FAA-RD-75-61

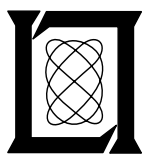
Project Report
ATC-48

DABS Downlink Coding

J. T. Barrows

12 September 1975

Lincoln Laboratory
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LEXINGTON, MASSACHUSETTS



Prepared for the Federal Aviation Administration,
Washington, D.C. 20591

This document is available to the public through
the National Technical Information Service,
Springfield, VA 22161

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
FAA-RD-75-61		

4. Title and Subtitle	5. Report Date
DABS Downlink Coding	12 September 1975
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
J. T. Barrows	ATC-48

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS) 45364 Project No. 034-241-021
Massachusetts Institute of Technology Lincoln Laboratory P. O. Box 73 Lexington, Massachusetts 02173	11. Contract or Grant No. IAG DOT-FA-72-WAI-261
	13. Type of Report and Period Covered Project Report
12. Sponsoring Agency Name and Address Department of Transportation Federal Aviation Administration Systems Research and Development Service Washington, D.C. 20591	14. Sponsoring Agency Code

15. Supplementary Notes

The work reported in this document was performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology under Air Force Contract F19628-76-C-0002.

16. Abstract

This report summarizes the encoding/decoding techniques incorporated into the DABS downlink design. It has been determined that the binary cyclic code used for the uplink is applicable to the downlink interference environment, thus rendering considerable hardware simplification in the transponder. The downlink environment leads to a characterization as a burst erasure channel, which allows the code to be used in a burst correction capacity.

This report concentrates on the error correction techniques including specific implementations. Evaluation of the performance of the code by simulation and/or bench test is presently being carried out and will be reported in a future document.

17. Key Words	18. Distribution Statement
Cyclic codes Burst error correction Erasures	Document is available to the public through the National Technical Information Service, Springfield, Virginia 22151.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	
Unclassified	Unclassified	40	

TABLE OF CONTENTS

ILLUSTRATIONS

I

INTRODUCTION

The Discrete Address Beacon System (DABS) will provide an evolutionary upgrading of air traffic surveillance capability together with an integrated ground/air data link. Both features are required to support the planned automation of air traffic control.

DABS includes a unique code as part of each interrogation which allows the ground sensor to address each aircraft individually so as to control the timing of replies from neighboring aircraft. This eliminates the self-interference due to overlapping replies (synchronous garble), which is a basic limitation of the present Air Traffic Control Radar Beacon System (ATCRBS). By providing for the inclusion of a message as part of an interrogation or a reply, data link communications can be accommodated on the same channel with a small increase in equipment complexity.

The major factors influencing the design of the DABS signal formats are: (1) the need to attain a reliability of service commensurate with the projected surveillance and communications demands of an automated ATC system, (2) electro-magnetic compatibility with ATCRBS to allow both systems to operate together in an extended period of evolution from ATCRBS to DABS, and (3) minimization of transponder cost to speed the transition from DABS to ATCRBS. Simply stated, DABS must operate reliably in a severe interference environment without degrading other systems.

In view of these requirements, a study was initiated to determine the costs/benefits resulting from the incorporation of error control techniques into both the DABS uplink and downlink messages. This report summarizes the results of the coding investigation for the downlink and is a continuation of a companion report [Ref. 1] detailing the coding analysis and results for the uplink. Of major importance was the decision to utilize the same shortened cycle code for the downlink as used for the uplink with a resulting simplification of transponder hardware. In view of this, the reader is referred to the uplink report for a review of the properties of the code and basic implementation considerations.

In the following paragraphs, the factors that impact on downlink coding considerations are first reviewed. It will be shown that these factors lead to a characterization of the downlink as a burst erasure channel. This, in turn, leads to the potential for maximum utilization of the error correction capability of the selected code.

Then the decoding and error correction techniques are discussed including specific implementations that are driven by the requirements and constraints of the DABS downlink configuration. The underlying thought in all the analysis and subsequent implementation was to achieve a (limited) correction capability to increase message throughput without measurably compromising the message reliability that is inherent in the code if used in a detection-only mode.

II

DOWNLINK CHANNEL CHARACTERISTICS IMPACTING ON ERROR CONTROL DESIGN

The same general constraints and requirements, which drove the investigation of coding techniques for the DABS uplink, also apply to the downlink. However, the emphasis is somewhat different for the downlink. In particular, the uplink decoder is limited to an error detection-only capability largely because of the need for inexpensive transponders operating in real time. These restrictions are of secondary importance in the downlink decoder (ground based) due to an inherently larger computation capability and the possibility of off-line processing. Thus, error correction techniques become a viable consideration for the improvement of throughput by reducing the incidence of rejected messages (containing errors). Other factors that facilitate an error correction capability for the downlink are the signaling technique, data rate, monopulse/receiver processing and the resulting structure or characteristics of transmission errors. These factors are summarized below and lead to a characterization of the downlink as a burst erasure channel.

II.1 Signaling Format

The center frequency of the DABS reply transmission is 1090 MHz, which is the same as for the ATCRBS reply. The downlink signaling will use

Nonreturn-to-Zero, Pulse Amplitude Modulation (NRZ/PAM) for the message field with a chip rate of 2 megachips per second. A pulse position modulation (PPM) format is used where each chip is delayed by one chip duration, complemented and interleaved with the original chip stream. This results in an effective data rate of 1.0 megabit per second. Two immediate advantages of this complemented PPM format are (1) a single ATCRBS reply pulse (0.45-μsec width) will not affect both chips comprising a single data bit in the same way, thus increasing the detectability of interference, and (2) the complementing ensures the presence of as many "ones" as there are data bits for use in a monopulse estimate.

The downlink message shown in Fig. 1 will contain 56 bits for all-call, Synchro-DABS or surveillance replies, and 112 bits for communications replies. The message (excluding preamble) will be systematically encoded using the same 24-bit parity check code as in the uplink, with the parity bits overlayed on the 24-bit address field.

II.2 Downlink Interference Sources

The major source of interference for the DABS downlink is the ATCRBS reply which operates at a center frequency of 1090 MHz. The ATCRBS reply signal characteristics are shown in Fig. 2 and are composed of 0.45-μsec duration PAM pulses spaced 1.45 μsec apart. Pulses F_1 and F_2 are framing pulses separated by 20.3 μsec which are always present; the X pulse is never present and the remaining 12 interior pulses are used to make up the responses to an ATCRBS Mode A or Mode C interrogation. The last pulse is a special purpose (pilot) indicator (SPI) that is transmitted only on initiation by the

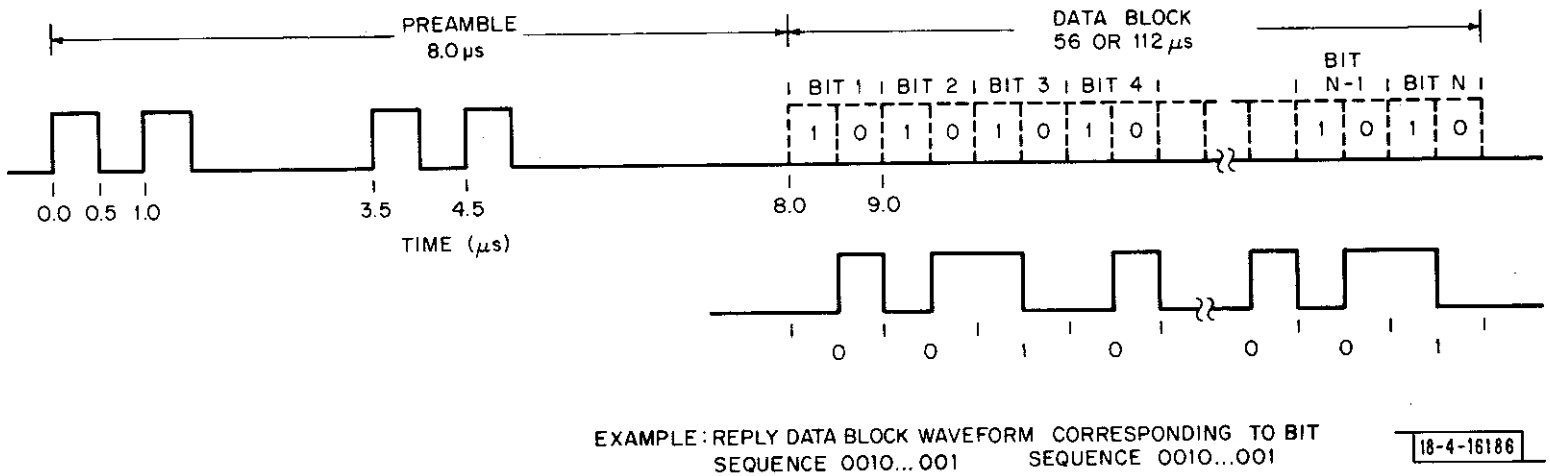


Fig. 1. DABS reply format.

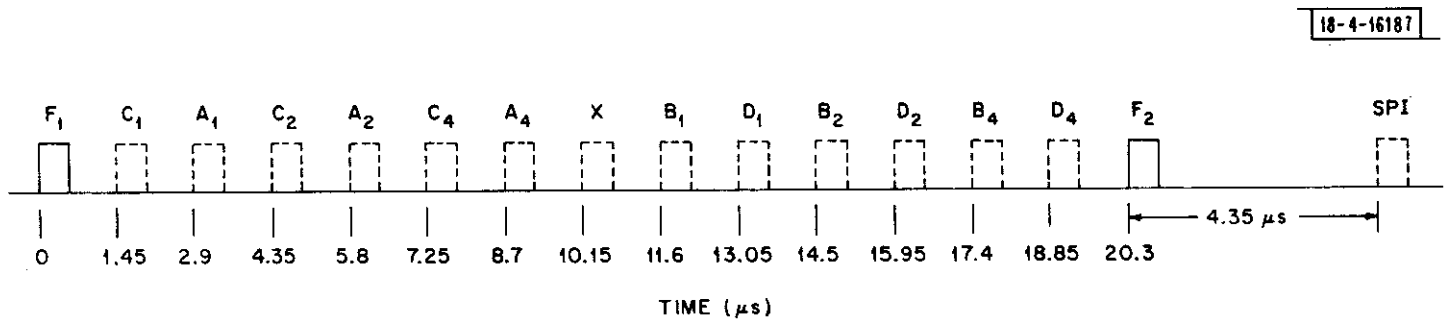


Fig. 2. ATCRBS transponder reply codes (downlink).

pilot. The important thing to note is that a single fruit* reply without an SPI pulse will adversely affect a span of less than 24 DABS bits. Moreover, the use of the SPI is not a frequently occurring event. Thus, this predominant interference source gives rise to a burst error channel. Furthermore, by using the 24-bit parity check code, this single fruit reply cannot create undetected errors in the DABS message.

Another likely source of interference arises from TACAN interrogation pulse pairs having a carrier frequency close to 1090 MHz (there exist some special use military TACANS using 1090 MHz). These interrogations have two 3.5 ± 0.5 μsec pulses separated by 12.0 ± 0.5 μsec and, again, are seen to result in burst errors spanning less than 24 DABS bits. Other interference sources such as multipath also lead to bursts of errors in the DABS message.

II.3 Summary of the DABS Receiver/Processor Features Relevant to DABS Coding

The DABS reply processing utilizes monopulse techniques that produce three data channels from the PPM formatted NRZ-PAM chip stream. These channels are the log sum, monopulse angle, and interference detection channels whose outputs are digitized by three A/D converters. Also the video outputs of the log sum and log omni (from the omni receiver) channels are compared and quantized. Thus a bit decision is based on

(1) Quantized video output wherein the two chips corresponding to a single bit are compared

(2) An amplitude estimate

*"Fruit" will be used in the sequel primarily to describe a nonsynchronous ATCRBS reply message overlapping a DABS message received at the interrogator/processor.

(3) An azimuth estimate

(4) An interference estimate (from difference beam).

A "confidence" measure of the bit decision process can be derived by using one or more of these estimates that can be used as added information in designing an error correction scheme. In particular, by flagging bit decisions in which a low confidence estimate is made, the input to the decoder can be characterized as an erasure channel. By changing the parameters associated with the confidence measure, the decoder input can be varied from a mixed errors-and-erasures channel to an essentially erasures-only channel. The trade-offs associated with this capability are beyond the scope of this report, and for the purposes of describing the decoder/correction possibilities it suffices to say that an erasures-only channel can be assumed by being sufficiently conservative in defining the confidence measure.

III

DOWNLINK ERROR CONTROL

Quite early in the coding study, a decision was made to utilize, if at all possible, the same code for the uplink and downlink. This would effect a measurable saving in transponder hardware and complexity. In realizing this goal it was necessary to determine whether or not a single code could be found that would yield satisfactory performance in the two different interference environments of the uplink and the downlink. After analyzing the 24-bit parity check code used for the uplink and assessing the downlink interference environment, it was determined that this same code would suffice for the downlink. The summary properties of the selected code are listed in the uplink report but are repeated here for convenience. This code is a binary cyclic code defined by the generator polynomial

$$g(z) = 1 + x^3 + x^{10} + x^{12} + x^{13} + x^{14} + x^{15} + x^{16} + x^{17} + x^{18} + x^{19} + x^{20} + x^{21} + x^{22} + x^{23} + x^{24}$$

$$= (1 + x)(1 + x^2 + x^4 + x^5 + x^6)(1 + x + x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^{10} + x^{13} + x^{15} + x^{16} + x^{17})$$

which has a natural length of about 2.75 million bits ($e = 21(2^{17} - 1)$). This code has been shortened to a length of $n = 112$ bits for the DABS message format and to $n = 56$ bits for the DABS all-call and surveillance replies.

This code was first discovered by Kasami (1964) and was determined to be able to correct any single 12-bit error burst or less in a code length of 72 bits (or it can correct any two erasure bursts each of 12 bits or less). Alternatively, this code guarantees detection (correction) of any single 24-bit error (erasure) burst over any length.

Although the actual Hamming distance or general burst distance properties have not been analytically determined, a fairly extensive exercising of this code over a word length of approximately 100 bits indicates that

- The Hamming distance is 6.
- The 4-bit burst distance is 4 over a code length of at least 100 bits.
- The 9-bit burst distance is at least 3 over 89 bits (in other words, any occurrence within 89 bits of two nine-bit error bursts is guaranteed detectable).
- The 10-bit burst distance is 3 over a code length of 89 bits.
- The 11-bit burst distance is 3 over a code length of 75 bits.
- The 12-bit burst distance is 3 over a code length of 72 bits.

III. 1 Encoding

One obvious advantage of using the same code for both uplink and downlink messages is that the same shift register used for decoding in the transponder can also be used for encoding the downlink message. Moreover, the transponder encoding circuit shown in Fig. 3 is essentially the same as the (ground based) encoder used for the uplink. However, there is a slight change from the uplink encoder in that in the present case the parity bits are directly overlayed on the address field. Thus in Fig. 3, gates G-1 and G-2 are enabled,

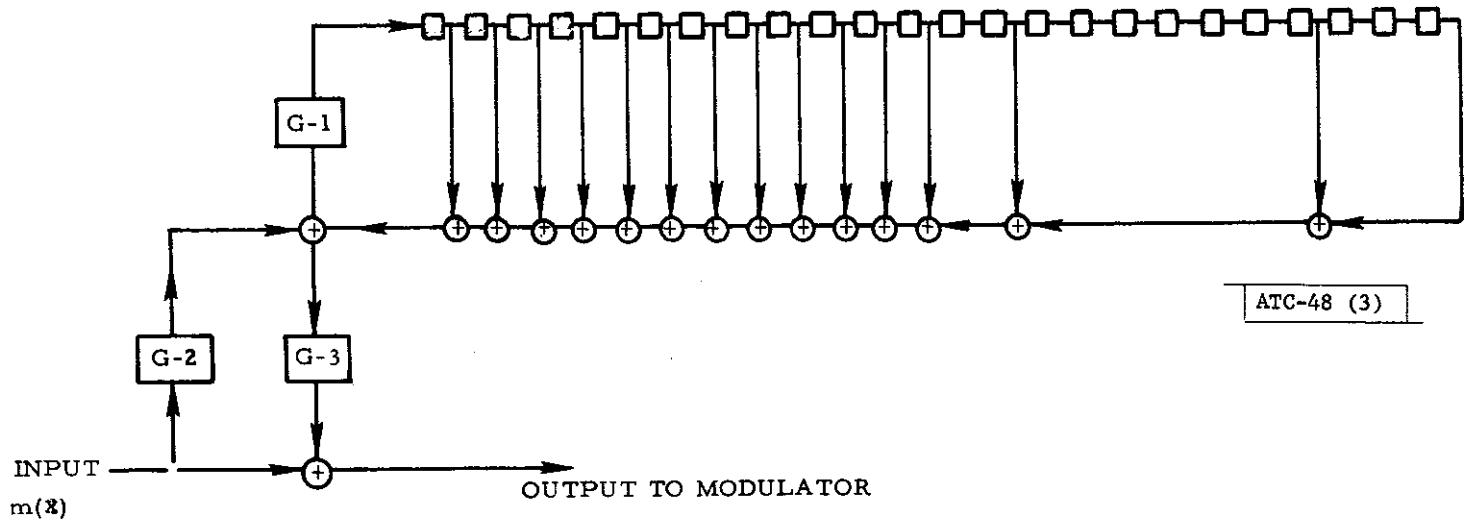


Fig. 3. Transponder encoder.

and G-3 is disabled during the first k-bit shifts. This allows the information bits to be input to the division circuit and simultaneously outputted unaltered to the downlink modulator. Thence for the next r = 24 shifts, gates G-1 and G-2 are disabled, and G-3 is enabled which causes the generated parity bits to be overlayed on the 24 address bits (added modulo -2, bit by bit). This is all that is necessary to be said regarding the transponder encoder. For more details the reader is refered to ATC-49 [Ref. 1]. The remainder of this report is devoted to the decoding considerations with emphasis on the correction of burst errors.

III. 2. Decoding for the Downlink: Burst Erasure Correction

A general block diagram for the downlink decoding process is shown in Fig. 4. There are two aspects to this process: the syndrome (remainder) calculation/address-compare, which is performed in real time, and then the correction process, which requires up to n more clock units (n is the length of the coded message) but can be performed faster than real time.

As mentioned in Section II. 3, the receiver/processor has the capability of ascribing a confidence measure to individual bit decisions made in the detection process. Thus there are two binary sequences input to the decoding circuitry. The demodulated binary data stream is fed into the division register (divide by g(x)) which calculates the syndrome in real time; it is also input to a storage register for later use in the correction process. After the last data bit is entered, the syndrome is checked against the appropriate address (as determined by the scheduling/interrogation protocols). If the address check is satisfied, the process can be terminated by reading out the stored message

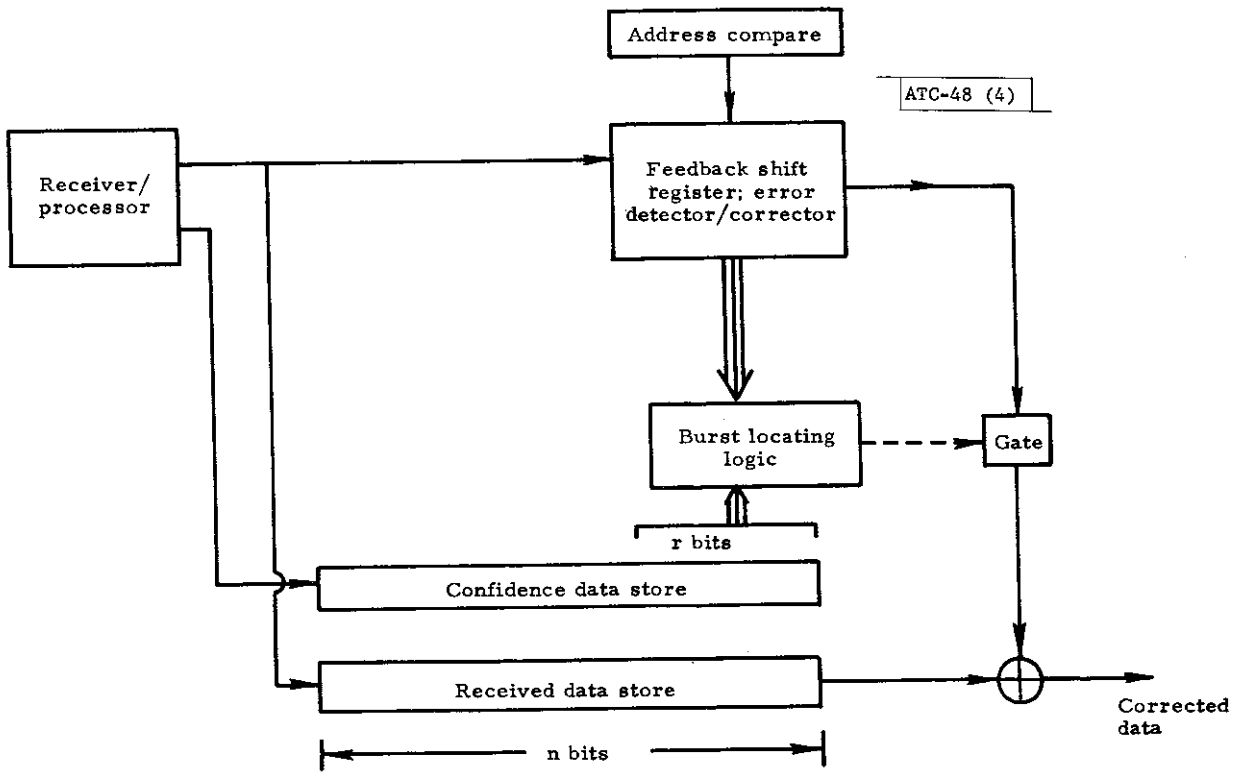


Fig. 4. Downlink decoding process.

bits to the display units. If the address does not check, there are several possible causes: (a) A transmission was received from the addressed aircraft, but with errors, (b) a transmission was received from some aircraft other than the one addressed, and (c) a combination of (a) and (b). Assuming that the sensor is in a roll-call mode (as opposed to all-call), the decoder assumes that case (a) has occurred and the assumed address is subtracted from the calculated syndrome/address sequence. (Alternately the address can be subtracted from the sequence before the division process, in which case the syndrome is tested against the all-zero sequence.) The resulting 24-bit remainder is used to initialize the correction process in the second phase of operation.

A confidence sequence is also outputted from the receiver/processor into a storage register to be used in the correction process. There are several methods of utilizing this extra information in the correction process, which are detailed in a later section. Suffice it to say at this point that the confidence sequence is used to determine if a correctable erasure burst is present in the received data and, if so, locate its position within the message.

In the following paragraphs, first the concept of burst erasure trapping is briefly discussed without regard to the specific implementation problems imposed by the DABS downlink message or the receiver/processor design. Following this, the utilization of the confidence sequence is discussed and shown to be an efficient but conservative locator of a correctable erasure burst. Then the complete set of decoding steps is summarized assuming the use of a full length cyclic code. Five specific implementations are then presented which focus on the use of the proposed shortened cyclic code. Finally, brief

mention is made of the technique used to decode (and encode) all-call messages. Since the performance of this correction process is dependent upon the design of the receiver/processor and the nature of the interference, results are being obtained by simulation and bench testing and will be presented in a separate report.

III. 2. 1 Burst Error Trapping: Correction of a Single 24-Bit Erasure Burst

After the last data bit has been input to the division register and the address has been subtracted out, the register contains the syndrome, which is a function only of the errors occurring in transmission. This 24-bit sequence is then used to initiate the burst trapping/error correction process.

Let $x^j B(x)$ denote a burst of errors (erasures) starting in the $(j + 1)$th bit of the code word, $c(x) = Q(x)\, g(x)$, and spanning not more than $r(=24)$ bits. The received sequence is then

$$\hat{c}(x) = c(x) + x^j B(x) \qquad (1)$$

where the location (j) and span of the burst are known from the confidence (erasure) sequence. The only thing not known is the structure of the burst. After dividing $\hat{c}(x)$ by $g(x)$, the syndrome (remainder), $S_0(x) = (s_0^0, s_1^0, \ldots, s_{r-1}^0)$, is obtained as

$$S_0(x) \equiv x^j B(x) \quad \text{modulo } g(x) \qquad (2)$$

where $\equiv$ denotes congruence. In the special case where $j = 0$, this division yields $S_0(x) = B(x)$, i. e., in the case where the burst is totally confined to the

parity bits, the syndrome is not just a function of the erasure burst, but is the burst itself. From this point it is a simple matter to correct the errors in the received sequence that has been stored.

Now suppose that $j > 0$. In this case the cyclic property of the code is utilized where $x^{e-j} \equiv x^{-j}$, modulo $g(x)$. In particular, starting with $S_0(x)$, and shifting the division register $e - j$ more times (with feedback but no added input) yields

$$S_{e-j}(x) \equiv x^{e-j} S_0(x) \equiv x^e B(x) \equiv B(x), \quad \text{modulo } g(x) . \tag{3}$$

But since both $S_{e-j}(x)$ and $B(x)$ have degree less than r, the congruence yields an equality, i.e., $S_{e-j}(x) = B(x)$. Now the structure of the error burst is exposed and can be subtracted out from the appropriate bit locations within the stored received sequence. Thus by shifting the contents of the register (which implies a multiplication by x for each shift and then a division by $g(x)$) a total of $e - j$ times, the error burst has been trapped in the parity locations of the shifted sequence.

This correction process can be implemented directly using a full length cyclic code ($e = n$), i.e., for each extra shift of the division register, a stored message bit is shifted out to the user until the error burst is exposed in the register and simultaneously located at the right end of the buffer (see Fig. 4). At this point, the division process is terminated and bit-by-bit correction of the burst takes place (gate is enabled) as the sequence is shifted out of the buffer. The only important requirement for this (automatic) sequential correction process is that the contents of the r-stage shift register must "line up" with

the corresponding r message bits in the right end of the buffer. This, in turn, requires a premultiplication of the message sequence by x^r before entering the division register. To see the necessity of this, let the received message be described as $\hat{c}(x) = \hat{c}_0 + \hat{c}_1 x + \hat{c}_2 x^2 + \ldots + \hat{c}_{e-1} x^{e-1} = (\hat{c}_0, \hat{c}_1, \ldots \hat{c}_{e-1})$ where the "hat" connotes possible bit errors. After the last message bit has been entered, the r bits in the right side of the message buffer in Fig. 4 are $(\hat{c}_{e-r}, \hat{c}_{e-r+1}, \ldots \hat{c}_{e-1})$ from left to right. However, without premultiplication, the contents of the shift register are associated with the r low-order bits of the code word (the parity field). Thus, since $x^e \equiv 1$ modulo $g(x)$, premultiplication of $\hat{c}(x)$ by x^r effectively shifts the high-order bits of the code word around to the low-order end, i.e.,

$$x^r \hat{c}(x) = \hat{c}_o x^r + \ldots + \hat{c}_{e-r-1} x^{e-1} + (\hat{c}_{e-r} x^e + \ldots + \hat{c}_{e-1} x^{e+r-1})$$

$$\equiv (\hat{c}_{e-r} + \ldots + \hat{c}_{e-1} x^{r-1}) + \hat{c}_o x^r + \ldots + \hat{c}_{e-r-1} x^{e-1}$$

and the corresponding bits are now aligned for the division process and subsequent correction.

III. 2. 2. Location of a Correctable Erasure Burst; Use of Confidence Sequence

Conceptually, the confidence sequence simply locates a correctable pattern of errors (i.e., a subsequence of low confidence bit decisions spanning no more than r = 24 bits). One way to implement this would be to monitor the confidence sequence as it enters the storage buffer and bracket a 24-bit interval beginning with the first low confidence bit that appears. Then any occurrence

of other low confidence bits outside this interval causes the message to be flagged as containing detectable but uncorrectable errors. In this event if the address check is satisfied, the message can still be acceptable; however, no correction is attempted on the message.

In practice, there may be numerous low confidence bits scattered throughout the message with or without bit inversions in the associated message bit decisions. This phenomenon is a function of the level of interference and the decision criteria used in the receiver/processor which is beyond the scope of this report. Suffice it to say that there are frequent cases wherein correct message bit decisions are made with high probability but labeled as low confidence (the presence of sidelobe fruit being one example), and it would be advantageous to allow such a message to be processed by the correction circuitry. Thus the presently proposed burst erasure locating technique tests the composition of each 24-bit segment of the confidence sequence (at the output end of the buffer during the correction process) against the corresponding 24 bits in the division register, and declares a correctable burst at the first occurrence of a match (see Fig. 5). This match is defined as

(a) The 1 (s) (errors) in the devision register match one-to-one with low confidence indications.

(b) No error is matched with a high confidence indicator.

However, there are dangers associated with this method which arise from two basic sources:

(1) There is an increased probability of undetected message error due to an incorrect correction of a message that is inherent in any correction process and is aggravated by this sequential erasure burst location process.

(2) Because of the address/parity overlay, there is the possibility of accepting a misdirected message having no errors but the wrong address (i.e., an incoming message that is not expected in that prescheduled time slot). In this case adding the wrong address into the address/parity locations will result in a burst of "errors" at the end of the message.

Both of these events can be illustrated by the same, albeit extreme example: if, for some reason there exists a 24-bit interval containing all low confidence indicators, the correction logic will always recognize this as a correctable burst since both of the above criteria will be automatically satisfied. The same potential arises with decreasing probability for lower but still high density clusters of low confidence bits. To guard against this eventuality, with its high likelihood of incorrect correction, a density sensing circuit is placed at the head of the confidence buffer and tests the number of occurrences of low confidence decisions in each 24-bit interval against a preset threshold as the buffer is being loaded (see Fig. 5). This circuit is implemented by a simple up/down counter with the threshold adjustable to any integer, T, between 4 and 14 low confidence bits (or equivalently between 10 and 20 high confidence bits). If the preset number T is exceeded at any time during the loading process, a flag is set and the correction circuitry is disabled.

A decision as to what the actual threshold should be depends on the decision process of the receiver/processor and the interference characteristics that can be determined only by simulation or testing in a realistic environment. The range of 4 to 14 is a somewhat arbitrary but conservative choice and is based on limiting the a posteriori probability of occurrence of such a random event to between 10^{-3} and 10^{-6}.

III.2.3. Decoding Steps for Full Length Cyclic Code

The complete set of steps in the decoding process is now summarized as well as illustrated in Fig. 5, using the r = 24 bit DABS code with full length encoding/decoding (i.e., n = e).

(1) The entire demodulated data sequence is input into the high-order end (premultiplication by x^{24}) of the division register and simultaneously stored in the e-stage message buffer. The assumed address is "stripped" from the last 24 bits of the input sequence before division. If no errors have occurred and the assumed address is the correct one associated with this message, the content of the register S_0 after the last bit has entered is the all-zero vector. In this event, the stored message is forwarded immediately to the user.

(2) Simultaneous with the above, the sequence of confidence bits outputted from the receiver/processor is being stored for use in the correction process. During this loading time, this sequence is being checked against the maximum allowable density of low confidence declarations in every 24-bit interval. If the syndrome, S_0, is nonzero and the density criterion has been exceeded anywhere within the confidence sequence, the message is flagged as having detectable but uncorrectable errors, and the process terminates without any attempt at correction.

(3) If the syndrome is not the all-zero sequence and the confidence bit density flag has not been set before the last bit is input, the correction process is initiated. Assume for concreteness that a correctable burst of errors has occurred in positions corresponding to $(b_0x^j + b_1x^{j+1} + \ldots + b_{r-1}x^{j+r-1})$. (Note that this does not preclude low confidence indications

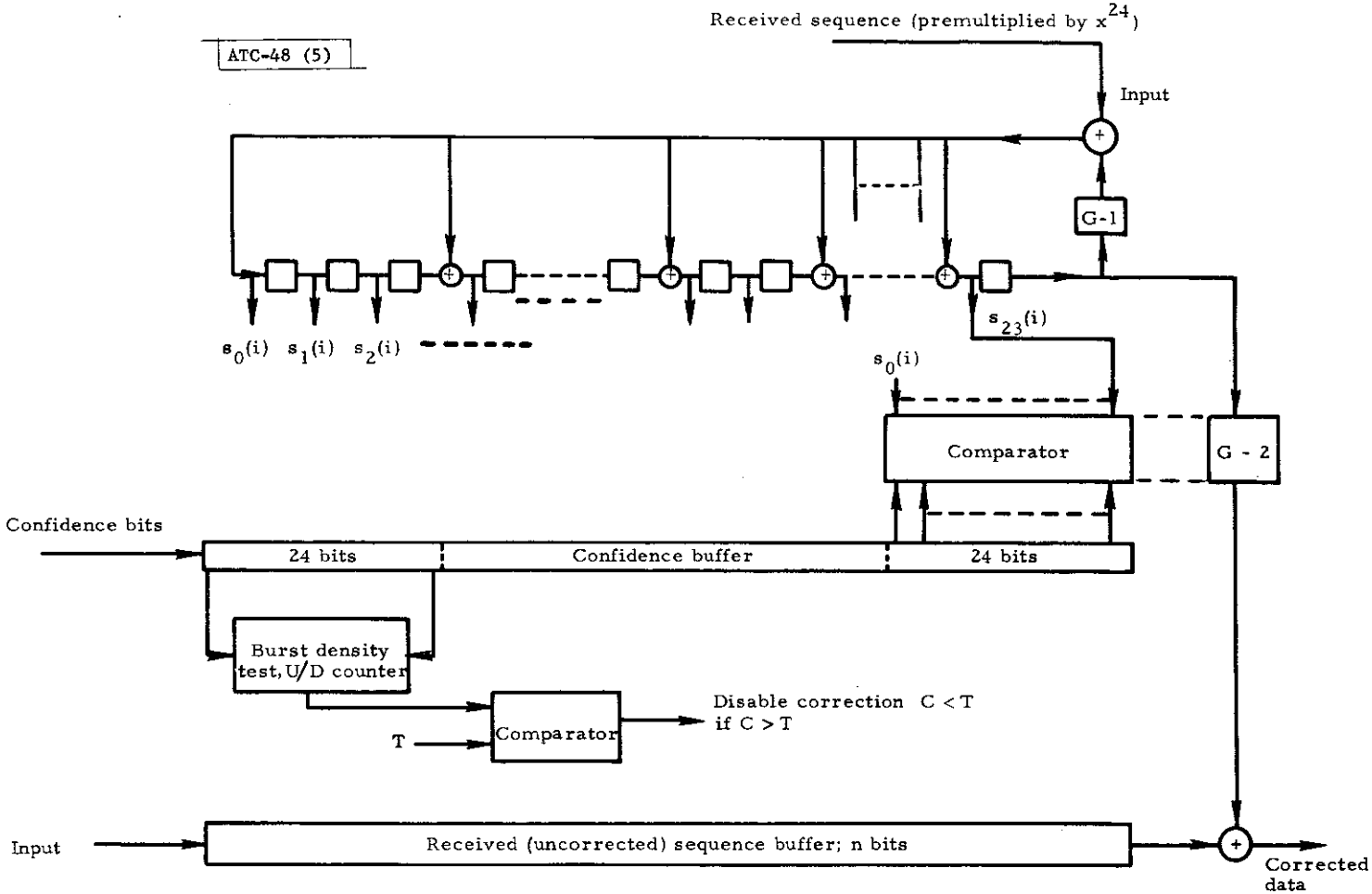


Fig. 5. Circuitry for burst erasure correction using full length code.

from appearing in other segments of the message.) The shift register is now clocked without input, and simultaneously both the message sequence and the confidence sequence are shifted out in synchronism. After each shift, the contents of the shift register are compared with the 24 confidence bits at the right end of the buffer. After the $(e - j)^{th}$ shift, the erasure burst is in the right end of the confidence buffer and the contents of the division register should be the exact reproduction of the errors within the burst. At this point a correction will take place if

(a) The 1(s) [errors] in the division register match one to one with low confidence indicators.

(b) No error is matched with a high confidence indicator.

(4) If the preceding two conditions are satisfied within the comparator circuitry, correction is effected by disabling gate G-1 (the division is terminated) and enabling gate G-2 so as to subtract, bit by bit, the contents of the shift register from the right-most 24 message bits as they emerge from the storage buffer.

(5) If conditions 3(a) and 3(b) are never satisfied after a total of e extra shifts, the decoder will flag the message to indicate a set of detectable but uncorrectable errors. The register and buffers are then zeroed out for the next incoming message.

III. 3. Implementation for a Shortened Cyclic Code

As described above, the burst trapping technique is tailor-made for a full length cyclic code. Recall, however, that the code chosen for the DABS downlink has been shortened from a natural length of $e \doteq 2.75 \times 10^6$ bits down

to n = 112 bits (or n = 56 bits). One way to directly utilize the correction methodology just described would be to append e - n zeros to the received message sequence before operating on it. However, this would introduce a substantial delay in decoding. In the following two subparagraphs, two implementations are given which circumvent this problem, both of which have their advantages and disadvantages.

III. 3. 1. Premultiplication by x^{e-n}

This technique is a direct extension of the argument just presented for premultiplication by x^r. Given a received codeword, $\hat{c}(x) = \hat{c}_0 + \hat{c}_1 x + \ldots \hat{c}_{n-1} x^{n-1}$, for $n < e$, then premultiplying by x^{e-n} before division in effect translates the sequence up to the high-order end of the full length codeword. Any further shifting with feedback (division), cycles the derived sequence back into the low-order end of the coefficient field. Furthermore, in order to perform the correction procedure as outlined for the full-length code above, an additional premultiplication by x^r is still required. Therefore, after all n bits have been clocked into the shift register (and the n stage storage buffer), the contents of the register will be the remainder after dividing $\hat{c}(x)\, x^{e-n+r}$ by g(x), i.e., division is performed on

$$\hat{c}(x)\, x^{e-n+r} = \hat{c}_0 x^{e-n+r} + \ldots + \hat{c}_{n-r} x^{e} + \ldots \hat{c}_{n-1} x^{e+r-1}$$

$$\equiv \hat{c}_{n-r} + \ldots \hat{c}_{n-1} x^{r-1} + 0 + \ldots + 0 + \hat{c}_0 x^{e-n+r} + \ldots$$

$$+ \hat{c}_{n-r-1} x^{e-1}$$

and after division, the r coefficients contained in the register are synchronized to the r high-order bit positions of the received sequence within the n-bit buffer.

The actual implementation of the premultiplication by x^{e-n-r} is facilitated by obtaining its remainder after dividing by g(x) and connecting the input leads appropriately. The technique is best explained by example, especially since there are two message lengths to consider (n = 112 and n = 56).

Case 1 $n = 112$ with $e = 21(2^{17} - 1)$ and $r = 24$.

This requires finding the r-bit remainder of x^{e-88} after division by $g(x) = 1 + x^3 + x^{10} + x^{12} + \ldots + x^{24}$ which is given by

$$x^{e-88} \equiv x^4 + x^5 + x^6 + x^7 + x^{10} + x^{13} + x^{14} + x^{15} + x^{16} + x^{17} + x^{18} + x^{19} + x^{21} \text{ modulo } g(x)$$

Case 2 $n = 56$ with $e = 21(2^{17} - 1)$ and $r = 24$.

Division of $x^{e-n+r} = x^{e-32}$ by g(x) yields

$$x^{e-32} \equiv 1 + x + x^5 + x^6 + x^7 + x^{13} + x^{15} + x^{17} + x^{18} + x^{22} + x^{23} \text{ modulo } g(x).$$

The complete implementation for each of these correction circuits is given in Figs. 6a and 6b, respectively. These two circuits can be collapsed

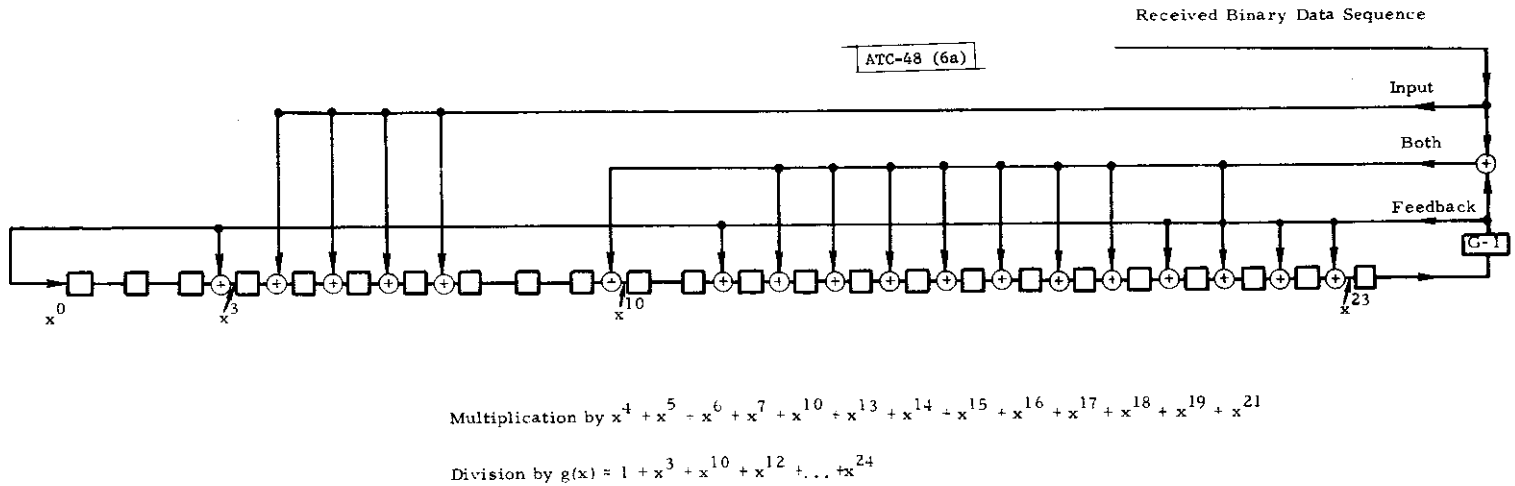
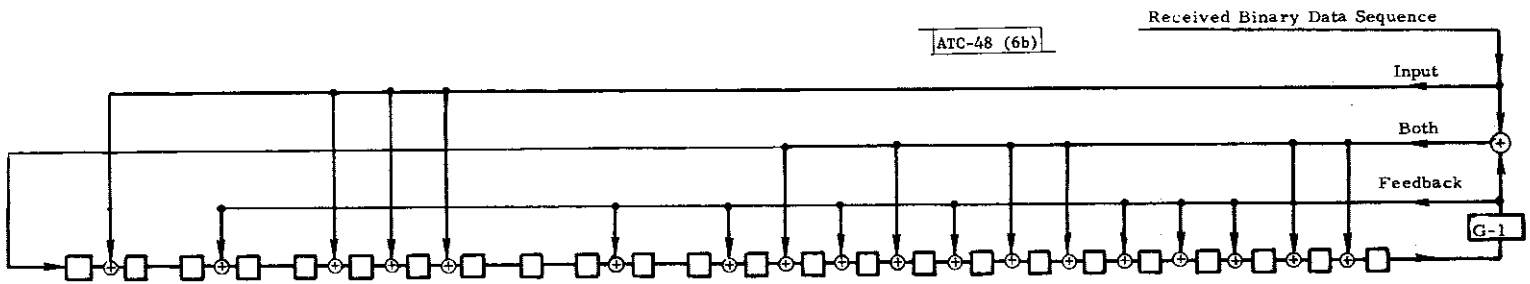


Fig. 6a. Division register with premultiplied input for n = 112 bits.



Multiplication by $1 + x + x^5 + x^6 + x^7 + x^{13} + x^{15} + x^{17} + x^{18} + x^{22} + x^{23}$

Division by $g(x) = 1 + x^3 + x^{10} + x^{12} + \ldots + x^{23} + x^{24}$

Fig. 6b. Division register with premultiplied input for n = 56 bits.

to one by using a few switches in the appropriate places. A more practical alternative allows the use of the single circuit for n = 112 bits if, whenever a 56-bit message is input, an extra 56 shifts are performed before the correction process is initiated. (This is due to the simple fact that $x^{e-32} = x^{56} \cdot x^{e-88}$.) In either case a storage buffer with a 112-bit capacity is needed. The correction process is the same as that given for the full-length code.

It is seen that this implementation requires a very modest increase in hardware complexity, no increase in processing complexity, and a maximum decoding delay of 112 extra clock units. Moreover, this configuration allows a completely serial operation and correspondingly simple logic.

III. 3. 2. Use of Reciprocal Code for Correction

From the previous descriptions it is seen that the correction process takes the calculated syndrome S_0, and shifts, in effect, an extra e - j times (for a correctable burst starting in the j^{th} bit). However, $x^{e-j} \equiv x^{-j}$ modulo g(x), which indicates that the burst would be trapped if somehow the circuitry can be made to shift (and divide) backward by j steps. The problem with this concept is that feedback shift registers are unidirectional devices. However, this can be implemented by using two division shift registers; a "forward acting" divide-by-g(x) to calculate the syndrome S_0 and check the address, and a "backward acting" divide-by-g(x) to perform the correction.

This backward acting device is actually implemented by using a division circuit corresponding to the reciprocal polynomial of g(x). This reciprocal polynomial is defined as $g^*(x) = x^r g(1/x)$ which has the same natural length and correction capability as g(x) and is seen to have the same coefficients as

g(x) written in reverse order. Thus if $S_0(x)$ is obtained from the first division circuit with the address stripped off and then dumped into the $g^*(x)$ division circuit with bits in the same order, it becomes $S_0^*(x)$, i.e.,

$$S_0^*(x) \equiv x^r(1/x)^j\,(B(1/x)) = x^{-j}\,B^*(x) \quad . \tag{4}$$

Now shifting this circuit j times with feedback (but no further input than the initialization of $S_0^*(x)$) yields

$$S_j^*(x) = x^j\,S_0^*(x) \equiv B^*(x) \ \text{ modulo } g^*(x) \tag{5}$$

which effectively yields $S_j^*(x) = B^*(x) = x^r B(1/x)$, the sought-for burst structure. All that remains to be done is to mechanize this and synchronize the correction to proper locations within the stored received sequence.

The implementation is shown in Figs. 7a and 7b with the process summarized below.

(1) The entire demodulated data sequence is input to the division register which divides by $g(x) = 1 + x^3 + x^{10} + x^{12} + x^{13} + \ldots + x^{23} + x^{24}$, and is simultaneously stored in the message buffer so that the first bit in resides in the right-most storage location. Similarly the confidence sequence from the receiver/processor is also stored for future use. It should be noted that there is no premultiplication by x^r before division. This means that after the last bit (n = 112 or n = 56) has been input, the content of the register is the address overlayed on the remainder (syndrome). If no errors have occurred, the address is immediately verified and the stored message is forwarded to the user.

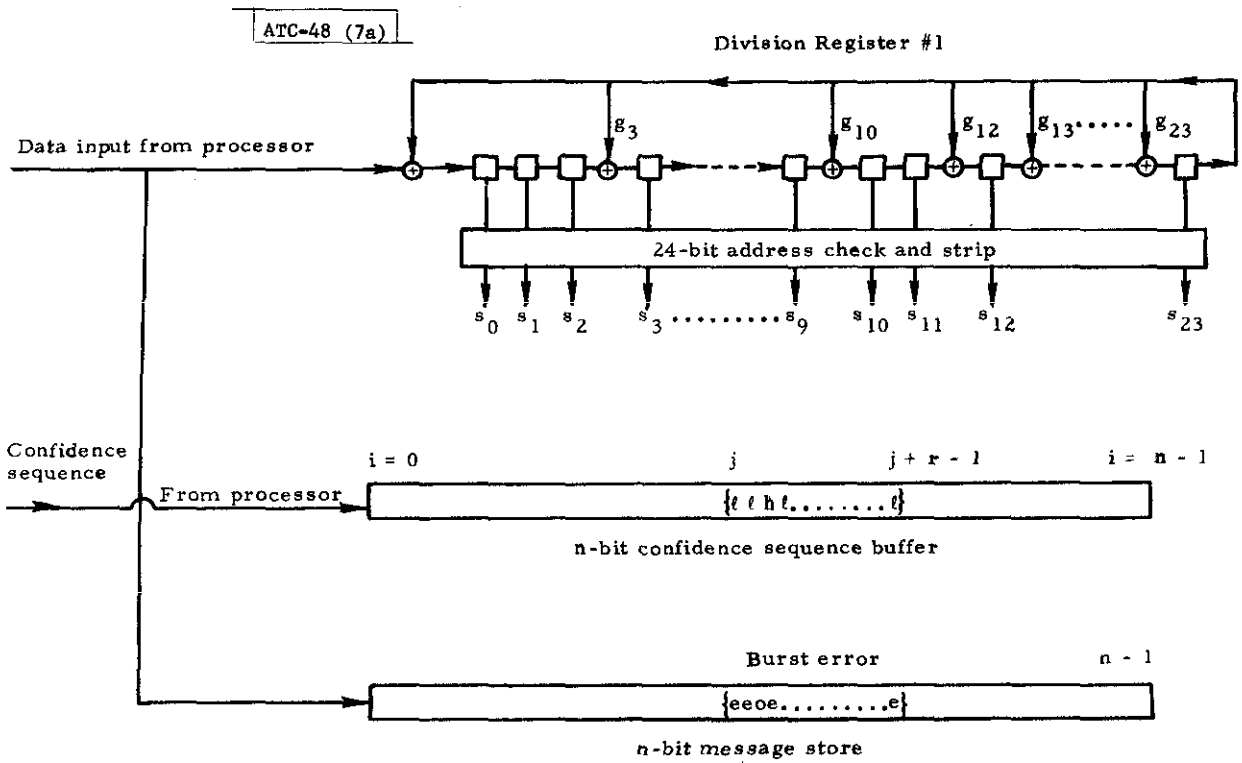


Fig. 7a. Decoder: syndrome calculation and address check.

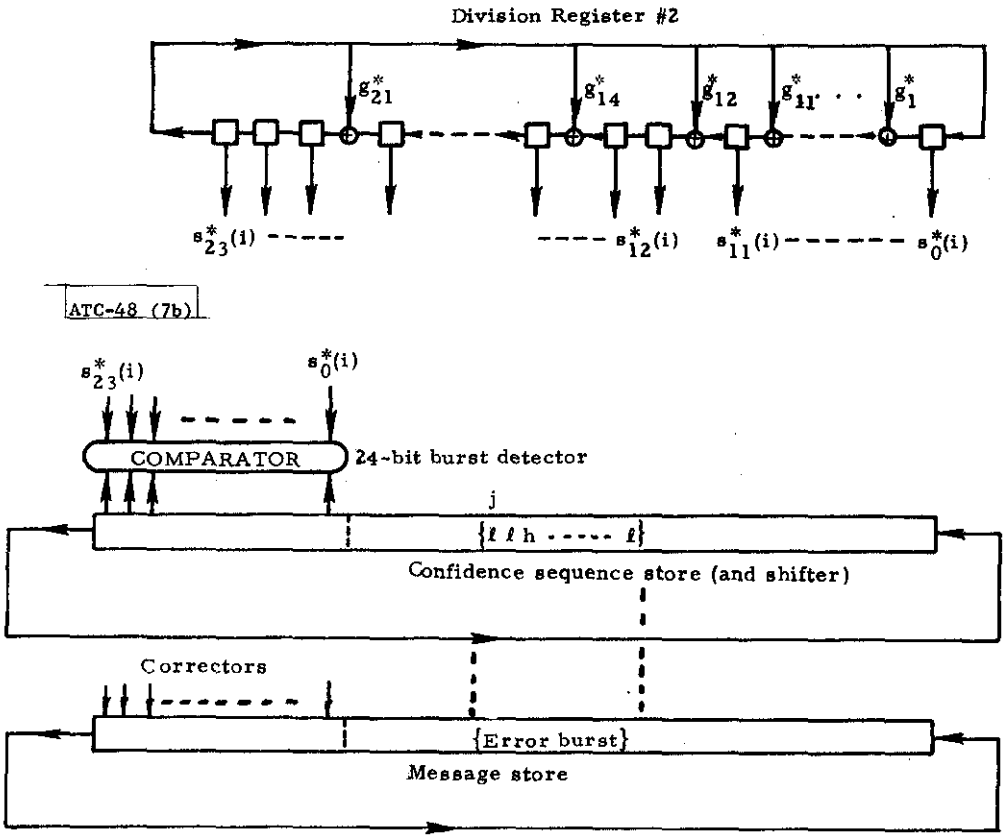


Fig. 7b. Correction of a 24-bit erasure burst.

(2) If the address is not recognized within the register locations, the (presumed) address is subtracted from the contents (added bit by bit, modulo-2) and the resulting syndrome, $S_0 = (s_0, s_1, \ldots, s_{23})$, read from left to right within the register, is dumped into division register no. 2 to initialize the correction procedure. Register no. 1 is then zeroed out for the next input.

(3) Division register no. 2, which divides by $g^*(x) = 1 + x + \ldots + x^{12} + x^{14} + x^{21} + x^{24}$, is initialized by

$$S_0^* = (s_0^*, s_1^*, \ldots, s_{23}^*)$$

read high index first from left to right in shift register no. 2. In other words, in comparing coefficients S_0 and S_0^*, $s_m^* = s_{23 - m}$, for m = 0, 1, 2, ..., 23, and the left-most flip-flop of register no. 2 is initialized by s_0 now denoted by $s_{23}^*(i = 0)$. This is done by parallel transfer.

(4) The division circuit is shifted a total of j times without further input. Simultaneously both the confidence sequence and the message sequence are cyclically shifted from right to left in sychronism with the division process. After each shift, the contents of the shift register are compared with the 24 confidence bits at the left end of the confidence register. After the jth shift, the erasure burst is in the left end of the confidence register, and the contents of the division register should be the exact reproduction of the errors within the burst. At this point, a correction takes place if the comparator recognizes that

(a) The 1's (errors) in division register no. 2 match one-to-one with low confidence indicators.

(b) No error is matched with a high confidence indicator.

(5) If the preceding two conditions are satisfied, the correction is effected by subtracting the 24-bit content of the shift register from the 24 message bits located in the left end of the message storage register. This can be done sequentially as in the previous implementations (with the division process disabled) or by parallel transfer. The corrected message is then reconstructed and forwarded to the user, and all registers are reset to zero. Because the received binary sequence has been cyclically shifted in the correction process, reconstruction of the transmitted message can be accomplished either by reversing the shifted sequence by j units before release from the right end of the register or by flagging the start-of-message bit and reading out from that point.

(6) If conditions 4a and 4b are never satisfied in n shifts, the decoder flags the sequence as containing a set of detectable but uncorrectable errors.

The main advantage of this mechanization is that correction takes place off line while the first division circuit can be processing the next incoming message for address check. Also, since there is no premultiplication by x^{24}, the address appears within the register after the last message bit has entered. The main disadvantage is the need for two hardware realizations of the division register and the need for parallel transfers. Also after correction, the message has to be "unshifted" so as to present the proper time sequence of message bits to the user.

III. 4. Encoding and Decoding of All-Call Messages

All-call reply messages are responses to the periodic transmission of the nondiscrete addressed all-call interrogation. As such, responses having

a particular aircraft address cannot be anticipated, which means the address has to be considered to be part of the information content of the message. Thus in the all-call reply mode, the transponder encoder does not overlay the parity on the address field but encodes it as part of the message. Conversely, the downlink decoder must recognize that an all-call reply is being processed, whereupon it disables the address check and looks for an all-zero remainder (syndrome). If the syndrome is nonzero, the correction process is initiated exactly as the previously described implementations.

IV

SUMMARY

The foregoing analysis has indicated the possible utilization of burst error correction techniques to improve the message throughput of the DABS downlink without sacrificing the error detection capability of the code. This possibility, and the fact that the same code is used for both uplink and downlink have an attractive and practical impact on the system design.

Performance of the proposed error correction technique is presently being evaluated and will be reported on in the near future. Preliminary results indicate a substantial improvement in message throughput without measurably degrading the very satisfactory error detection capability of the code.

REFERENCES

[1] J. T. Barrows, "DABS Uplink Coding," Project Report ATC-49, FAA-RD-75-62, Lincoln Laboratory, M.I.T. (25 July 1975).

[2] W. W. Peterson and C. J. Weldon, Jr., "Error Correcting Codes," Sec. Ed., M.I.T. Press, Cambridge, Massachusetts (1972).