FAA-RD-74-162

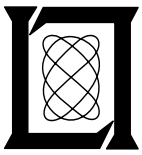
Project Report
ATC-40

DABS Uplink Encoder

J.R. Samson

4 March 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

ERRATA SHEET

DABS UPLINK ENCODER

PROJECT REPORT ATC-40 (FAA-RD-74-162)

Dated 4 March 1975

On page 26 the 29th row in Table II should read:

56 Bit 17	112 Bit 73	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
		1	1	1	1	1	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0

instead of:

56 Bit 17	112 Bit 73	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
		1	1	1	1	1	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0

Please make this change in the above-designated report.

7 April 1976

Publications Office
M. I. T. Lincoln Laboratory
P. O. Box 73
Lexington, Massachusetts 02173

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
FAA-RD-74-162		

4. Title and Subtitle	5. Report Date
DABS Uplink Encoder	4 March 1975
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
J.R. Samson	ATC-40

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS) 45364
Massachusetts Institute of Technology Lincoln Laboratory P.O. Box 73 Lexington, Massachusetts 02173	Project No. 034-241-012
	11. Contract or Grant No. IAG-DOT-FA72WAI-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 F19628-73-C-0002.

16. Abstract

This report explains the operation of the DABS uplink encoder and provides information useful in diagnosing its performance. Several techniques which may be useful in analyzing encoder operation are presented. One technique involves only the use of address-parity tables and modulo-2 addition. The address-parity tables included are based upon the encoding polynomial prescribed in "Provisional Signal Formats for the Discrete Address Beacon System," Lincoln Laboratory Project Report ATC-30 Rev.1, dated 25 April 1974.

The error detection and correction properties of polynomial encoding schemes are not the subject of this report.

17. Key Words		18. Distribution Statement
DABS Error Protection Parity Check Coding Data Transmission	Encoder Decoder	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	22. Price
Unclassified	Unclassified	46	

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION 1

INTRODUCTION

This report describes the functional operation and construction of the DABS [Ref. 1] uplink encoder, and provides a basis for analyzing uplink encoder operation. The explanation presented is based on a detailed examination of encoder operation, noting its linearity and its basic operations of division, multiplication, and addition. Although only uplink encoder processing is treated directly in this report, many of the ideas can be extended to the other encoding/decoding processing steps in the DABS uplink/downlink cycle.

Particular emphasis is given to the development of techniques and basic test patterns useful in checking for proper encoder operation. Two techniques are discussed in detail, one of which involves the use of address-parity tables and requires only modulo-2 addition to compute the correct encoded output for any data block input.

This report is divided into five major parts. The first section presents a general discussion of the processes involved in the DABS uplink encoder/decoder operation. The second section develops a standard method of representing binary sequences as algebraic polynomials and illustrates the algebraic manipulation of these polynomials which will be useful in "analyzing" encoder operation. An example is presented in the third section

illustrating how the notation and manipulations discussed in the previous section are applied to a DABS-type uplink encoder. The fourth section presents a procedure by which anyone armed with a set of address-parity tables and a knowledge of modulo-2 addition can compute the address/parity output for any data block sequence. The address-parity tables for the DABS uplink encoder, using the encoding polynomial prescribed in ATC-30 Rev. 1, "Provisional Signal Formats for the Discrete Address Beacon System" [Ref. 2], are included in this section. An explanation of how the tables are prepared is also given. The fifth section presents three basic tests which can be performed on any DABS-type encoder to indicate whether it is functioning properly.

SECTION 2

GENERAL

2.1 TERMINOLOGY AND APPROACH

A DABS interrogation data block consists of three segments: the sync burst, the information field, and the address or address/parity field. This segmentation of the clear text data block, prior to encoding, is illustrated in Fig. 1-a below. Figure 1-b illustrates the segmentation following the encoding process.

ATC-40(1)

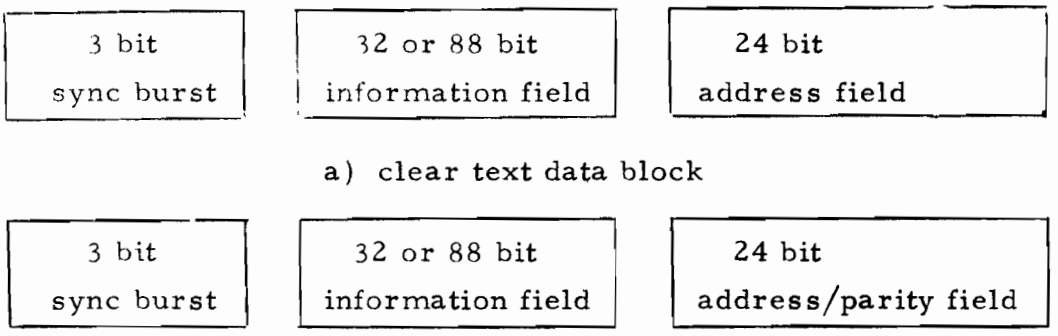


Fig. 1. Segmentation of DABS Interrogation Data Block.

The first 3 bits in the uplink data block form the sync burst, a 0 - 1 - 0 transmission in which the "1" in the 2nd bit is designated as the sync phase reversal. This sync phase reversal is used to 1) synchronize the transponder's airborne clock, and 2) provide an absolute reference for

the timing of transponder replies following the receipt of a valid interrogation. Since the sync burst bits have no effect on the encoding process, they will not be discussed further.

The remaining data block segments will be jointly referred to in this report as the "info/address block" or the "message block". The key to understanding encoder-decoder processing is the resolution of the processing into superposable operations. (The encoding and decoding mechanisms are linear processes, so that the application of the principle of superposition is valid.) The address/parity field output of the DABS uplink encoder can be considered as the superposition (a modulo-2 sum, symbolized by $\oplus$) of the outputs of two distinct processing steps. These two outputs are designated: 1) parity, and 2) encoded address; and the resultant address/parity field output can be expressed as:

$$\text{address/parity} = \text{parity} \oplus \text{encoded address}$$

where the summing is done on a bit by bit basis.

The parity is generated solely from the content of the information field. It is actually the final remainder following the "division" of the information field bit sequence by the bit sequence corresponding to the encoding polynomial.

The encoded address is generated solely from the content of the address field by the "multiplication" of the clear text address bit sequence by the sequence corresponding to the encoding polynomial.

In this analysis, the encoding process is explained as if it were two separate operations: multiplication, to yield the encoded address, and

division to yield parity. There is no interaction between the two processes until the results are superposed to yield address/parity. These operations take place, of course, in the encoding mechanism on a bit by bit basis, and while it is difficult to separate the operations as they are occurring simultaneously, it is very helpful to be able to analyze them independently.

Figure 2 illustrates an effective decomposition of DABS uplink encoder/decoder processing.

2.2 ENCODING

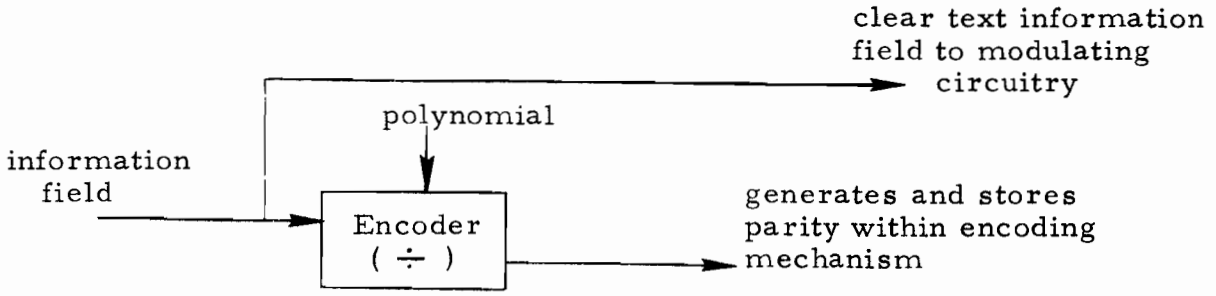
Starting with the first bit of the information field, the encoder performs a division operation which continually generates and "stores" the quotient (and indirectly a remainder) as each bit of the information field is clocked into the encoder. The remainder used in the computation of address/parity is the final remainder after the last bit of the information field has entered the encoder.* During this division process, the clear text (unencoded data) information field has been transmitted (Fig. 2-a).

Starting with the first bit of the address field, the encoder performs a multiplication operation generating the encoded address. As the encoded address is computed, the parity is added (modulo-2) and the combined address/parity transmitted (Fig. 2-b).

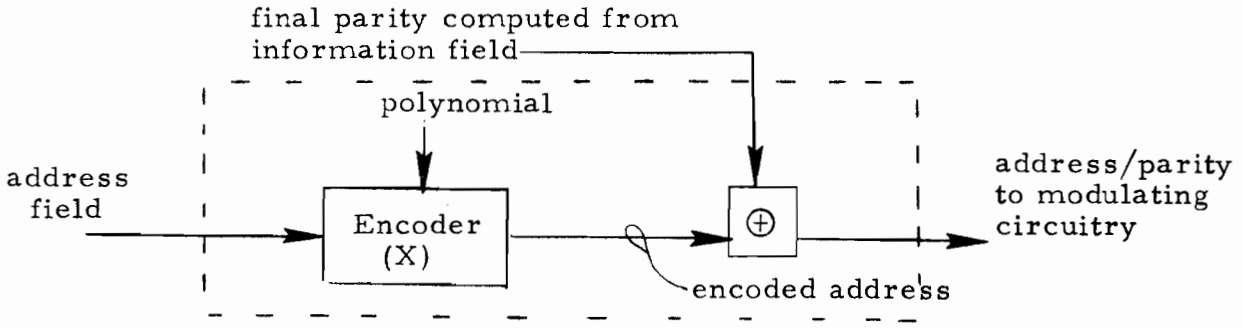
2.3 DECODING

As the clear text information field is received, it is simultaneously

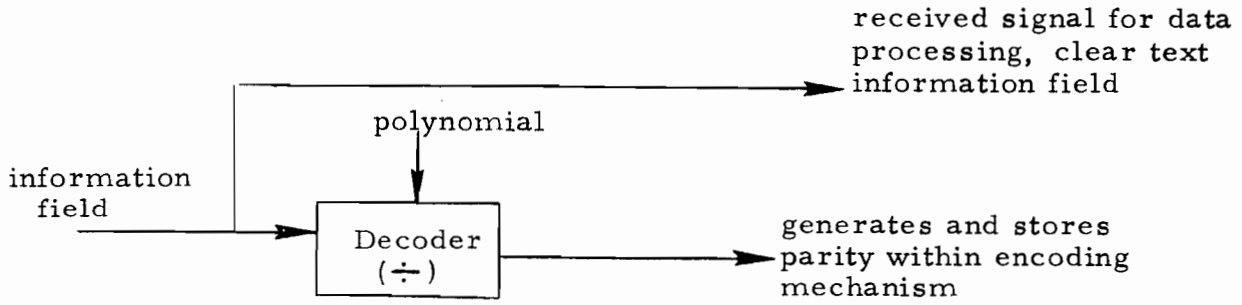
*This is a convenient way to think of the parity generating process, although the final remainder is actually computed during the encoding of the address field. This remainder has been "stored within the encoding mechanism" as the quotient of a division process, which, as it is clocked out of the encoder does yield the parity sequence.



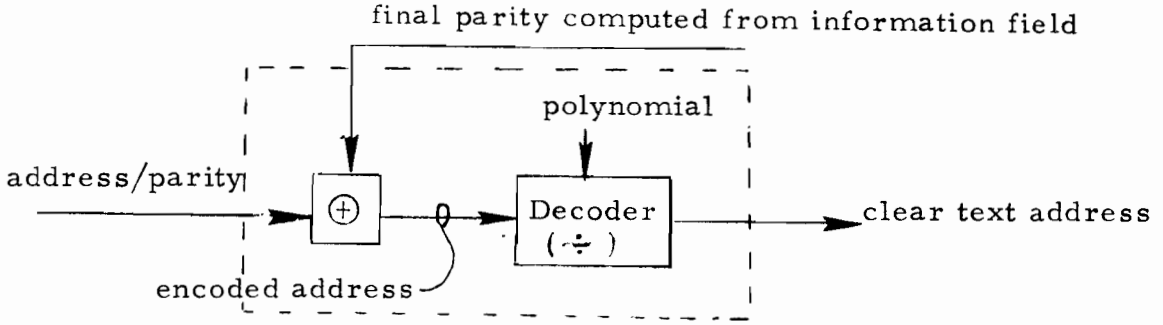
a) encoder processing during information field



b) encoder processing during address field



c) decoder processing during information field



d) decoder processing during address/parity field

Fig. 2. Effective Decomposition of DABS Uplink Encoder - Decoder Processing.

fed to the next stage of data processing and to the decoder. The decoder performs a division operation which generates and stores the parity (Fig. 2-c), in the same manner as the encoding process. As the address/parity field is received, the parity is subtracted (still a modulo-2 addition) yielding the encoded address. The decoder performs a division operation on the encoded address which should yield a clear text address (Fig. 2-d).

The transponder will compare the decoded address with its assigned address and respond only if they agree. Incorrect parity or decoded address (caused by interference), or just the wrong address (indicating that the interrogation was not meant for this transponder) will be flagged and the interrogation ignored by the transponder. This encoding-decoding processing markedly increases the reliability of the discretely addressable transponder system, since the probability of an undetected error in received information or address has been made extremely low.

2.4 THE DABS UPLINK ENCODER

The DABS encoder consists of a 24 bit shift register, parity checkers (EXCLUSIVE OR gates connected in tandem), and steering logic as shown in Fig. 3. The shift register outputs are tapped in sequence according to the encoding polynomial, and fed into the parity checkers. The input line marked "a" controls the steering logic which alters the processing operations performed on different portions of the data block. At the start of the encoding process, the shift register must be initialized to "0". (Note: If the initialization has been done prior to the sync burst bits, provision must be made to ensure that the sync burst bits have not entered the encoder, or that the encoder has been

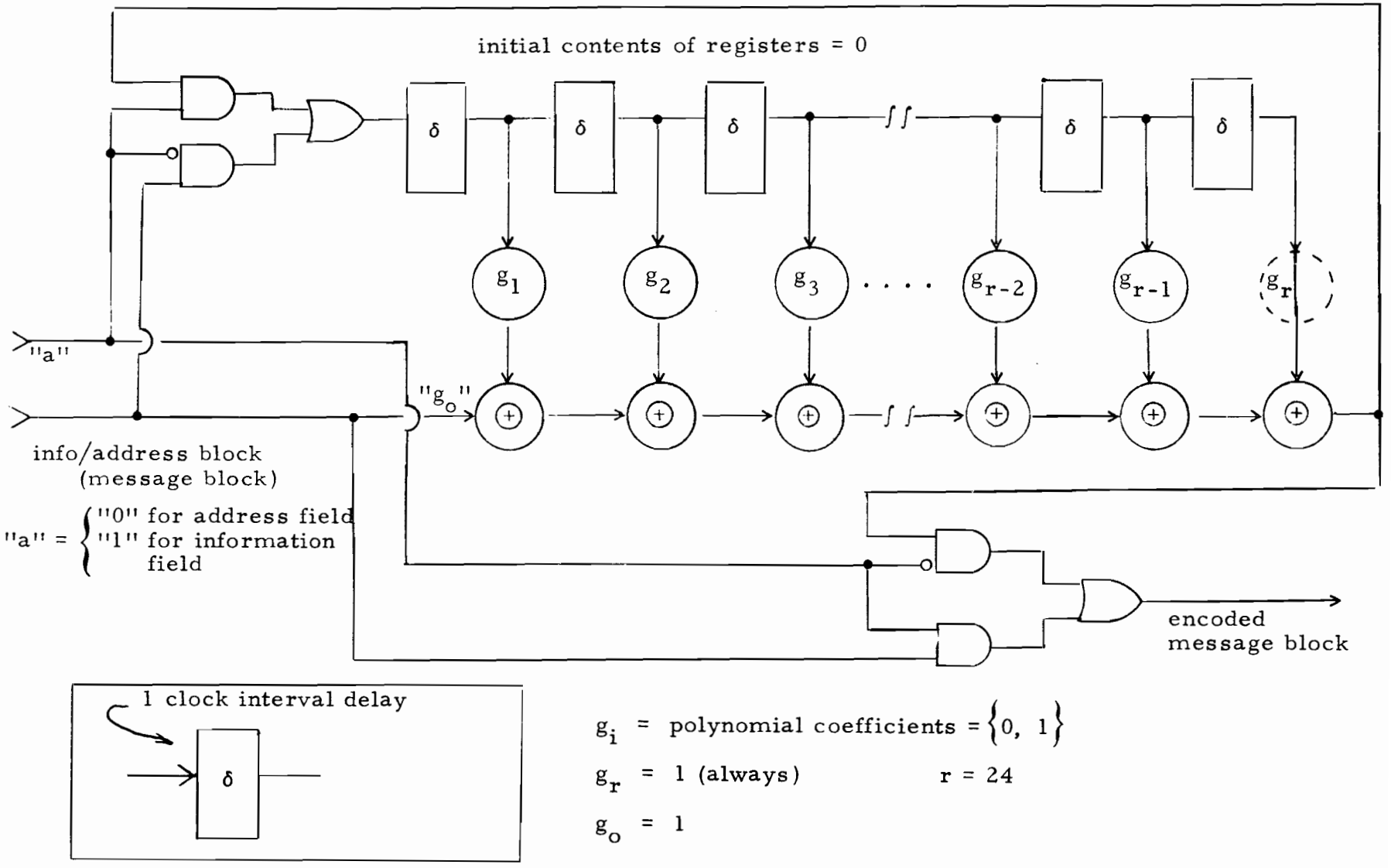
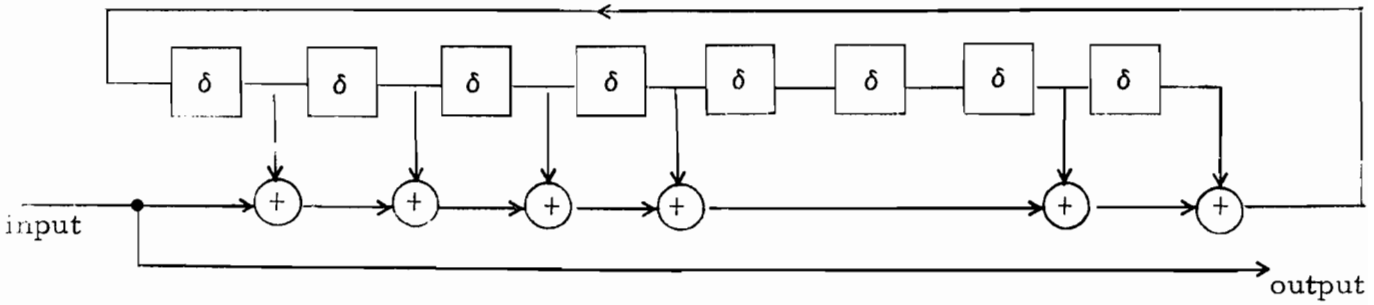


Fig. 3. DABS Uplink Encoder.

re-initialized prior to the 1st bit of the information field.)

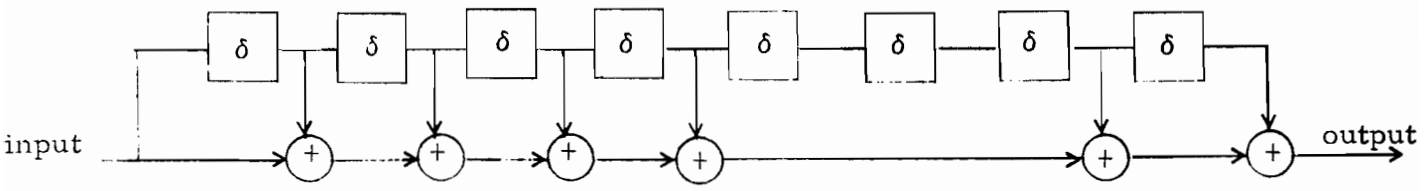
During the entry of the information field, the "a" line is held at a logical "1" level. This puts the encoder into the configuration:



which 1) allows the information field to enter the encoder serially, 2) causes the output of the last parity checker to be fed back into the input of the shift register, and 3) passes the information field (clear text) to the modulating circuitry. In this feedback configuration, the encoder performs its "division" operation on the incoming information field sequence. After the last bit of the information field is clocked into the encoder, the sequence needed to compute the final remainder (parity) is stored in the shift register. This sequence is actually the last 24 bits of the quotient generated by the division operation. The final parity bits are not computed until they are output at the proper time in the address/parity field.

After the last bit of the information field has entered the encoder, the "a-line" is switched to a logical "0" state. This puts the encoder into the configuration:

REGISTERS WITH STORED QUOTIENT



Now consider the action of the reconfigured encoder upon inputting the address field. If we assume for a moment that the information field was all 0's, yielding all 0 parity, the encoder will perform its multiplication operation on the incoming address sequence and yield simply the encoded address output sequence. However, since the encoder is a linear processor, the parity generated by the stored quotient will be added to the encoded address (modulo-2 addition dictated by the EXCLUSIVE OR gates) to yield the address/parity output. This addition process is performed on a bit by bit basis, as the encoded address and final parity bits are simultaneously computed. As far as the generation of address/parity is concerned, one could view the operation as entirely multiplication - considering the encoded address as the forced solution due to the address input, and the parity as the natural solution due to the initial conditions stored in the register at the time the solution starts (at the first address bit). The net response of this linear system is the forced solution plus the natural solution.

The encoder multiplication process may be recognized by clocking a non-zero bit through the shift register and parity checkers. For a single non-zero bit appearing at the input to the encoder in the no-feedback configuration, the encoder will generate an output sequence over the next 24

clock periods which corresponds to the coefficient pattern of the encoding polynomial. In essence, this single bit sequence has been multiplied by the sequence corresponding to the encoding polynomial. In a similar manner, the division process may also be recognized. For a single non-zero bit appearing at the input to the encoder in the feedback configuration, the shift register will contain the most recent 24 bits of the computed quotient over subsequent clock periods. In this case, the quotient results from the division of this single bit sequence by the sequence corresponding to the encoding polynomial.

Note: Clocking an entire data block through the shift register and parity checkers (switching the operating configuration appropriately) will yield the required address/parity. This technique may be useful in analyzing encoder operation.

SECTION 3

POLYNOMIAL NOTATION AND ARITHMETIC OPERATIONS

Bit sequences may be conveniently represented by the algebraic polynomial. The basic arithmetic associated with multiplying and dividing polynomials may be used (i.e., multiplication is performed on a term by term basis, where $x^i \cdot x^j = x^{i+j}$), but the encoding mechanism dictates that any addition (or subtraction) operations be performed in modulo-2 arithmetic.* The manipulations presented in this section correspond to the processes taking place in the encoder.

The algebraic polynomial in the variable x is expressed as a sum of increasing powers of x, with the degree of the polynomial specified as the highest power of x in the expression. The power of x indicates the bit position (delay) relative to the start of the sequence, and the coefficient of each power of x corresponds to the state of the bit (either 0 or 1).

For example, consider two 8-bit sequences:

sequence ⓐ 1 0 0 1 1 0 0 1

and sequence ⓑ 1 1 0 0 0 0 0 1

---

* modulo-2 arithmetic is provided by the EXCLUSIVE OR function, $0 \oplus 0=0$, $0 \oplus 1=1$, $1 \oplus 0=1$, and $1 \oplus 1=0$

These two sequences may be represented as *:

$$\text{sequence (a)} \qquad \underline{1} + \underline{0}x + \underline{0}x^2 + \underline{1}x^3 + \underline{1}x^4 + \underline{0}x^5 + \underline{0}x^6 + \underline{1}x^7$$

$$\text{sequence (b)} \qquad \underline{1} + \underline{1}x + \underline{0}x^2 + \underline{0}x^3 + \underline{0}x^4 + \underline{0}x^5 + \underline{0}x^6 + \underline{1}x^7$$

or more concisely as $1+x^3+x^4+x^7$ and $1+x+x^7$ respectively.

The power of x notation is a convenient bookkeeping procedure which keeps the sequences in proper time order; each unit increase in the power of x corresponds to 1 bit position delay from the start of the sequence.

The multiplication of polynomials representing these sequences is illustrated below:

$$\begin{array}{l} 1 + x^3 + x^4 + x^7 \\ \underline{\quad 1 + x + x^7} \\ 1 \;+\; x^3 \;+ x^4 \qquad + x^7 \\ \quad x \qquad + x^4 + x^5 \qquad + x^8 \\ \oplus \qquad\qquad\qquad\quad x^7 \qquad + x^{10} + x^{11} + x^{14} \\ \hline 1 + x + x^3 + x^5 + x^8 + x^{10} + x^{11} + x^{14} \end{array}$$

The output sequence corresponding to this multiplication would be 1 1 0 1 0 1 0 0 1 0 1 1 0 0 1, but normally if we multiply two 8-bit sequences, we probably desire (or are restricted to) an 8-bit output. This requires selective truncation of the total output sequence indicated above. Truncating after the 8th bit would yield the output sequence 1 1 0 1 0 1 0 0, the desired result.

---

* Coding theorists prefer to represent the 1st bit by the highest power of x - it seemed more advantageous to represent it as the lowest power of x. The results are the same in either case.

Division is handled similarly, with the exception that the subtraction to generate the partial remainders is still a modulo-2 addition. The division of sequence (a) by sequence (b) is illustrated in Fig. 4. The algebraic quotient, truncated to the first 8 terms, would yield a sequence 1 1 1 0 1 1 1 1. We may be interested in a particular remainder after a certain number of divisions. Take for example, the remainder after two divisions (point A in Fig. 3). If we are still interested in the first 8 terms, we might consider the remainder sequence 0 0 1 1 1 0 0 0. The power of x in the quotient indicates how many terms (bit position delay-wise from the start of the sequence) the division has progressed, i. e., a term x^4 indicates that the division has covered the first 5 bits in the sequence.

One item to keep in mind when working with these polynomial representations is that the sequences start with the term x^0 (= 1), i. e., the term x^7 corresponds to the 8th bit in the sequence.

In the next section, this method is applied to the computation of the address/parity output of a DABS-type uplink encoder.

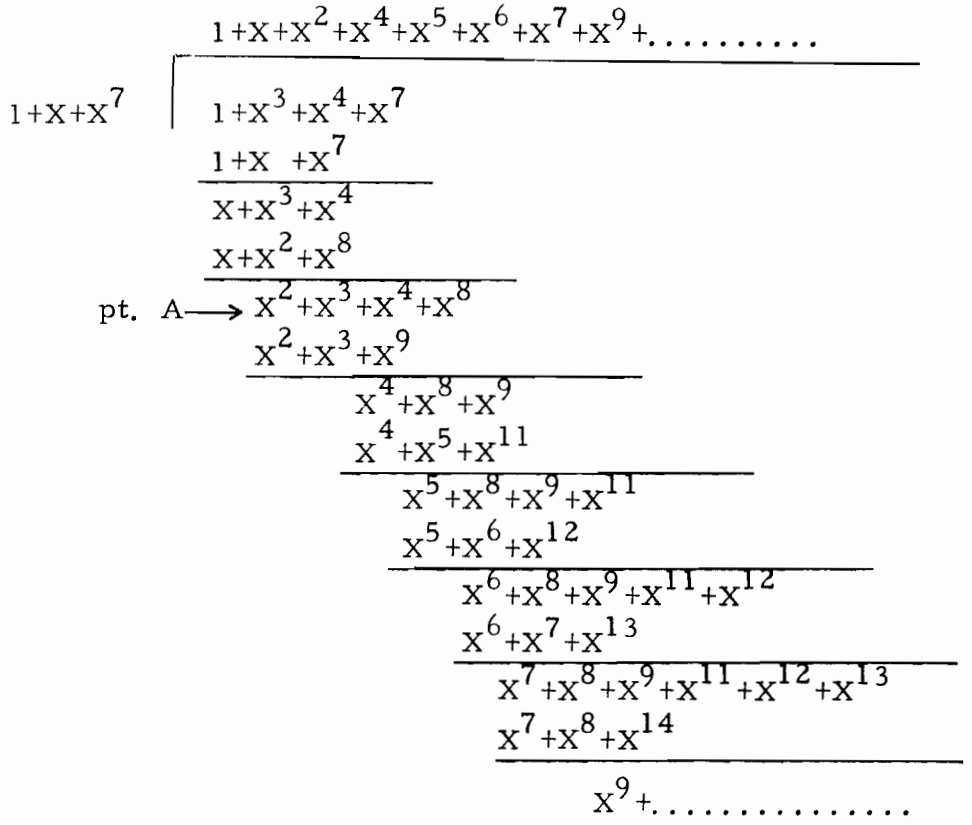


Fig. 4. Polynomial Division Corresponding to Division of Binary Sequences, 10011001 Divided by 11000001.

SECTION 4

COMPUTING THE ADDRESS/PARITY USING ALGEBRAIC POLYNOMIALS

The use of algebraic polynomials to compute the address/parity output of a DABS-type uplink encoder* is illustrated in this section. An example is worked out for a 56-bit message block using $1 + x^6 + x^{12} + x^{24}$ as the encoding polynomial.

The computation of address/parity is performed in three steps: 1) computation of parity, 2) computation of encoded address, and 3) superposition of the results of the first two steps.

In order to compute the parity, the polynomial representing the information sequence is divided by the encoding polynomial. The division is continued until it has progressed through the 32 bits of the information field (88 bits for a 112-bit message block). The stopping point is recognized when the quotient first shows a term x^y in which $y \geq 32$ (or 88). The remainder prior to this last division corresponds to the required parity.

This computation is illustrated in Fig. 5 for an information field sequence 10000000001100000000000000001000, represented by the polynomial

*In order to illustrate the procedure, an encoding polynomial with fewer non-zero coefficients than the actual DABS polynomial is assumed. The encoder processing is the same and the computational procedure can be applied to the encoder using the full DABS polynomial.

$1 + x^{10} + x^{11} + x^{28}$. The polynomial representing the parity, $x^{34} + x^{40} + x^{42} + x^{53}$, corresponds to an output sequence 00100000101000000000100 in the address/parity field.

In order to compute the encoded address, the polynomial representing the address sequence is multiplied by the encoding polynomial, and the resulting polynomial truncated after the x^{23} term. This computation is illustrated in Fig. 6 for an address field sequence 11000000000000100000000, represented by the polynomial $1 + x + x^{15}$. The polynomial representing the encoded address, $1 + x + x^6 + x^7 + x^{12} + x^{13} + x^{15} + x^{21}$, corresponds to the sequence 11000011000011010000100 in the address/parity field.

Note: The encoded address was computed using a "base-line" address, i.e., as if the address (and hence the resultant encoded address) started at bit #1. To shift the baseline address to its proper position in a 56-bit message block, the base-line address polynomial has to be multiplied by the term x^{32} (and likewise by x^{88} for a 112-bit message block). However, this would simply multiply the encoded address output polynomial by the same term. The output sequence in the address/parity field is the same in either case. The base-line computation is sufficient, as long as we remember to shift the encoded address output into the correct address/parity field. For the 56-bit message block in this example, the encoded address polynomial corresponding to the correct address/parity field is $x^{32} + x^{33} + x^{38} + x^{39} + x^{44} + x^{45} + x^{47} + x^{53}$.

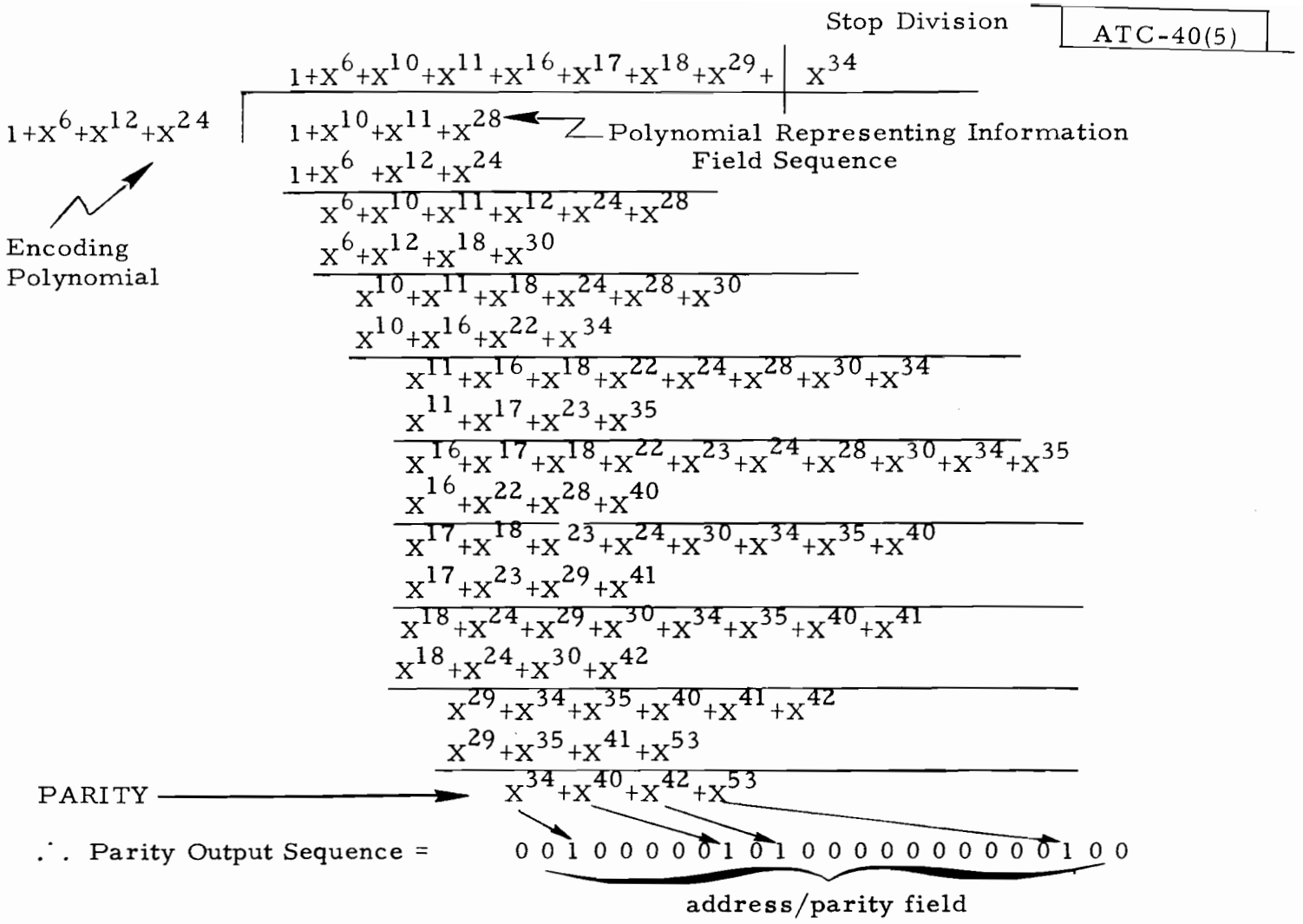
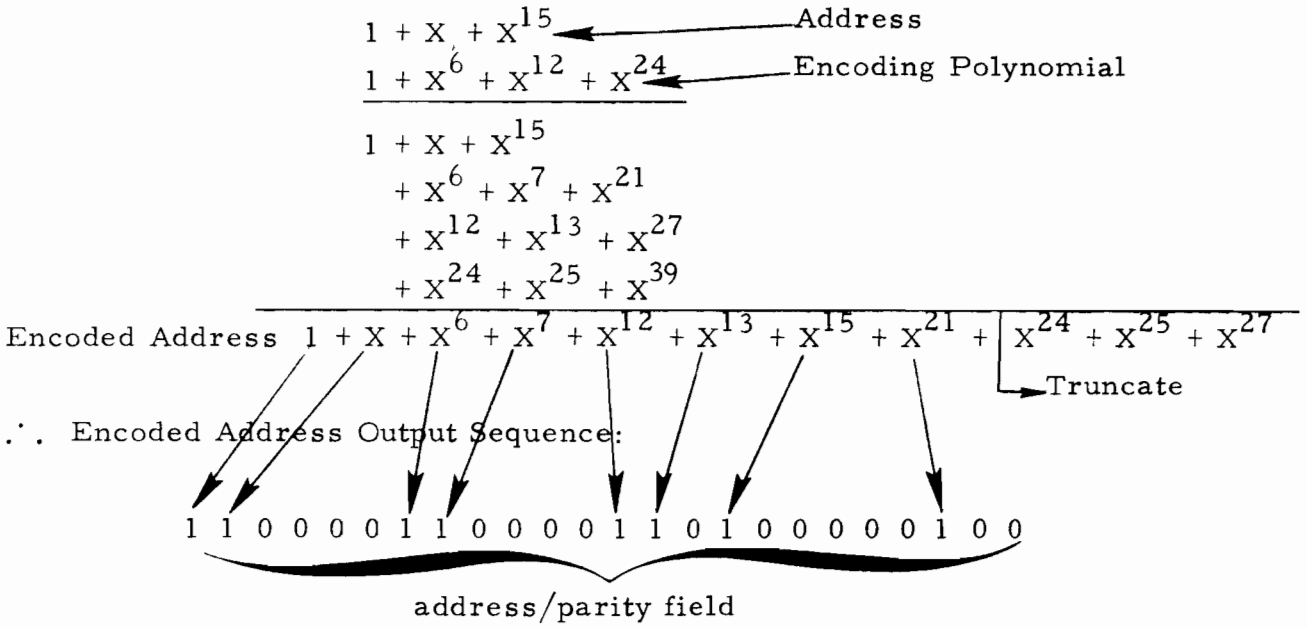


Fig. 5. Example: Computation of Parity Output Sequence Using Polynomial Division.



Note: This computation was performed on a "base-line" address, i. e., as if the address started with bit #1. This yielded an encoded address output which also started at bit #1. The address actually starts at bit #33 of a 56 bit message block and at bit #89 of a 112 bit message block, and so does the encoded address. The base-line computation simply has to be shifted up into the correct address/parity field.

Fig. 6. Example: Computation of Encoded Address Output Sequence Using Polynomial Multiplication.

In order to compute the address/parity, the polynomials corresponding to parity and encoded address are combined by the modulo-2 addition of terms with common powers of x (i. e., terms that correspond to the same position in the two sequences). Thus, the polynomial corresponding to address/parity is $x^{32} + x^{33} + x^{34} + x^{38} + x^{39} + x^{40} + x^{42} + x^{44} + x^{45} + x^{47}$, representing sequence 111000111010110100000000. (Note: This output sequence could have been obtained from a modulo-2 addition of the parity and encoded address sequences, without reverting to the polynomial representations.)

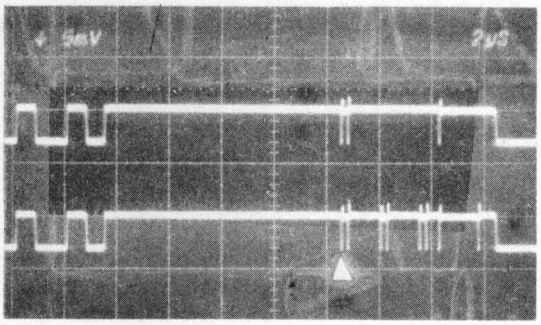
Figure 7 illustrates the results of this example as generated by the uplink encoder in the portable DABS Interrogator Simulator which was set up with the example polynomial. The waveforms are the detected RF uplink transmissions. The downward "spikes" within the DABS data block envelope correspond to phase changes generated by 1's in the DPSK modulation.* The photographs show the input-output relationships for the three steps in the computation, and verify the hand-computed sequences for parity, encoded address, and address/parity. The triangle on each photograph indicates the 33rd bit of the message block or the 1st bit of address/parity. At a 4-Megabit/sec rate, there are 8 bits per vertical division (2μ sec/vertical division).

Hand-computation of the address/parity output is laborious, particularly

*In DABS uplink interrogations, the data block is transmitted using DPSK (Differential Phase Shift Keying) Modulation. This modulation scheme uses the convention that a "1" for any bit generates a 180° phase reversal in the carrier for that bit, a "0" produces no phase reversal. Due to finite system bandwidth limitations, variations in carrier amplitude accompany each phase reversal. Envelope detection can discern these amplitude variations, providing a convenient display of the DPSK modulation.

Clear Text Address:
(w/all 0 information field)

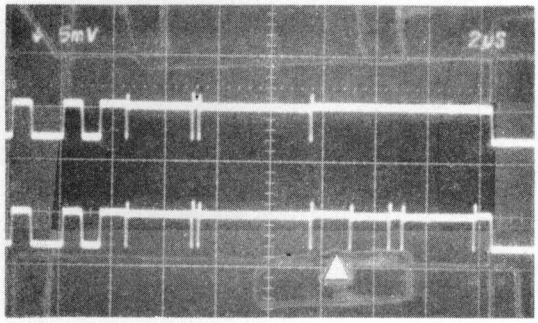
Encoded Address:



a) Input Address Only Encoded Address

Information Field:
(w/all 0 address field)

Parity:



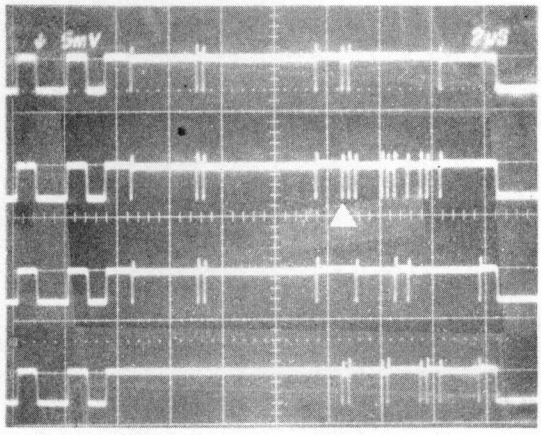
b) Input Information Field Only Parity

Information Field
Plus Clear Text Address:

Address/Parity:

Parity:

Encoded Address:



c) Input Information Plus Address Address/Parity

Fig. 7. Example: Encoded Address, Parity, and Address/ Parity Generated by DABS-type Uplink Encoder.
(Polynomial = $1 + x^6 + x^{12} + x^{24}$.)

for an info/address block and/or an encoding polynomial with many non-zero bits. In the next section, a simple method for computing address/parity, in which all the "difficult" work has been done, is presented.

SECTION 5

COMPUTING THE ADDRESS/PARITY BITS FROM TABLES

In the previous sections it was shown how resolving the encoding process into two distinct operations and then superposing the results simplifies the solution to (and understanding of) the problem of computing address/parity. The superposition aspect of the encoding operation may be further exploited by generating a set of tables which permit simple calculation of address/parity.

The first part of the procedure is to compute the contribution to address/parity due to each bit in the info/address block acting individually. Superposition then says that if the contribution of each individual bit is known, the address/parity due to the information field and address field input sequences may be computed by adding the results due each bit in the input acting individually. The tables provide the required output sequences.

The address and parity output sequences corresponding to the encoding polynomial prescribed in Reference 2* are given in Tables I and II respectively. In order to use the tables, one simply performs a modulo-2 addition of the

---

*The encoding polynomial prescribed in Reference 2 has the form:

$$g(x) = \sum_{i=0}^{i=24} g_i x^i, \text{ where } g_i = \begin{cases} 1 \text{ for } i = 0 \text{ through } 12,\ 14,\ 21 \text{ and } 24 \\ 0 \text{ otherwise} \end{cases}$$

TABLE I

ENCODED ADDRESS* (ADDRESS/PARITY OUTPUT SEQUENCES DUE TO A "1" IN EACH BIT OF ADDRESS)

Clear Text Address Bit Numbers	Address/Parity Bit Numbers →																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	1	0	0
2	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	1	0
3	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	1
4	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0
5	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0
6	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0
7	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0
8	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0
9	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0
10	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
11	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0
12	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
13	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

*Only for DABS uplink encoder using polynomial prescribed in Reference 2.

Note: The 1st bit in the address or address/parity field is the 33rd (89th) bit of the message block for a 56 (112) bit message.

TABLE II

PARITY* (ADDRESS/PARITY OUTPUT SEQUENCES DUE TO A "1" IN EACH BIT OF INFORMATION FIELD)

Information Field Bit Numbers		Address/Parity Bit Numbers →																						
56 Bit	112 Bit	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
-	1	0	0	1	1	1	0	0	1	0	0	1	1	0	1	0	1	1	1	1	0	1	0	1	0
-	2	0	0	0	1	1	1	0	0	1	0	0	1	1	0	1	0	1	1	1	1	0	1	0	1
-	3	1	1	1	1	0	0	0	1	1	0	1	1	0	1	1	1	0	1	1	1	1	1	1	0
-	4	0	1	1	1	1	0	0	0	1	1	0	1	1	0	1	1	1	0	1	1	1	1	1	1
-	5	1	1	0	0	0	0	1	1	1	0	0	1	0	1	1	1	1	1	0	1	1	0	1	1
-	6	1	0	0	1	1	1	1	0	0	0	1	1	0	0	0	1	1	1	1	0	1	0	0	1
-	7	1	0	1	1	0	0	0	0	1	1	1	0	0	0	1	0	1	1	1	1	0	0	0	0
-	8	0	1	0	1	1	0	0	0	0	1	1	1	0	0	0	1	0	1	1	1	1	0	0	0
-	9	0	0	1	0	1	1	0	0	0	0	1	1	1	0	0	0	1	0	1	1	1	1	0	0
-	10	0	0	0	1	0	1	1	0	0	0	0	1	1	1	0	0	0	1	0	1	1	1	1	0
-	11	0	0	0	0	1	0	1	1	0	0	0	0	1	1	1	0	0	0	1	0	1	1	1	1
-	12	1	1	1	1	1	0	1	0	0	1	1	1	1	1	0	1	0	0	0	1	0	0	1	1
-	13	1	0	0	0	0	0	1	0	1	1	0	0	0	1	0	0	1	0	0	0	1	1	0	1
-	14	1	0	1	1	1	1	1	0	1	0	0	1	1	0	0	0	0	1	0	0	0	0	1	0
-	15	0	1	0	1	1	1	1	1	0	1	0	0	1	1	0	0	0	0	1	0	0	0	0	1
-	16	1	1	0	1	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	1	0	1	0	0
-	17	0	1	1	0	1	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	1	0	1	0
-	18	0	0	1	1	0	1	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	1	0	1
-	19	1	1	1	0	0	1	0	1	1	1	1	1	0	0	0	1	1	0	0	0	0	1	1	0
-	20	0	1	1	1	0	0	1	0	1	1	1	1	1	0	0	0	1	1	0	0	0	0	1	1
-	21	1	1	0	0	0	1	1	0	1	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1
-	22	1	0	0	1	1	1	0	0	1	0	1	1	1	0	0	1	0	0	1	1	0	1	1	0
-	23	0	1	0	0	1	1	1	0	0	1	0	1	1	1	0	0	1	0	0	1	1	0	1	1
-	24	1	1	0	1	1	0	0	0	1	1	0	1	0	1	0	0	0	1	0	0	1	0	0	1
-	25	1	0	0	1	0	0	1	1	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
-	26	0	1	0	0	1	0	0	1	1	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0
-	27	0	0	1	0	0	1	0	0	1	1	1	0	0	1	0	0	0	0	0	0	1	0	0	0
-	28	0	0	0	1	0	0	1	0	0	1	1	1	0	0	1	0	0	0	0	0	0	1	0	0
-	29	0	0	0	0	1	0	0	1	0	0	1	1	1	0	0	1	0	0	0	0	0	0	1	0
-	30	0	0	0	0	0	1	0	0	1	0	0	1	1	1	0	0	1	0	0	0	0	0	0	1
-	31	1	1	1	1	1	1	0	1	1	0	1	1	0	1	0	0	0	1	0	0	0	1	0	0
-	32	0	1	1	1	1	1	1	0	1	1	0	1	1	0	1	0	0	0	1	0	0	0	1	0
-	33	0	0	1	1	1	1	1	1	0	1	1	0	1	1	0	1	0	0	0	1	0	0	0	1
-	34	1	1	1	0	0	0	0	0	0	1	0	0	1	1	0	0	1	0	0	0	1	1	0	0
-	35	0	1	1	1	0	0	0	0	0	0	1	0	0	1	1	0	0	1	0	0	0	1	1	0
-	36	0	0	1	1	1	0	0	0	0	0	0	1	0	0	1	1	0	0	1	0	0	0	1	1
-	37	1	1	1	0	0	0	1	1	1	1	1	1	0	0	1	1	1	0	0	1	0	1	0	1
-	38	1	0	0	0	1	1	1	0	0	0	0	0	0	0	1	1	1	1	0	0	1	1	1	0
-	39	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	1	1	1	1	0	0	1	1	1
-	40	1	1	0	1	1	1	0	0	0	1	1	1	1	0	1	0	1	1	1	1	0	1	1	1
-	41	1	0	0	1	0	0	0	1	1	1	0	0	0	1	1	1	0	1	1	1	1	1	1	1
-	42	1	0	1	1	0	1	1	1	0	0	0	1	1	0	0	1	1	0	1	1	1	0	1	1
-	43	1	0	1	0	0	1	0	0	0	1	1	1	0	1	1	0	1	1	0	1	1	0	0	1
-	44	1	0	1	0	1	1	0	1	1	1	0	0	0	0	0	1	0	1	1	0	1	0	0	0

-continued-

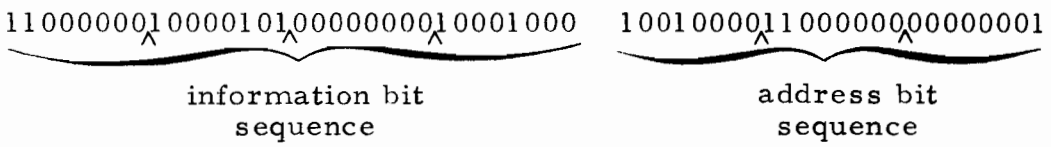
*Only for DABS encoder (uplink or downlink) using polynomial prescribed in Reference 2.

TABLE II (Continued)

Information Field Bit Numbers		Address/Parity Bit Numbers																							
56 Bit	112 Bit	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
-	45	0	1	0	1	0	1	1	0	1	1	1	0	0	0	0	0	1	0	1	1	0	1	0	0
-	46	0	0	1	0	1	0	1	1	0	1	1	1	0	0	0	0	0	1	0	1	1	0	1	0
-	47	0	0	0	1	0	1	0	1	1	0	1	1	1	0	0	0	0	0	1	0	1	1	0	1
-	48	1	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	0	0	0	1	0	0	1	0
-	49	0	1	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	0	0	0	1	0	0	1
-	50	1	1	0	0	0	0	1	0	1	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0
-	51	0	1	1	0	0	0	0	1	0	1	0	1	1	0	0	1	1	1	0	0	0	0	0	0
-	52	0	0	1	1	0	0	0	0	1	0	1	0	1	1	0	0	1	1	1	0	0	0	0	0
-	53	0	0	0	1	1	0	0	0	0	1	0	1	0	1	1	0	0	1	1	1	0	0	0	0
-	54	0	0	0	0	1	1	0	0	0	0	1	0	1	0	1	1	0	0	1	1	1	0	0	0
-	55	0	0	0	0	0	1	1	0	0	0	0	1	0	1	0	1	1	0	0	1	1	1	0	0
-	56	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	0	1	1	0	0	1	1	1	0
1	57	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	0	1	1	0	0	1	1	1
2	58	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0	0	1	0	1	1	0	1	1	1
3	59	1	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	1	1	1	1
4	60	1	0	1	1	1	1	1	1	1	1	0	0	1	0	0	1	0	0	1	0	1	0	1	1
5	61	1	0	1	0	0	0	0	0	0	0	0	1	1	1	1	0	1	0	0	1	0	0	0	1
6	62	1	0	1	0	1	1	1	1	1	1	1	1	0	1	0	1	0	1	0	0	1	1	0	0
7	63	0	1	0	1	0	1	1	1	1	1	1	1	1	0	1	0	1	0	1	0	0	1	1	0
8	64	0	0	1	0	1	0	1	1	1	1	1	1	1	1	0	1	0	1	0	1	0	0	1	1
9	65	1	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	1
10	66	1	0	0	0	1	0	1	0	1	1	1	1	1	0	0	0	0	1	0	1	0	0	1	0
11	67	0	1	0	0	0	1	0	1	0	1	1	1	1	1	0	0	0	0	1	0	1	0	0	1
12	68	1	1	0	1	1	1	0	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0
13	69	0	1	1	0	1	1	1	0	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0
14	70	0	0	1	1	0	1	1	1	0	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0
15	71	0	0	0	1	1	0	1	1	1	0	1	0	1	0	0	0	1	0	0	0	0	0	1	0
16	72	0	0	0	0	1	1	0	1	1	1	0	1	0	1	0	0	0	1	0	0	0	0	0	1
17	73	1	1	1	1	1	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0
18	74	0	1	1	1	1	1	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0
19	75	0	0	1	1	1	1	1	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	1
20	76	1	1	1	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0
21	77	0	1	1	1	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0
22	78	0	0	1	1	1	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0
23	79	0	0	0	1	1	1	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0
24	80	0	0	0	0	1	1	1	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0
25	81	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0
26	82	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0
27	83	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	0	1	1	0	0	0	0
28	84	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	0	1	1	0	0	0
29	85	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	0	1	1	0	0
30	86	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	0	1	1	0
31	87	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	0	1	1
32	88	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	1	0	0	1

output sequences due to each "1" bit in the info/address block. The modulo-2 addition corresponds to the EXCLUSIVE OR function for a many-variable input, i. e., if all the inputs are "0" - the output is "0", if the number of "1's" in the input is odd - the output is "1", and if the number of "1's" in the input is even - the output is "0"; there is no carry.

The procedure is best illustrated by an example. Consider the following input sequence (clear text) for a 56 bit info/address block:



First, compute the output parity sequence generated by the information bits. From the Parity Table (Table II) find the appropriate output sequence corresponding to each "1" in the information field. Modulo-2 addition of these sequences yields the output parity sequence:

from 1st bit	0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 1 0 1 1 0 0 1 1 1
from 2nd bit	1 1 1 1 1 1 1 1 0 0 1 1 1 0 0 0 1 0 1 1 0 1 1 1
from 9th bit	1 1 1 0 1 0 1 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 1 0 1
from 14th bit	0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0
from 16th bit	0 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 1
from 25th bit	0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 0
⊕ from 29th bit	0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 0 1 1 0 0
output parity sequence	0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0 0

Note: The modulo-2 addition is performed on each column of bits, all 0's → 0, even number of 1's → 0, odd numer of 1's → 1, and there is no carry. (This is equivalent to performing a multiple-input EXCLUSIVE OR function on each column.)

Next compute the encoded address sequence generated by the address bits. From the address table find the appropriate output sequence corresponding to each "1" in the address field. Modulo-2 addition of these sequences yields the encoded address sequence.

from 1st bit	1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 0 1 0 0
from 4th bit	0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 0
from 9th bit	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0
from 10th bit	0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1
⊕ from 24th bit	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
encoded address sequence	1 1 1 0 0 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 0 0 1 0

Modulo-2 addition of the output parity sequence and the encoded address sequence yields the resultant address/parity sequence:

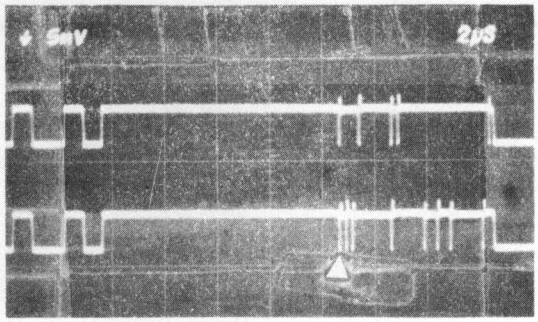
parity	0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0 0
⊕ enc. address	1 1 1 0 0 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 0 0 1 0
address/parity	1 1 0 0 1 0 0 1 1 1 0 0 1 1 1 1 1 1 0 1 0 1 1 0

Note: The modulo-2 additions could have been done in one step; it was broken down into three steps to emphasize the use of the two tables corresponding to the separation in processing that the encoder performs on the two segments of the input.

This example is graphically illustrated in Figs. 8 & 9. The photographs in these figures illustrate: 1) the contribution to address/parity due to the information field alone (parity), 2) the contribution due to the address field (encoded address), and 3) the resultant address/parity generated by the uplink encoder in the portable DABS Interrogator Simulator. The waveforms are detected uplink transmissions from the simulator. The downward "spikes"

Clear Text Address:
(w/all 0 information field)

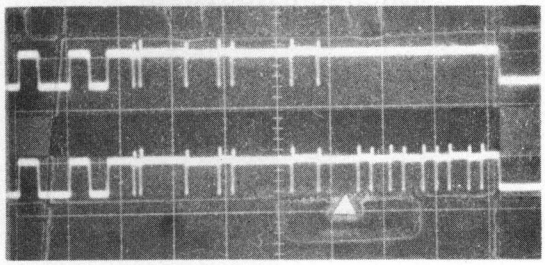
Encoded Address:



a) Input Address Only ⟶ Encoded Address

Information Field:
(w/all 0 address field)

Parity:



b) Input Information Field Only ⟶ Parity

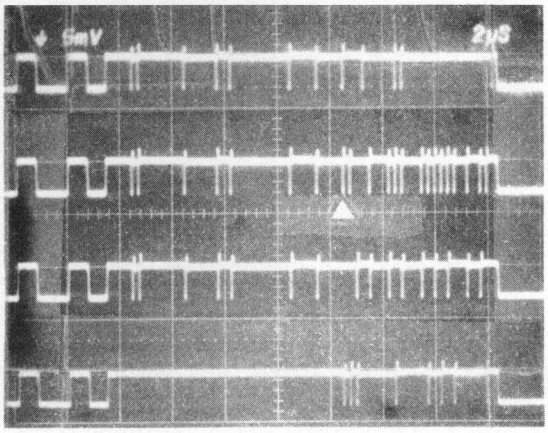
Information Field
Plus Clear Text Address:

Address/Parity:

=

Parity:

Encoded Address:



c) Input Information Plus Address ⟶ Address/Parity

Fig. 8. Example: Encoded Address, Parity, and Address/Parity Generated by DABS Uplink Encoder.

(Polynomial: As specified in Reference 1, "Provisional Signal Formats for the Discrete Address Beacon System Rev. 1".)

ATC-40(9)

Encoded Address:

Parity:

Address/Parity:

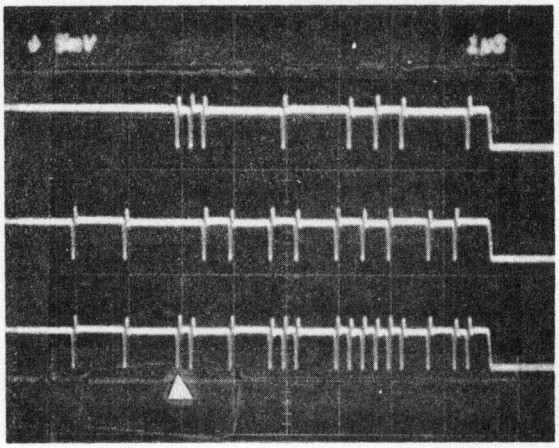


Fig. 9. Encoded Address, Parity, and Address/Parity Output Sequences from Example Illustrated in Fig. 8, Shown with Expanded Time Scale.

within the DABS data block envelope correspond to the DPSK bits. In these transmissions the sync phase reversal was not transmitted; if it had been transmitted it would not have affected the address/parity output. The triangle superimposed on the photographs indicates the 33rd bit of the info/address block or the 1st bit of address/parity for a 56-bit message sequence.

Tables I and II were generated using the encoding polynomial prescribed in Reference 2. The sequences listed in the tables were worked out by hand and verified with a working encoder. The address table is filled by multiplying the polynomial corresponding to the clear text address sequence by the encoding polynomial, and truncating after the 24th bit. The parity table is filled by dividing the information sequence polynomial by the encoding polynomial and taking the last 24 bits of the remainder as the parity sequence. These algebraic manipulations have been illustrated in the previous section, and one can use this procedure to 1) figure out the address/parity for a particular example, or 2) generate a set of address/parity tables corresponding to a different encoding polynomial.

The task of computing a new set of tables is not as formidable as it may seem. Actually the entire set of tables was generated with only one polynomial multiplication and one polynomial division. (Of course if one has an operating encoder, it can do the work for you.)

For the address table: The polynomial representing the clear text address used in the polynomial multiplication is $a(x) = 1$ (first bit of address is a "1", all of the rest are "0"). The resultant output for the encoded address sequence is simply the sequence corresponding to the coefficients of the encoding

polynomial. This multiplication yields the top row of the table. Consider next what the result would be given a "1" in only the second address bit. The basic multiplication is still the same, only the output is shifted one bit in sequence due to the one-bit shift in the address input (another aspect of the linearity of the encoder, delayed input produces delayed output). This result yields the 2nd row in the table. Similarly the table is filled up for all 24 address bits taken one at a time.

For the parity table: The polynomial representing the information sequence used in generating the parity table is

$$i(x) = 1$$

Dividing this polynomial by the encoding polynomial until the quotient shows the 89th bit of output has been computed, the last remainder prior to this division is the parity sequence corresponding to the first bit of the information sequence. Working backwards up through the division, the partial remainders provide the other sequences for the table.

It may be noted that the parity due to the last 32 bits of the 88-bit information sequence is identical to the parity for the 32-bit information sequence for a short message. A little thought should convince the reader that these parity sequences corresponding to the last 32 bits of an 88-bit information sequence are the partial remainders that would be obtained if the division had been stopped following the appearance of the 33rd bit in the quotient corresponding to a short message block.

SECTION 6

BASIC TEST PATTERNS

The test patterns presented in this section were designed to check the DABS uplink encoders used in the portable DABS Interrogator Simulators. The bit assignments agree with the proposed DABS uplink format specifications. In the following tests, the sync burst bits are not discussed since they have no bearing on the encoded output.

6.1 ADDRESS (POLYNOMIAL) TEST

The address polynomial test is useful to check, 1) that the encoder is performing the basic multiplication properly, 2) that the polynomial has been entered correctly into the encoder, and 3) that the encoder has switched from information field processing to address processing at the proper time.

Info/Address Block Input:		56-Bit Block Bit Nos.	112-Bit Block Bit Nos.
Information Field:	all 0's	1 through 32	1 through 88
Address Field:	1	33	89
	0's	34 through 56	90 through 112

so that the data block input sequence is:

00000000.....00000010000000000000000000000

The corresponding address/parity output should be:

00000000.....00000011111111111110100000010 0.

It should be noted that the address/parity output sequence is the coefficient pattern of the encoding polynomial.*

The input-encoded output relationships for this test are graphically illustrated in Fig. 10. The Figure shows the detected DPSK bits in the DABS data block envelope for this 56-bit message. The downward "spikes" are the detected reversals in the carrier phase, i.e., a "1" for that DPSK bit. This input-output relationship corresponds to the most elementary form of "multiplication" in the encoding mechanism.

6.2 PARITY TEST

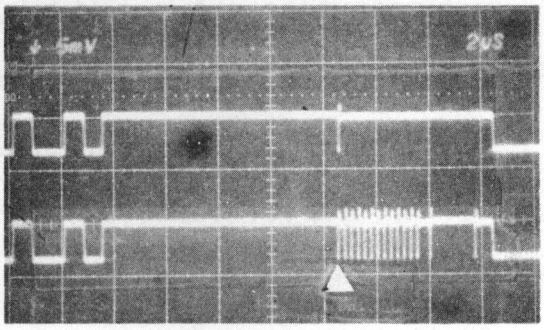
This test checks the generation of parity output from the encoder.

Info/Address Block Input:		56-Bit Block Bit Nos.	112-Bit Block Bit Nos.
Information Field:	0's	1-8	1-64
	1's	9-21	65-77
	0	22	78
	1	23	79
	0's	24-29	80-85
	1	30	86
	0's	31-32	87-88
Address Field:	all 0's	33-56	89-112

*This is the coefficient pattern of the polynomial except for the term corresponding to x^{24} - a 24th degree polynomial has 25 terms; since there is only room for 24 bits in the address/parity output, the last coefficient would come out as the "25th" address bit - except that the transmission of the data block ends after the 24th bit.

Input:

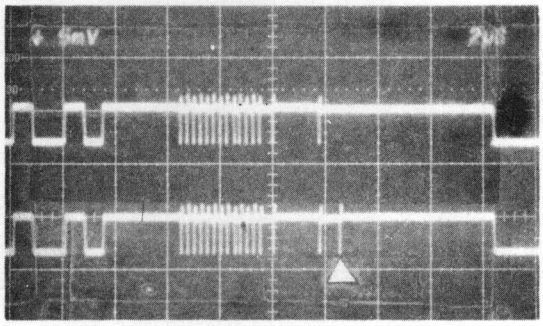
Address/Parity Output:



a) Address (polynomial) Test

Input:

Address/Parity Output:



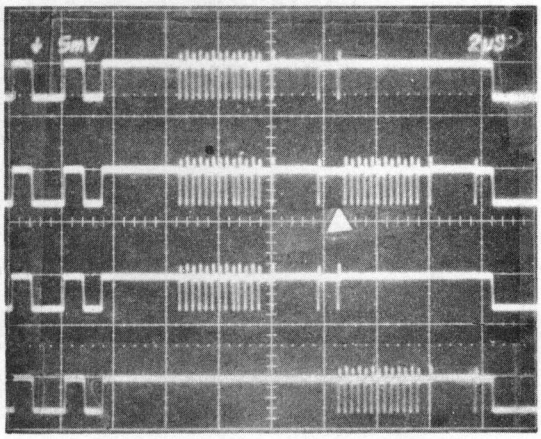
b) Parity Test

Input

Address/Parity Output:

Parity:
(from Parity Test)

Encoded Address:
(from Address Test)



c) Address-Parity Overlay Test

Fig. 10. Three Basic Test Patterns for DABS Uplink Encoder.

It may be noted that the last 24 bits of the information field are the coefficients of the first 24 terms of the encoding polynomial.

The data block input sequence is:

....0001111111111111101000000100 0000000000000000000000000

and the encoded output sequence is:

....0001111111111111101000000100 1000000000000000000000000

This input-output relationship corresponds to elementary division * in the encoding mechanism, where the parity output is the "remainder" after the division process.

The input-output relationship for this test is also illustrated in Fig. 10.

6.3 ADDRESS-PARITY OVERLAY TEST

Since the encoding process is a linear operation, the principle of superposition must hold. Thus the results of the two previous tests must superpose (overlay) directly. The overlay process is a modulo-2 addition on a bit by bit basis (i.e., $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, and $1 \oplus 1 = 0$ (no carry). Now if the testing sequences used in the first two tests are simultaneously input to the encoder, the new net input sequence is:

.....0001111111111111101000000100 1000000000000000000000000

Addition of the output sequence due to the parity test alone to the output due to the address test alone, yields the address/parity sequence:

---

*This sequence yields one non-zero bit of parity. If we had input the entire 25 terms of the polynomial, starting in bit 8 (or 64), then parity would be all zeros (no remainder). This leads to the possibly confusing conclusions: 1) it's working perfectly, or 2) it's not working at all.

(parity) ... 000111111111111101000000100 100000000000000000000000

(address) ⊕ ... 000000000000000000000000 111111111111101000000100

Address/Parity 000111111111111101000000100 011111111111101000000100

Note: Bit No. 33 = 0, since 1⊕1 = 0. The input-output relationships for this test are illustrated in Fig. 10.

If the DABS encoder passes these three tests, it is functioning properly.*

Two additional checks should be made to ensure that the parity generation starts with the 1st bit of the data block: 1) setting only bit no. 1 equal to a "1", the encoded output should be:

1000000..... 0000 001110010011010111101010

for a 112-bit data block, and

1000000..... 0000 000000011000010101100111

for a 56-bit data block (taken from Table II); and 2) the presence or absence of a "1" in any of the sync burst bits should not affect the encoder output.

With the address/parity tables in the preceding section, the user should be able to generate his own test patterns. Experience has shown: 1) working with the simplest (fewest 1's) input sequence and polynomial expression yields the best aid in debugging a faulty encoder, and 2) with the DABS encoder, it is helpful to look at the generation of parity separately from the generation of encoded address.

*This statement is implementation sensitive, e. g., the encoder may work with a polynomial having an even number of non-zero coefficients, and may not work with a polynomial having an odd number of non-zero coefficients, depending on what the designer did with the parity check inputs corresponding to the non-zero coefficients. In the latter case, an inversion of the output of the last parity checker will provide a quick remedy for the situation.

SECTION 7

GENERAL APPLICATION TO ENCODERS-DECODERS

The results presented in the previous sections are applicable to other polynomial encoders and decoders. Particularly significant is the technique of resolving the encoding process into superposable operations. With knowledge of the basic operations the particular encoder or decoder can perform, the results presented here (including the use of address-parity tables) can be applied.

The DABS encoding system employed in both the uplink and downlink is designed according to the same functional block diagram (Fig. 3). Specific operation is determined only by the control states of the steering logic. (Thus the transponder circuitry used to decode the uplink can also be used to encode the downlink.)

The techniques and TABLES presented in Sections 4 and 5 can be used accordingly in the other three cases:

Uplink Decode - The decoder is set into the feedback configuration which "divides" throughout the entire message block, first to compute the parity, second to obtain the unencoded address.

Downlink Encode* - During the information field, the encoder "divides" to generate the parity bits just as in the uplink encoder. Starting with the first bit of the address, instead of feeding the address into the shift register, the input of the shift register is held at a logical "0". This has the effect of adding the parity to the clear text address rather than to the encoded address. (Table II can be used directly, but the encoded address table (Table I) is not used at all.)

Downlink Decode - During the information field, the decoder "divides" to generate the parity bits. Starting with the 1st bit of address/parity, the input of the shift register is held at logical "0". The parity is added to address/parity yielding the clear text address. Note: In practice, the implementation of the downlink decoder is slightly different, but operationally equivalent to this description. This alternate realization is shown in Figure 11. It should be pointed out that this realization corresponds more closely to the mechanical manipulations of a hand computed division. With this realization, the division does not start until the first bit of data reaches the end of the shift register (24 clock counts), where it is multiplied by the polynomial coefficients and fed back into the shift register string as a modulo-2 subtraction from the previous dividend. Following the 56th or 112th clock, the outputs of the shift register contain the decoded address bits.

---

*The downlink uses pulse position modulation (PPM) and does not contain a sync burst field in the reply data block (see Reference 1). The encoder-decoder operation is independent of modulation type and the presence of any sync burst bits.

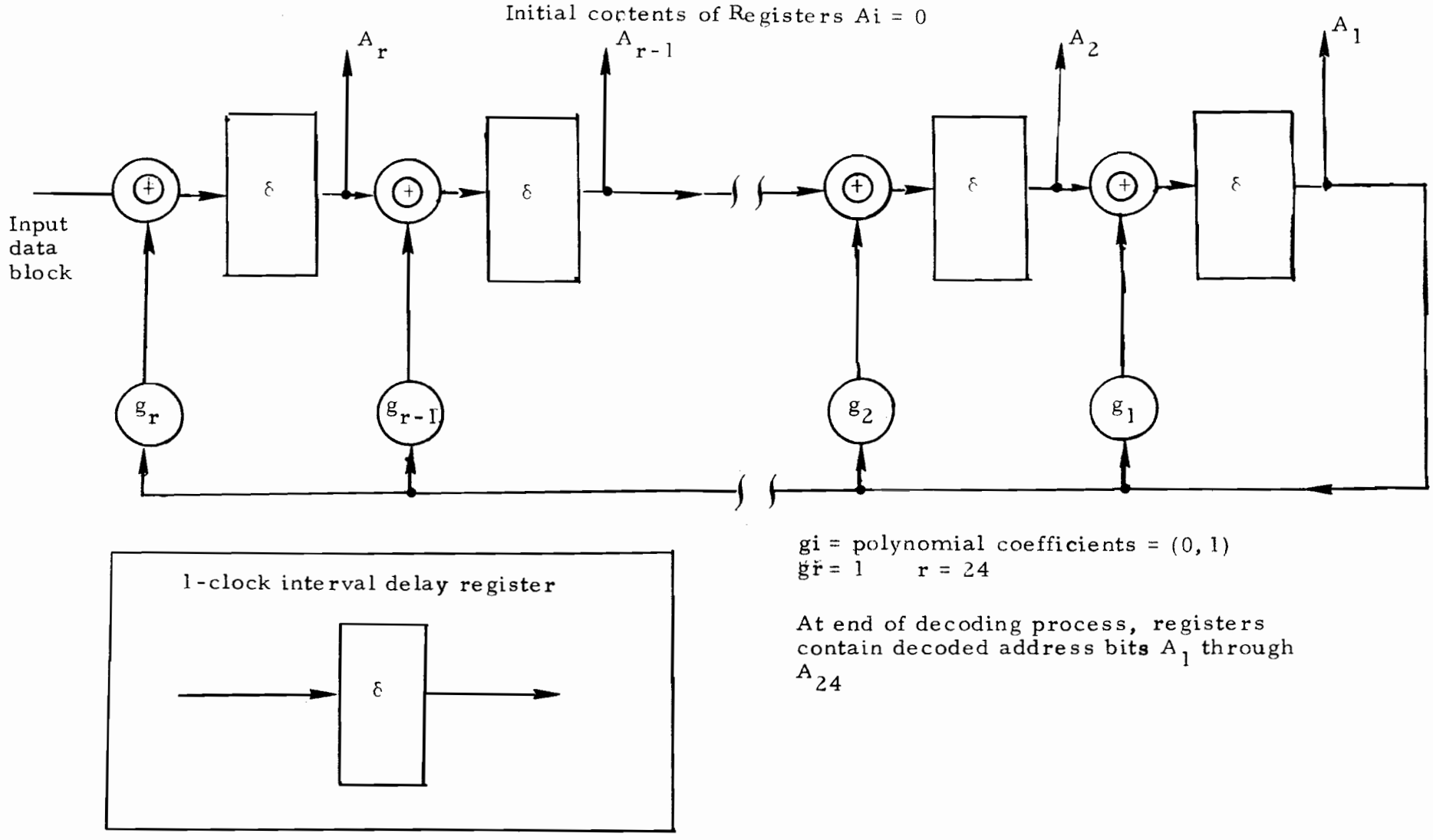


Fig. 11. DABS downlink decoder.

ACKNOWLEDGMENT

I wish to express my appreciation to J. Russell Johnson, colleague at Lincoln Laboratory, for providing initial insight into the theory of operation of the encoding mechanism. He is also responsible for suggesting basic test patterns which have proven to be valuable in checking encoders for proper operation [Ref. 3].

J. Samson

REFERENCES

[1] P. R. Drouilhet, "DABS: A System Description," Project Report ATC-42, Lincoln Laboratory, M. I. T. (18 November 1974) FAA-RD-74-189.

[2] P. R. Drouilhet, "Provisional Signal Formats for the Discrete Address Beacon System (Revision 1)," Project Report ATC-30, Revision 1, Lincoln Laboratory, M. I. T. (25 April 1974) FAA-RD-74-62.

[3] J. Russel Johnson, private communication (February 1974).