ERROR DETECTION AND CORRECTION

by Brian Patrick McArdle

Telecommunications channels are not so reliable that data can be passed over them without corruption. Since data communications has become important, satisfactory methods to identify and, if possible, correct errors have been developed.

CONSIDER the transfer of data as in Fig 1 between modems over a telephone line or radio link. The channel is corrupted by noise, which could result in a receiver detecting a 1 instead of a 0 or vice versa. If this were to happen in a number of places in a message the effect could be disastrous: the message might be completely unintelligible. Obviously, there is a need to be able to identify incorrect data bits. The subject is know as **error detection and correction**.

Parity

Parity is a familiar concept and is explained in most technician text books. Consider a block of seven bits that is given an additional bit known as the parity bit, so that the total block becomes eight bits. The term 'parity' refers to the entire new block and there are two categories:

(a) Even parity means that a block has an even number of 1s. For example, if the data is 1 0 1 0 1 0 1, which has four 1s and three 0s, the parity bit would be 0 giving four 1s and four 0s. The block would become 0 1 0 1 0 1 0 1 with the parity bit as the MSB wich is the usual location.

(b) Odd parity if it has an odd number of 1s. The previous example would be 1 1 0 1 0 1 0 1 with five 1s and three 0s.

The 8-bit ASCII code has seven data bits and a parity bit. Thus, the ASCII alphabet has a total of $2^7 = 128$ different symbols. The 8th bit for each symbol is chosen to give even parity. If an error occurs in transmission as in Fig. 1, a parity check would reveal an erroneous symbol. The receiver could ignore the particular symbol, provided that such an occurrence did not happen too often, or, alternatively, request a retransmission.

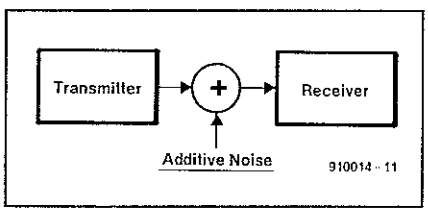


Fig 1 Communications channel

In reality, a parity check on each symbol gives very limited protection. The obvious problem is that two errors within a symbol could cancel each other. Alternatively when an erroneous symbol is identified, a receiver still cannot deduce the incorrect bit within the block and, consequently, correction is out. Therefore, although the addition of a parity bit into a block is the basis of error detection, it is rarely satisfactory. To identify an incorrect symbol is one thing, but to correct an error in one bit is quite another. The next section explains a method to identify and correct single-bit errors.

Block codes

In the following arrangement a block consists of data bits plus check bits. For example, a block of size n would have t data bits and $k = n - t$ check bits. The check bits are derived from the data bits. In reality, they are parity bits that are determined by various linear combinations of data bits. For the present, it is assumed that only one error will occur in each block. The significance of this point will be discussed in more detail later on. For a block of size n, there is a total of ${}^nC_1 = n$ errors where the corruption is limited to one bit per block. The k check bits can have $(2^k - 1)$ different combinations. This means that at most $(2^k - 1)$ errors can be detected. Hence, there is a requirement that $(2^k - 1) \geq n$ in the values that can be chosen for t and k. For example, if $k = 3$, $(2^k - 1) = 7$ which means that $n = 7$ and $t = 4$. Therefore three check bits can check up to four data bits only. The coded block would be $[d_4 d_3 d_2 d_1 c_3 c_2 c_1]$ or some transposition of this order.

In mathematical terms, the encoding operation can be described with the use of a matrix as follows:

$$\begin{bmatrix} 1\,0\,0 & \cdots & 0 \\ 0\,1\,0 & \cdots & 0 \\ 0\,0\,1 & \cdots & 0 \\ \vdots & & \vdots \\ 0\,0\,0 & \cdots & 1 \\ g_{t+1,1} & \cdots & g_{t+1,t} \\ \vdots & & \vdots \\ g_{n,1} & \cdots & g_{n,t} \end{bmatrix} \begin{bmatrix} d_t \\ d_{t-1} \\ \vdots \\ d_1 \end{bmatrix} = \begin{bmatrix} d_t \\ d_{t-1} \\ \vdots \\ d_1 \\ c_k \\ \vdots \\ c_1 \end{bmatrix}$$

$$c_k = \sum_{j=1}^{t} g_{t+1,j}\, d_{t+1-j} \bmod 2 \quad \text{[Eq 1]}$$

The matrix has n rows and t columns, which correspond to the lengths of the coded and data blocks respectively. The top part is an $(t \times t)$ identity matrix that does not require any explanation. The bottom part is a $(k \times t)$ matrix that generates c_1 to c_k bits from the d_t to d_1 bits. All the coefficients are 0 or 1 and the arithmetic is modulo 2. The requirement is that, if an error occurs in a data bit, the check bits will be able to identify the particular bit in question, which can be corrected. The correction process is simply to replace a 1 by a 0 or vice versa.

The next step is to devise a simple method to identify the exact location of an incorrect bit within the block. The **Hamming Code** is examined as a suitable example because it is widely known. For instance, a block of seven bits consists of four data bits and three check bits in the following format:

position:	7	6	5	4	3	2	1
bit:	d_7	d_6	d_5	c_4	d_3	c_2	c_1

in which the data and check bits are indicated as d and c respectively and d_7 as the most significant bit (MSB). As would be expected, the position of these bits within the overall block is very significant. The positions 3, 5, 6, 7 can be represented as various combinations of 1, 2, 4. Thus if d_3 is incorrect, c_1 and c_2 will not be validated and so on. The combinations that give c_1, c_2 and c_4 in terms of d_3, d_5, d_6 and d_7 are chosen accordingly as shown in Appendix A and the encoding operation is given by:

$$\begin{bmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 1\,1\,1\,0 \\ 0\,0\,0\,1 \\ 1\,1\,0\,1 \\ 1\,0\,1\,1 \end{bmatrix} \begin{bmatrix} d_7 \\ d_6 \\ d_5 \\ d_3 \end{bmatrix} = \begin{bmatrix} d_7 \\ d_6 \\ d_5 \\ c_4 \\ d_3 \\ c_2 \\ c_1 \end{bmatrix}$$

The matrix is know as the GENERATOR

MATRIX. The receiver verifies the block by using the PARITY CHECK MATRIX (see Appendix A) which should produce all zeroes. In simple terms the receiver generates c_1, c_2 and c_4 from the received data bits. If the generated c_1 and c_2 do not agree with the actual received values d_3 is incorrect. For example [1 0 1 1] becomes [1 0 1 0 1 0 1]. If d_5 is corrupted such that [1 0 0 0 1 0 1] is received the generated check bits are $c_1 = 0$, $c_2 = 0$, and $c_4 = 1$. Thus c_1 and c_1 do not agree with the detected values, which indicates that d_5 should not be 0 but 1. The same type of result occurs for any other error—provided there is only one error. If two errors had occurred with [1 0 1 1 0 0 1] detected a check would indicate that d_7 was incorrect. Consequently the block would be corrected to [0 0 1 1 0 0 1] which now has three errors with two in the four data bits. The importance of the original assumption that the code would only detect and correct an error in one bit per block is now fully clear. However, the overall method is typical of block codes. The various positions for a code with four check bits that can correct up to 11 data bits are given in Appendix B.

Cyclic codes

Cyclic codes are similar to block codes but there are important mathematical differences. A code word of length n bits is represented as a polynomial of degree $(n-1)$ as follows:

$$w(x) = [w_n\ w_{n-1}\ w_{n-2} \quad w_1] =$$

$$= w_n x^{n-1} + w_{n-1} x^{n-2} + \quad w_2 x + w_1 \qquad \text{[Eq 2]}$$

where w_n is the most significant bit (MSB). It is generated by a polynomial known as the GENERATOR POLYNOMIAL, $g(x)$ as follows:

$$w(x) = c(x)\ g(x) \bmod (x^n + 1) \qquad \text{[Eq 3]}$$

Most books on coding write the modulus as $(x^n - 1)$. In this case, the mathematics refer directly to digital electronics. Actually $g(x)$ is a factor of $(x^n + 1)$ but the other term, $c(x)$ does not represent the data as might be expected. In mathematical terms, $w(x)$ is the product modulo $(x^n + 1)$ of two polynomials with addition modulo 2 between the coefficients of the various terms. The exact coding mechanism will be shown later on.

A CHECK POLYNOMIAL, $h(x)$, has the same significance as in the previous section with

$$h(x)\ g(x) = x^n + 1 \qquad \text{[Eq 4]}$$

which means that $h(x)$ is the other factor of $(x^n + 1)$. If a received code word $w(x)$ has no error

$$h(x)\ w(x) = 0 \qquad \text{[Eq 5]}$$

Consider the example in the previous section with $n = 7$. In this section, this translates to $x^7 + 1 = 0 \bmod 2$. The factors are $(x^3 + x + 1)$ and $(x^4 + x^2 + x + 1)$ and we will use

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

and

$$h(x) = x^4 + x^2 + x + 1$$

as per equation 4. At this stage, the term 'cyclic' requires some explanation. Let $w(x)$ in equation 2 be rewritten in the form

$$w(x) = c_1 g(x) + c_2 x g(x) + c_3 x^2 g(x) +$$

$$\ldots + c_7 x^6 g(x) \qquad \text{[Eq 6]}$$

The various terms as shown in Appendix C consist of a set of polynomials that is generated by shifting $g(x)$ as in a shift register. Thus $w(x)$ may be considered as a linear combination of the seven states of $g(x)$ with x as a 'shift'. The full set of code words is generated by $g(x)$ and hence its name. The reader is referred to the topic of Cyclic Groups in Group Theory for the appropriate mathematical background. If the data is [1 0 1], an estimate of $w(x)$ which assumes that the check bits are all zeros is

$$w(x) = [1\ 0\ 1\ 0\ 0\ 0] = x^6 + x^5 + x^3$$

[Eq 7]

However, $g(x)$ does not divide $w'(x)$ evenly. There is a remainder of 1 in the least significant bit (LSB) position. The correct $w(x)$ is

$$w(x) = [1\ 0\ 1\ 0\ 0\ 0\ 1] = x^6 + x^5 + x^3 + 1$$

[Eq. 8]

in order to satisfy the mathematical conditions. The procedure may be written as a matrix as follows:

$$\begin{bmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \\ 1&1&1&0 \\ 0&1&1&1 \\ 1&1&0&1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

the derivation of which is in Appendix D. The PARITY CHECK MATRIX to authenticate the code word is explained in Appendix E. If the block was corrupted to [1 1 0 1 1 0 1], this new $w(x)$ on division by $g(x)$ would have a remainder of x^2. This indicates an error two shifts away from the LSB and the particular bit can be corrected. However, to correct two or more errors, a BCH code which uses a particular type of generator polynomial is required and this topic is outside the scope of this article.

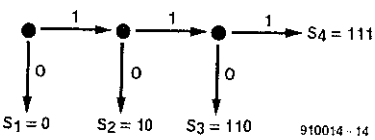
Huffman codes

Huffman codes are not for error detection and correction but deserve special mention because they may be used in conjunction with error detection and correction codes. The most important point is that the data blocks do not have a fixed length. The reasons for this structure structure lie in the nature of language. The symbol E in ASCII is 1 0 0 0 1 0 1 and Z is 1 0 1 1 0 1 0. Each symbol is represented by seven bits excluding the parity bit. However, E occurs 130 times more often in normal text than Z. It has been estimated that normal English is 70% redundant because of these statistical properties. Consequently, the total number of bits required for a full message could be greatly reduced if the common symbols were represented by short blocks and the more uncommon ones by larger blocks. A simple example is Morse code, in which E is one dot whereas Z is two dashes followed by two dots. Huffman codes utilize this property of redundancy to considerable advantage.

If an alphabet has m symbols the average number of bits per symbol is

$$\bar{r} = \sum_{i=1}^{m} p_i r_i \qquad \text{[Eq 9]}$$

where p_i is the probability of occurrence of the ith symbol, which is represented by r_i bits. A Huffman code seeks to keep $\bar{r}$ as small as possible by pairing the symbol with the highest probability with the shortest block. Consider an example of a set $\{S_1\ S_2\ S_3\ S_4\}$ with four symbols. Normally each symbol would be represented by a two-bit block, such as: $S_1 = 0\,0$, $S_2 = 0\,1$, $S_3 = 1\,0$, $S_4 = 1\,1$.



A Huffman code would adopt a branch type structure as shown above. In this design, S_1 is represented by only one bit, but S_3 has three bits. If the probabilities are $p_1 = 1/2$, $p_2 = 1/4$, $p_3 = 1/6$, and $p_4 = 1/12$, application of equation 9 gives

$$r = 1(1/2) + 2(1/4) + 3(1/6) + 3(1/12)$$

$$= 1.75 \text{ bits/symbol} \qquad \text{[Eq 10]}$$

which is a reduction of 25%. If the probabilities were assigned in reverse order $\bar{r} = 2.67$ which is an increase of 67%. Hence, the importance of pairing the largest probability with the shortest block. In the special case where the two values for $\bar{r}$ turned out to be equal, the assignment with the lowest variance would be taken.

The entire area of Huffman codes is quite involved and the reader is referred to the various text books in the references for further information. Its advantage is in reducing the total number of data bits. The sequence of data bits can be broken into blocks and encoded as under "Block codes" and Cyclic codes". The receiver simply reverses this process by first decoding the error detection and correction code followed by the Huffman code.

Implementation

The electronic implementation of the codes discussed is relatively straightforward once the actual coding mechanism is understood. Figure 2 shows a digital electronic circuit, which is essentially a shift register, to implement the example under 'Cyclic codes'. The arrangement is to generate the full code word of data and check bits. The four stages in the register correspond to the data block of four bits, which in turn generates the three check bits. The switch may be an AND gate with one input tied high (logic 1) or low (logic 0) as appropriate. The other input is simply the output of the bistable (flip flop). The addition modulo 2 represented by the ⊕ in the diagram is simply and EXOR (exclusive OR) gate. The only change is that three different switch arrangements (open = 0 and closed = 1) are required to generate the three check bits. This corresponds to the three linear combinations in mathematics. Thus the complete circuit for the coding operation could consist of three shift registers with fixed switch positions in parallel or one shift register with a mechanism to apply three different arrangements of the switches. Alternatively implementation could be by software that is to write a program using the AND and EXOR logic operations of a microprocessor. This method will probably become more common—especially where the codes are more complex than these simple examples.

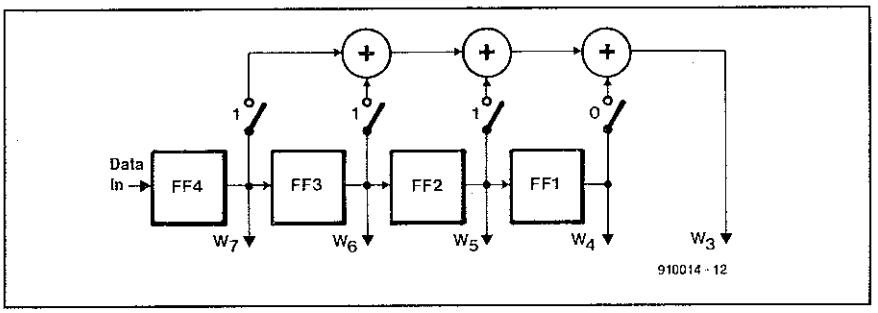


Fig. 2. Shift register as a Cyclic Encoder. The data block [$w_7w_6w_5w_4$] is shifted into the register. The switches represent the terms of the matrix, such as 1 1 1 0 to generate w_3. The switch is closed for 1 and open for 0. The same arrangement with different switch settings would be required to generate w_2 and w_1.

Figure 3 shows the circuit to check the parity bit generated by the first row of the PARITY CHECK MATRIX (as explained in Appendix E) applied to a code word. The seven stages correspond to the seven bits in the block. The other rows of the matrix simply require different arrangements of the switches. Both circuits are straightforward.

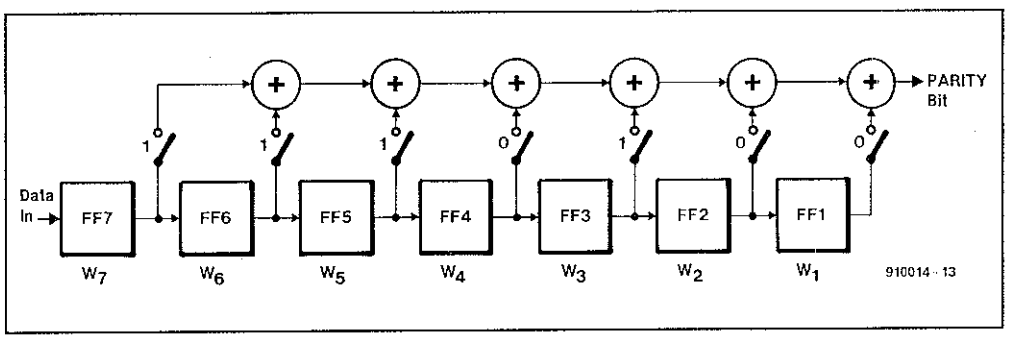


Fig. 3. Shift register to check a code word generated by the circuit in Fig. 2. If the PARITY BIT is 1, one of the seven bits in the shift register that has a closed switch is incorrect. For the first row of the PARITY CHECK MATRIX (see Appendix E) the switches are at 1 1 1 0 1 0 0. For successive rows, the appropriate settings must be used.

Conclusions

In this article I have attempted to explain the basis of error detection. It is not a detailed analysis and only deals with the correction of single errors. Multiple errors of two or more bits per block are not considered as these require more complex solutions. Nevertheless, it should provide an introduction for students who are familiar with digital electronics, but have not previously studied this tricky topic. An interesting point is the increasing connection between Modern Algebra and Electronics. Twenty years ago most engineers and technicians would not expect to use Modern Algebra. The reader is referred to the references for further study.

References

Coding and Information Theory by R W. Hamming. Prentice-Hall. 1980.

Information Theory and its Engineering Applications by D A. Bell. Pitman. 1968.

Codes and Cryptography by D. Welsh. Oxford University Press. 1988.

Telecommunications Engineering by H G. Brierly. Edward Arnold. 1986.

Appendix A

$c_4 = (d_7 + d_6 + d_5) \bmod 2$

$c_2 = (d_7 + d_6 + d_3) \bmod 2$

$c_1 = (d_7 + d_5 + d_3) \bmod 2$

If d_7 is incorrect, c_1, c_2 and c_4 do not agree with calculated values.

If d_6 is incorrect, c_2 and c_4 do not agree with calculated values.

If d_5 is in correct, c_1 and c_4 do not agree with calculated values.

If d_3 is incorrect, c_1 and c_2 do not agree with calculated values.

$$\begin{bmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \\ 1&1&1&0 \\ 1&1&0&1 \\ 1&0&1&1 \end{bmatrix} \begin{bmatrix} d_7 \\ d_6 \\ d_5 \\ d_3 \end{bmatrix} = \begin{bmatrix} d_7 \\ d_6 \\ d_5 \\ d_3 \\ c_4 \\ c_2 \\ c_1 \end{bmatrix} \qquad \begin{bmatrix} 1&1&1&0&1&0&0 \\ 1&1&0&1&0&1&0 \\ 1&0&1&1&0&0&1 \end{bmatrix} \begin{bmatrix} d_7 \\ d_6 \\ d_5 \\ d_3 \\ c_4 \\ c_2 \\ c_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Generator matrix — Encoded block — Parity check matrix

Appendix B

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

Bit: $d_{15}\,d_{14}\,d_{13}\,d_{12}\,d_{11}\,d_{10}\,d_9\;c_8\;d_7\;d_6\;d_5\;c_4\;d_3\;c_2\;c_1$

The check bits are thus at positions 1 2 4 and 8

Appendix C

$x^n = 1 \bmod 2$

$x^n + 1 = 0 \bmod 2$

For $n = 7$ and $g(x) = x^3 + x + 1$, representing the seven-bit state [0 0 0 1 0 1 1] of a shift register, the operation using shift left rather than shift right is

0 0 0 1 0 1 1	$g(x) = x^3 + x + 1$	
0 0 1 0 1 1 0	$xg(x) = x^4 + x^2 + x$	
0 1 0 1 1 0 0	$x^2g(x) = x^5 + x^3 + x^2$	
1 0 1 1 0 0 0	$x^3g(x) = x^6 + x^4 + x^3$	$x^7 = 1$
0 1 1 0 0 0 1	$x^4g(x) = x^5 + x^4 + 1$	
1 1 0 0 0 1 0	$x^5g(x) = x^6 + x^5 + x$	
1 0 0 0 1 0 1	$x^6g(x) = x^6 + x^2 + 1$	
0 0 0 1 0 1 1	$x^7g(x) = x^3 + x + 1$	

Thus, multiplication of $g(x)$ by x is equivalent to a left shift of one step of the shift register. The initial state is reproduced after seven steps.

Appendix D

$w(x) = c(x)\,g(x) = [w_7\,w_6\,w_5\,w_4\,w_3\,w_2\,w_1]$

where

$c(x) = c_4x^3 + c_3x^2 + c_2x + c_1$

and

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

whence

$$c(x)\,g(x) = c_4x^6 + c_3x^5 + (c_4+c_2)x^4 + (c_4+c_3+c_1)x^3 + (c_3+c_2)x^2 + (c_2+c_1)x + c_1$$

$w_7 = c_4$
$w_6 = c_3$
$w_5 = c_4 + c_2$ $\quad c_2 = w_7 + w_5$
$w_4 = c_4 + c_3 + c_1$ $\quad c_1 = w_7 + w_6 + w_4$
$w_3 = c_3 + c_2$ $\quad w_3 = w_7 + w_6 + w_5$
$w_2 = c_2 + c_1$ $\quad w_2 = w_6 + w_5 + w_4$
$w_1 = c_1$ $\quad w_1 = w_7 + w_6 + w_4$

$$\begin{bmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \\ 1&1&1&0 \\ 0&1&1&1 \\ 1&1&0&1 \end{bmatrix} \begin{bmatrix} w_7 \\ w_6 \\ w_5 \\ w_4 \end{bmatrix} = \begin{bmatrix} w_7 \\ w_6 \\ w_5 \\ w_4 \\ w_3 \\ w_2 \\ w_1 \end{bmatrix}$$

Remember that the arithmetic is modulo 2

Appendix E

$h(x) = x^4 + x^2 + x + 1$ PARITY CHECK POLYNOMIAL

$w(x) = w_7x^6 + w_6x^5 + w_5x^4 + w_4x^3 + w_3x^2 + w_2x + w_1$

$$h(x)\,w(x) = x^6(w_7 + w_6 + w_5 + w_3) + x^5(w_6 + w_5 + w_4 + w_2) + x^4(w_5 + w_4 + w_3 + w_1) + x^3(w_7 + w_4 + w_3 + w_2) + x^2(w_6 + w_3 + w_2 + w_1) + x(w_7 + w_5 + w_2 + w_1) + (w_7 + w_6 + w_4 + w_1)$$

$$\begin{bmatrix} 1&1&1&0&1&0&0 \\ 0&1&1&1&0&1&0 \\ 0&0&1&1&1&0&1 \\ 1&0&0&1&1&1&0 \\ 0&1&0&0&1&1&1 \\ 1&0&1&0&0&1&1 \\ 1&1&0&1&0&0&1 \end{bmatrix} \begin{bmatrix} w_7 \\ w_6 \\ w_5 \\ w_4 \\ w_3 \\ w_2 \\ w_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

From equation 6, the PARITY CHECK POLYNOMIAL applied to a correct code word should give 0. The PARITY CHECK MATRIX should give the zero vector.

Appendix F

An important parameter, known as the **Hamming Distance**, was not covered under 'Block codes' as it was not required. Consider the (7, 4) Hamming code. Any two code words differ in at least three bits. This minimum distance is known as the Hamming Distance. If the distance were only two, an error could only be detected but not corrected. Detection of a double error would require a distance of four.