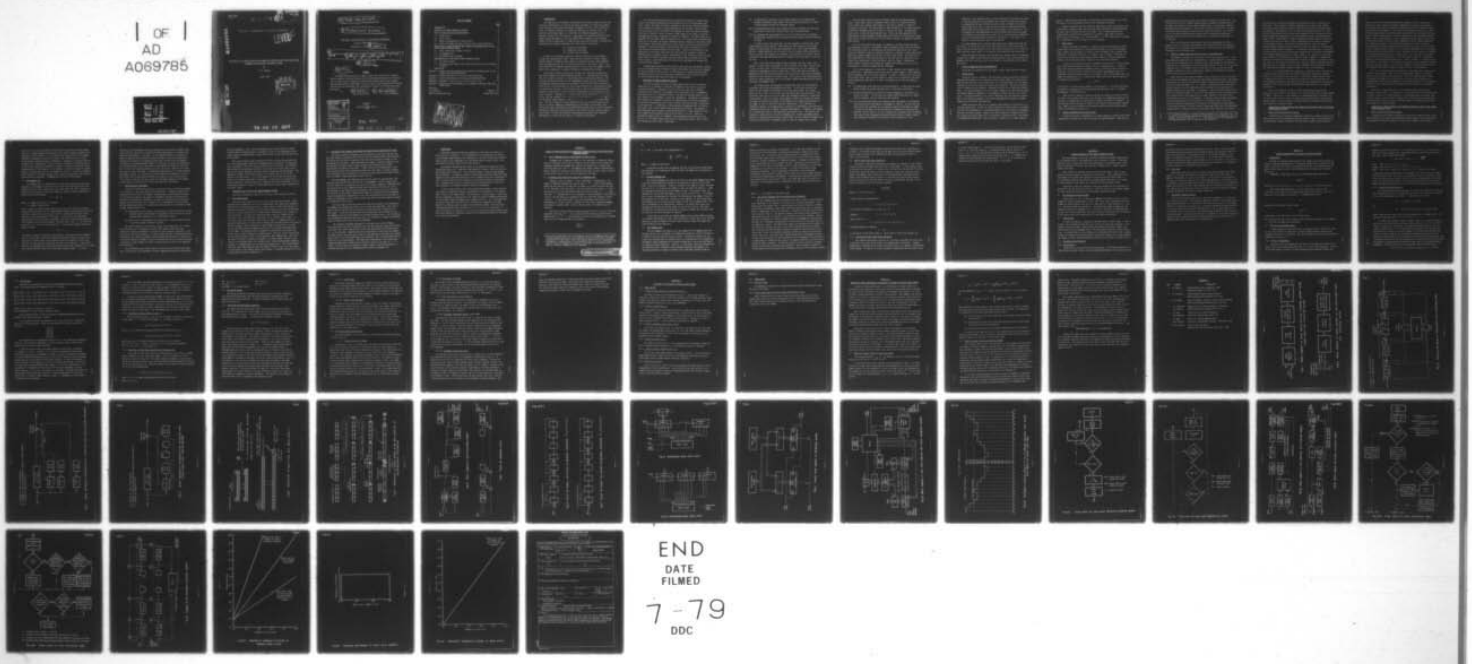
AD-A069 785 ROYAL AIRCRAFT ESTABLISHMENT FARNBOROUGH (ENGLAND) F/G 14/3
THE DESIGN OF AN ERROR DETECTION AND CORRECTION SYSTEM FOR USE --ETC(U)
APR 78 C W JACKSON

UNCLASSIFIED RAE-TM-IT-171 DRIC-BR-64334 NL

| OF |
AD
A069785



END
DATE
FILMED
7-79
DDC

TECH. MEMO
IT 171

TECH. MEMO
IT 171

ROYAL AIRCRAFT ESTABLISHMENT

THE DESIGN OF AN ERROR DETECTION AND CORRECTION SYSTEM FOR USE WITH SINGLE TRACK AIRBORNE DIGITAL MAGNETIC RECORDING SYSTEMS

by

C. W. Jackson

April 1978

(14) RAE-TM-IT-171

(9) Technical memo.

ROYAL AIRCRAFT ESTABLISHMENT

Technical Memorandum IT 171

Received for printing (11) 7 April ~~1~~ 1978

(6) THE DESIGN OF AN ERROR DETECTION AND CORRECTION SYSTEM FOR USE WITH SINGLE TRACK AIRBORNE DIGITAL MAGNETIC RECORDING SYSTEMS

by

(10) C. W. Jackson

(12) 59 p.

SUMMARY

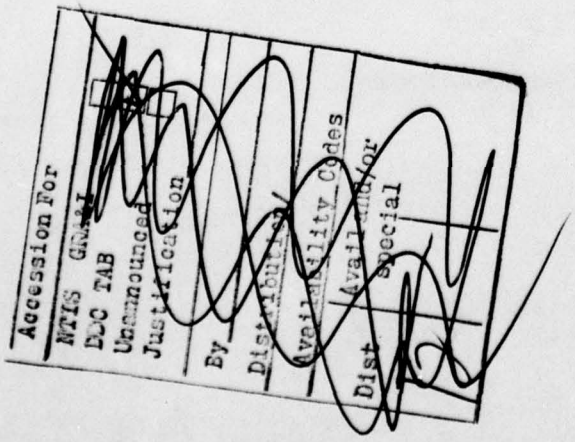
The Memorandum describes the types of error that can occur in airborne digital magnetic recording systems, and the problems associated with the correction of these errors on medium and high packing density single track systems. Some methods whereby record/replay errors on single track systems may be detected and corrected are explained, and a description is given of a system using these principles.

(18) DRIC (19) BR-64334

Accession For	
NTIS GRA&I	X
DDC TAB	
Unannounced	
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or special
A	

310 450

LIST OF CONTENTS



1 INTRODUCTION

The high costs of development and operational flying of modern aircraft has produced a growing need for accurate and rapid assessment of almost every aspect of an aircraft's design and performance, throughout its life. This inevitably leads to a demand to gather ever increasing quantities of airborne data. The ease with which large numbers of relatively low bandwidth parameters can be recorded, together with the need for a high degree of automation during the subsequent ground processing of the data, makes digital magnetic tape recording ideally suited to these types of airborne data acquisition systems. Airborne data recording can be broadly classified into three types of activities:-

(1) Flight test recording.
(2) Operational recording.
(3) Accident data recording.

Flight test recording encompasses aircraft certification, performance testing, and experimental flying. These activities generally require complex systems capable of handling large numbers of parameters giving a high combined input data sampling rate with a recording duration of a few hours. Reel to reel tape transports are normally employed using 14 or more tracks and making a record which can be replayed on a standard laboratory instrumentation recorder.

Operational recording systems and accident data recording systems have a much lower combined input data sampling rate but it is desirable in both cases that the recorders can work for long periods without requiring attention. The principal difference between the two requirements (apart from the considerable environmental hardening required by the ADR) is that the ADR may be allowed to re-write over old data whereas the entire tape produced by an operational recorder is required for analysis. The characteristics of the two recording systems are however so similar that in some aircraft, the ADR tapes are also used for operational analysis. Although half or quarter inch tape is often used in these applications and the recorders are invariably multitrack, efficient use of the tape in its multitrack format implies a very low tape speed, due to the low input data rate. The use of digital recording however allows the recording format on the tape to be varied completely independently of the format of the input, since digital data may be multiplexed and de-multiplexed easily with no loss of accuracy. Recording is therefore usually confined to one track at a time, using a single track format, to avoid the problems of maintaining acceptable tape transport performance at a very low tape speed in a vibration environment.

The required recording duration is obtained by writing the tracks sequentially, the recorder operating bi-directionally if not of the continuous loop type. Another advantage of digital recording is the greatly reduced dependence of the replay data quality on the performance of the recorder. When an error occurs, however, any of the digital bits making up a given word may be affected, thus even a single bit error can cause a large change in the value of the word encompassing the error, rendering it useless for analysis. For this reason it is desirable that erroneous words be identified so as to avoid a false interpretation being put on the data, and further advantage would be obtained if the errors could be corrected. Multitrack systems with error detection and correction have been designed[1,2] usually operating on the principle that only one track fails at a time and may be replaced using the other tracks. Clearly this technique is not applicable to the single track case, and errors must be corrected using data obtained from the erroneous track. Unfortunately, tape systems are prone to burst errors which, at high bit packing densities, may be several hundred bits long. Until recently, it was impractical to provide the large amount of storage such correction would require.

The Memorandum describes error detection and correction techniques, suitable for use with single track recording systems, and explains how they can be adapted for the correction of burst errors. A description of a system employing these techniques is given, and the theoretical performance of the system in terms of its error detection and correction capability is considered.

2 PRINCIPLES OF ERROR CORRECTING SYSTEMS

In order to provide a digital recording system with an error correction capability, it is necessary to increase the number of bits used to store the recorded information. The degree of error correction that can be achieved increases as the proportion of redundant bits is increased, but to accommodate these extra bits, it is necessary either to increase the packing density, or to use a greater area of medium to record the information. The degree of redundancy must therefore be optimised to provide the desired error correction performance, at the worst anticipated bit error rate, without inserting too high a proportion of redundant bits. In addition, a technique must be employed which is suited to the typical type of error distribution produced by the system. For example, a high redundancy system capable of working efficiently in a poor signal to noise ratio communications system, having an error rate of 1 in 10^2, would not necessarily function well in a magnetic tape recording system having an error rate of 1 in 10^6 caused almost exclusively by large burst errors. Error correction systems operate in one of three ways:-

(i) by replacing all the data in an erroneous region by the original data sequence occurring in that region, the data being reconstituted by using a coded sequence outside the erroneous region;

(ii) by determining the exact error sequence that has corrupted the data, and using this to invert the erroneous bits, thus correcting them;

(iii) by comparing the coded sequence on a bit by bit basis with all possible non-erroneous sequences and selecting the closest pattern, known as majority decoding.

To form an error correcting code, the data sequence is divided into fixed length sequences called blocks. These blocks are then encoded to form longer sequences incorporating the required redundancy bits. These longer sequences, which are termed codewords, are recorded in lieu of the data, and are decoded on replay to recover the original data sequence. The selection of an error correction method from those described above will depend on the code used in a particular case.

Error correcting codes themselves may be divided into four types. The simplest of these types conceptually is the systematic block code. In this type of code, the data block is recorded without modification, and the codeword is completed by appending a series of check bits. These check bits are formed by operating on the data block according to the rules of the code, and are therefore dependent on the contents of the data block. This class of error detecting and/or correcting codes encompasses most of the simpler coding techniques such as parity, repetition and Hamming codes.

The second group, the systematic convolutional codes are also formed by recording a data block and appending check bits. In this case, however, the check bits are formed by operating on two or more data blocks. As a result, there are two or more sets of check bits associated with each data block. Convolutional codes are often used to reduce the redundancy needed for a particular coding scheme, since, for a given total number of check bits required per block, the number of check bits required in each codeword is inversely proportional to the number of sets of check bits used by a given block. This reduction in redundancy, however, is accompanied by a reduction in performance, since errors in neighbouring blocks are no longer corrected independently. A commonly used example of a convolutional code is the Wyner-Ash[2] convolutional code. This may be considered as a convolutional form of the Hamming code. Unlike the Hamming code, however, it requires only one check bit per block, and thus combines a single error correction ability with a redundancy equal to that of a simple parity check.

The two other classes of error correcting codes are the non-systematic codes. These differ from the systematic codes in that the codewords recorded do not contain the original data blocks, *ie* the entire codeword is formed by operating on the data. For these types of codes therefore, it is always necessary to perform a decoding operation in order to retrieve the data. Non-systematic codes can be in block or convolutional form. These code types generally find applications in high redundancy systems where the incidence of single and/or small burst errors is high, and they are of only limited interest for magnetic recording applications.

For all of the above code types, the lengths of the data blocks and codewords used may vary over a considerable range, depending on the code selected and the constraints placed on the system by the particular application, but in all cases the codeword or group of codewords required to represent a single data block, must have a greater number of bits than the data block itself. This results in more combinations of bits being available in the code than are required to represent every combination of data bits which could be encoded. Combinations which actually represent data blocks are termed valid codewords, those not representing data blocks are termed invalid codewords.

For an error sequence to be detected, it must change a valid codeword into an invalid one. Undetectable errors occur when an error changes a valid codeword into a different, but still valid, codeword. In order to achieve error correction with a given block, only a certain number of errors can be tolerated in the codewords representing that block. If this is exceeded, one of three alternatives occurs:-

(i) An undetectable error may occur as described above. In this case the error passes through the system unchanged and is classified as good data.

(ii) An error pattern may occur which the system recognises as one which it is unable to correct. In this case the relevant data blocks would be classified as erroneous; not all codes have this property.

(iii) An error pattern may modify the codeword in the same manner as a different error pattern which the system is programmed to correct. In this case the wrong correction procedure will be applied, possibly increasing the number of errors in the data, and the data would be classified as corrected and therefore free of errors. It is not possible to distinguish data containing this type of error from genuine corrected errors unless a further stage of error detection is applied to the data to test the validity of the

correction. The resultant effect of the 'correction' is therefore the same as that occurring for an undetectable error except that the error pattern is changed and the data in which the error occurs is labelled 'corrected' rather than 'error free'. Such an error may thus be considered as being undetectable by the system without a second stage of error detection. For the purposes of this Memorandum, an error detection and correction system is considered to have detected an error only if it has correctly identified that error as being either one which it can correct, or one which it is unable to correct and must label as uncorrected.

As coding schemes will in general detect more error combinations than they can correct, it may be seen that if the above definition of error detection is used, a code will usually have a better error detection ability if it is used for error detection alone, than if it is used for error detection and correction.

The percentage of erroneous bits that may occur within a group of codewords representing a block of data, and still allow that data block to be corrected, is dependent upon the code used, but it will always be less than 50% and is usually much lower.

3 TYPES OF ERROR AND THEIR REPRESENTATION

Errors may be classified into two types - single errors and burst errors.

3.1 Single errors

Single bit errors in magnetic PCM recording systems are generally caused by temporal displacement of the transitions of the replay signal, *ie* timing jitter. This results from tape noise, thermal noise in the replay amplifiers, and speed instability of both the record and replay tape transports. The tape and thermal noise is random and thus produces a Gaussian distribution of transition displacements. The speed instability will have random components but may also have large non-random components dependent on the mechanical characteristics of the transports. The speed instability will also be dependent upon the vibration environment of the airborne transport, which may be predominately periodic or random depending on individual circumstances.

Speed variations alone are not normally large enough to cause errors in a working installation. Errors are usually caused by all these effects in combination and they approximate to a random distribution for the purposes of designing an error correction system. Timing jitter and consequential reading errors are also caused by pattern sensitivity, *ie* shifting of transitions due to the effect of neighbouring transitions. This may affect a single bit or a small number of

bits at a time, but the occurrence of errors due to this cause will be of a random nature if the data pattern itself can be considered as random.

Pattern sensitivity may also result in small burst errors (see section 3.2), particularly if channel codes such as Miller (Delay Modulation) and Enhanced NRZ are used, which are prone to base line wander. For a system producing only single errors, each bit has a probability of being in error equal to the bit error rate of the system, *ie* if the system BER is 1 in 10^6, then each bit recorded has a probability of 10^{-6} of being in error on replay.

3.2 Burst errors

A burst error may be defined as a region started and terminated by errors, within which, the probability of individual bits being in error is much higher than is normal for the system. For burst errors on magnetic tape systems, this probability is usually about 50%, *ie* a complete loss of signal. The length of burst errors usually has a high random content, subject to various system constraints, but the distribution of lengths is not necessarily Gaussian.

It can be seen that one possible arrangement only exists for a two bit burst error, this being two adjacent erroneous bits, *ie* a double error. A three bit burst error has two possible arrangements since its second bit may be correct or erroneous. A four bit burst error has four possible arrangements since its second and third bits may be correct or erroneous. Thus in the general case, the number of possibilities (p) of a particular length burst error may be related to its length in bits (n) by:-

$$p = 2^{(n-2)} .$$

A burst error correction system capable of correcting an n bit burst error must be capable of recognising and distinguishing between each of the p possible arrangements of the burst error.

Burst errors in magnetic recording systems are usually the result of loss of head to tape contact, often due to the presence of contaminating materials on the tape, or imperfections of the magnetic coating, resulting in 'dropouts'. In high packing density systems the length of such errors can reach several hundred bits. Small burst errors may also be caused by effects such as pattern sensitivity as described in section 3.1

3.3 Binary representation of burst errors

A burst error may be represented by a binary sequence, erroneous bits being represented by logical ones, and correct bits by logical noughts. Thus the two

possibilities for a three bit burst error would be written as 101 and 111*. The occurrence of errors in a binary sequence may be represented mathematically as the modulo two addition of the original sequence with a second binary sequence representing the error pattern. The process of modulo two addition is the same as that for binary addition, except that all carries are ignored. As a result zeros in the error sequence leave the data bits of equivalent significance unaltered, while ones in the error sequence cause the equivalent data bits to be inverted.

Those error correction systems which operate by generating the error sequence use it to perform a second modulo two addition with the data sequence. This results in the erroneous bits being given a second inversion and therefore returning to their original state.

3.4 Effect of single and burst errors on error correcting codes

The simplest technique for recording an error correcting code is to record each codeword sequentially on a single track, in a manner analogous to the transmission of such codewords over a telemetry link. The response of the system to single errors is independent of the relative positions of the bits of the recorded codewords, due to the random nature of this type of error, and for most tape systems, a codeword of several hundred bits capable of correcting only one bit would function well on this type of error.

Most errors on tape channels however, result from burst errors, which may be up to several hundred bits long with high packing density systems. Such errors are difficult to correct by traditional means, requiring very long codewords with very good correction capability. If a suitable coding method using such means could be devised, it would undoubtedly have a redundancy much higher than is desirable and would be uneconomic to implement. A more attractive approach is to alter the relative placement of the bits of each codeword so that a single burst error produces small correctable errors in many codewords rather than large uncorrectable errors in a few codewords. In multitrack tape systems this may be done by recording each bit of a codeword on a different track. Since with currently available instrumentation recorders, the transverse packing density (*ie* the track density) is several orders of magnitude lower than the longitudinal packing density, individual dropouts seldom affect more than one track, hence a

---

* It is sometimes useful to represent binary sequences in a polynomial form. The 0s and 1s of the sequence become the coefficients of the polynomial, thus the sequence 101 would be written $x^2 + 1$. The application of this symbology to error patterns gives rise to the term 'error polynomial'.

code with one bit correction capability will generally provide a high level of correction. This technique is clearly inappropriate to single track systems, but the same result can be achieved by reducing the effective longitudinal packing density of the codewords. This is done by interleaving each bit of a codeword with large numbers of bits from other codewords so that successive bits of a given codeword are separated by a distance greater than the anticipated dropout length. Such a system needs to store a comparatively large amount of data, both on record and replay, but may be readily implemented using currently available LSI integrated circuits. The required interleaving will result in parts of several thousand codewords being incorporated within the length of tape required to hold a single complete codeword. It is therefore desirable that a code should be selected which is suitable for parallel decoding, as this enables the codewords to be decoded sequentially using a single decoder. The codeword to be decoded may be presented to the decoder in a parallel format by means of a tapped shift register, the decoder being operated when a complete codeword is available at the taps (Fig 3). The length of the sections of the shift register between successive taps will equal the number of bits of other codewords placed between each bit of the codeword being decoded termed the number of 'ways' of interleaving.

If a serial decoding method is employed, an equivalent shift register must be used to hold the data, since the entire codeword must normally be decoded before it can be used for error correction. This shift register may be easier to construct than that for the parallel case since no taps are required, but this advantage is gained at the expense of far greater complexity in the decoding circuitry. This occurs because the interleaved codewords must now be decoded concurrently.

The use of a separate decoder for each codeword being simultaneously decoded, as shown in Fig 4, is unlikely to be cost effective, even if each decoder is simple. The use of a single decoder in a time-sharing mode is feasible (see Fig 5) but since at least one bit of memory would be required for each partially decoded codeword, the total amount of storage required is somewhat greater than that needed for a parallel system.

4 DESCRIPTION OF ERROR DETECTION AND CORRECTION TECHNIQUES USED IN THE SINGLE TRACK RECORDING SYSTEM

4.1 Error correction by means of parity

A codeword may be formed by assembling a block from a number of bits in the data stream, each bit being widely separated from the others, as described above, and performing a modulo two addition of these bits in order to form a parity

code with one bit correction capability will generally provide a high level of correction. This technique is clearly inappropriate to single track systems, but the same result can be achieved by reducing the effective longitudinal packing density of the codewords. This is done by interleaving each bit of a codeword with large numbers of bits from other codewords so that successive bits of a given codeword are separated by a distance greater than the anticipated dropout length. Such a system needs to store a comparatively large amount of data, both on record and replay, but may be readily implemented using currently available LSI integrated circuits. The required interleaving will result in parts of several thousand codewords being incorporated within the length of tape required to hold a single complete codeword. It is therefore desirable that a code should be selected which is suitable for parallel decoding, as this enables the codewords to be decoded sequentially using a single decoder. The codeword to be decoded may be presented to the decoder in a parallel format by means of a tapped shift register, the decoder being operated when a complete codeword is available at the taps (Fig 3). The length of the sections of the shift register between successive taps will equal the number of bits of other codewords placed between each bit of the codeword being decoded termed the number of 'ways' of interleaving.

If a serial decoding method is employed, an equivalent shift register must be used to hold the data, since the entire codeword must normally be decoded before it can be used for error correction. This shift register may be easier to construct than that for the parallel case since no taps are required, but this advantage is gained at the expense of far greater complexity in the decoding circuitry. This occurs because the interleaved codewords must now be decoded concurrently.

The use of a separate decoder for each codeword being simultaneously decoded, as shown in Fig 4, is unlikely to be cost effective, even if each decoder is simple. The use of a single decoder in a time-sharing mode is feasible (see Fig 5) but since at least one bit of memory would be required for each partially decoded codeword, the total amount of storage required is somewhat greater than that needed for a parallel system.

4 DESCRIPTION OF ERROR DETECTION AND CORRECTION TECHNIQUES USED IN THE SINGLE TRACK RECORDING SYSTEM

4.1 Error correction by means of parity

A codeword may be formed by assembling a block from a number of bits in the data stream, each bit being widely separated from the others, as described above, and performing a modulo two addition of these bits in order to form a parity

check bit. The code thus formed is ideal for the application described above, since it is readily generated by parallel means (special purpose components are available for this function in the standard logic ranges) and has the minimum theoretical redundancy of one bit per block, thus allowing the use of short blocks without incurring excessive redundancy. Although it is normally used as a simple error checking code, a parity system is capable of reconstituting any one of the bits making up a codeword, by means of a modulo two addition of the remaining bits of the codeword. It is, however, not possible using this system alone, to discover which bit of the codeword is in error. A separate technique must therefore be used which has good error detection properties, and is able to define the error positions. A Hamming code is used for this purpose.

4.2 The Hamming code

The Hamming code is a systematic block code with single error correction capability. The length of the data block to which the code can be practically applied varies over wide limits, but in all cases where the code is required to perform error correction a sufficient number of check bits must be employed to comply with the relationship

$$n \leqslant (2^r - 1)$$

where n = number of bits in the codeword
r = number of check bits.

Thus, a seven bit codeword would require three check bits and could therefore contain four data bits, a 31-bit codeword would require five check bits and could therefore contain 26 data bits, while a 255-bit codeword would require eight check bits and could contain 247 data bits. It may be seen the redundancy required varies over wide limits according to the block length, being proportionately highest for short block lengths. The code rates $n/(n - r)$ for the above cases are 1.75, 1.19 and 1.03 respectively. In the case of optimal length codewords, *ie* where

$$n = (2^r - 1)$$

the number of states the check bits may assume is one greater than the number of bits in the codeword. There are, therefore, sufficient states to indicate the position of a single error in any position within the codeword, with an extra (usually the zero) position indicating the error free condition. It may thus be seen that the Hamming code achieves the minimum theoretical redundancy for a single error correcting block code, for each optimal codeword length. It follows,

therefore, that the Hamming code has no multiple error detection capability and every error will result in a correction procedure appropriate to a single error condition being applied to the data. If the codeword length is non-optimal a degree of multiple error detection becomes available since some multiple error conditions will result in an apparent single error outside the real codeword. If, however, the minimum possible number of check bits are used, the probability of detecting multiple error conditions by this method will always be less than 50%. If the error correcting abilities of the code are not used (*ie* the code is only used for error detection), a considerable improvement in its error detection performance results. In this case one state indicates the presence of an error free block, and all other states indicate the presence of one or more errors. It may thus be seen that the error detection performance improves as the number of check bits is increased, and is thus best used with long blocks.

4.3 Synchronisation techniques

In order to identify the position of each word within a serial data stream, it is necessary to have some form of synchronisation. A number of ways of providing this facility may be envisaged, depending upon the synchronisation detection reliability required, the degree of redundancy that may be permitted for providing synchronisation information, the speed of synchronisation acquisition needed, and the properties of the channel code to be employed with the system. The two methods most suited to a system which records data in a continuous stream at high packing density are:-

(i) modification of the channel code used by the system to produce a unique encloded sequence which cannot be generated by the data (*eg* the use of 'missing edge' techniques with the Manchester codes);

(ii) the use of a pre-determined word (sync word) or word sequence inserted in the data stream at regular intervals.

The latter technique is used in the single track system since it is likely that the system will be required to operate with channel codes which have insufficient inherent redundancy to provide the extra sequences required to indicate the sync position by means of a unique encoded sequence (*eg* delay modulation). In addition, the inherent redundancy resulting from the multiple bit structure of a sync word reduces the number of false sync detections and can be used to provide a tolerance to small numbers of errors occurring within the word, thus improving sync detection.

In the proposed single track system, 16 data words are placed between each pair of sync words, this providing a suitable compromise between sync acquisition

time and redundancy. The 16 data words between any two sync words are termed a frame, and the encoding process is arranged to reset at the beginning of each frame, forming an encoded frame in which the position of data bits and check bits are invariant.

Since tape transport speed instabilities such as flutter may preclude the use of bit synchronisers capable of freewheeling for the duration of large dropouts, the number of bits lost during a dropout will not be known. The resultant mis-clocking of the decoding electronics will result in the sync words on each side of the dropout having an incorrect relative spacing in the data domain. This spacing is checked in the system by means of a counter, and is used for the detection of large errors. It is also possible for one or more successive sync words to be rendered undetectable by dropouts. In this case it is important to know how many sync words have been lost (and hence how much information). This is determined by means of a three bit frame identification count which increments on successive frames.

5 IMPLEMENTATION OF THE SINGLE TRACK RECORDING SYSTEM

The above techniques are applied in the single track system as follows:-

5.1 The record system

Parity bits are added to the end of each 12-bit data word (Fig 6). These parity bits are formed from 12 bits of the preceding data stream, each separated from the next by 80 words plus one bit. This separation, which corresponds to a separation in excess of 1000 bits of the encoded signal on the tape, is sufficient to give a very low likelihood of a single dropout spanning two bits forming the same parity bit[3]. The extra bit separation causes each bit forming a given parity bit to be of a different significance in its respective data word. By this means, every data bit has one parity bit to check it. Hamming codes are generated from groups of 16-data words including their appended parity bits (Fig 6). The data stream is then interleaved in such a manner that each bit of a given word is separated from its neighbours by three bits of words from another part of the frame. The resulting 224 bit sequence is divided into four 56-bit subframes each of which includes four complete words. Two 8-bit parity words (A and B) are formed as shown in Fig 7, thus producing a dual Hamming code. This code, the theory of which is explained in Appendix A, enables single and small burst errors to be corrected while also having an error detection performance approaching that of a Hamming code used for detection purposes only. The synchronisation words and frame identification words are added as shown in Fig 7, and the signal is then passed to the channel encoder and recorded. A detailed description of the record system is given in Appendix C.2.

5.2 Decoding of the single track format and detection and correction of errors

The error correction procedures are illustrated in flow charts 20a and 20b. The first stage of decoding is the detection of the synchronisation words (Fig 20a). The validity of the data is checked at this point by counting the number of bits separating the sync words. This check shows up frames of incorrect length (in the data domain) this fault usually being caused by mis-clocking of the channel decoders due to dropouts on the tape. A check carried out simultaneously on the frame identification count reveals the presence of sync words too badly corrupted to be detected by the sync detection circuits, and thus allows the system to determine the length of multiframe burst errors (Fig 17).

The dual Hamming code formed by parity words A and B is next decoded. The decoding circuitry for this code (Fig 19) is programmed to correct burst errors of up to three bits in length but as each bit of a given codeword is separated from the adjacent bit in the codeword by three bits from other codewords, due to the interleaving process employed during encoding, burst errors of up to 12 bits can be corrected at this stage. Errors which cannot be corrected by the dual Hamming code will generally be detected by it, thus the error position will be defined to within four subframes (each of which include four words) for each codeword in which an undetectable error occurs.

The next stage in the error detection/correction process is the decoding of the Hamming code having parity word C as its check bits (Fig 20b). This code is used for error detection only and locates the error position to within one 16 word frame. It may thus be seen that errors detected but not corrected by the dual Hamming code will also normally be detected by parity word C. By correlating the information obtained by these two tests, it is possible to define an erroneous region to within one or more four word subframes.

The erroneous subframes discovered in the above test, and the frames which have failed the sync recognition tests are now corrected using the distributed parity bits, which allow any one of the bits associated with a given parity bit to be corrected. Assuming that about 1000 words of intact data are present on either side of the burst error, then the wide spacing of the parity bits allows burst errors of up to 80 words long to be corrected. This is much greater than is likely to occur from dropouts. The theoretical performance of the system with random distributions of single errors and burst errors of the type 111 --- 111 (Fig 2) *ie* worst case burst errors is, shown in Figs 22 and 24 respectively. A detailed description of the decoding system is given in Appendix C.3.

6 CONCLUSIONS

Sufficient design work has been completed to show that the system is a practical one to construct. It is considered to be cost-effective in relation to the expense appertaining to the acquisition of aircraft trials data, particularly as most of the complexity is in the ground replay section, which, in an operational situation, would service many aircraft.

Considerable thought has been given to the theoretical operation of the system under many error conditions, and operation of some of the circuitry has been demonstrated practically. At the present stage, no major design changes appear necessary. A simulator capable of generating both pseudo-random and ramp inputs has been constructed and may be used to simulate most error conditions in order to check that the actual response of the decoder follows its theoretical one. Following these tests, the system will be evaluated while in use with a tape transport in a simulated airborne vibration environment.

As the low data rates of the record system would appear to offer considerable scope for sequential working, consideration is being given to the replacement of much of the random logic in the encoder by a microprocessor. This would have the advantage of reducing the size, weight and power consumption/dissipation of the airborne system and might give a reduction in cost. It is envisaged that the final design will take the form of an add-on unit compatible with most ARINC 573 or similar types of single track recording systems. This would give considerably enhanced data reliability by virtue of the very high degree of error correction and examination of the data corrected flags during replay should give early warning of incipient failures of the recording system and of the need for corrective maintenance.

Appendix A

THEORY OF ERROR DETECTION AND CORRECTION TECHNIQUES USED IN THE SINGLE TRACK RECORDING SYSTEM

A.1 Use of Hamming codes in the presence of burst errors

A Hamming code is generally used for the correction of single bit errors, since if sufficient check bits are added to the data block to be encoded, the syndrome* obtained by decoding will indicate the position of a single error occurring within the code[1,6]. The Hamming code cannot, however, differentiate between single bit errors and ones affecting more than one bit, thus if a multiple bit error occurs, it will not be corrected by the code.

A.1.1 Multiple error detection properties of Hamming codes

Consider a code of length n bits, containing k message bits. The number of different messages that may be sent is 2^k. Since a given message always has the same set of check bits attached, it follows that there can only be 2^k different valid code words. Since the length of the code is n bits, the degree of the error polynomial which can affect the code is $n - 1$, hence there are 2^n combinations of this polynomial. Owen and Harknett[5] show that for a cyclic code the modulo two addition of any two code words produces a third codeword. It thus follows that if the error polynomial (see section 3.3) corresponds to a valid codeword, addition of this to the transmitted message will produce a further valid codeword and the error will therefore not be detected.

Since the no error condition (error polynomial all 0 s) is one of the valid codewords, there are $2^k - 1$ error patterns which cannot be detected by the Hamming code. As $2^n - 1$ error patterns can occur, the probability of a random burst error not being detected by the code is:-

$$\frac{2^k - 1}{2^n - 1} .$$

---

* Codes which reconstitute the error sequence, such as the Hamming codes, produce a binary sequence when they are decoded called the syndrome, which indicates the position of the errors. Each error pattern which is to be corrected must correspond to a unique state of the syndrome so that it can be identified. The term is thus used because the situation is analogous to the medical case in which an illness is diagnosed by its symptoms.

If k and n are large, this approximates to:-

$$\frac{2^k}{2^n} = 2^{(k-n)} = \frac{1}{2^r}$$

where r = number of check bits.

It may thus be seen that the Hamming code has the potential for good burst error detection, if large parity words are used, and single error correction is not required.

A.2 Extended Hamming code

The extended Hamming code adds an extra bit to the normal Hamming code check bits, this being formed from the modulo two addition of each bit of the codeword[6]. Thus if the original codeword contains an even number of ones, the extra bits will be 0. If it has an odd number of ones, the extra bit will be 1. Thus the extended code always has an even number of ones. If no errors occur, the syndrome will be zero and the number of ones in the codeword even. If a single error occurs, the syndrome will indicate its position and the number of ones in the codeword will be odd, thus correction can be applied. If two errors occur, the syndrome will be non-zero and the number of ones in the codeword even. This can be recognised as an uncorrectable condition. If three errors occur, the syndrome will be non-zero and the number of ones in the codeword odd, thus the condition is incorrectly interpreted as being the single error condition and an incorrect correction procedure is applied.

It may thus be seen that an extended Hamming code provides 100% correction of a single error within its code, 100% detection of a double error, but only 50% detection of greater numbers of errors, and is thus not ideal for single error correction/burst error detection.

A.3 Dual Hamming code

The dual Hamming code makes use of the property of the Hamming code that multiple errors will, in general, be diagnosed as single bit errors occurring in a different position from the actual error(s). The relative position of the actual error pattern and its indicated error is a function only of the generator polynomial, being independent of the absolute position of the error, provided it is confined entirely within the codeword. If two parity words are generated for the same data block, therefore, using different generator polynomials, the indicated positions of single bit errors will be the same, whereas those for

multiple errors will in general be different. It is thus only necessary to subtract the two 'error positions' to differentiate between a single error and a multiple bit one. If the first parity word generated is included in the block of the second, single errors occurring within the first parity word can also be corrected. Greater elaboration is required, however, if single errors occurring in the second word are to be recognised as such. The multiple error detection properties of this code are very good, since to remain undetected, an error must correspond to a codeword of both systems, *ie* it must be divisible by both generator polynomials. Since the generator polynomials are by definition irreducible, they can have no common factors, hence their lowest common multiple is obtained by multiplying them together. This means that the parity words will have the same effect as one word of twice the length when used for burst error detection, *ie* the probability of a random burst error not being detected by the code is:-

$$\frac{1}{2^{2r}}$$

where r = the length of each parity word.

A.3.1 Use of dual Hamming code for multiple error correction

The dual Hamming code may alternatively be used to correct certain multiple error conditions at the expense of its burst error detection properties. This process uses the property that a certain pattern of error will produce a difference between the apparent single error positions predicted by the two syndromes, which is independent of the error position. A decoder for multiple error correction is shown in Fig 19. The apparent single error position from syndrome B is subtracted from that of syndrome A to give the multiple error syndrome. This is fed into the address of a Read Only Memory which is programmed to recognise the multiple error syndromes corresponding to the multiple error conditions which it is required to correct. The memory then outputs the appropriate error pattern to the correction circuitry and also to the error position correction circuitry. This circuitry adds the difference between the apparent error position and the actual one, to one of the decoded syndromes in order to obtain the actual position of the error. The 'error type' decoder memory may be programmed to detect as many error conditions as there are multiple error syndromes, but correction of a large number of conditions would result in complex correction circuitry, thus a practical limit is reached. In addition, each burst error condition corrected reduces the multiple error detection capability. This is because a family of errors exists for every multiple error syndrome, and use of the syndrome to detect

a multiple error condition presupposes that the correctable error condition has occurred. It may thus be seen that the multiple error correction facility should only be used to correct error combinations which are particularly likely to occur, *eg* burst errors.

A.4 Use of parity for error correction

The use of a parity bit generated by the modulo two addition of all the bits within its block for single error detection is well known. It can, however, also be used for the correction of a single error, provided that the position of the suspected error within the block is known by means of other error detection tests. Whereas the Hamming codes reconstitute the error polynomial, which may then be added to the erroneous data to correct it, the parity bit may be used to reconstitute any one bit of the data within its block, provided that the rest of the block is error free, *eg* for a received codeword

$$m_1 m_2 m_3 m_4 p$$

where $p = m_1 \oplus m_2 \oplus m_3 \oplus m_4$

a parity check is performed thus:-

$$C = m_1 \oplus m_2 \oplus m_3 \oplus m_4 \oplus p$$

if an error is suspected in m_1 put $m_1 = 0$,

therefore $$C = m_2 \oplus m_3 \oplus m_4 \oplus p$$

substituting for p

$$C = m_2 \oplus m_3 \oplus m_4 \oplus m_1 \oplus m_2 \oplus m_3 \oplus m_4 .$$

Performing modulo two addition

$$C = m_1 ,$$

ie the result of the parity check C can be used to replace the suspect bit.

A.4.1 Use of parity for burst error correction

The technique described in section A.4 is clearly unsuitable for burst error correction, since only one bit within the codeword may be corrected. A simple modification, enabling the correction of burst errors, is to employ a convolutional block system. If each bit in a given codeword is separated from each other bit by

n bits, a burst error of n bits can be corrected. This is because a burst error of p bits will result in p codewords having one bit in error if $p \leqslant n$. If $n < p < 2n$ only partial correction can be achieved since $2n - p$ codewords will contain one error and be corrected, and $p - n$ codewords will contain two errors and thus be uncorrectable. If $p = 2n$ no correction is possible unless the error overlaps the start of a new block.

Appendix B

DESIGN PRINCIPLES OF THE ERROR CORRECTING SYSTEM

The system employs a two tier correction system which enables small errors to be corrected prior to the large burst error correction circuitry. This minimises the loss of burst error correction performance in conditions where, due to pattern sensitivity or jitter, the incidence of small burst or single errors is higher than normal.

Small errors are detected and corrected by parity words A and B (the primary correction system) as described in section 5.2. Error combinations not correctable at this point will generally also be detected by parity word C , the information from the two detection systems enabling the error to be located to within groups of four words, for correction by the secondary system.

Large burst errors will generally cause mis-clocking and will therefore be detected by the sync detection circuitry. This will result in an integral number of frames requiring correction by the secondary correction system.

B.1 The primary correction system

Parity words A and B form a dual Hamming code which is used to correct single bit errors and small burst errors. The 'error type' decoder has been programmed to correct burst errors of up to three bits, *ie* error patterns 1, 11, 101, 111. The parity words are attached to a convolved block which extends over four frames as described in Appendix C.2.1. Each bit in the block is separated from the next by at least three bits, thus burst errors of twelve bits can be tolerated in the data as only three adjacent bits of any block are affected.

B.2 Interleaving

It would be possible to apply the convolved block of parity words A and B without first interleaving the data, with no loss of correction efficiency of the primary correction system. When uncorrectable errors were found, however, the affected block would encompass parts of 64 words, which would all require correction. By interleaving in such a fashion as to include *whole* words in the convolved blocks, the number of words affected by each block is reduced to 16, thus reducing the number of words requiring secondary correction as the result of small errors.

B.3 Secondary error detection

B.3.1 Parity word C

Parity word C is used only for error detection. As its block consists of a single complete frame of data, the four subframes included in a given parity word C

block each come from a different parity word AB block. Thus if an error pattern occurs which cannot be corrected by the primary system, the erroneous frames are detected by parity word C, allowing the error to be located to groups of four words. An important function of parity word C is the detection of incorrect burst error corrections made by parity words A and B (section A.3.1), since if the dual system makes a wrong assumption about the error pattern, the resulting erroneous blocks will be detected by parity word C.

B.3.2 Sync word

Large burst errors will almost certainly cause mis-clocking of the replay signal. As a result, the frame(s) in which error has occurred will have an apparent 'shortening' (in terms of bit count). In this case, the sync words at each end of an affected frame will not arrive simultaneously at the parallel output sections of the sync detection register, thus the sync will not be recognised. Frames thus affected, and any frames failing the tests described in Appendix C.3.1, thus causing the sync detection system to return to the search mode, are flagged erroneous throughout and require secondary correction.

B.4 Secondary correction system

The parity bits whose generation is described in Appendix C.2.1 are used for secondary correction. The subframes requiring correction are detected by the means described above, the correlation of the three error detection tests being described in Appendix C.3.2(d) (Fig 20). As each bit making up a given parity bit is separated from the other bits by 80 words and one bit, a burst error up to five frames long can be corrected, this length being in excess of that likely to occur in practice. The correction process is inhibited by the control circuit, if more than one bit requires correction from one parity bit, as correction is not possible under these circumstances (see Appendix A.4).

Appendix C

DETAILED DESCRIPTION OF ENCODING AND DECODING METHOD

C.1 Nomenclature

In order to facilitate description of the system it has been necessary to define a subscripted nomenclature. Although the symbols will be defined as they appear in the text, the overall rules governing the use of the nomenclature are given in this section, so as to avoid burdening the following text with them. These are:-

SUBSCRIPTS - these appear in order of decreasing significance thus:-

$$g_{20}(2,7)$$

would refer to the seventh bit of the second subframe of the twentieth frame.

USE OF UPPER AND LOWER CASE - the lower case letter refers to the *individual bit* occurring in the larger group defined by the letter. Use of an upper case letter imples the presence of a complete group thus:-

$$F_4$$

would refer to the entire fourth frame.

$$f_4(12)$$

would refer to the twelfth bit of the fourth frame.

NB The lower case letter thus always requires one more subscript for its definition than its upper case counterpart.

C.2 The record (encoding) system

A block diagram of the record system is shown in Fig 8. From this it will be seen that the data are subjected to several coding processes before being channel encoded prior to recording. The basic principles of the record system are described in section 5.1.

C.2.1 Parity bit generator

The data, which are presented to the input of the encoder as serial 12 bit words, first pass through twelve 1041 (80 × 13 + 1) bit shift registers. The output of each register is fed to a parity generator which performs a modulo two

addition of these lines (Fig 9). The resultant parity bit p is fed into the input of the first register via a two-way multiplexer, after the entry of each data word. Thus an input data stream,

$$.....M(i)\ M(i+1)\ M(i+2)..... \xrightarrow{\text{time}}$$

where $M(i) = m(i,1)\ \ m(i,2).....m(i,12)$

becomes $M(i)\ p(i)\ M(i+1)\ p(i+1)\ M(i+2)\ p(1+2)$

where $p(i) = m(i-80,12) \oplus m(i-160,11) \oplus ....m(i-880,2) \oplus m(i-960,1)$.

It should be noted that each data bit associated with a given parity bit comes from a different word. This allows the spreading of the bits required for burst error correction, as described in section 5.1 without spreading the actual data words. By this means the number of data words affected by each burst error is minimised thereby improving the burst error correction properties of the system.

C.2.2 Generation of parity word C

On leaving the parity bit generator, the consecutive sequence of words with their appended parity bits, are organised in blocks of 16, which will be termed *frames*, these being given the symbol F. Thus for the jth frame:-

$$F_j = f_j(1)f_j(2).....f_j(208)$$

and

$$F_j = M_j(1)\ p_j(1)\ M_j(2)\ p_j(2).....M_j(16)\ p_j(16)\ .$$

These frames are then used as blocks for the generation of a Hamming code. The generator polynomial used is $x^8 + x^4 + x^3 + x^2 + 1$.

The modulo two division circuit is shown in Fig 10. An eight-bit parity word results from the division, which is called parity word C (C_j for the jth frame). At the end of each frame a three-way multiplexer (Fig 8) is used to introduce parity word C and a second 8-bit word into the data stream. This 8-bit word (D) is subject to a lesser degree of correction than the main data stream, but may be used for the recording of documentary data since this is sufficiently predictable to allow efficient correction by means of software. The documentary data may thus be kept separate from the main data stream, this having advantages in systems where the documentary data controls the de-multiplexing programme. The data stream after the multiplexer may be represented by:-

$$.....F_j\ D_j\ C_j\ F_{j+1}\ D_{j+1}\ C_{j+1}\ F_{j+2}\ D_{j+2}\ C_{j+2}\ .....\ \text{(Fig 6)}.$$

C.2.3 Interleaving

Four-way bit interleaving of the frame and appended 8-bit words now takes place. The operation may be represented by 52 × 4 matrix:-

$$\begin{bmatrix} d_j(1) & M_j(1) & p_j(1) & d_j(2) & M_j(2) & p_j(2) & d_j(3) & M_j(3) & p_j(3) & d_j(4) & M_j(4) & p_j(4) \\ d_j(5) & M_j(5) & p_j(5) & d_j(6) & M_j(6) & p_j(6) & d_j(7) & M_j(7) & p_j(7) & d_j(8) & M_j(8) & p_j(8) \\ c_j(1) & M_j(9) & p_j(9) & c_j(2) & M_j(10) & p_j(10) & c_j(3) & M_j(11) & p_j(11) & c_j(4) & M_j(12) & p_j(12) \\ c_j(5) & M_j(13) & p_j(13) & c_j(6) & M_j(14) & p_j(14) & c_j(7) & M_j(15) & p_j(15) & c_j(8) & M_j(16) & p_j(16) \end{bmatrix}$$

where $M_j(i) = m_j(i,1), m_j(i,2).....m_j(i,12)$
and each element of $M_j(i)$ forms a separate column.

The interleaved frame F' may be obtained by reading sequentially down the columns of the matrix from left to right.

It is useful to define four subframes G(1), G(2), G(3), G(4) formed from the rows of the above matrix. Hence the above matrix for the jth frame may be written:-

$$\begin{bmatrix} Gj(1) \\ Gj(2) \\ Gj(3) \\ Gj(4) \end{bmatrix}$$

The interleaving is implemented using two 256 × 1 bit random access memories (Fig 14). For clarity of explanation, the write process (Fig 12) and the read process (Fig 13) are described separately.

During the write process, the memory address for each entry is selected by counters A and B . Counter A is a four bit presettable counter which, during the data entry, is programmed to return to a count of three after reaching full scale, thereby dividing the clock rate by 13 (2^4-3). Since the data (including parity bits) are in the form of 13-bit words M(i)p(i) at this point, counter A always contains a fixed number for a data bit of given significance. At the end of a word (with associated parity bit) counter A is set to 3 and counter B is incremented. On completion of the entry of the 16 words belonging to a given frame, counter A is set to 2 and counter B is reset. The eight spare bits are now entered from the DOC input, counter B being incremented by one for each. Parity word C (8 bits) is next entered, counter A remaining at 2 and counter B continuing to be incremented.

In the read mode, the memory addresses are selected by counters C , D and E. Counters C and E divide by 4 while counter D is programmed to return to a count of 2 at the end of its count, and thus performs a divide by 14 function. It can be seen that the arrangement of counters shown in Fig 13 will result in the memory being read out in such an order as to produce frame F' .

In order to use the random access memory for both write and read functions, it is necessary to multiplex its address inputs in the manner shown in Fig 14. As the reading and writing functions are required simultaneously (in the interest of smooth data flow) and the order of use of the locations is different for both functions, it is necessary to add a second RAM so that one may be written into whilst the other is read, their roles interchanging at the end of each frame.

C.2.4 Generation of parity words A and B

Two 8-bit parity words A_j and B_j are attached to the end of each interleaved frame F'j . The block from which B_j is formed consists of:-

$$G_{j-3}(4)\ G_{j-2}(3)\ G_{j-1}(2)\ G_j(1)\ ,$$

while A_j is formed from the same block plus the parity word B_j , *ie*

$$G_{j-3}(4)\ G_{j-2}(3)\ G_{j-1}(2)\ G_j(1)\ B_j \quad \text{(Fig 7)}\ .$$

Parity word B_j is formed by modulo two division of its blocks by $x^8 + x^6 + x^5 + x^4 + 1$ (Fig 11), while for A_j the reciprocal polynomial $x^8 + x^4 + x^3 + x^2 + 1$ is used (Fig 10).

C.2.5 Insertion of frame identification count and sync word

At the end of each frame a three bit identification count (Fig 8) is added which increments by 1 for successive frames. This is followed by a 13-bit Barker type sync word (Fig 16). The system timing is arranged so that the data stream is continuous at this point, the next frame following immediately after the end of the sync word (Fig 7). The data stream is thus ready for channel encoding[4] and is of the following form:-

$$.....F'_j\ B_j\ A_j\ \phi_j\ S\ F'_{j+1}\ B_{j+1}\ A_{j+1}\ \phi_{j+1}\ S$$

where ϕ_j is the frame identification word for the jth frame

where $0 \leq \theta_j \leq 7$

and $\phi_{j+1} = \phi_j + 1$ for $0 \leq \phi_j < 7$
and $\phi_{j+1} = 0$ for $\phi_j = 7$
and where S is the sync word.

C.3 The replay system

The replay system may be divided into two sections, the first recognising sync and identifying frames, and the second decoding the replayed data and correcting errors where possible. The basic principles of the replay system are given in section 5.2.

C.3.1 Sync detection and frame recognition

The channel decoded replay data first enters the sync detection register (Fig 15). This register has a total length of 272 bits, the first and last 16 bits having parallel outputs. The data is shifted serially through the register, and sync detection for frame j occurs when the register contains

$$\phi_{j-1}\ S\ F'_j\ B_j\ A_j\ \phi_j\ S \quad .$$

The sync words and frame identification words are at this time in the parallel access sections of the shift register. Modulo two addition takes place between each of the sync words and a 'mask' consisting of a static pattern of the inverse of the sync word. The output of the modulo two adders thus consists of 26 lines, each of which indicates the coincidence of a bit of the received sync words with the true sync word pattern. These are then summed in a binary adder, each input being given a weight of 1. A comparator then decides whether sufficient bits have been recognised for the sync to be labelled valid. A histogram showing the worst case coincidence pattern for a typical sync word is included (Fig 16). A two level sync recognition system is used, initial recognition requiring both syncs to be error free (search mode), and subsequent detection being timed to the expected appearance of the sync words and allowing a total of four errors to occur in any two consecutive sync words (run mode). The frame identification counts are also used in sync verification, the loss of either one of the two counts checked at any time being tolerated in the run mode. The detailed operation of the sync circuitry is shown in a flow chart (Fig 17a&b). Frames which have failed the above test will often be in gross error, hence all such frames are flagged as erroneous throughout. The setting of this flag overrules the results of all subsequent error detection tests.

C.3.1(a) Buffer store

Successfully identified frames are clocked into one of eight 240-bit shift registers in the eight frame buffer store (Fig 15), the register selected being determined by the frame identification count. When frames are not identified, the register into which they would normally be clocked remains unloaded. The registers are read out in sequence, the unloaded registers providing a space in the data stream for the reconstituted data to be inserted.

C.3.1(b) Output clock generator

Assuming that the data has been recorded on the tape using a self-clocking channel code (as is usual practice in digital recording) the channel decoder will provide both the decoded data and a clock signal. This clock signal may, in principle, be used to operate the error detection and correction electronics. In practice, however, the clock signal will fail during drop-outs, causing mis-clocking and hence failure of the decoding electronics. To avoid this problem, it is necessary to use a locally generated clock signal. This is done using a variable frequency oscillator whose frequency is controlled so as to keep a constant relationship between the clocking in (under conditions of good data) and clocking out of the buffer store.

C.3.2 Error detection and correction

The block diagram of the error detection and correction (decoding) system is shown in Fig 18.

C.3.2(a) Primary correction system

Parity word A and B dividers (Fig 18) operate over the same blocks that were used originally to generate the two parity words and also include the appropriate check bits as in normal Hamming code practice. The remainders in the dividers are the syndromes indicating the error condition of the respective blocks[1,6]. The syndromes are separately decoded in 256 × 8 bit Read Only Memories which output the apparent position of the error (Fig 19). The outputs are subtracted, and the result passed to the 'error type' decoder (a 256 × 4 bit ROM). If a correctable error is detected, the correction circuitry is activated and the 'detected and corrected' flag is set. If the error is non-correctable the 'detected not corrected' flag is set. As parity words A and B are formed from the four frames preceding them, it is necessary to provide a store of this length for the data, since correction can only take place after the check words have been read.

C.3.2(b) Re-formatting stores

The purpose of the re-formatting stores (Fig 18) is to return the data to the order it was in prior to the interleaving process described in the encoding section. The process is the exact reverse of that described in Appendix C.2.3 and the circuitry is identical, with the exception that counters C, D and E are used to write into the store, while counters A and B read from it.

C.3.2(c) Error detection by parity word C

The Hamming code whose generation is described in Appendix C.2.2 is now decoded using the circuit of Fig 10. The syndrome is not used to locate a single bit error as is normal practice, but merely to test for the existence of an error. The decoding of the syndrome thus consists only of an 8 input gate, testing for the all logical 0's condition.

C.3.2(d) Secondary correction control (Figs 20a&b)

The detection of an error by parity word C locates that error to within one frame. The four subframes making up this frame will all be associated with different A and B parity word pairs, and will thus each have a different pair of flags. If any of the four 'uncorrected error' flags are set, the correction request flag to the secondary correction system is set for the flagged subframes. If none of the 'uncorrected error' flags are set, any 'corrected error' flags which are set cause the 'correction request' flag for the appropriate subframes to be set in the event of an error being detected by parity word C . If none of the flags are set, correction of the entire frame is requested if parity word C indicates the presence of an error. If the sync/frame identification error flag is set, all the above is overridden, and the correction flags are set for all four subframes.

C.3.2(e) Secondary correction system

The secondary correction system consists of twelve 1041 bit shift registers each separated from the next by an arrangement of multiplexers (Fig 21). In parallel with the 1041 bit registers are twelve 20 bit registers, in which the correction request flag for each subframe within the main register is held. The multiplexers are normally in the position shown, but if a logical 1 appears at any one of the taps of the correction request flag register, the associated multiplexers in the main register are switched. This switches the relevant input of the modulo 2 adder to a logical 0 and the input of the next register to the output of the modulo 2 adder. Valid corrections can only be made once every 13 shifts, *ie* when a parity bit is at the input of the secondary correction register. The

flags are therefore gated with a correction enable line which inhibits corrections when data bits are at the input. The control also detects the simultaneous presence of more than one correction request flag. When this condition occurs, the correction enable pulses are inhibited, and all the words in each subframe requesting correction at that time are flagged uncorrected.

Appendix D

RESPONSE OF THE SYSTEM TO ISOLATED BURST ERRORS

D.1 Small errors

D.1.1 Error in data, parity bits and parity word C

As these all make up the interleaved frame F' , their error response is identical. Errors of 12 bits or less are corrected by the primary system. Errors greater than this will require some secondary correction, *eg* a 16-bit error will require secondary correction of up to four subframes.

D.1.2 Errors in parity word B

The primary correction system can correct a burst error of three bits in parity word B (this not being interleaved). Thus an error of four or more bits results in the primary system flagging an uncorrected error. As parity words A and B are not included in the block of parity word C , however, no error is found by parity word C , and no secondary correction is required.

D.1.3 Errors straddling parity word B/data

For errors less than 13 bits, no correction is necessary if the error does not affect the subframe whose parity word B is in the frame (always subframe 1). In the worst case, therefore, a 7-bit error could cause the correction of one subframe by the secondary system.

D.1.4 Errors in parity word A

All errors in parity word A are uncorrectable by the primary system, but by the argument used in section D.1.2 no secondary correction is required.

D.1.5 Errors in frame identification count

Loss of one frame identification count causes no error. If two successive frame identification counts are lost, the sync detection system goes into the search mode, and two frames require secondary correction.

D.1.6 Errors in sync word

Loss of four bits or less in any two successive sync words causes no error (regardless of their distribution). Loss of five bits or more results in two frames requiring secondary correction (assuming that there are no sync or frame errors in the following two frames).

D.2 Large errors

D.2.1 Errors in data

As explained in section B.3.2 large errors wholly within the data will cause the loss of one frame.

D.2.2 Errors affecting sync word or frame identification count

Since a large error will cause the sync detection system to go into the search mode, any error in the following sync word or frame identification count will cause the sync acquisition following the error to be invalidated, thus causing the loss of two frames.

Appendix E

THEORETICAL ERROR PERFORMANCE FOR RANDOMLY OCCURRING SINGLE AND BURST ERRORS

In order to allow small errors to be efficiently corrected, while also dealing with the large burst errors associated with high packing density systems, two levels of data correction are applied in the single track system. The first of these, termed the primary correction system employs a dual Hamming code which is capable of correcting a burst error of up to three bits within a given block and is also effective in detecting burst errors of longer duration and multiple error conditions, which it is unable to correct. Four way interleaving of the dual Hamming code blocks increases the effective burst error correction performance to 12 bits, for errors occurring entirely within the interleaved section.

Small errors which cannot be corrected by the primary correction system will normally be corrected by the secondary correction system, thus in considering the performance of the system in correcting small errors, it is of value to determine the number of words requiring correction by the secondary correction system (with consequent reduction in large error correction performance) as well as the number of erroneous words actually output from the overall system.

Large burst errors such as those produced by dropouts in high packing density systems are corrected by the secondary correction system. In this case correction is applied by means of parity. The spacing between consecutive data bits from which a given parity bit is formed is five frames (80 words) + one bit. For secondary correction to be possible, only one bit out of the block forming a given parity bit may be erroneous, thus the maximum length burst error that may be entirely corrected is five frames. In practice, burst errors are unlikely to last this long at currently achievable packing densities, thus two or more errors requiring secondary correction must occur, before the correction system fails.

E.1 *Effect of single errors on correction system*

The average probability p of a bit being in error in a PCM replay signal having an error rate of 1 in 10^m is 10^{-m}.

If the errors are entirely randomly occurring single bit ones, *ie* no burst errors are present except those produced from combinations of closely occurring single errors, then the probability of each bit being in error is independent of the probability of every other bit being in error and equals 10^{-m}. In this case, it can be shown that the probability p_k of k errors occurring in n bits is given by

$$p_k = {}^nC_k p^k (1-p)^{n-k} = \frac{n!}{k!(n-k)!} 10^{-mk}(1-10^{-m})^{n-k}$$

and the probability p_{k+} of k or more errors occurring in n bits is given by

$$p_k = \sum_k^n {}^nC_k p^k (1-p)^{n-k} = \sum_k^n \frac{n!}{k!(n-k)!} 10^{-mk}(1-10^{-m})^{n-k} \quad .$$

These formulae have been applied to the single track format in order to produce the single error correction performance graphs shown in Fig 22. For simplicity of calculation the following approximations were made:-

(a) the burst error correction properties of the primary correction system were ignored;

(b) errors affecting the data, frame identification words and sync words were treated as being independent;

(c) each data frame was treated as being independent of all the others.

(a) and (b) will cause the result to be pessimistic and (c) will cause the result to be optimistic but in each case, the departure from the correct result should be small over the range of error rates shown.

E.2 Effect of burst errors on correction system

Little quantitative work on the distribution of the lengths and intensities of dropouts occurring in magnetic systems tape used in airborne environments has been performed. Knight[3] has obtained experimental results which indicate that less than 1% of dropouts exceed 0.8 mm which corresponds to approximately 300 bits at 400 bits mm^{-1} (10 K bits in^{-1}) but he points out that his results varied considerably between samples of tape. In addition, it is not necessarily valid to assume that a given dropout length distribution will result in an equivalent distribution of burst error lengths, since this will depend on a number of factors determined by the replay system, *eg* the detection method employed for reconstituting the digital signal, the freewheeling ability of the replay clock generator, the synchronisation properties of the channel code employed etc.

It is thus considered impractical by the author to formulate an accurate theoretical burst error distribution, and the distribution shown in Fig 23 has therefore been taken as being a reasonable approximation to the practical case, but at the same time allowing the system performance to be determined with

relative ease. The 300-bit maximum length dropout is dependent on the packing density of the recording, and has been taken assuming a packing density of 400 bits mm^{-1} (10 K bits in^{-1}). The 10-bit minimum length is dependent on the characteristics of the replay electronics, and has been selected as a typical length error over which the electronics might lose count of the number of bits, thus causing a frame error.

The average length burst error, taking the distribution shown in Fig 23 is 155 bits. This figure can be divided into the bit error rate (assuming this is entirely due to burst errors) to give the burst error rate. Burst errors of 240 bits or less can corrupt only one sync word, and therefore can cause the loss of a maximum of two frames, (32 words). The probability of burst errors of length 10 to 240 bits corrupting a sync word is dependent upon the length of the error, thus each length error will have a different probability of causing one or two frames to be in error. Similarly burst errors of length 241 bits to 300 bits will result in either two or three frames being lost. It is thus possible to arrive at a relationship between the bit error rate (burst errors) off the tape, and the resulting frame error rate of the system (before correction). This can be shown to be:-

$$\text{FRAME ERROR RATE} = 2.74 \times \text{BIT ERROR RATE} .$$

Because all frame errors are assumed to be short enough to affect only one bit of a given secondary correction codeword, the probability of two or more bits in a codeword requiring correction is the same as if single errors were occurring at this point. The formulae given in section D.1 have thus been used, with the frame error rate replacing the bit error rate, to give the burst error response shown in Fig 24.

REFERENCES

No.	Author	Title, etc
1	R. Swanson	Understanding cyclic redundancy codes. Computing designs, November 1975
2	F.A. Bellise	A mult-channel digital sound recorder. IERE Conference Proceedings No.35 (1976)
3	G.A. Knight	Study of magnetic tape for airborne data recording. EMI Central Research Laboratories, May 1975
4	J.C. Mallinson J.W. Miller	Optimal codes for digital magnetic recording. IERE Conference Proceedings No.35 (1976)
5	A. Owens M.R. Harknett	Introduction to coding techniques. Electronic Engineering, August 1975
6	A. Owens M.R. Harknett	Basic and extended Hamming codes. Electronic Engineering, December 1975/January 1976
7	W.W. Peterson	Error correcting codes. The MIT Press and John Wiley & Sons Inc. (1972)

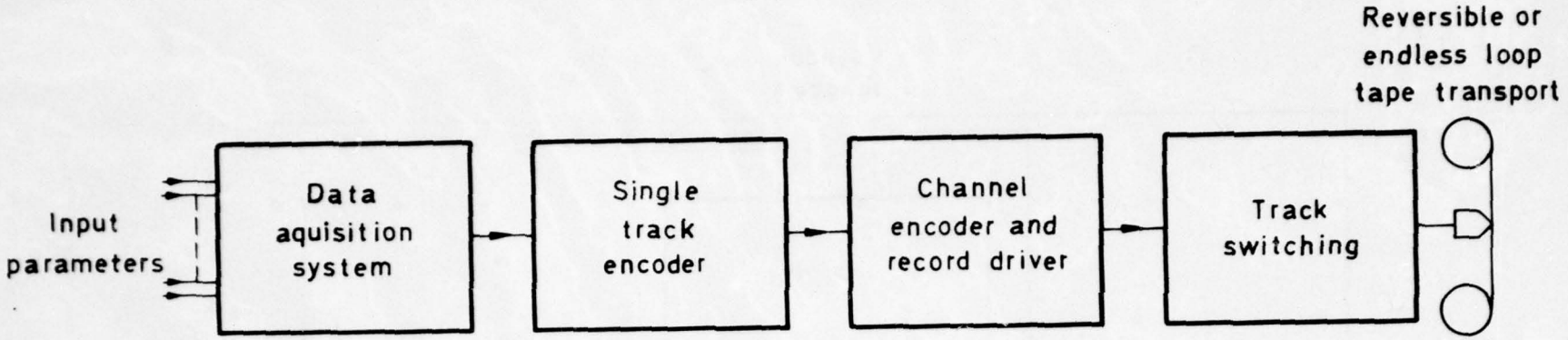


Fig 1 Block diagram of data aquisition and recording system using single track error correcting format

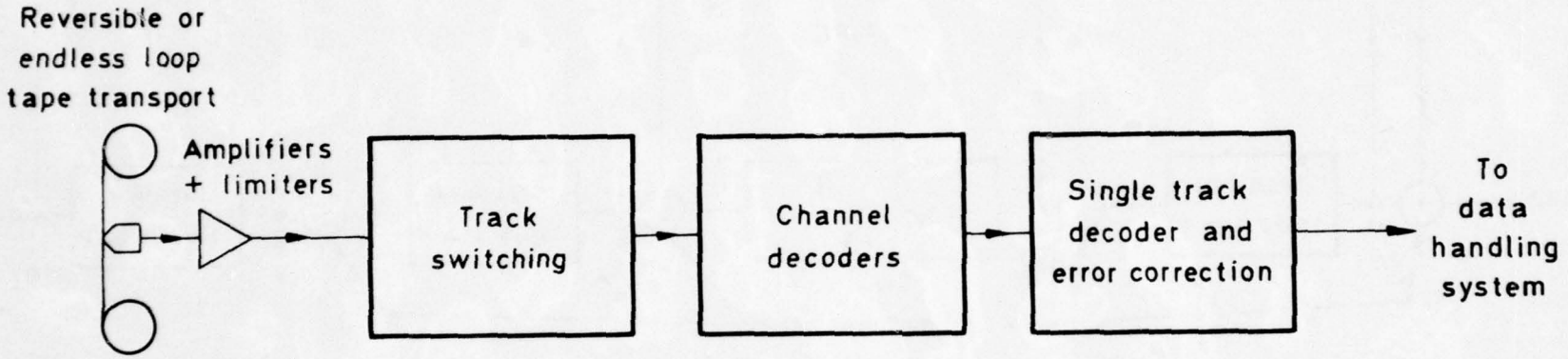


Fig 2 Block diagram of replay system for use with single track error correcting format

Figs 1&2

Fig 3

a = Number of ways of interleaving

n = Length of codeword in bits

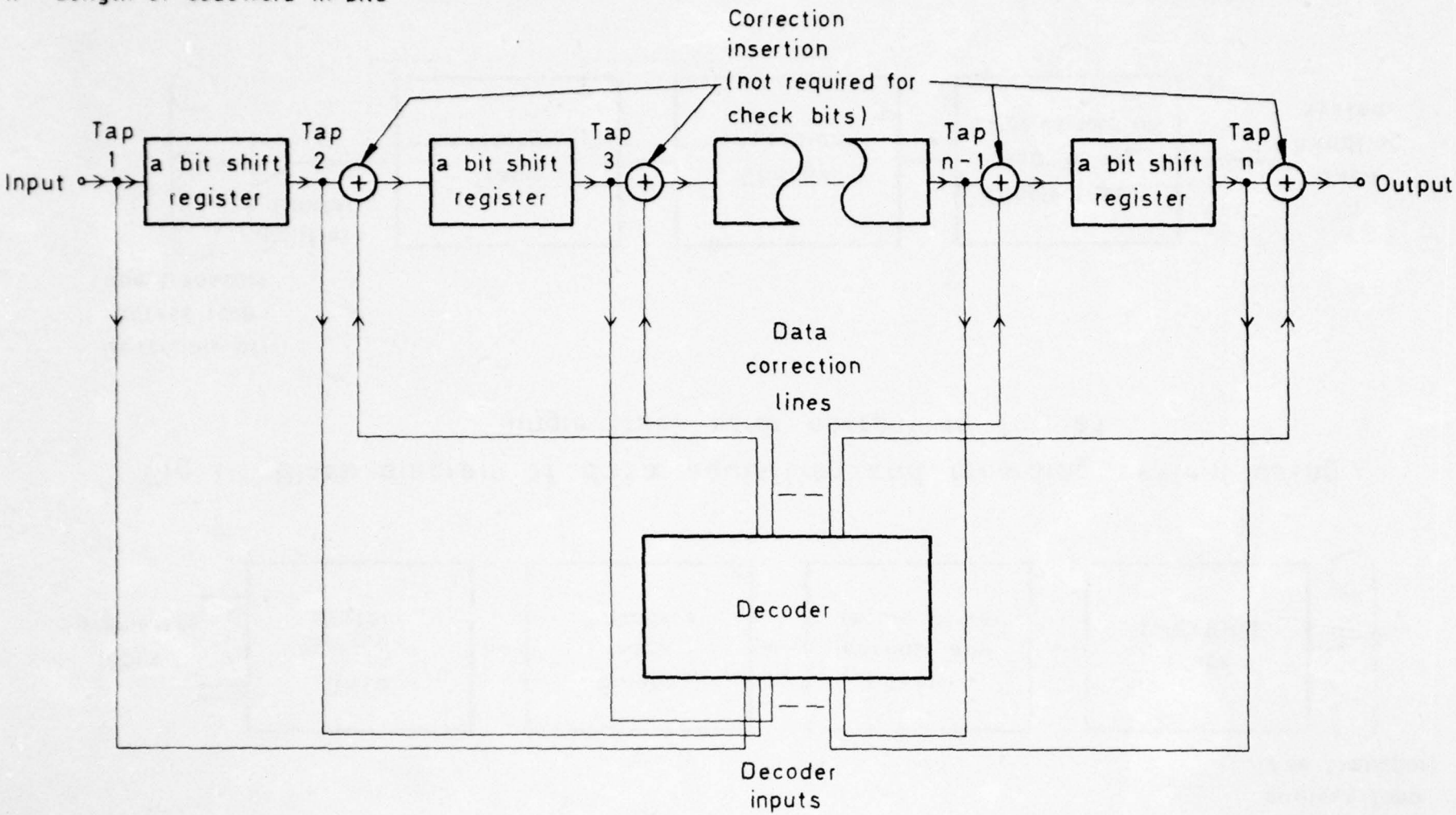


Fig 3 Parallel decoding of interleaved error correcting code

a = Number of ways of interleaving
b = Number of bits storage required to decode a single codeword
n = Length of codeword in bits

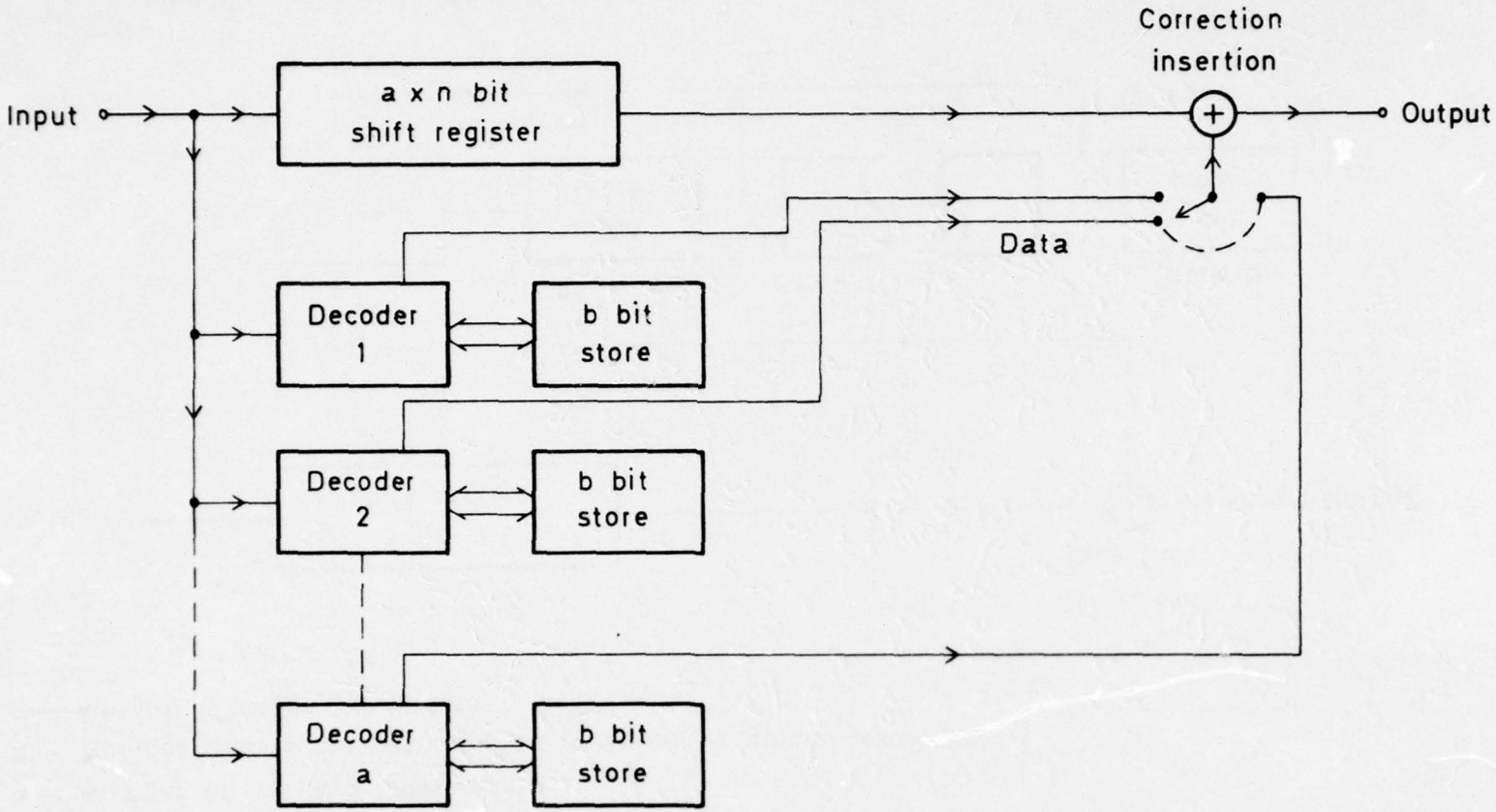


Fig 4 Serial decoding of interleaved error correcting code using multiple decoders

Fig 4

a = Number of ways of interleaving
b = Number of bits storage required to decode a single codeword
n = Length of codeword in bits

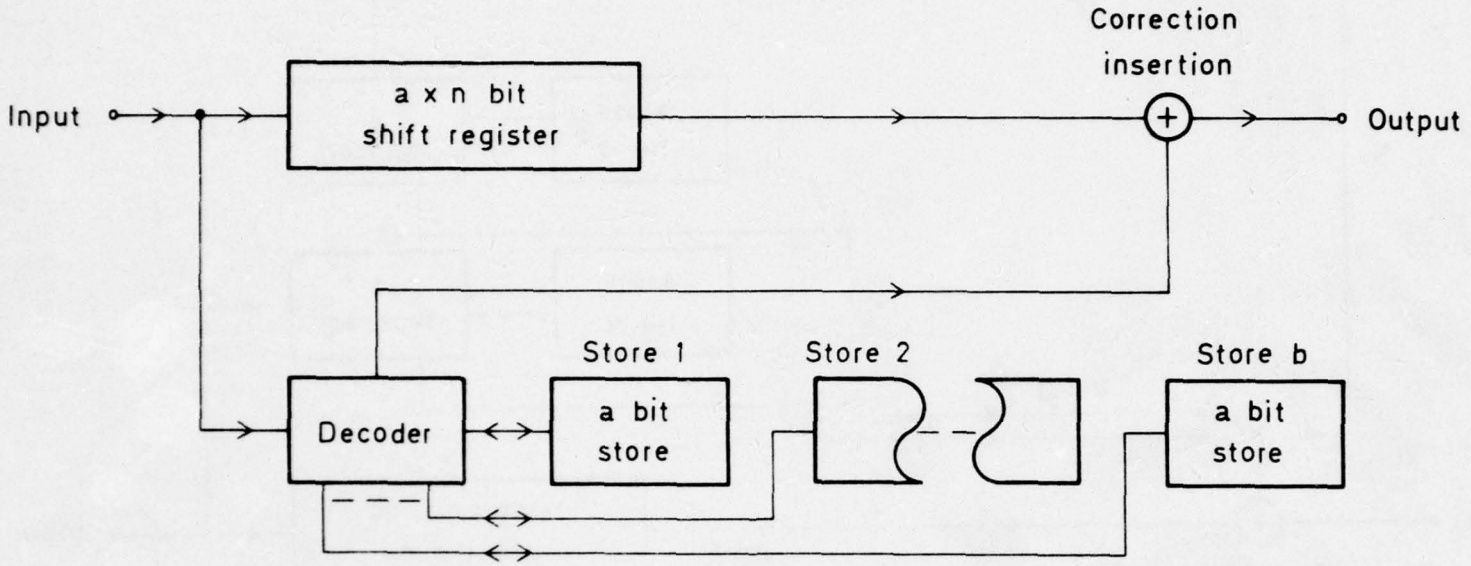


Fig 5 Serial decoding of interleaved error correcting code using single time – shared decoder

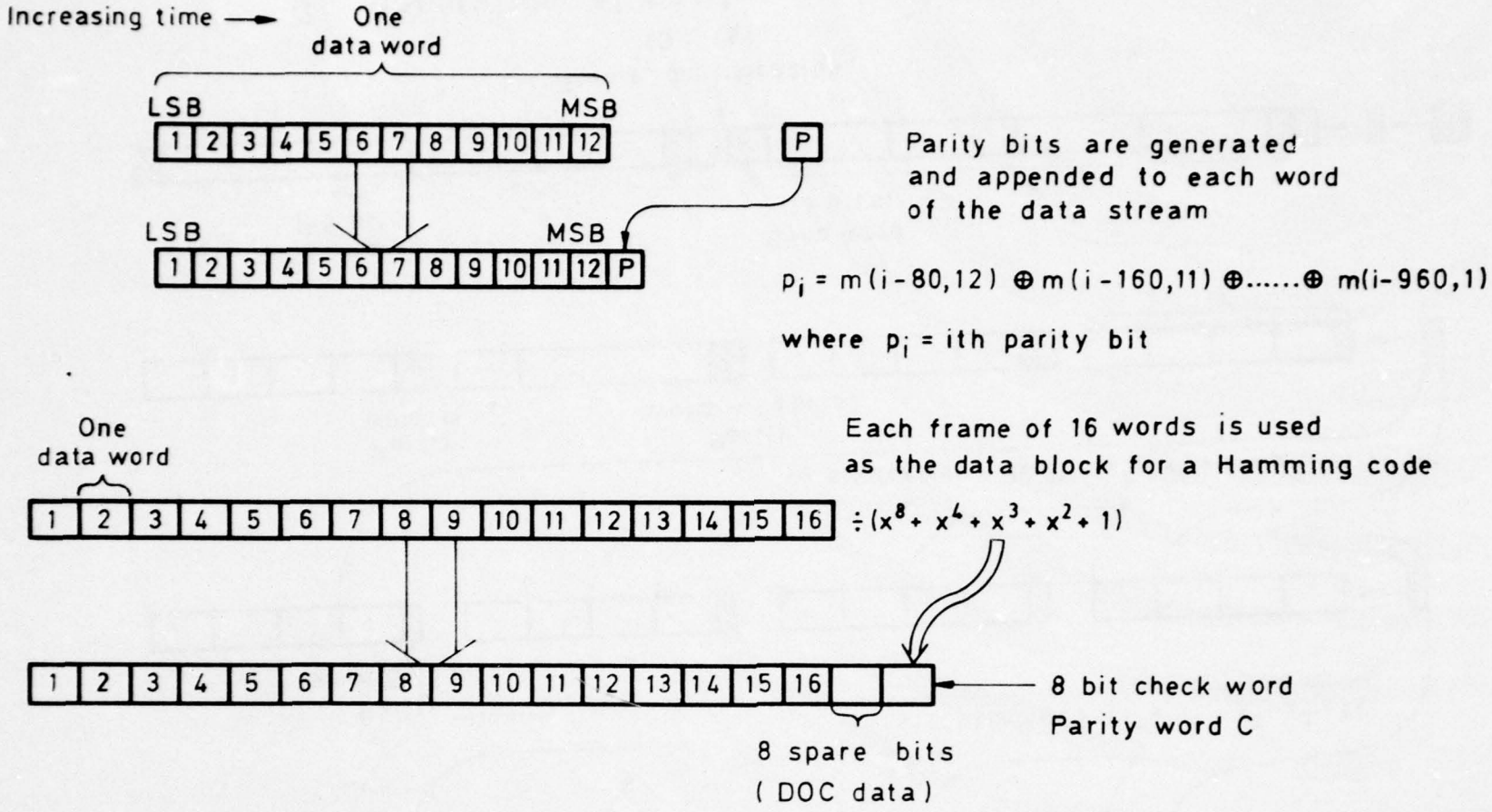


Fig 6 Generation of parity bits and parity word C

Fig 6

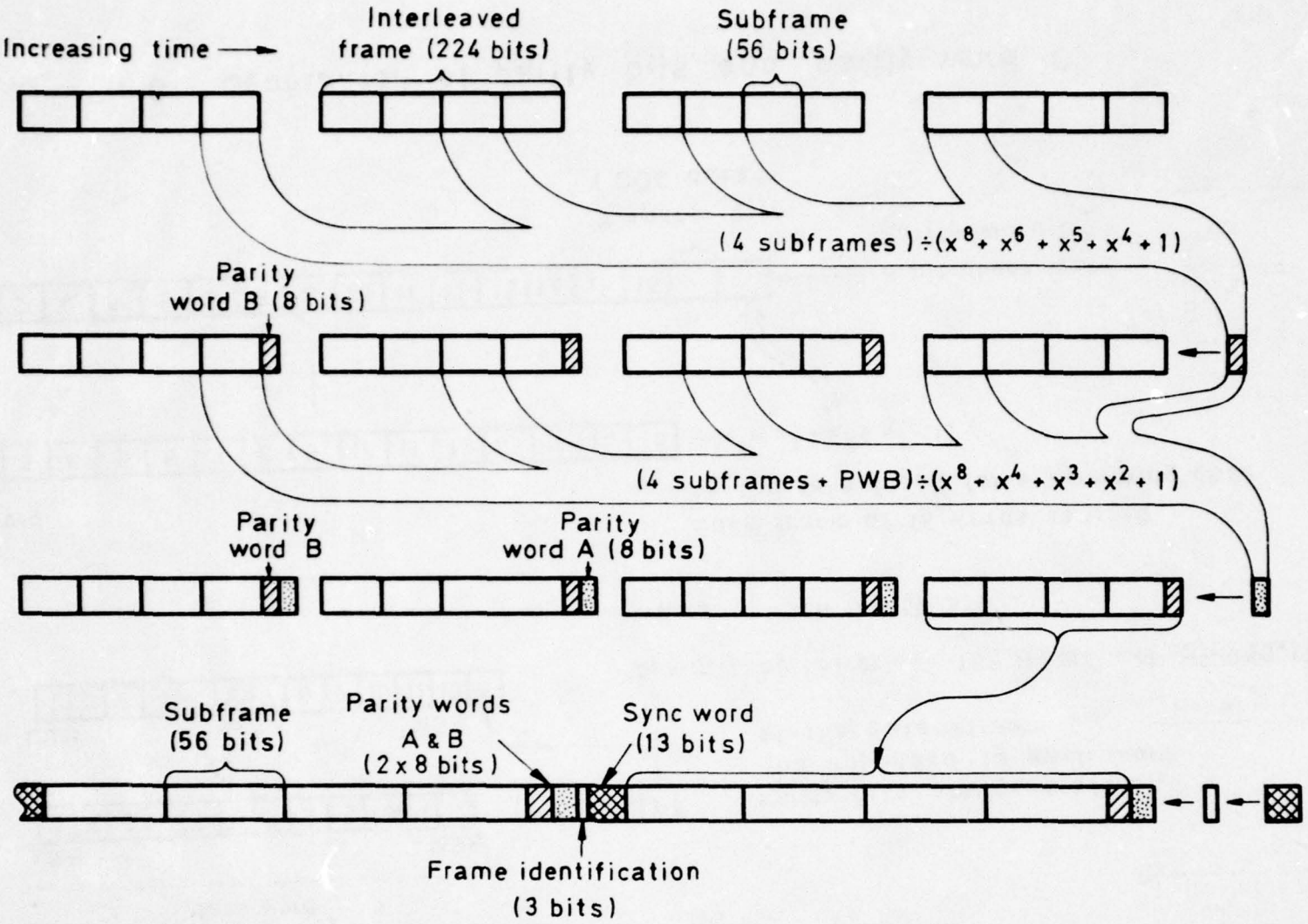


Fig 7 Generation of parity words A & B and insertion of synchronisation information

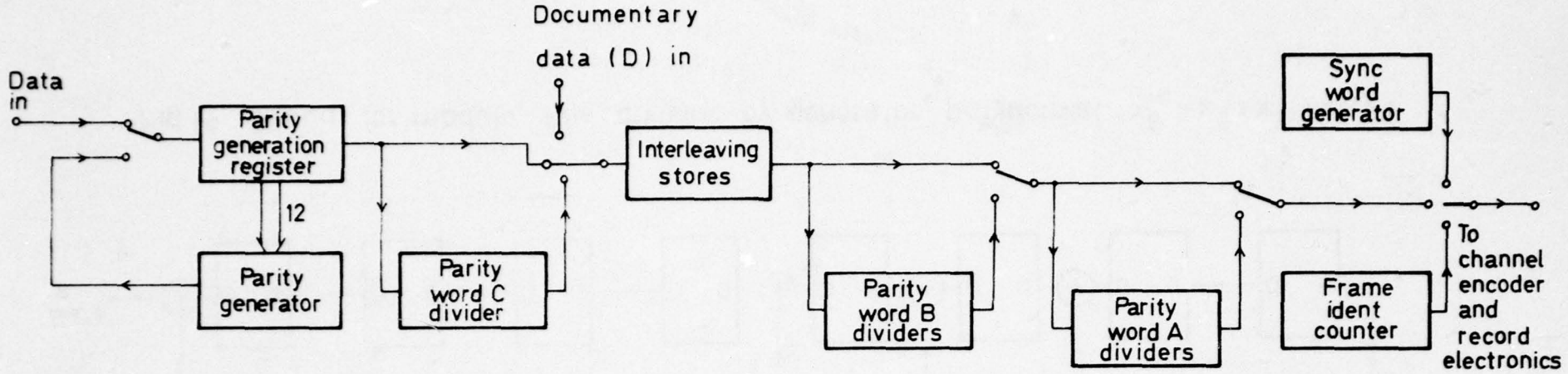


Fig 8 Block diagram of record (encoding) system

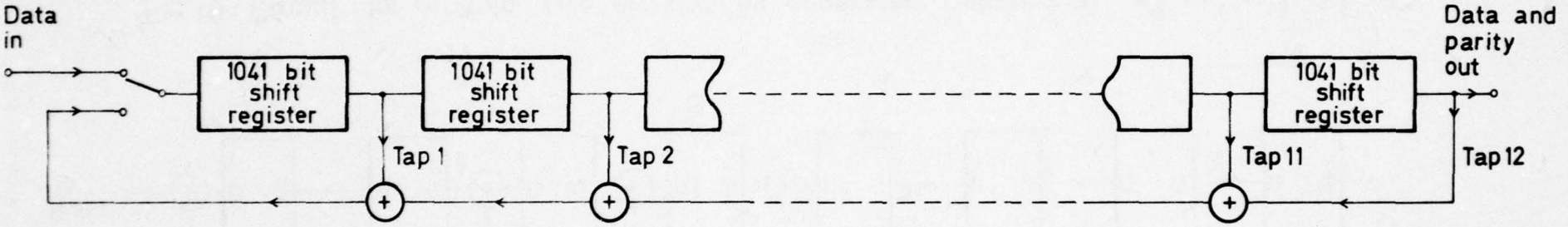


Fig 9 Parity bit generation circuit

Figs 8 & 9

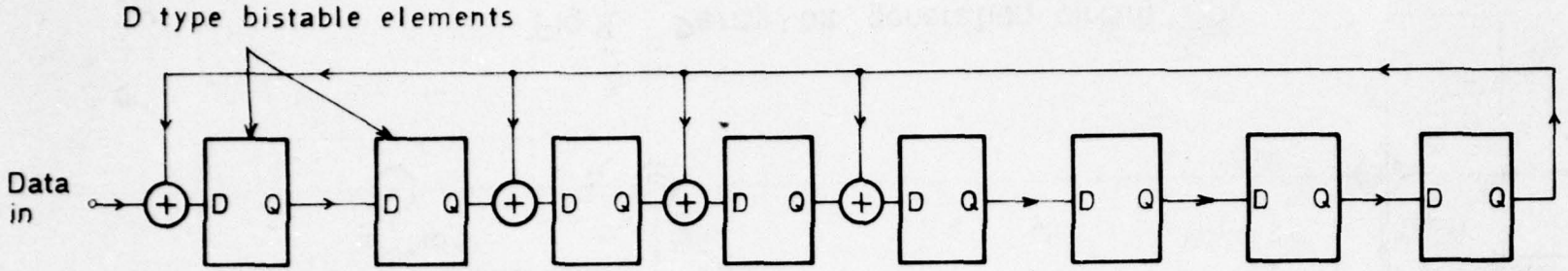


Fig 10 Circuit for modulo two division by generator polynomial $x^8 + x^4 + x^3 + x^2 + 1$

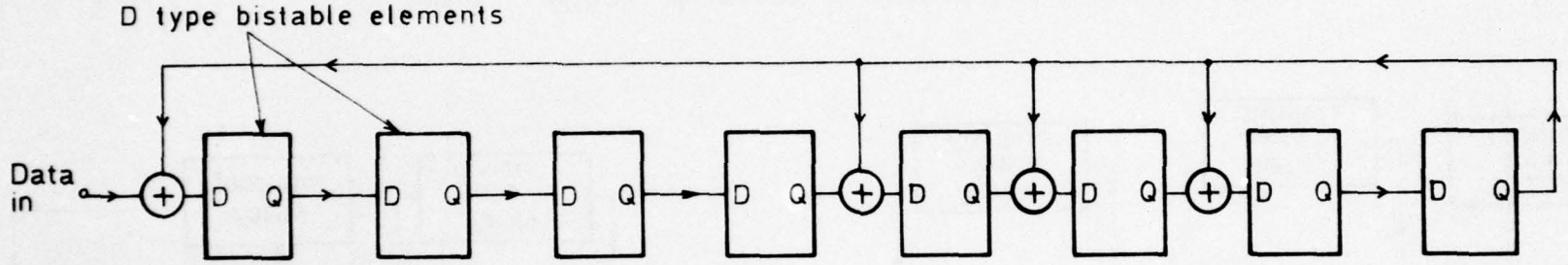


Fig 11 Circuit for modulo two division by generator polynomial $x^8 + x^6 + x^5 + x^4 + 1$

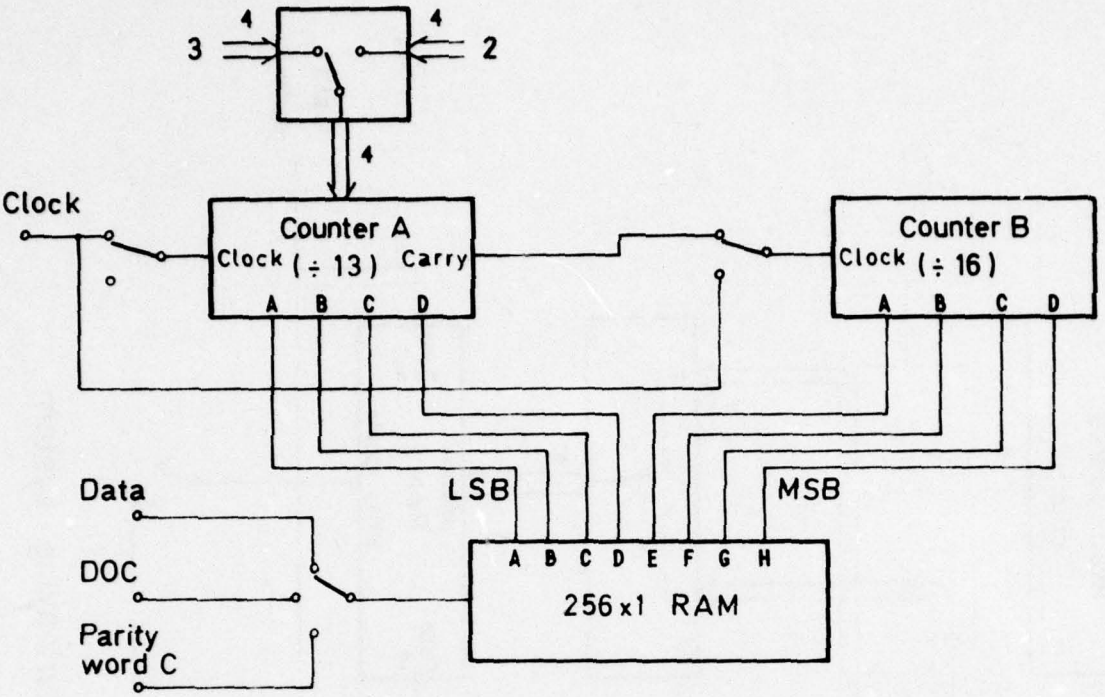


Fig 12 Interleaving store write circuit

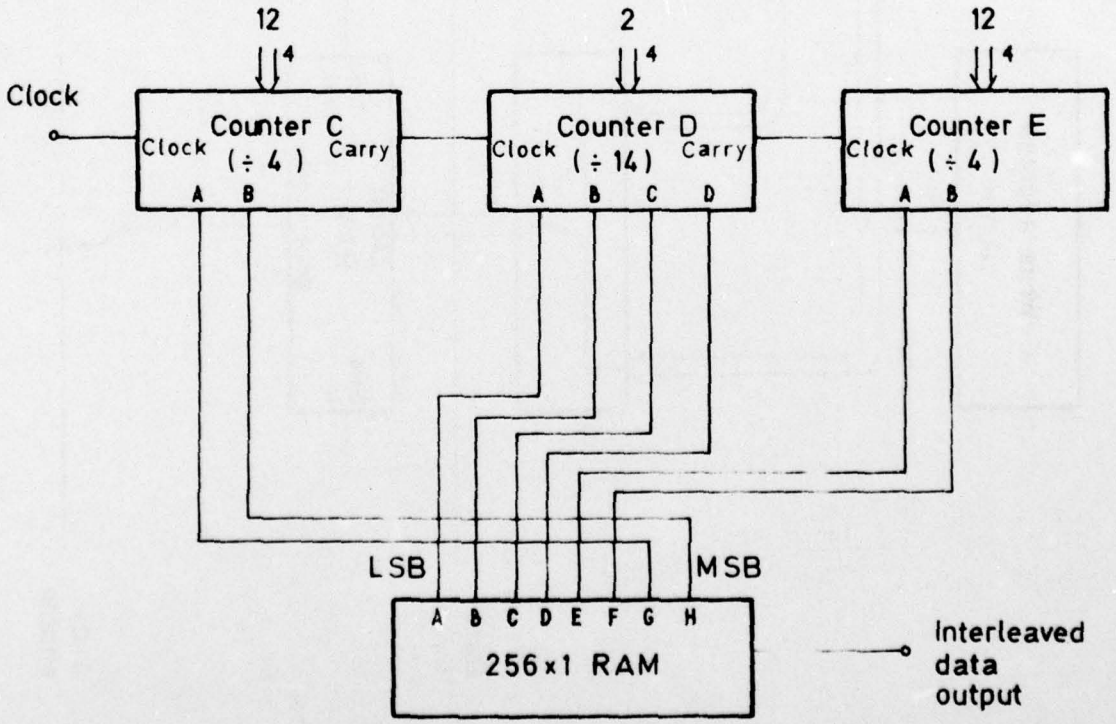


Fig 13 Interleaving store read circuit

Fig 14

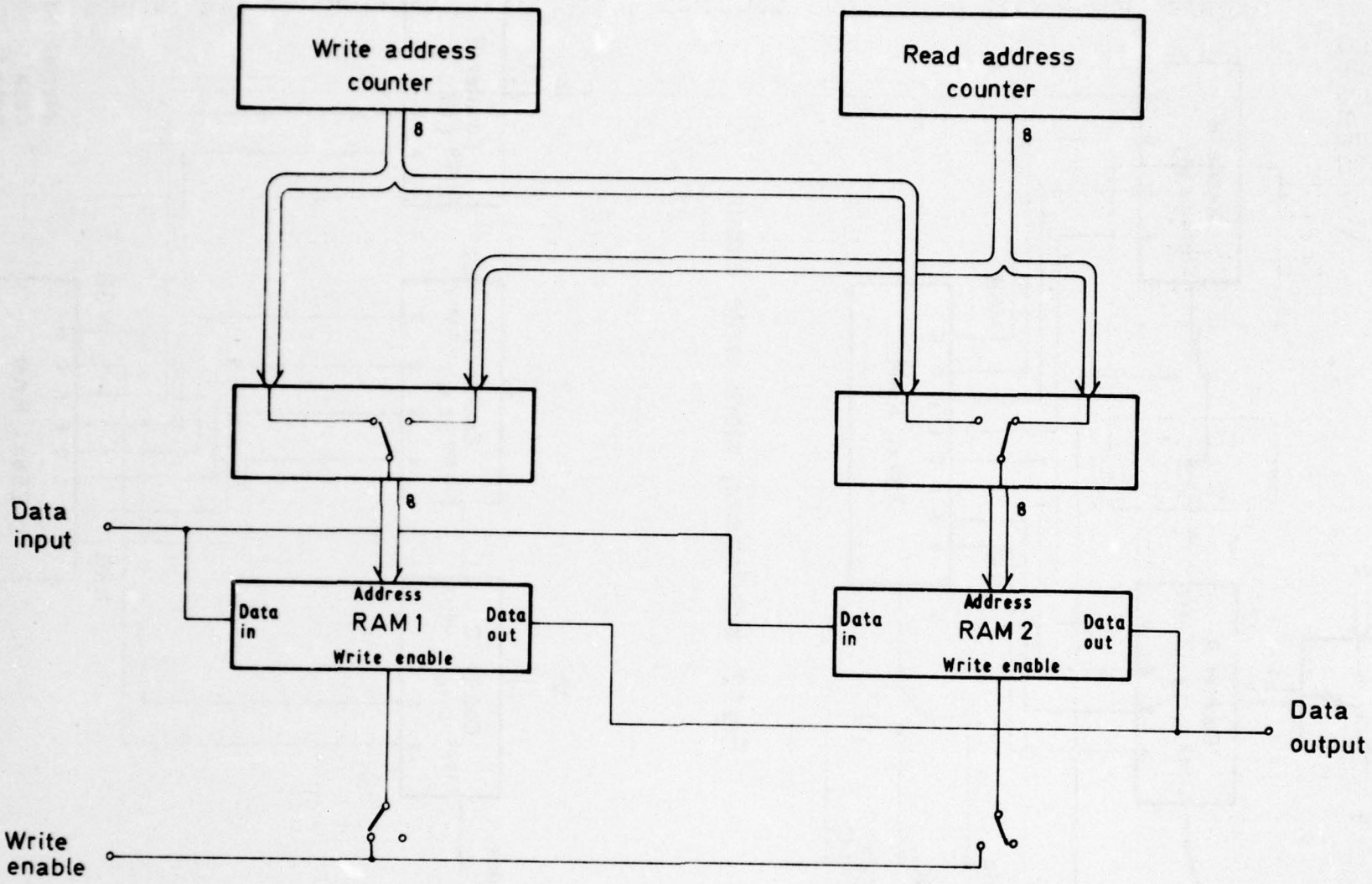


Fig 14 Overall block diagram of interleaving system

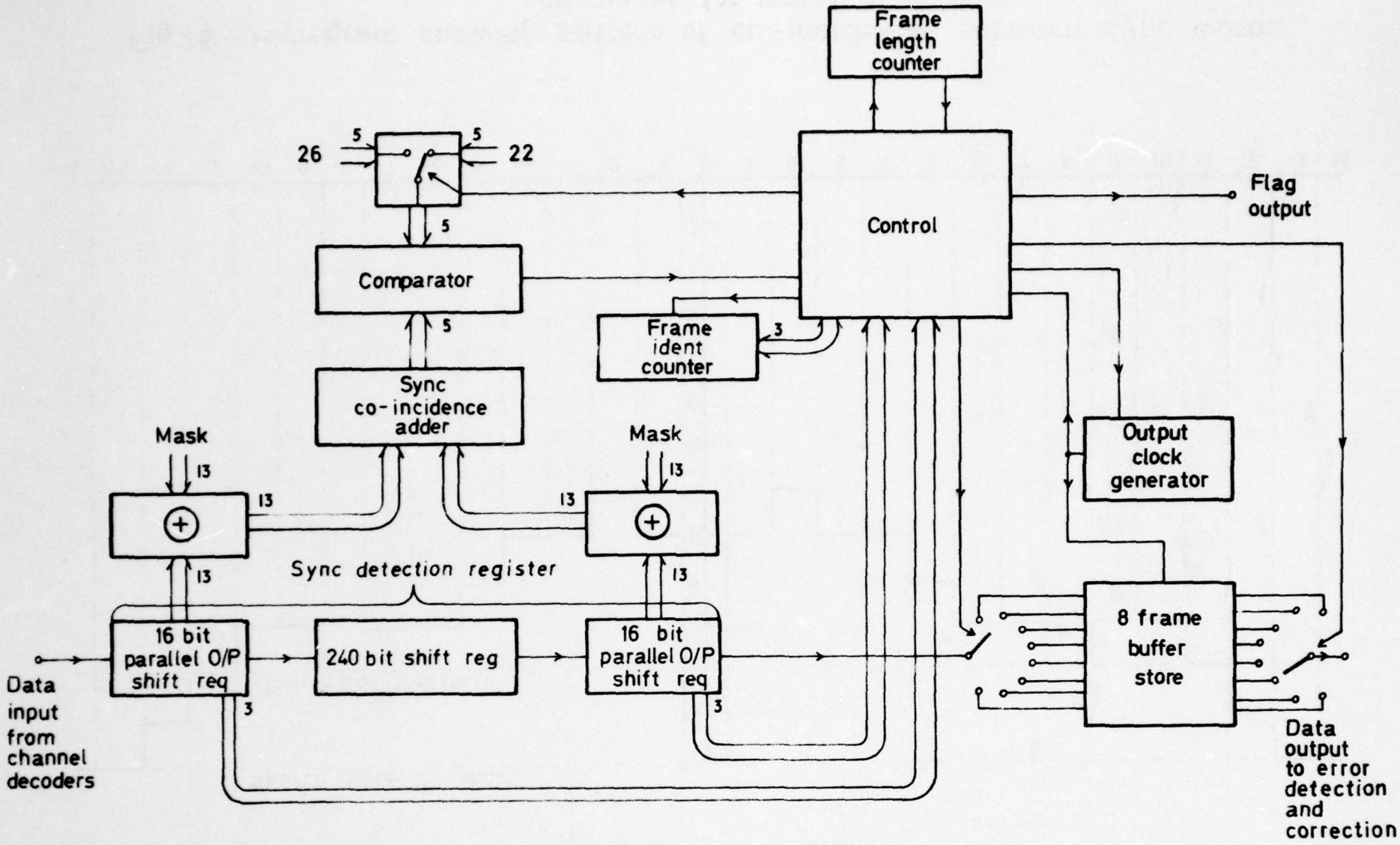


Fig 15 Block diagram of sync word detection and frame recognition system

Fig 15

Fig 16

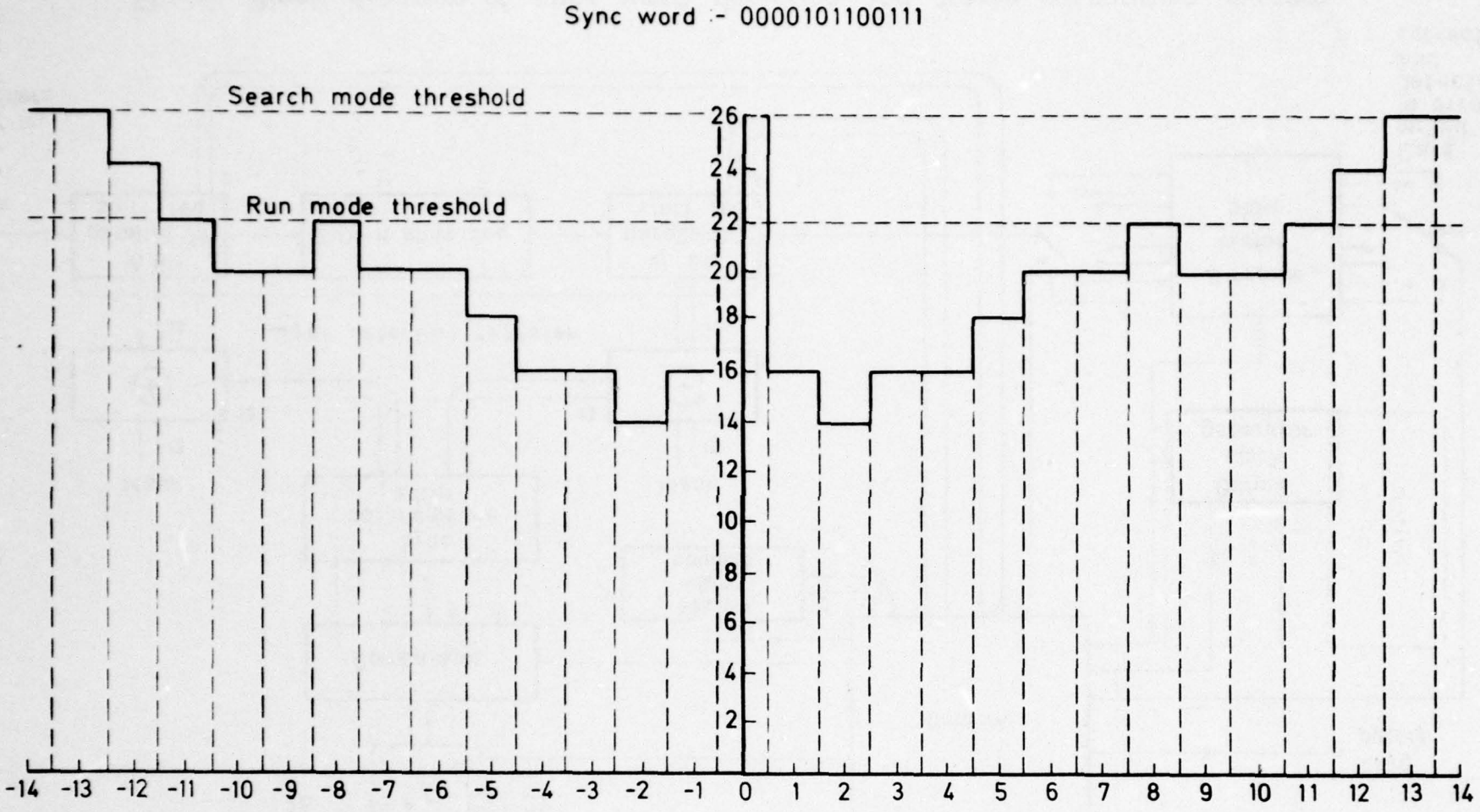
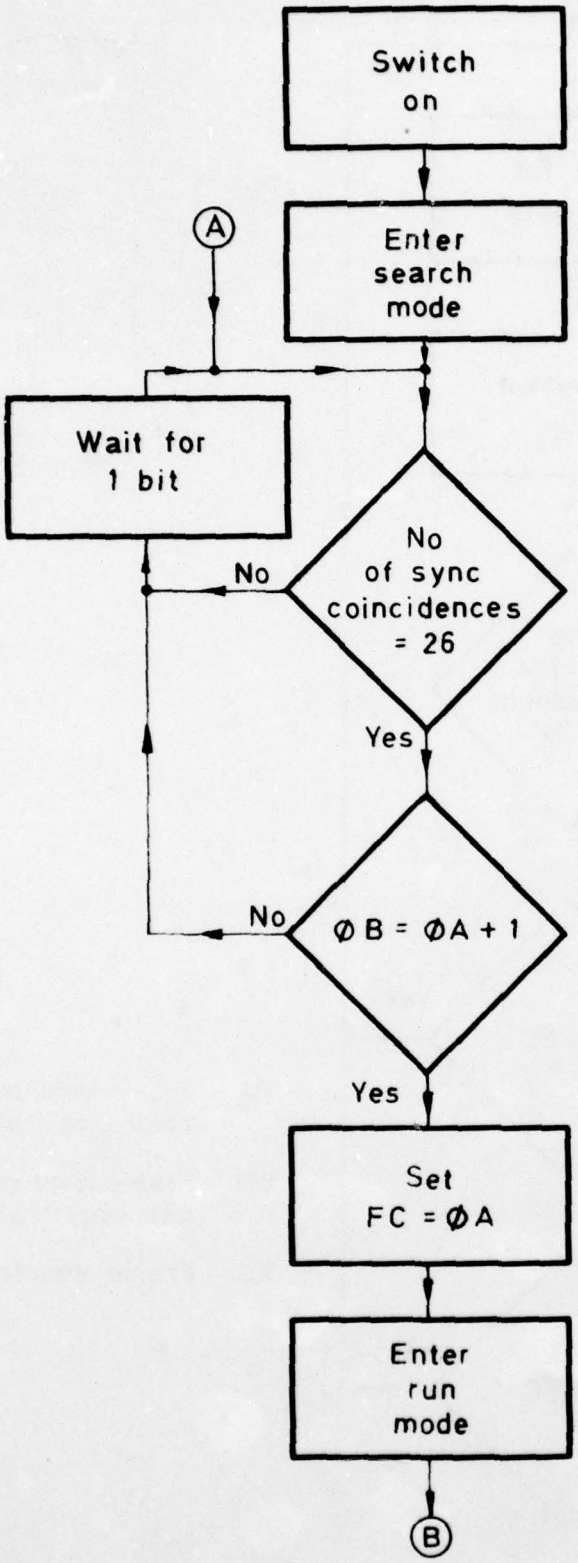


Fig 16 Histogram showing pattern of co-incidences between sync words and masks for worst case data



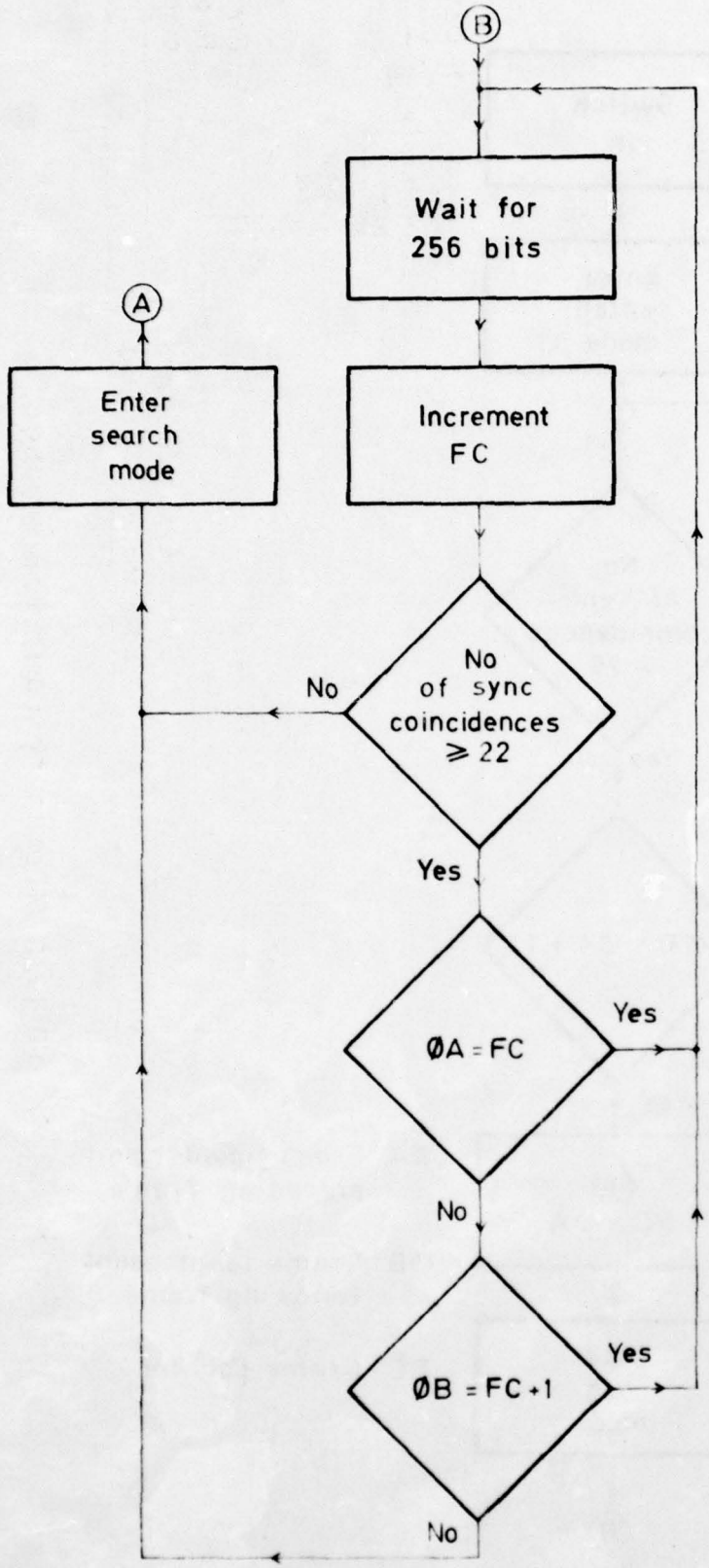
ØA Frame ident count preceding frame

ØB Frame ident count following frame

FC Frame counter

Fig 17a Flow chart for sync word detection (search mode)

Fig 17b



ØA Frame ident count preceding frame

ØB Frame ident count following frame

FC Frame counter

T Memo IT 171

Fig 17b Flow chart for sync word detection (run mode)

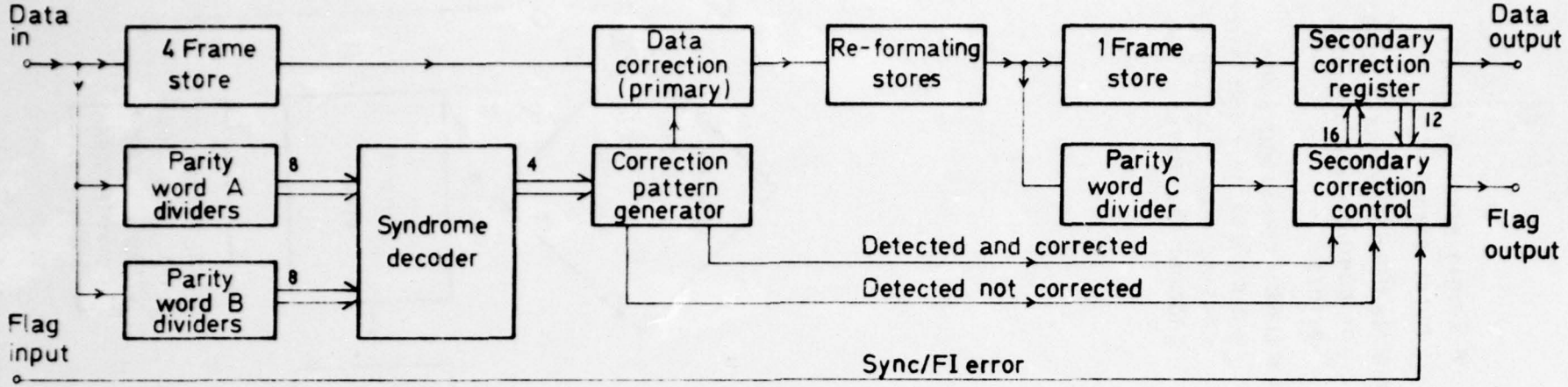


Fig 18 Block diagram of error detection and correction system

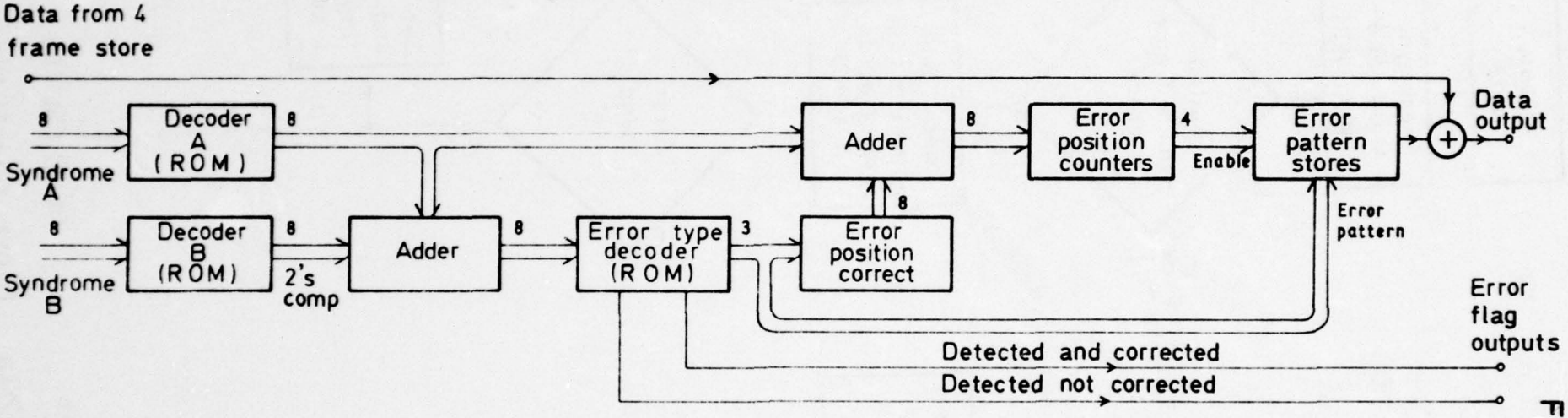
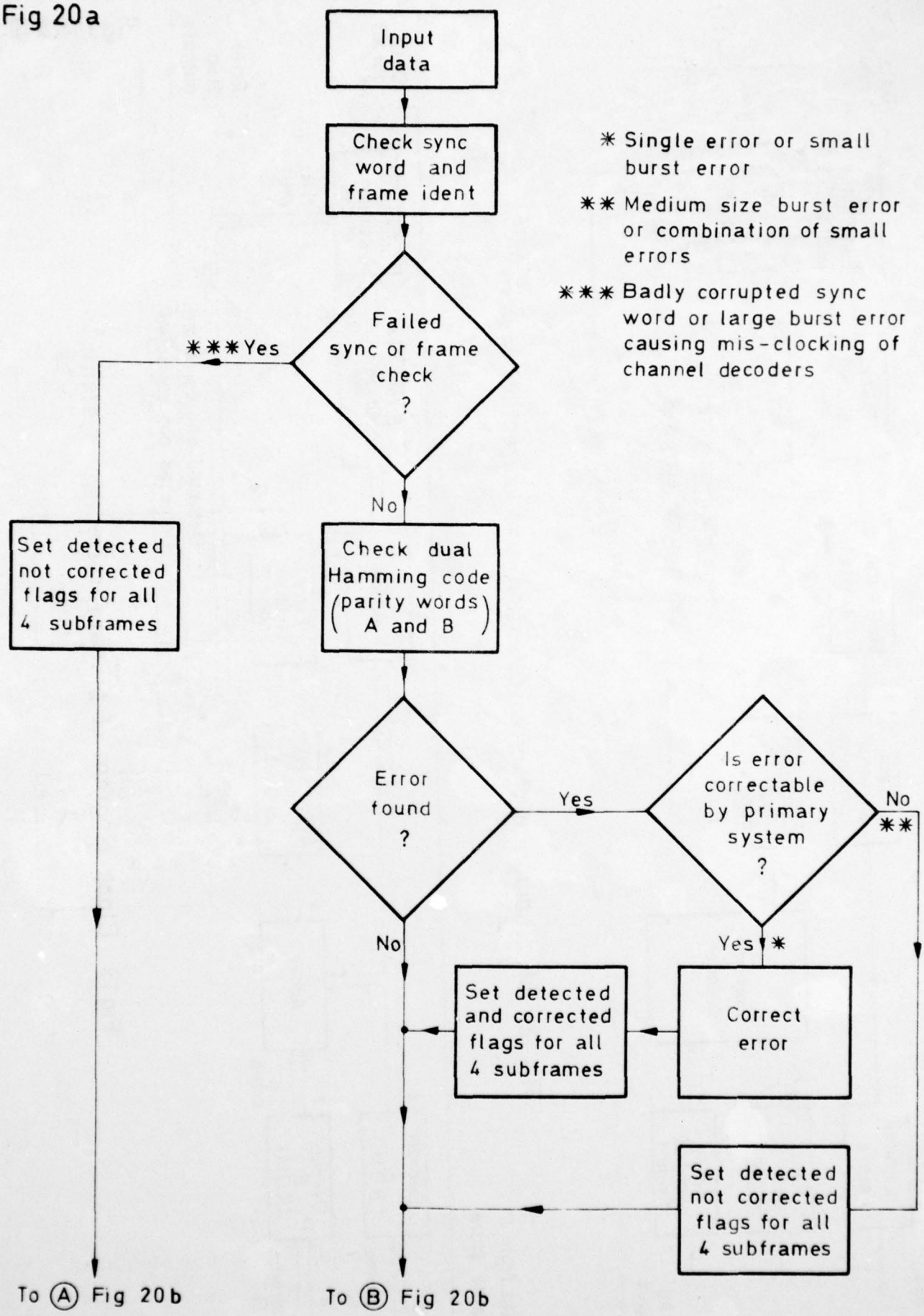


Fig 19 Block diagram of primary correction system

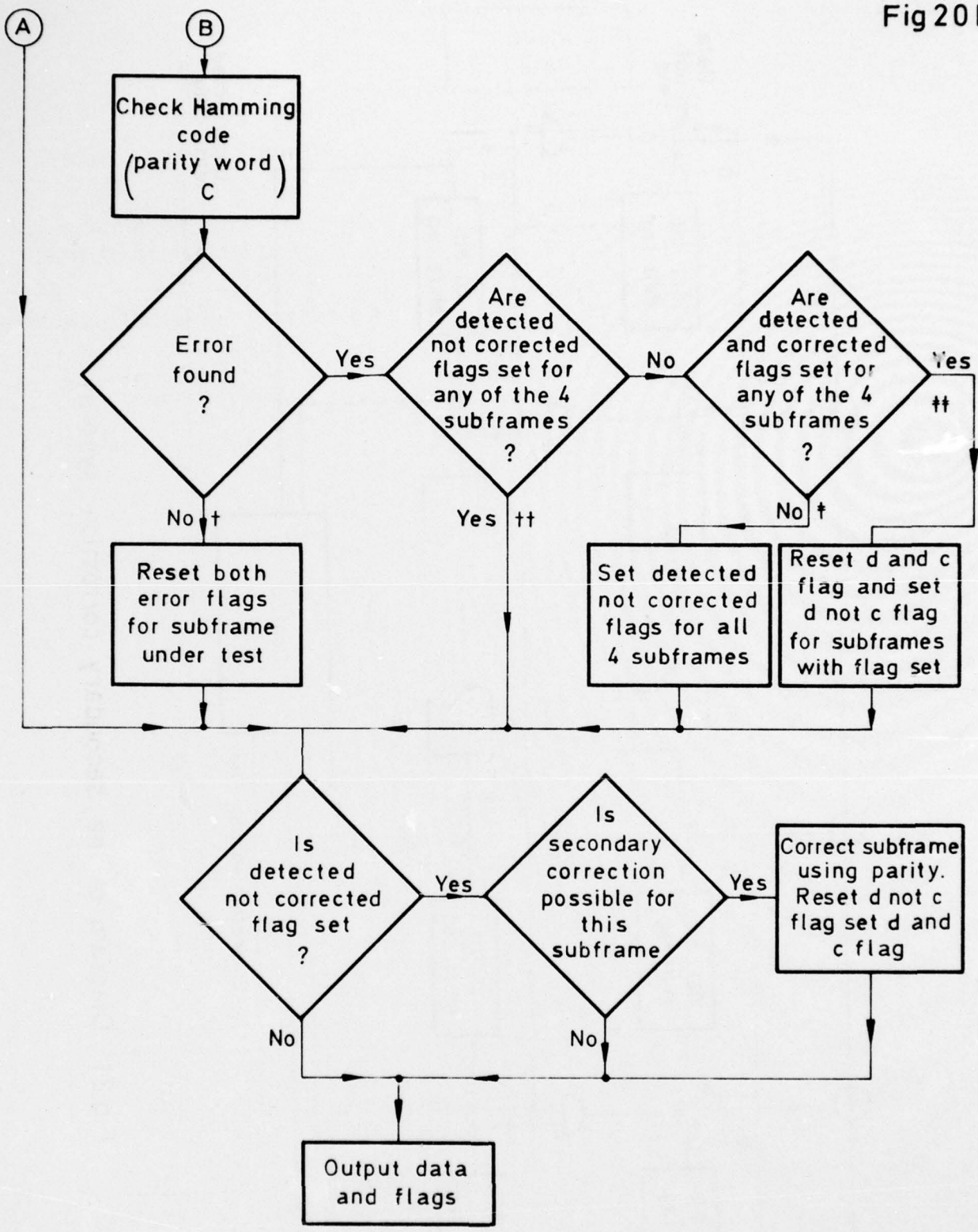
Figs 18&19

Fig 20a



T. Memo 11 171

Fig 20a Flow chart of error correction logic



T. Memo I T 171

† Assume entire frame is correct

†† Assume dual Hamming code has detected all errors

‡ Assume an error undetectable by dual Hamming code has occurred

‡‡ Assume dual Hamming code has applied wrong correction procedure

Fig 20b Flow chart of error correction logic

Fig 21

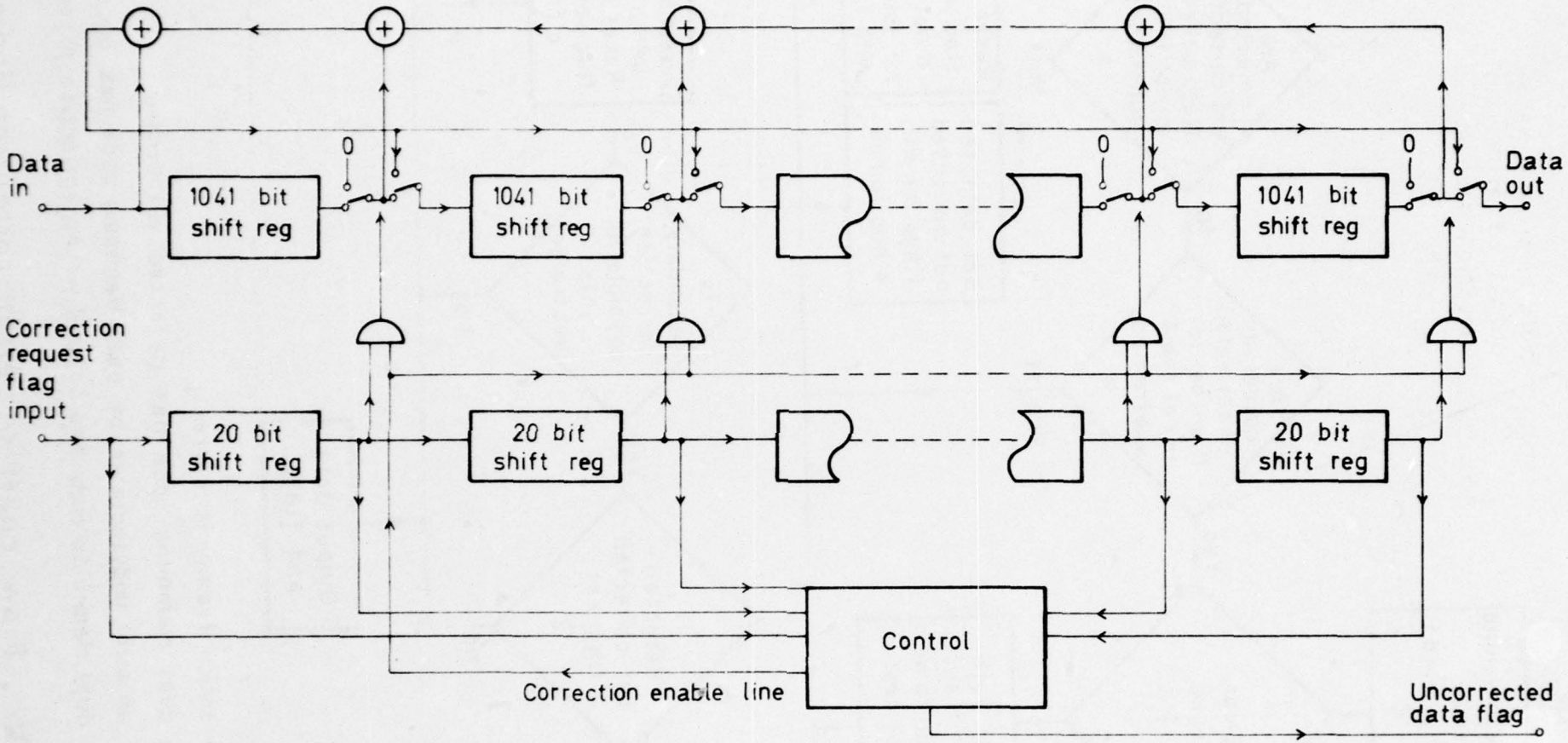


Fig 21 Diagram of the secondary correction system

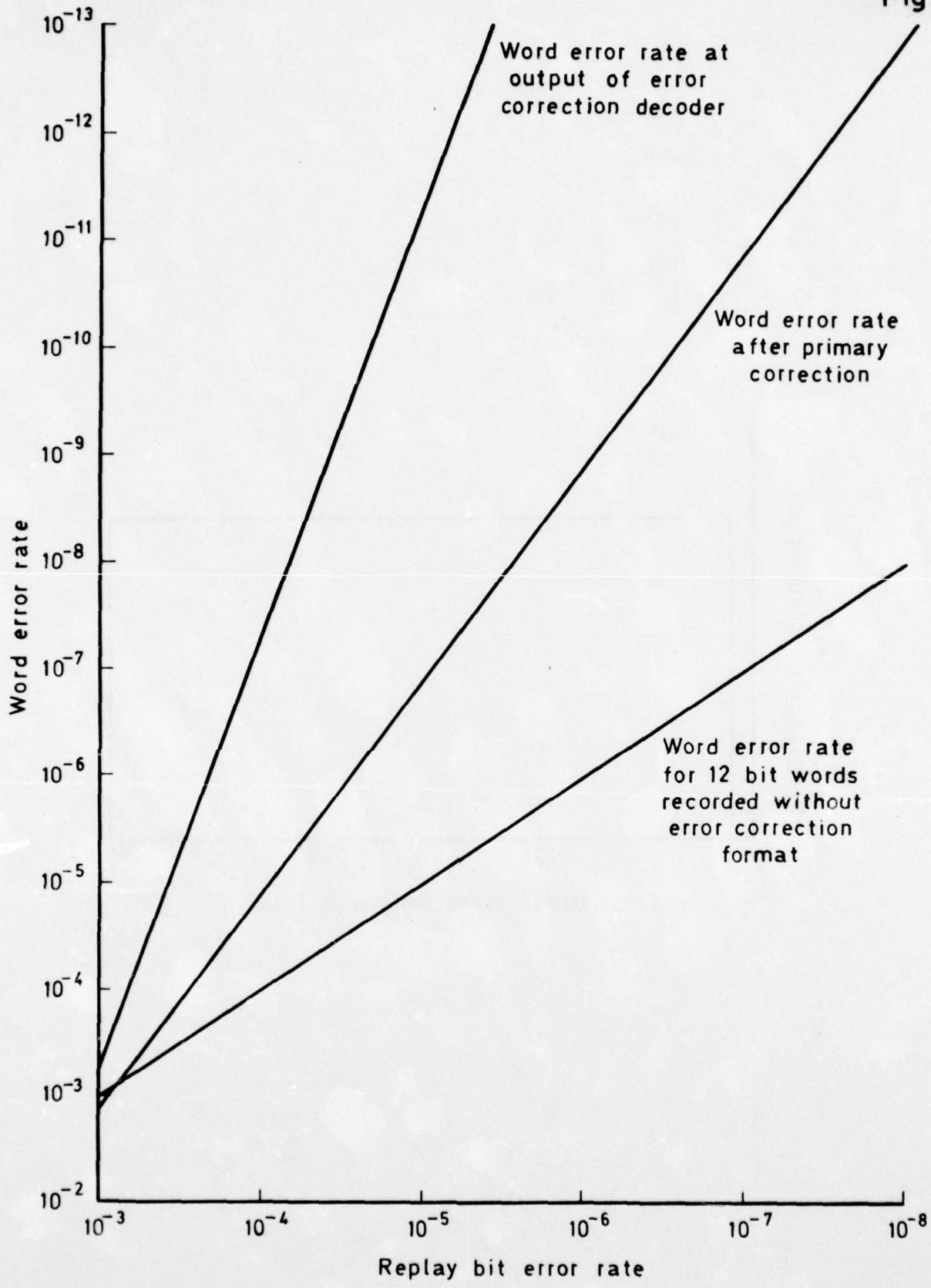


Fig 22 Theoretical response of system to random single errors

Fig 23

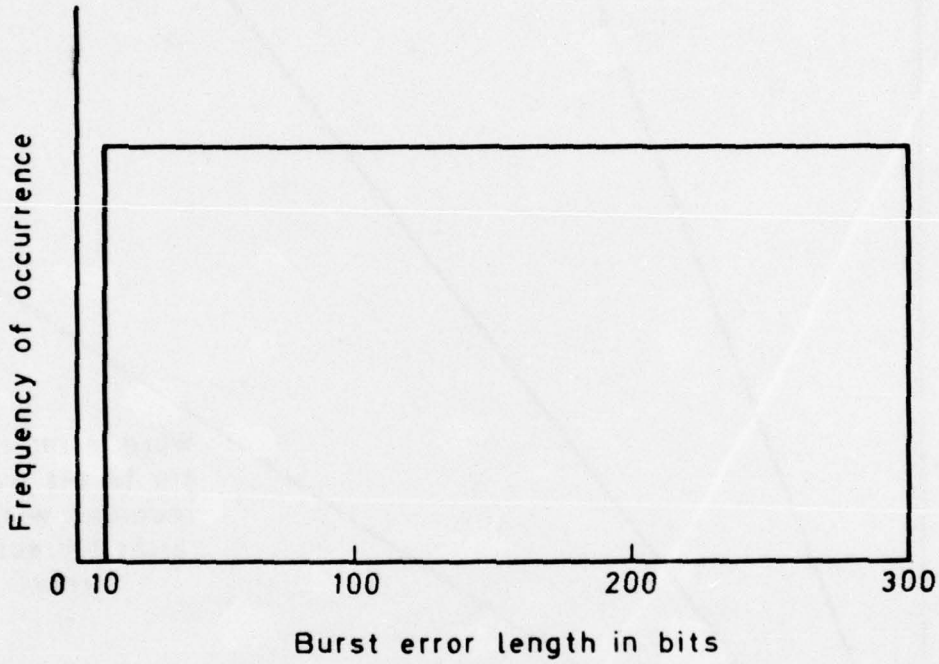


Fig 23 Assumed distribution of burst error lengths

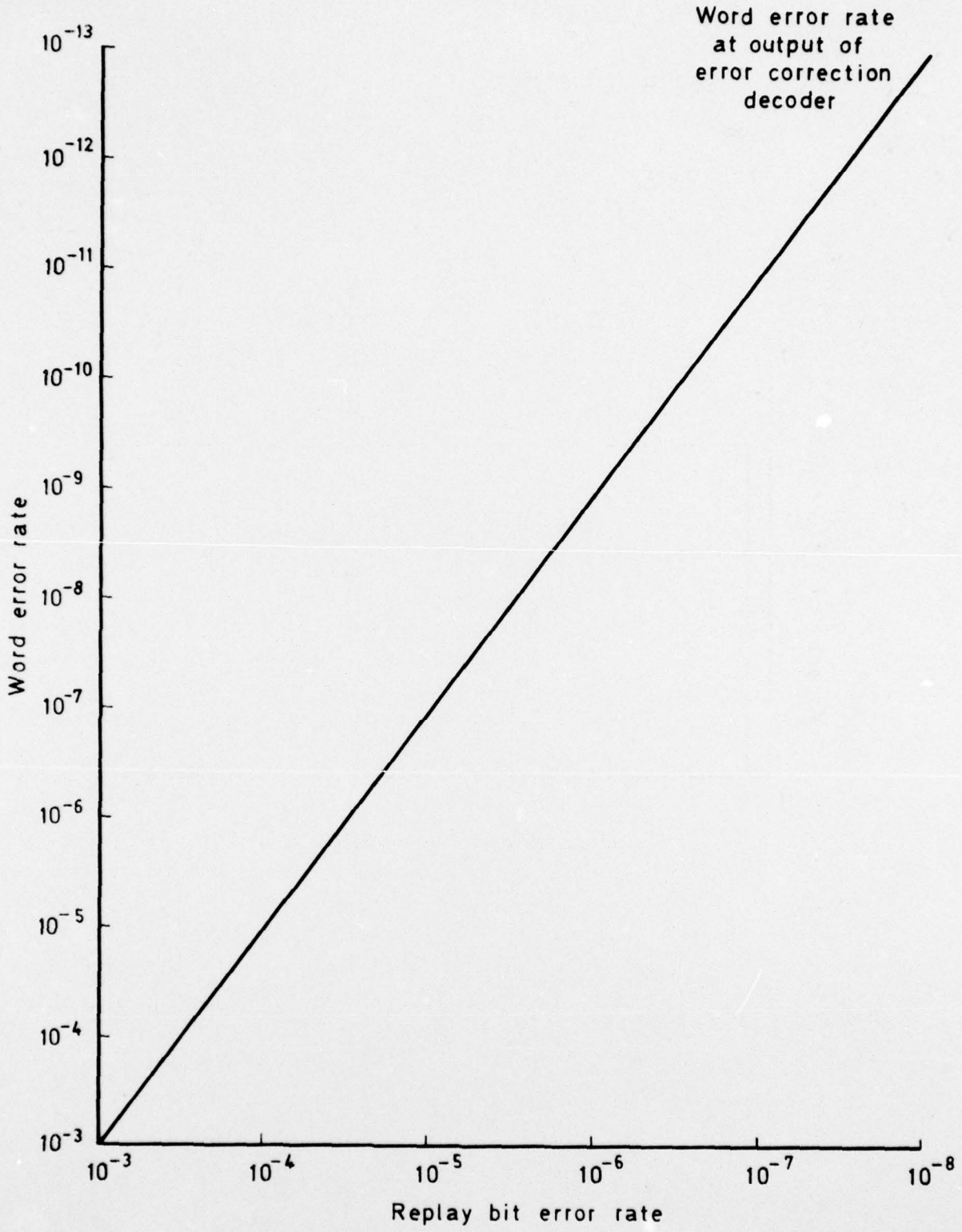


Fig 24 Theoretical response of system to burst errors

REPORT DOCUMENTATION PAGE

Overall security classification of this page

UNCLASSIFIED

As far as possible this page should contain only unclassified information. If it is necessary to enter classified information, the box above must be marked to indicate the classification, e.g. Restricted, Confidential or Secret.

1. DRIC Reference (to be added by DRIC)	2. Originator's Reference	3. Agency Reference	4. Report Security Classification/Marking
	RAE TM IT 171	N/A	UNCLASSIFIED

5. DRIC Code for Originator	6. Originator (Corporate Author) Name and Location
850100	Royal Aircraft Establishment, Farnborough, Hants, UK

5a. Sponsoring Agency's Code	6a. Sponsoring Agency (Contract Authority) Name and Location
N/A	N/A

7. Title: The design of an error detection and correction system for use with single track airborne digital magnetic recording systems.

7a. (For Translations) Title in Foreign Language

7b. (For Conference Papers) Title, Place and Date of Conference

8. Author 1. Surname, Initials	9a. Author 2	9b. Authors 3, 4	10. Date	Pages	Refs.
Jackson, C.W.			April 1978	57	7

11. Contract Number	12. Period	13. Project	14. Other Reference Nos.
N/A	N/A		

15. Distribution statement

(a) Controlled by – Unlimited

(b) Special limitations (if any) –

16. Descriptors (Keywords) (Descriptors marked * are selected from TEST)

Digital recording error detection and correction. Single track digital recording. Dual Hamming codes. Extended Hamming codes.

17. Abstract

The Memorandum describes the types of error that can occur in airborne digital magnetic recording systems, and the problems associated with the correction of these errors on medium and high packing density single track systems. Some methods whereby record/replay errors on single track systems may be detected and corrected are explained, and a description is given of a system using these principles.

F5910/1

RAE Form A143