DOT/FAA/PM-83/6

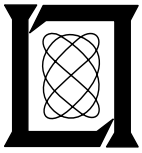
Project Report
ATC-117

Fundamentals of Mode S Parity Coding

J. L. Gertz

2 April 1984

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

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
DOT/FAA/PM-83/6		

4. Title and Subtitle	5. Report Date
Fundamentals of Mode S Parity Coding	2 April 1984
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Jeffrey L. Gertz	ATC-117

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
Lincoln Laboratory, M.I.T. P.O. Box 73 Lexington, MA 02173-0073	Proj. No. 052-281-04
	11. Contract or Grant No. DOT-FA72WAI-267
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address Department of Transportation Federal Aviation Administration Systems Research and Development Service Washington, DC 20591	Project Report
	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 F19628-80-C-0002.

16. Abstract

This report presents the details and basic theory of the coding scheme employed on Mode S uplink and downlink transmissions. Since ATCRBS interference is the main source of error for these signals, a cyclic burst detection code was chosen for Mode S. This code permits simple error detection at the transponder and more complex error correction at the sensor.

The theory behind cyclic encoding and decoding as used for Mode S is presented first. Then, since polynomial multiplication and division are required for these processes, circuits for these operations are described. Finally, the last chapter describes the actual implementations specified for encoding and decoding in both the transponder and sensor.

17. Key Words		18. Distribution Statement
Mode S Air Traffic Control Uplink Downlink	Coding Decoding Error correction	Document is available to the public through the National Technical Information Service, Springfield, Virginia 22161.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	42	

Form DOT F 1700.7 (8-69)

CONTENTS

ILLUSTRATIONS

FUNDAMENTALS OF MODE S PARITY CODING

1.0 BACKGROUND

Mode S, previously named the Discrete Address Beacon System (DABS), provides an evolutionary upgrade to the present third generation Air Traffic Control Radar Beacon System (ATCRBS). In particular, it provides an improved surveillance capability together with an integrated ground/air data link. Both features are required to support the planned automation of air traffic control.

Mode S includes a unique code as part of each interrogation. This 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 ATCRBS System. 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 that influenced the design of the Mode S signal formats were: (1) the need to attain service reliability 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 during the transition from ATCRBS to Mode S, and (3) minimization of transponder cost to speed the transition from ATCRBS to Mode S.

In view of these requirements, a study was undertaken to determine the benefits resulting from the incorporation of error control techniques into both the Mode S uplink and downlink messages. The results of this study were previously reported in [1] and [2]. These documents concluded:

(1) signal coding would significantly reduce the probability of receiving incorrect messages and the need for re-transmissions

(2) uplink error <u>detection</u> was feasible within the transponder cost constraints

(3) downlink error <u>correction</u> was possible at the sensor.

They also showed that the same code could be used on both links. The characteristics of the code, its error correction/detection properties, and the residual error probabilities were described in these documents.

This report updates these earlier documents by presenting the specific use and implementation of error detection and correction coding as it is now specified in the Mode S standards and specifications. This report also stresses understanding and basic theory over detailed theory to make it more usable by non theorists. Readers requiring more rigorous treatments are referred to [3].

The remainder of this chapter reviews the Mode S material that forms the background to signal coding. The following chapters then present and explain the various components of the overall Mode S coding design.

1.1 Coding Problem

One of the primary responsibilities of Mode S is the delivery and reception of various types of traffic control information to and from aircraft. It is necessary that such messages be validated before acceptance. Air Traffic Control voice radio commands are validated in the present system by repeating the command back to the ground. This same technique could be used with digital messages in Mode S as a low cost method of message validation. However, if a message could be validated in a single transaction, message delivery would require fewer transmissions and thus less channel capacity, and would also be less strongly affected by link reliability. Coding techniques offer just such a means of reliably validating a single transmission, and such coding techniques need not involve a great amount of circuit complexity. Thus, coding techniques were studied from the outset of the Mode S program as a promising means of providing a highly reliable and efficient message validation system with little cost impact on the transponder.

In order to eliminate the overhead associated with the redundant parity check bits in coding, a technique for combining parity and address bits was used as developed by the British in their early work on a discrete address system referred to as the ADSEL (Address Selection) beacon system. Instead of having the receiver check two separate message fields to determine if the received message should be accepted, a combined address/parity field allows the operation to be carried out by checking only one field. Whenever the parity check bits resulting from the received message are nonzero, the expected address/parity field is different from the actual received address/parity field, indicating the message should not be accepted. This scheme removes the overhead associated with the use of coding for message validation, which is an important step because of the constraints on message length.

Although the foregoing discussion makes coding appear attractive for message validation, the key problem is that of selecting a code that performs adequately in the channel environment. There are numerous error mechanisms affecting the Mode S uplink and downlink channels. A decision was made early in the coding investigation to concentrate on codes that can overcome errors caused by interference sources. Errors caused by noise only or caused solely by fading of the signal below threshold were not considered. A rationale for this posture is that errors due to noise alone can be dealt with by virtually any choice of code, while the fading mechanisms that arise from turning aircraft, over-the-horizon transmission, etc., have a duration that is longer than the Mode S message and therefore are beyond the control of any coding scheme.

Errors due to interference arise from ATCRBS interrogations, TACAN channels operating near or harmonically related to 1030 or 1090 MHz, continuous-wave (CW) interference, and multipath. Of these, ATCRBS interference is the dominant factor, and the code search was largely driven

by this fact. In an (uncontrolled) ATCRBS environment, an ATCRBS reply, consisting of several pulses, could overlap a Mode S message, resulting in a burst error channel. It is this channel that was given the major emphasis and that led to the eventual choice of a cyclic code, a class of codes specially suited to burst error detection.

Performance requirements, real-time operation, and low-cost considerations for the Mode S transponder were the major factors that resulted in the decision to have the error control technique provide an error detection-only capability on the uplink. In particular, the Mode S scheduling subsystem has sufficient capacity to accommodate an occasional retransmission because of a missed message. However, the emphasis is somewhat different for the downlink due to an inherently larger computational capability in the ground sensor decoder. 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, the data rate, the 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.

1.2 Downlink Channel and Receiver Characteristics

The center frequency of the Mode S reply transmission is 1090 MHz, which is the same as for the ATCRBS reply. The downlink signaling uses a pulse position modulation (PPM) format with a data rate of 1.0 megabit per second. Two immediate advantages of this PPM format are (1) a single ATCRBS reply pulse (0.45-μsec width) will not affect both 0.5 microsecond chip positions comprising a single data bit in the same way, thus increasing the detectability of interference, and (2) the complementing aspect of PPM ensures the presence of as many "ones" as there are data bits for use in a monopulse estimate.

The downlink message shown in Fig. 1-1 contains 56-bits for all-call or surveillance replies, and 112-bits for communications replies. The message (excluding preamble) is 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.

The major source of interference for the Mode S downlink is the ATCRBS reply. The reply, as shown in Fig. 1-2, is 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 and 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 position identification (SPI) that is transmitted only on initiation by the pilot. The important thing to note is that a single reply without an SPI pulse will adversely affect a span of less than 24 Mode S 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. By using a 24-bit cyclic parity check code, this single reply cannot create undetected errors in the Mode S message.

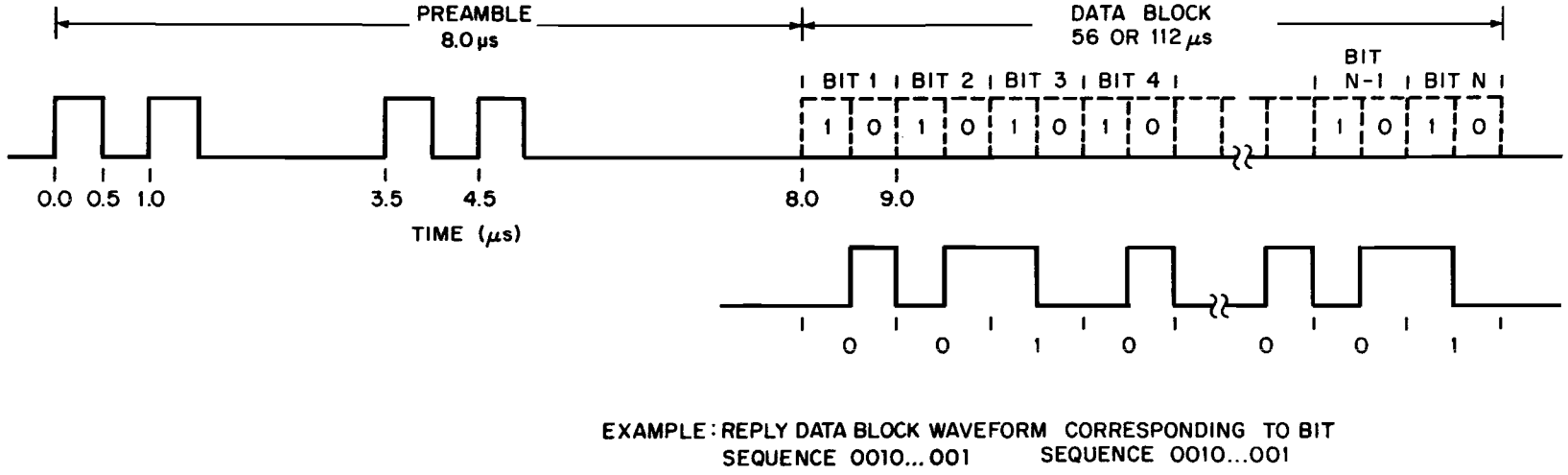


Fig. 1-1. Mode S repyl format.

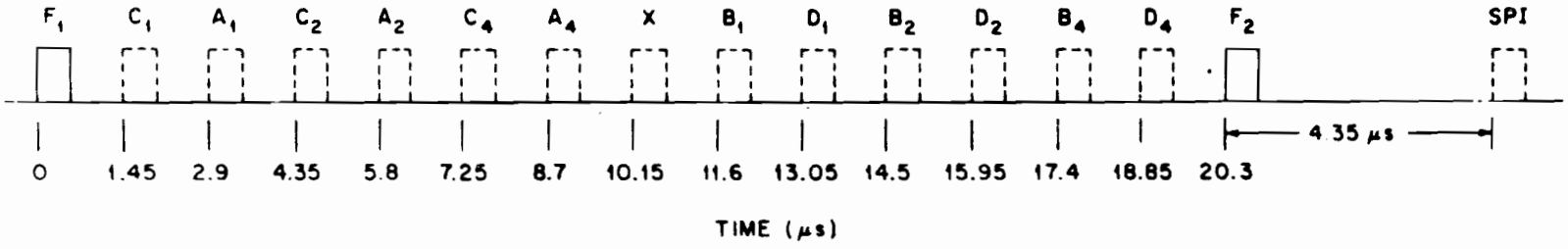


Fig. 1-2. ATCRBS reply format.

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 Mode S bits. Other interference sources such as multipath also lead to bursts of errors in the Mode S message.

The Mode S reply processor uses an amplitude comparator to determine the data from the PPM reply, assigning a '0' or '1' if the first or second chip respectively of the bit has greater amplitude. A separate detector determines whether interference is present for the bit. The data bit is flagged as high confidence if this detector finds (1) that no interference energy is present in the "other" chip, and (2) that the primary chip energy is in the mainbeam of the antenna rather than in a sidelobe. Other bit decisions are labelled as low confidence. The reply azimuth estimate is constructured by averaging the monopulse estimates of the first sixteen high confidence bits.

The confidence measures of the bit decision process are used in the error correction scheme. In particular, by assuming that bit errors can only occur in bits for which a low confidence estimate has been made, the input to the decoder can be characterized as an erasure channel. The correction ability for an erasure channel, in which the possible error locations are known, is the same as the detection ability on a normal channel, in which no such knowledge exists. Thus error correction becomes feasible on the downlink.

2.0 PARITY CODING

In general, every possible bit pattern in a message field can constitute a valid message. Thus if a transmitted message had one or more bits received in error, the message would be interpreted incorrectly as a different valid message with no errors. The method employed to permit error detection is to add parity bits to the message so that most channel errors will produce recognizably invalid messages. For example, the most common single parity bit is defined by:

$$p = \sum_{i=1}^{k} m_k \quad \text{where summation is performed using modulo 2 addition } (\oplus)$$

Then any valid message will have an even number of '1's; any single bit transmission error will produce an odd number of '1's. In general, if r parity bits are added to a message, only 2^{-r} of the possible received messages will be valid. The information rate in such a case decreases to k/(k+r), so channel capacity is reduced as protection is increased.

Each of the r parity bits is specified by a different function of the message to be encoded:

$$p_i = f_i\ (m_1, m_2, \ldots, m_k)$$

The functions are chosen to maximize the detection (or correction) properties of the coding scheme.

If random, uncorrelated errors are the main channel problem, the Hamming distance d of the code is the characteristic to be maximized. The Hamming distance between two valid code words is defined as the number of bits of the first that would have to be inverted to produce the second. d is the minimum such distance over all pairs of code words. The relationship between d and the properties of a code are given by:

d = t + 1	detect t errors
d = 2e + 1	correct e errors

The former rule states that errors can be detected as long as they cannot lead to another valid code word, while the latter rule states that errors can be corrected as long as they produce an error word nearer to the proper one than to any other valid word.

Another common channel problem is burst errors. On such a channel, errors tend to be bunched. A burst error of length b is defined as a sequence of b-bits containing errors in the first, last, and any set of interior bits. Fading and interference lead to burst errors. For Mode S, ATCRBS reply interference is the main source of errors. Thus the occurrence of burst errors is the dominant error mode.

Parity functions for burst detection codes tend to be quite different from those for random errors. No longer is distance important. Instead, the message bits that generate a specific parity bit must be widely spaced, or said another way, bits within a burst must independently determine different parity bits. A trivial code for detecting burst errors is as follows:

$$\begin{aligned} p_1 &= m_1 + m_{b+1} + m_{2b+1} + \cdots \\ p_2 &= m_2 + m_{b+2} + m_{2b+2} + \cdots \\ &\vdots \\ p_b &= m_b + m_{2b} + m_{3b} + \cdots \end{aligned} \qquad \text{(where + = sum modulo 2)}$$

This parity code detects any burst of up to b-bits, even though its Hamming distance is only 2, implying that it can only guarantee detection of a single random error.

An optimum burst code, such as that chosen for Mode S, will maximize the random error detection performance for a given burst length capability.

2.1 Cyclic Codes

A cyclic code is defined as one in which the set of valid code words is expressible as all multiples of a given generator polynominal G(x):

$$C(x) = H(x)\ G(x) \quad \text{for any } H(x) \tag{2-1}$$

where for burst detection applications G(x) will be of the same order as the burst length, namely b. The natural length n of a cyclic code, which is the number of bits each of its code words must contain to produce cyclic code properties, is the smallest integer for which:

$$\frac{x^n-1}{G(x)} \rightarrow \text{no remainder} \tag{2-2}$$

The code words produced by G(x) will have these n-bits broken up as:

$$n = k + b$$

where k is the number of information bits in the code word and b is the number of parity bits. Each potential code word, produced as in (2-1), is reduced to n-bits by being taken modulo x^n-1.

Cyclic codes are so-named because any code word, shifted cyclically, is still a code word. That is, if one code word is given by:

$$C_1(x) = H_1(x)\ G(x) = (a_{n-1}\ x^{n-1} + a_{n-2}\ x^{n-2} + \ldots + a_1\ x + a_0) \tag{2-3}$$

then a second one can be given by:

$$C_2(x) = xH_1(x)\ G(x) = H_2(x)\ G(x)$$

$$= a_{n-1}x^n + a_{n-2}x^{n-1} + \cdots + a_1x^2 + a_0x + (a_{n-1}-a_{n-1})$$

$$= (a_{n-2}x^{n-1} + a_{n-3}x^{n-2} + \ldots + a_0x + a_{n-1}) + a_{n-1}(x^n-1) \qquad (2\text{-}4)$$

But, since $a_{n-1}(x^n-1) = 0$ modulo x^n-1, the new code word is simply a cycled copy of the original one.

For a given message M(x), encoding consists of first shifting M(x) b-bits to the left to make room for the parity field, then dividing the resulting $x^bM(x)$ word by the generator G(x) to produce the remainder R(x), and finally adding this remainder to the shifted word to form the code word C(x).

To show that this process produces a cyclic code word, write M(x) in the following form:

$$x^b\ M(x) = m(x)\ G(x) + R(x) \qquad (2\text{-}5)$$

where the x^b multiplication shifts the message by b-bits, and the remainder R(x) is of order b-1. Adding R(x) in the parity field yields the code word:

$$C(x) = x^b\ M(x) + R(x) \qquad (2\text{-}6)$$

That C(x) is a code word follows from (2-5):

$$C(x) = x^b\ M(x) + R(x) = m(x)\ G(x) + R(x) + R(x) \qquad (2\text{-}7)$$
$$= m(x)\ G(x)$$

because by modulo 2 addition, a + a = 0.

Figure 2-1 provides an example of the encoding process. Note that, by convention, the first transmitted message bit is considered to be the highest order coefficient of the message polynomial. Also note, as explained above, the encoded transmission consists of the unmodified message followed by the remainder in the parity field.

To decode the received reply, the received word W(x) is processed to determine whether or not an error occurred (in either the message or parity fields, or both) during transmission. If not, the message is obtained directly from the high order bits of W(x).

Decoding with cyclic codes is merely the complementary process of encoding. That is, divide the received word W(x) by the generator to produce, in the general case:

$$\frac{W(x)}{G(x)} = w(x) + \frac{r(x)}{G(x)} \qquad (2\text{-}8)$$

If no error was encountered during transmission, r(x) = 0 by (2-7). Thus the presence of a non-zero remainder detects an error.

Message: | 1 | 0 | 1 | 1 | 0 |

1st bit ... 5th bit

$= x^4 + x^2 + x$

Generator: $x^2 + x$ → b = 2

Encode: find x^b M(x) / G(x):

$$
\begin{array}{r}
x^4 + x^3 + 0 + x + 1 \\
\hline
x^2 + x + 0 \;\big|\; x^6 + 0 + x^4 + x^3 + 0 + 0 + 0 \\
\underline{x^6 + x^5} \\
x^5 + x^4 + x^3 \\
\underline{x^5 + x^4} \\
x^3 \\
\underline{x^3 + x^2} \\
x^2 \\
\underline{x^2 + x} \\
x = R(x)
\end{array}
$$

Code Word: x^b M(x) + R(x) = $(x^6 + x^4 + x^3) + (x)$

$= x^6 + x^4 + x^3 + x$

= 1 0 1 1 0 | 1 0 |

message | parity

ATC-117

Fig. 2-1. Cyclic encoding example.

The proof that cyclic codes can detect burst errors is fairly straight-forward. First assume the burst error is in the low-order b-bits of the received word. That is:

$$W(x) = C(x) + E(x) \qquad E \text{ of order } b-1 \text{ (or less)} \tag{2-9}$$

Then, by (2-7):

$$\frac{W(x)}{G(x)} = \frac{m(x)\,G(x) + E(x)}{G(x)} = m(x) + \frac{E(x)}{G(x)} \tag{2-10}$$

and G(x) cannot divide E(x) as G(x) is of order b. Thus any low order burst error must produce a remainder. Now assume that the error occurred in a higher order b-bits. By the cyclic nature of code words, the received word can be cycled until the burst error is at the low-order end without affecting the code word properties. Then, since the division of the cycled word by G(x) yields a remainder, division of the original received word by G(x) must also yield a remainder (although not the same one). Otherwise, the received word would have been a code word, which contradicts the fact that the cycled one is not.

The code chosen for Mode S, for both the uplink and downlink, is a cyclic code based upon the generator polynomial:

$$G(x) = x^{24} + x^{23} + x^{22} + x^{21} + x^{20} + x^{19} + x^{18} + x^{17} + x^{16} + x^{15} + x^{14} + x^{13} + x^{12} + x^{10} + x^{3} + 1 \tag{2-11}$$

This code was first discovered by Kasami [4]. As its order is b=24, it can handle the expected 24 microsecond Mode S burst environment due to ATCRBS interference. In addition it has a Hamming distance of 6 to aid in the detection of random errors.

2.2 Shortened Cyclic Codes

The length n of cyclic codes and their code words is often quite long. For example, the code chosen for Mode S has a natural length, found as in (2-2), of $n = 21(2^{17}-1) \approx 2.75 \times 10^{6}$ bits. Clearly, the Mode S messages cannot be this long. Fortunately, cyclic codes can be shortened without loss of error detection properties.

An (n,k) cyclic code, meaning one with n-bit code words having k information bits (and thus n-k parity bits), can always be shortened to form an (n-i, k-i) code. This is done by only choosing from the total set of code words those whose first i information symbols are 0's, and deleting these symbols from the code words.

The resulting code, it should be noted, is not a cyclic code, in that not all cycled code words result in other code words. The code words are still generated by the same polynominal G(x), however, so all the encoding and decoding procedures still apply. Mathematically, whereas the natural cyclic code uses algebra based on modulo x^n-1, the shortened cyclic code uses algebra based on:

$$F(x) = x^{n-i}-R(x), \qquad R(x) = \text{remainder of } \frac{x^{n-i}}{G(x)} \tag{2-12}$$

Thus, a shortened cyclic code is also known as a pseudo-cyclic code.

For Mode S, the message codes employ 24 parity bits, and are shortened to either a length of 56 or 112-bits depending upon whether a 32-bit or 88-bit message is being transmitted.

3.0 MODE S CODING

This chapter shows how the general theory of cyclic code encoding and decoding presented in the previous chapter has been adapted to the Mode S uplink and downlink messages. The major considerations that led to the specific algorithms being employed are:

(1) transponder simplicity, and

(2) minimization of retransmissions

The first consideration took precedence over the second, so that only error detection is attempted on the uplink. On the downlink, however, error correction is implemented in accordance with the latter consideration.

The specific methods of encoding the parity fields on the two links differ, even though both include the Mode S address. These differences yield a transponder implementation, as presented in Chapter 5, that minimizes cost, and has the benefit that the same shift register circuit can be used for both decoding uplink messages and encoding downlink ones.

3.1 Uplink Coding Theory

If the only requirement on the uplink signal were error detection, the scheme presented in the previous chapter, namely appending the b-bit remainder to the shifted message, would be sufficient. However, a transponder must insure that it only processes messages intended for its aircraft. The unique aircraft address serves to identify the intended message destination. Thus an uplink signal must contain the data field, the address, and the parity field.

To minimize message length (and thus channel time), the address and parity fields have been merged into a single field. The transponder, upon decoding the message, will look for the pattern that would result if

(a) the address were its own, and
(b) no transmission errors have occurred

Failure to produce this pattern results in message rejection. Note that the transponder cannot determine whether (a) or (b) was violated. Differentiation between other aircraft messages and message errors was sacrificed by the address/parity merger.

The actual merged field is the sum of the parity bits and a function of the address. This function, rather than the address itself, was chosen to simplify the transponder decoding logic. The actual function is the high-order bits of the product of the address and the generator polynominal:

$$f = [A(x)\ G(x)]_{\text{high order b-bits}} \qquad (3\text{-}1)$$

For Mode S, b=24. Thus mathematically:

$$f = \frac{A(x)\ G(x)}{x^{24}} \text{ with remainder } \frac{r(x)}{x^{24}} \text{ lost} \qquad (3\text{-}2)$$

The actual uplink message can then be written as:

$$U = \underbrace{x^{24}\ M(x)}_{\text{message}} + \underbrace{R(x)}_{\text{parity}} + \underbrace{\frac{A(x)\ G(x)}{x^{24}}}_{\substack{\text{address}\\ \text{function}}} + \underbrace{\frac{r(x)}{x^{24}}}_{\substack{\text{lost}\\ \text{remainder}\\ (- = +)}} \qquad (3\text{-}3)$$

where, for review:

$$x^{24}\ M(x) = m(x)\ G(x) + R(x) \qquad (3\text{-}4)$$

The transponder decoding procedure (to decode the address) is to multiply the received message by x^{24} and then divide the result by G(x):

$$U' = \frac{x^{24}}{G(x)}\ U \qquad (3\text{-}5)$$

If no errors were made in transmission, the result would be by (3-3) and (3-4):

$$U' = \frac{x^{24}}{G(x)}\left[m(x)\ G(x) + R(x) + R(x) + \frac{A(x)\ G(x)}{x^{24}} + \frac{r(x)}{x^{24}}\right]$$

$$= \underbrace{x^{24}\ m(x)}_{\substack{\text{high}\\ \text{order}\\ \text{bits}}} + \underbrace{A(x)}_{\substack{\text{low}\\ \text{order}\\ \text{bits}}} + \underbrace{\frac{r(x)}{G(x)}}_{\text{remainder}} \qquad (3\text{-}6)$$

Thus the address will be directly readable as desired. If an error occurred:

$$U'' = \frac{x^{24}}{G(x)}\ [U + E(x)]$$

$$= U' + \frac{x^{24}\ E(x)}{G(x)} \qquad (3\text{-}7)$$

For a burst error of 24-bits or less, E(x) cannot be a multiple of G(x). Thus:

$$E(x) = e(x)\ G(x) + \varepsilon(x) \qquad \varepsilon \text{ of order } < 24 \tag{3-8}$$

producing by (3-7) and (3-8):

$$U'' = U' + \underbrace{x^{24}\ e(x)}_{\substack{\text{high}\\ \text{order}\\ \text{bits}}} + \underbrace{\frac{x^{24}\ \varepsilon(x)}{G(x)}}_{\substack{\text{low}\\ \text{order}\\ \text{bits}}} \tag{3-9}$$

The second term, being in the same bit field as the address from (3-6), produces a decoded address different from that being sought, and the message will be rejected.

3.2 Downlink Coding Theory

On the downlink, the desire is to minimize the transponder encoding complexity. Thus the combined parity/address field in this case is constructed as a simple addition:

$$AP = R(x) + A(x) \tag{3-10}$$

producing the downlink message:

$$D = \underbrace{x^{24}M(x)}_{\substack{\text{high}\\ \text{order}\\ \text{bits}}} + \underbrace{R(x) + A(x)}_{\substack{\text{low}\\ \text{order}\\ \text{bits}}} \tag{3-11}$$

The decoding logic, being performed in the sensor, is not as restricted in its complexity. In particular, error correction is employed in this case to minimize the need for retransmissions due to downlink errors. The basic steps of this process are:

1. compare the decoded remainder with the expected address to determine whether an error was made
2. locate the error using the receiver confidence bits
3. correct the error.

If step 1 indicates an error exists, the residual remainder after the address is removed is called the error syndrome. It is used to perform the step 2 and 3 operations as explained below.

Whenever an error is present, the syndrome will be non-zero. For a given 24-bit message segment, there are $2^{24}-1$ possible burst error patterns. Each of these patterns will produce a different one of the $2^{24}-1$ possible syndromes. Thus, there is a 1-to-1 correspondence between errors and syndromes. The syndrome pattern will differ from that of the error because of the division of the message by G(x) during the decoding process. The only time this pattern transformation does not occur is when the error burst is in the low-order 24-bits; in that case the syndrome and the error are identical.

Of course, the burst error can occur in <u>any</u> 24-bit segment. Since the errors in each such segment can generate every possible syndrome, the mapping from syndromes to errors is many-to-one. A given syndrome is produced by different error patterns in different segments, however, as the transformation is a function of error location. The syndrome thus contains no information concerning the location of the error; an independent source is needed for that function. Once the position of the error is located, though, the syndrome will specify the error pattern.

Each received bit is decoded in the Mode S sensor as a 0 or 1, with a separate confidence bit produced as described in Chapter 1 to indicate the receiver's certainty as to its decision. High confidence bits are assumed to be correct, and are not permitted to be changed. If the error syndrome indicates a burst error, and some 24-bit segments of the message has a pattern of low confidence bits that matches the 1's in the transformed syndrome that applies to that segment, the error will be assumed to be located, and these bits will be corrected. Errors due to multiple bursts, or errors in high confidence bits, can not be corrected. Of course, infrequently a wrong correction will be performed.

The first decoding step is division of the received message by the generator polynomial. If no error has occurred, and using (3-11) and (3-4):

$$\frac{D}{G(x)} = \frac{x^{24}M(x)}{G(x)} + \frac{R(x)}{G(x)} + \frac{A(x)}{G(x)}$$

$$= \frac{m(x)G(x)}{G(x)} + \frac{R(x)}{G(x)} + \frac{R(x)}{G(x)} + \frac{A(x)}{G(x)}$$

$$= m(x) + \frac{A(x)}{G(x)} \qquad (3\text{-}12)$$

Thus, if the remainder matches the address, no correction is needed. If, however, a burst error has occurred, then by (3-8):

$$\frac{D}{G(x)} = m(x) + \frac{A(x)}{G(x)} + \frac{E(x)}{G(x)}$$

$$= m(x) + e(x) + \frac{A(x)}{G(x)} + \frac{\varepsilon(x)}{G(x)} \qquad (3\text{-}13)$$

and the remainder differs from the address.

The maximum length 24-bit burst error polynomial, assumed to be located j bits from the end of the message, can be written as

$$E(x) = x^j F(x) \qquad F \text{ of order 23 (or less)} \qquad (3\text{-}14)$$

where E(x) is of order 23 + j. After the decoding division by G(x) (which is of order 24):

$$\frac{E(x)}{G(x)} = e(x) + \frac{\varepsilon(x)}{G(x)} \qquad e \text{ of order } j\text{-}1 \qquad (3\text{-}15)$$

and only the $\varepsilon(x)$ part is visible in the syndrome. If j=0, that is if the error were in the low order 24-bits, then e(x)=0, $\varepsilon(x)$ = E(x), and the known part would in fact be the error itself. For any other case, correction is not possible at this point in the process.

Since a cyclic code is being employed, the whole message could be cycled right j times without changing the decoding properties. At that time, the burst error would be in the low order bits, and the remainder syndrome would be the error pattern. Of course, j is unknown to the receiver. Thus the process would have to be:

1. subtract (add) the address from the received message to get N(x) = D(x) - A(x), which by (3-12) is a code word if no errors have been made

2. divide the message N(x) by G(x) to determine if a remainder exists, and thus an error has been made

3. compare the remainder (syndrome) pattern to the low order 24-bit confidence pattern

4. if all 1's in the syndrome are paired with a low confidence bit, the error has been located

5. else cyclically shift right the message and the confidence word

6. return to 2, using the new shifted message as N(x).

Fortunately, the need to perform the whole division process each time can be eliminated. This is because the syndrome S_1 of the left cyclically shifted message can be produced directly from the original syndrome S_0. By definition of the syndrome, it is the remainder when dividing N(x) by the generator G(x):

$$S_0 = N(x) - G(x)\ Q_0(x) \tag{3-16}$$

where $Q_0(x)$ is some quotient, and S_0 is of degree 23 or less. The left shifted message's syndrome S_1 is given by:

$$S_1 = xN(x) - G(x)\ Q_1(x) \tag{3-17}$$

where xN(x) is the left shifted message, $Q_1(x)$ is a different quotient, and S_1 is again of degree 23 or less. Then:

$$xS_0 - S_1 = -G(x)[xQ_0(x) - Q_1(x)] \tag{3-18}$$

But the degree of the left side is at most 24, while that of G(x) is exactly 24. Therefore the quantity in brackets must be a constant c (0 or 1) to keep the right side degree no greater than 24. Furthermore, if the degree of S_0 is less than 23 (that is, if its leading coefficient $s_{23}=0$), then the left side is of degree 23 or less, requiring c to be 0. Otherwise, if $s_{23}=1$, xS_0 is of degree 24 while S_1 is of degree 23 or less, so the left side cannot be zero, requiring c to be 1. Summarizing these results:

$$S_1 = xS_0 \qquad \text{if } s_{23} = 0$$

$$S_1 = xS_0 + G(x) \quad \text{if } s_{23} = 1 \tag{3-19}$$

This equation states that the syndrome of the left shifted message can be obtained from the original syndrome by a shift and add the divisor operation. The process can be implemented by entering the original syndrome into a division circuit set to divide by G(x), of the type shown in Fig. 4.4, and operating the circuit with no input. Each cycle then shifts the syndrome, and adds G(x) or 0 according to whether the rightmost stage (s_{23}) is 1 or 0 respectively, as required by (3-19).

This procedure would be directly applicable only to the full length message, that is, with $n = 2.75 \times 10^6$ bits of zeroes appended to its beginning. Thus the total number of left shifts required to bring the error burst to the right end of the message, namely n-j, would be enormous. Fortunately, a modification of this procedure, using the polynomial F(x) defined earlier in (2-12), is available for shortened codes.

Instead of employing the usual shortened code modification, though, the Mode S implementation has chosen to use a revised form of the above basic procedure. This alternate version in effect uses a backward (right shift) process. That is, in place of multiplying the message by the x^{n-j} needed for left cycling, it multiplies by the x^{-j} that produces right cycling. That $x^{n-j} = x^{-j}$ in x^{n-1} modulo arithmetic is shown as follows:

$$x^{n-j}-x^{-j} = x^n x^{-j}-x^{-j}$$

$$= x^{-j}(x^n-1) = 0 \bmod(x^n-1) \qquad (3\text{-}20)$$

Thus, a j cycle reverse shift and division process would be equivalent to the much longer n-j cycle normal left shift process. Since shift registers don't shift in reverse, the implementation of this process requires inserting the original syndrome S_0 in reverse bit order into another shift register, and setting its taps to divide by the reciprocal polynomial G'(x), where the coefficient g'_i is defined as the coefficient g_{24-i} of the original G(x). The implementation chapter presented later clarifies this procedure. The additional shift register required for this procedure is not a drawback, but rather a plus, as it frees the original register to process the next downlink message.

4.0 POLYNOMIAL ARITHMETIC

As seen in the previous chapter, encoding and decoding procedures require numerous polynominal arithmetic operations: addition, multiplication, and division (subtraction is the same as addition, as +1 = -1 in modulo-2 arithmetic). Addition is implemented simply by a modulo-2 adder, also known as an exclusive-or circuit. Multiplication and division, however, require fairly complex shift register implementations. This chapter presents the details of these circuits.

4.1 Polynomial Multiplication

The product of two polynomials, $P(x) = H(x)\,G(x)$, is formed by grouping together and summing modulo-2 all cross-coefficient terms having the same exponent sum. In particular, the product coefficient p_j is given by:

$$p_j = \sum_{i=0}^{r} g_i h_{j-i} \qquad \text{(r the degree of G)} \qquad (4\text{-}1)$$

where

$h_{j-i} = 0$ for $j-i < 0$

$h_{j-i} = 0$ for $j-i > k$ (k the degree of H)

Thus, the product coefficient can be computed if all the g coefficients are available along with the r+1 h coefficients from h_j down through h_{j-r}.

Figure 4-1 presents a tapped r-stage shift register circuit that implements this operation. The h coefficients are entered one by one, highest one first, into the register (initialized to all zeroes). After the last one (h_0) is entered, zeroes are fed in until the multiplication is completed. Thus the input plus the register always contains the sequence h_j, h_{j-1},..h_{j-r} as required, for all j from n=r+k to 0. The g coefficients are represented by the presence (g_i=1) or absence (g_i=0) of the inter-stage taps. Thus, each clock cycle, the summation box produces the product coefficient according to the above summation formula.

An alternate multiplication circuit is also commonly employed. This circuit, shown in Fig. 4-2, generates each product coefficient piece by piece as the input h coefficients are encountered. The above summation formula indicates that the coefficient h_i contributes to r+1 different product coefficients as follows:

$h_i g_0 \rightarrow p_i$

$h_i g_1 \rightarrow p_{i+1}$

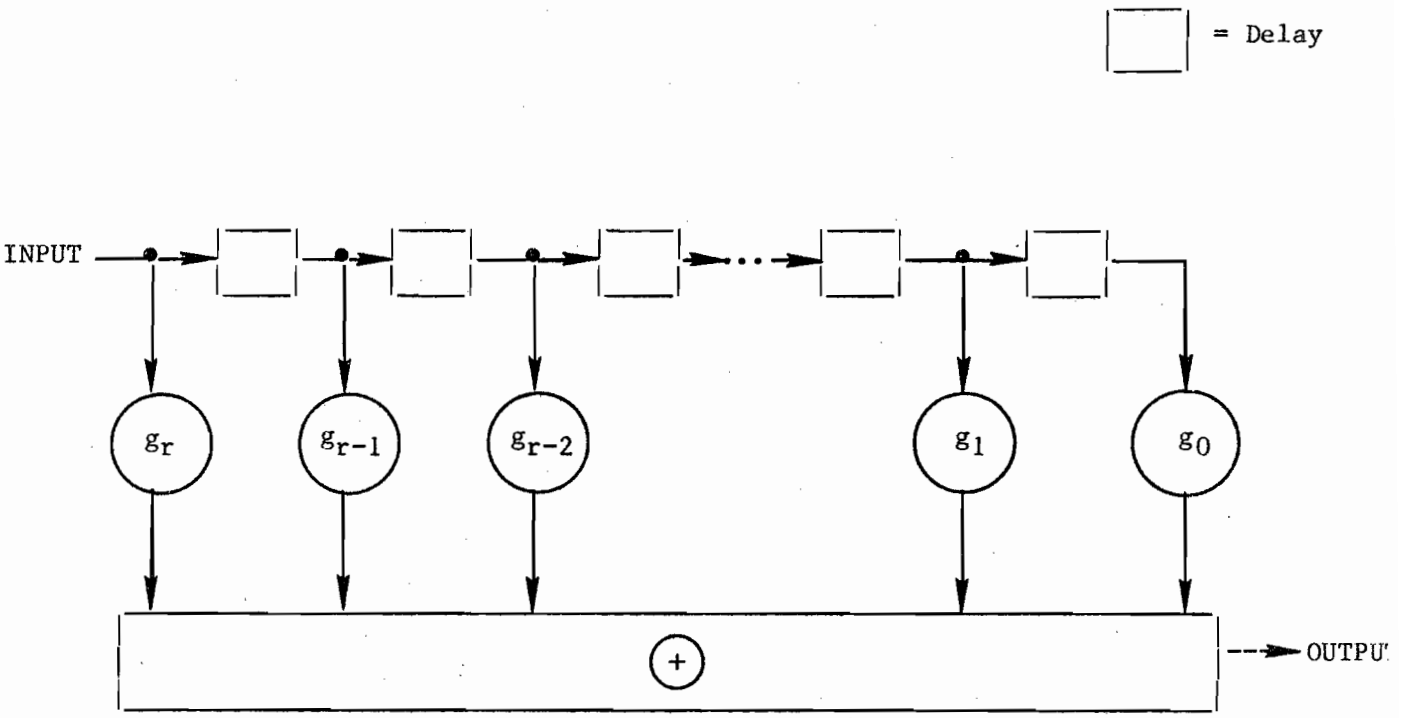
.

.

.

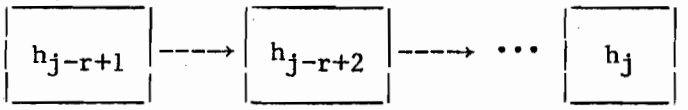
$h_i g_r \rightarrow p_{i+r}$

Multiply $P(x) = H(x)\ G(x)$



Input: $h_k, h_{k-1}, \cdots, h_1, h_0$

Register: for j^{th} product coefficient (p_j)

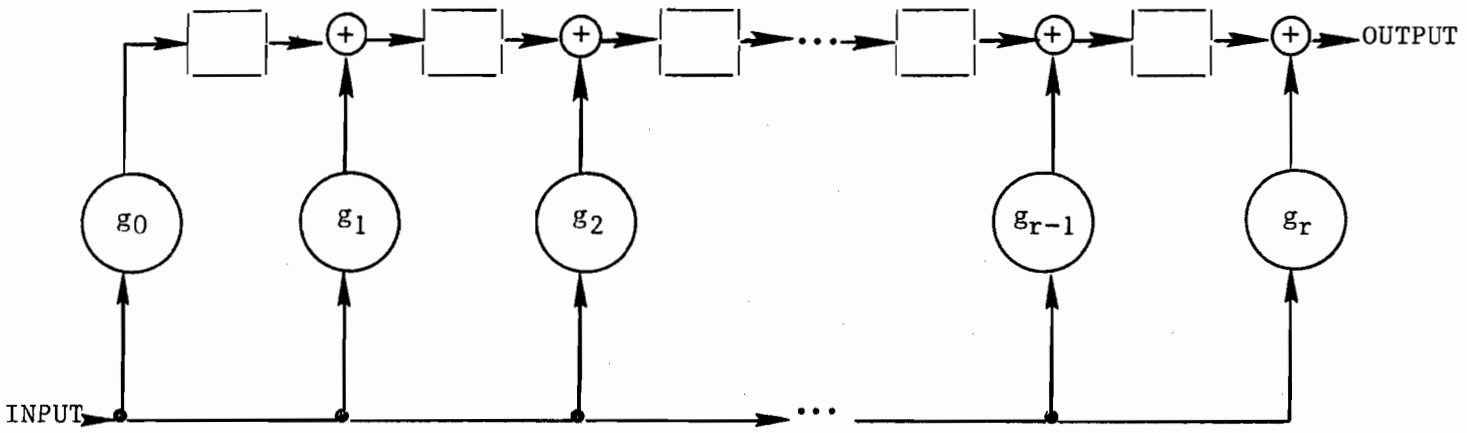


with h_{j-r} at the input.

ATC-117

Fig. 4-1. First multiplication circuit.

Multiply P(x) = H(x) G(x)



Input: $h_k, h_{k-1}, \cdots, h_1, h_0$

Register:

contains r partial product coefficients, each of which is generated one term at a time as they pass through the register stages.

p_j is output when h_{j-r} is input

ATC-117

Fig. 4-2. Second multiplication circuit.

Thus, in this second implementation, the shift register stages store the r product coefficients that are being generated. When the new input coefficient is input, its contribution completes one product coefficient which is then output, adds to r-1 already started coefficients, and initiates one new coefficient. The multiplication is completed r cycles after h_0 is input (zeroes being input during these cycles).

4.2 Polynomial Division

The process of polynomial division, $Q(x) = H(x)/G(x)$, as illustrated by the example in Fig. 4-3, is similar to long division. At each step, the high order coefficient g_r of the divisor is divided into the highest order coefficient of the current remainder, with the result being the new quotient coefficient. This quotient coefficient is then multiplied by the divisor, and the result subtracted from the old remainder to form the new remainder. Since modulo-2 arithmetic is being employed, two simplifications result. First, the quotient is 0 or 1 according to whether the highest order remainder coefficient is 0 or 1, and second, subtraction is equivalent to addition (-1 = +1).

A circuit to implement this process is shown in Fig. 4-4. The shift register stages store the highest order r coefficients of the remainder, which are the only ones affected in the next step. The new quotient coefficient, which is the output, is multiplied by the divisor via the shift register tap circuitry (compare with Fig. 4-2). This product is then added (i.e: subtracted) to the shift register stages to form the new remainder.

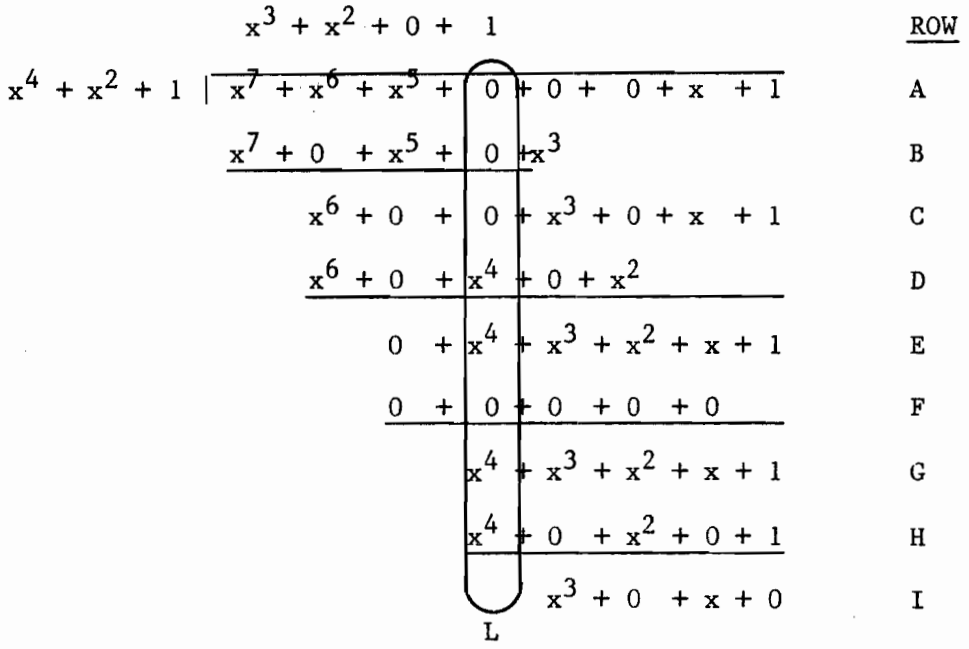
The shift register is initialized to zero. The first r shifts fill the registers with the first r coefficients of H(x), which is the first remainder. The remaining k-r+1 shifts then produce the quotient coefficients starting with q_{k-r}, the highest of the quotient. The division remainder then resides in the shift register, where it can be read in parallel or shifted out with the feedback disabled.

An alternate circuit for division is presented in Fig. 4-5. This implementation considers division to be a vertical column, rather than a horizontal row, operation. As stated earlier, only the high order remainder coefficient determines the quotient coefficient. This high order coefficient is generated by the actions in the column above it, as indicated in the Fig. 4-3 example. The coefficient value is initialized by an h coefficient. Then it is inverted at each division step for which the quotient is a 1 and the lining up g coefficient is also a 1. That is, the leading coefficient for the j^{th} division step is:

$$c_j = h_{k-j} + q_{k-j}\, g_0 + q_{k-j-1}\, g_1 + \ldots + q_{k-j-r+1}\, g_{r-1} \qquad (4\text{-}2)$$

This coefficient, which determines the quotient coefficient q_{k-j-r}, is thus affected by the previous r quotient coefficients. Conversely, each quotient coefficient affects the next r quotient coefficients to be produced.

Divide $Q(x) = H(x) / G(x)$



Row A: Input Stream
B, D, F, H: feedback for Fig. 4-4 type divider
C, E, G: intermediate remainders
I: final remainder

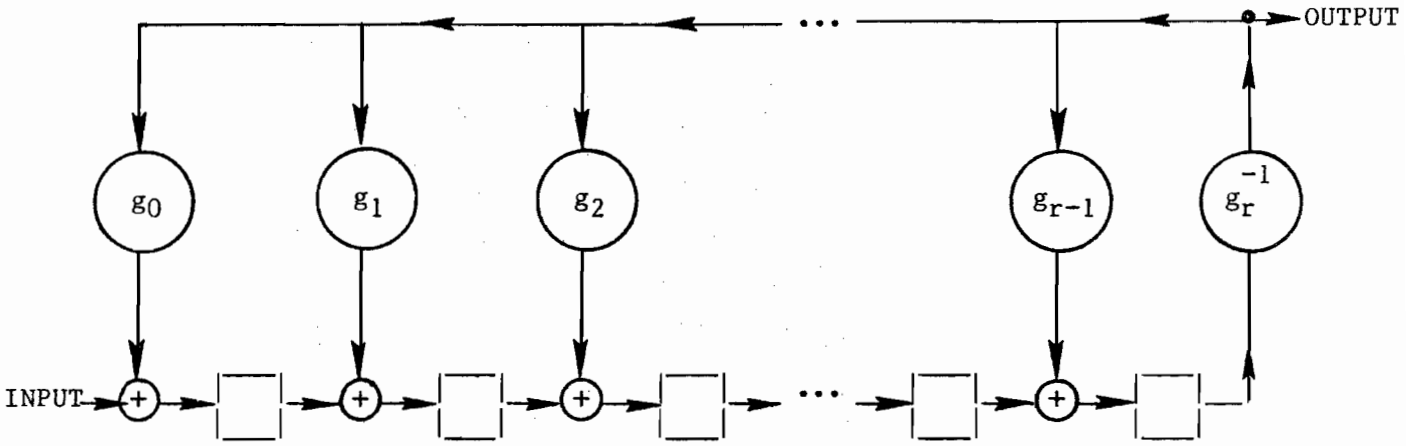
Column L: shows formation of leading coefficient of remainder in Fig. 4-5 type divider

initial value (Row A) - given by input
final value (Row G) - modified by prior quotient terms in rows B, D, F

ATC-117

Fig. 4-3. Sample division process.

Divide Q(x) = H(x) / G(x)



Input: h_k, h_{k-1}, ••• , h_1, h_0

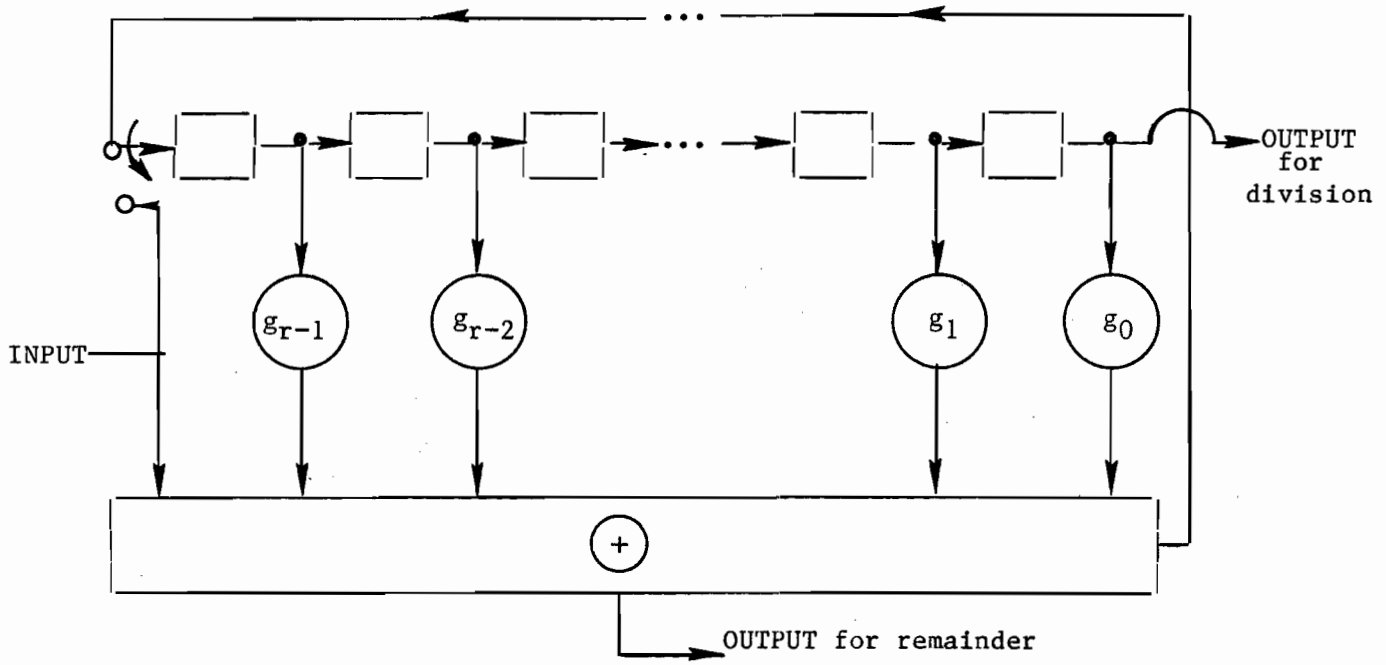
Register: stores intermediate remainder (see Fig. 4-3)

Feedback: multiplication of new quotient term by G(x) (see Fig. 4-3)
= high order remainder coefficient /g_r (g_r = 1 always)

ATC-117

Fig. 4-4. First division circuit.

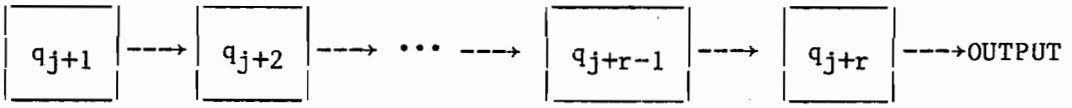
Divide Q(x) = H(x) / G(x)



Input: h_k, h_{k-1}, ••• , h_1, h_0 during division phase

q_{-1}, q_{-2}, ••• , q_{-r} during remainder phase

Register: as q_j is being generated (and q_{j+r} is being output)



final value: remainder quotients $q_{-r} \longrightarrow q_{-1}$
must be multiplied by G(x) to get remainder

Switch: thrown to disconnect feedback and connect input to produce the multiplier needed to generate remainder

Feedback: new quotient coefficient

ATC-117

Fig. 4-5. Second division circuit.

The shift register in this figure stores the last r quotient coefficients generated. As each input is entered, it is summed with the contributions of these quotient coefficients multiplied by the proper g coefficients, thereby producing the proper leading coefficient of the remainder (which as stated earlier is the new quotient coefficient) according to the formula (4-2).

Outputs from the register first appear when the $r+1^{st}$ input coefficient is encountered. This "delay" serves to match the fact that the highest quotient coefficient is q_{k-r}. When the last input coefficient arrives, the last quotient coefficient, q_0, is output. The register then contains quotient coefficients $q_{-1}, q_{-2}, \ldots, q_{-r}$, not the remainder. The remainder can only be produced by a multiplication of the register contents and G(x):

$$R(x) = q_{-1}g_r x^{r-1} + [q_{-1}g_{r-1} + q_{-2}g_r]x^{r-2} + \ldots + [q_{-1}g_{r-j+1} + q_{-2}g_{r-j+2} + \ldots]x^{r-j} + \ldots + q_{-r}g_r x^0 \qquad (4\text{-}3)$$

Comparing this division circuit of Fig. 4-5 with the multiplication circuit of Fig. 4-1, it is clear that the divider can be converted to a multiplier by switching out the feedback loop and switching in the input line as shown in the figure. Then the remainder coefficients must be read out of the register and re-introduced to the circuit as inputs, q_{-1} first. Finally, the remainder polynomial is produced at the output of the summing circuit, which as shown in Fig. 4-1 is the normal multiplier output port, after r more cycles of operation.

The result is that this division circuit requires at least r cycles more time to compute the remainder than did the previous one. This additional delay can be eliminated, though, in the special case in which the polynomial to be divided has its last r coefficients all zeros. This condition fortunately applies in the Mode S situation, as the message to be encoded is shifted out of the parity field, leaving that field all zero.

By referring to Fig. 4-5, it is clear that the quotient coefficients are actually generated r cycles before they are output, as they are the values placed in the feedback loop. Thus, if the output is taken at the feedback point, all coefficients q_{k-r} through q_0 will be obtained before the last r inputs have been introduced. These inputs, in general, are needed to determine the remainder. However, since they are all zero, they make no contribution in this special case.

By definition, the remainder is the difference between the input polynomial and the product of the quotient and divisor polynomials:

$$R(x) = H(x) - Q(x)\,G(x) \qquad (4\text{-}4)$$

Since R(x) is of order r-1, though, this can be written as:

$$R(x) = H(x)_{\text{low order}} - Q(x)\,G(x)_{\text{low order}} \qquad (4\text{-}5)$$

Using the facts that the low order H(x) terms are all zero, and that - = +, the final result is:

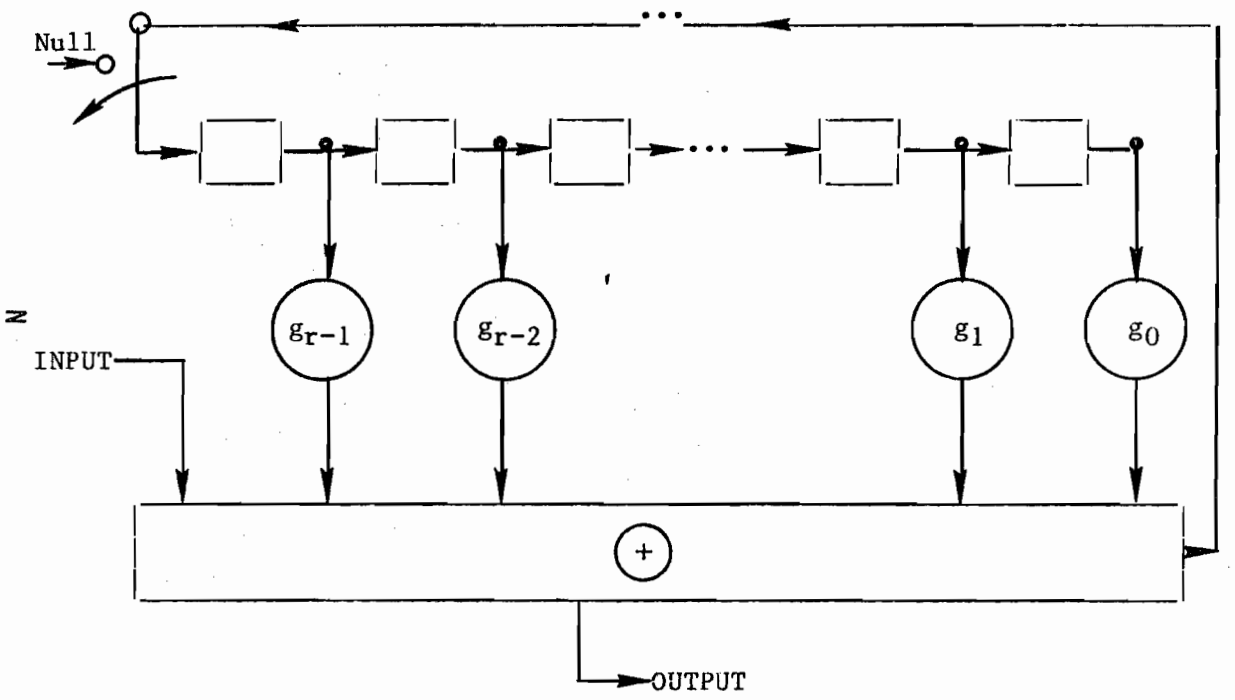
$$R(x) = Q(x)\ G(x)_{\text{low order}} \tag{4-6}$$

Expanding this result:

$$\begin{aligned} R(x) = {} & [q_0\ g_{r-1} + q_1\ g_{r-2} + \cdots + q_{r-1}\ g_0]x^{r-1} \\ & + [q_0\ g_{r-2} + q_1\ g_{r-3} + \cdots + q_{r-2} g_0]x^{r-2} \\ & + \cdots + [q_0\ g_1 + q_1\ g_0]x + q_0\ g_0 \end{aligned} \tag{4-7}$$

At the time the last quotient q_0 has been generated by the division circuit feedback, the register stages contain q_0 through q_{r-1}. Furthermore, q_0 is aligned with g_{r-1}, q_1 with g_{r-2}, etc. Thus the circuit, when placed in a multiplier configuration, will generate, sequentially, the remainder terms. The circuit in Fig. 4-6, therefore, implements both the required division and remainder generation operations. This second type of division circuit now requires no more cycles than the first to perform these operations.

Divide $Q(x) = H(x) / G(x)$



Input: $h_k, h_{k-1}, \cdots, h_{r+1}, h_r$

Register: same as Fig. 4-5
final value: quotients $q_0 \rightarrow q_{r-1}$

Switch: thrown to disconnect feedback after input h_r

Output: quotient coefficients $q_{k-r} \rightarrow q_0$ while input exists
remainder coefficients $R_{r-1} \rightarrow R_0$ in next r cycles

ATC-117

Fig. 4-6. Revised circuit for inputs with trailing zeroes.

5.0 MODE S IMPLEMENTATION

Now that the mathematical description of the Mode S uplink and downlink coding processes has been developed (in Chapter 3), and the polynomial arithmetic circuits have been described (in Chapter 4), the actual Mode S sensor and transponder coding implementations can be provided. This chapter presents the actual figures contained in the Mode S National Standard and Specification (FAA-E-2716). Each figure is functionally explained by putting together the knowledge provided by the previous two chapters.

5.1 Uplink Implementation

The sensor uplink encoder is shown in Fig. 5-1. Basically, it is the revised division circuit presented earlier in Fig. 4-6. Note that the output is taken at the feedback loop, rather than 24 cycles later when this value exits the shift register as in the true division circuit of Fig. 4-5. This change is thus equivalent to multiplying the input by x^{24}, yielding the division $x^{24}M(x)/G(x)$ as desired.

With the switch up, the division is performed, and the remainder quotient coefficients placed into the shift register as explained in the previous chapter. Then, when the switch is lowered to remove the feedback, two simultaneous operations occur in the now multiplier circuit (refer to Fig. 4-6). First, the remainder R(x) is generated in the manner explained in the last chapter when the input of the address is ignored. Second, the presence of this input through the switch causes it to be multiplied by G(x). This latter multiplication is not completed, however, as the 24 trailing zeroes needed to complete the formation of the product (see 4.1) are not input. Thus only the high-order bits are produced. The result, by superposition, is that the AP field output is:

$$AP = R(x) + A(x)\ G(x)_{\text{high order}} \qquad (5\text{-}1)$$

as desired.

The transponder decoder circuit, also shown in Fig. 5-1, is again of the type of Fig. 4-6. Only this time, it is always left in the division mode. Once again, the input is multiplied by x^{24} due to the position of the output. Thus the result is:

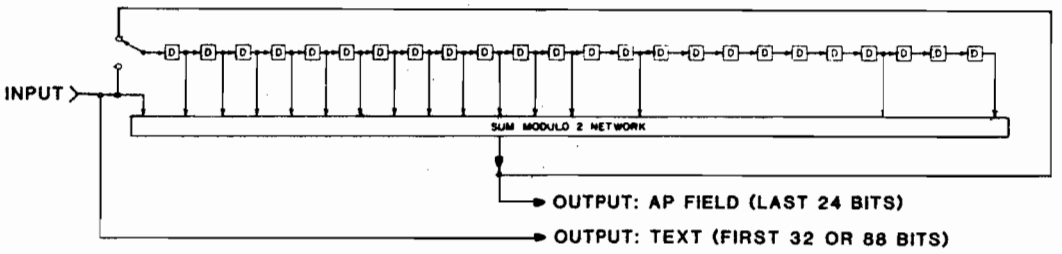
$$U' = \frac{x^{24}}{G(x)}\ U \qquad (5\text{-}2)$$

as desired.

5.2 Downlink Implementation

The transponder encoder circuit, also shown in Fig. 5-1, is virtually identical to the sensor uplink encoder. The difference is that the address is not input to the multiplier circuit through the switch for the second part of

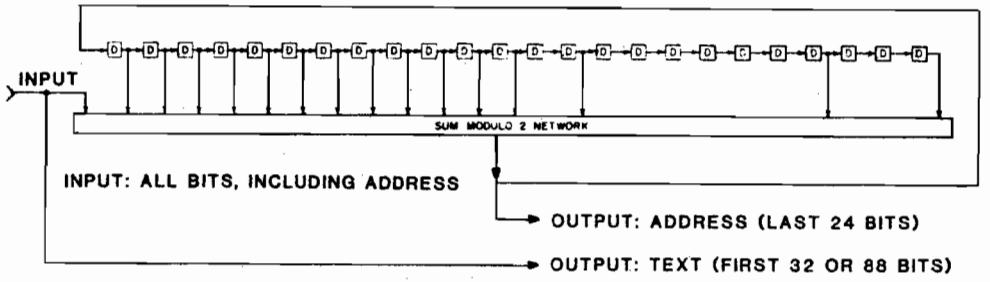
SENSOR ENCODER



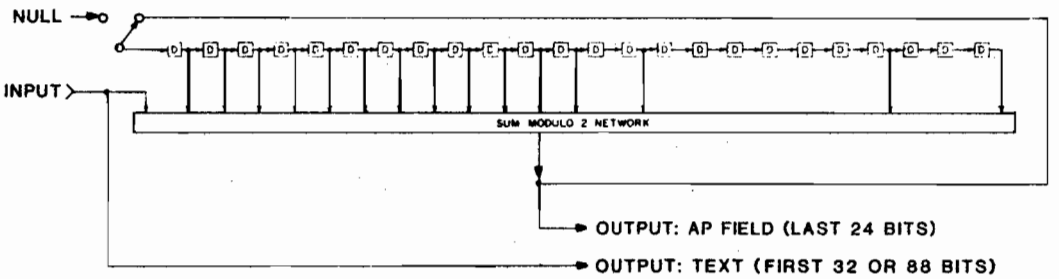
INPUT: ALL BITS, INCLUDING ADDRESS

SWITCH: UP EXCEPT FOR LAST 24 BITS

TRANSPONDER DECODER



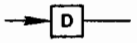
TRANSPONDER ENCODER



INPUT: ALL BITS, INCLUDING ADDRESS

SWITCH: RIGHT EXCEPT FOR LAST 24 BITS

ATC-117



1 BIT INTERVAL DELAY

FOR ALL CODERS: NULLS IN D AT START OF PROCESS

Fig. 5-1. Mode S implementation.

the operation. Thus, no multiplication of it by G(x) takes place. Instead, the input is merely added to the remainder being generated. Thus the AP field is now:

$$AP = R(x) + A(x) \qquad (5\text{-}3)$$

as desired.

The sensor decoder represents the major hardware complexity of the coding system. Figures 5-2, 5-3, and 5-4, taken from the Mode S specification, highlight the implementation. First, as shown in Fig. 5-2, the downlink message is entered into the A-Register, which is a division circuit of the type of Fig. 4-4. This circuit produces the remainder in parallel-readable form in the shift register. Thus, the remainder can be bit-by-bit added (compared) to the expected address to produce the error syndrome. Meanwhile, the message and confidence bits are being stored in the DB and CB registers respectively. The confidence test shown in the figure is discussed below.

If the syndrome is non-zero, an error burst is present. This burst can be in any 24-bit segment of the message. To produce the sequence of successively cyclic shifted syndrome patterns, the "reverse division" E-Register circuit of Fig. 5-3 is used. This circuit, as explained in Section 3.2, is filled by the initial syndrome in bit reversed order, and its taps implement the reciprocal polynomial G'(x) (compare the g coefficient order with Fig. 5-2). The explanation also indicated, as shown, that it has no input, only feedback. The CB and DB registers are tranferred to the L and M registers respectively, also in reverse order, so that the low-order 24-bits are the first set to be checked.

One shift at a time, the successively cycled syndrome is produced according to equation (3-19). In parallel, the message and confidence stream are cycled one-bit at a time. When the 1's of the syndrome pattern match the low confidence 1's in the low order 24-bits of the confidence bit pattern, the error has been trapped. The correction enable bit is then set by the error location function.

At this time, as shown in Fig. 5-4, the feedback of the E-Register is disabled, so that the syndrome can be read out serially. In parallel, the M register shifts out the message bits. Each bit corresponding to a 1 in the error syndrome is then corrected by adding the two streams bit-by-bit.

One further check is made during the detection phase of the correction process, namely the number of low confidence bits contained in each 24-bit segment of the message is determined. If the number of them ever exceeds a threshold, error correction is rejected. This is because the possibility of an erroneous correction goes up sharply with the number of low confidence bits. In the limit, if any 24 consecutive bits were low confidence, the syndrome pattern would be matched no matter what it was, and correction of those specific 24 bits would always occur.

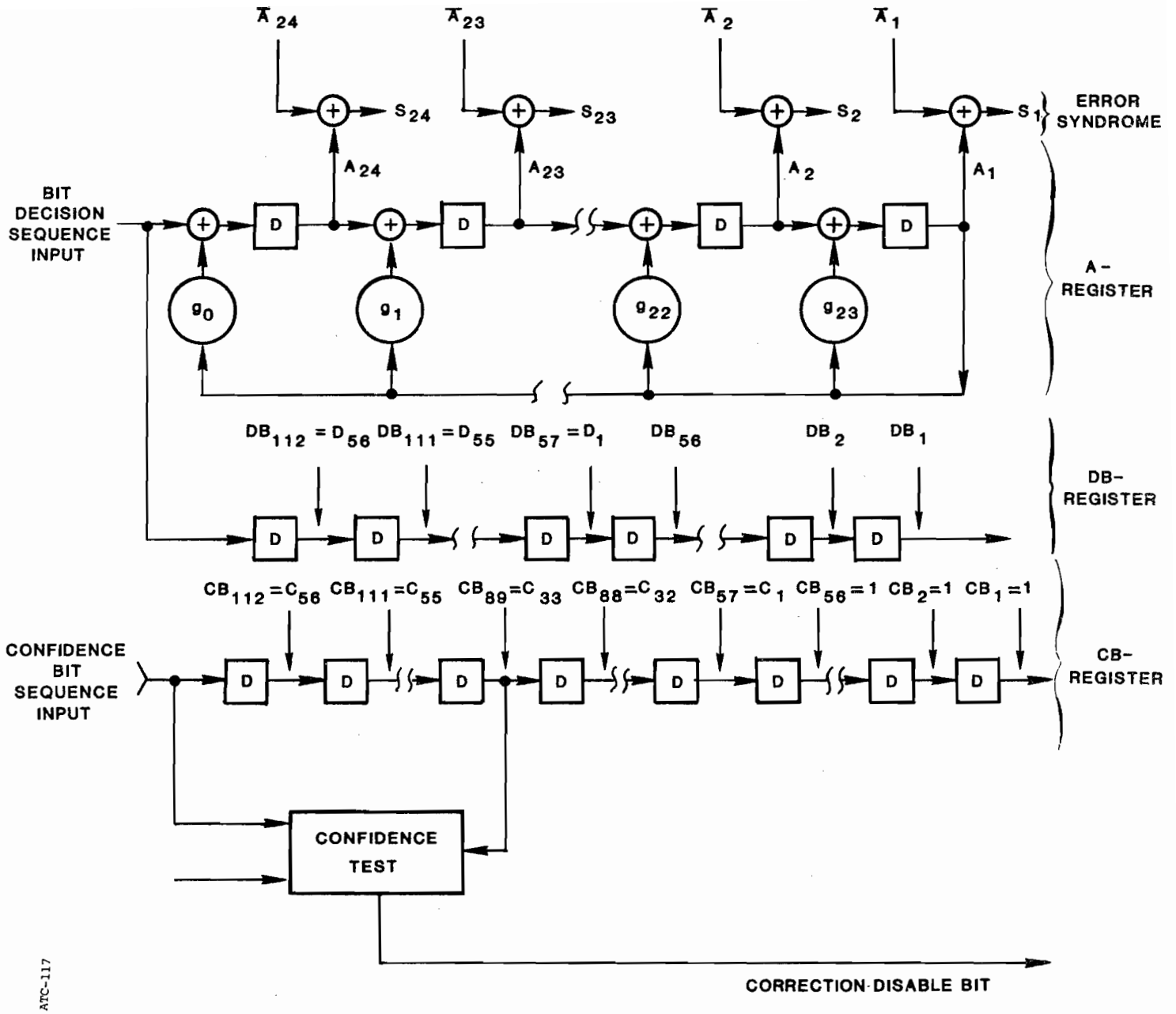


FIG. 5-2. ERROR DETECTION LOGIC

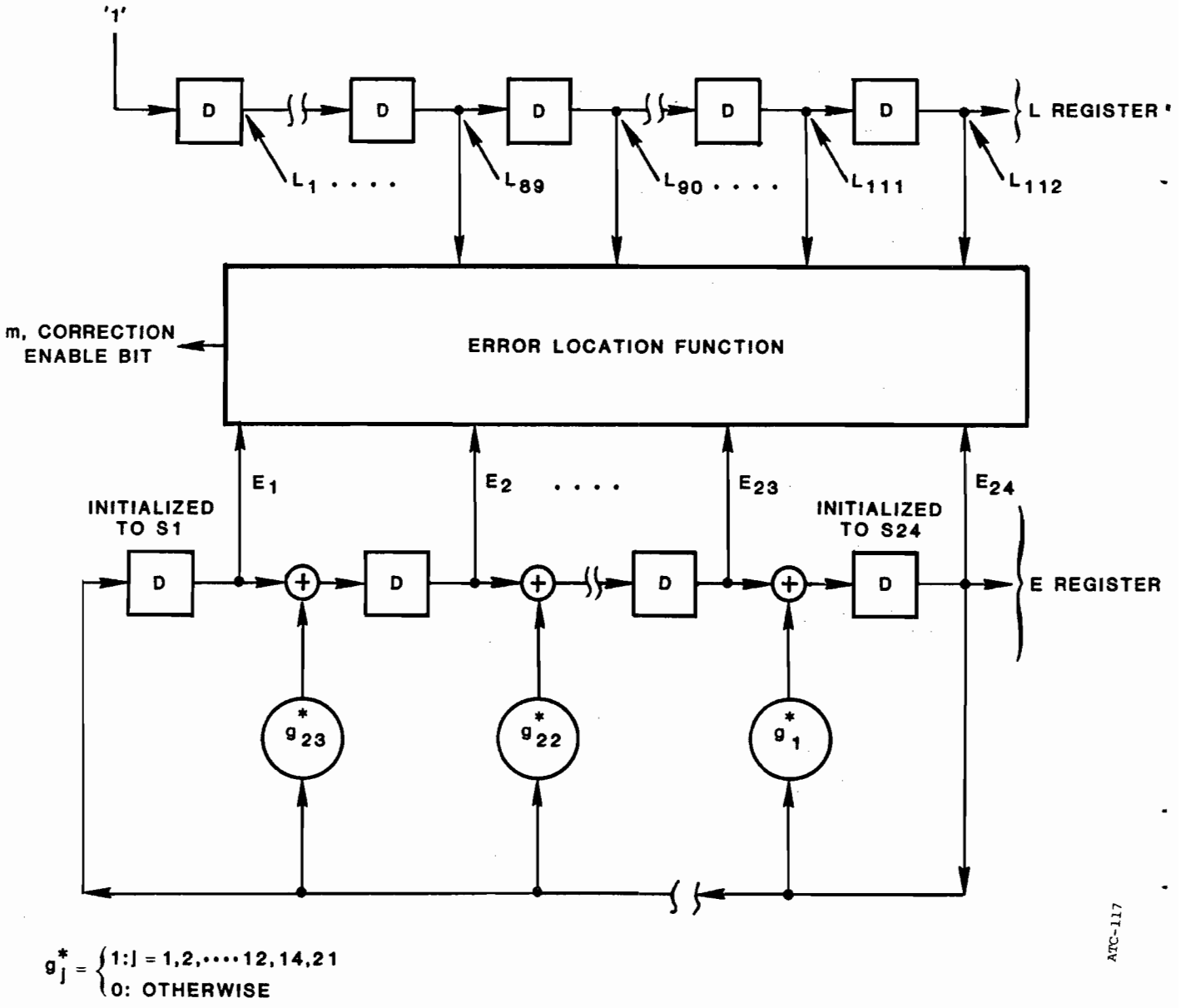
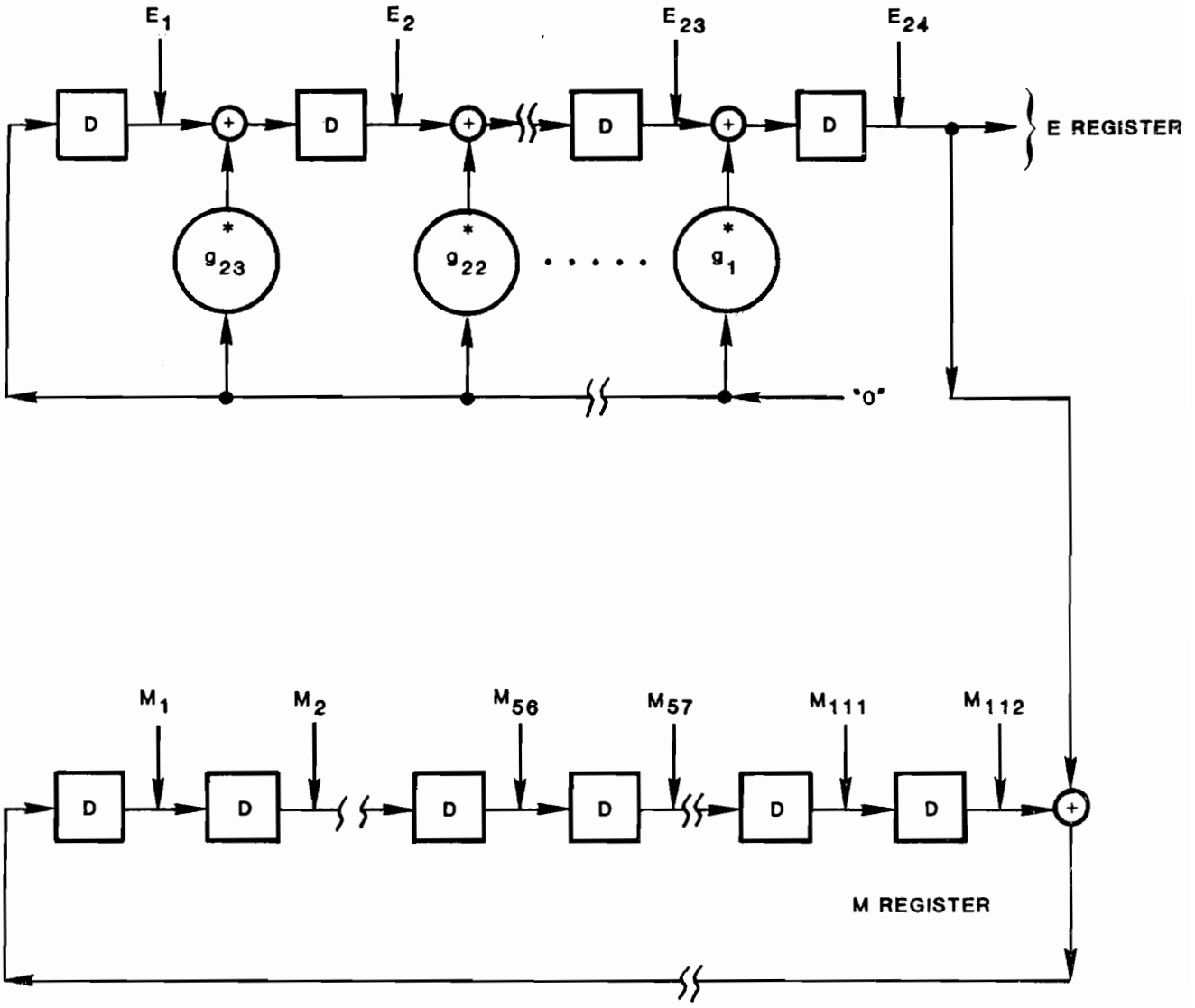


FIG. 5-3. ERROR LOCATION LOGIC



ATC-117

FIG. 5-4. ERROR CORRECTION LOGIC

REFERENCES

1. J. Barrows, "DABS Uplink Coding", ATC-49, Lincoln Laboratory, M.I.T. (25 July 1975).

2. J. Barrows, "DABS Downlink Coding", ATC-48, Lincoln Laborarory, M.I.T. (12 December 1975).

3. W. W. Peterson and C. J. Weldon, Jr., "Error Correcting Codes," Sec. Ed., M.I.T. Press, Cambridge, MA 1972).

4. T. Kasami and S. Matoba, "Some Efficient Shortened Cyclic Codes for Burst-Error Correction, "IEEE Trans. Inf. Theory IT-10, 252 (1964).