An Encoding Scheme for
Word Stuffing

by

M. M. Buchner, Jr. and E. J. Hronik
Bell Telephone Laboratories, Incorporated
Holmdel, New Jersey

ABSTRACT

The effectiveness of word stuffing for synchronization depends upon the ability to distinguish the stuff words from the data words at the destination and, thus, delete correctly the stuff words. If all input sequences are permitted, the data words must be encoded before stuffing occurs so that the stuff word can be distinct from the data words. Previously, a code was given in which a single transmission error could change a data word into the stuff word. In this paper, we give the code that maximizes the minimum distance between the stuff word and any data word. It follows that the probability of character synchronization loss because of confusion between data and stuff words due to transmission errors is considerably reduced. Some implementations are given which suggest that the increase in distance can be achieved at a small cost in hardware.

Some other encoding schemes, for which no immediate applications are known, are presented to show the variety of codes that can be designed in hopes of stimulating new applications.

An Encoding Scheme for
Word Stuffing

by

M. M. Buchner, Jr. and E. J. Hronik
Bell Telephone Laboratories, Incorporated
Holmdel, New Jersey

I. Introduction

When a digital communication network is used for data transmission, it may be necessary to adjust transmission rates within the network to achieve synchronization. Word stuffing [1] is a technique that can be used for this purpose. The basic idea is to group the transmitted bits into words which we call data words. The data words are simply formed for transmission and are not related to any word structure that may exist in the customer's data stream. Stuff words, which are distinguishable from the data words, are inserted into the stream of data words at the transmitter. Transmission rates are adjusted by inserting or deleting stuff words. At the destination, the stuff words are deleted whereas the data words are delivered to the customer. The effectiveness of word stuffing depends upon the ability to distinguish the stuff words from the data words at the destination and, thus, delete correctly the stuff words. If the stuff words are incorrectly deleted, bits will be inserted into or deleted from the customer's data stream. As a result, the received bits will be incorrectly formatted and incorrectly interpreted. When this

occurs, we say that character synchronization is lost. Character synchronization is important in data transmission because once it is lost, subsequent bits are erroneously interpreted even if transmitted correctly.

Two problems arise. First, if all input sequences are permitted, the data words must be encoded before stuffing occurs so that the stuff word can be distinct from the data words. Secondly, when transmission errors occur, it is possible for:

- (i) a stuff word to be transformed into a data word,
- (ii) a data word to be transformed into a stuff word, or
- (iii) a data word to be transformed into another data word.

In most cases, (i) and (ii) are more serious than (iii) because of the resulting loss of character synchronization. The prevention of type (iii) errors is not considered in this paper because error control, when required, is generally performed by the customer's terminal.

Previously, a format was proposed for the data words and the stuff word [1]. The format results in the stuff word being distance one^{*} from a data word. Therefore, a single transmission error can change a data word into the stuff word or vice versa with a corresponding loss of character synchronization.

* The distance, frequently called the Hamming distance, between two binary words X and Y is the number of positions in which X and Y differ.

We give an alternative encoding scheme that, for a given code efficiency, maximizes the minimum distance between the stuff word and any data word. It follows that the probability of character synchronization loss, because of transitions between data and stuff words due to transmission errors, is considerably reduced. Some implementations of moderate complexity are given to demonstrate that the increase in minimum distance can be achieved at a small cost in hardware.

Some other encoding schemes are presented because they are interesting variations on a theme. At the moment, no applications are known that fully use the properties of these codes. The codes are presented to indicate the wide range of capabilities that can be designed into the encoding scheme in hopes of stimulating new applications.

II. Preliminaries

A model of the processing before transmission is shown in Figure 1. The input binary data stream is segmented into k -bit data words denoted by A , where

$$A = (a_1, a_2, \dots, a_k).$$

We assume that all of the 2^k possible sequences for A are allowed. In order for the stuff word to be distinguishable from the data words, the alphabet is enlarged by adding redundancy to A . Thus, the encoder generates from A an n -tuple B where $n > k$ and

$$B = (b_1, b_2, \dots, b_n).$$

The sequence B is transmitted. The dimensions of such a code are denoted by (n, k) .

There are 2^n possible sequences for B of which 2^k are used to transmit data. Thus, there are $2^n - 2^k$ values of B that are not used for data but that can be used for other purposes including the stuff word. For $n > k$,

$$2^n - 2^k \geq 2^k. \quad (1)$$

The reader should appreciate the tremendous flexibility available in the design of the coding scheme because, by (1), never more than half of the possible B sequences are used as data words.

The processing after transmission is indicated in Figure 2. As shown later, it is possible to construct codes that have a minimum distance greater than one between the stuff word and any data word for practical values of n and k . For these codes, the first step at the destination is to delete the stuff word and all other received words "sufficiently close" to the stuff word. By "sufficiently close" we mean that the received word is more likely the result of a stuff word corrupted by errors than a data word corrupted by errors. Let B' denote the n -tuples that are not deleted where

$$B' = (b'_1, b'_2, \dots, b'_n).$$

The decoder operates on B' to form the k -bit word A' where

$$A' = (a'_1, a'_2, \dots, a'_k).$$

The system functions properly if

$$A' = A.$$

At this point, it is convenient to review the encoding and decoding schemes presented in [1]. An $(8,7)$ code is used, i.e., $n = 8$, $k = 7$. The encoder generates B from A by the relation

$$b_8 = 1,$$

$$b_i = a_i \quad \text{for} \quad 1 \leq i \leq 7.$$

Thus,

$$B = (a_1, a_2, \dots, a_7, 1).$$

The stuff word is (10000000). At the receiver, only the word (10000000) is deleted as the stuff word; all other received words are decoded as data words. Thus, the stuff word is interpreted as a data word if one or more transmission errors occur. Alternatively, because the data word (10000001) is distance one from the stuff word, a single error can convert this data word into the stuff word. Any other data word requires at least two errors for conversion into the stuff word. It follows that a single transmission error can cause a loss of character synchronization.

III. Basic Concepts

We construct a code that has one stuff word and 2^k data words such that the minimum distance between the stuff word and any data word is maximized. The problem is to divide the 2^n n-tuples into three sets; namely,

- (i) the stuff word denoted by S,
- (ii) the set D consisting of the 2^k data words, and
- (iii) the set U consisting of the $2^n - 2^k - 1$ unused words,

such that, for given values of n and k, the minimum distance between S and any element of D is maximized.

Choose one of the n-tuples to be the stuff word S. For given n, k and S, let d_m denote the maximum possible minimum distance between S and any element of D. We determine d_m by observing that if all elements of D are to be at least distance d_m from S, then U must contain all n-tuples that are distance $d_m - 1$ or less from S. Thus, the set U contains

$$\begin{aligned}
 &\text{the } \binom{n}{1} \text{ n-tuples distance } 1 \text{ from S,} \\
 &\text{the } \binom{n}{2} \text{ n-tuples distance } 2 \text{ from S,} \\
 &\quad \vdots \\
 &\text{the } \binom{n}{d_m-1} \text{ n-tuples distance } d_m - 1 \text{ from S,}
 \end{aligned} \tag{2a}$$

and ω n-tuples distance greater than $d_m - 1$ from S such that

$$\sum_{\ell=1}^{d_m-1} \binom{n}{\ell} \leq 2^n - 2^k - 1 < \sum_{\ell=1}^{d_m} \binom{n}{\ell} \tag{2b}$$

and

$$\omega = 2^n - 2^k - 1 - \sum_{\ell=1}^{d_m-1} \binom{n}{\ell}.$$

The value of d_m is determined by (2b). The data words are the remaining 2^k words. The code is specified by (2) up to the choice of the ω unused words that are distance greater than $d_m - 1$ from S . While the choice of these ω words does not alter d_m , the probability that a data word is transposed into the stuff word is minimized if the ω words are all chosen to be distance d_m from S . In practice, the choice of the ω words does not change this probability substantially and it appears preferable to assign the ω words to simplify encoder-decoder design.

What effect does the choice of the stuff word S have upon the structure of the code? Let C be a code with the stuff word S and minimum distance d_m . Let K be an arbitrary n -tuple and construct a new code C' from C by adding* K to each n -tuple in C . The stuff word in C' is $K \oplus S$, the data words in C' are $K \oplus D$,† and the unused words in C' are $K \oplus U$. Suppose that D_1 is a representative element of D . The distance‡ between S and D_1 , $d(S, D_1)$, is

$$d(S, D_1) = w[S \oplus D_1] \geq d_m.$$

* The addition is component-by-component modulo two addition and is denoted by $\oplus$.

† The notation $K \oplus D$ implies that K is added to each element of the set D to form a new set $K \oplus D$.

‡ The distance between two binary words X and Y is denoted by $d(X, Y)$. The weight of X is the number of nonzero components in X and is denoted by $w[X]$. Thus, $d(X, Y)$ is numerically equal to $w[X \oplus Y]$.

In C' , D_1 becomes $K \oplus D_1$ and the distance between the stuff word $K \oplus S$ and $K \oplus D_1$ is

$$d(K \oplus S, K \oplus D_1) = w[(K \oplus S) \oplus (K \oplus D_1)] = w[S \oplus D_1] = d(S, D_1).$$

Therefore, changing a code by adding a constant n -tuple to all words simply rotates the code with no change in the distance properties. It follows that the minimum distance in C' between the stuff word and any data word is exactly d_m . The important point is that the choice of S does not influence d_m . Conversely, d_m is independent of S and is determined by n and k . This is illustrated in (2b).

Transmission of the all-zero word can be avoided by choosing S so that the all-zero word is one of the unused words. However, it may be convenient, particularly when detecting stuff words, for S to be the all-zero word because then distance from S is equivalent to weight and can be computed by counting ones. These two seemingly contradictory objectives can be simultaneously satisfied as follows. Design the encoder and stuff word using a code C in which S is the all-zero word. Let S' be one of the unused words in C . Add S' to each word immediately before transmission and add S' to each word immediately after transmission. The double addition of S' permits the suppression of the all-zero word for transmission but is transparent for the encoding, stuff word detection, and decoding operations.

Because the data words and the stuff word form a subset of the set of all possible n -tuples, we may, on the average, transmit an unequal number of zeros and ones. By varying $w[S]$ from 0 to n , the relative number of zeros and ones in the data words can be varied from mostly ones to mostly zeros.

IV. Capabilities

The first and most important measure of code capability is d_m . A plot of d_m versus n for $k = n-1$ and $k = n-2$ is given in Figure 3. Observe that it is possible to design an $(8,7)$ code in which $d_m = 4$. Similarly, we can build a $(16,15)$ code with $d_m = 8$. The $(15,14)$ code provides the same number of data positions as two words of the $(8,7)$ code, has $d_m = 8$, and compared with two words from the $(8,7)$ code leaves one position for other purposes, e.g., signaling. However, if the transmission system uses an 8-bit word, a code with $n \neq 8$ requires special precautions. For example, the decoder can become confused and incorrectly group the received 8-bit words for decoding, particularly if an 8-bit word is inserted or deleted at a digital switch.

Consider the $(8,7)$ code with $d_m = 4$. Let R_s and R_d denote the stuff rate and the rate of occurrence of the data words that are distance four from S , respectively. When the stuff rate is low ($R_s < R_d$), the decision rule at the receiver is biased in favor of the data words by deleting as stuff words all received words distance one from S and decoding as data words all remaining received words. Conversely, for high stuff rates ($R_s > R_d$), the decision rule is biased in favor of the stuff words by deleting

as stuff words all received words distance two or less from S and decoding as data words all remaining received words. When the rates are approximately equal ($R_s \approx R_d$), the two decision rules give comparable performance.

It is possible to modify the (8,7) code by relaxing the minimum distance requirement so that all data words are at least distance three from S . Received words distance one from S are deleted as stuff words; the remaining received words are decoded as data words. This balanced code gives roughly the same performance as either of the biased codes when $R_s \approx R_d$. In fact, as we shall see, the performance of the balanced code is sufficiently good for the intended application to Bell System PCM channels for a fairly wide range of rates that the small improvement achievable by the biased codes is not necessary. However, the reduction in minimum distance from four to three allows greater flexibility in encoder-decoder design, provides for less error multiplication (see Section V), and makes possible the design of some interesting codes with multiple stuff words (see Section VI).

We now compute the mean time between erroneous conversions of a stuff word into a data word (denoted by $\bar{T}_{d|s}$) and the mean time between erroneous conversions of a data word into a stuff word (denoted by $\bar{T}_{s|d}$). For the computations, we use the following assumptions that are taken from [1] and that are representative of the PCM carrier facilities used in the Bell System.

- (i) Transmission errors are independent of the transmitted bits, independent of each other, and occur with probability $p = 10^{-7}$.
- (ii) The transmission rate is 64×10^3 bits per second or, because $n = 8$, 8000 words per second.
- (iii) All 7-bit input words are equally likely.
- (iv) The stuff rate is R_s stuff words per second.*

The balanced (8,7) code is applicable except for extremely high or low stuff rates and is used here and in Section V for illustration. Let N_3 denote the number of data words distance three from S (the exact value of N_3 depends upon the encoder). Then

$$\bar{T}_{d|s} \approx \frac{1}{3600 R_s \left[\begin{matrix} 8 \\ 2 \end{matrix} \right] p^2} = \frac{9.93 \times 10^8}{R_s} \text{ hours}$$

and

$$\bar{T}_{s|d} \approx \frac{1}{3600 (8000 - R_s) \left[\frac{\begin{matrix} 3 \\ 2 \end{matrix} N_3}{2^7} p^2 \right]} = \frac{1.18 \times 10^{12}}{N_3 (8000 - R_s)} \text{ hours.}$$

* The value of R_s will vary depending upon the application. For fine adjustment of clock rates, R_s typically would be less than 20 words per second. If word stuffing is used for speed padding as well as adjusting clock rates (for example, to send 50 kilobit service over a 64-kilobit line), R_s could be 850 words per second or larger.

For comparison, the format in [1] provides

$$\bar{T}_{d|s} \approx \frac{3.46 \times 10^2}{R_s} \text{ hours}$$

and

$$\bar{T}_{s|d} \approx \frac{3.57 \times 10^5}{8000 - R_s} \text{ hours.}$$

The (8,7) code with a minimum distance of three increases the mean time to loss of character synchronization by a factor of $10^5 - 10^6$ over the format in [1]. Similarly, if the (16,15) code is used in a balanced mode, a minimum distance of seven is achieved; no set of three or fewer errors can cause a loss of character synchronization; and $\bar{T}_{d|s}$ and $\bar{T}_{s|d}$ are extremely large when $p = 10^{-7}$.

V. Encoding, Decoding and Stuff Word Detection

In this section, we give some possible implementations for the balanced (8,7) code. The implementations, which show the modest complexity of the encoder, decoder and stuff word detector, can be easily modified for either of the biased (8,7) codes or for other values of n and k provided $n = k + 1$ (we are primarily interested in the $n = k + 1$ situation). For $n > k + 1$, the encoder and decoder can in some cases be extended.

Case 1: We begin with the simplest case. Let the stuff word be

$$S = (00000000).$$

From (2), the unused words are the 8 weight one words, the 28 weight two words, and 91 other words of weight greater than two. The encoder operates as follows:

$$\begin{aligned} \text{If } w[A] \geq 3, \\ b_8 &= 0 \\ b_i &= a_i \quad \text{for } 1 \leq i \leq 7. \end{aligned} \tag{3a}$$

$$\begin{aligned} \text{If } w[A] \leq 2, \\ b_8 &= 1 \\ b_i &= 1 \oplus a_i \quad \text{for } 1 \leq i \leq 7. \end{aligned} \tag{3b}$$

Notice that (3) can be rewritten as (4).

$$\begin{aligned} \text{If } w[A] \geq 3, \quad b_8 &= 0. \\ \text{If } w[A] \leq 2, \quad b_8 &= 1. \\ \text{Then } b_i &= b_8 \oplus a_i \quad \text{for } 1 \leq i \leq 7. \end{aligned} \tag{4}$$

The operations in (4) are inherently straightforward and are systematic in that they are performed on every word and do not require memory for code-book encoding. Figure 4 shows the encoder in (4).

Decoding is also straightforward. The operation necessary for decoding is

$$a'_i = b'_8 \oplus b'_i \quad \text{for } 1 \leq i \leq 7. \tag{5}$$

Figure 5 shows the decoder in (5).

The encoding-decoding technique in (3), (4) and (5) results in some error multiplication. Suppose that position i , $1 \leq i \leq 7$, is in error and that position eight is correct. Then

$$b'_i = 1 \oplus b_i,$$

$$b'_8 = b_8$$

and, from (5) and (4),

$$a'_i = b_8 \oplus 1 \oplus b_i = 1 \oplus a_i.$$

The error in position i is delivered to the customer. However, if position eight is in error,

$$b'_8 = 1 \oplus b_8$$

and, from (5) and (4),

$$a'_i = 1 \oplus b_8 \oplus b_i = 1 \oplus a_i \quad \text{for } 1 \leq i \leq 7.$$

The point is that now all seven data positions are in error. A single transmission error in position eight causes seven errors for the customer. However, synchronization is maintained.

Recall that for the balanced code the stuff word detector deletes the stuff word and the eight words distance one from the stuff word. Thus, for the code at hand, to decide whether a received word should be deleted, we simply determine whether there are less than two ones in the word.

A slight variation of (4) is given in (6). Notice however, that the performance of the code is essentially unchanged.

$$\begin{aligned}
 &\text{If } w[A] \geq 2, & b_8 &= 1. \\
 &\text{If } w[A] \leq 1, & b_8 &= 0. \\
 &\text{Then } b_i = b_8 \oplus a_i \oplus 1 & \text{for } 1 \leq i \leq 7.
 \end{aligned} \tag{6}$$

Case 2: By a modification of the ideas in case 1, a transmission error in position eight results in four rather than seven errors. The stuff word and the unused words are the same as case 1. The encoder is given in (7).

$$\begin{aligned}
 &\text{If } w[A] \geq 3, & b_8 &= 0. \\
 &\text{If } w[A] \leq 2, & b_8 &= 1. \\
 &\text{Then } b_i = b_8 \oplus a_i & \text{for } 1 \leq i \leq 4, \\
 & & b_i = a_i & \text{for } 5 \leq i \leq 7.
 \end{aligned} \tag{7}$$

The decoder performs the operations

$$\begin{aligned}
 &a'_i = b'_8 \oplus b'_i & \text{for } 1 \leq i \leq 4, \\
 &a'_i = b'_i & \text{for } 5 \leq i \leq 7.
 \end{aligned} \tag{8}$$

The algorithms in (7) and (8) can be incorporated into the encoder and decoder for case 1 as shown in Figures 6 and 7, respectively. The stuff word detector is the same as in case 1.

As in case 1, errors in positions one through seven are conveyed to the customer. However, an error in position eight now results in four errors rather than seven.

As noted in Section III, it is possible to design a code for which the stuff word is the all-zero word and then, for transmission, suppress the all-zero word by adding an unused word to each word before transmission. By properly choosing the word that is added, it is possible to combine the encoding and addition operations in a simple manner. We give two examples both based on case 2.

Case 3: Let the stuff word be

$$S = (00000001).$$

↑
position 8

The unused words are the 8 words distance one from S, the 28 words distance two from S, and 91 other words distance greater than two from S. The encoder operates as follows:

$$\text{If } w[A] \geq 3, \quad b_8 = 1.$$

$$\text{If } w[A] \leq 2, \quad b_8 = 0.$$

$$\text{Then } b_1 = 1 \oplus b_8 \oplus a_1 \quad \text{for } 1 \leq i \leq 4,$$

$$b_i = a_i \quad \text{for } 5 \leq i \leq 7.$$

The decoder performs the operations

$$a'_1 = 1 \oplus b'_8 \oplus b'_1 \quad \text{for } 1 \leq i \leq 4,$$

$$a'_i = b'_i \quad \text{for } 5 \leq i \leq 7.$$

Case 4: The stuff word is

$$S = (\underbrace{1111}_{\text{positions 1,2,3 and 4}}0000).$$

The encoder is as follows:

If $w[A] \geq 3$, $b_8 = 0$.

If $w[A] \leq 2$, $b_8 = 1$.

Then $b_i = 1 \oplus b_8 \oplus a_i$ for $1 \leq i \leq 4$,

$b_i = a_i$ for $5 \leq i \leq 7$.

The decoder is identical to the decoder in case 3.

VI. Some Other Codes

Instead of maximizing the minimum distance between one stuff word and the data words, there are other goals that can be sought in code design. In this section, three codes are given that are, in a sense, generalizations of the code in (2) and that show the wide range of capabilities that can be achieved. The codes, for $n = 8$, $k = 7$, still have 2^7 data words and a set of unused words. However, instead of a single stuff word, there are now s special words that exhibit interesting distance properties. With the exception of some nebulous ideas about multiple stuff words or signaling where one special word is used as the stuff word and another special word is used to signal the end of a call, no applications are known that fully use the properties of the codes. The codes are presented in hopes of stimulating new applications and to show that a system designer can reasonably expect more than a single stuff word.

Let $n = 8$ and $k = 7$. The 2^8 possible 8-tuples are divided into three sets:

- i) the set S consisting of the s special words,
- ii) the set D consisting of 2^7 data words, and
- iii) the set U consisting of the 2^7 -s unused words.

The codes are specified by giving S and U. The set D consists of all 8-tuples not in S or U.

Code 1: The set S consists of the four words,

(11000000)
(00110000)
(00001100)
(00000011).

The set U consists of

- (i) the all-zero word,
- (ii) the 8 weight one words,
- (iii) the 24 weight two words not in S,
- (iv) the 24 weight three words not of the form

$(i_1, \bar{i}_1, i_2, \bar{i}_2, i_3, \bar{i}_3, 0, 0)$

$(i_1, \bar{i}_1, i_2, \bar{i}_2, 0, 0, i_3, \bar{i}_3)$

$(i_1, \bar{i}_1, 0, 0, i_2, \bar{i}_2, i_3, \bar{i}_3)$

$(0, 0, i_1, \bar{i}_1, i_2, \bar{i}_2, i_3, \bar{i}_3)$

where i_1, i_2 , and i_3 are arbitrary binary digits,*

- (v) the 54 weight four words not of the form

$(i_1, \bar{i}_1, i_2, \bar{i}_2, i_3, \bar{i}_3, i_4, \bar{i}_4)$

where i_1, i_2, i_3 , and i_4 are arbitrary binary digits,

and

- (vi) any other 13 words not in S.

* Let i be a binary digit. Then $\bar{i} = 1 \oplus i$.

Notice that each word in S is exactly distance four from every other word in S and that each word in S is at least distance three from any data word. Thus, there is protection against confusing the elements of S with the elements of D and protection against confusing the elements S with each other.

Code 2: The set S consists of the four words

(00000000)

(any weight 1 word)

(any weight 7 word)

(11111111)

The set U consists of

- (i) the 7 weight one words not in S,
- (ii) the 28 weight two words,
- (iii) the 21 weight three words that have a one in the same position as the weight one word in S,
- (iv) the 21 weight five words that have a zero in the same position as the weight seven word in S,
- (v) the 28 weight six words,
- (vi) the 7 weight seven words not in S, and
- (vii) any other 12 words not in S.

This code is interesting because the elements of S can be grouped into two pairs such that the distance between the pairs is large. All elements of S are at least distance three from any element of D. Of course, this code can be modified so that

the weight one and seven words in S are included in U. The resulting code provides extreme distance between the remaining two elements of S and good distance between the elements of S and the elements of D.

Code 3: The set S consists of the 12 words

(00000000)

(the 8 weight one words)

(11000000)

(10100000)

(01100000).

The set U consists of

- (i) the 25 weight two words not in S,
- (ii) the 56 weight three words,
- (iii) the 35 weight four words not of the form

$(1, 0, 0, i_1, i_2, i_3, i_4, i_5)$

$(0, 1, 0, i_1, i_2, i_3, i_4, i_5)$

$(0, 0, 1, i_1, i_2, i_3, i_4, i_5)$

where i_1, i_2, i_3, i_4 , and i_5 are arbitrary binary digits such that exactly three are one, or

$(0, 0, 0, i'_1, i'_2, i'_3, i'_4, i'_5)$

where i'_1, i'_2, i'_3, i'_4 , and i'_5 are arbitrary binary digits such that exactly four are one.

Notice the large number of words in S. Also, each word in S is at least distance three from any word in D although the elements in S are, in some cases, distance one apart.

VII. Conclusions

The maximum possible minimum distance between a single stuff word and any data word is determined and codes that achieve the maximum distance are specified. Certain measures of the capabilities of these codes are given. Some possible implementations are presented to demonstrate the modest complexity of the encoder, decoder, and stuff word detector.

Some other encoding schemes are given because they are interesting variations on a theme. Although no applications are known that fully use the properties of these codes, they are presented in hopes of stimulating new applications and to show that a system designer can reasonably expect more than a single stuff word without increasing the redundancy.

REFERENCE

1. Mattesich, R. R. and Richters, J. S., Bell Telephone Laboratories, Incorporated, unpublished work.

FIGURE 1 - Processing before transmission.

FIGURE 2 - Processing after transmission.

FIGURE 3 - Maximum minimum distance d_m for various n .

FIGURE 4 - Encoder for Case 1.

FIGURE 5 - Decoder for Case 1

FIGURE 6 - Encoder for Case 2

FIGURE 7 - Decoder for Case 2

FIGURE 8 - Encoder for Case 3

FIGURE 9 - Decoder for Case 3