ASD-TR-78-17

LEVEL II

23

B-1 EMUX AUTOMATED LOGIC DIAGRAMMER DEMONSTRATION PROGRAM

RONALD B. BERGER

MAY 1978

TECHNICAL REPORT ASD-TR-78-17
Final Report for Period July 1975 to September 1977

Approved for public release; distribution unlimited.

ELECTRICAL POWER BRANCH
AVIONICS DIRECTORATE
DEPUTY FOR ENGINEERING
AERONAUTICAL SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

NOTICE

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

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

This technical report has been reviewed and is approved for publication.

W. W. RADLINGER, LT COL, USAF
Acting Chief, Avionics Division
Strategic Systems Program Office

DAVID R. GORDON
Acting Chief, Electrical Power Branch
Directorate, Avionics Engineering
Aeronautical Systems Division

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify ASD/YYEA, W-PAFB, OH 45433 to help us maintain a current mailing list".

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

AIR FORCE/56780/3 July 1978 — 100

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

(14) 1. REPORT NUMBER: ASD-TR-78-17

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

(6) 4. TITLE (and Subtitle): B-1 EMUX AUTOMATED LOGIC DIAGRAMMER DEMONSTRATION PROGRAM.

(9) 5. TYPE OF REPORT & PERIOD COVERED: Final Rept. July 1975 — September 1977

6. PERFORMING ORG. REPORT NUMBER

(10) 7. AUTHOR(s): Ronald B. Berger

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Electrical Power Branch (ASD/ENACD)
Directorate of Avionics Engineering
Aeronautical Systems Division
Air Force Systems Command, WPAFB, OH 45433

(16) 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 139A2

11. CONTROLLING OFFICE NAME AND ADDRESS
Avionics Division, Strategic Systems Program Office (ASD/YYEA)
WPAFB, OH 45433

(11) 12. REPORT DATE: May 1978

13. NUMBER OF PAGES: 49

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

(12) 53p.

15. SECURITY CLASS. (of this report): Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

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

18. SUPPLEMENTARY NOTES

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

Computer program, automated logic diagramming, logic diagrams.

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

The B-1 aircraft Electrical Multiplex System (EMUX) is a programmable computer network which provides control of aircraft electrical loads, provides automatic electrical load management, and provides digital data transfer. Boolean (or logic) equations are used to program the system. A computer program was developed in-house to demonstrate the feasibility of automatically creating logic diagrams of the boolean equations directly from the same EMUX computer files used to program the system. Program BOLD (B-1 (One) Logic Diagrammer), the result of this study, is documented in this report.

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

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

INTRODUCTION

The B-1 aircraft, developed by the B-1 Division of Rockwell International, utilizes an Electrical Multiplex System (EMUX) for:

- processing and transfer of serial-digital and discrete data throughout the aircraft
- power control of most electrical loads
- automatic electrical load management

There are two independent EMUX systems in the aircraft for redundancy - each having its own configuration and software. EMUX is programmed using an IBM 370 hosted compiler. Input to the compiler consists of logic equations, assembly language instructions, and EMUX system configuration data. EMUX can be programmed to output over two thousand signals as functions of over five thousand input signals. This magnitude of equations and signals complicates the understanding and analysis of the EMUX software.

In order to improve the Air Force EMUX software analysis capability, several in-house computer routines were developed from July 1975 to September 1977. All of these routines use copies of the same data files created and maintained by the B-1 Division for use in generating the EMUX flight software. All of the routines were developed on Aeronautical Systems Division's CDC 6600 Computer System using FORTRAN EXTENDED.

One ASD Technical Report and four Avionics Directorate (ENA) Engineering Reports have been written to document the various computer programs:

"B-1 EMUX Data Tape Conversion Software"
ASD-ENA-77-20

"B-1 EMUX Usage Analysis Software"
ASD-ENA-78-2

"B-1 EMUX Signal-Signal and Signal-Box Relationship Analysis Software"
ASD-ENA-78-3

"B-1 EMUX Logic Equation Regeneration Software"
ASD-ENA-78-4

"B-1 EMUX Automated Logic Diagrammer Demonstration Program"
ASD-TR-78-17

This report discusses the FORTRAN software written for use on the CDC 6600 to demonstrate the feasibility of automatically creating logic diagrams of boolean equations directly from the contractor maintained computer data files. Diagrams, while being logically equivalent to equations, are pictorial representations of the equations and, as such, facilitate understanding. However, logic diagrams are quite time-consuming to draw manually; estimates upwards of two or more man-years for a single version of EMUX software have been given. Thus, a computerized diagramming technique which did not require special input data could provide cost and time savings if the automated diagrams were of equal quality (accuracy, readability, layout, etc.) as manual diagrams. Program BOLD (B-1 (One) Logic Diagrammer) is the result of this feasibility demonstration.

PROGRAM DESIGN GROUNDRULES

Program BOLD was written in FORTRAN EXTENDED for use on Aeronautical Systems Division's CDC 6600 computer system using the following ground rules:

- the layout of the diagrams, expecially the positioning of the logic operator symbols, must be of equal or better quality when compared to manually drawn diagrams.

- any valid logic equation for EMUX must be diagrammable, including equations with time delays.

- the diagrams must accurately reflect the boolean equation logic.

- preparation of special input data must be minimized.

- program size, run time, and resultant cost must be less than manual costs.

All of these ground rules have been satisfied by BOLD as will be discussed in the following paragraphs.

The proper layout of the diagram is one of the most important measures of the success or failure of any computerized diagramming technique. Proper layout must consider the location of each of the logic operator symbols (gates) and the interconnection of the gates. A major portion of the logic in BOLD is involved in diagram layout. Listed below are the diagram layout rules which were established:

- flow of logic will be from left to right.

- no overlap of gates.

- gates will be arranged in columns (for readability and simplicity) and will be assigned to columns from right to left to minimize length of interconnect lines.

- interconnect lines will be straight whenever possible and will have at most two turns.

- no feedback or latch interconnect lines will be used but will be labeled as operator inputs (this is not inconsistent with EMUX operation but is probably not how the diagram would be done manually).

- crossover of interconnect lines will be minimized.

- time delays will be treated as two input operators. One input is the quantity to be delayed and the other input is the time delay duration.

Figures 1 and 2 are sample BOLD printouts of two different equation diagrams. As can be seen, the diagrams are "drawn" by the standard line printer rather than by a continuous line plotter (e.e., a CALCOMP plotter). This type of output was selected primarily because BOLD was intended to be a feasibility demonstration program rather than a final production-oriented program. As such, diagram output via line printer is simpler to obtain in terms of software logic complexity and program turnaround time. A logical modification to improve diagram readability to BOLD for production use would be the conversion to CALCOMP-type diagrams. While this would require the main program to be changed (approximately 25 to 30% rewrite), the methodology would be the same and no other subroutine changes should be required. At the same time, logic for feedback lines could be included if deemed desirable.

All equations diagrammed to date have accurately reflected the input equation logic. The only problems encountered but not solved were due to either improper input equation syntax or equation size in excess of BOLD capacities. Improper equation syntax (e.g., missing operator, missing parenthesis) would also cause problems for the contractor's EMUX compiler and can only be resolved by correcting the equation. Many data arrays are used in BOLD for storage of various types of equation information. The sizes of these arrays are related to each other to some extent but are somewhat arbitrarily established. There are no known theoretical limits to the equation size which can be processed; however, practical limits of computer memory available and desired turnaround time may prevent some "extreme" equations from being diagrammed. Present array sizes have rejected less than 1/2 of 1% of all equations input. Array size limitations would probably be improved by the conversion to CALCOMP-type diagrams. The present program is limited to a maximum of 8 pages of printout per equation which, when cut and taped together, will show the entire diagram. Limits on particular equation size parameters (e.g., number of parenthesis pairs, number of operators, number of operands per operator) are documented in the program listings.

BOLD uses the same data files maintained by the contractor for EMUX flight software programming. The only additional input required is the user's selection of the equation set to be diagrammed. The user has the flexibility to select individual equations or entire subsystems for diagramming.

While final conclusions on relative costs (automated diagrams versus manual diagrams) cannot be made until a production version of BOLD is created, a preliminary cost analysis resulted significantly in favor of automated diagrams.

PRINTOUT DESCRIPTION

While the BOLD printout, Figures 1 and 2, seems at first cryptic, it is understandable if examined in parts. The first line identifies the OCNEE signal designator (an aircraft standard signal identification scheme) for the equation diagrammed, the aircraft effectivity, and the section of EMUX (left or right) involved. (For Figure 1, the equation output signal designator is 2821-008 and the equation is used in the left section of EMUX on Aircraft 3.) The next three lines on the left identify the aircraft system, subsystem, and subsystem to which the signal is assigned. (For Figure 1, sub-subsystem 2821 is the Internal Transfer portion of the Fuel Distribution System.) The line to the right of the system labels is an abbreviated signal description. (For Figure 1, the signal is a power control signal for the tank 1 transfer pump number 2821PP1.) The next set of lines is the actual boolean equation to be diagrammed as input to BOLD. Following the equation is a table of all signals that are used in the equation. Included with each signal designator is the signal description. The final part of the printout is the diagram. Due to line printer limitations, the diagram is not as readable as possible. Each logic operator (gate) is indicated with a box of asterisks with a letter inside. AND gates use the letter A, OR gates use the letter O, EXCLUSIVE OR gates use the letter X, Type 1 time delays use a dollar sign ($), and Type 2 time delays use a question mark (?). Gates with more than two inputs have the left column of asterisks extended as necessary to accomodate all inputs. NOT'ed gate inputs are indicated with the letter O in place of an asterisk in the gate's leftmost column. Gate interconnecting lines use dashes, periods, and letter I's for their horizontal parts, corners, and vertical parts respectively. The readability of these line printer diagrams can be improved significantly merely by drawing over the connecting lines manually - as is done in Figure 2.

EMUX BOOLEAN EQUATION DIAGRAM FOR SIGNAL 2821-008 OF AIRCRAFT NO. 3 LEFT SIDE OF EMUX

FUEL SUBSYSTEM
DISTRIBUTION
INTERNAL TRANSFER

PWR-K36,TK 1 XFR(2821PP1)

2821-008=((2841A02K*2841A02A+2841L02K*2841L02A*2841A02A')*2821-001'*2821-003'*2821-VAA+2821-001*2841-109'*2425-015')*
2821-V01+2821-001*2821-183*2821-243*2821-303*2821-483*2821-633*2821-783*2821-843*(2841-109+2425-015)

DIAGRAM DESIGNATOR	EMUX SIGNAL DESIGNATOR	SIGNAL SOURCE OR DESTINATION
	2425-015	EMER GEN FLD RELAY CLOSED
	2821-VAA	$04921 AUTO ENAB
	2821-V01	$04968 EMGENLOCK
	2821-001	TK 1 XFR PMP(PP1&2),ON
	2821-003	TK 1 XFR PMP(PP1&2),OFF
	2821-183	TK 2 XFR PMP(PP3&4),OFF
	2821-243	TK 3 XFR PMP(PP5&6),OFF
	2821-303	TK 4 XFR PMP(PP7-9),OFF
	2821-483	L WG PMPS(10,11&18-20)OFF
	2821-633	R WG PMPS(12,13&21-23)OFF
	2821-783	FWD SB XFR PMP(14&15),OFF
	2821-843	INT SB XFR PMP(16&17),OFF
	2841A02A	PID,VALIDITY BIT
	2841A02K	PID,TK 1 ESS PMP ON CMD
	2841L02A	SID,VALIDITY BIT
	2841L02K	SID,TK 1 ESS PMP ON CMD
	2841-109	EMER GEN TEST(PMP SEQ),ON

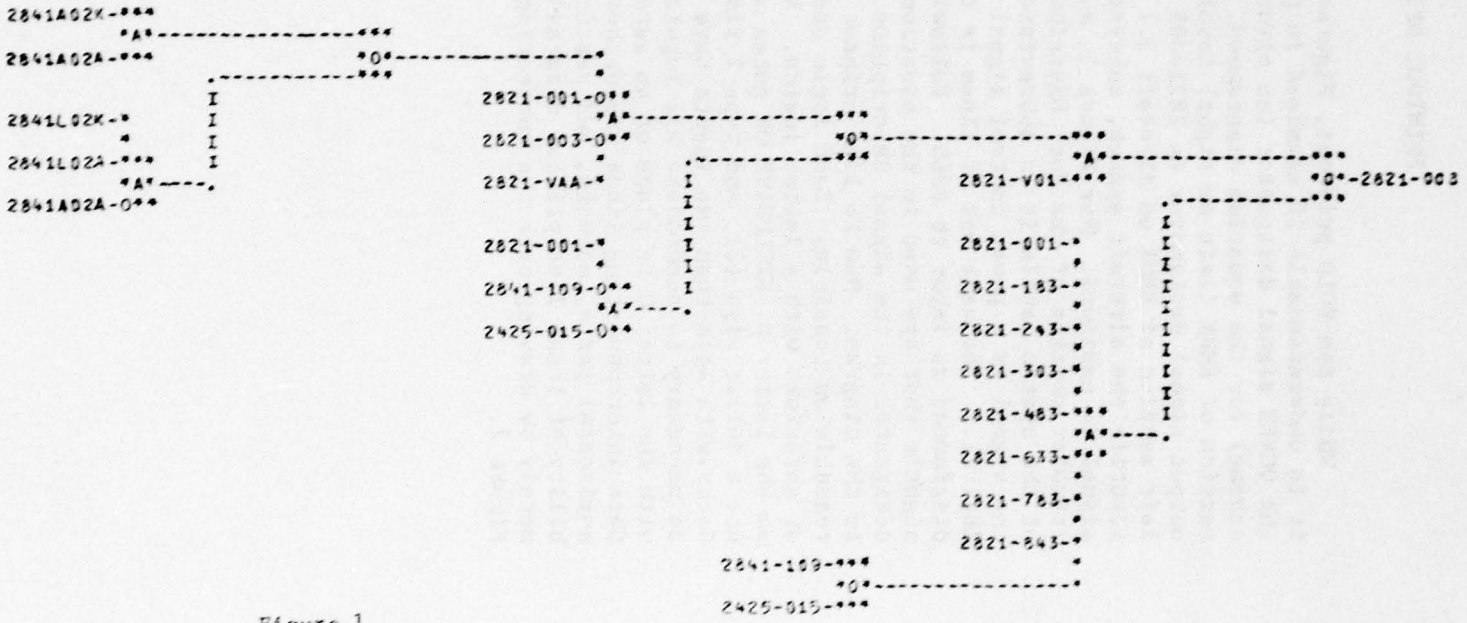


Figure 1

EMUX BOOLEAN EQUATION DIAGRAM FOR SIGNAL 2824-029 OF AIRCRAFT NO. 3 LEFT SIDE OF EMUX

FUEL SUBSYSTEM
DISTRIBUTION
FUEL COOLING LOOP

CLG FUEL LP XOVER ANN LT
TIME DELAY TYPE 2 FOR 10.00 SECONDS

2824-029=(2824-029+2824-001)*(2824-224'+2824-310'+2823-003'*2823-014'+2823-019'*2823-031'+3231-V01')*2824-178'

DIAGRAM DESIGNATOR	EMUX SIGNAL DESIGNATOR	SIGNAL SOURCE OR DESTINATION	DIAGRAM DESIGNATOR	EMUX SIGNAL DESIGNATOR	SIGNAL SOURCE OR DESTINATION
	2823-003	BOOST PMP(PP1)PRESS,PRESS		2824-178	C L XOVER V OP LIM,OPEN
	2823-014	BOOST PMP(PP2)PRESS,PRESS		2824-224	LH C L PMP PRESS,PRESSURE
	2823-019	BOOST PMP(PP3)PRESS,PRESS		2824-310	RH C L PMP PRESS,PRESSURE
	2823-031	BOOST PMP(PP4)PRESS,PRESS		3231-V01	$05827 AIRBORNE
	2824-001	CLG LP XOVER(VL5),NORMAL			

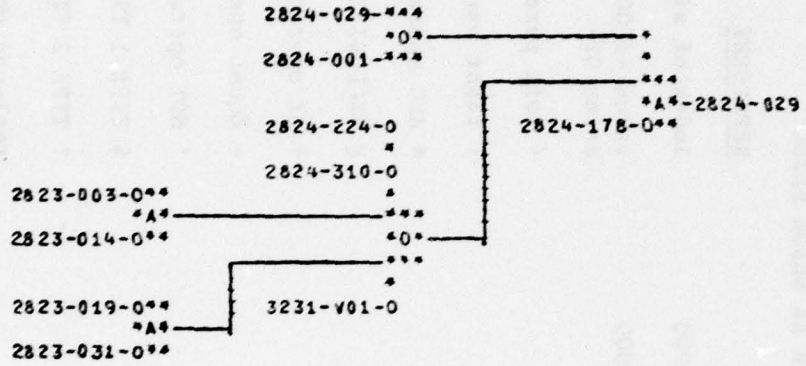


Figure 2

PROGRAM OPERATION

The process used by BOLD, as shown in Figure 3, starts with reading the next equation to be diagrammed into array EQN. In order to simplify processing throughout the rest of the program, the equation is converted into an internal coded integer format and is stored in array EQN1. Array VAR is loaded with signal names appearing in the equation. The integer code is shown below:

VALUES	REPRESENT
1 to 1000	index of signal name stored in VAR
1001 to 2000	value -1000 is the index of an operator in array OPR
2001	(left parenthesis
2002	) right parenthesis
2003	* AND operator
2004	@ EXCLUSIVE OR operator
2005	+ OR operator
2006	= EQUAL operator
2007	' NOT operator
2010	$ TYPE 1 TIME DELAY operator
2011	? TYPE 2 TIME DELAY operator
9999	replaced character
-2006	NOT EQUAL operator
-2000 to -1001	NOT'd operator, absolute value -1000 is the index of an operator in OPR
-1000 to -1	NOT'd operator input, absolute value is the index of a signal name stored in VAR

Due to parentheses and operator precedence, the equation cannot be simply scanned from left to right to determine the correct diagram.

Therefore, BOLD first scans EQN1 for parenthesis pairs and stores information on each pair in array PAR. Once the parentheses have been found, then the logic within each pair is sent to subroutine PARSE for analysis - the innermost pair being done first. As each pair is done, the logic and parentheses are replaced in EQN1 with the resultant output operator's index in OPR and with 9999's. After all parentheses have been removed, the entire equation is sent to PARSE for a final analysis.

PARSE semantically analyzes each portion of the equation, taking into account operator precedence, by scanning left to right five times (once for each operator type). The operator precedence used by PARSE is Type 1 Time Delays, Type 2 Time Delays, AND gates, EXCLUSIVE OR gates, and OR gates. During each of the five passes, the appropriate operator is tested for and, if found, is stored in array OPR along with its associated inputs. Thus, when PARSE is finished with the equation, all of the semantic equation information is stored in array OPR as shown below for operator i:

OPR (i, 1)	operator type code (2003, 2004, 2005, 2010, 2011)
OPR (i, 2)	number of inputs to this operator
OPR (i, 3)	x of x, y coordinates of location of operator on output page
OPR(i, 4)	y of x, y coordinates of location of operator on output page
OPR (i, 5) through OPR (i, 34)	operator input indices. Negative values indicate primed inputs. Absolute values from 1 to 1000 are signal name indices pointing to array VAR. Absolute values from 1001 to 2000 are inputs from other operator outputs. These values -1000 point to array OPR.

The present size of array OPR limits any one operator to 30 inputs maximum. It should be noted that OPR (i, 3) and OPR (i, 4), operator location information, are not defined by BOLD at this point in the equation processing.

The only remaining task before printing out the diagram is the determination of the location of each of the operators on the output page. This portion of the logic within BOLD was designed to:

- minimize crossover of interconnecting lines.
- minimize the number of operators which could not be connected due to the location of other operators

- maximize the use of straight line connections

- make all connections so that the logic flow is from left to right

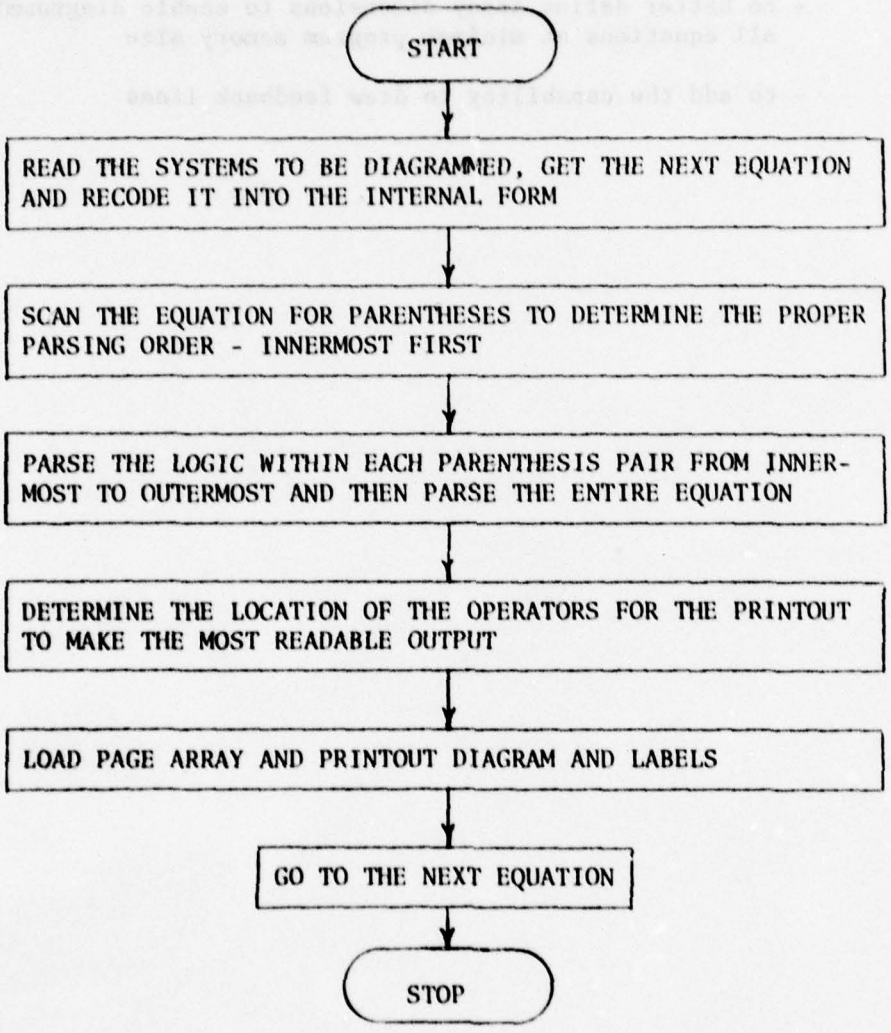
For both readability and ease of programming it was decided that each operator would be assigned to one of several available columns. Once an operator is assigned to a column then its vertical position in the column is determined based on providing the straightest connection from its output to the next operator's input. The operators are assigned to columns from right to left starting with the equation's overall output operator in the rightmost column. Its input operators are assigned to the next column to the left and their input operators are assigned likewise. This process is continued until all operators are assigned columns. Array OPOR is used to keep track of this process. When all operators are assigned columns, then OPR (i, 3) is defined for each operator.

OPR (i, 4) is defined by starting with the first, or top, operator in each column (right to left) and locating the operator if possible so that the connection from this operator to the one in the next column is straight. This is not always possible due to gate overlaps so the operator is moved down the column until no overlap occurs. As each column is finished, the next column to the left is done. This is continued until all columns are finished.

The final step in BOLD is to load the printout array PAGE with the operators, labels, and connection lines. This is a straightforward process once the location of each operator is defined. In order to reduce computer memory requirements, 10 characters per word are packed into PAGE. Present dimensions of PAGE are 25,200. Up to eight pages can be used for diagram output with up to 200 lines per diagram and up to 250 characters per line.

BOLD DIAGRAM

Figure 3



CONCLUSION

Program BOLD has successfully demonstrated the feasibility of producing automated logic diagrams which are of equal quality to manual diagrams, which are less expensive than manual diagrams, and which can be obtained in much less time. Further development is required prior to production use of BOLD:

- to develop logic for utilization of plotter drawn diagrams
- to better define array dimensions to enable diagramming of all equations at minimum program memory size
- to add the capability to draw feedback lines

PROGRAM LISTING

```
      PROGRAM BOLD (INPUT=/80,OUTPUT=400B,TAPE4=400B,TAPE7=400B,
     A    TAPE5=INPUT,TAPE6=OUTPUT)
C
C   THIS PROGRAM (B-1 (ONE) LOGIC DIAGRAMMER) DRAWS LOGIC DIAGRAMS, AS
C                  -    -    -    -
C   SELECTED BY INPUT, FROM RI'S EMUX MASTER FILE FOR ONE SIDE OF ONE
C   AIRPLANE EFFECTIVITY.  THIS VERSION OUTPUTS LINE PRINTER DIAGRAMS
C   ONLY.
C
C   FILE DESCRIPTIONS:
C
C         TAPE4      EMUX MASTER FILE - EQUATIONS FILE (SEQUENTIAL)
C         TAPE5      INPUT DATA CARDS
C         TAPE6      OUTPUT FILE
C         TAPE7      EMUX MASTER FILE - RANDOM ACCESS (SEE PROGRAM RANDOM
C                    DESCRIPTION)
C
C   INPUT DESCRIPTION:
C
C         THE ONLY SET OF INPUT DATA CARDS IS IN LIST DIRECTED FORMAT AND
C         IS USED TO SELECT THE SYSTEM, SUBSYSTEM, OR SUB-SUBSYSTEM TO
C         BE DIAGRAMMED.  THEY ARE SELECTED BY SURROUNDING THE FOUR DIGIT
C         OCNEE NUMBERS WITH QUOTES, BY USING COMMAS AS SEPARATORS, AND
C         BY PLACING A SLASH AT THE END.  SINGLE EQUATIONS CAN ALSO BE
C         SELECTED BY INPUTTING THE SIGNAL DESIGNATOR OF THE DESIRED
C         EQUATIONS AS DESCRIBED ABOVE.  UP TO 40 CAN BE SELECTED.
C         AN ADDITIONAL FORTY-FIRST VALUE, IF INPUT AS NON-ZERO, WILL
C         FORCE THE ERROR PRINTOUT FOR ALL DIAGRAMS.
C
C   OUTPUT DESCRIPTION:
C
C         ONE OR MORE PAGES ARE OUTPUT FOR EACH EQUATION.  IN ADDITION TO
C         THE LOGIC DIAGRAM, THE PRINTOUT WILL CONTAIN THE EQUATION AS
C         READ FROM THE MASTER FILE, A TABLE OF SIGNAL DESCRIPTION
C         INFORMATION FOR THE SIGNALS IN THE EQUATION, AND A HEADING WITH
C         SYSTEM, SUBSYSTEM, AND SUB-SUBSYSTEM NAMES.
C
C   VARIABLE DESCRIPTIONS AND LIMITS:
C
C     NAME     SIZE                  DESCRIPTION
C
C     CARD     80        LAST CARD READ FROM TAPE4.  ONE CHARACTER PER WORD,
C                        LEFT JUSTIFIED WITH BLANK FILL.
C
C     CHAR     1         CURRENT CHARACTER FROM EQN1.  **SEE NOTE 10
C
C     COL      12        NEXT ROW INDICATOR FOR EACH OF THE OPERATOR
C                        COLUMNS IN THE PRINTOUT.  COL(I) CORRESPONDS TO
C                        THE ITH COLUMN FROM THE RIGHT OF THE DIAGRAM.
C                        COL(1) IS THE OUTPUT OPERATOR COLUMN, COL(2) IS THE
C                        COLUMN OF OPERATORS THAT INPUT TO THE OUTPUT
C                        OPERATOR, ... ,COL(NCOL) IS FOR THE LEFTMOST
C                        COLUMN.
C
C     DATA     400       STORAGE ARRAY FOR TAPE7 RECORD READ BY GETDAT.
```

```
C
C     EFF       1         EFFECTIVITY CODE FOR EQUATION.  TWO CHARACTERS LEFT
C                         JUSTIFIED WITH BLANK FILL.
C
C     EQN       4460      CURRENT EQUATION BEING DIAGRAMMED.  ONE CHARACTER
C                         PER WORD LEFT JUSTIFIED WITH BLANK FILL.
C                         **SEE NOTE 1
C
C     EQN1      960       INTERNAL FORM OF EQN.  **SEE NOTES 2,10
C
C     EQUATN    SIZEE     ARRAY FROM TAPE7 RECORD THAT CONTAINS THE EQUATION
C                         CARD COUNT FOR EACH EQUATION ON THE FILE.
C
C     ERR       6         ERROR REDEFINITION ARRAY USED BY SYSTEMC.
C                         **SEE NOTE 9
C
C     IFM       1         LEFT END OF CONNECTION COLUMN.
C
C     II        1         EQN1(II) IS LEFTMOST END OF LOGIC TO BE PARSED.
C
C     IK        1         X OF (X,Y) COORDINATES OF VERTICAL SECTION OF
C                         CONNECTION PATH.
C
C     INDEXM    SIZEM     MASTER INDEX ARRAY FOR TAPE7.
C
C     INPUTS    30        INDICES OF VARIABLES AND OPERATORS FROM EQN1
C                         BEFORE ASSIGNMENT TO OPR.  **SEE NOTES 8,10
C
C     ISYS      1         INPUT SIGNAL TO FORCE ERROR PRINTOUT.
C
C     ITO       1         RIGHT END OF CONNECTION COLUMN.
C
C     JFM       1         Y OF (X,Y) COORDINATES OF LEFT DASH OF OPERATOR
C                         OUTPUT.
C
C     JJ        1         EQN1(JJ) IS RIGHTMOST END OF LOGIC TO BE PARSED.
C
C     JTO       1         Y OF (X,Y) COORDINATES OF RIGHT DASH OF OPERATOR
C                         INPUT.
C
C     KEY       1         INDEX KEY (NAME TYPE) USED TO GET RECORDS FROM
C                         TAPE7.
C
C     LOPR      1         LAST OPERATOR TYPE ENCOUNTERED.  **SEE NOTE 10
C
C     NCOL      1         NUMBER OF COLUMNS IN THE DIAGRAM (ENTRIES IN COL).
C
C     NEQN      1         NUMBER OF CHARACTERS IN EQN.
C
C     NEQN1     1         NUMBER OF ENTRIES IN EQN1.
C
C     NIN       1         NUMBER OF ENTRIES IN INPUTS.
C
C     NOPR      1         NUMBER OF OPERATORS IN OPR.
C
```

```
C     NPAR    1         NUMBER OF PARENTHESIS SETS IN PAR.
C
C     NVAR    1         NUMBER OF NAMES IN VAR.
C
C     OPOR    120,3     OPERATOR DRAWING ORDER INFORMATION.  ORDER IN OPOR
C                       DETERMINES DRAWING ORDER FROM RIGHT TO LEFT AND TOP
C                       TO BOTTOM.
C                               OPOR(I,1) OPR INDEX OF OPERATOR IN THIS
C                                         POSITION.
C                               OPOR(I,2) OPERATOR LEVEL (=1 IS OUTPUT
C                                         OPERATOR, =2 IS INPUT OPERATOR TO
C                                         LEVEL 1, =3 IS INPUT OPERATORS TO
C                                         LEVEL 2, ...).
C                               OPOR(I,3) INDEX IN OPR OF OPERATOR WHICH
C                                         THIS OPERATOR INPUTS TO.
C                       **SEE NOTE 7
C
C     OPR     120,34    ARRAY OF OPERATOR INFORMATION.  FOR OPERATOR I,
C                       VALUES IN OPR ARE DESCRIBED BELOW.
C                               OPR(I,1)  OPERATOR TYPE (2003, 2004, 2005,
C                                         2010, 2011).
C                               OPR(I,2)  NUMBER OF INPUTS TO THIS
C                                         OPERATOR.
C                               OPR(I,3)  IS X OF (X,Y) COORDINATES OF
C                                         UPPER LEFT CORNER OF OPERATOR
C                                         BOX.
C                               OPR(I,4)  IS Y OF (X,Y) COORDINATES OF
C                                         UPPER LEFT CORNER OF OPERATOR
C                                         BOX.
C                               OPR(I,5)  OPERATOR INPUT INDICES.  NEGATIVE
C                               THROUGH   VALUES ARE PRIMED INPUTS.
C                               OPR(I,34) ABSOLUTE VALUES BETWEEN 1 AND
C                                         1000 ARE VARIABLE INDICES IN VAR.
C                                         ABSOLUTE VALUES OVER 1000 ARE
C                                         1000 PLUS OPERATOR INDICES IN
C                                         OPR.
C                       **SEE NOTE 5
C
C     PAGE    25,200    ARRAY CONTAINING THE DIAGRAM TO BE PRINTED.
C                       **SEE NOTE 6
C
C     PAR     50,3      PARENTHESES NESTING INFORMATION.  FOR PARENTHESIS
C                       SET I, THE FOLLOWING APPLIES.
C                               I,1 IS THE NESTING LEVEL (1 IS INNERMOST).
C                               I,2 IS THE SUBSCRIPT OF THE LEFT
C                                   PARENTHESIS IN EQN1.
C                               I,3 IS THE SUBSCRIPT OF THE RIGHT
C                                   PARENTHESIS IN EQN1.
C                       **SEE NOTE 4
C
C     SIDE    1         EMUX SECTION CODE (LEFT OR RIGHT).
C
C     SIZEE   1         MAX NUMBER OF EQUATIONS POSSIBLE - SET BY GETDAT.
C
C     SIZEM   1         DIMENSION OF INDEXM.
```

```
C
C     SIZER    1        WORD COUNT OF LAST RECORD READ BY GETDAT.
C
C     SIZM     1        MAXIMUM NUMBER OF RECORDS POSSIBLE ON TAPE7.
C
C     SYS      40       INPUT ARRAY CONTAINING ONEE 4 DIGIT NUMBERS FOR
C                       THE SYSTEMS, SUBSYSTEMS, OR SUB-SUBSYSTEMS TO BE
C                       DIAGRAMMED.
C
C     VAR      100,2    VAR(I,1) IS THE ARRAY OF VARIABLE NAMES OR TIME
C                       DELAY DURATIONS IN THE PRESENT EQUATION.  ONE EIGHT
C                       CHARACTER NAME PER WORD, LEFT JUSTIFIED WITH BLANK
C                       FILL.  VAR(I,2) INDICATES (WITH TWO LETTER LABELS)
C                       WHICH VAR(I,1) NAMES WERE REPLACED WITH TWO LETTER
C                       LABELS IN THE DIAGRAM.  **SEE NOTE 3
C
C     A,B,C,   1        MISCELLANEOUS TEMPORARIES SET BY BOLD.
C     D,I,J,
C     K,L,M,
C     N,R,S,
C     T
C
C     E,F,G,   1        MISCELLANEOUS TEMPORARIES SET BY PARSE.
C     P,Q
C
C     H,J,V,   1        MISCELLANEOUS TEMPORARIES SET BY READ4.
C     W,X
C
C     Y,Z,KK,  1        MISCELLANEOUS TEMPORARIES SET BY PACK.
C     LL,MM,
C     NN
C
C  A CHANGE IN DIMENSION OF ANY ARRAY IN LABELED COMMON WOULD AFFECT
C  AS A MINIMUM -
C    - ALL LABELED COMMON STATEMENTS
C    - VARIABLE DESCRIPTIONS IN BOLD
C    - DATA STATEMENTS IN BOLD
C    - REINITIALIZATION STATEMENTS IN BOLD
C    - WRITE AND FORMAT STATEMENTS IN ECHO AND ERROR
C
C  THE FOLLOWING TABLE SHOWS THOSE ADDITIONAL SUBROUTINES WHICH MAY BE
C  AFFECTED BY A DIMENSION CHANGE -
C
C  ARRAY    ---------ROUTINES---------
C              BOLD   PACK  PARSE  READ4
C  CARD                              X
C  COL         X
C  EQN         X                     X
C  EQN1        X              X
C  ERR         X
C  INPUTS                     X
C  OPOR        X
C  OPR         X              X
C  PAGE        X      X
C  PAR         X
```

```
C  SYS        X                    X
C  VAR        X
C
C
C  FOR THE FOLLOWING NOTES -
C  I IS THE MAXIMUM NUMBER OF PARENTHESIS PAIRS PER EQUATION
C  J IS THE MAXIMUM NUMBER OF UNIQUE OPERATORS PER EQUATION
C  K IS THE MAXIMUM NUMBER OF INPUTS PER OPERATOR
C  L IS THE MAXIMUM NUMBER OF LINES PER DIAGRAM
C  M IS THE MAXIMUM NUMBER OF UNIQUE VARIABLES PER EQUATION
C  THE PRESENT VALUES ARE - I=50, J=120, K=30, L=200, M=100
C
C  THE FOLLOWING TABLE SHOWS WHICH ARRAYS ARE A FUNCTION OF I,J,K,L,M
C
C  ARRAY    I  J  K  L  M
C
C  EQN      X  X           X
C  EQN1     X  X           X
C  INPUTS         X
C  OPOR        X
C  OPR         X  X
C  PAGE              X
C  PAR      X
C  VAR                     X
C
C
C        NOTE       DESCRIPTION
C         1         DIMENSION DETERMINED BY 2I+3J+40M.
C         2         DIMENSION DETERMINED BY 2I+3J+5M.
C         3         DIMENSION DETERMINED BY M.
C         4         DIMENSION DETERMINED BY I,3.
C         5         DIMENSION DETERMINED BY J,K+4.
C         6         DIMENSION DETERMINED BY 25,L.
C         7         DIMENSION DETERMINED BY J,3.
C         8         DIMENSION DETERMINED BY K.
C         9         DIMENSION DETERMINED BY COC SYSTEM ERROR RESET USAGE.
C        10         AN INTERNAL INTEGER CODE IS USED TO STORE AND ANALYZE
C                   THE EQUATIONS.  ALL OR PART OF THIS CODE IS USED BY
C                   OPR, EQN1, AND CHAR.  THE CODE IS SHOWN BELOW.
C
C                     VALUE          USED BY          DESCRIPTION
C                    FROM  TO
C
C                   -2006            EQN1            CODE FOR NOT EQUAL.
C                   -2000  -1        OPR,INPUTS      PRIMED OPERATOR INPUT.
C                                                    ABSOLUTE VALUE IS
C                                                    1000 + INDEX OF
C                                                    OPERATOR IN OPR.
C                   -1000  -1        OPR,INPJTS      PRIMED VARIABLE INDEX.
C                                                    ABSOLUTE VALUE IS INDEX
C                                                    OF VARIABLE IN VAR.
C                   1      1000      OPR,CHAR,       INDEX OF VARIABLE IN
C                                    EQN1,INPUTS     VAR.
C                   1001   2000      OPR,CHAR,       VALUE - 1000 IS INDEX
C                                    EQN1,INPUTS     OF OPERATOR IN OPR.
```

```
C                2001  2002     EQN1             CODE FOR ( AND )
C                2003  2005     EQN1,CHAR,OPR    CODE FOR * AND @ AND +
C                2006            EQN1             CODE FOR =
C                2007            EQN1,CHAR,OPR    CODE FOR '
C                2010  2011     EQN1,CHAR,OPR    CODE FOR $ AND ?
C                9999            EQN1,CHAR        CODE FOR REPLACED
C                                                 CHARACTER - SKIPPED
C                                                 BY LOGIC.
C
C                VALUES NOT LISTED ARE NOT VALID CODES.
C                NOTE SHOULD BE MADE OF THE FACT THAT INCREASING J
C                OVER 1000 WILL AFFECT THE INTERNAL INTEGER
C                CODE AS USED BY BOLD AND PARSE.  THIS SHOULD BE
C                AVOIDED.
C
C
C  SUBROUTINE DESCRIPTIONS:
C
C    NAME     DESCRIPTION
C
C    ECHO     DIAGNOSTIC PRINTOUT ROUTINE CALLED BY THE OPERATING SYSTEM
C             WHEN FATAL EXECUTION ERRORS OCCUR.
C
C    ERROR    DIAGNOSTIC PRINTOUT ROUTINE CALLED WHEN SYNTAX ERRORS ARE
C             FOUND AND WHEN LIMITS WOULD BE EXCEEDED.  EQUATION INVOLVED
C             IS SKIPPED.  ARGUMENT 1 TELLS ERROR WHAT THE PROBLEM IS.  A
C             NON-STANDARD RETURN IS USED TO TERMINATE PROCESSING OF
C             CURRENT EQUATION.
C
C    GETDAT   ROUTINE TO GET RANDOM ACCESS DATA FROM TAPE7.  IF DATA
C             CANNOT BE FOUND THEN QUESTION MARKS ARE PUT IN ARRAY DATA.
C
C    INIT     GENERAL INITIALIZATION SUBROUTINE.  ARGUMENT 1 IS THE
C             STARTING LOCATION, ARGUMENT 2 IS THE NUMBER OF LOCATIONS
C             TO BE INITIALIZED, ARGUMENT 3 IS THE INITIALIZATION VALUE.
C
C    LABL     THIS SUBROUTINE PROVIDES CHARACTER (LETTER OR NUMBER)
C             OUTPUT FOR USE IN LABELING OPERATOR INPUT AND/OR OUTPUT
C             LINES.  ARGUMENT 1 IS A SIGNAL TO OUTPUT ALPHABETIC
C             OR NUMERIC CHARACTERS.  ARGUMENT 2 IS THE VALUE TO BE
C             USED TO SELECT THE PROPER OUTPUT CHARACTERS.  ARGUMENT 3
C             IS THE OUTPUT TWO CHARACTERS - LEFT JUSTIFIED WITH BLANK
C             FILL.  NON-STANDARD RETURN USED AS ABOVE.
C
C    PACK     THIS SUBROUTINE PACKS CHARACTERS INTO ARRAY PAGE
C             (10 CHARACTERS PER WORD) FOR STORAGE OF THE DIAGRAM.  AN
C             ENTRY POINT (UNPACK) PROVIDES THE CAPABILITY TO GET
C             CHARACTERS OUT OF PAGE.  ARGUMENT 1 IS THE CHARACTER(S) TO
C             BE STORED (PACK CALL) OR CHARACTER(S) TO BE RETRIEVED
C             (UNPACK CALL).  ARGUMENT 2 IS THE NUMBER OF CHARACTERS IN
C             ARGUMENT 1.  ARGUMENT 3 IS THE PRINTOUT COLUMN FOR THE
C             FIRST CHARACTER.  ARGUMENT 4 IS THE PRINTOUT ROW.
C
C    PARSE    THIS SUBROUTINE WILL PERFORM THE ACTUAL SEMANTIC PARSING OF
C             EQN1(II) THROUGH EQN1(JJ).  NON-STANDARD RETURN USED WHEN
```

```
C            ERROR OCCURS.
C
C     READ4  SUBROUTINE TO SEARCH TAPE4 FOR THE NEXT EQUATION TO BE
C            DIAGRAMMED.  NON-STANDARD RETURN USED WHEN ERROR OCCURS OR
C            FOR NORMAL JOB TERMINATION.
C
C
C   WRITTEN BY R.B. BERGER  ASD/YHEJ  9 SEPT 75  FOR CDC 6600 FORTRAN
C   EXTENDED VERSION 4.4.
C
C
C   REVISED 1 AUGUST 1977
C
      COMMON /INFO/
     A   CARD(80),ERR(6),SYS(40),ISYS,EQN(4460),PAGE(25,200),VAR(100,2),
     B   EFF,SIDE,
     C   COL(12),EQN1(960),INPUTS(30),OPOR(120,3),OPR(120,34),PAR(50,3),
     D   CHAR,II,IFM,IK,ITO,JFM,JJ,JTO,LINE,LOPR,NCOL,NEQN,NEQN1,
     E   NIN,NOPR,NPAR,NVAR,A,B,C,D,E,F,G,H,I,J,K,KK,L,LL,M,MM,N,
     F   NN,P,Q,R,S,T,U,V,W,X,Y,Z
      COMMON INDEXM(16001),EQUATN(2300),DATA(400),SIZEM,SIZM,SIZEE,
     A   SIZER,KEY
      INTEGER
     A   CARD,ERR,SYS,EQN,PAGE,VAR,EFF,SIDE,COL,EQN1,OPOR,OPR,PAR,CHAR,
     B   A,B,C,D,E,F,G,H,P,Q,R,S,T,U,V,W,X,Y,Z
      INTEGER EQUATN,DATA,SIZEM,SIZM,SIZEE,SIZER
      EXTERNAL ECHO
      DATA CARD,SYS,EQN,PAGE,VAR,EFF,SIDE / 9782*1H  /
      DATA ERR / 0,0,0,-1,-1,-1 /
      DATA ISYS,COL,EQN1,INPUTS,OPOR,OPR,PAR,CHAR,II,IFM,IK,ITO,JFM,JJ,
     A   JTO,LINE,LOPR,NCOL,NEQN,NEQN1,NIN,NOPR,NPAR,NVAR,A,B,C,D,E,
     B   F,G,H,I,J,K,KK,L,LL,M,MM,N,NN,P,Q,R,S,T,U,V,W,X,Y,Z
     C   / 5639*0 /
C
C        SETUP SYSTEM ERROR RECOVERY LOGIC
C
      CALL RECOVR(ECHO,77B,0)
C
C        REDEFINE ERROR 104 (CANNOT FIND RECORD IN TAPE7) AS NON-FATAL
C
      CALL SYSTEMC(104,ERR)
C
C        OPEN TAPE7 AND SET MASTER INDEX
C
      CALL GETDAT(4HOPEN)
C
C        POSITION TAPE4 INITIALLY
C
      REWIND 4
C
C        READ SYSTEMS, SUBSYSTEMS, OR SUB-SUBSYSTEMS TO BE DIAGRAMMED
C
      READ(5,*) SYS,ISYS
      IF(SYS(1).NE.1H ) GO TO 20
      WRITE(6,3200)
```

```
      GO TO 999
C
C        RESET PART OF COMMON INFO BETWEEN EACH EQUATION
C
   20 CALL INIT(EQN,9662,1H )
      CALL INIT(COL,12,-9999)
      CALL INIT(EQN1,5626,0)
C
C        GET NEXT EQUATION TO BE DIAGRAMMED - STORE IN EQN
C
      CALL READ4, RETURNS(20,999)
C
C        SET EQN1 AND VAR
C
C        TEST FOR OPERATOR - IF FOUND, THEN PUT IN PROPER CODE NUMBER -
C             ( IS 2001
C             ) IS 2002
C             * IS 2003
C             @ IS 2004
C             + IS 2005
C             = IS 2006
C             ' IS 2007
C             $ IS 2010
C             ? IS 2011
C        IF EQN(I) IS NOT AN OPERATOR, THEN IT IS EITHER A VARIABLE
C        NAME OR A TIME DELAY DURATION.  VARIABLES AND TIME DELAYS WILL
C        BE REPLACED BY THEIR LOCATION IN VAR.
C
      I=1
   30 J=0
      IF(EQN(I).EQ.1H() J=2001
      IF(EQN(I).EQ.1H)) J=2002
      IF(EQN(I).EQ.1H*) J=2003
      IF(EQN(I).EQ.1H@) J=2004
      IF(EQN(I).EQ.1H+) J=2005
      IF(EQN(I).EQ.1H=) J=2006
      IF(EQN(I).EQ.1H') J=2007
      IF(EQN(I).EQ.1H$) J=2010
      IF(EQN(I).EQ.1H?) J=2011
      IF(J.EQ.0) GO TO 40
C
C        OPERATOR FOUND - SET EQN1(NEQN1) AND INCREMENT I
C
      NEQN1=NEQN1+1
      IF(NEQN1.GT.960) CALL ERROR(2), RETURNS(20)
      EQN1(NEQN1)=J
      I=I+1
      IF(I-NEQN) 30,30,140
C
C        EITHER VARIABLE NAME OR TIME DELAY DURATION STARTING AT EQN(I)
C        TEST EQN(I+4) TO FIGURE OUT WHICH
C
   40 IF(EQN(I+4).NE.1H- .AND. EQN(I+4).NE.1H> .AND.
     A    (EQN(I+4).LT.1HA .OR. EQN(I+4).GT.1HZ)) GO TO 70
C
```

```
C         VARIABLE NAME IN EQN(I) THROUGH EQN(I+7) - PACK INTO VAR(NVAR)
C         IF IT IS A NEW NAME AND THEN SET EQN1(NEQN1) AND INCREMENT I
C
      ENCODE(8,2000,K) EQN(I),EQN(I+1),EQN(I+2),EQN(I+3),EQN(I+4),
     A    EQN(I+5),EQN(I+6),EQN(I+7)
      DO 50 J=1,NVAR
   50 IF(K.EQ.VAR(J,1)) GO TO 60
      IF(NVAR.EQ.100) CALL ERROR(1), RETURNS(20)
      NVAR=NVAR+1
      VAR(NVAR,1)=K
      J=NVAR
   60 NEQN1=NEQN1+1
      IF(NEQN1.GT.960) CALL ERROR(2), RETURNS(20)
      EQN1(NEQN1)=J
      I=I+8
      IF(I-NEQN) 30,30,140
C
C         TIME DELAY DURATION IN EQN(I) THROUGH EQN(I+3) - PACK INTO
C         VAR(NVAR) IF IT IS A NEW VALUE AND THEN SET EQN1(NEQN1) AND
C         INCREMENT I
C
   70 ENCODE(4,2000,K) EQN(I),EQN(I+1),EQN(I+2),EQN(I+3)
      DO 80 J=1,NVAR
   80 IF(K.EQ.VAR(J,1)) GO TO 90
      IF(NVAR.EQ.100) CALL ERROR(1), RETURNS(20)
      NVAR=NVAR+1
      VAR(NVAR,1)=K
      J=NVAR
   90 NEQN1=NEQN1+1
      IF(NEQN1.GT.960) CALL ERROR(2), RETURNS(20)
      EQN1(NEQN1)=J
      I=I+4
      IF(I-NEQN) 30,30,140
C
C         SCAN EQN1 FOR PARENTHESES TO DETERMINE PROPER PARSING OF
C         EQUATION - SET PAR AND NPAR
C
  140 J=K=L=0
      DO 170 I=1,NEQN1
C
C         LOOK FOR LEFT OR RIGHT PARENTHESIS
C
      IF(EQN1(I).EQ.2002) GO TO 150
      IF(EQN1(I).NE.2001) GO TO 170
C
C         LEFT PARENTHESIS FOUND - SAVE SUBSCRIPT OF MOST CURRENT LEFT
C         PARENTHESIS IN J, INCREMENT K (THE NESTING LEVEL), SET L (THE
C         MAXIMUM NESTING LEVEL), AND SET PAR(J,1) AND PAR(J,2)
C
      IF(NPAR.EQ.50) CALL ERROR(3), RETURNS(20)
      NPAR=NPAR+1
      J=NPAR
      K=K+1
      L=MAXO(L,K)
      PAR(J,1)=K
```

```
      PAR(J,2)=I
      GO TO 170
C
C        RIGHT PARENTHESIS FOUND - SET PAR(J,3), DECREMENT K (NESTING
C        LEVEL) AND RESET J (RIGHTMOST LEFT PARENTHESIS FOUND WITHOUT
C        A RIGHT PARENTHESIS)
C
  150 IF(J.EQ.0) CALL ERROR(4), RETURNS(20)
      K=K-1
      PAR(J,3)=I
  160 J=J-1
      IF(J.EQ.0) GO TO 170
      IF(PAR(J,3).NE.0) GO TO 160
  170 CONTINUE
C
C        CHECK FOR UNMATCHED PARENTHESES
C
      IF(J.NE.0) CALL ERROR(8), RETURNS(20)
      IF(NPAR-1) 240,220,180
C
C        RESET NESTING LEVEL AS STORED IN PAR SO THAT INNERMOST SET
C        HAS ONE FOR ITS LEVEL
C
  180 DO 190 I=1,NPAR
  190 PAR(I,1)=L-PAR(I,1)+1
C
C        ARRANGE PAR FROM LOW TO HIGH LEVEL AND, FOR EACH LEVEL, FROM
C        LEFT TO RIGHT POSITION OF LEFT PARENTHESIS.
C
      K=NPAR-1
      DO 210 I=1,K
      L=I+1
      DO 210 J=L,NPAR
      IF(PAR(I,1)*10000+PAR(I,2).LE.PAR(J,1)*10000+PAR(J,2)) GO TO 210
      DO 200 M=1,3
      N=PAR(I,M)
      PAR(I,M)=PAR(J,M)
  200 PAR(J,M)=N
  210 CONTINUE
C
C        FOR EACH PARENTHESIS SET, CALL PARSE TO ANALYZE LOGIC IN
C        EQN1(II) THROUGH EQN1(JJ) AND THEN WIPE OUT PARENTHESES.
C
  220 DO 230 I=1,NPAR
      II=PAR(I,2)+1
      JJ=PAR(I,3)-1
      CALL PARSE, RETURNS(20)
  230 EQN1(II-1)=EQN1(JJ+1)=9999
C
C        NO MORE PARENTHESES - PARSE ENTIRE EQUATION
C
  240 II=3
      JJ=NEQN1
      CALL PARSE, RETURNS(20)
C
```

```
C        THE EQUATION HAS BEEN PARSED - NOW NEED TO DETERMINE WHERE THE
C        OPERATORS WILL BE LOCATED ON THE PAGE PRINTOUT TO MAKE THE MOST
C        READABLE DIAGRAM (MINIMUM CROSSOVER OF OPERATOR CONNECTIONS,
C        MINIMUM NUMBER OF UNCONNECTED OPERATORS, MAXIMUM USE OF
C        STRAIGHT LINE CONNECTIONS, ALL CONNECTIONS GOING LEFT TO RIGHT)
C
C        CHECK FOR ONLY ONE OPERATOR IN THE EQUATION
C
      IF(NOPR-1) 20,250,260
  250 OPR(1,3)=10
      OPR(1,4)=OPR(1,2)+MOD(OPR(1,2),2)-1
      NCOL=1
      COL(1)=OPR(1,4)+OPR(1,2)-MOD(OPR(1,2),2)+3
      IF(COL(1).GT.203) CALL ERROR(11), RETURNS(20)
      GO TO 400
C
C        MORE THAN ONE OPERATOR IN THE EQUATION.  BEFORE OPR(I,3) AND
C        OPR(I,4) CAN BE SET, NEED TO DEFINE OPOR(I,1), OPOR(I,2), AND
C        OPOR(I,3).  ORDER OF ENTRIES IN OPOR IS DIAGRAM ORDER STARTING
C        WITH OUTPUT OPERATOR ON THE RIGHT AND WORKING TO THE LEFT -
C        TOP TO BOTTOM IN EACH COLUMN.
C             OPOR(I,1) IS THE INDEX OF THE OPERATOR IN OPR
C             OPOR(I,2) IS THE OPERATOR LEVEL (=1 FOR OUTPUT OPERATOR,
C                       =2 FOR INPUT OPERATORS TO LEVEL 1, =3 FOR INPUT
C                       OPERATORS TO LEVEL 2, ...)
C             OPOR(I,3) IS THE INDEX IN OPR OF THE OPERATOR WHICH THIS
C                       OPERATOR INPUTS TO
C
  260 M=OPOR(1,2)=1
      OPOR(1,1)=NOPR
C
C        TAKE 12 PASSES THROUGH OPR - ONE FOR EACH LEVEL.  IF ALL OF
C        OPOR IS NOT SET, THEN THE EQUATION REQUIRES MORE THAN 12 COLUMN
C
      DO 272 L=1,12
C
C        FIND LEVEL L IN OPOR AND THEN GO TO OPERATOR J IN OPR AND
C        ASSIGN ITS INPUT OPERATORS, IF ANY, TO LEVEL L+1
C
      DO 271 J=1,NOPR
      IF(OPOR(J,2).NE.L) GO TO 271
      S=OPOR(J,1)
      N=OPR(S,2)
      DO 270 K=1,N
      R=IABS(OPR(S,K+4))
      IF(R.LT.1000) GO TO 270
      M=M+1
      OPOR(M,1)=R-1000
      OPOR(M,2)=L+1
      OPOR(M,3)=S
      IF(M.EQ.NOPR) GO TO 290
  270 CONTINUE
  271 CONTINUE
  272 CONTINUE
C
```

```
C        FAILED TO ASSIGN ALL OPERATORS
C
      CALL ERROR(6), RETURNS(20)
C
C        SET NCOL SO THAT IT IS THE MAX LEVEL (NUMBER OF COLUMNS)
C
  290 NCOL=OPOR(1,2)
      DO  300 I=2,NOPR
  300 NCOL=MAXO(NCOL,OPOR(I,2))
C
C        SET OPR(I,3) SO THAT THE OPERATORS WILL START IN COLUMNS
C        10, 30, 50, 70, 90, 110, 130, 150, 170, 190, 210, 230 IN ARRAY
C        PAGE
C
      DO  310 I=1,NOPR
  310 OPR(OPOR(I,1),3)=20*(NCOL-OPOR(I,2)+1)-10
C
C        SET OPR(I,4) BY GOING THROUGH OPOR IN ORDER.  PROCEDURE IS TO
C        PLACE THE OPERATOR VERTICALLY SO THAT THE CONNECTION PATH IS
C        STRAIGHT - IF POSSIBLE.  OPERATORS ARE PLACED TOP TO BOTTOM
C        WITHIN A GIVEN LEVEL - COL IS USED TO KEEP TRACK OF THE NEXT
C        AVAILABLE POSITION IN EACH LEVEL SO THAT THE OPERATORS WILL
C        HAVE AT LEAST TWO LINES BETWEEN THEM.
C
C        SET OPR(NOPR,4) AND THEN DO THE REST
C
      OPR(NOPR,4)=OPR(NOPR,2)+MOD(OPR(NOPR,2),2)-1
      COL(1)=OPR(NOPR,4)+OPR(NOPR,2)-MOD(OPR(NOPR,2),2)+3
C
C        SET OPR(I,4) - NEED TO KNOW ITS INPUT POSITION.  THE FIRST
C        EXPRESSION IN THE MAXO IS THE HIGHEST AVAILABLE POSITION
C        WITHOUT OVERLAP AND THE SECOND EXPRESSION IS THE DESIRED
C        VERTICAL POSITION FOR A STRAIGHT CONNECTION PATH
C
      DO  330 I=2,NOPR
      S=OPOR(I,3)
      N=OPR(S,2)
      DO  320 J=1,N
      R=IABS(OPR(S,J+4))
      IF(R.NE.OPOR(I,1)+1000) GO TO 320
      R=R-1000
      K=OPOR(I,2)
      M=OPR(R,2)
      OPR(R,4)=OPR(S,4)-N-MOD(N,2)+2*J-1
      IF(COL(K).NE.-9999) OPR(R,4)=MAXO(OPR(R,4),COL(K)+M+MOD(M,2)-2)
      COL(K)=OPR(R,4)+M-MOD(M,2)+3
      GO TO 330
  320 CONTINUE
  330 CONTINUE
C
C        NOW THAT OPR(I,4) HAS BEEN SET (BASED ON THE OUTPUT
C        OPERATOR BEING IN LINE 1), NEED TO INSURE THAT THE TOP OF
C        THE HIGHEST OPERATOR IS IN LINE 1
C
C        LET L BE THE TOP OF THE HIGHEST OPERATOR
```

```
C
      L=OPR(1,4)-OPR(1,2)-MOD(OPR(1,2),2)+2
      DO  340 I=2,NOPR
  340 L=MIN0(L,OPR(I,4)-OPR(I,2)-MOD(OPR(I,2),2)+2)
C
C         NOW NEED TO RESET OPR(I,4) AND COL(I)
C
      DO  350 I=1,NCOL
  350 COL(I)=COL(I)-L+1
      DO  360 I=1,NOPR
  360 OPR(I,4)=OPR(I,4)-L+1
C
C         CHECK COL TO SEE THAT THE DIAGRAM WILL FIT IN ARRAY PAGE
C
      DO  390 I=1,NCOL
  390 IF(COL(I).GT.203) CALL ERROR(11), RETURNS(20)
C
C         LOAD OPERATORS INTO PAGE AND THEN CONNECT THEM
C
  400 DO  590 N=1,NOPR
      A=OPR(N,2)
      J=OPR(N,3)
      K=OPR(N,4)
      L=K-A-MOD(A,2)+2
      D=K+A-MOD(A,2)
C
C         OUTPUT THE BOX (EVERYTHING EXCEPT INPUT DASHES, PRIMES, AND
C         LEFTMOST ASTERISKS)
C
      CALL PACK(2H**,2,J+1,K), RETURNS(20)
      IF(OPR(N,1).EQ.2003) CALL PACK(3HA*-,3,J+1,K+1), RETURNS(20)
      IF(OPR(N,1).EQ.2004) CALL PACK(3HX*-,3,J+1,K+1), RETURNS(20)
      IF(OPR(N,1).EQ.2005) CALL PACK(3HO*-,3,J+1,K+1), RETURNS(20)
      IF(OPR(N,1).EQ.2010) CALL PACK(3H1*-,3,J+1,K+1), RETURNS(20)
      IF(OPR(N,1).EQ.2011) CALL PACK(3H2*-,3,J+1,K+1), RETURNS(20)
      CALL PACK(2H**,2,J+1,K+2), RETURNS(20)
C
C         SET THE INPUT DASHES, PRIMES, AND LEFTMOST ASTERISKS
C
      DO  420 M=L,D
      S=MOD(M-L+1,2)
      IF(S.EQ.0) CALL PACK(2H *,2,J-1,M), RETURNS(20)
      IF(S.NE.0 .AND. OPR(N,(M-L+2)/2+4).GE.0) CALL PACK(2H-*,2,J-1,M),
     A    RETURNS(20)
  420 IF(S.NE.0 .AND. OPR(N,(M-L+2)/2+4)..T.0) CALL PACK(2H-0,2,J-1,M),
     A    RETURNS(20)
C
C         CONNECT THE OPERATORS AND LABEL THE INPUTS FROM VARIABLES.
C         THE ONLY LINE CONNECTION PATH TO BE ATTEMPTED WILL BE
C         OVER-UP(DOWN)-OVER.  IF THERE IS NO CLEAR PATH AVAILABLE, THE
C         OPERATOR OUTPUT AND INPUT WILL BE LABELED WITH A TWO
C         DIGIT NUMBER. OPERATOR INPUTS FROM VARIABLES WILL BE LABELED
C         WITH A TWO LETTER LABEL.  A TABLE WILL BE PRINTED TO IDENTIFY
C         THE TWO LETTER LABELS WITH RI'S SIGNAL DESIGNATORS.
C
```

```
C         LOOK AT EACH INPUT TO EACH OPERATOR AND CONNECT OR LABEL
C         THOSE INPUTS FROM OTHER OPERATORS.  INPUTS FROM VARIABLES
C         WILL BE DONE NEXT.
C
      DO 590 I=1,A
      L=IABS(OPR(N,I+4))
      IF(L.LE.1000) GO TO 590
      L=L-1000
C
C         INPUT IS FROM ANOTHER OPERATOR - TRY TO CONNECT THE OPERATORS
C
      ITO=J-5
      JTO=K-A-MOD(A,2)+2*I
      IFM=OPR(L,3)*7
      JFM=OPR(L,4)*1
      S=MIN0(JFM,JTO)
      T=MAX0(JFM,JTO)
C
C         FIND CONNECTION PATH
C
      DO 580 IK=IFM,ITO
C
C         SEE IF PATH IS CLEAR FROM (IFM,JFM) TO (IK,JFM) - BLANK OR
C         I OK
C
      DO 490 M=IFM,IK
      CALL UNPACK(C,1,M,JFM), RETURNS(20)
  490 IF(C.NE.1H  .AND. C.NE.1HI) GO TO 580
C
C         SEE IF PATH IS CLEAR BETWEEN (IK,JFM) AND (IK,JTO) - BLANK OR
C         DASH OK
C
      DO 500 M=S,T
      CALL UNPACK(C,1,IK,M), RETURNS(20)
  500 IF(C.NE.1H  .AND. C.NE.1H-) GO TO 580
C
C         SEE IF PATH IS CLEAR BETWEEN (IK,JTO) AND (ITO,JTO) - BLANK OK
C
      DO 510 M=IK,ITO
      CALL UNPACK(C,1,M,JTO), RETURNS(20)
  510 IF(C.NE.1H ) GO TO 580
C
C         CONNECTION PATH IS CLEAR BUT BEFORE DRAWING NEED TO
C         MAKE SURE THAT IK IS IN THE PROPER COLUMNS (17-25, 37-45,
C         57-65, 77-85, 97-105, 117-125, 137-145, 157-165, 177-185,
C         197-205, 217-225)
C
      IF(((IK.GT. 25 .AND. IK.LT. 37).OR.(IK.GT. 45 .AND. IK.LT. 57).OR.
     A   (IK.GT. 65 .AND. IK.LT. 77).OR.(IK.GT. 85 .AND. IK.LT. 97).OR.
     B   (IK.GT.105 .AND. IK.LT.117).OR.(IK.GT.125 .AND. IK.LT.137).OR.
     C   (IK.GT.145 .AND. IK.LT.157).OR.(IK.GT.165 .AND. IK.LT.177).OR.
     D   (IK.GT.185 .AND. IK.LT.197).OR.(IK.GT.205 .AND. IK.LT.217))
     E   GO TO 580
C
C         PATH FOUND WITH CORNERS AT (IK,JFM) AND (IK,JTO) - LOAD IT INTO
```

```
C       PAGE - USE PERIODS FOR CORNER CHARACTERS
C
      L=IFM-3
      D=ITO+3
      IF(JFM.NE.JTO) GO TO 530
      DO 520 M=L,D
      CALL UNPACK(C,1,M,JFM), RETURNS(20)
  520 IF(C.EQ.1H ) CALL PACK(1H-,1,M,JFM), RETURNS(20)
      GO TO 590
  530 S=S+1
      T=T-1
      DO 550 M=S,T
  550 CALL PACK(1HI,1,IK,M), RETURNS(20)
      DO 560 M=L,IK
      CALL UNPACK(C,1,M,JFM), RETURNS(20)
  560 IF(C.EQ.1H ) CALL PACK(1H-,1,M,JFM), RETURNS(20)
      DO 570 M=IK,D
      CALL UNPACK(C,1,M,JTO), RETURNS(20)
  570 IF(C.EQ.1H ) CALL PACK(1H-,1,M,JTO), RETURNS(20)
      CALL PACK(1H.,1,IK,JFM), RETURNS(20)
      CALL PACK(1H.,1,IK,JTO), RETURNS(20)
      GO TO 590
  580 CONTINUE
C
C       NO PATH POSSIBLE - INSTEAD OF CONNECTING OPERATORS, LABEL THE
C       OUTPUT AND INPUT WITH A TWO DIGIT NUMBER.
C
      CALL LABL(2,L,B), RETURNS(20)
      CALL PACK(B,2,IFM-3,JFM), RETURNS(20)
      CALL PACK(B,2,ITO+2,JTO), RETURNS(20)
  590 CONTINUE
C
C       LOOK AT EACH INPUT TO EACH OPERATOR AND, FOR THOSE INPUTS FROM
C       VARIABLES, LABEL WITH VAR(L,1) OR WITH A 2-LETTER LABEL
C
      DO 620 N=1,NOPR
      A=OPR(N,2)
      J=OPR(N,3)
      K=OPR(N,4)
      DO 620 I=1,A
      L=IABS(OPR(N,I+4))
      IF(L.GT.1000) GO TO 620
C
C       INPUT IS EITHER A VARIABLE OR A TIME DELAY DURATION VALUE -
C       LABEL WITH VAR(L,1) OR A 2 LETTER LABEL
C
      T=K-A-MOD(A,2)+2*I
      IF( OPR(N,1).NE.2010 .AND. OPR(N,1).NE.2011) .OR.
     A   AND(VAR(L,1),777777777778).NE.555555555558) GO TO 600
C
C       INPUT IS TIME DELAY DURATION
C
      CALL UNPACK(C,5,J-6,T), RETURNS(20)
      IF(C.NE.1H ) GO TO 610
      CALL PACK(VAR(L,1),4,J-5,T), RETURNS(20)
```

```
      GO TO 620
C
C        INPUT IS A VARIABLE
C
  600 IF(J.EQ.10) CALL UNPACK(C,8,J-9,T), RETURNS(20)
      IF(J.NE.10) CALL UNPACK(C,9,J-10,T), RETURNS(20)
      IF(C.NE.1H ) GO TO 610
      CALL PACK(VAR(L,1),8,J-9,T), RETURNS(20)
      GO TO 620
C
C        INPUT COULD NOT BE FIT IN - USE A 2 LETTER LABEL
C
  610 CALL LABL(1,L,VAR(L,2)), RETURNS(20)
      CALL PACK(VAR(L,2),2,J-3,T), RETURNS(20)
  620 CONTINUE
C
C        ADD OUTPUT VARIABLE
C
      J=OPR(NOPR,3)
      K=OPR(NOPR,4)+1
      IF(EQN1(2).EQ.-2006) CALL PACK(1HO,1,J+2,K), RETURNS(20)
      CALL PACK(VAR(1,1),8,J+4,K), RETURNS(20)
C
C        OUTPUT HEADER
C
      CALL GETDAT(VAR(1,1))
      DECODE(160,2400,DATA) L,M,N
      ENCODE(10,1900,KEY) VAR(1,1)
      CALL GETDAT(KEY)
      LINE=8
      WRITE(6,2800) VAR(1,1),EFF,SIDE,(DATA(I),I=1,7),L,M,N,
     A    (DATA(I),I=8,21)
C
C        OUTPUT EQUATION
C
      J=125
      IF(NEQN.GT.124) GO TO 630
      LINE=LINE+2
      WRITE(6,2300) (EQN(K),K=1,NEQN)
      GO TO 670
  630 J=J-1
      IF(EQN(J).NE.1H* .AND. EQN(J).NE.1H) .AND. EQN(J).NE.1H+ .AND.
     A    EQN(J).NE.1H$ .AND. EQN(J).NE.1H?) GO TO 630
      LINE=LINE+2
      WRITE(6,2300) (EQN(K),K=1,J)
  640 I=J+1
      IF(J+124.LT.NEQN) GO TO 650
      LINE=LINE+1
      WRITE(6,2200) (EQN(K),K=I,NEQN)
      GO TO 670
  650 J=J+125
  660 J=J-1
      IF(EQN(J).NE.1H* .AND. EQN(J).NE.1H) .AND. EQN(J).NE.1H+ .AND.
     A    EQN(J).NE.1H$ .AND. EQN(J).NE.1H? .AND. I.NE.J) GO TO 660
      LINE=LINE+1
```

```
      WRITE(6,2200) (EQN(K),K=I,J)
      GO TO 640
C
C         REARRANGE VAR FROM LOW TO HIGH
C
  670 I=NVAR-1
      DO 680 J=2,I
      K=J+1
      DO 680 L=K,NVAR
      IF(VAR(J,1).LE.VAR(L,1)) GO TO 680
      M=VAR(J,1)
      VAR(J,1)=VAR(L,1)
      VAR(L,1)=M
      M=VAR(J,2)
      VAR(J,2)=VAR(L,2)
      VAR(L,2)=M
  680 CONTINUE
C
C         OUTPUT SIGNAL DESCRIPTION TABLE
C
      WRITE(6,3000)
      B=NVAR/2+MOD(NVAR,2)
      LINE=LINE+B+6
      DO 700 I=1,B
      A=C=D=J=K=R=S=T=1H
      L=10HTIME DELAY
      M=10H DURATION
      N=10H(SEC)
      IF(AND(VAR(I,1),7777777777777B).EQ.5555555555555B) GO TO 690
      CALL GETDAT(VAR(I,1))
      DECODE(160,2400,DATA) L,M,N
      DECODE(20,2100,VAR(I,2)) J,K
  690 IF(B+I.GT.NVAR .OR. NVAR.EQ.1) GO TO 700
      A=10HTIME DELAY
      D=10H DURATION
      S=10H(SEC)
      C=VAR(B+I,1)
      IF(AND(C,7777777777777B).EQ.5555555555555B) GO TO 700
      CALL GETDAT(C)
      DECODE(160,2400,DATA) A,D,S
      DECODE(20,2100,VAR(B+I,2)) T,R
  700 WRITE(6,3100) J,K,VAR(I,1),L,M,N,  T,R,C,A,D,S
C
C         OUTPUT DIAGRAM
C
      J=NCOL-6
      K=A=-9999
      DO 710 I=1,NCOL
      IF(I.LE.J) A=MAX0(A,COL(I))
  710 IF((I.GE.J .AND. J.GT.0) .OR. J.LE.0) K=MAX0(K,COL(I))
      CALL GETDAT(VAR(1,1))
      DECODE(160,2400,DATA) L,M,N
      ENCODE(10,1900,KEY) VAR(1,1)
      CALL GETDAT(KEY)
C
```

```
C ---     SEE IF HEADER PAGE CAN BE USED
C
      IF(LINE+K.GT.55) GO TO 730
      WRITE(6,2700)
      WRITE(6,2900) ((PAGE(I,J),I=1,13),J=1,K)
C
C        HEADER PAGE USED - SEE IF ADDITIONAL PAGE(S) NEEDED
C
      IF(A.EQ.-9999) GO TO 740
      IF(LINE+K.LE.51) WRITE(6,2600)
      WRITE(6,2800) VAR(1,1),EFF,SIDE,(DATA(I),I=1,7),L,M,N,
     A    (DATA(I),I=8,21)
      J=LINE-7
      DO 720 I=1,J
  720 WRITE(6,3400)
      WRITE(6,3300) ((PAGE(I,J),I=14,25),J=1,A)
      GO TO 740
C
C        HEADER PAGE CANNOT BE USED - GO TO A NEW PAGE
C
  730 IF(LINE.LE.51) WRITE(6,2500)
      WRITE(6,2800) VAR(1,1),EFF,SIDE,(DATA(I),I=1,7),L,M,N,
     A    (DATA(I),I=8,21)
      WRITE(6,2700)
      WRITE(6,2900) ((PAGE(I,J),I=1,13),J=1,K)
C
C        HEADER PAGE NOT USED - SEE IF ADDITIONAL PAGE(S) NEEDED
C
      IF(A.EQ.-9999) GO TO 740
      IF(LINE+K.LE.51) WRITE(6,2600)
      WRITE(6,2800) VAR(1,1),EFF,SIDE,(DATA(I),I=1,7),L,M,N,
     A    (DATA(I),I=8,21)
      WRITE(6,2700)
      WRITE(6,3300) ((PAGE(I,J),I=14,25),J=1,A)
C
C        THIS EQUATION IS DONE - GO TO NEXT ONE
C
  740 IF(ISYS.NE.0) CALL ERROR(7), RETURNS(20)
      GO TO 20
C
C        NO MORE EQUATIONS
C
  999 CONTINUE
 1900 FORMAT(A4* LABEL*)
 2000 FORMAT(8A1)
 2100 FORMAT(2A1)
 2200 FORMAT(11X,124A1)
 2300 FORMAT(/2X,124A1)
 2400 FORMAT(34X,2A10,A5)
 2500 FORMAT(//50X*$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$*/
     A          50X*$$$$ DIAGRAM ON THE NEXT PAGE $$$$*/
     B          50X*$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$*)
 2600 FORMAT(//50X*$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$*/
     A          50X*$$$$     CIAGRAM CONTINUED      $$$$*/
     B          50X*$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$*)
```

```
2700 FORMAT(*0*)
2800 FORMAT(*1*///23X*EMUX BOOLEAN EQUATION DIAGRAM FOR SIGNAL *A8
     A    * OF AIRCRAFT NO. *A2,2X,A5* SIDE OF EMUX*//
     B    15X,7A10,5X,2A10,A5/
     C    15X,7A10/
     D    15X,7A10)
2900 FORMAT(1X,13A10)
3000 FORMAT(///
     A    10X* DIAGRAM        EMUX     SIGNAL SOURCE OR DESTINATION*
     B    10X* DIAGRAM        EMUX     SIGNAL SOURCE OR DESTINATION*/
     C    10X*DESIGNATOR    SIGNAL*
     D    42X*DESIGNATOR    SIGNAL*/
     E    10X*              DESIGNATOR*
     F    40X*              DESIGNATOR*/)
3100 FORMAT(13X,2A1,8X,A8,3X,2A10,A5,16X,2A1,8X,A8,3X,2A10,A5)
3200 FORMAT(* ERROR IN INPUT DATA CARDS - NO SYSTEM NUMBERS READ - RUN*
     A    * TERMINATED*)
3300 FORMAT(1X,12A10)
3400 FORMAT(1X)
     END
```

```
      SUBROUTINE ECHO(IEXCH,ENDRUN,IRA)
C
C   THIS ROUTINE IS USED IN CONJUNCTION WITH SYSTEM ROUTINE RECOVR
C   TO REINITIALIZE EXECUTION WITHOUT RELOADING IN THE EVENT OF
C   EXECUTION TIME FATAL ERRORS.  EXECUTION WILL BE RESTARTED ONLY TO
C   PRINT OUT COMMON DATA.
C
C
      COMMON /INFO/
     A    CARD(80),ERR(6),SYS(40),ISYS,EQN(4460),PAGE(25,200),VAR(100,2),
     B    EFF,SIDE,
     C    COL(12),EQN1(960),INPUTS(30),OPOR(120,3),OPR(120,34),PAR(50,3),
     D    CHAR,II,IFM,IK,ITO,JFM,JJ,JTO,LINE,LOPR,NCOL,NEQN,NEQN1,
     E    NIN,NOPR,NPAR,NVAR,A,B,C,D,E,F,G,H,I,J,K,KK,L,LL,M,MM,N,
     F    NN,P,Q,R,S,T,U,V,W,X,Y,Z
      COMMON INDEXM(16001),EQUATN(2360),DATA(400),SIZEM,SIZM,SIZEE,
     A    SIZER,KEY
      INTEGER
     A    CARD,ERR,SYS,EQN,PAGE,VAR,EFF,SIDE,COL,EQN1,OPOR,OPR,PAR,CHAR,
     B    A,B,C,D,E,F,G,H,P,Q,R,S,T,U,V,W,X,Y,Z
      INTEGER EQUATN,DATA,SIZEM,SIZM,SIZEE,SIZER
      DIMENSION IEXCH(17),IRA(1)
C
C         FIGURE OUT ERROR ADDRESS AND NUMBER
C
      ENDRUN=1.
      IADDR=AND(SHIFT(IEXCH(1),24),7777778)
      IF(IADDR.NE.0) GO TO 10
      IERROR=AND(SHIFT(IRA(IADDR),12),78)
      IADDR=AND(SHIFT(IRA(IADDR),30),7777778)
      IIIIII=5HMODE
      GO TO 20
   10 IERROR=AND(IEXCH(1),778)
      IIIIII=5HTYPE
C
C         PRINT COMMON
C
   20 WRITE(6,1000) IIIIII,IERROR,IADDR,
     A    ISYS,CHAR,II,IFM,IK,ITO,JFM,JJ,JTO,LINE,LOPR,NCOL,NEQN,
     B    NEQN1,NIN,NOPR,NPAR,NVAR,A,B,C,D,E,F,G,H,I,J,K,KK,L,LL,M,MM,
     C    N,NN,P,Q,R,S,T,U,V,W,X,Y,Z,SIZEM,SIZM,SIZEE,SIZER,
     D    EFF,SIDE,KEY,A,B,C,D,E,F,G,H,I,J,K,KK,L,LL,M,MM,N,NN,P,Q,
     E    R,S,T,U,V,W,X,Y,Z,
     F    CARD,ERR,SYS,EQN,((VAR(I,J),J=1,2),I=1,100),COL,EQN1,
     G    INPUTS,((OPOR(I,J),J=1,3),I=1,12)),((OPR(I,J),J=1,34),I=1,120),
     H    ((PAR(I,J),J=1,3),I=1,50),DATA
      CALL EXIT
 1000 FORMAT(*1*//*  JOB RECOVERED FROM ERROR *A5,02* AT ADDRESS *06/
     A    * SIMPLE INTEGERS*/2(5X,20(I5,1X)/),5X,11(I5,1X)//
     B    * SIMPLE HOLLERITHS*/5X,3A11/4(5X,6021/)5X,5021//
     C    * CARD*/8(5X,10A11/)//
     D    * ERR*/5X,6(I5,1X)//
     E    * SYS*/4(5X,10A11/)//
     F    * EQN*/44(5X,100A1/),5X,60A1//
     G    * VAR*/100(5X,2A11/)//
```

```
H     * COL*/5X,12(I5,1X)//
I     * EQN1*/48(5X,20(I5,1X)/)//
J     * INPUTS*/5X,20(I5,1X)/5X,10(I5,1X)//
K     * OPOR*/120(5X,3(I5,1X)/)//
L     * OPR*/120(5X,20(I5,1X)/5X,14(I5,1X)/)//
M     * PAR*/50(5X,3(I5,1X)/)//
N     * DATA*/40(5X,10A11/))
      END
```

```
      SUBROUTINE ERROR(IERROR), RETURNS(AAAAAA)
C
C   DIAGNOSTIC PRINTOUT AND EQUATION PROCESSING TERMINATION SUBROUTINE.
C   IERROR SIGNALS THE TYPE OF ERROR FOR PRINTOUT ERROR MESSAGE
C   SELECTION.  THE NON-STANDARD ERROR RETURN IS ALWAYS USED TO
C   TERMINATE PROCESSING OF THE OFFENDING EQUATION.
C
      COMMON /INFO/
     A    CARD(80),ERR(6),SYS(40),ISYS,EQN(4460),PAGE(25,200),VAR(100,2),
     B    EFF,SIDE,
     C    COL(12),EQN1(960),INPUTS(30),OPOR(120,3),OPR(120,34),PAR(50,3),
     D    CHAR,II,IFM,IK,ITO,JFM,JJ,JTO,LINE,LOPR,NCOL,NEQN,NEQN1,
     E    NIN,NOPR,NPAR,NVAR,A,B,C,D,E,F,G,H,I,J,K,KK,L,LL,M,MM,N,
     F    NN,P,Q,R,S,T,U,V,W,X,Y,Z
      COMMON INDEXM(15001),EQUATN(2300),DATA(400),SIZEM,SIZM,SIZEE,
     A    SIZER,KEY
      INTEGER
     A    CARD,ERR,SYS,EQN,PAGE,VAR,EFF,SIDE,COL,EQN1,OPOR,OPR,PAR,CHAR,
     B    A,B,C,D,E,F,G,H,P,Q,R,S,T,U,V,W,X,Y,Z
      INTEGER EQUATN,DATA,SIZEM,SIZM,SIZEE,SIZER
      WRITE(6,2000) IERROR
      GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22),
     A    IERROR
    1 WRITE(6,1001)
      GO TO 500
    2 WRITE(6,1002)
      GO TO 500
    3 WRITE(6,1003)
      GO TO 500
    4 WRITE(6,1004)
      GO TO 500
    5 WRITE(6,1005)
      GO TO 500
    6 WRITE(6,1006)
      GO TO 500
    7 WRITE(6,1007)
      GO TO 500
    8 WRITE(6,1008)
      GO TO 500
    9 WRITE(6,1009)
      GO TO 500
   10 WRITE(6,1010)
      GO TO 500
   11 WRITE(6,1011)
      GO TO 500
   12 WRITE(6,1012)
      GO TO 500
   13 WRITE(6,1013)
      GO TO 500
   14 WRITE(6,1014)
      GO TO 500
   15 WRITE(6,1015)
      GO TO 500
   16 WRITE(6,1016)
      GO TO 500
```

```
   17 WRITE(6,1017)
      GO TO 500
   18 WRITE(6,1018)
      GO TO 500
   19 WRITE(6,1019)
      GO TO 500
   20 WRITE(6,1020)
      GO TO 500
   21 WRITE(6,1021)
      GO TO 500
   22 WRITE(6,1022)
      GO TO 500
  500 WRITE(6,3000)
     A    ISYS,CHAR,II,IFM,IK,ITO,JFM,JJ,JTO,LINE,LOPR,NCOL,NEQN,
     B    NEQN1,NIN,NOPR,NPAR,NVAR,A,B,C,D,E,F,G,H,I,J,K,KK,L,LL,M,MM,
     C    N,NN,P,Q,R,S,T,U,V,W,X,Y,Z,SIZEM,SIZM,SIZEE,SIZER,
     D    EFF,SIDE,KEY,A,B,C,D,E,F,G,H,I,J,K,KK,L,LL,M,MM,N,NN,P,Q,
     E    R,S,T,U,V,W,X,Y,Z,
     F    CARD,ERR,SYS,EQN,((VAR(I,J),J=1,2),I=1,100),COL,EQN1,
     G    INPUTS,((OPOR(I,J),J=1,3),I=1,120),((OPR(I,J),J=1,34),I=1,120),
     H    ((PAR(I,J),J=1,3),I=1,50),DATA
      RETURN AAAAAA
 1001 FORMAT(* TOO MANY VARIABLES IN EQUATION*)
 1002 FORMAT(* EQUATION TOO BIG IN REDUCED FORM FOR EQN1*)
 1003 FORMAT(* TOO MANY PARENTHESIS PAIRS*)
 1004 FORMAT(* MISSING LEFT PARENTHESIS*)
 1005 FORMAT(* TOO MANY OPERATORS IN EQUATION*)
 1006 FORMAT(* COULD NOT ASSIGN ALL OPERATORS TO COLUMNS*)
 1007 FORMAT(* FORCED ERROR OUTPUT - NO ERROR*)
 1008 FORMAT(* MISSING RIGHT PARENTHESIS*)
 1009 FORMAT(* PACK OR UNPACK CALLED WITH NAA <=0 OR >=11*)
 1010 FORMAT(* EQUATION ON FILE HAS TOO MANY CHARACTERS FOR ARRAY EQN*)
 1011 FORMAT(* EQUATION DIAGRAM HAS TOO MANY ROWS FOR PAGE ARRAY*)
 1012 FORMAT(* BAD VALUE INPUT TO SUBROUTINE VAL*)
 1013 FORMAT(* INVALID EQUATION SYNTAX - OPERATOR INPUT NOT FOUND*)
 1014 FORMAT(* TOO MANY OPERATOR INPUTS FOR ONE OPERATOR*)
 1015 FORMAT(* MISSING LEFT SIDE VARIABLE*)
 1016 FORMAT(* MISSING EQUAL SIGN*)
 1017 FORMAT(* RIGHT SIDE OF EQUATION IS MISSING*)
 1018 FORMAT(* IMPROPER USE OF PRIME OPERATOR*)
 1019 FORMAT(* INVALID EQUATION SYNTAX - OPERATOR (AND, EXCLUSIVE OR,*
     A    * OR, TIME DELAY) NOT FOUND*)
 1020 FORMAT(* EQUATION TOO BIG FOR PAGE ARRAY*)
 1021 FORMAT(* BAD DATA FOR UNPACK*)
 1022 FORMAT(* PACK OR UNPACK CALLED WITH BB OR CC <=0*)
 2000 FORMAT(*1*//*  ERROR NUMBER *I3)
 3000 FORMAT(* TERMINATE PROCESSING FOR THIS EQUATION*/
     A    * SIMPLE INTEGERS*/2(5X,20(I5,1X)/),5X,11(I5,1X)//
     B    * SIMPLE HOLLERITHS*/5X,3A11/4(5X,6021/)5X,5021//
     C    * CARD*/8(5X,10A11/)//
     D    * ERR*/5X,6(I5,1X)//
     E    * SYS*/4(5X,10A11/)//
     F    * EQN*/44(5X,100A1/),5X,60A1//
     G    * VAR*/100(5X,2A11/)//
     H    * COL*/5X,12(I5,1X)//
```

```
      I    * EQN1*/48(5X,20(I5,1X)/)//
      J    * INPUTS*/5X,20(I5,1X)/5X,10(I5,1X)//
      K    * OPOR*/120(5X,3(I5,1X)/)//
      L    * OPR*/120(5X,20(I5,1X)/5X,14(I5,1X)/)//
      M    * PAR*/50(5X,3(I5,1X)/)//
      N    * DATA*/40(5X,10A11/))
       END
```

```
      SUBROUTINE GETDAT(KEYREQ)
C
C   ROUTINE TO GET RANDOM ACCESS DATA FROM TAPE7
C
C   THE ONLY INPUT IS KEYREQ WHICH IS THE KEY OF THE REQUESTED RECORD
C
C   THE RECORD IS OUTPUT IN DATA(1) THROUGH DATA(SIZER).  IF NO RECORD
C   CAN BE FOUND, THEN DATA IS SET TO QUESTION MARKS.
C
C
      COMMON INDEXM(16001),EQUATN(2300),DATA(400),SIZEM,SIZM,SIZEE,
     A   SIZER,KEY
      INTEGER EQUATN,DATA,SIZEM,SIZM,SIZEE,SIZER
C
C        TEST FOR REQUEST TO OPEN TAPE7
C
      IF(KEYREQ.NE.4HOPEN) GO TO 5
C
C        OPEN TAPE7 AND INITIALIZE BLANK COMMON
C
      SIZEM=16001
      SIZEE=2300
      CALL OPENMS(7,INDEXM,SIZEM,1)
      CALL READMS(7,EQUATN,SIZEE,9HEQUATIONS)
      CALL INIT(DATA,400,1H )
      SIZM=(SIZEM-1)/2
      SIZER=0
      KEY=1H
      RETURN
C
C        INITIALIZE
C
    5 CALL INIT(DATA,400,10H??????????)
C
C        FIGURE OUT RECORD SIZE AND SET SIZER
C
      SIZER=0
      DECODE(10,1000,KEYREQ) IIIIII,KKKKKK,JJJJJJ
      IF(JJJJJJ.NE.2H  ) GO TO 10
C
C        EITHER DISCRETE OR SERIAL-DIGITAL DATA RECORD REQUESTED
C
      SIZER=8
      IF(IIIIII.EQ.1H- .OR. IIIIII.EQ.1H>) SIZER=16
      IF((IIIIII.GE.1HA .AND. IIIIII.LE.1HZ) .AND. KKKKKK.NE.1H0)
     A   SIZER=16
      GO TO 40
C
C        EITHER EQUATION OR LABEL OR EQUATION SIZE RECORD REQUESTED
C
   10 IF(JJJJJJ.NE.2HEQ) GO TO 30
C
C        EQUATION REQUESTED
C
      DECODE(10,1100,KEYREQ) IIIIII
```

```
      DO 20 JJJJJJ=1,SIZEE
      DECODE(10,1200,EQUATN(JJJJJJ)) KKKKKK,LLLLLL
      SIZER=LLLLLL*8
   20 IF(IIIIII.EQ.KKKKKK) GO TO 40
      SIZER=0
      RETURN
C
C        EITHER LABEL OR EQUATION SIZE RECORD REQUESTED
C
   30 IF(JJJJJJ.EQ.2HEL) SIZER=21
      IF(JJJJJJ.EQ.2HS ) SIZER=SIZEE
      IF(SIZER.EQ.0) RETURN
C
C        GET DATA
C
   40 CALL READMS(7,DATA,SIZER,KEYREQ)
 1000 FORMAT(4X,A1,2X,A1,A2)
 1100 FORMAT(A8)
 1200 FORMAT(A8,I2)
      END
```

```
      SUBROUTINE INIT(X,N,XVAL)
C
C  ROUTINE TO INITIALIZE A SELECTED STORAGE AREA (FROM X(1) TO X(N)) TO
C  THE VALUE OF XVAL.
C
      DIMENSION X(1)
      INTEGER X,XVAL
      DO 10 I=1,N
   10 X(I)=XVAL
      RETURN
      END
```

```
      SUBROUTINE LABL(ALPNUM,VAL,CHAR), RETURNS(AAAAAA)
C
C   THIS SUBROUTINE WILL OUTPUT A TWO CHARACTER LABEL.  ALPNUM SIGNALS
C   WHETHER AN ALPHABETIC (ALPNUM=1) OR A NUMERIC (ALPNUM=2) IS DESIRED.
C   VAL IS THE INPUT VALUE TO BE USED TO GET THE LABEL.  CHAR IS THE
C   OUTPUT TWO CHARACTERS - LEFT JUSTIFIED WITH BLANK FILL.
C
      INTEGER ALPNUM,VAL,RCHAR,CHAR,R
      IF(ALPNUM.EQ.2) GO TO 10
C
C        MAKE AN ALPHABETIC LABEL
C        ( A, B, C, D,..., Z,AA,AB,...,AZ,BA,BB,...,ZX,ZY,ZZ)
C        I'S AND O'S ARE NOT USED
C
      IF(VAL.LT.1 .OR. VAL.GT.600) CALL ERROR(12), RETURNS(20)
      LCHAR=L=1+(VAL-1)/24
      RCHAR=R=VAL-24*(L-1)
      IF(R.GE.9) RCHAR=RCHAR+1
      IF(R.GE.14) RCHAR=RCHAR+1
      IF(L.EQ.1) LCHAR=55B
      IF(L.GE.2 .AND. L.LE.9) LCHAR=LCHAR-1
      IF(L.GE.15) LCHAR=LCHAR+1
      CHAR=OR(SHIFT(LCHAR,54),SHIFT(RCHAR,48),55555555555555558)
      RETURN
C
C        MAKE A NUMERIC LABEL (01,02,03,...,10,11,12,...,97,98,99)
C
   10 IF(VAL.LT.1 .OR. VAL.GT.99) CALL ERROR(12), RETURNS(20)
      LCHAR=VAL/10
      RCHAR=VAL-10*LCHAR
      CHAR=OR(SHIFT(LCHAR+33B,54),SHIFT(RCHAR+33B,48),55555555555555558)
      RETURN
   20 RETURN AAAAAA
      END
```

```
      SUBROUTINE PACK(AA,NAA,BB,CC), RETURNS(AAAAAA)
C
C   ROUTINE EITHER TO PACK CHARACTERS INTO ARRAY PAGE (10 CHARACTERS PER
C   WORD) OR TO UNPACK CHARACTERS FROM ARRAY PAGE.
C
C   ARRAY PAGE IS USED TO STORE THE DIAGRAM PRIOR TO PRINTOUT.  PAGE(I,J)
C   WILL BE PRINTED ON THE JTH LINE STARTING AT COLUMN 10*(I-1)+1.
C
C   AA IS THE CHARACTER(S) TO BE PACKED INTO PAGE OR UNPACKED FROM PAGE -
C   LEFT JUSTIFIED WITH BLANK FILL
C   NAA IS THE NUMBER OF CHARACTERS IN AA
C   BB IS THE PRINTOUT COLUMN WHERE AA IS TO BE PACKED INTO OR UNPACKED
C   FROM
C   CC IS THE PRINTOUT LINE WHERE AA IS TO BE PACKED INTO OR UNPACKED
C   FROM
C
C   ENTRY POINT PACK PUTS NAA CHARACTERS FROM AA INTO PAGE AT ROW CC AND
C   COLUMNS BB THROUGH BB+NAA-1.
C
C   ENTRY POINT UNPACK GETS NAA CHARACTERS FOR AA FROM PAGE AT ROW CC AND
C   COLUMNS BB THROUGH BB+NAA-1.
C
C
      COMMON /INFO/
     A    CARD(80),ERR(6),SYS(40),ISYS,EQN(4460),PAGE(25,200),VAR(100,2),
     B    EFF,SIDE,
     C    COL(12),EQN1(960),INPUTS(30),OPOR(120,3),OPR(120,34),PAR(50,3),
     D    CHAR,I1,IFM,IK,ITO,JFM,JJ,JTO,LINE,LOPR,NCOL,NEQN,NEQN1,
     E    NIN,NOPR,NPAR,NVAR,A,B,C,D,E,F,G,H,I,J,K,KK,L,LL,M,MM,N,
     F    NN,P,Q,R,S,T,U,V,W,X,Y,Z
      COMMON INDEXM(16001),EQUATN(2300),DATA(400),SIZEM,SIZM,SIZEE,
     A    SIZER,KEY
      INTEGER
     A    CARD,ERR,SYS,EQN,PAGE,VAR,EFF,SIDE,COL,EQN1,OPOR,OPR,PAR,CHAR,
     B    A,B,C,D,E,F,G,H,P,Q,R,S,T,U,V,W,X,Y,Z
      INTEGER EQUATN,DATA,SIZEM,SIZM,SIZEE,SIZER
      INTEGER AA,BB,CC
C
C        FIGURE OUT WHERE TO START IN PAGE - AT PAGE(LL,KK) MM IS THE
C        CHARACTER POSITION WITHIN PAGE(LL,KK) TO START AT - MM=1 IS
C        THE LEFTMOST CHARACTER AND MM=10 IS THE RIGHTMOST CHARACTER
C
      IF(BB.LE.0 .OR. CC.LE.0) CALL ERROR(22), RETURNS(999)
      LL=(BB-1)/10+1
      MM=MOD(BB,10)
      IF(MM.EQ.0) MM=10
      KK=CC
      IF(NAA.LE.0 .OR. NAA.GE.11) CALL ERROR(9), RETURNS(999)
      IF(LL.GT.25 .OR. KK.GT.200) CALL ERROR(20), RETURNS(999)
C
C        PACK AA INTO PAGE
C
C        THE PACKING OF AA INTO PAGE IS ACCOMPLISHED BASED ON ONE OF
C        FIVE POSSIBLE CONDITIONS -
C
```

```
C          STATEMENT  CONDITION
C            LABEL
C               10      MM=1 AND MM+NAA<11
C                       LEFTMOST NAA CHARACTERS OF AA GO INTO PAGE(LL,KK)
C                       STARTING AT THE LEFTMOST POSITION AND PAGE(LL+1,KK)
C                       IS NOT AFFECTED
C               20      MM=1 AND MM+NAA=11
C                       ALL OF AA GOES INTO PAGE(LL,KK) STARTING AT THE
C                       LEFTMOST POSITION AND PAGE(LL+1,KK) IS NOT AFFECTED
C               30      MM>1 AND MM+NAA<11
C                       LEFTMOST NAA CHARACTERS OF AA GO INTO PAGE(KK,LL)
C                       STARTING AT THE MMTH POSITION BUT DO NOT FILL OUT
C                       THE RIGHT END OF PAGE(LL,KK) AND PAGE(LL+1,KK) IS
C                       NOT AFFECTED
C               40      MM>1 AND MM+NAA=11
C                       LEFTMOST NAA CHARACTERS OF AA GO INTO PAGE(LL,KK)
C                       STARTING AT THE MMTH POSITION AND DO FILL OUT
C                       PAGE(LL,KK) AND PAGE(LL+1,KK) IS NOT AFFECTED
C               50      MM>1 AND MM+NAA>11
C                       LEFTMOST NAA CHARACTERS GO INTO PAGE(LL,KK)
C                       STARTING AT THE MMTH POSITION AND RUN INTO
C                       PAGE(LL+1,KK) STARTING AT THE LEFTMOST POSITION
C
C        DETERMINE PROPER CONDITION
C
      IF(MM.EQ.1 .AND. MM+NAA.LT.11) GO TO 10
      IF(MM.EQ.1 .AND. MM+NAA.EQ.11) GO TO 20
      IF(MM.GT.1 .AND. MM+NAA.LT.11) GO TO 30
      IF(MM.GT.1 .AND. MM+NAA.EQ.11) GO TO 40
      GO TO 50
C
   10 Y=10-NAA
      ENCODE(10,1100,Z) NAA,Y
      ENCODE(10,Z,PAGE(LL,KK)) AA,PAGE(LL,KK)
      RETURN
C
   20 PAGE(LL,KK)=AA
      RETURN
C
   30 Y=MM-1
      NN=11-NAA-MM
      ENCODE(10,1200,Z) Y,NAA,NN
      ENCODE(10,Z,PAGE(LL,KK)) PAGE(LL,KK),AA,PAGE(LL,KK)
      RETURN
C
   40 Y=MM-1
      ENCODE(10,1300,Z) Y,NAA
      ENCODE(10,Z,PAGE(LL,KK)) PAGE(LL,KK),AA
      RETURN
C
   50 IF(LL+1.GT.25) CALL ERROR(20), RETURNS(999)
      Y=MM-1
      NN=21-NAA-MM
      ENCODE(10,1200,Z) Y,NAA,NN
      ENCODE(20,Z,PAGE(LL,KK)) PAGE(LL,KK),AA,PAGE(LL+1,KK)
```

```
      RETURN
C
C        ENTRY UNPACK
C
C        WILL START AT PAGE(LL,KK), MM IS THE CHARACTER POSITION WITHIN
C        PAGE(LL,KK) TO START AT - MM=1 IS THE LEFTMOST CHARACTER AND
C        MM=10 IS THE RIGHTMOST CHARACTER
C
      ENTRY UNPACK
      LL=(BB-1)/10+1
      MM=MOD(BB,10)
      IF(MM.EQ.0) MM=10
      KK=CC
      IF(NAA.LE.0 .OR. NAA.GE.11) CALL ERROR(9), RETURNS(999)
      IF(LL.GT.25 .OR. KK.GT.200) CALL ERROR(21), RETURNS(999)
      IF(MM+NAA.GT.11 .AND. LL+1.GT.25) CALL ERROR(21), RETURNS(999)
C
C        SET AA FROM PAGE
C
      Y=MM-1
      ENCODE(10,1000,Z) Y,NAA
      DECODE(20,Z,PAGE(LL,KK)) AA
      RETURN
  999 RETURN AAAAAA
 1000 FORMAT( *(* I1 *X,A* I2 *)* )
 1100 FORMAT( *(A* I1 *,R* I1 *)* )
 1200 FORMAT( *(A* I1 *,A* I1 *,R* I1 *)* )
 1300 FORMAT( *(A* I1 *,A* I1 *)* )
      END
```

```
      SUBROUTINE PARSE, RETURNS(AAAAAA)
C
C  SUBROUTINE TO SEMANTICALLY ANALYZE (PARSE) EQN1(II) THROUGH EQN1(JJ)
C
C
      COMMON /INFO/
     A    CARD(80),ERR(6),SYS(40),ISYS,EQN(4460),PAGE(25,200),VAR(100,2),
     B    EFF,SIDE,
     C    COL(12),EQN1(960),INPUTS(30),OPOR(120,3),OPR(120,34),PAR(50,3),
     D    CHAR,II,IFM,IK,ITO,JFM,JJ,JTO,LINE,LOPR,NCOL,NEQN,NEQN1,
     E    NIN,NOPR,NPAR,NVAR,A,B,C,D,E,F,G,H,I,J,K,KK,L,LL,M,MM,N,
     F    NN,P,Q,R,S,T,U,V,W,X,Y,Z
      COMMON INDEXM(16001),EQUATN(2300),DATA(400),SIZEM,SIZM,SIZEE,
     A    SIZER,KEY
      INTEGER
     A    CARD,ERR,SYS,EQN,PAGE,VAR,EFF,SIDE,COL,EQN1,OPOR,OPR,PAR,CHAR,
     B    A,B,C,D,E,F,G,H,P,Q,R,S,T,U,V,W,X,Y,Z
      INTEGER EQUATN,DATA,SIZEM,SIZM,SIZEE,SIZER
C
C        THE BASIC PROCEDURE IS TO SCAN EQN1(II) THROUGH EQN1(JJ)
C        LEFT TO RIGHT - FIRST FOR TIME DELAYS (TYPE 1 THEN TYPE 2),
C        THEN FOR AND OPERATORS, THEN FOR EXCLUSIVE OR
C        OPERATORS, AND THEN FOR OR OPERATORS.  AS THESE ARE FOUND,
C        THEY AND THEIR INPUTS ARE REMOVED FROM EQN1 AND REPLACED WITH
C        1000+NOPR AND 9999'S.  THE INFORMATION FROM EQN1 IS PLACED
C        IN OPR
C
C        INITIALIZE DO LOOP FOR G = 2010,2011,2003,2004,2005 ($,?,*,@,+)
C
      DO 130 Q=1,5
      G=Q+2000
      IF(G.LE.2002) G=G+9
      E=II-1
      LOPR=NIN=F=CHAR=0
      CALL INIT(INPUTS,30,0)
C
C        LOOK FOR OPERATOR INPUT (EITHER VARIABLE OR OUTPUT OF
C        ANOTHER, PREVIOUSLY PARSED, OPERATOR)
C
   20 E=E+1
      CHAR=EQN1(E)
      IF(CHAR.EQ.9999) GO TO 120
      IF(CHAR.LT.1 .OR. CHAR.GT.2000) CALL ERROR(13), RETURNS(1000)
      NIN=NIN+1
      IF(NIN.GT.30) CALL ERROR(14), RETURNS(1000)
      IF(NIN.EQ.1) F=E
      INPUTS(NIN)=CHAR
C
C        LOOK FOR OPERATOR - G IS CURRENT DESIRED OPERATOR
C
   30 E=E+1
      IF(E.LE.JJ) GO TO 40
      IF(LOPR.EQ.G) 70,130
   40 CHAR=EQN1(E)
      IF(CHAR.EQ.9999) GO TO 30
```

```
      IF(CHAR.NE.2007) GO TO 50
C
C        PRIME FOUND - RESET INPUTS(NIN) AND GET NEXT CHARACTER
C
      IF(INPUTS(NIN).EQ.0) CALL ERROR(18), RETURNS(1000)
      INPUTS(NIN)=-INPUTS(NIN)
      GO TO 30
C
C        TEST CHAR FOR 2003, 2004, 2005
C
   50 IF(CHAR.NE.2003 .AND. CHAR.NE.2004 .AND. CHAR.NE.2005 .AND.
     A   CHAR.NE.2010 .AND. CHAR.NE.2011) CALL ERROR(19), RETURNS(1000)
C
C        TEST FOR DESIRED OPERATOR - IF FOUND THEN SET LOPR AND LOOK
C        FOR MORE OPERATOR INPUTS
C
      IF(CHAR.NE.G) GO TO 60
      LOPR=CHAR
      GO TO 120
C
C        DESIRED OPERATOR NOT FOUND - IF PREVIOUS OPERATOR (LOPR)
C        WAS NOT THE DESIRED OPERATOR, THEN CLEANOUT NIN AND INPUTS
C        AND START OVER LOOKING FOR OPERANDS
C
   60 IF(LOPR.NE.G) GO TO 100
C
C        LOPR WAS THE DESIRED OPERATOR - CURRENT OPERATOR IS NOT NOW
C        DESIRED, THEREFORE EQN1(F) THROUGH EQN1(E-1) CAN BE PUT INTO
C        OPR AND REMOVED FROM EQN1
C
   70 LOPR=0
      NOPR=NOPR+1
      IF(NOPR.GT.120) CALL ERROR(5), RETURNS(1000)
      OPR(NOPR,1)=G
      OPR(NOPR,2)=NIN
      DO 80 P=1,NIN
   80 OPR(NOPR,P+4)=INPUTS(P)
      CALL INIT(EQN1(F+1),E-F-1,9999)
      EQN1(F)=1000+NOPR
C
C        CLEANOUT NIN AND INPUTS
C
  100 NIN=F=0
      CALL INIT(INPUTS,30,0)
C
C        TEST E TO SEE IF DONE WITH THIS PASS THROUGH EQN1
C
  120 IF(E.LT.JJ) GO TO 20
C
C        DONE WITH THIS OPERATOR
C
  130 CONTINUE
C
C        EITHER RETURN OR TAKE CARE OF EQUAL OPERATOR
C
```

```
      IF(II.NE.3) RETURN
C
C        TAKE CARE OF EQUAL OPERATOR
C
      IF(EQN1(1).LT.1 .OR. EQN1(1).GT.1000) CALL ERROR(15),
     A   RETURNS(1000)
      IF(EQN1(2).NE.2006) CALL ERROR(16), RETURNS(1000)
      IF(NOPR.EQ.0 .AND. INPUTS(1).EQ.0) CALL ERROR(17), RETURNS(1000)
      IF(INPUTS(1).LT.0) EQN1(2)=-2006
      RETURN
C
C        ERROR ENCOUNTERED - USE NON-STANDARD RETURN
C
 1000 RETURN AAAAAA
      END
```

```
      SUBROUTINE READ4, RETURNS(AAAAAA,BBBBBB)
C
C   ROUTINE TO GET NEXT EQUATION FROM TAPE4 - LOADED INTO EQN - 1
C   CHARACTER PER WORD LEFT JUSTIFIED WITH BLANK FILL.
C
C   STANDARD RETURN IS USED WHEN THE DESIRED EQUATION IS FOUND WITH NO
C   PROBLEMS
C
C   RETURN AAAAAA IS USED WHEN THE EQUATION IS TOO LARGE FOR EQN
C
C   RETURN BBBBBB IS USED WHEN AN END OF FILE IS ENCOUNTERED
C
C
      COMMON /INFO/
     A    CARD(80),ERR(6),SYS(40),ISYS,EQN(4460),PAGE(25,200),VAR(100,2),
     B    EFF,SIDE,
     C    COL(12),EQN1(960),INPUTS(30),OPOR(120,3),OPR(120,34),PAR(50,3),
     D    CHAR,II,IFP,IK,ITO,JFM,JJ,JTO,LINE,LOPR,NCOL,NEQN,NEQN1,
     E    NIN,NOPR,NPAR,NVAR,A,B,C,D,E,F,G,H,I,J,K,KK,L,LL,M,MM,N,
     F    NN,P,Q,R,S,T,U,V,W,X,Y,Z
      COMMON INDEXM(16001),EQUATN(2300),DATA(400),SIZEM,SIZM,SIZEE,
     A    SIZER,KEY
      INTEGER
     A    CARD,ERR,SYS,EQN,PAGE,VAR,EFF,SIDE,COL,EQN1,OPOR,OPR,PAR,CHAR,
     B    A,B,C,D,E,F,G,H,P,Q,R,S,T,U,V,W,X,Y,Z
      INTEGER EQUATN,DATA,SIZEM,SIZM,SIZEE,SIZER
C
C        TEST CARD(1) TO SEE IF PROGRAM SHOULD STOP, IF A CARD SHOULD
C        BE READ, OR IF EQN SHOULD BE LOADED
C
      IF(CARD(1).EQ.-1) RETURN BBBBBB
   10 IF(CARD(1).EQ.1H1) GO TO 30
C
C        READ A CARD
C
   20 READ(4,1000) CARD
      IF(EOF(4).NE.0.) RETURN BBBBBB
      GO TO 10
C
C        SEE IF A DESIRED EQUATION HAS BEEN FOUND
C
   30 ENCODE(10,6000,H) CARD(2),CARD(3),CARD(4),CARD(5),CARD(6),CARD(7),
     A    CARD(8),CARD(9)
      ENCODE(10,2000,U) H
      ENCODE(10,4000,V) H
      ENCODE(10,5000,W) H
      DO 40 X=1,40
   40 IF(H.EQ.SYS(X) .OR. U.EQ.SYS(X) .OR. V.EQ.SYS(X) .OR. W.EQ.SYS(X))
     A    GO TO 50
C
C        DO NOT WANT THIS EQUATION - TRY NEXT ONE
C
      GO TO 20
C
C        WANT THIS EQUATION - SET EFFECTIVITY (EFF) AND EMUX SECTION
```

```
C         (SIDE) AND STORE EQUATION, WITHOJT BLANKS, IN EQN
C
   50 ENCODE(2,3000,EFF) CARD(74),CARD(75)
      IF(CARD(80).EQ.1HL) SIDE=5H LEFT
      IF(CARD(80).EQ.1HR) SIDE=5HRIGHT
      H=2
      NEQN=0
   60 IF(CARD(H).NE.1H ) GO TO 70
      IF(H.EQ.69) GO TO 80
      H=H+1
      GO TO 60
   70 IF(NEQN.EQ.4460) CALL ERROR(10), RETURNS(999)
      NEQN=NEQN+1
      EQN(NEQN)=CARD(H)
      IF(H.EQ.69) GO TO 80
      H=H+1
      GO TO 60
C
C         END OF CARD - READ NEXT ONE TO SEE IF EQUATION IS CONTINUED
C
   80 READ(4,1000) CARD
      IF(EOF(4).NE.0.) GO TO 100
      IF(CARD(1).EQ.1H1) GO TO 90
      H=11
      GO TO 60
   90 IF(NEQN.NE.0) RETURN
      H=2
      GO TO 60
  100 IF(NEQN.EQ.0) RETURN BBBBBB
      CARD(1)=-1
      RETURN
  999 RETURN AAAAAA
 1000 FORMAT(80A1)
 2000 FORMAT(A4)
 3000 FORMAT(2A1)
 4000 FORMAT(A3*0*)
 5000 FORMAT(A2*00*)
 6000 FORMAT(8A1)
      END
```


*U.S.Government Printing Office: 1978 — 757-080/757