AFCRL-63-28

TWELFTH SCIENTIFIC REPORT

Computer Construction and Evaluation of Long Burst-Error Correcting Codes

J. J. Metzner

January 31, 1963

K UNIVERSITY

NEERING

ering

AFCRL-63-28

COMPUTER CONSTRUCTION AND EVALUATION OF LONG BURST-ERROR CORRECTING CODES

J. J. Metzner

NEW YORK UNIVERSITY
COLLEGE OF ENGINEERING
DEPARTMENT OF ELECTRICAL ENGINEERING
Laboratory for Electroscience Research

University Heights
New York 53, New York

TWELFTH SCIENTIFIC REPORT

Contract AF19(604)-6168
Project 4610
Task 461003

January 31, 1963

Approved by
Leonard S. Schwartz
Project Director

Prepared
for

ELECTRONICS RESEARCH DIRECTORATE
AIR FORCE CAMBRIDGE RESEARCH LABORATORIES
OFFICE OF AEROSPACE RESEARCH
UNITED STATES AIR FORCE
BEDFORD, MASSACHUSETTS

NOTICE

Requests for additional copies by Agencies of the Department of Defense, their contractors, and other Government agencies should be directed to the:

ARMED SERVICES TECHNICAL INFORMATION AGENCY
ARLINGTON HALL STATION
ARLINGTON 12, VIRGINIA

Department of Defense contractors must be established for ASTIA services or have their "need-to-know" certified by the cognizant military agency of their project or contract.

All other persons and organizations should apply to the:

U. S. DEPARTMENT OF COMMERCE
OFFICE OF TECHNICAL SERVICES
WASHINGTON 25, D. C.

ABSTRACT

This report presents a specific method of programming a digital computer to construct and evaluate codes suitable for use in a burst-error correction decoding scheme described in the Tenth Scientific Report.[1] Two (100,50) codes of this type have been constructed using the methods described. One is found capable of correcting uniquely all bursts of length 21 or less.

The method used to evaluate burst-error correcting capabilities is applicable to any group code, not just the class under consideration. Memory limitations have restricted the present program to code lengths not exceeding 105 digits, of which at most 63 may be check digits.

GLOSSARY

B	- set of error burst sequences
b	- maximum correctible burst length
c	- number of check digits per code word
c_i	- i^{th} check digit
$[e_i]$	- partial error sequence
$[e_i]_A$	- error sequence
i_j	- j^{th} information digit
k	- number of information digits per code word
[M]	- matrix relating information digits to check digits
$[M_1]$, $[M_2]$	- submatrices of [M]
n	- number of digits in a code word
P	- record of column under consideration
P_s	- record of columns already considered, including present
p_i	- i^{th} parity digit
[P]	- parity check, or coding, matrix
$[P_1]$, $[P_2]$,.......,$[P_q]$	- submatrices of [P], all of which have inverses
[p]	- parity sequence
q	- number of decoding matrices
$[Q_1]$, $[Q_2]$, $[Q_3]$,....	- decoding matrices
R	- random number
r_i	- i^{th} received digit
$[R_1]$, $[R_2]$	- submatrices used in second construction method

[r]	- received sequence
S_i	- i^{th} set of columns
X	- first part of set of columns
x	- amount of overlap of two matrices
Y	- second part of set of columns
Y_o	- initial second part of set of columns

TABLE OF CONTENTS

A. Introduction and Basic Theory

The Tenth Scientific Report[1] described a coding and decoding method for burst-error correction. The coding and decoding processes are fairly simple, but the actual construction and evaluation of a code is difficult. The purpose of this report is to give a specific method for constructing such a code and evaluating its correction capabilities, and to illustrate codes actually constructed by this method.

The theory of the burst-error correction procedure for randomly chosen group codes was given in the Tenth Scientific Report.[1] Only the basic ideas needed for an understanding of the results will be repeated here.

In an (n,k) group code,[2] the n-k = c check digits c_i may be computed from the k information digits i_j by the matrix equation

$$\begin{bmatrix} c_1 \\ c_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ c_c \end{bmatrix} = [M] \cdot \begin{bmatrix} i_1 \\ i_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ i_k \end{bmatrix} , \qquad (1)$$

where [M] is a c x k matrix of ones and zeroes, and addition is modulo two.

Let the last c digits in a transmitted code word be the check digits. When a sequence of binary digits r_i (i=1, 2, ..., n) representing message plus noise is received, a parity sequence can be formed:

$$\begin{bmatrix} p_1 \\ p_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ p_c \end{bmatrix} = \left[\begin{array}{c:c} [M] & [I] \end{array} \right] \begin{bmatrix} r_1 \\ r_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ r_n \end{bmatrix} , \qquad (2)$$

where [I] is a c x c unit diagonal matrix. In abbreviated form, (2) may be written as

$$[p] = [P]\,[r] \; . \qquad (3)$$

The received sequence can be considered as the modulo two digit-by-digit sum of the transmitted sequence and a noise (or error) sequence. The group of 2^n possible error sequences can be partitioned into 2^c cosets of the group of code words. All the sequences in each coset have the same parity sequence and no two sequences in different cosets have the same parity sequence.[2]

The error correction procedure is based on a set of hypotheses that the errors are all within given blocks of c positions of the received sequence. A set of c x c matrices $[P_1]$, $[P_2]$, ..., $[P_q]$ is chosen, such that $[P_1]$ consists of the columns 1 to c of [P], $[P_2]$ consists of the columns c - x + 1 to 2c - x ($x \geq b$), etc., so that q is the smallest integer for which each matrix overlaps the preceding one in at least b positions, and every position is included in at least one matrix. The value of q is the smallest integer equal to or greater than $(n-b)/(c-b)$.

Let B be the set of error sequences consisting of bursts of length b or less. Assume that:

1) no two elements of B appear in the same coset and
2) each matrix $[P_i]$ has an inverse.

The decoding procedure can then correct uniquely every element of B. The quantities

$$[e_i] = [P_i]^{-1} [P] [r] = [Q_i] [r] \tag{4}$$

are calculated. The error sequence corresponding to $[e_i]$ is $[e_i]_A$, where $[e_i]_A$ is obtained from $[e_i]$ by augmenting $[e_i]$ with c zeroes corresponding to the positions hypothesized to be error-free. If one of the $[e_i]_A \in B$, the procedure is to add that $[e_i]_A$ to $[r]$. If no $[e_i]_A \in B$, the code word is rejected and a repeat is requested.

B. Specific Codes

Two (100,50) codes were constructed in a random manner with the aid of a Packard Bell pb 250 computer. The better of the two codes is illustrated in this section. The other is shown in Appendix H. Additional codes could be constructed and evaluated at a rate of about one every 4-5 hours.

Methods of construction are described in Section C, and further details of the construction program are given in Appendices B and C. The method of evaluation is given in Section D and Appendix D.

B-1. Coding and Decoding Matrices for a Particular Code

The coding matrix [P] for a particular (100,50) code is shown in figure 1. The digits of the matrix are represented octally, each octal digit representing three successive binary digits in a row of the matrix. Since 100 is not a multiple of 3, the last octal digit in a row represents only a single binary digit, being a 4 or a 0, according as the binary digit is a 1 or a 0, respectively. The + signs should be ignored.

Three decoding matrices are required, covering the sets of positions 1-50, 26-75, and 51-100, respectively. These matrices are of the form:

$$[Q_1] = [P_1]^{-1}\,[P] \quad ; \tag{5}$$

$$[Q_2] = [P_2]^{-1}\,[P] \quad ; \tag{6}$$

and

$$[Q_3] = [P_3]^{-1}\,[P] = [P] \tag{7}$$

30102$30102	30103$30103	30104$30104	30105$30105	30106$30106
D+7650103	D+4226222	D+4150000	D+0000000	D+0000000
+6026515	D+7632523	+4304000	+0000000	+0000000
+2073145	+3567264	+2522000	+0000000	+0000000
+1705543	+7122033	+6521000	+0000000	+0000000
+1047617	+6336432	+1500400	+0000000	+0000000
+1450526	+2016664	+4160200	+0000000	+0000000
+3340014	+2560623	+0500100	+0000000	+0000000
+0732456	+1177552	+2040040	+0000000	+0000000
+1151523	+1247332	+7560020	+0000000	+0000000
+3332160	+0141464	+5760010	+0000000	+0000000
+4730620	+2047633	+5660004	+0000000	+0000000
+1043562	+5717040	+2140002	+0000000	+0000000
+5612152	+2067673	+3400001	+0000000	+0000000
+0746154	+5346121	+3400000	+4000000	+0000000
+4637071	+5244371	+0040000	+2000000	+0000000
+6572605	+5123643	+1300000	+1000000	+0000000
+2576237	+4544251	+2560000	+0400000	+0000000
+6204066	+1712676	+7340000	+0200000	+0000000
+6771731	+7404122	+4100000	+0100000	+0000000
+5274760	+2405260	+0000000	+0040000	+0000000
+6335700	+4400366	+0460000	+0020000	+0000000
+4776712	+1752202	+3320000	+0010000	+0000000
+6470220	+5617650	+7560000	+0004000	+0000000
+0622164	+5707721	+2700000	+0002000	+0000000
+5264424	+2310460	+1220000	+0001000	+0000000
+1104771	+5206123	+2340000	+0000400	+0000000
+7536347	+7254276	+2520000	+0000200	+0000000
+0653760	+0307347	+4260000	+0000100	+0000000
+7515446	+6322055	+4200000	+0000040	+0000000
+4515122	+5435675	+4500000	+0000020	+0000000
+0157712	+2032431	+0200000	+0000010	+0000000
+3023503	+6025552	+2720000	+0000004	+0000000
+4051431	+1704437	+3220000	+0000002	+0000000
+7660354	+7464577	+5120000	+0000001	+0000000
+2501316	+0163264	+2360000	+0000000	+4000000
+4350454	+1002101	+3100000	+0000000	+2000000
+3320706	+4061557	+2040000	+0000000	+1000000
+6770605	+4772551	+3700000	+0000000	+0400000
+3257041	+4041441	+7360000	+0000000	+0200000
+2312161	+4327236	+6760000	+0000000	+0100000
+4302121	+2007713	+3640000	+0000000	+0040000
+0242560	+6367475	+6460000	+0000000	+0020000
+5556700	+1412400	+2060000	+0000000	+0010000
+7571741	+0017715	+1200000	+0000000	+0004000
+1040433	+2275346	+2020000	+0000000	+0002000
+0353252	+4107564	+2060000	+0000000	+0001000
+1214722	+3322517	+1040000	+0000000	+0000400
+3212635	+4420066	+3060000	+0000000	+0000200
+4004100	+2530000	+1460000	+0000000	+0000100
+5400273	+2512507	+2100000	+0000000	+0000040

Figure 1 - Coding (and Decoding) Matrix [P] for a (100, 50) Code [key = +3574263]

Since $[P_3]^{-1} = [I]$, the third decoding matrix is just [P] itself. The other two are shown in figures 2 and 3.

(Actually, any one of the three matrices in figures 1, 2, and 3, can be chosen as the coding matrix. The only difference would be in the check digit locations. If figure 2 were the coding matrix, the check digits would occupy positions 26-75, while if figure 3 were the coding matrix they would occupy positions 1-50).

Although the above code appears to be random, and can be considered to be random for its intended use, it was actually generated by the computer in a deterministic manner by successive multiplications and retention of the minor product.[3] The entire code can always be regenerated by means of the key number + 3574263. How this is done will be explained in Appendix B. The code described in Appendix H was not generated in this manner, but was constructed from random numbers taken partly from a table of random numbers,[4] and partly from digits selected from the decimal expansion of the natural logarithmic base e. It is recommended, however, that any future codes be constructed by the much simpler process of computer generation.

B-2. <u>Error-Correcting Capabilities</u>

It was found that the code whose matrix is shown in figure 1 is capable of correcting uniquely all error bursts of length 21 or less, but not including all bursts of length 22. The other chosen code was found capable of correcting all error bursts of length 18 or less, but not all bursts of length 19.

30102	30103	30104	30105	30106
D+4000000	D+0000000	D+0006103	D+6322465	D+2303300
+2000000	+0000000	+0017121	+2245415	+2022200
+1000000	+0000000	+0016766	+1245101	+5361540
+0400000	+0000000	+0015546	+6040247	+2340300
+0200000	+0000000	+0003475	+2123674	+6653040
+0100000	+0000000	+0011624	+1453546	+7651740
+0040000	+0000000	+0015463	+7560105	+3772040
+0020000	+0000000	+0003053	+6655232	+2473200
+0010000	+0000000	+0013317	+1426420	+1255600
+0004000	+0000000	+0010624	+0403022	+4464340
+0002000	+0000000	+0000350	+6501514	+4017300
+0001000	+0000000	+0000114	+5115146	+6173300
+0000400	+0000000	+0007427	+1644737	+7205300
+0000200	+0000000	+0012674	+5363242	+5351700
+0000100	+0000000	+0001503	+3061475	+0225400
+0000040	+0000000	+0012750	+6273072	+5747700
+0000020	+0000000	+0001000	+6216235	+4206140
+0000010	+0000000	+0001703	+6672427	+6777600
+0000004	+0000000	+0002303	+2625124	+5606100
+0000002	+0000000	+0000402	+2063250	+7325400
+0000001	+0000000	+0013525	+2472766	+2647700
+0000000	+4000000	+0010231	+0142716	+4003000
+0000000	+2000000	+0006327	+0471633	+4271700
+0000000	+1000000	+0000432	+6375440	+5007200
+0000000	+0400000	+0015214	+6473312	+0035740
+0000000	+0200000	+0007033	+6436304	+6561200
+0000000	+0100000	+0017502	+0605744	+0335640
+0000000	+0040000	+0015367	+5641033	+4141600
+0000000	+0020000	+0000066	+0052601	+4433440
+0000000	+0010000	+0006011	+6362537	+1423440
+0000000	+0004000	+0000562	+0531153	+4326540
+0000000	+0002000	+0002766	+2671542	+7050400
+0000000	+0001000	+0006223	+6677363	+4361100
+0000000	+0000400	+0011015	+3051023	+7413600
+0000000	+0000200	+0013247	+1445447	+0361440
+0000000	+0000100	+0012504	+4156510	+4430340
+0000000	+0000040	+0013713	+3313627	+1043300
+0000000	+0000020	+0014035	+0225213	+3410400
+0000000	+0000010	+0003656	+0337470	+6547240
+0000000	+0000004	+0004174	+1750457	+4441000
+0000000	+0000002	+0015252	+7101135	+3766540
+0000000	+0000001	+0000042	+7524566	+0320200
+0000000	+0000000	+4011440	+7401210	+1552600
+0000000	+0000000	+2011411	+3652532	+7170400
+0000000	+0000000	+1010453	+1326050	+0753300
+0000000	+0000000	+0402733	+7531071	+6514340
+0000000	+0000000	+0216363	+7302015	+4500100
+0000000	+0000000	+0115501	+5776422	+3371100
+0000000	+0000000	+0056426	+4507675	+1320340
+0000000	+0000000	+0031124	+7167735	+0544040

Figure 2 - Decoding Matrix Q_1 for the (100, 50) Code

30102	30103	30104	30105	30106
D+3713536	D+6200000	D+0000000	D+0000016	D+6451500
+7774554	+1500000	+0000000	+0000114	+6323340
+1551671	+1440000	+0000000	+0000171	+2672440
+5355132	+0420000	+0000000	+0000733	+7027300
+1137304	+5010000	+0000000	+0000757	+7543100
+0402050	+2004000	+0000000	+0000466	+5230340
+1557221	+2002000	+0000000	+0000335	+4366640
+5670712	+5001000	+0000000	+0000102	+5063040
+7733445	+5400400	+0000000	+0000324	+4561500
+7611022	+4000200	+0000000	+0000310	+1434300
+6771365	+5000100	+0000000	+0000320	+3004040
+5441063	+7400040	+0000000	+0000041	+0703200
+1523136	+2000020	+0000000	+0000553	+7771640
+6467554	+6000010	+0000000	+0000616	+2743140
+4427670	+6400004	+0000000	+0000515	+5454540
+7762060	+4000002	+0000000	+0000243	+5230200
+1407741	+2000001	+0000000	+0000277	+0601000
+5764352	+0000000	+4000000	+0000526	+1376340
+5601145	+1400000	+2000000	+0000034	+5063240
+3565222	+0000000	+1000000	+0000231	+7565040
+3073765	+4000000	+0400000	+0000433	+3015000
+4412063	+6000000	+0200000	+0000157	+5550640
+4615136	+5400000	+0100000	+0000067	+7065400
+7273554	+0000000	+0040000	+0000406	+1315400
+0277671	+2400000	+0020000	+0000374	+3022400
+7411132	+5000000	+0010000	+0000531	+4453700
+3367304	+1000000	+0004000	+0000232	+4235140
+0762051	+3400000	+0002000	+0000504	+6460100
+4126272	+4400000	+0001000	+0000451	+6726200
+3101204	+0000000	+0000400	+0000272	+6136500
+7120650	+2400000	+0000200	+0000204	+5142440
+0332620	+6000000	+0000100	+0000651	+0713240
+1324640	+5000000	+0000040	+0000672	+4563200
+5112101	+0000000	+0000020	+0000205	+7013700
+5560252	+6400000	+0000010	+0000722	+2471200
+6023745	+5400000	+0000004	+0000164	+1270740
+3010623	+1400000	+0000002	+0000410	+0714240
+4366037	+7400000	+0000001	+0000521	+5026100
+7667426	+3400000	+0000000	+4000512	+6431300
+3266635	+0000000	+0000000	+2000044	+7606040
+0310702	+0400000	+0000000	+1000750	+5236240
+4102725	+3400000	+0000000	+0400021	+1630240
+7515363	+3400000	+0000000	+0200313	+3107240
+7732736	+3000000	+0000000	+0100517	+2466000
+3775155	+4000000	+0000000	+0040366	+6615440
+4005742	+4000000	+0000000	+0020135	+4104040
+7447425	+0000000	+0000000	+0010503	+6130000
+0617562	+1000000	+0000000	+0004376	+7405140
+0262265	+5400000	+0000000	+0002654	+1325300
+4665662	+4000000	+0000000	+0001071	+5735400

Figure 3 - Decoding Matrix Q_s for the (100, 50) Code

It was shown in Section V.B of reference 1 that of the (100,50) codes for which all $[P_1]$ have inverses, at least 79 percent can correct all bursts of length 19 or less, and at least 18 percent can correct all bursts of length 20 or less. The bound does not guarantee that any (100,50) code could correct all bursts of length 21 or less. The fact that one of two selected codes could correct all bursts of length 21 or less suggests that the bound is somewhat pessimistic. A definite upper bound on the maximum burst length is c/2, or 25 in this case.

The method by which the error-correcting capability of the codes were determined is described in Section D and Appendix D.

B-3. Implementation of the Decoding Procedure

Although the selection and evaluation of a code is difficult, its actual implementation is not difficult. The matrix decoding operations can be carried out by means of a very elementary type of magnetic core matrix, as shown in figure 4.

A core is placed at an intersection if and only if the intersection of row i with column j of the matrix is a one. (Dummy cores can be placed at other intersections, if necessary). The cores are originally set in their zero state. When the first digit is received, the first vertical wire is energized if and only if the digit is a one, and a pulse passes along all horizontal wires which intersect with the first vertical wire at a core. Later received digits act in a similar manner. Flip-flops on each horizontal wire record zero or one according as an even or odd number of impulses have occurred on that wire.

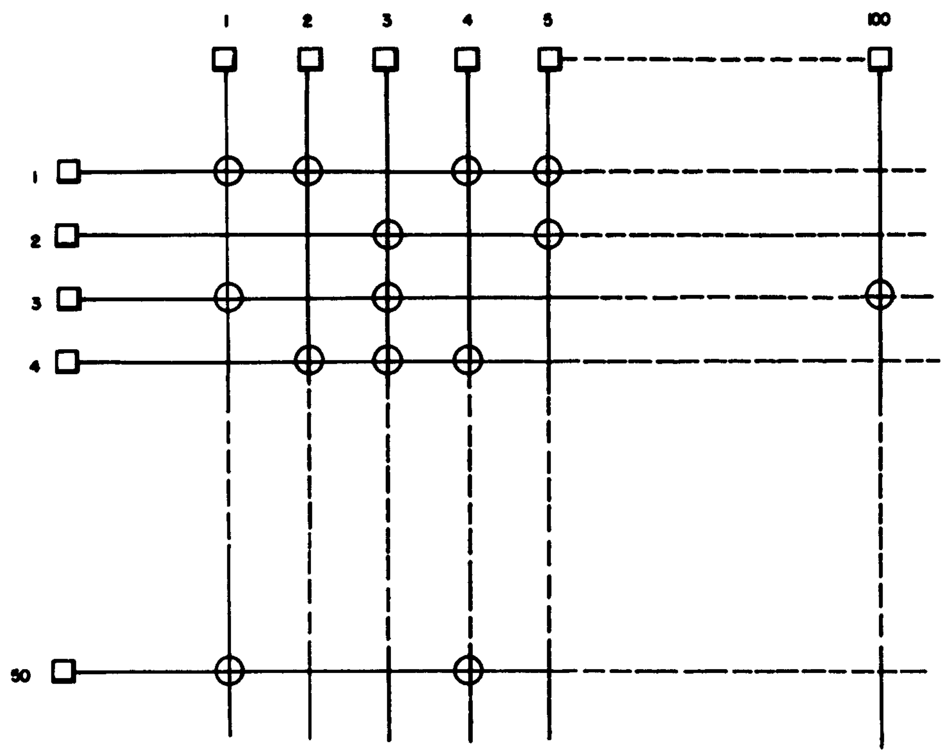

FIG. 4

A MAGNETIC CORE DECODING MATRIX

A total of three such matrices are required for the (100,50) code. The contents of the 50 flip-flops in the i^{th} matrix represent $[e_i]$, and if one of the three $[e_i]$ corresponds to a burst of length b or less, that $[e_i]$ is added to the appropriate 50 digits of the 100-digit received sequence. Otherwise, the sequence is rejected and a repeat requested.

The matrix requires only two wires through each core instead of the usual three and in this respect is easier to construct than standard core matrices. On the other hand, the fact that genuine cores are placed only at fairly random intersections may make the matrix somewhat more difficult to construct.

C. Construction of the Code

If [P] is the coding matrix of a (100,50) group code, it is required for the burst-error correcting procedure that the following sets of columns of [P] be linearly independent:

columns 1 to 50;

columns 26 to 75;

columns 51 to 100.

One of these sets may be chosen to correspond to the parity check digit positions and is thus a unit diagonal matrix. The remaining 50 columns, which are designated as [M], may be chosen in a random manner, subject to the restriction that the appropriate sets of columns of [P] be linearly independent.

One method of finding a suitable matrix [M] is to select matrices at random until one is found which satisfies the linear independence conditions. Approximately one matrix in twelve is suitable for codes of rate 1/2 (see p. 31 of ref. 1).

A second method of finding a suitable code is to begin with a diagonal matrix and then form random combinations of rows. This method has the advantage that the linear independence requirement is automatically satisfied without an indefinite number of successive trials, but the disadvantage that the coding matrix is not constructed in one whole unit, but must be built up from the random matrices with inverses which have been constructed.

The first method is preferable given the present programs, because it is more compatible with the evaluation program and requires fewer manual operations.

C-1. First Method

Figure 5 shows the general block diagram for this procedure. The connectors a_6, a_7, etc., refer to more detailed diagrams given in Appendices B and D.

First, a random (100,50) code is generated by constructing the matrix [P] of random numbers except for a unit diagonalized 50 x 50 matrix in columns 26-75. Then, the linear independence of columns 1-50 is investigated (i=1) with the aid of the code evaluation program. (This program is described in Section D and Appendix D). If they are linearly dependent, a new code is generated. If they are linearly independent, columns 51-100 are tested for linear independence, (i=3). If these are also linearly independent, a suitable code has been found.

Immediately after a code has been found suitable, the form of the coding matrix is such that a single one appears in each of columns 51-100. Rows are then interchanged so that columns 51-100 form a unit diagonal matrix, thereby yielding the matrix [P]. By setting i = 1 and a = 0, as shown in figure 5, columns 1-50 can be diagonalized, forming the decoding matrix [Q_1]. The other decoding matrix, [Q_2], is obtained by setting i = 2 and a = 0. It is always possible to return to the coding matrix [P] (which is also decoding matrix [Q_3]) by setting i = 3 and a = 0.

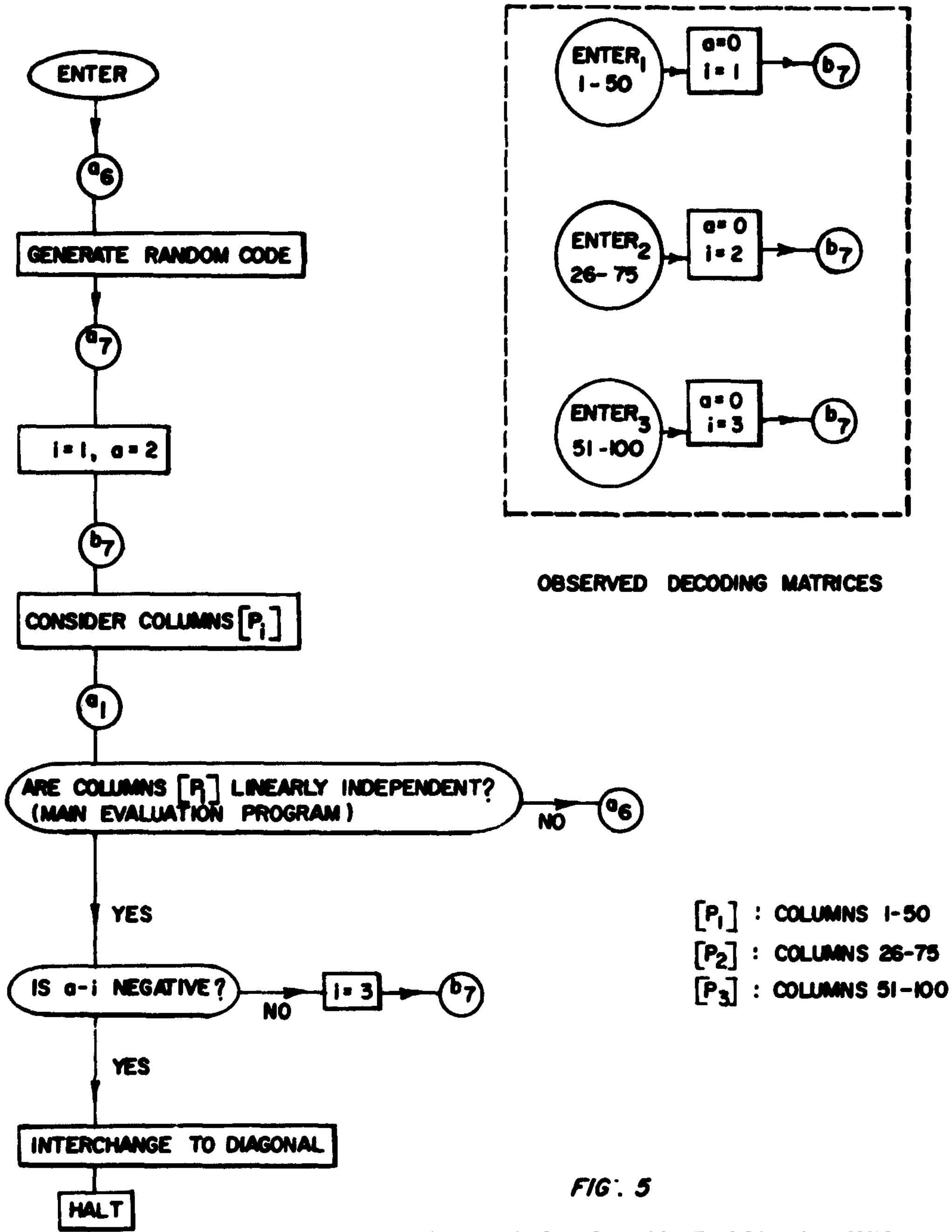

FIG. 5

FIRST METHOD OF CODE CONSTRUCTION

The first method was used to construct the code described in Section B. Details of a program utilizing the method are given in Appendix B.

C-2. Second Method

The procedure used here for finding a suitable code is to begin with a diagonal matrix and then form random combinations of rows as follows. Replace the first row by the sum of the original first row with a random selection of the other rows. Then replace the second row by the sum of the second row with a random selection of the other rows, using the new first row if this row should be included in the sum. Continue until all rows have been changed. The resulting matrix has an inverse.

In order to use the above procedure, the unit diagonal matrix was chosen to occupy positions 26-75, and the matrix [M] was partitioned into 25 x 25 submatrices as follows:

$$[M] = \left[\begin{array}{c|c} R_1 M_1 & M_2 \\ \hline M_1 & R_2 M_2 \end{array}\right] , \qquad (8)$$

where the matrices $[M_1]$ and $[M_2]$ are chosen in a random manner, but subject to the restriction that they have inverses. The matrices $[R_1]$ and $[R_2]$ can be selected entirely at random.

The matrices $[P_1]$, $[P_2]$, and $[P_3]$ are then

$$[P_1] = \left[\begin{array}{c|c} R_1 M_1 & I_{25} \\ \hline M_1 & 0 \end{array}\right] \quad ; \tag{9}$$

$$[P_2] = \left[\begin{array}{c|c} I_{25} & 0 \\ \hline 0 & I_{25} \end{array}\right] \quad ; \tag{10}$$

$$[P_3] = \left[\begin{array}{c|c} 0 & M_2 \\ \hline I_{25} & R_2 M_2 \end{array}\right] \quad . \tag{11}$$

The submatrix I_{25} is a 25 x 25 unit diagonal matrix. The inverses of these matrices are

$$[P_1]^{-1} = \left[\begin{array}{c|c} 0 & M_1^{-1} \\ \hline I_{25} & R_1 \end{array}\right] \quad ; \tag{12}$$

$$[P_2]^{-1} = [P_2] \quad ; \tag{13}$$

$$[P_3]^{-1} = \left[\begin{array}{c|c} R_2 & I_{25} \\ \hline M_2^{-1} & 0 \end{array}\right] \tag{14}$$

The three decoding matrices are then

$$[Q_1] = [P_1]^{-1}\,[P] = \left[\begin{array}{c|c|c|c} I_{25} & 0 & M_1^{-1} & M_1^{-1} R_2 M_2 \\ \hline 0 & I_{25} & R_1 & [I + R_1 R_2] M_2 \end{array}\right]; \qquad (15)$$

$$[Q_2] = [P_2]^{-1}\,[P] = \left[\begin{array}{c|c|c|c} R_1 M_1 & I_{25} & 0 & M_2 \\ \hline M_1 & 0 & I_{25} & R_2 M_2 \end{array}\right]; \qquad (16)$$

$$[Q_3] = [P_3]^{-1}\,[P] = \left[\begin{array}{c|c|c|c} [R_2 R_1 + I] M_1 & R_2 & I_{25} & 0 \\ \hline M_2^{-1} R_1 M_1 & M_2^{-1} & 0 & I_{25} \end{array}\right]. \qquad (17)$$

One method of finding $[Q_1]$, $[Q_2]$, and $[Q_3]$ is to construct $[M_1]$, $[M_2]$, $[M_1]^{-1}$, $[M_2]^{-1}$, and the other various matrix products and sums involved in (12), (13), and (14). This can be done, and a program for performing all these operations is included in Appendix C. However, once $[Q_2]$ is known, it was found to be more practical to use the code evaluation program to find $[Q_1]$ and $[Q_3]$, as illustrated in figure 5.

The second method was used to construct the code shown in Appendix H. Further details about the program for carrying out this second method are given in Appendix C.

D. Evaluation of the Error-Correcting Capabilities of the Code

D-1. Object and General Approach

It is desired to find the largest value of b for which the code corrects uniquely all bursts of length b or less. For this purpose it is necessary and sufficient to show that each burst of length b or less is contained in a different coset of the group of code words: i.e., that each burst of length b or less is associated with a different parity check sequence.

There are two possible approaches. One is to generate each error burst and observe the decoder outputs. The other is to investigate the structure of the decoding matrix.

The first approach would be practical if a fast special-purpose decoder for the code were already available, but would require a prohibitive length of time for most general-purpose computers. For example, if $b = 19$, there are about 22 million bursts of length b or less. Since it would be optimistic to assume that the pb 250 could check more than one per second, the procedure would take at least 6000 hours of computing time. Moreover, each additional unit increase of b would approximately double the required time. The second approach is the one used here. A procedure is found which requires about 3 hours of computing time for $b = 21$.

D-2. Underlying Principle of the Evaluation Procedure

If the columns of a submatrix [S] consisting of certain columns of [P] are linearly independent, then no two bursts which together cover positions included entirely in [S] can have the same parity sequence. To prove that a code is capable of correcting uniquely all bursts of length b or less, it is sufficient to show that the following sets of columns, each containing 2b elements, are all linearly independent:

$[S_1]$: 1 to b, b + 1 to 2b

$[S_2]$: 1 to b, b + 2 to 2b + 1

.
.
.
.

$[S_{n-2b+1}]$: 1 to b, n - b + 1 to n

$[S_{n-2b+2}]$: 2 to b + 1, b + 2 to 2b + 1

.
.
.
.

$[S_{2n-4b+1}]$: 2 to b + 1, n - b + 1 to n

.
.
.
.

$\left[S_{\frac{(n-2b+2)(n-2b+1)}{2}}\right]$: n - 2b + 1 to n - b, n - b + 1 to n

These sets are sufficient because any two bursts must be both included by at least one of the sets.

D-3. The Method

The linear independence of the sets $[S_i]$ is investigated in the order presented above by adding rows of the [P] matrix to other rows of the [P] matrix. Such operations do not change the linear independence or dependence of any set of columns of the matrix.[5] The method used is described below with reference to the five parts of figure 6.

A Gauss-Jordan reduction type of operation[6] is employed to determine linear independence. Thus, the first column of the set is searched for a non-zero element (search for pivot - part 3). When one is found, the (pivot) row of [P] containing that element is added modulo two digit-by-digit to all rows whose first column has a non-zero element (perform pivot operation - part 4). Then, a search (search for pivot) is made for a non-zero element in the second column of the set (select new column - part 1), excluding from consideration the previous pivot row. This process is continued until either:

(a) all members of the set of columns are searched and a non-zero pivot is found for every column, in which case the set is linearly independent; or,

(b) a column is found with no acceptable pivot, in which case the set is linearly dependent.

If $[S_1]$ is found to be linearly independent, $[S_2]$ is next investigated. (Change set of columns - part 2). Since $[S_1]$ and $[S_2]$ are identical except in one column, it is only necessary to investigate one column of $[S_2]$ to determine its linear independence once $[S_1]$ has been found linearly independent. (Change one part of set - transfer to c_1). After $[S_{n-2b+1}]$ has been investigated, however, a large number of new columns must be investigated, so that it is most convenient

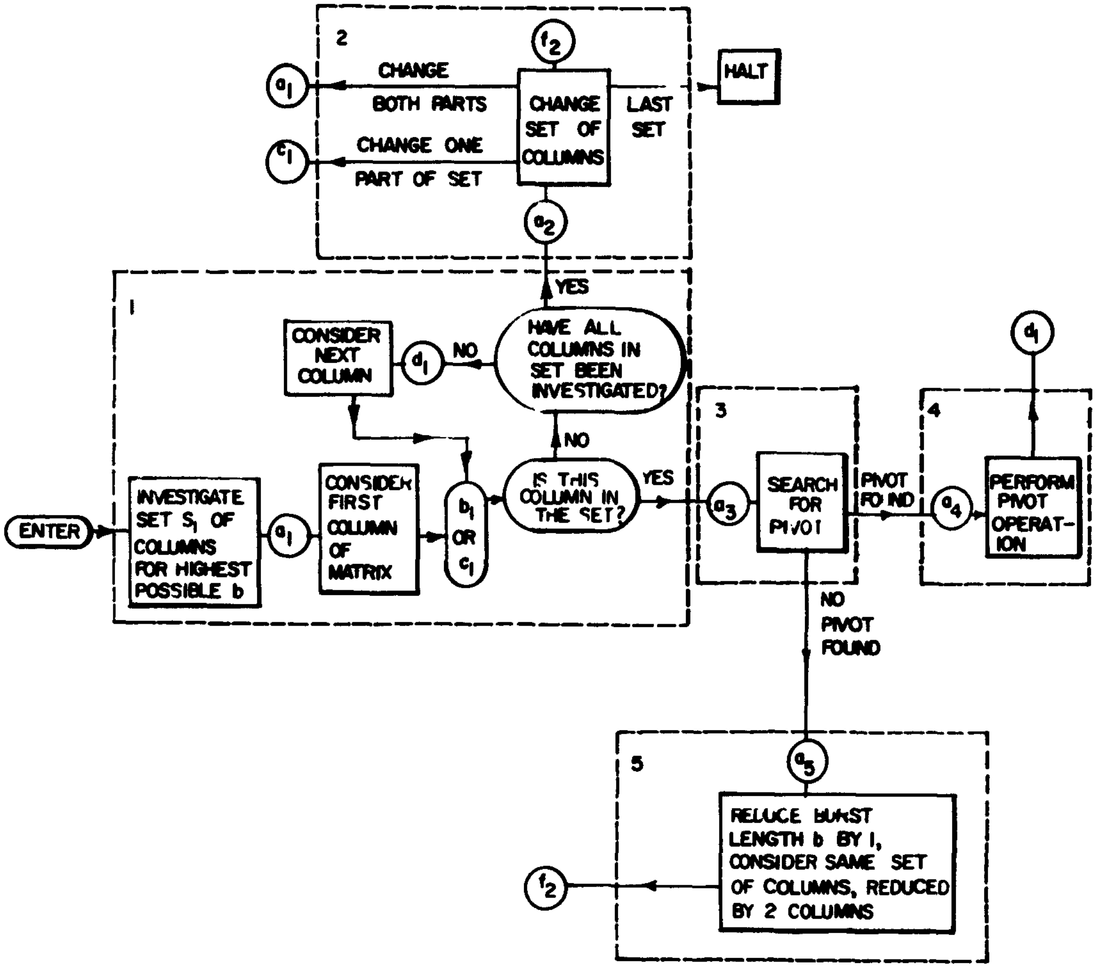

FIG. 6

MAIN EVALUATION PROGRAM - GENERAL DIAGRAM to investigate all columns of the new set, even though there is a slight duplication of effort. (Change both parts of set - transfer to a_1).

If some set $[S_i]$ is found to be linearly dependent, then not all bursts of length b or less can be corrected for the chosen value of b. The value of b is then reduced by one (reduce burst length - part 5), the number of columns per set by two, and the search is continued (transfer to f_2). It is not necessary to begin all over again when the burst length has been reduced, since all sets found linearly independent include sets for smaller values of b. If, for example, the set j to $b_o + j - 1$, k to $b_o + k - 1$ was the last to have been found linearly independent, the search can continue starting with j + 1 to $b_o + j - 1$, k + 1 to $b_o + k - 1$. Actually, for convenience, some duplication has been permitted, and the search continues starting with j + 1 to $b_o + j - 1$, $b_o + j$ to $2b_o + j - 2$.

The total number of submatrices to be investigated increases as the square of the code length. This property ensures that the computer does not have to compete with an exponential growth of computations with code length, as would be the case if each error burst had to be individually generated and checked.

Further details of the code evaluation procedure are given in Appendix D.

D-4. <u>Best Codes</u>

A program which permits both construction and evaluation is capable of finding a "best" code. For example, to find a code capable of correcting all bursts of length b or less, the computer could generate a code, test its capability to correct bursts of length b or less and, if it could not, generate another code. However, the pb 250 program is too slow (about 2-4 hours per code) to make this procedure practical if the fraction of codes correcting all bursts up to length b is very small.

E. Conclusions

Two programs have been described for the construction of codes suitable for the burst-error correction procedure described in reference (1). Also, a program for evaluating the burst-error correcting capabilities of the constructed code has been presented. The general logical procedures described are valid for any code length, n, and number of check digits, c. The specific pb 250 programs were written for $n = 100$ and $c = 50$, but with minor modifications they could be used for any $n \leq 105$ and $c \leq 63$. (These modifications are explained in Appendix G). Larger codes could also be handled if additional memory lines were available.

The first method of construction is preferable given the present programs, because it is more compatible with the evaluation program and requires fewer manual operations.

The evaluation program permits determination of the burst-error correction capabilities (maximum value of b) for <u>any</u> group code, not just the class under consideration.

APPENDIX A

Characteristics of the pb 250 and Conventions Used

The Appendices to follow contain various descriptions of the programs constructed. Some of the discussion is general, but other parts will be intelligible only to someone familiar with the Packard-Bell pb 250 computer, and are presented chiefly for reference value. However, a few basic pb 250 characteristics and conventions used are explained here as an aid to understanding the principal ideas.

The pb 250 stores data or commands in a number of lines, each containing 256 sectors numbered (octally) from 000 through 377 (400 octal = 256 decimal). The lines available on the particular machine used are numbered 00, 01, 02, 03, 04, 05, 06, and 07. Each command location stores 21 bits plus a sign bit. There are also three arithmetic registers, labelled A, B, and C.

The following conventions are employed in the pb 250 logic diagrams to follow:

(1) Parentheses signify "contents of".

(2) A long dash, as (A)———(B), implies contents of A replace contents of B.

(3) Three-figure numbers, as 002, signify octal sector numbers.

(4) Two-figure numbers, as 02, signify octal line numbers.

(5) The number at the upper right corner is the address of the first instruction in the box - sector number followed by line number.

(6) The words "(A) and (B)" signify the logical product of (A) and (B) as binary sequences. A one is placed in every position of "(A) and (B)", where both (A) and (B) contain a one. A zero is placed in all other positions.

(7) The words "(A) or (B)" signify the logical sum of (A) and (B). A one is placed in every position of "(A) or (B)" where either (A) or (B) or both contain a one. A zero is placed in all other positions.

(8) A connector a_i has a subscript i which indicates it is in the i^{th} section of the program.

APPENDIX B

Code Construction Program - First Method

1. Generation of a Random Code

The process of generating a random code consists of selecting n - c columns of the [P] matrix as random sequences of zeroes and ones, and of selecting a unit diagonal matrix for the c columns corresponding to the positions to be occupied by check digits. The key as to which columns are to contain the unit diagonal matrix is contained as a set of numbers in sector 277, lines 02, 03, 04, 05, and 06. The location of zeroes in these numbers corresponds to the c columns of the matrix [P](located in the first n columns of sectors 300-377, lines 02, 03, 04, 05, and 06), which contain the unit diagonal matrix. For the (100,50) code with check digits in position 26-75, the key numbers are, in octal form,

```
277 02$    +7777777
277 03$    +7400000
277 04$    +0000000
277 05$    +0000777
277 06$    +7777740
```

The columns corresponding to ones in the 277 sector numbers are filled with random numbers. The manner in which this is done is illustrated in Figures 7-8. Conventions used are as described in Appendix A.

Part 1 of the Random Code Generation Program (figure 7) consist of filling in the random numbers. Initially, some randomly-chosen positive 21-bit number is placed in (R). The first random number generated in the minor product of (R) with +7303425 (octal) which is the highest power of five capable of being placed in one location.[3] This minor product then replaces (R) for use in generating the second random number. The digit-by-digit logical product of the first random number with the contents of 277 02 are placed in 301 02 to form the first 21 bits in the first row of matrix [P]. Similarly, all the addresses from 301 through 362, line 02, are filled with random numbers. The program then moves to line 03, etc. The initial, or "key" random number R_o is stored in 240 05.

Part 2 of the Random Code Generation Program (shown in figure 8) is for the purpose of inserting the unit diagonal matrix. Beginning with +4000000 (octal) in (d), the contents of d are shifted and compared with the contents of sector 277 (first line 02, then line 03, etc.) until the single one in (d) corresponds in position to a zero in (277,x). Then the binary digit-by-digit logical sum of (d) with (301,x) is stored in (301,x). The contents of d are shifted once more and the new contents stored in (302,x) (unless, just previously, (d) = +0000001, in which case +4000000 would be stored in (302,x+01)). Each time (d) is stored, the sector address key S is incremented by one, but each time (d) has a one in a position corresponding to a one in (277,x) the key S is not incremented, and (d) is not stored. After line 06 has been passed, the construction is complete.

2. <u>Suitability of the Code</u>

The above discussion has described the details of the

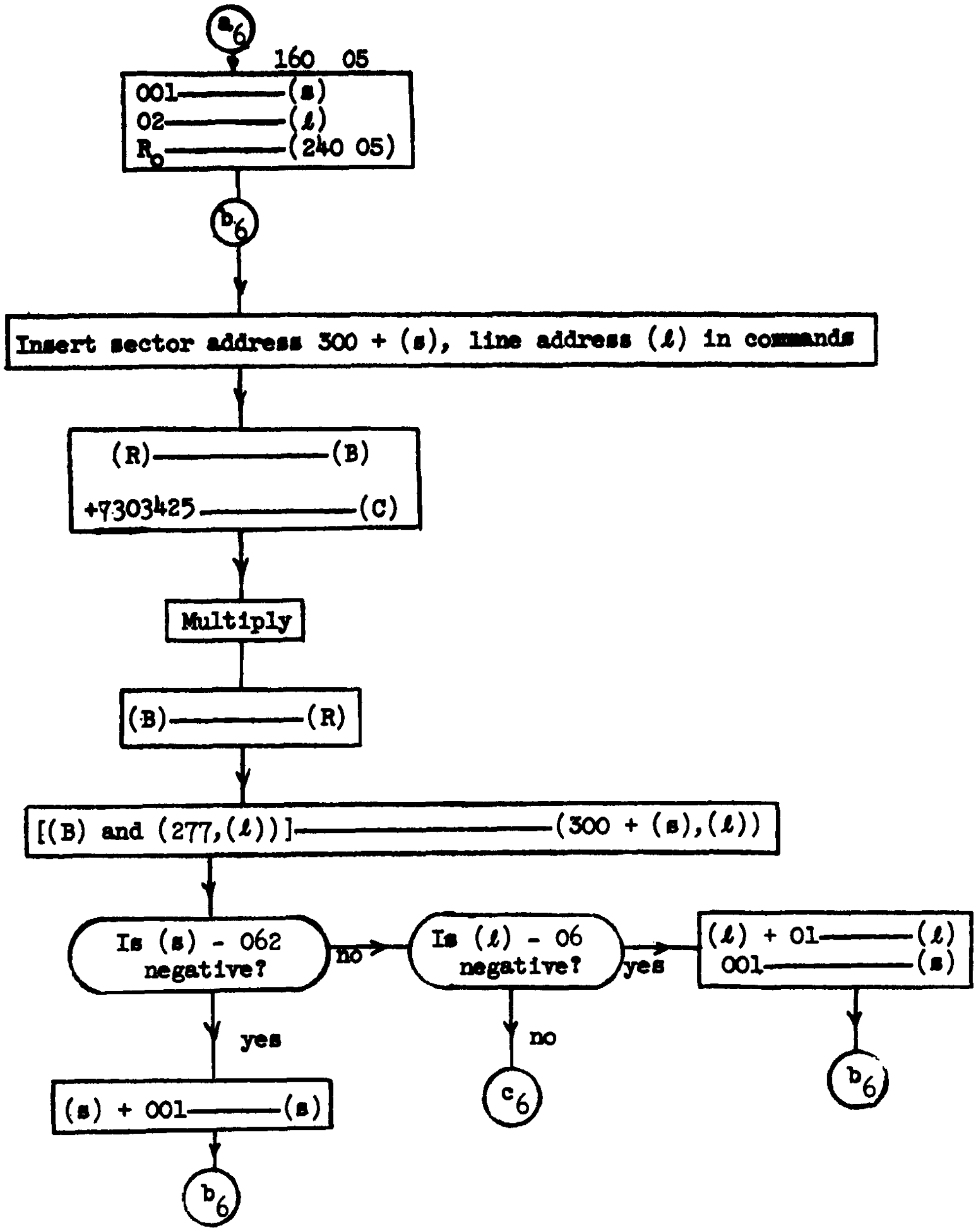

Figure 7 - Random Code Generation - Part 1

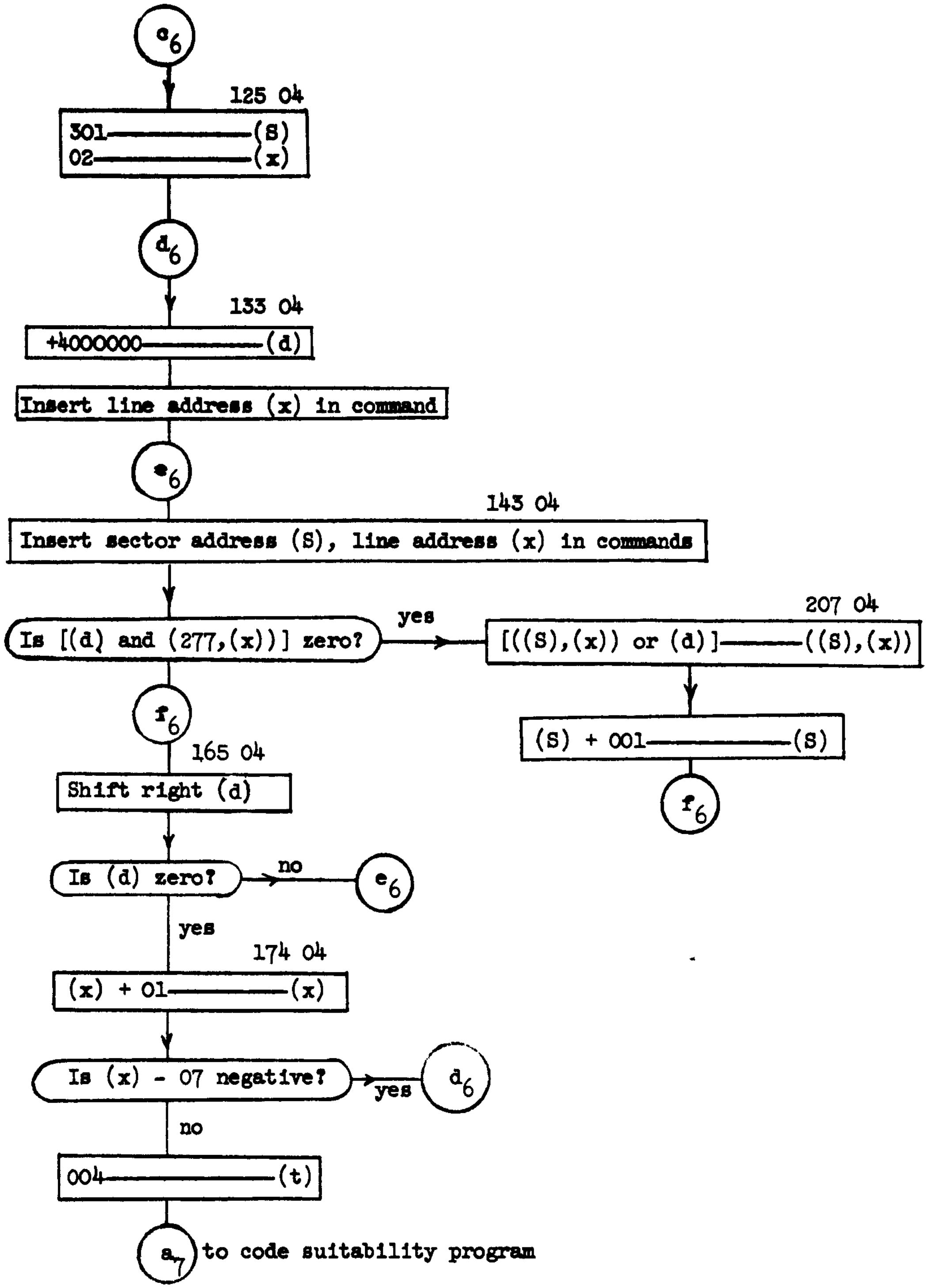

Figure 8 - Random Code Generation - Part 2

"Generate Random Code" box in figure 5, Section C.1. The following is a description of the remaining parts of figure 5.

A key is first set up to indicate for which sets of columns linear independence is to be searched. For the (100,50) code this is set up as follows.

sector ↓ / line →	02	03	04	05	06
267	-7777777	-7400000	+0000000	+0000000	+0000000
270	+0000000	+0377777	-7760000	+0000000	+0000000
271	+0000000	+0000000	+0017777	-7777000	+0000000
272	+0000000	+0000000	+0000000	+0000777	-7777740

For other codes, sectors 273, 274, 275, and 276, are also available to permit linear independence search over more sets of matrices. In the present case, keys 267 and 270 are used to search for linear independence of columns 1-50, while 271 and 272 are used to search columns 51-100. In the program, the keys to be used are expressed as 267 +(p), 270 +(p).

Figure 9 shows the logic diagram. Initially, p = 0 is set. Most of the logic shown is for the purpose of adapting the main evaluation program to the present requirements. The number +7000000 is placed in a key position in the main program to cause it to branch differently from its normal procedure. (The dashed lines indicate the paths normally taken by the main program. See Section D and Appendix D for a description of the main program).

The section beginning with 014 04 places the contents of 267 +(p) and 270 +(p), lines 02 through 06, in that part of the main program which determines the set of columns to be searched for linear independence (quantities X and Y in the main program). In order that the code be suitable it is necessary that columns 1-50 be linearly independent and that columns 51-100 be linearly independent. (Columns 26-75 are already linearly independent by construction). Thus, the search is first run through for p = 0, then for p = 2. If the set for p = 0 is found linearly dependent, a new code is generated.

The time required to generate a code is about 15 seconds. It requires 3 or 5 minutes to determine if the code is suitable. The time required to find a suitable code is variable, usually between 1/2 and 1 hour for a (100,50) code.

The coding matrix obtained in the above manner appears in a mixed form. To obtain a usable form, it is necessary to interchange rows such that the i^{th} row represents the i^{th} check rule of the code. This is done by the procedure illustrated in figure 10.

Again, the main evaluation program is used for the bulk of the operations. A key of +7700000 is placed in (Y_o,02) to obtain proper branching. Then, the main evaluation program is entered. Suppose that columns 51-100 are the ones under consideration. These columns already each contain exactly a single one, but the one is generally not in the desired position. The first column considered is column 51, and (i) is set equal to 001. The search for pivot (see Appendix D) discovers a one in row j of column 51. The key then causes the program to branch to 325 07 instead of proceeding to "perform pivot operation". The portion of the program following 325 07 interchanges row i and row j of the matrix. Column 51 now contains a one only in row one. The number (i) is then increased by 001, and the main program re-entered at d_1, where the column under consideration is shifted to 52.

After fifty interchanges have been made, (i) - 063 (octal) will be zero, and the program will halt. A unit diagonal matrix now appears in columns 51-100. To observe the decoding matrix with columns 1-50 diagonalized, the program is entered at a_9, where (p) and (x) are both set equal to 000 (see figure 9). The code suitability program is then run through for just (p) = 000, yielding single ones in each of columns 1-50. The rows are then interchanged by the "Interchange to diagonal" program. Similarly, to observe the 26-75 diagonalized decoding matrix, a_{10} is the entry, and (p) = 001 is used. Entry at a_{11} will return to the 51-100 diagonalized matrix.

Once a suitable code has been found, it can be regenerated at any time by inserting the key number for that code in 201 05 and performing the random code generation program.

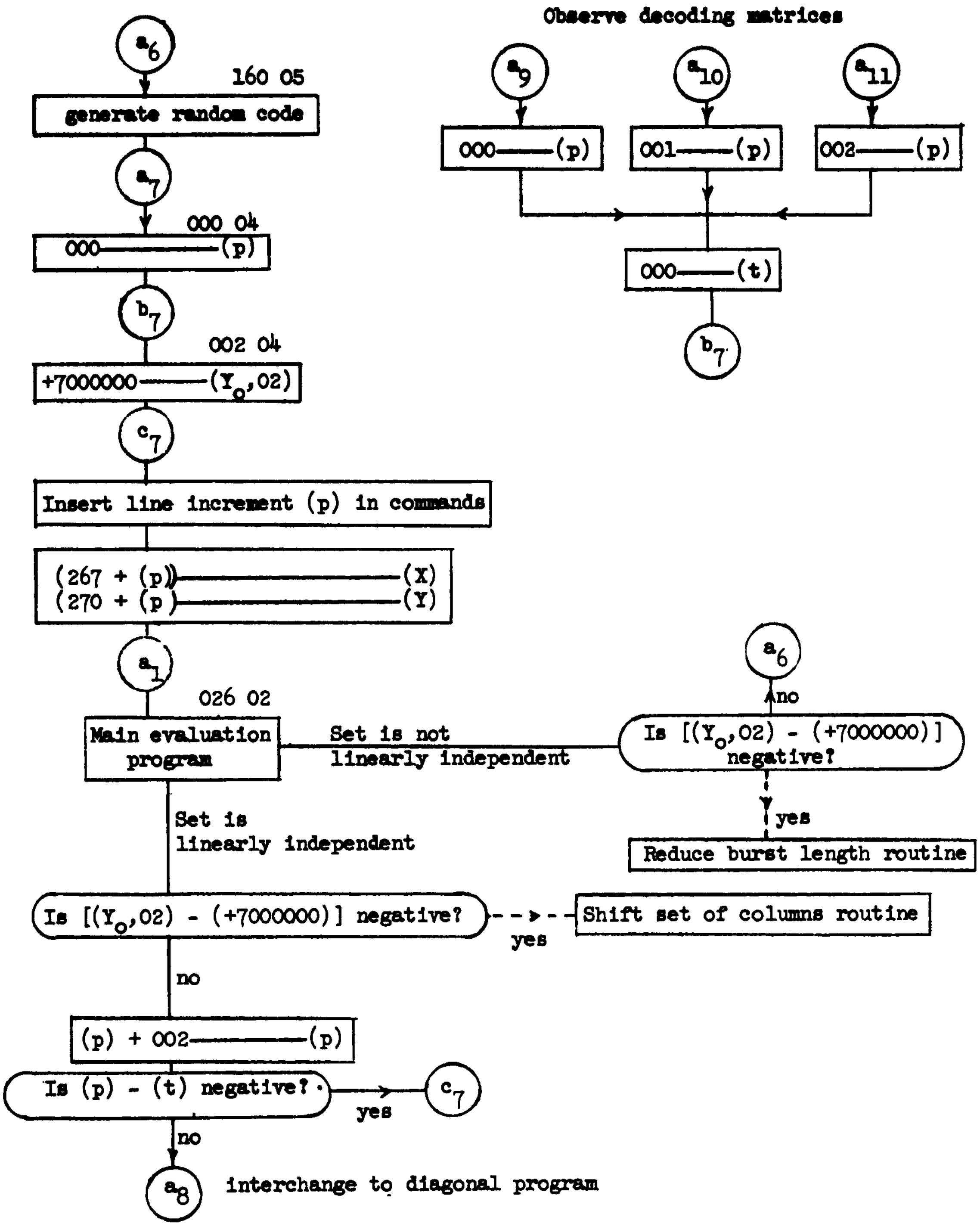

Figure 9 - Determination of Code Suitability and Observation of Decoding Matrices

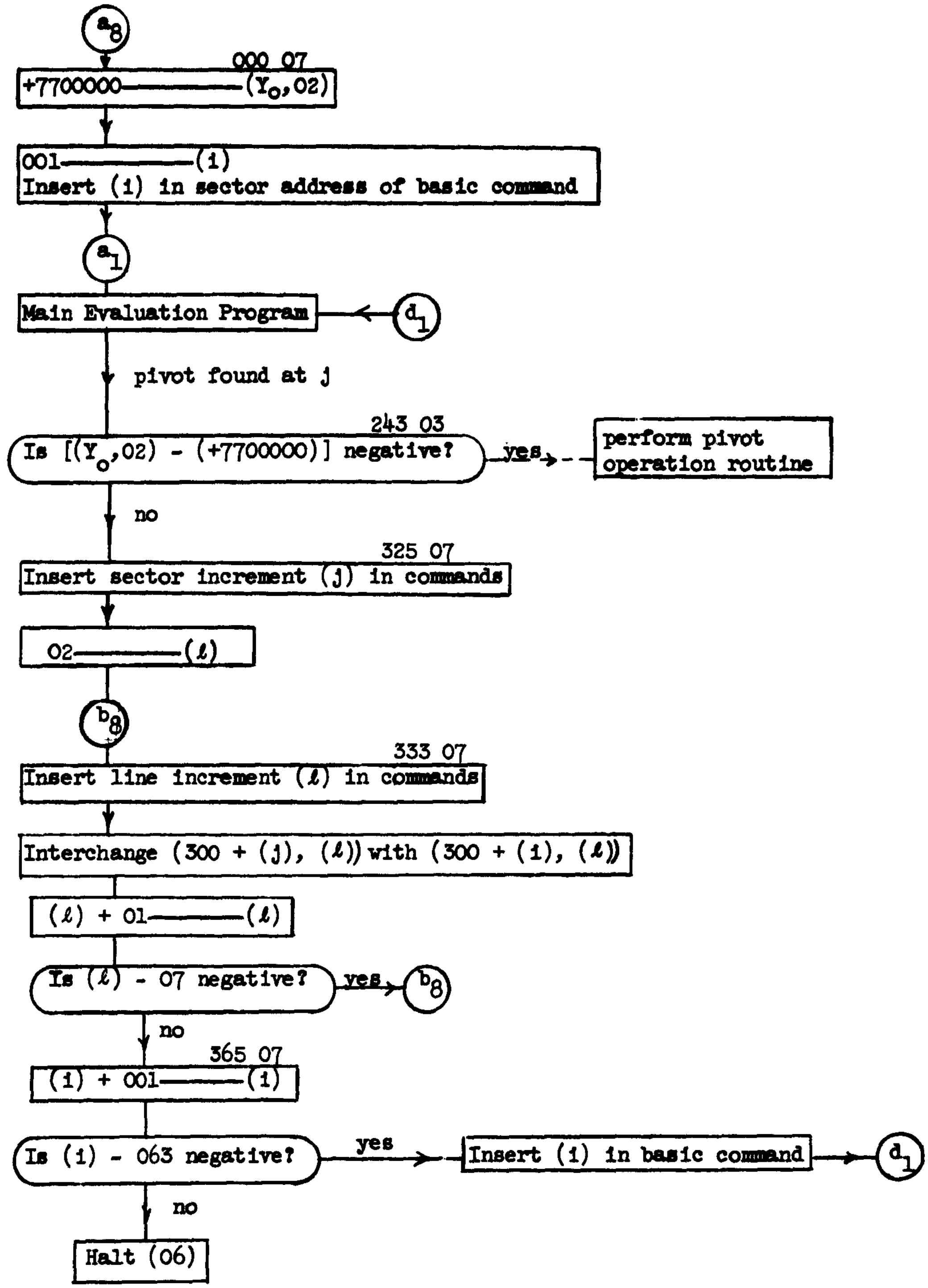

Figure 10 - Interchange to Diagonal Program

APPENDIX C

Code Construction Program - Second Method

1. Theory

The matrix $[M_1]$ may be considered as defined by the product of matrices

$$[M_1] = [T_{25}][T_{24}] \dots [T_1] \quad ,$$

where

$$[T_i] = \begin{bmatrix} 1 & \ddots & & & & 0 & & \\ & 0 & & 1 & & & & \\ d_{i,1} & \cdots & & d_{i,i-1} & 1 & d_{i,i+1} & \cdots & d_{i,25} \\ & & & & & 1 & \ddots & 0 \\ & & 0 & & & & \ddots & 1 \end{bmatrix}$$

and the d_{ij}, for all j except $j = i$, are chosen at random. The matrix $[T_i]$ represents the operation of replacing the i^{th} row by the modulo two sum of the i^{th} row with a random selection of the other rows. It can readily be shown by taking the product that

$$[T_i][T_i] = [I] \tag{18}$$

From (18), it follows that the inverse of M_1 is given by

$$[M_1]^{-1} = [T_1][T_2] \dots [T_{25}] \quad , \tag{19}$$

since

$$\{[T_1][T_2] \dots [T_{25}]\}\{[T_{25}] \dots [T_2][T_1]\} = [I] \quad . \tag{20}$$

Thus, if a matrix is formed by using a set of row combination rules as above, its inverse is found by performing the combinations in reverse order. This allows the use of practically the same computer program to obtain both a matrix and its inverse.

The product of two matrices can also be found with only a slight change in the basic program. The i^{th} row of the product [A][B] is the modulo two digit-by-digit sum of those rows of [B] such that the j^{th} row is included in the sum if and only if digit a_{ij} of [A] is a one. This operation is very similar to that performed in constructing $[M_1]$. If [A] replaces the row combination rules, [B] replaces the diagonal matrix, and if the sum of rows is stored in a new location instead of replacing the old row of [B], the program used for constructing $[M_1]$ can be used to find the product [A][B].

2. Description of the pb 250 Program

Although only 25 x 25 matrices are being constructed, the program permits construction, inversion, and products of matrices of any size up to 63 x 63 without modification. The key which tells the program when to stop is a negative number placed after the last combination rule. The combination rules were stored in sectors 201 through 231, lines 02 and 03, the first 21 digits of a row being stored in line 02, and the remaining 4 in line 03 (with a + sign). If a larger matrix were used, sectors 201-276 are available, and each combination could be stored in lines 02, 03, and 04, making up to 63 (decimal) dimensions possible.

Figure 11 shows the basic outline of the matrix construction program. Random numbers are generated to form the combination rules. To find a matrix and its inverse, a unit diagonal matrix is initially stored in sectors 301 through 377, lines 02, 03, and 04. The resultant matrix or inverse will appear in these locations upon completion of the program. The program is divided for convenience into five separate parts, which are individually described in figures 12-17. Conventions are as described in Appendix A.

Part 1 (Figure 12)

This section describes the main program. The three boxes indicated by asterisks are modified according to whether the purpose is to construct a matrix, find its inverse, or find the product of two matrices. An index (i) specifies the sector address of the combination rule (or row of matrix A in the product AB), while (j) corresponds to the j^{th} digit of the combination rule. If the j^{th} digit is a one, row j of the matrix stored in 301-377, lines 02-04, is added digit-by-digit modulo two to the basic row i of that matrix, or of the product matrix in the case where the product is taken.

The program initially picks up line 02 of the combination rule corresponding to (i_o). This is tested to see if it is negative, since a negative number indicates that all combination rules have already been considered. If it is positive, the rule is stored in (T), and the number 25 (octal) = 21 (decimal) is stored in (s) to specify the number of digits of this part of the combination rule which are yet to be considered. The number (T) is then placed in register (B) and shifted left,

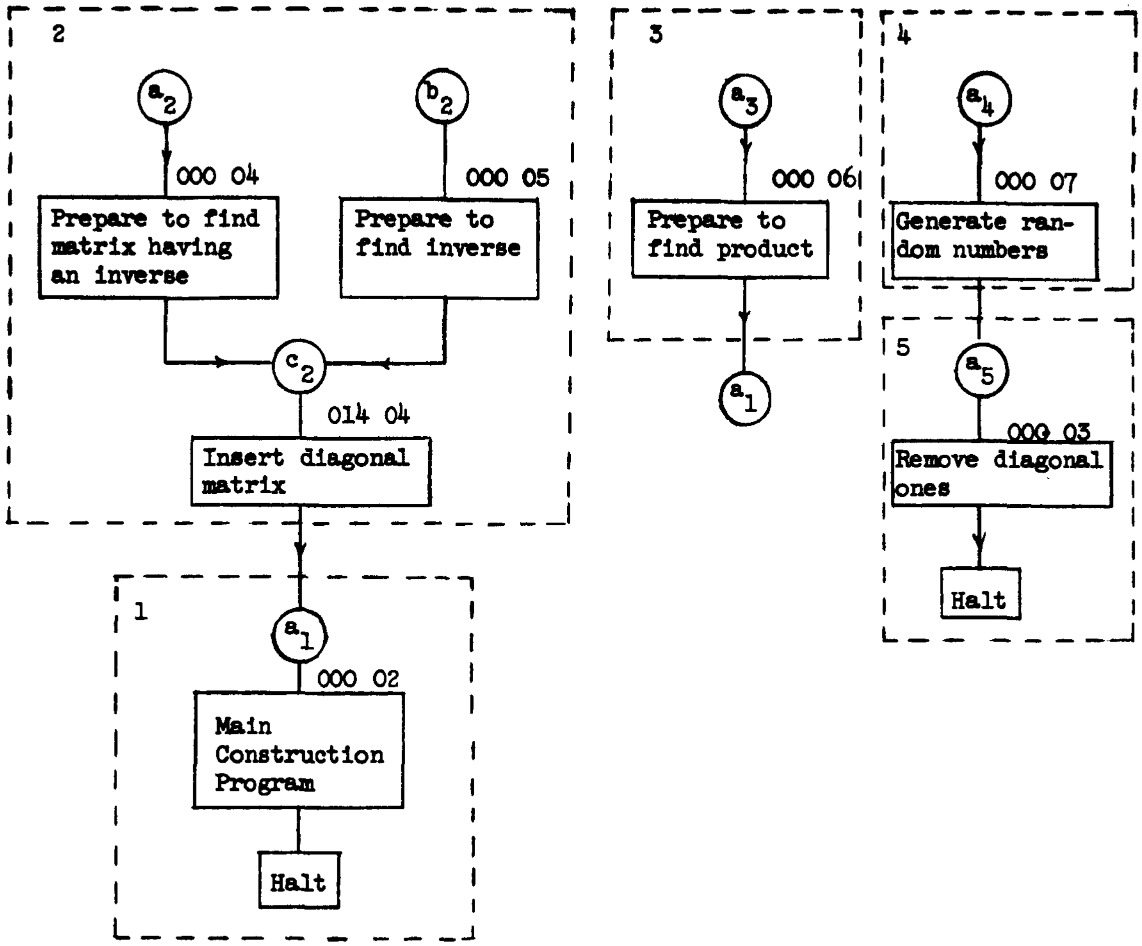

Figure 11 - Second Method of Code Construction

while simultaneously (s), which has been placed in register (C), is decremented by one. The number (C) is then tested for sign. If it were negative, this would indicate that all 21 digits of the combination rule had been considered. In this case, however, the number is positive. Then, (B) is tested for sign. If the first digit of the combination rule were a one, this digit would be shifted into the sign position of (B), yielding a negative number. If the first digit were a zero, the shift would yield a positive (B). Thus, if (B) is negative, row j is added to row i to form a new row i. (This never happens for i = j because the combination rules are generated such that the i^{th} digit of rule i is always a zero.) Then, the program proceeds to g_1 and j is incremented by one. If (B) is positive, the program proceeds directly to g_1. The number (j) continues to be incremented once per loop and (s) decremented until (s) is negative.

Then, the line address increment (q) of the combination rule is incremented by one. If the resulting (q) -03 is then non-negative, the entire combination rule has been considered, in which case a new rule is selected. This is done by incrementing (i) by 001 for matrix construction or product formation, or by decrementing (i) by 001 if the inverse is being sought.

In performing the addition of rows i and j, basic sector numbers (S_0) and (ℓ_0) are inserted. For matrix construction or inversion, $(S_0) = 300$ and $(\ell_0) = 02$, while for product formation $(S_0) = 200$ and $(\ell_0) = 05$.

Part 2 (Figures 13-14)

To construct a matrix given the combination rules, the program is started at 000 04. The section from a_2 to c_2 modifies the main program for the reasons explained in Part 1. The program following c_2 inserts a diagonal matrix in sectors 301-377, lines 02-04. The section between c_2 and f_2 clears to zero the contents of sectors 301-377, lines 02-04. Ones are then inserted along the diagonal in the section following f_2. First, the number (d) = +4000000 is stored in (301 02). The number (d) is then successively shifted right one unit at a time, each time storing the new number in the next successive sector.number location of line 02, until (d) is shifted to zero, in which case (d) = +4000000 is stored in line 03 of the next sector number. This continues until lines 02, 03, and 04 have all been completed. The program then proceeds to a_1.

To find the inverse, the program is started at 000 05. The section from b_2 to c_2 modifies the main program for reasons explained in Part 1. The remainder of the program is then the same as for constructing a matrix.

Part 3 (Figure 15)

To find the product AB of matrices A and B, the program is started at 000 06. Previously, matrix A is stored in sectors 201-277, lines 02-04, and B is stored in sectors 301-377, lines 02-04. The product appears in sectors 200-277, lines 05-07. The section from b_3 to a_1 clears to zero the contents of sectors 200-277, lines 05-07. The main program is then entered.

Part 4 (Figure 16)

This section generates the random combination rules needed to construct a matrix. This procedure is much the same as the one described in Appendix B (Figure 7 and associated explanation). A key number in sector 177, lines 02, 03, and 04, tells the program where random numbers should be placed. For a 25 x 25 matrix, the key is:

177 02$ +7777777

177 03$ +7400000

177 04$ +0000000

The columns corresponding to ones in the 177 sector numbers are filled with random numbers. The other columns are filled with zeros. After the random numbers are generated, the program proceeds to a_5.

Part 5 (Figure 17)

It is necessary to remove the diagonal ones from the set of random numbers in order to ensure that the i^{th} digit of combination rule i is always a zero. Initially, (W) = -3777777 is set, and the contents of 201 02 are replaced by the digit-by-digit logical product of (W) with the previous contents of 201 02. The number (W) is then successively shifted right one unit at a time (the nimus sign causes ones to be introduced from the left), each time taking the logical product of (W) with the contents of the next successive sector number location of line 02, until (W) becomes equal to -7777777, in which case the line address is incremented to 03, and (W) is returned to -3777777. This continues until all sectors through 277 have been included. The program then halts.

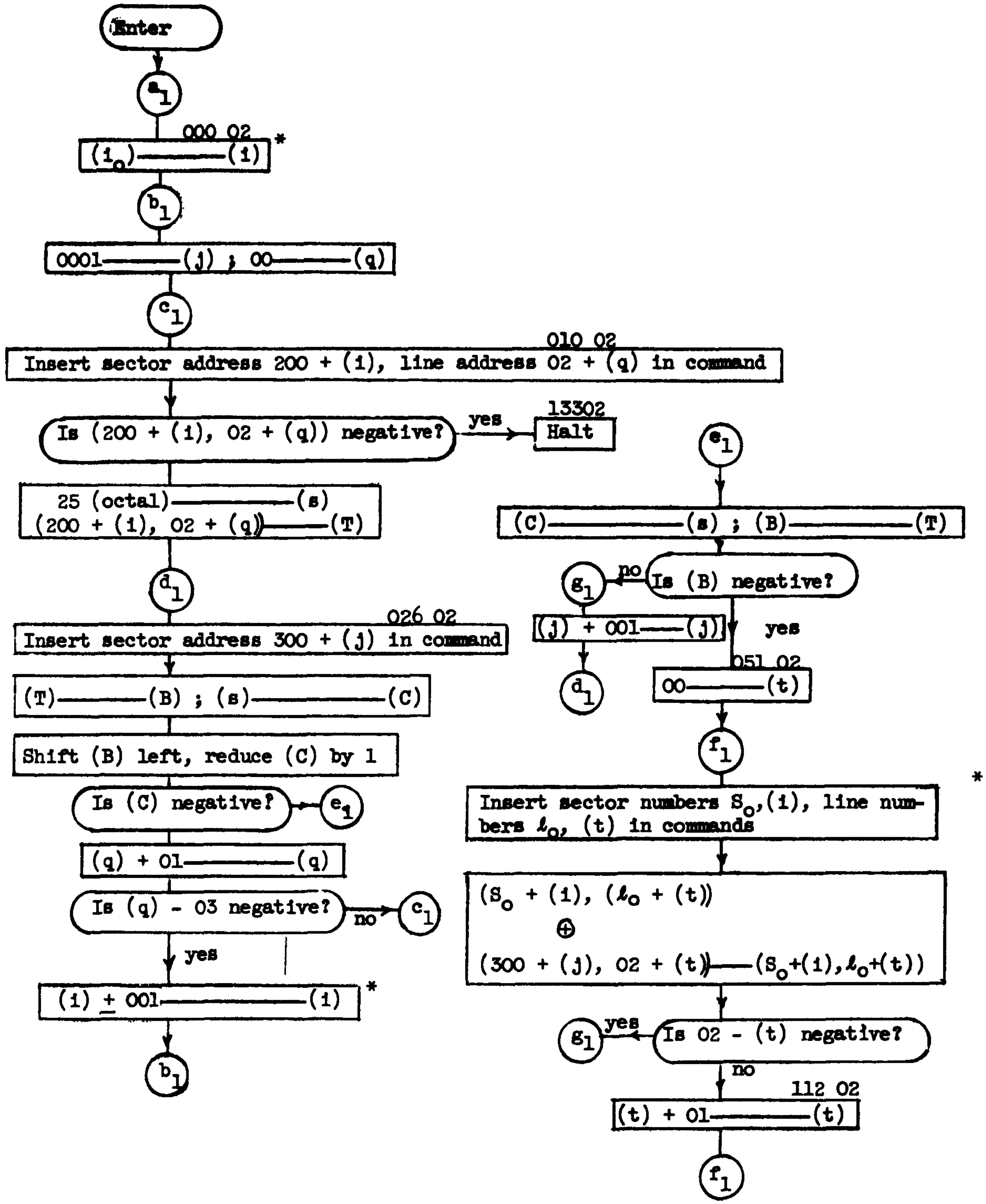

Figure 12 - Second Method of Code Construction - Part 1 -
Main Program

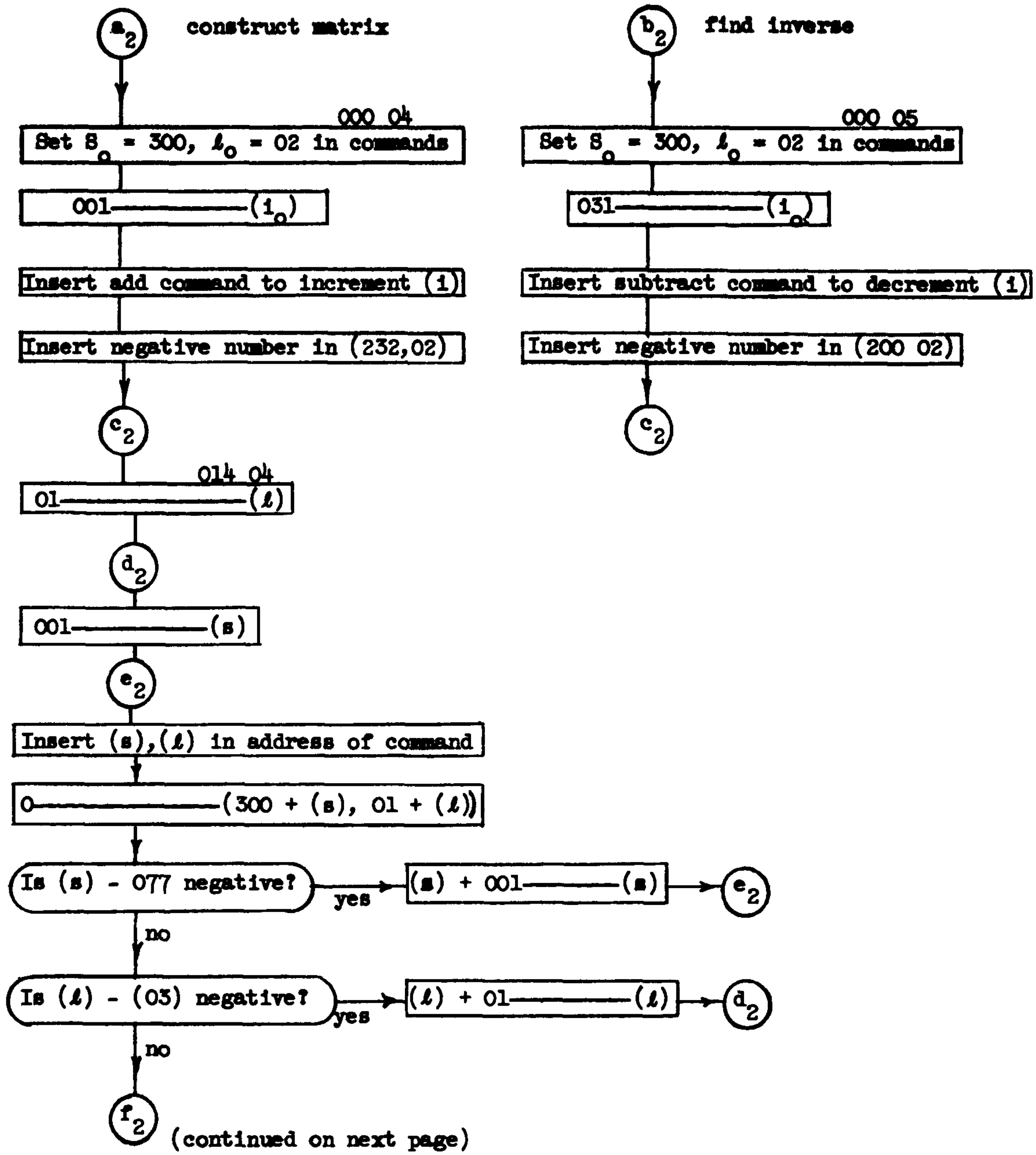

Figure 13 - Second Method of Code Construction - Part 2a -
Construction of a Matrix and its Inverse

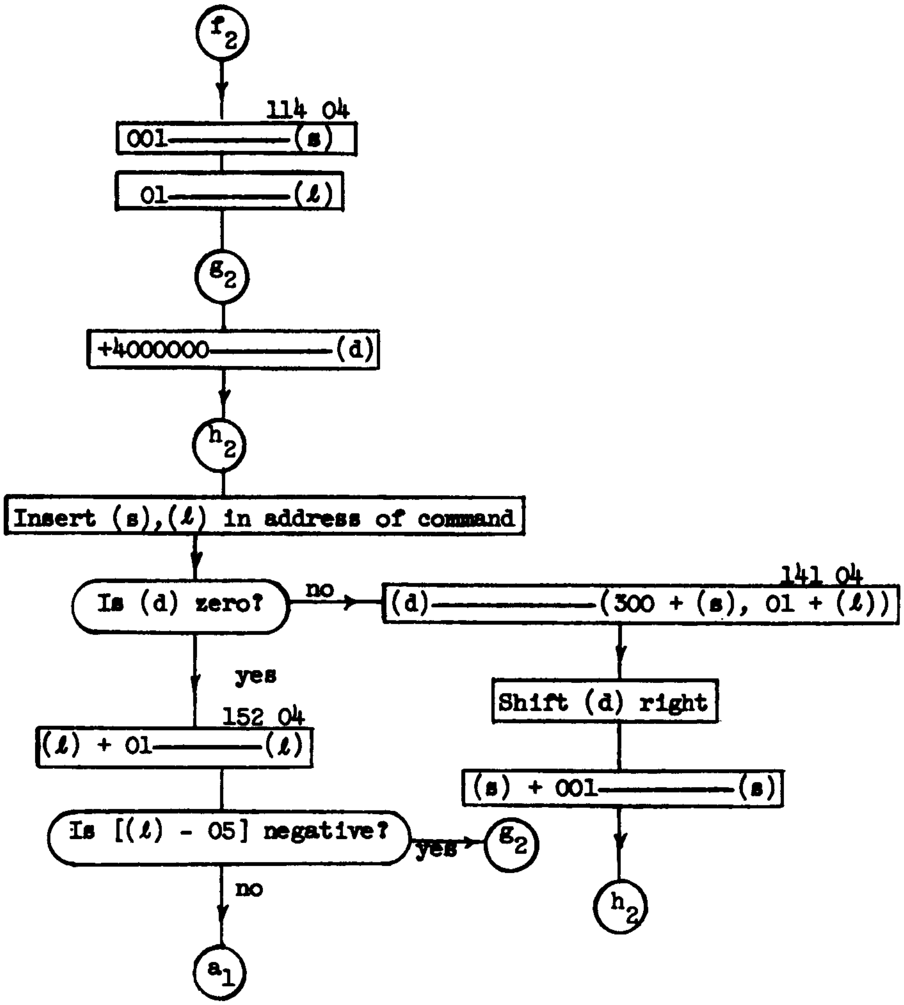

Figure 14 - Second Method of Code Construction - Part 2b - Construction of a Matrix and Its Inverse

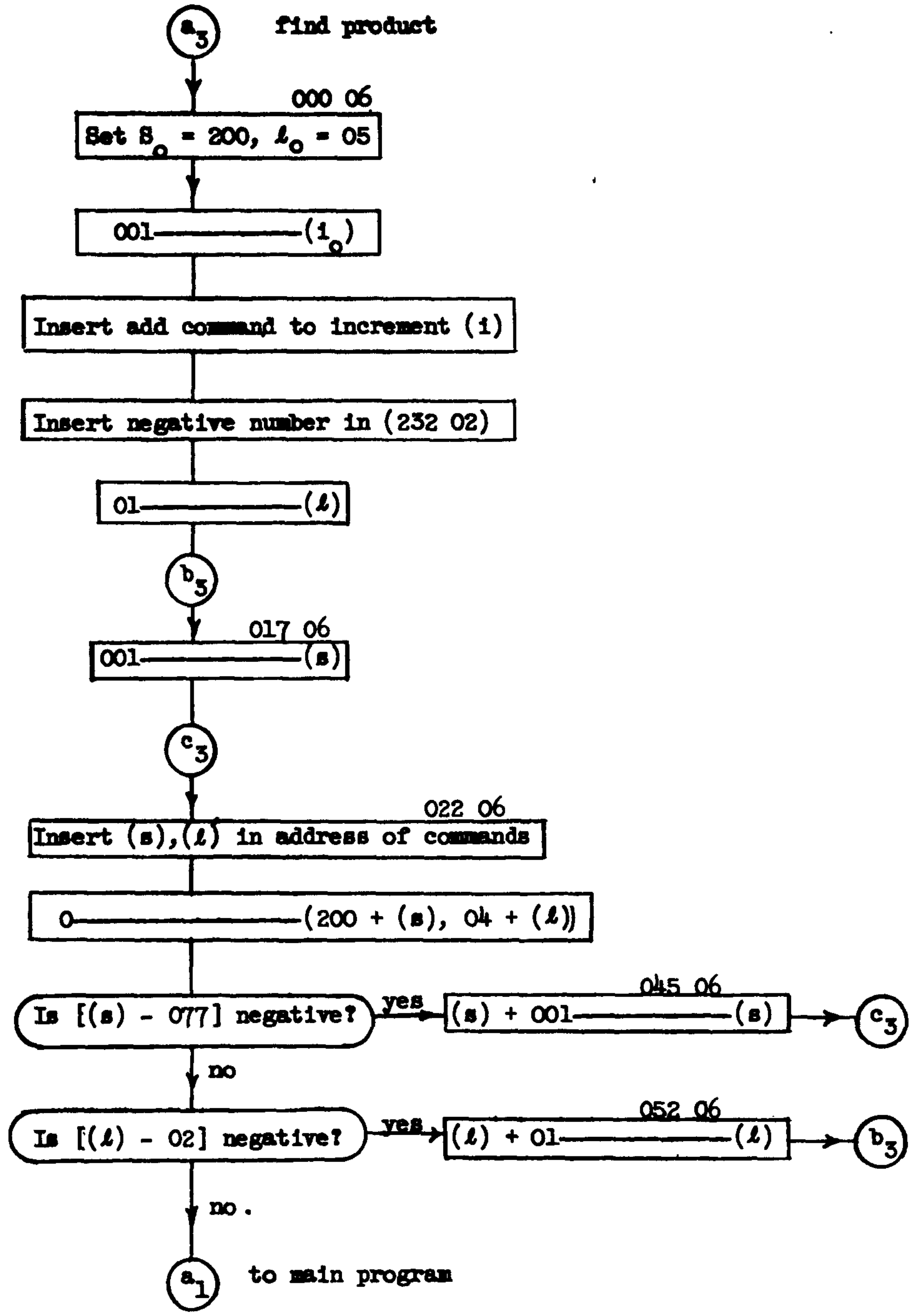

Figure 15 - Second Method of Code Construction - Part 3 -
Finding the Product of Two Matrices

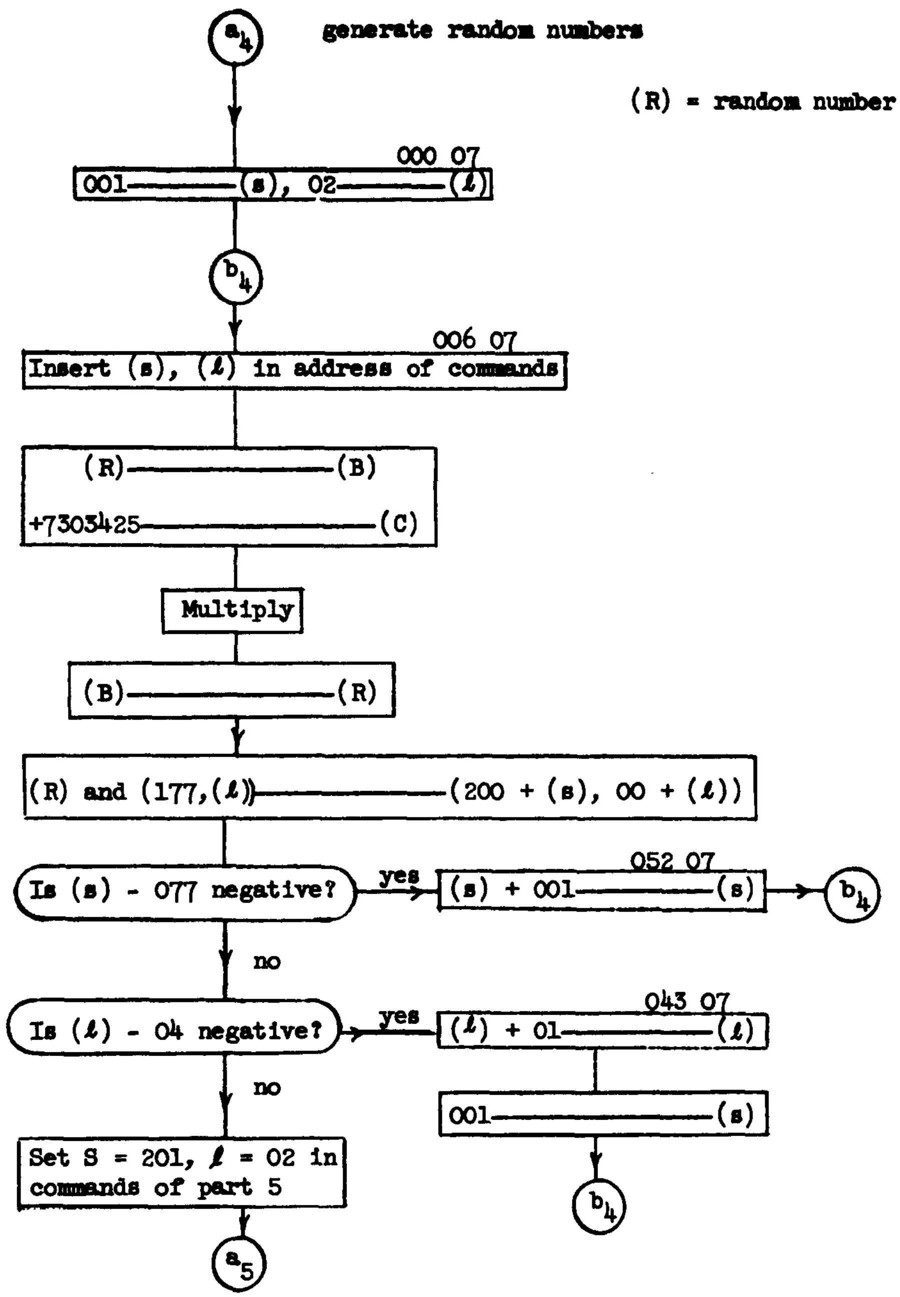

Figure 16 - Second Method of Code Construction - Part 4 -
Generation of Random Numbers

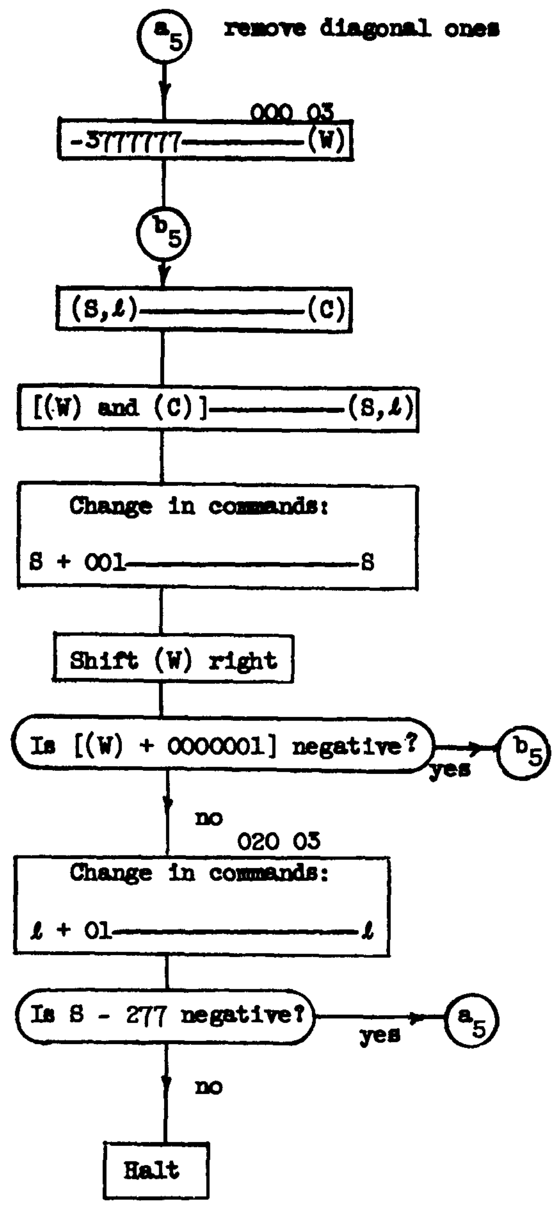

Figure 17 - Second Method of Code Construction - Part 5 - Removal of Diagonal Ones

APPENDIX D

Details of the Code Evaluation Program

Figures 18-22 show the logic of the various parts of the evaluation program in greater detail. The five parts are the same as referred to in figure 7. The notation used is similar to that described in Appendix A with an additional special convention. Since some quantities, as Y and X, represent contents stored in more than one location (the same sector, but different lines, the notation (Y, 02) will represent the contents of line 02 of the sector address of Y, while the notation (Y) will represent the contents of all lines in which Y is stored.

The 50 x 100 matrix [P] is stored in sectors 301-362, lines 02 through 06. The addresses in lines 02, 03, 04, and 05 each contain 21 bits of a row of the matrix, while only 16 bits from the addresses in line 06 are included. (Since 4 x 21 + 16 = 100).

The operation of the program is best understood by considering the various parts and their purposes.

Part 1 (Figure (18)

The purpose of Part 1 is to select the column for which a pivot search is to be carried out. Before this can be done, the set of columns currently being searched for linear independence must be specified. This set is defined by the quantities X and Y. The quantity X is a 100-digit number stored in lines 02, 03, 04, 05, and part of 06, sector 260, and consists of zeros except for a sequence of b successive ones. The quantity Y is another 100-digit number, stored in lines 02, 03, 04, 05, and 06, sector 261, and also has ones only in b successive positions, but none in position corresponding to ones in X. A given pair X, Y define some set [S_1] of columns of the type described in Section D-3. The part of the program beginning at 000 02 sets X, Y, and Y_0 (which will be discussed later) to the initial values corresponding to [S_1] and the value of b.

A number is required to keep a record of the column under consideration. This is provided by (P), together with line address (x). The binary number (P) contains but a single one, that being in the position corresponding to the column under consideration, and (x) gives the line address of the column under consideration. A convenient related quantity is (P_s), which is a 100-digit number stored in lines 02, 03, 04, 05, and 06 of sector 262. This number is to contain a one in all those positions up to, and including, the column under consideration.

Following a_1, (P) and (P_s) are set to their initial values, which corresponds to consideration of the first column. To determine if the column under consideration is in the set, it is necessary only to compare (P) with line x of X and of Y. If it is, a search for pivot is made on that column (proceed to a_s). If not, there are two possibilities: either the set to be investigated has not yet been reached, or it has been passed (in which case all columns have been investigated. These alternatives are separated by determining whether (P_s) has any ones in common with Y. If it does, all columns have been investigated, and a new set of columns is selected (to a_s). If not, (P) is shifted right one space to consider a new column. If the right shift causes (P) to be zero, the new column should be the first in the succeeding line address, so that x is incremented by 01 and (P) is reset to +4000000. Also, line x of P_s is set to -4000000. The negative sign is used for P_s because the minus sign is represented in the computer as a one, and this one is shifted right on a right shift command, while retaining the minus sign. Thus, successive right shifts change -4000000 to -6000000 to -7000000, etc. When (P) is not zero after shifting, line x of (P_s) is shifted right one space. In the latter case, the program then returns to c_1. In the former, however, where x has been changed, the return must be to b_1 in order to change x in certain commands.

Part 2 (Figure 19)

This section provides a change in the set of columns for which linear independence is being searched. The order of searching is as follows: begin with X covering columns 1 to b, and $Y = Y_O$ covering columns b + 1 to 2b. Then shift Y one space right until the last column covered by Y is the one-hundredth. The next step is to shift X and Y_O one space right and set $Y = Y_O$. The number Y is then shifted, and shifting continues in this manner until Y_O covers the one-hundredth column position.

The test following a_2 is to determine whether Y has reached position 100. If not, a key V is set for shifting Y. If it is, the key is set for shifting Y_O. Shifts of X, Y, and Y_O all share the same subroutine.

The process of shifting X, Y, or Y_0 is somewhat complex because each occupies five separate addresses. However, advantage can be taken of the fact that the non-zero elements appear in succession. First, a search is made for the lowest line-numbered address ℓ of V for which the contents of V, (ℓ) are non-zero. This number in V, (ℓ) is placed in register A. If (A) is now negative, the sign must be made positive, in order that the subsequent right shift of (A) will introduce a zero from the left. The contents of registers A and B are then shifted right one position. The new value of (A) is then stored in (V, (ℓ)). If the last digit in the original V, (ℓ) were a zero, the shift will cause (B) to become positive. In this case, the shifting of V is complete. If the shift causes (B) to become negative, there are two possibilities: if (V, (ℓ) + 01) were zero, it should now be made +4000000, and shifting is then complete; if (V, (ℓ) + 01) is non-zero, the remainder of the shift can be accomplished after observing the highest line-numbered address q of V for which the contents of V, (q) are non-zero. When such a non-zero V, (q) is found and stored in (A), it is shifted right one place, and the new (A) replaces (V, (q)). If (B) is now negative, -4000000 is placed in (V, (q) + 01). The shift of V is then complete.

After the shifting has been performed, it is necessary to observe the key to determine the proper branch. If the key is Y_0, it is still necessary to shift X, so that the key X replaces the key Y, and the program returns to b_s . If the key is X, shifting of both parts of the set are completed by $(Y_0) \longrightarrow (Y)$. If the shifted (Y_0) now covers the 101st column, the program is completed and the computer halts.

If not, the program returns to a_1 and begins considering columns, starting with the first. If the key is Y, only one new column is introduced in the new set, so that the return is to c_1 to consider just that new column.

Part 3 (Figure 20)

The purpose of this section is to search for a pivot element in a particular column; i.e., a non-zero element preceded in its row by all zeros in the columns of the set under investigation. The first step is to set up a quantity Z, stored in lines 02, 03, 04, 05, and 06 of sector 121, which contains ones just in those positions of columns of the set under investigation which have been considered, including the present one. A one is placed in a position of Z if and only if it appears as a one both in P_s and in either X or Y.

Using the quantities (P) and (x) which specify the column, the rows are next searched to find one with a one in that column. If one is found for some sector address (300 + j), a comparison is made to see if there are any non-zero elements in row j corresponding to non-zero elements of Z, exclusive of the particular pivot column. If there are none, this row is a suitable pivot, and the program proceeds to a_4 (perform pivot operation). If it is not a suitable pivot, j is incremented by one, and the search is repeated for a new row. If all rows of the matrix have been searched without success, it is known that the set of columns is linearly dependent. The program then proceeds to a_5 (reduce burst length).

Part 4 (Figure 21)

The purpose of this section is to perform the pivot operation; i.e., to add the pivot row (row j) digit-by-digit modulo two to all other rows which contain a non-zero element in the column under consideration. The parameter (k) specifies the row to which possible addition of row j is being considered. The case k = j is eliminated from consideration at 003 06. Otherwise, if a non-zero element is found in the proper column, row j is added to row k, the sum forming a new row k. The addition must be carried out for line addresses s = 02, 03, 04, 05, and 06. Computer speed is most critical in the section from 017 07 (c_4) through 143 06. Each command in this section must be excuted about 250,000 times in the course of the program. For this reason all but 3 of the 24 commands in this section are sequence tagged for rapid operation.

Part 5 (Figure 22)

This section causes the maximum burst length b to be reduced by one unit if a linearly dependent set of columns has been found using the previous value of b.

Suppose that the set found to be linearly dependent consists of columns x to b + x - 1 and y to b + y - 1 (we must have $y \geq b + x$). All column sets previous to this set were found linearly independent for the value b. It is then only necessary to begin the search with b - 1 at the set of columns x + 1 to b + x - 1 and y + 1 to b + y - 1, because all previous sets of columns for burst length b - 1 have been included in the search with burst length b. For convenience, however, some duplication is permitted, and the new search starts with x + 1 to b + x - 1 and b + x to 2b + x - 2. The program shown removes a single one from the right of the sequence of ones in X and Y_0. (Previous to the change, the sequence of ones in Y_0 was from b + x to 2b + x - 1.) It is actually desired to have a one removed from the extreme left of X rather than the right, and this is corrected when the program proceeds to f_2 and shifts X (but not Y_0) right one place. Then (Y_0) —— (Y), and the program proceeds to a_1.

The ones are removed in the following manner. First the sector address key 264 is set, which is the address of Y_0. Then, a number K containing a single one, initially +0000001 in line 06, is shifted left and compared with the contents of 264 having the same line address until the two compared numbers have a one in common. If K is shifted left until it becomes -0000000, the line address is reduced by 01, and K is again replaced by +0000001. When a common one is formed, K is subtracted from the contents of 264, line ℓ, and the key 260 is set, which is the address of X. After the process is repeated for 260, the program proceeds to f_2 for shifting X right and (Y) —— (Y).

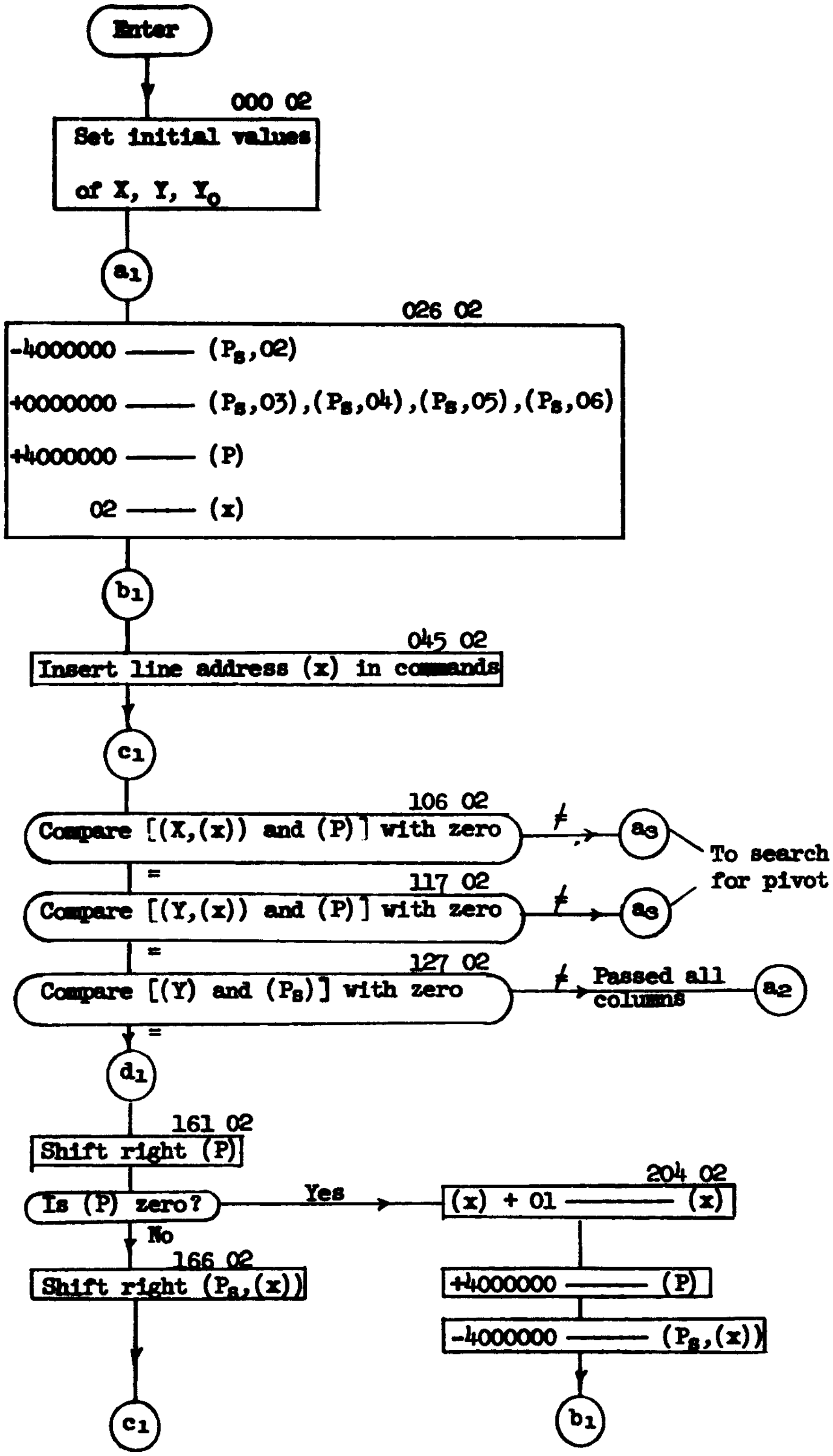

Figure 18 - Code Evaluation Program - Part 1
Selection of a Column for Pivot Search

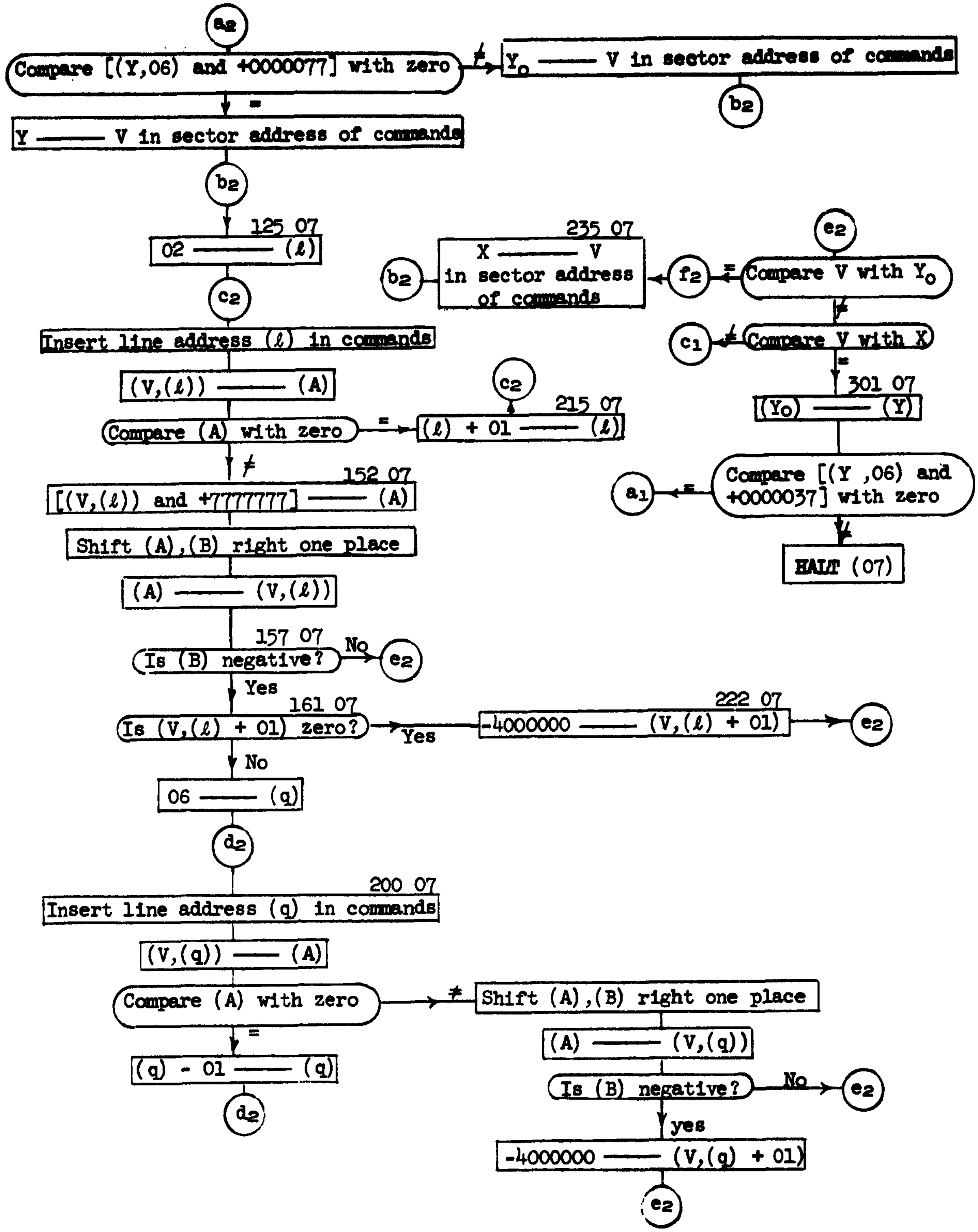

Figure 19 - Code Evaluation Program - Part 2
Selection of Set of Columns for Linear Independence Search

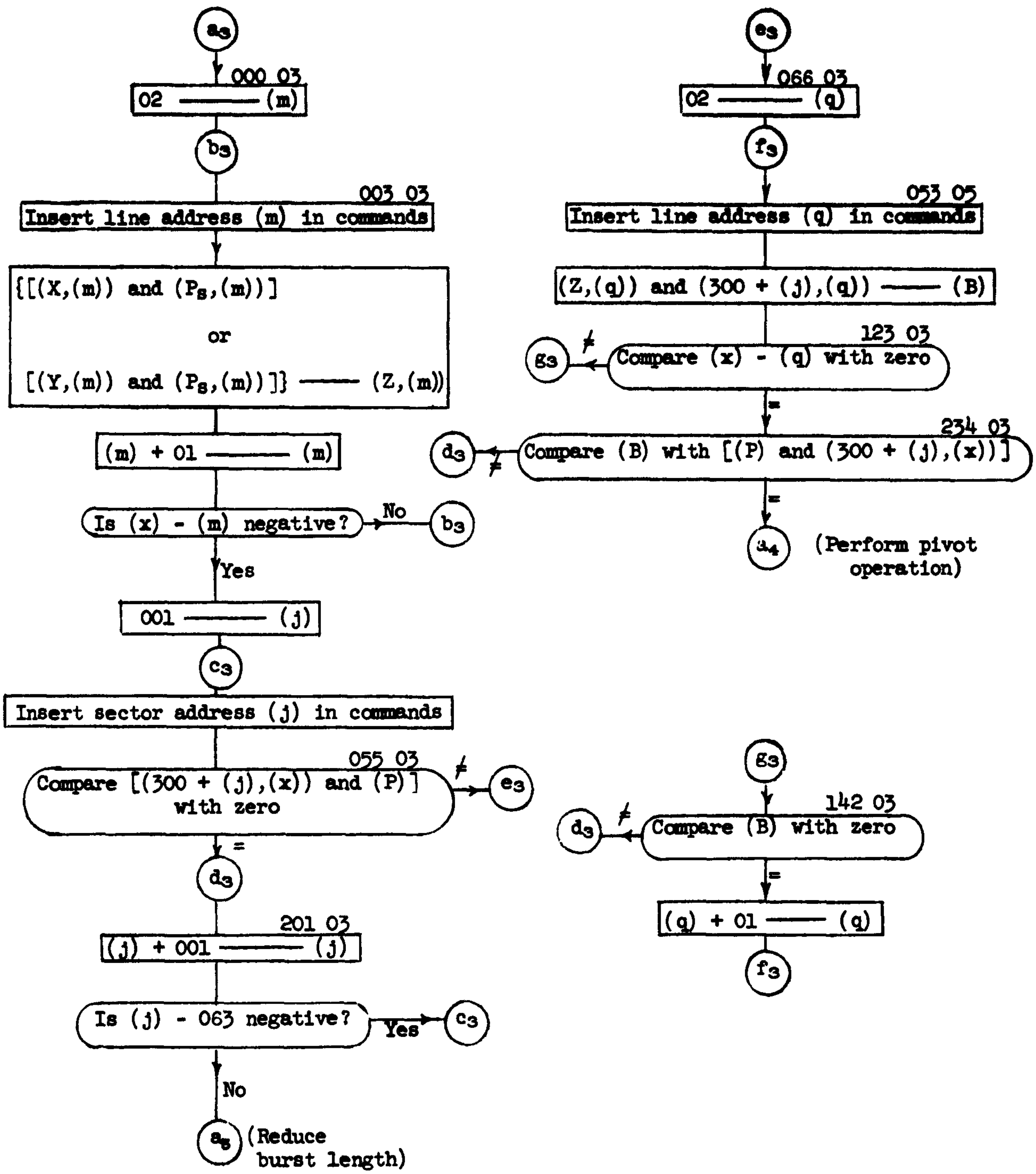

Figure 20 - Code Evaluation Program - Part 3
Search for Pivot

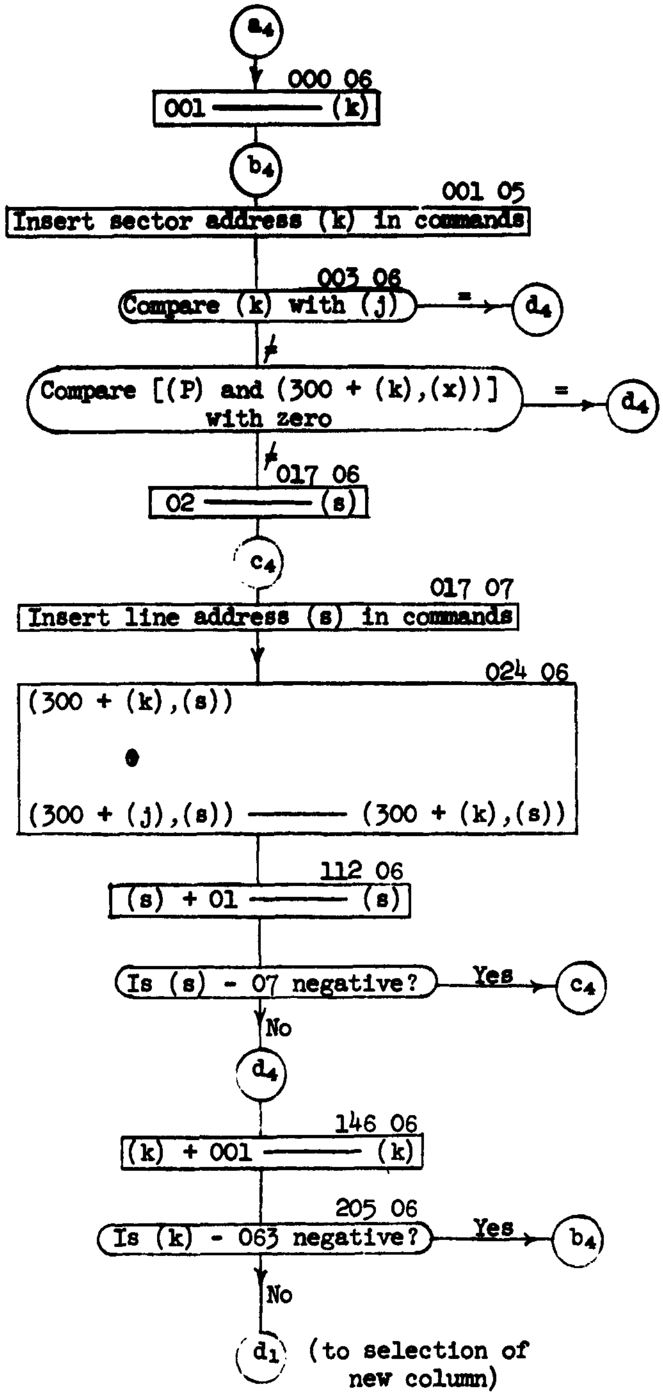

Figure 21 - Code Evaluation Program - Part 4
Perform Pivot Operation

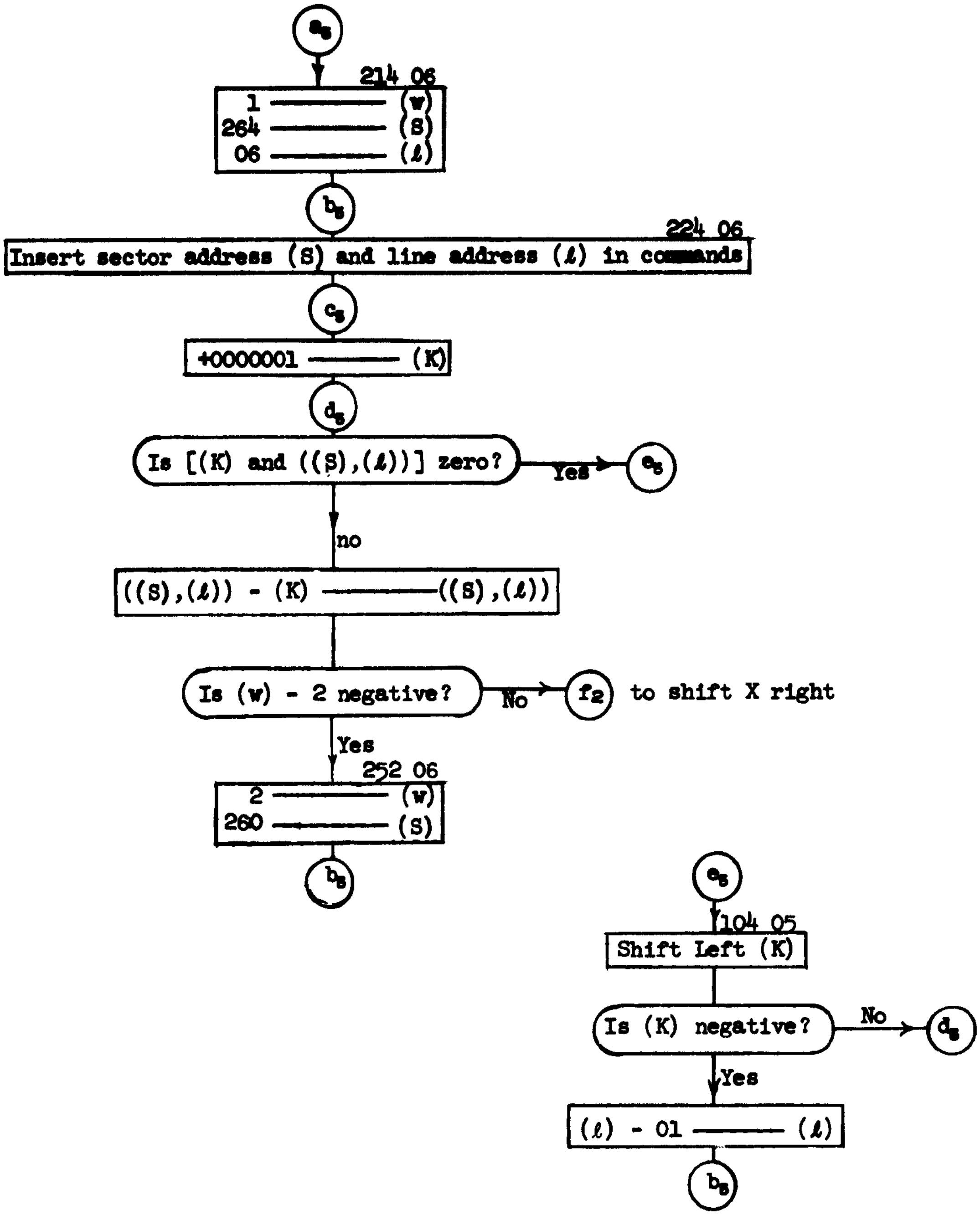

Figure 22 - Code Evaluation Program - Part 5
Reduce Burst Length

APPENDIX E

List of Programs

1. Evaluation Program and First Construction Method

The programs described in Appendices B and D can all be stored in the computer at the same time. The commands are listed in figure 23 as read out from the computer. All locations containing a number with a line through it need not be read into the computer. Some of these locations are not used, while others are filled in the course of the program.

The contents of sectors 301-362, lines 02-06, which are not shown, contain the coding matrix.

2. Second Construction Method

Figure 24 lists the commands for this program. These commands cannot be stored in the computer at the same time as the evaluation program.

	00002	00003	00004
000	C263 0502;	C001S0503;	C000 4500;
	261 1102;	000 0002;	010 1104;
	264 1102;	006 1103;	003S0504;
	263 0503;	004S0503;	160 0000;
	261 1103;	262 0400;	264 1102;
	264 1103;	006S1403;	006S0504;
	263 0504;	~~000 0003~~;	267 0500;
	261 1104;	024 1103;	010S1404;
010	264 1104;	011S0503;	~~002 0000~~;
	012S4502;	260 4200;	024 1104;
	~~007S4502~~;	006 1403;	126 0507;
	261 1105;	025 1103;	015 1104;
	261 1106;	015S0503;	015S0504;
	264 1105;	261 4200;	~~000 0007~~;
	264 1106;	006 1403;	017S1404;
	260 1104;	030 1103;	260 1100;
C20	260 1105;	021S0503;	032 1104;
	260 1106;	121 1000;	052 1403;
	265 0502;	006 1403;	034 1104;
	260 1102;	040 1103;	024S0504;
	265 0503;	~~262 0402~~;	~~267 0500~~;
	260 1103;	~~260 4202~~;	015 1404;
	027S0502;	027S1203;	031 1104;
	300 0000;	~~017S7400~~;	052 1403;
030	262 1102;	~~261 4202~~;	033 1104;
	032 4502;	032 2403;	~~267 0506~~;
	032 4502;	033S0303;	~~260 1106~~;
	262 1103;	~~033S0303~~;	~~270 0506~~;
	262 1104;	~~033S0303~~;	~~261 1106~~;
	262 1105;	027 0403;	015 0504;
	262 1106;	037S0003;	160 1403;
	040S0502;	~~037S0003~~;	015 1104;
040	100 0000;	~~121 1002~~;	363 1507;
	056 1103;	006 0503;	014 3504;
	011 1106;	043S1403;	026S3702;
	001 0503;	000 0001;	000 0000;
	124 1103;	006 1103;	045S0504;
	124 0503;	124 0503;	300 4202;
	047S1402;	006 1503;	047S1400;
	300 4200;	051 3503;	~~047S1400~~;
050	002 1105;	003S3703;	057S1103;
	124 0503;	052S0503;	~~000 0000~~;
	053S1402;	001 0000;	~~000 0000~~;
	260 4200;	007 1100;	~~000 0000~~;
	107 1102;	044S3704;	~~000 0000~~;
	124 0503;	056S0403;	~~000 0000~~;
	057S1402;	~~000 0400~~;	000 0000;
	261 4200;	303 4202;	000 0000;

Figure 23a - List of Combined Evaluation-Construction Commands
Lines 02, 03, 04

060	117 1102;	061S0303;	061S0504;
	130 1102;	~~061S0303;~~	300 4200;
	124 0503;	~~061S0303;~~	067S1400;
	077S1402;	064S5603;	~~000 0000;~~
	~~000 0000;~~	000 0000;	~~000 0000;~~
	~~000 0000;~~	201 7503;	~~000 0000;~~
	~~000 0000;~~	240 1103;	~~000 0000;~~
	~~000 0000;~~	070S0503;	~~000 0000;~~
070	~~000 0000;~~	000 0002;	102S1105;
	~~000 0000;~~	076S1100;	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	262 0400;	053S3705;	~~000 0000;~~
100	127 1102;	~~121S0402;~~	~~000 0000;~~
	124 0503;	~~000 0000;~~	~~000 0000;~~
	103S1402;	~~000 0000;~~	~~000 0000;~~
	300 4200;	~~000 0000;~~	107S0500;
	045 1104;	~~000 0000;~~	~~000 0000;~~
	045 1104;	~~000 0000;~~	~~000 0000;~~
	056 0403;	~~000 0000;~~	~~000 0000;~~
	260 4202;	~~000 0000;~~	~~000 0000;~~
110	111S0302;	~~000 0000;~~	006 1106;
	~~111S0302;~~	~~000 0000;~~	112S1404;
	~~111S0302;~~	~~000 0000;~~	300 0500;
	114S5602;	~~000 0000;~~	020 1107;
	000 0000;	~~000 0000;~~	055S3703;
	241 7502;	~~000 0000;~~	~~000 0000;~~
	000S3703;	~~000 0000;~~	~~000 0000;~~
	261 4202;	~~000 0000;~~	~~000 0000;~~
120	121S0302;	~~000 0000;~~	~~000 0000;~~
	~~017S7400;~~	~~370 0000;~~	~~000 0000;~~
	~~121S0302;~~	313 4202;	~~000 0000;~~
	124S5602;	124S0503;	~~000 0000;~~
	000 0000;	000 0002;	~~000 0000;~~
	127 7502;	136S1500;	126S0504;
	000S3703;	~~000 0000;~~	301 0000;
	~~262 0402;~~	~~000 0000;~~	146 1104;
130	~~261 4202;~~	~~000 0000;~~	131S0504;
	000 0302;	~~000 0000;~~	000 0002;
	156 5602;	~~000 0000;~~	141 1104;
	135 7502;	~~000 0000;~~	134S0504;
	243S3702;	~~000 0000;~~	100 0000;
	124 0503;	~~000 0000;~~	155 1104;
	140 1102;	~~000 0000;~~	137S0504;
	140S0502;	140S5603;	277 4200;

Figure 23b - List of Combined Evaluation-Construction Commands
Lines 02, 03, 04

140	~~000 0001;~~	000 0000;	141S1404;
	142S1502;	234 7503;	~~141S1404;~~
	000 0001;	143S0303;	156 1104;
	140 1102;	~~143S0303;~~	144S0504;
	145S1502;	~~143S0303;~~	000 0400;
	000 0002;	146S5603;	146S1404;
	161 3502;	000 0000;	~~146S1404;~~
	150S0502;	151 7503;	141 1404;
150	261 0500;	201S3703;	210 1104;
	140 1402;	156S0500;	152S1404;
	154 1102;	~~000 0000;~~	000 0400;
	000 2402;	~~000 0000;~~	213 1104;
	~~261 0505;~~	~~000 0000;~~	155S0404;
	156S5602;	~~000 0000;~~	~~155S0404;~~
	000 0000;	~~000 0000;~~	~~155S0404;~~
	137 7502;	160S1403;	160S0304;
160	243S3702;	000 0001;	~~160S0304;~~
	056 0503;	176S1100;	~~160S0304;~~
	164 2210;	~~000 0000;~~	163S5604;
	164S5602;	~~000 0000;~~	000 0000;
	000 0000;	~~000 0000;~~	207 7504;
	204 7502;	~~000 0000;~~	155 0504;
	056 1103;	~~000 0000;~~	170 2210;
	011 1106;	~~000 0000;~~	170S5604;
170	266 0502;	~~000 0000;~~	000 0000;
	124 1403;	~~000 0000;~~	174 7504;
	200 1102;	~~000 0000;~~	155 1104;
	174S0502;	~~000 0000;~~	143S3704;
	262 1100;	~~000 0000;~~	141 0504;
	124 1403;	~~000 0000;~~	176S1404;
	202 1102;	~~000 0000;~~	000 0001;
	215 1102;	053S3705;	141 1104;
200	262 0502;	~~053S3705;~~	201S1504;
	203 2210;	207S0500;	000 0007;
	~~262 1102;~~	~~000 0000;~~	133 3504;
	106S3702;	~~000 0000;~~	204S0504;
	124 0503;	~~000 0000;~~	004 0000;
	160 1403;	~~000 0000;~~	254 1102;
	124 1103;	~~000 0000;~~	000S3704;
	174 1402;	~~000 0000;~~	155 0504;
210	215 1102;	211S1403;	~~155 0504;~~
	040 0502;	001 0000;	212S0004;
	056 1103;	227S1100;	~~212S0004;~~
	011 1106;	~~000 0000;~~	~~212S0004;~~
	027 0502;	~~000 0000;~~	146 0504;
	~~262 1102;~~	~~000 0000;~~	216S1404;
	045S3702;	~~000 0000;~~	001 0000;
	261 0406;	~~000 0000;~~	146 1104;

Figure 23c - List of Combined Evaluation-Construction Commands
Lines 02, 03, 04

220	221S4202;	~~000 0000;~~	~~000 0000;~~
	000 0057|	~~000 0000;~~	~~000 0000;~~
	223S0303;	~~000 0000;~~	~~000 0000;~~
	~~223S0303;~~	~~000 0000;~~	~~000 0000;~~
	~~223S0303;~~	~~000 0000;~~	~~000 0000;~~
	226S5602;	~~000 0000;~~	~~000 0000;~~
	000 0000;	~~000 0000;~~	~~000 0000;~~
	234 7502;	~~000 0000;~~	~~000 0000;~~
230	231S0502;	231S1503;	~~000 0000;~~
	264 0500;	063 0000;	~~000 0000;~~
	131 1107;	044 3504;	~~000 0000;~~
	125S3707;	250S3703;	~~000 0000;~~
	235S0502;	235S0303;	~~000 0000;~~
	261 0500;	~~235S0303;~~	~~000 0000;~~
	131 1107;	~~235S0303;~~	~~000 0000;~~
	125S3707;	240S5603;	~~000 0000;~~
240	~~000 0000;~~	~~000 1000;~~	~~000 0000;~~
	056 0403;	243 7503;	~~000 0000;~~
	117S3702;	201S3703;	~~000 0000;~~
	264 0502;	264 0502;	~~000 0000;~~
	245S1502;	245S1503;	~~000 0000;~~
	160 0000;	176 0000;	~~000 0000;~~
	217 3502;	000 3506;	~~000 0000;~~
	010 0504;	325S3707;	~~000 0000;~~
250	251S1402;	264 0502;	~~000 0000;~~
	002 0000;	245 1502;	~~000 0000;~~
	010 1104;	254 3503;	~~000 0000;~~
	254S1502;	160S3705;	~~000 0000;~~
	000 0000;	214S3706;	~~000 0000;~~
	005 3504;	~~000 0002;~~	~~000 0000;~~
	000S3707;	~~010 0000;~~	~~000 0000;~~
	~~000 0000;~~	~~000 0003;~~	~~000 0000;~~
260	~~017S7777|~~	~~374 0000;~~	~~000 0000;~~
	~~000 0000;~~	~~003S7777|~~	~~377 0000;~~
	~~377S7400;~~	~~000 0000;~~	~~000 0000;~~
	000 0000;	007S7777|	377S0000;
	~~000 0000;~~	~~003S7777|~~	~~377 0000;~~
	177S7777|	370 0000;	000 0000;
	262 0500;	~~000 0000;~~	~~000 0000;~~
	377S7777|	370 0000;	000 0000;
270	000 0000;	007S7777|	377S0000;
	000 0000;	000 0000;	000 7777|
	000 0000;	000 0000;	000 0000;
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	177S7777|	170 0000;	000 0000;

Figure 23d - List of Combined Evaluation-Construction Commands
Lines 02, 03, 04

	00005	00006	00007
000	~~C000 0000~~;	C052 0503;	C001S0507;
	002S0505;	004 1100;	176 0000;
	~~300 4202~~;	001S3705;	264 1102;
	004S1400;	004S0500;	052 0503;
	~~004S1400~~;	~~004S0500~~;	366 1107;
	012 1106;	006S5606;	251 1407;
	007S0505;	~~003 0000~~;	254 1107;
	300 0500;	146 7506;	026S3702;
010	024S1400;	011S0406;	000 0000;
	000 4500;	~~000 0400~~;	000 0500;
	010 1104;	~~362 4202~~;	264 0000;
	030S3705;	000 0300;	~~260 0500~~;
	015S0505;	015S5606;	000 3100;
	001 0000;	000 0000;	000 0000I
	010 1104;	146 7506;	260 0000;
	030S3705;	020S0506;	020S0507;
020	021S0505;	000 0002;	~~302 0500~~;
	002 0000;	022S1100;	022S1400;
	010 1104;	~~022S1100~~;	~~022S1400~~;
	030S3705;	017S3707;	024S1106;
	~~000 0000~~;	~~332 0506~~;	~~024S1106~~;
	026S1107;	026S1100;	026S0507;
	~~026S1107~~;	~~026S1100~~;	~~362 0500~~;
	042S0505;	044S3706;	042S1400;
030	000 4500;	~~000 0000~~;	~~000 0000~~;
	254 1102;	~~000 0000~~;	~~000 0000~~;
	002S3704;	~~000 0000~~;	~~000 0000~~;
	~~000 0000~~;	~~000 0000~~;	~~000 0000~~;
	~~000 0000~~;	~~000 0000~~;	~~000 0000~~;
	~~000 0000~~;	~~000 0000~~;	~~000 0000~~;
	~~000 0000~~;	~~000 0000~~;	~~000 0000~~;
	~~000 0000~~;	~~000 0000~~;	000 0400;
040	~~000 0000~~;	~~000 0000~~;	~~000 0000~~;
	~~000 0000~~;	~~000 0000~~;	~~000 0000~~;
	300 1200;	~~000 0000~~;	~~000 0000~~;
	044S1400;	~~000 0000~~;	044S1106;
	~~044S1400~~;	~~361 0506~~;	~~044S1106~~;
	046S1107;	050S1100;	046S0507;
	~~046S1107~~;	~~050S1100~~;	~~362 1200~~;
	003S3706;	~~050S1100~~;	062S1400;
050	~~000 0000~~;	~~050S1100~~;	~~000 0000~~;
	~~000 0000~~;	052S0506;	~~000 0000~~;
	~~000 0000~~;	177S77771	~~000 0000~~;
	054S0505;	066S1500;	~~000 0000~~;
	121S0400;	~~000 0000~~;	~~000 0000~~;
	056S1400;	~~000 0000~~;	~~000 0000~~;
	~~056S1400~~;	~~000 0000~~;	~~000 0000~~;
	100S1103;	~~000 0000~~;	~~000 0000~~;

Figure 23e - List of Combined Evaluation-Construction Commands
Lines 05, 06, 07

060	000 0000;	~~000 0000;~~	~~000 0000;~~
	062S0505;	~~000 0000;~~	~~000 0000;~~
	000 0003;	~~000 0000;~~	~~000 0000;~~
	255 1103;	~~000 0000;~~	111S1106;
	065S0505;	~~000 0000;~~	~~000 0000;~~
	205 0507;	~~000 0000;~~	~~000 0000;~~
	241 1106;	~~000 0000;~~	~~000 0000;~~
	037 1407;	070S0106;	~~000 0000;~~
070	243 1106;	~~070S0106;~~	~~000 0000;~~
	014 1407;	106S0600;	~~000 0000;~~
	234 1106;	~~000 0000;~~	~~000 0000;~~
	233S3706;	~~000 0000;~~	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
100	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	102S0505;	~~000 0000;~~	~~000 0000;~~
	~~303 4200;~~	~~000 0000;~~	~~000 0000;~~
	116S1400;	~~000 0000;~~	~~000 0000;~~
	106 2110;	~~000 0000;~~	~~000 0000;~~
	110 3605;	~~000 0000;~~	~~000 0000;~~
	000 0200;	~~000 0000;~~	~~000 0000;~~
	234S3706;	110S4600;	~~000 0000;~~
110	257 0503;	~~110S4600;~~	~~000 0000;~~
	142 1502;	361 1206;	~~000 0000;~~
	257 1103;	122S0500;	024S3706;
	224S3706;	~~000 0000;~~	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	122S1103;	~~000 0000;~~	~~000 0000;~~
120	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	~~000 0077+~~	~~377377730;~~	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	100S3703;	124S1406;	~~000 0000;~~
	~~000 0000;~~	000 0001;	~~000 0000;~~
	~~000 0000;~~	142S1100;	126S0507;
	~~000 0000;~~	~~000 0000;~~	000 0002;
	~~000 0000;~~	~~000 0000;~~	133 1107;
130	~~000 0000;~~	~~000 0000;~~	131S0507;
	~~000 0000;~~	~~000 0000;~~	~~260 0500;~~
	~~000 0000;~~	~~000 0000;~~	133S1407;
	~~000 0000;~~	~~000 0000;~~	~~000 0002;~~
	~~000 0000;~~	~~000 0000;~~	146 1107;
	~~000 0000;~~	~~000 0000;~~	136S1407;
	~~000 0000;~~	~~000 0000;~~	000 0400;
	~~000 0000;~~	~~000 0000:~~	156 1107;

Figure 23f - List of Combined Evaluation-Construction Commands
Lines 05, 06, 07

140	~~000 0000;~~	~~000 0000;~~	141S1407;
	~~000 0000;~~	~~000 0000;~~	000 0001;
	~~000 0000;~~	~~000 0000;~~	224 1107;
	~~000 0000;~~	144S1506;	144S1507;
	~~000 0000;~~	000 0007;	000 0400;
	~~000 0000;~~	017 3507;	161 1107;
	~~000 0000;~~	164S0500;	~~260 0502;~~
	~~000 0000;~~	~~000 0000;~~	150S5607;
150	~~000 0000;~~	~~000 0000;~~	000 0000;
	~~000 0000;~~	~~000 0000;~~	215 7507;
	~~000 0000;~~	~~000 0000;~~	000 0100;
	~~000 0000;~~	~~000 0000;~~	246 4207;
	~~000 0000;~~	~~000 0000;~~	000 0307;
	~~000 0000;~~	~~000 0000;~~	157 2210;
	~~000 0000;~~	~~000 0000;~~	~~260 1102;~~
	~~000 0000;~~	~~000 0000;~~	161 3607;
160	236 0505;	~~000 0000;~~	225S3707;
	212 1105;	~~000 0000;~~	~~260 0503;~~
	237 0505;	~~000 0000;~~	163S5607;
	172 1105;	~~000 0000;~~	000 0000;
	201 0505;	~~000 0000;~~	222 7507;
	240 1105;	166S1406;	166S0507;
	212 0505;	001 0000;	000 0006;
	170S1405;	204S1100;	172 1107;
170	300 1200;	~~000 0000;~~	131 0507;
	172S1405;	~~000 0000;~~	172S1407;
	~~172S1405;~~	~~000 0000;~~	~~000 0003;~~
	210 1105;	~~000 0000;~~	201 1107;
	172 0505;	~~000 0000;~~	136 1407;
	176S1405;	~~000 0000;~~	206 1107;
	277 4200;	~~000 0000;~~	177S1407;
	207 1105;	~~000 0000;~~	000 0001;
200	201S0605;	~~000 0000;~~	213 1107;
	RANDOM NUMBER	~~000 0000;~~	~~260 0503;~~
	203S0405;	~~000 0000;~~	203S5607;
	166 1605!	~~000 0000;~~	000 0000;
	233 3200;	~~000 0000;~~	241 7507;
	201 1205;	206S1506;	207 2210;
	000 0200;	063 0000;	~~260 1103;~~
	~~000 0200;~~	210S5606;	211 3607;
210	~~000 0200;~~	000 0000;	225S3707;
	212S0505;	161 7502;	212S0507;
	~~212S0505;~~	001S3705;	300 0000;
	214S1505;	000 0000;	~~260 1104;~~
	062 0000;	142 0502;	225S3707;
	232 3505;	255 1103;	133 0507;
	172 0505;	011 0507;	217S1407;
	220S1505;	012 1407;	000 0001;

Figure 23g - List of Combined Evaluation-Construction Commands
Lines 05, 06, 07

220	000 0006;	013 1107;	133 1107;
	223 3505;	222S0506;	130S3707;
	125S3704;	000 0006;	223S0507;
	172 0505;	257 1103;	300 0000;
	225S1405;	013 0507;	~~260 1103;~~
	000 0001;	257 1403;	131 0507;
	172 1105;	241 1106;	227S5607;
	236 0505;	037 1407;	264 0500;
230	212 1105;	243 1106;	235 7507;
	166S3705;	014 1407;	232S5607;
	212 0505;	234 1106;	260 0500;
	236 1405;	015 0407;	301 7507;
	212 1105;	~~260 4203;~~	106S3702;
	166S3705;	000 0300;	236S0507;
	001 0000;	226 5602;	260 0500;
	000 0002;	104 7505;	131 1107;
240	~~073S6+541~~	256 1103;	125S3707;
	~~000 0000;~~	~~260 0503;~~	172 0507;
	~~000 0000;~~	256 1503;	177 1507;
	~~000 0000;~~	~~260 1103;~~	172 1107;
	~~000 0000;~~	255 0503;	170S3707;
	~~000 0000;~~	001 1503;	125S3707;
	~~000 0000;~~	252 3506;	177S7777I
	~~000 0000;~~	235S3707;	000 0000;
250	~~000 0000;~~	~~000 0000;~~	251S0507;
	~~000 0000;~~	~~000 0000;~~	300 0500;
	~~000 0000;~~	~~000 0000;~~	366 1407;
	~~000 0000;~~	~~000 0000;~~	254S1107;
	~~000 0000;~~	~~000 0000;~~	~~362 0500;~~
	~~000 0000;~~	~~000 0000;~~	161S3702;
	~~000 0000;~~	~~000 0000;~~	300 0400;
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
260	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	000 0000;	000 0000;	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	000 0000;	000 0000;	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	000 0000;	000 0000;	~~000 0000;~~
270	000 0000;	000 0000;	~~000 0000;~~
	377S7400;	000 0000;	~~000 0000;~~
	000 0377I	377S7730;	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	~~000 0000;~~	~~000 0000;~~	~~000 0000;~~
	000 0377I	177S7730;	004 0000;

Figure 23h - List of Combined Evaluation-Construction Commands Lines 05, 06, 07

300	~~000 0000;~~	340	341S1407;
	264 0502;		000 0400;
	261 1102;		355 1107;
	264 0503;		254 0507;
	261 1103;		334 1407;
	264 0504;		350 1107;
	261 1104;		341 1407;
	264 0505;		354 1107;
310	261 1105;	350	~~362 0506;~~
	264 0506;		~~362 0406;~~
	261 1106;		353S0107;
	000 0100;		~~353S0107;~~
	315S4207;		~~362 1106;~~
	000 00471		~~362 1006;~~
	317S0307;		334 0507;
	~~317S0307;~~		360S1407;
320	~~317S0307;~~	360	000 0001;
	322S5607;		334 1107;
	000 0000;		363S1507;
	026 7502;		000 0007;
	000 0007;		333 3507;
	256 0507;		366S0507;
	327S1400;		~~063 0000;~~
	~~327S1400;~~		052 1403;
330	336 1107;	370	366 1107;
	145 0502;		372S1507;
	334 1107;		063 0000;
	334S0507;		250 3507;
	~~000 0007;~~		000 0005;
	336S1407;		~~000 0000;~~
	~~362 0400;~~		~~000 0000;~~
	351 1107;		~~000 0000;~~

Figure 23i - List of Combined Evaluation-Construction Commands-Line 07

```
       00002$134 0502;
       011 1102;
       003S0502;
       001 0000;
       031 1102;
       006S4500;

       016 1102;
010    011S0502;

       013S1402;

       065 1102;
       016S1402;

       020 1102;
020
       133 3602;
       023S0402;
       +0000025
       034 1002;
       036 1202;
       027S0502;
       300 0602;
030    031S1402;

       057 1102;
       034S0402;

       036S0602;

       041 2100;
040    117 3402;
       034 1002;
       036 1202;
       051 3602;
       031 0502;
       046S1402;
       001 0000;
       031 1102;
050    026S3702;
       052S4500;

       055 1102;
       055S0502;

       057S1402;

060    102 1102;
       062S1402;
       000 0700;
       077 1102;
       065S0502;

       067S1402;

070    055 1402;
       104 1102;
       073S1402;
       000 3400;
       103 1102;
       076S0502;
       +7777777

100    101S0100;




       055 0502;
       107S1502;
       000 0002;
110    112 3502;
       044S3702;
       055 0502;
       114S1402;
       000 0001;
       055 1102;
       054S3702;
       016 0502;
120    121S1402;
       000 0001;
       016 1102;
       124S1502;
       000 0003;
       010 3502;
       011 0502;

130    001 0000;
       011 1102;
       002S3702;
       000 0006;
```

Figure 24a - List of Second Construction Method Commands-Line 02

000	00003$100 0503;	00005$001S0504;
	077 1103;	~~001S0504;~~
	~~300 0405;~~	067 1102;
	077 4203;	004S0505;
	~~300 1205;~~	031 0000;
	002 0503;	134 1102;
	101 1403;	007S0505;
	002 1103;	130S1502;
010	004 0503;	127 1102;
	101 1403;	012S0504;
	004 1103;	~~012S0504;~~
	077 0503;	200 1102;
	016 2210;	014S3704;
	077 1103;	
	102 1403;	
	002 3503;	
020	002 0503;	
	103 1403;	
	002 1103;	
	004 0503;	
	103 1403;	
	004 1103;	
	104 0503;	10005$000 0005;
	002 1503;	10003$-3777777
030	100 3505;	001 0000;
	000S3703;	+0000001
		000 0001;
		277 0404;

Figure 24b - List of Second Construction Method Commands-Lines 03, 05

000 00004$001S0504;
100 0400;
067 1102;
004S0504;
001 0000;
134 1102;
007S0504;
130S1402;

010 127 1102;
012S0504;
300 0000;
232 1102;
015S0504;
000 0001;
027 1104;
020S0504;

020 001 0000;
025 1104;
023S0504;
300 1101;
025S1404;
~~077 0000;~~
027S1404;
~~000 0003;~~

030 033 1104;
032S4500;
~~032S4500;~~
~~377 1104;~~
025 0504;
036S1504;
077 0000;
045 3504;

040 027 0504;
042S1504;
000 0003;
052 3504;
114S3704;
025 0504;
047S1404;
001 0000;

050 025 1104;
022S3704;
027 0504;
054S1404;
000 0001;
027 1104;
017S3704;

11404$115S0504;
001 0000;
130 1104;
120S0504;

120 000 0001;
132 1104;
123S0504;
100 0000;
135 1104;
126S0504;
300 1101;
130S1404;

130 ~~100 0000;~~
132S1404;
~~000 0004;~~
141 1104;
135S0504;
~~000 0000;~~
137S5604;
000 0000;

140 152 7504;
~~000 1104;~~
144S2210;
~~144S2210;~~
135 1104;
130 0504;
147S1404;
001 0000;

150 130 1104;
125S3704;
132 0504;
154S1404;
000 0001;
132 1104;
157S1504;
000 0005;

160 122 3504;
000S3702;
000 0000;

Figure 24c - List of Second Construction Method Commands-Line 04

000	00006$001S0506;
	000 0402;
	067 1102;
	004S0504;
	~~004S0504;~~
	134 1102;
	007S0504;
	~~007S0504;~~
010	127 1102;
	012S0504;
	~~012S0504;~~
	232 1102;
	015S0506;
	000 0001;
	027 1106;
	020S0506;
020	001 0000;
	025 1106;
	023S0506;
	200 1104;
	025S1406;
	~~077 0000;~~
	027S1406;
	~~000 0003;~~
030	033 1106;
	032S4500;
	~~032S4500;~~
	~~277 1107;~~
	025 0506;
	036S1506;
	077 0000;
	045 3506;
040	027 0506;
	042S1506;
	000 0003;
	052 3506;
	000S3702;
	025 0506;
	047S1406;
	001 0000;
050	025 1106;
	022S3706;
	027 0506;
	054S1406;
	000 0001;
	027 1106;
	017S3706;

Figure 24d - List of Second Construction Method Commands-Line 06

000	00007$001S0507;	040	000 0004;
	001 0000;		043 3507;
	032 1107;		056S3707;
	004S0507;		012 0507;
	000 0002;		045S1407;
	012 1107;		000 0001;
	032 0507;		012 1107;
	010S1407;		001 0507;
010	200 1200;	050	032 1107;
	012S1407;		006S3707;
	~~000 0004~~;		032 0507;
	030 1107;		001 1407;
	012 0507;		032 1107;
	016S1407;		006S3707;
	177 4200;		057S0507;
	027 1107;		201 0402;
020	021S0607;	060	002 1103;
	RANDOM NUMBER		062S0507;
	023S0407;		201 1202;
	166 16051		004 1103;
	052 3200;		000S3703;
	021 1207;		
	000 0200;		
	~~177 4204;~~		
030	~~262 1204;~~		17702$+7777777
	032S0507;		17703$+7400000
	~~062 0000;~~		17704$+0000000
	034S1507;		
	077 0000;		
	052 3507;		
	012 0507;		
	040S1507;		

Figure 24e - List of Second Construction Method Commands-Line 07 and Sector 177

APPENDIX F

Adaptability to Other Code Sizes

The following is an enumeration of the changes which must be made according to the value of n and c chosen.

1. Number of Rows of Matrix

```
372 07$ 0XX 0000;
206 06$ 0XX 0000;
231 03$ 0XX 0000;
214 05$ 0YY 0000;
```

$XX = 1 + (C)_{octal}$; $YY = (C)_{octal}$

The above quantities specify when the last row has been reached.

2. Number of Columns of Matrix

```
                         for n = 100,
217 02$ 267 040X;        217 02$ 261 0406;

221 02$ +XXXXXXX         221 02$ +0000077
```

This command and number tell the program if the set of columns Y has reached inclusion of column n. If one writes $n = 21p + q + 1$, where $q < 21$, then the line address of the command in 217 02 should be $02 + p$, and the number in 221 02 should contain zeros in the leftmost q positions, and ones in the remaining positions.

Also

	for n = 100,
313 07$ 264 040X;	313 07$ 264 0406;
315 07$ +XXXXXXX	315 07$ +0000037
	or
	+0000077

This command and number tell the program if the set of columns Y_0 has reached inclusion of column n. (Logically, it would appear from the program that n + 1 should be used instead of n, but the search for linear independence over the last set of columns is redundant because of the method of code construction.)

3. <u>Key for Generating Code</u>

	n = 100, c = 50
277 02$ —	+7777777
277 03$ —	+7400000
277 04$ —	+0000000
277 05$ —	+0000777
277 06$ —	+7777740

The first n digits of those beginning with the leftmost in 277 02 and ending with the rightmost in 277 06 correspond in position to the n columns of the [P] matrix. A set of c consecutive digits of these are made zero, and the other n - c are each set equal to one. The c zeros correspond to the positions of the check digits. The code generation program fills the columns corresponding to ones in 277 with random numbers, and places a c x c unit diagonal matrix in the check digit positions. In certain cases it will be necessary to use a sector address other then 277, as explained below.

4. Key for Decoding Matrices

267-276, lines 02, 03, 04, 05, 06

In order for a code to be suitable, certain sets of columns of [P] must be linearly independent, namely $[P_1]$, $[P_2]$, ..., $[P_q]$. Sectors 267, 270 are filled such that each position corresponding to a column of $[P_1]$ contains a one in either 267 or 270, and all other positions are zero in both 267 and 270. The contents of 267 and 270 replace the quantities X and Y in the main program to provide a test of the linear independence of $[P_1]$. For the (100, 50) code, the following were the contents:

line	02	03	04	05	06
267	+7777777	-7400000	+0000000	+0000000	+0000000
270	+0000000	+0377777	-7400000	+0000000	+0000000

Similarly, 271 and 272 can represent $[P_2]$, etc. (In some cases, 270 and 271 together can represent one of the matrices as well.)

An additional charge needed is

254 02$ 0XX 0000;

where XX = $(2q)_{octal}$, and q is the number of matrices which must be linearly independent in order to make the code suitable. In case $q > 4$, it is necessary for the decoding matrix key to occupy sectors 277 and some suceeding sectors. This will only happen if c is considerably below the maximum, in which case the first row of the [P] matrix can be stored further down than sector 301. The key for generating the code would also have to be moved to a sector other than 277. This would require the following changes.

Replace 300 4200; by XXX 4200; in

047 02

103 02

061 04

Replace 300 0500; by XXX 0500; in

112 04

007 05

251 07

Replace 300 1200; by XXX 1200; in

042 05

170 05 .

Replace 300 0400; by XXX 0400; in

256 07 .

Replace 300 0000; by YYY 0000; in

126 04 .

Replace 277 4200; by ZZZ 4200; in

137 04 .

The quantity YYY is the sector location of the first row, and XXX = YYY - 001. The address ZZZ is for the new key location.

APPENDIX G

Checking the Computer Result

In the course of the main evaluation program, the computer executes more than ten million commands. While it is very unlikely that a computer error would occur without being coupled with a premature halt in the program, it was felt that some check was needed to ensure that the computer performed the program correctly.

An error committed in performing the various additions of rows to the matrix could readily be detected by operating on the final derived matrix after computer halt so as to re-diagonalize columns 51 - 100. This is done simply by starting the program at 017 05 (a_{11} in Figure 9, Appendix B). If the resulting matrix is not identical to [P], then some error has been made, while if it is identical, then it is virtually certain that no error has been made in performing the additions of rows to other rows. No such error was ever observed.

An additional check on errors is to run the program through twice and compare the two final derived matrices. This was done for the code described in Section B, and the same result was obtained in both cases.

The code shown in Appendix H was checked in a different manner, in addition to the check by return to [P]. Instead of the "reduced burst length" portion of the program, the computer was caused to halt when search at that burst length yielded a set of linearly dependent columns. The program halted for bursts of length 21, and visual observation showed that the set of columns (3-23, 53-73) were indeed linearly dependent (see figure 25, especially column 73). As a fur ther check, a return to the original matrix was made, and this same set of columns was checked for linear independence. Again, the set was found linearly dependent. The same was done for b = 20, and b = 19. Finally, the program ran to completion for b = 18. This final result was checked further by return to the original matrix [P].

30102	30103	30104	30105	30106
D+6000000	D+3123755	D+0200000	D+0004343	D+7044600
+4000000	+5437434	+1300000	+0004116	+5625400
+2100000	+0152407	+1040000	+0004250	+5754000
+0200000	+1553110	+0200000	+0004143	+6250400
+0040000	+1760000	+1300000	+0000004	+6107040
+0002000	+1546072	+0300000	+0000130	+2273600
+7000000	+1116422	+0000000	+0004620	+0037640
+0010000	+0374173	+0340000	+0000236	+3727600
+0020000	+0347370	+0340000	+0004451	+6111200
+0000400	+0110467	+1140000	+0004012	+1563440
+6001000	+0434726	+0140000	+0004672	+3251200
+0400000	+1443272	+1000000	+0004623	+0074640
+4000200	+0515526	+1340000	+0000725	+1565240
+6000100	+1230125	+1000000	+0000304	+2345600
+4000004	+0372530	+1300000	+0000310	+3504200
+2004000	+0414202	+0240000	+0004716	+6272440
+0000040	+0475606	+0040000	+0000265	+5512040
+6000002	+1773564	+1300000	+0004637	+1553400
+2000020	+0272635	+1240000	+0000502	+5305200
+6000000	+1331364	+1200000	+0204152	+1207640
+2000000	+0715112	+0240000	+0040540	+2500400
+0000010	+1370320	+1300000	+0000437	+2015400
+4000001	+0420711	+1340000	+0000774	+2666600
+0000000	+0177070	+0050000	+0000447	+2133240
+6000000	+0225726	+1000000	+1000401	+7040200
+2000000	+0417775	+1304000	+0000067	+3204400
+2000000	+0461177	+0101000	+0004427	+6037640
+4000000	+1554667	+1340040	+0000320	+4254240
+4000000	+0437343	+0340400	+0000560	+4627040
+0000000	+0221321	+1100020	+0004750	+6204440
+6000000	+1245370	+1140000	+0010543	+5767640
+6000000	+1210437	+1040200	+0000017	+2765040
+0000000	+1253475	+0000010	+0004741	+2571400
+6000000	+1106734	+0340004	+0000004	+3364640
+4000000	+0221601	+1140002	+0000670	+4637400
+2000000	+0525740	+1240001	+0000200	+4720200
+4000000	+0657630	+0240000	+4000536	+3125240
+2000000	+0465001	+0300000	+2000001	+6325600
+2000000	+0671553	+0240000	+0404777	+5753240
+4000000	+1023124	+0040000	+0020546	+7660240
+6000000	+1757257	+2300000	+0000312	+6237440
+4000000	+1633240	+4200000	+0000432	+7072040
+2000000	+1605224	+0300100	+0004706	+3400600
+2000000	+0136200	+0540000	+0000260	+1146200
+2000000	+1713424	+1140000	+0000715	+5251340
+4000000	+0515350	+0302000	+0004175	+1100600
+6000000	+1102650	+0360000	+0000177	+4016200
+0000000	+1042665	+1140000	+0104255	+6655400
+4000000	+1067006	+0200000	+0002615	+0517640
+0000000	+0473303	+1200000	+0001102	+2570040

Figure 25 - Transformed Code of Appendix H Showing Linear Dependence of Columns 3-23, 53-73

APPENDIX H

An Additional (100, 50) Code

Figures 26-28 illustrate the coding and decoding matrices for a (100, 50) code which is capable of correcting all bursts of length 18 or less. This code was constructed by means of the second method of code construction, as described in Section C.2 and Appendix C. The individual 25 x 25 matrices from which the code was built are shown in figure 29, and related matrices are shown in figure 30. Octal digits represent three binary digits in a row, except the last octal digit, which is either 4 or 0 according as the last binary digit is 1 or 0.

30102	30103	30104	30105	30106
D+1260240	D+3737210	D+2550000	D+0000000	D+0000000
+4223705	+1320204	+0704000	+0000000	+0000000
+5105507	+5474222	+5342000	+0000000	+0000000
+2573477	+5141607	+2641000	+0000000	+0000000
+3174122	+2112216	+1720400	+0000000	+0000000
+4422412	+6702652	+2340200	+0000000	+0000000
+1674110	+1404760	+0340100	+0000000	+0000000
+6327013	+6277332	+0620040	+0000000	+0000000
+3225022	+4232132	+4520020	+0000000	+0000000
+1377231	+4021544	+7420010	+0000000	+0000000
+0471331	+4040047	+6700004	+0000000	+0000000
+4710510	+1121032	+6060002	+0000000	+0000000
+5070603	+0566612	+6420001	+0000000	+0000000
+7452661	+5726531	+3460000	+4000000	+0000000
+4566667	+7350465	+6260000	+2000000	+0000000
+4070055	+2436127	+1420000	+1000000	+0000000
+4105317	+2565753	+5160000	+0400000	+0000000
+6545073	+0041113	+4440000	+0200000	+0000000
+5111711	+2005355	+5400000	+0100000	+0000000
+5576223	+0714010	+5300000	+0040000	+0000000
+4175114	+1203771	+5200000	+0020000	+0000000
+7235466	+2215070	+5300000	+0010000	+0000000
+4633410	+0307372	+1740000	+0004000	+0000000
+3420610	+7343671	+3240000	+0002000	+0000000
+3552244	+3245307	+6740000	+0001000	+0000000
+4265341	+6304561	+1620000	+0000400	+0000000
+6310004	+7025035	+1640000	+0000200	+0000000
+2620000	+2515710	+3500000	+0000100	+0000000
+3560354	+1575322	+5720000	+0000040	+0000000
+0311377	+6426076	+1040000	+0000020	+0000000
+4621413	+6372367	+0640000	+0000010	+0000000
+6060175	+5107103	+1160000	+0000004	+0000000
+4654602	+3426564	+3620000	+0000002	+0000000
+5150766	+6322633	+3160000	+0000001	+0000000
+7566725	+7617451	+6140000	+0000000	+4000000
+1461347	+4407562	+4660000	+0000000	+2000000
+2467523	+7373620	+6300000	+0000000	+1000000
+1524636	+2510027	+2640000	+0000000	+0400000
+6112305	+5443117	+2520000	+0000000	+0200000
+6015777	+6473743	+7620000	+0000000	+0100000
+4722463	+7120367	+5200000	+0000000	+0040000
+4322236	+0527262	+4160000	+0000000	+0020000
+4243514	+3042413	+1360000	+0000000	+0010000
+3721266	+5551473	+7640000	+0000000	+0004000
+7017526	+2547631	+5040000	+0000000	+0002000
+5431021	+0636051	+1500000	+0000000	+0001000
+4503030	+5601257	+3340000	+0000000	+0000400
+3107150	+1676127	+3540000	+0000000	+0000200
+7235541	+1651233	+2440000	+0000000	+0000100
+7235501	+6624560	+6060000	+0000000	+0000040

Figure 26 - Coding (and Decoding) Matrix [P] for a (100, 50) Code

30102	30103	30104	30105	30106
D+4000000	D+0000000	D+0017016	D+4240676	D+6651300
+2000000	+0000000	+0010754	+4520014	+0422140
+1000000	+0000000	+0006412	+1111176	+6250400
+0400000	+0000000	+0012700	+0433105	+3336440
+0200000	+0000000	+0002530	+1142055	+2251300
+0100000	+0000000	+0003777	+3217530	+3211600
+0040000	+0000000	+0004660	+3615015	+6466700
+0020000	+0000000	+0005410	+6261044	+3432340
+0010000	+0000000	+0006064	+7043324	+7261340
+0004000	+0000000	+0001546	+6334523	+1261500
+0002000	+0000000	+0015053	+5207141	+1615500
+0001000	+0000000	+0015500	+3506431	+4653300
+0000400	+0000000	+0000310	+3665476	+5107340
+0000200	+0000000	+0004676	+3734705	+2732600
+0000100	+0000000	+0016115	+7461442	+1515140
+0000040	+0000000	+0012257	+6350673	+1176540
+0000020	+0000000	+0011775	+3437243	+7370200
+0000010	+0000000	+0000110	+6072217	+6274400
+0000004	+0000000	+0016420	+4456022	+0216240
+0000002	+0000000	+0007644	+5430314	+7123240
+0000001	+0000000	+0000672	+3570325	+2256240
+0000000	+4000000	+0014551	+2271330	+1534300
+0000000	+2000000	+0014314	+1544354	+0245400
+0000000	+1000000	+0012523	+2433305	+4640540
+0000000	+0400000	+0017023	+2761032	+6576700
+0000000	+0200000	+0005036	+4160540	+6405040
+0000000	+0100000	+0005027	+0427557	+5141300
+0000000	+0040000	+0015562	+7055743	+4046340
+0000000	+0020000	+0002562	+6120337	+4775040
+0000000	+0010000	+0016032	+1305225	+7046740
+0000000	+0004000	+0014016	+7260577	+2032640
+0000000	+0002000	+0013322	+2612543	+4102540
+0000000	+0001000	+0004425	+0354357	+6456440
+0000000	+0000400	+0002614	+6726646	+3670740
+0000000	+0000200	+0004036	+4256365	+0621540
+0000000	+0000100	+0007441	+2600751	+2611640
+0000000	+0000040	+0003060	+1326262	+7453140
+0000000	+0000020	+0006316	+0655012	+7200400
+0000000	+0000010	+0006505	+4135001	+2431140
+0000000	+0000004	+0005170	+0757406	+4042140
+0000000	+0000002	+0003050	+1065323	+4213140
+0000000	+0000001	+0007645	+1776247	+4630600
+0000000	+0000000	+4004425	+3171166	+1407700
+0000000	+0000000	+2005725	+5074405	+3664040
+0000000	+0000000	+1011670	+5403771	+1336200
+0000000	+0000000	+0406653	+2257053	+1610440
+0000000	+0000000	+0217304	+3710354	+2502440
+0000000	+0000000	+0110605	+0266360	+2214300
+0000000	+0000000	+0054177	+0537653	+2003640
+0000000	+0000000	+0033655	+4644523	+3340040

Figure 27 - Decoding Matrix [Q_a] for the (100, 50) Code

30102	30103	30104	30105	30106
D+2646200	D+5600000	D+0000000	D+0000771	D+0062600
+6572025	+5100000	+0000000	+0000445	+0671540
+2756776	+6040000	+0000000	+0000620	+7277740
+2430716	+6020000	+0000000	+0000732	+5322240
+6646050	+4410000	+0000000	+0000606	+6000000
+4275144	+0004000	+0000000	+0000320	+1440100
+1027542	+1002000	+0000000	+0000274	+3740140
+4327442	+4401000	+0000000	+0000240	+2542240
+6401365	+7400400	+0000000	+0000233	+6540700
+5006434	+5000200	+0000000	+0000432	+1735300
+7631360	+3400100	+0000000	+0000150	+4607200
+1406572	+3400040	+0000000	+0000317	+3571440
+4031714	+2400020	+0000000	+0000757	+7020540
+6420207	+2400010	+0000000	+0000352	+5024340
+6337063	+1400004	+0000000	+0000304	+2440000
+7560350	+1000002	+0000000	+0000475	+1426240
+4416270	+6400001	+0000000	+0000306	+3005740
+2475136	+2400000	+4000000	+0000656	+4503740
+5433721	+5000000	+2000000	+0000272	+7634700
+1511221	+0000000	+1000000	+0000344	+3267240
+0134217	+7000000	+0400000	+0000540	+3527540
+1745436	+7000000	+0200000	+0000330	+3677700
+4246452	+6000000	+0100000	+0000256	+4251340
+4622114	+2400000	+0040000	+0000561	+6756040
+4255433	+5000000	+0020000	+0000505	+7577100
+6712124	+5000000	+0010000	+0000671	+1276200
+4221412	+5000000	+0004000	+0000716	+2745300
+3142664	+6400000	+0002000	+0000231	+4232100
+4656160	+1400000	+0001000	+0000330	+3160440
+0365574	+6400000	+0000400	+C000241	+2057440
+6162650	+3000000	+0000200	+0000504	+5711040
+5121773	+2000000	+0000100	+0000436	+7613200
+1256162	+0000000	+0000040	+0000516	+5442700
+4420645	+6000000	+0000020	+0000154	+3127300
+3757663	+5000000	+0000010	+0000127	+0516700
+4111412	+6000000	+0000004	+0000142	+3553740
+6035312	+7400000	+0000002	+0000507	+6525000
+7256475	+0000000	+0000001	+0000144	+4134100
+7752202	+5000000	+0000000	+4000520	+2622440
+3321772	+3400000	+0000000	+2000213	+3751340
+6265227	+4000000	+0000000	+1000111	+5451640
+5270652	+7400000	+0000000	+0400370	+4436000
+4130135	+3000000	+0000000	+0200766	+7247300
+1123125	+3400000	+0000000	+0100417	+5363240
+3236066	+5400000	+0000000	+0040134	+2135040
+2425315	+0400000	+0000000	+0020222	+4271400
+6717022	+7400000	+0000000	+0010371	+4757640
+7036352	+5400000	+0000000	+0004426	+4400340
+5301721	+3000000	+0000000	+0002052	+2411400
+2335153	+7400000	+0000000	+0001447	+0726700

Figure 28 - Decoding Matrix $[Q_s]$ for the (100, 50) Code

	2646200 54		7710062 60
	6572025 50		4450671 54
	2756776 60		6207277 74
	2430716 60		7325322 24
	6646050 44		6066000 00
	4275144 00		3201440 10
	1027542 10		2743740 14
	4327442 44		2402542 24
	6401365 74		2336540 70
	5006434 50		4321735 30
	7631360 34		1504607 20
	1406572 34		3173571 44
	4031714 24		7577020 54
$R_1M_1=$	6420207 24	$M_2=$	3525024 34
	6337063 14		3042440 00
	7560350 10		4751426 24
	4416270 64		3063005 74
	2475136 24		6564503 74
	5433721 50		2727634 70
	1511221 00		3443267 24
	0134217 70		5403527 54
	1745436 70		3303677 70
	4246452 60		2564251 34
	4622114 24		5616756 04
	4255433 50		5057577 10

	6712124 50		6711276 20
	4221412 50		7162745 30
	3142664 64		2314232 10
	4656160 14		3303160 44
	0365574 64		2412057 44
	6162650 30		5045711 04
	5121773 20		4367613 20
	1256162 00		5165442 70
	4420645 60		1543127 30
$M_1=$	3757663 50	$R_2M_2=$	1270516 70
	4111412 60		1423553 74
	6035312 74		5076525 00
	7256475 00		1444134 10
	7752202 50		5202622 44
	3321772 34		2133751 34
	6265227 40		1115451 64
	5270652 74		3704436 00
	4130135 30		7667247 30
	1123125 34		4175363 24
	3236066 54		1342135 04
	2425315 04		2224271 40
	6717022 74		3714757 64
	7036352 54		4264400 34
	5301721 30		0522411 40
	2335153 74		4470726 70

Figure 29 - Individual Components of [P]

$R_1 =$	$R_2 =$
241720700	676420530
241342134	640410160
667134264	170445270
127130500	303416550
701505424	224434364
600735300	605524470
555113050	011740070
221241660	576664144
130633530	464265124
201721270	043311704
362053000	100117560
143005530	242065414
314703264	355425504
324260564	655262714
247403674	721153454
142404324	074256304
372247770	353727234
221254744	102227110
275264360	012733300
473426014	630021260
332551274	407763240
754217440	432161260
430241330	616764370
607742574	707562650
572663220	512617570

$M_1^{-1} =$	$M_2^{-1} =$
740721200	611342344
436622500	052072350
320504444	233620720
534002154	372645364
125404610	054174210
177755074	754756150
233017064	216206234
260431304	055350744
303234214	645466634
066331560	437123430
642565034	017345154
664016430	767441460
014417324	220056550
233717560	106236524
704676304	167707744
512771640	240757240
477656174	256545034
004430350	105026274
721022270	323167750
372226140	317463210
033516740	474122320
626451344	402536670
614606620	574256730
525152154	522466510
741153704	451341414

Figure 30 - Related 25 x 25 Matrices

REFERENCES

1. Metzner, J.J., "Two Classes of Codes for Improving Feedback Communication Efficiency," New York University, Tenth Scientific Report, Contract AF 19(604)-6168, June 15, 1962 UNCLASSIFIED.

2. Slepian, D., "A Class of Binary Signalling Alphabets," Bell Sys. Tech. J., vol. 35 (1956), pp. 203-234.

3. Ledley, R.S., Digital Computer and Control Engineering, McGraw-Hill Book Co., New York (1960), p. 213.

4. Williams, J.D., The Compleat Strategyst, McGraw-Hill Book Co., New York (1954), pp. 219-225.

5. Hildebrand, Methods of Applied Mathematics, Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1952), p. 21.

6. Hildebrand, op. cit., pp. 1-2.

DISTRIBUTION LIST

LIST A

Code	Organization	Copy No.
AF 5	AFMTC (AFMTC Tech Library-MU-135) Patrick AFB, Fla.	1
AF 18	AUL Maxwell AFB, Ala.	2
AF 32	OAR (RROS, Col. John R. Fowler) Tempo D 4th and Independence Ave., Washington 25, D.C.	3
AF 33	AFOSR, OAR (SRYP) Tempo D 4th and Independence Ave., Washington 25, D.C.	4
AF 43	ASD (ASAPRD - Dist) Wright-Patterson AFB, Ohio	5
AF 124	RADC (RAYLD) Griffiss AFB, New York Attn: Documents Library	6
AF 139	AF Missile Development Center (MDGRT) Holloman AFB, New Mexico	7
AF 314	Hq. OAR (RROSP, Maj. Richard W. Nelson) Washington 25, D.C.	8
AF 318	ARL (ARA-2) Library AFL 2292, Building 450 Wright-Patterson AFB, Ohio	9
Ar 5	Commanding General USASRDL Ft. Monmouth, N. J. Attn: Tech Doc. Ctr. SIGRA/SL-ADT	10
Ar 9	Department of the Army Office of the Chief Signal Officer Washington 25, D.C. Attn: SIGRD-4a-2	11
Ar 50	Commanding Officer Attn: ORDTL-012 Diamond Ordnance Fuze Laboratories Washington 25, D.C.	12
Ar 67	Redstone Scientific Information Center U.S. Army Missile Command Redstone Arsenal, Alabama	13
G 31	Office of Scientific Intelligence Central Intelligence Agency 2430 E Street, N.W. Washington 25, D.C.	14

Code	Organization	Copy No.
C 2	ASTIA (TIPAA) Arlington Hall Station Arlington 12, Virginia	15-24
G 68	Scientific and Technical Information Facility Attn: NASA Representative (S-AK/DL) P.O. Box 5700 Bethesda, Maryland	25
G 109	Director Langley Research Center National Aeronautics and Space Administration Langley Field, Virginia	26
N 9	Chief, Bureau of Naval Weapons Department of the Navy Washington 25, D.C. Attn: DLI-31	27-28
N 29	Director (Code 2027) U.S. Naval Research Laboratory Washington 25, D.C.	29-30
I 292	Director, USAF Project RAND The Rand Corporation 1700 Main Street Santa Monica, California THRU: AF Liaison Office	31
M 6	AFCRL, OAR (CRXRA - Stop 39) L. G. Hanscom Field Bedford, Mass	32-51
AF 253	Technical Information Office European Office, Aerospace Research Shell Building, 47 Cantersteen Brussels, Belgium	52
Ar 107	U. S. Army Aviation Human Research Unit U. S. Continental Army Command P. O. Box 428, Fort Rucker, Alabama Attn: Maj. Arne H. Eliasson	53
G 8	Library Boulder Laboratories National Bureau of Standards Boulder, Colorado	54-55
M 65	Institute of the Aerospace Sciences, Inc. 2 East 64th Street New York 21, New York Attn: Librarian	56

Code	Organization	Copy No.
M 64	AFCRL, OAR (CRXR, J. R. Marple L. G. Hanscom Field Bedford, Mass.	57
N 73	Office of Naval Research Branch Office, London Navy 100, Box 39 F.P.O., New York, New York	58-66
U 32	Massachusetts Institute of Technology Research Laboratory Building 26, Room 327 Cambridge 39, Massachusetts Attn: John H. Hewitt	67
U 431	Alderman Library University of Virginia Charlottesville, Virginia	68
G 9	Defence Research Member Canadian Joint Staff 2450 Massachusetts Avenue, N.W. Washington 8, D.C.	69
G 108	Office of the U.S. Research and Development Coordinator SHAPE Air Defense Technical Center P. O. Box 174, The Hague, Netherlands Attn: Mr. B. Barrow	70
U 441	Massachusetts Institute of Technology Lincoln Laboratory Post Office Box 73 Lexington 73, Mass. Attn: Mr. H. Sherman	71

LIST B-E

Code	Organization	Copy No.
AF 2	ADC (ADLAN-C) Ent AFB, Colorado	72
AF 19	AMS (AMSRS/TIPD) Scott AFB, Illinois	73
AF 34	SAC Offutt AFB, Nebraska	74
AF 35	TAC Langley AFB, Virginia	75
AF 57	USAF Security Service (ESD) Attn: Directorate of Systems Engineering ESC/Communications-Electronics San Antonio, Texas	76
AF 55	WADD (WCLRK) Wright-Patterson AFB, Ohio	77
AF 76	Air Proving Ground Command SAGE Project Office (Building 4) c/o RAND Corporation P.O. Box 98 Lexington 73, Massachusetts	78
AF 153	AF Ballistic Missile Division (WDSOT) Air Force Unit Post Office Los Angeles 45, California	79
AF 179	AFOSR (Dr. Harold Wooster) Director Research Information Office Washington 25, D.C.	80
AF 225	RADC (RCUMT) Griffiss AFB, New York	81
AF 332	WADD ([illegible]) Wright-Patterson AFB, Ohio	82
Ar 4	Director Evans Signal Laboratory Belmar, New Jersey Attn: Mr. S. Grossman	83
Ar 20	Chief, U. S. Army Security Agency Arlington Hall Station Arlington 12, Virginia Attn: ACOFS, G4, TE Section	84
Ar 88	Commanding General U. S. Army Electronic Proving Ground Fort Huachuca, Arizona Attn: Technical Library, SIGPG-DH	85
Ar 111	USASRDL (SIGRA/SLNDC, Mr. J. Barton) Ft. Monmouth, New Jersey	86
Ar 121	USASRDL (SIGRA/SL-SRC, Mr. O. Daiano) Ft. Monmouth, New Jersey	87
G 41	Federal Aviation Agency Bureau of Research and Development Washington 25, D. C.	88
G 75	Director National Security Agency Fort George G. Meade, Maryland Attn: C3/TDL	89

Code	Organization	Copy No.
G 109	Director Langley Research Center National Aeronautics and Space Administration Langley Field, Virginia	90
I 33	ADCOM 238 Main Street Cambridge, Mass. Attn: Phillip Brilo	91
I 38	Signatron, Inc. Miller Building 594 Marret Road Lexington 73, Mass. Attn: Dr. Julian Bussgang	92
I 51	HRB Singer, Inc. 1517 Science Avenue State College, Pa.	93
I 76	Autonetics 9150 E. Imperial Highway Downey, California Attn: Technical Library, 3[illegible]	94
I 96	Sandia Corporation Sandia Base, P. O. Box 5800 Albuquerque, New Mexico Attn: Classified Document Division	95
I 126	Convair, A Division of General Dynamics Corp Fort Worth, Texas Attn: K. G. Brown, Division Research Librarian	96
I 130	Parke Mathematical Labs, Inc. Bedford Road, Carlisle, Mass. Attn: Dr. Nathan Grier Parke, III	97
I 190	Motorola, Inc. Phoenix Research Laboratory 3102 No. 56th Street Phoenix, Arizona Attn: Technical Librarian	98
I 194	Rand Corporation 1700 Main Street Santa Monica, California Attn: Dr. A. L. Hiebert	99
I 228	Philco Corporation Research Division, Plant No. 50 4700 Wissahickon Avenue Philadelphia 44, Pa. Attn: S. Zebrowitz	100

Code	Organization	Copy No.
I 921	Litton Systems, Inc. 221 Crescent Street Waltham, Massachusetts Attn: Dr. David Van Meter	111
I 982	The Mitre Corp. Middlesex Turnpike Bedford, Mass.	112
I 997	Alton T. Butson 8030 109th Street, N.E. Bellevue, Washington	113
M 35	ESD (ESRDC, Directorate of Technology) Applied Research Division L.G. Hanscom Field Bedford, Massachusetts	114
N 37	Office of Naval Research Department of the Navy Washington 25, D.C. Attn: Code 427, Electronics Branch	115
N 57	Chief, Bureau of Ships Department of the Navy Washington 25, D. C. Attn: Mr. D. Reed, Code 685	116
N 59	U.S. Naval Air Development Center Johnsville Pennsylvania Attn: Librarian	117
N 85	Commanding Officer and Director U.S. Navy Electronics Laboratory (Library) San Diego 52 California	118
U 26	Massachusetts Institute of Technology Lincoln Laboratory P.O. Box 73 Lexington 73, Mass. Attn: Mary A. Granese, Librarian	119
U 37	University of Michigan Engineering Research Institute Willow Run Laboratories Willow Run Airport Ypsilanti, Michigan Attn: Librarian	120
U [illegible]	Northeastern University Electronics Research Laboratory Boston, Massachusetts Attn: M. W. Essigmann	121

Code	Organization	Copy No.
I 238	General Electric Company Knolls Research Laboratory 1 River Road Schenectady, New York Attn: Dr. R. L. Shuey	101
I 266	ITT Federal Laboratories Technical Library 500 Washington Avenue Nutley 10, New Jersey	102
I 291	Radio Corporation of America Defense Electronic Products Building 10, Floor 7 Camden 2, New Jersey Attn: Mr. Harold J. Schrader, Staff Engineer Organization of Chief Technical Administrator	103
I 312	STL Technical Library Document Acquisitions Space Technology Laboratories, Inc. P.O. Box 95001 Los Angeles 45, California	104
I 376	Hughes Aircraft Company Culver City, California Attn: Dr. S. G. Lutz Communication and Miniaturization Department	105
I 445	The Rand Corp. 1700 Main Street Santa Monica, California Attn: Dr. R. E. Kalaba	106
I 561	Melpar, Inc. 11 Galen Street Watertown, Massachusetts Attn: Librarian	107
I 674	Stanford Research Institute Menlo Park, California Attn: W. R. Vincent	108
I 675	Hazeltine Research Corp. 59-25 Little Neck Parkway Little Neck 62, New York Attn: Donald Richman	109
I 784	Sylvania Electronic Systems Applied Research Laboratory 100 First Avenue Waltham, Massachusetts Attn: Dr. Seymour Stein	110

Code	Organization	Copy No.
U 6[illegible]	New York University College of Engineering Department of Electrical Engineering University Heights, Bronx 53, New York Attn: Dr. Robert F. Cotellessa	122
U 144	Jet Propulsion Laboratory California Institute of Technology 4800 Oak Grove Drive Pasadena 3, California Attn: Dr. Eberhardt Rechtin	123
U 1[illegible]	California Institute of Technology Jet Propulsion Laboratory 4800 Oak Grove Drive Pasadena 3, California Attn: Mr. I. E. Newlan, Manager Technical Reports Section	124
U 286	University of Michigan Office of Research Administration North Campus, Ann Arbor, Michigan Attn: Gordon Roberts	125
U 97	Polytechnic Institute of Brooklyn Microwave Research Institute 55 Johnson Street Brooklyn, New York (Attn: Prof. E.J. Smith)	126
U 396	Syracuse University Syracuse 10, New York Attn: Prof. F. Reza	127
U 406	Massachusetts Institute of Technology Department of Electrical Engineering Cambridge, Mass. (Attn: Mr. Jack Cowan)	128
U 408	Johns Hopkins University School of Engineering 34th and Charles Baltimore 18, Maryland (Attn: W. H. Huggins)	129
U 411	Montana State College Electronics Research Laboratory Bozeman, Montana (Attn: Dr. W. Kilmer)	130
U 395	Advanced Electronics Center at Cornell University General Electric Company Ithaca, New York (Attn: Dr. Robert Turner)	131
	Hq. AFCRL, Office of Aerospace Research (CRRC) L. G. Hanscom Field, Bedford, Mass.	132
	Hq. AFCRL, Office of Aerospace Research (CRRS) L.G. Hanscom Field, Bedford, Mass.	133-142

AD________________ Accession No.________________

Electronics Research Directorate, Air Force Cambridge Research Laboratories, Office of Aerospace Research, United States Air Force, Bedford, Massachusetts.

AFCRL-63-28

COMPUTER CONSTRUCTION AND EVALUATION OF LONG BURST-ERROR CORRECTING CODES

Twelfth Scientific Report, 31 January 1963, 89 + vii pp. including 30 figures, 6 references; distribution list.

UNCLASSIFIED REPORT

This report presents a specific method of programming a digital computer to construct and evaluate codes suitable for use in a burst-error correction decoding scheme described in the Tenth Scientific Report. Two (100,50) codes of this type have been constructed using the methods described. One is found capable of correcting uniquely all bursts of length 21 or less.

The method used to evaluate burst-error correcting capabilities is applicable to any group code, not just the class under consideration. Memory limitations have restricted the present program to code lengths not exceeding 105 digits, of which at most 63 may be check digits.

1. Computers
2. Communication Systems
3. Coding

I. Project 4610, Task 461003
II. Contract AF19(604)-6168
III. New York University, College of Engineering, Department of Electrical Engineering. University Heights, New York 53, New York.
IV. Metzner, J. J.
V. In ASTIA collection

AD________________ Accession No.________________

Electronics Research Directorate, Air Force Cambridge Research Laboratories, Office of Aerospace Research, United States Air Force, Bedford, Massachusetts.

AFCRL-63-28

COMPUTER CONSTRUCTION AND EVALUATION OF LONG BURST-ERROR CORRECTING CODES

Twelfth Scientific Report, 31 January 1963, 89 + vii pp. including 30 figures, 6 references; distribution list.

UNCLASSIFIED REPORT

This report presents a specific method of programming a digital computer to construct and evaluate codes suitable for use in a burst-error correction decoding scheme described in the Tenth Scientific Report. Two (100,50) codes of this type have been constructed using the methods described. One is found capable of correcting uniquely all bursts of length 21 or less.

The method used to evaluate burst-error correcting capabilities is applicable to any group code, not just the class under consideration. Memory limitations have restricted the present program to code lengths not exceeding 105 digits, of which at most 63 may be check digits.

1. Computers
2. Communication Systems
3. Coding

I. Project 4610, Task 461003
II. Contract AF19(604)-6168
III. New York University, College of Engineering, Department of Electrical Engineering. University Heights, New York 53, New York.
IV. Metzner, J. J.
V. In ASTIA collection

AD________________ Accession No.________________

Electronics Research Directorate, Air Force Cambridge Research Laboratories, Office of Aerospace Research, United States Air Force, Bedford, Massachusetts.

AFCRL-63-28

COMPUTER CONSTRUCTION AND EVALUATION OF LONG BURST-ERROR CORRECTING CODES

Twelfth Scientific Report, 31 January 1963, 89 + vii pp. including 30 figures, 6 references; distribution list.

UNCLASSIFIED REPORT

This report presents a specific method of programming a digital computer to construct and evaluate codes suitable for use in a burst-error correction decoding scheme described in the Tenth Scientific Report. Two (100,50) codes of this type have been constructed using the methods described. One is found capable of correcting uniquely all bursts of length 21 or less.

The method used to evaluate burst-error correcting capabilities is applicable to any group code, not just the class under consideration. Memory limitations have restricted the present program to code lengths not exceeding 105 digits, of which at most 63 may be check digits.

1. Computers
2. Communication Systems
3. Coding

I. Project 4610, Task 461003
II. Contract AF19(604)-6168
III. New York University, College of Engineering, Department of Electrical Engineering. University Heights, New York 53, New York.
IV. Metzner, J. J.
V. In ASTIA collection

AD________________ Accession No.________________

Electronics Research Directorate, Air Force Cambridge Research Laboratories, Office of Aerospace Research, United States Air Force, Bedford, Massachusetts.

AFCRL-63-28

COMPUTER CONSTRUCTION AND EVALUATION OF LONG BURST-ERROR CORRECTING CODES

Twelfth Scientific Report, 31 January 1963, 89 + vii pp. including 30 figures, 6 references; distribution list.

UNCLASSIFIED REPORT

This report presents a specific method of programming a digital computer to construct and evaluate codes suitable for use in a burst-error correction decoding scheme described in the Tenth Scientific Report. Two (100,50) codes of this type have been constructed using the methods described. One is found capable of correcting uniquely all bursts of length 21 or less.

The method used to evaluate burst-error correcting capabilities is applicable to any group code, not just the class under consideration. Memory limitations have restricted the present program to code lengths not exceeding 105 digits, of which at most 63 may be check digits.

1. Computers
2. Communication Systems
3. Coding

I. Project 4610, Task 461003
II. Contract AF19(604)-6168
III. New York University, College of Engineering, Department of Electrical Engineering. University Heights, New York 53, New York.
IV. Metzner, J. J.
V. In ASTIA collection

The Research Division of the College of Engineering is an integral part of the educational program of the College. The faculty of the College takes part in the work of the Research Division, often serving as co-ordinators or project directors or as technical specialists on the projects. This research activity enriches the educational experience of their students since it enables the faculty to be practicing scientists and engineers, in close touch with developments and current problems in their field of specialization. At the same time, this arrangement makes available to industrial and governmental sponsors the wealth of experience and special training represented by the faculty of a major engineering college. The staff of the Division is drawn from many areas of engineering and research. It includes men formerly with the research divisions of industry, governmental and public agencies, and independent research organizations.

Following are the areas represented in the research program: Aeronautical Engineering, Chemical Engineering, Civil Engineering, Electrical Engineering, Engineering Mechanics, Industrial and Management Engineering, Mechanical Engineering, Metallurgical Engineering, Mathematics, Meteorology and Oceanography, and Physics. In addition, an interdisciplinary research group is responsible for studies which embrace several disciplines. Inquiries regarding specific areas of research may be addressed to the Director, Research Division for forwarding to the appropriate research group.