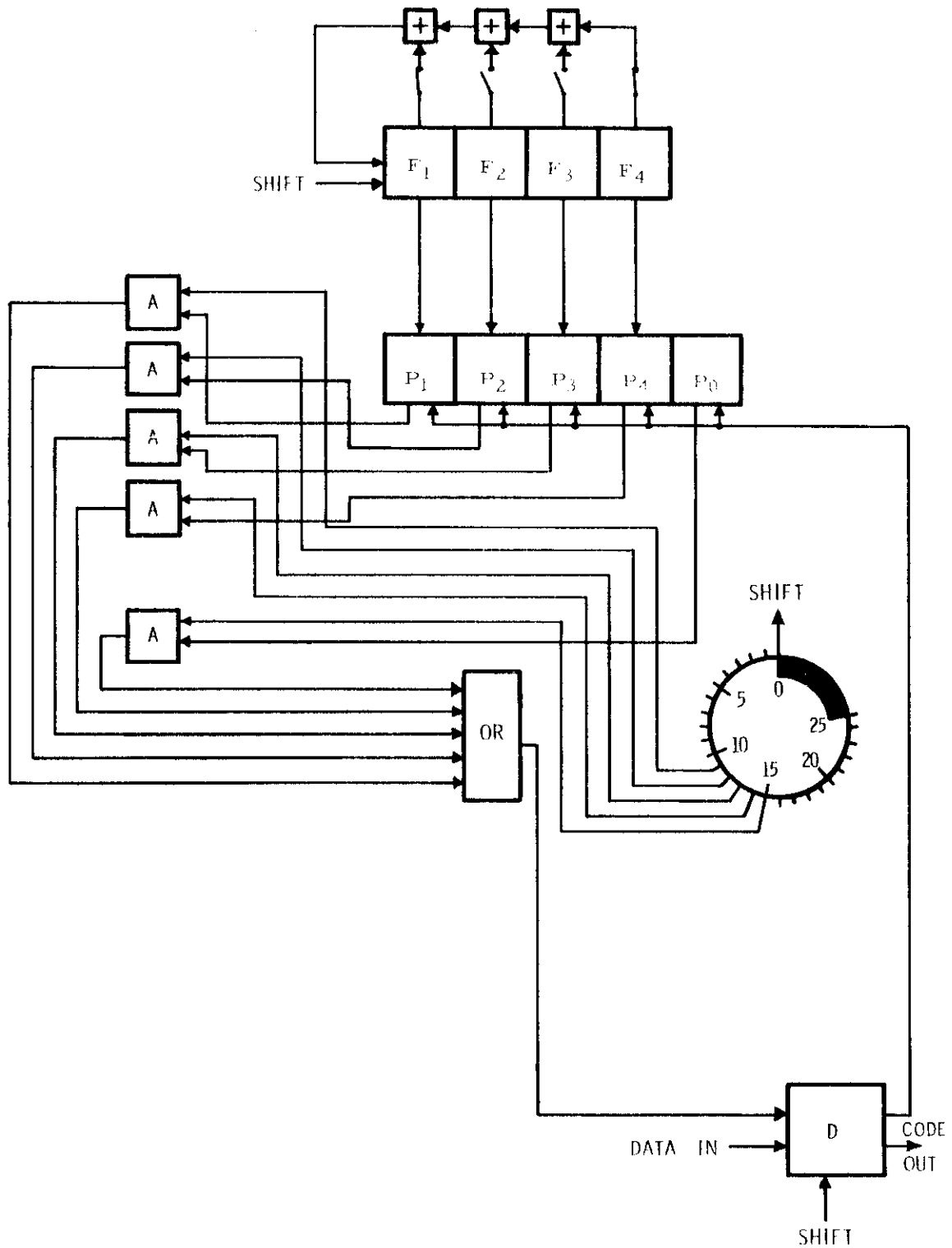
FIGURE 1 - ENCODER

Incoming Line or Buffer

F or R

P

D

Output Line or Buffer
S

Parity Bits

R

10

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

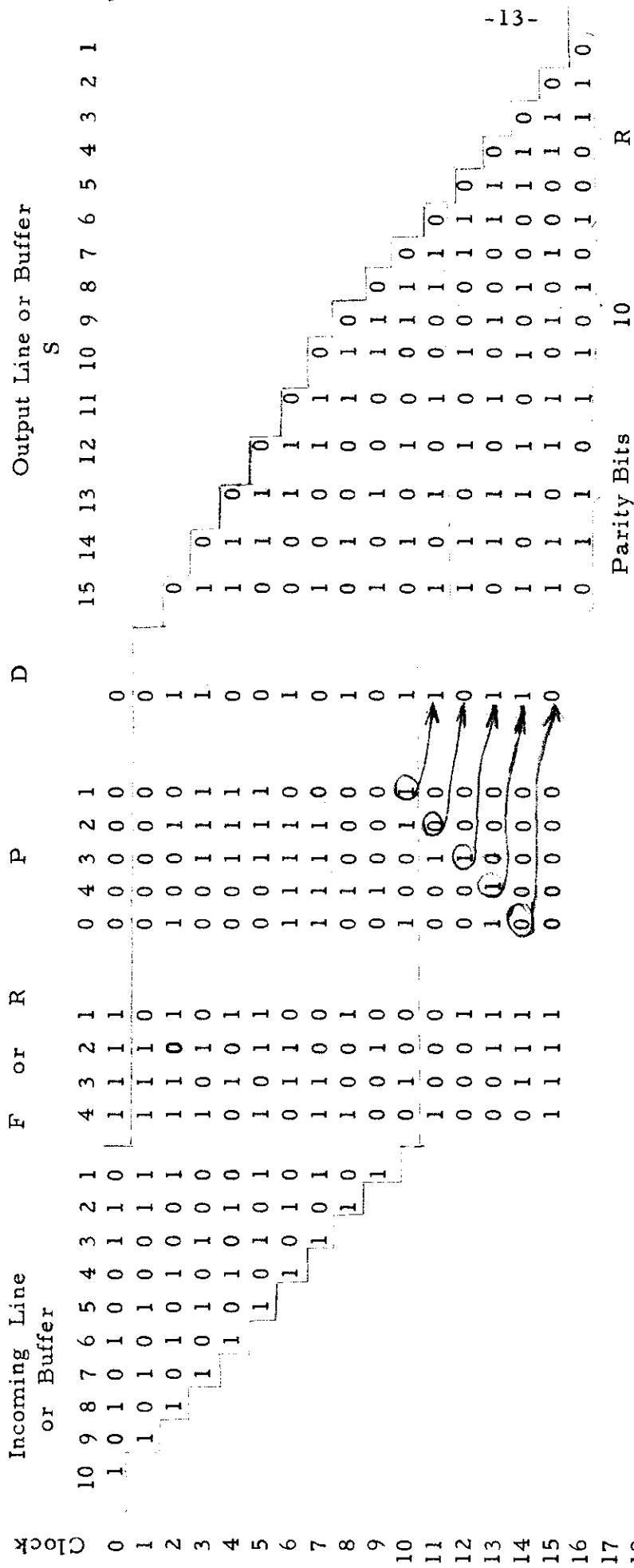Clock
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
0

FIGURE 2 – EXAMPLE OF ENCODING

For example:

$$D_1 + D_3 + D_6 + D_7 + D_9 + D_{10} = 1, \text{ so } P_1 = 1$$

$$D_2 + D_4 + D_7 + D_8 + D_{10} + D_1 = 0, \text{ so } P_2 = 0$$

$$D_3 + D_5 + D_8 + D_9 + P_1 + P_2 = 1, \text{ so } P_3 = 1$$

$$D_4 + D_6 + D_9 + D_{10} + P_2 + P_3 = 1, \text{ so } P_4 = 1$$

$$\sum_{1 \text{ to } 10} D_i + \sum_{1 \text{ to } 4} P_i = 0, \qquad \text{so} \quad P_0 = 0$$

01101

This calculation from Table III agrees with the derivation in Fig. 2. Note that the registers F and P are automatically returned to the starting positions.

Decoding

The basic procedure is to compute the parity bits again during receipt of the information bits, and to compare the recomputed parity bits with the received parity bits. If there is a discrepancy, the checking number obtained will tell in which bits the errors occurred. In practice the computing of the new parity bits and the comparison with the transmitted ones can be combined so that the received parity bits are used in computing the new parity bits so that the computed bits left in registers P_0 P_4 P_3 P_2 P_1 after Step 15 are the checking number bits.

The same basic circuits are used as in the encoder, plus the inclusion of a 10-bit buffer shift register and some additional switching logic is required for the decoder. A possible circuit is shown in Fig. 3. Each square or rectangle represents a trigger (flip-flop) unless marked otherwise. A "$+$" indicates a mod 2 adder, and "A" an "AND" circuit. In this circuit the computation and comparison are overlapped into a combined operation.

The counter starts at zero. The P_i and F_i registers start as in encoding. Information bits coming in are shifted from D into S_{10} and step by step to S_1. The encoding is duplicated except the received parity bits are used in the calculation of the checking number.

On step 16 the shift is omitted and S_{15} (P_0) is added mod 2 to S_{14}, S_{13}, S_{12}, and S_{11}. The compliment of F_0 is added to F_1, F_2, F_3

(similar to R_2, R_3, R_4 of Ref. 5). This also sets up register S_0 if $P_0 = 0$, indicating a double error. The function of flip-flop S_0 is to provide that when the first error is corrected during steps 17-25, the register S_0 is set up so that the next bit will also be corrected by changing the bit in S_1 in addition to changing the bit in D^1.

During steps 17-25 the shift register F_i and the message buffer $S_1 - S_{10}$ shifts during the complete set of cycles 17-25, 0 so that the full ten information bits will be shifted out through register D^1.

The shift register F_i shifts until the compare circuit shows a coincidence. Then the bit in error is in register D^1, so A "1" is added to the content of D^1 to effect correction. If there was a double error the content of S_1 would also be changed.

One addition to the circuit is still required for automatic operation. Note that in Fig. 4 on the reset (0) F = $\overleftarrow{0100}$ and P = $\overleftarrow{10110}$. These should be F = 1111 and P = 00000. A reset control has to be added to accomplish this.

Single Error Correction (SEC)

Consider the case of a single error in digit 6 which converts the letter "R" to "J". The sequence of operations is traced in Fig. 4. Note that the difference between decoding and encoding is that on steps 11-15, the received parity bits are used in calculating the next values of P in decoding, (Fig. 4), while the values of P on the diagonal shown in Fig. 2 are used in the original encoding.

Examination of step 15 in Fig. 4 shows that the checking number is in the P register. The fact that P_0 = 1 indicates a single error. The number $P_4 P_3 P_2 P_1$ is the compliment of the value of $F_4 F_3 F_2 F_1$ (when started from 1111) at the correct step for correcting the error. The checking number 11001 can be verified from Table III by noting that $P_0 P_4 P_3 P_2 P_1$ = XX00X in column D_6, which corresponds to an error in the 6th bit.

Examination of Fig. 3 for step 16 shows that the "1" bit from P_0 is added to P_4, P_3, P_2, and P_1. Then on step 22: $F_4 F_3 F_2 F_1$ = $P_4 P_3 P_2 P_1$. The compare circuit then changes the bit in D^1, so the message is corrected as in step 22 in Fig. 4.

Double Adjacent Error Correction (DAEC)

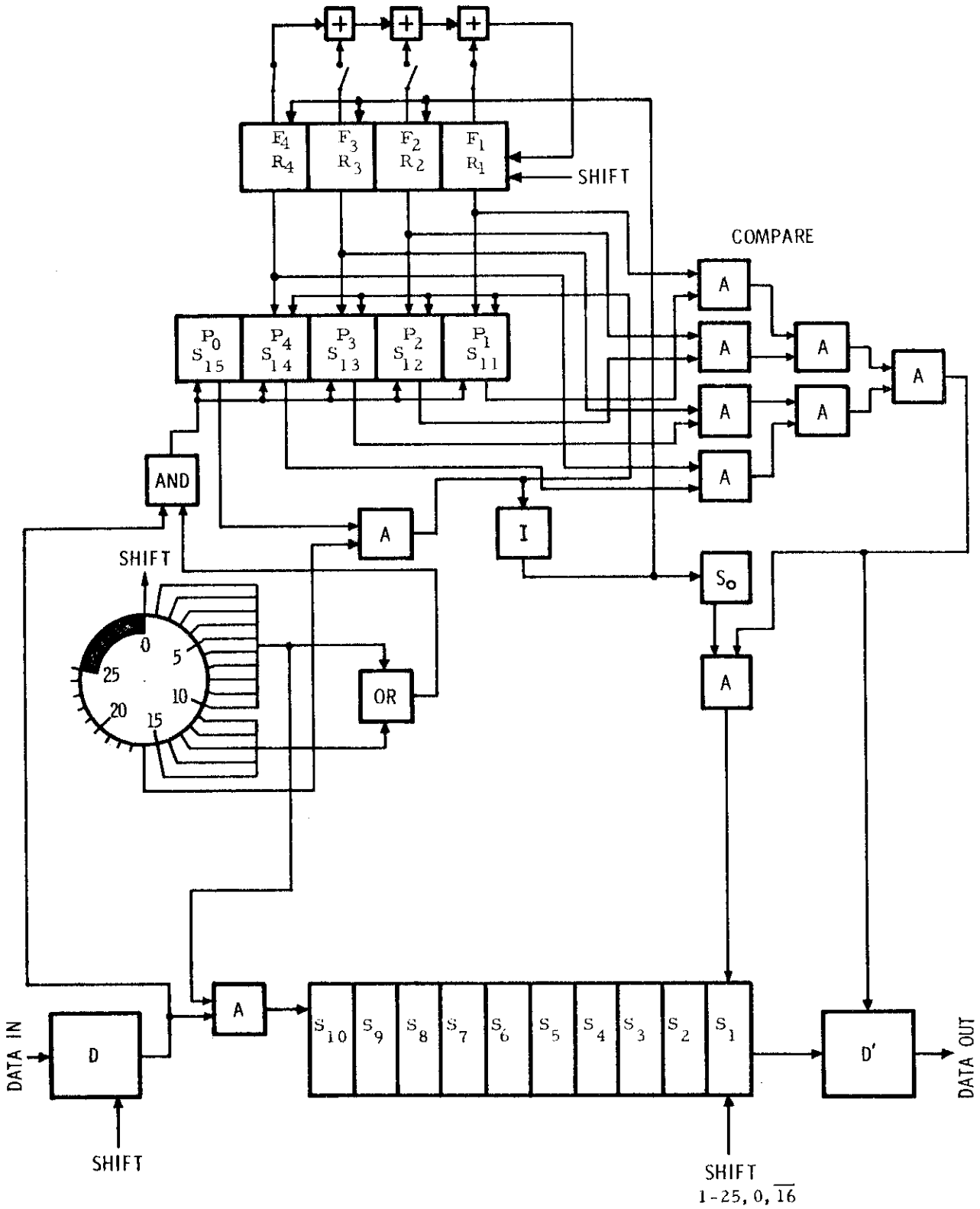The example used here to illustrate double error correction is "J"

FIGURE 3 - DECODER

Transmitted Symbol: 0 1 1 0 1 1 0 1 0 1 0 0 1 1 0

Received Symbol: 0 1 1 0 1 1 0 1 0 1 0 0 0 1 1 0

Output Line

Incoming Line   F or R   P   D   Buffer   Output Line

Compare/correct

FIGURE 4 – Decoding with Single Error

from station "14". The parity checks are computed in Fig. 5. This gives

$$00001 \quad 1110 \quad 000110$$
$$\text{Parity} \quad 14 \qquad J$$

The error introduced in this example is a double adjacent error in positions 7 and 8 making the station identification "13" instead of "14".

The difference between SEC and DAEC shows up in Step 16 in Fig. 6. At the end of step 15, $P_0 = 0$, indicating a double error. Adding P_0 to P_4, P_3, P_2 and P_1 leaves the P register the same. Adding the complement of P_0 to F_4, F_3, and F_2 sets the F register to $F_4 \ F_3 \ F_2 \ F_1 = 0001$ (or $F_1 F_2 F_3 F_4 = 1000$) which means the second column of Fig. 4 (of Ref. 3) is used in double-error-correction. Examining this second column shows that shift 7 from 1000 gives 1010 (or $F_4 F_3 F_2 F_1 = 0101$).

On step 23, the 7th shift, $F_i = P_i$, so D^1 is changed. Back at step 16, the $P_0 = 0$ was inverted at S_0; so S_0 was set to "1", so S_1 is also changed, thus accomplishing the double error correction.

Transistorized Encoder and Decoder

A preliminary estimate of the transistors needed to construct the encoder and decoder of Figures 1 and 3 is summarized in Table VI. These estimates are based upon using "standard" or "proposed standard" transistor logic cards from the IBM Standards Book. CTRL units have been used extensively in these estimates which limit these estimates to bit rates of 20, 000 bits/sec or less.

For ten information bits 62 transistors are required for the encoder. For decoding 82 transistors are required, provided a buffer storage is available for use to store the incoming character during receiving and error-correcting. If such a buffer is not available , a decoder buffer of 20 transistors would have to be added. If half-duplex operation is used, i. e. , transmission is in one direction at a time, the addition of four twelve point relays would switch the logic elements between encoding and decoding modes of operation.

Examination of columns "b" and "m" in Table VI shows that the number of transistors required is approximately proportional to the number of parity bits k. Thus, increasing the information bits from 10 to 2400 requires change from 5 to 13 parity bits, resulting in a change from 82 transistors to 210 transistors for the half duplex operation per terminal.

	Into D									D	F				P				
	10	9	8	7	6	5	4	3	2	1	4	3	2	1	0	4	3	2	1
0											1	1	1	1	0	0	0	0	0	⟵ Start
1	1	1	1	0	0	0	0	1	1	0	1	1	1	0	0	0	0	0	0
2		1	1	1	0	0	0	0	1	1	1	1	0	1	1	0	0	1	0
3			1	1	1	0	0	0	0	1	1	0	1	0	0	0	1	1	1
4				1	1	1	0	0	0	0	0	1	0	1	0	0	1	1	1
5					1	1	1	0	0	0	1	0	1	1	0	0	1	1	1
6						1	1	1	0	0	0	1	1	0	0	0	1	1	1
7							1	1	1	0	1	1	0	0	0	0	1	1	1
8								1	1	1	1	0	0	1	1	0	0	0	1
9									1	1	0	0	1	0	0	1	1	0	0
10										1	0	1	0	0	1	0	1	1	1
11								$P_1 \rightarrow$		1	1	0	0	0	0	0	0	0	0
12								$P_2 \rightarrow$		0	0	0	0	1	0	0	0	0	0
13								$P_3 \rightarrow$		0	0	0	1	1	0	0	0	0	0
14								$P_4 \rightarrow$		0	0	1	1	1	0	0	0	0	0
15								$P_5 \rightarrow$		0	1	1	1	1	0	0	0	0	0	⟵ Start Position
16																			

FIGURE 5 - Encoding of J-14

FIGURE 6 - DOUBLE ERROR CORRECTION
(Same steps as single error decoding.)

A curve of number of transistors versus the number of redundant bits is plotted in Fig. 7. The relationship is linear except at transistion points such as those caused by jumps due to the allowable information bits being a function of a power of two while the number of transistor is related to a function modulo 3.

A limiting economic factor of this error-correcting code is that a buffer storage is required to store the block of digits between decoding and error-correction. In most proposals for data terminal sets a buffer storage unit has been included, so that the buffer need not be included in the incremental cost of adding double-adjacent-error-correction.

Extension to Multiple Adjacent Error-Correcting

A. B. Brown and S. T. Meyers (6) have postulated thr~~ ~~ of the probability that a burst covers h ~~bit~~ Ref. 6 (Fig. 1) is used ~~~~

During steps 17-25 the shift register F_i and the message buffer S_1-S_{10} shift under control of the clock. The message buffer S_1-S_{10} shifts during the complete set of cycles 17-25, 0 so that the full ten information bits will be shifted out through register D^1.

~~~~ ~~longer~~ than two bit times. The ~~~~ a second bit will be in error when the burst ~~~~ is longer than two bit periods will depend upon the ~~~~ation-demodulation system and the symbol being transmitted. Under some conditions a burst length of three bit times will at most cause errors in two out of three bits. Further research will indicate whether DAEC codes are sufficient for IBM needs.

For bursts longer than two ~~bit times~~, be considered:

(1) Use the multiple adjacent error correction (11) code developed by P. Fire at Sylvania Electric Products. This code is an extension of Abramson's code to higher orders of multiple errors.

(2) Use the burst error-correcting code developed at Bell Telephone Laboratories by Hagelbarger. (7-9). These start with a fifty per cent redundancy, but are capable of correcting longer bursts of errors.

(3) Use of the double adjacent error code of Abramson with a longitudinal block check to detect triple adjacent errors and certain other error patterns.
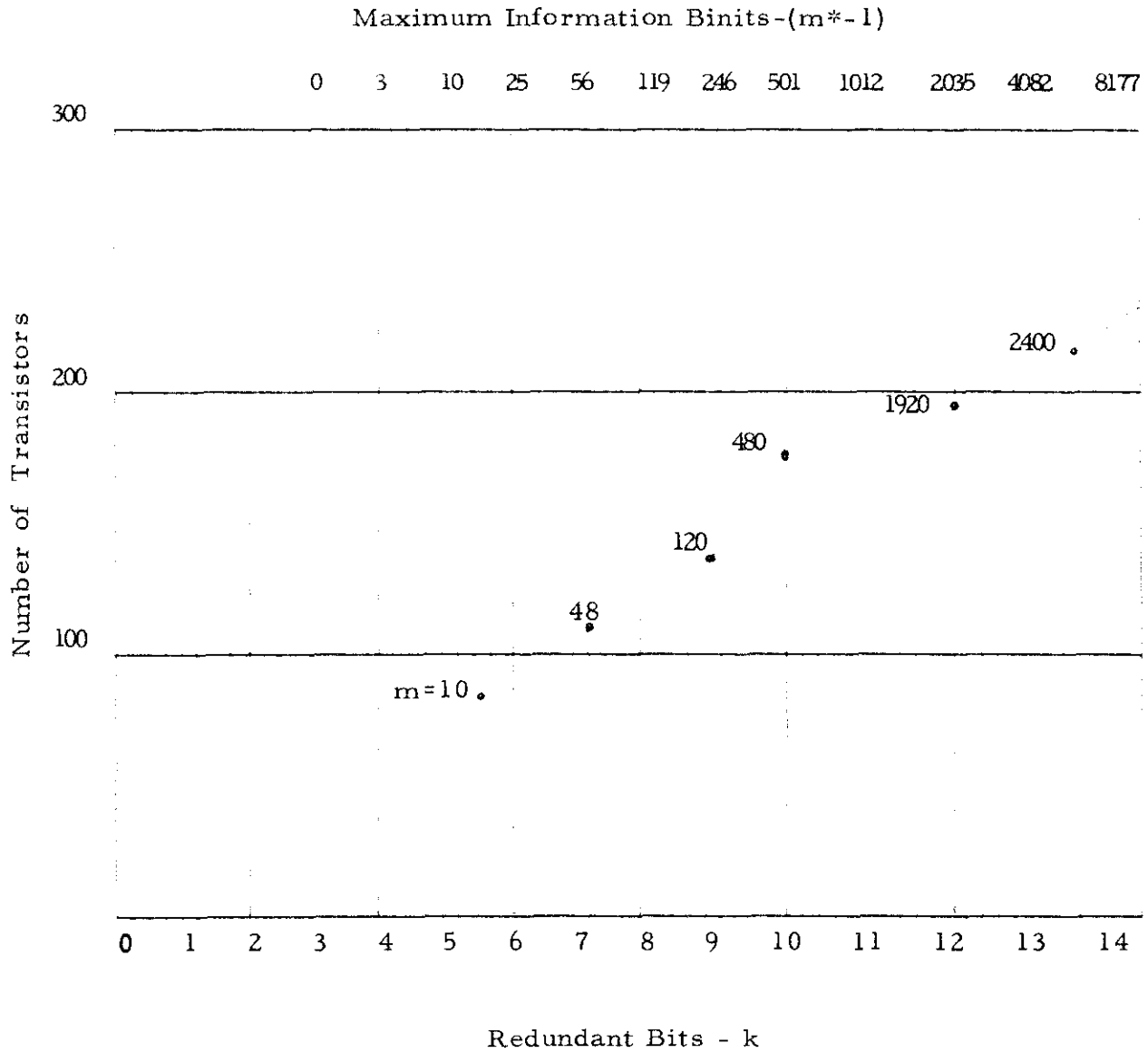
TABLE VI

ESTIMATE OF NUMBER OF TRANSISTORS REQUIRED FOR ENCODING AND DECODING

Bits		Common Elements		Encoder (Separate)				Decoder (Separate)				Combined Encoder Decoder, Switched Half Duplex Total	Decoder Buffer (If not available in Data Set)	Relays for Switching between Encode and Decode in Half-Duplex
Information - m	Parity - k	Sequence Generator	Parity Register	Clock	In/Cut	Gates & Switches	Total	Clock	In/Out	Gates & Switches	Total			
a -	b	c -	d	e-	f-	g-	h	i-	j-	k-	l	m	n	o/p
10-5		14-	10	26-	2-	12-	64	32-	4-	22 -	82	82	20	4/12pt.
48-7		22-	14	38-	2-	17-	93	48-	4-	28-	116	116	96	6/12pt.
120-9		30-	18	59-	2-	21-	130	65 -	4-	30-	147	147	240	6/12pt.
480-10		34-	20	58-	2-	24-	146	68 -	4-	32-	178	178	960	6/12pt.
1920-12		42-	24	71-	2-	28-	167	81 -	4-	45-	196	196	3840	8/12pt.
2400-13		46-	26	77-	2-	21-	172	88-	4-	48-	210	210	4800	8/12pt.

Figure 7

Number of Transistors for Combined Encoder-Decoder
Using CTRL Units for Half Duplex

Maximum Information Binits-(m*-1)

(4) Use a combination of a parity bit on each character
     plus a longitudinal block check such as the seven-bit
     code or the 4-out-of-8 code with a longitudinal check
     on each row. Studies of some of these codes have
     been made in Endicott (12) and Poughkeepsie(13).

     The use of the Hamming (1) Code has been excluded
     on account of the lower efficiency. The OMNICODE
     may be specially suitable to error-correction with-
     in a computer, but the large redundancy makes it
     efficient for data transmission . (14-15)

     It may be possible to develop an extension of
     Abramson's DAEC code which would correct for a
     set of burst error patterns of the most probable
     distribution. The work of B. Elspas at Stanford
     Research Institute has resulted in an alternative
     logic for performing the encoding and decoding in
     N. Abramson's SEC-DAEC code (16) which may
     lead to a simplified system.

Conclusions

The following of these detailed steps in the operation of an encoder
and decoder for N. M. Abramson's SEC-DAEC code demonstrates the
feasibility and simplicity of the system. The small number of parity
bits required and the simple logic circuits required make it potentially
attractive for data transmission use. The estimated number of transis-
tors per terminal for the error-correcting logic is relatively low.

The unknown factor in evaluating this code is the statistics of
error bursts. The American Telephone & Telegraph Company is mak-
ing statistical analyses of error bursts on their lines using their FM
Data Subset. When this data is available, it may give an indication of
the error pattern distribution. If triple adjacent errors or double
errors separated by a correct bit are significant, the advantages of
this code diminish. However, there are possibilities of deriving
another code from this SEC-DAEC code that would correct other error
patterns more efficiently than the alternative code listed.

APPENDIX - SUPPLEMENTARY NOTES

In comparing different codes it is significant to know the "minimum distance" or minimum number of digits in which each character differs from all the other characters or code points. This minimum distance can be considered as the minimum number of edges of an n-dimensional cube between any two code points when represented by an n-dimensional space. An elementary illustration of the separation between code points is given by Colin Cherry.

The $(AC)^2$ - SEC-DAEC code of N. Abramson has a minimum distance of four. The term $(AC)^2$ means that the code is "almost complete" and that it is an "all-check" code. The term "complete" means that the inequality relating m and k is an equality, similarly to the term "close-packed" used by Shapiro and Slotnik. Since complete or close-packed codes generally do not exist except for trivial or special cases, practical codes which are "almost complete" are developed by calculating m* from the equality of ref. 1, eq. 3, and then using $m = m* - 1$, which allows one less information digit than the impossible complete code for the same k. The term "all check" means that one of the parity bits provide a check on all the bits.

For convenience in understanding the double-adjacent error correcting code, a sample case of $m = 3$, $k = 4$ is illustrated in Table VII. The X's indicate which bits each parity bit checks. Then a table of distances is prepared from this and is shown in Table VIII. In this table the distance between the information and parity sub set are shown separately and are added together to get the total distance of each code point from all the others. In this case the minimum distance is equal to four and in fact the distance between every separate point is four. A Hamming code with minimum distance of four could provide single error correction with double-error detection or it could provide detection only of single, double, and triple errors.

A table of checking numbers for single, double, triple, and quadruple adjacent errors for the parity table of Table VII is shown as Table IX. From this one can see that if a system was designed so that no single errors occurred the SEC - DAEC code could be used to correct double adjacent errors and triple adjacent errors. If we count the check numbers for single and double-adjacent errors, we see they add up to fourteen of the possible sixteen. The checking number 0000 means no errors occurred of the type for which the code is designed to correct.

In the third section of this report it was stated that the remaining unused check number: all one's indicates that an odd number of errors greater than one occurred. The possible triple errors are shown in Table X. The checking number of all one's occurs for six of the 35 possible triple errors. The other possible triple errors would give checking numbers which would be interpreted as single errors.

TABLE VII

Parity Table for $m = 3$, $k = 4$

	D_1	D_2	D_3	P_1	P_2	P_3	P_0
P_1	x	x		x			
P_2		x	x		x		
P_3			x	x		x	
P_0	x	x	x	x	x	x	x

TABLE VIII

Table of Distances for SEC-DAEC Code   m= 3, k = 4

	000/0000	001/0111	010/1110	011/1001	100/1011	101/1100	110/0101	111/0010
000/0000	0							
001/0111	1+3=4	0						
010/1110	1+3=4	2+2=4	0					
011/1001	2+2=4	1+3=4	1+3=4	0				
100/1011	1+3=4	2+2=4	2+2=4	3+1=4	0			
101/1100	2+2=4	1+3=4	3+1=4	2+2=4	1+3=4	0		
110/0101	2+2=4	3+1=4	1+3=4	2+2=4	1+3=4	2+2=4	0	
111/0010	3+1=4	2+2=4	2+2=4	1+3=4	2+2=4	1+3=4	1+3=4	0

TABLE IX

CHECKING NUMBER FOR SINGLE, DOUBLE, TRIPLE and QUADRUPLE ADJACENT ERRORS

Error In	Check No.	Error In	Check No.
D_1	1001	$D_3 P_1 P_2$	1001
D_2	1101	$P_1 P_2 P_3$	1101
D_3	0111	$P_2 P_3 P_0$	0111
P_1	1011	$P_3 P_0 D_1$	1011
P_2	0101	$P_0 D_1 D_2$	0101
P_3	0011	$D_1 D_2 D_3$	0011
P_0	0001	$D_2 D_3 P_1$	0001
$D_1 D_2$	0100	$D_2 D_3 P_1 P_2$	0100
$D_2 D_3$	1010	$D_3 P_1 P_2 P_3$	1010
$P_3 P_1$	1100	$P_1 P_2 P_3 P_0$	1100
$P_1 P_2$	1110	$P_2 P_3 P_0 D_1$	1110
$P_2 P_3$	0110	$P_3 P_0 D_1 D_2$	0110
$P_3 P_0$	0010	$P_0 D_1 D_2 D_3$	0010
$P_0 D_1$	1000	$D_1 D_2 D_3 P_1$	1000

TABLE X

TRIPLE ERRORS

Error In	Check No
$D_1 D_2 D_3$	0011
$D_1 D_2 P_1$	1111
$D_1 D_2 P_2$	0001
$D_1 D_2 P_3$	0111
$D_1 D_2 P_0$	0101
$D_1 D_3 P_1$	0101
$D_1 D_3 P_2$	1011
$D_1 D_3 P_3$	1101
$D_1 D_3 P_0$	1111
$D_1 P_1 P_2$	0111
$D_1 P_1 P_3$	0001
$D_1 P_1 P_0$	0011
$D_1 P_2 P_3$	1111
$D_1 P_2 P_0$	1101
$D_1 P_3 P_0$	1011
$D_2 D_3 P_1$	0001
$D_2 D_3 P_2$	1101
$D_2 D_3 P_3$	1001
$D_2 D_3 P_0$	1011
$D_2 P_1 P_2$	0011
$D_2 P_1 P_3$	0101
$D_2 P_1 P_0$	0111
$D_2 P_2 P_3$	1011
$D_2 P_2 P_0$	1001
$D_2 P_3 P_0$	1111
$D_3 P_1 P_2$	1001
$D_3 P_1 P_3$	1111
$D_3 P_1 P_0$	1101
$D_3 P_2 P_3$	0001
$D_3 P_2 P_0$	0011
$D_3 P_3 P_0$	0101
$P_1 P_2 P_3$	1101
$P_1 P_2 P_0$	1111
$P_1 P_3 P_0$	1001
$P_2 P_3 P_0$	0111

REFERENCES

1. R. W. Hamming, "Error Detecting and Error Correcting Codes," B.S.T.J., Vol. 20, April 1950, pp. 147-160.

2. D. Slepian, "A Class of Binary Signalling Alphabets," B.S.T.J., Vol. 35, 1956, pp. 203-239.

3. N. M. Abramson, "A Class of Systematic Codes for Non-Independent Errors," Stanford Electronic Laboratories, Technical Report No. 51, Dec. 30, 1958.

    A more complete bibliography of the preceding literature on coding and binary shift register is included in this report.

4. N. M. Abramson, "Single Error and Double Adjacent Error Encoder," IBM Disclosure No. 80780.

5. N. M. Abramson, "Single Error and Double Adjacent Error Decoder," IBM Disclosure No. 80779.

6. A. B. Brown and S. T. Meyers, "Evaluation of Some Error Correcting Methods Applicable to Digital Data Transmission," IRE Nat. Conv. Record, Vol. 6, Part 4, 1958, pp. 37-55, Fig. 1.

7. "A New Correcting Code for Bursts of Errors," Bell Laboratories Record, Jan. 1959, pp. 33-34.

8. "Error-Correction Code Eliminates Bursts of Errors," Electrical Design News, March 4, 1959, p. 7.

9. "Digital Error Corrector," Wire and Radio Communications, March 1959, p. 26.

10. F. B. Wood, "Theoretical Error Frequency in Different Codes," Memorandum SJA-16, July 5, 1956, p. 11.

11. Philip Fire, "A Class of Multiple-Error-Correcting Binary Codes for Non-Independent Errors," Mountain View, California: Sylvania Reconnaissance Systems Laboratory, Report RSL-E-2, March 1959. (This report is also being issued by Stanford University.)

12. M. P. Marcus, G. J. Saxenmeyer, M. J. Schatzoff and L. H. Tung, "Coding Study Report," Endicott: Product Development Laboratory, Coding Study Group, Preliminary copy, Jan. 28, 1959.

13. W. E. Brandt, "Performance of Error Detecting and Correcting Codes," Poughkeepsie, IBM-SEPD Report, TN 09.01012.001, August 1, 1958.

14. C. M. Davis, J. M. MacDonald, "OMNICODE," Poughkeepsie Product Development, Reliability and Serviceability Bulletin No. 14, June 23, 1958. Code: TN 00.01010.286.

15. J. E. MacDonald, C. M. Davis, "An Improved OMNICODE Adder." Poughkeepsie Product Development, Reliability and Serviceability Bulletin, No. 23, Oct. 6, 1958. Code: TN 00.01010,316.

16. Norman Abramson (Stanford University) and Bernard Elspas (Stanford Research Institute), "Double-Error-Correction Encoders and Decoders for Non-Independent Binary Errors." (Manuscript, March 1959.)

17. Colin Cherry, On Human Communication, N.Y.: Wiley (1957) p. 187, Fig. 5,6.

Distribution List:

N. M. Abramson
A. G. Anderson
M. M. Astrahan
P. Beebe (SEPD, Poughkeepsie)
B. J. Bennett (PD, San Jose)
C. A. Bergfors (3 copies)
W. E. Brandt (Poughkeepsie)
F. C. Chiang
W. A. Christopherson
C. M. Davis (PD, Poughkeepsie)
W. E. Dickinson
A. B. Fontaine (Lamb Estate)
M. M. Gibson (PD, San Jose)
W. A. Goddard
R. L. Haug
A. S. Hoagland
E. Hopner
C. J. Hoppel
R. B. Johnson
W. J. Johnson, Jr.
R. E. Kaufmann
A. Kusnick (SEPD, Poughkeepsie)
R. E. Lawhead
F. A. Litz
J. E. MacDonald (PD, Poughkeepsie)
M. P. Marcus (PD, Endicott)
H. G. Markey
J. A. McLaughlin
C. M. Melas
L. O. Nippe (PD, Poughkeepsie)
R. W. Porter (SEPD, San Jose)
E. A. Quade (WHQ)
L. E. Rickard
B. G. Riddell
J. T. Smith
J. B. Sponsler
L. A. Tate (PD, Poughkeepsie)
R. M. Walker (Watson Lab.)
W. Wittenberg (SEPD, Poughkeepsie)
F. B. Wood