NEW CLASSES OF SYNCHRONOUS CODES

by DENNIS JOHN CLAGUE, B.Sc. (Hons.), Adelaide.

Ph. D. THESIS

MATHEMATICS DEPARTMENT

JULY, 1966

I give consent to this copy of my thesis, when deposited in the University Library, being available for loan and photocopying.

Date 28/6/67 Signed

TABLE OF CONTENTS

PRECIS

This thesis deals with classes of fixed length binary codes which can be used for information transmission in a two-state channel. These codes are called synchronous codes.

The synchronous codes constructed here are in general more efficient than codes known to be in use at the moment[*], for all possible code lengths. In fact, there is some similarity between the new codes and the less-restricted comma-free codes of Golomb, Gordon and Welch[**].

Gilbert uses the technique of fixing certain positions in every code word to achieve synchronisation; in this thesis an entirely new technique of achieving synchronisation is developed, which allows the codes which use it to have high efficiency; the technique consists of constraining pairs of positions instead of fixing positions. The pure technique, which provides highly efficient codes for small code lengths, is developed first, and is used to construct synchronous codes analogous to certain well-known linear and cyclic codes (e.g. the Fire codes); then a hybrid technique is developed which gives high efficiencies for all other code lengths.

[*] E.N. Gilbert, 'Synchronisation of Binary Messages', I.R.E. Transactions on Information Theory, September, 1963.

[**] 'Comma-free Codes', Canadian Journal of Maths, 1958.

Precis (cont.).

Synchronous codes of even length are also developed, and these are shown to be only slightly less efficient than corresponding odd length codes.

Next, Gilbert's technique is generalised, and a class of codes is developed which is as efficient as any class of codes which uses a fixed position technique can be.

Finally, it is proved that a synchronous code cannot be as efficient as a comma-free code, except for very small code lengths. In fact, an upper bound on the efficiency of synchronous codes is developed which is considerably lower than that on comma-free codes, and codes are constructed which reach this upper bound for all code lengths up to 93, and which approach the bound for code lengths greater than this.

To the best of my knowledge and belief, none of the material in this thesis has either been presented for the award of a degree or diploma in any University or written or published by another person, except where reference is made in the text.

D.J. Clague.

ACKNOWLEDGEMENTS

I would like to express my thanks to Mr. I. Hinckfuss, late of the Weapons Research Establishment, Salisbury, South Australia, for introducing me to the topic; to Mr. G. Rose, of the Engineering Department, University of Adelaide, for help concerning the feasibility of implementation of the techniques described herein; and especially to Dr. J.H. Michael of the Mathematics Department, University of Adelaide, for the many helpful suggestions he has made concerning the presentation of this material.

Also, I thank my wife Aldith, both for typing this thesis and for the moral support she has given me during its compilation.

INTRODUCTION

In these days of rocketry, space exploration and earth satellites, there is an increasing demand for reliable, efficient methods of transmitting information over long distances. One of the more reliable media for transmitting information under these circumstances is the two-state channel, where the two states are an 'on' state and an 'off' state, i.e. a signal of a certain power or the absence of any signal whatsoever. Use of such a system eliminates errors which can occur in systems where the power of the signal is important.

These two states can be represented by the binary digits 0 and 1.

In a three-state channel, we can send binary data and reserve the third state for commas (spaces) between words. But use of such a system can lead to difficulties when the receiver has to distinguish between the three states, especially when the message is distorted due to external factors, e.g. atmospheric static or tape 'noise'. Therefore, there is a need of codes which are suitable for transmission in a two-state channel. But we must use both of the states to transmit information; and we have no straightforward method of indicating the point at which one information batch, or code word, ends and the next code word starts. And unless we can be sure that we can find the

start of a code word, either at the beginning of the message or after a loss of synchronisation between the receiver and the transmitter due to external interference, the code is useless.

There are two ways of doing this. The first is that employed by the variable length encodings. With such codes as the Huffman code /6/ or the alphabetic code /2/, the code will synchronise itself after only a finite amount of information has been lost; the codes are constructed so that this will occur.

These variable length encodings are excellent for the transmission of alphabetic data where the letters have a variable probability of occurrence.

Numeric data requires a different approach. We assume every digit has the same probability of occurrence; and it is desirable that every code word has the same length. This is especially true if we wish to transmit information in parallel channels.

For this reason, there is a need for codes in which every code word has the same length and which can be used in a two-state channel. But once such a code gets out of synchronization, it cannot resynchronise itself, as the variable length codes do. So we must empower the receiver with some method of resynchronising itself. The only way this can be achieved is via the code format.

For example, in a class of codes discovered by Gilbert /3/, a certain sequence of binary digits occurs at the beginning of each

code word, and at no other point in any code word. This provides a sure method of finding the starting point of a word. But when the input message is synchronised, these digits are wasted, since no information can be transmitted in them.

This thesis introduces codes which can be used in the same situations as Gilbert's codes; from among the various classes of codes we shall show that, for a given code length, we can always find at least one code which will transmit more information per word than Gilbert's code of the same length. To achieve this, we have to complicate our synchronising procedure a little; but we show that this disadvantage is not serious, for the first code word received correctly after a loss of synchronisation will still be recognised and decoded.

It can be appreciated that there must be a certain minimum number of positions which must be restricted in order that the receiver can resynchronise itself. We will find the value of this minimum and show that some of the codes we will construct are therefore as efficient as codes of this type can be.

1. GENERAL

1.1. BASIC DEFINITIONS

1.1.1. A Word

A word is a finite sequence of elements of a set F.

1.1.1.1. The number of terms in the sequence is the length of the word, and the ith term will be called the ith digit of the word. A word of length n can thus be regarded as an n-tuple of elements of F.

1.1.1.2. We shall only consider the case where F consists of the binary digits 0 and 1. Thus a word of length n will be a sequence of n terms, each term being either a zero or a unit

e.g. 0 1 0 1 is a word of length 4. We shall represent this as

0101

1.1.2. A Code

A code is a non-empty set of words.

1.1.2.1. In this thesis we will only consider codes in which all the words have the same length. This common length will be called the length of the code.

1.1.3. The (Hamming) Weight of a Code Word

The weight of a word in a code of the above type is defined to be the number of units in it.

The example given in section 1.1.1.2. is of weight 2.

1.1.4. The set of all words of a given length

For each positive integer n, let $W^{(n)}$ denote the code consisting of all words of length n.

A code of length n is thus a non-empty subset of $W^{(n)}$.

1.1.4.1. The number of words in $W^{(n)}$ with weight w is $\binom{n}{w}$, since there are w units to be placed in n positions.

1.1.4.2. In this thesis, we shall continue to use n to represent the length of the code under consideration.

1.1.5. A Composite Word

If $a = a_1 a_2 \ldots a_m$, and $b = b_1 b_2 \ldots b_n$, are two words of length m and n respectively, then we will denote by ab the word

$$a_1 a_2 \ldots a_m b_1 b_2 \ldots b_n$$

1.1.6. A Comma-free Code

A code C of length n is said to be comma-free if, given any two code words

$$a = a_1 a_2 \ldots a_n \quad \text{and} \quad b = b_1 b_2 \ldots b_n \text{, then}$$

no word of the form

$$a_{r+1} \ldots a_n b_1 \ldots b_r, \quad 0 < r < n, \quad \text{is in C.}$$

1.1.6.1. We shall define $a_{r+1} \ldots a_n b_1 \ldots b_r$, $0 < r < n$, to be a false word with respect to the comma-free code C. Thus a code C is comma-free if none of its false words is identical with a code word.

1.1.6.2. As an example of a comma-free code C, consider the code of length 4 which has as code words a and b, where

a = 0001 and b = 0011

If we consider the composite words given below, we see that they yield the corresponding false words.

aa : 0010, 0100, 1000

ab : 0010, 0100, 1001

ba : 0110, 1100, 1000

bb : 0110, 1100, 1001

Thus we can see that since C is comma-free 0010, 0100, 1000, 1001, 0110 and 1100 can not be code words.

Suppose c = 0111 is also a code word. This implies that the words given below are false words;

ac : 0010, 0101, 1011

bc : 0110, 1101, 1011

ca : 1110, 1100, 1000

cb : 1110, 1100, 1001

cc : 1110, 1101, 1011

Thus the code C with code words a, b and c, has as false words

0010, 0100, 1000

0101, 0110, 1001

1100, 1011, 1101

1110

The only members of $W^{(4)}$ represented in neither C nor the set of false words are

d = 0000 , e = 1010 and f = 1111.

Now d could not possibly be a member of a comma-free code of length 4, since dd generates false words identical with d; the same applies to f, and ee also generates a false word identical with e. We can see that none of a, b and c is in the set of false words. So the code consisting of a, b and c is comma-free.

We have thus constructed a comma-free code of length 4 to which no more words can be added. We have not shown that there is no better choice of a, b and c which would allow another member g to be added to the code, in the same way as we added c to the original choice of a and b.

1.1.7. A Synchronous Code

A comma-free code S of length n is said to be synchronous if some of the positions in the code are sufficiently arbitrary that binary information may be transmitted in those positions.

To this end, we postulate that, in a synchronous code, positions (or sets of two or more positions) contain digits which are totally independent of the digits in any other positions (if we disregard any artificial dependence which may be introduced by parity checking of the digits). Each position in a synchronous code S is either a fixed position (i.e. position which contains the same digit for every code word), or completely arbitrary,

e.g. The word $a_1a_2 \cdots a_{n-1}0$

is in S if and only if

$a_1a_2 \cdots a_{n-1}1$ is in S,

or a member of a set of positions which is arbitrary:

e.g. The word $a_1a_2 \cdots a_{n-2}01$

is in S if and only if

$a_1a_2 \cdots a_{n-2}10$

is in S.

As an example of a code which combines these attributes, consider the one consisting of the six words

01000
01001
01100
01101
01110
01111

This code is comma-free, since any false words beginning with 01 must begin in position 4 of a code word, and the third and fourth positions of the false word would then be 0 and 1 respectively; no code word has 0 and 1 in these positions.

The first two positions are fixed; they are 0 and 1 respectively for every code word.

The last position is completely arbitrary; it may contain 0 or 1 irrespective of the digits in the third and fourth positions.

The third and fourth positions form a semi-arbitrary set. They may contain 00, 10 or 11, no matter what the last position contains. But they may not contain 01.

Note that neither the third nor fourth position is completely arbitrary.

e.g. 01111 and 01000 are in the code

01011 and 01010 are not.

1.1.7.1. A comma-free code can only be synchronous if some of the positions in every code word can be filled arbitrarily, without considering the characters in other positions of the code word. If there is some dependence among the characters in a code word, apart from that introduced intentionally in the form of parity checks, an artificial correspondence must be established between coded messages and meanings.

If there are many code words, need of a 'dictionary' of this type would defeat the whole purpose of synchronous codes, which is to enable binary information to be transmitted economically in a two-state channel, without any third state to denote the break between words; and although consideration of efficiency will take precedence over considerations of hardware, we cannot neglect the latter. So we will only consider cases where the extra property possessed by synchronous codes supplies the method of implementing the information transmission.

1.1.8. Efficiency of a Code

We shall define the efficiency of a code of length n with N code words to be $(\log_2 N)/n$.

1.2. THE AIM OF THIS THESIS

1.2.1. The Frame of Reference

As stated above, we shall consider only binary codes in which each code word has the same length. In addition, we shall impose the restriction that synchronous codes must be able to carry information; this can only be achieved if some of the positions in every code word are independent of other positions in the code word as in the above example, and can be filled with either 0 or 1 as the information input dictates, regardless of the characters in other positions in the code word.

1.2.2. What we Shall Attempt

In section 2, we shall find an upper bound on the number of code words possible in a synchronous code of length n, and shall describe briefly a class of comma-free codes which reach this upper bound for at least some values of n; this class of codes, which we shall call the Golomb codes, are a subset of those discovered by S.W. Golomb and others /4/. Unfortunately, these codes do not obey the additional restriction imposed on synchronous codes.

We shall attempt to show that this upper bound is unrealistic for synchronous codes; we shall find another bound on codes constructed by methods of the types used here, and give evidence that it is an upper bound on all synchronous codes.

1.2.3. What we shall Achieve

We shall construct classes of synchronous codes which lie within our frame of reference and whose efficiency approaches our upper

bound mentioned above; we shall also construct synchronous codes analogous to well-known linear and cyclic codes.

We shall give the formats for classes of synchronous codes which encompass many code lengths, including even code lengths. We shall compare the efficiencies of codes from the different classes which have the same code length. And, finally, we shall adduce considerable evidence that codes within our frame of reference have an upper limit on their efficiencies of $1 - Kn^{-\frac{1}{2}}$, where K is a constant, instead of the value $1 - \log_2 n$ attained by the Golomb comma-free codes.

In section 3, we shall develop a new idea on synchronous codes; the new codes use pairs of digits, partially restricted to achieve synchronisation. We will compare these new codes, the M codes, with the classical class of codes discovered by Gilbert /3/ and demonstrate by this comparison that the new codes are highly efficient for small, odd, lengths.

We also have to show that the codes can be implemented without too much additional hardware. This is the purpose of section 4, where we establish a transformation from binary data to the form required for transmission by the M codes, i.e. into the pairs of digits the new codes require for synchronisation.

But the M codes are not particularly efficient for large values of n. To cover this case, a class of codes, the K codes, also of odd length have been constructed; these codes are really a cross between the M codes and Gilbert's codes.

Like the M-codes, the K codes are of odd length, and so, although we can now construct a highly efficient synchronous code of any odd length, there has so far been no attention given to the case of even length synchronous codes. In section 6, the even-length counterpart of the K codes, the E codes, are constructed. The even length case is shown to be more complex than the odd length case, and the modifications which have to be made to the format result in loss of efficiency. However, the fact that such codes can be constructed, with efficiencies approaching our upper bound, shows that there is not too much difference in the efficiencies attainable with even and odd length synchronous codes, at least in the binary case.

In Section 7, a class of codes, the F codes, are constructed; these codes are very similar to Gilbert's codes, but more efficient. They are not quite as efficient as the K codes, but their greater simplicity of operation may make them preferable in practical applications.

The last section, section 8, deals with a class of codes for which only simple examples can be constructed; in fact, the main purpose of this section is to demonstrate the probability that synchronous codes can never be as efficient as the corresponding Golomb's codes; the upper bound it imposes is considerably lower than that derived in section 2. And, in conclusion, evidence is provided that even this latest upper bound may not be the least upper bound for all values of n.

2. CYCLES

2.1. Introduction

The concept of a cycle was used by Golomb et al in their discovery of the upper bound. We will expand their idea in order to develop a closer relationship between the length of a code and the weight of the words in it. This is important in synchronous codes, since the greater the 'spread' of code word weights, the greater the average number of positions in which code words differ, and the easier it is to correct errors. This becomes apparent if we consider the two two-word synchronous codes S_1 and S_2

S_1 : 01000 and 01111 are code words

S_2 : 01100 and 01101 are code words

Any single error in S_1 is immediately correctable; with S_2, a single error in position 5 is not correctable.

2.1.1. Throughout section 2, n will be a fixed positive integer and $W^{(n)}$ will be abbreviated to W.

2.2. The Operator B

We define B on W as follows:

For each word $a = a_1a_2 \ldots a_n$ of W, define

$$B(a) = a_2a_3 \ldots a_na_1$$

Then $$B^2(a) = a_3a_4 \ldots a_na_1a_2$$

and $$B^n(a) = a$$

The operator B corresponds to multiplication by the polynomial x in the representation of the code word as the coefficients of a polynomial modulo x^n-1 over GF(2).

The "order" of a word a is defined to be the smallest positive integer r such that $B^r(a) = a$. Since $B^n(a) = a$, it is clear that the order of a divides n.

2.3. Definition

If a is a word with order r, then the set

$$a, B(a), B^2(a), \ldots, B^{r-1}(a)$$

of r distinct words is called a cycle of order r. Clearly, any member of a cycle generates the cycle. Also, all the words in a cycle have the same weight. We will refer to this as the "weight" of the cycle. The following properties of cycles are easily verified.

2.4. Properties of Cycles

2.4.1. The order of every cycle divides n.

2.4.2. Different cycles are disjoint and W is the union of mutually disjoint cycles.

2.4.3. If W_w denotes the subset of W consisting of all words of weight w, then W_w is the union of all cycles of weight w.

2.4.4. The only cycles of order 1 consist of

(a) the word 00 ... 0; or (b) the word 11 ... 1.

2.4.5. If the word length n is a prime p, then all the non-trivial cycles are of order p.

This follows from 2.4.1.

2.4.5.1. For the sake of neatness in some of the expressions in later sections, we shall designate combinatorial (n, w), i.e. n!/(w! (n-w)!), as Comb(n,w).

2.4.6. If n is a prime and $1 < w < n$, then W_w is the union of Comb(n,w)/n cycles.

This is a consequence of 2.4.5.

2.4.7. If $a = a_1 \ldots a_n$ is a word of order $r < n$, then a can be written in the form

$$a = a_1 \ldots a_r a_1 \ldots a_r \ldots a_1 \ldots a_r$$

Then, since

$$w = \sum_{i=1}^{n} a_i, = \frac{n}{r} \times \sum_{i=1}^{r} a_r,$$ it follows that n/r divides w.

i.e. If there is a cycle of order r and weight w then n/r divides w.

2.4.8. If S is a synchronous code, two words from the same cycle cannot be in S, because if a is in S, then the words which aa causes to be omitted are all the other members of the cycle containing a.

2.4.9. For the same reason, no word a of order $r < n$, can be in a synchronous code, since one of the words eliminated by aa would be a itself.

2.5. Fermat's Theorem

By 2.4.4. and 2.4.5., we can see that, if n is a prime, we have $2^n - 2$ words divided into disjoint cycles each containing n words. Therefore, n divides $2^n - 2$, n prime.

This is a special case of Fermat's theorem.

We have now established the method by which all the members of W can be grouped into cycles, and the fact that all the words of a certain weight can be collected as a set of cycles of that weight. We shall now investigate more closely the relationship between the cycles and the values of w for a fixed value of n.

The next theorem establishes the number of cycles of a given weight w if n and w have no common divisor greater than 1. This case is straightforward, and provides for the majority of cases. However, the case where a common divisor exists is much more interesting; it will be considered in sections 2.7 et seq.

2.6. Theorem

If $(n,w) = 1$, then the number of cycles of weight w is $\mathrm{Comb}(n,w)/n = \mathrm{Comb}(n-1, w-1)/w$.

Proof:

Consider an arbitrary cycle with weight w and order r. Since r divides n, we can write $n = rk$, $(k > 1)$ and since $n/r = k$ divides w, k must be 1. Thus, every cycle of weight w has order n. Since the total number of words with weight w is Comb(n,w) and these words are divided into mutually disjoint cycles, each containing n words, it follows that the number of cycles is

$$\text{Comb}(n,w)/n$$

This is easily shown to be equal to

$$\text{Comb}(n-1,\ w-1)/w$$

This takes care of the case when $(n,w) = 1$. The next few sections will develop the case when $(n,w) > 1$.

We will first develop the basic argument by considering the case where $n = p^a$, p a prime; later this will be generalised to encompass more general values of n.

2.7. Lemma

Let $n = p^a$ (where p is prime and $a > 1$) and let $1 \leqslant r \leqslant a$.

(i) If p^{a-r+1} divides w, then there are

$$\text{Comb}(p^r,\ w/p^{a-r}) - \text{Comb}(p^{r-1},\ w/p^{a-r+1})$$

words of weight w that belong to cycles of order p^r.

(ii) If p^{a-r} divides w, but p^{a-r+1} does not, then there are

$$\text{Comb}(p^r,\ w/p^{a-r})$$

words of weight w belonging to cycles of order p^r.

Proof:

A word b of length $n = p^a$ belongs to a cycle with order less than or equal to p^r if and only if b has order p^s, where $s \leqslant r$ i.e. if and only if the order of b divides p^r, hence if and only if b consists of a word c repeated p^{a-r} times. If b has weight w, then c has weight w/p^{a-r} and length p^r. Thus the number of words of length n and weight w which belong to cycles of order less than or equal to p^r is

$$\text{Comb}(p^r, w/p^{a-r})$$

(i) If p^{a-r+1} divides w, the number of words belonging to cycles of order exactly p^r is

$$\text{Comb}(p^r, w/p^{a-r}) - \text{Comb}(p^{r-1}, w/p^{a-r+1})$$

(ii) If p^{a-r} divides w, but p^{a-r+1} does not, there are no cycles with order less than p^r, hence the number of words belonging to cycles of order p^r is

$$\text{Comb}(p^r, w/p^{a-r})$$

2.8. Theorem

Suppose $n = p^a$ and $w = c.p^k$ (where p is a prime, $a \geqslant 1$, $k \geqslant 0$, $c > 0$, $w < n$, and p does not divide c).

Then the total number of cycles of weight w is

$$\text{Comb}(p^a, c.p^k)/p^a + (p-1) \sum_{r=1}^{k} (\text{Comb}(p^{a-r}, c.p^{k-r})/p^{a-r+1}).$$

Proof:

(i) When $w = n$ or $w = 0$, the theorem is trivial.

(ii) When $k = 0$, the theorem is the same as theorem 2.5.

(iii) Suppose $0 < w < n$ and $k \geqslant 1$. It follows from lemma 2.7 that for each r such that $a-k+1 \leqslant r \leqslant a$, there are

$$(\mathrm{Comb}(p^r, c.p^{k+r-a}) - \mathrm{Comb}(p^{r-1}, c.p^{k+r-a-1}))/p^r$$

cycles of order p^r and weight w, and when $r = a-k$ there are

$$\mathrm{Comb}(p^r, c.p^{k+r-a})/p^r = \mathrm{Comb}(p^r, c)/p^r$$

cycles of order p^r and weight w.

Since the only cycles that can occur are of order p^r and since n/p^r divides w, then p^{a-r} must divide w; hence $a-r \leqslant k$, so that $a-k \leqslant r \leqslant a$.

Thus the total number of cycles

$$= \mathrm{Comb}(p^{a-k}, c)/p^{a-k}$$

$$+ \sum_{r=a-k+1}^{a} (\mathrm{Comb}(p^r, c.p^{k+r-a}) - \mathrm{Comb}(p^{r-1}, c.p^{k+r-a-1}))/p^r$$

$$= \mathrm{Comb}(p^a, c.p^k)/p^a + (p-1) \sum_{r=1}^{k} (\mathrm{Comb}(p^{a-r}, c.p^{k-r})/p^{a-r+1})$$

$$= n^{-1} q \sum_{r=0}^{k} p^{r-1} \mathrm{Comb}(p^{a-r}, c.p^{k-r}),$$

where $q = p$, $r = 0$,

$q = p - 1$ otherwise.

It is unfortunate that the combinatorial symbol is not a multiplicative function; if it were, we would be able to obtain a much neater expression for this result. As it is, we are denied this satisfaction.

In the case where $n = p_1^{a1}\, p_2^{a2}$, the total number of cycles of weight w where $(n\ ,\ w) = p_1^{k1}\, p_2^{k2}$ and $w = c.p_1^{k1}\, p_2^{k2}$, is

$$n^{-1} q_1 q_2 \sum_{r=0}^{k_1} \sum_{s=0}^{k_2} p_1^{r-1}\, p_2^{s-1}\, \mathrm{Comb}(p_1^{a1-r}\, p_2^{a2-s},\ c.p_1^{k1-r}\, p_2^{k2-s})),$$

where $q_1 = p_1$, $r = 0$

$q_2 = p_2$, $s = 0$

$q_i = p_i - 1$ otherwise.

The pattern now becomes apparent, but again simplification is impossible. And so we will not proceed further with this line of enquiry.

In the next few sections, we shall investigate the cycles of order n, which give much neater results.

In section 2.13, we will derive an alternative formula which is applicable in more general cases than those just given. It is interesting that the two formulae are identical, since they are so different in form.

The next theorem derives a neat result for the number of cycles of order n and weight w; it will be used in section 2.13 to derive the alternative formula mentioned above.

2.9 **Theorem**

Suppose $(n , w) = T$.

Then the total number of cycles of length n, weight w and order n is

$$n^{-1} \sum u(d)\ \mathrm{Comb}(n/d,\ w/d)$$

where the summation is taken over the distinct divisors d of T, $1 \leqslant d \leqslant T$, 1 being regarded as a divisor, and where u is the Moebius function; i.e.

$u(d) = 1$ if $d = 1$

$= (-1)^s$ if $d = q_1 q_2 \dots q_s$ and $q_1, \dots, q_s$ are distinct primes,

$= 0$ if $d = q_1 q_2 \dots q_s$ and two of the primes are the same.

Proof:

Let $n = kT$. A positive integer r divides n and is such that n/r divides w if and only if $r = kt$, where t divides T. For each divisor t of T, let A_t denote the set of all words of length n, weight w and order kt.

Put $B_t = \bigcup_{d/t} A_d$

A word b belongs to B_t if and only if it consists of a word a of length kt repeated $n/(kt) = T/t$ times. The weight of a is wt/T, hence the number of words in B_t is $\mathrm{Comb}(kt,\ wt/T)$.

For each divisor t of T, let X_t, Y_t denote the characteristic functions of the set A_t, B_t. Then

$$X_T(x) = \sum_{d/T} u(d) Y_{T/d}(x), \qquad (1)$$

because (i) if x is in A_T, then $Y_{T/d}(x) = 1$ when $d = 1$ and $Y_{T/d}(x) = 0$ when $d > 1$.

(ii) if x is not in A_T, but x is in B_T, then x belongs to a cycle of order kt', where t' divides T and $t' < T$, hence

$$\sum_{d/T} u(d) T_{T/d}(x) = \sum_{d/(T/t')} u(d) = 0$$

Now the number of words in A_T

$$= \sum_{X} X_T(x)$$

and by (1)

$$= \sum_{d/T} u(d) \sum_{x} Y_{T/d}(x)$$

$$= \sum_{d/T} u(d) \; \mathrm{Comb}(n/d, w/d)$$

2.10 Corollary

The total number of cycles of order n among the words of $W^{(n)}$ is

$$n^{-1} \sum_{w=1}^{n-1} \sum_{d/(n,w)} u(d) \; \mathrm{Comb}(n/d, w/d)$$

The next theorem, however, gives a much neater formula for this evaluation.

This next theorem gives the number of cycles of order n in the form used by Golomb et al. It specifies the upper bound on the number of words in a synchronous code because

(i) No word can be in a synchronous code if it is in a cycle of order less than n, (by section 2.4.9) and

(ii) not more than one word from any cycle can be in a synchronous code (by section 2.4.8).

However, it can be readily appreciated that it may be impossible to choose one word from every cycle and obtain a synchronous code, although comma-free codes, which are less restricted, are known which reach this bound. In section 8, we will give evidence that this bound is not attainable by any but the simplest synchronous codes.

2.11 Theorem

The number of cycles of order n among the words of $W^{(n)}$ is

$$n^{-1} \sum_{d/n} u(d)\ 2^{n/d}$$

Proof:

Consider all the words of length n divided into their respective cycles. Each cycle is either of order n or of order $d < n$, where d divides n.

If we define f(d) as the number of words of length n in cycles of order d, it is obvious that

$$2^n = \sum_{d/n} f(d)$$

since each word has a certain order, hence belongs to a cycle of that order, and so it is counted once and only once in the summation.

By the well-known Moebius inversion formula,

$$f(n) = \sum_{d/n} u(d)\ 2^{n/d}$$

But each cycle of order n contains n words.

Therefore, the number of cycles = $n^{-1}\ f(n)$.

2.12 Corollary

It follows from 2.10 and 2.11 that the following is true:

$$\sum_{w=1}^{n-1} \sum_{d/(n\ ,\ w)} u(d)\ \mathrm{Comb}(n/d,\ w/d) = \sum_{d/n} u(d)\ 2^{n/d}$$

Note: when $w = n$, $\sum_{d/n} u(d) = 0$

We will now derive a formula which can be used instead of that given in 2.8, and which can also be used in more general cases. The fact that its form remains the same for all values of n invites favourable comment when it is observed that the formulae given in section 2.8 become complex when two or more distinct primes divide n.

2.13 The total number of cycles of weight w in $W^{(n)}$.

We know from theorem 2.9 that the number of cycles of order n and weight w is

$$n^{-1}\sum_{d|T} u(d)\ \mathrm{Comb}(n/d,\ w/d),\quad \text{where } T = (n,w).$$

Now the other words of weight w are members of cycles of order n/e, $e>1$ where e divides (n, w).

Among the words in short cycles in W, we will find all the words in $W^{(n/e)}$, repeated e times.

The number of short cycles generated by words of order n/e is therefore

$$en^{-1}\sum u(d)\ \mathrm{Comb}(n/(de),\ w/(de)),$$

where the summation is now over the divisors of $(n/e, w/e)$.

Thus the total number of cycles with weight w in a code of length n is

$$n^{-1}\sum_{e/(n\ ,\ w)} e \sum_{d/((n\ ,\ w)/e)} u(d)\ \mathrm{Comb}(n/(de),\ w/(de))$$

We can rearrange this expression to

$$n^{-1}\ (n\ ,\ w)\sum_{e/(n\ ,\ w)} e^{-1}\sum_{d/e} u(e/d)\ \mathrm{Comb}(nd/(n\ ,\ w),\ wd/(n\ ,\ w)),$$

by putting $e' = (n\ ,\ w)/e$ and $d' = e'/d$. In this form the role of the greatest common divisor is easily observable.

2.14 The Total Number of Cycles in $W^{(n)}$

Section 2.11 has shown us that the total number of cycles of order n is

$$n^{-1} \sum_{d/n} u(d)\, 2^{n/d}$$

Applying a similar argument to that used in 2.13, we can show the total number of cycles of all orders to be

$$\sum_{d/n} d^{-1} \sum_{e/d} u(e)\, 2^{d/e}$$

2.15 The Golomb Codes

Golomb et al descovered a new class of comma-free codes. They built up their proof that the codes were comma-free by demonstrating the following properties of words of length n, where n is odd.

2.15.1 Every sequence of digits of length n, can be represented by a sequence of n signs, each sign being + or - ; the ith character in the sign sequence is a plus if the ith digit is less than the (i+1)th digit (mod n), but a minus otherwise, $i < n$; the last sign is a result of comparing the last and first digits.

If we apply this transformation to every one of the 2^n words in $W^{(n)}$ we will get duplication of sign sequences.
e.g. 011 and 010 both give the sign sequence + - - .

Thus, this representation presents a method of grouping words other than by cycles, and the sets of words so obtained have varying weights.

If every one of the words which have a certain sign sequence representation is of order n, then every word in the set can be a member of a comma-free code; and, since we are not interested in words of smaller order, we discard these and consider only the words of order n with the given representation.

Thus application of this idea reduces the complexity of the operation of choosing one word from each cycle of order n, since, once we have chosen one word, we can choose words, from other cycles, with the same sign sequence representation without destroying comma-freedom.

But the main purpose of applying this transformation is to enable us to find the word we want in a cycle. We want from each cycle of order n a word whose representation begins with an odd number of pluses and ends with an even number of minuses, or vice versa.

In the binary case, the second alternative does not apply, since there can be no sequence of two or more plus signs in the representation.

Here we shall prove the following lemma using this hypothesis, although it has been proved in the more general case.

2.15.2 Lemma

In every cycle of odd length $\geqslant 3$ there is at least one word whose representation begins with a plus sign and ends with an even number of minus signs.

Proof:

The sign representations of the words in a cycle themselves form a cycle.

Since no two plus signs are consecutive, we know that there is a sequence of one or more minus signs between each pair of plus signs. If we can show that one of these sequences contains an even number of minus signs, we can choose the word whose representation begins at the plus sign following this sequence.

(i) If there are an odd number of plus signs, there are an even number of minus signs, since n is odd, and these are divided into an odd number of sequences by the plus signs.

By parity at least one of the sequences must contain an even number of minus signs.

(ii) If there are an even number of plus signs, there are an odd number of minus signs divided into an even number of sequences.

Again, by parity, there must be at least one even sequence.

This completes the proof.

This does not imply that there will be only one even sequence; if this is so, there is no trouble. We shall resolve later the case where there are two or more even sequences.

2.15.3 Notes on Golomb's Work

Golomb et al have omitted the last sign of the sequence; thus their sequences are of length n-1, and begin with an odd number of pluses and end with an odd number of minuses.

They have shown that the choice of one word with this property from each cycle of order n will give a comma-free code, provided $n \leqslant 15$ and n is odd. Jiggs /7/ established that the result is also true for n = 17. However, Selfridge, in the latter paper, casts doubt upon their conjecture that comma-free codes constructed in this manner attain the upper bound for all odd n.

When n = 9, they have suggested a lexicographic ordering for choosing a word, from a cycle, with representation

\+ - - + - - + - -

(Note that we do not choose a word from every cycle with this representation; some words with this representation are of order 3).

They have shown that this method of choice leads to a comma-free code.

However, they have not specified a method of choice in the case where n is prime. The following method seems to give correct results.

2.16 Choice between two sequences

If there is only one even string of minuses in a sequence, we have no option. However, if there are two or more even strings, we

have our choice of two or more representative sequences; we choose the one with its longest string most nearly positioned at the centre of the sequence. If two sequences are equal in this respect, we take into account the sequences on either side; in cases of mirror images we choose the one with the longer strings towards the start of the sequence.

This may or may not work in all cases. It is possible that Golomb et al chose their representative sequences by some other method; it is also possible that theirs is a better method. However, our method does work in some cases, as shown by the following examples.

2.17 Example

We shall demonstrate this technique in the case where n = 11; this is the smallest prime value for which a choice has to be made.

We shall construct the code and prove that it is comma-free. The complexity of the operation, which is compounded many times over for larger n, illustrates the reason that no codes have been constructed for $n > 17$.

We shall represent the sign sequences by the length of the minus sequences, e.g. 2, 2, 4 represents the sequence

\+ - - + - - + - - - -

The sequences we must consider are

10	1,1,6	1,1,1,4
1,8	1,3,4	1,1,3,2
3,6	2,2,4 XA	1,2,2,2 B
5,4	3,1,4	1,3,1,2
7,2	1,5,2	2,1,2,2 XB
	2,4,2 A	2,2,1,2 XB
	3,3,2	3,1,1,2
	4,2,2 XA	1,1,1,1,2
	5,1,2	

The sequences marked with A in the second column are cyclic permutations of each other; these are the ones with more than one even-numbered string of minus signs. So we can choose our representative, and, in accordance with our policy outlined above, we choose 2,4,2 and omit the other two, which we have marked with crosses.

Similarly, the sequences marked with B in the third column also represent the same cycle, and we choose 1,2,2,2 as representative.

If the code represented by these sign sequences is not comma-free, it can only be because one sign sequence in the set is identical with the overlap of two other sign sequences. (This is analogous to the argument given in section 1.1.7).

Of course, if an overlap begins at the start of a sequence, the pattern of the overlap is identical with a code pattern; this does not affect the property of comma-freedom, since in this case any word with this pattern is a code word.

(i) Suppose a false sequence begins with the plus sign and last minus string of a sequence.

The number of minuses in this string is either 8, 6, 4 or 2.

Such a false sequence could only be identical with 2,4,2.

But then, for the false sequence to be identical with a sequence in the set, 4,2, the only possibility, would have to begin a sequence in the set; this is not so, and so this case is eliminated.

(ii) Suppose the false sequence begins at the second last plus sign of a sequence. Its first two minus strings are

1,6 3,4 4,2 1,4 5,2 1,2 or 3,2

Of these, only 1,2 begins a sequence in the set; the sequence is 1,2,2,2.

But the false sequence cannot be 1,2,2,2 because 2,2 does not begin any sequence in the set.

(iii) Suppose the false sequence begins at the third last plus sign of a sequence. Its last three minus strings are

1,1,4 1,3,2 3,1,2 2,2,2 or 1,1,2

None of these begin a sequence in the set.

(iv) The only other possibility is that 1,1,1,2 begins a sequence in the set. This is not true.

This completes the proof.

2.18 Example

We shall now give a similar construction and proof for the case n = 13.

The rate at which the number of possibilities to be considered increases demonstrates the impractibility of manual proofs for n very much larger.

The sequences we must consider are those which have the minus string configurations.

12	1,1,8	1,1,1,6	1,1,1,1,4
1,10	3,1,6	1,1,3,4	1,1,1,3,2
3,8	2,2,6 XA	1,2,2,4 XC	1,1,2,2,2 G
5,6	1,3,6	1,3,1,4	1,1,3,1,2
7,4	5,1,4	2,1,2,4 XD	1,2,1,2,2 XH
9,2	4,2,4 XB	2,2,1,4 XE	1,2,2,1,2 H
	3,3,4	3,1,1,4	1,3,1,1,2
	2,4,4 B	1,1,5,2	2,1,1,2,2 XG
	1,5,4	1,2,4,2 D	2,1,2,1,2 XH
	7,1,2	1,3,3,2	2,2,1,1,2 XG
	6,2,2 XA	1,4,2,2 E	3,1,1,1,2
	5,3,2	1,5,1,2	1,1,1,1,1,2
	4,4,2 XB	2,1,4,2 XE	
	3,5,2	2,2,3,2 XF	
	2,6,2 A	2,3,2,2, F	
	1,7,2	2,4,1,2 C	
		3,1,3,2	
		3,2,2,2 XF	
		3,3,1,2	
		4,1,2,2 XC	
		4,2,1,2 XD	
		5,1,1,2	

We now have to choose representatives from sequences represented more than once.

Again we mark with letters the members of the string set which are cyclic permutations of each other. There was a need of only two such letters in the previous example; here we need ten. Such is the rate at which the complexity increases.

In our choice F, we have the mirror image case with 2,3,2,2 and 2,2,3,2; we choose the former with good reason, as we will show later.

Suppose the false sign sequence begins with the last plus sign of a sequence in the set. Its first minus string has length 8,6,4 or 2. The only possibilities for false sequences are

2,4,4 2,6,2 2,3,2,2 and 2,4,1,2. One of these could only be idential with a sequence of the set if one of

4,4 6,2 3,2,2 or 4,1,2 began a sequence in the set.

This is not the case.

Suppose the false sequence begins with the second last plus sign. It must begin with one of the following configurations:-

1,8 1,6 3,6 1,4 3,4 4,4 5,4

1,2 3,2 5,2 6,2 7,2 4,2 2,2

The only possibilities for false sequences are:-

1,4,2,2

1,2,4,2

1,2,2,1,2

But no sequence in the set begins with either 2,2 4,2 or 2,1,2 .

Suppose the false sequence begins with the third last plus sign of a sequence of the set.

The possible configurations are

1,1,6 1,3,4 3,1,4 1,1,4 1,5,2 2,4,2 3,3,2 4,2,2

5,1,2 3,2,2 4,1,2 1,3,2 3,1,2 1,1,2 2,2,2 or 2,1,2

A possible false sequence identical with sequences in the set is

1,1,2,2,2

However, no sequence in the set begins with 2,2.

If we had chosen 2,2,3,2 instead of 2,3,2,2, the proof would break down at this point; in more complicated cases, the choice is not so simple.

Suppose the false sequence begins with the fourth last plus sign in a sequence in the set.

The possible configurations are

1,1,1,4 1,1,3,2 1,2,2,2 1,3,1,2 2,2,1,2 3,1,1,2 or 1,1,1,2.

However, none of these start a sequence in the set.

1,1,1,1,2, the sequence beginning at the fifth last place of the last sequence in the set, does not begin any sequence in the set.

Thus the sets of words which have the representations given above form a comma-free code; and one word from each cycle of order n has one of the representations given. Thus the number of words in the comma-free code attains the upper bound.

2.19 Notes

We have shown that it is possible to construct comma-free codes of lengths 11 and 13 which have the maximum possible number of words. But mainly we have shown the difficulty attendant upon a proof that the code is comma-free lies in the extremely high number of cases which have to be considered; a comparison of the two examples shows clearly how the number of cases is increasing, and it becomes evident that further attempts at proof in this direction would achieve little, especially in view of Selfridge's result /7/.

In fact, Selfridge's result is important to our contention that the number of cycles of order n is an unrealistic upper bound on the number of words in a synchronous code, since the restrictions imposed by Selfridge are much less strict than those we have imposed on synchronous codes.

So for a while we shall concern ourselves with the construction of synchronous codes which may have a definite use, and not concern ourselves too much with the relationship of the efficiency attained by them to this now probably unattainable upper bound, although we shall of course consider the comparative efficiencies of the different types of codes constructed. After a while, a pattern will emerge, and it will become apparent that synchronous codes constructed by the types of methods we shall use here seem to have an upper bound on their efficiencies which is much less than that given above.

3. M CODES AND GILBERT'S CODES

3.1. General

In this section we shall show that the cycles of W can be divided into two disjoint subsets; for any odd value of n greater than 1, a synchronous code, the M code of that length, can be assembled from one word in each cycle in one of the subsets.

We need some standard with which to compare the efficiency of the M codes; for this we will use codes of the class discovered by Gilbert. This class of codes is in use at many installations at the moment (e.g. Weapons Research Establishment, Salisbury, South Australia).

We shall see from this comparison that the M code represents a definite advance for small values of n; however, for large values of n, Gilbert's code is far the more efficient. Thus the M codes do not present a solution to the problem of devising a highly efficient synchronous code of large length; this problem will be tackled in section 5. In section 4, subcodes of M codes will be shown to be analogous to well-known types of linear and cyclic codes. This versatility gives us another interest in the M codes.

First we will define a device which is used by many synchronous codes, including all those described in this thesis; the device, called the herald, provides a simple method of finding a possible first position in a word among the sequence of digits

arriving at the receiver. We shall then describe Gilbert's codes, whose property of synchronisation is in large part due to the herald; and while describing these codes we shall highlight the arguments which lead us to expect that, for any given length, a more efficient synchronous code than Gilbert's code of that length can be constructed.

3.2 The Herald of a Code

When a sequence of digits is arriving at our receiver, and we have to distinguish the start of a word, it helps us if we do not have to consider (n-1) wrong choices for the first position before finding the digit which actually starts a code word. We could, of course, find the correct first position by taking each digit in turn and comparing the sequence of n digits beginning at that digit with all the possible code words until an identity is obtained; this is the method we might have to use with a comma-free code. But the procedure is too laborious for the uses we have in mind, and the size of the 'dictionaries' we would have to employ is prohibitive for all but the smallest n.

The alternative is to standardise the first digit in each code word in some way; for example, if we stipulate that the first digit in every code word is a zero, we rule out all units as possible starting positions. This would reduce our search time by half.

If we want to reduce our search time even further, we could, for example, specify that the second position of every code word is also a zero. By this and further stipulations, we can reduce our search to a reasonable magnitude.

This 'fixing' of the leading positions in every code word is thus a useful device. We call the string of positions so fixed the herald of the code.

3.3 Methods of Achieving Synchronisation

We have shown that a herald is a useful device in a synchronous code. But use of a herald in itself does not guarantee that a code is synchronous, except in a few exceptional cases when the herald is more than half as long as the code word itself.

For aherald shorter than this, it is possible that the herald sequence could be duplicated in the remainder of the code word, if the right digits are chosen in the correct order, and provided nothing is done to prevent this occurrence. If it is possible that the string of n digits begun by this false 'herald' and overlapping into the next word could be identical with any code word, again assuming the right (or wrong, depending on the point of view) choice of digits, then the code would not be synchronous.

3.3.1. False Words Identical with Code Words

When we are considering codes with heralds, the only positions where false words identical with code words may possibly

begin are those where a false 'herald' can begin.

If it is possible for a false 'herald' to occur in a code word, there must be some restriction imposed on the digits occupying certain positions in the code words; these restrictions must prevent any false word from being identical with a code word.

3.3.2 The Methods

We have just described the two methods of restricting the code format which, with the use of a herald, will ensure that the code is synchronous. We must place restrictions other than the herald on the format of a synchronous code; this much has been shown. We can either

(i) ensure that no false words are identical with code words; or

(ii) ensure that no false 'herald' can occur; in this case there are no false words identical with code words, and so the condition of (i) is automatically satisfied.

It is obvious that the second method is far stronger than the first method; this means that it is stronger than necessary. Thus any code which uses this second method of achieving synchronisation (by preventing false heralds) should not be as efficient as the most efficient code which uses the other method.

Gilbert's codes are constructed so that no false 'heralds' may occur. This makes for extremely easy synchronisation; the receiver just seeks the first herald sequence, which it knows to be

the start of a code word, and begins decoding at that point. But if we can afford a more sophisticated receiver, and we can guarantee that, after a loss of synchronisation, the first correctly received word is decoded, there is no reason why we should not get greater efficiency by using a different code.

By a more sophisticated receiver, we mean one that can examine a complete string of n digits, instead of just a herald sequence, to determine whether the string represents a code word or a false word. But we do not want to get back to the 'dictionary' system. So we have to devise simple tests on certain positions of the digit string which give different results for code words and false words. This, combined with a 'herald' test, enables us to pick up the first correctly received word after a loss of synchronisation.

To make these tests work, we have to restrict certain positions in every code word. This can be done in two ways.

Firstly, we can stipulate that a certain position be a zero (or a unit) in every code word. Positions so fixed are normally amid positions left completely arbitrary to carry information.

It can be appreciated that, if we fix enough positions, the code so obtained will be synchronous. But while the digit string is arriving without abnormalities, and the receiver is putting out words one after the other, such fixed positions represent a complete loss

of time which, if not used to transmit the fixed digits, could be used to transmit information. But the fixed positions must be included in case synchronisation is lost; and it is obvious that there is a minimum number of positions which must be fixed if the code is to be a synchronous code, i.e. if the false words are to give different results to the tests to the code words.

This minimum covers methods which employ tests on single positions; we look at the ith position and say "This should be a unit, if the digit string represents a code word - is it?" The 'dictionary' test, on the other hand, employs a test on all n positions; and, if it could be implemented, it would give much higher efficiency.

Is there, then, a method which has the best features of both these systems - a method which gives higher efficiencies than the single position test method, but which can be used without referring to a 'dictionary'? There is; it consists of testing pairs of digits together.

We shall show in section 3.15 that this method does give high efficiencies, although final development of the method for maximum efficiency for all n will be left until section 5; and we shall also show that the tests are easy to apply.

Another point comes to mind; if testing pairs of digits increases efficiency, will not testing triplets or quadruplets yield still more efficient codes. Strangely, this is not the case; for

some reason, the two-digit testing systems gives up optimum results, and we shall prove this in section 8.

We shall now set up Gilbert's codes, and prove that they are synchronous; use of the stronger method of achieving synchronisation, as stated at the beginning of this section, makes this proof particularly straightforward.

3.4 Gilbert's Codes

We shall demonstrate the construction for a given value of n.

Choose b as the least integer such that

$$b^2 \geqslant n-1$$

We now stipulate that every code word has zeroes in positions 1,2 ... , b, and units in positions b+1, 2b+1, ... , (k-1)b+1 where $(k-1)b < n-1 \leqslant kb$, and a unit in the last position. All the other positions of the code word are arbitrary, and are used for the transmission of information. Thus, if we have n=15, we choose b=4, and we encode the information string ABCDEFG as

0000lABC1DEF1G1

Even if b-1 information symbols, such as ABC, are all zeroes, we cannot obtain b zeroes together except in the herald of a code word. Therefore there are no false 'heralds', and no false words; and so the code is synchronous.

3.4.1 Attainment of Synchronisation

After a loss of synchronisation, the receiver picks the first string of b zeroes as the start of a code word. This is a very easy system to implement.

3.4.2 The Choice of b

We shall show in this section that our choice of b gives the greatest efficiency with one of Gilbert's codes.

We have fixed b positions as zeroes and k positions as units. Therefore, by our choice of b, we have fixed

(i) 2b positions if $b(b-1) < n-1 \leqslant b^2$

(ii) 2b-1 positions if $(b-1)^2 < n-1 \leqslant b(b-1)$.

Consider another code of length n with c leading zeroes, and m units positioned in the same way as before. We know $(m-1)c < n-1 \leqslant mc$.

This code fixes c+m places. Suppose this code is more efficient than our original code; i.e. it must fix at least one less place.

Suppose $b(b-1) < n-1 \leqslant b^2$

then $mc > b(b-1)$ (1)

For greater efficiency, $c+m \leqslant 2b-1$ (2)

Squaring (2) and subtracting 4 x (1), we get

$$(c-m)^2 < 1$$

Therefore, $c = m$

By (2),

$$c = m \leqslant b-1$$

Therefore, $mc \leqslant (b-1)^2$, contradicting (1).

Suppose $(b-1)^2 < n-1 \leqslant b\ (b-1)$

then $mc > (b-1)^2$

For greater efficiency, $c+m \leqslant 2b-2$.

Repeating our argument above, we arrive at the same contradition.

Therefore there is no better choice for b than the one we have made.

3.4.3 The Efficiency of Gilbert's Codes

We showed in section 3.4.2 that for synchronisation we need to fix

(1) 2b positions if $b(b-1) < n-1 \leqslant b^2$ (i)

(2) 2b-1 positions if $(b-1)^2 < n-1 \leqslant b\ (b-1)$ (ii)

We can now calculate the efficiency of Gilbert's code of any length; however, to simplify comparisons which will be made in the following sections, we will consider the values of the efficiency at local maxima to be representative of the class as a whole.

There would appear to be dangers inherent in this policy; but we shall also show that the efficiency varies between sufficiently small limits to make this comparison meaningful.

As a general rule, the larger the value of n, the greater the efficiency of Gilbert's code of that length. However, if we plot efficiency against n on a graph, we find local maxima where $n = b^2+1$ and $n = b(b-1)$, where b is an integer. We can demonstrate this in the following way.

Suppose b is an integer and $n = b^2+1$. By (i) above, we have 2b fixed positions and $(b-1)^2$ arbitrary positions. If $n' = n-1$, we still have 2b fixed positions, but now only $(b-1)^2 - 1$ arbitrary position ; and if $n'' = n+1$, we need 2b+1 fixed positions, which leaves us with no more than $(b-1)^2$ arbitrary positions again.

In these last two cases, the ratio of fixed places to arbitrary positions is greater which means that the efficiency is less than in the case $n = b^2+1$. A similar argument applies when $n = b(b-1)$.

If $n = b^2+1$, the efficiency of Gilbert's code is

$$1-2n^{-\frac{1}{2}} + n^{-3/2} + O(n^{-5/2})$$

If $n = b(b-1)$, the efficiency of Gilbert's code is

$$1-2n^{-\frac{1}{2}} - \tfrac{1}{4}n^{-3/2} + O(n^{-5/2})$$

We will now give the local minima.

When $n = b^2+2$, the efficiency of Gilbert's code is

$$1-2n^{-\frac{1}{2}} -n^{-1} + n^{-3/2} + 0\,(n^{-5/2})$$

and when $n = b(b-1)+1$, the efficiency is

$$1-2n^{-\frac{1}{2}} - n^{-1} - \tfrac{1}{4}n^{-3/2} + 0\,(n^{-5/2}).$$

Now the local minima are n^{-1} less than the respective local maxima; this is reasonable, since in each case the fact that n is greater by 1 at the local minima than at the local maxima means only that we must have an extra fixed position to ensure that the code is synchronous. Fixing this extra position ensures this drop in efficiency of n^{-1}.

We now know that the efficiency of Gilbert's code is

$$1 - 2n^{-\frac{1}{2}} + 0\,(n^{-1}) \text{ for all } n.$$

It is this expression, with particular regard to the coefficient of $n^{-\frac{1}{2}}$, which will form the basis of our later comparisons.

3.5 The Box Principle

If we have a word $a = a_1a_2 \ldots a_n$ of length n, we can partition the word into pairs of digits

i.e. a_1a_2, a_3a_4, a_5a_6, ...

We shall call each of these pairs of elements a box.

If n is odd, we find that a_n is left over. In this case we have divided the word into $\frac{1}{2}(n-1)$ boxes and a_n.

These boxes are the devices with which we shall implement our tests on pairs of digits together, mentioned in section 3.3. We shall show, in our proof that an M code is synchronous, that this system of testing leads to an improvement in efficiency for small n; later we will consider the case for large n.

We need the following definition to simplify our proof.

3.5.1. A Doublet

A doublet is a pair of digits in a word; the first digit is a zero and the second a unit. However, we specify that, for an 01 pair to be a doublet, the zero must occupy the second position in a box when the word is partitioned in the manner given above.

It can be seen that every zero followed by a unit represents either an 01 box or a doublet in a word.

3.6. The M Codes

We shall construct the M code of length n, where n is odd.

We know that a word of length n, n odd, can be partitioned into $\frac{1}{2}(n-1)$ boxes, with a_n left over. We stipulate that the first of these boxes is 01, and that each of the other boxes is either 00, 10 or 11 arbitrarily; a_n can be either 0 or 1 arbitrarily.

The code consisting of all such words will be called the M code of length n.

We will define m to be the number of boxes which are not 01, in a code word.

Thus $m = \frac{1}{2}(n-3)$

We can represent a word in a M code as

$$01A_1A_2 \cdots A_m a_n$$

where (a) A_i = 00 or 10 or 11.

(b) a_n = 0 or 1.

Later, in section 4, when we discuss error correcting codes, we will use the digit in this last position as a parity check on digits in other positions. The error correcting codes discussed there will be proper subsets of the M code.

3.6.1. Example

Suppose we consider the M code of length 5.

The words are

01000	01001
01100	01101
01110	01111

We can show that the code is synchronous.

The only places where synchronisation could be misplaced in a correctly received message is that the 01 which ends either 01001 or 01101 could be mistaken for the beginning of a word. However,

the next two digits in the false word would be those beginning the next word in the message, and these are always 01; thus any false word would have 01 for its second box, and so it could not be identical with a code word.

This argument will be used extensively for the rest of this thesis. It states that if no false word is identical with a code word, then the code is synchronous.

Consider the result of dividing $W^{(5)}$ into cycles. We obtain 8 cycles in all.

1.	00000				
2.	00001	00010	00100	01000	10000
3.	00011	00110	01100	11000	10001
4.	00101	01010	10100	01001	10010
5.	00111	01110	11100	11001	10011
6.	01011	10110	01101	110100	10101
7.	01111	11110	11101	11011	10111
8.	11111				

We have underlined the words in the M code. It can be seen that we have chosen one word from every possible cycle; of course, we cannot choose words from cycles of order less than 5, in this case the trivial cycles.

If we recall the result of section 2.11, we see that the maximum number of cycles of order 5, and hence the maximum number of

words in a synchronous code of length 5, is

$$5^{-1} \sum_{d/n} u(d) 2^{n/d}$$

$$= 5^{-1} (2^5 - 2) = 6$$

Thus this synchronous code represents one of the few that attain the upper bound on comma-free codes. In fact, the M code for n=7 is the last M code which attains this bound; the reason for this is made clear in section 3.9.

The argument above shows that an M code can be synchronous. The next theorem proves that every M code is synchronous.

3.7 Theorem

An M code is synchronous.

Proof:

Consider two words, a and b, in M, the M code of length n.

Let z be a false word, consisting of n consecutive digits drawn from the sequence of $2n$ digits ab, but neither beginning at the first digit of a nor ending at the last digit of b. We have to show that z is not identical with any member of M.

Such a word z must begin at some position in a. Its first two digits must be 01, and since no box in a is 01 except the first, z must start at a doublet in a. This doublet must begin either in the second box of a or further to the right, since the second digit of the first box of a is a unit.

Therefore, z includes b_1 and b_2, the first two digits in b; and these are 0 and 1 respectively.

Since z begins at the second position in a box, and since n is odd, $b_1 b_2$ = 01 appears as a box in z.

Hence z has a 01 box other than the first.

Hence z is not in M.

Thus no false word can be identical with a code word.

As an example of this, suppose the doublet begins in position a_{n-3}.

In the false word z, the first box contains a_{n-3} and a_{n-2}, the second a_{n-1} and a_n, and the third b_1 and b_2.

ab: ... a_{n-4} a_{n-3} a_{n-2} a_{n-1} a_n 01 ...

z: 0 1 a_{n-1} a_n 01 ...

3.7.1. A possible adaption of the M code format

To ensure that a code like the M code is synchronous, we have to prevent 01 occurring in a box. We can do this by fixing the first position of a box as 1 and leaving the second position arbitrary, or by fixing the second position as a 0 and leaving the first position arbitrary.

However, this cuts the number of possibilities in a box from

3 to 2, and so reduces the efficiency of the code; for this reason it should not be used except where necessary. However, it is a useful adaption, as will be shown in section 4.

3.8 The Subsets of $W^{(n)}$, n odd

In this section, we shall divide the cycles of $W^{(n)}$ into two subsets - those which provide a member of an M code and those that do not. In this argument we will disregard the cycles of order 1, i.e. those generated by 000 ... 0 and 111 ... 1. For this reason, we shall define X as the subset of $W^{(n)}$ which contains all the words of $W^{(n)}$ which belong to the non-trivial cycles, i.e. all the words of $W^{(n)}$ except the two mentioned above.

We shall now define U to be the subset of X which consists of all words in cycles generated by members of the M code.

We now define V to be the subset of X which consists of all words in cycles generated by all the words b which have the following properties:

(1) b begins with 01, and has at least one other 01 box.

(2) There is a doublet in b somewhere between the first and the last 01 box of b.

This does not imply that every word of the form of b generates a different cycle; indeed, it will be shown that every word in V will be developed at least three times by different generators.

We shall show in the ensuing sections that every word in X is in either U or V, but that no word of X is in both.

First, we shall show that b is not the only word in its cycle which satisfies the restrictions laid upon it; in fact, there are at least two other words in the cycle containing b which obey the restrictions.

3.8.1 Theorem

If b is a word of length n, n odd, which begins with 01 and has at least one other 01 box, and a doublet somewhere between its first and last 01 boxes, then any word c, in the cycle generated by b, which begins with 01 also has at least one other 01 box, and a doublet between its first and last 01 boxes.

Proof:

By the conditions, b is of the form

$$01 \ldots 01 \ldots 01 \ldots b_n$$

box doublet last 01 box

If b is of this form, we shall say b has property BDB; c can start with the 01 represented in b by

(1) the last box

(2) a doublet before the last box

(3) a box before the last box and before a doublet

(4) a box before the last box and after all doublets

(5) a doublet after the last box.

It will clarify the proof if we construct the following table, which shows the relationship of a box or a doublet in b to the corresponding box or doublet in c, depending on whether the box or doublet in b occurs before or after the position in b corresponding to the first position in c.

(a) If c starts at a box in b:

	Before starting point		After starting point	
b:	Box	Doublet	Box	Doublet
c:	Doublet	Box	Box	Doublet

(b) If c starts at a doublet in b:

	Before starting point		After starting point	
b:	Box	Doublet	Box	Doublet
c:	Box	Doublet	Doublet	Box

e.g. reading from the first entry of the first table, if a box in b is previous to the starting point of c, it will be represented as a doublet in c.

Now we have this table, let us consider the possible starting points of c in order.

(1) The first box of b becomes a doublet in c.
The doublet before the last box of b becomes a box in c.

(2) The last box of b becomes a doublet in c.
The first box of b is still a box in c.

(3) The doublet of b after the starting point of c remains a doublet in c.
The last box of b is still a box in c.

(4) The doublet of b before the starting point of c becomes a box in c.

The first box of b becomes a doublet in c.

(5) The last box of b is still a box in c.

The doublet of b before the last box is still a doublet in c. If b generates a cycle of order n, then, since there are at least three zeroes followed by units in b, there are at least two other words, in the cycle generated by b , which also have property BDB.

This theorem has proved another important point. We can see that no member of the M code of length n can be in V, since every word in V beginning with 01 has at least one other 01 box. Since the cycles in W' are disjoint, U and V are disjoint subsets of X. But we still have to prove that their union is the whole of X.

Since no word from a cycle of order less than n can be in an M code, U does not contain any of the words in 'short' cycles. Thus, if U and V together contain every element of X, we have to show that there is at least one word in each 'short' cycle which possesses property BDB. The next theorem accomplishes this end.

3.9 Theorem

If c is a non-trivial word in $W^{(n)}$ of order less than n, then c is in V.

Proof:

We shall prove that c is in a cycle generated by b, where b has property BDB.

Suppose c is of order r, $1 < r < n$. We know by section 2.4.1 that $n = kr$, where k is an integer and $k > 1$.

Since n is odd, k and r are both odd.

Suppose b is a word in the same cycle as c, where b begins with 01. By section 2.4.7, b consists of a word of length r repeated k times.

i.e. $b = b_1 b_2 \ldots b_r b_1 b_2 \ldots b_r \ldots b_1 b_2 \ldots b_r$

By our choice of b,

$$b_1 b_2 \text{ is } 01$$

Therefore, $b = 01 \ldots b_r\ 01 \ldots b_r \ldots 01 \ldots b_r$

Now since r is odd, the second 01 pair given above is a doublet in b, and the last 01 pair given is a box in b.

So b has property BDB, and so b is a generator of a cycle in V; hence c is in V.

We have now shown that V contains some cycles of order n and all the non-trivial cycles of order less than n. We also know that U consists of cycles of order n. The next theorem will prove that U contains all the cycles of order n which are not in V.

Since all the words in V beginning with 01 have a doublet before their last 01 box, we can say that all the words beginning with 01 which are not in V do not have a doublet before their last 01 box.

We shall show that if a word c belongs to a cycle generated by a word d, which begins with 01 but has no doublet before its last 01 box, then c is in U; and since every word beginning with 01 must either have property EDB or be of the form of d, this means that U and V between them contain every word of X.

3.10 Theorem

If c is not in a cycle generated by a word b, where b has property EDB, then c is in U.

Proof:

Let d be a word beginning with 01 in the same cycle as c. d has no doublet before its last 01 box.

Let e be the word in the same cycle as c and d whose first two positions correspond to the last 01 box in d.

If d has only one 01 box, the first, then d is in the M code of length n; hence d generates a cycle in U; hence c is in U.

Now suppose d has more than one 01 box but does not have property EDB.

By our definitions of d and e, the only 01 pairs in d are represented by boxes up to the position corresponding to the first position of e, and doublets after this position.

But by the table given in 3.7.1, all these 01 pairs in d are represented by doublets in e.

Hence e has no 01 box other than its first, and so is in the M code of length n.

Hence c is in U.

We have thus shown that any word not in U is in a cycle generated by a word with property BDB. No word of odd length less than 9 can possess this property, since we need at least four boxes in a word in order to fit a doublet between the first and last boxes. This ties in with the fact that no M code of length greater than 7 is as efficient as Golomb's code of the same length, for we shall see that the larger the value of n, the greater proportion of words of $W^{(n)}$ belong to cycles generated by words with property BDB.

Since we have now shown that U and V together contain all the non-trivial words in $W^{(n)}$, the ratio between the number of words in each of them is important. We shall then exhibit an interesting fact. We shall find the ratio of words beginning with 01 in U to words beginning with 01 in V. One would expect that this ratio would be approximately the same as the overall ratio of the sizes of U and V. As a second possibility, the idea that the difference between these ratios would be in some way dependent on n would seem to be a reasonable one.

However, neither of these guesses is correct. We shall now calculate the ratios and show the relationship between them.

3.11 The Ratio of the Sizes of U and V

3.11.1 Lemma

The number of words in an M code is

$$2.3^n$$

Thus the number of words in U is

$$2n.3^n$$

Proof:

In a word b in the M code of length n, there are n boxes which can each be filled by any of the 3 pairs 00, 10 or 11, and b_n can be either 0 or 1. The other two positions, the first two, are a fixed 01 box.

Thus there are 2.3^n words in an M code.

Each one of these words generates a cycle of order n, i.e. each of these cycles contains n words.

Thus the number of words in U is $2n.3^n$.

We now know that the number of words in V is

$$2^n - 2 - 2n.3^n ,$$

and that the ratio of the size of U to the size of X is

$$2n.3^{\frac{1}{2}(n-3)} \; : \; 2^n - 2$$

We shall now find the ratio of the number of words beginning with 01 in V to the number of words beginning with 01 in X.

3.11.2 Lemma

The number of words in V which begin with 01 is

$$2^{n-2} - 8.(n+2)3^{n-2}$$

Proof:

A word in V which begins with 01 and has its last 01 box as its kth box, and which has its last doublet before the kth box beginning in the second position of its jth box ($j < k-1$), can be written in the form

$$01\ x_3x_4x_5x_6 \cdots x_{2j-1}\,01\ y_{2j+2}y_{2j+3} \cdots y_{2k-3}\ x_{2k-2}\ 01\ y_{2k+1} \cdots y_{n-1}\ x_n$$

where every x_i is either 0 or 1 arbitrarily and each pair

$$y_{2j+2}\ y_{2j+3},\ \cdots\ ,\ y_{2k-4}\ y_{2k-3},\ y_{2k+1}\ y_{2k+2},\ \cdots\ ,\ y_{n-2}\ y_{n-1}$$

is either 00, 10 or 11 arbitrarily.

Hence the number of words in V of this form is

$$2^{2j-3} \cdot 3^{k-2-j} \cdot 2.3^{n+1-k} \cdot 2$$

$$= 2^{2j-1} \cdot 3^{n-j-1}$$

Therefore the total number of words in V which begin with 01 is

$$\sum_{k=4}^{n+1} \sum_{j=2}^{k-2} 2^{2j-1} \cdot 3^{n-j-1}$$

$$= \sum_{k=4}^{n+1} 2^3.3^{n-2}\ ((4/3)^{k-3} - 1)$$

$$= 2^{2n+1} - 8.(n+2)3^{n-2}$$

Since there are altogether 2^{2m+1} words beginning with 01 in X, we now know that there are $8.(m+2).3^{m-2}$ words in U beginning with 01. The ratio of the number of words beginning with 01 in U to those beginning with 01 in X is

$$8.(m+2).3^{m-2} : 2^{2m+1}$$

If we compare this with the ratio found in 3.11.1, which is the same as

$$2(2m+3).3^{n} : 2^{2m+3} - 2,$$

we see that the ratios are not nearly equal. In fact, as n becomes large, this latter ratio tends asymptotically to 9/8 of the other. The reason for this is not apparent; it is certainly not the result we would expect.

We shall now examine the weight distribution of words in an M code and compare it with the weight distribution in $W^{(n)}$. We have already illustrated, in section 2.1., that the better the weight distribution of a code, the better the possibilities for error correction, for, although in many cases, the reduction in the number of parity digits necessary to correct every syndrome of weight e or less is not significant, the number of cases where an error of greater syndrome weight than e can be corrected is often greatly increased. Later we shall compare the weight distribution of an M code with that of Gilbert's code of the same length.

3.12 Theorem

If $0 < w \leqslant m+1$, then the number of words, of length $n = 2m+3$, with weight w, beginning with a 01 box and having no other 01 boxes is

$$\sum \text{Comb}(m,p).\text{Comb}(m+1-p,\ w-2p-1)$$

where the summation is taken over all integers p such that $0 \leqslant p \leqslant \frac{1}{2}(w-1)$.

NOTE: Since the number of words of weight w is the same as the number of words of weight n-w, by symmetry, the above formula can be used for words of weight greater than m+1.

Proof:

Each word which obeys the above condition has w-1 units and n-w-1 zeroes arranged among the last m boxes and the last digit.

Suppose the last m boxes of the word comprise p 11 boxes, q 10 boxes and r 00 boxes.

First, we shall find the number of words which obey the conditions of our theorem and which end with a zero.

In this case, $p + q + r = m$

$$2p + q = w-1$$

For any given p, q and r, the p 11 boxes can be distributed among the m boxes in Comb(m,p) ways.

For each of these distributions, the q 10 boxes can be arranged among the remaining (m-p) boxes in Comb(m-p, q) ways.

The remaining boxes can be filled in just one way with 00 boxes.

Hence the total number of words which begin with 01, have no other 01 box and which end with a zero is

$$\sum_{p,q} \text{Comb}(m,p)\ \text{Comb}(m-p,\ q) \qquad \text{A.}$$

where the summation is over all (p,q) such that $p \geqslant 0$, $q \geqslant 0$, $p + q \leqslant m$ and $2p+q = w-1$.

Similarly, the total number of words which begin with 01, have no other 01 box and which end with a unit is again

$$\sum_{p,q} \text{Comb}(m,p)\ \text{Comb}(m-p,q) \qquad \text{B.}$$

but here the summation is over all (p,q) such that $p \geqslant 0$, $q \geqslant 0$, $p + q \leqslant m$ and $2p+q = w-2$.

Now A. is equal to

$$\sum_{p=0}^{\left[\frac{1}{2}(w-1)\right]} \text{Comb}(m,p)\ \text{Comb}(m-p,\ w-2p-1) \qquad \text{C.}$$

Also B. is equal to

$$\sum_{p=0}^{\left[\frac{1}{2}(w-2)\right]} \text{Comb}(m,p)\ \text{Comb}(m-p,\ w-2p-2) \qquad \text{D.}$$

Hence the total number of words which begin with 01 and have no other 01 box is C + D.

C + D

$$= \text{Comb}(m, \tfrac{1}{2}(w-1))\ \text{Comb}(m-\tfrac{1}{2}(w-1), 0)$$

$$+ \sum_{p=0}^{\left[\frac{1}{2}(w-2)\right]} \text{Comb}(m,p)\ \text{Comb}(m-p, w-2p-1) + \text{Comb}(m,p)\ \text{Comb}(m-p, w-2p-2)$$

$$= \text{Comb}(m, \tfrac{1}{2}(w-1)) + \sum_{p=0}^{\left[\frac{1}{2}(w-2)\right]} \text{Comb}(m,p)\ \text{Comb}(m-p+1, w-2p-1) \qquad E.$$

by the well-known property of combinatorials.

If $p = \frac{1}{2}(w-1)$, the summand is

$$\text{Comb}(m, \tfrac{1}{2}(w-1)).\text{Comb}(m-\tfrac{1}{2}(w-1)+1, 0)$$

$$= \text{Comb}(m, \tfrac{1}{2}(w-1))$$

Therefore, $E. = \sum_{p=0}^{\left[\frac{1}{2}(w-1)\right]} (\text{Comb}(m,p)\ \text{Comb}(m-p+1, w-2p-1))$

We shall now apply this theorem to determine particular values of w for n=17 and n=51. The values are plotted on graphs, which also have Gilbert codes plotted for comparisons.

3.13 Notes

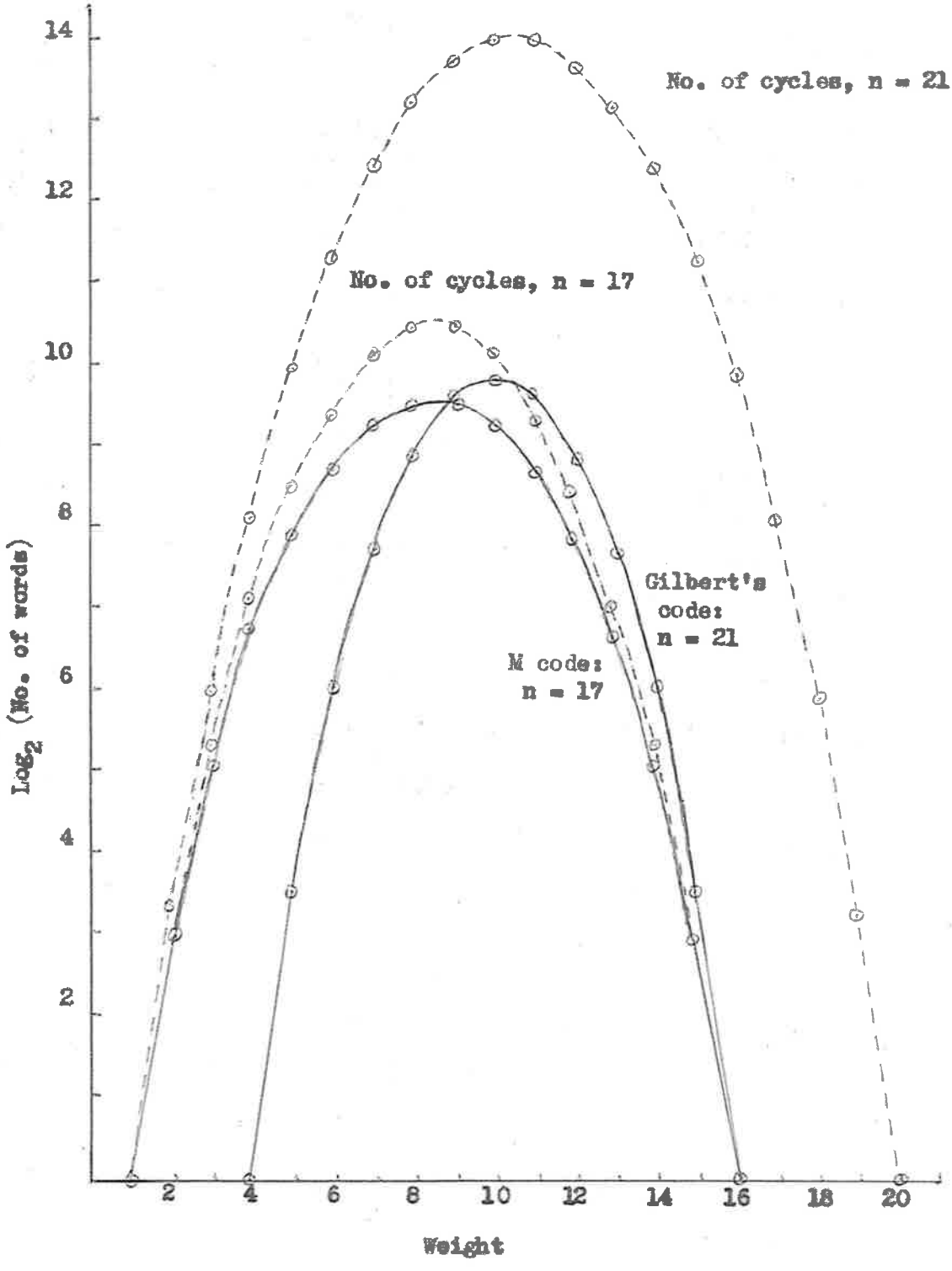
For a comparison of the M code and the fixed place code of the same length (n=51), and a comparison of the M code (n=17) and a fixed place code (n=21) with approximately the same number of words, see the following graphs.

(i) In graph 1 we see a comparison of the M code of length 17, which has 4374 words and the fixed place code of length 21 which has 4096 possible words (redundancy is not considered here), together with the graphs of the number of cycles for each weight. The log scale used does not allow the difference to be immediately apparent, but it can still be seen that the M code has many words of both small and large weight, while the weights of words in the fixed place code fall between much stricter limits (this can also be seen in graph 2). Thus the average distance between words in the M code is greater, although the minimum distance may not be (see later), and this means that a greater percentage of errors of syndrome weight greater than the minimum are correctable with the M code than with the corresponding fixed place code.

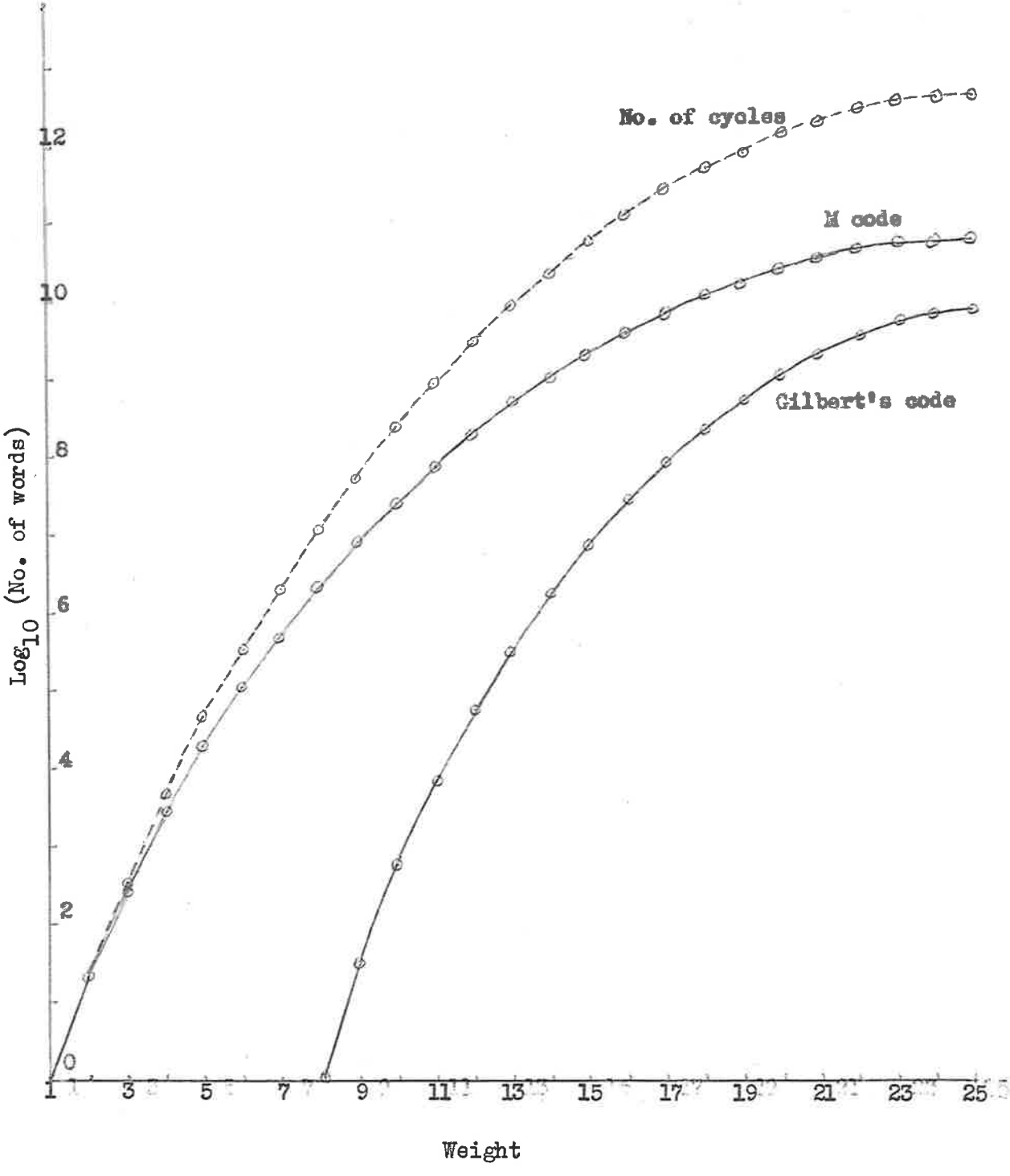
(ii) Graph 2 compares the M code and the fixed place code for $n = 51$ and it can be seen immediately that the M code has more words of every weight. As will be shown in the next lemma, the M code is theoretically preferable for $n \leqslant 79$. However, as shown in section 4, we encounter a difficulty in representing parity checks. An improvement is given in section 5 where the synchronous codes have arbitrary positions, and the parity checks are used as in a normal linear code.

M code, n = 17: 4,374 words

Gilbert's code, n = 21: 4,096 words



M code and Gilbert's code: n = 51



3.14 Lemma

The M code is more efficient than the fixed place code of length n with no redundancy for $n \leqslant 79$.

Proof:

The number of words in the M code is

$$2.3^{\frac{1}{2}(n-3)}$$

The number of words possible in the fixed place code of length n, making no allowance for redundancy, is 2^F, where F is the number of arbitrary positions, and is approximately $n-2n^{\frac{1}{2}}$.

The M code is the more efficient if

$$F \text{ is less than } 1+\tfrac{1}{2}(n-3)\log_2 3$$

This equation solves to $n \leqslant 79$.

It is interesting to note that, if we use the approximation $\log_2 3 = \frac{3}{2}$, which we will see gives accurate values under the encoding methods in the next section, the inequality solves to $n \leqslant 53$.

3.14.1 The Efficiency of an M code

We know that an M code has $2.3^{\frac{1}{2}(n-3)}$ words. Its efficiency is therefore

$$(\tfrac{1}{2}(n-3)\log_2 3+1)/n$$

or about

$$0.8 - 1.4n^{-1}$$

approximating $\log_2 3$ by 1.6.

It can be seen from this that for large values of n, Gilbert's code will be more efficient, since M codes have a limiting efficiency of 0.8; the limiting efficiency on Gilbert's codes is 1. If we use the approximation for $\log_2 3$ of 1.5, which will become appropriate in section 4, the efficiency of the M code is

$$0.75 - 1.25n^{-1}$$

3.15 Average Distance

We know that the average distance between words in a code is not as important as the minimum distance, as far as error correcting potentialities are concerned. However, a greater average distance between words means that the number of words for which an erroneous word can be mistaken is probably less, for given syndrome weight. Thus, greater average distance between words is another advantage of the M code over the fixed place code of length n.

3.15.1 Theorem

For $n \leqslant 79$, the M code has greater average distance between words than the fixed place code of length n.

Proof:

Consider the array consisting of all the words in an M code listed one under another in a column. We have n-3 columns of our word column containing members of either 11, 10 or 00 boxes, and

one column containing 1's or 0's equally.

There are N words; $N = 2.3^{\frac{1}{2}(n-3)}$

We have $\frac{1}{2}(n-3)$ columns containing $\frac{2N}{3}$ units, $\frac{N}{3}$ zeroes.

We have $\frac{1}{2}(n-3)$ columns containing $\frac{N}{3}$ units, $\frac{2N}{3}$ zeroes.

We have 1 column containing $\frac{N}{2}$ units, $\frac{N}{2}$ zeroes.

Summing differences, we get

$$A = (2N/3)(N/3)(n-3) + (N/2)(N/2)$$

differences as the sum over $\frac{1}{2}N(N-1)$ pairs of words

Average distance between two words $= 2A/(N(N-1))$

$$= 2N^2(8n-15)/(36N^2(1-1/N))$$

$$\simeq 4n/9 - 1$$

Now, with the fixed place code with no redundancy, we have F columns containing 1's or 0's equally.

The number of words is $N' = 2^F$

Average distance between two words

$$= F(\tfrac{1}{2}N')(\tfrac{1}{2}N')/(\tfrac{1}{2}N'(N'-1))$$

$$= \tfrac{1}{2}F/(1-1/N')$$

$$\simeq \tfrac{1}{2}F$$

Now, $\frac{1}{2}F < 4n/9-1$ for all $n < 300$. Therefore, in the range $n \leqslant 79$, where the M code has greater efficiency than the fixed place code, it also has a greater average distance between words.

e.g. n = 79:

M code: No. of words $2^{61.1}$ Av. distance = 34.1

Gilbert's code: No. of words $2^{61.0}$ Av. distance = 30.5

3.16 Method of Mechanisation

We have now demonstrated everything about the M code except that it could be useful in practical applications. We shall not demonstrate here the ways that information can be adapted for transmission via an M code format; this subject will be treated fully in section 4. However, to complete this section, we shall show that, if information is being transmitted using an M code, or a subset of one, synchronisation can be regained at the first correctly received word after a loss of synchronisation.

3.16.1 To Achieve Synchronisation

Suppose that, after a burst error of length greater than the error-correcting abilities of the code can handle, the signal is picked up at an arbitrary point in a word. (It is perhaps desirable that we attempt to confirm in some way that we are considering symbols from the transmitted message and not those at the end of the burst, e.g. If the burst is a long string of zeroes, wait until a unit has been received before beginning the following operations).

Once the burst pattern has ended, the machine scans the incoming symbols until it finds a zero followed by a unit.

By the construction of the M code, this must either represent a 01 herald of a word or a doublet in the middle of a word.

We treat the n digits beginning with this 01 as a possible word, and scan it for 01 boxes. In the former case, none will be found; but in the latter case, we will find one corresponding to each doublet in the originally transmitted word, after the one on which we have fixed, and also one for the 01 herald of the next word.

This latter 01 box will obviously be the last found in the word under operation, and so, if we shift our synchronisation mechanism to begin again from this point, synchronisation will have been regained. It must be noted that random errors not part of the burst may upset this procedure, and so the use of extended scanning (see 3.16.3) is desirable. And so we can summarise the operation thus:

Fix on the first 01 combination found, and scan the (n-2) symbols after this for 01 boxes. If none are found, the 01 combination is a herald; if some are found, the last of these boxes is the herald of the next code word.

This operation will obviously regain synchronisation in time to decode the first complete code word, provided that the word is correctly received.

3.16.2 Error Correction

Once synchronisation has been attained, we can deal with errors in the herald by methods similar to those given later (in section 3.16.3). However, it will be appreciated that errors can affect the process outlined above. So we must emphasise that synchronisation will be regained at the first complete word only if it is correctly received.

3.16.3 Additional Safeguards

These will be discussed later in more detail (see section 5), but briefly, they fall under two headings:-

(a) Extension of the herald.

Obviously, the longer the herald, the less possibility of a false herald; so a long herald can be an aid to synchronisation, but a herald longer than necessary costs us valuable arbitrary positions.

(b) Extended scanning.

If, after finding a herald sequence, we scan the n digits after the sequence, we shall expect to find another herald sequence at the end of this scan. Failure to do so makes it probable that our herald sequence is not the herald of a code word. This is advisable, since it costs nothing but a longer scan, and guards against the possibility of errors in the first received word causing a false synchronisation to be obtained.

4. METHODS OF ENCODING FOR A BINARY CHANNEL

4.1. General

In section 3 we showed that any code of length n which consists of a subset of the M code is synchronous. For a code to be such a subset, all the words in the code must have a 01 first box and no 01 box among the other n boxes. So in this section, we consider various methods of adapting information symbols (and parity checks) so that they obey the above format.

These methods fall into two broad classifications. Firstly, we can take an error correcting code of any type and, by use of the transformation function ϕ defined below, transmute the code into one which obeys the M code format. Alternatively, we can take our information input, represented by a binary number, and, by applying ϕ only to this, arrive at a set of boxes which obeys the format but which has no redundancy. We must then construct our parity checks so that they also obey the format.

It should be noted that remarks about information symbols, especially as regards efficiency of a code, also include parity checks, whichever classification we are considering. This is feasible, since the amount of information which can be sent via a channel and the rigour with which it can be checked are dependent only on the channel and on the error-correcting ability of the code, and not the code format. And so information symbols and parity checks play equivalent roles in determining the efficiency of a code.

So, in this section we consider "specialist" codes, i.e. codes which give their best efficiency only for certain values of n. In order to transmute the code (or set of information symbols) into the M code format, we split each word into sets of three consecutive binary symbols (triplets) and apply ϕ to each triplet in turn. We then add "formatized" parity checks, if necessary, (see later) and transmit the resultant symbols.

This addition of parity checks, not necessary in the former classification, can be accomplished in this latter case by regarding the 00, 10 and 11 boxes as respectively equivalent to 0, 1 and 2 (mod 3) and constructing the parity checks to give parity modulo 3. The obvious drawback is that we are using two symbols to obtain one parity check, but this is not serious in small-redundancy codes.

Codes constructed in these ways have the advantage that, once synchronisation has been attained, the one-third of all the random errors which result in a 01 box can be immediately located. Another advantage is the minimum number of fixed places. This means that the information symbols are spread over a larger proportion of the word, and so it is not possible for a burst of a certain length to affect as many information symbols as in the case of the fixed place code.

We will now develop a transformation which acts upon binary digits and represents them in a form suitable for transmission in a code which is a subset of an M code. We must apply some transformation since if we use arbitrary binary information, it is certain that many boxes other than the first will contain 01.

4.2. ϕ and The Ternary System

We define the transformation function ϕ in the following way:

ϕ is a function which, when acting upon a set of three binary digits (the sets being representable in the normal way by the integers 0, 1 ,... 7) transmutes them into a set of 4 digits, divided into two boxes, as in the following table.

If we consider the representation of the boxes 00, 10 and 11 by the numbers 0, 1, 2 respectively, we can obtain a number equivalent to a set of these boxes (e.g. 1011 is equivalent to 12, which represents 5 in a ternary system analogous to that which gives the binary equivalent of a number). This number will be called the ternary equivalent of a set of boxes, usually a pair of boxes.

No.	Binary Representation x	$\phi(x)$	Ternary Equivalent
0	000	0000	0
1	001	0010	1
2	010	1000	3
3	011	0011	2
4	100	1110	7
5	101	1111	8
6	110	1100	6
7	111	1011	5
		1010 has no representation	

So, given any message of length 3L, we split it up into L triplets, apply ϕ to each of them, and we obtain a message of 2L boxes, none of which is 01. To ensure that such a word obeys the M code format, we only have to give it a fixed 01 herald and a last symbol, which can be an overall parity check on the word.

Note that the original message may or may not contain built-in error-correcting apparatus.

We now consider the following difference tables of triplets and their mappings under ϕ. The element in the jth row and the ith column of the triangular matrix gives, in each case, the number of places in which the ith and jth words differ, i.e. the Hamming distance between them.

0	000		0	0000	
1	001	1	1	0010	1
2	010	12	3	1000	12
3	011	211	2	0011	213
4	100	1223	7	1110	3223
5	101	21321	8	1111	43321
6	110	231212	6	1100	231412
7	111	3221211	5	1011	3221213
				1010	21121221

It will be observed, then, that the distance between $\phi(x_1)$ and $\phi(x_2)$ is at least as great as that between x_1 and x_2. (Thos differences which are greater are underlined). Thus, if we define the ϕ-message to be the mapping of a message of length 3L under ϕ, the ϕ-message should be at least as susceptible to error correction as the message itself. This will be proved in section 4.4.

Also we notice in passing that, since the Hamming weight of $\phi(x)$ is the same as that of x (modulo 2), the overall parity check can be made either on the ϕ-message or on the input message.

We will assume hereafter that there is negligible probability of error in effecting this encoding, and concern ourselves only with errors that occur in the transmission of the ϕ-encoded word. It will be shown that the redundancy introduced by the lengthening of the message under ϕ-encoding (from 3L to 4L) to achieve the required form will also increase the burst-error correction capabilities of the code.

3.3 Notes

Since by using this system we are only getting 3 input information symbols out of every 4 ϕ-encoded symbols which are transmitted, the maximum efficiency of a code which uses this system is 3/4.

It will be shown in section 8 that if we try to define any function which translates blocks of input information symbols into packets which use "boxes" of 3 or more symbols as building units, then the resultant code is not as efficient as the one defined by ϕ.

3.3.1 Why 1010 was left out of $\phi(x)$

We should perhaps explain our choice of 1010 as the one left out in the table of $\phi(x)$. This was done for two reasons. Firstly, 1010 is more likely to produce herald sequences, both by itself and in combination with the box pairs on either side of it, than any other box pair. Thus omission of this particular box pair does more to obviate the possibility of a false synchronisation due to a transmission error than the omission of any other box pair.

The other and more important reason for the omission of 1010 can be observed from the table. If we test the Hamming distance between 1010 and any other box pair in the table, we find the distance is never more than 2 and is in half the cases only 1.

This means that if it was allowed in a transmitted message of ϕ-encoded data, single errors in the ϕ-message would be hard to correct. In fact, since for every possible triplet x_1 there is another triplet x_2 whose Hamming distance from x_1 is 3, the use of 1010 as a box pair representing x, would mean that the Hamming distance of $\phi(x_1)$ from $\phi(x_2)$ would be less than the Hamming distance between x_1 and x_2.

This argument is sufficient to remove 1010 from the set of box pairs used in $\phi(x)$. However, we can also see that, among the 9 possible box pairs (including 1010), the probability of a chosen digit occurring in any position is twice that of the occurrence of the other (e.g. if we observe the first positions of box pairs, 6 contain 1 and 3 contain 0; observing the second positions 3 are 1 and 6 are 0 etc.).

In any code, the average distance is maximum if in the table of code words 0 and 1 occur as nearly as possible the same number of times in every position. Since this is so, 6 to 3 in every column is not a very efficient ratio. However, removal of 1010 from the set makes the ratio 5 to 3 in every column; removal of any other box pair would leave the ratio 6 to 2 in at least one column.

We have now shown that the eight elements in $\phi(x)$ are the ones we should use to represent the eight triplets; we have not

justified our particular choice of representations. They have no justification, except that no other choice can be better for our purposes; many choices do not satisfy the condition that the Hamming distance between $\phi(x_1)$ and $\phi(x_2)$ be at least as great as that between x_1 and x_2.

It should be emphasised at this point that our choice of 01 as the herald of an M code was purely arbitrary; 10 would have given us a similar code, the 'mirror-image' code of the M code. In the same way, different choices for the representations will give codes which are different in appearance but identical in properties.

4.4. Error correction

We will now consider methods of error correction of ϕ-messages.

Using the representation of the 3 boxes by numbers, we can employ boxes as parity checks, in the sense that we assume that, for example, given 11, 10 and 11, then 10 gives parity, since $2 + 2 + 1 + 1 = 0 \pmod 3$, where these numbers are the ternary equivalents of the single boxes concerned. Thus we can add parity checks, in the form of these boxes, and still have a word which obeys the M code format. So, in codes where the parity checks are added after the input information symbols are ϕ-encoded, the error correction can be accomplished in the normal way, except that the

parity is modulo 3. However, where the complete code has been ϕ-encoded, and synchronisation has been attained by addition of a 01 herald and an overall parity check we have to obtain the original input information symbols (or as close as we can get) from the ϕ-message before beginning error correction.

Since we have assumed that synchronisation has been attained, we know the sets of four symbols in the received ϕ-message which are purported to have come from each of the triplets. So we take each pair of boxes and apply the inverse function ϕ^{-1}. We will show that there are three possible results of this application.

Firstly, if the pair of boxes has been correctly received, then, under the supposition of error-free application of ϕ^{-1}, the correct tripletwill be obtained with no further complications. But, if an error (or errors) have been made then one of two things can happen. The received signal, $\phi(x) + S$, where S is the error syndrome, can be either one of the combinations that ϕ^{-1} can decode or not.

We define the error syndrome of the ϕ-message in the normal binary way and not in terms of its ternary representation, since the error pattern is better represented by the former definition, and a knowledge of the probable error patterns helps in determining the parity check system of the original code.

If $\phi(x) + S = \phi(y)$, i.e. $\phi^{-1}(\phi(x) + S) = y$, then, since the Hamming distance between $\phi(x)$ and $\phi(y)$, is, as shown by the triangular distance matrices in section 4.2, not less than the Hamming distance between x and y, application of ϕ^{-1} to the ϕ-message will not increase the number of erroneous positions.

There are 16 possible sequences of 4 places, each arbitrarily 0 or 1. The transformation ϕ only uses 8 of these, the others being 1010 and those with a 01 box in them.

Thus, ϕ^{-1} only recognises these eight of the sixteen possible sequences, and, once an error has been made, only seven of the remaining fifteen sequences will be recognised. These which will not be recognised are 1010 and those which contain one or more 01 boxes.

If ϕ^{-1} cannot decode $\phi(x) + S$, then we have to decide on a method of decoding. This decision will depend on a number of factors major among which will be the error patterns with which we are most likely to be dealing. Although every different transmission system will have its own optimum method, we will give now two methods for particular systems which can be adapted in other cases.

4.5 Example

Suppose we require a synchronous code of length $n = 2m + 3$ which will correct one error in a word. This is the kind of code to use if the system of transmission gives only random errors, and so

few of these that we can discount the possibility of two errors in the one word. Suppose also that we want to send 3L information digits per word.

4.5.1 We can ϕ-encode the information digits to obtain 4L symbols in 2L boxes. Applying the conditions of the single-error-correcting ternary Hamming code to the boxes used, we see that we can construct a code which will correct any erroneous box by the addition of, say, b boxes. After the addition of the 01 herald and the last symbol, we find that $n = 4L + 2b + 3$.

Thus, we have obtained a synchronous code of length $4L+2b+3$. with $3L$ information digits, which will correct at least one error per word.

We notice that the last symbol has not been used yet. By a certain application of this symbol (e.g. make it equal to the sum of the symbols in the odd positions), we can correct a burst of length 2 or less in a word.

4.5.2 Alternatively, we add a minimum of three digits to the 3L digits and construct a single-error-correcting linear code. We then ϕ-encode the 3L' digits, add the herald and make the last parity check digit the sum of all the digits in the odd positions.

Now assume there is no more than one error in transmission; if $\phi(x) + S$ is one of the eight box pairs in the set, i.e. $\phi(y)$, then $\phi^{-1}(\phi(x) + S) = y$, and y will contain one error; so it can be

corrected; if ϕ^{-1} will not operate on $\phi(x) + S$ due to the formation of a 01 box, the correctness of the last parity check added to the ϕ-message will determine if the mistake is in the first or second digit of such a box; and if ϕ^{-1} will not operate because of a 1010 combination, choose either 1011 or 1110, decode and check; the wrong choice has been made, if there is still an error.

This is a code of length 4L + 7 with 3L input information digits. As can be seen, the decoding requires moderately complicated apparatus; but the encoding is simple, and this is the portion that has to be simple.

4.5.3 Adaption of the M code format

If we wish to use an M code with an odd number of boxes, we cannot ϕ-encode binary information into the last, odd, box. But we can fix a unit in the first position in the box, and leave the second position arbitrary; the fixed unit ensures that the box is never 01.

4.5.4 Efficiency of a ϕ-encoded M code

Since we are putting three information digits into the four positions of two boxes, no ϕ-encoded M code can ever have an efficiency greater than 3/4. Using the approximation of $\log_2 3$ as 1.5, as mentioned in section 3.14.1, in the expression for the efficiency of the M code, gives the same results as we get by

considering the ϕ-encoded M code. So we will use this approximation in future when we are considering a code which is likely to need ϕ-encoding; this applies particularly in section 5.

4.5.5 Definition

We will define $(A)_m$ as the smallest integer not less than A.

4.6 Notes

We see in section 4.5 that we can have different values of n for two codes with the same number of input information digits. In the following sections, we will concern ourselves with the following problem:- "Can we construct synchronous codes which possess similar properties to well-known linear and cyclic codes, and which can be treated in the same way."

To this end, we will construct codes in various ways, patterned on methods of constructing binary codes, except that we use three types of boxes instead of two types of symbols (1 and 0) as building units.

We will show that these various types of codes only attain their greatest efficiency for certain values of n, and try to find which code is best for given n.

4.7 Reed-Muller Code (Recurrent code) /8/ and /9/

In this class of codes we construct the (p + 1) basis vectors as binary representations of the numbers in ascending order, and then delete the last row, e.g. for p = 2

v_1 0 0 0 0 1 1 1 1

v_2 0 0 1 1 0 0 1 1 Delete v_3

v_3 0 1 0 1 0 1 0 1

This leaves us with p basis row vectors. From the analysis given for such a code, we know the length is 2^{p+1}, the number of information digits is $\sum$comb(p,i) where the summation is over $0 \leqslant i \leqslant R$, where we require the code to correct all errors of syndrome weight $2^{p-r} - 1$ or less, and detect all errors of weight 2^{p-r}.

As can be seen from the example given above, deletion of t he last row means that the (2j+1)th and (2j+2) digits are the same in all the basis vectors, all j, then they are also the same in any vector which is the product or linear sum of these vectors.

Thus every box of a word encoded as in a Reed Muller code using these basis vectors is composed entirely of 00 and 11 boxes. Thus, if we add a 01 herald and a last symbol which is a parity check on the sum over j of the (2j+1)th symbols, the code so formed is a subset of the M code of length $2^{p+1} + 3$.

Owing to the additional parity check the code can now correct errors of syndrome weight 2^{p-r}. In construction, the code is similar to a recurrence code, and any recurrence code which uses adjacent recurrence can be adapted in the same way into a synchronous code.

Also, mechanisation of such a code is extremely simple, since, apart from the herald location and parity check machinery, each input symbol is merely sent twice, in two consecutive positions. Thus, if we are willing to reduce the efficiency of any code (not synchronous) by half to achieve synchronisation, this is an excellent method of achieving this object without using bulky machinery. Decoding, also, is extremely simple, because of the normal properties of recurrence codes.

4.8 Fire Code

Codes of the class discovered by Fire /1/ are particularly adaptable to the ϕ transformation, and the results are quite efficient. In this section, we will define Fire codes, and then prove a lemma necessary to our main objective, a class of codes analogous to the ternary Fire codes; this latter class will be constructed in the next section.

A Fire code (in the binary sense) is defined to be a code of length $n = \text{L.C.M.}(a, f)$, generated by a polynomial $g(x)$ of the form $g(x) = R(x)(x^a - 1)$ where $R(x)$ is an irreducible polynomial over GF(2) of degree R whose roots have order f.

Using the representation of a binary code word as a polynomial modulo $x^n - 1$, we say that a polynomial $g(x)$ generates a code if and only if for each code word the corresponding polynomial $G(x)$ has the form $G(x) = g(x)h(x) \pmod{x^n - 1}$, where $h(x)$ is another polynomial.

A Fire code as described above has the following properties:

(i) The number of check symbols is $a + R$, i.e. The number of information digits is $n - a - R$.

(ii) It can correct burst errors of length b or less, and simultaneously detect burst errors of length d or less, provided

(a) $b + d \leqslant a + 1$

(b) $R \geqslant \min(b, d)$

(iii) The parity checks associated with the $x^a - 1$ factor are interlaced parity checks spaced n/a apart.

We now give, with only a heuristic proof, the following rather trivial lemma. Although applicable to all "box" burst-correcting codes, it is of particular use in determining the length of a burst correctable by a ternary Fire code of the type given in section 4.9.

4.8.1 Lemma

Assuming that a word is encoded in boxes, in order to correct a burst of length b, it is necessary and sufficient that we

can correct a burst over b' boxes, where b' = $(\frac{1}{2}(b + 1))m$.

Proof:

The critical case is the one where the burst extends from the $(2i + 2)$th digit to the $(2j + 1)$th digit inclusive. The burst involves $2(j - i)$ digits and $j - i + 1$ boxes.

4.9 A Ternary Fire Code System

Consider a primitive polynomial $R(x)$ over $GF(3)$ of degree f. Its roots are of order $3^f - 1$.

Choose a with only the restriction that $3^f - 1$ does not divide a.

We can then construct a Fire code generated by the polynomial $R(x)(x^a - 1)$ with the following properties.

Length = n' = L.C.M. $(3^f - 1, a)$

Number of parity checks = a + f

Therefore; Number of information digits = n - a - f.

We can correct any burst of length not greater than Min $(\frac{1}{2}(a + 1), f)$.

Now we proceed to represent the elements modulo 3 by the boxes 00, 10 and 11, and use these boxes instead of the elements modulo 3 in the Fire code given above.

We arrive at a sequence of n' boxes (2n' binary digits), of which n' - a - f boxes are arbitrarily 00, 10 or 11, and a + f are used as parity checks. Thus the code can carry $3(n' - a - f)/2$ binary information digits under ϕ-encoding.

Also, this sequence of n' boxes can correct any burst error which affects no more than Min $(\frac{1}{2}(a + 1), f)$ boxes; so, by lemma 4.8.1., it can correct any burst of a length which affects strictly less than twice this number of binary digits.

To construct a synchronous code, we place this sequence of boxes after a 01 herald and add a concluding parity check.

The resultant code has a length of $2n' + 3$ digits; the number of information symbols is the greatest integer not greater than $3(n' - a - f)/2$; and the code can correct any burst of less than $2.\text{Min}(\frac{1}{2}(a + 1), f)$. By the properties of Fire codes, we can also detect another burst whose length obeys the same restriction.

The optimum efficiency of the system occurs when $\frac{1}{2}(a + 1) = f$ i.e. when $a = 2f - 1$ (Notice $3^f - 1$ does not divide $2f - 1$).

In these circumstances we obtain a code with

$$\text{length} = 2(3^f - 1)(2f - 1) + 3$$

$$\text{Number of information symbols} = 3(3^f(2f - 1) - 5f + 2)/2$$

$$\text{Maximum length of correctable burst} = 2f - 1$$

$$\text{Efficiency} = 3/4 - 3.\log n/(n .\log 3) + O(n^{-1})$$

As an example of this, choose $f = 2$, $a = 3$.

We need for the code generator an irreducible polynomial over GF(3) of degree 2, whose roots have order 8; such a polynomial is $x^2 - x - 1$.

From the formulae, we obtain

n = 51

No. of information digits = 28.

Length of correctable burst = 3 or less.

Alternatively, we could apply the ϕ-transformation to a complete Fire code. This has certain advantages in the correction of long bursts, but for small bursts the difficulties encountered in ϕ-decoding erroneous messages (see section 4.4) do not make this procedure worth while.

4.10 Cyclic code systems (Hamming Codes)

This method can be used to correct random errors in a received ϕ-message. However, if the probability of random errors is sufficiently small, it is best to completely ϕ-encode a burst-error correcting code which will correct bursts of length 2 or less and then allow this to decipher any double boxes upon which ϕ' will not operate (see section 4.5). However, in the case of frequent random errors, we can use a cyclic type of code using the admissible boxes as below.

If we consider the representation of 00, 10 and 11 as elements modulo 3 as in section 4.9, then we arrive at a code with statistics as below.

It was shown by Hamming /5/ that, for a cyclic code, using the Hamming bound for the number of parity checks necessary to correct p errors, we need at least

$$\log_3 \left(\sum_{i=0}^{p} 2^i \, \mathrm{Comb}(n,i) \right) = Q(n,p) \text{ parity checks}$$

Thus, adding the herald and the concluding parity check in the usual way, we arrive at the following statistics.

The number of parity checks

$= P = Q(\frac{1}{2}(n-3), p)$ + 1 for the concluding parity check

The number of information digits

$= 3(n - 3 - 2P)/4$

The efficiency of the above code is

$(3 - 6P/n + O(1/n))/4$

or, since $P = \log_3(n^p/p! + O(n^{p-1}))$

Efficiency is $(3 - 6p \log_3 n/n + O(n^{-1}))/4$.

4.11 Burst-Locating Codes

In this section we will consider a recurrent code. This recurrence, however, will be of boxes, not single symbols. We arrange the boxes in such a way that any burst error will be automatically located, and then we can use the recurrence relations to correct it. We define a burst-locating code in the following way.

Consider a code made up of 2uv boxes for some u and v; each box is either 00, 10 or 11. This code, as it stands, is not synchronous since the herald has not yet been added. We now employ a recurrence relation which makes the ith and the $(u(v(2 - j) + i - 1) + j)$th box the same, where

$(j - 1)v < i \leqslant jv$ and $j \leqslant u$,

i.e. if $u = 2$, $v = 3$.

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

It can be shown that the inverse relationship is that the i'th and the $(v(u(1 - j') + i' - 2) + j')$th box are the same, where $(j' - 1)u < i' \leqslant j'u$ and $v < j'$.

Now, if $i \leqslant uv$, it can be seen that the minimum number of boxes between the ith box and its partner is $u + v - 2$ (e.g. in the case above, two boxes between the fourth and its partner, the eighth). However, the minimum burst length we cannot be sure of correcting is the shortest burst which will affect two boxes whose partners are only the same distance apart as the boxes themselves, or less. The minimum distance (in boxes) between such a pair of boxes will be shown in the next lemma to be $U + 1$, where U is the greatest integer $< \text{Min}(u, v)$. Now the shortest burst which will affect two boxes U apart is $2U + 2$.

Thus, using the recurrence property of the code, we can correct all bursts of length $2U + 1$ or less.

If we specify that none of the 2uv boxes is Ol, and we surround these boxes with a Ol herald and a last parity check, we have a synchronous code which will correct all bursts of length 2U + 1 or less. Correct use of the last parity check gives us the ability to correct all bursts of length 2U + 2 or less. And, since a burst which affects only two boxes separated by other boxes, the critical case considered in the next lemma, is more usually considered to represent random errors than a burst, we can correct a large percentage of bursts of all lengths less than half the word length; this is obviously the theoretical maximum length of a burst which could be corrected.

It has already been noted that the critical case for burst length looks more like random errors than a burst, and so, if random errors are more probable than burst errors, this code is not a good one to use.

4.12 Lemma

If, in a burst-locating code of the type described in section 4.11, where $1 < u < v$, we consider any two boxes, the pth and the qth, such that

$$|p-q| \leqslant U < u$$

then the boxes related to p and q, the p'th and q'th respectively, are such that

$$|p' - q'| > U$$

Proof:

Case 1. Suppose $p > q$

$1 \leqslant q < p \leqslant uv$

By the construction of the burst-locating code, we know

$p' = u(v(2-j_1) + p - 1) + j_1$,

$$(j_1 - 1)v < p \leqslant j_1 v$$

$q' = u(v(2-j_2) + q - 1) + j_2$

$$(j_2 - 1)v < q \leqslant j_2 v$$

We know $j_1 \geqslant j_2$

$$|p' - q'| = |uv(j_2 - j_1) + u(p - q) - (j_2 - j_1)|$$

$$= |(uv - 1)(j_2 - j_1) + u(p - q)|$$

Case 1.1 If $j_2 = j_1$,

$$|p' - q'| = |u(p - q)| \geqslant u > v$$

Case 1.2 If $j_1 > j_2$,

$$|p' - q'| = |(uv - 1)(j_1 - j_2) - u(p - q)|$$

$$\geqslant uv - 1 - u(u - 1)$$

Since $v > u$,

$$|p' - q'| \geqslant u(u + 1) - 1 - u^2 + u$$

Therefore, $|p' - q'| \geqslant 2u - 1$

Note: if v were equal to u, we would get $|p' - q'| \geqslant u - 1$

Case 2. $uv < q < p \leqslant 2uv$

The proof is similar to case 1, except that the inverse relationships are used.

Case 3. $q \leqslant uv < p < 2uv$

$$p' = v(u(1-j_1) + p - 2) + j_1,$$

$$(j_1 - 1)u < p \leqslant j_1 u$$

$$q' = u(v\ (2 - j_2) + q - 1) + j_2$$

$$(j_2 - 1)\ v < q \leqslant j_2 v$$

Since $|p - q| < u < v$ we know $j_1 = v + 1$ and $j_2 = u$

$$|q' - p'| = uq + 2uv - u^2v - pv + uv^2 + v + 1$$

Since $q \geqslant uv - v + 1$ and $p \leqslant uv + u$

$$|q' - p'| \geqslant u^2v - uv + u + 2uv - u^2v - uv^2 - uv + uv^2 + v + 1$$

i.e. $|q' - p'| \geqslant u + v + 1$

QED.

4.12.1 Corollary

If we want to use a burst-locating code with $u = v$, we can see from Case 1. of the proof of the preceding lemma that we require $U \leqslant v - 2$. Thus we can only correct bursts of length $\leqslant 2u - 3$.

4.13 Note

So far no method of correcting single or double errors in one box have been given for the burst-locating code. However, if the code is transmitted with herald and concluding parity check, as in the M code, the use of the concluding parity check as a parity check over all the digits in the first $(n - 3)/4$ boxes will be sufficient in all cases except the one in which a 11 box is changed to a 00 box or vice versa. Since, in most systems, the probability of the former occurrence is so much greater than that of the latter, we should include a mechanism that concludes that 11 was sent in this case. But, if the probability of error due to this is sufficient, then boxes with fixed units in the first position and parity checks in the last positions may be added to the code.

4.14 Comparison of Codes

In this section we will, for the sake of simplicity, disregard any necessity for restriction on n to give optimum results (e.g. "Integral part" signs omitted).

We will compare single-error correcting codes where practicable. However, we should remember that the burst-error correcting and Fire codes given will correct bursts of length greater than 1.

CODE	Length n	No. of inf. digits I	Efficiency I/n
Gilbert's	$a^2 + 1$	$a^2 - 2a - 2$	$1 - 2/\sqrt{n} + O(1/n)$
Reed-Muller	$2^{p+1} + 3$	2^p	$\frac{1}{2} - 3/(2n)$
Fire Code	$2a^2 + a + 3$	$\frac{1}{2}(3a^2 - a)$	$\frac{3}{4} - 1/\sqrt{8n} + O(1/n)$
Cyclic code	n	$\frac{3}{4}(n - 6)$	$\frac{3}{4} - 9/(2n) + O(1/n^2)$
Burst locating	$4uv + 3$	$\frac{3}{2}uv$	$\frac{3}{8} - 9/(8n)$

In the next table, we give actual values of n, I, which can be obtained for the various codes when $n \leqq 50$ or $n \leqq 35$ (ε = no. of errors code can correct).

Code	n	I	ε	n	I	ε
Fixed place	50	12	5	37	13	3
Reed-Muller	67	26	5	35	11	3
Fire code	43	18	4	35	12	3 + detection
Cyclic code	51	16	5	35	12	3
Burst locating	51	18	5	35	12	3

It will be noticed that some of these values do not obey the formulae given above. The reason is that, for small n, the integer signs which have been omitted play a large part.

4.15 Notes

4.15.1 Reed-Muller

This code has an extremely simple mechanisation. However, the extreme restriction on length means that it is only of use in certain very special cases.

4.15.2 Fire codes - General

Remember the conditions for a Fire code, i.e. the code is generated by $R(x)\ (x^a - 1)$, where $R(x)$ is a polynomial of degree R whose roots are of order f.

Length = LCM(a, f)

Number of parity checks = a + R

Error-correcting ability:

Any burst of length $\leqslant$ Min $\left\{\frac{a+1}{2}\ ,\ R\right\}$

Thus, Fire codes can be constructed in a similar way to that given above for a wider range of a and f than that given above; however, the length of the correctable bursts does not then reach the optimum value.

4.15.3 Synchronisation

Since these codes are all subsets of M codes, they all regain synchronisation on the next correctly received word.

4.15.4 Efficiency

Note that the upper bound of the ratio I/n for an M code is 3/4. This follows directly from the fact that we have only 3/4 of the 4 possible choices with which to fill a given box. Note also that all the codes given approach this bound as n becomes large, but that for n moderately small ($\leqslant 79$) the full potential is not realised.

4.15.5 Fixed place code

The values for the number of information symbols given in the table were obtained from the Hamming bound.

4.15.6 Error-correcting ability of Reed-Muller code

It will be noticed that the figures given for the number of errors which the Reed-Muller code can correct appears to be high. This is due to, first of all, the additional parity checks and mainly, the recurrence in the code which gives twice as many determinations of each symbol as in the normal Reed-Muller code.

5. REFINEMENTS AND THE K CODES

5.1 Error correction of the herald

Since the herald of an M code is only used to obtain and hold synchronisation, it is a waste of information digits to extend the herald. For the same reason, parity checks which apply only to the herald, and are therefore also fixed, are to be avoided if possible. Therefore, we have two alternatives - to use parity checks which include information digits as well as the herald, or to extend the herald by fixing extra positions and see if we can derive some compensating benefit for the information digits wasted.

Now, synchronisation need only be regained on the first correctly received word, and so the herald of the first word to go through the decoder will be correct. After this, extended scanning (section 3.15) can be used to correct single errors in the herald of the next word; more errors than this require a check on synchronisation of the code by the mechanism - this can be obtained by scanning the boxes in the word for a herald, with automatic correction if the herald is not found.

Therefore, correction of the herald can be accomplished by the mechanism installed to obtain synchronisation, provided extended scanning is used.

5.2 Use of an extended herald

Having just exploded the idea of a need for an extended herald, we proceed to revive it again. Since the pair 1010 has not been used in the ϕ transformation (section 4.1) at no time can three 10 boxes follow each other.

This means that the combination 0101 can only occur in the body of the code in one of the following contexts

0010,10 ..; 0010,11..;

..00,1011; or 10,1011; where the commas are not part of the word, but are used to show breaks between box pairs; and, since the total probability of this occurrence in an arbitrary sequence is 4 in 64 compared with the probability of 1/3 for a pair which can be mistaken for the herald of an M code, the chances of a wrong combination being selected as a starting point are considerably reduced. And, if we use 010101 as a herald, then the probability of a sequence identical with it in the body of the code is only one in 81, while use of 01010101 removes the possibility altogether, and even the probability of a random error creating a false herald is only one in 2048. Thus, depending on the probability of error in the system, the operation of synchronisation can be simplified mechanically. With only 01 used in synchronisation, a burst error of length sufficient to require resynchronisation may end half way through a 11 box, thus giving an erroneous 01 box which will be

picked up by the mechanism as a starting point, and which will not be corrected until the next word has been lost (unless extended scanning is used). With 0101, provided 01 is not used as an information box, synchronisation will definitely be regained at the first correctly received word, unless a random error (1 to 0) follows the burst error with one correct unit in between. This is a most unlikely event.

As can be seen above, the length of the herald can be adjusted to give any degree of accuracy of synchronisation required. The optimum length to use in any particular case must therefore depend upon the peculiaritiss of the system itself, for example, the expected error patterns, the error density, the availability of extended scanning, etc..

5.3 Summary

Let us consider the points for and against an extended herald in an M code.

5.3.1 Against

(a) It increases the length of a word without any compensating increase in information digits.

(b) The machinery which picks up synchronisation (and, after all, this is the only purpose of the herald) does not need the extension.

5.3.2 For

(a) It simplifies the machinery needed to obtain synchronisation.

(b) It lessens the possibility of an error in synchronisation.

5.4 Note

Extended scanning is useful in both cases to check the synchronisation. The possibility of an error in synchronisation with an extended herald and extended scanning is negligible, especially if this extended scanning takes in the next two heralds (see section 3.15). However, for codes of considerable length this is not normally practicable.

We have shown that the advantages and disadvantages of the extended herald for the M code almost balance; however, if the machinery is available, the short herald is preferable - we have, so far, obtained no compensation for the wasted space. But the fact that the herald is easier to read means that we may be able to relax our restrictions on the M code; and this indeed happens to be the case.

5.5 K Code

All this brings us back to our starting point - do we or do we not use an extended herald, and, if we do, can we obtain

any benefit from it? So far, we have only obtained assistance in the synchronisation process. However, in systems with small probability of error, this may very well be unnecessary, and only result in loss of information digit space.

It is in this latter case that the code defined below is most helpful. It is in reality a cross between an M code and a Gilbert code, and its distinction from the Gilbert code is that the space totally wasted by the former in the fixed unit positions is at least partially used in the K code.

5.5.1. Definition of a K code

We shall define here the K code of length n, where n is odd.

Every word in a K code begins with a herald consisting of a sequence of u 01 boxes, where $u \geqslant 1$.

We stipulate that every other string of 2u consecutive digits in the code word begins with a box, i.e. there is either 00, 10 or 11 in each of the position pairs 2ku + 1, 2ku + 2, k an integer and $k \geqslant 1$. Each of the other digits in a word is arbitrarily 0 or 1. This includes the last (n^{th}) digit in the case where n = 2ku + 1, some k. Thus the last string of arbitrary digits may contain 2u - 1 digits, although every other string of arbitrary digits contains 2u - 2

digits, e.g. a word in a K code is of the form

$$0101 \ldots 01b_1b_2xxx \ldots xb_3b_4 \ldots b_{2s-1}b_{2s}xx \ldots x$$

2u digits 2u digits 2u-1-2c digits

where the $b_{2i-1}b_{2i}$ represent the s boxes in a word and c is an integer, $0 \leqslant c \leqslant u-1$. The K code of length n consists of all words of length n which have this format. When we are talking about K codes, we shall restrict the use of the word boxes to mean the pairs $b_{2i-1}b_{2i}$.

The similarity of a K code to a Gilbert code lies in the fact that there is in both cases no string of arbitrary digits of as great a length as the herald; however, in a K code the possibility of a herald sequence other than at the start of a word is reduced, but not eliminated, by the device.

Note that there is no box at the end of a word in a K code, although there is a fixed digit at the end of a Gilbert code.

The similarity of a K code to an M code lies in the fact that they both use boxes to achieve synchronisation. This will be demonstrated in the proof in the next section.

In fact, it can be seen that if we choose u=1, the K code of a given length n is identical with the M code of the same length.

If u is greater than 1, the possibility of a herald sequence beginning at a position other than the start of a word is less than with a M code since the information digits (the arbitrary digits and the boxes) will form such a pattern in fewer cases. Thus this extension of the herald gives us a bonus; but we do not wish to claim this bonus if it means reduced efficiency.

We shall prove in the following sections that, far from reducing our efficiency, efficiency is greater for large u and n sufficiently large.

First, however, we should prove that a K code is synchronous. This is done by the following argument.

5.6 Theorem

The K code of length n, n odd, is synchronous.

Proof:

As with the M code, we shall prove that no false word is identical with a code word.

There is no possibility of a herald sequence beginning after the last box ($b_{2s-1}b_{2s}$) or before the first box (b_1b_2) of a word, because either the code word has a unit where the false word, to be identical with a code word, requires a zero (or vice versa) or the code word has its first box where the false word requires a 01 pair (or vice versa). Thus any false word which is identical with a code word must start between the first and last boxes of a code word.

Since a herald sequence is of length 2u, any 'herald' of a false word must overlap at least one box in a code word. If the false word starts at an odd-numbered position in a code word, its 'herald' will include a box of the code word where the false word, to be identical with a code word, needs a 01 pair.

Thus any false word which is identical with a code word must start in an even-numbered position of a code word, and must include the herald of the next code word.

Since n is odd, and since the false word begins in an even-numbered position in the false word, there are 2u consecutive 01 pairs beginning in odd-numbered positions in the false word.

But in a code word there is a box beginning in an odd position somewhere in every string of 2u digits.

Thus in this case the false word has 01 in two positions where a code word has a box. Thus no false word is identical with any code word, and a K code is synchronous.

We mentioned in section 5.5 that a K code does not need a box in its last two positions, although a Gilbert code has a fixed unit in its last position. We proved above that the lack of this concluding box does not upset synchronisation; however, it does give rise to a possibility of a sequence of more than u 01 pairs. In this case, we shall have to tell the receiver to fix on the last u 01 pairs of any sequence of more than u 01 pairs as a possible herald of a word.

5.7 Adaption of the Format

The K code format allows for any number of boxes, s, either odd or even; however, if we apply the ϕ transformation to binary information digits, we need an even number of boxes. If we want to use a K code with an odd number of boxes, we fix a unit in the first position of a box, or a zero in the second position of a box, and leave the other position arbitrary; this box now cannot contain 01. We then proceed to ϕ-encode binary information digits into the remaining boxes.

Normally, we ϕ-encode 3 binary information digits into 2 boxes; this means that every box is theoretically worth $1\frac{1}{2}$ binary digits, and so fixing a position in a box, which leaves the box able to carry no more than 1 binary digit, has cost us a theoretical $\frac{1}{2}$ digit. We shall in the following sections give the number of information digits which can be carried by a code in terms of s without regard as to whether s is odd or even; in some cases, such formulae give a fractional value for the number of infomation digits.

We can overcome this difficulty by specifying that the actual number of information digits which the code can carry is the greatest integer not greater than the value specified by the formula. We shall designate the formula value by the letter I.

5.8 Specifications of the K code

Let us evaluate our K code. It contains in all $2us + 2u + 1 - 2c$ positions; $2u$ of these are fixed in the herald, and another $2s$ of them are positioned in boxes. If we assume that the ϕ transformation is used on binary digits to fill the greatest possible even number of boxes, we obtain the following statistics:-

$$n = 2us + 2u + 1 - 2c$$

$$I = 2us - s + 1 - 2c$$

Thus we have lost no more than $2u + \frac{1}{2}(s + 1)$ digits to achieve synchronisation.

The ratio I/n, the efficiency of the code, is greatest when c is smallest i.e. $c = 0$. In these circumstances

$$n = 2us + 2u + 1$$

$$I = 2us - s + 1,$$

where the equality is really an equivalance, since every box is assumed able to carry $1\frac{1}{2}$ 'arbitrary digits'.

But if we cannot find a satisfactory pair (u, s) such that $n = 2us + 2u + 1$, then we must choose $c > 0$.

It is easy to see that the advantage of the K code of a certain length over a Gilbert code of the same length, and approximately the same herald length, is that, although the K code has twice as many restricted positions as the Gilbert code, it can carry 3 bits

of information in every 4 of its restricted positions, while the restricted positions in the Gilbert code are fixed and not available to carry any information. This results in a net gain in the number of information digits that can be carried of half the number of fixed units in the Gilbert code; this is at least one greater than the number of boxes in the K code, because of the last fixed digit in the Gilbert code.

In this argument we have assumed that we know the best choice for u and s for a given length; to aid the comparison we chose 2u as close as possible to the length of the herald of the Gilbert code. But this may not be the best choice; in the next section we will derive a formula which relates the best choice of u to the value of n.

5.8.1 An Approximation we Shall Use

We shall first of all develop the K code in its most efficient form; a K code in this form can only be used for transmitting information if binary information can be transmitted in the arbitrary positions and ternary information in the boxes at the same time.

To allow for simplicity in figuring, and so that we can quickly adapt the results to allow for the case when we must use information which is entirely in binary form, and hence ϕ-encode the information which is placed in the boxes, we will use the approximation of $\log_2 3$ as 1.5. This is accurate enough to illustrate the points we wish to make.

This approximation also facilitates adaption of the formulae to cover the ϕ transformation case, since it says in effect that we can carry (theoretically) 1.5 information bits in each box. This ties in precisely with the figure of 3 bits in 2 boxes which the ϕ transformation gives us; the only difference is that we assume 1.5 information bits, and not 1 information bit, can be carried in an odd box.

When we have developed final formulae using this approximation, we shall give formulae which use a more precise value of $\log_2 3$.

5.9 The Optimum Length of the Herald

We know that if $n = 2us + 2u + 1$ (i.e. $c = 0$) for some u and s, then the K code of this length is more efficient than that of length $2us + 2u - 1$ (i.e. $c = 1$), if the same values of u and s are used in both cases. However, we have not yet shown that there is not another choice of u, s and c which would not lead to more efficient codes in either, or both, cases.

In this section we shall find the optimum choice of u, v and c in terms of n. There is always more than one acceptable integer triplet (u, s, c) which solves $n = 2us + 2u + 1 - 2c$, provided n is odd.

The difference between n and I is

$2u + \frac{1}{2}s$ arbitrary digits, by our approximation.

(The exact value is $2u + (2-\log_2 3)s$).

The most efficient code of a given length n is the one generated by the triplet (u, s, c) which loses the least number of arbitrary digits, i.e. the one with $2u + \frac{1}{2}s$ minimum.

We will proceed to a solution of our problem by finding first of all particular values of n for which one triplet (u, v, c) gives a more efficient code than any other triplet. Once we have this information, it will not be difficult to discover other values of n for which one choice of u and s gives at least as efficient codes as any other choice.

If we minimise $2u + \frac{1}{2}s$ subject to $2us + 2u + 1 - 2c =$

n, $u \geqslant 1$ $s > 0$, $c \geqslant 0$

without insisting that u, v and c be integers, we obtain the solution

$$c = 0$$

$$s = 4u - 1$$

Consider the particular value of n which is equal to $2u(4u - 1) + 2u + 1 - 0$ for given u.

i.e. $n = 8u^2 + 1$

The number of 'digits' lost to synchronisation

$$= 2u + \tfrac{1}{2}(4u - 1)$$

$$= 4u - \tfrac{1}{2}$$

(We do not really lose half a digit; our boxes have cost us an approximation of half a digit).

Consider a code of length $8u^2 + 1$ with $(u + 1)$ 01 pairs for a herald.

$$2(u + 1)s' + 2(u + 1) + 1 - 2c = 8u^2 + 1$$

Therefore, $s' > 4u - 5$

$$L = 2(u + 1) + \tfrac{1}{2}s' \geqslant 4u$$

Consider a code of length $8u^2 + 1$ with $(u - 1)$ 01 pairs for a herald.

$$2(u - 1)s'' + 2(u - 1) + 1 - 2c = 8u^2 + 1$$

$$s'' > 4u - 3$$

Therefore, $2(u - 1) + \tfrac{1}{2}s'' \geqslant 4u$

Thus a K code with u 01 pairs for a herald is the most efficient K code to use for $n = 8u^2 + 1$.

We assume that, given a choice of two distinct triplets (u, v, c) which give the same values for n and which have the same efficiency, we would choose the triplet which provided the longer herald, the smaller number of boxes and the larger number of arbitrary digits, since the longer herald makes for easier synchronisation, and the less boxes and more arbitrary digits we have, the easier is our decoding procedure.

Since the best value to choose for u when $n = 8u_0^2 + 1$ is $u = u_0$, as shown previously, we know that, for progressively larger values of n, we should choose progressively larger values of u. We shall now find the best value to choose for

u for given n, keeping in mind the assumption made above that if two choices give equal efficiencies, we shall choose the larger value of u.

We shall only compare codes obtained with u and u + 1; we know that there is at least one value of n for which u + 2 and u do not give as efficient a code as u + 1, viz. $8(u + 1)^2 + 1$.

If the codes are of equal lengths and have equal efficiencies

$$2us + 2u + 1 - 2e = 2(u + 1)s' + 2(u + 1) + 1 - 2e' \qquad A$$

$$2u + \tfrac{1}{2}s = 2(u + 1) + \tfrac{1}{2}s' \qquad B$$

From B, $s = s' + 4$ C

We note in passing from C that s and s' have equal parity modulo 2; therefore, happily, we do not have to consider which one better suits the ϕ transformation, which we would have to do if they were of opposite parity.

The upper bound for the values of n for which we shall use u will occur when $e = 0$ (since if the K code obtained for a certain value of n by using u is more efficient than the K code obtained by using u + 1 for $e > 0$, the K code obtained for a greater n, $= 2us + 2u + 1$, by using u will also be more efficient than that obtained by using u + 1).

Consider $s = 4u + 1$, and suppose for the moment $u \geqslant 2$.

$2us + 2u + 1 = 8u^2 + 4u + 1 = n$

If $2(u + 1)s' + 2(u + 1) + 1 = 2c' = n$

$s' > 4u - 3$

i.e. $s' \geqslant 4u - 2$

For this value of n,

$$2u + \tfrac{1}{2}s = 4u + \tfrac{1}{2}$$

$$2(u + 1) + \tfrac{1}{2}s' = 4u + 1$$

Therefore, for this value of n, we should use u 01 pairs for a herald.

Now consider $n = 8u^2 + 4u + 3$

We know the best choice for s is now $4u + 2$, since $8u^2 + 4u + 1$ is the maximum length of a K code with u 01 pairs for a herald and only $4u + 1$ boxes.

Therefore, $2u + \tfrac{1}{2}s = 4u + 1$.

However, if $s' = 4u - 2$, $2(u + 1) + \tfrac{1}{2}s' = 4u + 1$ also, and

$2(u + 1)s' + 2(u + 1) + 1$

$= 8u^2 + 6u - 1 \geqslant 8u^2 + 4u + 3$,

since $u \geqslant 2$, except in the case of the M code, which will be discussed in detail later.

Therefore, we can choose $c' \geqslant 0$ so that the triplet $(u + 1, 4u - 2, c')$ satisfies the equation for n.

$$2(u+1) + \tfrac{1}{2}(4u - 2) = 4u + 1 = 2u + s$$

Therefore, this latter triplet gives as efficient a code as $(u, 4u + 2, c)$, and so we use it to construct our K code of length $8u^2 + 4u + 3$.

Unfortunately, the values of n for which we choose u Ol pairs for a herald have not been mentioned yet; there is one other case to consider.

The maximum length of a K code which can be constructed with $u + 1$ Ol pairs for a herald and $4u - 2$ boxes is

$$8u^2 + 6u - 1$$

Consider $n = 8u^2 + 6u + 1$

The minimum choice of s' is $4u - 1$

Therefore, $2(u + 1) + \frac{1}{2}s' = 4u + 1\frac{1}{2}$

However, if $s = 4u + 2$,

$2us + 2u + 1 = 8u^2 + 6u + 1$

Therefore it is possible to construct a K code of length $8u^2 + 6u + 1$ with u Ol pairs for a herald and $4u + 2$ boxes.

In this case

$$2u + \tfrac{1}{2}s = 4u + 1$$

Therefore we should use this latter construction where we want a code of length $8u^2 + 6u + 1$.

It can easily be shown that there are no values of n greater than this for which use of u 01 pairs for a herald gives a more efficient code than the use of u + 1 01 pairs.

By reducing the value of u by 1, we can find the upper bound on n for which we should use (u - 1) 01 pairs for a herald, and hence the lower bound on the values of n for which we should use u 01 pairs.

In fact, we use u 01 pairs for a K code herald if

$$8u^2 - 12u + 7 \leqslant n \leqslant 8u^2 + 4u + 1$$

or $n = 8u^2 + 6u + 1$

but $n \neq 8u^2 - 10u + 3$ (when we use u - 1 01 pairs).

When $u = 1$, $8u^2 + 6u + 1 = 8u^2 + 4u + 3$.

So we can still use our formula if, for

$n \geqslant 8u^2 + 4u + 5$, we use (u + 1) 01 pairs for our herald in all cases.

The only case we need consider is when u = 1. In this case the ϕ-encoded K code has the same efficiency as the ϕ-encoded M code; and in this case we choose the code with the longer herald.

We can therefore apply our formula for all values of u.

We should note at this point that our formula only holds if we are using binary input; if ternary input is available, the choice of u may be different in certain borderline cases.

We have now shown that for every odd value of n we can choose values of u and s which are optimum with regard to efficiency. We need this fact in the next proof; for the basis of our proof is that we cannot construct a more efficient synchronous code than the K code, at least if we attempt to construct it using a herald and boxes in certain set positions in a code word. This may not seem to be very important at the moment; but it will become important in section 8, where we shall see that, unless we broaden our conception of a herald, we cannot construct a more efficient synchronous code.

In the following theorem, a 'herald and boxes' code is one which has a herald composed of a certain number of 01 pairs and boxes in certain set positions throughout the remainder of each word.

5.10 Theorem

For a given odd length n, we can choose u and s with which to construct a K code which is at least as efficient as any 'herald and boxes' code of the length.

Proof:

Suppose we can construct a more efficient (herald and boxes) code. Let it begin with u 01 pairs and have s boxes in some set positions, which we shall not prescribe, throughout each word.

Suppose W and X are two consecutive code words in a message, and that a false word Z begins somewhere in the digit string WX. We shall define Z' as an arbitrary code word with which Z may or may not be identical.

Z cannot be identical with Z' if and only if some restricted positions of WX which are in Z correspond to certain restricted positions in Z', and are unacceptable in Z'. For example, if Z begins at the second position of W, Z begins with a 1, and Z' must begin with a 0. Thus Z' cannot begin in the second position of W.

The K code will be synchronous only if Z is prevented from beginning in all the positions in W from 2 to n (i.e. n - 1 cases).

The four cases when Z cannot be identical with Z' are the following:

(1) The first position of Z coincides with a 1 in the herald of W (u cases).

(2) The (2u)th position of Z (the last in its 'herald') coincides with a 0 in the herald of X (u cases).

(3) Z receives a box from WX in positions where Z' requires a 01 pair for its herald (us cases - one for each possible pair and box).

(4) Z receives a 01 pair from the herald of X in positions where Z' requires a box (again us cases).

Now these cases may not all be distinct, for example, both (1) and (3) may be satisfied for a given starting position, or (4) may be satisfied twice. But if we spread these cases as thinly

as possible, and ensure that Z is not identical with Z' for only one reason in any starting position, we can prevent a false word beginning in no more than (us + us + u + u) = 2us + 2u positions.

But we know that this figure must not be less than n - 1, if the code is to be synchronous. Thus a synchronous code with u 01 pairs and s boxes is of length at most 2us + 2u + 1 which is the length of the K code with the same u and s, and with c = 0.

This proves that the K code is as efficient as any 'herald and boxes' code for any value of n at which the K code attains the local maximum on its efficiency, as illustrated in section 5.9. The result must be true for other values of n as well, for if for any value of n we choose u and s which give the most efficient 'herald and boxes' code (i.e. 2us + 2u + 1 - 2c = n), we can construct a K code using the same values of u, s and c.

This completes the proof.

We have now shown that a K code is a highly efficient code, at least by comparison with any 'herald and boxes' code of the same length. Therefore, a calculation of its efficiency would be helpful in evaluating its efficiency compared with the upper bound derived by Golomb et al, and also in comparison with the other codes mentioned so far.

5.11 The Upper Bound on the Efficiency of a K Code

We have already shown in section 5.9 that the most efficient K code for given u and s is the one with c = 0; we also know that for given u, the most efficient K code has s = 4u - 1.

Thus the greatest efficiency attained by K codes constructed with u 01 pairs for a herald occurs when $n = 8u^2 + 1$. For this value of n, the efficiency of the K code is

$$1 - (2(1 - 1/n))^{\frac{1}{2}} + \tfrac{1}{2}n^{-1}$$

This is the value obtained by our approximation of $\log_2 3$ as 1.5.

If we require that binary data be ϕ-encoded to fill the boxes, the fact that our value of s is odd (= 4u - 1) means that we lose another $\frac{1}{2}$ of an 'arbitrary digit' (the last, odd, box can now only carry one arbitrary digit instead of the $1\frac{1}{2}$ 'arbitrary digits' allowed by our approximation. Thus the maximum efficiency of a K code for which binary data has to be ϕ-encoded to fill the boxes is

$$1 - (2(1 - 1/n))^{\frac{1}{2}}$$

However, if we can use the boxes fully, and not have to ϕ-encode binary data to fill them, we get maximum efficiency from our K code. This efficiency approaches

$$1 - (4K(n-1))^{\frac{1}{2}}/n + Kn^{-1}$$

where $K = 2 - \log_2 3$

It never reaches this bound, since s would have to be irrational, $((n-1)/K) - 1$, if it were reached. However, it is approached closely for certain values of u and s.

5.11.1 Comparison of Efficiencies

We can see why the K code of given n is more efficient than the M code of the same length, for most values of n; the efficiency of an M code is always less than 0.8, no matter what the value of n; but, as a general rule, the larger the value of n, the more efficient the K code we can construct for it.

The efficiencies of the various types of codes considered so far are

(i) K code (theoretical) $1 - 1.29n^{-\frac{1}{2}} + O(n^{-1})$

(ii) K code (ϕ-encoded) $1 - 1.41n^{-\frac{1}{2}} + O(n^{-1})$

(iii) Gilbert's code $1 - 2.00n^{-\frac{1}{2}} + O(n^{-1})$

(iv) M codes $.792 \quad + O(n^{-1})$

We showed in section 3.15 that the M codes were more efficient than Gilbert's codes for $n \leqslant 79$. We shall now show that the K codes are an improvement on the M codes for $n \geqslant 17$.

Since we can see from the formulae that the K codes are much more efficient than Gilbert's code for large values of n, it follows that the K code of a given length should be highly

efficient compared to any synchronous code of that length. We pursue this thought further in section 8.

5.11.2 Lemma

For $n \geqslant 17$, we can construct a K code which is at least as efficient as the M code of the same length, provided ϕ-encoding must be used.

Proof:

We observed previously, in section 5.5.1, that an M code is identical with the corresponding K code with one 01 pair for a herald. Since this is so, we can use the formula derived in section 5.9 to obtain the result stated above.

i.e. We choose u 01 pairs for a herald in preference to u - 1 pairs if $n \geqslant 8u^2 - 12u + 7$, except when $n = 8u^2 - 10u + 3$

i.e. When $u = 2$, $n \geqslant 15$ and $n \neq 15$. Q.E.D.

If ternary input is available, the K code is the more efficient for $n \geqslant 21$.

So far, so good. But we have so far only considered the values of n which gives local maxima of efficiency in establishing the above formulae. There is still a possibility that there are some values of n for which the most efficient K code of that length which can be constructed falls far short of this bound.

We have so far only proved, in section 5.10, that there is a K code of any odd length which is as efficient as any 'herald and boxes' code of the same length; we do not yet know if the 'herald and boxes' system of synchronous code construction is particularly efficient for all (odd) values of n.

This next theorem proves that the efficiency of the K code of any length approaches the upper bound. This is reasonable, since the two values of n which should provide the greatest disparity in efficiency are those for which $n = 2us + 2u + 1$, some u and s, and the value of n two greater than this; in this latter case we can use $s + 1$ boxes, and put $c = u - 1$

$$\text{i.e. } n' = 2u(s + 1) + 2u + 1 - 2(u - 1)$$

This is the only case where there are not two more arbitrary digits in the code of length greater by 2; but there must be at least 1½ 'arbitrary digits' extra, because there is another box in place of the two arbitrary digits.

If we remember the results of section 5.9, in some cases we may be able to choose a different value of u with which to construct the larger K code, and so obtain a higher efficiency than by using one more box; but the case we have illustrated is the worst that can occur.

5.12 **Theorem**

For any given odd length n, we can construct a K code of efficiency at least

$$1 - (4K(n-1))^{\frac{1}{2}}/n + O(n^{-1})$$

Proof:

The only values of n for which the most efficient K code that can be constructed is less efficient than the most efficient K codes that can be constructed for both n two greater and n two less are those values of n for which the most efficient K code is constructed with certain u and s and c = u - 1.

This is the same argument as in the preamble, except that it is stated differently; we now say that, since we can construct a code of length two less with one less box and the same number of arbitrary digits, this latter code will be more efficient than the code of length n; the code of length two greater has two more arbitrary digits, and is therefore more efficient.

If c = u - 1,

$$n = 2us + 2u + 1 = 2u + 2$$

$$= 2us + 3$$

This code loses $2u + \frac{1}{2}s$ digits to synchronisation. Its efficiency is therefore at worst

$$1 - (2/n)^{\frac{1}{2}} + O(n^{-1})$$

when the K code is used in its ϕ-encoded form, or

$$1 - (4K/n)^{\frac{1}{2}} + O(n^{-1})$$

where $K = 2 - \log_2 3$, if the code is used in its theoretically best form.

Incidentally, these worst cases occur when we use $n = 8u^2 - 8u + 3$ i.e. when $s = 4u - 4$.
If $s = 4u - 5$, $n = 8u^2 - 10u + 3$; by section 5.9, we use $(u - 1)$ 01 pairs for our K code and obtain a highly efficient code.

Thus the coefficients of $n^{-\frac{1}{2}}$ are the same as in the formulae in section 5.11; in fact, if we analyse the coefficients of n^{-1}, we find that they differ by $\frac{1}{2}$. This is in line with our conjecture that efficiency is only less for a value of n two larger when the longer code has one more box and no more arbitrary digits; and the coefficient agrees with the difference between two arbitrary digits and the $1\frac{1}{2}$ 'arbitrary digits' in a two-digit box.

5.13 Another Adaption of the K Code Format

If the most efficient K code of a given length has an odd number of boxes, we can of course adapt the format in the same way as we did with the M code in section 4.5.3; if we have an odd box after β-encoding binary data into pairs of boxes, we can fix the first digit in such a box as a unit and leave the second position arbitrary.

There is another method which is just as efficient and perhaps helpful from the viewpoint of error correction; we can include another box, in place of two arbitrary digits, to pair with the odd box. This costs nothing in efficiency; we ϕ-encode three arbitrary digits into the pair of boxes, where before we had two arbitrary digits and the odd box. The advantage of this system is that the extra box is an aid to synchronisation; the disadvantage is that it converts three arbitrary digits into two boxes, which have to be ϕ-encoded and ϕ-decoded.

We shall now consider an example of the effects of the necessity for such format adaptions on code efficiency; we choose for our example the values of n for which it may affect our decision as to whether we use an M code or a K code; these are the values of n between 11 and 20.

5.13.1 Example

We know from section 5.11.2, that, theoretically, the K code is more efficient for $n \geqslant 17$; we shall see if the practical case, using ϕ-encoding, gives any different values.

Each type of code will be assigned three headings; the number of boxes, the number of arbitrary digits, and the total number of information bits they can carry.

n	M code			K code		
	Boxes	a.d.	Bits.	Boxes	a.d.	Bits.
11	4	1	7	2	3	6
13	5	1	8	2	5	8
15	6	1	10	3	5	9
17	7	1	11	3	7	11
19	8	1	13	4	7	13
21	9	1	14	4	9	15

We see from the above table that for n = 13 the ϕ-encoded K code, as efficient as the ϕ-encoded M code, would now be preferable, because of its longer herald; but in all the other cases, the practical choice is no different from the theoretically best choice.

5.14 Comparing the M Code and K Code

The major difference between the two is that all the digits in the M code are at least as restricted and some are more restricted than their counterparts in the K code; this compensates for the shorter herald. Now, for synchronisation purposes alone, it will probably be better to use a K code; the longer herald,

and the subsequent reduction in the probability of a false herald, makes the location of the herald easier. (Another factor to be considered is that a 01 pair beginning in an odd position in the middle of a word, allowable in a K code, is to be welcomed, since it removes the possibility of a false herald, which requires all 10 combinations at that point). But, since so many of the digits in a K code are arbitrary, and since the synchronisation mechanism only observes a small fraction of the digits in a word, we lose some of the error detection ability provided by a synchronised M code. This latter, however, is not really important with small error probability.

5.14.1 Weight Distribution of the K Code

In the next few sections we shall determine the weight distribution of the words in a K code, and then we shall compare it with the weight distribution of the M code and Gilbert's code.

In the following sections, we shall illustrate the K code first, and then the $\emptyset$-encoded version; however, the formulae we obtain will be complicated, and a simple approximation will be found in section 5.17.

5.15 THE WEIGHTS OF WORDS IN A K CODE

5.15.1 Lemma

In the s boxes of a K code, not β-encoded, we can arrange w' units in

$$\sum_{p=0}^{\left[\frac{w'}{2}\right]} \text{Comb}(s,p)\,\text{Comb}(s-p,\, w-2p) \qquad \text{ways.}$$

Proof:

Suppose we choose p 11 boxes, q 10 boxes and r 00 boxes to fill the s positions.

Then $2p + q = w'$

We can arrange the p 11 boxes among the s positions in Comb(s,p) ways; we can arrange the q 10 boxes among the remaining s - p positions in Comb(s - p, q) ways; and we then place the r 00 boxes in the remaining s - p - q positions in just one way.

For given p and q, the total number of arrangements of the boxes is

$$\text{Comb}(s,p)\ \text{Comb}(s-p,\, q)$$

For all possible choices of p and q, the total number of arrangements is

$$\sum_{p=0}^{\left[\frac{w'}{2}\right]} \text{Comb}(s,p)\ \text{Comb}\ (s-p,\, w'-2p)$$

5.15.2 Lemma

In the s boxes of a ϕ-encoded K code, where s is even, we can arrange w' units in

$$\sum_{p=0}^{\left[\frac{w'}{2}\right]} 2^{w'-2p} \text{ Comb}(\tfrac{1}{2}s,\ w' - 2p) \text{ Comb}(s - w' + p,\ p) \qquad \text{ways}$$

Proof:

We note in passing that this result can be applied to a ϕ-encoded K code with an odd number of boxes which has been format-adapted, if minor alterations are made to n and w.

The s boxes are divided into $\frac{1}{2}s$ pairs of boxes for the purpose of ϕ-encoding, $q \leqslant \frac{1}{2}s$, where p,q and r are respectively the number of 11, 10 and 00 boxes among the s boxes, as before.

The total number of ways of allocating the 10 boxes among the $\frac{1}{2}s$ pairs of boxes, one 10 box to a pair, is

$$\text{Comb}(\tfrac{1}{2}s,\ q).$$

But, since the 10 boxes can go in two possible positions in each pair, then the total number of ways of allocating the 10 boxes is

$$2^q \text{ Comb}(\tfrac{1}{2}s, q).$$

To each way of allocating the q 01 boxes, there is Comb(s - q, p) ways of allocating the 11 boxes.

Hence, for given p and q, there are altogether

$$2^q \text{ Comb}(\tfrac{1}{2}s, q) \text{ Comb}(s - q, p)$$

ways of allocating the boxes. But $w' = 2p + q$. Therefore, the number of ways of allocating the w' units among the s boxes is

$$\sum_{p=0}^{\left[\frac{w'}{2}\right]} 2^{w'-2p} \text{ Comb}(\tfrac{1}{2}s, w' - 2p) \text{ Comb } (s - w' + p, p)$$

We are now in a position to develop the formulae for the number of words of weight w in both the K code and the ϕ-encoded K code of length n. However, it will be appreciated that the formulae are too complicated for practical use; a reasonable approximation will be derived later.

5.15.3 The number of words of weight w in the K code and the ϕ-encoded Kcode of length n are

$$\sum_{w'=0}^{w-u} \text{Comb}(n - 2u - 2s, w - u - w') \sum_{p=0}^{\left[\frac{w'}{2}\right]} \text{Comb}(s,p) \text{ Comb}(s-p, w' - 2p)$$

and

$$\sum_{w'=0}^{w-u} \text{Comb}(n - 2u - 2s, w - u - w') \text{ times}$$

$$\sum_{p=0}^{\left[\frac{w'}{2}\right]} 2^{w'-2p} \text{ Comb}(\tfrac{1}{2}s, w' - 2p) \text{ Comb}(s-w' + p,p)$$

respectively.

Proof:

The result follows from the two preceding lemmas. For each value of w and each value of w', we have $w - u - w'$ units to distribute among the $n - 2u - 2s$ arbitrary positions; this can be done in $\mathrm{Comb}(n - 2u - 2s, w - u - w')$ ways.

Hence the total number of arrangements of the digits, i.e. the total number of words, is as above.

5.16 The Average Distance Between Words in a K Code

We mentioned in section 3.15 that a large average distance between code words is desirable since overall it decreases the probability that a word which collects errors during transmission will be mistaken for another word by the decoder.

We also showed at that time that the average distance between words in an M code was $4n/9 - 1$; between words in a Gilbert code it was about $\frac{1}{2}n - n^{\frac{1}{2}}$.

Since there are generally more words in a K code of length n than in either the M code or Gilbert's code of that length, we should not be disappointed if the average distance between words in a K code does not approach the figure $\frac{1}{2}n$ which is the average distance between all the words in $W^{(n)}$. It does in fact approach this figure fairly closely; this proximity allows us to make the approximations of the formulae in section 5.15.3; and so we do derive some practical benefit from the result.

5.16.1 Lemma

In a K code of length n with u 01 pairs for a herald, the average distance between words is approximately

$$\tfrac{1}{2}n - n^{\frac{1}{2}}$$

Proof:

Consider all the words of the K code arranged in a column.

Suppose the K code has s boxes, and c = 0

The code has $(2u - 2)s + 1$ arbitrary digits and $2s$ digits in boxes.

In the columns of our word column which correspond to arbitrary digits, there are $\frac{1}{2}N$ zeroes and $\frac{1}{2}N$ units, where N designates the number of words in the K code.

In the columns, which represent boxes, there are $\frac{2}{3}N$ of one kind and $\frac{1}{3}N$ of the other kind. (This latter is the same result as for the M code, see section 3.14).

Thus the total number of differences between words, spread over $\frac{1}{2}N(N - 1)$ pairs of words, is

$$((2u - 2)s + 1)N^2/4 + 2s.\ 2N^2/9$$

$$= N^2((2us + 2u + 1)/4 - s/18 - u/2)$$

$$\doteq N^2(n/4 - 3u/4),$$

assuming $s \doteq 4u$.

Therefore, since N soon becomes very large, the average distance between words is

$$\frac{N^2(n - 3u)}{2N(N - 1)} \simeq \tfrac{1}{2}(n - 3u)$$

$$\simeq \tfrac{1}{2}n - n^{\frac{1}{2}}, \text{ if } n \simeq 8u^2$$

This is the same result as we derived for Gilbert's code of length n; so although the K code of length n has more words than Gilbert's code of that length, the average distance between words is no less.

If we apply the same argument to the column of all words in the $\emptyset$-encoded K code of length n, we find that the average distance is even greater (i.e. approximately $\tfrac{1}{2}n - 0.7n^{\frac{1}{2}}$).

5.17 Approximation to Formulae in 5.15.3

We can now say that the distribution of weights of words in the K code of length n is somewhat similar to the weight distribution of the words in $W^{(n)}$. So we shall make our approximation, for the number of words of weight w in a K code of length n, a number proportional to the number of words of weight w in $W^{(n)}$.

Thus we say that the number of words of weight w in a K code of length n is about

$$2^{n-2u-2s} \cdot 3^s \cdot \mathrm{Comb}(n,w)/2^n$$

where (i) $2^{n-2u-2s} \cdot 3^s$ is the number of words in the K code of length n

(ii) $\mathrm{Comb}(n,w)$ is the number of words of weight w in $W^{(n)}$

(iii) 2^n is the total number of words in $W^{(n)}$

i.e. the number of words is approximately

$$3^s \, \mathrm{Comb}(n,w)/2^{2u+2s}$$

By the same method, the number of words in the ϕ-encoded K code of length n is about

$$2^{n-2u-\frac{1}{2}s} \, \mathrm{Comb}(n,w)/2^n$$

$$= \mathrm{Comb}(n,w)/2^{2u+\frac{1}{2}s}$$

We will not attempt to justify this approximation; we can only say that it has been tested in the two cases given in the next section and found to give good results for values of w not too close to either 0 or n, although, understandably, the approximations near both 0 and n are not good; in these areas almost every cycle is represented in the K code, and this is not the result we get with our formula.

We can state our approximation in another way; since there are $2^{n-2u-2s} \cdot 3^s$ words in a K code of length n, and about $2^n/n$ cycles of length n, the ratio of the number of words in the K code of weight w to the number of words of weight w in $W^{(n)}$ should be approximately constant at

$$n/(2^{2u+2s} \cdot 3^{-s})$$

for the K code

and at $$n/2^{2u+\frac{1}{2}s}$$

for the ϕ-encoded K code.

Again, this approximation should only be used for w not too close to 0 or n.

We will now finish this section with some illustrative graphs which show clearly the weight distributions and efficiencies of the various types of codes for certain values of n.

5.18 Notes on the Graphs

5.18.1 We have shown on the first graph the efficiencies attainable with K codes which use four different values of u, not because the codes with lower efficiencies are practical, but to show the 'leapfrogging' effect as a certain value of u gives the most efficient K code for only a few values of n.

5.18.2 It is apparent from the difference between the M code and the ϕ-encoded M code graphs that the necessity of adapting to binary input greatly affects the efficiency of the M code as well as requiring a more complicated encoder. However, we notice that the ϕ-encoded M code is still more efficient than the fixed-place code.

5.18.3 It is a pity that we have to use a log scale, since it makes the crossovers on the n = 51 graph appear more important than they are. When we remember that the number of words of weight up to 12 is only a very small fraction of the total number of words, we see the importance of these crossovers in a truer perspective.

5.18.4 Apart from this, notice the way in which the K code gains ground on its rivals around the most important point - the high point of the graphs. For example, in the n = 21 graph it has only about 2/5 as many words of weight 4 as the M code, but about 15/11 of weight 10. In the n = 51 graph, it improves from about equality with the M code for w = 12 up to about 7/2 as many words for w = 25.

5.18.5 Always the number of cycles remains an unapproachable upper limit. This is also most apparent towards the centre of the graph, where the lack of the balanced 01 combinations tells against both the K code and the M code in achieving words of weight about half the length. This indicates the conclusion that efficiency cannot be much improved using the 'herald and boxes' structure.

5.18.6 Notice that for moderately large value of w (e.g. over the range $n/4 < w < 3n/4$ the graph of the K code runs approximately parallel to that of the number of cycles. This helps us to check our approximation.

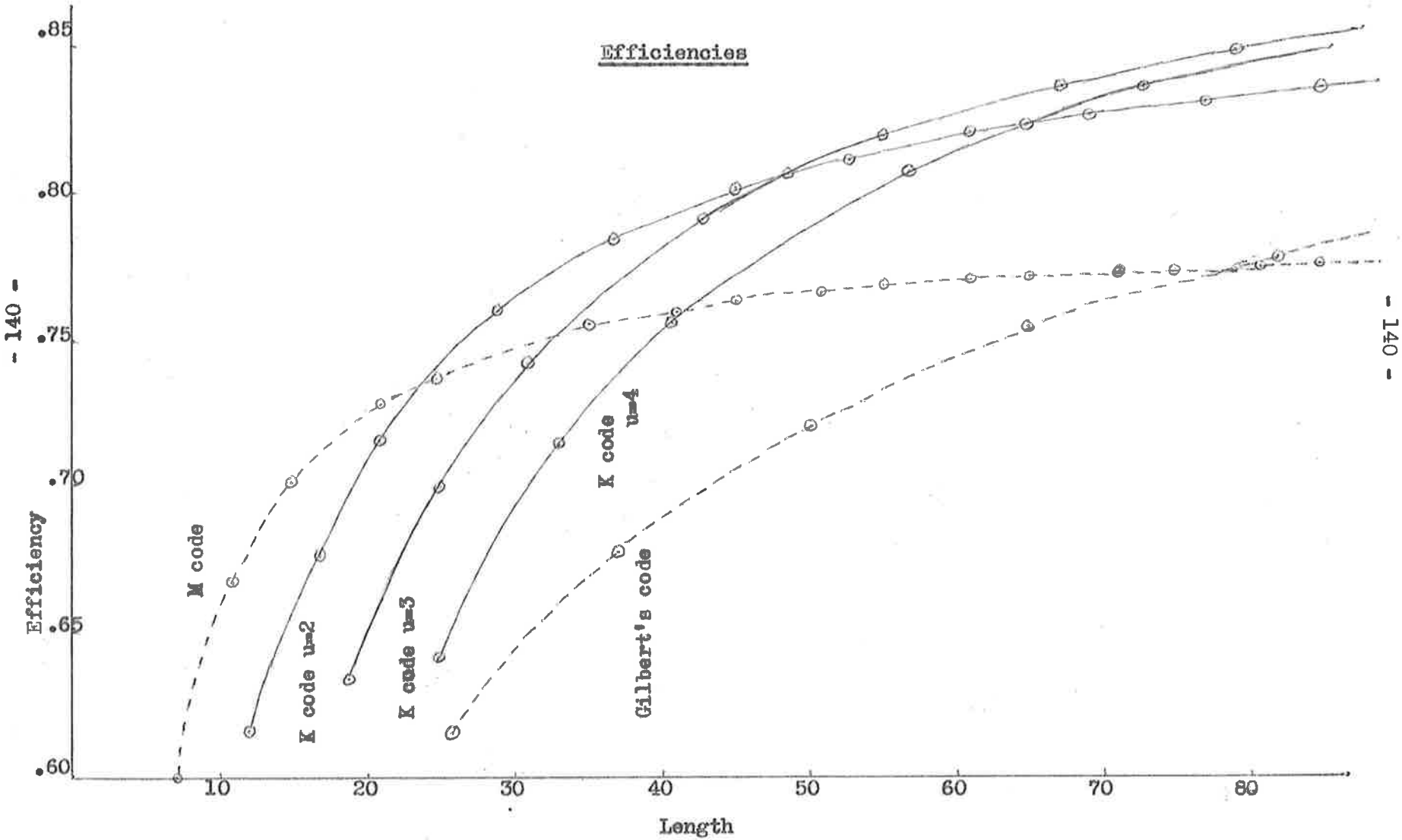
e.g. for n = 21, when $u = \frac{1}{2}s = 2$, $n/2^{2u+\frac{1}{2}s} \simeq \frac{1}{3}$

It can be seen from the graphs that the difference between the graphs is about $\log_2 3$, or about 1.6.

For n = 51, when u = 3, s = 8, the fraction is about 1/20, and the gap between the graphs is about $\log_{10} 20 = 1.3$.

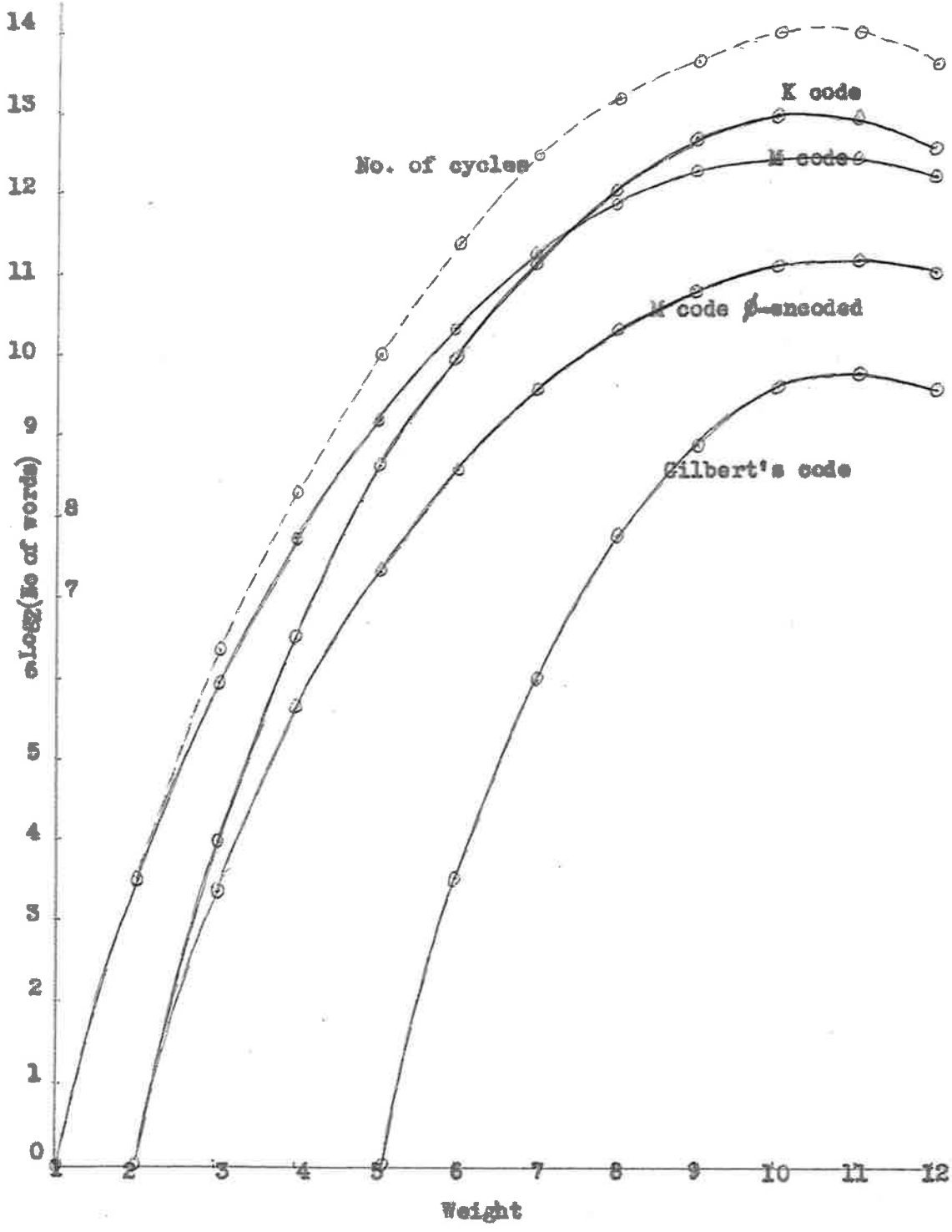
5.18.7 On the graph of efficiencies, the characteristic "leapfrogging" of the various K codes for different u is shown. We can also clearly see that the efficiency of the fixed place code lags far behind that of the most suitable K code, although it catches the M code at n = 79.

5.18.8 The last graph gives a greater magnification of the crossover between two representative K codes using different u codes, and shows the way in which actual efficiency differs from the the theoretical values obtained previously, if we use β-encoding on the boxes.

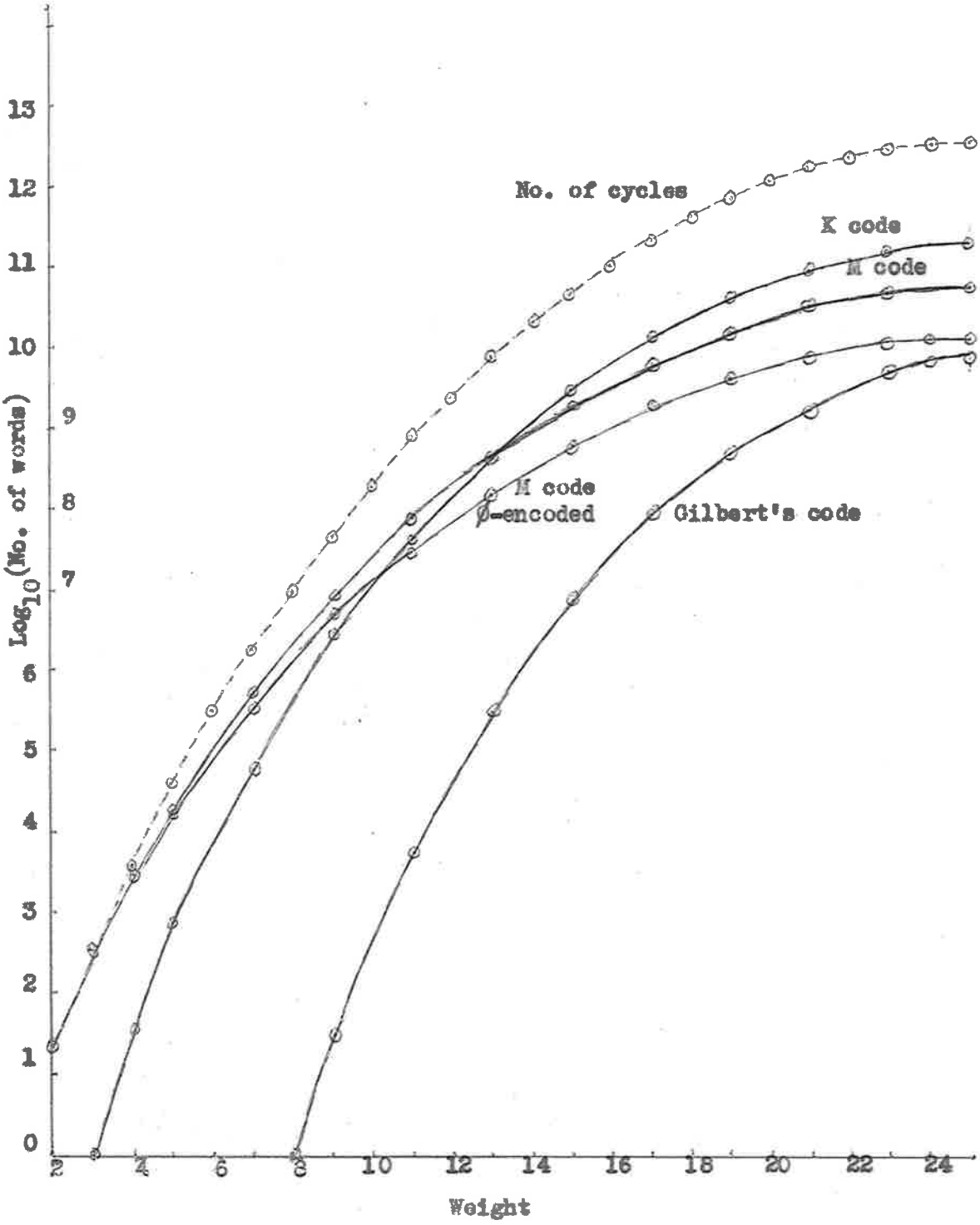

Efficiencies
.85
.80
.75
.70
.65
.60
Efficiency
M code
K code u=2
K code u=3
K code u=4
Gilbert's code
10
20
30
40
50
60
70
80
Length

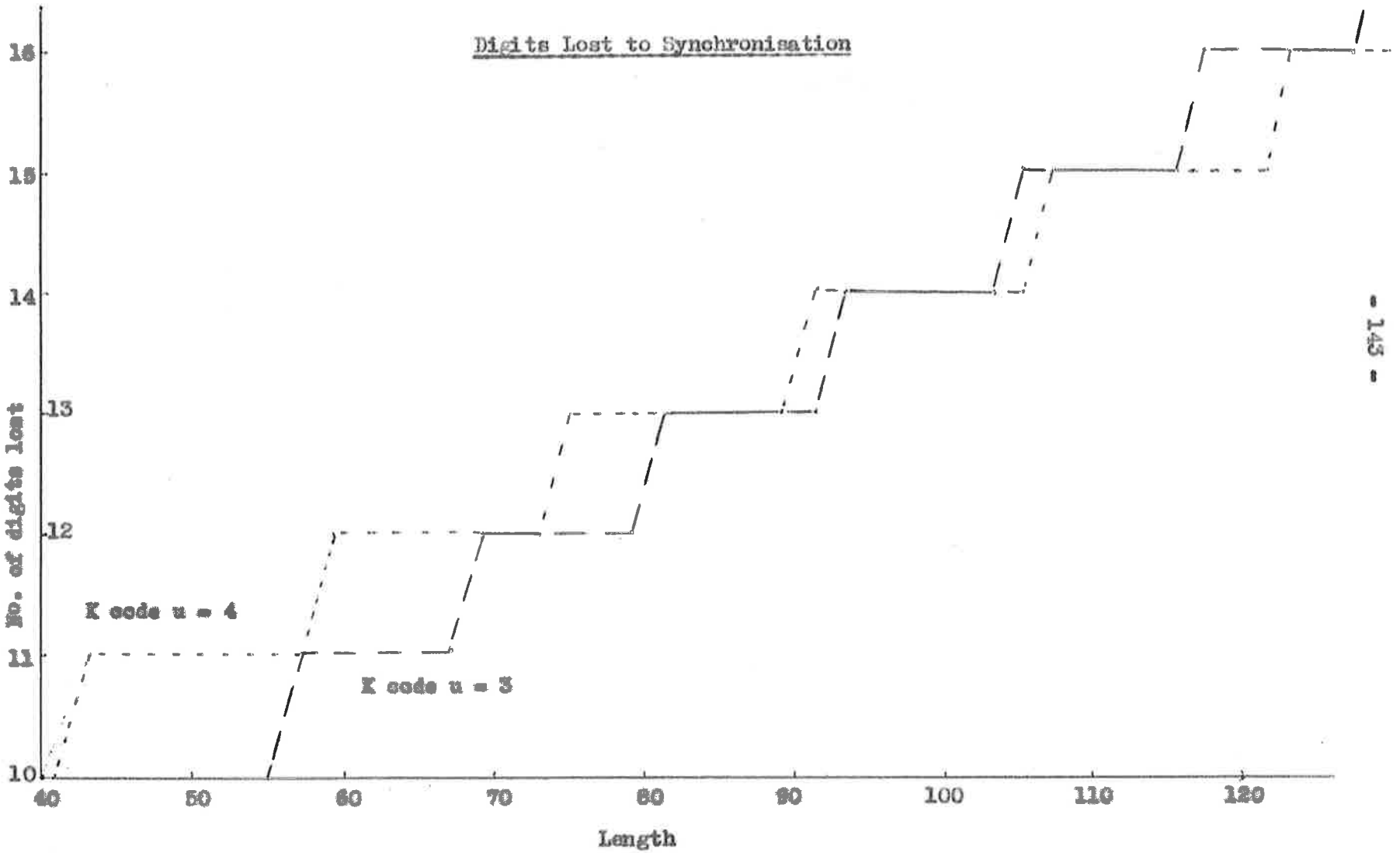
- 140 -

The Various Codes n = 21



The Various Codes n = 51




Digits Lost to Synchronisation
16
15
14
13
12
11
10
No. of digits lost
K code u = 4
K code u = 3
40
50
60
70
80
90
100
110
120
Length

6. CODES OF EVEN LENGTH

6.1. Introduction

So far, the codes we have considered (the M codes and the K codes) have been codes of odd length. With both these codes, we ensured that any false 'heralds' would begin in an even-numbered position; and, because of the odd length, even numbered positions in the next real word, which the false word overlapped, became even-numbered positions in the false word, and odd-numbered positions in the next real word became odd-numbered positions, in the false word. In particular, the herald of the next real word began at an odd-numbered position in the false word, and this ensured that a 01 pair would be found where the false word, to be identical with a code word, needed a box.

In our examination of the synchronous property of codes in this section, the only false words which will concern us are those which are identical with code words. So we say that a false word requires a unit in a certain position if and only if every code word has a unit in that position; only if this requirement is fulfilled can the false word possibly be identical with a code word.

With codes of even length, we could still ensure that all false words begin in even-numbered positions by beginning all our boxes in odd-numbered positions. However, in this case the herald of the next real word would begin at an even-numbered position in

the false word, and would give a 10 where the false word requires a box - and this would not prevent the false word from being identical with a code word. So the boxes must be positioned differently; some must begin in even-numbered positions and some in odd-numbered positions. One technique which is used later begins each of the boxes before the middle of the information-carrying part of the word, (the whole of the word except for the herald) in an odd-numbered position, and each of the boxes after this point in an even-numbered position. It will be shown later (in section 6.5) that it is possible to construct synchronous codes by applying this idea.

For the rest of this section, we will compare the codes constructed here only with codes which use both boxes and fixed places to achieve synchronisation. However, it will be shown in section 8 that there is strong evidence for assuming that synchronous codes constructed using the 'herald and boxes' technique are at least as efficient as synchronous codes which do not use this technique. And so, when we say that a code constructed in this section is the most efficient 'herald and boxes' code of a certain length which can be constructed, it is definitely possible that no other synchronous code of that length (except a generalised 'herald and boxes' code (see section 8)) could be more efficient.

A 'herald and boxes' code is a synchronous code which has a herald composed of 01 pairs, and boxes imbedded in the remainder of the code word.

We should stress that the codes constructed here are not as efficient as codes of the same length constructed by the Golomb method. However, Golomb codes are not synchronous codes, and in section 8 evidence is adduced that synchronous codes cannot have the efficiency given by Golomb's upper bound.

In this section, when we refer to 'the E code' we will mean 'the E code of a given length'.

6.2 Definition

We shall define an E code as a code of the 'herald and boxes' type which consists of all the possible words which have the format

$$01\,01\,\ldots\,01 b_1 b_2 xx \ldots xb_3 b_4 xx \ldots xb_5 b_6 \ldots b_{2s-1} b_{2s} xx \ldots x$$

where

(i) The herald consists of u 01 pairs.

(ii) $b_{2i-1} b_{2i}$ represents the ith box, $i = 1, \ldots, s$; each of these boxes is either 00, 10 or 11.

(iii) p_i consecutive arbitrary digits follow the ith box.

(iv) As in the K code, $p_i \leqslant 2u - 2$, $i = 1, \ldots, s-1$, $p_s \leqslant 2u - 1$

Now $n = 2u + 2s + \sum_{i=1}^{s} p_i$, and n is even.

Therefore, it follows that, in an E code, $\sum_{i=1}^{s} p_i$ must be even.

We shall show that by suitable positioning of the boxes and, in some cases, reduction of the number of digits, we can find synchronous E codes.

If we allow any p_i, i = 1, ... , s - 1, to be larger than 2u - 2, it is possible for false words to start at both even and odd-numbered positions among these p_i digits; the restriction that $p_i \leqslant 2u - 2$ means that false words whose first 2u digits are identical with the code herald may begin in either even-numbered or odd-numbered positions among the p_i, but not both.

(i) e.g. u = 3, $p_i = 2u = 6$

... $b_{2i-1}b_{2i}$ x x x x x x $b_{2i+1}b_{2i+2}$ ... 0 1 0 1 0 1

Possible false words	. . 0 1 0 1 0 1 ...	. . $b_j b_{j+1}$ ...
	. 0 1 0 1 0 1 ...	. $b_j b_{j+1}$ ...
	0 1 0 1 0 1 ...	$b_j b_{j+1}$ ...

We see that false words can begin at 3 consecutive positions.

Suppose that $b_j b_{j+1}$ represents positions where a false word, to be identical with a code word, would need a box. In two of the cases the next code herald will ensure that 01 go in these positions, and so prevent the false word from looking like a code word; however in the middle case the next code herald would put 10 in $b_j b_{j+1}$ and, since this is permissible in a box, the false word could be identical with a code word. In these circumstances we would have to have another box coincident with the next real herald at the same time,

e.g. 0 1 0 10 1

$$b_j b_{j+1} x b_k b_{k+1}$$

But this means that $p_j \leqslant 2u - 5$. (a)

In fact using this method we can show that if $p_i > 2u - 2$, $i < s$, the there must be a p_j such that $p_i + p_j < 4u - 4$. Since if p_i and p_j are both equal to $2u - 2$, $p_i + p_j = 4u - 4$, there is nothing to be gained by allowing any of the p_i's to exceed $2u - 2$. The case for p_s is slightly different, for here the next digit after the end of the string of p_s arbitrary digits is not just the first position in a box, which may be 0 or 1, as it is for the other p_i's, but a fixed 0, the first digit of the next real word.

Now a false word identical with a code word cannot begin at the first position of the last box. It cannot begin at the second position of the last box only if the first 'box' in the false word overlaps the first 01 pair of the herald of the next code word; this is the case if $p_s \leqslant 2u - 1$.

Any false word beginning in the last arbitrary string will not be identical with a code word, since the code herald supplies either a unit where the false 'herald' needs a 0, or a 01 pair in the first 'box' of the false word.

A code is synchronous if and only if none of the set of false words which may occur is identical with any code word. The next theorem defines the positions where false words could begin in a string of synchronous code words, by virtue of the fact that it is possible for a false herald sequence to begin there. The theorems etc. which follow this theorem will show us that, if we impose certain restrictions on the E code, it is possible to ensure that no false words beginning in these positions can be identical with any code word.

6.3 Theorem

Let W, X be words in an E code with u 01 pairs for a herald and s boxes. Suppose that the sequence of n consecutive digits of WX beginning at the pth position of W ($2 \leqslant p \leqslant n$) is a word Z of the E code.

Then $s \geqslant 2$ and

$$p = 2u + \sum_{u=1}^{j} (p_i + 2) - 2k,$$

where $1 \leqslant j \leqslant s - 1$ and k is an integer, $0 \leqslant k \leqslant \frac{1}{2}p_j$

Proof:

Let $W = 0101 \ldots 01\ w_1 w_2 w \ldots w w_3 w_4 \ldots w_{2s-1}\ w_{2s} w \ldots w$

$X = 0101 \ldots 01\ x_1 x_2 x \ldots x x_3 x_4 \ldots x_{2s-1}\ x_{2s} x \ldots x$

$Z = 0101 \ldots 01\ z_1 z_2 z \ldots z z_3 z_4 \ldots z_{2s-1}\ z_{2s} z \ldots z$

Clearly, the herald of Z cannot overlap the herald of W, because this would imply that $w_1 w_2$ are 01 respectively. Similarly Z cannot begin at w_1.

Thus

$$p \geqslant 2u + 2 \qquad (1)$$

Again, the herald of Z cannot overlap the herald of X, since this would imply that $z_1 z_2$ are 01 respectively. Similarly, the herald of Z cannot end at the last digit of W.

Thus

$$p \leqslant n - 2u \qquad (2)$$

If $s = 1$, we have

$$n = 2u + p_1 + 2, \text{ and } p_1 \text{ is even.}$$

But by (1) and (2)

$$n - 2u \geqslant 2u + 2$$

i.e. $$p_1 \geqslant 2u$$

This contradicts our assumption that $p_1 \leqslant 2u - 1$. (We can show that this inequality must hold, as there are no other p_j's from which we can choose one to satisfy the inequality (a),

$$p_i + p_j < 4u - 4,$$

given in section 6.2.).

Thus $s \geqslant 2$.

Let j be a positive integer such that

$$2u + \left(\sum_{i=1}^{j-1} (p_i + 2)\right) + 1 < p \leqslant 2u + \left(\sum_{i=1}^{j} (p_i + 2)\right) + 1 \qquad (3)$$

Since $2u + (\sum_{i=1}^{s-1} (p_i + 2)) + 1 = n - (p_s + 2) + 1$

and $p_s \leqslant 2u - 1$,

$$2u + (\sum_{i=1}^{s-1} p_i + 2)) + 1 \geqslant n - 2u$$

$$\geqslant p_s \text{ by (2).}$$

From this it follows that $j \leqslant s - 1$.

Since $p_j \leqslant 2u - 2$, the herald of Z must completely overlap $w_{2j+1}w_{2j+2}$ (which must be 10) or else the last digit of the herald Z must correspond to w_{2j+1} ; this latter case can only occur when $p_j = 2u - 2$ and $p = w_{2j}$. Since w_{2j+1} occupies the position

$$2u + (\sum_{i=1}^{j} (p_i + 2)) + 1$$

of W and $w_{2j+1}w_{2j+2}$ cannot be 01, then

$$2u + (\sum_{i=1}^{j} (p_i + 2)) + 1 - p \text{ must be odd}$$

Put

$$2k = 2u + (\sum_{i=1}^{j} (p_i + 2)) - p \qquad (4)$$

By (3), $k \geqslant 0$ and $k \leqslant \frac{1}{2}p_j$

From (4), $p = 2u + \left(\sum_{i=1}^{j} (p_i + 2)\right) - 2k$

QED.

In this theorem, we proved that, if s = 1, an E code is always synchronous. However, this is not a very important case, since, for $u > 2$, the fact that there are more fixed places than boxed positions means that the E code so constructed will not be very efficient; a more efficient code could be constructed with u less and s greater. Since an E code so closely resembles a K code, we would suppose that the maximum efficiency of the E code would occur when the ratio of 01 pairs to boxes is about the same as for the most efficient K code; and this ratio is about one 01 pair to four boxes (see section 5.9).

We shall now consider the case where $s \geqslant 2$. The next theorem describes a necessary and sufficient condition that an E code with $s \geqslant 2$ must satisfy to be synchronous.

6.4 Theorem

An E code of a given length with two or more boxes is synchronous if and only if for every pair of integers j, k such that

$$1 \leqslant j \leqslant s - 1 \tag{1}$$

$$0 \leqslant k \leqslant \tfrac{1}{2}p_j \tag{2}$$

and

$$\sum_{i=j+1}^{s} (p_i + 2) + 2k \geqslant 2u - 1 \tag{3}$$

there exist integers m, q such that either

(a) $$0 \leqslant m \leqslant s - 1 \tag{4}$$

$$0 \leqslant q \leqslant u - 1 \tag{5}$$

and

$$2u + \sum_{i=1}^{m} (p_i + 2) - \left(\sum_{i=j+1}^{s} (p_i + 2) + 2k \right) = 2q + 1 \tag{6}$$

or

(b) $$j \leqslant m \leqslant s - 1 \tag{7}$$

$$1 \leqslant q \leqslant u - 1 \tag{8}$$

and

$$2k + \sum_{i=j+1}^{m} (p_i + 2) = 2q - 1 \tag{9}$$

Proof:

Suppose that W and X are words in an E code, where

$$W = 0101 \ldots 01w_1w_2ww \ldots ww_3w_4 \ldots w_{2s-1}w_{2s}w \ldots w$$

$$X = 0101 \ldots 01x_1x_2xx \ldots xx_3x_4 \ldots x_{2s-1}x_{2s}x \ldots x$$

Suppose also that the sequence of n consecutive digits of WX beginning at the pth position of W, $2 \leqslant p \leqslant n$, is a word Z in the E code, where

$$Z = 0101 \ldots 01z_1z_2zz \ldots zz_3z_4 \ldots z_{2s-1}z_{2s}z \ldots z$$

By the last theorem, there exist integers j and k such that (1) and (2) are satisfied and

$$p = 2u + \sum_{i=1}^{j} (p_i + 2) - 2k \qquad (10)$$

The herald of Z cannot overlap the herald of X, because this would imply that z_1z_2 are 01 respectively; hence, since the herald of Z ends in position $p + 2u - 1$ of W, we have

$$p + 2u - 1 \leqslant n$$

i.e. $$4u + \sum_{i=1}^{j} (p_i + 2) - 2k - 1 \leqslant 2u + \sum_{i=1}^{s} (p_i + 2)$$

i.e. $$\sum_{i=j+1}^{s} (p_i + 2) + 2k \geqslant 2u - 1$$

If: Now for j, k satisfying (1), (2) and (3), suppose that there exist integers m, q such that (a) (4), (5) and (6), or (b) (7), (8) and (9), are satisfied.

(b) Now suppose (7), (8) and (9) are satisfied

w_{2u+1} occurs in position

$$2u + \sum_{i=1}^{m} (p_i+2) + 1$$

of the word W, hence in position

$$2u + \sum_{i=1}^{m} (p_i+2) + 2 - p$$

of the word Z

i.e. the position

$$\sum_{i=j+1}^{m} (p_i + 2) + 2k + 2 = 2q + 1$$

of Z.

Now the herald of Z goes as far as the 2uth position of Z; and so the $(2q + 1)$th position of Z should be 0 and the $(2q+2)$th position of Z should be 1. But $w_{2u + 1}$ and w_{2u+2}, which fall in these positions, are not 01; This contradicts our assumption that Z is identical with a code word.

(a) Now suppose (4), (5) and (6) are satisfied, and consider

z_{2m+1} and z_{2m+2}.

z_{2m+1} occurs in the position

$$r = 2u + \sum_{i=1}^{m}(p_i + 2) + 1$$

of Z, hence

in position $t = p + r - 1 - n$

of X.

Then

$$t = 2u + \sum_{i=1}^{j}(p_i + 2) - 2k + 2u + \sum_{i=1}^{m}(p_i + 2) + 1 - 1 - (2u + \sum_{i=1}^{s}(p_i + 2))$$

$$= 2u + \sum_{i=1}^{m}(p_i + 2) - (\sum_{i=j+1}^{s}(p_i + 2)) - 2k$$

$$= 2q + 1$$

Now, since $0 \leqslant q \leqslant u - 1$, the $(2q + 1)$th position of X is an odd-numbered position in the herald of X. Hence z_{2m+1} and z_{2m+2} contain 0 and 1 respectively. Again, this contradicts our assumption that Z is identical with a code word.

Only if:

Suppose now that the E code is synchronous and that j and k are a pair of integers satisfying (1), (2) and (3). Put

$$p = 2u + \sum_{i=1}^{j} (p_i + 2) - 2k$$

We shall now show that under these conditions, if there is no pair of integers m, q satisfying (7), (8) and (9), then there exists a pair of integers m, q satisfying (4), (5) and (6).

Consider the word W

W = 0101 ... 010000 ... 00101 ... 0100 ... 0

2u digits 2u digits

Suppose that the second string of u 01 pairs begins in the pth position of W. For each i, i = 1, ... , s, let $w_{2i-1}w_{2i}$ indicate the pair of digits in positions

$$e_i = 2u + \sum_{u=1}^{i-1} (p_u + 2) + 1$$

and $e_i + 1$, of W

We shall now prove that W is a word in the E code. It is sufficient to show that for each i, $w_{2i-1}w_{2i}$ are not 01 respectively.

When $i \leqslant j$, we have

$$p - e_i = \sum_{u=i}^{j} (p_u + 2) - 2k - 1$$

$$\geqslant \sum_{u=i}^{j} (p_u + 2) - p_j - 1$$

$$= \sum_{u=i}^{j-1} (p_u + 2) \quad \geqslant \quad 1$$

Hence w_{2i-1} and w_{2i} are both 0.

When $i \geqslant j + 1$, we have

$$e_i - p = \sum_{u=j+1}^{i-1} (p_u + 2) + 2k + 1$$

Therefore either

(1) $e_i - p > 2u - 2$, in which case $w_{2i-1} = 1$ and $w_{2i} = 0$

(if $e_i - p = 2u - 1$) or $w_{2i-1} = 0$ and $w_{2i} = 0$

or

(2) $2k + 1 \leqslant e_i - p \leqslant 2u - 2$

If we put $i - 1 = m$, we see that

$$j \leqslant m \leqslant s - 1$$

and

$$2k \leqslant \sum_{u=j+1}^{m} (p_u + 2) + 2k \leqslant 2u - 3$$

Now, by our assumption that (§) is not satisfied

$$\sum_{u=j+1}^{n} (p_u + 2) + 2k \text{ is even}$$

Therefore $e_i - p$ is odd. Thus $w_{2j-1} = 1$ and

$$w_{2j} = 0$$

This proves that W is a word in the E code.

Also, clearly, if

$$X = 0101 \ldots 01111 \ldots 1$$

$$2u \text{ digits}$$

then X is a word in the E code.

The sequence of n consecutive digits of WX beginning at the pth digit of W is represented by

$$Z = 0101 \ldots 01\, z_1 z_2 z \ldots z z_3 z_4 \ldots z_{2s-1} z_{2s} z \ldots z$$

Now, when z_{2i-1} is in W

$z_{2i-1} z_{2i} = 00$, and when z_{2i} lies to the right of the herald of X,

$$z_{2i-1} z_{2i} = 11.$$

Now, since the Z code has been assumed synchronous, Z is not identical with a code word. This being the case, there must exist an integer m, $0 \leqslant m \leqslant s - 1$, such that

(i) $z_{2m+1}z_{2m+2}$ is entirely contained within the herald of X and

(ii) $z_{2m+1} = 0$ and $z_{2m+2} = 1$.

i.e. there exists an integer q such that z_{2m+1} lies in the $(2q + 1)$th position of X, $0 \leqslant q \leqslant u - 1$.

Then

$$p + 2u + \sum_{i=1}^{m} (p_i + 2) = n = 2q + 1$$

$$\text{i.e. } 2u + \sum_{i=1}^{j} (p_i + 2) = 2k + 2u + \sum_{i=1}^{m} (p_i + 2)$$

$$= (2u + \sum_{i=1}^{s} (p_i + 2)) = 2q + 1$$

$$\text{i.e. } 2u + \sum_{i=1}^{m} (p_i + 2) - (\sum_{i=j+1}^{s} (p_i + 2) + 2k) = 2q + 1$$

This is equation (6), and, since our choices of m and q satisfied equations (4) and (5) respectively, this completes the proof.

The author is indebted to Dr. J.H. Michael for this proof of the theorem.

The reason that we made j and k satisfy equation (3) was illustrated in the proof; if this inequality does not hold for any pair of integers j, k, it is impossible for a false word identical with a code word to begin at that position p defined by j and k, since it would overlap the next real herald (that of X). If, for given j, k satisfying (1) (2) and (3), there exist integers m, q satisfying (4), (5) and (6), the false word cannot begin in that position p since the positions where such a false word to be identical with a code word, would require a box, it will have a 01 pair from the next real herald. And if there are m and q satisfying (7), (8) and (9) then a rearrangement of (9) gives

$$2u + \sum_{i=1}^{j} (p_i + 2) - 2k + 2q = 2u + \sum_{i=1}^{m} (p_i + 2) + 1$$

$$\text{i.e.} \quad p + 2q = w_{2m-1}$$

Thus a false word cannot start in this position since where it would require a 01 pair in its herald, it will find a box of W. This is not the same case of interference with a false herald as was exhibited in theorem 6.3. That result indicated positions where a false word could not begin because of the restriction imposed by the box in W which was the first after the pth position of W, at which the false word started. This latter

result contains this restriction (which was helpful in determining our choice of p), but also generalises it to include any box of W which comes within the string of 2u digits of W beginning at p.

Unfortunately, this extra restriction is not often useful. In an E code with high efficiency, most of the values of p_i are close to 2u - 2. If $p_i = 2u - 2$, a string of 2u digits can only include either $w_{2i-1}w_{2i}$ or $w_{2i+1}w_{2i+2}$ — never both. And so the box of W which is the first after p is, in the majority of cases, the only one which affects positions at which a false word may begin.

We will now apply the preceding results to particular cases. The definition of a particular E code of a certain length with an odd number of boxes has two points of interest. Firstly, it is remarkable in its similarity to the K code whose length is greater by 1; in fact, it is precisely that K code with one arbitrary digit removed from the middle string of arbitrary digits. This explains the similarity between the results derived concerning the efficiency of the E code with s odd, and those results derived for the K code in section 5. Secondly, it illustrates the use of the idea given in section 6.1 of beginning all the boxes before the midpoint of the string of information-carrying digits in odd-numbered positions, and all the boxes after this point in even-numbered positions.

We shall see that the implementation of this idea in the E code with s odd gives us an even-length code of high efficiency.

6.5. Definition

An E_o code is an E code such that

(i) s is odd and $\geqslant 3$,

(ii) $p_s = 2u - 1$

(iii) $p_{\frac{1}{2}(s+1)} = 2u - 3$

(iv) $p_i = 2u - 2$, $i \neq s$, $i \neq \frac{1}{2}(s + 1)$

We will show in the next few sections that an E_o code of a given length is synchronous, and that it is at least as efficient as any other synchronous code of the same length which uses the 'herald and boxes' principle. Obviously, if we have fixed u and s, we cannot expect to nominate a length n at random and suppose that by locating the s boxes correctly we can obtain a synchronous code, no matter how large n may be. For example, we have already seen that it is advisable to restrict the p_i's to $2u - 2$, $i < s$, and $2u - 1$, $i = s$, and that values of p_i's greater than these do not appear to enhance the efficiency of the code. And so we can say that for given u and s, there is a certain length n_1 such that if a code of length n is constructed with u 01 pairs and s boxes, and $n > n_1$, then the code will not be synchronous. We will show later,

in section 7, that this bound is almost the same as for the K code, and that the E_0 code attains this bound. To this end, we should note at this point that the length of the E_0 code with u 01 pairs for a herald and s boxes is

$$2us + 2u.$$

The reason that we have particularly chosen s odd in this case is that the mid-point of the string of information-carrying digits is then about in the middle of a string of digits between two boxes. The importance of this will become apparent later when we consider the case of s even.

6.6. Theorem

An E_0 code of a given length is synchronous.

Proof:

In this proof we make repeated use of the result of theorem 4.

Consider arbitrary integers j, k such that

$$1 \leqslant j \leqslant s - 1 \qquad (1)$$

and $$0 \leqslant k \leqslant \tfrac{1}{2}p_j \qquad (2)$$

Define $$m = s - j \qquad (3)$$

and

$$h = 2u + \sum_{i=1}^{m} (p_i + 2) - \Big(\sum_{i=j+1}^{s} (p_i + 2) + 2k \Big). \qquad (4)$$

(a) If $j < \frac{1}{2}(s+1)$, then $m \geq \frac{1}{2}(s+1)$

and $h = 2u + m(2u) - 1 - (s-j)(2u) - 2k$

i.e.

$$h = 2u - 2k - 1 \qquad (5)$$

(b) If $j \geq \frac{1}{2}(s+1)$, then

$$m < \frac{1}{2}(s+1)$$

and $h = 2u + m(2u) - ((s-j)(2u) + 1 + 2k)$

i.e.

$$h = 2u - 2k - 1$$

Now, if we choose $q = u - k - 1$

then

(i) $h = 2q + 1$

and (ii) $q \leq u - 1$

also since $k \leq \frac{1}{2}p_j \leq u - 1$

$$q \geq 1$$

and, by our definition of m,

$$1 \leq m \leq s - 1.$$

Thus there exist m and q such that the first alternative of theorem 4 is satisfied.

We see that m and q can be found for all possible pairs j, k. Theorem 4 then asserts that the E_o code is synchronous.

This proof depends largely on the fact that the boxes in the E_0 code begin in odd-numbered positions before the mid-point of the arbitrary digit string of a code word and in even-numbered positions after it. Thus false words which begin before the mid-point must begin in even-numbered positions, by theorem 6.3. We go forward until we strike the next real herald, which begins at an even-numbered position in the false word. But at the end of this next real herald we are more than half way through the false word, which then, to be a code word, requires boxes beginning in even-numbered positions. The E_0 code is so constructed that a 01 doublet occupies the positions where the false word needs a box.

It is surprising that we cannot use a similar construction to the E_0 code in the case of an E code with an even number of boxes. The latter case is in fact not nearly as simple, and the only reason which can be given is that the mid-point of the arbitrary digit string in the case where s is even and the p_i's approach 2u - 2 falls close to or actually in the $(\frac{1}{2}s + 1)$th box. This would tend to allow false words beginning close to this box to be identical with code words. In the case of the E_0 code the mid-point fell in the middle of the centre digit string, and this problem was not encountered.

We can note at this point that the E_o code consists of 2u fixed positions for the herald, 2s digits in boxes and 2s.(u - 1) arbitrary digits. This last figure is the one we should specially notice, as it will be used as a lower bound for the number of arbitrary digits in an E code with one more box, in section 6.8, and we shall meet it again later.

We will now determine the upper bound on the length of an even-length synchronous 'herald and boxes' code for fixed u and s. This bound is determined by the number of positions which can be excluded as starting points for a false word because of the format of the code. A position can be eliminated as a starting point for any one of three reasons.

(i) It can suffer direct interference from a real herald. For example, a false word cannot begin in the second position of a code word, which contains a 1.

(ii) To be identical with a code word, a false word needs u consecutive 01 pairs for its herald. If the position where any one of these pairs is needed coincide with positions where a code word has a box, a false word beginning in that position cannot be identical with a code word.

(iii) To be identical with a code word, a false word needs boxes in certain pairs of positions. If any one of these pairs of positions

coincides with a 01 pair of the herald of the next code word, the false word beginning in that position cannot be identical with a code word.

6.7. **Theorem**

A synchronous E code with only u 01 pairs for a herald and no more than s boxes positioned throughout its length has a maximum length of

$$2us + 2u$$

Proof:

(i) Clearly, no false word can begin in a position where a code word has a fixed unit. This means that no false word can start in the positions 2, 4, 6, ... , 2u, u positions in all. Also, no 'herald' of a false word can end in a position where a code word has a fixed 0. This eliminates the positions n, n - 2, ... n - 2u + 2, again u positions in all.

There are no other exclusions possible under case (i).

(ii) Clearly, each box in a real code can clash once and only once with each 01 pair in a hypothetical 'herald' of a false word. This means that the boxes of a code word can prevent a false word from starting in at most us positions.

(iii) Similarly, each 01 pair in the herald of the next code word can occupy the same positions as a hypothetical 'box' of a false word once and only once. Thus the pairs of the next real herald can prevent false words beginning in at most us positions.

Each of these three values given is an upper bound. In the last two cases, it may or may not be attained; for a given value of p there may be two or more clashes between positions where the false word requires 01 and the code word gives a box, or vice versa. In this case the number of eliminated starting positions may be less than the figure given.

The same considerations apply to the sum of these values. The code format may eliminate 2us + 2u positions as starting positions, but the actual figure could be less if there is "doubling up" of eliminations, as suggested above. But if we ensure that each position p is eliminated for only one reason, we can indeed eliminate 2us + 2u positions.

Now we obviously need to eliminate all the positions except the first position as possible starting points. And so the maximum number of positions, or digits, in an E code with fixed u and s is at most 2us + 2u + 1.

But we know that an E code is of even length. Therefore its maximum length is

$$2us + 2u.$$

This is exactly the same figure that we obtained for the length of the E_o code with u 01 doublets and s boxes in section 6.5. Thus, in the E_o code we have a code as efficient as any 'herald and boxes' synchronous code can be.

We cannot generalise this and say that the E_o code of a certain length is as efficient as any synchronous code of that length. In fact, in section 8 we will give examples of synchronous codes which, although somewhat similar to the E_o code, use a more refined format to achieve greater efficiency. The positioning of the u 01 pairs in the herald means that they only eliminate 2u positions; if the same number of 0's and 1's were spaced randomly throughout the word, they could eliminate as many as $2u^2$ positions as starting points for false word. While for other reasons this bound is not attainable, there is obviously room for improvement in this direction.

We are now equipped to consider the case of an E code with an even number of boxes. As previously stated, the construction is not as simple or as straightforward as that of the E_o code, which we obtained by removing one arbitrary digit from the middle string of arbitrary digits of the corresponding K code. Rather than set our sights too high and attempt to construct E codes with an even number of boxes which reach the bound attained by the E_o codes, we shall first find a more realistic lower bound and then attempt to construct codes to surpass this bound.

As a starting point, we shall make the assumption that an E code with u 01 pairs and s even can have at least as many arbitrary digits as the E_o code with the same u and s one less, i.e. $(s - 1)(2u - 2)$. This may appear to be a low estimate for the

number of arbitrary digits, since at first sight it would appear that the code could contain 2u - 2 digits more than this. However, use of this estimate will lead us to the upper bound for codes of this type.

6.8 The E' Code

We define an E' code to be a synchronous E code with an even number of boxes. We shall make the assumption that an E' code has at least $(s - 1)(2u - 2)$ arbitrary positions. We can give at this stage no justification for this assumption, except that, if it is not the case, the E' code is not as efficient as the E_o code with one less box, and if this is so then it is not a practical proposition. But, practical considerations aside, we must allow later results to justify our choice of a lower bound.

Obviously, then, the values of p_i cannot all be very much less than 2u - 2, although a particular p_i may become small and even vanish. However, there are restrictions on the sums of different p_i's, as illustrated in the next lemma.

6.8.1 Lemma

In an E' code

(i) $p_i + p_{i'} > 2u - 4$, $i \neq i'$.

(ii) $p_i + p_{i'} + p_{i''} > 4u - 6$, $i \neq i' \neq i''$.

Proof:

By our assumption,

$$\sum_{j} p_j \geqslant (2u - 2)(s - 1)$$

We also know that

$$\sum_{j,\ i \neq j,\ i' \neq j} p_j \leqslant (2u - 2)(s-2) + 1$$

Therefore, $p_i + p_i' \geqslant 2u - 3.$

Similarly,

$$\sum_{j,\ i \neq j,\ i' \neq j,\ i'' \neq j} p_i \leqslant (2u - 2)(s - 3) + 1$$

whence

$$p_i + p_i' + p_i'' \geqslant 4u - 5.$$

We now have to define some terms we will use later.

6.8.2 Preparation

We have to proceed slowly from here to a proof that an E' code can contain no more than $(s - 1)(2u - 2)$ arbitrary digits. Since the E' code is synchronous, we can apply the results of theorem 6.4. However, this result will be easier to apply if we adapt it slightly; the following lemma verifies our adaption.

6.8.2.1 Lemma

We know from theorem 6.4 that if

$$1 \leqslant j \leqslant s - 1 \tag{1}$$

$$0 \leqslant k \leqslant \tfrac{1}{2}p_j \tag{2}$$

$$\text{and} \quad \sum_{i=j+1}^{s} (p_i + 2) + 2k \geqslant 2u - 1 \tag{3}$$

then there exists integers m, q such that either

$$0 \leqslant m \leqslant s - 1 \tag{4}$$

$$0 \leqslant q \leqslant u - 1 \tag{5}$$

$$\text{and } 2u + \sum_{i=1}^{m} (p_i + 2) - \Big(\sum_{i=j+1}^{s} (p_i + 2) + 2k \Big) = 2q + 1 \tag{6}$$

or

$$j \leqslant m \leqslant s - 1 \tag{7}$$

$$1 \leqslant q \leqslant u - 1 \tag{8}$$

$$\text{and} \quad 2k + \sum_{i=j+1}^{m} (p_i + 2) = 2q - 1 \tag{9}$$

Suppose for given j and k the second alternative holds. By equation (9)

$$2q - 1 - 2k = \sum_{i=j+1}^{m} (p_i + 2)$$

i.e. by (8)

$$2u - 1 - 2k > \sum_{i=j+1}^{m} (p_i + 2)$$

$$\text{Therefore, } 2u - 1 > \sum_{i=j+1}^{m} (p_i + 2).$$

But, by lemma68.1,

$$\sum_{i=j+1}^{j+2} (p_i + 2) > 2u$$

Therefore, $m = j + 1$.

Putting this in equation (9) we find

$$2k = 2q - 5 - p_{j+1} \leqslant 2u - 5 - p_{j+1}$$

Therefore, we find that there must exist m and q satisfying (4), (5) and (6) above unless

(i) p_{j+1} is odd

and (ii) $k \leqslant \frac{1}{2}(2u - 5 - p_{j+1})$

We can thus rephrase theorem 4 in the following way.

For every integral j, k, such that

$$1 \leqslant j \leqslant s - 1 \qquad (1')$$

$$0 \leqslant k \leqslant \tfrac{1}{2}p_j \qquad (2')$$

and $$\sum_{i=j+1}^{s} (p_i + 2) + 2k \geqslant 2u - 1 \qquad (3')$$

there exists integers m, r such that

$$0 \leqslant m \leqslant s - 1 \qquad (4')$$

$$0 \leqslant r \leqslant u - 1 \qquad (5')$$

and $$\sum_{i=j+1}^{s} (p_i + 2) + 2k - 2r = \sum_{i=1}^{m} (p_i + 2) + 1 \qquad (6')$$

where

$$G = \quad 0, \ p_{j+1} \quad \text{even}$$

$$= \quad \tfrac{1}{2}(2u - 3 - p_{j+1}), \ p_{j+1} \quad \text{odd}$$

It is this form of theorem 4 which we shall use in subsequent proofs.

6.8.2.2. Definition

Now we define (k_j) as the set of all integers between 0 and $\frac{1}{2}p_j$ inclusive which must with certain m and r satisfy equation (6'). If there exist m and r which satisfy equation (6') for some k_j, and m' and r' which satisfy equation (6') for some $k_j' > k_j$, there must exist m" and r" which satisfy the equation for k_j'' if $k_j' > k_j'' > k_j$, always provided that $k_j'' > \frac{1}{2}(2u - 5 - p_{j+1})$. If $m = m'$, then we can always provide a solution by choosing r" between r and r'. However, in some cases we will not be able to find a common value for m and m'; and in this case the finding of values of m" and r" which satisfy equation (6') for k" assumes a greater importance.

In a large majority of cases, k_i and k_j will overlap, we will find in the next lemma the only conditions for which there is no overlapping.

6.8.2.3 Lemma

The sets (k_j) and (k_i) have no value in common if and only if

(i) $i = j + 1$

(ii) p_{j+1} is odd

(iii) $p_{j+1} < u - 2$

Proof:

We know $0 \leqslant 2k_i \leqslant p_i$, where k_i is a member of (k_i).

We also know from the argument given in section 6.2.1 above that (k_j) includes 0 unless p_{j+1} is odd. Since 0 is included in (k_i), we know that p_{j+1} is odd.

Now in this case

$$2u - 3 - p_{j+1} \leqslant 2k_i \leqslant p_j,$$

so that,

if there is no overlapping,

$$2u - 5 - p_{j+1} \geqslant p_i$$

therefore, $p_{j+1} + p_i \leqslant 2u - 5$.

By lemma 6.1, this can only occur if $j + 1 = i$.

In this case it follows immediately that

$$p_{j+1} < u - 2$$

We are now in a position to prove a lemma which is important in the proof of the succeeding theorem. We are setting up an apparatus for examining the conditions which the format of the E' code has to obey in order to be synchronous; in particular we shall examine the properties of false words which could begin at or near the middle of the arbitrary digit string of a code word, because, as we mentioned before, that is where the main difficulties arise. And we shall show that these difficulties are of such a magnitude that the only way to overcome them is to reduce the central arbitrary digit strings to such an extent that the E' code begins to resemble the E_o code with one less box.

To simplify the argument, we shall define the set of all k_j, positive integer values for which there exist j, m and q which satisfy equation (6') of lemma 8.2.1, as $A_j(m)$. Obviously, by the fact that the E' code is synchronous, a given k_j must be contained in $A_j(m)$ for at least one value of m.

It is of course quite possible that $A_j(m)$ will be empty; this will certainly be the case if

$$\sum_{i=j+1}^{s} (p_i + 2) \quad \text{and} \quad \sum_{i=1}^{m} (p_i + 2)$$ are either both odd or both even, for then one side of equation (6') will be odd and the other even. The next lemma will show the consequence of such an empty set.

9. Lemma

If $A_j(m)$ is not empty and $A_j(m-1)$ is empty, then every element of (k_j) is contained in $A_j(m')$ for some $m' \geqslant m$.

Suppose the lemma is not true, and that there exists some values of j and k_j for which equation (6') is satisfied for some $m'' \leqslant m-2$ and suitable r, but for no $m' \geqslant m$, whatever value of r we may choose. Since we know that there is at least one element in $A_j(m)$ which we shall call k' and we have assumed an element in $A_j(m-2)$, which will be called k", we know that there are values of r, r' and r", which satisfy the equations

$$\sum_{i=j+1}^{s} (p_i + 2) + 2k' - 2r' = \sum_{i=1}^{m} (p_i + 2) + 1. \quad - \quad (7')$$

$$\sum_{i=j+1}^{s} (p_i + 2) + 2k'' - 2r'' = \sum_{i=1}^{m-2} (p_i + 2) + 1. \quad - \quad (8')$$

We also know that k" is not in $A_j(m)$. Therefore, $A_j(m)$ must have a least member k_1. If the value of r which satisfies equation (8') for k_1 were greater than 0, we could find a lesser value of k to satisfy it.

Therefore,

$$\sum_{i=j+1}^{s} (p_i + 2) + 2k_1 = \sum_{i=1}^{m} (p_i + 2) + 1. \quad - \quad (9')$$

Subtracting (8') from (9'), we get

$$2k_1 - 2k'' + 2r'' = p_{m-1} + p_m + 4$$

$$\geqslant 2u + 1$$

i.e. $k_1 - k'' + r'' \geqslant u + 1$

i.e. $k_1 - k'' \geqslant u + 1 - r'' \geqslant 2$

Since this holds for any element k'' not in $A_j(m)$ then k'' + 1 is neither in $A_j(m)$ nor in $A_j(m - 2)$.

But, by our definition of k' and k'', we know that all the values between them should be in $A_j(m_1)$ for some m_1, if the code is synchronous. This contradicts our assumption that the lemma was false.

Now it is obvious that every time we find an odd p_i, we will also find an empty $A_j(m)$, for if p_m is odd and $A_j(m - 1)$ is not empty, $A_j(m)$ must be empty, since the addition of an odd number to the right-hand side of equation (6') upsets the parity. And, as we have seen in the preceding lemma, the effect of this empty set $A_j(m)$ is to ensure that all the possible values of k_j are in $A_j(m_1)$ for some $m_1 \geqslant m$.

We can by similar means prove the complementary result that if $A_j(m-1)$ is empty and $A_j(m-2)$ is not empty, then all the values of j and k_j can be satisfied for some value of $m_1 \leqslant m-2$ and suitable choice of r. This of course means that unless either $A_j(m)$ or $A_j(m-2)$ is empty we may, if the code is synchronous, finish up with two solution pairs of m and r for each pair of j and k_j.

This does not mean that either $A_j(m)$ or $A_j(m-2)$ is always empty; in fact, for s even it very often is not. But if for any given j and k_j there are two pairs of m and r, which satisfy equation (6'), we are 'doubling up' on our false word 'stoppers'. If we refer to section 6.7, we see that this is not a good thing, since the efficiency of the E_o code only attains the upper bound given because it spreads its 'stoppers' as thinly as possible - only one for each position which must be stopped. But unfortunately, it is impossible to achieve this thin spread in an E' code.

6.9 Lemma

We need one other thing before we embark on the proof of the theorem; we need to show that $A_j(m)$ and $A_m(j)$ are identical.

We can do this by showing that equation (6') is reversible.

We consider equation (6')

$$\sum_{i=j+1}^{s} (p_i + 2) + 2k - 2r = \sum_{i=1}^{m} (p_i + 2) + 1$$

We subtract both sides of the equation from

$\sum_{i=1}^{s} (p_i + 2)$, and we obtain

$$\sum_{i=1}^{j} (p_i + 2) - 2k + 2r = \sum_{i=m+1}^{s} (p_i + 2) - 1,$$

which, under reorganisation, gives

$$\sum_{i=m+1}^{s} (p_i + 2) + 2k - 2r = \sum_{i=1}^{j} (p_i + 2) + 1. \quad - \quad (10')$$

This is exactly what we should have expected. It says that, if we cannot start a false word at a certain point because a box of the code word interferes with the 'herald' of the false word, then there is a corresponding position where the false word cannot begin because there is a 01 pair in the next code herald which occurs where the false word, to be identical with a code word, needs a box. This one-to-one correspondence simplifies our work to some

extent, since if for example $j < \frac{1}{2}s$ and $m \geqslant \frac{1}{2}s$, we can say that if no false word can begin in the arbitrary digit string following the jth box of a real word , no false word can begin in the arbitrary digit string following the $(m + 1)$th box. For example, in the preceding lemma we see by inverting $A_j(m)$ and $A_j(m - 1)$ that $A_m(j)$ is non-empty and $A_{m-1}(j)$ is empty. But this still does not help us with the case of the box in the middle of the arbitrary digit string.

The next theorem strikes at the heart of our trouble - this mid-point of the arbitrary digit string. We first of all locate the boxes which surround the mid-point, and then prove certain properties of the arbitrary digit strings close to this point.

6.10 Theorem

If, in an E' code, we choose t such that

$$\sum_{i=1}^{t} (p_i + 2) + g = \tfrac{1}{2}S < \sum_{i=1}^{t+1} (p_i + 2) \qquad (1)$$

$$\text{where} \quad (i) \quad S = \sum_{i=1}^{s} (p_i + 2)$$

$$\text{and} \quad (ii) \quad 0 \leqslant g \leqslant p_{t+1} + 1$$

then (i) p_{t+1} is odd

(ii) $A_t\,(t + 1)$ is not empty.

Proof:

If $A_j(j)$ is not empty, there is a solution of the equation

$$\sum_{i=t+1}^{s} (p_i + 2) + 2k - 2r = \sum_{i=1}^{t} (p_i + 2) + 1$$

Adding $\sum_{i=1}^{t} (p_i + 2)$ to both sides, we get

$$\sum_{i=1}^{s} (p_i + 2) + 2k - 2r = 2\left(\sum_{i=1}^{t} (p_i + 2)\right) + 1$$

which is a contradiction of our definition of S.

Therefore, $A_t(t)$ is empty.

Suppose $A_t(t + 1)$ is also empty; this will be the case if p_{t+1} is even. And if it is so, then we know that $A_t(t + 2)$ and $A_t(t + 3)$ (or $A_t(t - 1)$ and $A_t(t - 2)$) together contain (k_t), since if $A_t(t + 4)$ is not empty, for some k_t and r,

$$\sum_{i=t+1}^{s} (p_i + 2) + 2k_t - 2r = \sum_{i=1}^{t+4} (p_i + 2) + 1. \qquad (2)$$

and by subtracting each term of (1) from S we get

$$\sum_{i=t+1}^{s} (p_i + 2) - g = \tfrac{1}{2}S > \sum_{i=t+2}^{s} (p_i + 2) - 1 \qquad (3)$$

$$\text{i.e.} \quad \tfrac{1}{2}S + (p_{t+1} + 2) > \sum_{i=t+1}^{s} (p_i + 2) \qquad (4)$$

Substituting in (2)

$$\tfrac{1}{2}S + (p_{t+1} + 2) + 2k_t - 2r > \sum_{i=1}^{t+4} (p_i + 2) + 1$$

But from (1) we get

$$\sum_{i=1}^{t+1} (p_i + 2) + (p_{t+1} + 2) + 2k_t - 2r > \sum_{i=1}^{t+4} (p_i + 2) + 1$$

Therefore, $p_{t+1} + 2 + 2k_t - 2r > \sum_{i=t+2}^{t+4} (p_i + 2) + 1$

$$\text{i.e.} \quad p_{t+1} + p_t + 2 \geq p_{t+2} + p_{t+3} + p_{t+4} + 7$$

By Lemma 6.8.1, this is impossible.

Now if we suppose that $A_t(t+2)$ is empty (i.e. p_{t+2} is also even), then we assume that $A_t(t+3)$ or $A_t(t-1)$ contains (k_t); in the former case, since we have assumed p_{t+1} even, we can choose k_t as 0. This leads us to the equation

$$\sum_{i=t+1}^{s} (p_i + 2) - 2r = \sum_{i=1}^{t+3} (p_i + 2) + 1$$

$$\text{i.e. from (3)} \qquad \tfrac{1}{2}S + g - 2r = \sum_{i=1}^{t+3} (p_i + 2) + 1$$

and from (1)

$$\tfrac{1}{2}S - 2r + p_{t+1} + 1 \geqslant \tfrac{1}{2}S + p_{t+2} + p_{t+3} + 5$$

i.e. $2u + 1 - 2r \geqslant 2u + 2$

which is impossible, by 6.8.1.

Now, if $A_t(t + 2)$ is empty, $A_t(t - 1)$ must contain (k_t). If we choose $k_t = \tfrac{1}{2}(p_t - d)$ where $d = 0$ or 1 depending on the parity of p_t, we get

$$\sum_{i=t+1}^{s} (p_i + 2) + p_t - d - 2r = \sum_{i=1}^{t-1} (p_i + 2) + 1$$

and from equation (4)

$$\tfrac{1}{2}S + g + p_t - d - 2r = \sum_{i=1}^{t} (p_i + 2) - p_t - 1$$

$$\leqslant \tfrac{1}{2}S + g - p_t - 1$$

which is impossible.

Therefore $A_t(t + 2)$ is certainly not empty, and therefore by the preceding lemma

$A_t(t + 2)$ and $A_t(t + 3)$ contain (k_t). We also know that p_{t+2} is odd.

We now have to find a value of m, which we shall call m_1, so that $A_{t+1}(m)$ is not empty.

Obviously, increase in j (here increased to t + 1) should not be accompanied by increase in m. Hence we will assume that $m_1 \leqslant t + 2$.

We find that $A_t(t + 1)$ is empty, by our first supposition, and we know $A_{t+1}(t + 1)$ is empty.

From this we know that either $A_{t+1}(t + 2)$ contains (k_{t+1}) or $A_{t+1}(t - 1)$ contains (k_{t+1}). In the former case, putting $k_{t+1} = \frac{1}{2}(p_{t+2} - d)$ we get

$$\sum_{i=t+2}^{s} (p_i + 2) + 2k_{t+1} - 2r = \sum_{i=1}^{t+2} (p_i + 2) + 1$$

And from (3) and (1)

$$\tfrac{1}{2}s + p_{t+2} - d - 2r \geqslant \tfrac{1}{2}s + p_{t+2} + 3,$$

which is impossible.

Similarly, in the second case putting $k_{t+2} = \frac{1}{2}(p_{t+2} - d)$ we get

$$\sum_{i=t+2}^{s} (p_i + 2) + 2k_{t+2} - 2r = \sum_{i=1}^{t-1} (p_i + 2) + 1$$

But since we can choose $k_t = 0$

$$\sum_{i=t+1}^{s} (p_i + 2) + 0 - 2r' = \sum_{i=1}^{t+2} (p_i + 2) + 1$$

Subtracting

$$p_{t+1} + 2 + 2r - 2r' - p_{t+2} + d = p_{t+2} + p_{t+1} + p_t + 6$$

which is impossible.

This means that our basic assumption that $A_t(t+1)$ was empty must be wrong. Therefore $A_t(t + 1)$ is not empty, and this implies p_{t+1} is odd.

This means that the mid-point of the information digit string of the E' code must fall in an odd-numbered string of arbitrary digits. This ties in very well with the observations made concerning the E_o code; there the mid-point falls precisely in the middle of a string of (2u - 3) digits. But if all the strings of an E' code except the last and one other were 2u - 2 digits long, the mid-point would fall in the middle of a box, and any shortening of the string on either side of this box (to make the string of odd length) must be accompanied by greater shortening of a string on the other side of the box so that the mid-point will fall in the odd-numbered string.

So the first result that this verifies is that, while we can construct an E_o code with efficiency which reaches the upper bound given, it now becomes apparent that an E' code cannot reach this bound.

We shall show in the next theorem that the upper bound on the efficiency of an E' code is even lower than this reasoning would indicate; for if the E' code could reach the bound attained by the E_o code, it would have the same configuration as the E_o code, except that another box and another string of (2u - 2) arbitrary digits would be inserted in the information digit string. So far the theorems have only shown that the mid-point of the information string must be adjusted to fall in the middle of an odd-numbered arbitrary digit string; this should be possible with the removal of only a few digits from the arbitrary digit strings. However, the next theorem will show that we have to remove the equivalent of a complete digit string, and, incidentally, will justify our choice of a lower bound on the total number of arbitrary positions.

As an example, let us consider an E' code with four 01 pairs for a herald and two boxes. In order to prevent a false word identical with a code word beginning in the tenth position (the last position of the first box), the second box must begin in

the sixteenth position (or fourteenth, or twelfth). Now consider a false word beginning in the eleventh position. The first position of the second 'box' of the false word corresponds to an even-numbered position in the next code word, and so cannot correspond to a 01 pair in the code word. Since the E' code is synchronous, the false word is not identical with a code word, and so the first 'box' of the false word must correspond to a 01 pair in the next code word. This can be so only if the code word has length no greater than eighteen. In comparison, we can construct a K code with four 01 pairs, two boxes and length 27. This simple example shows us that we cannot expect to obtain as high an efficiency with an E' code as with a K code.

There are two results we shall need before we begin the final assault. One follows directly from the last theorem.

We know that $A_t(t+1)$ is not empty. This implies that for some k_t, r

$$\sum_{i=t+1}^{s} (p_i + 2) + 2k_t - 2r = \sum_{i=1}^{t+1} (p_i + 2) + 1 \qquad - (1)$$

Since $A_{t+1}(t)$ is also not empty, for some k_{t+1}, r'

$$\sum_{i=t+1}^{s} (p_i + 2) + 2k_{t+1} - 2r' = \sum_{i=1}^{t} (p_i + 2) + 1 \qquad - (2)$$

The minimum value of k_t in $A_t(t+1)$ is, by 6.8.2.2 at most

$$\tfrac{1}{2}(2u - 3 - p_{t+1}).$$

We shall now prove the following lemma, which we shall need in our proof of the succeeding theorem. To enable us to give particular values to k_t, we need to show first that such values will be in the set $A_t(t + 1)$.

Part of the proof rests on the duality of $A_t(t + 1)$ and $A_{t+1}(t)$, which follows from the fact that equation (2) is identical with equation (1) with $(p_{t+1} + 2)$ substracted from both sides.

6.11 Lemma

The integer $\frac{1}{2}(2u - 3 - p_{t+1})$ is in $A_t(t + 1)$ and $A_{t+1}(t)$.

The integer $\frac{1}{2}(p_t - d)$ is in either $A_t(t + 2)$ or both $A_t(t + 1)$ and $A_{t+1}(t)$,

where $d = 1$, p_t odd

0, p_t even

Proof:

We know by the duality of $A_t(t + 1)$ and $A_{t+1}(t)$ that any integer that is in one is also in the other.

Lemma 6.8.2.1 tells us $\frac{1}{2}(2u - 3 - p_{t+1})$ is in $A_t(t + 1)$.

Suppose $\frac{1}{2}(p_t - d)$ is not in $A_t(t + 1)$ because it is under the lower limit given by lemma 6.8.2.1.

$$\text{i.e. } \tfrac{1}{2}(p_t - d) \leqslant \tfrac{1}{2}(2u - 5 - p_{t+1})$$

$$\text{Therefore, } p_t + p_{t+1} \leqslant 2u - 5 + d$$

$$\leqslant 2u - 4.$$

This contradicts lemma 6.8.1.

Therefore, $\frac{1}{2}(p_t - d)$ is in $A_t(t + 1)$ or in $A_t(t + 2)$. If in the former, it is by duality also in $A_{t+1}(t)$.

Having now obtained a set of values of integers which are in $A_t(t + 1)$, we shall use these and equation 1 from section 6.10 to prove the next theorem. Since we cannot prove that $\frac{1}{2}(p_t - d)$ is in $A_t(t + 1)$, we must prove the theorem for the case when it is and the case when it is not.

6.12 <u>Theorem</u>

The maximum length of an E' code for given u and s is

$$2us + 2$$

Since 2u of these are in the herald, and 2s are in boxes, we know that an E' code can contain no more than

$$2us + 2 - 2u - 2us = 2(u - 1)(s - 1)$$

completely arbitrary digits (not counting those in boxes).

Proof:

We shall use our equations

$$\sum_{i=t+1}^{s} (p_i + 2) + 2k_t - 2r = \sum_{i=1}^{t+1} (p_i + 2) + 1 \qquad (1)$$

and

$$\sum_{i=t+2}^{s} (p_i + 2) + 2k_{t+1} - 2r' = \sum_{i=1}^{t} (p_i + 2) + 1 \qquad (2)$$

and substitute into them either $\frac{1}{2}(p_t - d)$ or $\frac{1}{2}(2u - 3 - p_{t+1})$, which, lemma 6.11 tells us, are in $A_t(t + 1)$ or $A_t(t + 2)$

(i) Suppose $\frac{1}{2}(p_t - d)$ is in $A_t(t + 1)$

(a) $t \geqslant \frac{1}{2}s$

Then $\sum_{i=t+2}^{s} (p_i + 2) \leqslant 2u(\frac{1}{2}s - 1) + 1$

Substituting $k_{t+1} = \frac{1}{2}(2u - 3 - p_{t+1})$ in equation (2)

$$\sum_{i=t+2}^{s} (p_i + 2) + 2u - 1 \geqslant \sum_{i=1}^{t+1} (p_i + 2) + 1$$

Therefore $\sum_{i=1}^{t+1} (p_i + 2) \leqslant 2u(\frac{1}{2}s - 1) + 1 + 2u - 2$

Therefore, $S \leqslant 2u\,(s - 1)$

(b) $t < \frac{1}{2}s$

Then $\sum_{i=1}^{t}(p_i+2) \leq 2u(\frac{1}{2}s - 2) + p_t + 2$

Using equation (2), and choosing $k_{t+1} = \frac{1}{2}(p_t - d)$

$$\sum_{i=t+1}^{s}(p_i + 2) + p_t - d - 2r \leqslant 2u(\tfrac{1}{2}s - 2) + p_t + 3$$

$$\sum_{i=t+1}^{S}(p_i+2) \leqslant 2u(\tfrac{1}{2}s - 1) + 2$$

Therefore, $S \leqslant 2u(s - 3) + 4 \leqslant 2u(s - 1)$

(ii) Suppose $\frac{1}{2}(p_t - d)$ is in $A_t(t + 2)$.

Therefore,

$$\sum_{i=t+1}^{s}(p_i + 2) + p_t - d - 2r_i = \sum_{i=1}^{t+2}(p_i + 2) + 1 \qquad (3)$$

(a) As before, if $t \geqslant \frac{1}{2}s$ we can choose $k_t = 2u - 3 - p_{t+1}$

and show that

$$S \leqslant 2u(s - 1)$$

(b) $t < \frac{1}{2}s$

$$\sum_{i=1}^{t} (p_i + 2) \leqslant 2u(\tfrac{1}{2}s - 1)$$

Substituting $k_t = u - 2 - K$ in equation (2)

we get

$$\sum_{i=t+2}^{s} (p_i + 2) \leqslant 2u(\tfrac{1}{2}s - 1) + 1 - 2u + 4 + 2K + 2(u - 1)$$

and $p_{t+1} + 2 \leqslant 2u - 1 - 2K$

Therefore, $S \leqslant 2u(s - 1) + 2$

This latter case provides the maximum value for the length of the information string for given u and s. If we add on the 2u digits in the herald, we arrive at the word length

$$n = 2us + 2$$

while if we subtract the 2s digits in the boxes, we find that there are $2(u - 1)(s - 1)$ arbitrary digits.

Now if we refer back to section 6.6 and to section 6.8, we discover that this is precisely the number of arbitrary digits in an E_0 code with one less box than the E' code we have just discussed; it is also the figure we chose as the lower bound on the number of digits if the E' codes were to be practically worthwhile.

We now know that it is also an upper bound, but we still have not settled the question as to whether an E' code can be constructed with this number of arbitrary digits. It would be remarkable if the only way to transform an E_o code into an E' code happened to be to remove two or more arbitrary digits to make way for the extra box.

In fact, this is not the case. We shall construct in the next section a particular class of E' codes which attain this new upper bound and then we will prove that codes in this new class are synchronous.

6.13 Definition

An E_E code is an E' code with the following values for the p_i.

$p_s = 2u - 1$

$p_{is} = 2u - 3$

$p_{is+1} = 0$

$p_i = 2u - 2$ otherwise.

It is not necessary to have one of the p_i equal to 0; the format may be altered quite considerably and still remain that of a synchronous code. But we have proved that there can be no increase

in the total number of arbitrary digits if the code is to remain synchronous; and the zero placed where it is relates the E_E code closely to the E_o code, since it is now apparent that the E_E code is just an E_o code with two boxes after the odd-numbered digit string instead of one. Since this is the case, we should expect the E_E code to be synchronous; our expectations are fulfilled.

6.14 Theorem

The E_E code is synchronous.

Proof:

By theorem 6.4, the code is synchronous if for all j , k, such that $1 \leqslant j \leqslant s - 1$, $0 \leqslant k \leqslant \frac{1}{2}p_j$, there exist m and q such that

$$0 \leqslant m \leqslant s - 1$$

$$\text{and } 0 \leqslant q \leqslant u - 1$$

$$\text{and } 2u + \sum_{i=1}^{m} (p_i + 2) - \left(\sum_{i=j+1}^{s} (p_i + 2) + 2k \right) = 2q + 1$$

(i) Suppose $j \neq \frac{1}{2}s$.

Choose $m = s - j$, $q = u - 1 - k$

(a) $j \leqslant \frac{1}{2}s - 1$ i.e. $m \geqslant \frac{1}{2}s + 1$

$$\sum_{i=1}^{m} (p_i + 2) = 2u(m - 1) - 1 = 2u(s - j - 1) - 1$$

$$\sum_{i=j+1}^{s} (p_i + 2) = 2u(s - j - 1)$$

Therefore, LHS $= 2u - 2k - 1 =$ RHS

(b) $j \geqslant \frac{1}{2}s + 1 \qquad m \leqslant \frac{1}{2}s - 1$

$$\sum_{i=1}^{m} (p_i + 2) = 2um = 2u(s - j)$$

$$\sum_{i=j+1}^{s} (p_i + 2) = 2u(s - j) + 1$$

Again, L.H.S. = R.H.S.

(ii) $j = \frac{1}{2}s$ Choose $m = \frac{1}{2}s - 1$

Since $p_j = 2u - 3$, $k \leqslant u - 2$

Therefore, we can choose $q = u - 2 - k$.

$$\sum_{i=1}^{m} (p_i + 2) = 2u(\tfrac{1}{2}s - 1)$$

$$\sum_{i=j+1}^{s} (p_i + 2) = 2u(s - \tfrac{1}{2}s - 1) + 3$$

Therefore, L.H.S. = $2u - 2k - 3$ = R.H.S.

Thus for every permissible pair of j and k_j we can find m and q which satisfy the equation. This proves the theorem.

We have now proved that the E_o codes and the E_E codes are the most efficient E codes for s odd and even respectively.

We are now, then, in a position to seek an answer to the question: What is the most efficient code for a given even length?

There is another complication which enters the picture at this point. Obviously, if we can find an E_o code which has about the maximum length possible for the given u and s, it will obviously be more efficient than the corresponding K code of length one less. But if the most efficient code for a certain even length is an E_E code, we have to consider the question: is it more efficient than the K code of length one less?

To use an E_o code or E_E code for a given length n we have to be able to solve one of the equations

$$2us + 2u = n$$

$$\text{or} \quad 2us + 2 = n \quad \text{with integers u and s.}$$

If these were the only values of n for which E_o codes or E_E codes were suitable, they would not be very important. But, because of their similarity to the K codes, it can readily be understood that we can construct a synchronous code of any given length by finding a larger value of n which satisfies one of the equations above, and removing 2i digits from the arbitrary digit strings to reduce the length to the value required. We shall call the classes of codes which can be derived in this way from the E_o and E_E codes the E_o' and E_E' codes respectively.

If we remove digits with a certain amount of finesse, we can ensure that the resulting codes are still synchronous. For example, removal of the first arbitrary digit in the first string will ensure that the left hand side of equation (1) from section 6.10 remains unaltered.

However, in practice, we may want to find the minimum code length which will carry a certain amount of information. In this case, if the E_E' code can not carry more information than the K` code of length one less, it is not practicable to use the E_E' code. We shall also attempt in the next few sections to construct codes which satisfy these requirements.

6.15 Efficiencies of the E codes

We know that an E_o code, of length 2us + 2u, has 2s digits in boxes and 2((u - 1)s) arbitrary digits. This means that it has a maximum efficiency when 5u = s; the value of this efficiency is

$$1 - 1.288n^{-\frac{1}{2}} + .415\ n^{-1}$$

This compares favourably with the efficiency of the K code, at best

$$1 - 1.288n^{-\frac{1}{2}} + .415n^{-1} + \tfrac{1}{2}n^{-3/2} + O(n^{-5/2});$$

This comparison reflects the similarity of their construction.

But if we consider an E_E code of length 2us + 2, we find that it has 2s digits in boxes but only 2((u-1)(s-1)) arbitrary digits. Again considering the best case, when 5u = s, we find the efficiency is

$$1 - 1.288n^{-\frac{1}{2}} + .644n^{-3/2} + O(n^{-5/2}).$$

There is no term in n^{-1} in this expression; and so this code is not going to compare in efficiency with the E_0 code and the K code. Although it may be argued that this term will become exceedingly small as n becomes large, it should be remembered that the number of words is 2^{Yn}, where Y denotes the efficiency; and so even a term in n^{-1} can make a great difference to the number of possible code words.

Having proved that the E_E code is theoretically inferior in efficiency to the E_0 code for most n we shall now determine the precise values of n for which it is superior.

The formulae given only refer to the case when c = 0; for some values on n it may not be possible to choose a value for c close to 0 for either an E_0' code or an E_E' code of a certain length, while if we want to use an E_0' code, we cannot find any c' near 0 which give this value of n. It is in this case that it may be better to use an E_E code for certain n.

6.16 Theorem

The only values of n for which the E_E' code can be superior in efficiency to the most efficient E_o' code of the same length are those values which are solutions of the equation

$$2us + 2 = n$$

for some u and s.

Proof:

We can quickly show that an E_o code of length $2u_os_o+2u_o = n$ is superior in efficiency to an E_E' code with one more box and two less arbitrary digits, since the box is less efficient than the two arbitrary digits it replaces. Since this is the case, if both the E_o code and the E_E' code are reduced by two arbitrary digits, the E_o' code of length $(n - 2)$ so obtained must be superior in efficiency to the E_E' code of the same length, since this latter E_E' code still has a box in place of two arbitrary digits of the E_o' code.

This state of affairs must persist for the lengths $(n - 4)$, $(n - 6)$ etc. until we reach the value of n which is equal to $2u_o(s_o - 1) + 2$. Since this value is greater than

$2u_o(s_o - 2) + 2u$, the most efficient E_o code of this length has $u = u_o$ and $s = s_o$. Thus it has $2s_o$ digits in boxes and only $2(u_o - 1)(s_o - 1) - 2u_o$ arbitrary digits.

But we know the E_E code of length $2u_o(s_o - 1) + 2$ has $2(s_o - 1)$ digits in boxes and

$2(u_o - 1)(s_o - 1) - 2u_o + 2$ arbitrary digits.

Thus the E_E code has two arbitrary digits instead of the digits inside the extra box in the E_o' code; the E_E code is the more efficient for that particular length. However, for a length 2 less we can use an E_o code with $(s_o - 2)$ boxes, and so by induction we can arrive at the result stated in the theorem.

This is not to say that all the solutions of the given equation represent codes which are more efficient than the best E_o' code of the same length; indeed an E_E code of length 94, which is the length given by substituting $u = 23$ and $s = 2$ in the equation above, only has 44 arbitrary positions and 4 digits in boxes. If we use an E_o code with $u = 3$, $s = 15$ and $c = 1$, we obtain 58 arbitrary positions and 30 digits in boxes.

So we could rephrase our statement and say that if s is close to $5u$, the E_E code of length $2us + 2$ may have a higher

efficiency than any E_o' code of the same length. But this statement is not definite enough to be applied strictly; and we shall use it with caution when we begin representing actual values of n.

We have now shown that the E_o' code of a given length is generally superior in efficiency both to the E_E' code of the same length and the K' code of length one less. However, it is instructive to compare the latter two, and see again how heavily the restrictions that both the length of the code and the number of boxes are even weigh against the efficiency of the E_E' code.

We shall now compare the E_E codes with K codes. Since the formula for the length of the E_E codes does not bear the same similarity to that for the K codes as does that of the E_o codes, we aid our ability to make a suitable comparison by defining a K' code as a K code which may have extra boxes in place of arbitrary digits. By means of this device, we can construct a code which is a close approximation to a K code and which has a similar number of 01 pairs and boxes to any E_E code, and a length close to that of the E_E code.

Obviously, a K' code with $u = u_o$, $s = s_o$ and length n cannot be more efficient than an E_E' code with $u = u_o$, $s = s_o$ and length $n + 1$ whatever value c takes in both cases, for the former is

always one arbitrary digit shorter than the latter. But there is no restriction on the parity of s in the former case, and so we shall compare a K' code, with $u = u_o$ and $s = s_o - 1$, with the E_E' code given.

6.17 Theorem

The E_E' code of a certain length is never more efficient than a K' code of length one less. However, the E_E' code is capable of carrying more information than the K' code.

Proof:

Since the lengths are different by only one

$$2u_o s_o + 2 - 2c_E = (2u_o(s_o - 1) + 2u_o + 1 - 2c_K) + 1$$

This equation solves with $c_E = c_K$.

Now the E_E' code has $2s_o$ digits in boxes and $(2u_o s_o - 2u_o - 2c_E - 2s_o + 2)$ arbitrary digits.

The K' code has $2(s_o - 1)$ digits in boxes and $(2u_o(s_o - 1) - 2(s_o - 1) + 1 - 2c_K)$ arbitrary digits. Thus the K' code has one more arbitrary digit and one less box than the corresponding E_E' code.

Since the one arbitrary digit is more efficient than the box, and since in all other things the K' code and the E_E' code are equal, the K' code is the more efficient.

However, the extra box will always carry at least as much information as the one arbitrary digit; the E_E' code can always carry more information than the K' code of length one less. This applies especially to the E_E codes with suitable choices of u and s, and means that they can carry at least as much information as any code of their length or less.

6.18 Reactions to ϕ-encoding (see 4.2 et seq)

So far, the codes have been compared under the tacit assumption that mixed binary and ternary input is available to be transmitted in the information string. If we consider the effect of ϕ-encoding on each type of code in turn, we shall find certain changes in the relationships between them.

Firstly, consider the E_o' codes. Under ϕ-encoding, we will have to apply the same format adaption as we used on the K codes in 5.13; we place a fixed 0 in the second position of one of the boxes. But this only costs about $\frac{1}{2}$ of an arbitrary digit (since two boxes can carry as much information as three arbitrary digits) compared to a code with an even number of boxes.

Similarly, if we consider a K' code with an odd number of boxes (the only kind we worked with in this section), we again lose about $\frac{1}{2}$ an arbitrary digit, compared to, in particular, an E_E' code. Of course, since an E_E' code has an even number of boxes, it requires no format adaption to make it suitable for ϕ-encoding. Thus if only binary input is available, the E_E' code becomes a slightly better proposition compared to the other codes.

In fact, under this system, it is at least as efficient to use an E_E' code as an E_O' code of the same length. If we refer back to theorem 6.16, we see that if an E_O' code and an E_E' code with the same value of u are of the same length, then the E_O' code has two arbitrary digits in place of the box in the E_E' code. Now, if we compare the odd box and two arbitrary digits in the E_O' code with the two boxes to which they correspond in the E_E' code, we find that each set can carry three arbitrary information digits. Thus in every case the E_E' code can carry at least as much information as the E_O' code of the same length; in the particular case given in theorem 6.16, the E_E' code is more efficient than the E_O' code of the same length.

Since both the E_O' code and the K' code with an odd number of boxes react the same way to ϕ-encoding, the E_E code will be more efficient than the K' code of length one less.

So under ϕ-encoding, the E_E' codes have developed from codes useful only in isolated cases, if binary and ternary input are both available, to codes which are as efficient as any code when even-length codes are required.

6.19 Summary

In this section we have constructed 'herald and boxes' synchronous codes with even length. We have shown that if the constructed code has an odd number of boxes, it is at least as efficient as any 'herald and boxes' code of the same length.

If, however, the constructed code must have an even number of boxes, the efficiency drops and the code, although it is proved to be as efficient as any like it, has no advantage over one with an odd number of boxes.

But if we must use ϕ-encoding, this latter code assumes much greater significance, and under these conditions is shown to be at least as efficient as any of the other codes given.

In many places the constructed codes are compared with the K' codes of length one less; in the majority of cases they show

up favourably, and thus ensure that they should be considered in attempts to find the minimum synchronous code length to carry a given number of information digits.

7. FIXED PLACE SYNCHRONOUS CODES WITH FALSE HERALDS

7.1 Introduction

We have seen that codes like Gilbert's code prohibit false heralds; but we know from our considerations of the M, K and E codes that such a prohibition is not necessary. In fact, herald sequences which are identical with a code word herald can begin at numerous positions in the information strings of the latter three types of code mentioned; the codes only ensure that no complete false word can be identical with a code word.

So far we have only compared these 'herald and boxes' codes which allow false heralds, to Gilbert's codes. In this comparison, Gilbert's code suffers because of the restriction on it that there must be no possibility of a false herald sequence, as well as from the fact that we have shown that fixed places, which cost one arbitrary position whenever they are used, are not as efficient synchronising units as boxes, which cost one digit for every two of them used if the binary input must be $\emptyset$-encoded, or slightly less than this if ternary input is available (the value in this latter case works out to about 4 arbitrary digits lost for every 10 boxes used).

However, because of the Gilbert's code's restriction on false herald sequences, the comparisons we have made are not fair to the class of fixed position synchronous codes in general; we would tend to expect that we could find a synchronous code which uses only fixed positions to achieve synchronisation, but which allows false herald sequences, and we should further expect that such a code would be more efficient than Gilbert's code of the same length. As yet we cannot say whether it would be as efficient as one of the M, K or E codes for a certain length and for a certain practical purpose, for although we know that its theoretical efficiency will not be as high as that of a 'herald and boxes' code (because boxes are superior in efficiency to fixed placed for carrying information) we cannot yet assess the effect of the reduction in efficiency of the 'herald and boxes' codes, due to such things as ϕ-encoding, in a particular case. Also, in a particular case, ϕ-encoding may not be considered worthwhile; and it is here that we require a code which uses only fixed places to achieve synchronisation, and which has the maximum number of arbitrary digits in its given length.

Before we begin to consider particular codes, we will determine the relationship between the length of the code and the minimum number of fixed places to enable the code to be synchronous. For example, a code with a fixed 01 herald followed by ten arbitrary digits is not synchronous; in fact, if we reduce the number of arbitrary digits to two, the code is still not synchronous.

For both these lengths, we need to fix more positions before the code can hope to be synchronous. In the latter case, there is no difficulty, if we fix the third positions in the code word as either 0 or 1, and leave only the last position arbitrary, the code is obviously synchronous. But fixing three positions in order to transmit one arbitrary digit is extremely wasteful, although, surprisingly, it is the best we can do for a code of length 4.

In the former case (of the 01 followed by ten arbitrary digits) there is no such easy solution. We could construct Gilbert types of codesby either fixing all the even positions among the arbitrary digits as 0, or by fixing the two positions directly after the 01 herald as units, and fixing the sixth and last digits of the arbitrary digit string as zeroes.

We cannot prove by examination that the latter code (which is more efficient than the former) is a most efficient fixed place synchronous code of length 12 (in fact, it is not, as will be shown later).

The next theorem will determine the minimum number of fixed places in a synchronous code of given length; but first we need a neat mathematical expression of the condition "A fixed place code is synchronous". This is given to us by the next lemma.

7.2 Lemma

If we have a code with 0's fixed in the positions a_i , i = 1, ... p, and 1's fixed in the positions b_j , j = 1, ...,q, the code is synchronous if and only if for each t = 1, ... , n - 1, there exist positive integers i, j such that either

$$a_i - b_j = t \pmod n$$

or

$$b_j - a_i = t \pmod n$$

Proof:

Suppose $b_j - a_i = t \pmod n$.

Now consider a false word beginning in the $(t' + 1)$th position of a code word.

The $(t' + s) \pmod n$th position of the code word (or the next code word) corresponds to the sth position of the false word for all s between 1 and n inclusive. In particular, putting $s = a_i$, the b_jth position of the code word corresponds to the a_ith position of the false word.

But the b_jth position of the code word holds a fixed 1, and the false word, to be identical with a code word, needs a 0 in its a_ith position. Thus no false word can start in the $(t' + 1)$th position.

The same argument applies if $a_i - b_j = t' \pmod n$.

On the other hand, if there is no solution of either equation for a given t', a false word can begin in the (t' + 1)th position, since all the positions where the false word has fixed digits correspond to positions in a code word which contain either arbitrary digits or the 'right' fixed digit, and these arbitrary digits could be arranged to make the false word identical with a code word.

We can now proceed to our theorem; once we have found our lower bound on the number of fixed places, we shall construct codes which attain this bound.

7.3 Theorem

A fixed place type synchronous code of length n has at least r fixed places, where r is the least integer such that

$$r \geqslant (2(n-1))^{\frac{1}{2}}$$

Proof:

We will use the notation of the previous lemma.

Since the code is synchronous, for every t we need a pair (a_i , b_j) which satisfies one of the equations. Now it is possible for a given t that there may be more than one solution pair. But in the best possible case, there will be one solution

pair (a_i , b_j) for each value of t between 1 and n - 1. In this case, if

$$a_{i_1} - b_{j_1} = t \pmod n$$

then $a_{i_2} - b_{j_2} \neq t \pmod n$ for any $j_2 \neq j_1$ $i_2 \neq i$,

and $b_j - a_i \neq t$ (any i, j).

But even in this case, p values of a_i and q values of b_j cannot provide more than pq solutions for values of t in the first equation, and a similar number of solutions of t in the second equation. Thus there can be no more than 2pq values of t which are solutions of either equation.

We know from our previous lemma, that there must be at least (n - 1) solutions of one or other of the equations, so that there can be at least one solution for every value of t from 1 to n - 1.

This leads us to the inequality

$$2pq \geqslant n - 1.$$

But r = p + q, and the solution of the problem 'Minimise p + q subject to $2pq \geqslant n - 1$' can quickly be shown to be

$$p + q \geqslant (2(n - 1))^{\frac{1}{2}}$$

whence

$$r \geqslant (2(n - 1))^{\frac{1}{2}}$$

This gives us our lower bound on the number of fixed places necessary. We can construct codes which attain this bound for all n. These codes, which we shall call the F codes, are thus the most efficient fixed place synchronous codes possible.

7.4 Definition

An F code of length n has

(i) fixed 0's in the positions 1, 2, ... , r_1, and

(ii) fixed 1's in the positions n, n - r_1 , ... n - $(r_2 - 1)r_1$

where (a) r is the least integer not less than $(2(n-1))^{\frac{1}{2}}$

(b) r_1 is the least integer not less than $\frac{1}{2}r$

(c) $r_2 = r - r_1$

This code has r_1 fixed 0's and r_2 fixed 1's — r fixed positions in all. As such, it reaches the bound of theorem 7.3. We now have to prove that it is synchronous, and lemma 7.2 tells us this will be so if and only if we can find solutions for one of the equations for every t from 1 to n - 1.

We shall later compare the F code of a certain length with other types of codes of the same length. Since it is at least as efficient as Gilbert's code of the same length, we shall only compare it with codes of the 'herald and boxes' type.

Before we proceed to the proof of the theorem, we need the following lemma. It will be used in the theorem to show that false words cannot begin in positions close to the centre of a code word; thus, as is the case with the E codes, the positions we have to worry most about are those near the centre of the code word.

7.4.1 Lemma

If r is the least integer not less than $(2(n-1))^{\frac{1}{2}}$, r_1 the least integer not less than $\frac{1}{2}r$, and $r_2 = r - r_1$,

$$\text{then } r_1 \cdot r_2 \geqslant \tfrac{1}{2}(n-1)$$

Proof:

$$r^2 \geqslant 2(n-1)$$

(i) r is even

$$r_1 = r_2 = \tfrac{1}{2}r$$

$$r_1 \cdot r_2 = \tfrac{1}{4}r^2 \geqslant \tfrac{1}{2}(n-1)$$

(ii) r is odd

r^2 is odd, and so

$$r^2 - 1 \geqslant 2(n-1)$$

$$r_1 = \tfrac{1}{2}(r+1),\ r_2 = \tfrac{1}{2}(r-1)$$

$$r_1 r_2 = \tfrac{1}{4}(r^2 - 1) \geqslant \tfrac{1}{2}(n-1)$$

Therefore, In all cases $r_1 r_2 \geqslant \tfrac{1}{2}(n-1)$.

Now that we have this lemma, we can proceed with the theorem.

7.5 Theorem

An F code is synchronous.

Proof:

In the notation of lemma 7.2, the a_i's are i, i = 1, 2, ... , r_1 and the b_j's are $n - (j - 1)r_1$, j = 1, 2, ... , r_2.

By lemma 7.2, we only have to show that for all t from 1 to n - 1, there is a solution of either $a_i - b_j = t \pmod n$,

or $b_j - a_i = t \pmod n$.

Considering $a_i - b_j$	Considering $b_j - a_i$
$a_1 - b_1 = 1 \pmod n$	$b_1 - a_1 = n - 1 \pmod n$
$a_2 - b_1 = 2 \pmod n$	$b_1 - a_2 = n - 2 \pmod n$
...	...
$a_{r_1} - b_1 = r_1 \pmod n$	$b_1 - a_{r_1} = n - r_1 \pmod n$
$a_1 - b_2 = r_1 + 1 \pmod n$	$b_2 - a_1 = n - r_1 - 1 \pmod n$
...	...
$a_{r_1} - b_{r_2} = r_1 r_2 \pmod n$	$b_{r_2} - a_{r_1} = n - r_1 r_2 \pmod n$

Thus $(a_i - b_j)$ satisfy all the values of t from 1 to $r_1 r_2 \pmod n$ and $(b_j - a_i)$ satisfy all the values of t from $n - r_1 r_2$ to n - 1.

But lemma 7.4 showed us that

$$2r_1r_2 \geqslant (n - 1)$$

therefore, $r_1r_2 \geqslant n - 1 - r_1r_2$

i.e. $r_1r_2 + 1 \geqslant n - r_1r_2$.

Therefore, since $n - r_1r_2$ and all greater values have as a solution pair a_i and b_j of $b_j - a_i$, $r_1r_2 + 1$ also has a solution pair a_i' and b_j' of $b_j - a_i$.

Thus there is a solution for all the numbers from 1 to n - 1.

By lemma 7.2, this proves that the F code is synchronous.

By the definition of the F code, the F code is the most efficient fixed place synchronous code for any given length n. From the proof above, it is evident that the F code can be constructed for both even and odd lengths. This is one of its advantages over the M, K and E codes, in that we do not have to first determine whether we want an even or an odd length code; we know that, whatever the parity, the F code of that length will be synchronous.

Of course, the big advantage of the F code over the M, K and E codes is that it requires neither ternary input nor a ϕ-encoder; in fact it needs very little more apparatus than a Gilbert code, and so it represents an advance on Gilbert's code. It transmits more information in a given length while still retaining the synchronising property, and thus relieves slightly the greatest burden on synchronous code efficiency - the digits which are restricted in order to ensure that synchronisation may be regained at the first correctly received word after it is lost, but which, when the message is coming through normally, only take up what could be much-needed space.

This, then, is the reason that F codes can do the work of Gilbert's codes, and do it better. When we use a Gilbert code, we scan the incoming digits for the first herald sequence that appears after synchronisation is lost. This requires a holding area for the length of the herald. With an F code, we scan the incoming digits for the first acceptable word; this also requires a holding area, this time the length of the word, but this is the only change from the hardware necessary to implement Gilbert's code.

Obviously, synchronisation will be regained with the first correct code word that comes through, since if the digit string is error free the first position of a code word, and only the first position, can begin a string of n digits which is acceptable as a code word.

We have proved before that fixed places are not as efficient as boxes in a synchronous code; therefore the F code of a given length can never, at least theoretically, be as efficient as the most efficient 'herald and boxes' code of that length. However, in practice this may not be true, since if for example only binary input is available, the F code may be as efficient as the corresponding K code (or E code) after it has been ϕ-encoded. In this next section we will investigate this possibility.

There is another side to the argument. If a ϕ-encoded K code is only slightly more efficient than a F code of the same length, would it not be better to save ourselves the trouble of a ϕ-encoder and use an F code? We cannot answer this question directly, since it depends largely on practical considerations. But we must remember in all our arguments that a K code may not be better to use than a F code of the same length just because it is more efficient.

We will first compare the F code with the ϕ-encoded K code of the same length. This comparison must fall into two parts; the case when the K code has an even number of boxes, and the case when the number of boxes is odd. The comparison will be more favourable to the F code in the latter case, when ϕ-encoding reduces the efficiency of the K code more. So we shall consider the former case first.

7.6 Comparison of the F codes and K codes

Suppose we have a length equal to $8a^2 + 2a + 1$ where a is any integer. With a K code, if we have 4a boxes and a 01 pairs we obtain a code able to carry $8a^2 - 2a + 1$ binary input digits of information — $8a^2 - 8a + 1$ by arbitrary digits, and 6a in the 2a boxes.

Now an F code of the same length requires at least $(16a^2 + 4a)^{\frac{1}{2}}$ fixed positions. Thus it requires 4a + 1 fixed positions, and so has only $8a^2 - 2a$ arbitrary positions.

Now if we consider similar codes each with two arbitrary digits less, the new F code will still require 4a + 1 fixed positions and so it will still have one 'arbitrary digit' less than the corresponding K code. We can continue considering codes with the number of arbitrary digits less by two and repeating the argument until we reach the stage where the F code only requires 4a fixed positions; the first case is when $n = 8a^2 + 1$.

If we have a K code two arbitrary digits shorter than the K code of length $8a^2 + 3$, the new K code will only be as efficient as the F code of the same length, since the F code now only requires 4a fixed positions and so one of the fixed positions can become an arbitrary digit. However, at this length, since we have removed 2a arbitrary digits from the original K code, we must consider another K code — the one with 4a - 1 (an odd number of boxes) boxes and a 01 pairs for a herald.

We know that the F code of length $8a^2 + 1$ has $8a^2 - 4a + 1$ arbitrary digits, the same as the former K code reduced to that length. The later K code can carry $8a^2 - 10a + 3$ binary input digits through its arbitrary digits and $6u - 3$ binary input digits through $4u - 2$ of its boxes. But in the odd box, under ϕ-encoding, we can do nothing but fix the last digit as a zero, and so we can only carry one binary digit in it. Thus this latter K' code of length $8a^2 + 1$ can also only carry $8a^2 - 4a + 1$ binary input digits.

Thus, for $n = 8a^2 + 1$, the F code is as efficient as any ϕ-encoded K code, and we can consider the codes shorter by two arbitrary digits without altering this relationship until we have to consider the K code with u 01 pairs for a herald and $4u - 2$ boxes. The first case we must consider is when $n = 8a^2 - 2a + 1$. We are again comparing an F code with a K code with an even number of boxes, and the results are similar to those obtained at the start of this section - the F code can carry one less binary input digit than the K code of the same length.

We have now shown that, if the most efficient ϕ-encoded K code of a certain length has an odd number of boxes in it, the F code of the same length is just as efficient, but if the ϕ-encoded K code has an even number of boxes, the F code can carry one less binary input digit.

We also saw that between $n = 8a^2 + 2a + 1$ and $n = 8a^2 + 1$, the most efficient K code had an even number of boxes; between $8a^2 + 1$ and $8a^2 - 2u + 1$, an odd number. This indicates that for half of all possible cases the F code of a certain length is as efficient as the corresponding K code; for the other half it is less efficient.

We shall now compare the F codes with the E_o and E_E codes. Since we have already shown (in section 6) a close affinity between the E_o codes and the K codes, and the reactions of ϕ-encoding on them and on the E_E codes, we have a good idea of what to expect from this comparison. There are no unexpected results.

7.7 Comparison of the F codes and ϕ-encoded E codes

In section 6.18 we showed that under ϕ-encoding an E_E' code is at least as efficient as the E_o code of the same length. So we will only compare the F codes and the E_E codes.

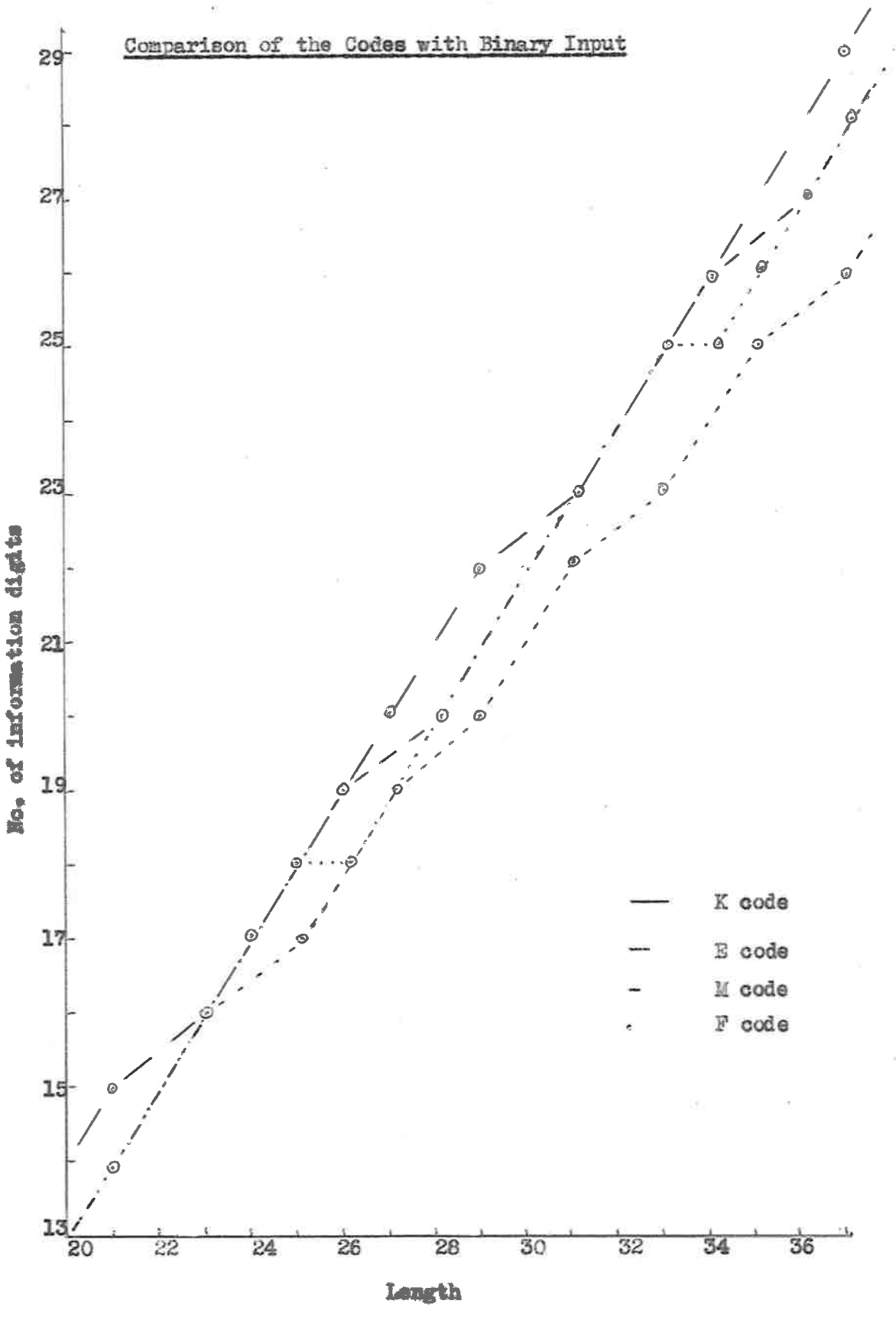
If we choose $s = 4a$ and $u = a$ for some integer a, we arrive at an E_E code of length $8a^2 + 2$ which can carry $8a^2 - 4a + 2$ binary input digits, and we know that an F code of that length requires $4a + 1$ fixed positions, and so can only carry $8a^2 - 4a + 1$ binary input digits. But for $n = 8a^2$, we remove two arbitrary

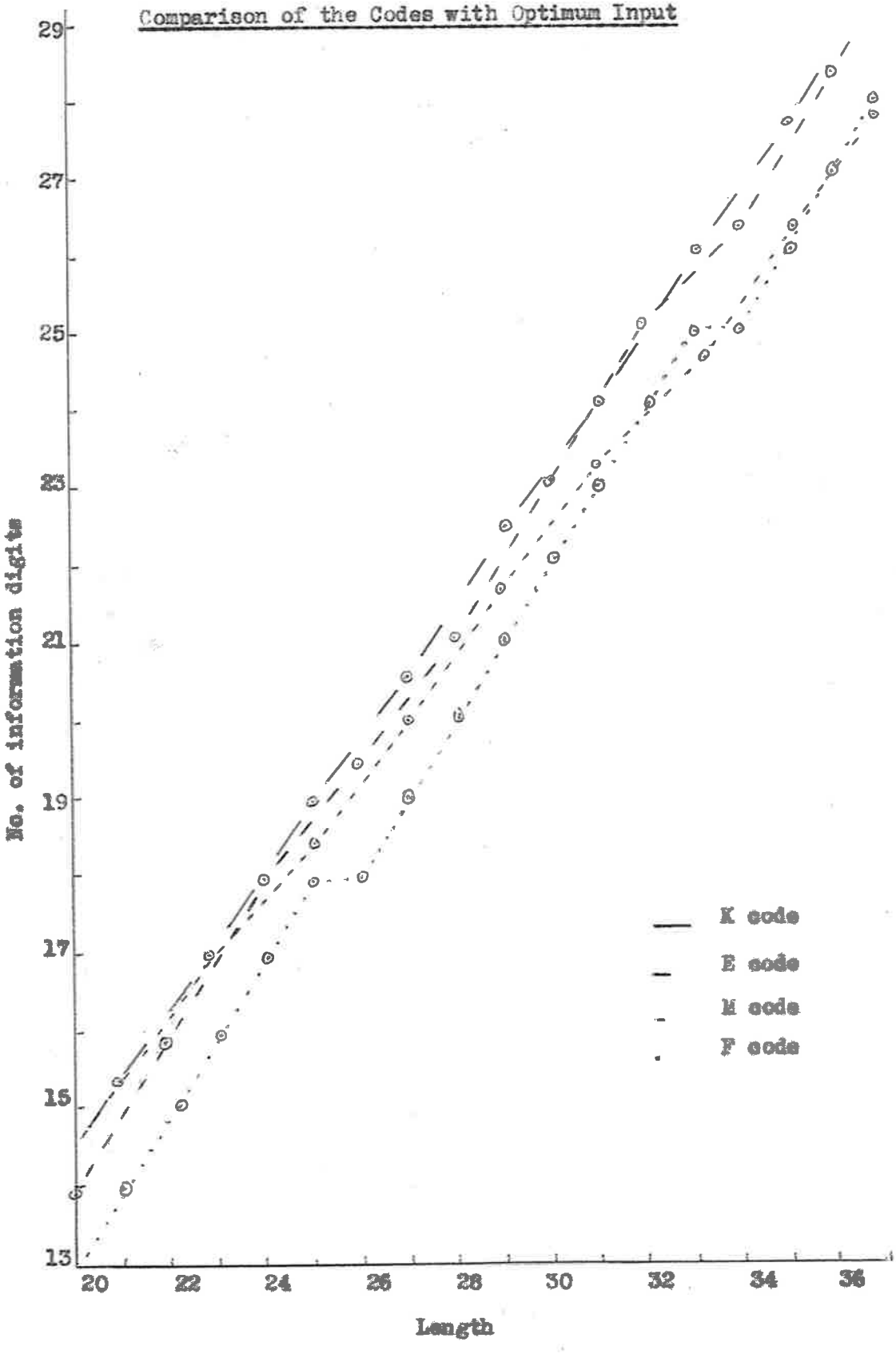
digits from our original E_E code, and the E_E' code we obtain can carry only $8a^2 - 4a$ binary input digits. But we showed before that an F code of this length only requires 4a fixed positions, and so it too can carry $8a^2 - 4a$ binary input digits.

We have now shown that for synchronous codes of even length, the F code is as efficient as the most efficient $\emptyset$-encoded 'herald and boxes' code except in a few isolated cases.

We can sum up the results of the past two sections as follows: if $\emptyset$-encoding is necessary with a 'herald and boxes' code, it is preferable to use an F code unless efficiency is important, and then, if the length required is odd and the most efficient K code of this length has an even number of boxes, it may be preferable to use a K code.

The following graphs show these properties clearly. The first one compares the F code with the $\emptyset$-encoded 'herald and boxes' codes; the second compares the F codes with the 'herald and boxes' codes in their most efficient state. The effect of $\emptyset$-encoding on the latter codes is well illustrated.


Comparison of the Codes with Binary Input
No. of information digits
Length
13
15
17
19
21
23
25
27
29
20
22
24
26
28
30
32
34
36
K code
E code
M code
F code


Comparison of the Codes with Optimum Input
No. of information digits
Length
29
27
25
23
21
19
17
15
13
20
22
24
26
28
30
32
34
36
K code
E code
M code
F code

7.8 Notes on the Graphs

The first thing we must note is that the position of the F code is exactly the same on both graphs. Once this is recognised, the difference ϕ-encoding makes to the efficiency of the 'herald and boxes' codes is immediately apparent.

If however, ϕ-encoding must be used to implement 'herald and boxes' codes, their efficiency advantages disappear. Over a large part of the range chosen (and over any range) the F code can be seen to be the equal in efficiency of any of the 'herald and boxes' codes, and, if this is so, there is no need to complicate our hardware with a ϕ-encoder; we use the F code of the length required.

Later, when we compare all these codes with the code bound derived in section 8, we shall see examples of what happens for large n. But the results we have collected here on these graphs are representative of all the spectrum of n values; anyway, we can see this from the theoretical results already obtained.

7.9 Summary

In this section we have constructed a class of fixed place synchronous codes which are in many cases as efficient in handling binary input dataas the more sophisticated 'herald and boxes' codes, although they are theoretically not as efficient. We have also shown that the F codes are as efficient as any fixed place synchronous code can be.

8. A NEW UPPER BOUND ON THE EFFICIENCY OF SYNCHRONOUS CODES

8.1 Introduction

So far we have discussed several new classes of synchronous codes, the M codes, K codes, E codes and F codes. Each are useful in particular cases; the M codes are the most efficient synchronous codes for small n, the K codes and E codes are the most efficient 'herald and boxes' synchronous codes of odd and even lengths respectively, and the F codes are the most efficient fixed place synchronous codes possible. But one question remains unanswered: how efficient is the most efficient synchronous code of a given length?

We know that we can disregard all but the K and E codes in the discussion, since we have proved, and verified from the graphs, that the F codes are not as efficient as the former two classes for given n. This proves that the most efficient synchronous code possible will not be a fixed place code, but will rely on the 'herald and boxes' principle to some extent.

We shall show the 'two digits to a box' system is more efficient than the 'three digits to a box' system with more complicated boxes; so we shall assume that somewhere among the possibilities of modification of the 'herald and boxes' theme lies the class of most efficient synchronous codes.

In 1958. Golomb, Welch and Gordon published their paper on comma-free codes. Comma-free codes have the synchronisation property that no false words are code words. However, they do not appear important in the field of information transmission, since the digits are interrelated, and so cannot accept arbitrary input.

The important thing about the comma-free codes they constructed is that they proved for odd n up to 15, and possibly beyond, the comma-free codes reached the upper bound given in section 2 on the number of possible words in a self-synchronising code. Jiggs in 1963 verified this result for n = 17. But comma-free codes do not submit to the restriction on synchronous codes that some digits must be sufficiently arbitrary to carry information, and it appears that this additional restriction means that the above bound is not a least upper bound for synchronous codes.

We shall construct a bound for generalised 'herald and boxes' codes which, if there is no other more efficient way of constructing synchronous codes, will be an upper bound for all synchronous codes. We shall then show that this bound is realistic in that it can be attained for all n up to 93, but we shall also show that in some cases it is unattainable and, in fact, for values of n greater than 93, it may never be attainable. But even then it is a great deal closer to an actual value than the bound given in section 2.

8.2 The generalised 'herald and boxes' code

We have shown, in our proofs that the K codes and F codes are the most efficient 'herald and boxes' codes possible, that we have in these two cases stretched to the limit the number of starting positions which can be eliminated because of the fact that the false word has 01 where a code word needs a box, or vice versa; i.e. if we have u 01 pairs and s boxes, we can eliminate 2us starting positions, and no more, because of the interactions between them. The only other feature of the code which could be adjusted to improve synchronisation is the herald.

In the herald of a code word are u fixed 0's and u fixed 1's. We have seen, from our proof that the F code is synchronous, that if these were placed correctly throughout the word, we could eliminate $2u^2$ starting positions. Since the F code does not provide for boxes of two digits, even in the generalised form outlined below, we cannot expect to obtain $2u^2$ eliminated starting positions from our herald; but we could expect some better return than the 2u eliminated positions we have obtained in the past.

The problem then becomes; how can we arrange the 01 pairs so that we obtain the best return in eliminated starting positions.

This is an easy question to answer by itself, although when combined with other factors the solution becomes more complex. If we could merely ensure that no two digits in the 01 pairs interacted with other digits for the same starting position, we would obtain the maximum number $2u^2$ (given above), of eliminated positions. But since we must retain the box character of the 01 pairs, in every case there must be u interactions for a false word beginning in positions 2 and n of a code word, for in the former case all the leading zeroes of the 01 pairs of the false word interact with the units directly following the zeroes in the code word; in the latter case the units of the false word interact with the zeroes of the code word.

However, apart from these two positions, it is possible to position the boxes in such a way that no two positions of a code word interact with two positions of a false word for any value of n. For example, consider the synchronous code of length 7 with 01 pairs beginning in the first and fourth positions, and the positions where a false word could start.

Two code words		0	1	x	0	1	x	x	0	1	x	0	1	x	x
	2		0'	1	x	0'	1	x	x						
False word	3			0	1'	x	0	1	x	x					
starting	4				0	1	x	0	1'	x	x				
positions	5					0'	1	x	0	1	x	x			
	6						0	1	x	0'	1	x	x		
	7							0	1'	x	0	1'	x	x	

The tags indicate positions where the false word interacts with the code words.

Except for positions 2 and 7, which we have already explained we can do nothing about, there is only one interaction between the code word and the false word for every starting position. This is the condition we must aim for in positioning our 01 pairs.

The above is not a good example of the type of code we hope to obtain, since it contains only fixed places and no boxes. But use of such a good example would have made the code word too long.

Of course, there will be more than one way of positioning the 01 pairs in a particular case; for example, in the above case, fixing the 01 pairs as beginning in the first and fifth positions would have accomplished the same result. And, since placing becomes important in the more complex cases when we have to handle boxes as well, the above is not a complete answer to the problem. But, anyway, we now have our limit; we know that with u 01 pairs correctly placed we can eliminate $2(u^2 - u + 1)$ starting positions (we lose $(u - 1)$ in both the second and nth positions).

There is one other generalisation we should mention, although so far it has not helped to enhance the efficiency of any

synchronous code. We have assumed that the two digits in a box or 01 pair must be next to each other. No reason has been given for this; it just seemed the logical thing to do. But none of our arguments would have altered if the digits in the boxes and 01 pairs were two, or three, or p digits apart, so long as each 01 pair and box had the same distance between its digits. And so we must consider this aspect also before we decide that a synchronous code of a certain length and efficiency cannot be constructed.

We should note here that this technique of splitting the boxes cannot affect the number of starting positions eliminated by the herald, since there will still be u interactions with a false word starting in the second of the paired positions and the complementary position.

We are now in a position to derive the maximum bound. The derivation is closely allied to those used in reference to the K and E codes, and so we will give here only the bare bones of the argument.

8.3. Theorem

A synchronous code with u 01 pairs and s boxes cannot have a length greater than

$$2u\,(u + s - 1) + 3$$

Proof:

Interactions between the boxes and the 01 pairs eliminate no more than 2us starting positions.

Interactions between the digits of the 01 pairs eliminate no more than $2(u^2 - u + 1)$ starting positions.

So there are no more than $2(u^2 + us - u + 1)$ starting positions eliminated altogether.

So the length is at most $2(u^2 + us - u + 1) + 1$.

A K code of this length would require u 01 pairs and $(s + u - 1)$ boxes. So if we can construct a code which attains this bound, we can convert $(u - 1)$ of the boxes in the K code into arbitrary digits, and this gives us four arbitrary digits instead of the three binary input digits the K code could carry in the boxes under ϕ-encoding.

For small u, this is not very important; for large u, where it would be important, we may not be able to construct the codes. But we shall show first that in the particular cases when $u = 2$ and $u = 3$ that codes can be constructed, and also show that when $u = 4$ and $s = 1$ a code of the maximum length given by the formula can not be constructed.

This latter result seems to indicate that the bound we are seeking may not be a least upper bound for $u \geq 4$.

But first we will show that when u = 2 and u = 3, codes can be constructed which attain this bound. For the first part of this demonstration, we shall prove the following theorem.

8.4 Theorem

A code of length 4s + 7, with 01 pairs beginning in the first and (2s + 4)th positions, and boxes beginning in positions 3, 5, ... , 2s + 1, is synchronous.

Proof:

No false word can begin in the positions numbered below for the reasons given.

(1) 2 and 4s + 7; both 01 pairs interact between the false word and the code word.

(2) 3, 5, ... 2s + 1; the first 01 pair of the false word interacts with a box of the code word.

(3) 4, 6, ... 2s + 2; one of the boxes of the false word interacts with the second 01 pair.

(4) 2s + 3, 2s + 5; the first 01 pair of the false word interacts with the second 01 pair of the code word.

(5) 2s + 4, 2s + 6; the second 01 pair of the false word interacts with the first 01 pair of the next code word.

(6) 2s + 7, 2s + 9, ..., 4s + 5; the second 01 pair of the false word interacts with a box of the next real word.

(7) 2s + 8, 2s + 10 , ... , 4s + 6; the boxes of the false word interact with the first 01 pair of the next real word.

This proves that the code is synchronous. But in theorem 8.3 we showed that 4s + 7 was the maximum length possible with only two 01 pairs. So for u = 2 we can construct codes which reach this upper bound.

We can obviously construct a synchronous code of length 4s + 5, 4s + 3 etc. by successively removing two of the arbitrary digits; but the fact that we can construct even length synchronous codes by this process of digit removal is not so evident.

In fact, it is not possible to construct a synchronous code of length 4s + 6 using only two 01 pairs. It is possible to construct a synchronous code by removing three digits from the last digit string, but removal of only one digit leaves a non-synchronous code; the second 01 pair of the code word and the first 01 pair of the false word correspond at the same time as the second 01 pair of the false word corresponds to the first 01 pair of the next code word, and so it would be possible for a false word identical with a code word to begin at the second 01 pair in a code word.

This is not a proof that it is impossible to construct a synchronous code of length 4s + 6; the proof follows similar lines to that given later in section 8.6.

This completes the discussion of the case u = 2. We shall now consider the case when u = 3; this case is remarkably similar to the case just discussed.

8.5 Theorem

A code of length $6s + 15$, with 01 pairs beginning in the positions 1, $2s + 4$ and $4s + 12$ and boxes beginning in the positions $3, 5, 7, \ldots, 2s + 1$, is synchronous.

Proof:

No false word can start in the positions indicated for the reasons given.

(1) 2 and $6s + 15$; all three 01 pairs interact between the false word and the code word.

(2) $3, 5, \ldots, 2s + 1$; the first 01 pair of the false word interacts with a box of the code word.

(3) $4, 6, \ldots, 2s + 2$; a box of the false word interacts with the second 01 pair of the real word.

(4) $2s + 3, 2s + 5$; the first 01 pair of the false word interacts with the second 01 pair of the real word.

(5) $2s + 4, 2s + 6$; the third 01 pair of the false word interacts with the first 01 pair of the next code word.

(6) $2s + 7, 2s + 9, \ldots, 4s + 5$; the third 01 pair of the false word interacts with a box of the next code word.

(7) $2s + 8, 2s + 10$; the third 01 pair of the code word interacts with the second 01 pair of the false word.

(8) $2s + 12, 2s + 14, \ldots, 4s + 10$; the third 01 pair of the code word interacts with a box of the false word.

(9) $4s + 7$, $4s + 9$; the third 01 pair of the false word interacts with the second 01 pair of the next code word.

(10) $4s + 11$, $4s + 13$; the third 01 pair of the code word interacts with the first 01 pair of the false word.

(11) $4s + 12$, $4s + 14$; the second 01 pair of the false word interacts with the first 01 pair of the next code word.

(12) $4s + 15$, ... , $6s + 13$; the second 01 pair of the false word interacts with a box of the next code word.

(13) $4s + 16$, ... , $6s + 14$; the first 01 pair of the next code word interacts with a box of the false word.

This proves that the code is synchronous.

Again, we can construct synchronous codes of odd lengths less than $6s + 15$ by removing pairs of arbitrary digits. And again, the code of length $6s + 14$ is impossible to construct with only 3 01 pairs. Also, it is impossible to construct a synchronous code of length $6s + 12$ with only 3 01 pairs, but the code of length $6s + 10$ can easily be constructed.

We seem to have found a pattern, which comprises the same arrangement of the first $2s + 5$ digits and the other 01 pairs spread throughout the word. But this pattern does not work for $u = 4$;

in fact no pattern works for $u = 4$ and $s = 1$, as the following theorem shows. And so once again we find ourselves in the position of not knowing whether any code we construct is the most efficient possible, or whether there is another more efficient code, the result of a more judicious arrangement of the 01 pairs and the boxes.

From the formula given in theorem 8.3, with $u = 4$ and $s = 1$ we could hope to construct a code of length $2u(u + s - 1) + 3 = 35$. We shall show in the next theorem that such a code cannot be constructed, at least with 01 pairs and boxes which have adjacent digits. It is unfortunate that the method of proof is so tedious; so many cases have to be considered that proof of the generalised theorem, with a certain number of positions between the digits of each 01 pair and each box, will not be attempted here; however, every attempt to construct a synchronous code of length 35 with only 4 01 pairs and one box has failed; in this search many different separations have been tried. But there is no reason why a different separation than the one used in the proof would enable such a code to be constructed; in fact, the experimental evidence indicates that a similar proof could be applied for any separation, for the results bear a close resemblance to the results obtained when trying to construct the code with adjacent digits in 01 pairs and boxes.

The proof is also instructive for another reason; it shows that there is an element which affects the feasibility of constructing synchronous codes which does not submit to the rationalisation we have used to obtain our upper bound; indeed, the proof indicates that we are expecting too fine an accuracy if we assume that we can spread our position eliminations as thinly as possible in all synchronous code constructions; since we achieved this maximum spread with the K and E_0 codes, we have tended to assume that we can do it with all similar codes, although our experience with the E_E code could have been taken as a warning.

8.6 Theorem

If the digits in the 01 pairs and in the boxes are adjacent it is impossible to position 4 01 pairs and one box in such a way as to make a code of length 35(= $2us + 2(u^2 - u + 1) + 1$) be synchronous.

Proof:

Let us suppose, without loss of generality, that the box begins the code words; we can do this, since any cyclic permutation of a synchronous code is itself synchronous.

Suppose the 01 pairs begin in the positions a, b, c, d, where

$$2 < a < b < c < d < 35$$

The positions where false words cannot begin are the positions as follows:

(i) a, b, c, d, where positions in which the false word, to be identical with a code word, needs a box, coincide with a 01 pair of the code word.

(ii) 37-a, 37-b, 37-c, 37-d, where positions in which the false word needs a 01 pair coincide with the box of the next code word.

(iii) 2, b-a, b-a+2, c-a, c-a+2, d-a, d-a+2,
c-b, c-b+2, d-b, d-b+2, d-c, d-c+2,
35, 37+a-b, 35+a-b, 37+a-c, 35+a-c, 37+a-d, 35+a-d,
37+b-c, 35+b-c, 37+b-d, 35+b-d, 37+c-d, 35+c-d,

where positions in which the false word needs a zero coincide with a fixed unit in the code word or vice versa.

All the above expressions are modulo 35. If a, b, c and d can be chosen so that the resulting code is synchronous, then the above expressions must represent every residue modulo 35 from 2 to 35 inclusive. Since in this case the residues are spread as thinly as possible, no two residues may be equal. This is our basic premiss. Let us consider which of the set could represent 3 (mod 35).

There are 5 possibilities

a, b - a, c - b, d - c, 37 - d

We shall consider these possibilities in turn.

Case 1. a = 3.

We shall consider which of the remainder could represent 4 (mod 35).

There are 4 possibilities

$b - a,\ c - b,\ d - c,\ 37 - d$

Again we shall consider these possibilities in turn.

Case 1.1 $b - a = 4$

We shall now consider which of the remainder could represent 5 (mod 35).

There are 3 possibilities

$c - b,\ d - c,\ 37 - d$

Again, we consider the possibilities in turn.

Case 1.1.1. $c - b = 5$

Using these equations, the set of residues is

3, 7, 12, d, 34, 30, 25, 37-d,

2, 4, 6, 9, 11, d-3, d-1, 5, 7, d-7, d-5, d-12, d-10

35, 33, 31, 28, 26, 40-d, 38-d, 32, 30, 44-d, 42-d, 49-d, 47-d.

The least of these expressions in d

$d = 12$ or $37 - d$, must be 8.

$\therefore$ $d = 20$ or $d = 29$

But then $40 - d = d$ or $38 - d = 9$.

Thus two residues are equal; this contradicts our premiss.

Case 1.1.2. $d - c = 5$

But then $d - b + 2 = c$, contrary to our premiss.

Case 1.1.3. $37 - d = 5$

But then $35 + a - d = b - a + 2$, contrary to our premiss.

Case 1.2. $c-b = 4$.

Again, we shall consider which of the remainder could represent 5 (mod 35).

There are 3 possibilities

$$b - a,\ d - c,\ 37 - d$$

Case 1.2.1. $b-a = 5$

The set of residues is

$$3,\ 8,\ 12,\ d\ ,\ 34,\ 29,\ 25,\ 37-d$$

$$2,\ 5,\ 7,\ 9,\ 11,\ d-3,\ d-1,\ 4,\ 6,\ d-8,\ d-b,\ d-12,\ d-10$$

$$35,\ 32,\ 30,\ 28,\ 26,\ 40-d,\ 38-d,\ 33,\ 31,\ 45-d,\ 43-d,\ 49-d,\ 47-d$$

The least of these expressions in d,

$d - 12$ or $37 - d$, must be 8

$d = 20$ or $d = 29$

Therefore, $d = 40 - d$ or $38 - d = 9$

Thus two of the residues are the same, contrary to our premise.

This latter section of the proof is identical with the same part of the proof in case 1.1.1.. If we proved every possible case, we would meet this duality often. So we shall only prove this case (case 1 : $a = 3$) in detail; the proof of the cases $c - b = 3$ and $d - c = 3$ are very similar to case 2; the proof of the case $37 - d = 3$ is similar to case 1.

Case 1.2.2. $d - c = 5$

The set of residues is

3, b, b+4, b+9 34, 37-b, 33-b, 28-b

2, b-3, b-1, b+1, b+3, b+6, b+8, 4, 6, 9, 11, 5, 7

35, 40-b, 38-b, 36-b, 34-b, 31-b, 29-b, 33, 31, 28, 26, 32, 30

The least of these expressions in b,

$b - 3$ or $28 - b$ must be 8

Therefore $b = 11$ or $b = 20$

Therefore, $b = d + 2$ or $40 - b = b$

Thus, again, two residues must be equal.

Case 1.2.3. $37 - d = 5$

But then $35 + a - d = c - b + 2$, contrary to our premiss.

Case 1.3. $d - c = 4$

The expressions which could represent 5 (modulo 35) are

$b - a$, $c - b$, $37 - d$

Case 1.3.1. $b - a = 5$

This case is similar to case 1.1.2.

Case 1.3.2. $c - b = 5$

This case is similar to case 1.2.2.

Case 1.3.3. $37 - d = 5$

In this case $d - c + 2 = 35 + a - d$.

Case 1.4. $37 - d = 4$

There are three expressions which could represent 5

$b - a$, $c - b$, $d - c$

Case 1.4.1. $b - a = 5$.

This is similar to case 1.1.3.

Case 1.4.2. $c - b = 5$

This is similar to case 1.2.3.

Case 1.4.3.

This is similar to case 1.3.3.

We have now proved that it is impossible to have a synchronous code of length 35, with 4 01 pairs and one box beginning in the first position, which has a 01 pair beginning directly after the box in the third position. The case when $37 - d = 3$ is very similar, since this represents a code with a 01 pair directly before the box, at least in the message sequence, in positions 34 and 35.

The only other expressions which could be equal to 3 (mod 35) are $b - a$, $c - b$ and $d - c$. These three cases are very similar, and so we shall only expound the proof of the first one

Case 2. $b - a = 3$

There are 4 possible ways to represent 4 (mod 35)

a, $c - b$, $d - c$, $37 - d$

Case 2.1. $a = 4$

If $b - a = 3$, $b - a + 2 = 5$. We must then consider the expressions which could equal 6.

They are $c - b$, $d - c$, $37 - d$.

Case 2.1.1. $c - b = 6$

The set of residues is

4, 7, 13, d 33, 30, 24, 37-d

2, 3, 5, 9, 11, d-4, d-2, 6, 8, d-7, d-5, d-13, d-11

35, 34, 32, 38, 26, 41-d, 39-d, 31, 29, 44-d, 42-d, 50-d, 48-d

The least of these expressions in d

$d - 13$ or $37 - d$, must be 10

Therefore, $d = 23$ or $d = 27$

Therefore, $d - 4 = 42 - d$ or $d - 13 = 41 - d$

This contradicts our premise.

Case 2.1.2. $d - c = 6$

The set of residues is

4, 7, c, c + 6 33, 30, 37-c, 31-c

2, 3, 5, c-4, c-2, c+2, c+4, c-7, c-5, c-1, c+1, 6, 8

35, 34, 32, 41-c, 39-c, 35-c, 33-c, 44-c, 42-c, 38-c, 36-c, 31, 29

The least of these expressions in c

$c - 7$ or $31 - c$, must be 9

Therefore, $c = 16$ or $c = 22$

Therefore, $c + 4 = 36 - c$ or $c - 7 = 37 - c$,

contrary to our premise.

Case 2.1.3. $37 - d = 6$

The set of residues is

4, 7, c, 31 33, 30, 37-c, 6

2, 3, 5, c-4, c-2, 27, 29, c-7, c-5, 24, 26, 31-c, 33-c

35, 34, 32, 41-c, 39-c, 10, 8, 44-c, 42-c, 13, 11, c+6, c+4

The least of these expressions in c,

$c - 7$ or $31 - c$, must be 9

Therefore, $c = 16$ or $c = 22$

Therefore, $c - 5 = 11$ or $c - 7 = 37 - c$,

contrary to our premiss.

Case 2.2. $c - b = 4$

In this case $c - b + 2 = 6$. We know also $c - a = 7$.

We have to find expressions which could equal 8.

There are 3 of these

a, d - c, 37 - d

Case 2.2.1. $a = 8$

The residues are

8, 11, 15, d 29, 26, 22, 37-d

2, 3, 5, 7, 9, d-8, d-6, 4, 6, d-11, d-a, d-15, d-13

35, 34, 30, 28, 45-d, 43-d, 33, 31, 46-d, 44-d, 52-d, 50-d

The least of these expressions in d

$d - 15$ or $37 - d$ must be 10

Therefore, $d = 25$ or $d = 27$

$44 - d = d - 6$ or $d - 8 = 46 - d$

contrary to our premiss.

Case 2.2.2. $d - c = 8$

The residues are

a, a+3, a+7, a+15, 37-a, 34-a, 30-a, 22-a

2, 3, 5, 7, 9, 15, 17, 12, 14, 4, 6, 8, 10

35, 34, 32, 30, 28, 22, 20, 25, 23, 33, 31, 29, 27.

The least of the expressions in a,

a or 22-a, must be 11

Therefore a = 11

Therefore, a + 3 = 14, contrary to our premiss.

Case 2.2.3. $37 - d = 8$

The case is similar to case 2.2.1.

Case 2.3. $d - c = 4$

The case is similar to case 2.2.

Case 2.4. $37 - d = 4$

The case is similar to case 2.1

8.6.1. <u>The Definite Upper Bound</u>

We now know that we can construct a synchronous code of a certain length which attains the upper bound on its efficiency if the most efficient code of that length has two or three 01 pairs for its generalised 'herald'. If the most efficient code requires four or more 01 pairs, we do not know whether we can construct it or not.

So the first value of n for which our upper bound may not be the least upper bound is the lowest value of n for which a generalised 'herald and boxes' code with three 01 pairs would be less efficient than one with four 01 pairs, if the latter could be constructed. This value of n is 94.

Consider n = 93 and n = 94.

(i) n = 93

(a) u = 3, s = 13.

There are 61 arbitrary digits and 13 boxes in this code.

Total information-carrying capacity

= 81.6 'arbitrary digits'.

(b) u = 4, s = 9

There are 67 arbitrary digits and 9 boxes in this code.

Total information carrying capacity

= 81.3 'arbitrary digits'

(ii) n = 94

(a) u = 3, s = 14

The 60 arbitrary digits and 14 boxes could carry

82.2 'arbitrary digits'.

(b) u = 4, s = 9

The 68 arbitrary digits and 9 boxes could carry

82.3 'arbitrary digits'.

For values less than 94, the code with 3 01 pairs is more efficient; for n greater than this, the codes with 4 01 pairs would

be the more efficient and would attain the upper bound on their efficiency if they could be constructed; but if they cannot be constructed, our upper bound is no longer a least upper bound.

8.7 Other Types of Synchronous Codes

We have investigated both the 'herald and boxes' codes, typified by the M, K and E codes, and the fixed place codes, Gilbert's codes and the F codes, in considerable detail. The reason we have given for this is that these codes are the easiest to implement in a practical application; in fact there does not seem to be any other type of synchronous code which could be used for information transmission.

However, we have made reference to the fact that no more efficient class of synchronous codes can be constructed; we have assumed in making this statement that there is no method of constructing synchronous codes other than one of the methods which use interactions between fixed positions or restricted strings of positions (e.g. boxes) to achieve synchronisation. We have also stated that longer strings of digits do not provide greater efficiency.

Intuitively, there are arguments for and against this latter statement; we can either argue that, since the K code, which uses boxes, is more efficient than the F code, which uses fixed

places, 'boxes' of 3 or more digits will provide greater efficiency; or we can say that the greater clumsiness of larger boxes will reduce the efficiency of the resultant code. We will show that the latter argument is correct.

This ties in with the result of theorem 8.6.. To construct such a code of length 35, we would not choose u = 4 and s = 1; in fact choosing u = 2 and s = 7 gives a code which reaches the upper bound. The main reason that we chose such an unlikely pair, apart from the fact that it was the simplest case with u = 4, was to illustrate the fact of uncertainty about the upper bound being a least upper bound; in fact, it appears that because of the constructional difficulties of long synchronous codes of this type which first become apparent in this proof, the upper bound may still be lowered further. But this is all premature; we have not yet shown that it is an upper bound for all methods of construction.

We will now consider the problem of constructing synchronous codes using other methods than the fixed place method and the 'herald and boxes' method. The only method which appears to have any possibilities of usefulness in a practical application is a method which uses box-p's, which we shall define to be sets of p digits, $p > 2$, in the same way as a 'herald and boxes' code uses boxes containing two digits.

We shall not consider hybrid cases, where a code uses, say, boxes and box-3's, to achieve synchronisation. Such a hybridisation was used out of necessity when we had to ϕ-encode binary data and required an odd number of boxes; at this time we fixed the first unit of the odd box as a unit, and the resultant code became a cross between a fixed place code and a 'herald and boxes' code, although we did not refer to it as such. But, just as in this case the unit placed an unnecessary restriction on the first position of the box, so would a box place an unnecessary restriction on the first two places of a box-3. So it is reasonable to assume that if we can find a synchronous code which uses box-p's and is more efficient than the generalised 'herald and boxes' code, it will use only one kind of box-p's.

Even if such a code could be constructed, it would have a drawback. With the ϕ-transformation, we had only to encode sets of 3 binary digits into 8 box pairs; thus we needed a 'dictionary' of 8 entries. But if a code using box-3's achieved high efficiency, we would still have to use one triplet as a herald, as we used 01 in the 'herald and boxes' codes; this triplet could not then be used in the positions corresponding to boxes. So we would have at most seven triplets available for use in the body of the code word.

8.7.1 Example

Suppose we had seven possible triplets into which to encode our information. We could set up a 1-1 transformation of

two arbitrary digits into the triplet set

e.g. $00 \rightarrow 000$,

but this is highly inefficient, since the maximum efficiency of a code using this transformation is 2/3.

We could encode 5 digits into the set of box-3 pairs, and this gives a maximum efficiency of 0.83, which is better than the maximum efficiency of the M code; but now we need a 'dictionary' with 32 entries. And, with the longer herald, this improvement is not attained for moderately small values of n; for example, the M code of length 97 can carry 142 arbitrary information digits; if we could construct a box-3 code of the type outlined above of the same length, it could carry no more than 155, and probably less.

8.7.2 The Use of p-tuples in Boxes

The reason that the above code may not carry 155 'arbitrary digits' is that although there must be at least one triplet, the herald triplet, that we cannot use in the body of the code word, there may be others which we have to omit to ensure that the code is synchronous.

As an example, consider an M type box-3 code which uses 001 for a herald; and suppose that its length is equal to 1 (mod 3). With the M code, the fact that 01 was not used as an information box meant that no false word identical with a code word could start in an odd-numbered position modulo 2, and the odd length sufficed to

place the 01 herald of the next real word in a box position of any false word beginning in an even numbered position. With a box-3 code, no false word can start in positions equal to 1 (mod 3) or 2 (mod 3) for the same reasons given above; but false words can certainly start in positions equal to 0 (mod 3) as in the following examples:

Suppose 001 $b_1b_2b_3b_4b_5b_6a_n$ is a word in a box-3 code, where the b_i's represent digits in box-3's, any digit triplet except 001, and a_n represents an arbitrary digit. A message string appears as

$$001\ b_1b_2b_3b_4b_5b_6a_n\ 001\ \ldots$$

No false word identical with a code word can begin with b_1, for $b_1b_2b_3$ is not 001, or at b_2, for then the next 001 would appear as a box-3 in the false word. But a false word beginning at b_3 could certainly be identical with a code word.

So, with only 001 for a herald, we would have to prevent $b_3b_4b_5$ from being 001. We could do this in two ways; we could fix b_3 as 1, which would lose us 4 of our so far acceptable triplets, or ensure b_4b_5 was not 01, which would mean that we could not use 010 or 011 in the body of a code word. In this latter case, with only 5 acceptable triplets (001, 010 and 011 being unavailable), we could not transform 5 arbitrary digits into the set of available box-3 pairs; thus the efficiency of a code constructed in the preceding example would be even less.

There is a more efficient method of accomplishing synchronisation in this particular case. If we construct a similar code with herald, 0010, we would be able to use 011 in positions equivalent to the box-5's in the previous example, and would only have to prohibit 001 and 010 from box-3's. But we still have only six of the eight triplets available to carry information; this is no better than the M code, which has three of the four boxes available to carry information; and any advantage in the efficiency of the hypothetical code is more than compensated by the simpler 'dictionary' we can use with the M code.

There is yet another method of achieving synchronisation in a box-3 code if it is rather longer than in our previous example. We can construct a semi-generalised box-3 code of which all words of the following type are members

001 a_4 001 $b_1b_2b_3$ ... $b_4b_5b_6b_7b_8b_9b_{10}b_{11}b_{12}$ a_{n-1} a_n

where again b_{3i+1} b_{3i+2} b_{3i+3} represents a box-3 not 001, and a_i represents an arbitrary digit. In this code, no false word which begins in a position equal to 0 (mod 3) can be identical with a code word, since either the second 001 overlaps the next code word herald or it occupies a box-3 in the false word. In this latter case, we can use all seven triplets not 001, and so we can probably construct more efficient box-3 codes using this method than any other. But we have fixed six places instead of the three we originally intended to fix.

8.7.3 Summary

What we have said about box-3 codes applies equally to box-p codes for $p > 3$, except that the bigger 'building blocks' make the latter codes even more cumbersome. So box-p codes have the following properties;

(i) They have comparatively longer 'heralds' (or alternatively, less freedom in the choice of arbitrary boxes) than the M code.

(ii) If only binary input is available, then something like ϕ-encoding will be required; this involves a much larger 'dictionary' than the M code and its ϕ-transformation, although the size of such a 'dictionary' is not even remotely close to the size of the one we would have to use with a comma-free code, when the number of 'entries' would be the number of code words.

(iii) Transformation of the same kind as was used to generalise the M codes to the K codes is conceivable, but could scarcely be as easy to implement.

8.8 Definition of the Problem

However, the result that we want to develop requires that we consider all possibilities such as this. So we will assume we can construct a box-p code with the following properties:

(i) The fixed places in the code are made up of u identical p-tuples (corresponding to the 01 pairs used in the generalised 'herald and boxes' code) and certain other fixed 0's and 1's. These are positioned throughout the word in positions we shall not pinpoint.

(ii) There are s p-tuples of positions which cannot contain the p-tuple mentioned in (i); they may or may not be able to contain any other p-tuples.

(iii) There are also positions in the word which are completely arbitrary.

(iv) We will assume a fixed p-tuple occupies the first p positions and that it contains at least one unit and one zero. We shall first develop the maximum efficiency of such a code in a general way, and then refine our choice of variables until we obtain a class of codes which are the most efficient obtainable using the box-p methods described above. We shall then compare this class of codes with the generalised 'herald and boxes' codes.

8.9 The p-tuples we can use

Before we attempt our proof, we must analyse the way in which we shall construct a box-p code of high efficiency.

We know that it will be of a form analogous to that of a generalised 'herald and boxes' code, i.e. a code word will consist of 'herald' p-tuples and box-p's imbedded in arbitrary digits. We have to discover the p-tuples we cannot use in the box-p's; obviously we cannot use the 'herald' p-tuples — are there any others?

To answer this question, we refer to section 8.7.2., in which we showed that the most efficient way to construct a box-3

code analogous to the M code was to use two offset 'herald' p-tuples. However, this method will not work in a generalised box-3 code; even if we can position our 'herald' p-tuples in such a fashion and still retain maximum starting-position eliminating potential (and theorem 8.6 showed that this latter is not always possible, even if we do not attempt to place additional restrictions on our positioning), the arbitrary digits between box-p's, where a false word identical with a code word may start, do not restrict any starting positions; the restrictions are imposed only by 'herald' p-tuple interactions between themselves and with box-p's.

If we consider the K code, the boxes are positioned in such a way that no false word can start in an odd-numbered position without having a box in its 'herald'. This probability of predictability of positions eliminated by the code format is not possessed by the generalised 'herald and boxes' code, where expediency is the key to the placing of the restricted positions.

In connection with this, it is interesting to note that there are cyclic permutations of the codes given in sections 8.4 and 8.5 which have all their 01 pairs beginning in odd-numbered positions. There may be a relationship between this property and the fact that we cannot construct the synchronous code of length 35 with $u = 4$ and $s = 1$.

As an example of the difference of M type and K type box-p codes, consider the following examples of code words of the two types

$$001\ a_4\ 001\ b_1b_2b_3 \cdots b_4b_5b_6b_7b_8b_9b_{10}b_{11}b_{12}\ a_{n-1}\ a_n$$

$$001\ a_4\ 001\ b_1b_2b_3 \cdots b_4b_5b_6\ a_pa_qa_r b_{10}b_{11}b_{12}\ a_{n-1}\ a_n\ ,$$

where the b_i represent digits in box-p's, and the a_j represent arbitrary digits.

The first example is the one we used in section 8.7.2 to show that we could construct an M type box-3 code which uses all seven other 3-tuples in the box-3's. But in the K type code we get by making $b_7b_8b_9$ independent arbitrary digits, a false word which is identical with a code word can start at a_p.

We know from our experience with the M code that an M type box-3 code will not in general be as efficient as a K type box-3 code. So there will be many arbitrary digits between box-3's, and the problem illustrated by the above example will arise many times in the most efficient box-3 code.

With the K code, the boxes ensured that any false words without 01 in their herald began in an even-numbered position in a code word; it was this regularity that enabled us to prove so easily that a K code is synchronous. It is likely that in a K type box-3 code, the boxes will enforce a similar restriction on the starting positions of false words which have 'heralds' identical with that of the code words.

Ideally, then, we would expect that false heralds could only start in positions a (mod 3), where a was a fixed digit. This would involve eliminating 010 and 011 from the set of 3-tuples available for box-3's, as in section 6.7.2; in general, it would involve the elimination of a large percentage of the otherwise available box-p's.

It may be unnecessary to restrict starting positions of false words so strictly; but we must discard at least one p-tuple in any case, and in the final analysis the number we discard will probably be considerably greater than this, especially for large values of p.

We will show that, for any given value of p, even if we have to discard only one other p-tuple, the generalised 'herald and boxes' code of a particular length should be more efficient than any box-p code of that length; we will not consider particular values of n, but show that the upper bound on the efficiency of box-p codes is less than that on generalised 'herald and boxes' codes.

We will in turn find formulae for the maximum efficiencies of the generalised 'herald and boxes' code, and box-p codes for $p = 3$, $p = 4$, and $p = 5$ assuming in the last three cases that $2^p - 2$ p-tuples can be used in the box-p's. (This assumes that the 'herald' p-tuples and one other cannot be used). But first we need

the following lemma, which gives us a precise value of the number of positions which can be eliminated as starting points for false words identical with code words by interaction between 'herald' p-tuples.

8.10.1 Lemma

In a code with u 'herald' p-tuples, the maximum number of positions which can be eliminated as possible starting positions for false words identical with code words because of interaction between the 'herald' p-tuples is

$$2(p - 1)(u^2 - u + 1).$$

Proof:

If two 'herald' p-tuples begin in positions b_i and b_j respectively, the starting positions which can be eliminated are those positions between and including the position where b_{i+p-1} in the false word coincides with b_j in the code word, and the position where b_{j+p-1} in the code word coincides with b_i in the false word, excluding the position where b_i in the false word coincides with b_j in the code word.

e.g.

```
Code Word          . . . 0 0 1 . . .
                (    . . . 0 0 1 . . .
False           (      . . . 0 0 1 . . .
Words           (        . . . 0 0 1 . . .   Not eliminated
                (          . . . 0 0 1 . . .
                (            . . . 0 0 1 . . .
```

Thus each pair of p-tuples can eliminate at most 2(p - 1) starting positions, and there are u^2 pairs of p-tuples.

But if i = j, each of the u p-tuples is interacting with itself at the same time. This reduces the number of effective pairs of p-tuples by u - 1.

Q.E.D..

We shall now devise an approximation which simplifies our calculations in the next section.

8.10.2 Lemma

A close approximation to the theoretical number of digits lost to synchronisation by the s box-p's in a box-p code is

$$ps.2^{1-p}$$

Proof:

The number of ways that the digits can be arranged in a box-p is

$$(2^p - 2)$$

Arbitrary digits could be arranged in the positions in 2^p ways.

The theoretical fraction of a digit lost to synchronisation by each position in a box-p is then

$$p - \log_2(2^p - 2)$$

$$= \quad p - \log_2(2^p - 2)$$

$$= \quad p - \log_2 2^p(1 - 2^{1-p})$$

$$= \quad - \log_2(1 - 2^{1-p})$$

which, to a first approximation is

$$2^{1-p}$$

But there are altogether sp digits in box-p's.

Q.E.D.

We are now in a position to examine the following cases.

8.11 The Efficiencies of the box-p codes

8.11.1 The generalised 'herald and boxes' code has

$n = 2(u^2 - u + 1) + 2us + 1$ and

$L = 2u + \frac{1}{2}s$,

where L represents the theoretical number of 'arbitrary digits' lost to synchronisation.

We again approximate $\log_2 3$ by 1.5 at this stage, where it is accurate enough for our purposes.

The maximum efficiency occurs when

$$s = 2u + 1$$

and is

$$1 - (3/2n)^{\frac{1}{2}} + O(n^{-1}).$$

In this and the following sections we consider the maximum theoretical value of n as if a code could be constructed of this length.

8.11.2 The box-p codes

We showed in lemma 8.10.1 that interaction between the u 'herald' p-tuples can eliminate at most $2(p-1)(u^2-u+1)$ starting positions.

We know that interaction between u 'herald' p-tuples and s box-p's can eliminate at most 2us starting positions.

The maximum length of a box-p code is one greater than the sum of these,

$$\text{i.e. } n = 2(p-1)(u^2-u+1) + 2us + 1$$

for this, $L = up + ps.2^{1-p}$,

by lemma 8.10.2.

8.11.3 The box-3 code

$$n = 4(u^2-u+1) + 2us + 1$$

$$L = 3u + 3s/4$$

The maximum efficiency occurs when

$$s = 2$$

and is $1 - (9/4n)^{\frac{1}{2}} + O(n^{-1})$

8.11.4 The box-4 code

$$n = 6(u^2 - u + 1) + 2us + 1$$

$$L = 4u + \tfrac{1}{2}s$$

The maximum efficiency occurs when

$$s = 2u + 3$$

and is $1 - (5/2n)^{\frac{1}{2}} + O(n^{-1})$

8.11.5 The box-5 code

$$n = 8(u^2 - u + 1) + 2us + 1$$

$$L = 5u + 5s/16$$

The maximum efficiency occurs when

$$s = 8u - 4$$

and is $1 - (75/32n)^{\frac{1}{2}} + O(n^{-1})$.

Although this code appears more efficient than the box-4 code, we can reasonably assume that in practice we would have to discard more than one 5-tuple apart from the herald; a similar argument would apply to greater values of p. This comparison may appear pointless, since we have not shown that we can construct a generalised 'herald and boxes' code of a given length with given u and s; in fact, we showed that it was quite likely that for many values of n we could not construct a code as efficient as the upper bound would indicate.

However, for any value of n we can construct a K code, an E code or an M code. For a given length n, the generalised 'herald and boxes' code will be at least as efficient as the most efficient code from any of these classes.

We will only compare box-p codes with K codes, since E codes are closely related to K codes and for the values of n where the M code is more efficient than a K code, the box-p codes are inefficient. We can now prove the following theorem.

8.12 Theorem

The most efficient generalised 'herald and boxes' code for a given length n is the most efficient synchronous code of that length.

Proof:

The efficiency of a K code of a given length n is, by the formula in 5.11,

$$1 - (2/n)^{\frac{1}{2}} + O(n^{-1})$$

This code can be constructed, and it is much more efficient than any of the box-p codes given; these latter codes may or may not be capable of construction.

We also showed that a K code of a given length is more efficient than an F code of the same length.

Since the K code is a particular case of a 'herald and boxes' code, the generalised 'herald and boxes' code of a given length must be at least as efficient as the K code of that length.

8.12.1 If we use the actual value of $\log_2 3$ instead of the approximation 1.5, we find that the upper bound on the efficiency of a generalised 'herald and boxes' code is

$$1 - 1.15n^{-\frac{1}{2}} + O(n^{-1}).$$

8.13 The sign representation

In section 2.15, we showed that Golomb's comma-free code of a certain length consists of all the words which have certain sign representations. We shall show that our generalised 'herald and boxes' codes can be represented in similar fashion.

When we find the sign representation of a word in a generalised 'herald and boxes' code, in the same way as we found the sign representation of a word in a comma-free code, there is always a plus sign in the position in the sign sequence corresponding to the first position in a 01 pair in a code word, and we always have a minus sign in the position corresponding to the first position in a box. Apart from these restrictions, and the restriction that there cannot be two consecutive plus signs in a sign representation of a word in a binary code, the generalised 'herald and boxes' code of a certain length will be shown to consist of all words with permissible

sign representations.

e.g. Consider the M code of length n = 11. This is not a good example of a generalised 'herald and boxes' code, but it is the most efficient synchronous code of that length.

The sign representation of a word in the M code is

$$+ - - x - x - x - x ,$$

where the x's represent signs which are either plus or minus arbitrarily.

The second and last sign in the string must be minuses, since there cannot be two consecutive plus signs.

If we use the representation of the sign sequences by the lengths of the strings of minus signs, as in section 2.15, we find that the permissible sign sequences are represented by

10	2, 1, 5	2, 1, 1, 3
2, 7	2, 3, 3	2, 1, 3, 1
4, 5	2, 5, 1	2, 3, 1, 1
6, 3	4, 1, 3	4, 1, 1, 1
8, 1	4, 3, 1	2, 1, 1, 1, 1
	6, 1, 1	

If we compare these representations with those given in section 2.16, we find that each of the above representations is a cyclic permutation of one of the representations of the comma-free code. The representations in the latter set for which there is no counterpart in the M code are those with more than one even-numbered minus string.

Because of this, we can quickly see that the M code of length 11 must be synchronous, since the only even-numbered minus string in each representation is the first one. Thus any false code word beginning with 01 (i.e. whose representation begins with a plus sign) would have an odd-numbered first minus string.

Every M code has the same type of set of representations, i.e. every sign sequence has a first even-numbered string of minus signs, and no other even-numbered string. This is not so important when we consider n = 11, for there are only two representations in the comma-free code set with more than one even-numbered string; but for large values of n, there are many such representations, and so the M code represents a smaller and smaller percentage of the total number of cycles of order n.

If we consider the M code of length 13, we find that there are now ten representations in the comma-free code set given in 2.17 which have no counterpart. But we shall consider in detail the K code of length 13 which has two 01 pairs for a herald; although this is not as efficient as the M code of that length, it gives us a better idea of the comparison for large n.

A word in the K code of length 13 has a sign representation of

+ - + - - x x x - x x x x,

where the x's again represent signs arbitrarily plus or minus.

The permissible sign configurations are

1, 10	1, 2, 7	1, 2, 1, 5	1, 2, 1, 1, 3
	1, 3, 6	1, 2, 3, 3	1, 2, 1, 2, 2
	1, 4, 5	1, 2, 4, 2	1, 2, 1, 3, 1
	1, 6, 3	1, 2, 5, 1	1, 2, 3, 1, 1
	1, 7, 2	1, 3, 2, 3	1, 3, 2, 1, 1
	1, 8, 1	1, 3, 3, 2	1, 4, 1, 1, 1
		1, 3, 4, 1	1, 2, 1, 1, 1, 1
		1, 4, 1, 3	
		1, 4, 2, 2	
		1, 4, 3, 1	
		1, 6, 1, 1	

Again we can quickly show that the K code of length 13 is synchronous, since every representation begins with 1, followed by an even number or 3. The only possible false words which may have representations identical to those of code words would have a representation beginning with 1, 3, 1, which does not begin any representation in the set.

The K code, unlike the M code, can have representations with more than one even-numbered minus string; however, if we compare this set with the one given in 2.18, we find that the majority of the representations in the latter set without an equivalent in the K code are the ones with a small number of minus strings; for example, 1, 10 is the only one of the six in the first column with an equivalent in the K code. This is explained by the fact

that a word in the K code of length 13 with two 01 pairs for a herald has at least two plus signs in its representation, whereas a comma-free code may have only one. But, since for larger and larger n, a smaller and smaller percentage of the total number of cycles is represented by representations containing a few large minus strings, the efficiency of the K code is not impaired to the same degree as that of the M code.

With any generalised 'herald and boxes' code with more than one fixed 01 pair, we find that the necessity of at least two plus signs in every representation means that not many of the representations in the comma-free code with only a few minus strings are represented. This is another way of stating the reason why a generalised 'herald and boxes' code can never be as efficient as a comma-free code of the same length for $n > 7$.

However, we have shown that there is a marked correlation between the sign representations of the words in a generalised 'herald and boxes' code and a comma-free code, and that the codes are not as dissimilar as they appear on the surface.

9. SUMMARY

We have constructed many different types of synchronous codes. In particular, we have constructed codes of the 'herald and boxes' type which attain a high degree of efficiency. In fact, we have shown that unless there is a radically different method of constructing synchronous codes, the K codes (and the E_0 codes) are not far below the limit on efficiency we established in section 8; generalised 'herald and boxes' codes can be constructed which reach this bound for $n < 94$.

We shall use the following table to compare the main types of codes we have constructed for a range of values of n .

Efficiency

	n=9	n=15	n=17	n=37	n=85	n=201	n=393	n=1353	n=10369
Cycle Upper Bound	.648	.739	.760	.859	.925	.962	.978	.992	.999
Herald & Boxes Bound	.639	.701	.711	.808	.872	.919	.940	.967	.988
K code	.509	.651	.691	.802	.866	.911	.936	.965	.987
K code (ø-encoded)	.444	.600	.647	.784	.847	.900	.929	.962	.986
F code	.556	.600	.647	.757	.847	.900	.929	.962	.986
M code	.639	.701	.711	.755	.776	.786	.789	.791	.792
M code (ø-encoded)	.556	.667	.647	.703	.729	.741	.746	.749	.750
Gilbert's code	.333	.467	.529	.676	.776	.856	.898	.945	.980

BIBLIOGRAPHY

1. Fire, P. 'A Class of Multiple-Error-Correcting Binary Codes for Non-Independent Errors', Sylvania Report RSL-E-2, Sylvania Reconnaissance Systems Laboratory, Mountain View, California (1959).

2. Gilbert, E.N. and Moore, E.F.. 'Variable Length Binary Encoding', Bell System Tech. Journal, 38, July, 1959.

3. Gilbert, E.N.. 'Synchronisation of Binary Messages', I.R.E. Transactions on Information Theory, September, 1963.

4. Golomb, S.W., Gordon, B., and Welch, L.R.. 'Comma-Free Codes', Canadian Journal Math., 10 (1958).

5. Hamming, R.W.. 'Error Detecting and Correcting Codes', Bell System Tech. Journal, 29 (1950).

6. Huffman, D.A.. 'A Method for the Construction of Minimum Redundancy Codes', Proceedings I.R.E., 40, September, 1952.

7. Jiggs, B.H.. 'Recent Results in Comma-free Codes', Canadian Journal Math, 1963.

8. Muller, D.E.. 'Application of Boolean Algebra to Switching Circuit Design and to Error Detection', I.R.E. Transactions, EC-3 (1954).

9. Reed, I.S.. 'A Class of Multiple-Error-Correcting Codes and the Decoding Scheme', I.R.E. Transactions, PGIT-4 (1954).