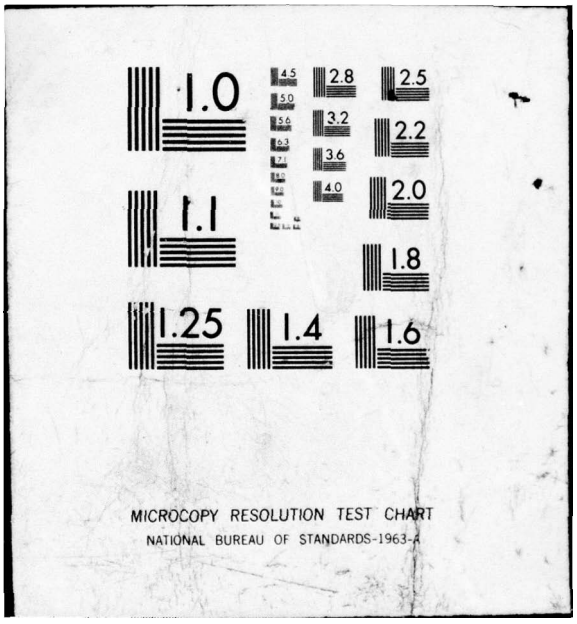
AD-A031 983 ELECTRONIC COMMUNICATIONS INC ST PETERSBURG FLA RESE--ETC F/G 17/2.1
ADAPTIVE ERROR CORRECTING TECHNIQUES FOR USE IN AIRBORNE SATELL--ETC(U)
SEP 76 R K SMITH F33615-75-C-1231

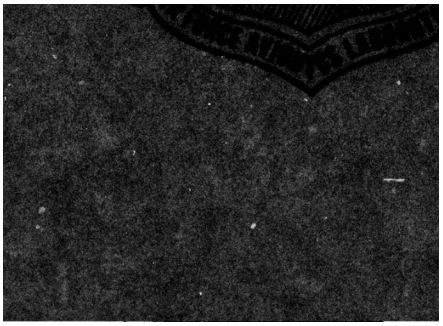
UNCLASSIFIED AFAL-TR-76-103 NL

| OF |
AD
A031983

END

DATE
FILMED
1-76


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A



NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

Paul F. Humel

PAUL F. HUMEL 1Lt, USAF
Project Engineer

FOR THE COMMANDER

James D. Everett

JAMES D. EVERETT Lt Col, USAF
Chief, System Avionics Division
Air Force Avionics Laboratory

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

AIR FORCE - 20 OCTOBER 76 - 100

63431 F
AFAL

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFAL-TR-76-103	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): Adaptive Error Correcting Techniques for use in Airborne Satellite Communications Systems.	5. TYPE OF REPORT & PERIOD COVERED: Final Repl. April 1975-December 1975.
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Richard K. Smith	8. CONTRACT OR GRANT NUMBER(s): F33615-75-C-1231 NEW
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Electronic Communications, Inc. P.O. Box 12248 St. Petersburg, Florida 33733	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 1227[illegible] 32
11. CONTROLLING OFFICE NAME AND ADDRESS: Air Force Avionics Laboratory Wright-Patterson AFB, Ohio 45433	12. REPORT DATE: Sep[illegible]76
	13. NUMBER OF PAGES: 70 (69 p.)
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This report has been reviewed by the Information Office (OI) and is releaseable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Approved for public release, distribution unlimited.

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Error Control
Coding Theory
Satellite Communications
Ionospheric Scintillation
Gallager Algorithm

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Airborne satellite communication systems have been shown to frequently suffer severe degradation of performance due to ionospheric scintillation. This type of interference produces deep fades in received signal power resulting in long error bursts in the demodulated data stream. (Continued)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

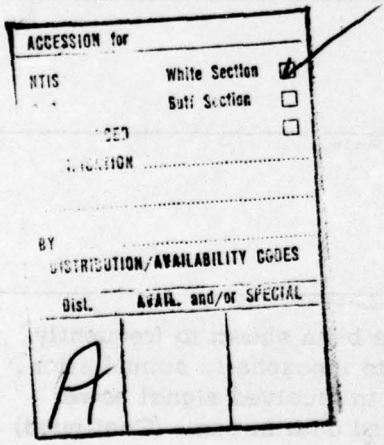
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

407 065
bps

The objective of this study was to lay the groundwork for a quantitative evaluation of the performance improvement that can be achieved through the use of an adaptive error control technique. Specifically, the adaptive Gallager algorithm and the extended Gallager algorithm were examined to determine which of these provides the better solution to the error control problem on this channel.

The first phase of this study dealt with the heuristic analysis of these two error control techniques on the basis of binary error sequences generated from flight test records of received signal level. Phase I culminates with a recommendation as to which technique should be used. Suggestions regarding the values of important algorithm parameters are also provided.

Phase II of this study was devoted to the development of a computer simulation algorithm and the design of a hardware evaluator for the recommended coding technique. The algorithm, used with the AFAL channel simulator, permits the quantitative evaluation of code performance on the simulated scintillation channel. The hardware evaluator, which is an adaptive Gallager encoder/decoder with switch selectable parameters, is designed to be interface compatible with the existing AFAL flight test modem.



PREFACE

The work covered by this report was accomplished under Air Force Contract F33615-75-C-1231. The effort is documented under Project Work Unit 12273211, and has been administered under the direction of Mr. John Garrett (AFAL/AA) of the Air Force Avionics Laboratory, Wright-Patterson Air Force Base, Ohio.

This report covers work performed from April 1975 to December 1975, and was submitted by the author in February, 1976.

This program was conducted by Electronic Communications, Inc., St. Petersburg, Florida, under the direction of Mr. Richard A. Saraydar, Program Manager, and Mr. Richard Kent Smith, Project Engineer. Significant contributions were made to the program by Mr. Myung Kim.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION I

INTRODUCTION

Airborne satellite communication systems have been shown to frequently suffer severe degradation of performance due to ionospheric scintillation. This type of interference produces deep fades in received signal power resulting in long error bursts in the demodulated data stream.

The objective of this study was to lay the groundwork for a quantitative evaluation of the performance improvement that can be achieved through the use of an adaptive error control technique. Specifically, the adaptive Gallager algorithm and the extended Gallager algorithm were examined to determine which of these provides the better solution to the error control problem on this channel.

The first phase of this study dealt with the heuristic analysis of these two error control techniques on the basis of binary error sequences generated from flight test records of received signal level. Phase I culminates with a recommendation as to which technique should be used. Suggestions regarding the values of important algorithm parameters are also provided.

Phase II of this study was devoted to the development of a computer simulation algorithm and the design of a hardware evaluator for the recommended coding technique. The algorithm, used with the AFAL channel simulator, permits the quantitative evaluation of code performance on the simulated scintillation channel. The hardware evaluator, which is an adaptive Gallager encoder/decoder with switch selectable parameters, is designed to be interface compatible with the existing AFAL flight test modem.

SECTION II

PHASE I - ANALYSIS OF CODES

1. INTRODUCTION

The objective of Phase I of this study was to determine heuristically, based on error records supplied by AFAL, whether the adaptive Gallager or the extended Gallager error correcting algorithm offers the best solution to the error control problem for this ionospheric scintillation channel. In doing this, a search was conducted for new convolutional codes with the properties necessary for use with the extended algorithm. This search was unsuccessful in that no new high rate convolutional codes with the properties required for use with the extended algorithm were discovered. (The extended algorithm requires two convolutional codes; one containing the other.)

The search for new convolutional codes was conducted using the trial-and-error technique with the aid of a threshold decoding simulator that was developed for this purpose. Although no new "contained" codes were discovered, two trial polynomials were confirmed as generators for 1/2 rate convolutional codes with good distance properties.

2. ADAPTIVE GALLAGER ALGORITHM

A diagram of a typical 1/2-rate Gallager encoder is shown in Figure 1. It consists of a shift register B+X+k bits long with several taps at the left end and one tap on the right-most stage. The configuration of taps at the left end of the register is defined by the code generator polynomial which is chosen based on good random error correction and detection properties.

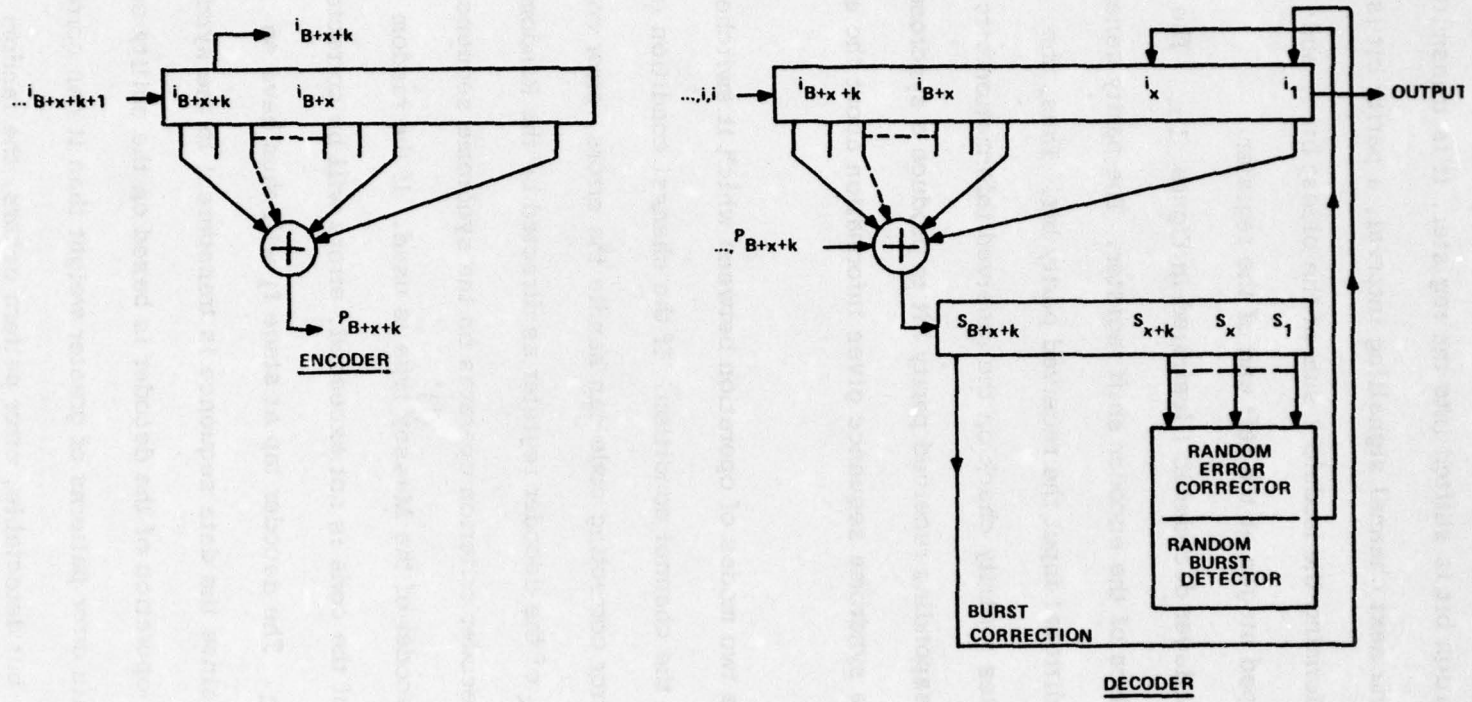


Figure 1. Adaptive Gallager Encoder/Decoder

As each information bit is shifted into the register, it is transmitted over the channel. In the next channel signalling interval, a parity bit is generated and sent by forming the modulo 2 sum of the oldest bit i, and the new bits in the tapped stages at the left end of the register.

The adaptive Gallager decoder is illustrated in Figure 1b. The decoder contains a replica of the encoder shift register. The parity generator, however, has as an additional input the received parity bit. Thus, the modulo 2 adder generates a parity check on the received information bits and compares it to the corresponding received parity bit to produce a syndrome bit. Examination of the syndrome sequence gives information about the error pattern.

The decoder has two modes of operation between which it switches automatically based on the channel condition. If the channel condition is such that the random error correcting code can handle the errors, error correction is performed at stage i_x of the decoder register as directed by the Random Error Corrector logic. The decoder criterion operates on the syndrome sequence; typically a threshold decoder of the Massey[1] type is used. If the random error correcting power of the code is not exceeded, errors will be corrected before reaching stage i_1. The decoder tap at stage i_1 will thus have no effect on the syndrome since the data sequence is transparent to the syndrome.

The principle of operation of the decoder is based on the ability of the code to detect certain error patterns of greater weight than it can correct. When an uncorrectable, but detectable, error pattern occurs, the random error correction control algorithm is suspended and burst error correcting

takes over. The burst correcting algorithm is very simple. The rule is to decide that i_1 is wrong and change it if and only if the most recent syndrome bit $S_{B+X+k} = 1$. If the error burst is no more than 2B bits long, it will have passed the taps at the left end of the decoder register before the decoder enters the burst mode. If a "clean" guard space follows the error burst, the current syndrome will be non-zero (flagging a correction) if and only if i_1 itself is in error. If a guard space of error-free bits approximately equal to the burst length is present at the input to the decoder, the entire error burst will be decoded by this rule. Although not shown in the diagram, when a correction is made the effect of the error on the syndrome is removed by complimenting S_{B+X+k} (making it a binary zero). Because of this, as the error burst passes out of the decoder, the syndrome register fills up with zeros. The control algorithm examines a region around the right end of the syndrome register for an all zero condition. When enough consecutive zero syndromes are observed, control is passed back to the random error correcting algorithm.

The advantage of the adaptive Gallager algorithm over many of the other burst correcting codes is that it requires a very short error-free guard space approximately equal in length to the actual burst that is being corrected. Other techniques typically require a guard space about three times the length of the maximum correctable burst. The reason for this is that the Gallager technique corrects most but not 100%, of the bursts less than its maximum designed length. The requirement for a guard-space-to-burst ratio of 3 applies to the idealized case of guaranteed error-free burst correction.

Note that the occurrence of an error in the guard space will cause a short burst of errors in the output (one each time it passes through a tapped stage of the register) if the error is in an information bit and a single output error if the error is in a parity bit. This problem can be more or less serious depending on the error statistics on the channel of interest. For the scintillation channel, the random error rate in the interburst intervals is rather high. For this reason, the effect of guard space errors is an important consideration.

One approach to dealing with the problem of guard space errors is to modify the rules that govern the burst/random mode switching. Indications as to the modifications that may be required can be obtained by running computer simulations using the standard algorithm and an error sequence generator whose statistics accurately model the channel of interest. The simulation should log the occurrence of decoded bit errors and keep track of the state of the decoder (burst or random) when each of these failures occurs. Analysis of the results of such a simulation will give clues as to the failure modes of the algorithm. For example, if a preponderance of the errors occur in the burst mode, the criteria for entering the burst mode may need to be strengthened so that the decoder doesn't switch to the burst mode so readily. On the other hand, weakening the requirements for switching back to the random mode may be the answer. The encoder and decoder simulation algorithms provided in Section III of this report can be used in conjunction with the AFAL channel simulation program to conduct an analysis such as this. The hardware evaluator design presented in Section III provides for a limited amount of experimentation in this regard through the provision of a switch that allows the random-to-burst criterion to be strengthened. In addition, the design permits the use of any desired algorithm

(stored in programmable read-only memory) to control the basic random-to-burst switching decision. Computer simulation can be used to determine the optimum ROM program for the scintillation channel. Finally, the ease with which the decoder returns to the random mode can be affected by varying the 'Y' - parameter.

3. EXTENDED GALLAGER ALGORITHM

While the performance of the adaptive Gallager decoder can be optimized for specific channels by employing the proper mode-change strategies, there is a definite limit to the performance that can be achieved in this manner. Another approach to the problem of errors in the guard space has been suggested by Sullivan.[2] Sullivan's generalized (or extended) Gallager decoder depends, in its principle of operation, upon the use of two convolutional codes, C and C*, where C* contains C. At the encoder the information sequence is first encoded using code C; after a fixed delay, it is also encoded with a "shortened" version of C*, which is added to the parity bits of C. A functional diagram of the extended Gallager decoder is shown in Figure 2. In the random error correcting mode, the decoder is equivalent in operation to the ordinary Gallager decoder using the C- code. In the burst mode, the C* decoder is switched in to remove random errors from the interval following the error burst.

In his paper, Sullivan presented an example of the extended Gallager algorithm using convolutional codes that yield an equivalent coding rate of 1/3. He also points out the difficulty in the general application of his scheme by warning that there is "a fundamental problem . . . resulting

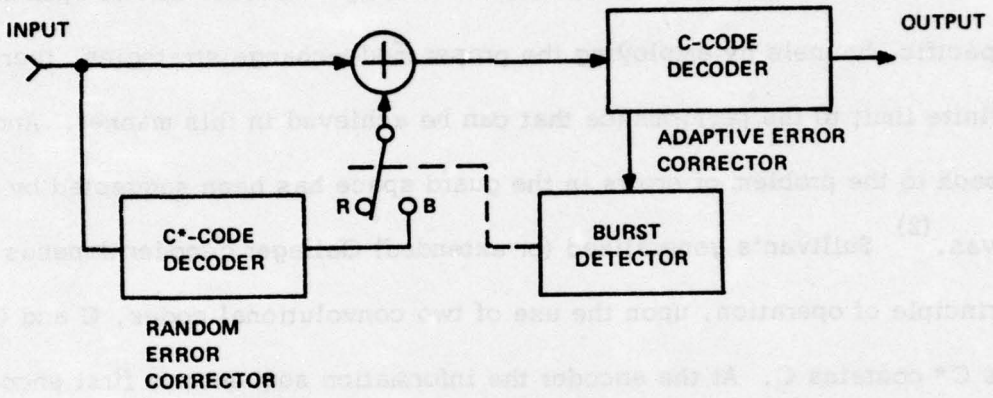


Figure 2. Extended Gallager Decoder

from the scarcity of good constructive random-error-correcting convolutional codes, particularly at high rates. This problem becomes even more pronounced with the constraint that one of the two codes used must contain the other." He goes on to say, "It therefore appears likely that full utilization of this scheme must await further developments in constructive techniques for the encoding and decoding of random-error-correcting convolutional codes."

Subsequent to the appearance of Sullivan's paper, several papers which bear on the problem have appeared. In May, 1972, a correspondence entitled, "Contained Convolutional Codes," appeared in the Transactions on Information Theory (Ferguson, 3). Here, Ferguson shows that the conditions of containment imply a very simple factorization of the codes. More recently, Wu(4), (5) in a two-part paper, has presented a code-generation algorithm for high rate convolutional codes. Since these codes are threshold decodable, Wu's algorithm has potential application to the problem of discovering good codes for use with the Gallager algorithm. Unfortunately, the appearance of Wu's paper was subsequent to the completion of Phase I of this study.

4. SEARCH FOR NEW CODES

During the conduct of Phase I of this study, M. Kim of ECI conducted a search for new convolutional codes with the properties required for application to the extended Gallager algorithm. This search was carried out using the trial and error method[1] together with a threshold decoding simulation that was developed to test the properties of codes generated by the various trial generator polynomials. A listing of this computer simulation is presented in Appendix A.

The impetus for this attempt to find a pair of more powerful convolutional codes for use with the extended Gallager algorithm came from the fact that the known codes of rate 1/3 are not powerful enough to correct 3 consecutive errors in the random mode or to correct two consecutive errors in the guard space in the burst mode.

Although the search for contained codes was unsuccessful, Kim did succeed in discovering two new polynomials that generate codes with good distance properties. These are shown in Table 1 where the properties of some of the known codes as well as the new codes are summarized. The new code of constraint length 24 was selected for the proposed hardware evaluator design.

5. CHANNEL ERROR CHARACTERISTICS

A communication channel can, in general, be categorized into one of three basic classes: random error channels, burst error channels and compound channels (combination random and burst). The UHF channel affected by ionospheric scintillation fading falls into the compound channel category.

Eight files of measured received signal level taken from actual flight tests together with binary error sequences generated by a computer model of the modem were furnished to ECI to provide a basis for an heuristic analysis of code performance.

TABLE 1. CANDIDATE CONVOLUTIONAL CODES

GENERATING POLYNOMIAL	EFFECTIVE CONSTRAINT LENGTH (N_E)	GUARANTEED CORRECTABLE ERRORS	DETECTABLE ERRORS	REMARKS
FOR ORIGINAL GALLAGER DECODING (1/2-RATE)				
(111101)	16	2/20	3/20	
(111001)	11	2/12	3/12	
(100000110111)	22	3/24	3/24	
(101101110111)	24	3/24	4/24	New Code
(10110111)	16	2/18	3/18	New Code
FOR EXTENDED GALLAGER DECODING (1/3-RATE)				
$G_1^{(2)} = (1101)$ $G_1^{(3)} = (1100)$	--	2/12	3/12	C-Code
$G_2^{*(3)} = (1100)$ $G_1^{*(3)} = (0111)$	--	1/12	2/12	C*-Code

The following is a brief synopsis of the AFAL program that was used to analyze the channel error statistics.

The flight data file number and position on disk are read in from the keyboard. To approximate a 75 BPS transmission rate, a variable is set so as to use only every 14th data point from flight file. This is arrived at by knowing that each file is approximately 5 minutes long. There being 301 blocks of data in a file makes a single block correspond to one second. Each block contains 1066 data points which roughly corresponds to 14 X 75 (1050). The data points are not averaged (14 points at a time) because the digitized power level was very smooth. A printout of a digitized block is included in the material. Seven burst threshold values were investigated which are .15, .12, .10, .08, .05, .03, .01. Each burst threshold value was evaluated at 4 different .001 threshold values that correspond to -136 dbm, -134 dbm, -132 dbm, -130 dbm. The minimum burst length is set at 1/4 second. Burst and guard counts cleared and flags initialized.

The program now evaluates burst length versus guard length for 300 blocks of a flight data file. The following criteria are used for burst and guard analysis.

The detection of a burst is anytime the bit error probability exceeds or equals the burst threshold level. This generally signifies the end of a guard space. All burst counts below 1/4 second are padded to at least a 1/4 second burst by part of the guard space to the burst. All burst counts equaling 4 seconds duration are counted and the burst count is put back to one. This is done because a guard count equaling 1/4 of the burst

length may not arise so that useful information would be lost. A burst count is complete whenever the guard count is at least equal to 1/4 of the burst length. The burst lengths for each file are tabulated in 1/4 second increments .25 - .50, .50 - .75, etc. up to 3 seconds with burst between 3 and 4 seconds in one group and those over 4 seconds counted separately. A guard count is started whenever the bit error probability falls below the burst threshold level and the previous burst length being at least 1/4 second. A guard count terminates whenever a burst condition occurs (anytime bit error probability exceeds or equals burst threshold level). Whenever a guard count is less than 1/4 of the burst count, the guard count is added to the burst count. The guard lengths for each file are tabulated corresponding to 1/4 of the previous burst length .25 - .50 BL, .50 - .75 BL, etc. up to 3 times the burst length. All guard lengths greater than 3 times length are tabulated together. When all 300 blocks of data (corresponding to 22,500 transmitted bits) have been processed, the tabulated data is printed out on the lineprinter and written out on the disc.

Among the 8 files of received signal level that were provided by AFAL, File No. 4 appears to represent the worst case. Tabulated burst statistics for both the aggregate of all 8 files and File No. 4 are shown in Table 2. The conclusions presented in the following section are based primarily on analysis of the binary error sequences generated from File No. 4 data.

5. CONCLUSIONS

Analysis of binary error sequences generated from File No. 4 data reveals that the longest error burst is 639 bits in length. Most of the long error bursts are separated by guard spaces less than 2B bits in length. The

TABLE 2. BURST ERROR STATISTICS

-134 DRM THRESHOLD LEVEL FOR .001 ERROR RATE (COMBINED DATA OF 8 FILES)

BL	BL CNT	.25*BL	.5*BL	.75*BL	1.00*BL	1.25*BL	1.50*BL	1.75*BL	2.00*BL	2.25*BL	2.50*BL	2.75*BL	3.00*BL
0.25	237	6	8	6	3	2	1	5	8	7	5	3	183
0.50	51	1	3	0	2	0	3	0	1	2	0	0	39
0.75	29	6	0	1	1	2	0	0	0	1	2	1	15
1.00	14	0	1	2	1	0	0	1	0	0	1	0	8
1.25	7	1	0	0	0	0	0	0	1	1	0	1	3
1.50	3	1	0	0	0	0	1	0	0	1	0	0	0
1.75	5	1	0	0	2	0	1	0	0	0	0	0	1
2.00	3	0	0	0	0	0	0	0	0	0	0	0	3
2.25	1	0	0	0	0	0	0	0	0	0	0	0	1
2.50	1	0	0	0	0	0	0	0	1	0	0	0	0
2.75	0	0	0	0	0	0	0	0	0	0	0	0	0
3.00	4	0	1	0	1	1	0	0	0	0	0	0	1

BURST LENGTHS OVER 4 SEC = 0

-134 DRM THRESHOLD LEVEL FOR .001 ERROR RATE (FILE 4)

BL	BL CNT	.25*BL	.5*BL	.75*BL	1.00*BL	1.25*BL	1.50*BL	1.75*BL	2.00*BL	2.25*BL	2.50*BL	2.75*BL	3.00*BL
0.25	20	1	0	1	1	2	1	1	1	1	2	0	9
0.50	8	0	1	0	0	0	0	0	0	0	0	0	7
0.75	9	5	0	0	0	1	0	0	0	0	0	0	3
1.00	5	0	0	0	1	0	0	0	0	0	0	0	4
1.25	3	1	0	0	0	0	0	0	1	0	0	1	0
1.50	0	0	0	0	0	0	0	0	0	0	0	0	0
1.75	3	0	0	0	2	0	1	0	0	0	0	0	0
2.00	2	0	0	0	0	0	0	0	0	0	0	0	2
2.25	0	0	0	0	0	0	0	0	0	0	0	0	0
2.50	1	0	0	0	0	0	0	0	1	0	0	0	0
2.75	0	0	0	0	0	0	0	0	0	0	0	0	0
3.00	3	0	1	0	1	1	0	0	0	0	0	0	0

BURST LENGTHS OVER 4 SEC = 3

total number of error burst occurrences in the 21,500 bit record was 27. The error burst duty cycle was approximately 33% with an average burst length of 263 bits. While the average guard space-to-burst ratio was approximately 3, for long bursts, the G/B ratio was close to unity. The background error rate was moderately low. However, there was a relatively high incidence of short error clusters consisting of 2 or 3 consecutive errors.

Because of the high incidence of consecutive errors in the guard spaces, it is concluded that the performance improvement that would result from the use of the extended algori thm would not be significantly better than the benefit that can be derived from the original Gallager algorithm. This is because the known extended Gallager algorithm codes of rate 1/3 are not powerful enough to correct 3 consecutive errors in the random mode or 2 consecutive errors in the guard space in the burst mode. Furthermore, the extended algorithm requires 1.5 times more transmission time and nearly twice the hardware complexity of the original Gallager algorithm. Based on this heuristic reasoning, it was concluded that Phase II of this program should concentrate on the development of software and hardware tools for the quantitative evaluation of the performance of the ordinary adaptive Gallager algorithm on the UHF ionospheric scintillation channel. The new convolutional code of constraint length 24 specified by the generator polynomial given in Table 1 should provide a significant improvement in decoded error rate for this channel. Since this code is capable of correcting 3 consecutive errors in the random mode, this power coupled with a burst/random mode change strategy that is optimized for the scintillation channel should approach the performance that could be achieved with the more costly 1/3 rate extended Gallager algorithm.

Based on the channel error statistics, it appears that the encoder/decoder buffers should be approximately 340 and 20 bits long for the first buffer, B and the second buffer, X, respectively. These parameters are, of course, adjustable in both the simulation algorithm and the hardware evaluator.

SECTION III

PHASE II - DESIGN OF EVALUATION TOOLS

1. INTRODUCTION

As a result of Phase I of this program, it was determined that the expected performance improvement provided by the extended Gallager algorithm on the scintillation channel does not warrant the substantial added complexity. Because of this, Phase II of this program has been devoted to the development of tools for evaluating the performance of the original Gallager algorithm on a real scintillation channel. Specifically, software algorithms tailored to the PDP-11 have been developed for use with the AFAL channel simulation program for the adaptive Gallager encoder and decoder. In addition, a detailed hardware design has been completed for a variable parameter adaptive Gallager encoder-decoder that will enable real-channel evaluation of the effectiveness of this error correcting scheme.

Description of these software and hardware performance evaluation tools is the subject of this section.

2. GALLAGER ALGORITHM USING $g(D) = 1+D^2+D^3+D^5+D^6+D^7+D^9+D^{10}+D^{11}$

Block diagrams of the coder and decoder for the code selected are shown in Figures 3 and 4 respectively. The encoder is extremely simple, consisting of a single long buffer register, a parity tree and a commutator that alternately sends data and parity bits. As designated by the generator polynomial, parity bits are computed from certain of the past twelve information bits together with the information bit in the last stage of the buffer. Since this is a 1/2 rate code, two channel symbols (one information and one

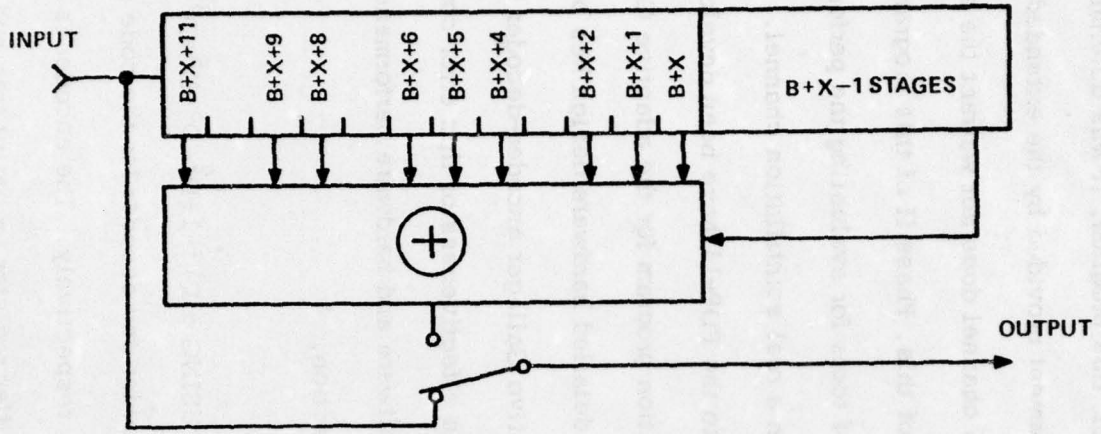


Figure 3. Block Diagram - Adaptive Gallager Encoder (Kim's Polynomial)

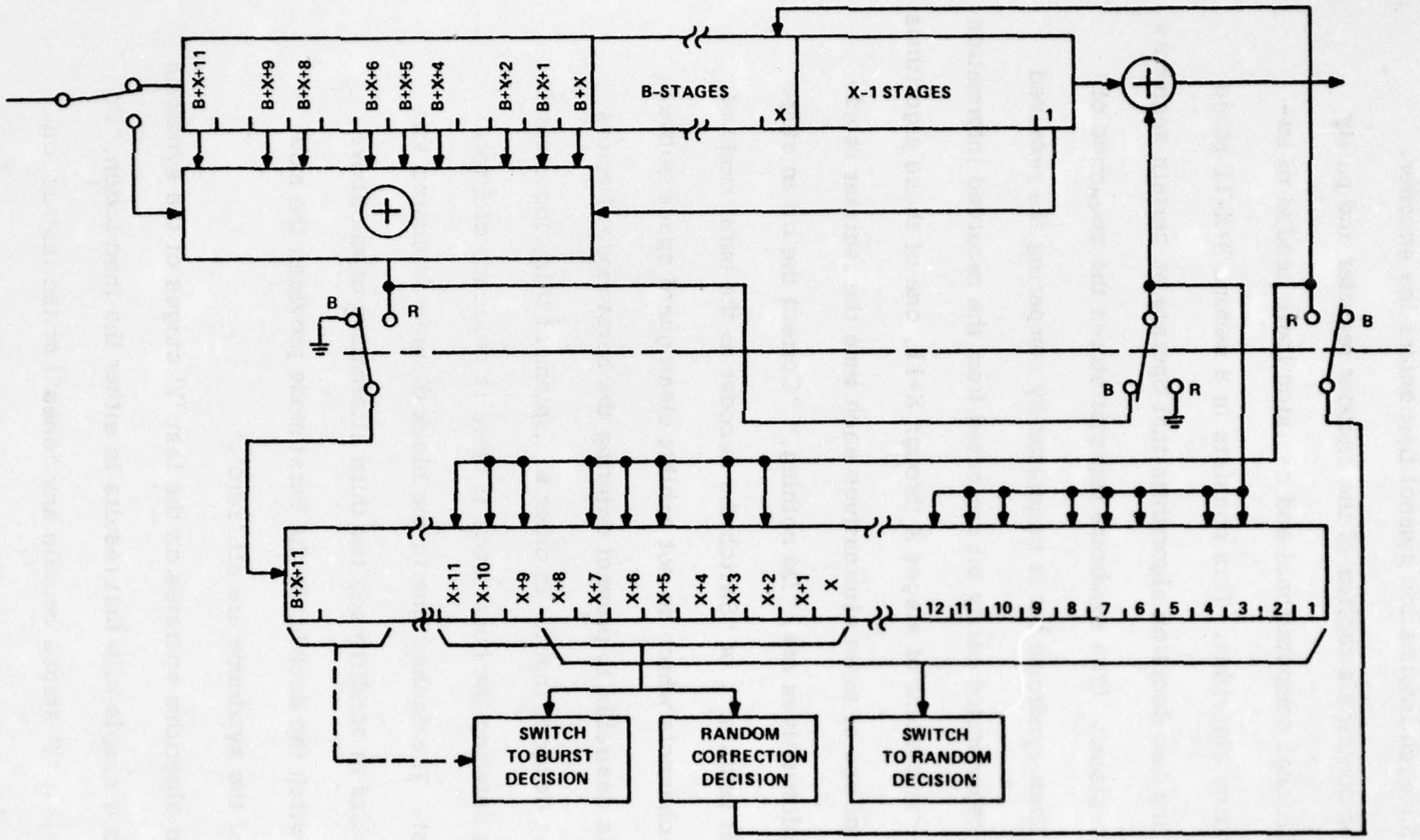


Figure 4. Block Diagram - Adaptive Gallager Decoder (Kim's Polynomial)

parity) are sent for each information symbol that enters the encoder.

The decoder contains a replica of the encoder register and parity tree plus the additional computational and decision logic needed to implement the decoding algorithm. This consists in a second B+X+11 stage syndrome buffer and two decision algorithms that operate on certain segments of the syndrome register. (The syndrome register stores the sequence of syndrome bits. Each syndrome bit is generated by comparing the received parity bit with a regenerated parity bit computed from the received information bits). Based on the content of stages X through X+11, one of these algorithms decides in favor of one of three alternatives each time the register is advanced. These alternatives are, "*Do nothing*," "*Correct the bit in stage 'X' of the decoder register*", or "Switch the decoder to the burst mode of operation." On channels which do not exhibit clean guard space between error bursts, it is desirable to prevent entering the burst mode if errors are present at the decoder input. In order to implement this, logic can be added to prevent entering the burst mode if there is evidence of errors in the recent past. The dashed line in the block diagram indicates this option which results in conditioning the third alternative of the above algorithm to, "Switch the decoder to the burst mode provided the most recent 'N' bits of the syndrome are all 'zero'."

The second algorithm operates on the last 'Y' stages of the syndrome register. This is a simple rule that results in either the instruction, "Do nothing" (if the last 'Y' stages contain any "ones") or the instruction, "Switch the decoder to the random-error mode" (if "all-zeros" are present in the last 'Y' stages).

3. ALGORITHMS FOR PDP-11 SIMULATION

Because of the fact that the AFAL channel simulation was implemented using encoder and decoder subroutines written in PDP-11 assembly language, the algorithms provided here are structured to simplify their implementation using PDP-11 registers and instructions. The schematic drawing shown in Figure 5. depicts the representation of the encoder and decoder buffers in the PDP-11 memory.

Although the encoder and decoder algorithms are specified here in the form of considerably detailed flow charts, the tasks of coding and interfacing these subroutines with the AFAL simulation program remain. (The coding for the major portion of the encoder subroutine is included here as an example of one way the long buffer registers can be implemented and manipulated within the framework of the PDP-11.)

Flow charts of the encoding and decoding algorithms are shown in Figures 6 and 7 respectively. These algorithms are general specifications of the encoding and decoding rules. In the interest of simplifying the implementation on the PDP-11, the programmer may find it convenient to impose some restrictions on the lengths of the 'B' and 'X' registers. This has been done to some extent in the example encoder program given in Table 3. Here the encoder register is represented in PDP-11 memory by a block of contiguous words. This approach (rather than using one memory word per bit) restricts the buffer length to a multiple of 16. This is of no great consequence performance-wise, however, on the scintillation channel where bursts of errors are typically much longer than 16-bits.

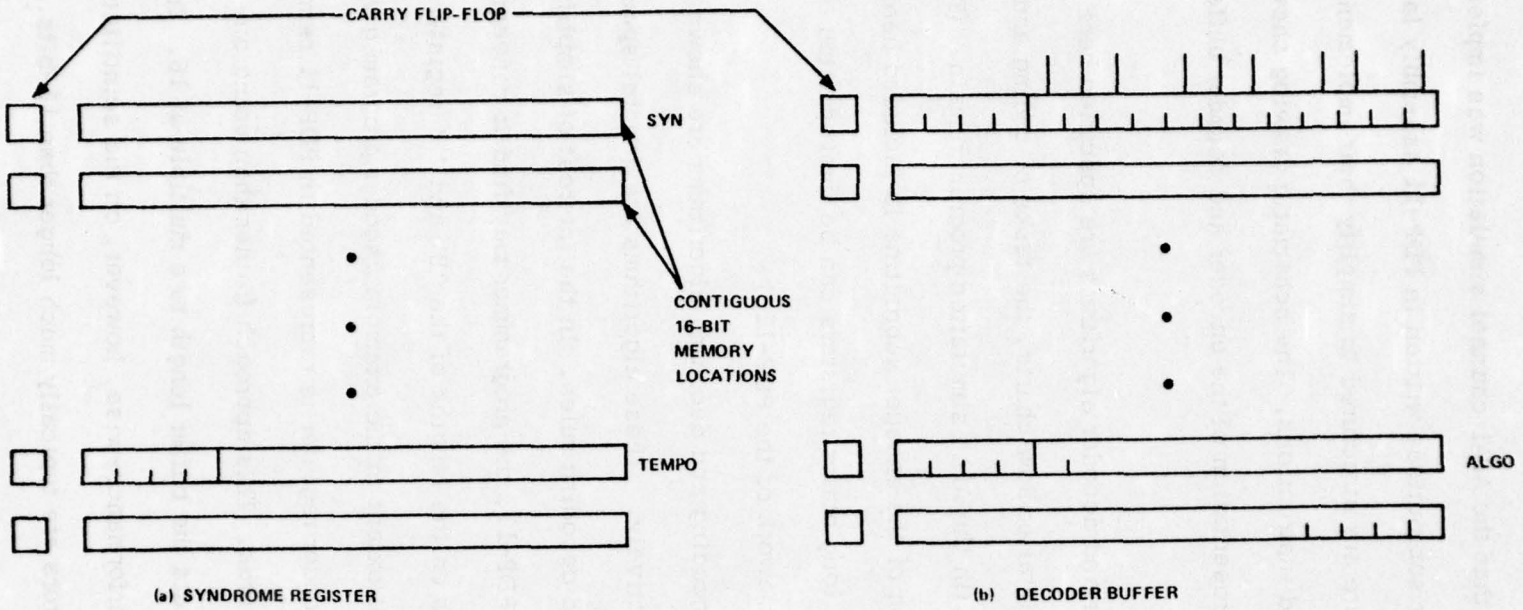


Figure 5. Representation of Decoder Registers

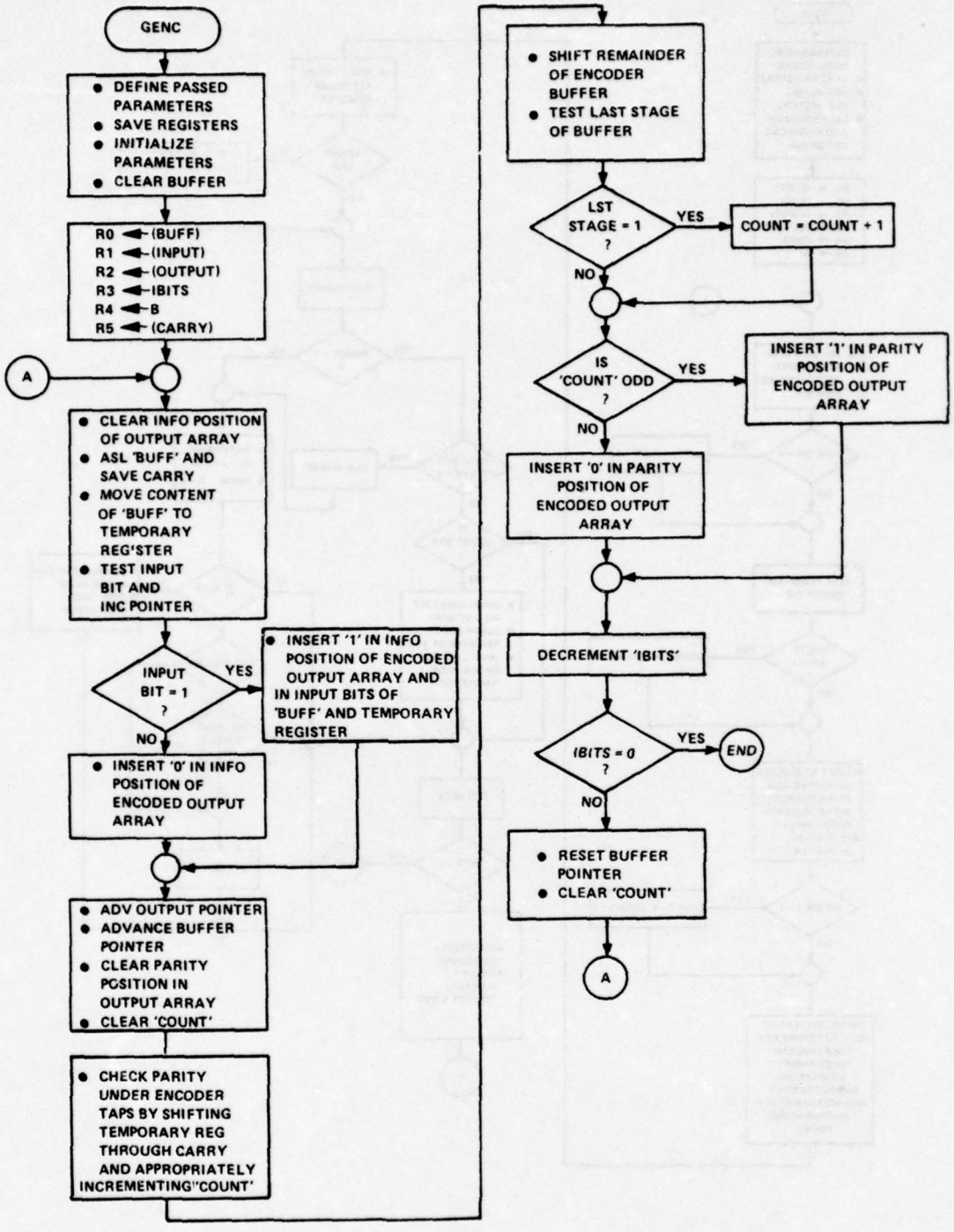


Figure 6. Flow Chart - Adaptive Gallager Encoder Simulation

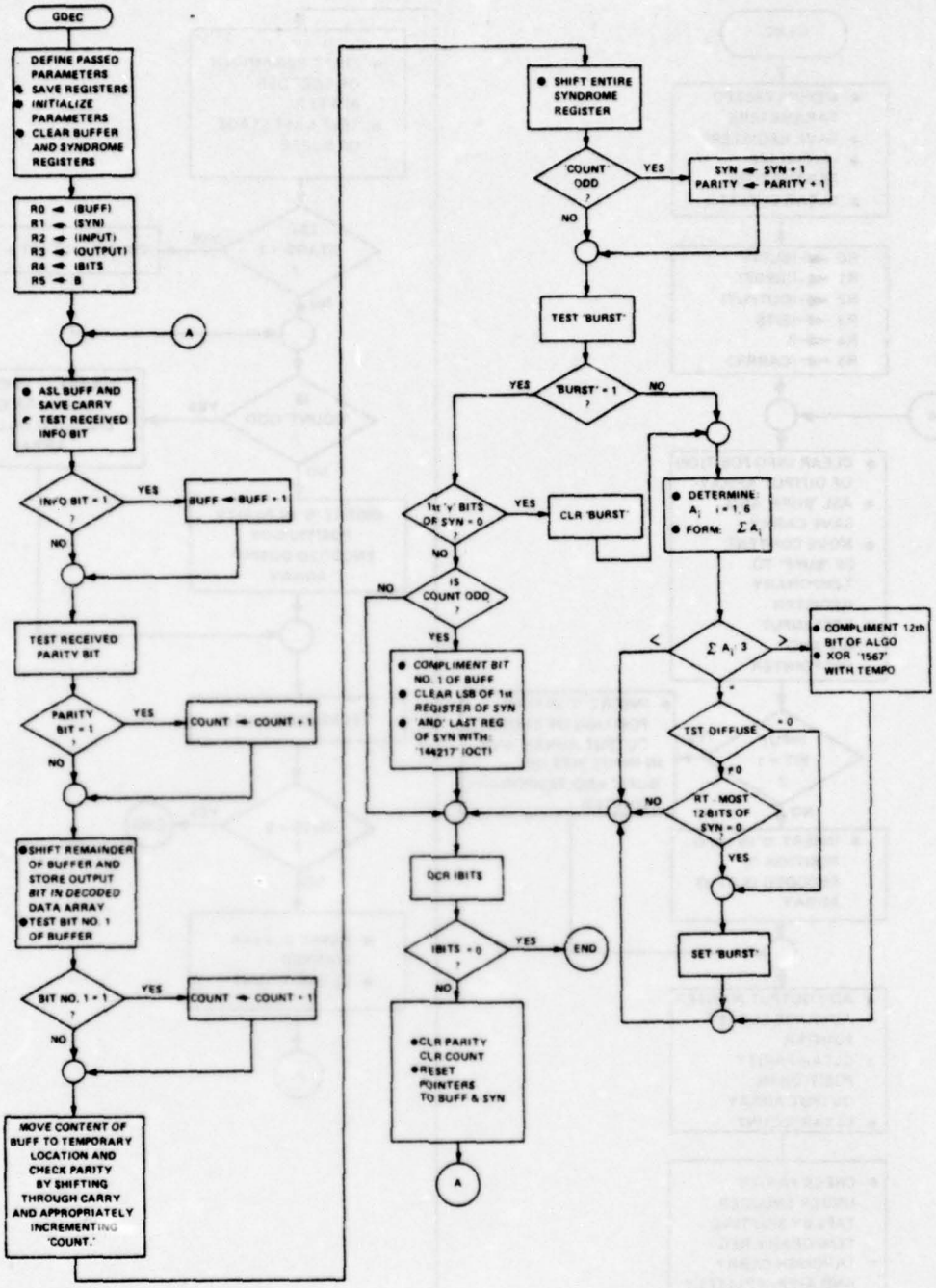


Figure 7. Flow Chart - Adaptive Gallager Decoder Simulation

TABLE 3. EXAMPLE ENCODER PROGRAM

```
;ADAPTIVE GALLAGER ENCODER - CODING EXAMPLE (REF: FIGURE 3-3)
;CODING MUST BE ADDED TO THIS ROUTINE TO DEFINE PASSED PARAMETERS.
;DEFINE STORAGE,SAVE REGISTERS AND CLEAR THE ENCODER BUFFER.

        MOV #SHREG,R0           ;R0 - POINTER TO SHREG
        MOV #INPUT,R1           ;R1 - POINTER TO INPUT
        MOV #OUTPUT,R2          ;R2 - POINTER TO OUTPUT
        MOV ISITS,R3            ;R3 - NUMBER OF INPUT MESSAGE BITS
        MOV E,R4                ;R4 - NUMBER OF BUFFER REGISTER STAGES
        MOV #CARRY,R5           ;R5 - POINTER TO CARRY WORKING REGISTER
LOOP:   CLR CARRY               ;
        MOV #SHREG,R0           ;
        CLR (R2)                ;
        ASL (R0)
        BCC TAP
        INC CARRY
TAP:    MOV (R0),HOLD           ;MOVE CONTENT OF SHREG TO TEMPORARY REGISTER
        TST (R1)+               ;SEE IF INPUT BIT IS A ≠1≠
        BEQ HOP                 ;IF NOT, LEAVE BIT 1 OF SHREG = 0 = OUTPUT BIT
        INC (R2)+               ;SET OUTPUT BIT = 1 AND ADVANCE POINTER
        INC (R0)                ;SET SHREG INPUT BIT = 1
        INC HOLD                ;SET HOLD INPUT BIT = 1
        BR  HOP1
HOP:    TST (R2)+               ;COME HERE IF INPUT BIT = 0, LEAVE OUTPUT
HOP1:   TST (R0)+               ;BIT, SHREG & HOLD INPUT BITS = 0, AND
        CLR (R2)                ;ADVANCE INPUT AND BUFFER POINTERS.
        CLR COUNT               ;INITIALIZE PARITY COUNTER.
        ASR HOLD                ;THE FOLLOWING CODING COMPUTES THE PARITY
        BCC A1                  ;OVER THE TAPS OF THE BUFFER INPUT REGISTER
        INC COUNT               ;DEFINED BY THE GENERATOR POLYNOMIAL.
A1:     ASR HOLD
        ASR HOLD
        BCC A2
        INC COUNT
A2:     ASR HOLD
        BCC A3
        INC COUNT
A3:     ASR HOLD
        ASR HOLD
        BCC A4
        INC COUNT
A4:     ASR HOLD
        BCC A5
        INC COUNT
A5:     ASR HOLD
        BCC A6
        INC COUNT
A6:     ASR HOLD
        ASR HOLD
        BCC A7
        INC COUNT
```

TABLE 3. EXAMPLE ENCODER PROGRAM (CONTINUED)

```
A7:     ASR HOLD
        BCC A8
        INC COUNT
A8:     ASR HOLD
        BCC A9
        INC COUNT                   :FINISHED TESTING PARITY AT SHREG TAPS
A9:     INC R5                      :THIS LOOP SHIFTS THE ENTIRE ENCODER
        MOV (R5)+,CARRY             :BUFFER REGISTER
        CLR (R5)
        ASL (R0)
        BCC OVER
        INC (R5)
OVER:   ADD -(R5),(R0)+
        DEC R4
        BNE A9
        MOV ENDO,TESTO              :PUT LAST BUFFER BIT INTO TEMPORARY REGISTER
        BIT #1,TESTO                :TEST, AND
        BEQ TAB2                    :IF EQUAL TO #1#,
        INC COUNT                   :INCREMENT PARITY COUNT
TAB2:   ASR COUNT                   :TEST PARITY SUM
        BCC A10                     :IF EVEN, GO TO A10
        INC (R2)                    :IF ODD, SET OUTPUT PARITY BIT EQUAL TO #1#
A10:    TST (R2)+                   :INCREMENT OUTPUT POINTER
        DEC R3                      :TEST FOR END OF INPUT MESSAGE
        BNE LOOP                    :IF NOT DONE, GO BACK TO START
        END
```

The decoding algorithm presented here is specialized to the extent that the conventional majority decision approach of using a set of orthogonal equations as the decision variable is adopted. If desired, however, the algorithm can easily be modified to accomodate any alternate decision rule for making the correction/mode change decision.

The decoding algorithm includes the option of conditioning the random to burst mode-change decision on the requirement that there have been no errors at the decoder input in the "recent past" as suggested by Forney[6]. The purpose of this is to prevent entering the burst mode too easily since this can result in an increase in the decoded bit error rate on channels that exhibit diffuse rather than dense error bursts. This option is invoked by setting the 'DIFFUSE' flag equal to one.

4. HARDWARE

The literature on error correcting codes is replete with testimony and words of caution with regard to the pitfalls and inaccuracies that can be encountered in attempts to predict error control code performance on the basis of channel models. It is almost universally agreed that the "proof of the pudding" can only come through real-channel testing.

Since AFAL has the facilities for scintillation channel flight testing and a test modem that currently contains a feedback encoder/decoder card, it was proposed that ECI design a flexible adaptive Gallager encoder/decoder with compatible interfaces. Accordingly, a design has been produced for a variable-parameter, adaptive Gallager decoder. With the implementation described here, all of the encoder and decoder parameters affecting performance

can be varied over a wide range. With the exception of the low-power Schottky read-only memory, the entire codec has been designed with CMOS logic.

5. ADAPTIVE GALLAGER ENCODER.

A schematic diagram of the encoder is shown in Figure 8. This Figure shows that the Gallager encoding algorithm is simple to implement in hardware. Parity bit generation is implemented with a single MSI package (a CMOS 12-bit parity tree) while the parity and data are commutated to form the output data stream using an and-or select circuit. An output clock at twice the input clock frequency is generated using a combination of exclusive-or gates and a one-shot.

A normal design implementation of the Gallager encoder, including output clock generation, would require only 7 I.C.'s if an MOS LSI serial register were used for the encoder buffer. Here, however, interest lies in a flexible test codec with parameter variability. For this reason, the encoder buffer has been designed with two variable-length sections. The primary variable buffer is labeled R-1 in Figure 8. A schematic diagram of the implementation of the buffer using three CMOS LSI Quad 64-bit shift registers (Fairchild 34731's) is shown in Figure 9. As shown in the table accompanying the Figure, this implementation provides a very wide selection of buffer lengths. The buffer length selector switch is ganged to the corresponding decoder buffer length selectors.

The need for the second (8-stage) variable register arises from the decoder design. In addition to the basic buffer length parameter, the decoder has two other design parameters that affect the operation of the device. One

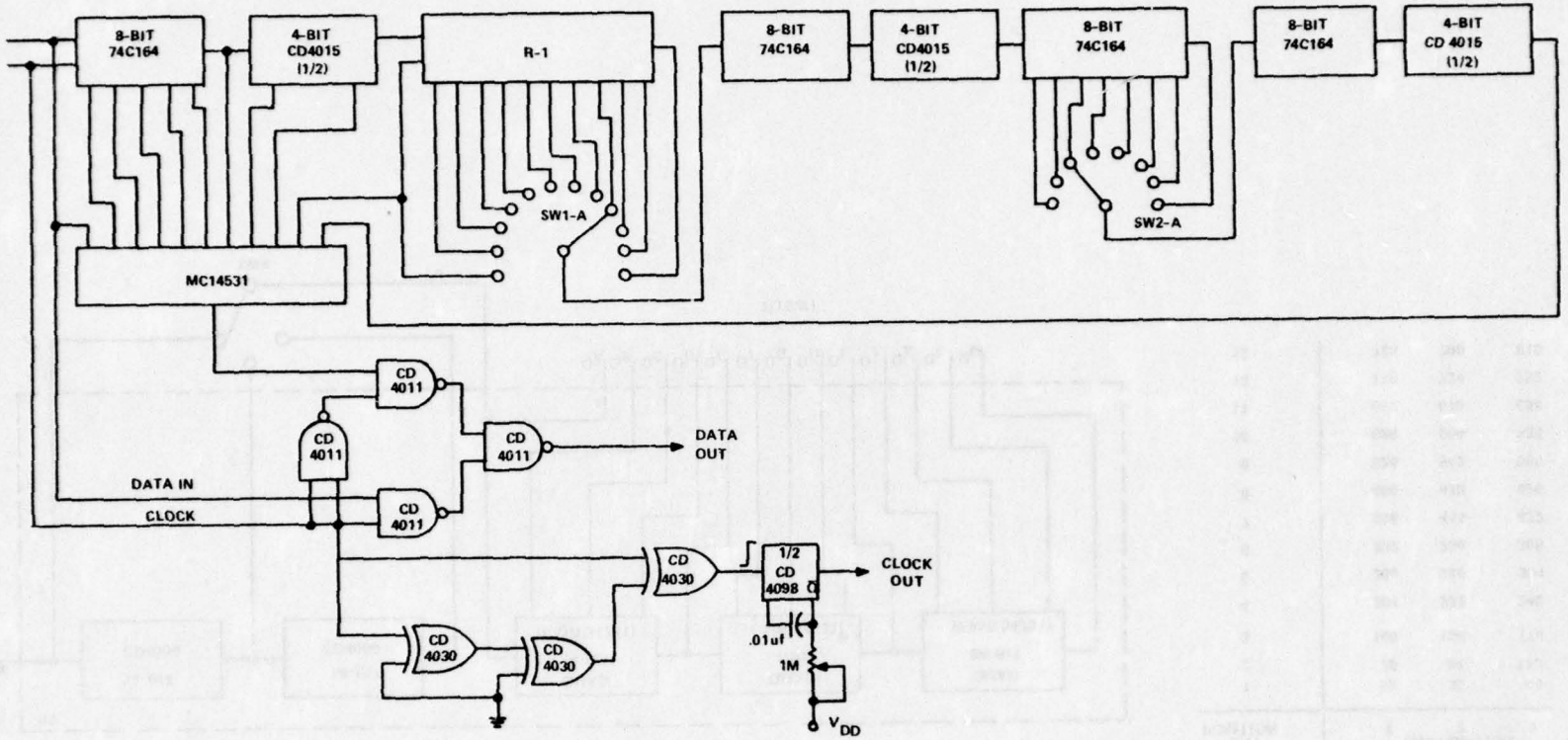
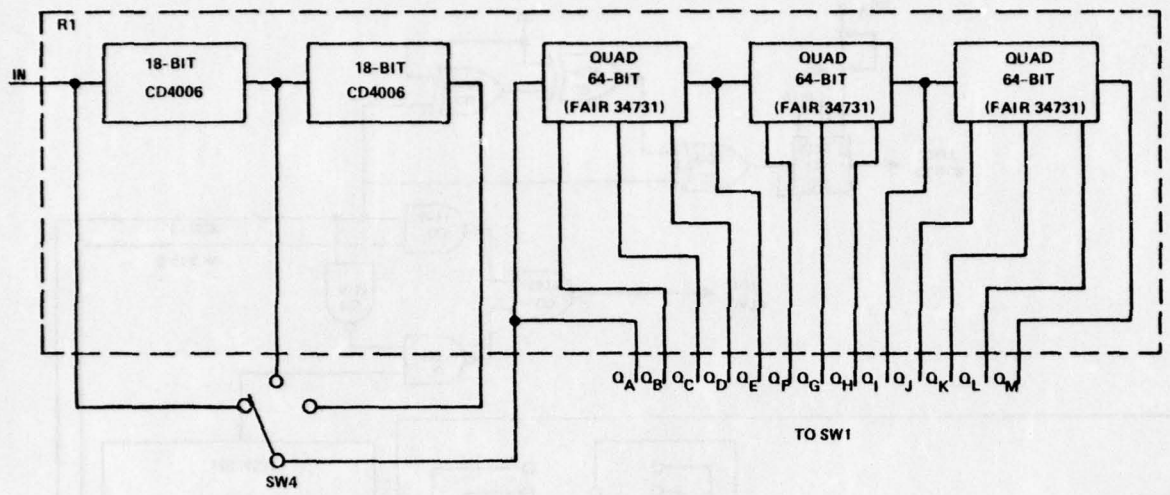


Figure 8. Schematic Diagram - Variable Parameter Adaptive Gallager Encoder (Kim's Polynomial)



R1 BUFFER LENGTH

SW1 POSITION	SW4 POSITION 1	2	3
1	12	30	48
2	76	94	112
3	140	158	176
4	204	222	240
5	268	286	304
6	332	350	368
7	396	414	432
8	460	478	496
9	524	542	560
10	588	606	624
11	652	670	688
12	716	734	752
13	780	798	816

Figure 9. Variable Buffer Implementation

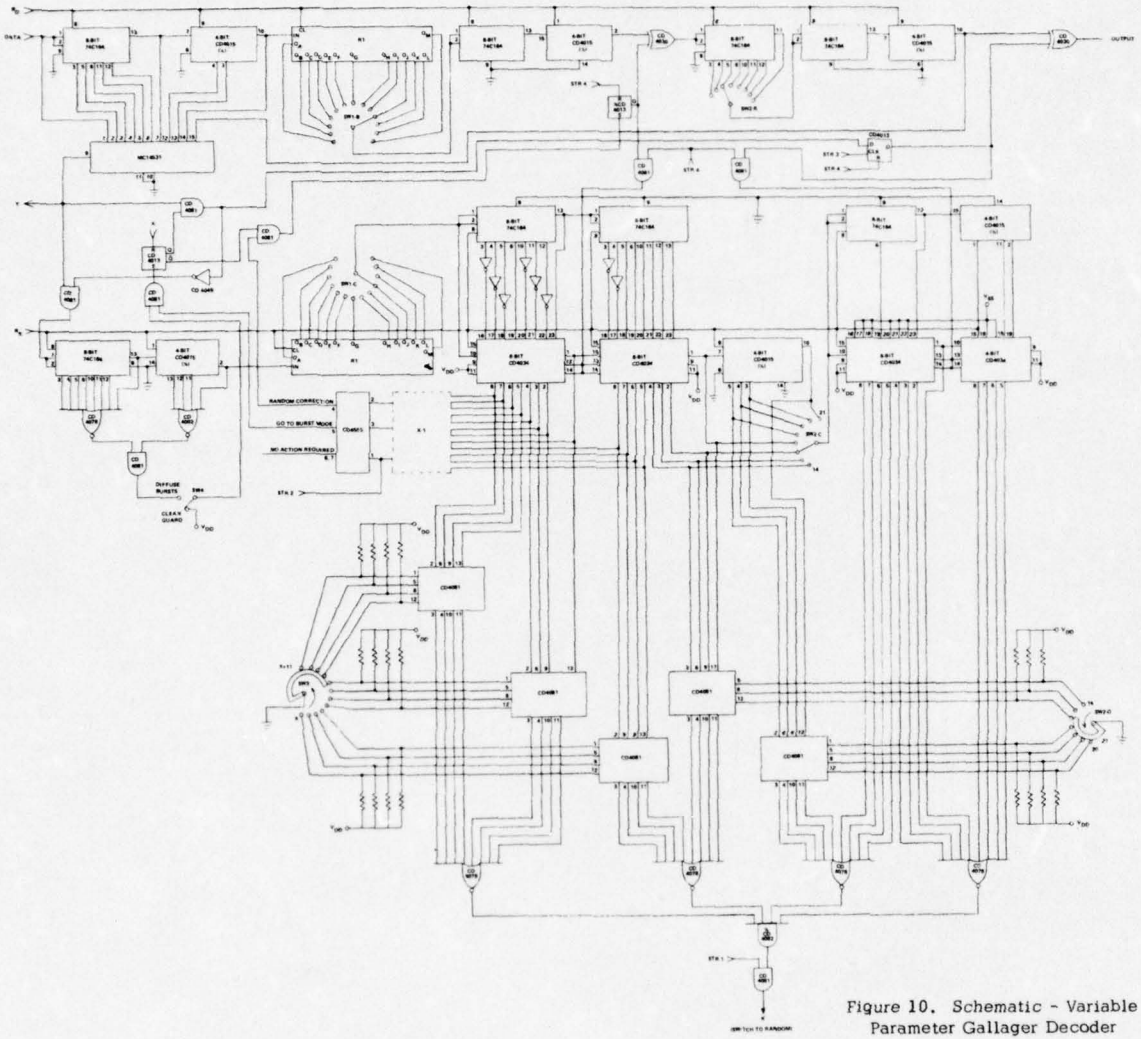


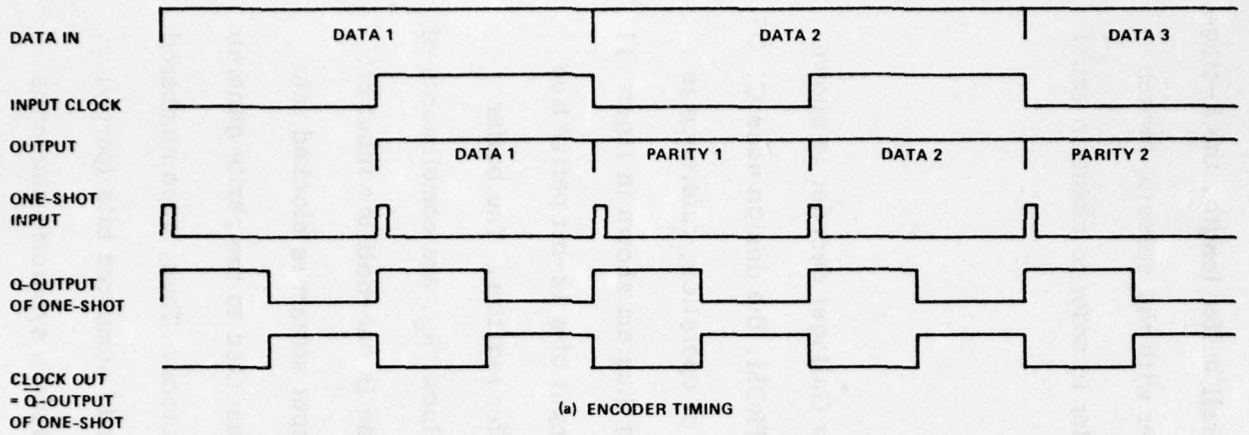
Figure 10. Schematic - Variable Parameter Gallager Decoder

of these parameters (denoted as the X-parameter) is related to the integration time allotted to the random-to-burst mode switch decision. The X-parameter is equal to one greater than the number of stages of delay between the random error and burst error correction points in the decoder buffer. Since the X-parameter is variable and affects the overall buffer length, the 8-stage variable section is included in the encoder buffer with the selector switch ganged to the corresponding switch in the decoder in order to maintain equal buffer lengths between encoder and decoder.

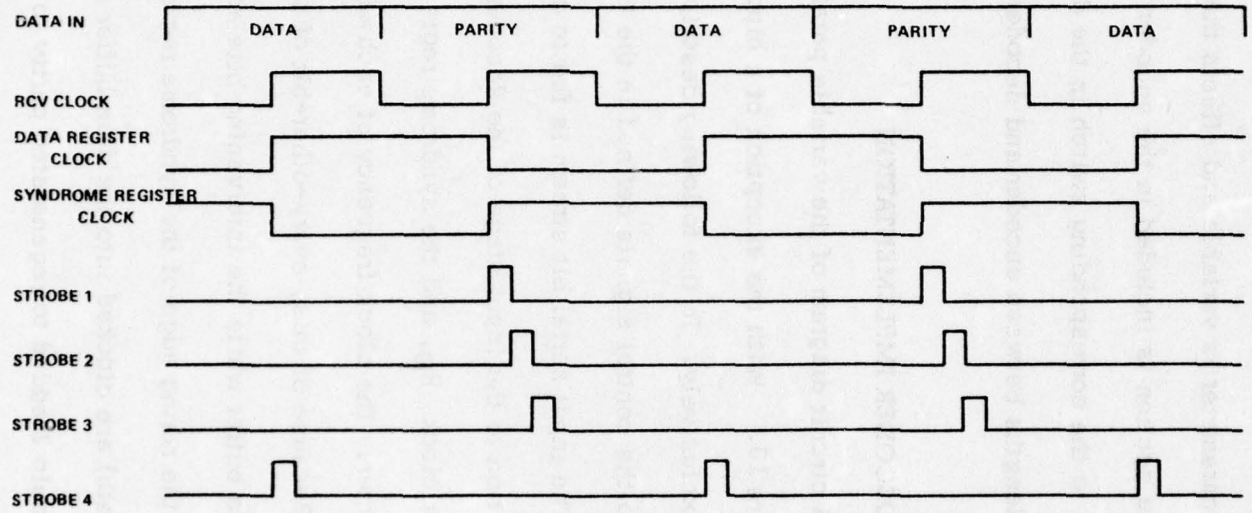
6. DECODER IMPLEMENTATION

A circuit diagram of the variable parameter Gallager decoder is shown in Figure 10. With the exception of a bipolar PROM, the design uses CMOS logic exclusively. In the following description of operation, reference is made to the control signals defined in the timing diagram shown in Figure 11

The input serial bit stream is fed to one input of a 12-bit parity tree in addition to the input stage of the decoder buffer register. The buffer register clock, R_D, and the syndrome register clock, R_S, are compliments of each other. The clock frequency of each of these is one-half the input bit rate. Because of this, every-other-bit of the input stream is clocked into the data buffer while the intervening bits are presented to the parity generator during the rising edge of the syndrome register clock. Thus, even numbered bits (data) are clocked into the data buffer and odd numbered bits (parity) are modulo 2 added to regenerated parity to produce a syndrome sequence that is shifted into the syndrome register. Upon start-up, the initial clock phases may be such that parity bits are clocked into the data buffer



(a) ENCODER TIMING



(b) DECODER TIMING

Figure 11. Timing Diagram

and data bits are presented to the parity checker. This synchronization problem and its solution are discussed in Section 7.

The following description of the sequence of decoder operations is illustrated by the flow chart shown in Figure 12 together with the schematic diagram of Figure 10 The description of operation begins with the occurrence of a leading edge of the syndrome register clock. When this occurs, the syndrome register is right-shifted one bit position. If the decoder is in the BURST mode and the current syndrome bit is a 'one', a 'zero' is shifted into the first stage of the syndrome register. The reason for this is explained later. After a short delay to allow for shift register and gate delays, strobe STR1 samples the output of the NOR gate network that is used to look for all zeros in the last 'Y' stages of the syndrome register. If an all-zero condition exists, a reset pulse to the MODE flip-flop is generated. In the reset state, the MODE flip-flop indicates the RANDOM error correction mode.

The next event that occurs, is the examination of stages X through X+11 of the syndrome register in order to determine if a random error correction is to be performed, if the unit is to be switched to the burst mode of operation, or if no action is required. The rule that is used in deciding in favor of one of these alternatives is defined by the logic contained in the block labeled X-1 in the schematic diagram. Two alternate implementations of the block X-1 are shown in Figure 13.

Approach X-1a uses a majority logic decision algorithm based on a set of orthogonal equations involving the syndrome bits. The majority decision is performed by a 64 x 2 field programmable read-only memory. There are four possible ROM outputs, two of which are "do nothing" indications. A

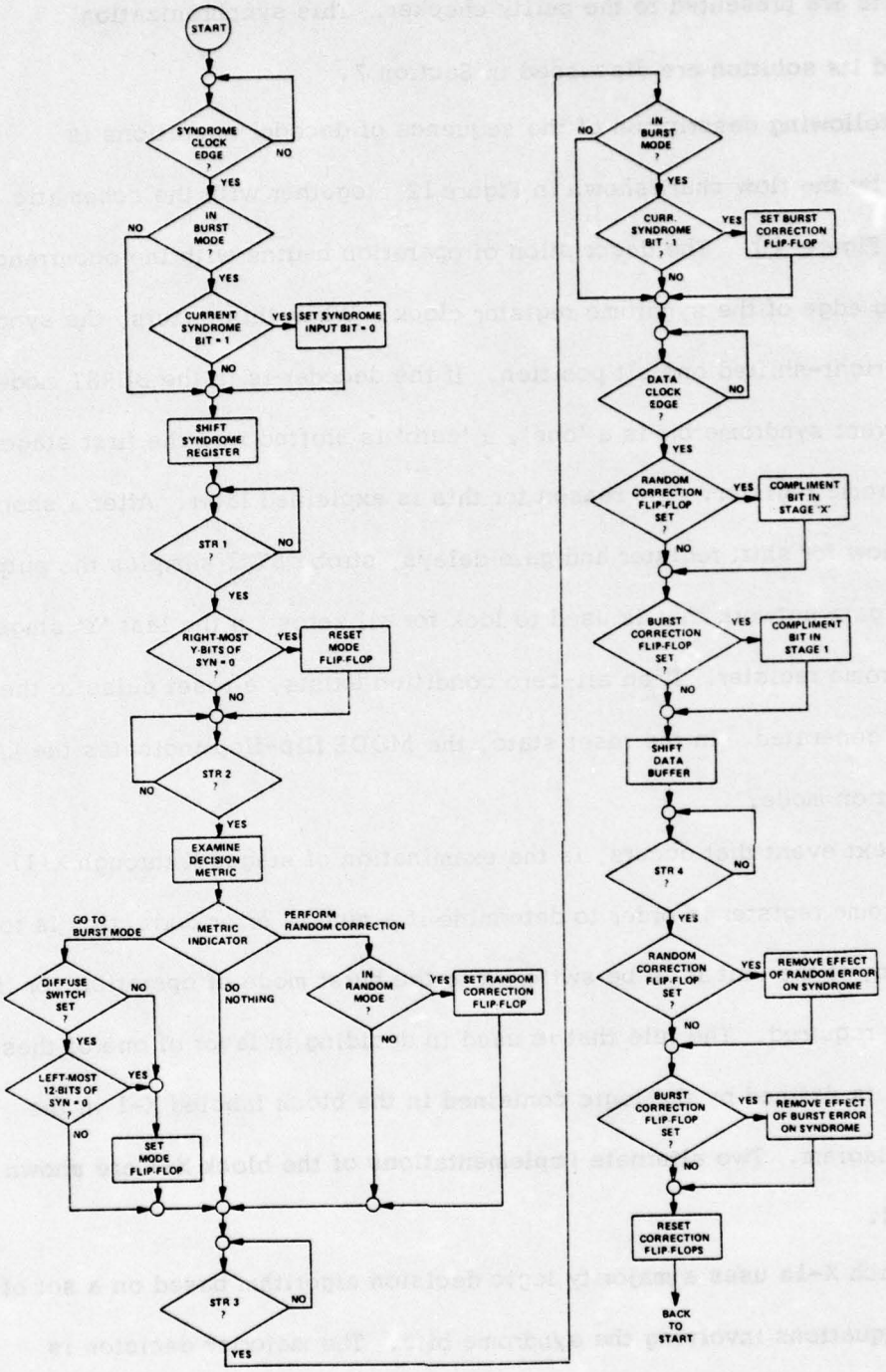


Figure 12. Flow Chart of Variable Parameter Decoder Hardware

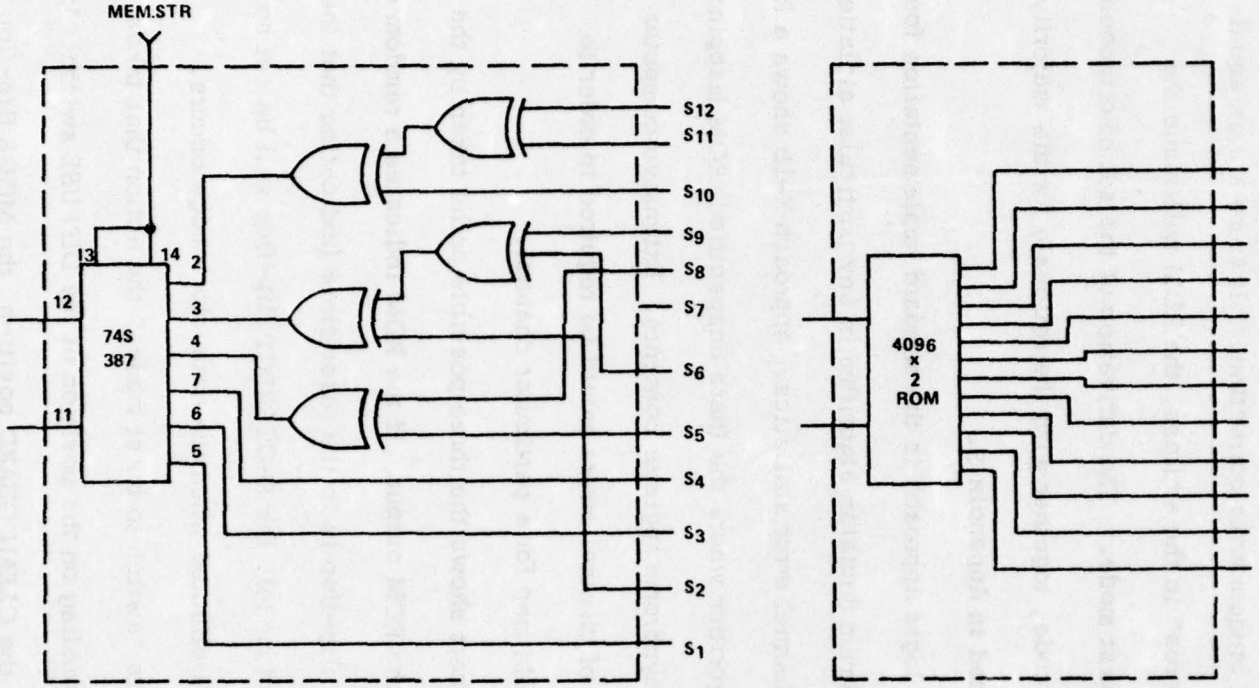


Figure 13. Alternate Decision Block Implementations

"do nothing" output results when there are four or more "zeros" on the address lines. If there are four or more "ones" in the address, the ROM will indicate, "perform a random error correction." If there are an equal number of "ones" and "zeros" in the address, the ROM will issue the command to "switch to burst mode." The derivation of the set of orthogonal equations for the 24, 12 code, together with the rationale for the majority logic approach is presented in Appendix B.

While the majority logic approach is the standard implementation found in the literature, the optimum decision algorithm in any particular situation depends greatly on the channel error statistics. Approach X-1b shows a ROM implemented decision algorithm where the (here unspecified) ROM mapping operates directly on the syndrome register contents. Extensive computer simulation using records of channel errors would be required in order to determine the optimum ROM map for a particular channel.

The decoder flow chart shows the three possible paths taken by the decoder as indicated by the ROM output. If the ROM indicates a ramdom error correction and the MODE flip-flop is in the reset state (indicating that the decoder is in the RANDOM mode), the R-CORRECT flip-flop will be set so that the correction will be effected when the data clock edge occurs.

If the ROM indicates "switch to burst mode," the action that takes place is conditional, depending on the position of the DIFFUSE switch. If the DIFFUSE switch is in the CLEAN GUARD position, the MODE flip-flop is unconditionally set effecting a switch to BURST mode. If the DIFFUSE switch is in the DIFFUSE BURST position, the switch to BURST mode is

effected only if there have been no recent errors, i.e., the last twelve syndrome bits are all zero.

If the ROM gives a "do nothing" output, nothing happens, of course, until strobe STR3 occurs.

When STR3 occurs, the B-CORRECT flip-flop will be set to the 'one' state if the D-input line is high. This condition requires that the decoder be in the BURST mode and that the current syndrome bit be a 'one'. (The principle of operation depends upon the assumption that the entire error burst is contained in the data buffer so that if the parity check fails - indicated by a syndrome of 'one' -- this could only be caused by an error in the output stage.) If the B-CORRECT flip-flop is set, the content of the output stage will be inverted the next time the data buffer is shifted.

The next event that takes place (after the occurrence of STR3) is the shifting of the data buffer and the correction of errors if so indicated by the states of the correction flip-flops.

The final action in the decoding cycle is triggered by the occurrence of STR4. The operations initiated by STR4 include removal of the effects of any errors that were corrected on the content of the syndrome register and the resetting of the correction flip-flops in preparation for the next decoding cycle. In the case where a random error correction has been performed, removal of the effect of the error on the syndrome sequence consists in complimenting certain bits in stages X through X+11 of the syndrome register. This is accomplished by incorporating two parallel-out registers that redundantly store syndrome bits S_X through S_{X+11}. After a random error correction is performed, the contents of these registers, with the appropriate bits inverted, are "jammed" into syndrome register stages X through X+11.

The same general method is used to take out the error effect in the burst mode. In this case, however, the effect is removed by clearing certain of the first twelve stages of the syndrome register and clearing the most recent syndrome bit. The former is accomplished synchronous with STR4 while the latter is accomplished by clocking a 'zero' into the syndrome register whenever a burst correction is about to be performed.

The decoder design provides for a wide range of variability of the important Gallager decoder parameters. The parameter to which decoder performance will be most sensitive is the basic buffer length, B. As selected using switches SW1 and SW4 (see Figure 9) the buffer length can be varied from 12 stages to 816 stages in relatively fine-grain increments. The formula for buffer length, B, in terms of the SW4 setting, i, and the SW1 setting, j, is given by:

$$B = 18i + 64j + 12 \; ; \; i = 0,3 \; ; \; j = 0,12$$

Both SW1 and SW4 are 3-section rotary switches so that the encoder, decoder, and syndrome buffers are varied together.

The X-parameter is equal to one greater than the number of stages of delay between the points in the data buffer at which random and burst correction is performed. The time the decoder has in which to make a random-to-burst mode change decision is thus proportional to this parameter. Very little analytical or empirical information is available to guide the selection of this parameter value (Brayer[7] used a value of X = 15 in his simulations). Here, the value has been given a range of variation of from 14 to 21 under

control of selector switch SW2. A simple wiring change (eliminating the final 4-stages of each of the buffers) would shift this selection to the range X = 10 to X = 17. Selector switch SW2 is a 4 section rotary switch. One section is devoted to each of the three buffers; the fourth section is used to disable the inputs to the burst-to-random mode switch decision logic that are associated with unused syndrome buffer stages.

The Y-parameter is equal to the number of syndrome bits that are examined in making the burst-to-random mode change decision. The Y parameter sets the degree of confidence needed in the decision that the burst has passed in order to decide in favor of a change to the random mode. If Y is made too small, errors may remain in the data buffer which will be passed to the output. The Y parameter is set by selector switch SW3 which is used to disable from 0 to 12 inputs to the mode change decision logic. This parameter can thus be varied from Y = X to Y = X+11.

7. SYNCHRONIZATION

Since this is a convolutional code, no real word sync is required. The sync problem consists only of determining the proper phasing of the decoder data and syndrome register clocks so that data bits, not parity, are clocked into the data buffer.

The two system clocks shown in Figure 11 are derived simply by dividing the received bit clock by two and using the Q and $\overline{Q}$ outputs of the divide-by-two flip-flop as shown in Figure 14. The synchronization problem arises from the uncertainty that exists with regard to the relationship between the incoming bit stream and the initial clock states. If the start-up state is

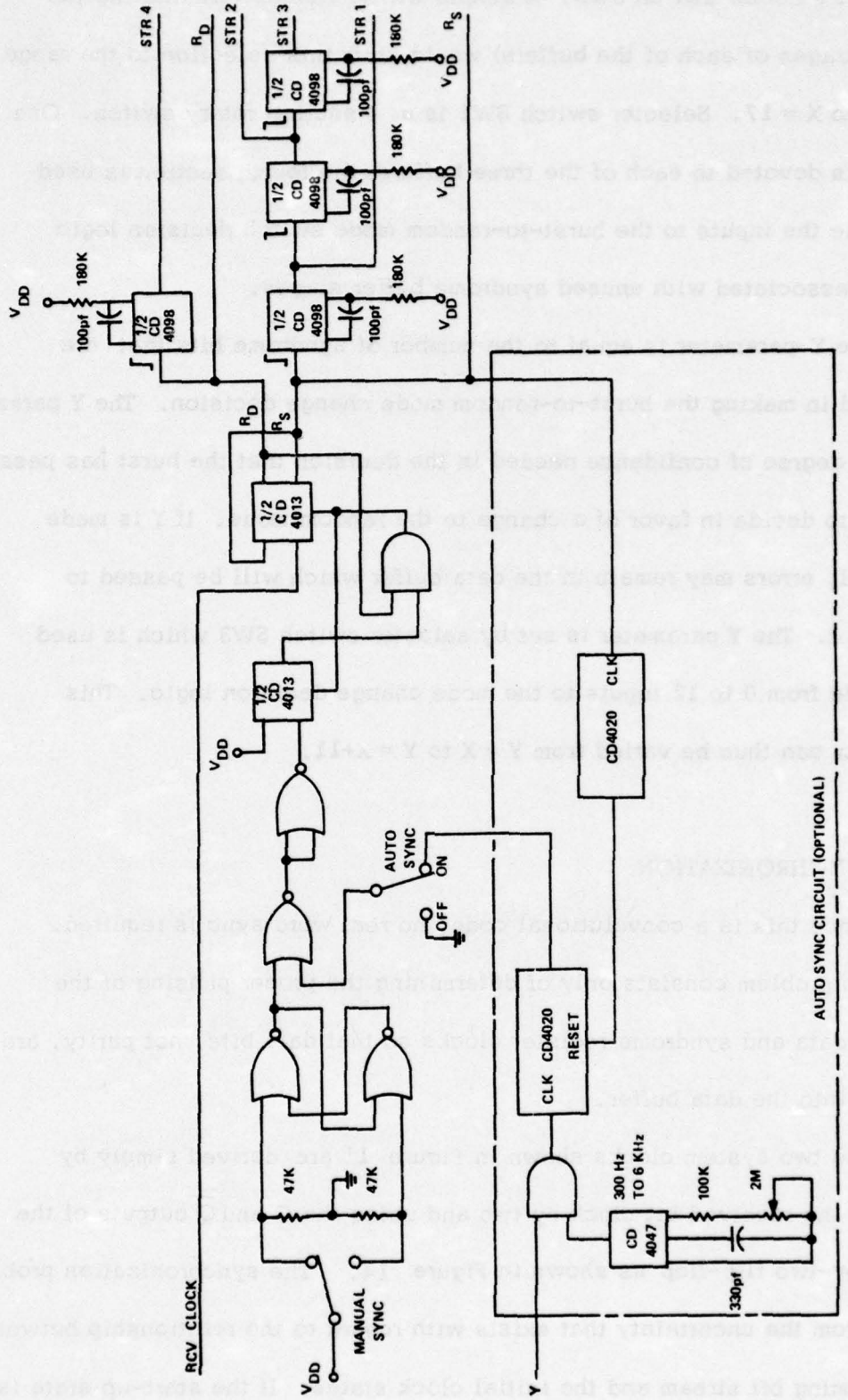


Figure 14. Timing and Synchronization Circuitry

such that the rising edge of the data register clock occurs when a parity bit is present at the input, parity bits will erroneously be clocked into the data register. This condition will occur upon start-up with probability 1/2.

Two means are provided to cure an erroneous start-up condition. One of these, the manual method, provides a pushbutton switch that can be depressed to invert both clock signals if the operator observes that the output is 'garbage.' The alternate method of sync acquisition is by means of an automatic sync circuit whose principle of operation is based on the fact that an out-of-sync condition will result in the occurrence of nearly 50% 'ones' in the syndrome sequence.

The auto sync circuit is comprised of a monostable with a variable rate from 300 Hz to 6KHz, an 'and' gate and two 14-stage ripple counters. The syndrome sequence is "and-ed" with the monostable output so that a series of clock pulses is sent to the integrating counter each time a 'one' occurs in the syndrome sequence. By varying the monostable frequency, the number of pulses produced for each occurrence of a 'one' can be varied from 4 to 80. The integrating counter is periodically reset to prevent the flagging of an "out-of-sync" indication during proper in sync operation. The frequency of these resets is adjustable by selecting the proper output tap on the reset generating counter. The auto sync circuit parameters (both counter outputs and the monostable frequency) should be adjusted experimentally so that "out-of-sync" indications are not falsely generated when even the longest error bursts are present.

A parts list for the hardware evaluator is shown in Table 4.

TABLE 4. ENCODER/DECODER PARTS LIST

Qty.	Part Number	Description	Vendor
1	CD4001	Quad 2-Input NOR Gate	RCA
2	CD4002	Dual 4-Input NOR Gate	RCA
6	CD4006	18-Stage Shift Register	RCA
1	CD4011	Quad 2-Input NAND Gate	RCA
3	CD4013	Dual D-Flip-Flop	RCA
5	CD4015	Dual 4-Bit Shift Register	RCA
2	CD4020	14-Stage Ripple Counter	RCA
7	CD4030	Quad Exclusive OR Gate	RCA
4	CD4034	8-Stage Bidirectional Shift Register	RCA
1	CD4047	Multivibrator	RCA
2	CD4049	Hex Inverter	RCA
5	CD4078	8-Input NOR Gate	RCA
1	CD4082	Dual 4-Input AND Gate	RCA
3	CD4098	Dual Neonostable Multivibrator	RCA
8	CD4081	Quad 2-Input AND Gate	RCA
1	CD4555	Dual Binary to 1 of 4 Decoder	RCA
12	74C164	8-Bit Parallel-Out Shift Register	National
2	74S387	1024-Bit Programmable ROM	TI
2	MC14531	12-Bit Parity Tree	Motorola
9	34731	Quad 64-Bit Shift Register	Fairchild
3	851-000-U2J0-101J	100 pf Ceramic Capacitor	Erie
1	8121-100-C0G0-33K	330 pf Ceramic Capacitor	Erie
1	8121-050-651-103M	.01 µf Ceramic Capacitor	Erie
1		100KΩ 1/4W 5% Resistor	
3		180KΩ 1/4W 5% Resistor	
1	3099P	1MΩ Cermet Trimpot	Bourns
1	3099P	2MΩ Cermet Trimpot	Bourns
1	SA31SDT6	Pushbutton Switch	Cutler Hammer
2	SF11SCT691	SPDT Toggle Switch	Cutler Hammer
1	2H50A16-3	2-16 Position 3 Pole Switch	Cutler Hammer
1	2E00A24-1 - Progressive Opening	24 Position Switch	Cutler Hammer
1	399720JC	2-10 Position Consecutive Shorting Switch	Oak
1	399475JC	4-Section 2-12 Position Switch	Oak

SECTION IV

CONCLUSIONS

Although the original Gallager algorithm was chosen for further study on the basis of the Phase I analysis, the extended Gallager algorithm should not be excluded from consideration for use on the scintillation channel in the future. There are two reasons for this. First, advances in coding theory may soon yield constructive procedures for the generation of more powerful convolutional codes with the containment property required by the extended algorithm. Secondly, advances in integrated circuit technology are driving the cost of digital hardware down at such a rate that the difference in implementation complexity may soon become an insignificant consideration.

This study has produced the tools necessary for conducting a quantative determination of the effectiveness of the adaptive Gallager error correcting method. The evaluation can be approached via two avenues: computer simulation using the algorithm provided or flight tests using the hardware evaluator. Since the hardware involved is relatively simple, the design is largely completed and flight tests will yield the most meaningful results, this is the approach that is recommended.

APPENDIX A

THRESHOLD DECODER SIMULATION

PROGRAM LISTING

```
THRESHOLD DECODER SIMULATOR---M.S. KIM  PAGE 1


// JOB
// FOR
*ONE WORD INTEGERS
      SUBROUTINE CODER
      COMMON LNGTH,ILAST,NR,KR,LX,LB
      COMMON INPUT(100)
      COMMON KREG(15,3),KGEN(15,3,4),NCODE(100,3)
      COMMON MODE, LL, JTH, LDCOD, JPCHS
      COMMON JOUT(100,2),JSTRG(100,2),IEROR(300)
      COMMON JSYND(100,2),JE(3),JROUT(3),KORTH(15,2,15),KORTI(15,2,15)
C  CONVOLUTIONAL CODE GENERATOR, R= KR/NR
C     I --- DATA INDEX (BIT NUMBER)        ILAST --- THE NUMBER OF DATA
C     K --- SHIFT REG INDEX                KREG((L,K)--- KR SHIFT REGISTERS
C     LNGTH --- CONSTRAINT BLOCK LENGTH    J --- CODE WORD INDEX
C     NR --- OUTPUT BIT NUMBER (N)         NCODE(J,N) --- CODE WORD STRG
C     KR --- INPUTT BIT NUMBER (L)
C     KGEN(N,L,K,)  --- CODE GENERATORS
C  DEFINITION OF FUNC MODTU(M)
      MODTU(M)=M-(M/2)*2
C  INPUT AND OUTPUT PRINTING FORMAT -----
    4 FORMAT(10X, 7011)
    6 FORMAT(1X//// 30X,≠CONVOLUTIONAL ENCODER≠///
     110X,≠DATA INDEX≠,10X,≠INPUT DATA≠,10X,≠ENCODED DATA≠//
     250X,≠G1(I)≠,3X≠G2(I)≠,3X,≠G3(I)≠,3X,≠G4(I)≠//)
C  INITIALIZATION -----
      LDLAY=LB+LX
      LSYND=LX+LB+LNGTH
      DO 105 L=1,KR
      DO 105 K=1,LNGTH
      KREG(K,L)=0
  105 CONTINUE
C  INPUT INFORMATION -----
      READ(2,4) (INPUT(I),I=1,ILAST)
C     CONVOLUTIONAL ENCODING -----
      I=1
      JNR=(ILAST/KR)
      DO 490 J=1,JNR
      DO 290 L=1,KR
      K=LNGTH
  211 K1=K-1
      IF(K1)220,220,215
  215 KREG(K,L)=KREG(K1,L)
      K=K-1
      GO TO 211
  220 KREG(K,L)=INPUT(I)
      I=I+1
  290 CONTINUE
C  PARITY CHECK BITS ---
      N=1
      DO 390 N=1,NR
      IF (N-KR) 380,380,310
  310 NTEMP=0
```

```
      L=1
      K=1
      DO 350 L=1,KR
      DO 340 K=1,LNGTH
      NTEMP=KGEN(K,L,N)*KREG(K,L)+NTEMP
  340 CONTINUE
  350 CONTINUE
      NCODE(J,N)=MODTU(NTEMP)
      GO TO 390
  380 NCODE(J,N)=KREG(1,N)
  390 CONTINUE
  490 CONTINUE
      RETURN
      END
// DUP
*STORE       WS  UA  CODER
// FOR
*ONE WORD INTEGERS
      SUBROUTINE JRDOM
      COMMON LNGTH,ILAST,NP,KR,LX,LB
      COMMON INPUT(100)
      COMMON KREG(15,3),KGEN(15,3,4),NCODE(100,3)
      COMMON MODE, LL, JTH, LDCOD, JPCHS
      COMMON JOUT(100,2),JSTRG(100,2),IEROR(300)
      COMMON JSYND(100,2),JE(3),JROUT(3),KORTH(15,2,15),KORT1(15,2,15)
C  INPUT VARIABLES---
C   LDCOD= PARITY CHECK LENGTH
C   JS = NUMBER OF PARITY EQ.
C   JTH = THRESHOLD LEVEL
C   JPCHS = DIMMENSION OF PARITY EQ.
C   KORTH(K,NS,JS) = LINEAR COMBINATION OF PARITY EQ.
      MODTU(M)=M-(M/2)*2
      NS=NP-KR
      ML=LL
      JPADD=0
      DO 2100 JS=1,JPCHS
      JORTH=0
      DO 2090 N=1,NS
      K=LB+LNGTH
      K1=LDCOD
 2020 JORTH=JSYND(K,N)*KORTH(K1,N,JS)+JORTH
      IF(K1-1)2030,2030,2025
 2025 K=K-1
      K1=K1-1
      GO TO 2020
 2030 GO TO 2090
 2090 CONTINUE
      JPADD=MODTU(JORTH)+JPADD
 2100 CONTINUE
      IF (JPADD-JTH) 2110,2120,2130
 2110 JE(ML)=0
      GO TO 2200
```

```
 2120 JE(ML)=0
      JBDET=1
      MODE=1
      GO TO 2200
 2130 JE(ML)=1
      J1=LR
      JTEMP=JSTRG(J1,1)+1
      JSTRG(J1,ML)=MODTU(JTEMP)
 2200 RETURN
      END
// DUP
*STORE       WS  UA  JRDOM
// FOR
*ONE WORD INTEGERS
** THRESHOLDING DECODER, WRITTEN BY M. KIM
* IOCS(CARD,1132 PRINTER,TYPEWRITER,KEYBOARD)
      DIMENSION JRCVD(100,3),JPGEN(4)
      COMMON LNGTH,ILAST,NR,KR,LX,LR
      COMMON INPUT(100)
      COMMON KREG(15,3),KGEN(15,3,4),NCODE(100,3)
      COMMON MODE, LL, JTH, LDCOD, JPCHS
      COMMON JOUT(100,2),JSTRG(100,2),IERROR(300)
      COMMON JSYND(100,2),JE(3),JROUT(3),KORTH(15,2,15),KORT1(15,2,15)
C  DEFINITION OF FUNCTION
      MODTU(M)=M-(M/2)*2
      MODUL(M)=M-(M/NR)*NR
C  INPUT AND OUTPUT FORMAT ---
    5 FORMAT(10X,I3)
    9 FORMAT(1X///10X,≠ORTHOGONAL LINEAR COMBINATION≠//)
   10 FORMAT (10X,≠G≠,2I1,≠(D)=≠,15I1/)
   14 FORMAT(10X,70I1)
   15 FORMAT(1X/////
     110X,≠THRESHOLD DECODING SIMULATION FOR GALLAGER CODING≠//
     21X,≠TIME INDEX≠,3X,≠DATA INPUT≠,3X,≠ENCODED DATA≠,3X,≠ERROR≠,4X,
     3≠RCVD DATA≠, 4X, ≠DECODED DATA≠//)
   16 FORMAT(5X,I4,10X,I1,13X,I1,11X,I1,10X,I1,14X,I1)
   17 FORMAT(10X,I3)
   18 FORMAT(19X,I1,13X,I1,11X,I1,10X,I1,14X,I1)
   19 FORMAT(33X,I1,11X,I1,10X,I1)
   20 FORMAT(3X,≠MODE=≠,I1,5X,≠SYND(≠,I1,≠)---S1=≠,I1,2X,≠SR=≠,I1,
     12X,≠SR=≠,I1)
   21 FORMAT(50X,I3,≠TH RANDOM ERROR CORRECTION=≠,I1)
   22 FORMAT(1X/≠ERROR COUNT=XXX≠/)
   23 FORMAT (I3)
   24 FORMAT(1X/≠ERROR POSITION,F≠,I3,≠=XXX≠/)
   25 FORMAT(I3)
C  INPUT STATEMENTS---
      READ(2,17) LNGTH
      READ(2,17) LR
      READ(2,17) LX
      READ(2,17) ILAST
      READ(2,17) NR
      READ(2,17) KR
```

```
      READ(2,17) LDCOD
      READ(2,17) JPCH1
      READ (2,5) JTH
      READ (2,5) JPCHS
C  INITIALIZATION ---
      LSYND=LX+LR+LNGTH
      LDLAY=LR+LX
      NS=NR-KR
      JNR=(ILAST/KR)
      LASTE=(ILAST/KR)*NR
      DO 500 M=1,JNR
      DO 500 N=1,NR
      JRCVD(M,N)=0
  500 CONTINUE
      DO 510 K=1,LNGTH
      DO 510 L=1,KR
      KREG(K,L)=0
  510 CONTINUE
      DO 512 L=1,KR
      DO 512 J=1,LDLAY
  512 JSTRG(J,L)=0
      DO 514 N=1,NS
      DO 514 K=1,LSYND
  514 JSYND(K,N)=0
      DO 516 I=1,LASTE
  516 IEROR(I)=0
      WRITE(1,22)
      READ (6,23) NN
      DO 3333 N=1,NN
      WRITE (1,24) N
      READ (6,25) I
      IEROR(I)=1
 3333 CONTINUE
      MODE=0
      DO 520 N=1,NR
      DO 520 L=1,KR
      READ(2,14)(KGEN(K,L,N),K=1,LNGTH)
  520 CONTINUE
      DO 530 N=1,JPCH1
      DO 530 L=1,KR
      READ (2,14) (KORTH(K,L,N),K=1,LDCOD)
  530 CONTINUE
      WRITE (3,9)
      DO 540 L=1,NS
      DO 540 N=1,JPCH1
      WRITE (3,10) N,L,(KORTH(K,L,N),K=1,LDCOD)
  540 CONTINUE
C  NOISE EMBEDDING -----
      CALL CODER
      I=1
      DO 610 J=1,JNR
      DO 605 N=1,NR
```

```
      JTEMP=NCODE(J,N)+IEROR(I)
      JRCVD(J,N)=MODTU(JTEMP)
      I=I+1
  605 CONTINUE
  610 CONTINUE
C  DECODING -----
      DO 2222 J=1,JNR
      DO 1500 N=1,NR
      IF (N-KR) 1001,1001,1400
 1001 M=LX+LR
      K=LNGTH
      JROUT(N)=JSTRG(M,N)
      JOUT(J,N)=JROUT(N)
 1010 M1=M-1
      JSTRG(M,N)=JSTRG(M1,N)
      IF (M1) 1020,1020,1016
 1016 M=M-1
      GO TO 1010
 1020 JSTRG(M,N)=KREG(LNGTH,N)
 1030 K1=K-1
      KREG(K,N)=KREG(K1,N)
      IF (K1) 1040,1040,1035
 1035 K=K-1
      GO TO 1030
 1040 KREG(K,N)=JRCVD(J,N)
      GO TO 1490
 1400 N1=N-KR
      JTEMP=0
      K=1
      L=1
      DO 1420 L=1,KR
      DO 1410 K=1,LNGTH
      JTEMP= KGEN(K,L,N)*KREG(K,L)+ JTEMP
 1410 CONTINUE
 1420 CONTINUE
      JTEMP=JTEMP+JRCVD(J,N)
      JRGEN(N1)=MODTU(JTEMP)
 1490 GO TO 1500
 1500 CONTINUE
C  SYNDROME REG SHIFTING AND ESTIMATION-----
      N1=1
      DO 1640 N1=1,NS
      K=LSYND
 1601 K1=K-1
      JSYND(K,N1)=JSYND(K1,N1)
      IF (K1) 1610,1610,1607
 1607 K=K-1
      GO TO 1601
 1610 JSYND(K,N1)=JRGEN(N1)
      WRITE (3,20) MODE,N1,JSYND(1,N1),JSYND(LDLAY,N1),JSYND(LSYND,N1)
 1640 CONTINUE
      IF (MODE) 1800,1800,1900
```

```
 1800 DO 1802 L=1,KR
      LL=L
      CALL JRDOM
      WRITE (3,21) J, JF(L)
 1802 CONTINUE
      DO 1890 L=1,KR
      IF (JF(L)-1) 1890,1820,1820
 1820 DO 1860 N=1,NS
      N1=N+KR
      K1=LR+LNGTH-1
      DO 1850 K=2,LNGTH
      JTEMP=JSYND(K1,N)+KGEN(K,N,N1)
      JSYND(K1,N)=MODTU(JTEMP)
      K1=K1-1
 1850 CONTINUE
 1860 CONTINUE
 1890 CONTINUE
      MODE=0
      GO TO 2222
C  BURST MODE ----
 1900 DO 1999 N=1,NS
      N1=N+KR
      IF(JSYND(1,N)) 1902,1990,1902
 1902 DO 1960 L=1,KR
      DO 1950 K=2,LNGTH
      K1=LNGTH+LR+LX-K+1
      IF (KGEN(K,L,N1)) 1911,1912,1911
 1911 JSYND(K1,N)=0
 1912 GO TO 1950
 1950 CONTINUE
      JSYND(1,N)=0
      JF(L)=1
      JTEMP=JSTRG(LDLAY,L)+JF(L)
      JSTRG(LDLAY,L)=MODTU(JTEMP)
 1960 CONTINUE
 1990 GO TO 1999
 1999 CONTINUE
 2222 CONTINUE
C PRINTOUT STATEMENTS------
      WRITE(3,15)
      J=0
      JIN=1
      L=1
      LAST=JNR*NR
      DO 2100 I=1,LAST
      IF (L-1) 2002,2001,2002
 2001 J=J+1
      WRITE (3,16) J,INPUT(JIN),NCODE(J,L),IEROR(I),JRCVD(I,L),JOUT(J,L)
      JIN=JIN+1
      GO TO 2099
 2002 IF (L-KR) 2005,2005,2006
 2005 WRITE (3,18) INPUT(JIN),NCODE(J,L),IEROR(I),JRCVD(J,L),JOUT(J,L)
```

```
      JIN=JIN+1
      GO TO 2099
2006  WRITE (3,19) NCODE(J,L),TEROR(I),JRCVD(J,L)
      IF (L-NR) 2007,2008,2008
2007  GO TO 2099
2008  L=1
      GO TO 2100
2099  L=L+1
2100  CONTINUE
      CALL EXIT
      END
```

APPENDIX B

DERIVATION OF ORTHOGONAL EQUATIONS

APPENDIX B

DERIVATION OF ORTHOGONAL EQUATIONS

For the half-rate convolutional code generated by,

$$G(D) = 1 + D^2 + D^3 + D^5 + D^6 + D^7 + D^9 + D^{10} + D^{11} \tag{1}$$

parity bits are generated by the formula,

$$P_{B+x+11} = I_{B+x} + I_{B+x+1} + I_{B+x+2} + I_{B+x+4} + I_{B+x+5} + I_{B+x+6} + I_{B+x+8} + I_{B+x+9} + I_{B+x+11} \tag{2}$$

where P denotes a parity bit, I denotes an information bit, and the subscript notation is in accordance with the register labels used in Figure B1.

At the decoder, shown in Figure B1, a syndrome is generated by re-calculating parity from the "noisy" information bits and comparing this parity bit with the received parity bit. The current syndrome bit is thus given by,

$$S_{B+x+11} = I'_{B+x} + I'_{B+x+1} + I'_{B+x+2} + I'_{B+x+4} + I'_{B+x+5} + I'_{B+x+6} + I'_{B+x+8} + I'_{B+x+9} + I'_{B+x+11} + P'_{B+x+11} \tag{3}$$

where the prime (') is used to indicate "noisy" received bits which, due to errors, may not equal the original bits.

Formally,

$$I'_k = I_k + E^i_k \tag{4a}$$

$$P'_k = P_k + E^p_k \tag{4b}$$

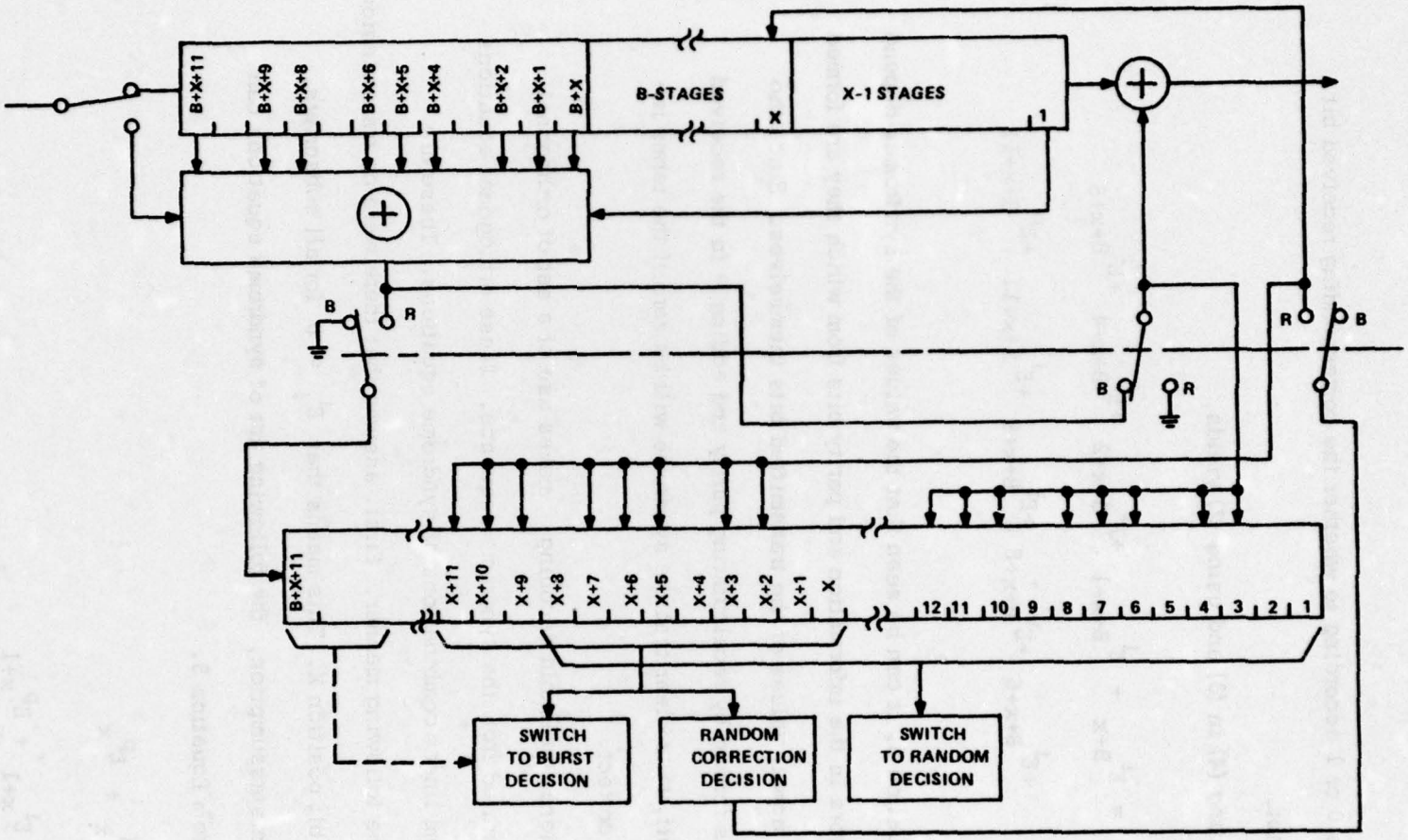


Figure B-1. Block Diagram - Adaptive Gallager Decoder (Kim's Polynomial)

Where $E_{\_}^{\_}$ is 0 or 1 according to whether the corresponding received bit is in error or not.

Substituting (4) in (3) and using (2) yields,

$$S_{B+x+11} = E^{i}_{B+x} + E^{i}_{B+x+1} + E^{i}_{B+x+2} + E^{i}_{B+x+4} + E^{i}_{B+x+5} + E^{i}_{B+x+6} + E^{i}_{B+x+8} + E^{i}_{B+x+9} + E^{i}_{B+x+11} + E^{p}_{B+x+11} \quad (5)$$

From Equation 5, it can be seen that the values of the syndromes depend only on the errors in the information and parity bits from which they are formed and not on the actual values of the transmitted bits themselves. Since the syndrome bit is formed by recalculating parity and adding it to the received version of itself, it is clear that the syndrome will be zero if the terms involved are all correct.

Conventional threshold decoding makes use of a set of orthogonal equations generated from the syndrome sequence. These orthogonal equations are derived from linear combinations of syndrome equations. These are generated in the following manner. First, assume that there were no transmission errors prior to bit position X. This means that $E^{i}_{j} = 0$ for all subscripts, $j < x$. With this assumption, the following set of syndrome equations can be generated from Equation 5.

$$S_x = E^{i}_{x} + E^{p}_{x}$$

$$S_{x+1} = E^{i}_{x+1} + E^{p}_{x+1}$$

$$S_{x+2} = E^i_x + E^i_{x+2} + E^p_{x+2}$$

$$S_{x+3} = E^i_x + E^i_{x+1} + E^i_{x+3} + E^p_{x+3}$$

$$S_{x+4} = E^i_{x+1} + E^i_{x+2} + E^i_{x+4} + E^p_{x+4}$$

$$S_{x+5} = E^i_x + E^i_{x+2} + E^i_{x+3} + E^i_{x+5} + E^p_{x+5}$$

$$S_{x+6} = E^i_x + E^i_{x+1} + E^i_{x+3} + E^i_{x+4} + E^i_{x+6} + E^p_{x+6}$$

$$S_{x+7} = E^i_x + E^i_{x+1} + E^i_{x+2} + E^i_{x+4} + E^i_{x+5} + E^i_{x+7} + E^p_{x+7}$$

$$S_{x+8} = E^i_{x+1} + E^i_{x+2} + E^i_{x+3} + E^i_{x+5} + E^i_{x+6} E^i_{x+8} + E^p_{x+8}$$

$$S_{x+9} = E^i_x + E^i_{x+2} + E^i_{x+3} + E^i_{x+4} + E^i_{x+6} + E^i_{x+7} + E^i_{x+9} + E^p_{x+9}$$

$$S_{x+10} = E^i_x + E^i_{x+1} + E^i_{x+3} + E^i_{x+4} + E^i_{x+5} + E^i_{x+7} + E^i_{x+8} + E^i_{x+10} + E^p_{x+10}$$

$$S_{x+11} = E^i_x + E^i_{x+1} + E^i_{x+2} + E^i_{x+4} + E^i_{x+5} + E^i_{x+6} + E^i_{x+8} + E^i_{x+9} + E^i_{x+11} + E^p_{x+11}$$

(6)

The following set of orthogonal equations is generated by forming linear combinations of Equations 6. The subscripts (with the x's dropped) indicate which syndrome equations were used to form the resulting equation, e.g., $A_{4,7}$ denotes a linear combination of syndrome equations S_{x+4} and S_{x+7}.

$$
\begin{aligned}
A_o &= E^i_x + E^p_x \\
A_2 &= E^i_x + E^i_{x+2} + E^p_{x+2} \\
A_3 &= E^i_x + E^i_{x+1} + E^i_{x+3} + E^p_{x+3} \\
A_{4,7} &= E^i_x + E^p_{x+4} + E^i_{x+5} + E^i_{x+7} + E^p_{x+7} \\
A_{1,5,8} &= E^i_x + E^p_{x+1} + E^p_{x+5} + E^i_{x+6} + E^i_{x+8} + E^p_{x+8} \\
A_{9,10,11} &= E^i_x + E^i_{x+4} + E^p_{x+9} + E^i_{x+10} + E^p_{x+10} + E^i_{x+11} + E^p_{x+11}
\end{aligned}
\tag{7}
$$

This set of equations has the orthogonal properties: (1) E^i_x appears in every equation, and (2) no other bit error term appears more than once in the entire set. Since the set is orthogonal on E^i_x, it is possible to solve for this term if not more than three errors are present in the terms involved in the equations.

REFERENCES

1) Massey, J.L., Threshold Decoding, MIT Press, 1963.

2) Sullivan, D.D., "A Generalization of Gallager's Adaptive Error Control Scheme," IEEE Transactions on Information Theory, Vol. IT-17, No. 6, November 1971

3) Ferguson, M.J., "Contained Convolutional Codes," IEEE Transactions on Information Theory, Vol. IT-18, No. 3, May 1972

4) Wu, W.W., "New Convolutional Codes - Part I," IEEE Transactions on Communications, Vol. COM-23, No. 9, September 1975

5) Wu, W.W., "New Convolutional Codes - Part II," IEEE Transactions on Communications, Vol. COM-24, No. 1, January 1976

6) Forney, G.D., Jr. and Kohlenberg, A., "Convolutional Coding for Channels with Memory," IEEE Transactions on Information Theory, Vol. IT-14, No. 5, September 1968

7) Brayer, K., "Error Correction Code Performance On HF, Troposcatter and Satellite Channels," IEEE Transactions on Communication Technology, Vol. COM-19, No. 5, October 1971

☆ U. S. GOVERNMENT PRINTING OFFICE: 1976 — 657-630/1055