United States Patent [19]

Spangler et al.

[11] **4,012,725**

[45] **Mar. 15, 1977**

[54] **PROGRAMMABLE CALCULATOR**

[75] Inventors: **Richard M. Spangler; Eugene V. Burmeister; Frank E. Cada; Wayne F. Covington; Chris J. Christopher; Myles A. Judd; Freddie W. Wenninger; Robert E. Watson; Kent W. Simcoe,** all of Loveland, Colo.

[73] Assignee: **Hewlett-Packard Company,** Palo Alto, Calif.

[22] Filed: **May 30, 1974**

[21] Appl. No.: **469,727**

Related U.S. Application Data

[63] Continuation of Ser. No. 269,899, July 7, 1972, abandoned.

[52] **U.S. Cl.** **340/172.5;** 235/152

[51] **Int. Cl.²** **G06F 15/40;** G06F 15/02; G06F 3/14; G06F 7/38

[58] **Field of Search** ..... 340/172.5, 324 A, 324 AD; 235/156, 152

[56] **References Cited**

UNITED STATES PATENTS

3,248,705	4/1966	Dammann	340/172.5
3,355,714	11/1967	Culler	340/172.5
3,364,473	1/1968	Reitz	340/172.5
3,389,404	6/1968	Koster	340/172.5
3,501,746	3/1970	Vosbury	340/172.5
3,505,665	4/1970	Lasoff	340/324
3,581,290	5/1971	Sugarman	340/172.5
3,610,902	10/1971	Rahenkamp	340/172.5
3,618,032	11/1971	Goldsberry	340/172.5
3,648,245	3/1972	Dodds	340/172.5
3,699,531	10/1972	Heiman	340/172.5
3,760,171	9/1973	Wang	340/172.5
3,764,986	10/1973	Spademan	340/172.5

Primary Examiner—Gareth D. Shaw
Assistant Examiner—James D. Thomas
Attorney, Agent, or Firm—Roland I. Griffin; William E. Hein

[57] **ABSTRACT**

An adaptable programmable calculator is provided by employing a modular read-write and read-only memory unit capable of being expanded to provide the calculator with additional program and data storage functions oriented towards the environment of the user, a central processing unit capable of performing both serial binary and parallel binary-coded-decimal arithmetic, and an input-output control unit capable of bidirectionally transferring information between the memory or central processing units and a number of input and output units. The memory, central processor, and input-output control units are controlled by a microprocessor included in the central processing unit.

The input and output units include a keyboard input unit with a plurality of alphanumeric keys, a magnetic tape cassette reading and recording unit capable of bidirectionally transferring programs and data between a magnetic tape and the calculator and, a solid state output display unit capable of displaying every alphabetic and numeric character and many other symbols individually or in combination. All of these input and output units are included within the calculator itself. An output printer, an X-Y plotter, a typewriter, a teletypewriter, a magnetic or paper tape reading and recording unit, an extended read-write memory unit, a magnetic disc reading and recording unit, a modem for connecting the calculator via telephone lines to remotely located computer, and many other peripheral input and output units may also be employed with the calculator.

The calculator may be operated manually by the user from the keyboard input unit or automatically by a program stored within the memory unit to perform calculations and provide an output indication of the results thereof. It may also be employed to load programs into the memory unit from the keyboard input unit, to separately or collectively transfer data and programs bidirectionally between the memory unit and an external magnetic tape and to code programs or sections thereof stored in the memory unit as being secure when they are transferred to an external magnetic tape, thereby preventing users of the calculator from again transferring them to an external magnetic tape or obtaining any indication of the individual program steps once they are reloaded into the calculator. In addition, the calculator may be employed to edit

programs stored in the memory unit and to print out program lists, labels, and messages.

The calculator employs an extended version of BASIC computer language and allows the user to enter a line comprising an alphanumeric statement into the calculator from the keyboard input unit while visually observing an alphanumeric display of that line to check for errors therein, permitting the user to case the entered lines to be immediately executed by the calculator or stored as part of a program within the memory unit, and permitting the user to subsequently recall the executed or stored line so that it may be reinspected, reevaluated, and, if necessary, edited and executed or re-executed, or restored in edited form. Any entered or recalled information may be edited by employing the keyboard input unit to selectively delete or replace incorrect or undesired portions of the information or to selectively insert corrected or previously omitted portions thereof on a line-by-line or character-by-character basis. Syntax errors are automatically detected by the calculator when the entered statement is terminated, and execution errors are automatically detected upon attempted execution of the statement or statements. Both types of errors are indicated to the user via error messages displayed by the output display unit. In the event the calculator is being used in combination ith an external printer, unit indications of syntax or execution errors may, if desired, be printed.

The calculator employs a compiler for converting each statement entered into the calculator in BASIC language into an internal stored format. It also employs an uncompiler for generating in the BASIC language statement any entered line converter to the internal stored format. The compiler and uncompiler operate on a line-by-line basis.

The magnetic tape cassette reading and recording unit employed in the calculator allows the user to chain together several program segments and allows program manipulation of several blocks of data on an individual basis, thereby providing more efficient utilization of the available calculator memory. An interrupt feature of the cassette unit facilitates searching for a particular file located on a magnetic tape at the same time the calculator is performing other functions.

41 Claims, 277 Drawing Figures

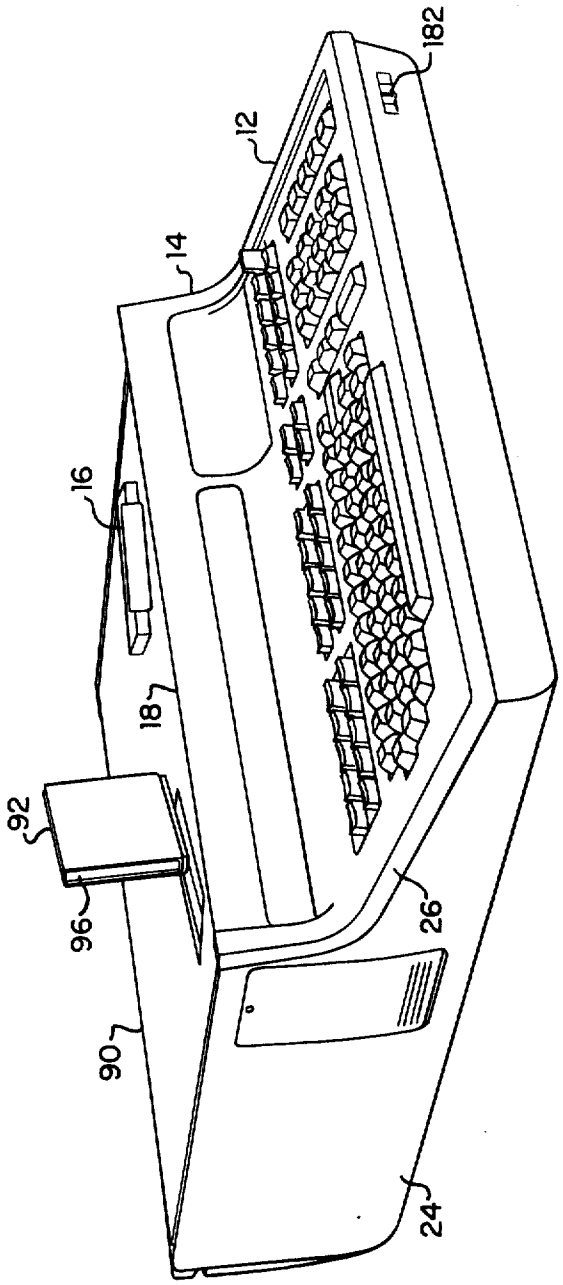


Figure 1

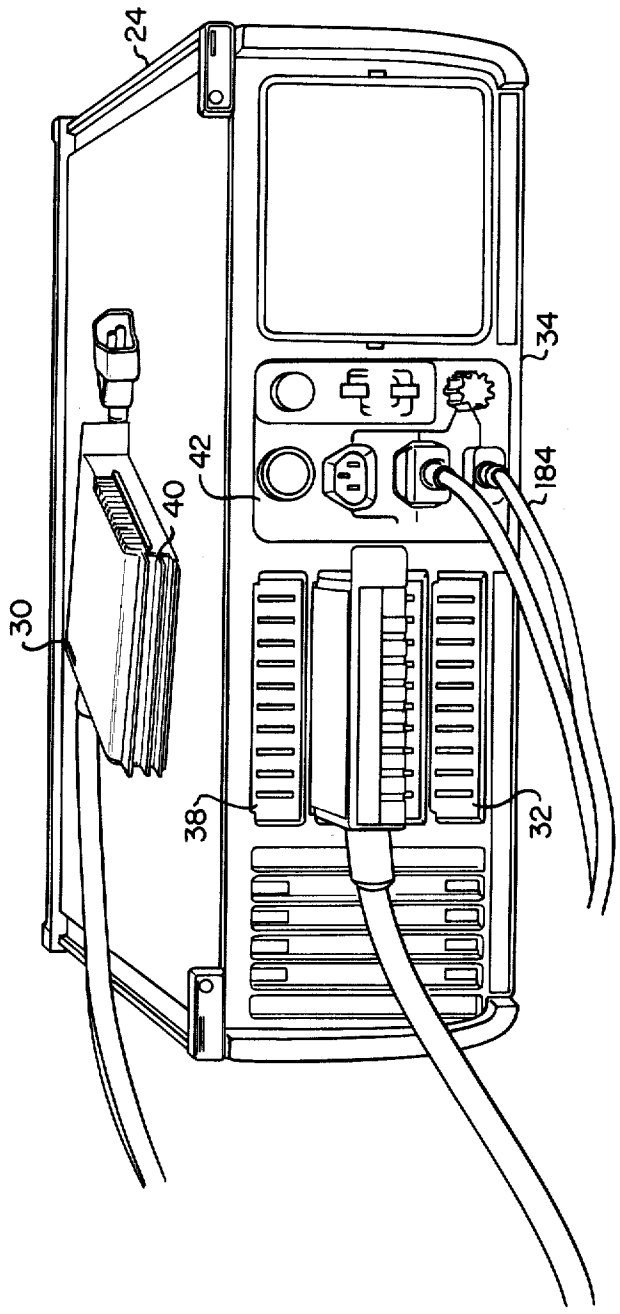


Figure 2

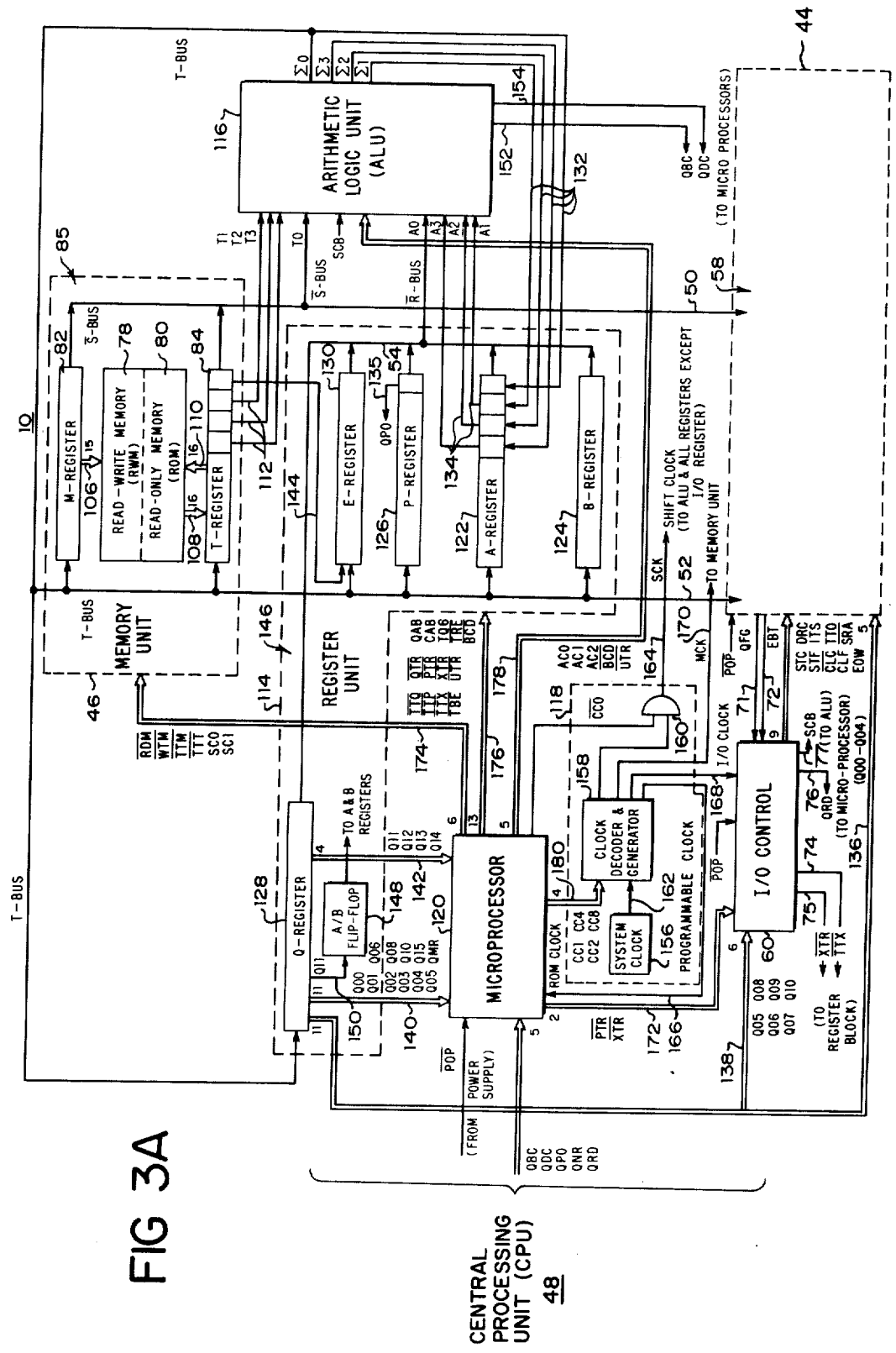

ARITHMETIC LOGIC UNIT (ALU)
116
154
152
132
T-BUS
Σ0
Σ3
Σ2
Σ1
T1
T2
T3
T0
S̄-BUS
SCB
R̄-BUS
A0
A3
A2
A1
QBC
QDC
(TO MICRO PROCESSORS)
44
50
58
10
85
82
M-REGISTER
106
15
78
READ-WRITE MEMORY (RWM)
80
READ-ONLY MEMORY (ROM)
84
110
16
108
T-REGISTER
112
144
130
E-REGISTER
135
54
126
OPO
P-REGISTER
134
122
A-REGISTER
124
B-REGISTER
SHIFT CLOCK (TO ALU & ALL REGISTERS EXCEPT I/O REGISTER)
SCK
52
TO MEMORY UNIT
MEMORY UNIT
46
146
114
REGISTER UNIT
RDM
WTM
TTM
TTT
SC0
SC1
QAB
CAB
I06
TBE
BCD
QTR
PTR
XTR
UTR
TTO
TTP
TTX
TBE
178
ACO
AC1
AC2
BCD
UTR
164
118
CCO
170
MCK
160
POP
QFG
EBT
71
72
9
STC
STF
CLC
CLF
DRC
ITS
TTO
SRA
EOW
5
SCB
(TO ALU)
77
I/O CLOCK
168
I/O CONTROL
76
74
75
60
XTR
TTX
ORD
(TO MICRO-PROCESSOR)
136
(000-004)
Q05
Q06
Q07
Q08
Q09
Q10
(TO REGISTER BLOCK)
138
158
CLOCK DECODER & GENERATOR
162
PROGRAMMABLE CLOCK
156
SYSTEM CLOCK
ROM CLOCK
CC1
CC2
CC4
CC8
180
MICROPROCESSOR
120
142
148
A/B FLIP-FLOP
Q11
Q00
Q01
Q02
Q03
Q04
Q05
QMR
Q06
Q08
Q10
Q15
Q12
Q13
Q14
TO A&B REGISTERS
128
Q-REGISTER
150
140
POP
PTR
XTR
172
166
POWER SUPPLY
(FROM
QBC
QDC
QPO
QNR
QRD
CENTRAL PROCESSING UNIT (CPU)
48

FIG 3A

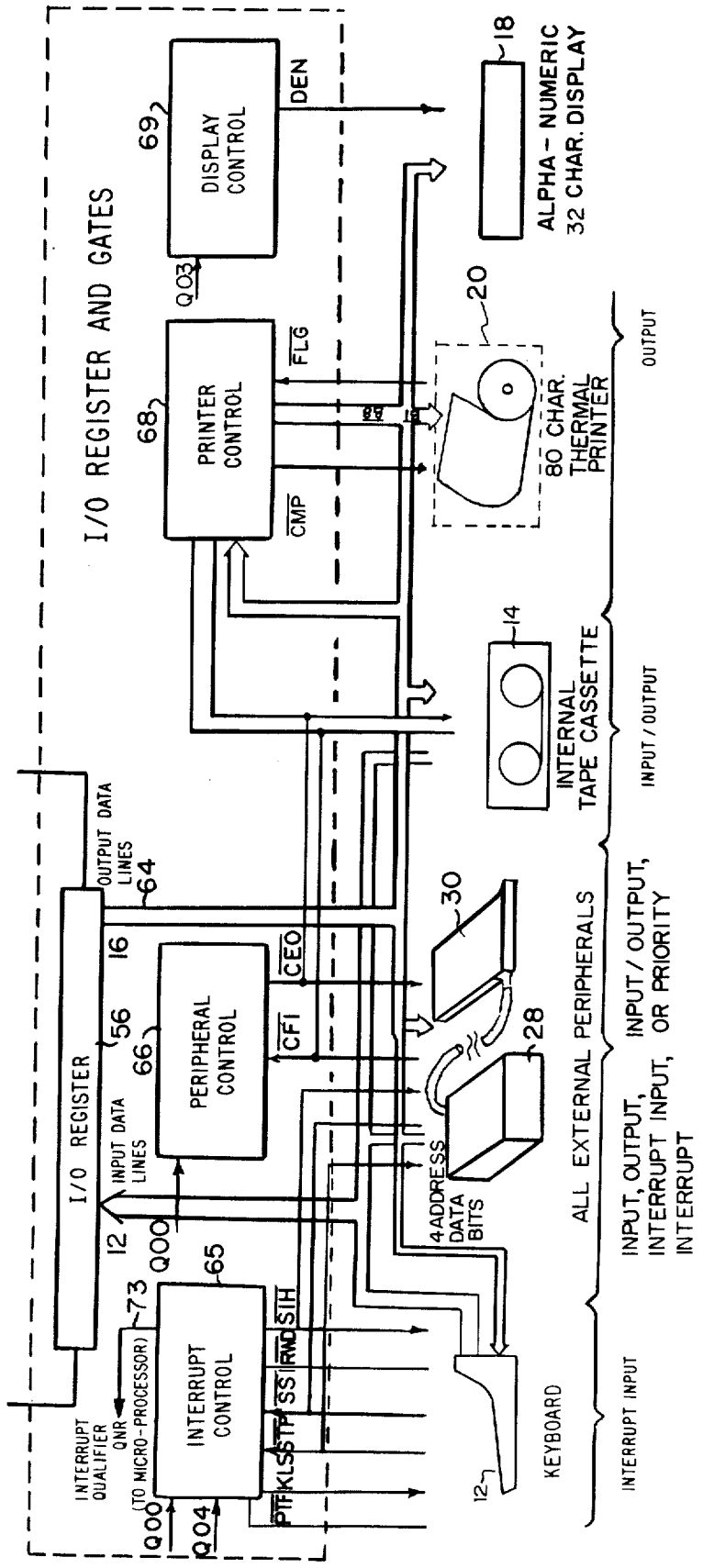

I/O REGISTER AND GATES
DISPLAY CONTROL
69
DEN
Q03
18
ALPHA - NUMERIC
32 CHAR. DISPLAY
PRINTER CONTROL
68
FLG
CMP
20
80 CHAR. THERMAL PRINTER
OUTPUT
INTERNAL TAPE CASSETTE
14
INPUT / OUTPUT
OUTPUT DATA LINES
64
I/O REGISTER
16
56
INPUT DATA LINES
PERIPHERAL CONTROL
66
CEO
CFI
30
28
4 ADDRESS DATA BITS
12
Q00
65
INTERRUPT CONTROL
73
INTERRUPT QUALIFIER
QNR
(TO MICRO-PROCESSOR)
Q00
Q04
PTF
KLS
STP
SS
RND
SIH
ALL EXTERNAL PERIPHERALS
INPUT, OUTPUT, INTERRUPT INPUT, OR PRIORITY INTERRUPT
INPUT / OUTPUT
KEYBOARD
12
INTERRUPT INPUT

FIG 3B

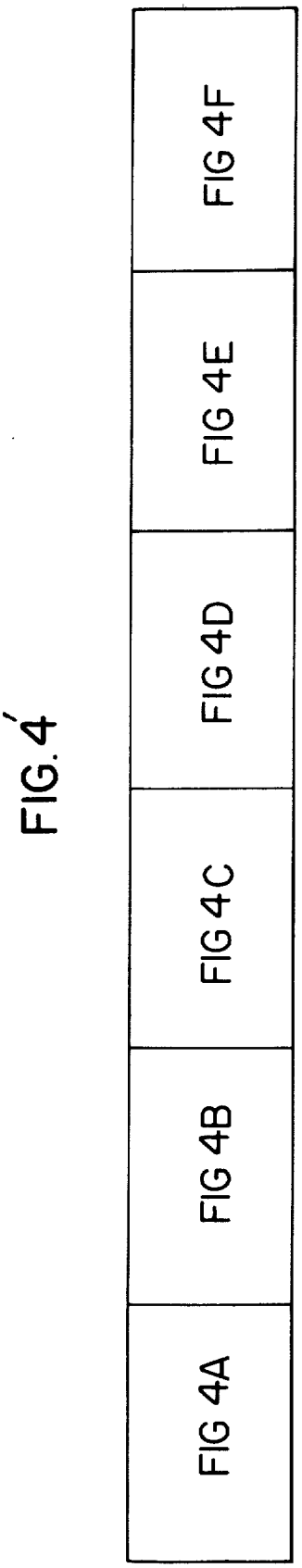


FIG. 4'

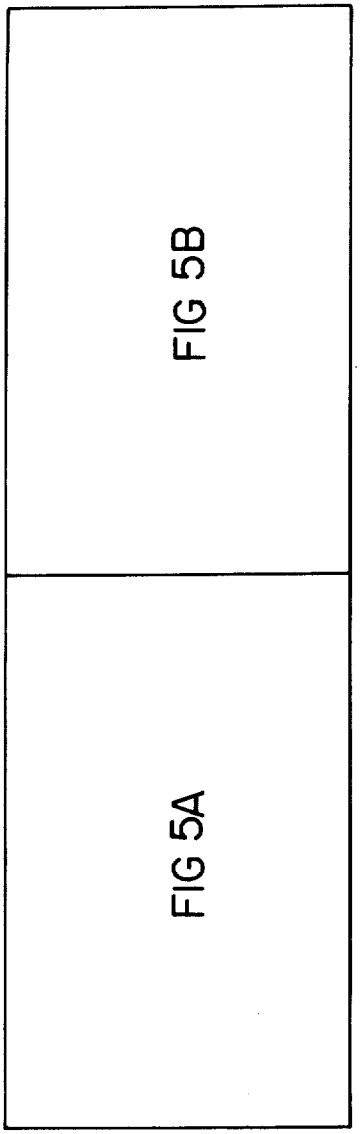


FIG. 5'

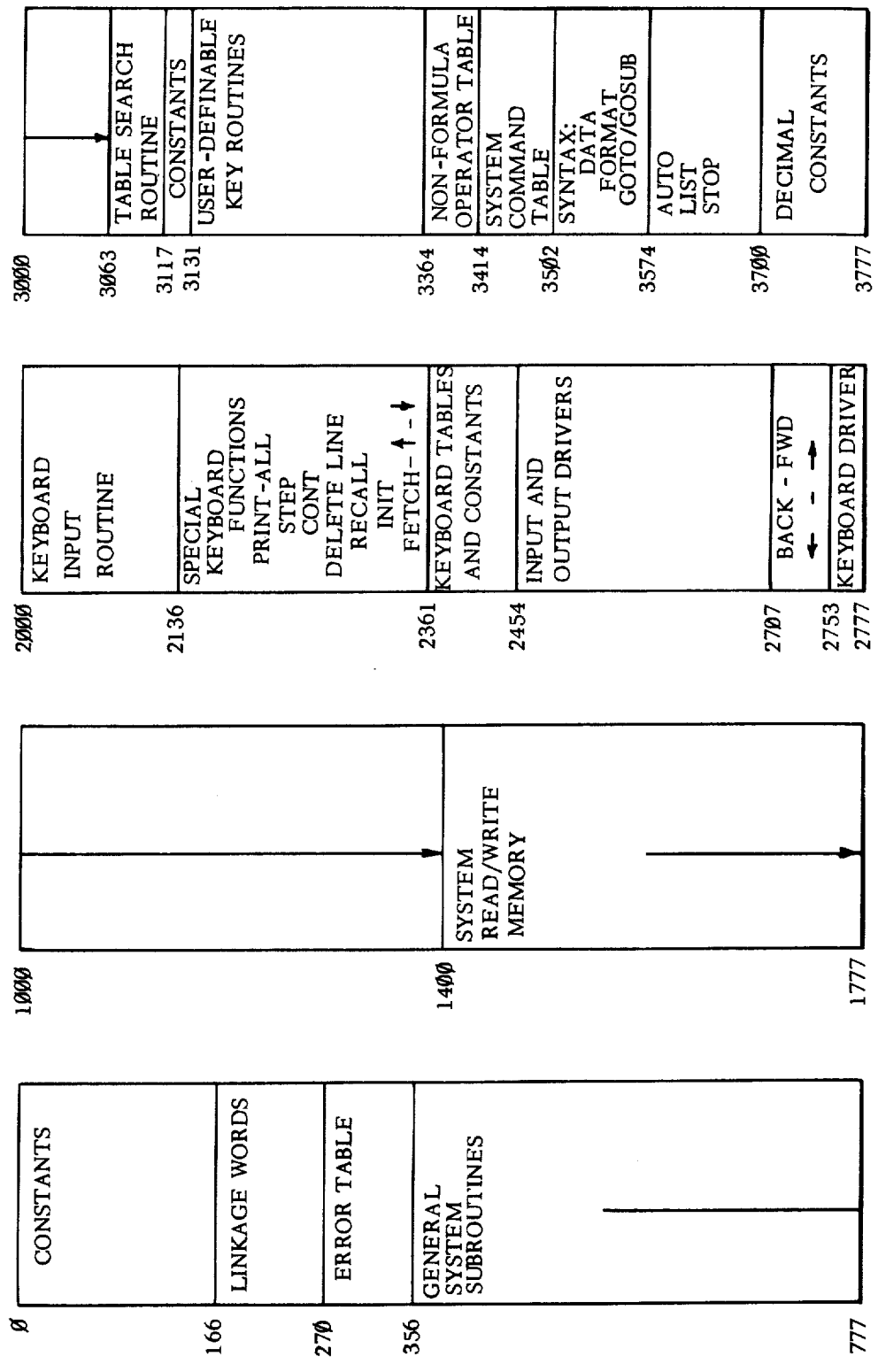

Ø
CONSTANTS
166
LINKAGE WORDS
270
ERROR TABLE
356
GENERAL SYSTEM SUBROUTINES
777
1000
1400
SYSTEM READ/WRITE MEMORY
1777
2000
KEYBOARD INPUT ROUTINE
2136
SPECIAL KEYBOARD FUNCTIONS
PRINT-ALL
STEP
CONT
DELETE LINE
RECALL
INIT
FETCH- ↑ - ↓
2361
KEYBOARD TABLES AND CONSTANTS
2454
INPUT AND OUTPUT DRIVERS
2707
BACK - FWD ← - →
2753
KEYBOARD DRIVER
2777
3000
3063
TABLE SEARCH ROUTINE
3117
CONSTANTS
3131
USER-DEFINABLE KEY ROUTINES
3364
NON-FORMULA OPERATOR TABLE
3414
SYSTEM COMMAND TABLE
3502
SYNTAX: DATA FORMAT GOTO/GOSUB
3574
AUTO
LIST
STOP
3700
DECIMAL CONSTANTS
3777

FIG 4A

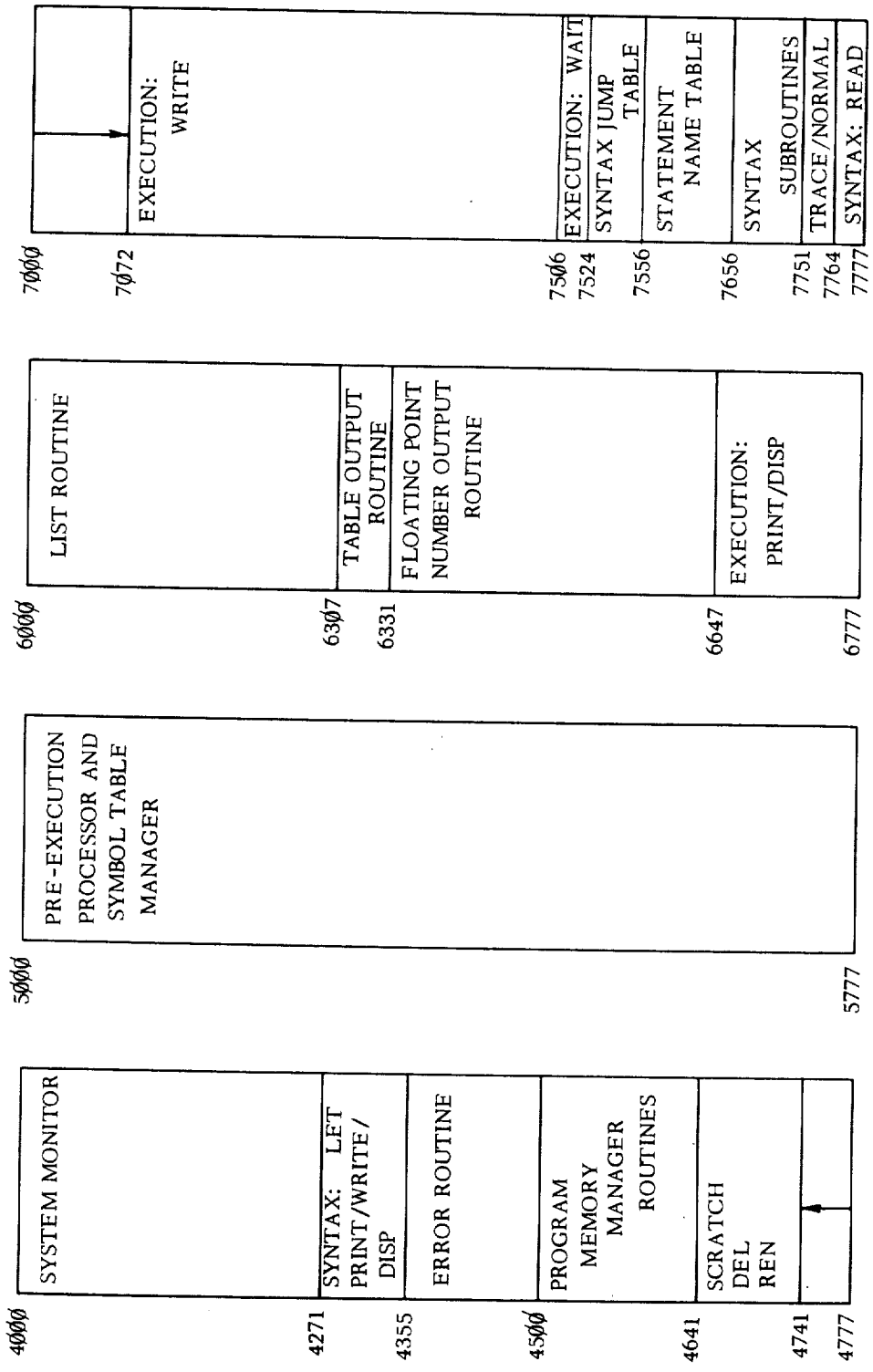

4000
SYSTEM MONITOR
4271
SYNTAX: LET
PRINT/WRITE/
DISP
4355
ERROR ROUTINE
4500
PROGRAM
MEMORY
MANAGER
ROUTINES
4641
SCRATCH
DEL
REN
4741
4777
5000
PRE-EXECUTION
PROCESSOR AND
SYMBOL TABLE
MANAGER
5777
6000
LIST ROUTINE
6307
TABLE OUTPUT
ROUTINE
6331
FLOATING POINT
NUMBER OUTPUT
ROUTINE
6647
EXECUTION:
PRINT/DISP
6777
7000
7072
EXECUTION:
WRITE
7506
7524
EXECUTION: WAIT
SYNTAX JUMP
TABLE
7556
STATEMENT
NAME TABLE
7656
SYNTAX
SUBROUTINES
7751
7764
TRACE/NORMAL
7777
SYNTAX: READ

FIG 4B

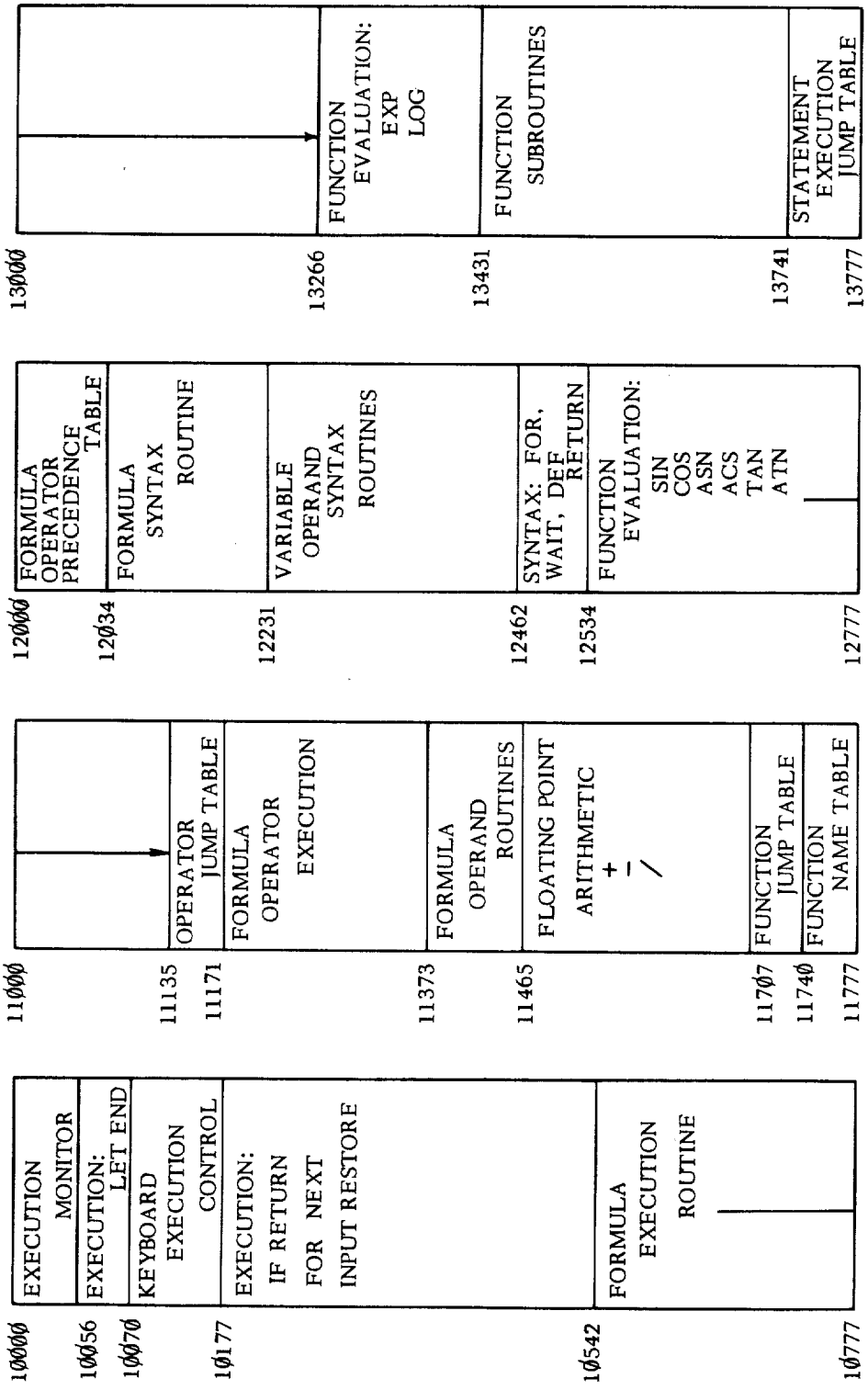

10000
EXECUTION MONITOR
10056
EXECUTION: LET END
10070
KEYBOARD EXECUTION CONTROL
10177
EXECUTION: IF RETURN FOR NEXT INPUT RESTORE
10542
FORMULA EXECUTION ROUTINE
10777
11000
11135
OPERATOR JUMP TABLE
11171
FORMULA OPERATOR EXECUTION
11373
FORMULA OPERAND ROUTINES
11465
FLOATING POINT ARITHMETIC + - / *
11707
FUNCTION JUMP TABLE
11740
FUNCTION NAME TABLE
11777
12000
12034
FORMULA OPERATOR PRECEDENCE TABLE
12231
FORMULA SYNTAX ROUTINE
12462
VARIABLE OPERAND SYNTAX ROUTINES
12534
SYNTAX: FOR, WAIT, DEF RETURN
12777
FUNCTION EVALUATION: SIN COS ASN ACS TAN ATN
13000
13266
FUNCTION EVALUATION: EXP LOG
13431
FUNCTION SUBROUTINES
13741
STATEMENT EXECUTION JUMP TABLE
13777

FIG 4C

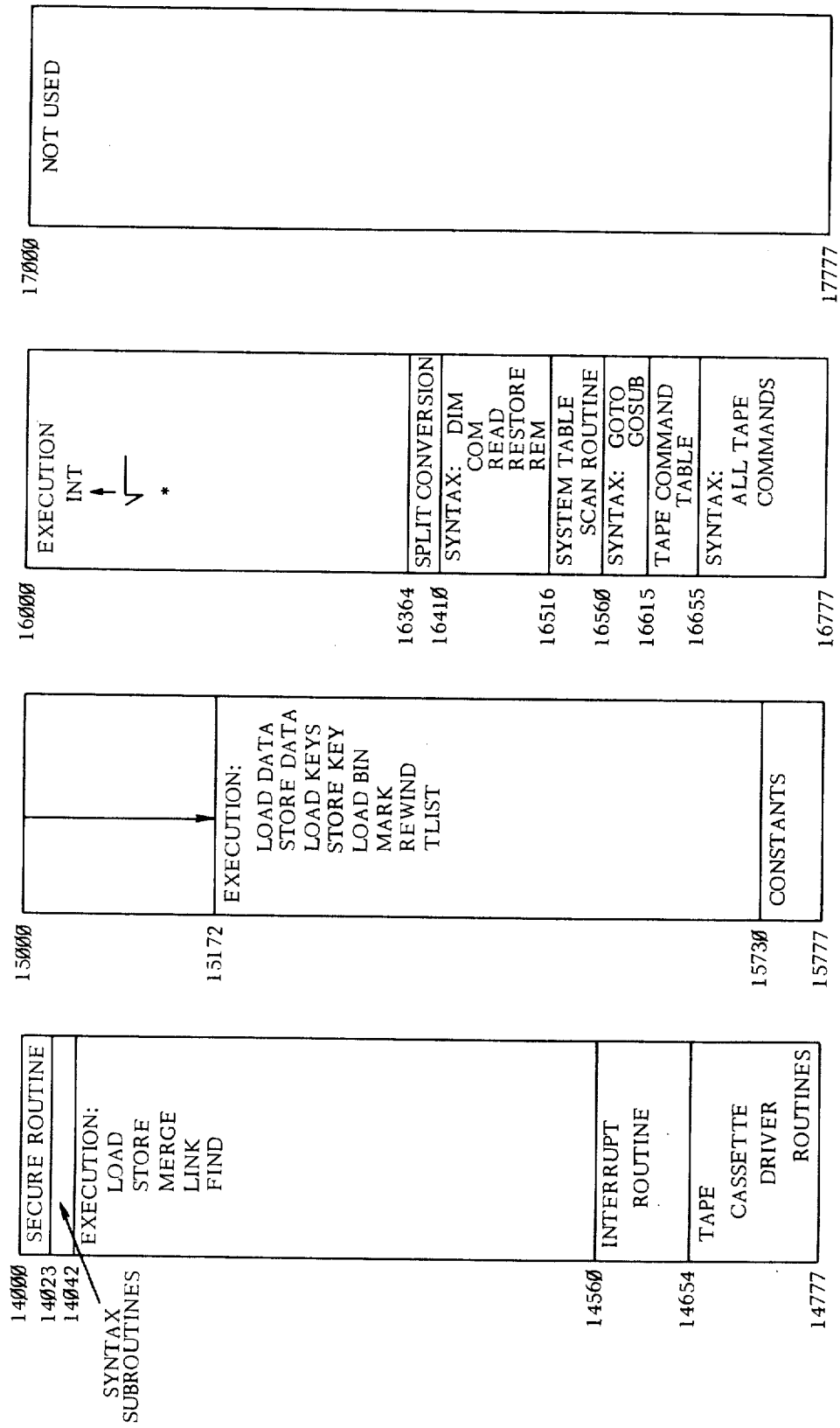

NOT USED
17000
17777
EXECUTION
INT
SPLIT CONVERSION
SYNTAX: DIM
COM
READ
RESTORE
REM
SYSTEM TABLE
SCAN ROUTINE
SYNTAX: GOTO
GOSUB
TAPE COMMAND
TABLE
SYNTAX:
ALL TAPE
COMMANDS
16000
16364
16410
16516
16560
16615
16655
16777
EXECUTION:
LOAD DATA
STORE DATA
LOAD KEYS
STORE KEY
LOAD BIN
MARK
REWIND
TLIST
CONSTANTS
15000
15172
15730
15777
SECURE ROUTINE
EXECUTION:
LOAD
STORE
MERGE
LINK
FIND
SYNTAX
SUBROUTINES
INTERRUPT
ROUTINE
TAPE
CASSETTE
DRIVER
ROUTINES
14000
14023
14042
14560
14654
14777

FIG 4D

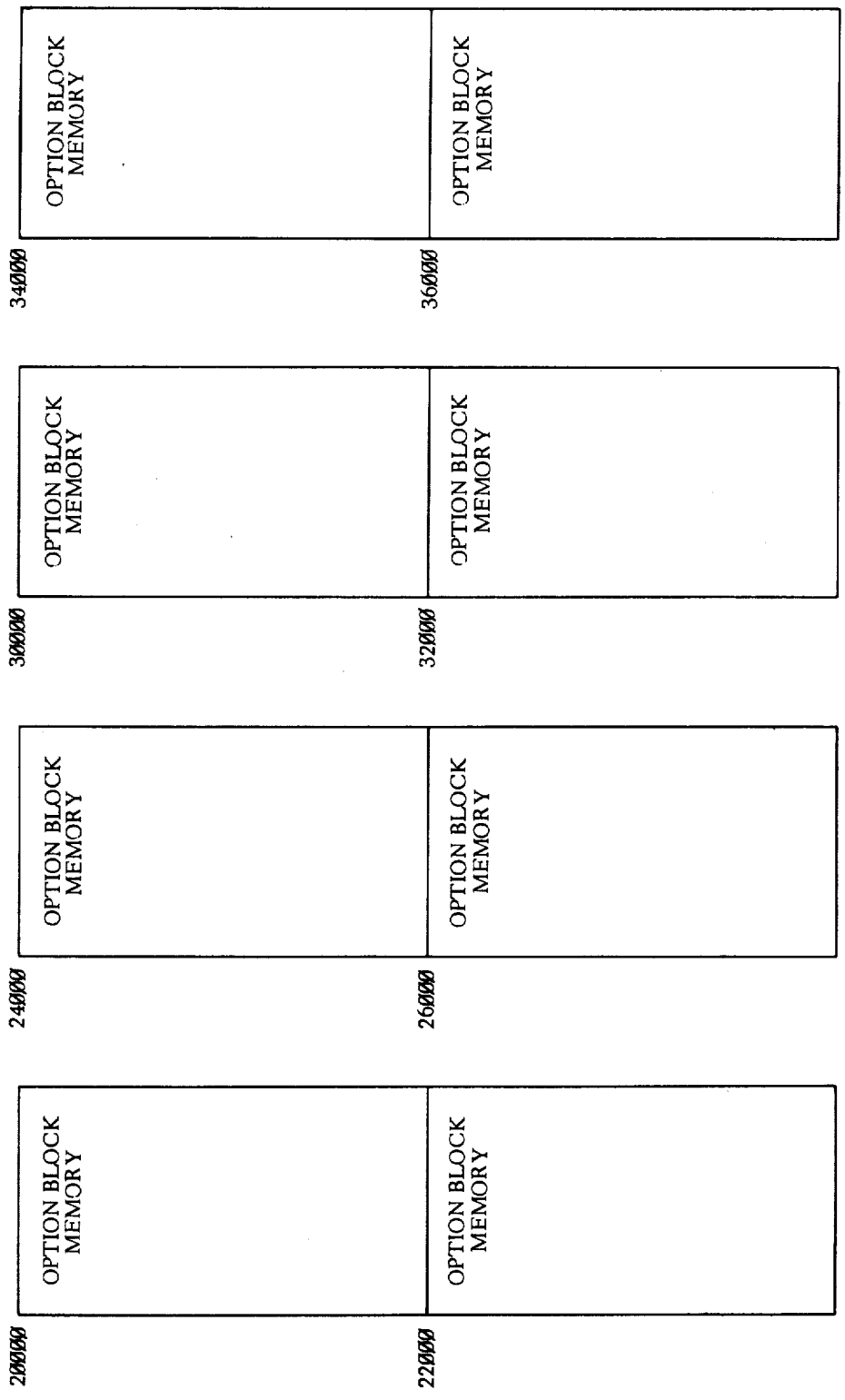

OPTION BLOCK MEMORY
OPTION BLOCK MEMORY
OPTION BLOCK MEMORY
OPTION BLOCK MEMORY
OPTION BLOCK MEMORY
OPTION BLOCK MEMORY
OPTION BLOCK MEMORY
OPTION BLOCK MEMORY

FIG 4E

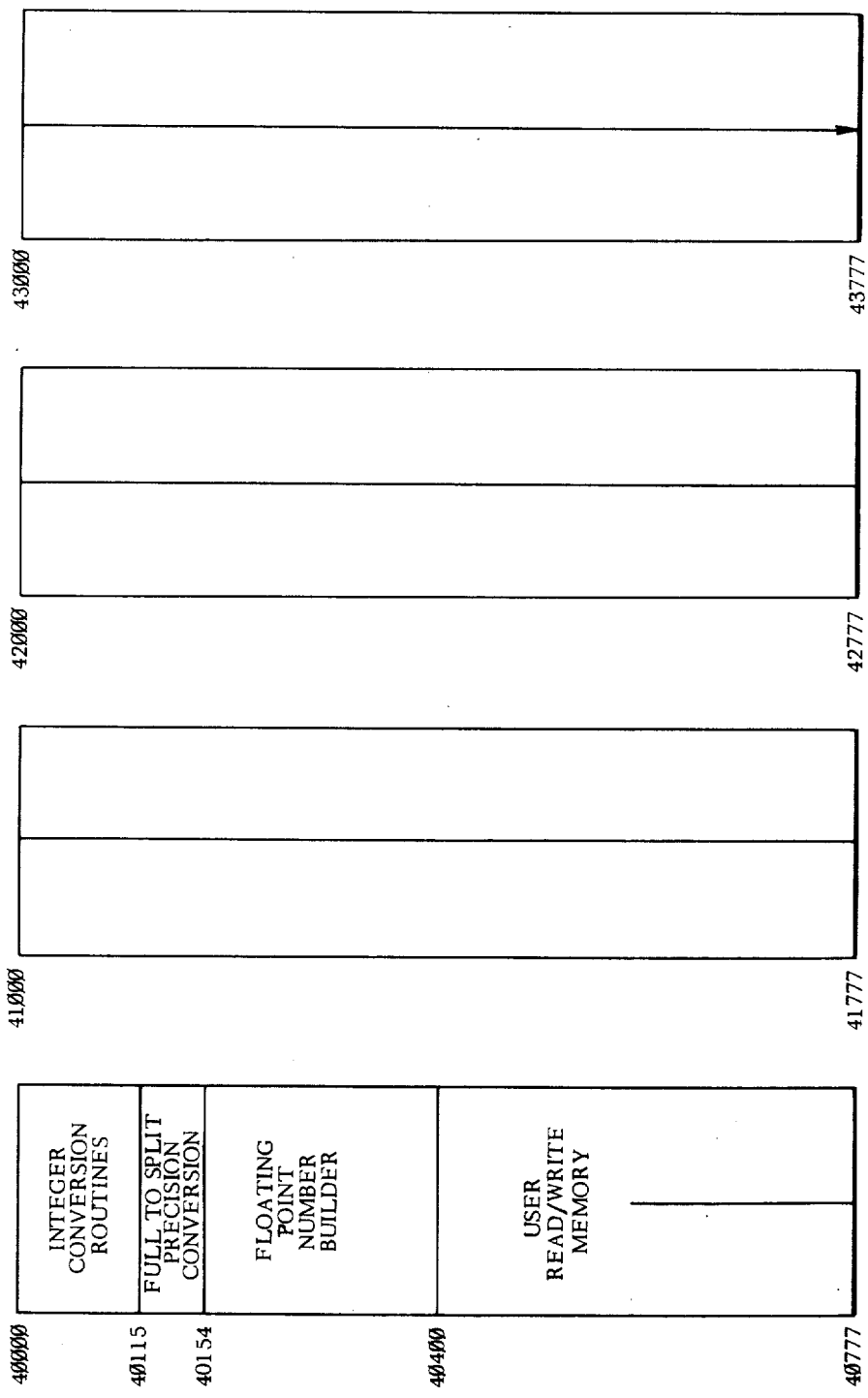

400000
INTEGER CONVERSION ROUTINES
401154
FULL TO SPLIT PRECISION CONVERSION
401154
FLOATING POINT NUMBER BUILDER
404000
USER READ/WRITE MEMORY
407777
410000
417777
420000
427777
430000
437777

FIG 4F

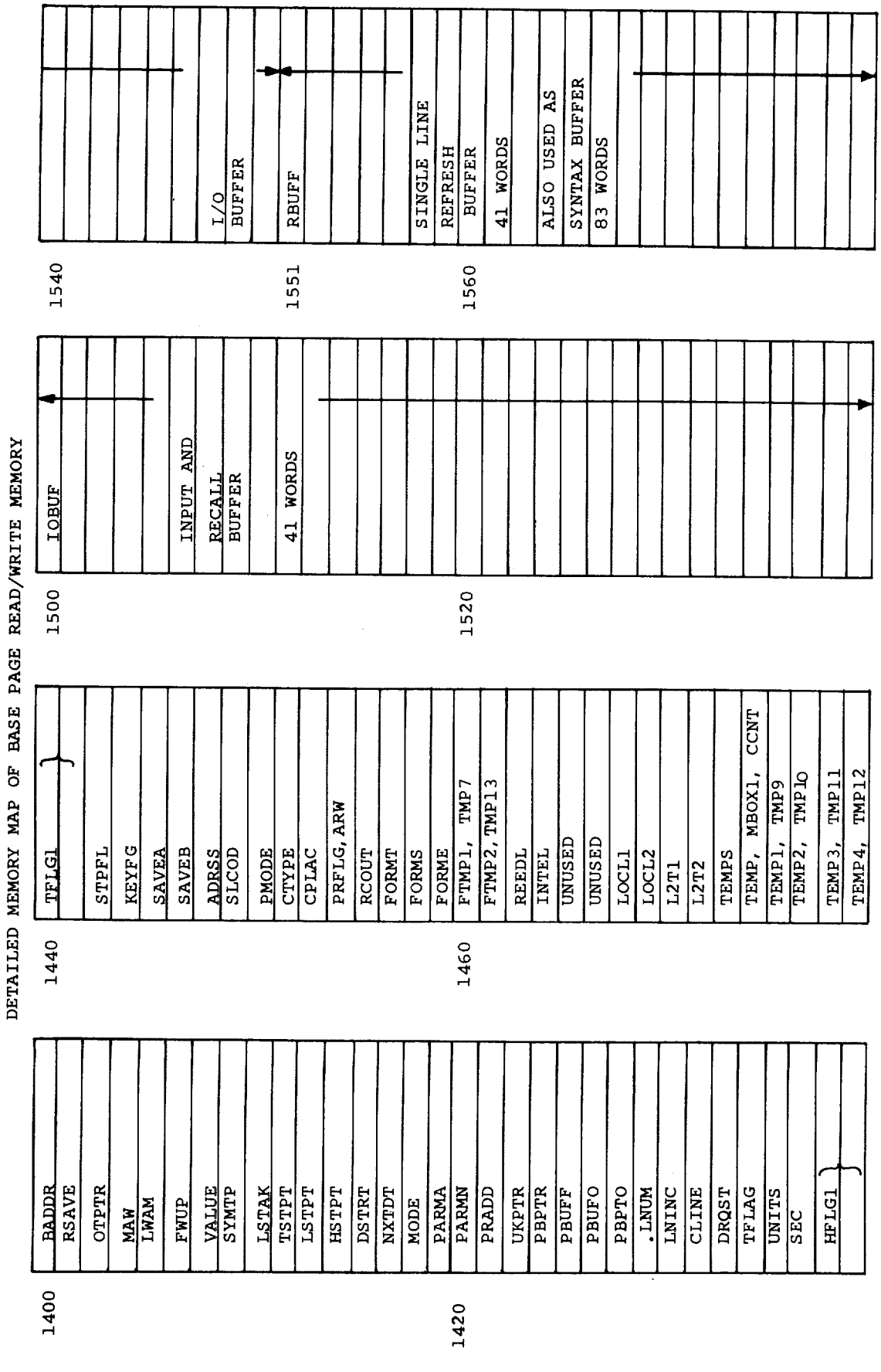

DETAILED MEMORY MAP OF BASE PAGE READ/WRITE MEMORY
1400
1420
BADDR
RSAVE
OTPTR
MAW
LWAM
FWUP
VALUE
SYMTP
LSTAK
TSTPT
LSTPT
HSTPT
DSTRT
NXTDT
MODE
PARMA
PARMN
PRADD
UKPTR
PBPTR
PBUFF
PBUFO
PBPTO
.LNUM
LNINC
CLINE
DRQST
TFLAG
UNITS
SEC
HFLG1
1440
1460
TFLG1
STPFL
KEYFG
SAVEA
SAVEB
ADRSS
SLCOD
PMODE
CTYPE
CPLAC
PRFLG,ARW
RCOUT
FORMT
FORMS
FORME
FTMP1, TMP7
FTMP2,TMP13
REEDL
INTEL
UNUSED
UNUSED
LOCL1
LOCL2
L2T1
L2T2
TEMPS
TEMP, MBOX1, CCNT
TEMP1, TMP9
TEMP2, TMP10
TEMP3, TMP11
TEMP4, TMP12
1500
1520
IOBUF
INPUT AND
RECALL
BUFFER
41 WORDS
1540
1551
1560
I/O
BUFFER
RBUFF
SINGLE LINE
REFRESH
BUFFER
41 WORDS
ALSO USED AS
SYNTAX BUFFER
83 WORDS

FIG.5A

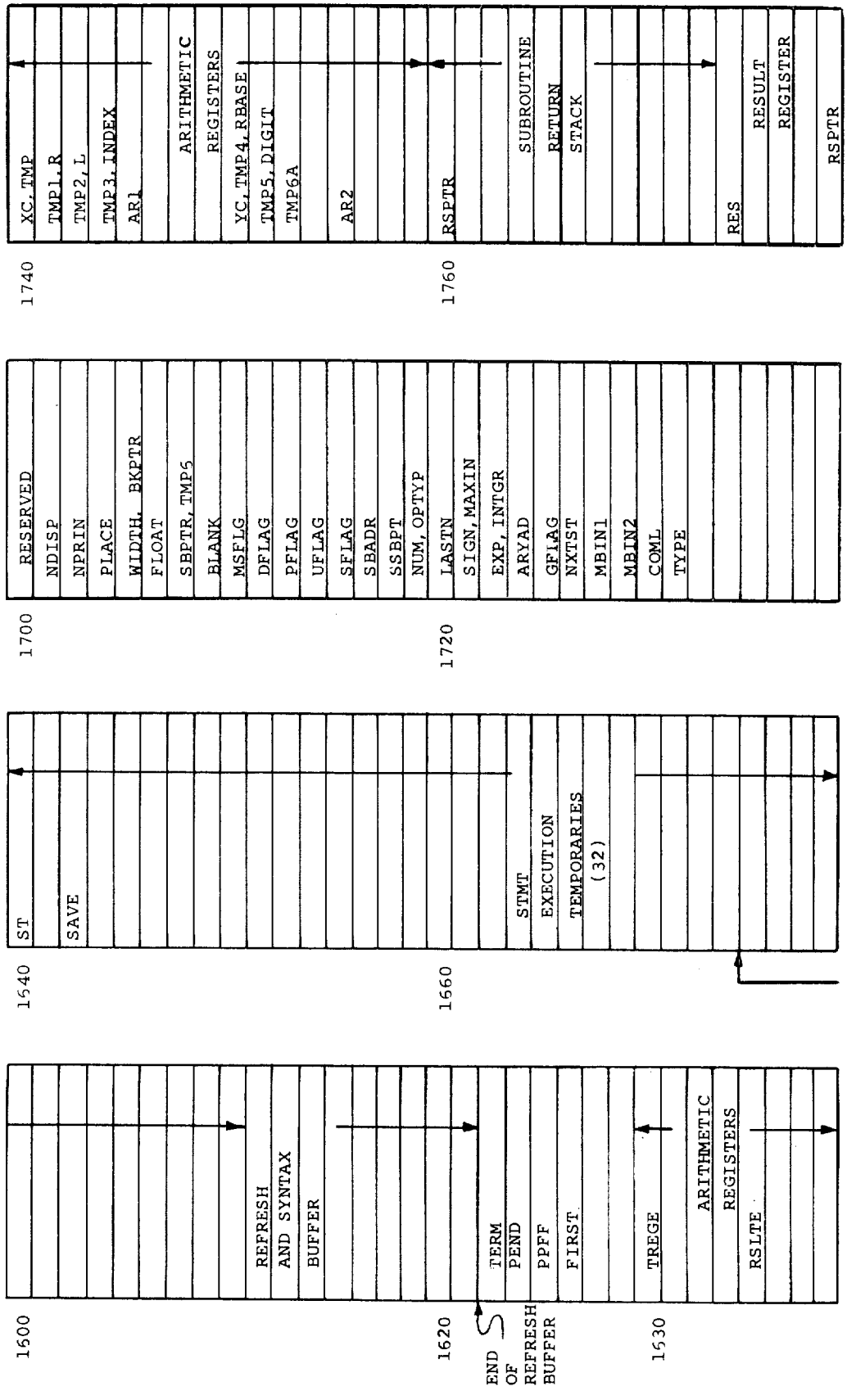

1600
REFRESH
AND SYNTAX
BUFFER
1620
TERM
PEND
PPFF
FIRST
1630
TREGE
ARITHMETIC
REGISTERS
RSLTE
END OF REFRESH BUFFER
1640
ST
SAVE
1660
STMT
EXECUTION
TEMPORARIES
(32)
END OF SYNTAX BUFFER
1700
RESERVED
NDISP
NPRIN
PLACE
WIDTH, BKPTR
FLOAT
SBPTR, TMP5
BLANK
MSFLG
DFLAG
PFLAG
UFLAG
SFLAG
SBADR
SSBPT
NUM, OPTYP
1720
LASTN
SIGN, MAXIN
EXP, INTGR
ARYAD
GFLAG
NXTST
MBIN1
MBIN2
COML
TYPE
1740
XC, TMP
TMP1, R
TMP2, L
TMP3, INDEX
AR1
ARITHMETIC
REGISTERS
YC, TMP4, RBASE
TMP5, DIGIT
TMP6A
AR2
1760
RSPTR
SUBROUTINE
RETURN
STACK
RES
RESULT
REGISTER
RSPTR

FIG 5B

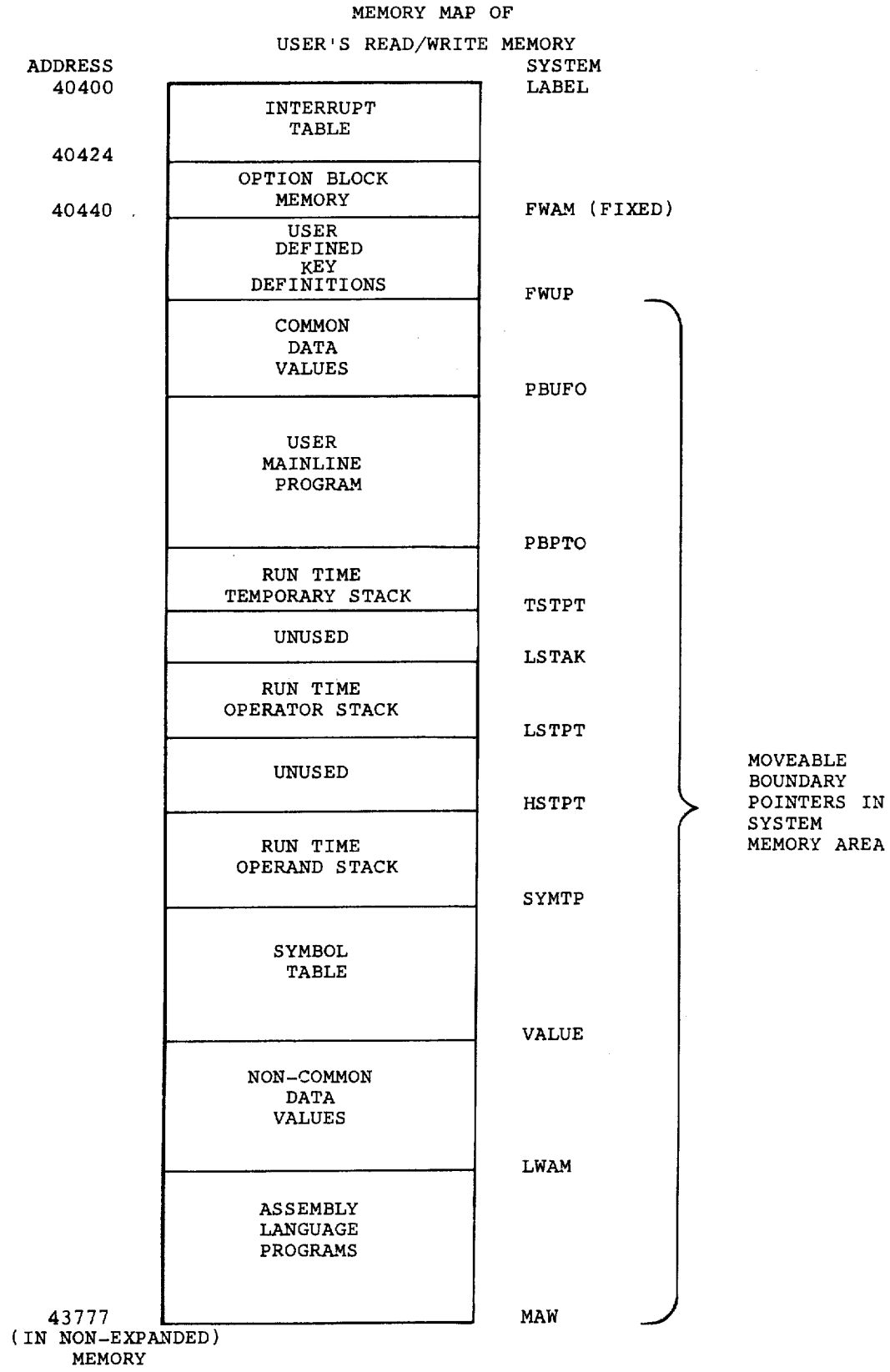

MEMORY MAP OF
USER'S READ/WRITE MEMORY
ADDRESS
40400
40424
40440
43777
(IN NON-EXPANDED)
MEMORY
SYSTEM
LABEL
INTERRUPT TABLE
OPTION BLOCK MEMORY
USER DEFINED KEY DEFINITIONS
COMMON DATA VALUES
USER MAINLINE PROGRAM
RUN TIME TEMPORARY STACK
UNUSED
RUN TIME OPERATOR STACK
UNUSED
RUN TIME OPERAND STACK
SYMBOL TABLE
NON-COMMON DATA VALUES
ASSEMBLY LANGUAGE PROGRAMS
FWAM (FIXED)
FWUP
PBUFO
PBPTO
TSTPT
LSTAK
LSTPT
HSTPT
SYMTP
VALUE
LWAM
MAW
MOVEABLE BOUNDARY POINTERS IN SYSTEM MEMORY AREA

FIG. 6

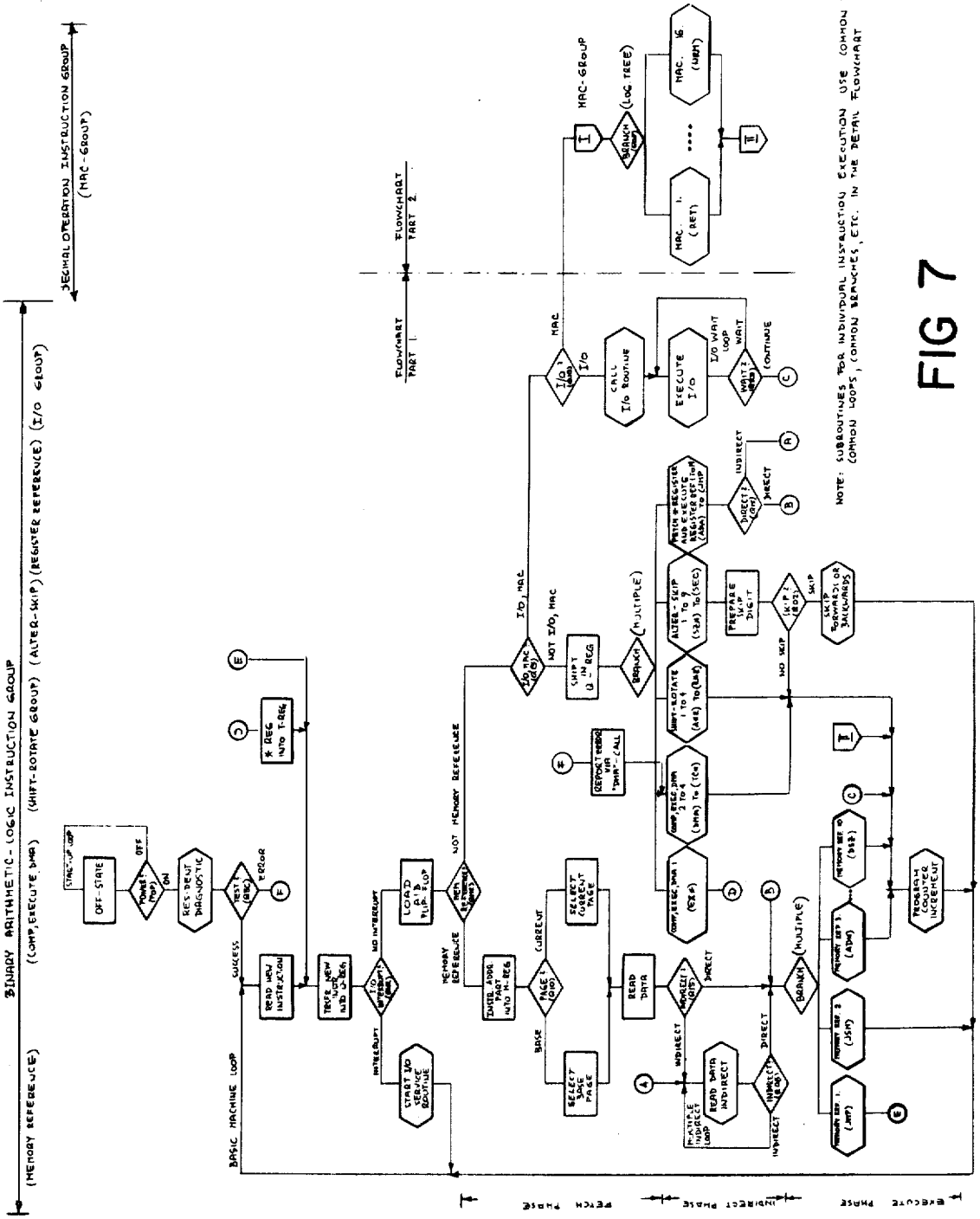

BINARY ARITHMETIC - LOGIC INSTRUCTION GROUP
(MEMORY REFERENCE) (COMP, EXECUTE, BMR) (SHIFT-ROTATE GROUP) (ALTER-SKIP) (REGISTER REFERENCE) (I/O GROUP)
DECIMAL OPERATION INSTRUCTION GROUP
(MRC-GROUP)
FLOWCHART PART 1
FLOWCHART PART 2
OFF-STATE
POWER (MAN)
RES-DEBT DIAGNOSTIC
TEST (MRC)
ERROR
SUCCESS
READ NEW INSTRUCTION
TRANS. NEW INSTR. INTO U-REG
TO INTERRUPT
INTERRUPT
NO INTERRUPT
START I/O SERVICE ROUTINE
LOAD A & B FLIP-FLOP
MEMORY REFERENCE
NOT MEMORY REFERENCE
DATA ADDR. INTO M-REG
PAGE 1 (DIR)
CURRENT
BASE
SELECT BASE PAGE
SELECT CURRENT PAGE
READ DATA
INDIRECT (DIR)
INDIRECT
DIRECT
MULTIPLE INDIRECT LOOP
READ DATA INDIRECT
INDIRECT (DIR)
BRANCH (MULTIPLE)
COMP, EXEC, BMR 1 TO 4 (BMR) TO (10C)
SHIFT-ROTATE 1 TO 9 (SRE) TO (SHE)
ALTER - SKIP 1 TO 9 (SRA) TO (SHS)
PERFORM REGISTER SUBEXECUTE (REG) TO (CMP)
DIRECT (MRC)
INDIRECT
PREPARE NEXT DIGIT
SKIP (SKP)
SKIP INSTRUCTION OR SKIP/ARGS
NO SKIP
MEMORY REF 0 (JMP)
MEMORY REF 1 (JSP)
MEMORY REF 2 (ADM)
MEMORY REF 3 (XCH)
PROGRAM COUNTER INCREMENT
NOT I/O, MRC
I/O, MRC
I/O (MRC)
CALL I/O ROUTINE
EXECUTE I/O
I/O WAIT LOOP
WAIT (MRC)
CONTINUE
MRC-GROUP
BRANCH (LOG TREE)
MRC. 1. (REST)
MRC. 16 (MRM)
NOTE: SUBROUTINES FOR INDIVIDUAL INSTRUCTION EXECUTION USE COMMON PATHS, COMMON LOOPS, COMMON BRANCHES, ETC. IN THE DETAIL FLOWCHART
FETCH PHASE
INDIRECT PHASE
EXECUTE PHASE

FIG 7

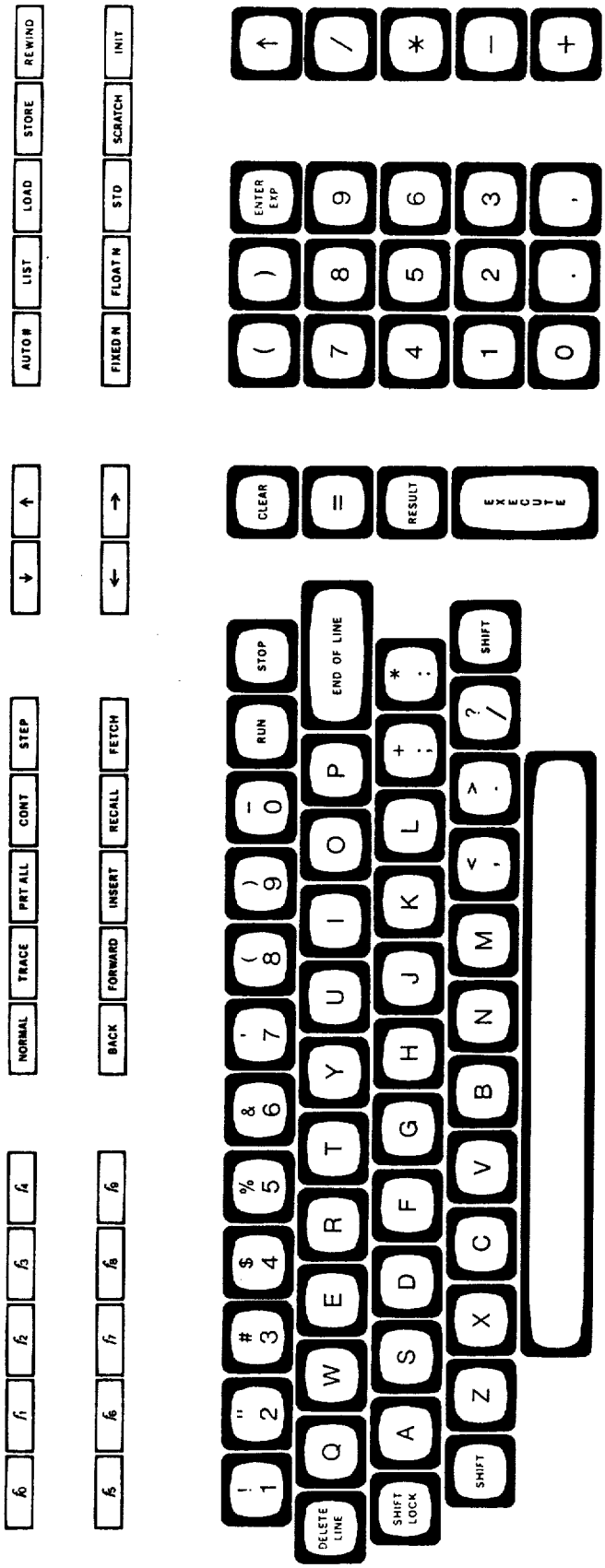

f6 f1 f2 f3 f4
f6 f6 f7 f8 f9
NORMAL TRACE PRT ALL CONT STEP
BACK FORWARD INSERT RECALL FETCH
AUTO # LIST LOAD STORE REWIND
FIXED N FLOAT N STD SCRATCH INIT
DELETE LINE
SHIFT LOCK
SHIFT
STOP
END OF LINE
CLEAR
RESULT
EXECUTE
ENTER EXP

FIG 8

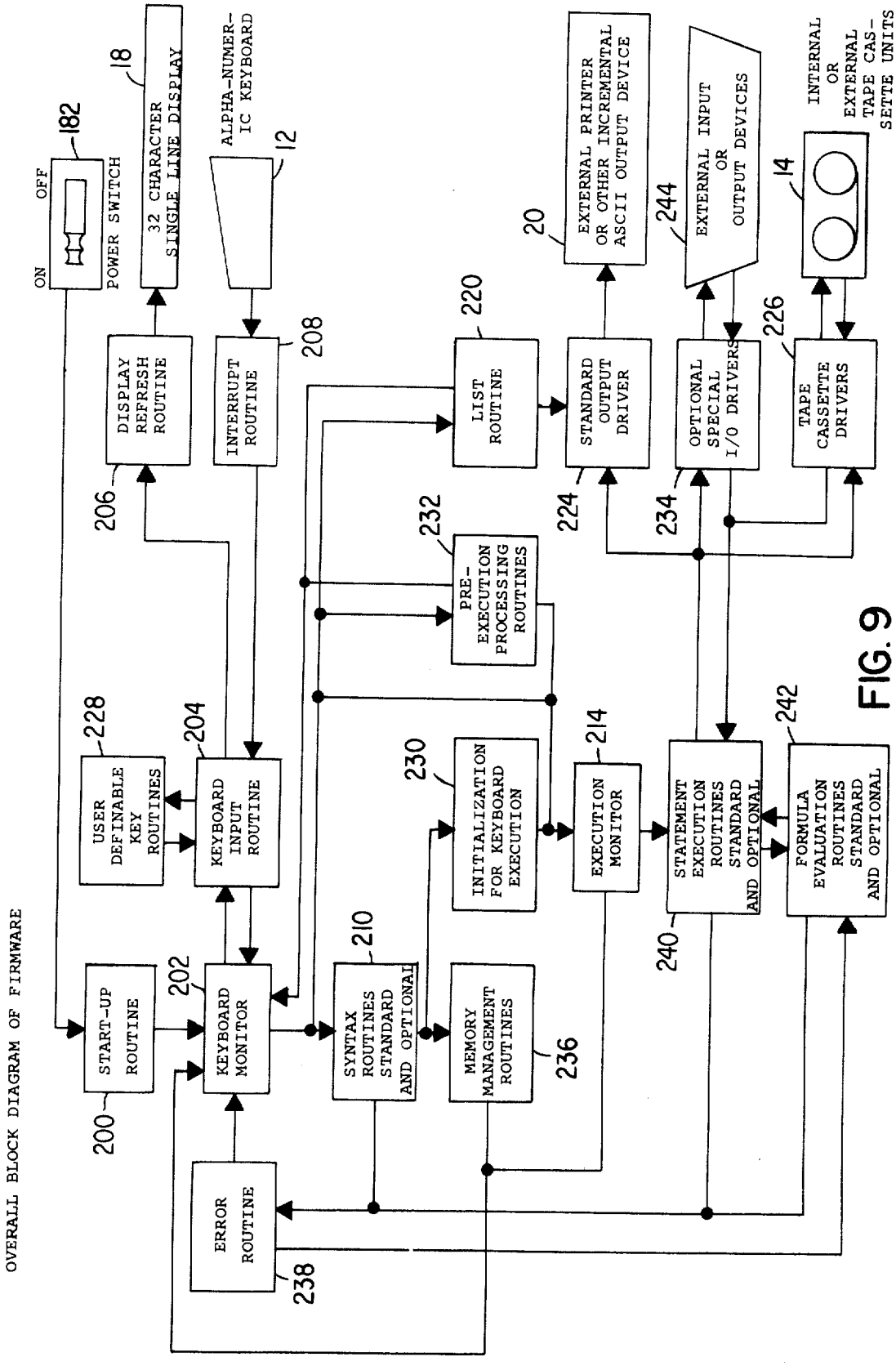

OVERALL BLOCK DIAGRAM OF FIRMWARE
ON
OFF
POWER SWITCH
182
32 CHARACTER SINGLE LINE DISPLAY
18
ALPHA-NUMERIC KEYBOARD
12
EXTERNAL PRINTER OR OTHER INCREMENTAL ASCII OUTPUT DEVICE
20
EXTERNAL INPUT OR OUTPUT DEVICES
244
INTERNAL OR EXTERNAL TAPE CASSETTE UNITS
14
DISPLAY REFRESH ROUTINE
206
INTERRUPT ROUTINE
208
LIST ROUTINE
220
STANDARD OUTPUT DRIVER
224
OPTIONAL SPECIAL I/O DRIVERS
234
TAPE CASSETTE DRIVERS
226
PRE-EXECUTION PROCESSING ROUTINES
232
USER DEFINABLE KEY ROUTINES
228
KEYBOARD INPUT ROUTINE
204
INITIALIZATION FOR KEYBOARD EXECUTION
230
EXECUTION MONITOR
214
STATEMENT EXECUTION ROUTINES STANDARD AND OPTIONAL
240
FORMULA EVALUATION ROUTINES STANDARD AND OPTIONAL
242
START-UP ROUTINE
200
KEYBOARD MONITOR
202
SYNTAX ROUTINES STANDARD AND OPTIONAL
210
MEMORY MANAGEMENT ROUTINES
236
ERROR ROUTINE
238

FIG. 9

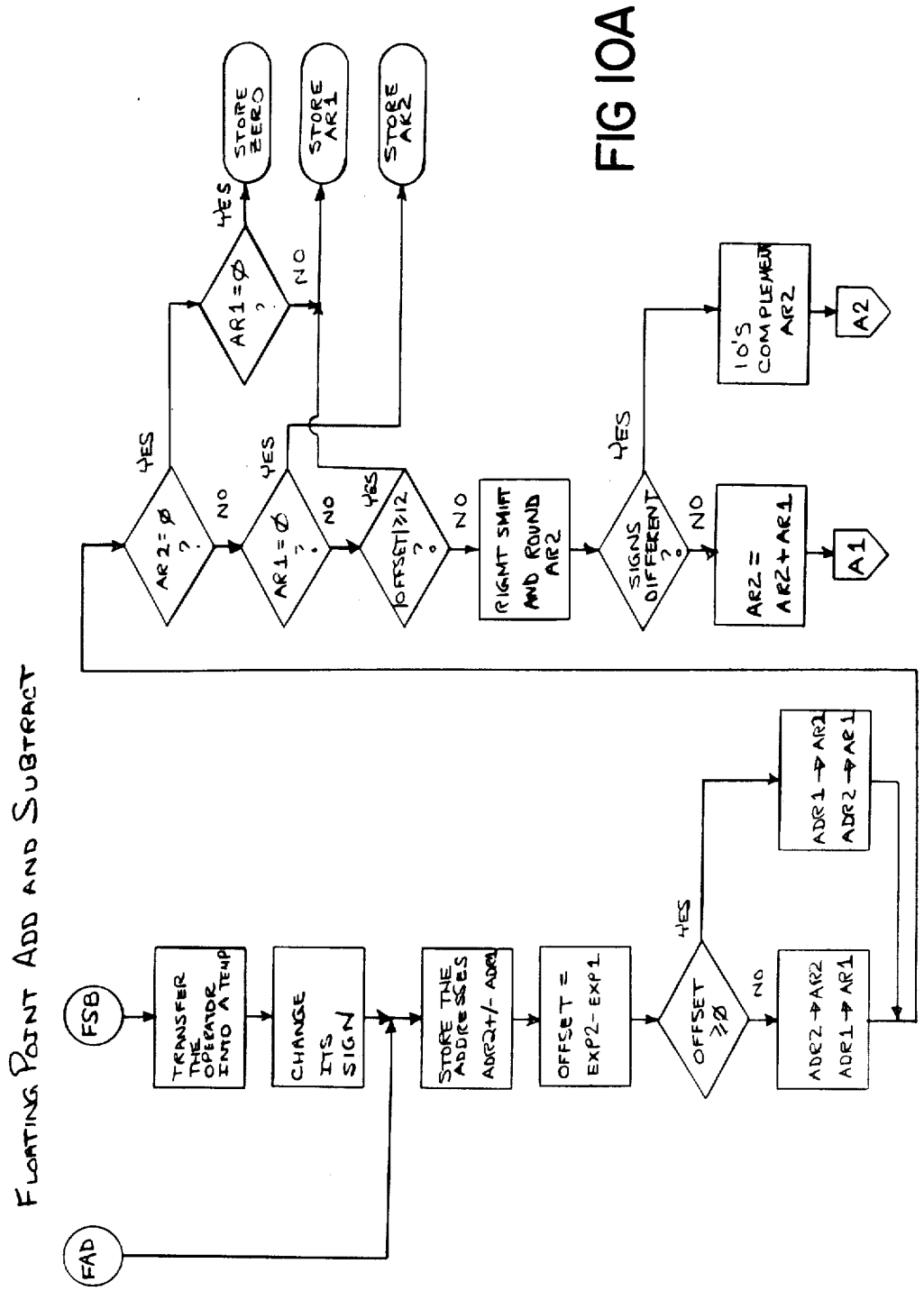

FLOATING POINT ADD AND SUBTRACT
FSB
FAD
TRANSFER THE OPERATOR INTO A TEMP
CHANGE ITS SIGN
STORE THE ADDRESSES ADDR2+/- ADDR1
OFFSET = EXP2 - EXP1
OFFSET ≥ Ø
YES
NO
ADDR1 → AR2
ADDR2 → AR1
ADDR2 → AR2
ADDR1 → AR1
AR2 = Ø ?
AR1 = Ø ?
|OFFSET| ≥ 12 ?
RIGHT SHIFT AND ROUND AR2
SIGNS DIFFERENT ?
10'S COMPLEMENT AR2
AR2 = AR2 + AR1
A1
A2
STORE ZERO
STORE AR1
STORE AR2

FIG 10A

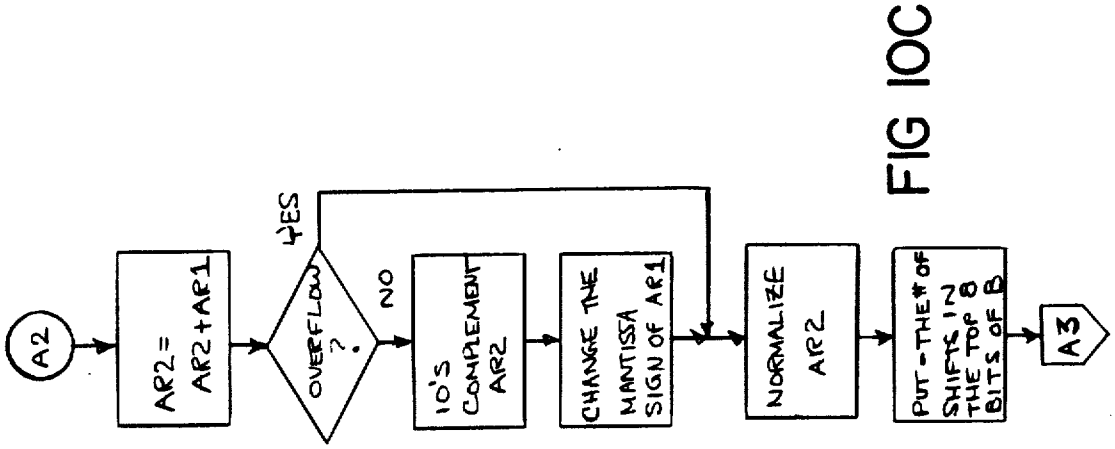


FIG 10C

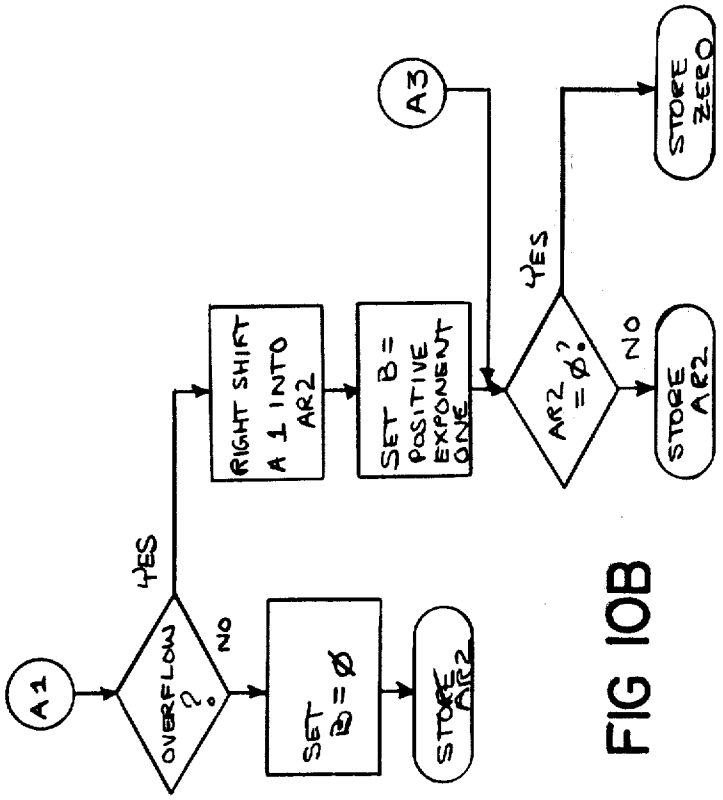


FIG 10B

FLOATING POINT MULTIPLY

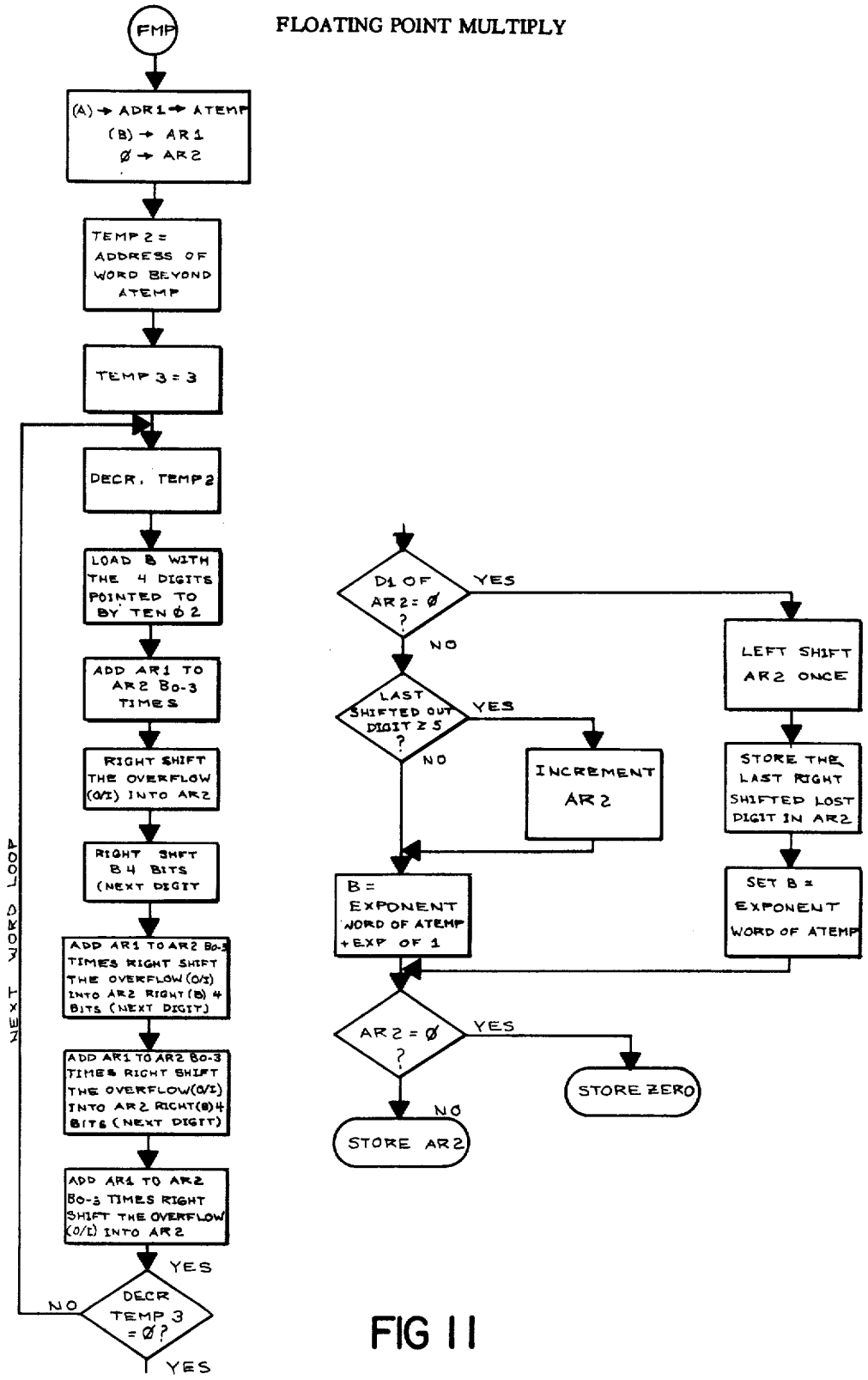
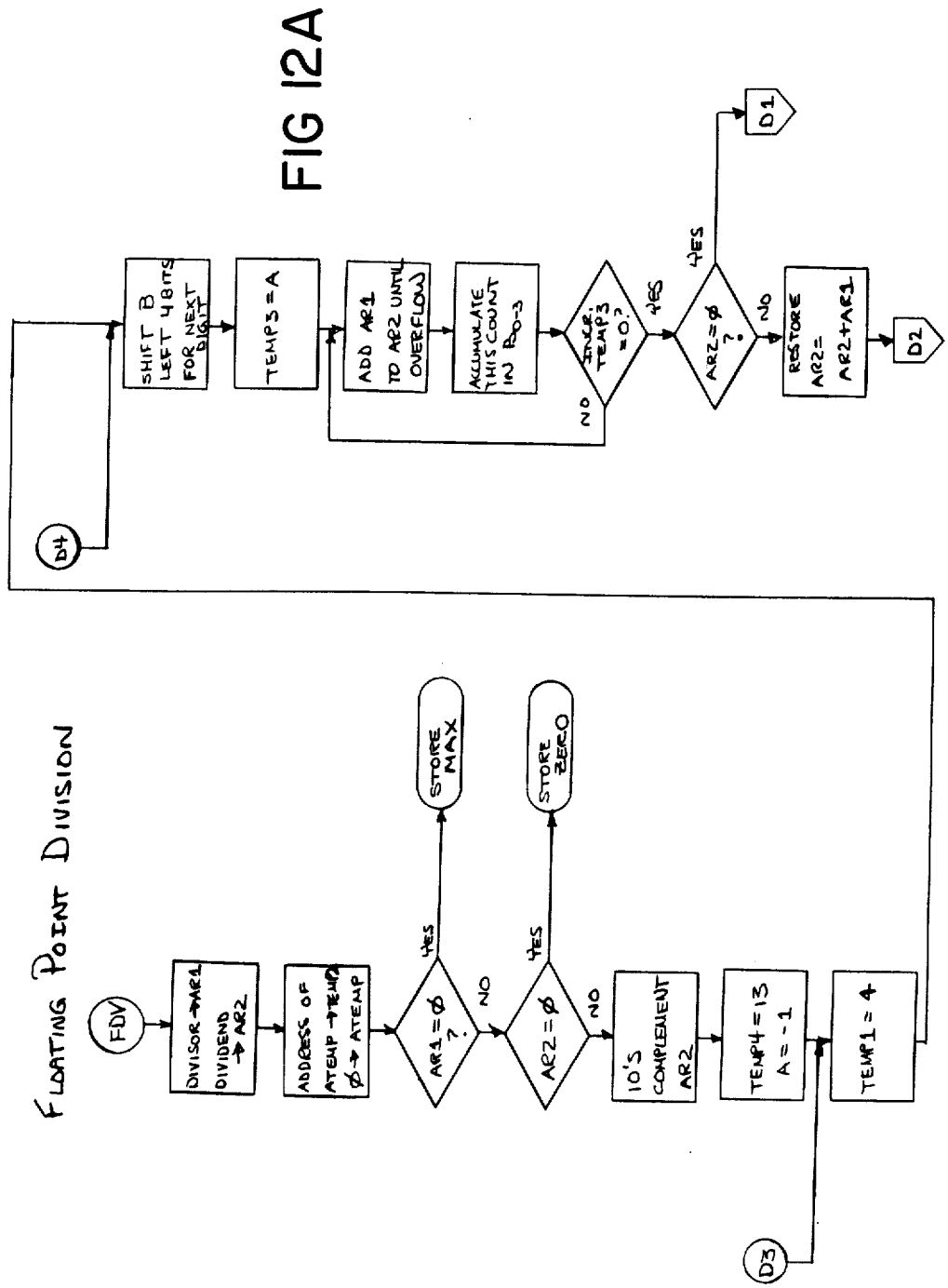


FIG 11

FIG 12A


FLOATING POINT DIVISION
FDV
DIVISOR → AR1
DIVIDEND → AR2
ADDRESS OF ATEMP → TEMP
Ø → ATEMP
AR1 = Ø ?
YES
NO
STORE MAX
AR2 = Ø
YES
NO
STORE ZERO
10'S COMPLEMENT AR2
TEMP4 = 13
A = -1
TEMP1 = 4
D3
D4
SHIFT B LEFT 4 BITS FOR NEXT DIGIT
TEMP3 = A
ADD AR1 TO AR2 UNTIL OVERFLOW
ACCUMULATE THIS COUNT IN B0-3
SUCC. TEMP3 = 0 ?
YES
NO
AR2 = Ø ?
YES
NO
D1
RESTORE AR2 = AR2 + AR1
D2

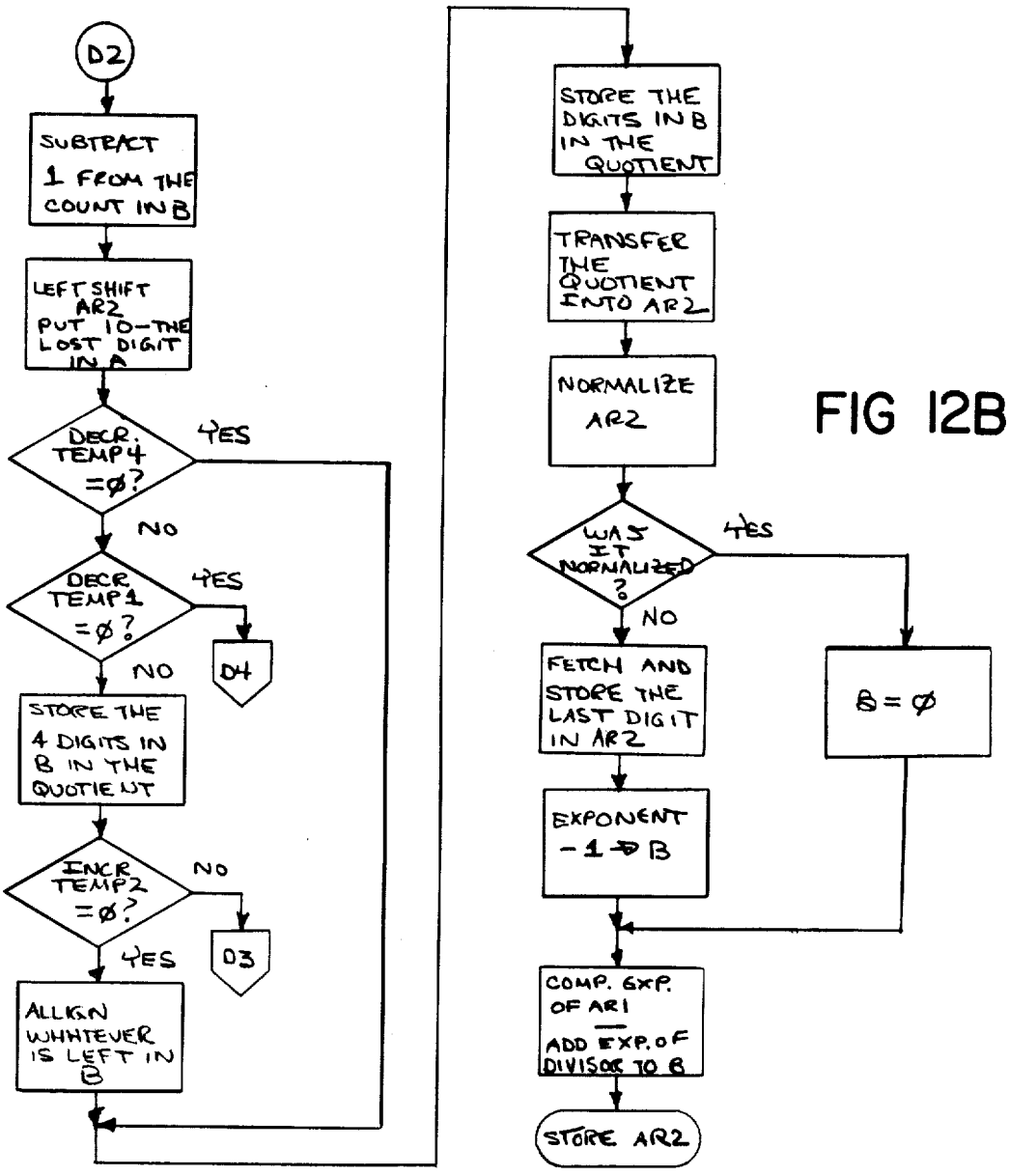

D2
SUBTRACT 1 FROM THE COUNT IN B
LEFT SHIFT AR2 PUT IO-THE LOST DIGIT IN A
DECR. TEMP4 =Ø?
YES
NO
DECR TEMP1 =Ø?
YES
D4
NO
STORE THE 4 DIGITS IN B IN THE QUOTIENT
INCR TEMP2 =Ø?
NO
D3
YES
ALLIGN WHHTEVER IS LEFT IN B
STORE THE DIGITS IN B IN THE QUOTIENT
TRANSFER THE QUOTIENT INTO AR2
NORMALIZE AR2
WAS IT NORMALIZED ?
YES
NO
FETCH AND STORE THE LAST DIGIT IN AR2
B = Ø
EXPONENT -1 → B
COMP. EXP. OF AR1
ADD EXP. OF DIVISOR TO B
STORE AR2

FIG 12B

FLOATING POINT SQUARE ROOT

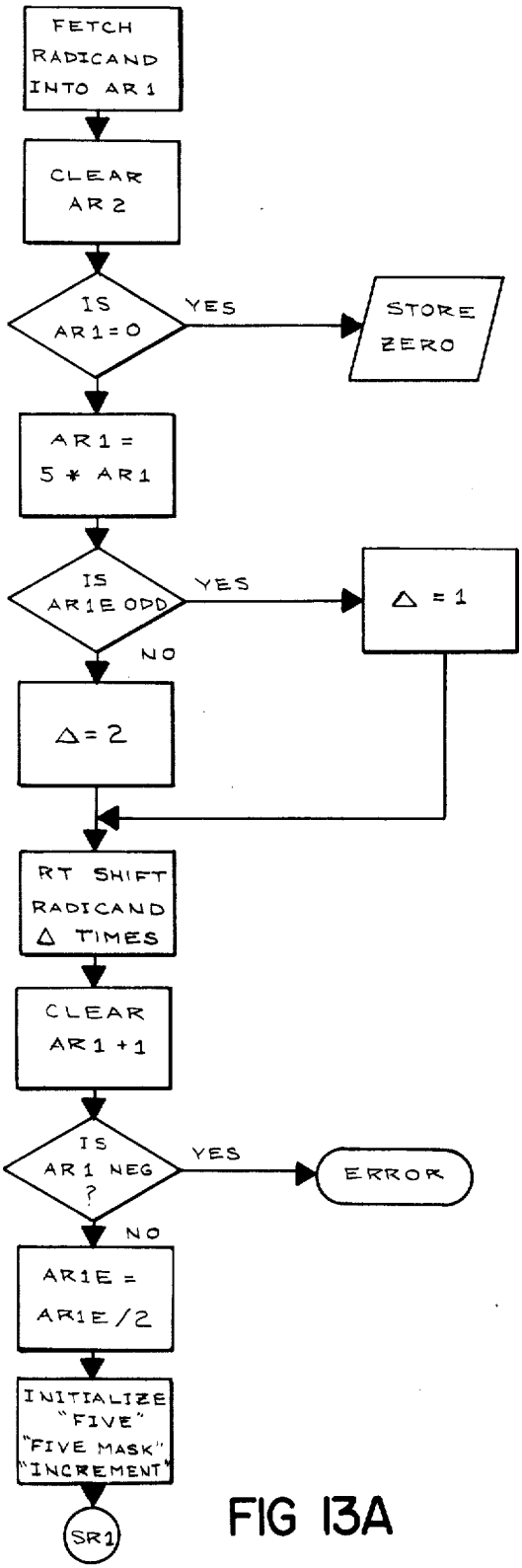


FIG 13A

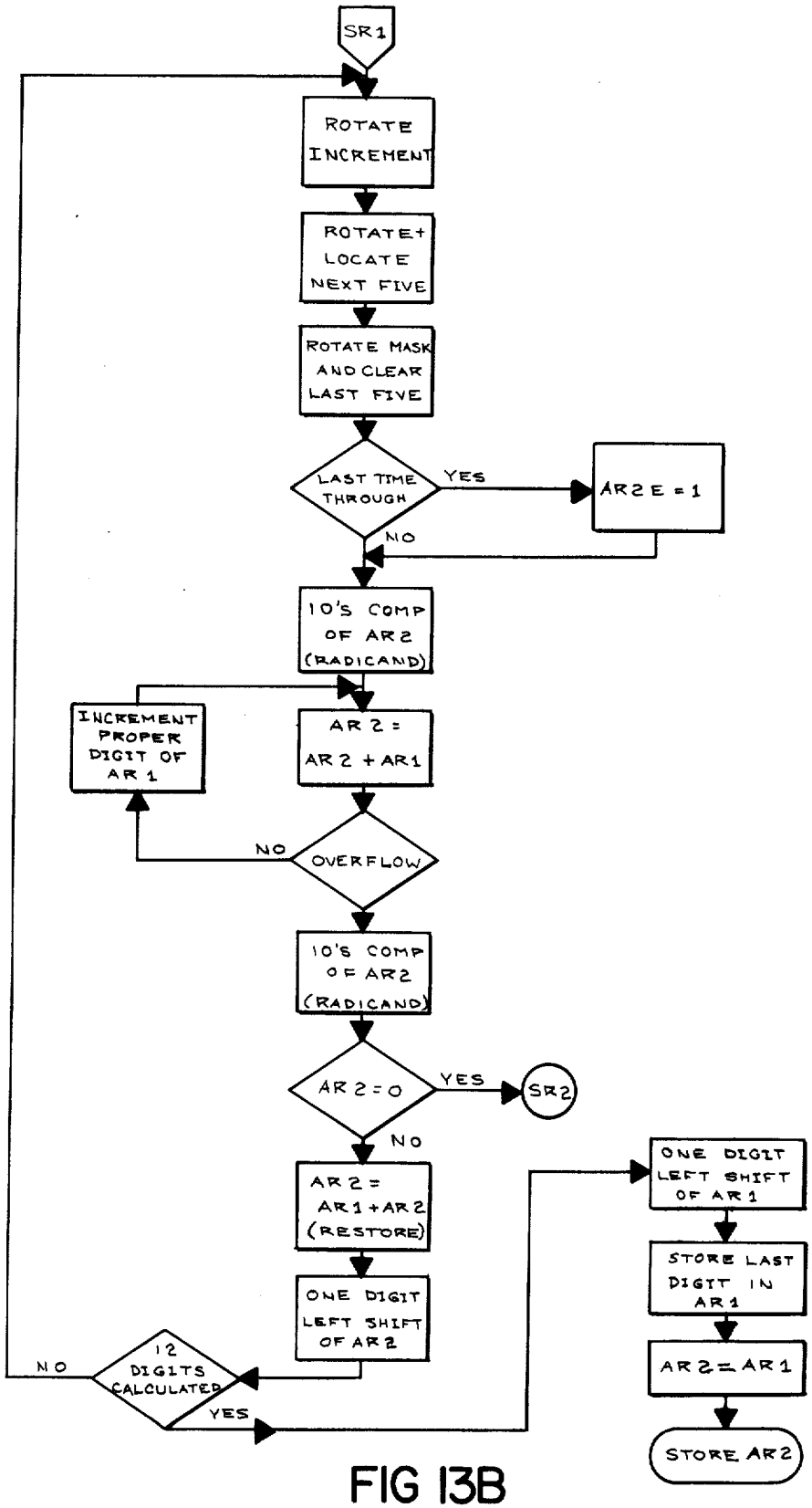

SR1
ROTATE INCREMENT
ROTATE+ LOCATE NEXT FIVE
ROTATE MASK AND CLEAR LAST FIVE
LAST TIME THROUGH
YES
AR2 E = 1
NO
10'S COMP OF AR2 (RADICAND)
INCREMENT PROPER DIGIT OF AR 1
AR 2 = AR 2 + AR1
NO
OVERFLOW
10'S COMP OF AR2 (RADICAND)
AR 2 = 0
YES
SR2
NO
AR 2 = AR1 + AR2 (RESTORE)
ONE DIGIT LEFT SHIFT OF AR2
12 DIGITS CALCULATED
NO
YES
ONE DIGIT LEFT SHIFT OF AR 1
STORE LAST DIGIT IN AR1
AR 2 = AR1
STORE AR2

FIG 13B

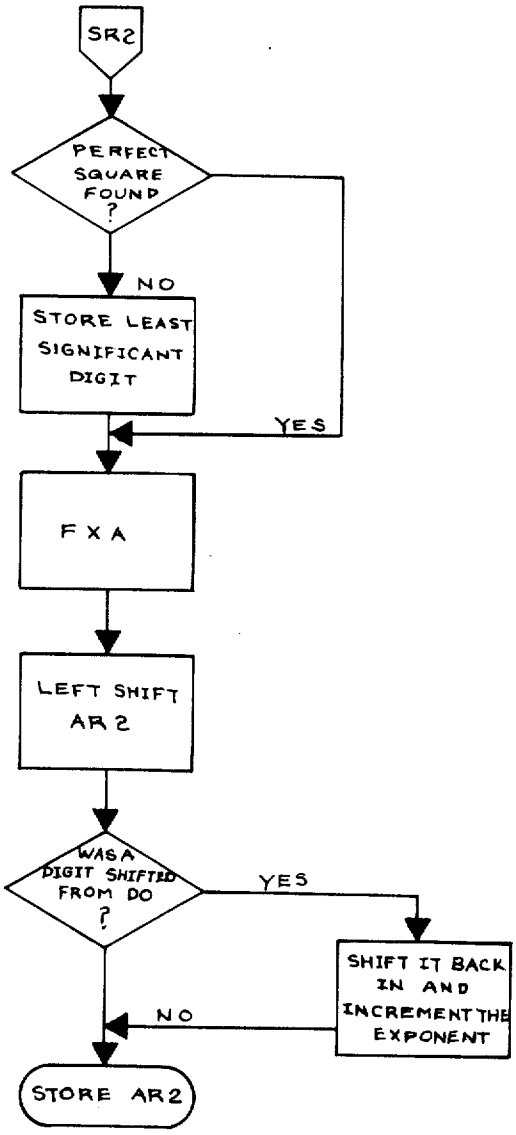

SR2
PERFECT SQUARE FOUND ?
NO
STORE LEAST SIGNIFICANT DIGIT
YES
F X A
LEFT SHIFT AR 2
WAS A DIGIT SHIFTED FROM DO ?
YES
SHIFT IT BACK IN AND INCREMENT THE EXPONENT
NO
STORE AR2

FIG. 13C

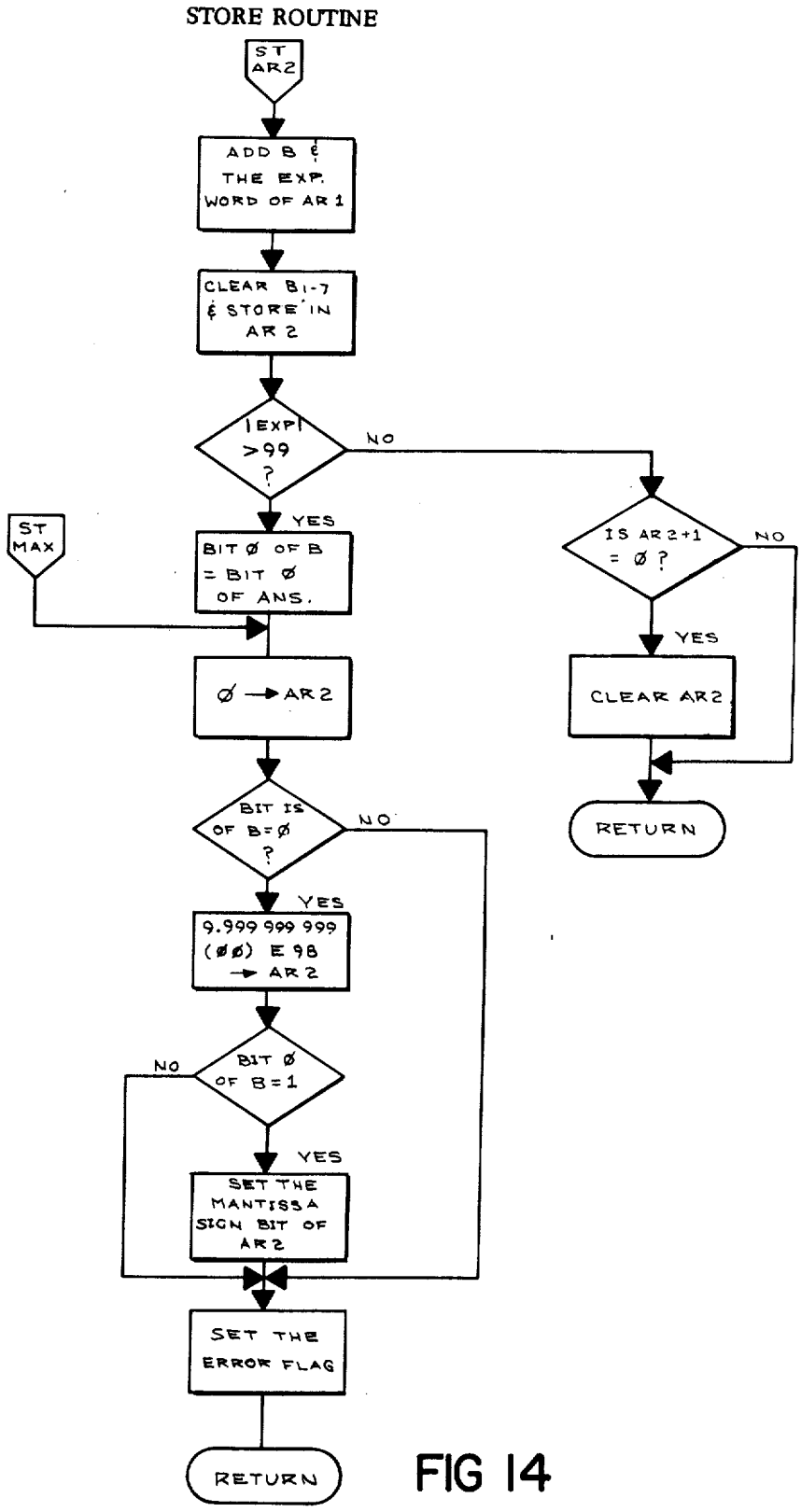

STORE ROUTINE
ST AR2
ADD B & THE EXP. WORD OF AR 1
CLEAR B1-7 & STORE IN AR 2
|EXP| >99 ?
NO
YES
ST MAX
BIT Ø OF B = BIT Ø OF ANS.
Ø → AR2
BIT IS OF B=Ø ?
NO
YES
9.999 999 999 (ØØ) E 98 → AR 2
BIT Ø OF B=1
NO
YES
SET THE MANTISSA SIGN BIT OF AR2
SET THE ERROR FLAG
RETURN
IS AR 2+1 = Ø ?
NO
YES
CLEAR AR2
RETURN

FIG 14

FIG 15

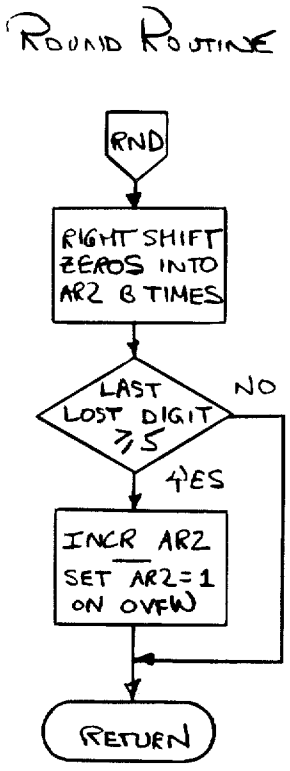
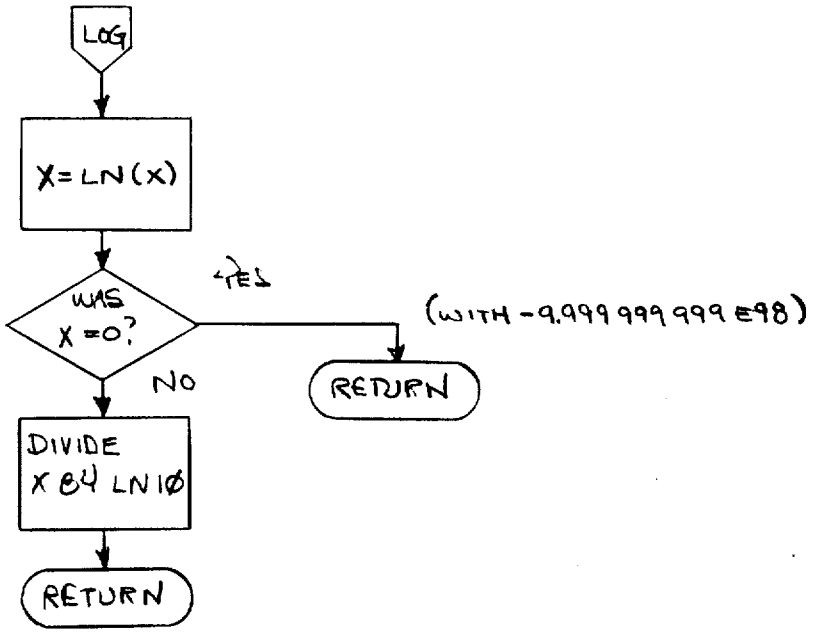

ROUND ROUTINE
RND
RIGHT SHIFT ZEROS INTO AR2 B TIMES
LAST LOST DIGIT ≥ 5
NO
YES
INCR AR2
SET AR2=1 ON OVFW
RETURN

FIG 26


LOG
X=LN(X)
WAS X=0?
YES
(WITH -9.999 999 999 E98)
NO
RETURN
DIVIDE X BY LN 10
RETURN

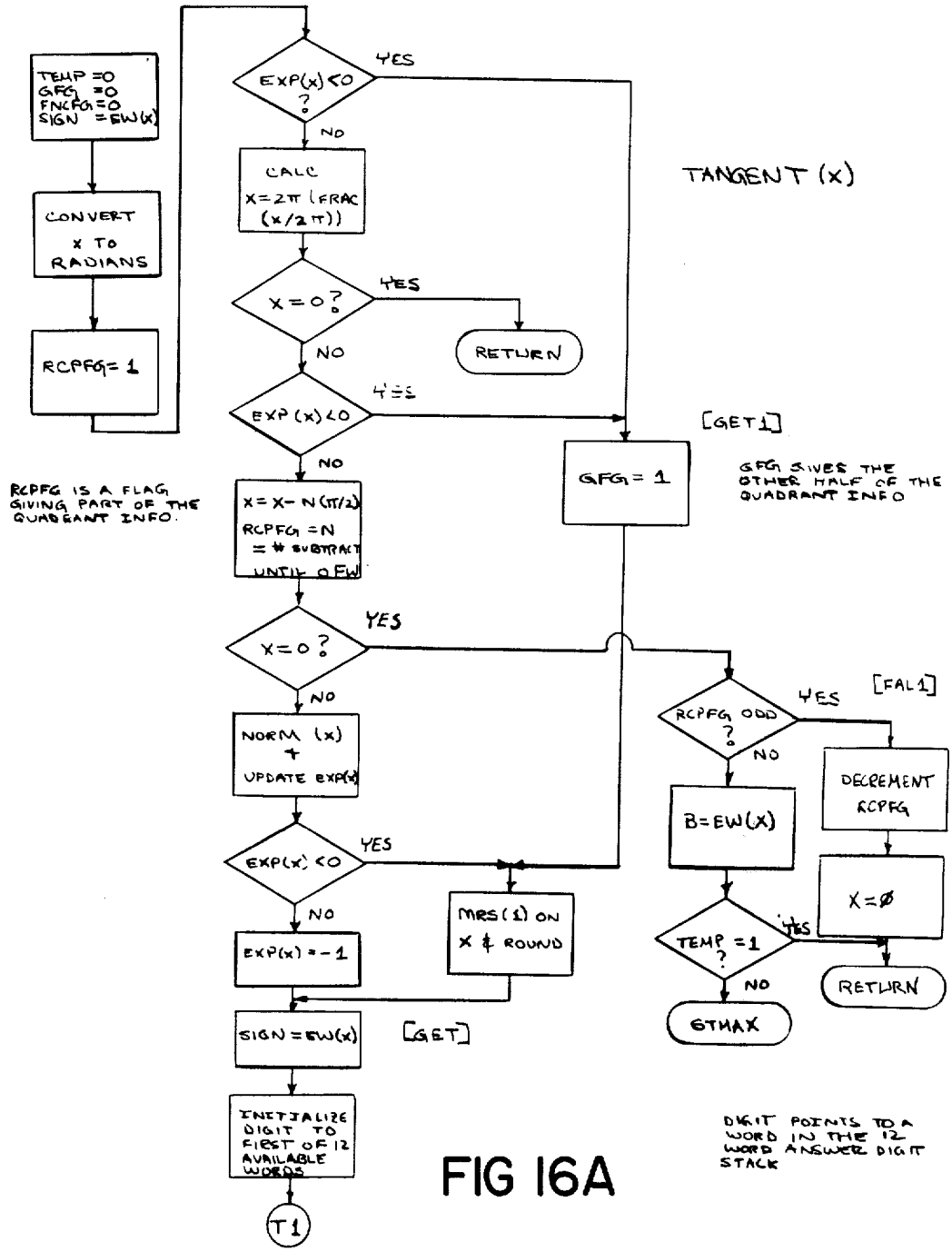

TANGENT (x)
TEMP =0
GFG =0
FNCFG=0
SIGN =EW(x)
CONVERT x TO RADIANS
RCPFG= 1
RCPFG IS A FLAG GIVING PART OF THE QUADRANT INFO.
EXP(x)<0 ?
YES
NO
CALC X=2π (FRAC (x/2π))
X=0 ?
YES
RETURN
NO
EXP(x)<0
YES
NO
X=X-N(π/2)
RCPFG =N = # SUBTRACT UNTIL 0 FW
X=0 ?
YES
NO
NORM (x) + UPDATE EXP(x)
EXP(x) <0
YES
NO
EXP(x) = -1
MRS(1) ON X & ROUND
SIGN =EW(x)
[GET]
INITIALIZE DIGIT TO FIRST OF 12 AVAILABLE WORDS
T1
[GET1]
GFG= 1
GFG GIVES THE OTHER HALF OF THE QUADRANT INFO
RCPFG ODD ?
YES
[FAL1]
NO
B=EW(X)
DECREMENT RCPFG
X=Ø
TEMP =1 ?
YES
NO
RETURN
6THAX
DIGIT POINTS TO A WORD IN THE 12 WORD ANSWER DIGIT STACK

FIG 16A

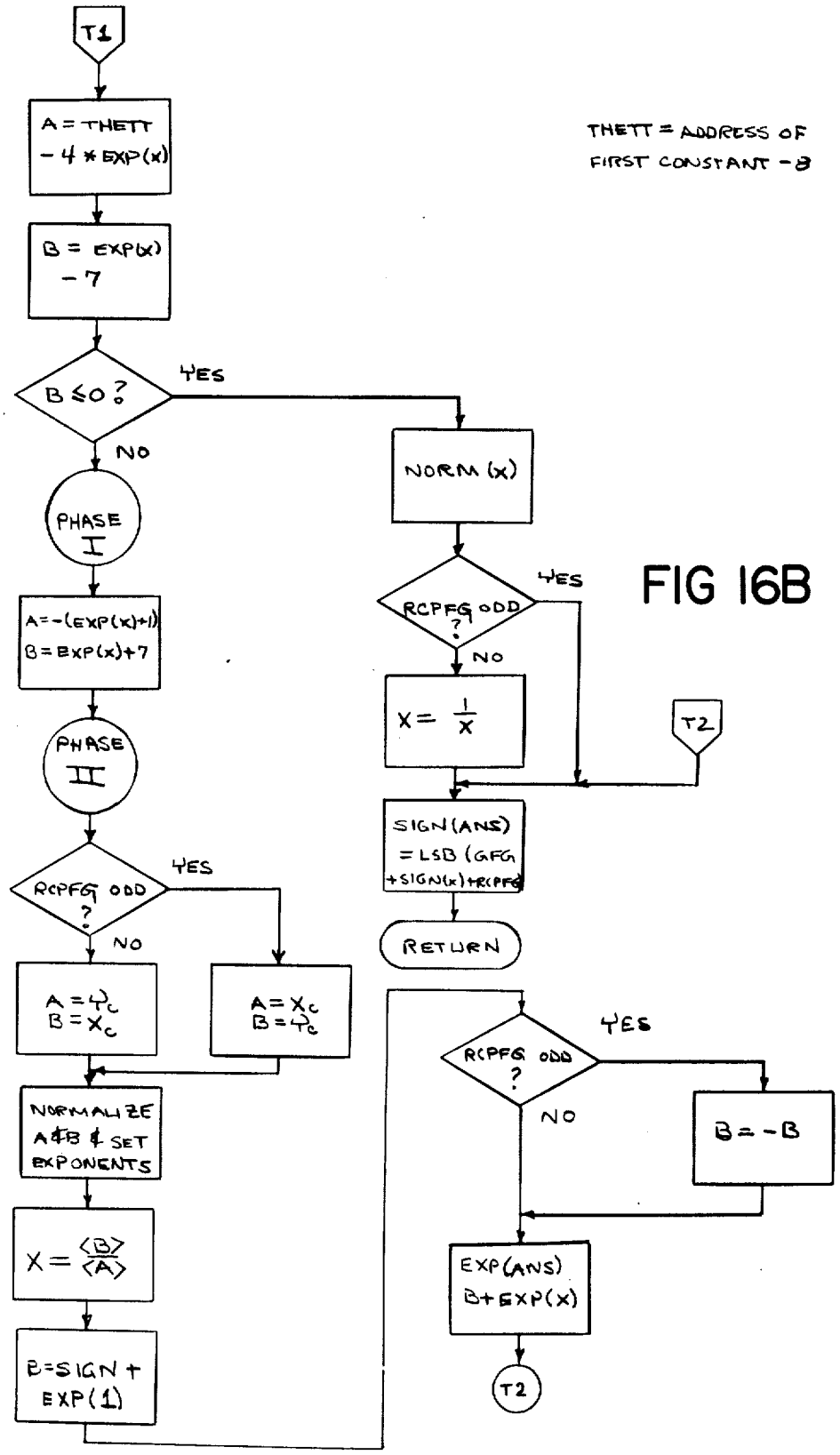
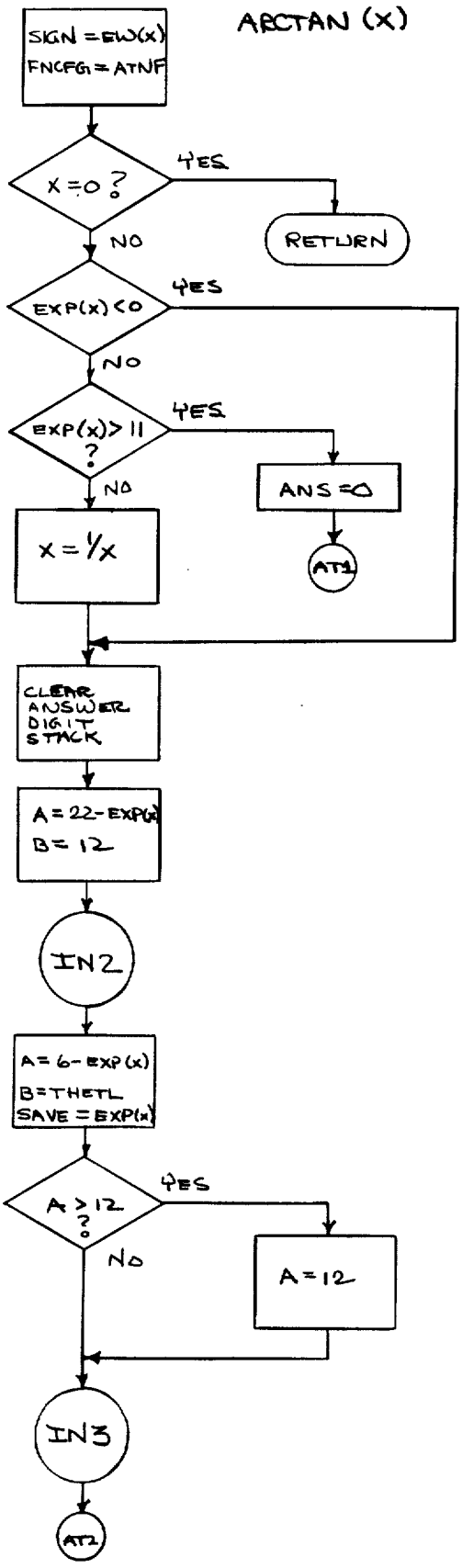

T1
A = THETT - 4 * EXP(x)
B = EXP(x) - 7
B ≤ 0 ?
YES
NO
PHASE I
A = -(EXP(x)+1)
B = EXP(x)+7
PHASE II
RCPFG ODD ?
YES
NO
A = Yc
B = Xc
A = Xc
B = Yc
NORMALIZE A & B & SET EXPONENTS
X = <B> / <A>
B = SIGN + EXP(1)
NORM (x)
RCPFG ODD ?
YES
NO
X = 1/X
T2
SIGN(ANS) = LSB (GFG +SIGN(x)+RCPFG
RETURN
RCPFG ODD ?
YES
NO
B = -B
EXP(ANS) B+EXP(x)
T2
THETT = ADDRESS OF FIRST CONSTANT - 8

FIG 16B



ATNF IS THE ARCTAN FLAG

FIG 17A

THETL = ADDRESS OF LAST CORDIC CONSTANT -4

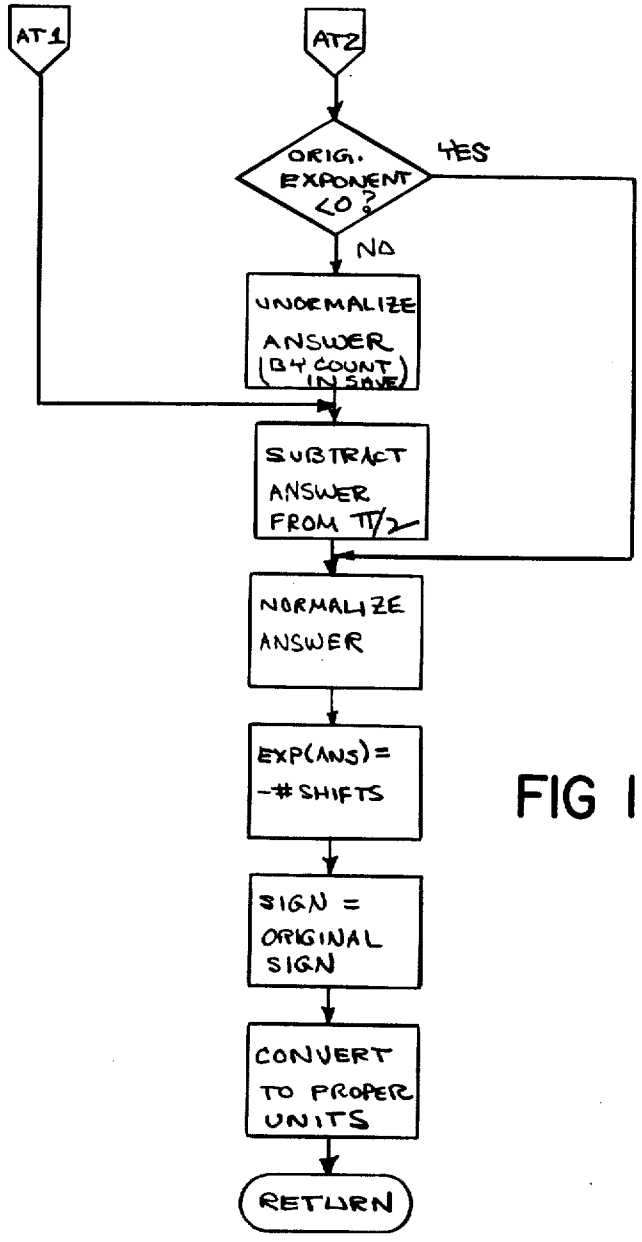

AT1
AT2
ORIG. EXPONENT <0?
YES
NO
UNNORMALIZE ANSWER (BY COUNT IN SAVE)
SUBTRACT ANSWER FROM TT/2
NORMALIZE ANSWER
EXP(ANS) = -# SHIFTS
SIGN = ORIGINAL SIGN
CONVERT TO PROPER UNITS
RETURN

FIG 17B

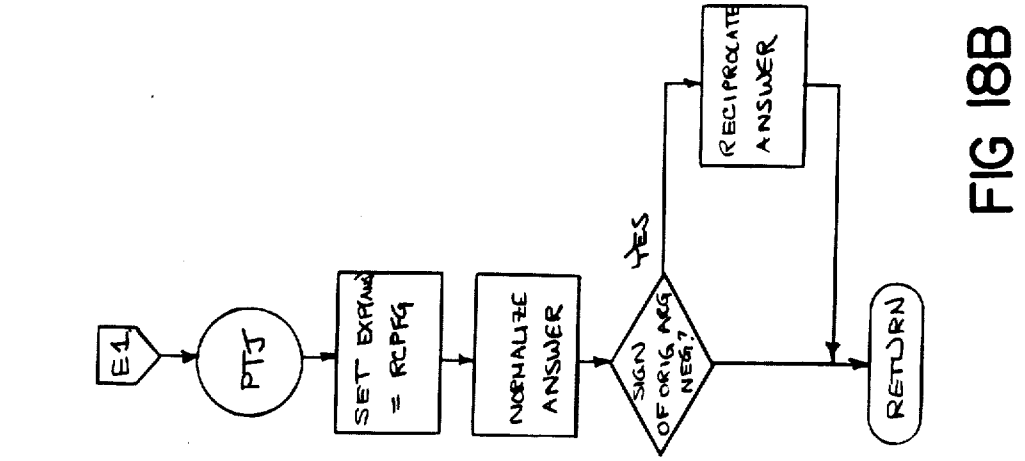


FIG 18B

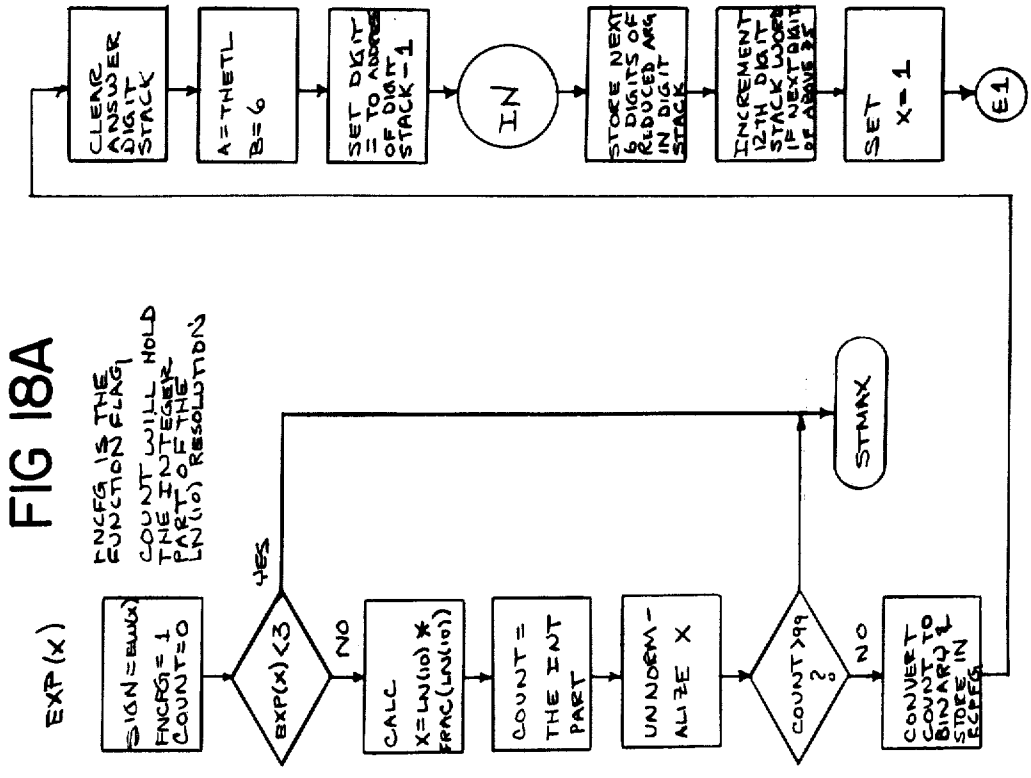


FIG 18A

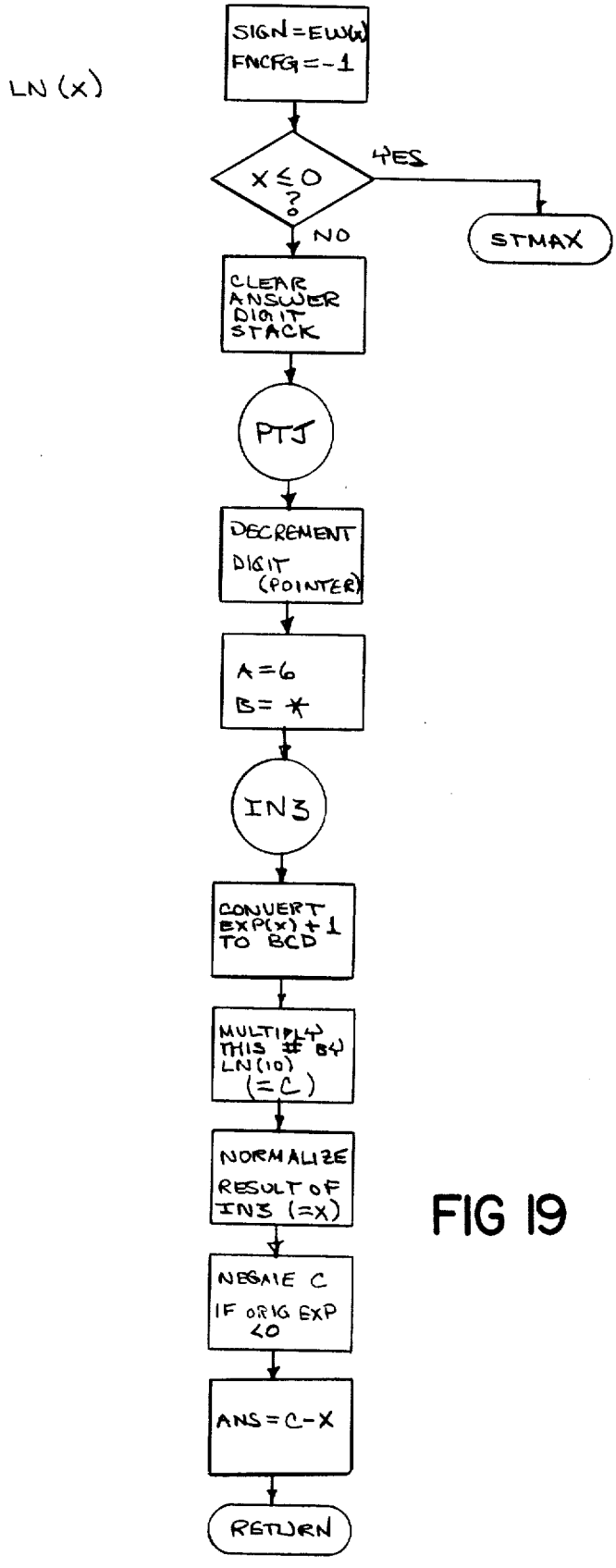

LN (X)
SIGN = EXP(X)
FNCFG = -1
X ≤ 0 ?
YES
NO
STMAX
CLEAR ANSWER DIGIT STACK
PTJ
DECREMENT DIGIT (POINTER)
A = 6
B = *
IN3
CONVERT EXP(X) + 1 TO BCD
MULTIPLY THIS # BY LN(10) (=C)
NORMALIZE RESULT OF IN3 (=X)
NEGATE C IF ORIG EXP < 0
ANS = C - X
RETURN
FNCFG = FUNCTION FLAG
DIGIT IS A POINTER FOR THE ANSWER DIGIT STACK
* IS ADDRESS OF LAST CORDIC CONSTANT - 4

FIG 19

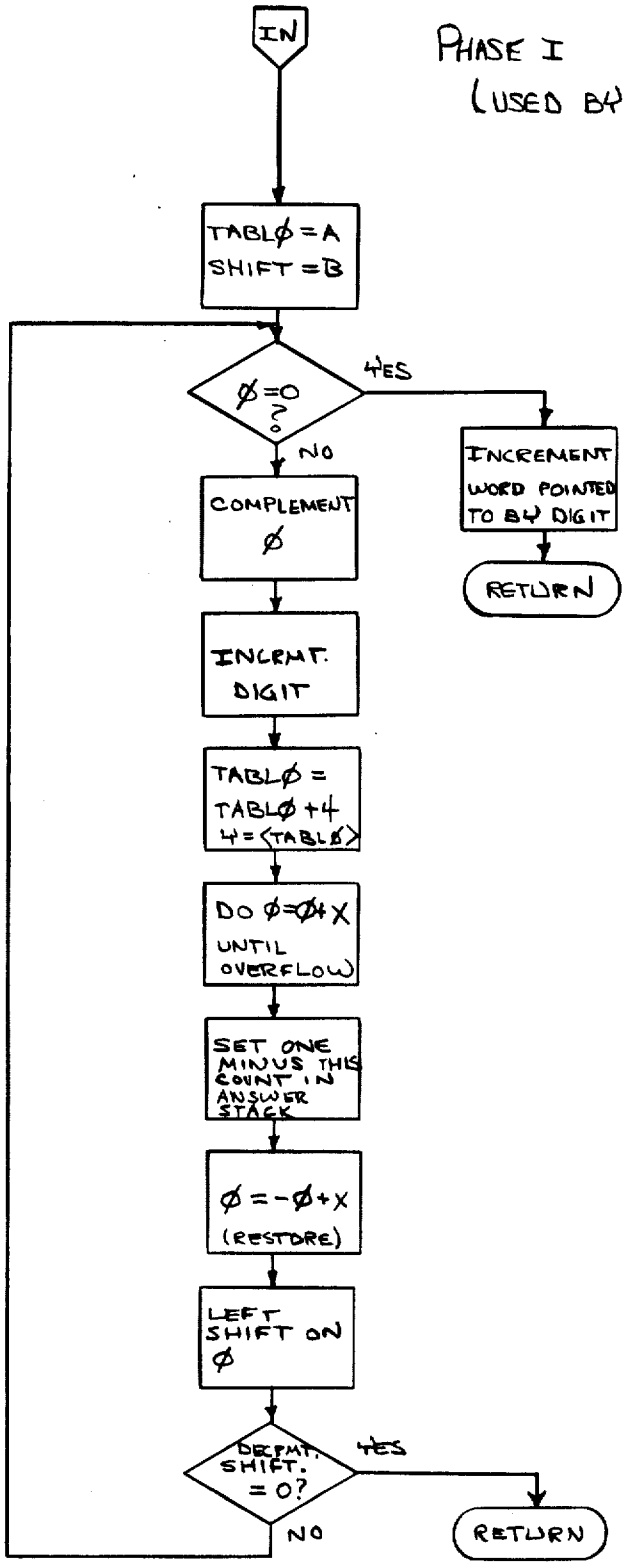
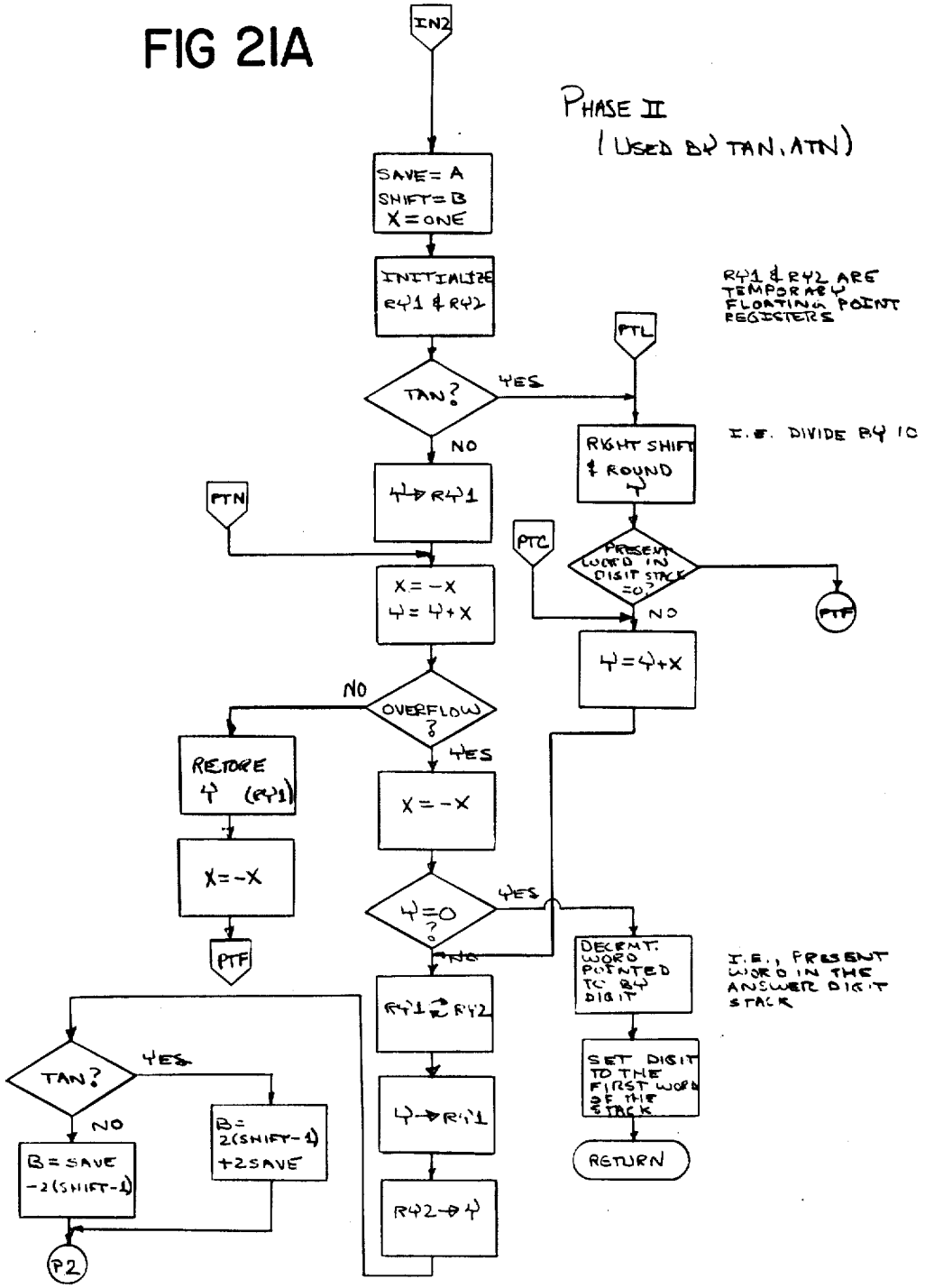

IN
PHASE I
(USED BY TAN, EXP)
TABLØ = A
SHIFT = B
TABLØ HOLDS THE ADDRESS OF A CORDIC CONSTANT
SHIFT = # TIMES TO GO THRU THE MAIN LOOP
Ø = 0 ?
YES
NO
INCREMENT WORD POINTED TO BY DIGIT
RETURN
COMPLEMENT Ø
INCRMT. DIGIT
TABLØ = TABLØ + 4
4 = <TABLØ>
SET X = TO NEXT CORDIC CONSTANT
DO Ø = Ø + X UNTIL OVERFLOW
SET ONE MINUS THIS COUNT IN ANSWER STACK
THE COUNT IS LOCATED IN THE STACK VIA DIGIT
Ø = -Ø + X (RESTORE)
LEFT SHIFT ON Ø
I.E., MULTIPLY Ø BY 10
DECMT. SHIFT. = 0?
YES
NO
RETURN

FIG 20

FIG 21A


IN2
PHASE II
(USED BY TAN, ATN)
SAVE = A
SHIFT = B
X = ONE
INITIALIZE RY1 & RY2
RY1 & RY2 ARE TEMPORARY FLOATING POINT REGISTERS
TAN?
YES
NO
PTL
RIGHT SHIFT & ROUND Y
I.E. DIVIDE BY 10
Y → RY1
PTN
PTC
PRESENT WORD IN DIGIT STACK = 0?
NO
PTF
X = -X
Y = Y + X
Y = Y + X
NO
OVERFLOW ?
YES
RESTORE Y (RY1)
X = -X
X = -X
Y = 0 ?
YES
NO
PTF
DECRMT. WORD POINTED TO BY DIGIT
I.E., PRESENT WORD IN THE ANSWER DIGIT STACK
RY1 ⇄ RY2
SET DIGIT TO THE FIRST WORD OF THE STACK
RETURN
TAN?
YES
NO
B = 2(SHIFT-1) + 2SAVE
B = SAVE - 2(SHIFT-1)
Y → RY1
RY2 → Y
P2

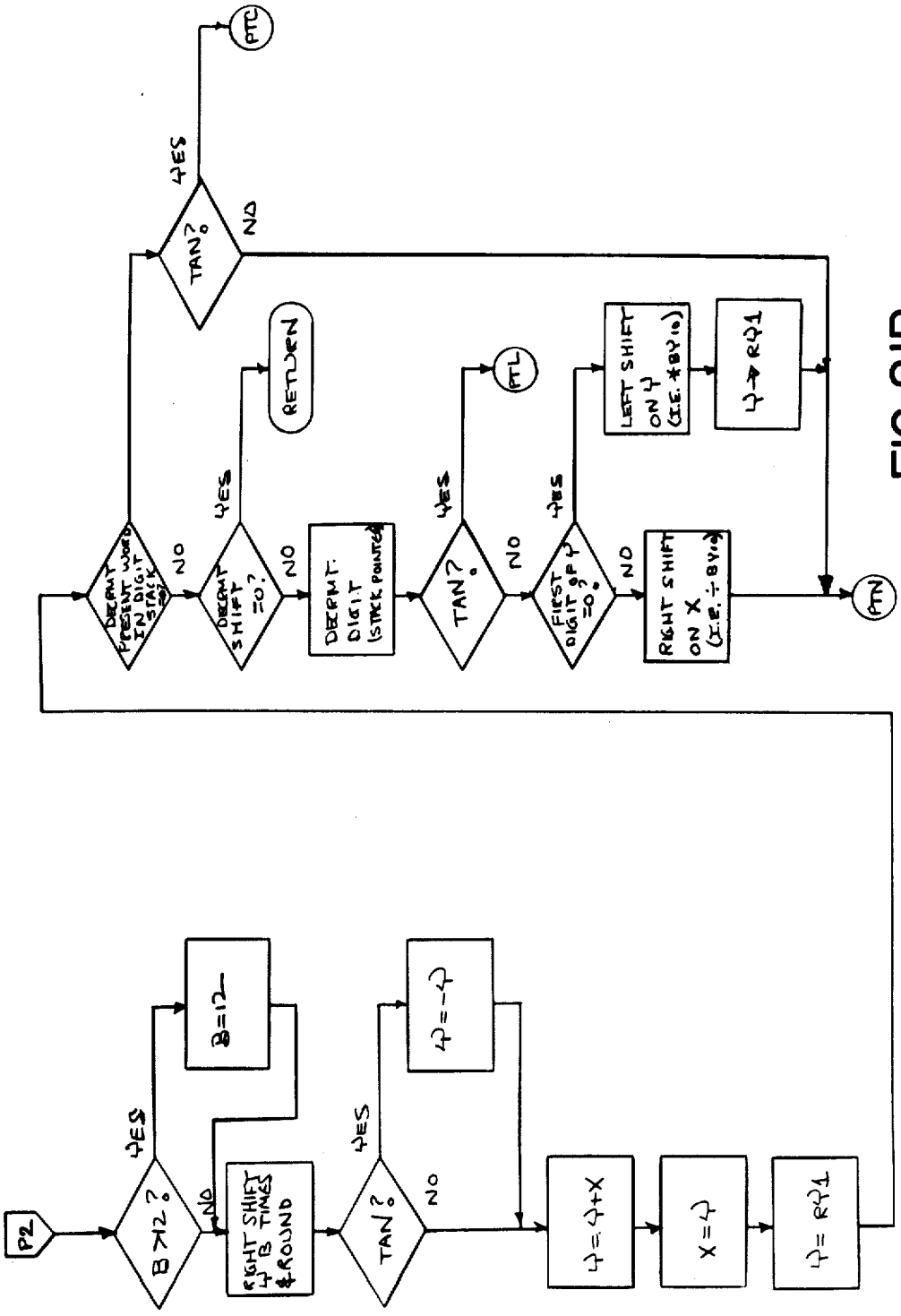

P2
B>12?
YES
NO
B=12-
RIGHT SHIFT Y B TIMES #ROUND
TAN?
YES
NO
Y=-Y
Y=Y+X
X=Y
Y=RY1
TAN?
YES
NO
PTL
FIRST DIGIT OF Y =0.?
YES
NO
LEFT SHIFT ON Y (I.E. *BY10)
Y→RY1
RIGHT SHIFT ON X (I.E. ÷BY10)
DECIMT SHIFT =0.?
YES
NO
RETURN
DECPNT DIGIT ISTACK POINTER
DECPNT PRESENT WORD IN DIGIT STACK
NO
TAN?
YES
NO
PTC
PTN

FIG 21B

FIG 22A

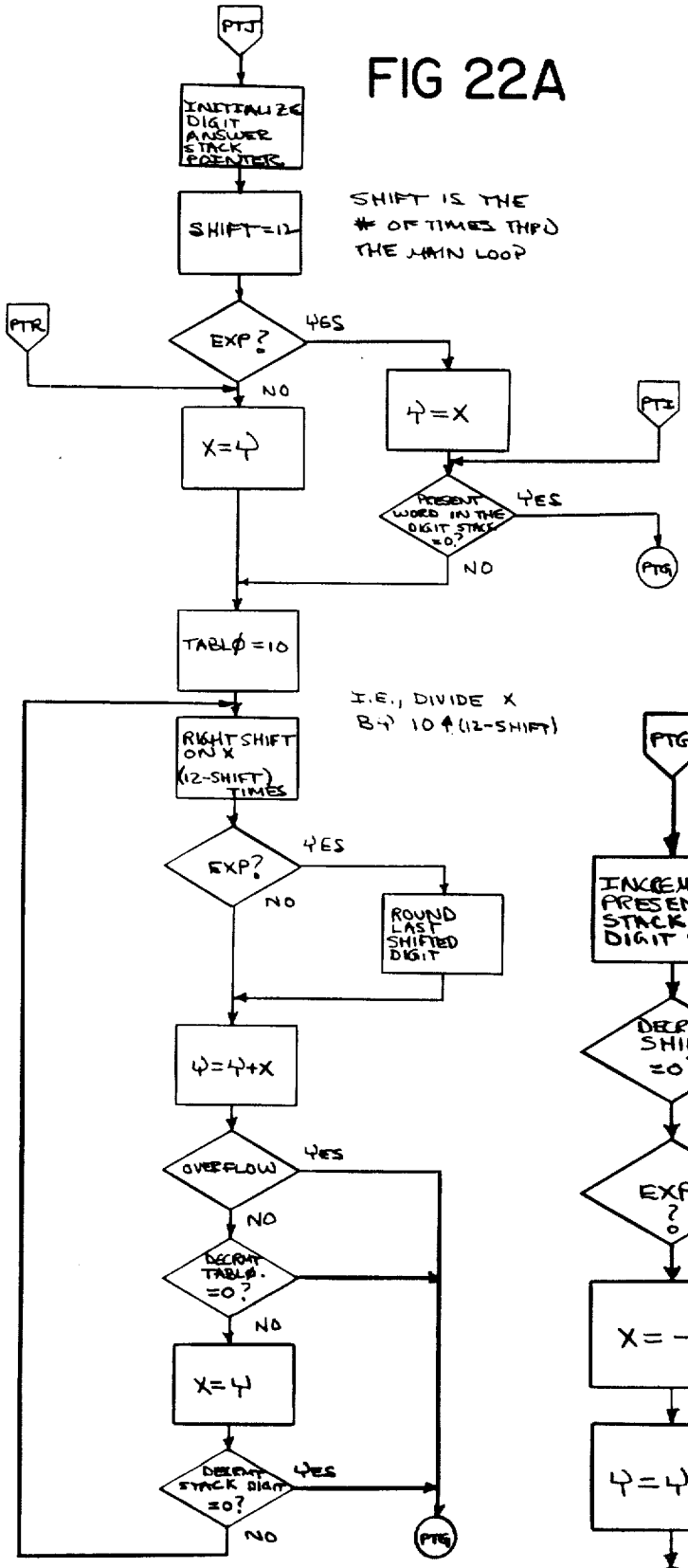


FIG 22B

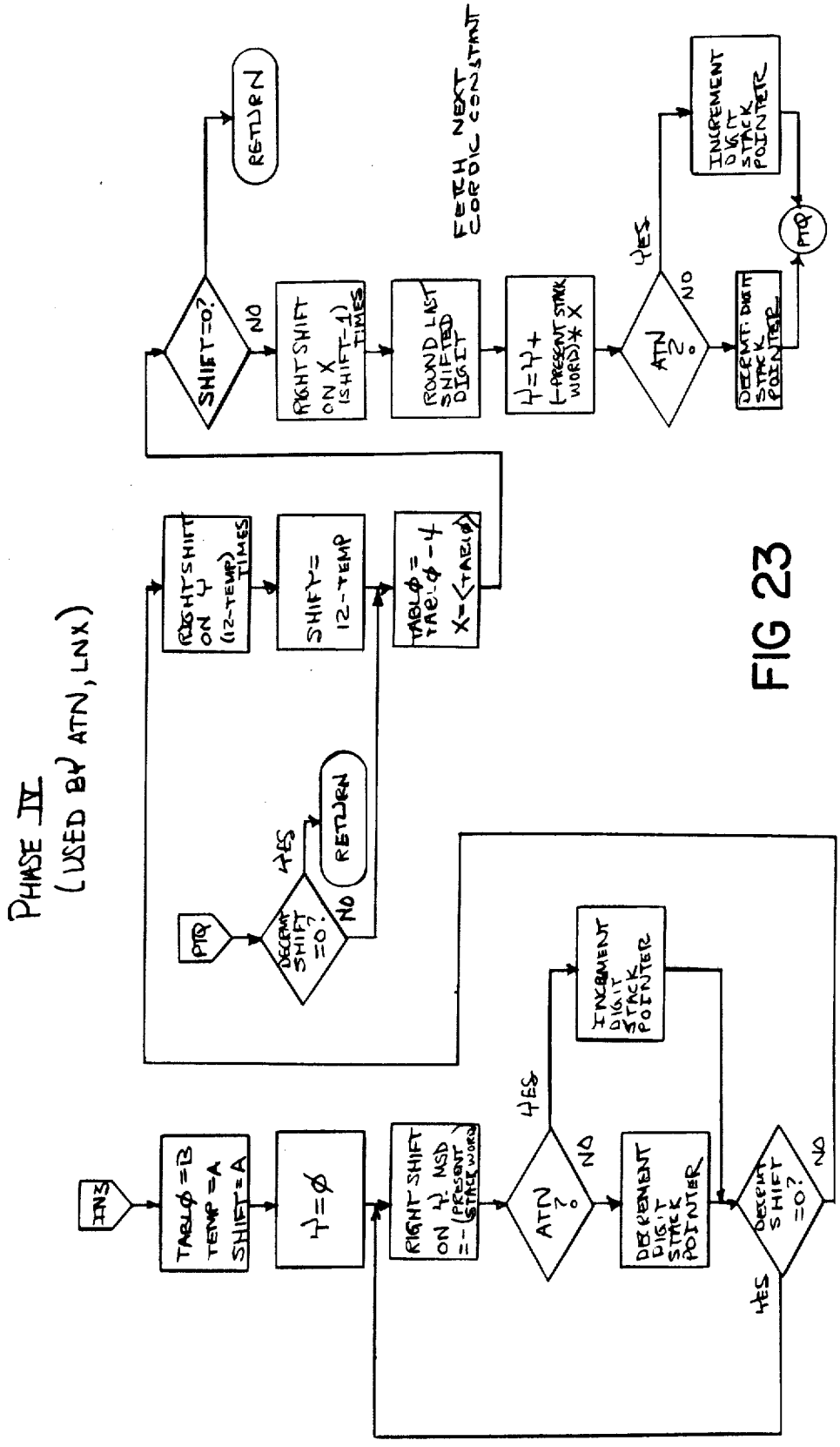
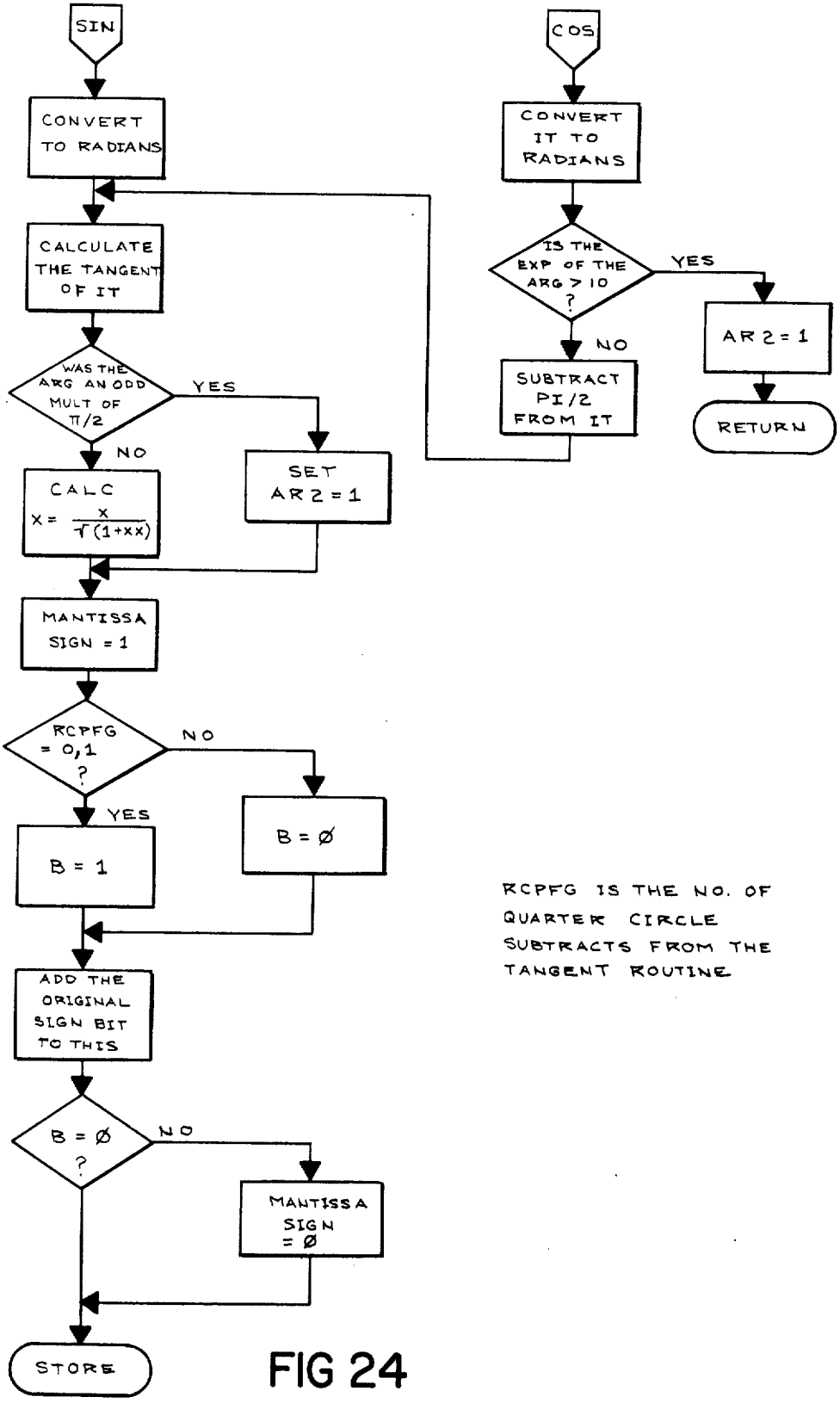

PHASE IV
(USED BY ATN, LNX)
FETCH NEXT CORDIC CONSTANT
INX
TABLΦ = 13
TEMP = A
SHIFT = A
4 = Φ
RIGHT SHIFT ON 4 NSD = (PRESENT STACK WORD)
ATN ?
YES
NO
INCREMENT DIGIT STACK POINTER
DECREMENT DIGIT STACK POINTER
DECRMT SHIFT = 0?
YES
NO
PTR
DECRMT SHIFT = 0?
YES
NO
RETURN
RIGHT SHIFT ON 4 (12-TEMP) TIMES
SHIFT = 12-TEMP
TABLΦ = TABLΦ - 4
X = <TABLΦ
SHIFT = 0?
NO
RETURN
RIGHT SHIFT ON X (SHIFT-1) TIMES
ROUND LAST SHIFTED DIGIT
4 = 4 + (PRESENT STACK WORD) * X
ATN ?
YES
NO
INCREMENT DIGIT STACK POINTER
DECRMT DIGIT STACK POINTER
PTR

FIG 23

SINE AND COSINE ROUTINE



RCPFG IS THE NO. OF QUARTER CIRCLE SUBTRACTS FROM THE TANGENT ROUTINE

FIG 24

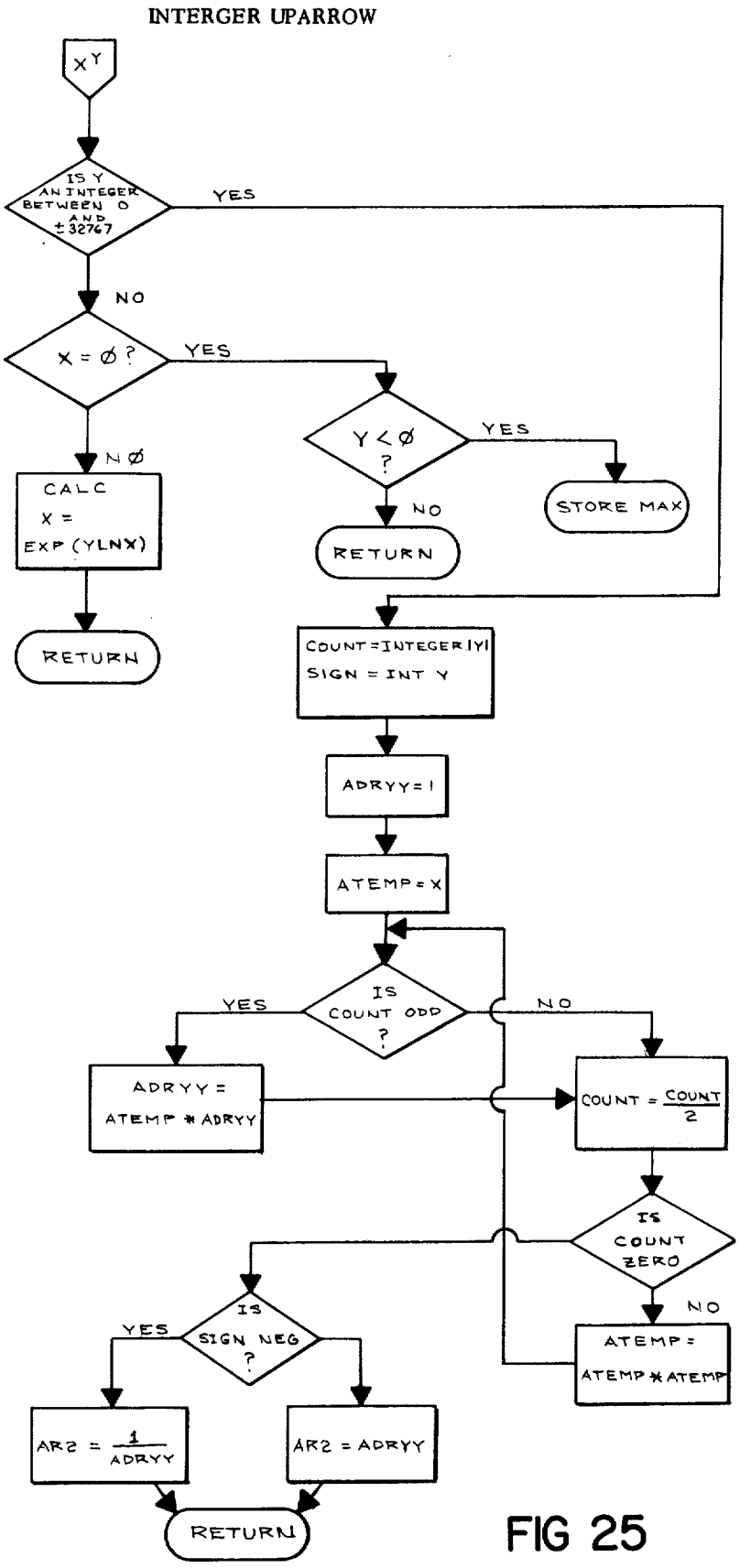
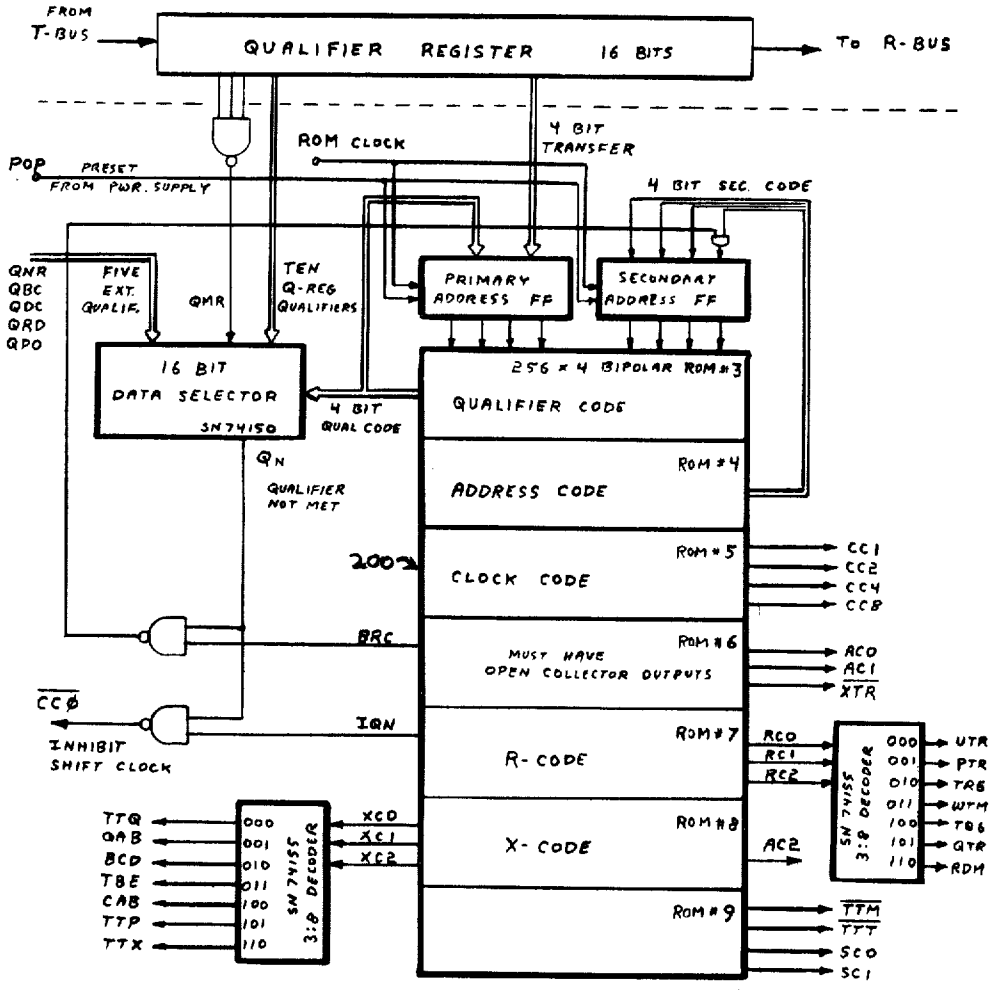

INTERGER UPARROW
XY
IS Y AN INTEGER BETWEEN 0 AND ±32767
YES
NO
X = Ø ?
YES
NØ
Y < Ø ?
YES
NO
CALC X = EXP (YLNX)
STORE MAX
RETURN
RETURN
COUNT = INTEGER|Y|
SIGN = INT Y
ADRYY = 1
ATEMP = X
IS COUNT ODD ?
YES
NO
ADRYY = ATEMP * ADRYY
COUNT = COUNT / 2
IS COUNT ZERO
NO
ATEMP = ATEMP * ATEMP
IS SIGN NEG ?
YES
AR2 = 1 / ADRYY
AR2 = ADRYY
RETURN

FIG 25

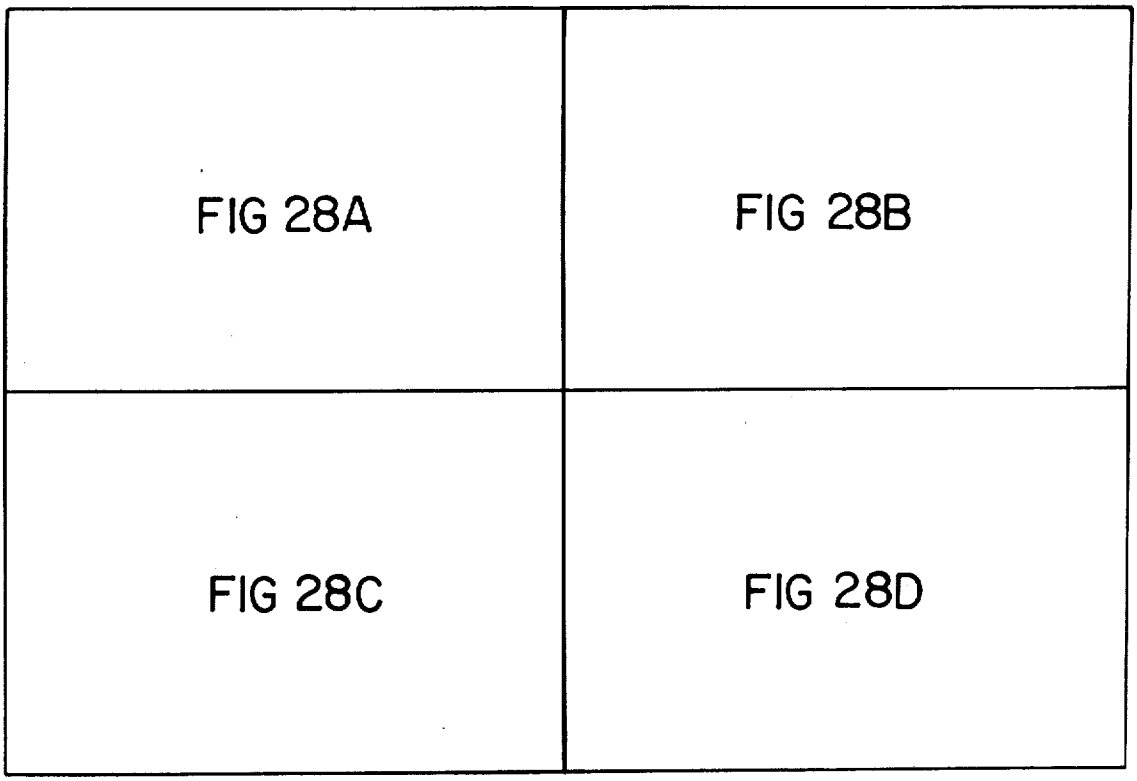
MICRO PROCESSOR



ROM ORGANIZATION: 256 WORDS x 28 BITS, EXPANDED TO 36 μ-INSTRUCTIONS

FIG 27

FIG 28'



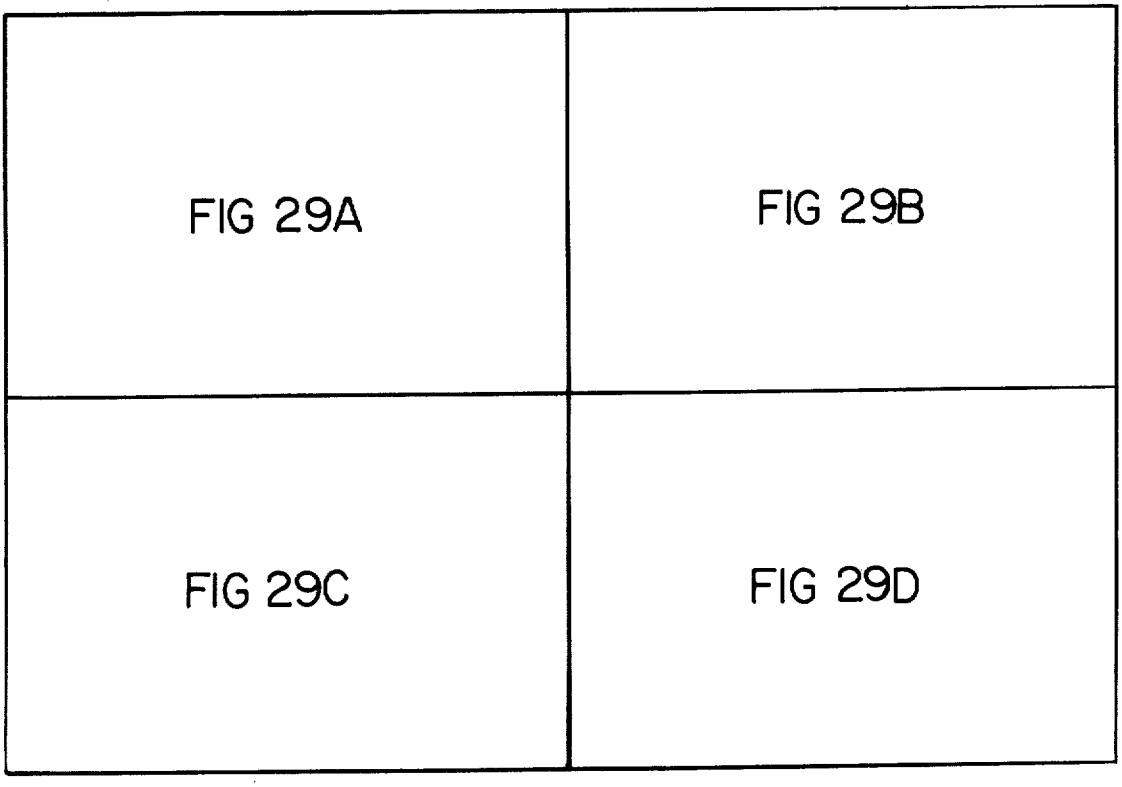


FIG 29'

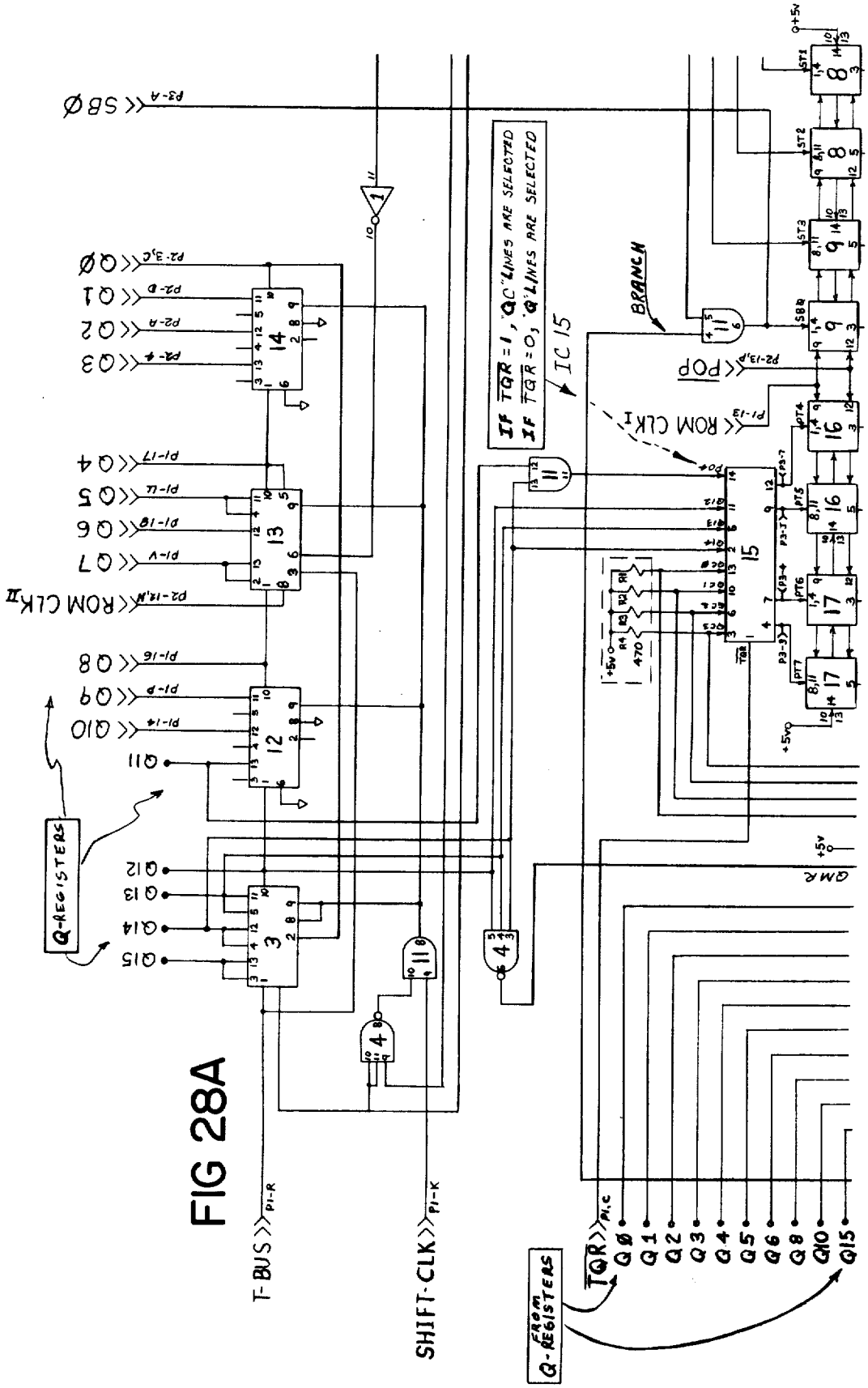
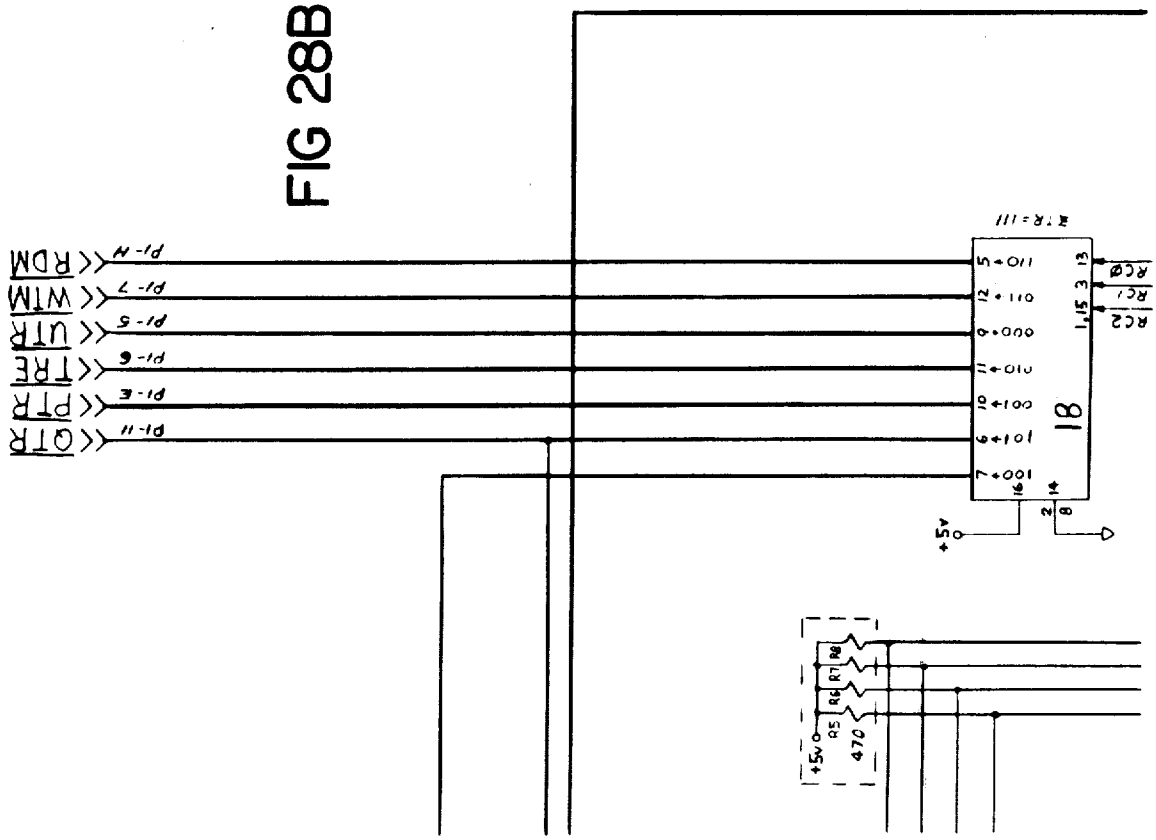
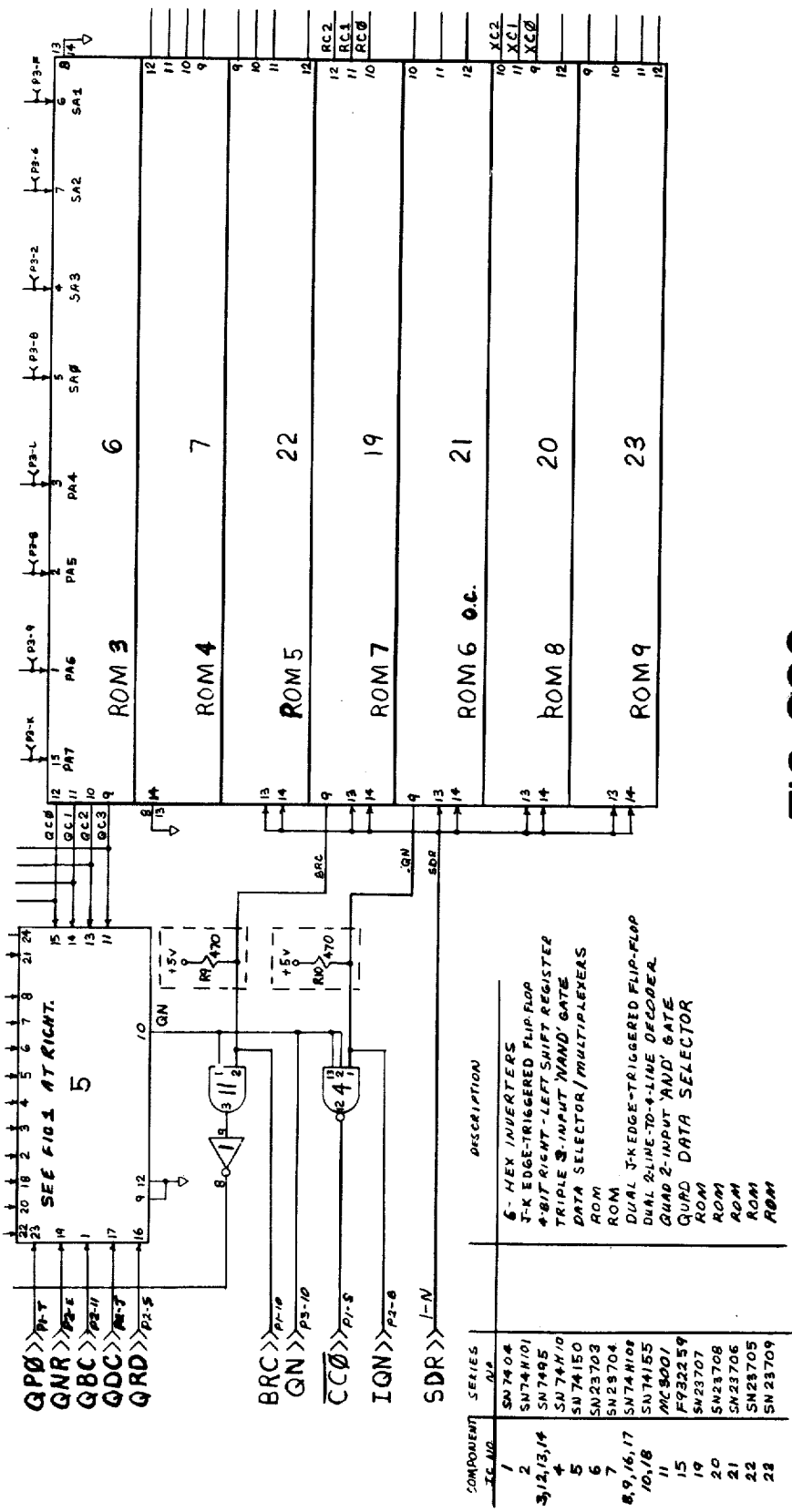

IF TQR = 1, 'Q' LINES ARE SELECTED
IF TQR = 0, 'Q' LINES ARE SELECTED
IC 15
BRANCH
ROM CLK I
ROM CLK II
Q-REGISTERS
Q-REGISTERS
T-BUS
SHIFT-CLK
TQR
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q8
Q10
Q15

FIG 28A

FIG 28B





COMPONENT IC NO	SERIES NO	DESCRIPTION
1	SN7404	6- HEX INVERTERS
2	SN74H101	J-K EDGE-TRIGGERED FLIP-FLOP
3,12,13,14	SN7495	4-BIT RIGHT-LEFT SHIFT REGISTER
4	SN74H10	TRIPLE 3-INPUT 'NAND' GATE
5	SN74150	DATA SELECTOR/MULTIPLEXERS
6	SN23703	ROM
7	SN23704	ROM
8,9,16,17	SN74H108	DUAL J-K EDGE-TRIGGERED FLIP-FLOP
10,18	SN74155	DUAL 2-LINE-TO-4-LINE DECODER
11	MC3001	QUAD 2-INPUT 'AND' GATE
15	F932259	QUAD DATA SELECTOR
19	SN23707	ROM
20	SN23708	ROM
21	SN23706	ROM
22	SN23705	ROM
23	SN23709	ROM

FIG 28C

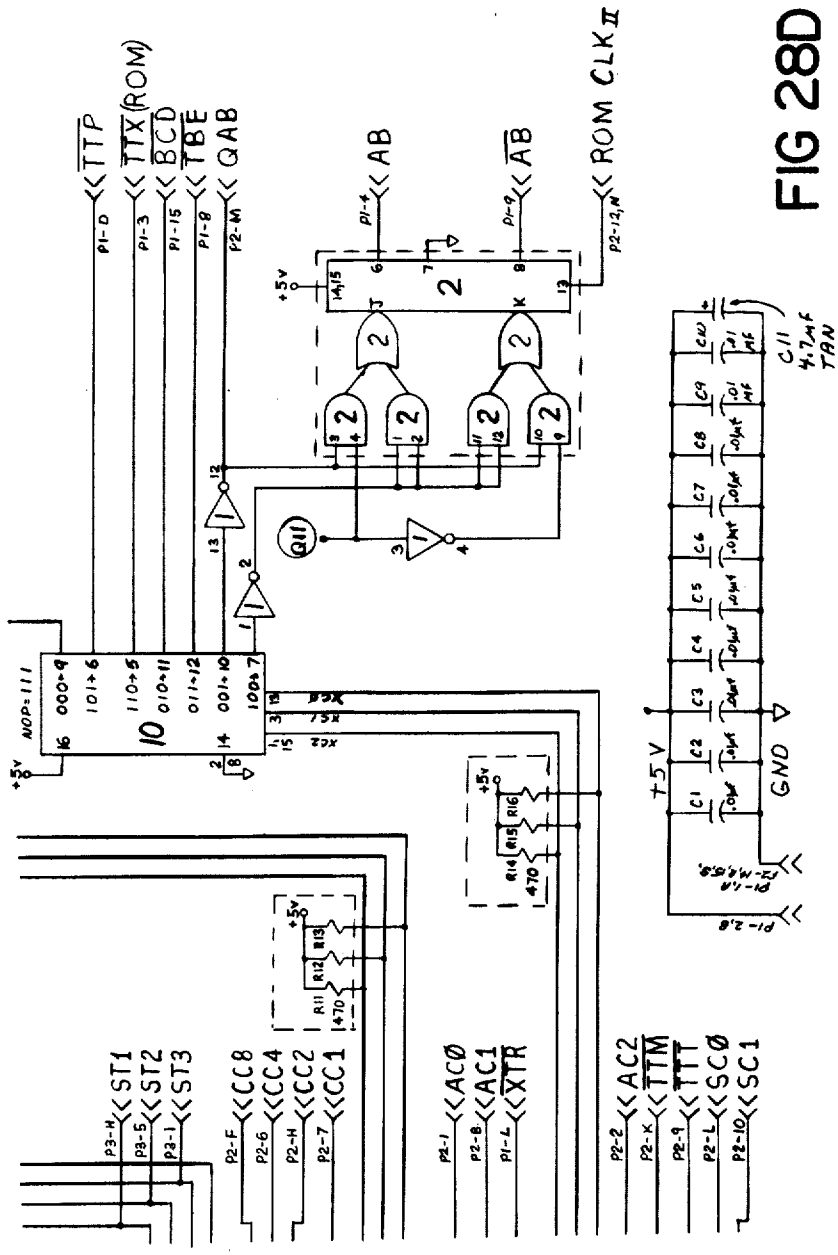

TTP
TTX (ROM)
BCD
TBE
QAB
AB
AB
ROM CLK II
Q11
MOD=111
000→9
101→6
110→5
010→11
011→12
001→10
100→7
10
+5V
GND
ST1
ST2
ST3
CC8
CC4
CC2
CC1
AC0
AC1
XTR
AC2
TTM
TTT
SC0
SC1
C11
4.7MF
TAN

FIG 28D

FIG 29"

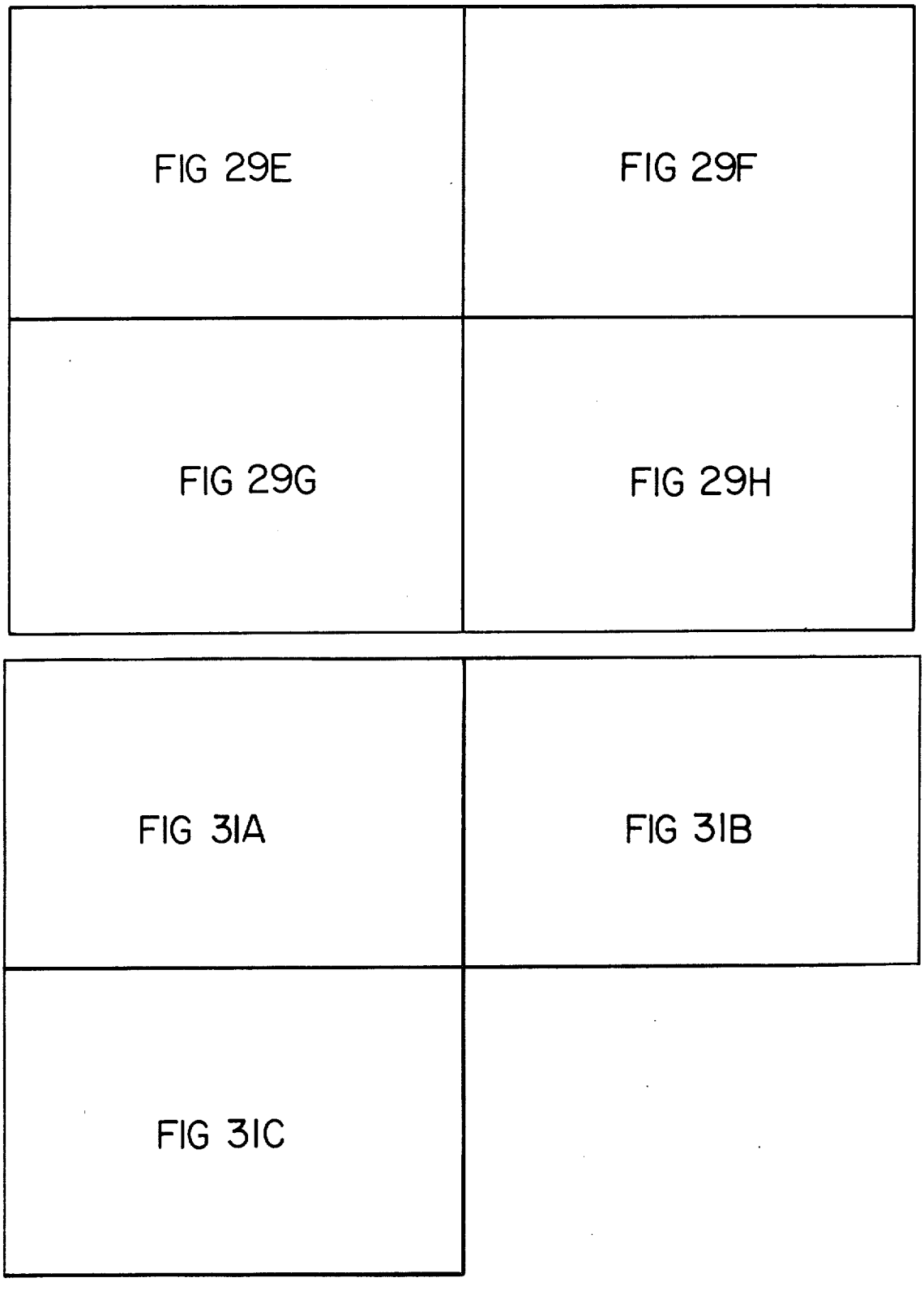
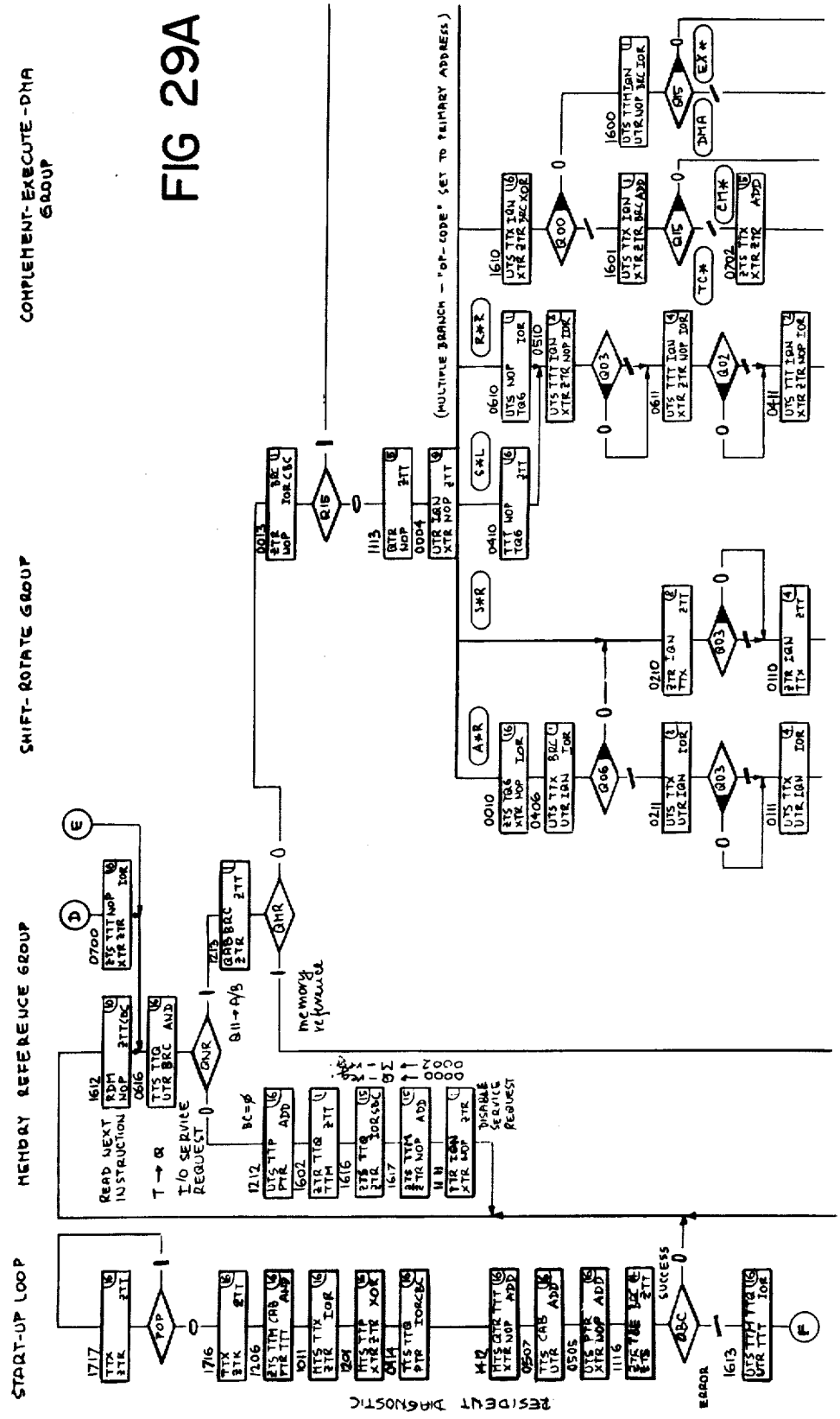


FIG 31'

FIG 29A


START-UP LOOP
MEMORY REFERENCE GROUP
SHIFT-ROTATE GROUP
COMPLEMENT-EXECUTE-DMA GROUP
RESIDENT DIAGNOSTIC
READ NEXT INSTRUCTION
I/O SERVICE REQUEST
DISABLE SERVICE REQUEST
MEMORY REFRESH
(MULTIPLE BRANCH - "OP-CODE" SET TO PRIMARY ADDRESS)
ERROR
SUCCESS

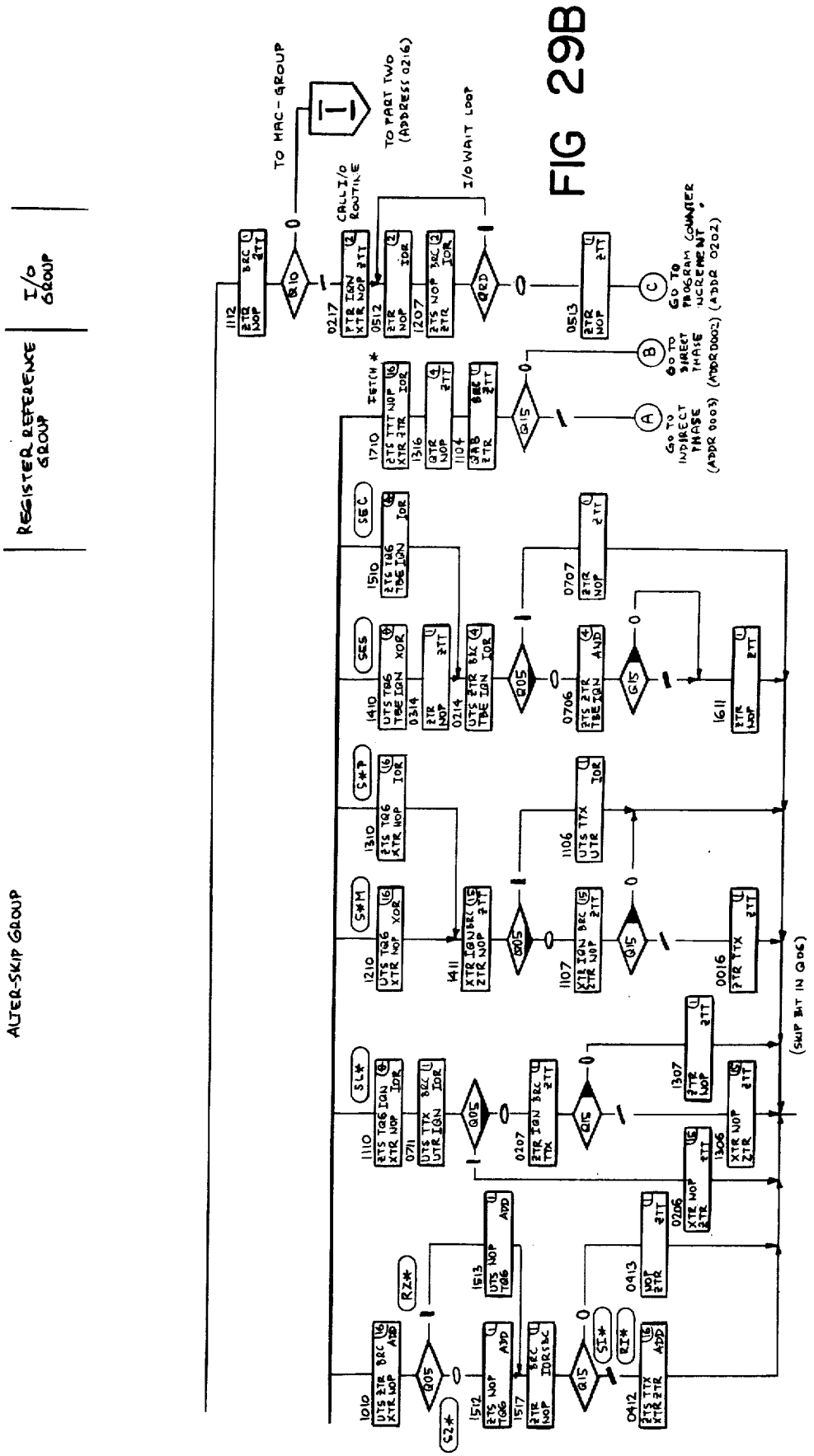

REGISTER REFERENCE GROUP
I/O GROUP
ALTER-SKIP GROUP
TO MAC-GROUP
CALL I/O ROUTINE
TO PART TWO (ADDRESS 0216)
I/O WAIT LOOP
GO TO PROGRAM COUNTER INCREMENT (ADDR 0202)
GO TO SHIFT PHASE (ADDR 0002)
GO TO INDIRECT PHASE (ADDR 0003)
(SKIP BIT IN Q06)

FIG 29B

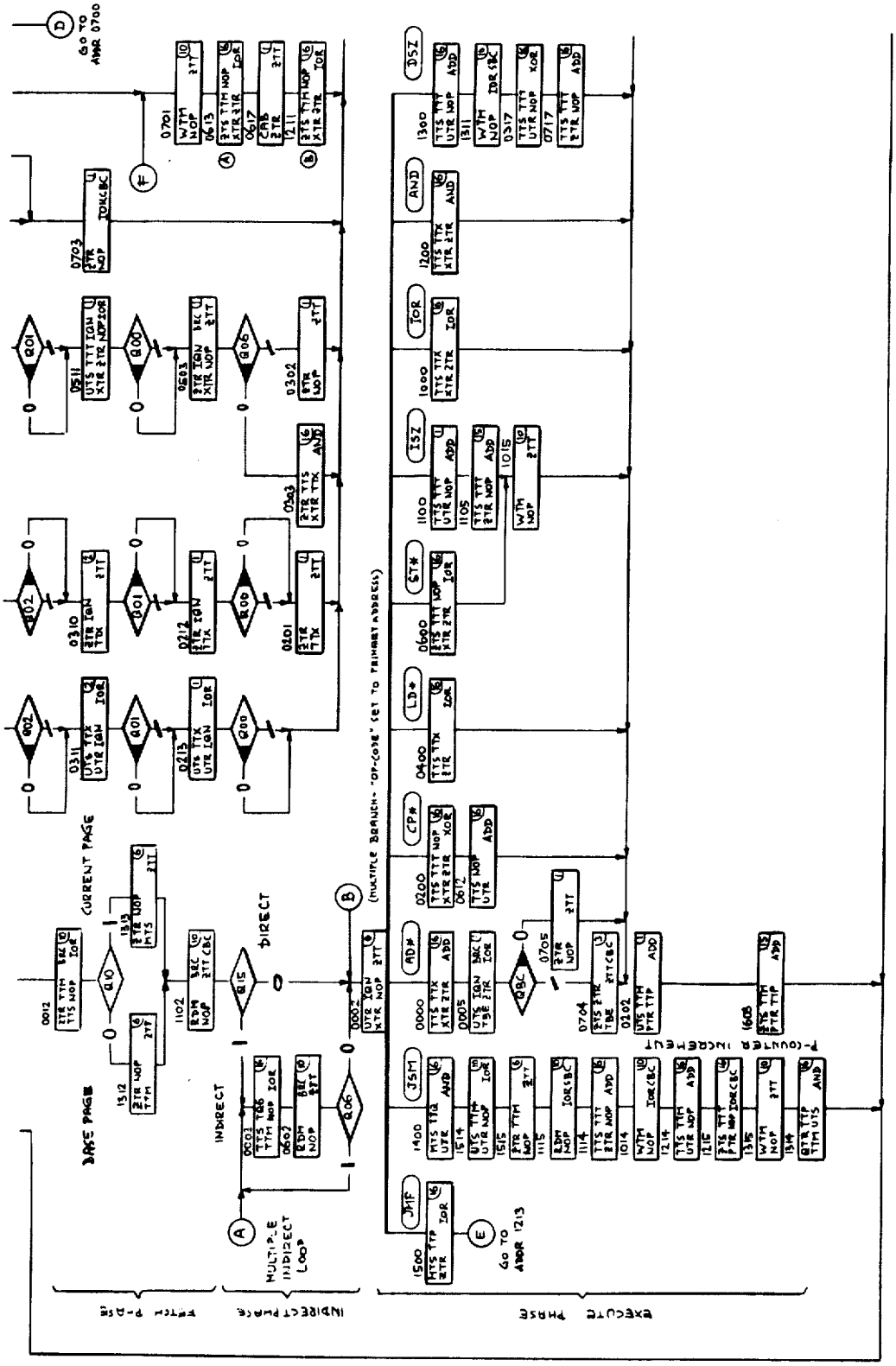


FIG 29C

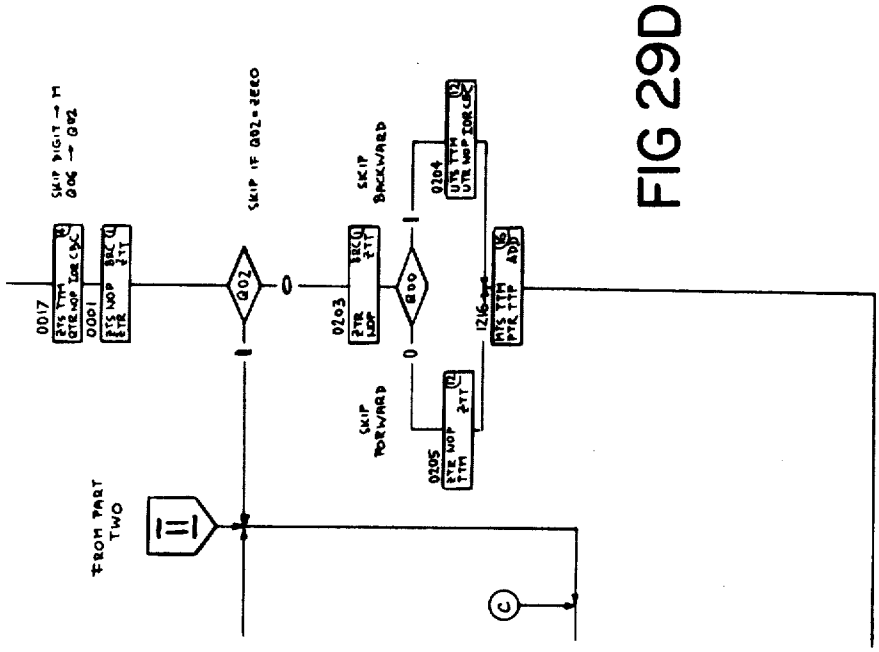
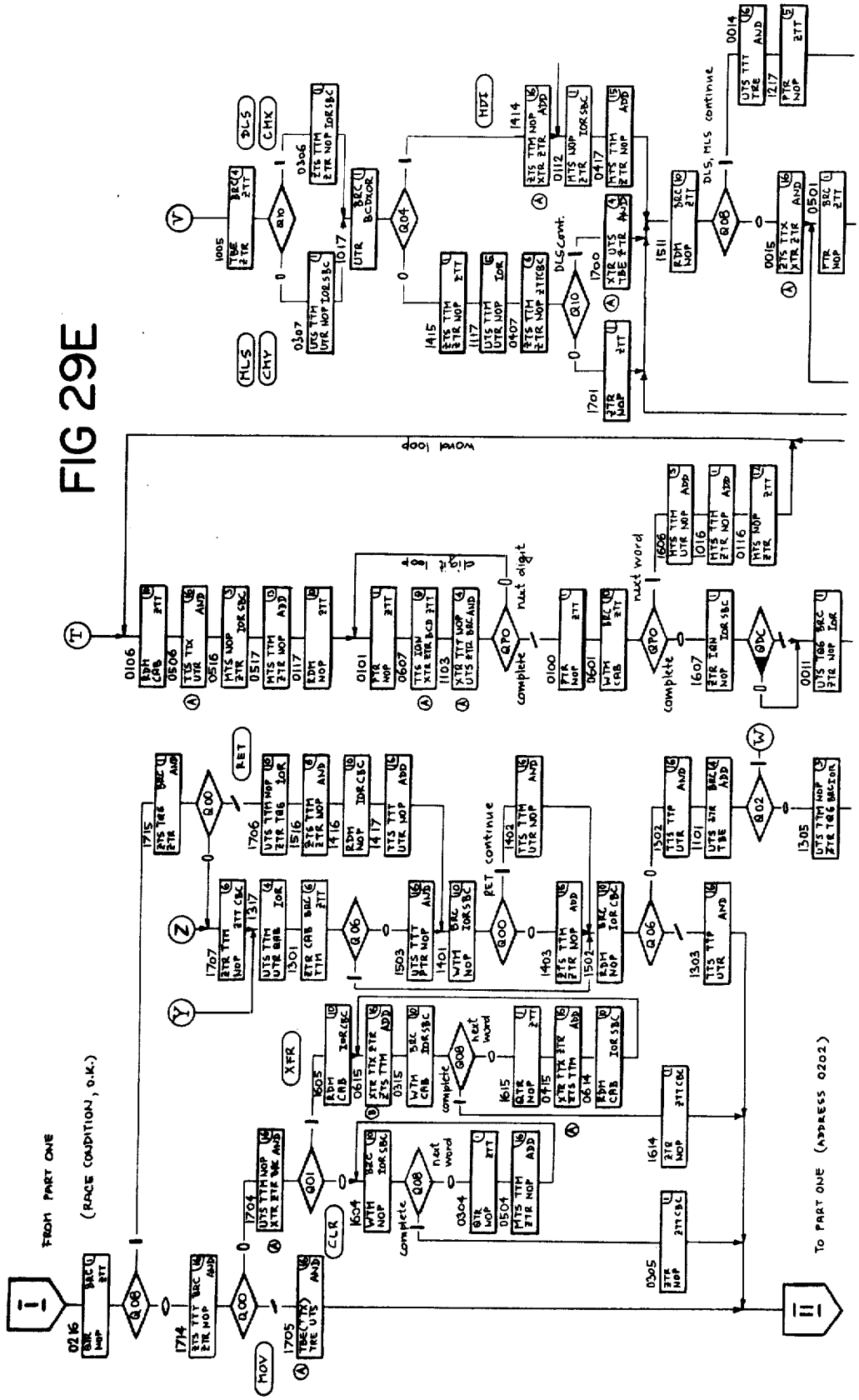

FROM PART TWO
SKIP BACKWARD
SKIP FORWARD
SKIP IF Q02 = ZERO
0017
0001
0203
0204
0205
1216
Q02
Q00
C

FIG 29D

FIG 29E


FROM PART ONE
(RACE CONDITION, O.K.)
TO PART ONE (ADDRESS 0202)
next word
complete
next digit
word loop
DLS, MLS continue
DLS cont.
RET
XFR
CLR
MOV
DLS
CMX
MLS
CMY
HOLD

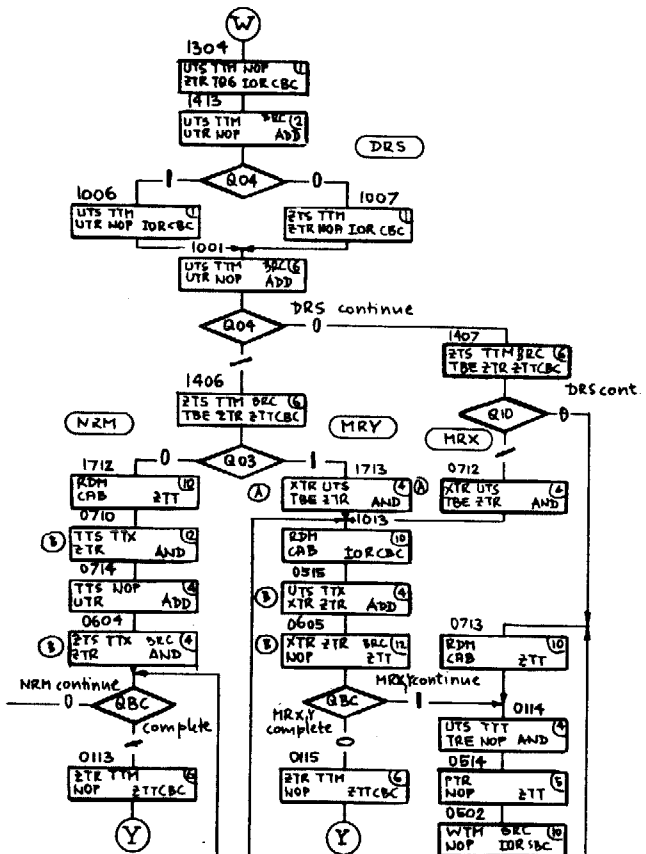


FIG 29F

① MAC-INSTRUCTION CODING TABLE :

MNEMONIC	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OCTAL
RET	1	1	1	1	0	0	0	1	0	0	0	0	0	1	0		170402
MOV	1	1	1	1	•	•	•	0	0	0	•	0	•	0	1	0	170002
CLR	1	1	1	1	0	•	•	0	•	•	0	0	0	0	0	0	170000
XFR	1	1	1	1	0	0	•	0	•	•	0	0	•	1	•	0	170004
MRY	1	1	1	1	1	0	•	1	•	•	1	1	1	•	•	•	174470
MLS	1	1	1	1	•	0	1	1	•	•	•	•	0	•	•	•	171400
DRS	1	1	1	1	0	0	0	1	•	•	0	0	1	0	0	0	170410
DLS	1	1	1	1	1	0	1	1	•	•	•	•	•	0	0	0	175400
FXA	1	1	1	1	0	•	0	1	0	1	1	1	0	•	0	0	170560
FMP	1	1	1	1	0	0	0	1	1	0	0	1	1	0	•	0	171460
FDV	1	1	1	1	0	0	0	1	1	0	0	0	1	1	0	0	170420
SMX	1	1	1	1	1	0	•	1	0	•	0	•	0	•	•	0	174400
CMY	1	1	1	1	0	0	0	1	0	0	0	0	0	0	•	0	170400
MDI	1	1	1	1	•	0	•	1	0	1	1	0	0	0	0	0	170540
NRM	1	1	1	1	0	•	1	1	0	0	1	0	1	0	0	0	171450
MRX	1	1	1	1	1	0	0	1	0	0	0	1	1	0	0	0	174430

② USED DATA FORMAT :

ADDRESS	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M	SE	exponent														SM
M+1	D_1				D_2				D_3				D_4			
M+2	D_5				D_6				D_7				D_8			
M+3	D_9				D_{10}				D_{11}				D_{12}			

D_1 ... most significant digit
D_{12} ... least significant digit
se ... sign of exponent
sm ... sign of mantissa
decimal point is located between D_1 and D_2

③ USED ADDRESSES AND CONSTANTS (OCTAL)

0001 counter constant address
1700 P-register contents storage
1777 stack-pointer address
1744 - 47 AR1 } arithmetic pseudoregisters
1754 - 57 AR2 }
043060 counter constant

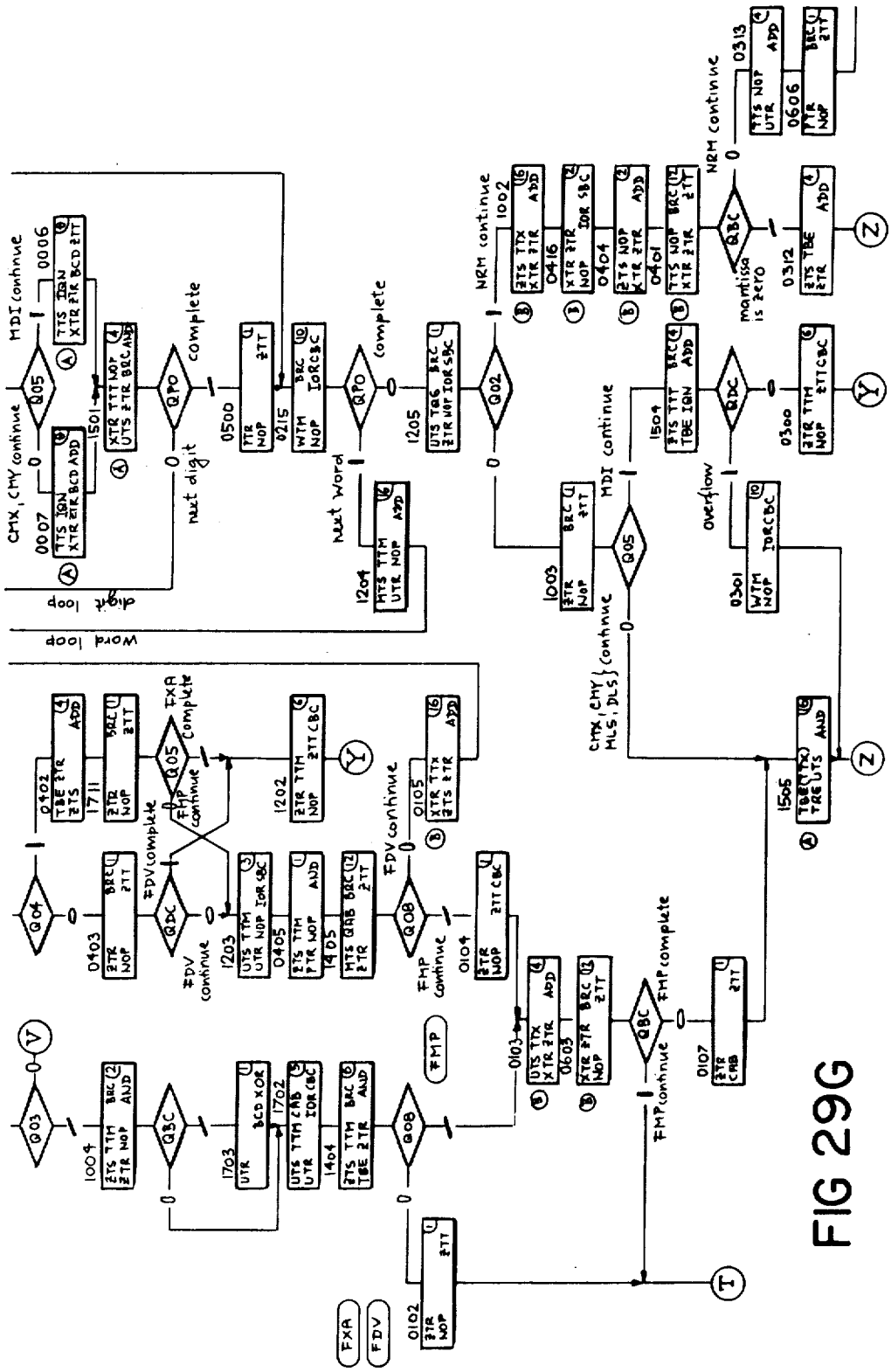


FIG 29G

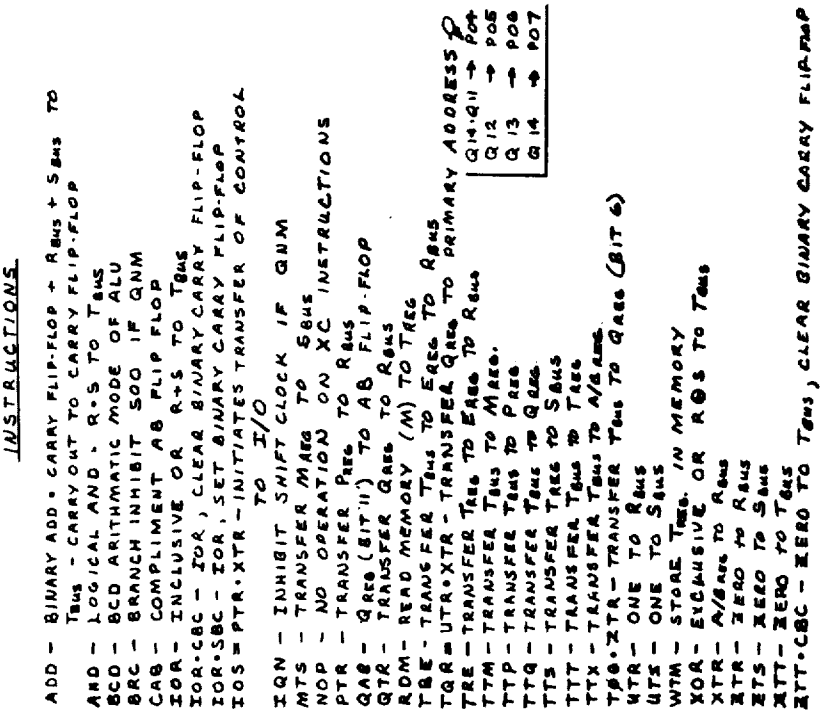

INSTRUCTIONS
ADD - BINARY ADD • CARRY FLIP-FLOP → RBUS + SBUS TO TBUS - CARRY OUT TO CARRY FLIP-FLOP
AND - LOGICAL AND . R•S TO TBUS
BCD - BCD ARITHMATIC MODE OF ALU
BRC - BRANCH INHIBIT SOO IF QNM
CAB - COMPLIMENT AB FLIP FLOP
IOR - INCLUSIVE OR R+S TO TBUS
IOR•CBC - IOR, CLEAR BINARY CARRY FLIP-FLOP
IOR•SBC - IOR, SET BINARY CARRY FLIP-FLOP
IOS = PTR•XTR - INITIATES TRANSFER OF CONTROL TO I/O
IQN - INHIBIT SHIFT CLOCK IF QNM
MTS - TRANSFER MREG TO SBUS
NOP - NO OPERATION ON XC INSTRUCTIONS
PTR - TRANSFER PREG TO RBUS
QAB - QREG (BIT 11) TO AB FLIP-FLOP
QTR - TRANSFER QREG TO RBUS
RDM - READ MEMORY (M) TO TREG
TBE - TRANSFER TBUS TO EREG TO RBUS
TQR = UTR•XTR - TRANSFER QREG TO PRIMARY ADDRESS
TRE - TRANSFER TREG TO EREG TO RBUS
TTM - TRANSFER TBUS TO MREG.
TTP - TRANSFER TBUS TO PREG
TTQ - TRANSFER TBUS TO QREG
TTS - TRANSFER TREG TO SBUS
TTT - TRANSFER TBUS TO TREG
TTX - TRANSFER TBUS TO A/S REG.
TXB•XTR - TRANSFER TBUS TO QREG (BIT 6)
UTR - ONE TO RBUS
UTS - ONE TO SBUS
WTM - STORE TREG. IN MEMORY
XOR - EXCLUSIVE OR RBUS TO TBUS
XTR - A/S REG TO RBUS
ZTR - ZERO TO RBUS
ZTS - ZERO TO SBUS
ZTT - ZERO TO TBUS
ZTT•CBC - ZERO TO TBUS, CLEAR BINARY CARRY FLIP-FLOP
Q11:Q11 → P04
Q12 → P05
Q13 → P06
Q14 → P07
complete
Q10
next word
word loop
1506
HTS CAB
2TT
0716
1507
HTS TTM
ETR CAB
ADD
HTS TTM
ETR NOP
ALUD
0715
HTS UOP
2TR
2TT
0316
TBE(TTX)
TRE UTS
BRCUS
AND
NEXT
continue
3RS complete
1012
ETR TTM
UOP
2TT CBC

FIG 29H

PROGRAMMABLE CLOCK

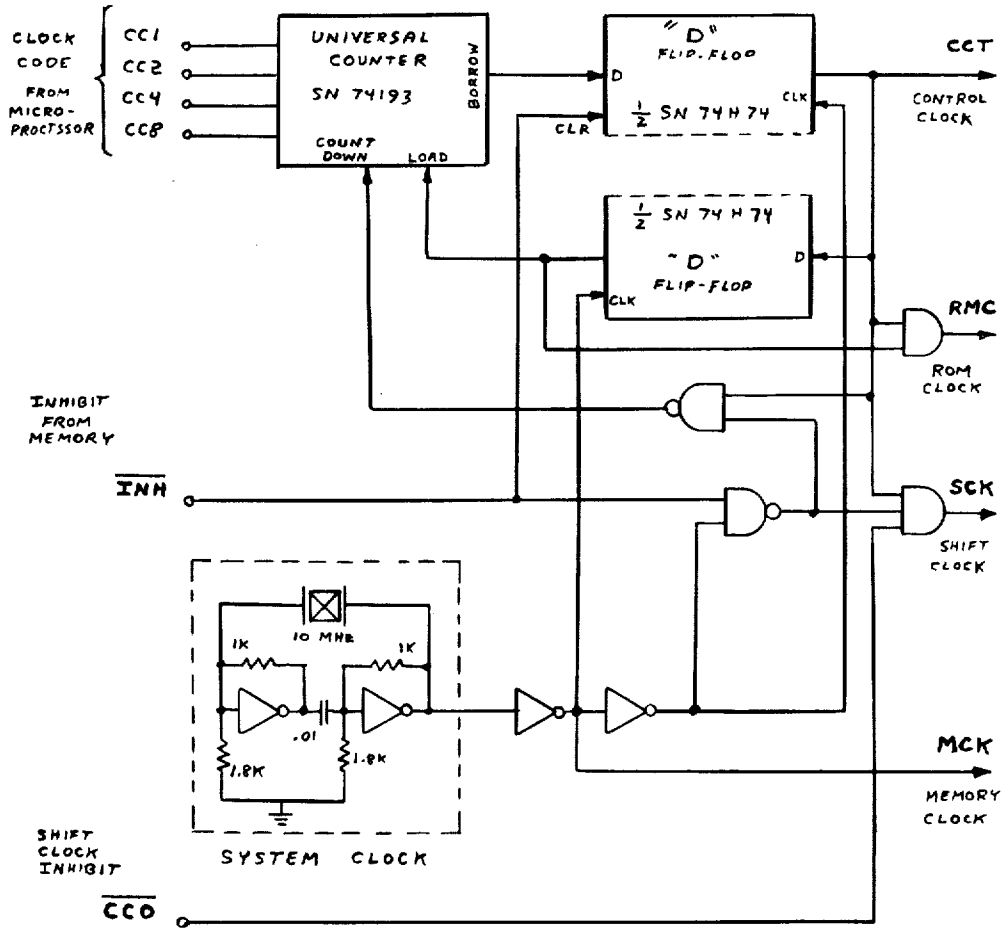
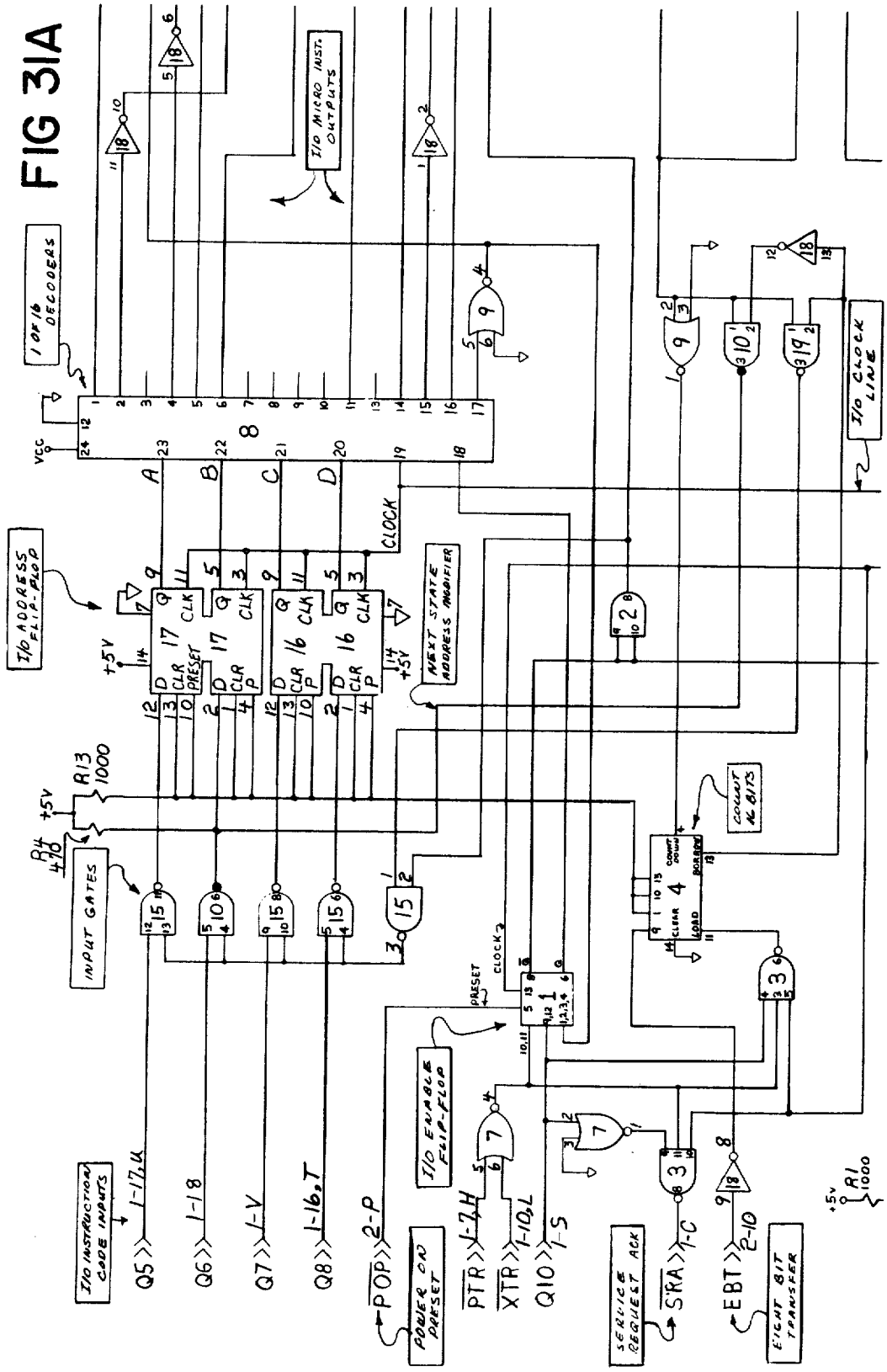


FIG 30

FIG 31A


I/O MICRO INST. OUTPUTS
1 OF 16 DECODERS
I/O ADDRESS FLIP-FLOP
NEXT STATE ADDRESS MODIFIER
I/O CLOCK LINE
COUNT 16 BITS
INPUT GATES
I/O INSTRUCTION CODE INPUTS
I/O ENABLE FLIP-FLOP
POWER ON PRESET
SERVICE REQUEST ACK
EIGHT BIT TRANSFER
CLOCK
PRESET
VCC
+5V
R13 1000
R14 470
R1 1000
Q5>> I-17,U
Q6>> I-18
Q7>> I-V
Q8>> I-16,T
POR>> 2-P
PTR>> I-7,H
XTR>> I-10,L
Q10>> I-S
SRA>> I-C
EBT>> 2-10

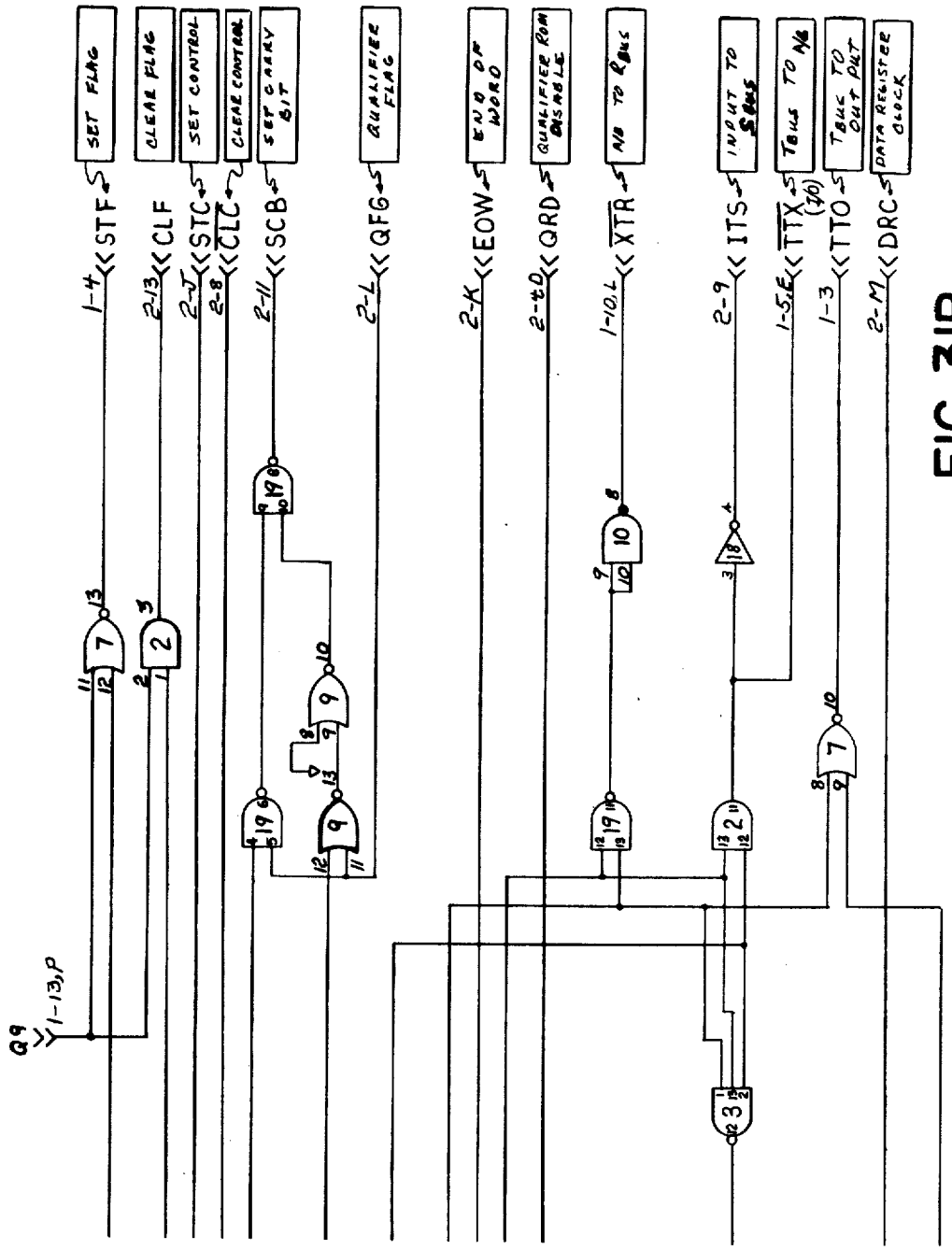

SET FLAG
CLEAR FLAG
SET CONTROL
CLEAR CONTROL
SET CHARY BIT
QUALIFIER FLAG
END OF WORD
QUALIFIER ROM ENABLE
NB TO RBUS
INPUT TO SBUS
TBUS TO NB
TBUS TO OUT PUT
DATA REGISTER CLOCK
STF
CLF
STC
CLC
SCB
QFG
EOW
QRD
XTR
ITS
TTX
TTO
DRC
Q9
1-13,P

FIG 31B

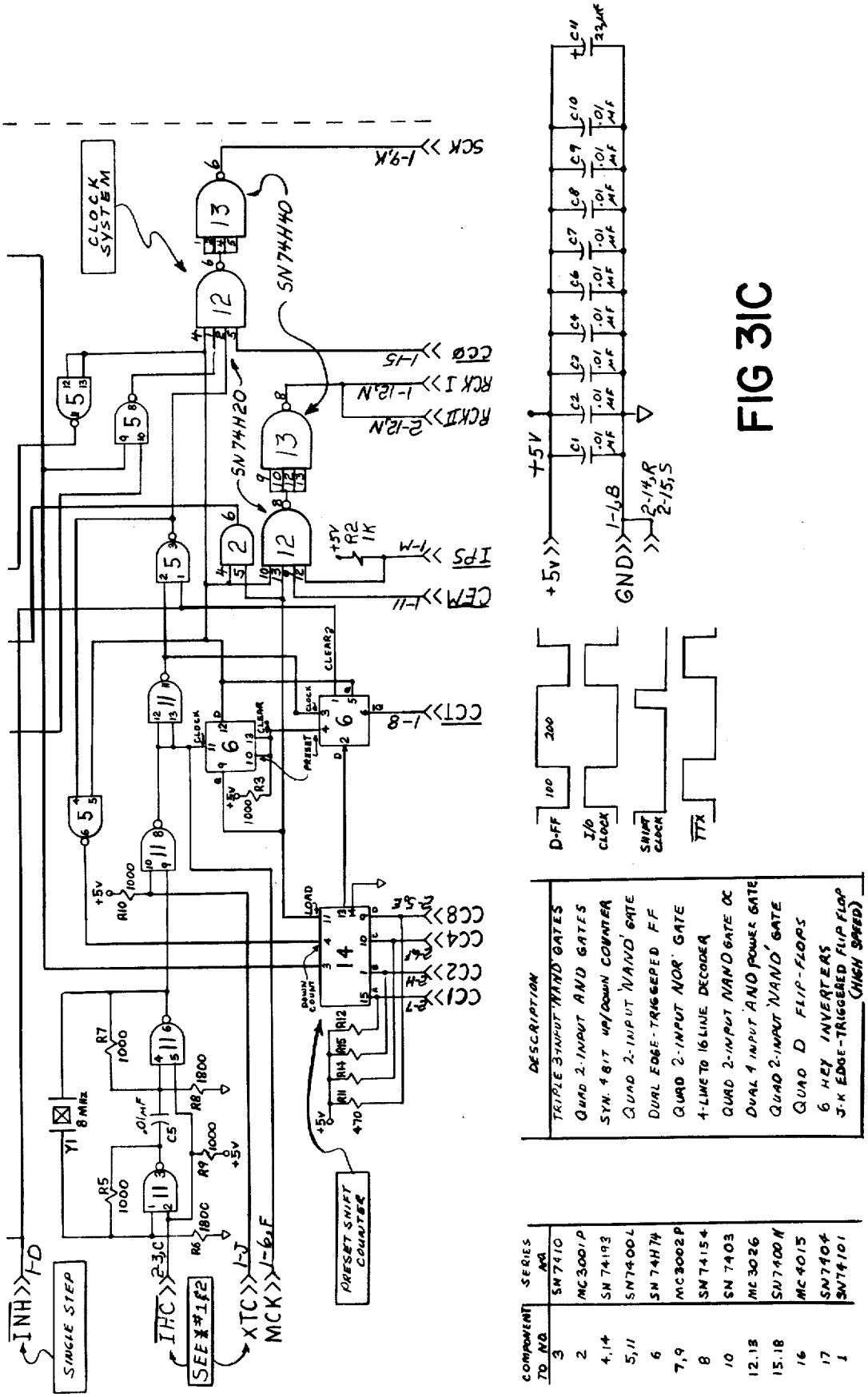


FIG 3IC

COMPONENT TO NO	SERIES AA	DESCRIPTION
3	SN7410	TRIPLE 3-INPUT 'NAND' GATES
2	MC3001P	QUAD 2-INPUT AND GATES
4,14	SN74193	SYN. 4 BIT UP/DOWN COUNTER
5,11	SN7400L	QUAD 2-INPUT 'NAND' GATE
6	SN74H74	DUAL EDGE-TRIGGERED FF
7,9	MC3002P	QUAD 2-INPUT NOR GATE
8	SN74154	4-LINE TO 16 LINE DECODER
10	SN7403	QUAD 2-INPUT NAND GATE OC
12,13	MC3026	DUAL 4 INPUT AND POWER GATE
15,18	SN7400N	QUAD 2-INPUT 'NAND' GATE
16	MC4015	QUAD D FLIP-FLOPS
17	SN7404	6 HEX INVERTERS
1	SN74101	J-K EDGE-TRIGGERED FLIP FLOP (HIGH SPEED)

FIG 32 TIMING WAVEFORMS - CENTRAL PROCESSOR

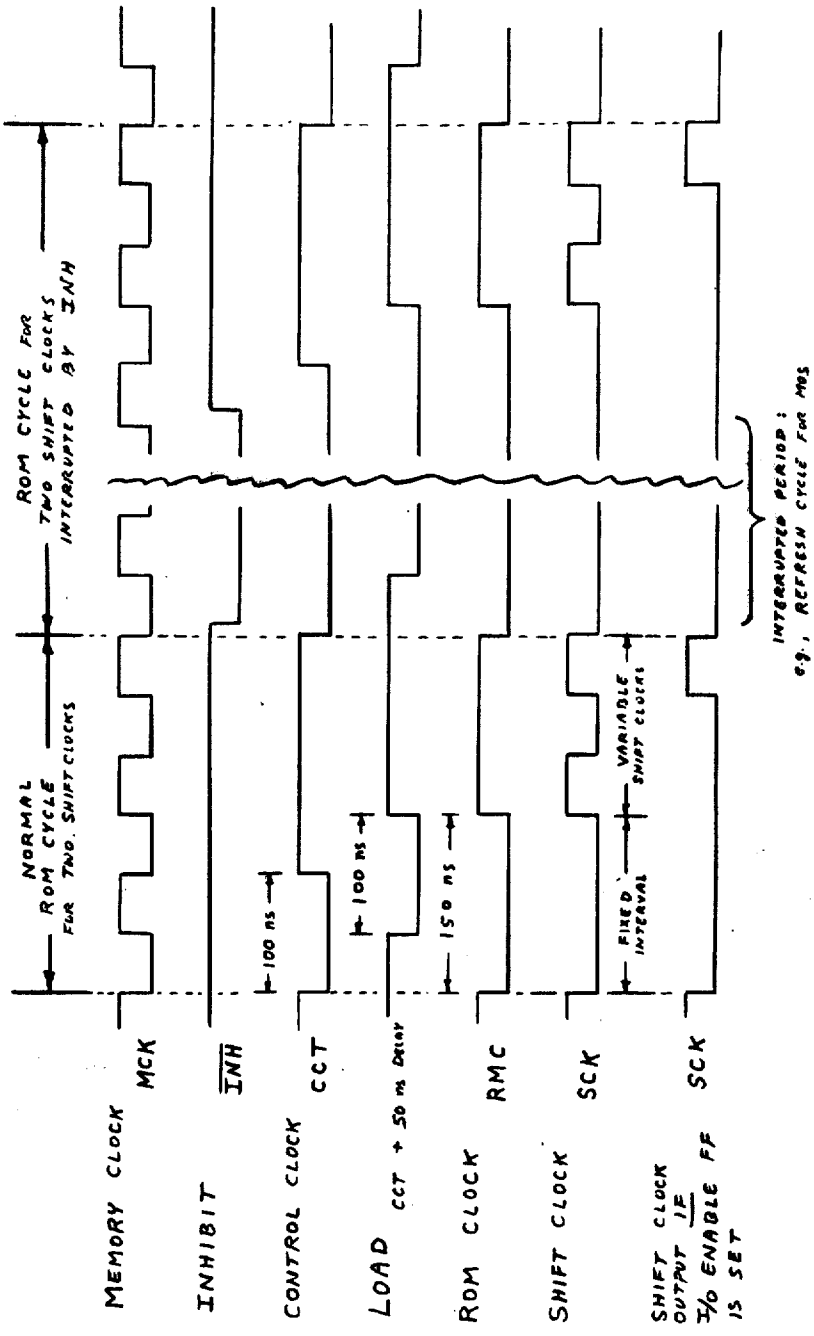
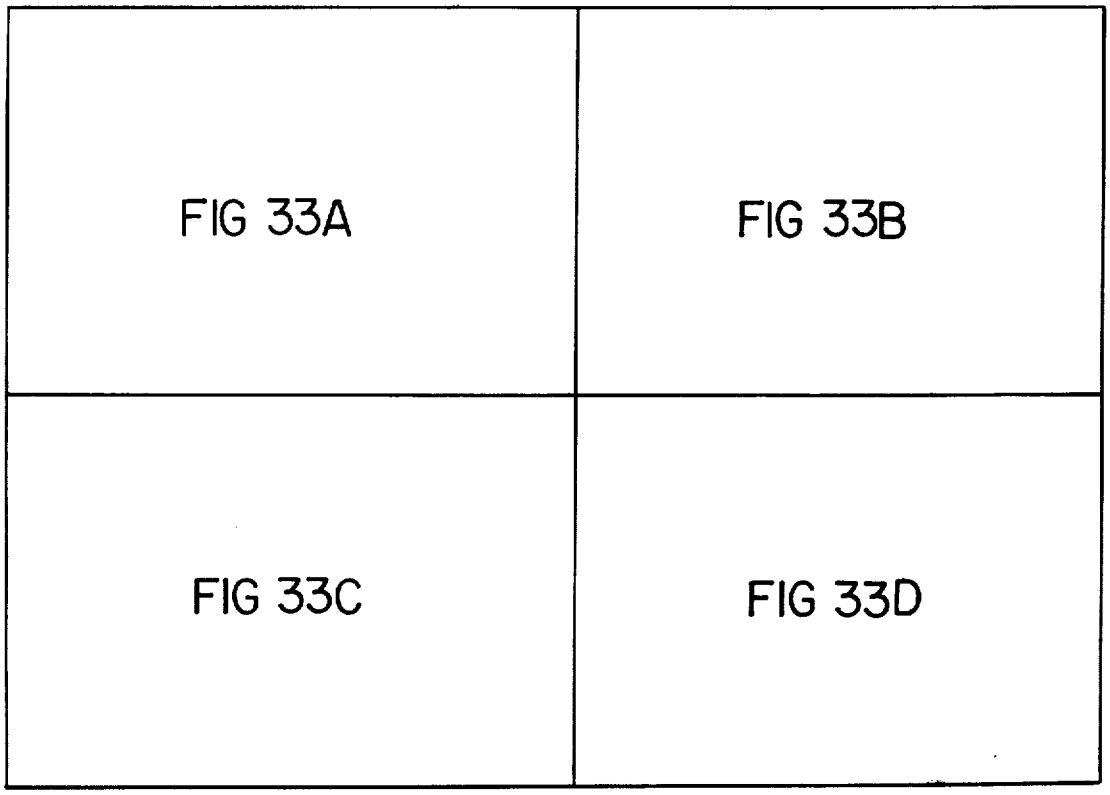


FIG 33'



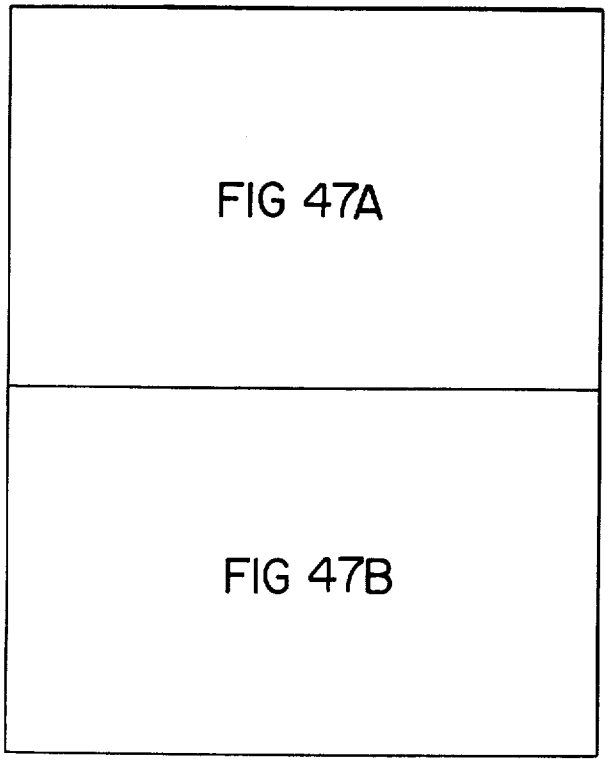


FIG 47'

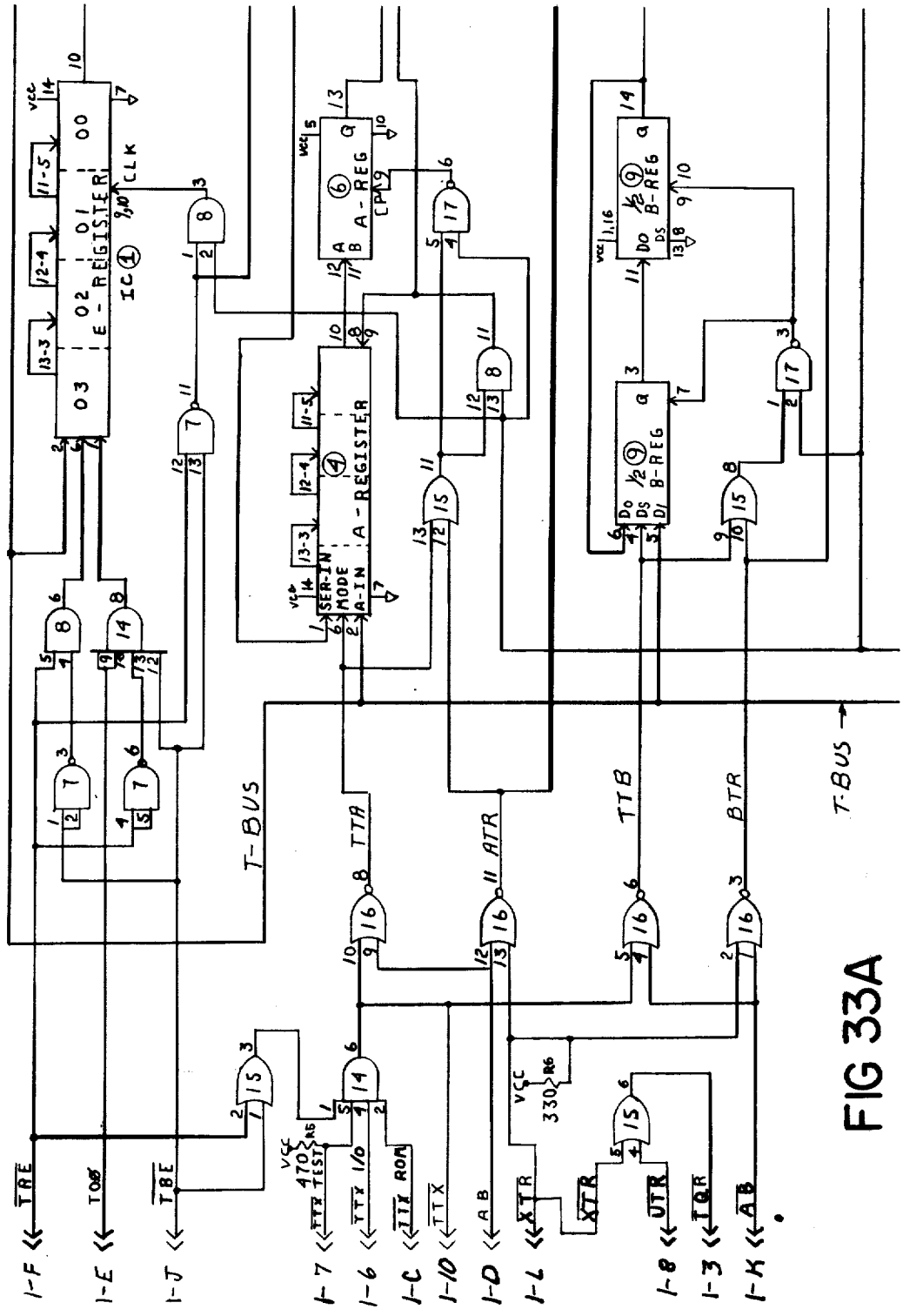
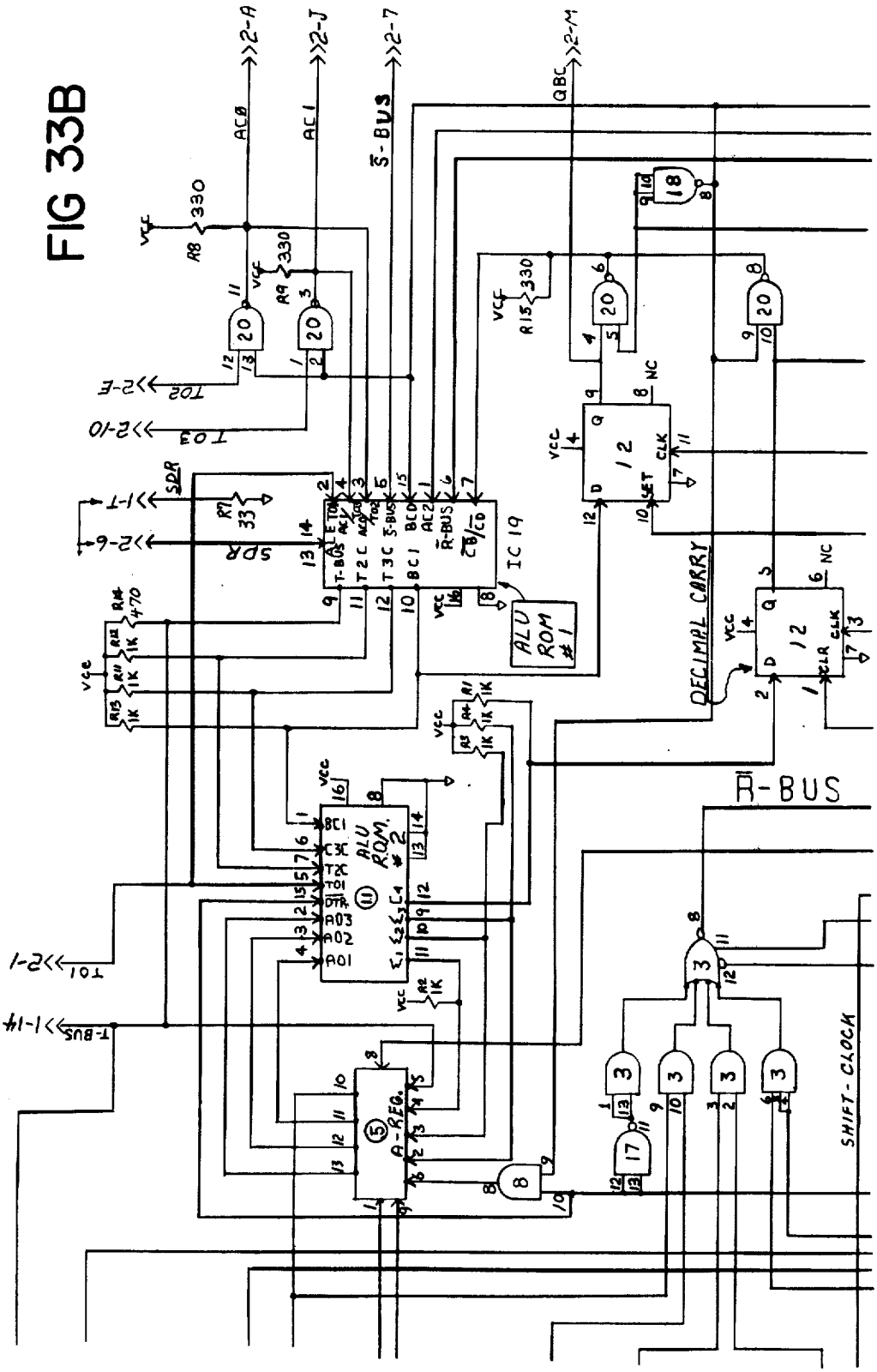
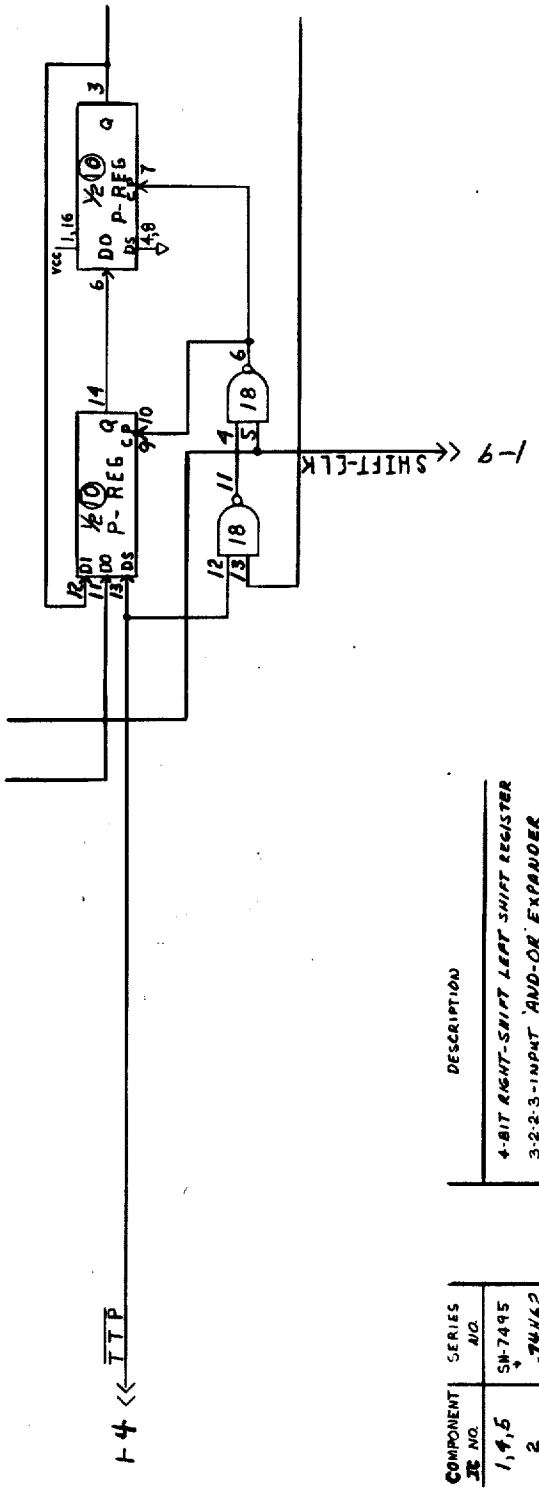

T-BUS
TTB
ATR
BTR
T-BUS
1-F
1-E
1-J
1-7
1-6
1-C
1-10
1-D
1-L
1-8
1-3
1-K
TRE
TOS
TBE
TTX TEST
TTX I/O
TTX ROM
TTX
A B
XTR
XTR
UTR
TBR
AB
VCC
470
330
IC ① E-REGISTER
⑥ A-REG
④ A-REGISTER
⑨ B-REG
⑨ B-REG
SER-IN
MODE
A-IN
CLK

FIG 33A

FIG 33B


S-BUS
ALU ROM #1
ALU ROM #2
A-REG.
DECIMAL CARRY
A-BUS
SHIFT-CLOCK
T-BUS
SDR
IC 19



COMPONENT IC NO.	SERIES NO	DESCRIPTION
1,4,5	SN-7495	4-BIT RIGHT-SHIFT LEFT SHIFT REGISTER
2	-74H62	3-2-2-3-INPUT 'AND-OR' EXPANDER
3	-74H53	EXPANDABLE 2-2-2-3-INPUT 'AND-OR' INVERTED GATE
6	-7491	8 BIT SHIFT REGISTER
7,13,17,18	-74H00	QUAD 2-INPUT 'NAND' GATES
8	MC-3001P	QUAD 2-INPUT 'AND' GATES
9,10	F932859	8-BIT SHIFT REGISTER, DUAL
11		BI-POLAR 'ROM'
12	SN-7474	DUAL, EDGE-TRIGGERED FLIP-FLOPS
14	SN-74H21	DUAL, 4-INPUT 'AND' GATES
15	MC3003P	QUAD, 2-INPUT 'OR' GATES
16	MC3002P	QUAD, 2-INPUT 'NOR' GATES
19		BI-POLAR 'ROM'
20	SN74H01	QUAD 2-INPUT 'NAND' GATE (OPEN COLLECTOR OUTPUT)

FIG 33C

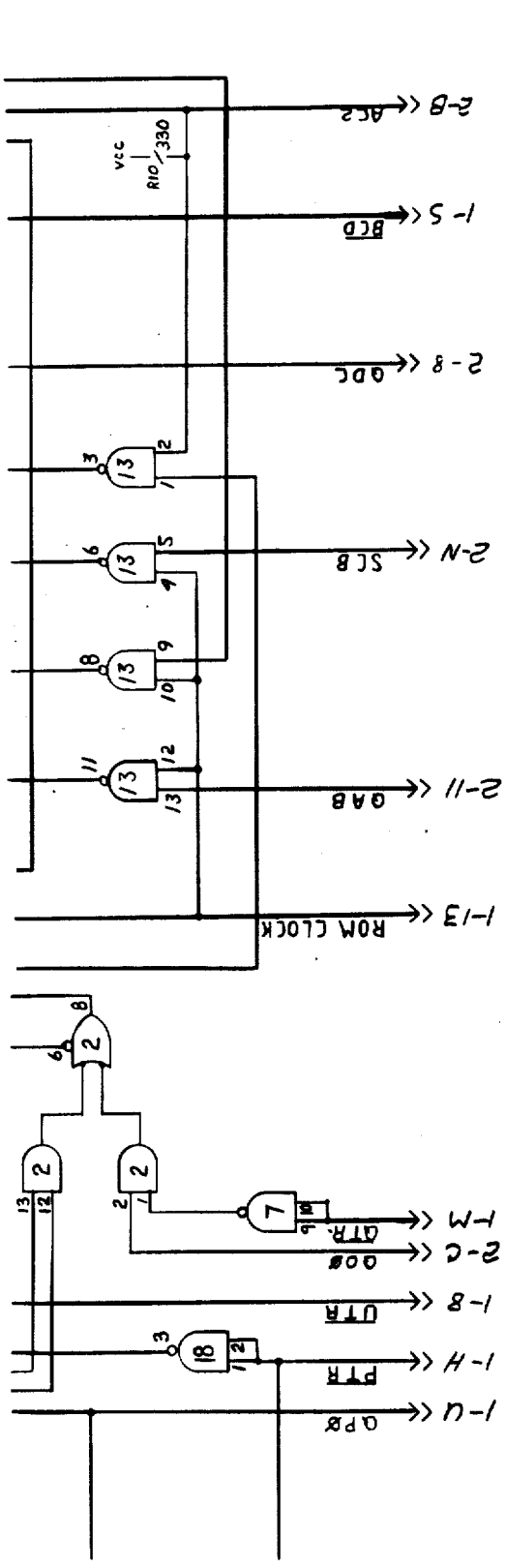

2-B
1-S
2-8
2-N
2-11
1-13
1-M
2-C
1-8
1-H
1-U
ROM CLOCK
VCC
R10
330
BCD
GND
VCC +5V
C10
TAN

FIG 33D

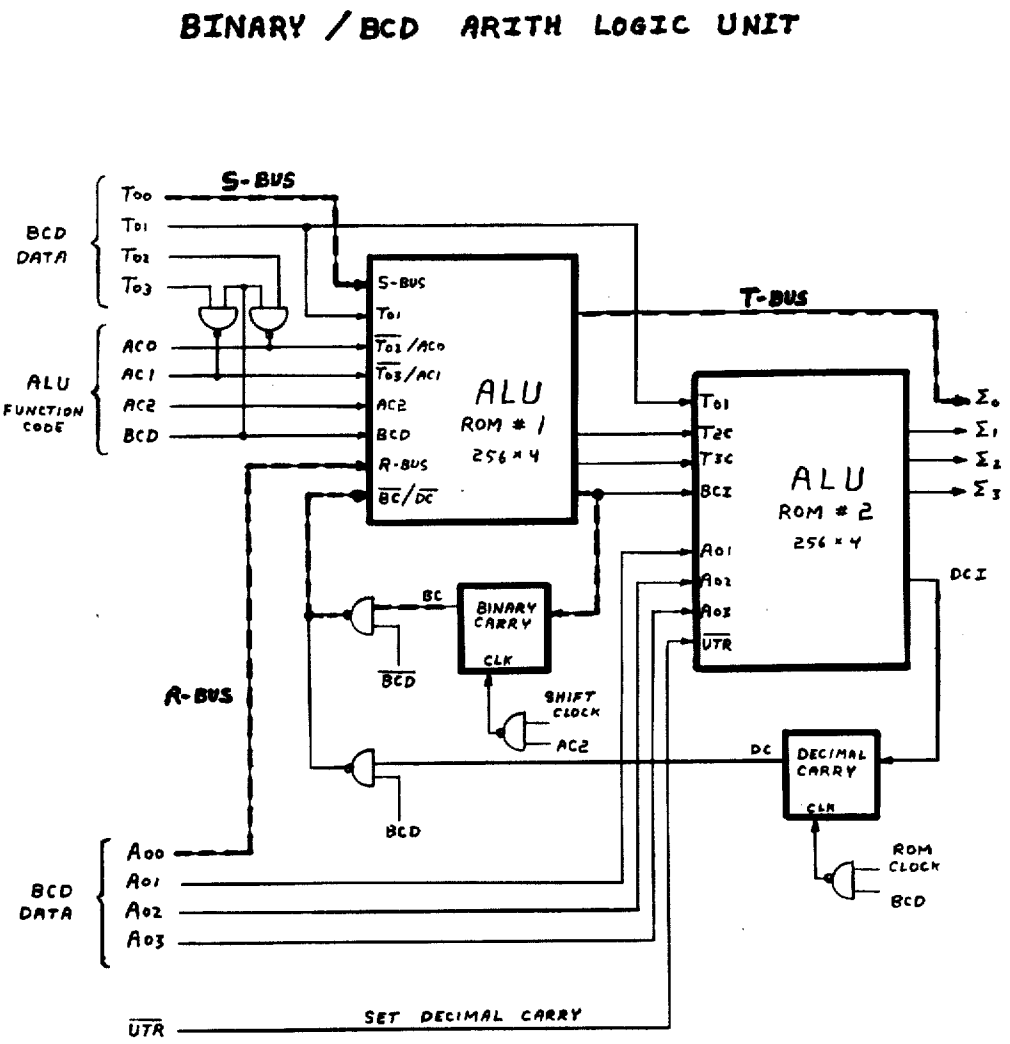

BINARY / BCD ARITH LOGIC UNIT
S-BUS
T-BUS
BCD DATA
ALU FUNCTION CODE
ALU ROM # 1 256 × 4
ALU ROM # 2 256 × 4
BINARY CARRY
DECIMAL CARRY
SHIFT CLOCK
ROM CLOCK
A-BUS
SET DECIMAL CARRY

FIG 34

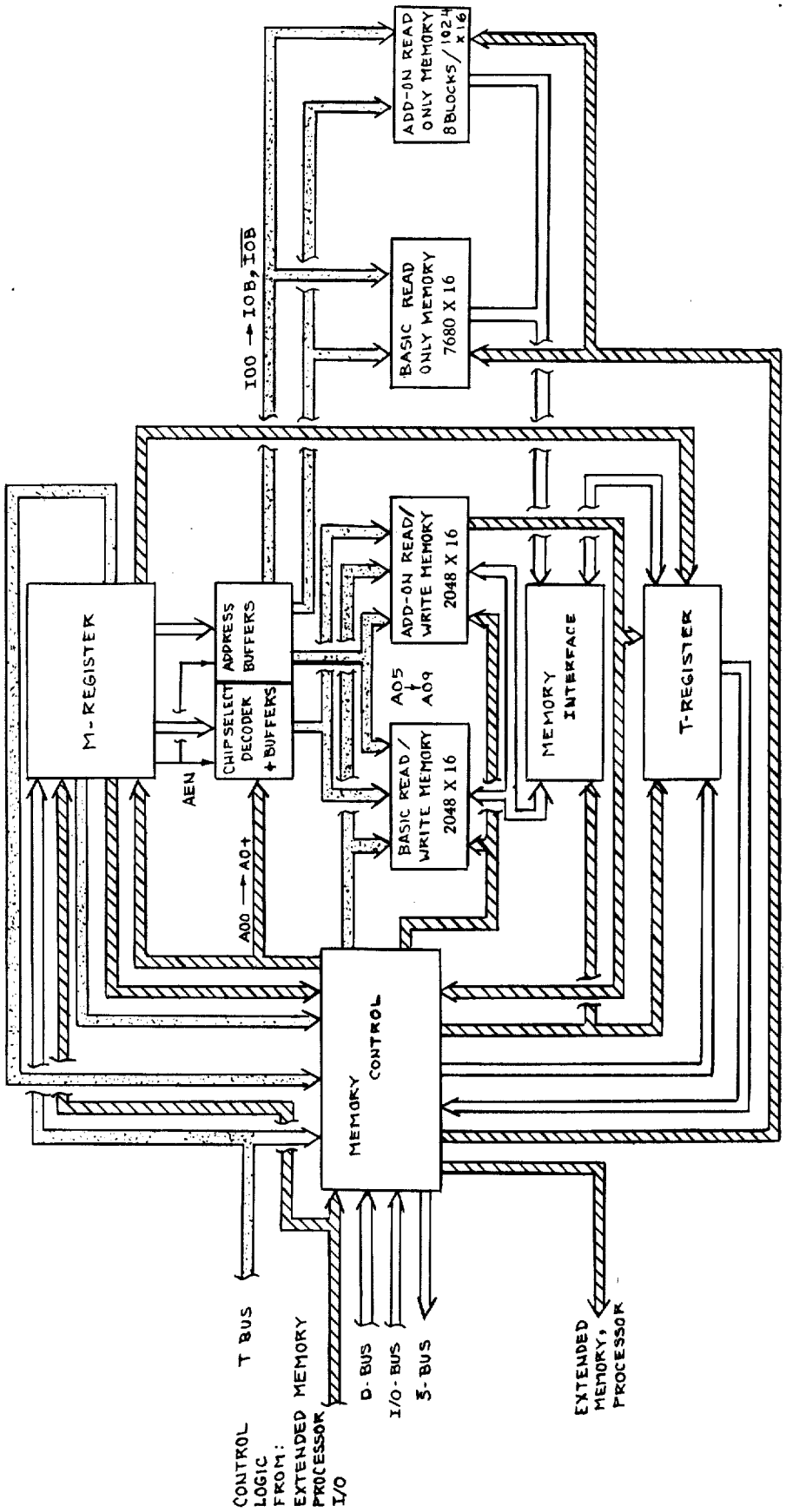

ADD-ON READ ONLY MEMORY 8 BLOCKS / 1024 x 16
BASIC READ ONLY MEMORY 7680 X 16
IOO → IOB, IOB
ADD-ON READ/ WRITE MEMORY 2048 X 16
BASIC READ/ WRITE MEMORY 2048 X 16
A05 → A09
M-REGISTER
ADDRESS BUFFERS
CHIP SELECT DECODER + BUFFERS
AEN
A00 → A04
MEMORY INTERFACE
T-REGISTER
MEMORY CONTROL
T BUS
D-BUS
I/O-BUS
S-BUS
CONTROL LOGIC FROM: EXTENDED PROCESSOR I/O
EXTENDED MEMORY, PROCESSOR
LEGEND:
CONTROL SIGNAL
DATA
ADDRESS INFORMATION

FIG 35

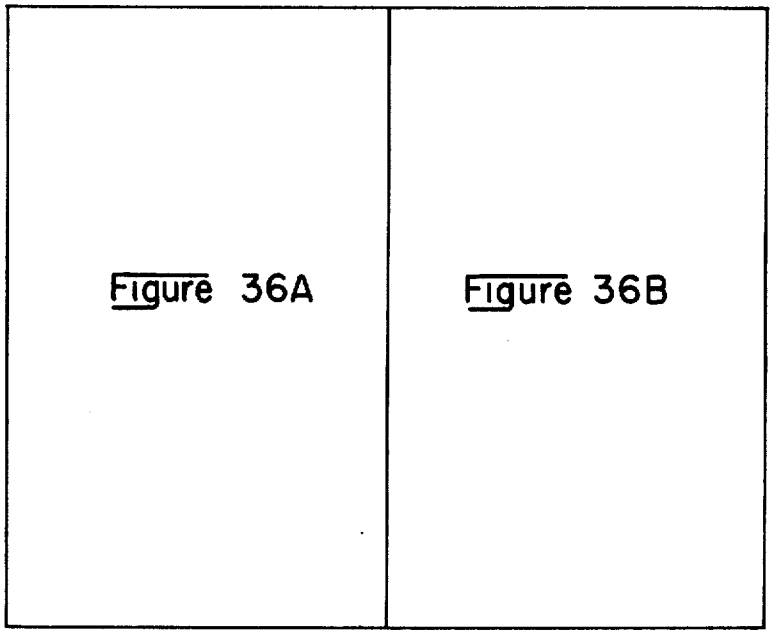
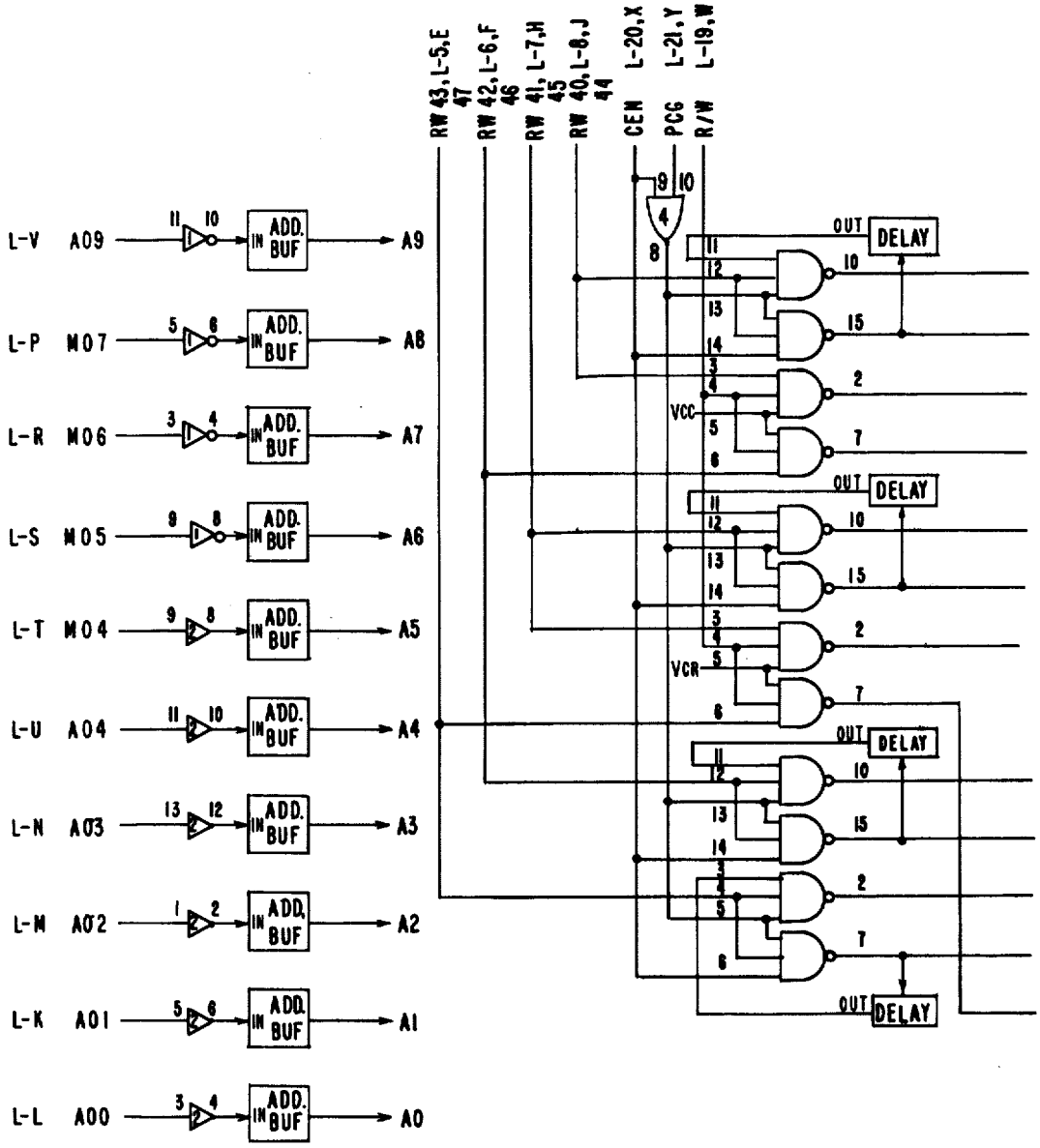


Figure 36'


RW 43, L-5, E
RW 42, L-6, F
RW 41, L-7, H
RW 40, L-8, J
CEN
L-20, X
L-21, Y
L-19, W
PCG
R/W
L-V A09
L-P M07
L-R M06
L-S M05
L-T M04
L-U A04
L-N A03
L-M A02
L-K A01
L-L A00
ADD. BUF
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
DELAY
OUT
VCC
VCR

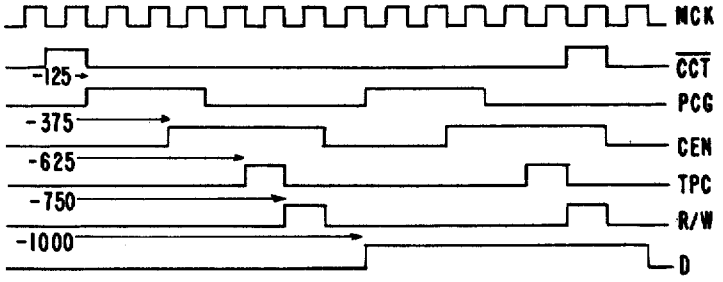
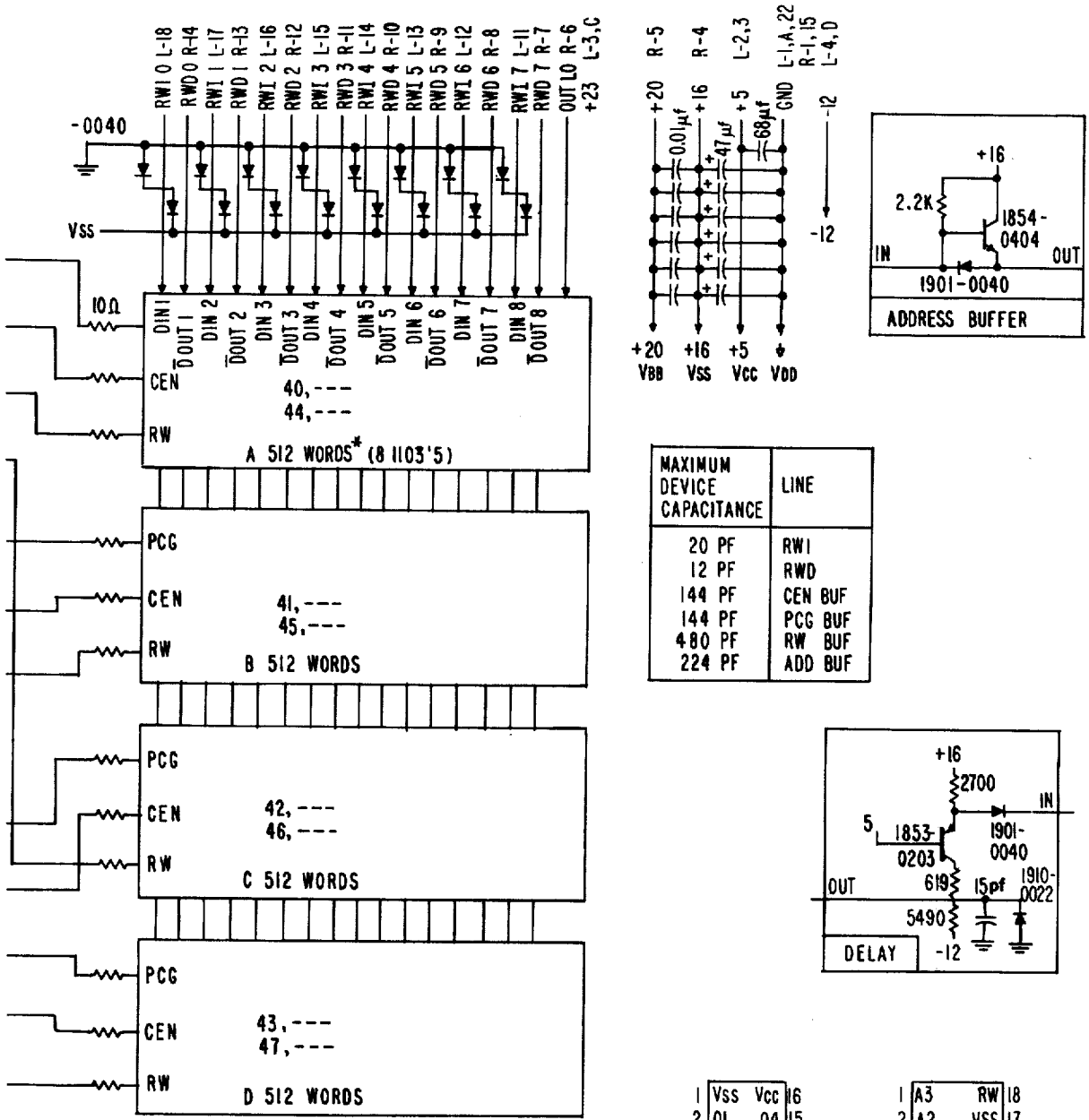

MCK
-125
CCT
PCG
-375
CEN
-625
TPC
-750
R/W
-1000
D

Figure 36A



MAXIMUM DEVICE CAPACITANCE	LINE
20 PF	RWI
12 PF	RWD
144 PF	CEN BUF
144 PF	PCG BUF
480 PF	RW BUF
224 PF	ADD BUF

*RW, PCG, CEN ARE CONNECTED IN COMMON TO ALL 8 PACKAGES. DIN 1 IS CONNECTED TO DIN OF PACKAGE 1 IN BLOCKS A, B, C, & D.

1	Vss	Vcc	16
2	O1	O4	15
3	DI	D4	14
4	E1	E4	13
5	E2	E3	12
6	D2	D3	11
7	O2	O3	10
8	GND	VBB	9

I3207

1	A3	RW	18
2	A2	VSS	17
3	A1	CEN	16
4	A0	A4	15
5	PCG	DOUT	14
6	A9	A8	13
7	A6	DIN	12
8	A5	VDD	11
9	A7	VBB	10

INTEL 1103

Figure 36B

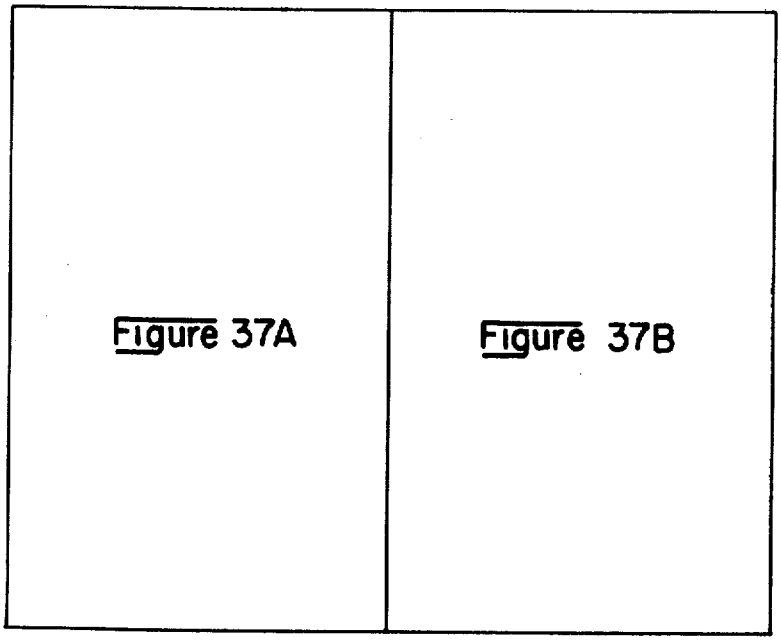
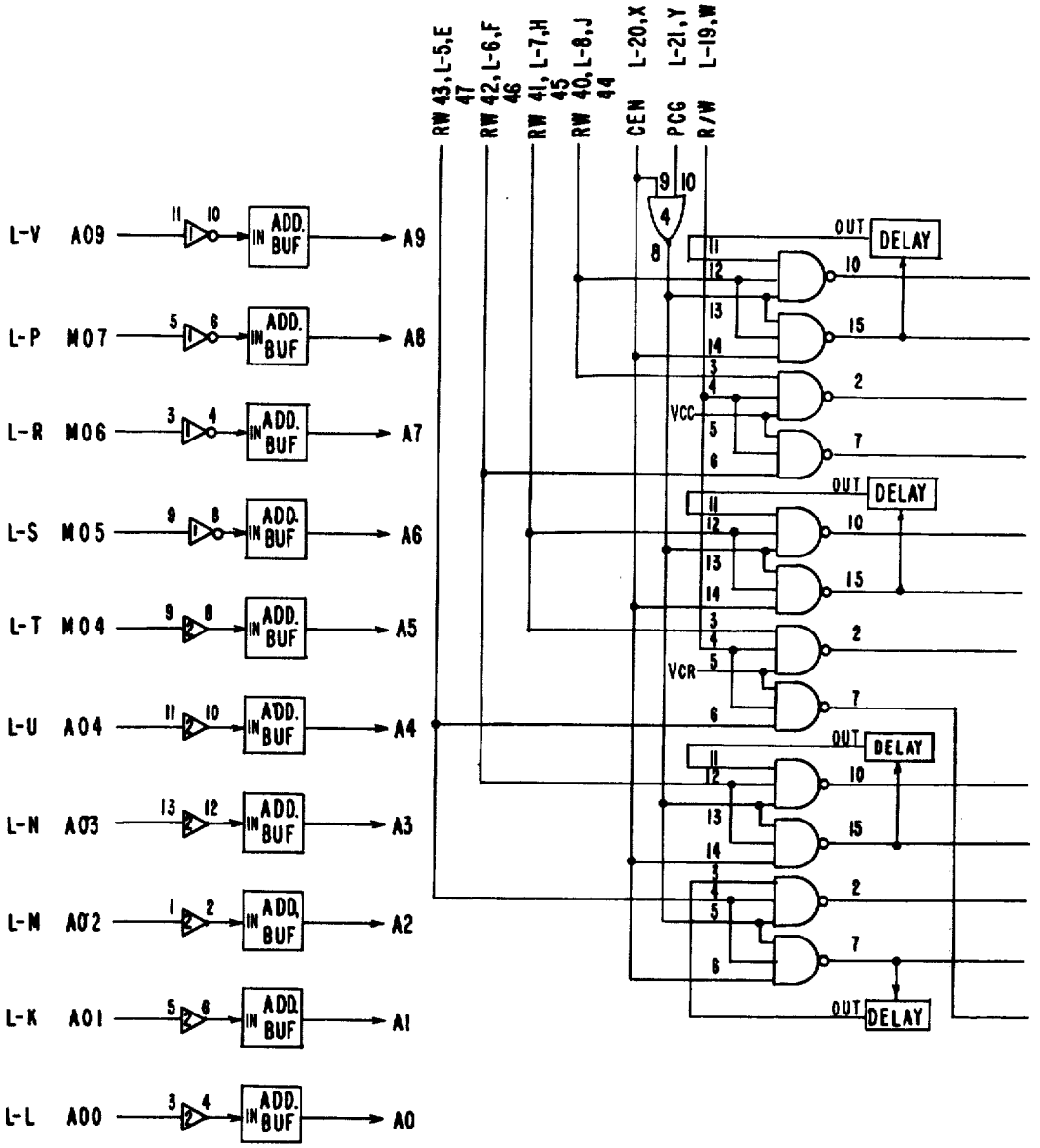


Figure 37'


RW 43, L-5, E
47
RW 42, L-6, F
46
RW 41, L-7, H
45
RW 40, L-8, J
44
CEN
L-20, X
PCG
L-21, Y
R/W
L-19, W
L-V A09
L-P M07
L-R M06
L-S M05
L-T M04
L-U A04
L-N A03
L-M A02
L-K A01
L-L A00
ADD. BUF
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
OUT
DELAY
VCC
VCR

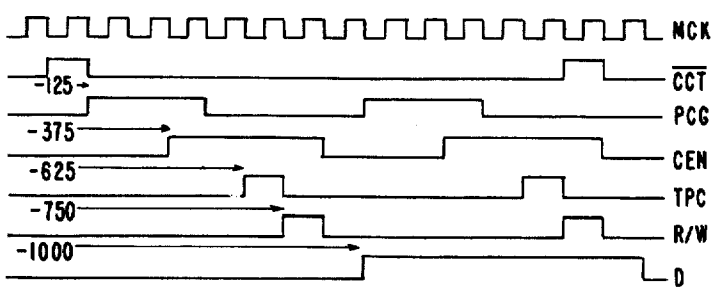
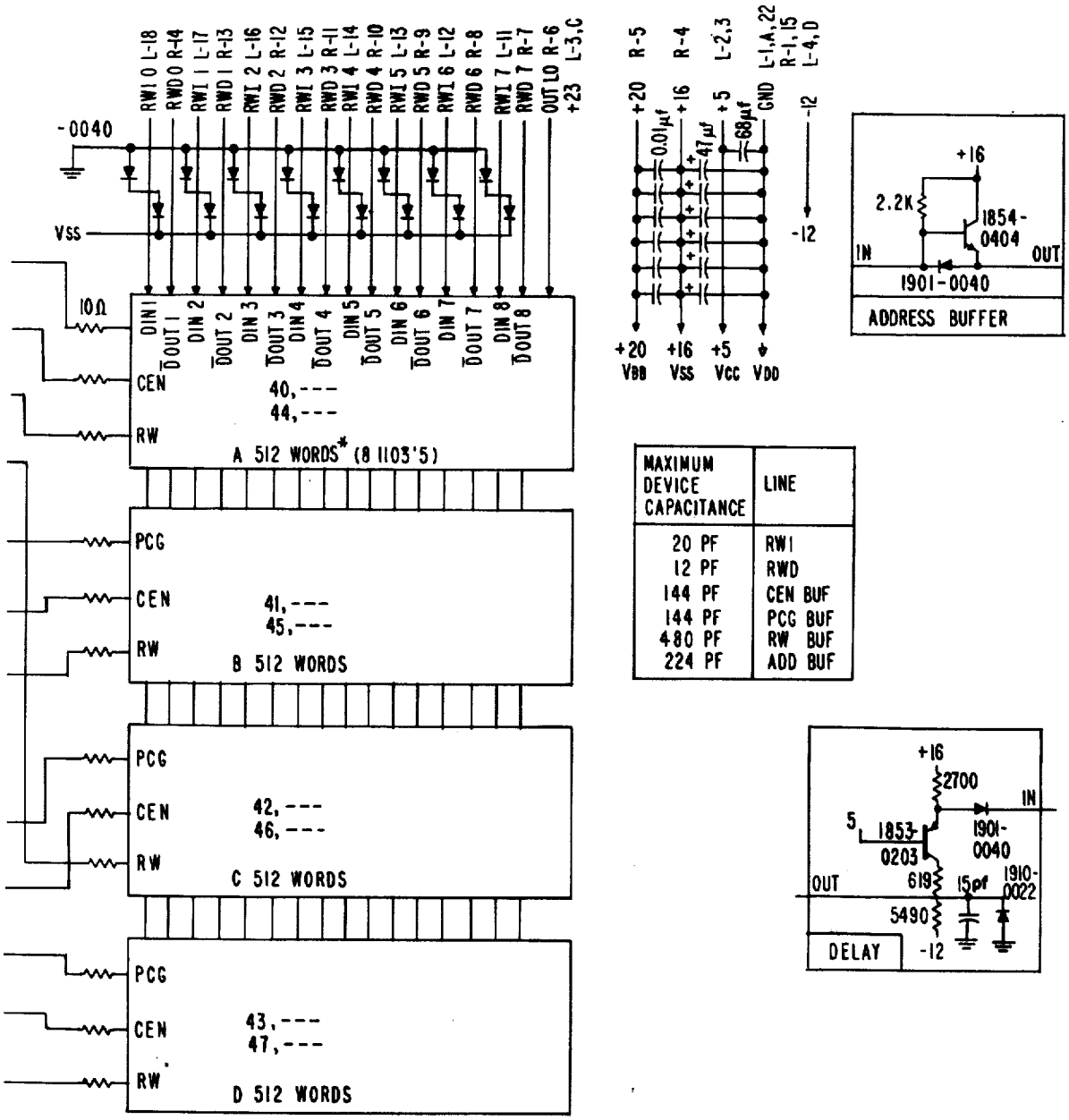

MCK
-125
CCT
PCG
-375
CEN
-625
TPC
-750
R/W
-1000
D

Figure 37A



MAXIMUM DEVICE CAPACITANCE	LINE
20 PF	RWI
12 PF	RWD
144 PF	CEN BUF
144 PF	PCG BUF
480 PF	RW BUF
224 PF	ADD BUF

*RW, PCG, CEN ARE CONNECTED IN COMMON TO ALL 8 PACKAGES. DIN 1 IS CONNECTED TO DIN OF PACKAGE 1 IN BLOCKS A, B, C, & D.

Figure 37B

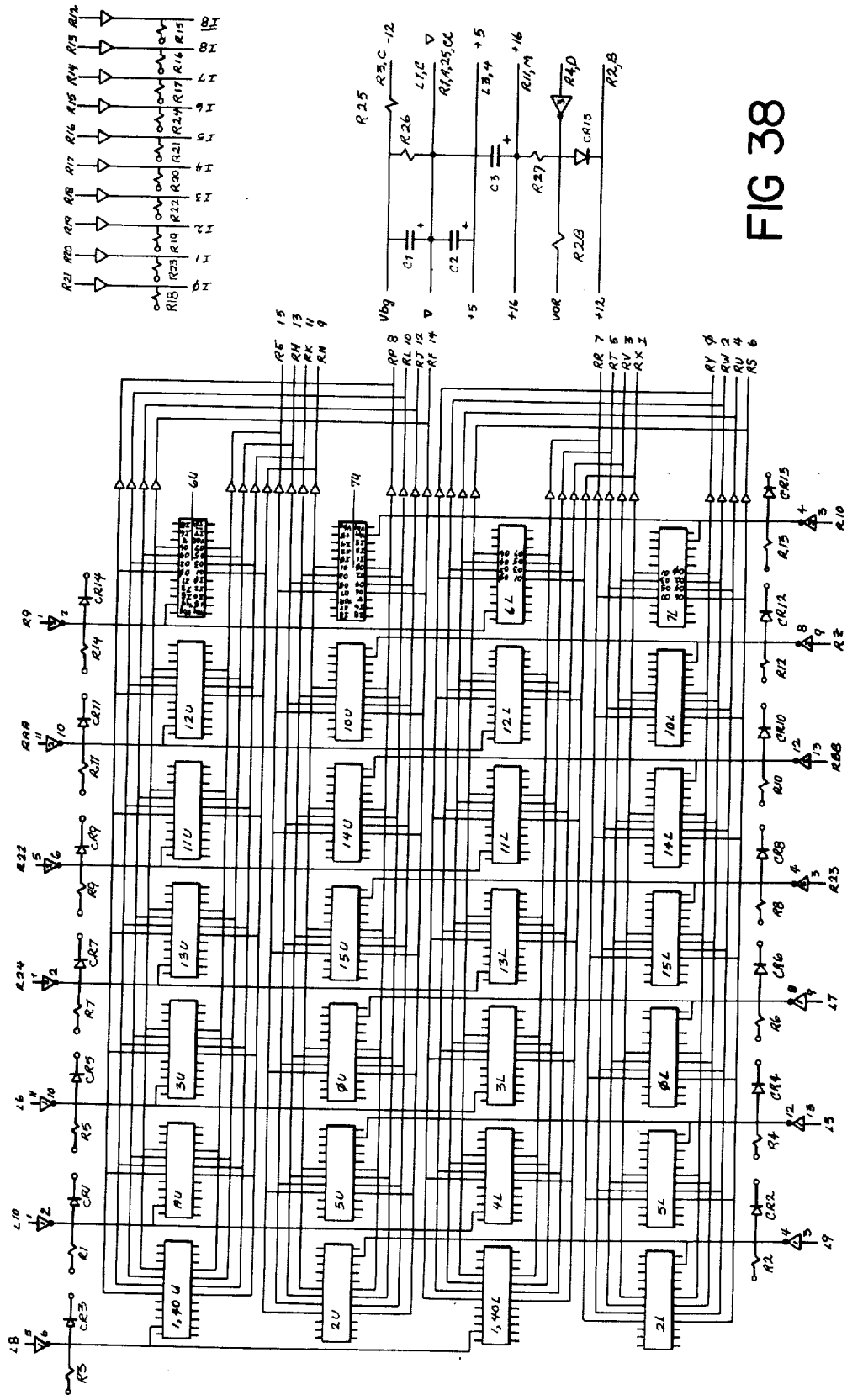


FIG 38

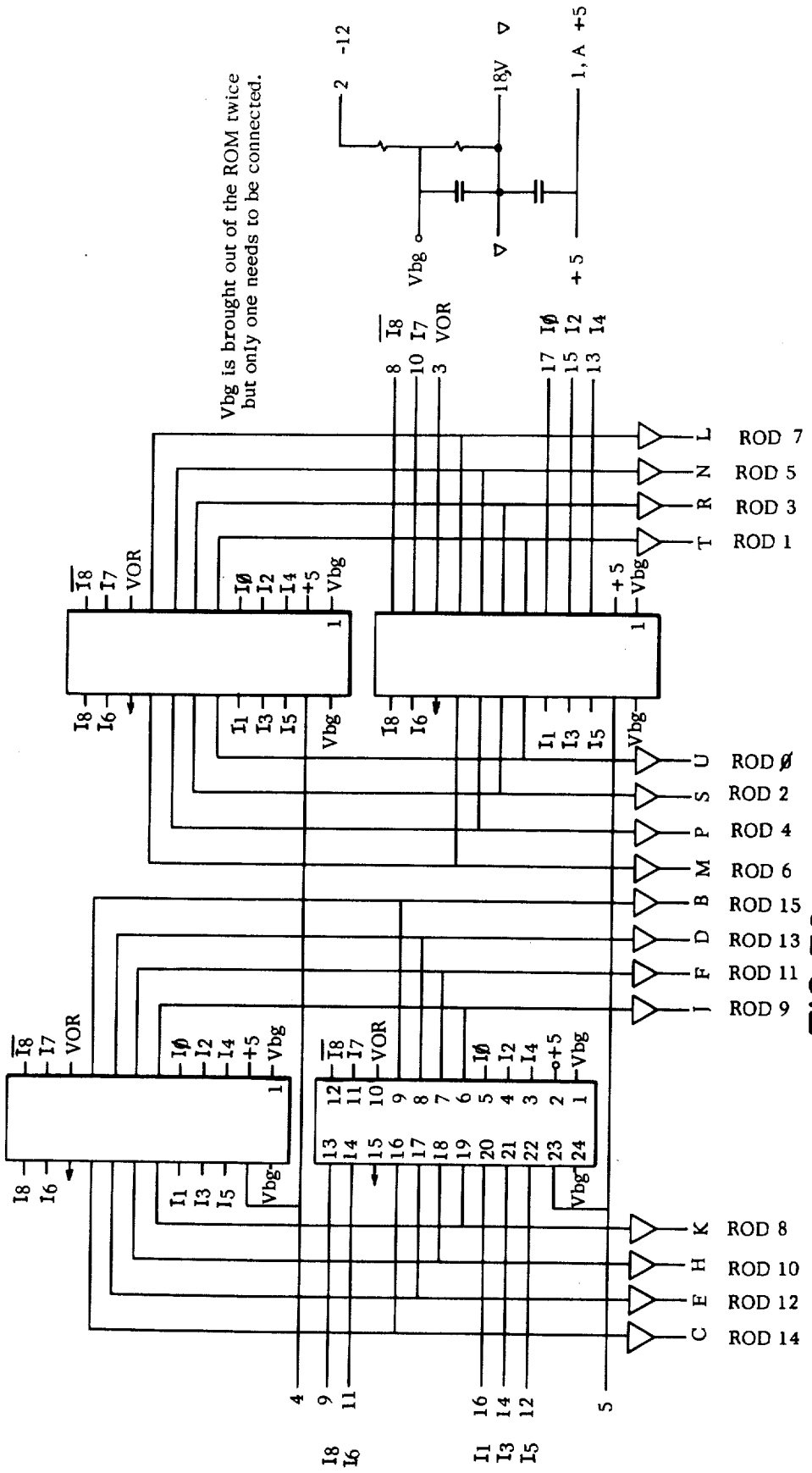

Vbg is brought out of the ROM twice but only one needs to be connected.
2 -12
18V
1, A +5
Vbg
+ 5
ROD 7
ROD 5
ROD 3
ROD 1
ROD Ø
ROD 2
ROD 4
ROD 6
ROD 15
ROD 13
ROD 11
ROD 9
ROD 8
ROD 10
ROD 12
ROD 14

FIG 39

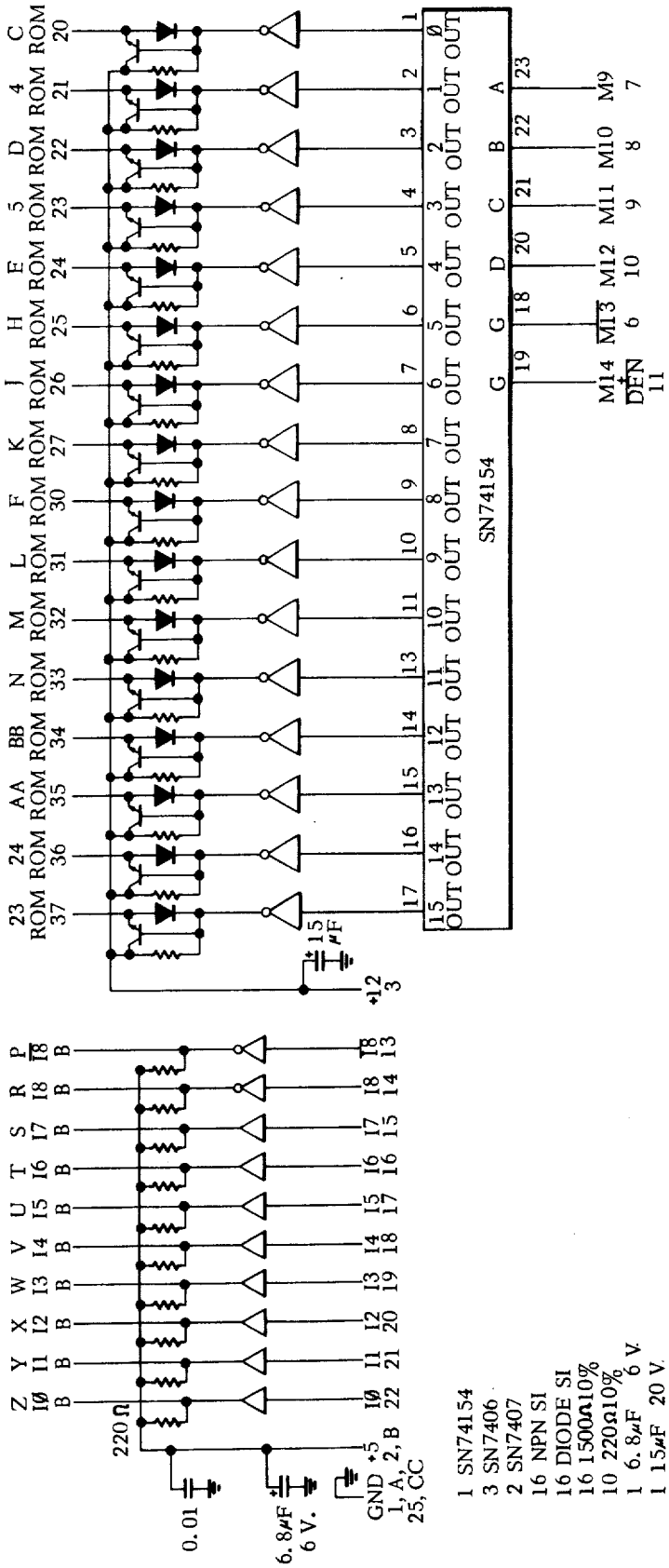

SN74154
1 SN74154
3 SN7406
2 SN7407
16 NPN SI
16 DIODE SI
16 1500Ω10%
10 220Ω10%
1 6.8μF 6 V.
1 15μF 20 V.
1 0.01 μ F

FIG 40

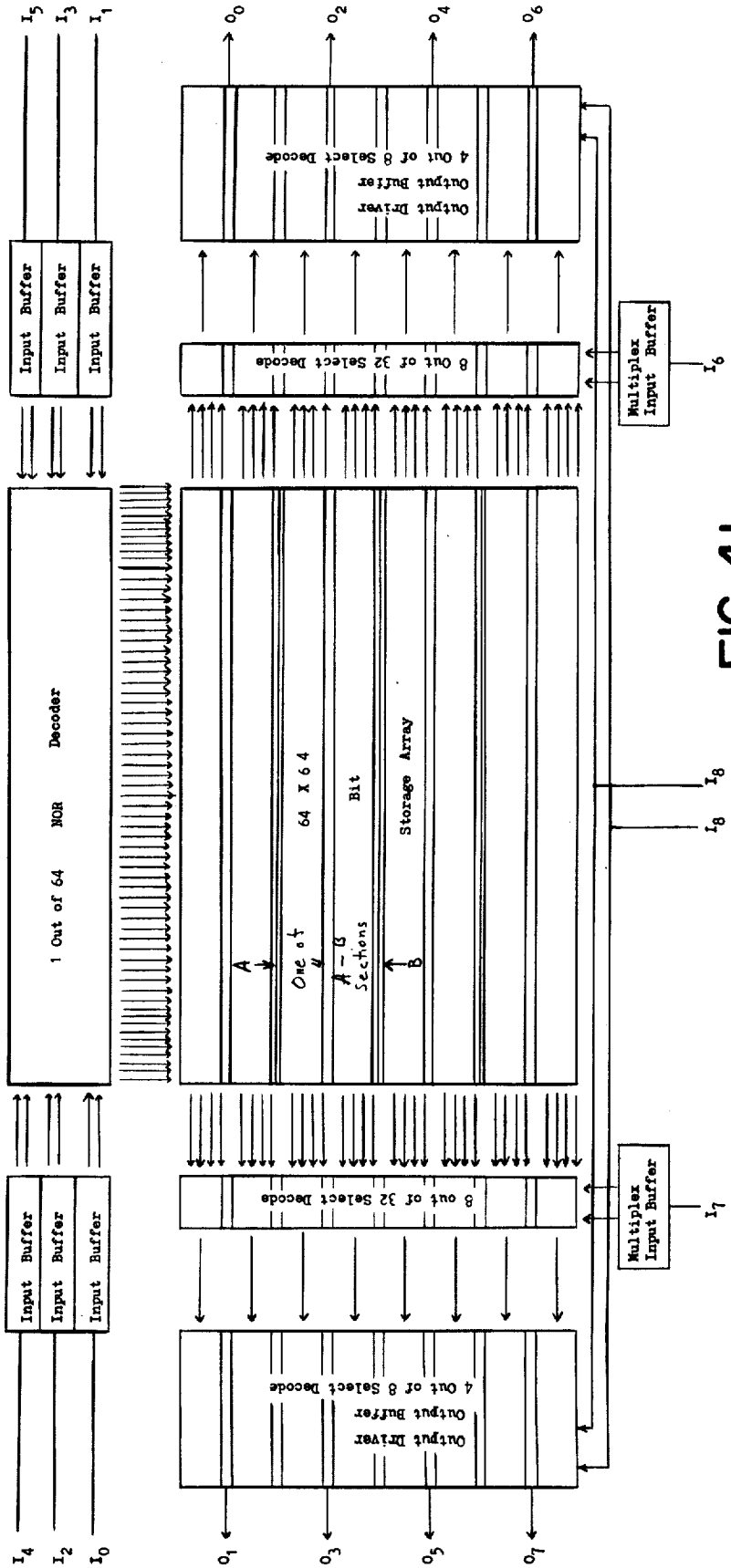

I5
I3
I1
O0
O2
O4
O6
Input Buffer
Input Buffer
Input Buffer
Output Driver
Output Buffer
4 Out of 8 Select Decode
8 Out of 32 Select Decode
Multiplex Input Buffer
I6
1 Out of 64 NOR Decoder
A
One of 64 A-G Sections
B
64 X 64 Bit Storage Array
I8
I8
8 Out of 32 Select Decode
Multiplex Input Buffer
I7
Output Driver
Output Buffer
4 Out of 8 Select Decode
I4
I2
I0
O1
O3
O5
O7

FIG 41

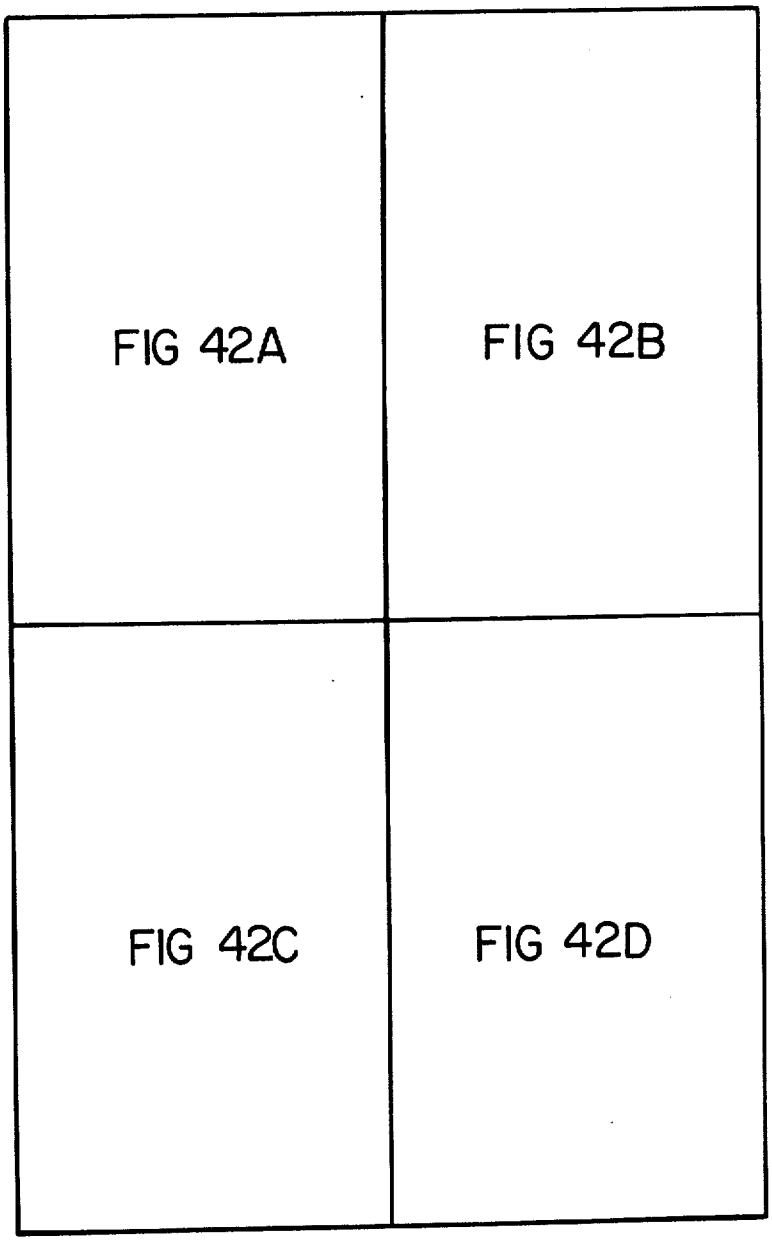


FIG 42'

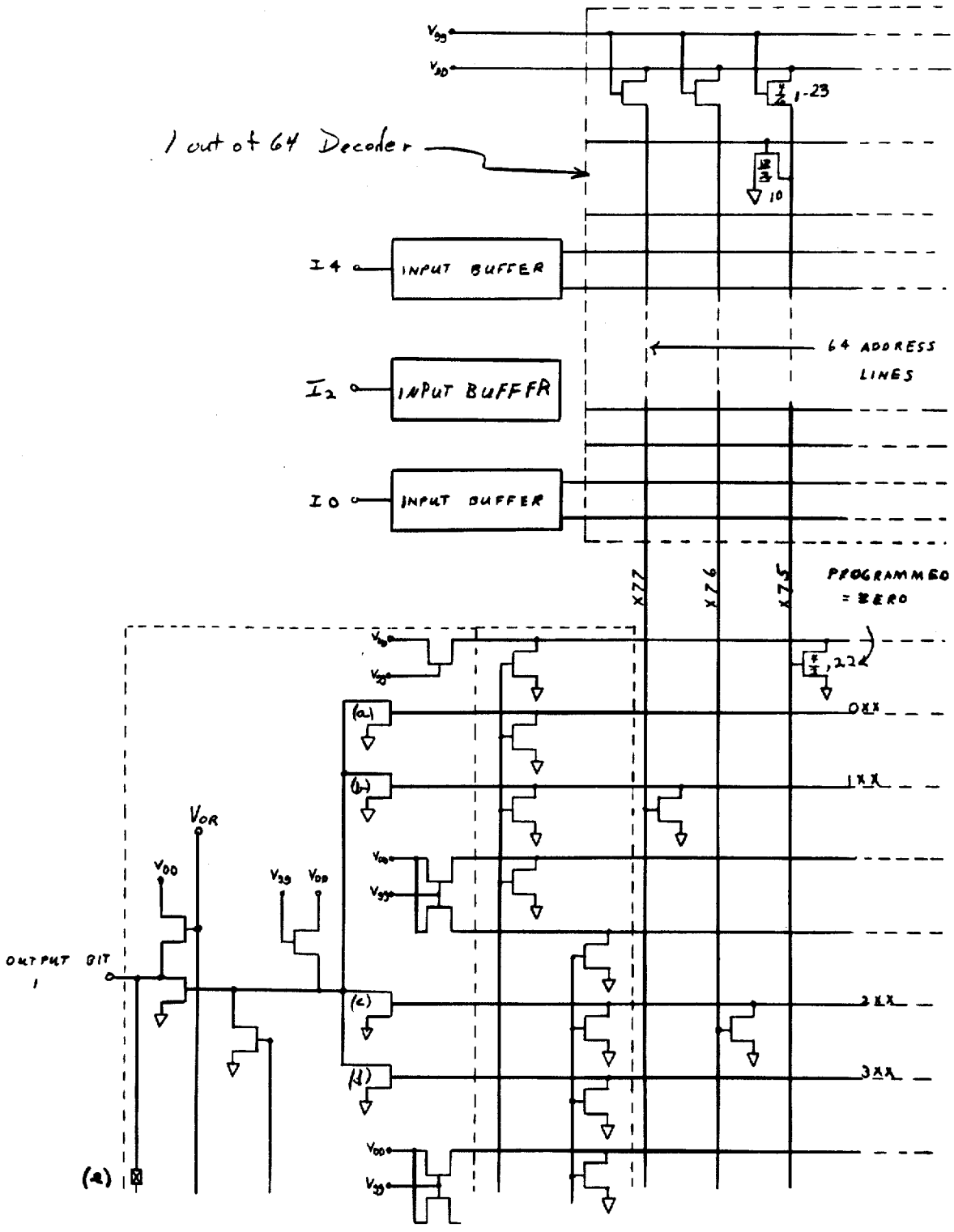

1 out of 64 Decoder
I4
INPUT BUFFER
I2
INPUT BUFFFR
I0
INPUT BUFFER
64 ADDRESS LINES
X77
X76
X75
PROGRAMMED = ZERO
OUTPUT BIT 1
VOR
VDD
VSS
(a)
(b)
(c)
(d)
(e)

FIG 42A

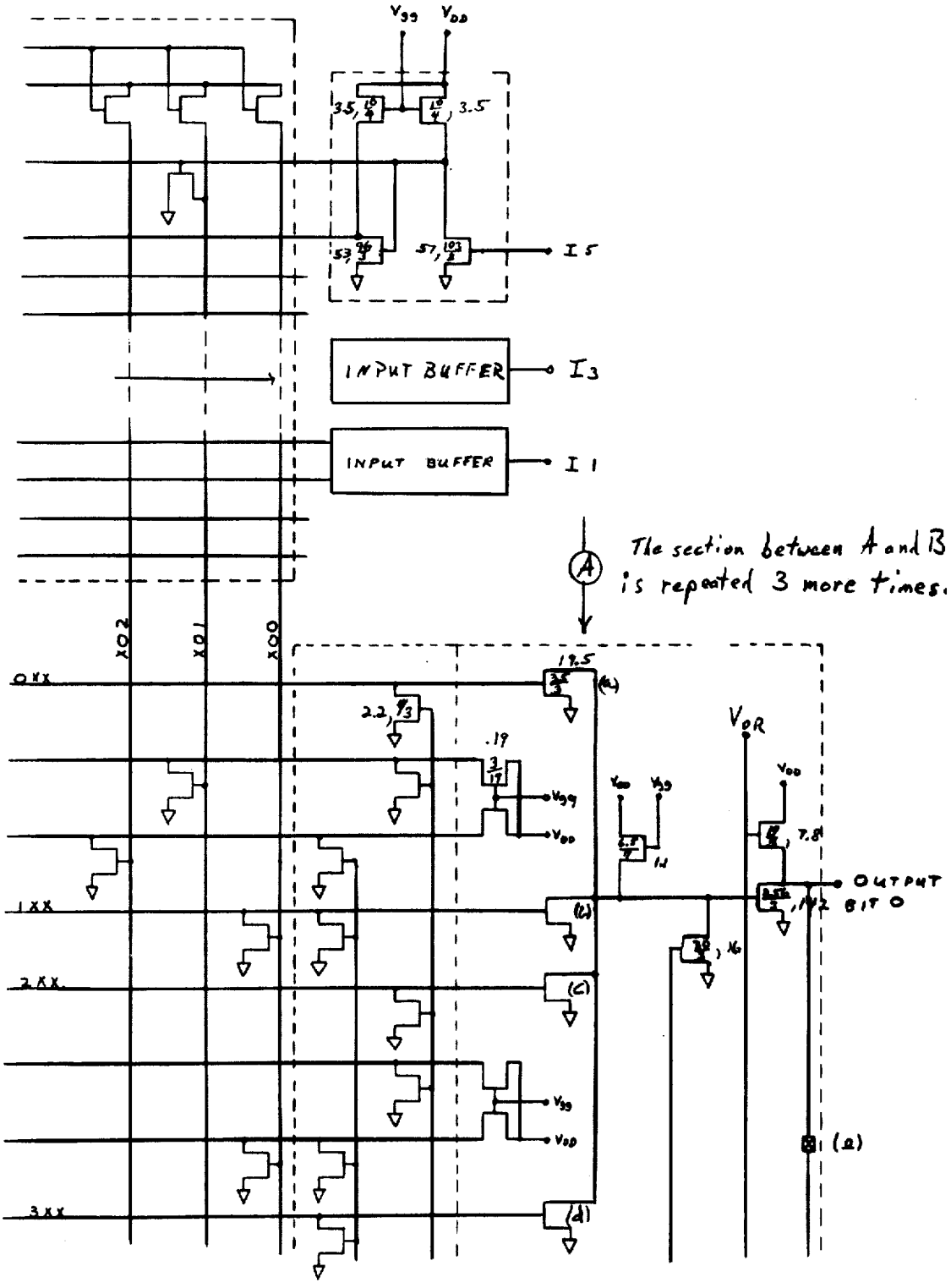
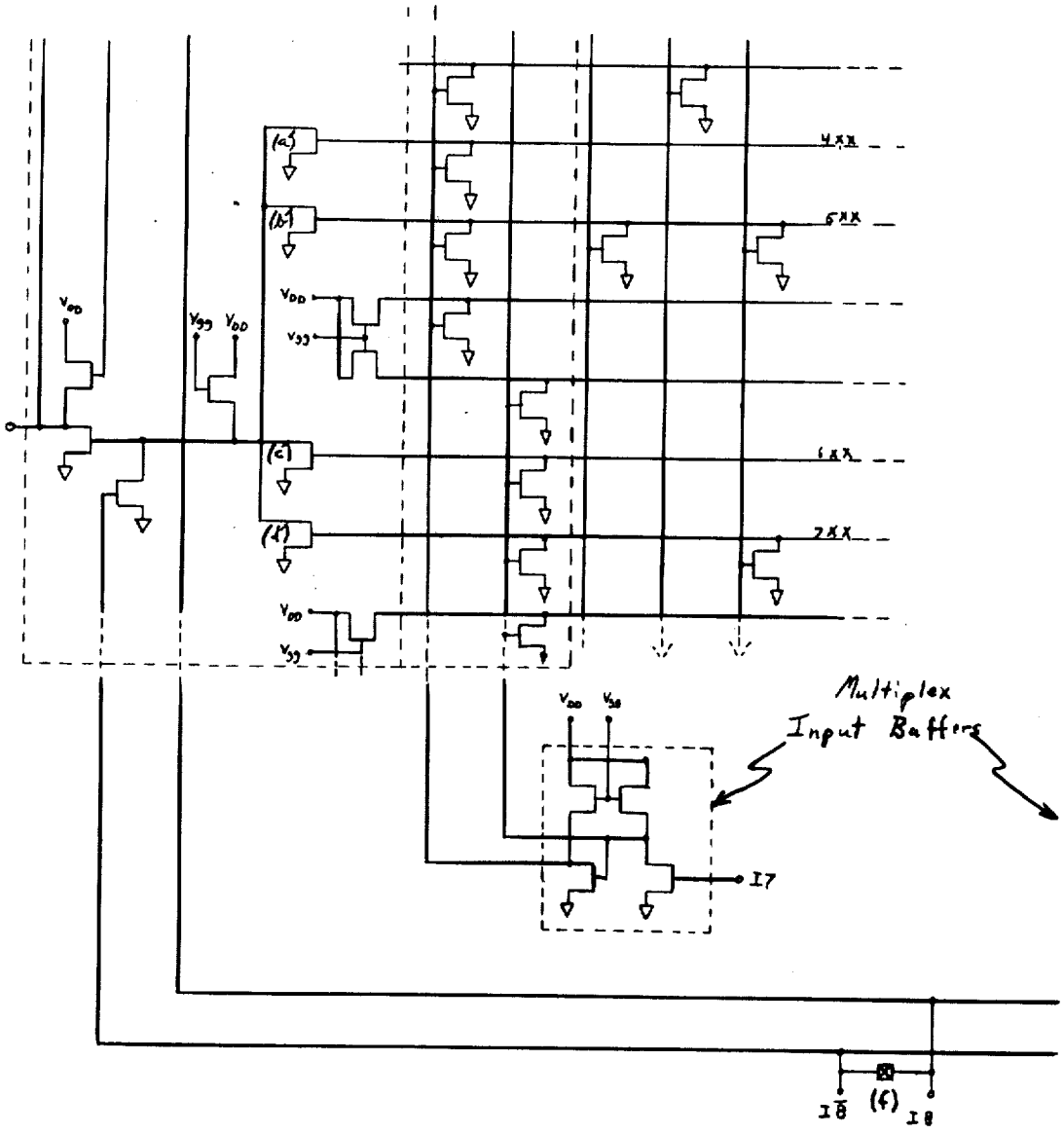

INPUT BUFFER
INPUT BUFFER
I 5
I3
I 1
The section between A and B is repeated 3 more times.
A
B
X02
X01
X00
0XX
1XX
2XX
3XX
OUTPUT BIT 0
(a)
(b)
(c)
(d)

FIG 42B



NOTES →

This symbol indicates connections which can be programmed open or short. ⊠

① FOR A 512 X8 :
- ⓐ TIE OUTPUT PAIRS TOGETHER
- ⓑ DO NOT TIE I8 AND $I\bar{8}$ TOGETHER

② FOR A 256 X16 :
- ⓐ DO NOT TIE OUTPUT PAIRS TOGETHER
- ⓑ TIE I8 AND $I\bar{8}$ TOGETHER

③ V_{DD} = +5V V_{SS} = +12V (switched)
(POSITIVE TRUE LOGIC)

④ RESISTANCES ASSUME 10 Ω □
ALL W/L's ASSUME A 3 μ TOTAL SIDE DIFFUSION
ALL CONTACTS SHOWN AS (■) ARE PROGRAMMABLE WITH GATE MASK

FIG 42C

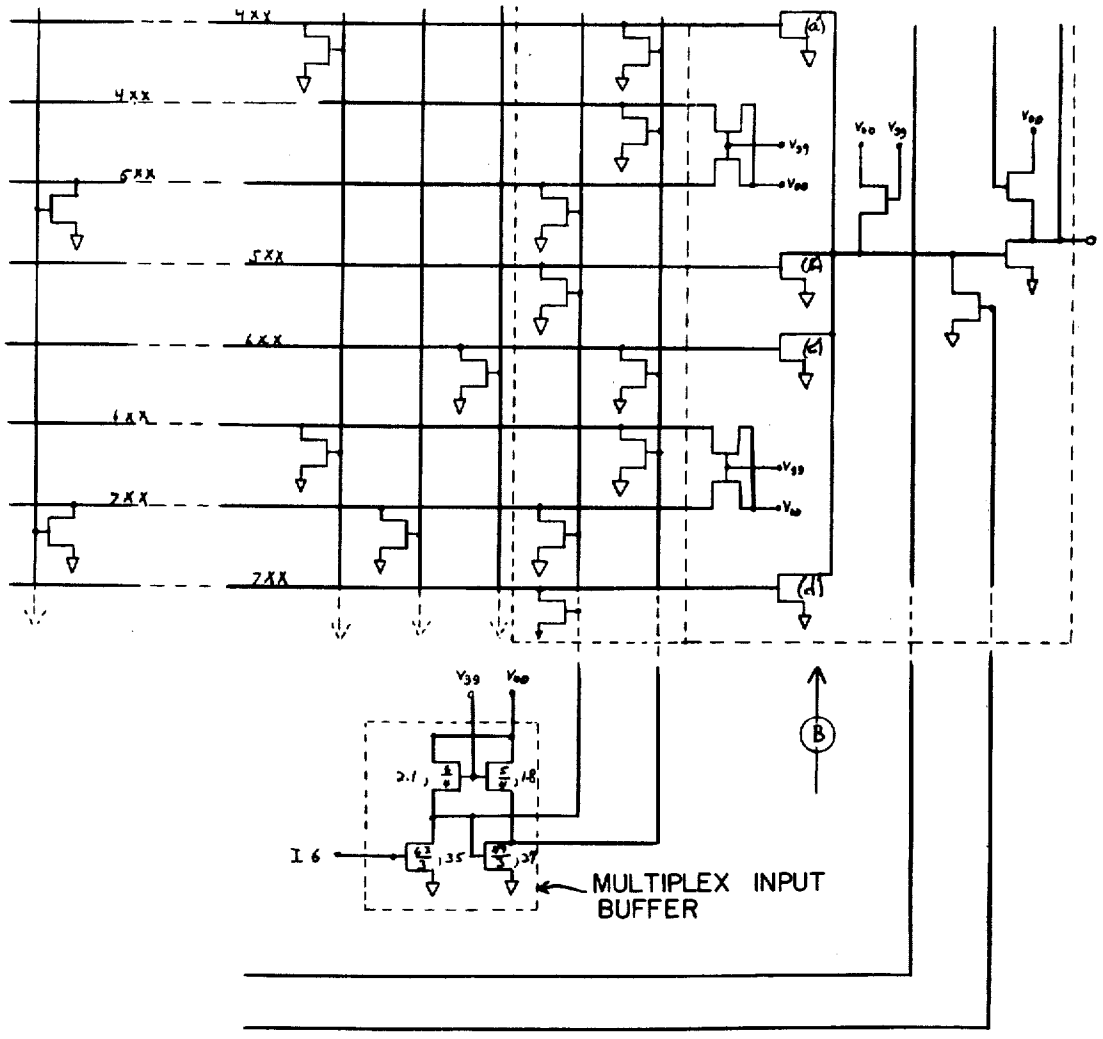


FIG 42D

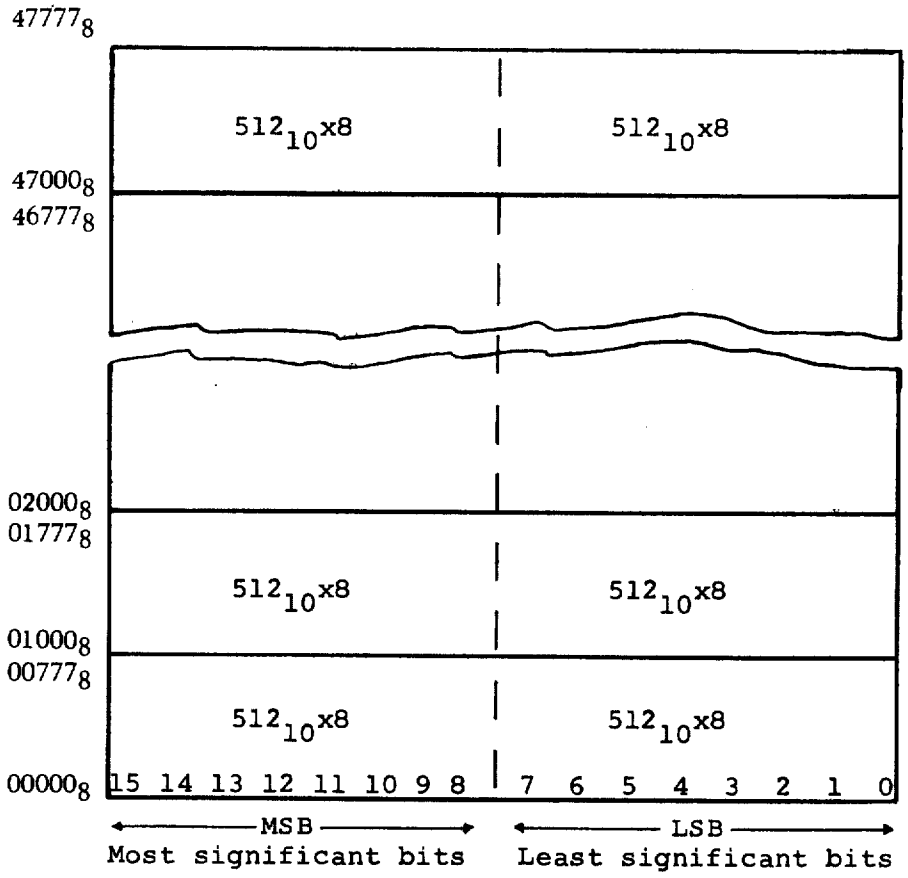

47777_8
512_{10}x8
512_{10}x8
47000_8
46777_8
02000_8
01777_8
512_{10}x8
512_{10}x8
01000_8
00777_8
512_{10}x8
512_{10}x8
00000_8
15 14 13 12 11 10 9 8
7 6 5 4 3 2 1 0
MSB
LSB
Most significant bits
Least significant bits

FIG 43

BIT NUMBERS	ACTUAL BIT
1	15 or 7
2	14 or 6
3	13 or 5
4	12 or 4
5	11 or 3
6	10 or 2
7	9 or 1
8	8 or 0

FIG 45

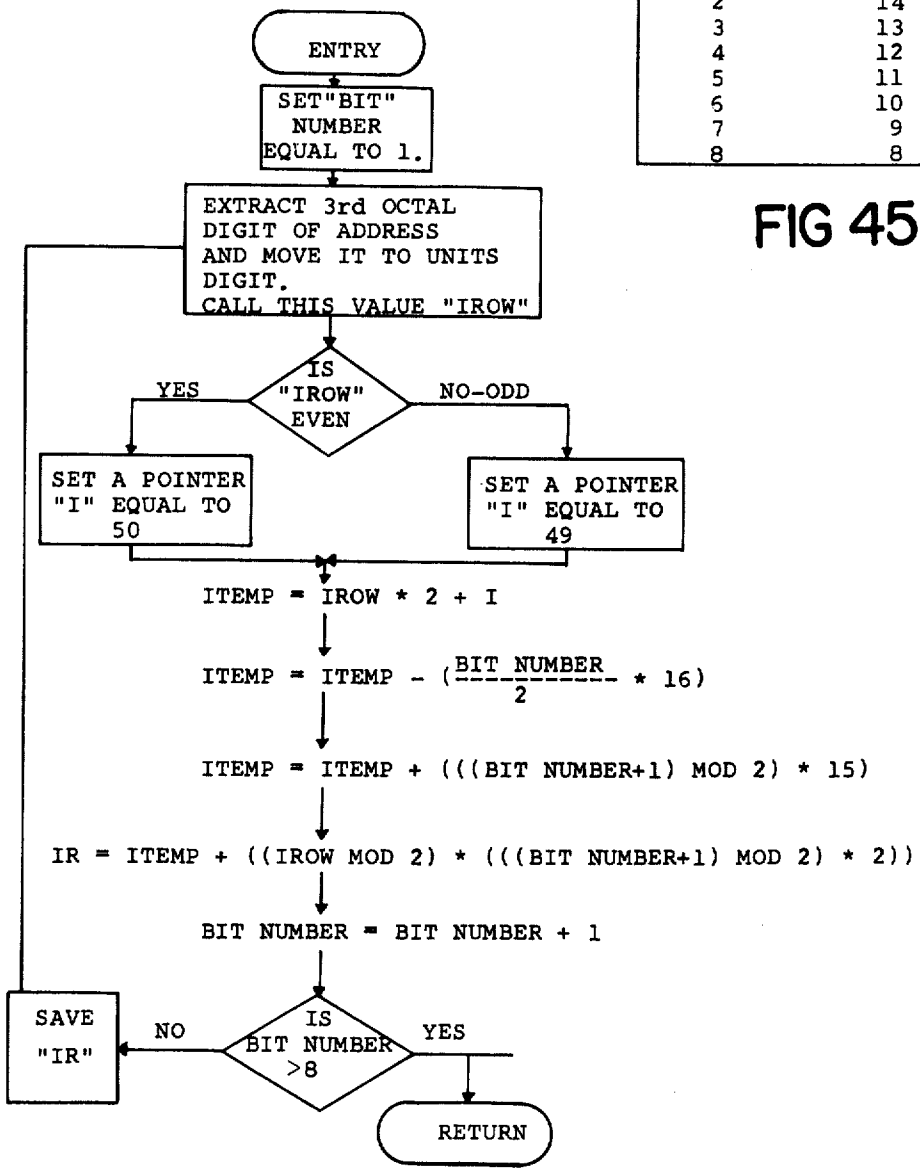


FIG 44

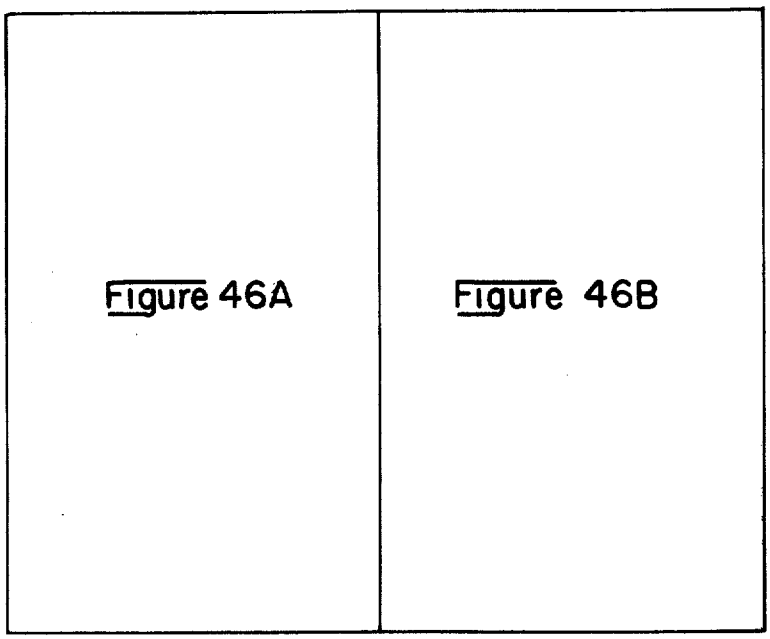


Figure 46'

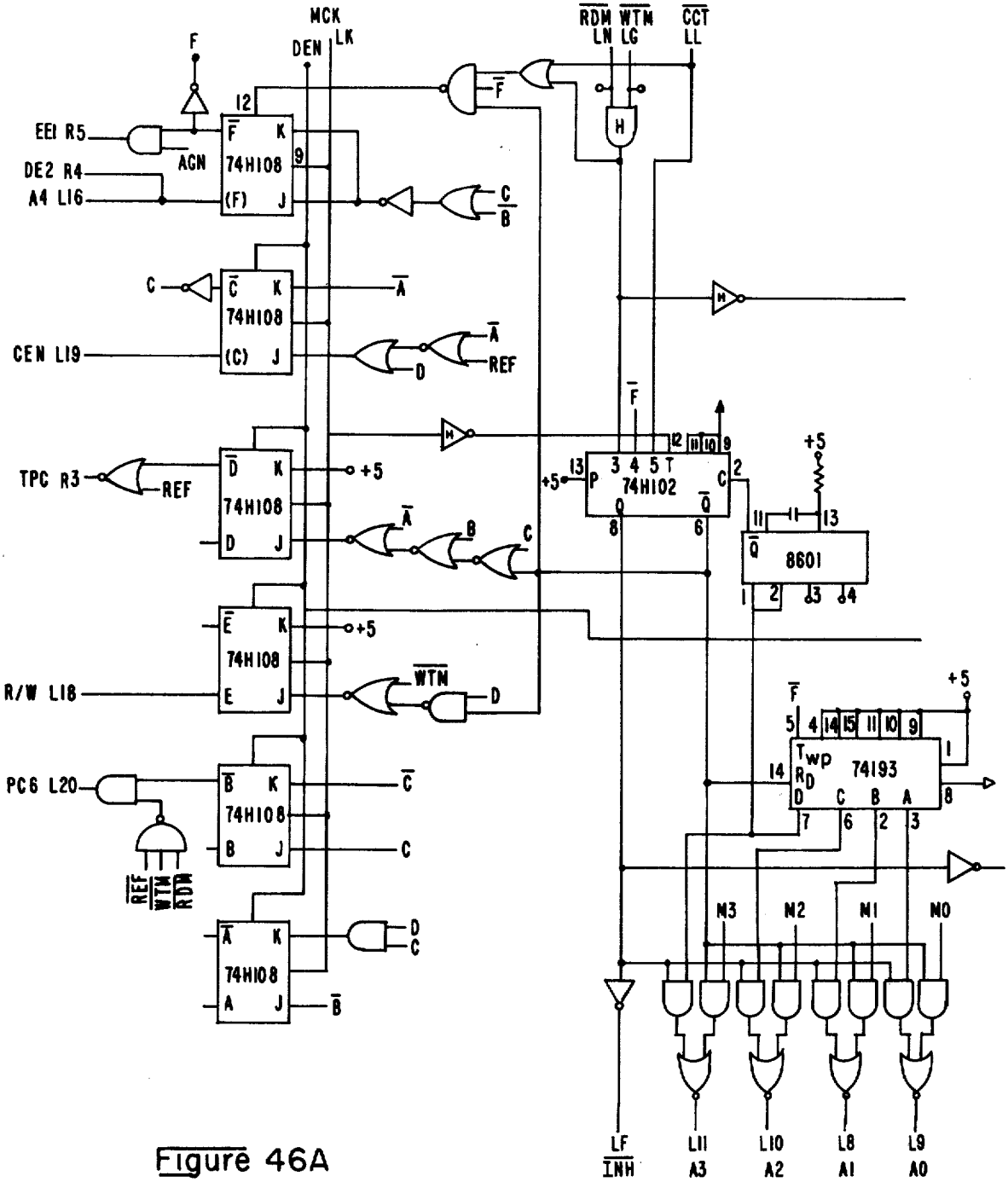

MCK
DEN
LK
F
RDM LN
WTM LG
CCT LL
EEI R5
AGN
DE2 R4
A4 LI6
74H108
74H102
74193
8601
C
CEN LI9
TPC R3
REF
R/W LI8
WTN
PC6 L20
M3
M2
M1
M0
LF INH
LII A3
LI0 A2
L8 A1
L9 A0

Figure 46A

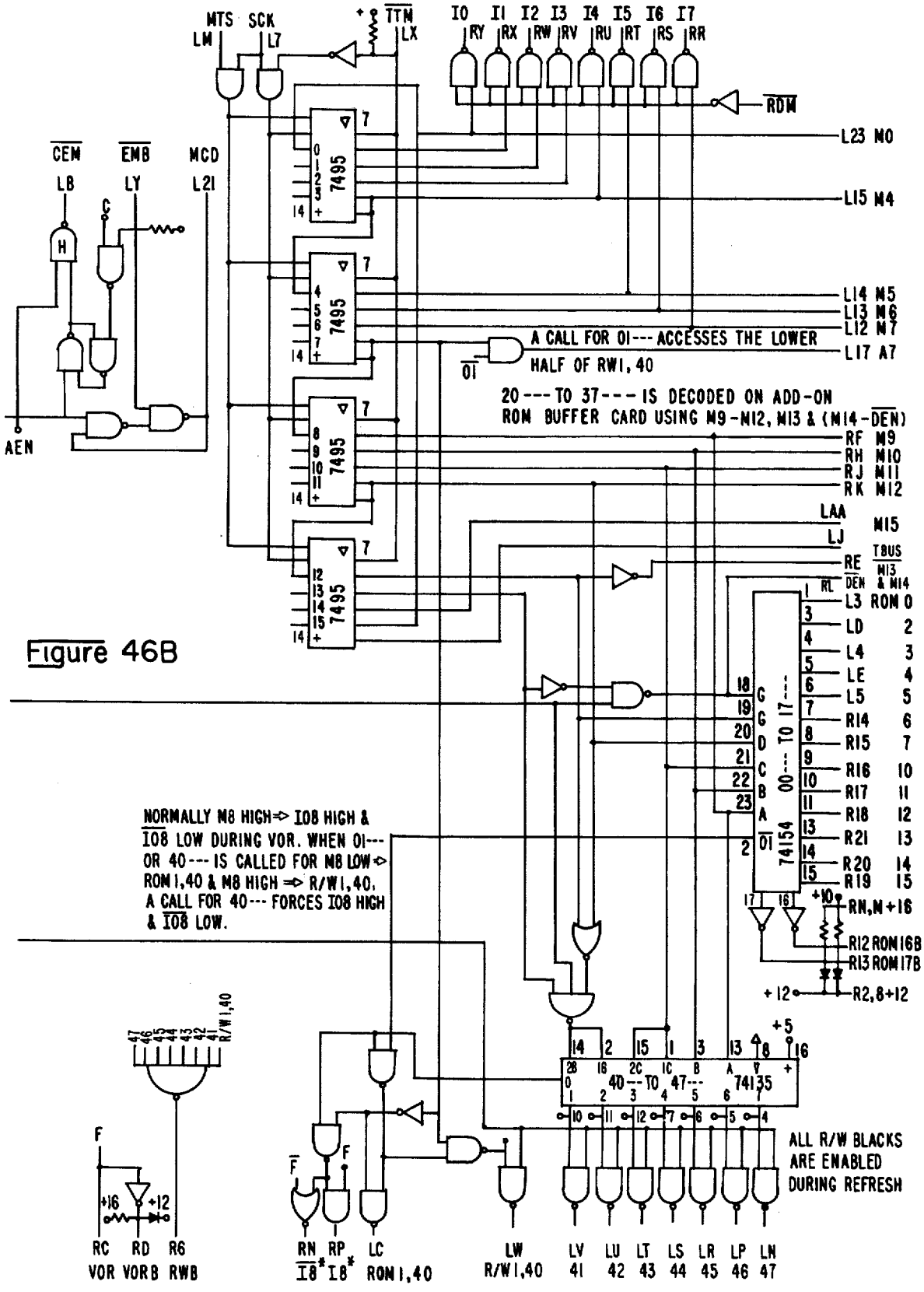

MTS SCK
LM L7
TTM LX
I0 I1 I2 I3 I4 I5 I6 I7
RY RX RW RV RU RT RS RR
RDM
L23 M0
L15 M4
L14 M5
L13 M6
L12 M7
L17 A7
CEN EMB MCD
LB LY L21
AEN
7495
A CALL FOR 01--- ACCESSES THE LOWER HALF OF RW1,40
20---TO 37--- IS DECODED ON ADD-ON ROM BUFFER CARD USING M9-M12, M13 & (M14-DEN)
RF M9
RH M10
RJ M11
RK M12
LAA M15
LJ
RE
T BUS M13 & M14
L3 ROM 0
LD 2
L4 3
LE 4
L5 5
R14 6
R15 7
R16 10
R17 11
R18 12
R21 13
R20 14
R19 15
74154
00---TO 17---
RN,M+16
R12 ROM16B
R13 ROM17B
R2,8+12
NORMALLY M8 HIGH⇒ I08 HIGH & I08 LOW DURING VOR. WHEN 01--- OR 40--- IS CALLED FOR M8 LOW⇒ ROM 1,40 & M8 HIGH ⇒ R/W1,40. A CALL FOR 40--- FORCES I08 HIGH & I08 LOW.
40---TO 47---
74135
ALL R/W BLACKS ARE ENABLED DURING REFRESH
RC RD R6
VOR VORB RWB
RN RP LC
I8* I8* ROM1,40
LW LV LU LT LS LR LP LN
R/W1,40 41 42 43 44 45 46 47

Figure 46B

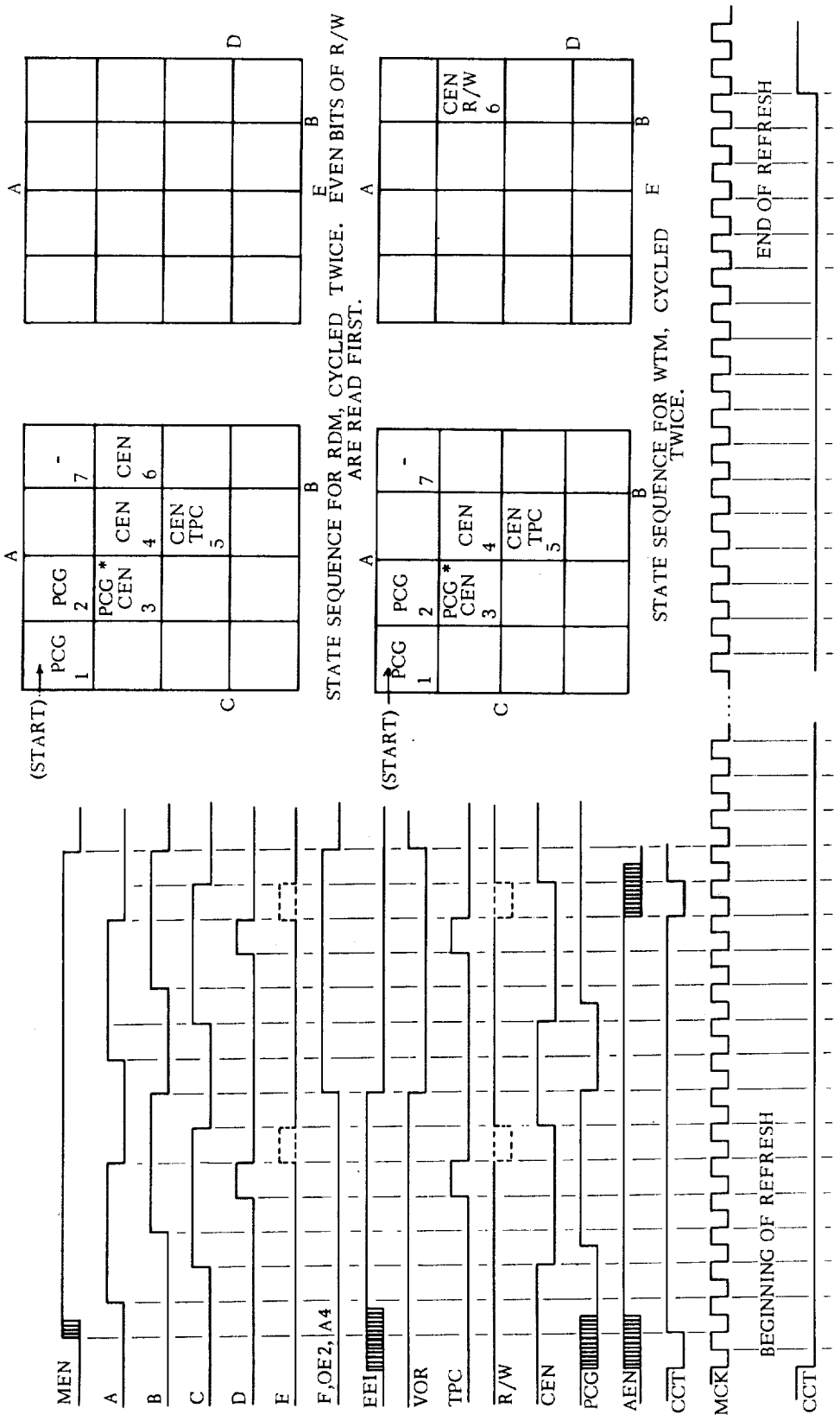
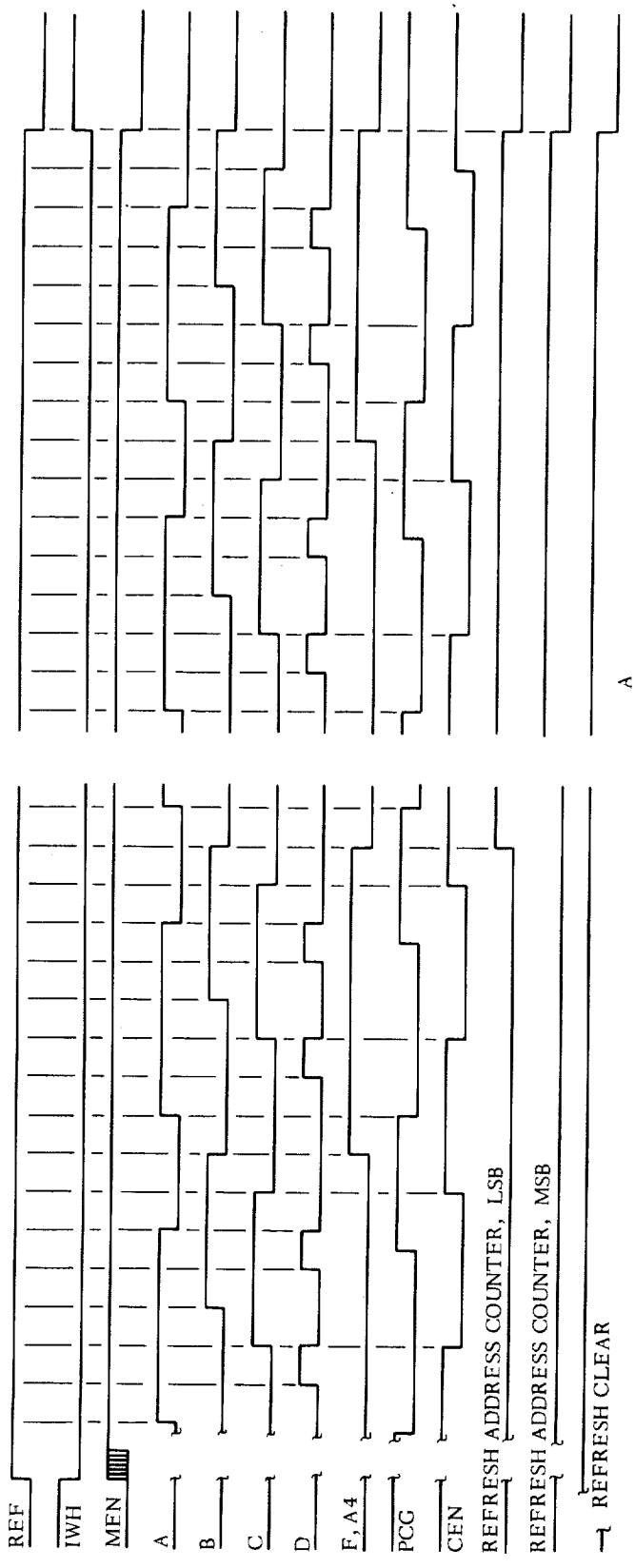
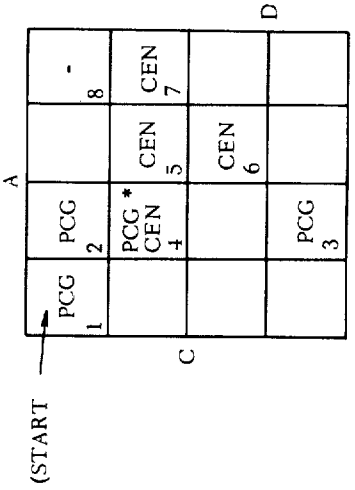

(START)
PCG 1
PCG 2
PCG * CEN 3
CEN 4
CEN TPC 5
CEN 6
- 7
A
B
C
D
E
STATE SEQUENCE FOR RDM, CYCLED TWICE. EVEN BITS OF R/W ARE READ FIRST.
CEN R/W 6
STATE SEQUENCE FOR WTM, CYCLED TWICE.
MEN
A
B
C
D
E
F,OE2,A4
FEI
VOR
TPC
R/W
CEN
PCG
AEN
CCT
MCK
BEGINNING OF REFRESH
END OF REFRESH
CCT

FIG 47A

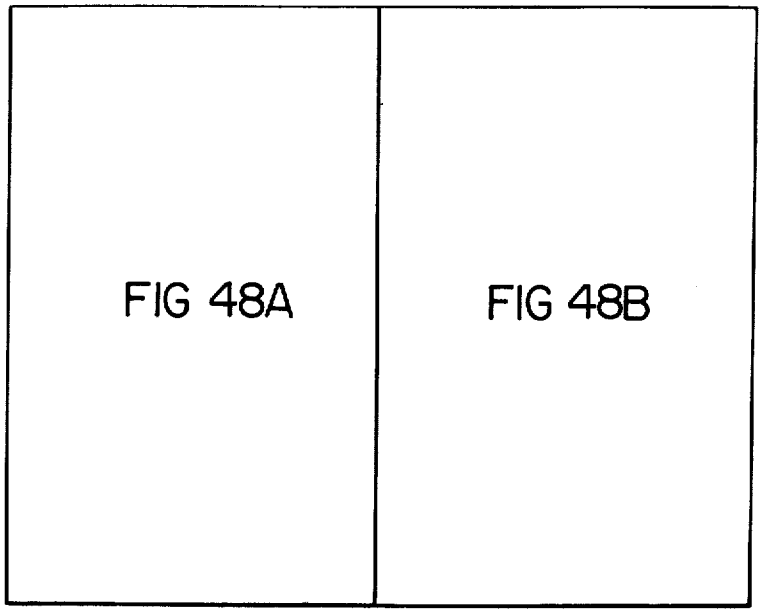




STATE SEQUENCE FOR REFRESH, CYCLED 32 TIMES

FIG 47B

FIG 48'



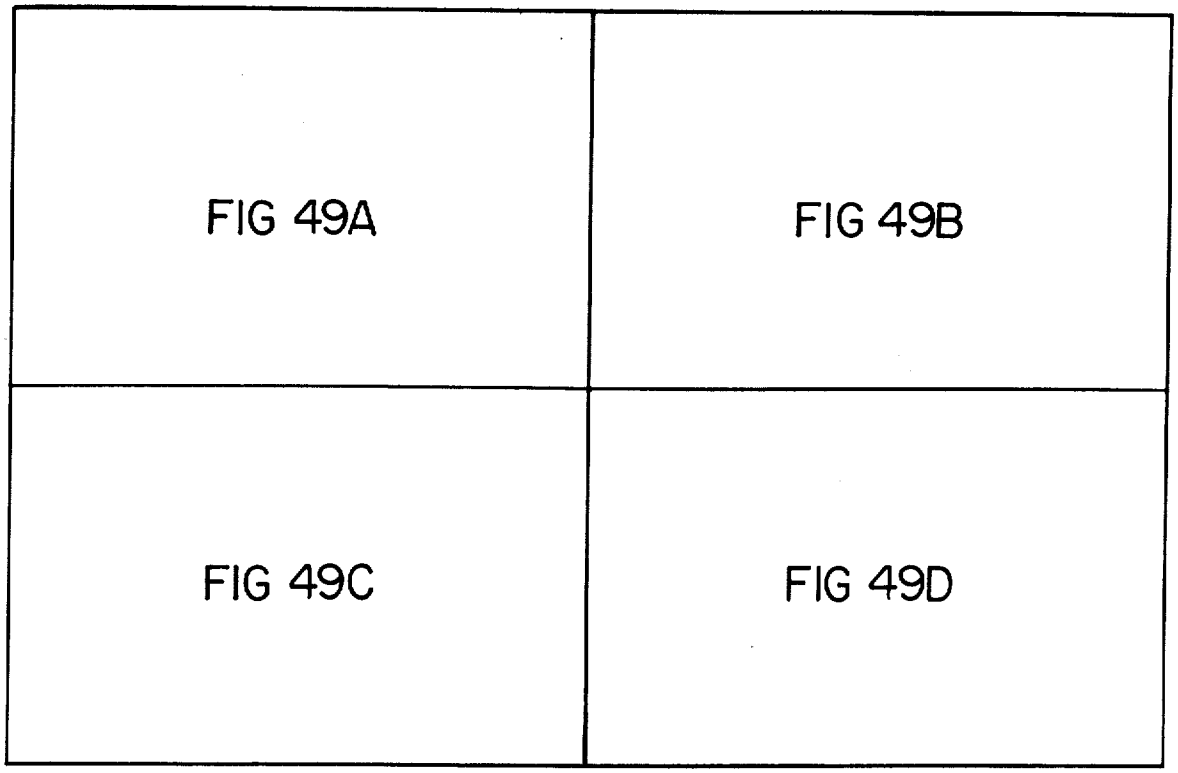


FIG 49'

L-M MTS
L-P STS
L-R TOD
L-AA MOΦ
L-W SCI
L-21 SCΦ
S-Bus L-11
L-V TTT
L-J T-Bus
L-23 D-Bus
L-22 RDT
L-7 WTm
L-H RDm
L-Y MCD
L-8 SCK
R-14 RWD 7
R-E ROD 15
R-F ROD 14
R-15 RWD 6
R-H ROD 13
R-J ROD 12
R-16 RWD 5
R-K ROD 11
R-L ROD 10
R-17 RWD 4
R-N ROD 9
R-P ROD 8
R-18 RWD 3
R-R ROD 7
R-S ROD 6
R-19 RWD 2
R-T ROD 5
R-U ROD 4
R-20 RWD 1
R-V ROD 3
R-W ROD 2
R-21 RWD Φ
R-X ROD 1
R-Y ROD Φ
+12
+V REF
TO ALL S.R.'s
R-13 INT LO
R-6 RWE
R-4 TPC
L-K I3
L-T I2
L-S I1
L-P IΦ

FIG 48A

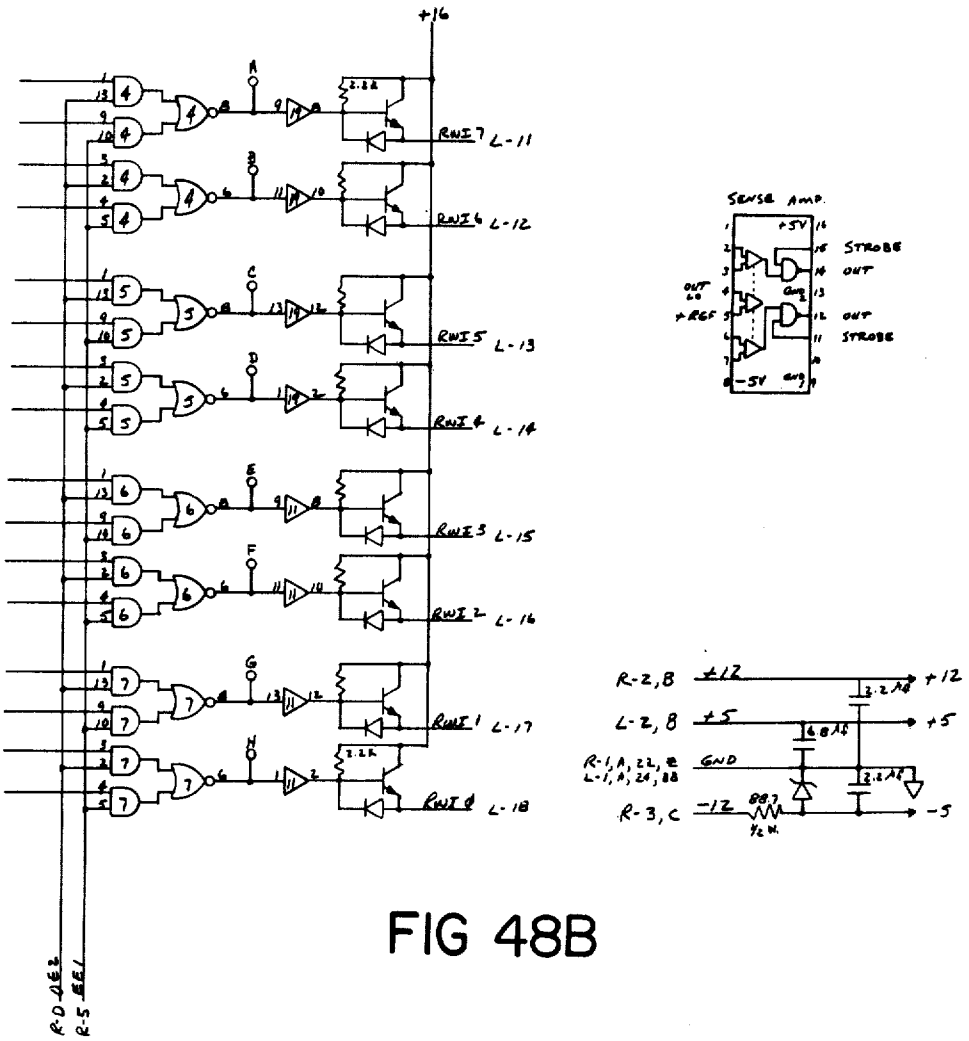

+16
A
B
C
D
E
F
G
H
RWI 7
RWI 6
RWI 5
RWI 4
RWI 3
RWI 2
RWI 1
RWI 0
L-11
L-12
L-13
L-14
L-15
L-16
L-17
L-18
R-D AEL
R-S AEL
SENSE AMP.
+5V
STROBE
OUT
OUT
OUT
STROBE
-5V
R-2, B
L-2, B
R-1, A, 22, 88
L-1, A, 24, 88
R-3, C
+12
+5
GND
-12
-5

FIG 48B

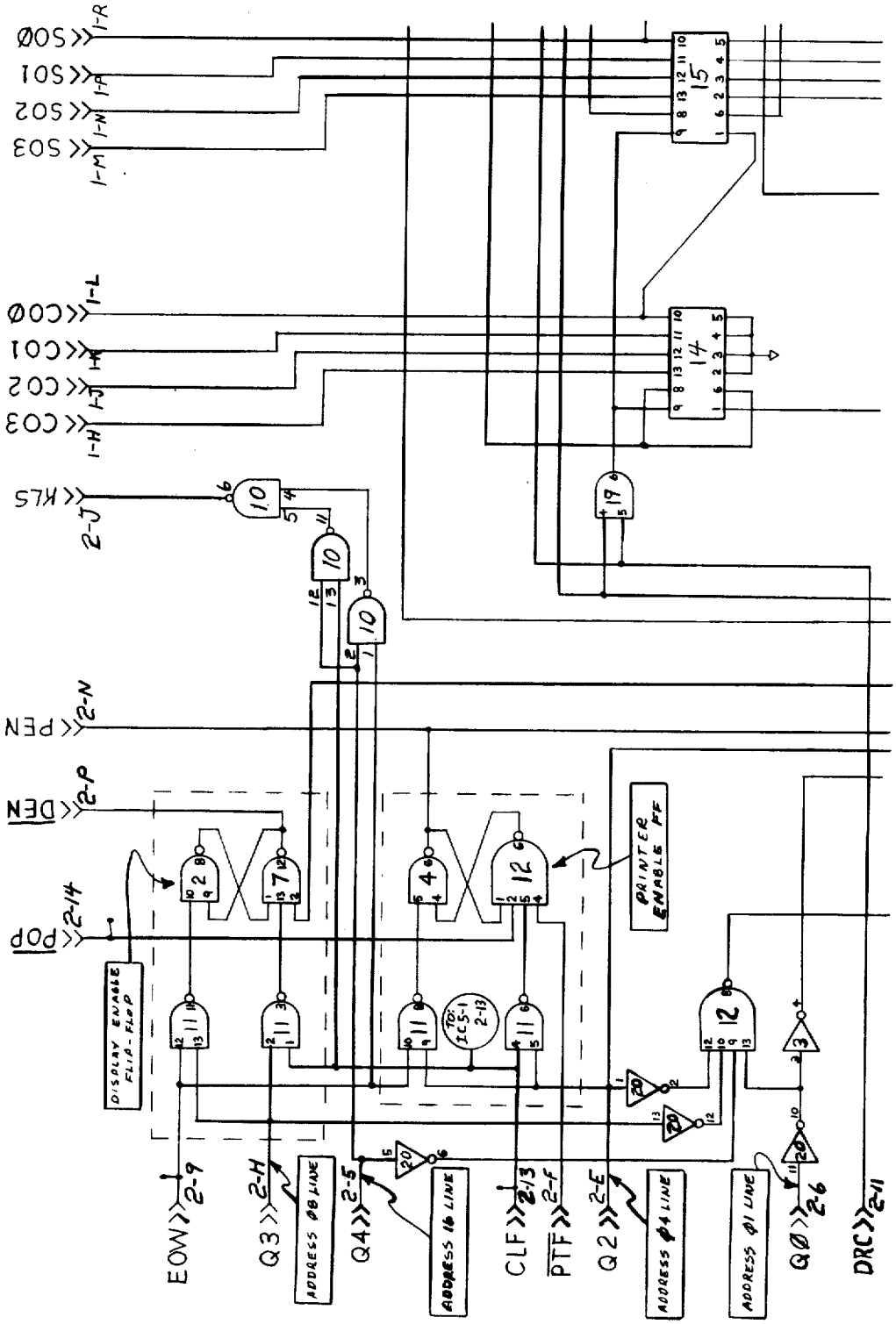
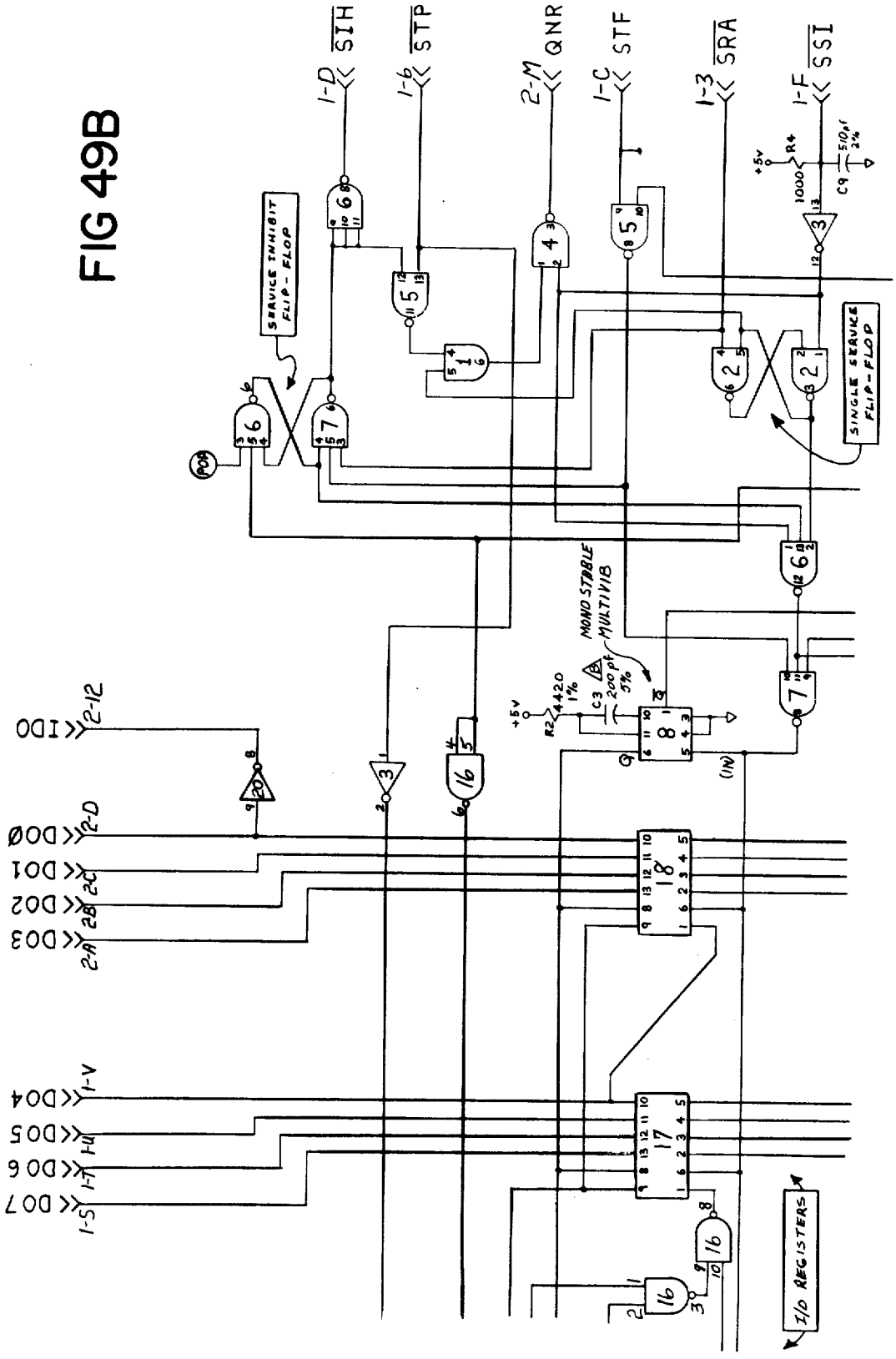

S0Ø
S01
S02
S03
C0Ø
C01
C02
C03
PEN
DEN
POP
EOW
Q3
Q4
CLF
PTF
Q2
QØ
DRC
PRINTER ENABLE FF
DISPLAY ENABLE FLIP-FLOP
ADDRESS Ø8 LINE
ADDRESS 16 LINE
ADDRESS Ø4 LINE
ADDRESS Ø1 LINE

FIG 49A

FIG 49B


1-D SIH
1-6 STP
2-M QNR
1-C STF
1-3 SRA
1-F SSI
SERVICE INHIBIT FLIP-FLOP
SINGLE SEAVICE FLIP-FLOP
MONO STABLE MULTIVIB
+5V
R2 4420 1%
C3 200 pf 5%
R4 1000
C9 .01μf 2%
I/O REGISTERS
2-12 IDO
2-D DO0
DO1
DO2
DO3
1-V DO4
DO5
DO6
DO7

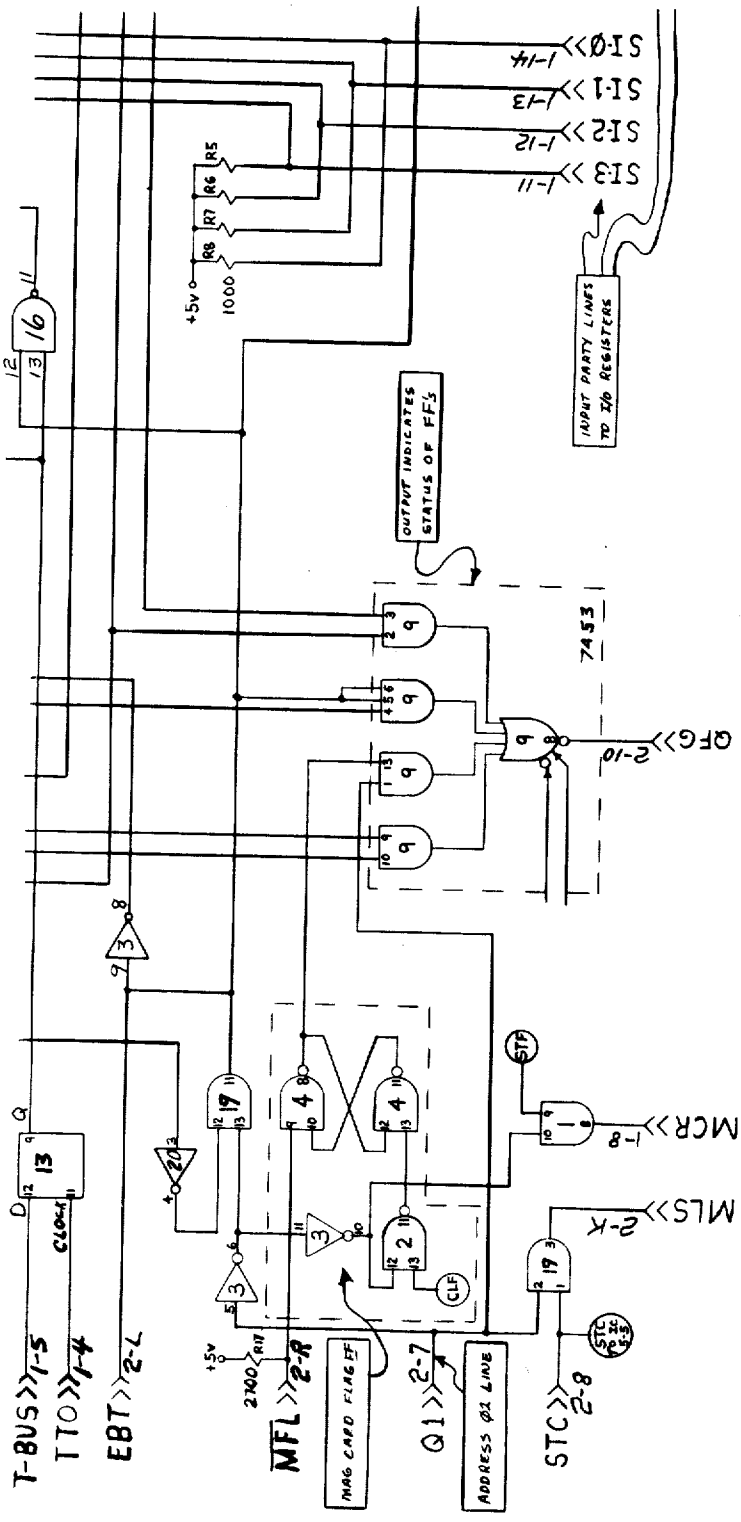


FIG 49C

COMPONENT IC NO.	SERIES NO.	DISCRIPTION
1, 19	MC-3001P	QUAD 2-INPUT 'AND' GATES
2, 4, 5, 10, 11, 16	SN-7400	QUAD 2-INPUT 'NAND' GATES
3, 20	-7404	6 FK INVERTERS
6, 7	-7410	TRIPLE 3-INPUT 'NAND' GATES
8	-74121	MONOSTABLE MULTIVIBRATOR
9	-7453	EXPANDABLE 4-WIDE 2-INPUT 'AND-OR-INVERTED GATES
12	-7420	DUAL 4-INPUT 'NAND' GATES
13	-7474	DUAL EDGE-TRIGGERD FLIP-FLOPS
14, 15, 17, 18	-7495	4-BIT RIGHT SHIFT, LEFT SHIFT REGISTER

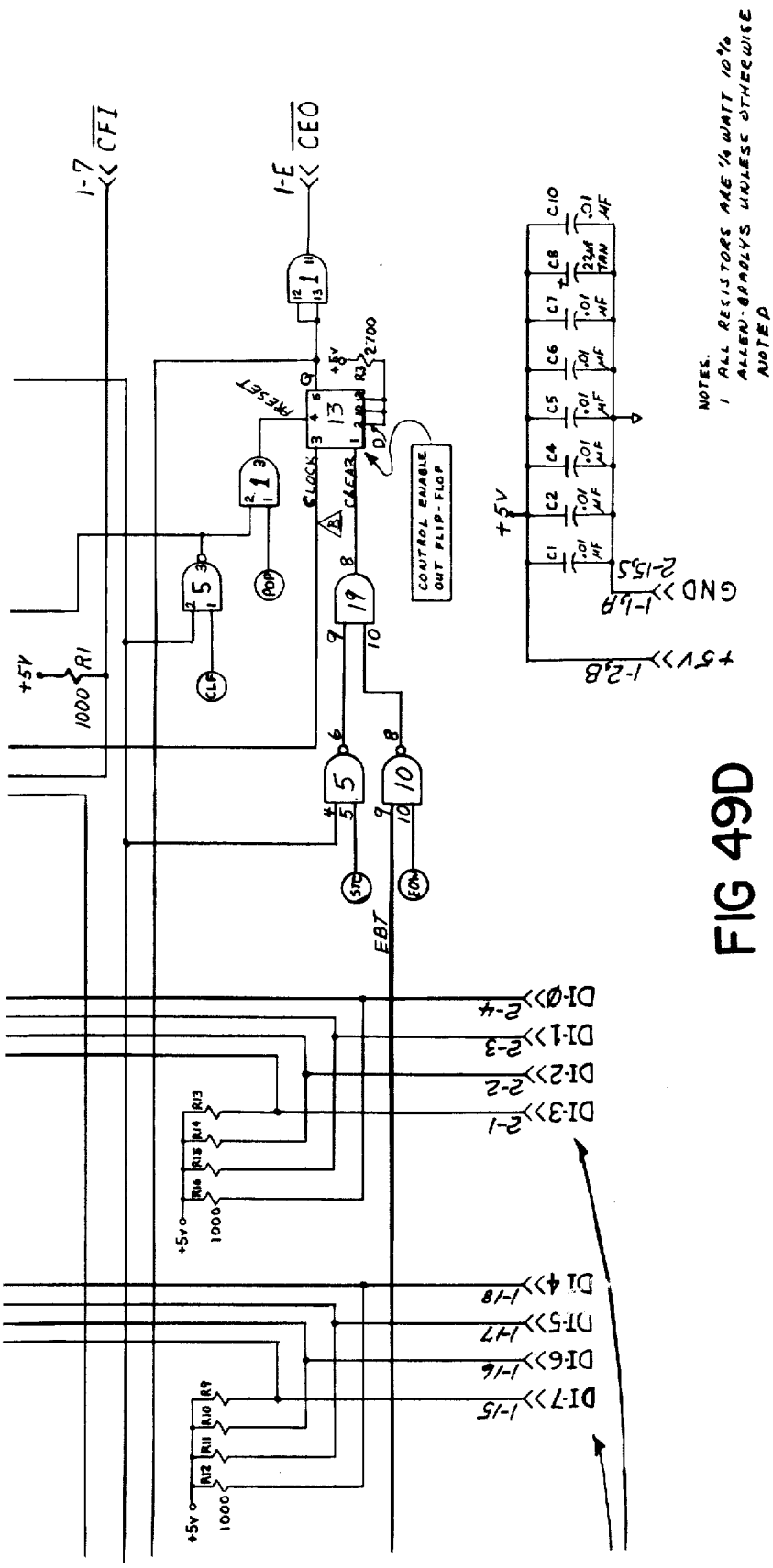

1-7 CFI
1-E CEO
+5V
1000
R1
CLEAR
CLOCK
PRESET
CONTROL ENABLE OUT FLIP-FLOP
EBT
DI-0
DI-1
DI-2
DI-3
DI-4
DI-5
DI-6
DI-7
GND
+5V
NOTES.
1 ALL RESISTORS ARE ¼ WATT 10% ALLEN-BRADLYS UNLESS OTHERWISE NOTED

FIG 49D

Relationship of the I/O Lines.

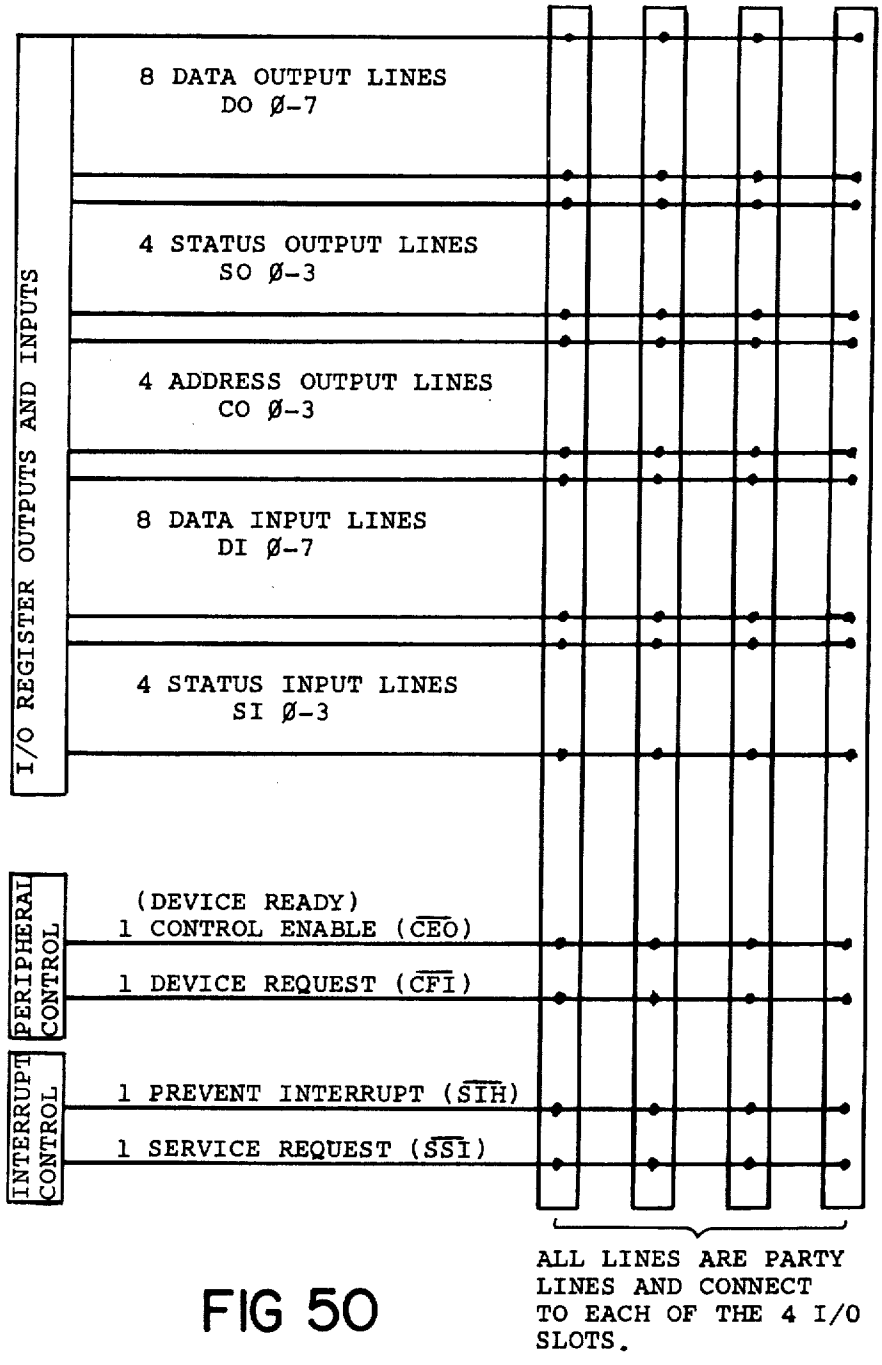


FIG 50

I/O Control Board Timing Diagrams Showing Execution of STC

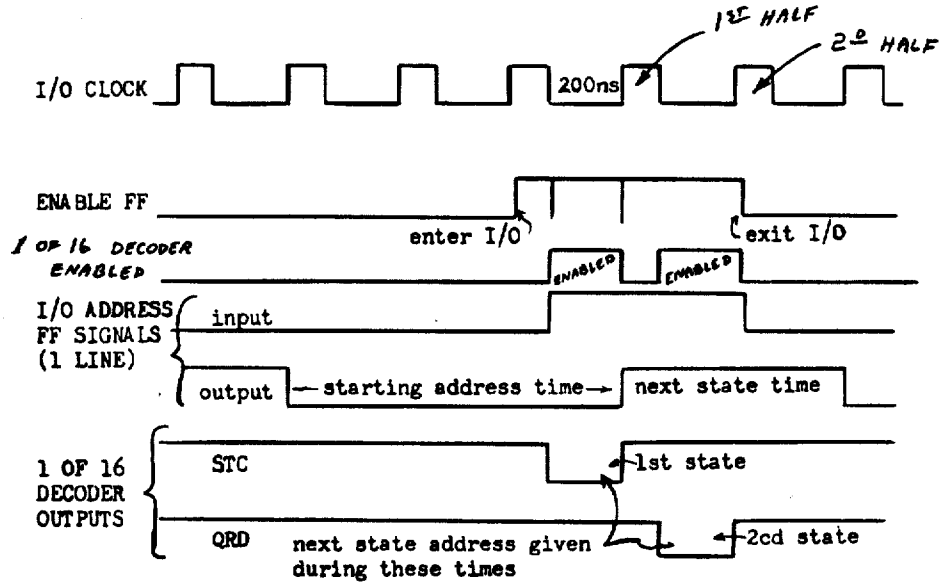


FIG 51

FIG 52

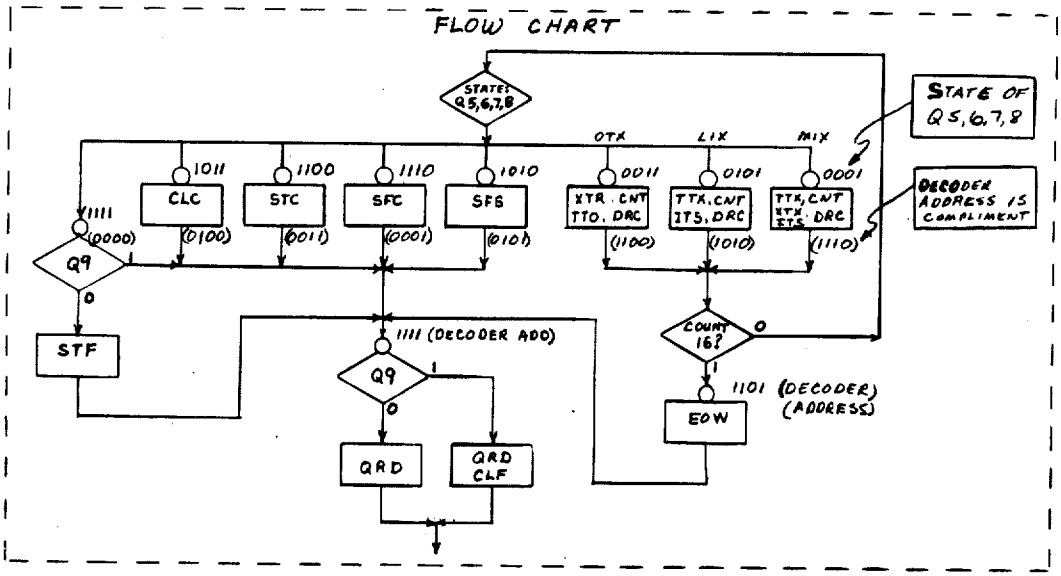
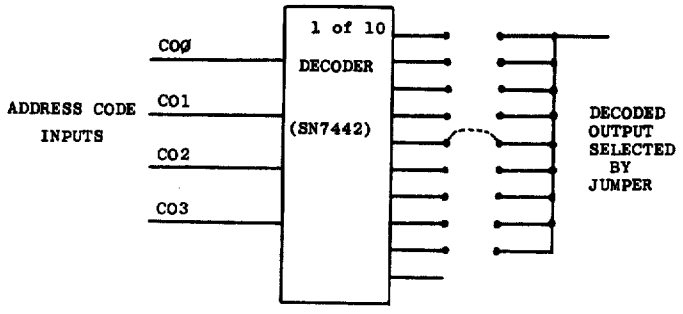

FLOW CHART
STATES Q5,6,7,8
STATE OF Q5,6,7,8
DECODER ADDRESS IS COMPLIMENT
1011
CLC
1100
STC
1110
SFC
1010
SFS
0011
XTR. CNT TTO. DRC
0101
TTX. CNT 1T5. DRC
0001
TTX. CNT XTX. ETS. DRC
(1111)
(0000)
Q9
(0100)
(0011)
(0001)
(0101)
(1100)
(1010)
(1110)
STF
1111 (DECODER ADD)
Q9
COUNT 16?
1101 (DECODER) (ADDRESS)
EOW
QRD
QRD CLF

FIG 53


1 of 10
DECODER
(SN7442)
CO0
CO1
CO2
CO3
ADDRESS CODE INPUTS
DECODED OUTPUT SELECTED BY JUMPER

GENERAL MODE INPUT WAVEFORMS.

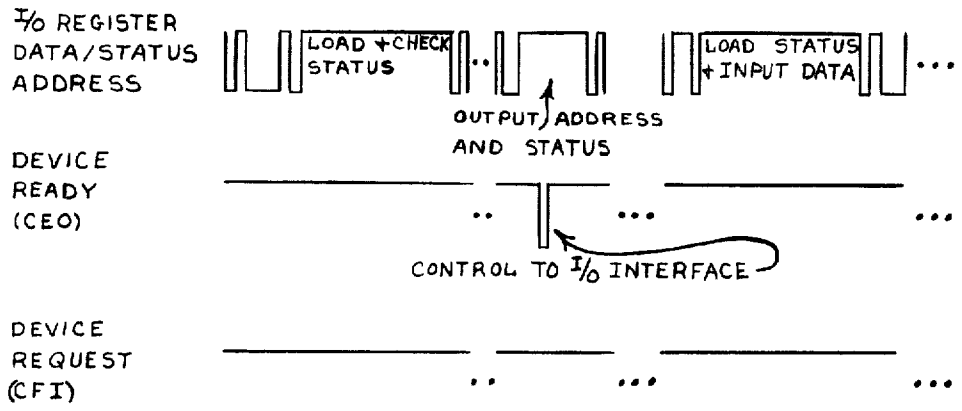


FIG 54

GENERAL MODE OUTPUT WAVEFORMS

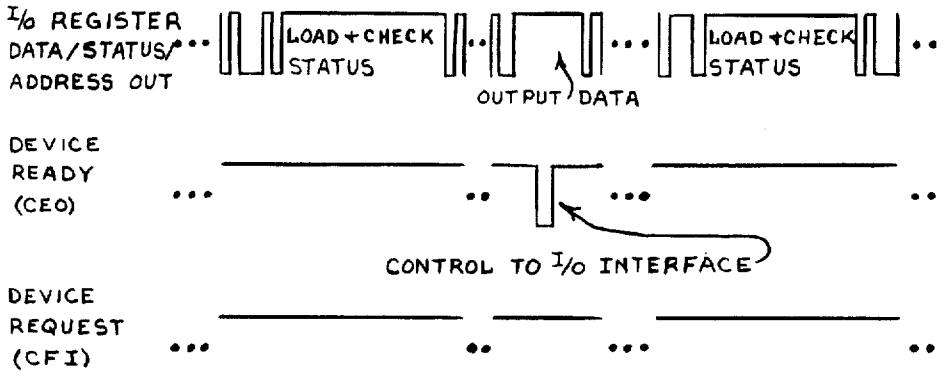


FIG 55

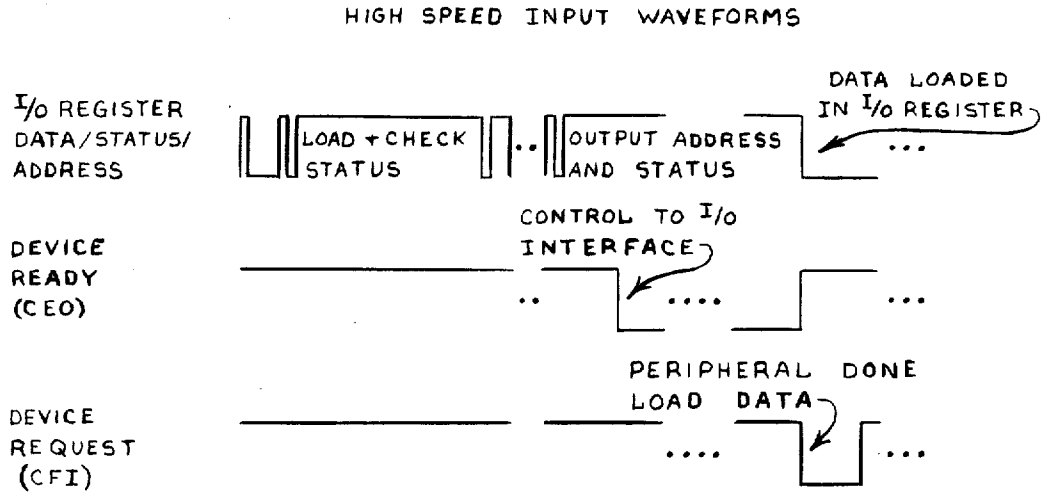


FIG 56

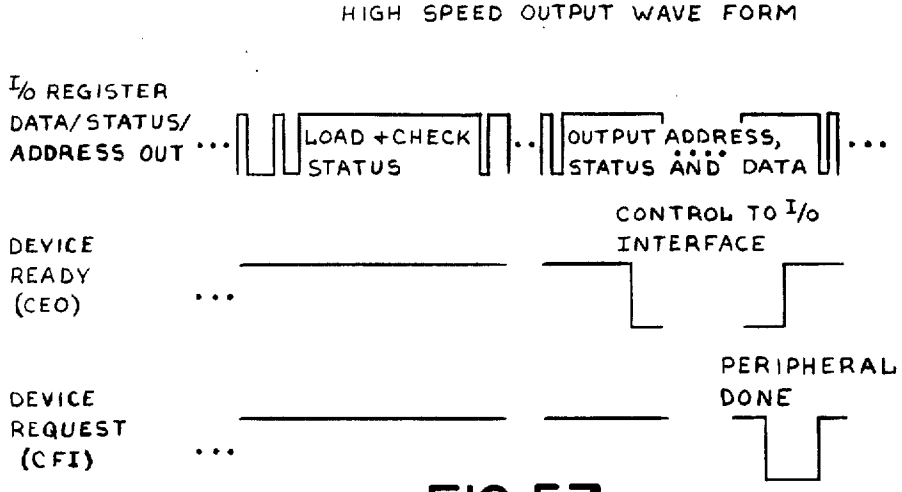


FIG 57

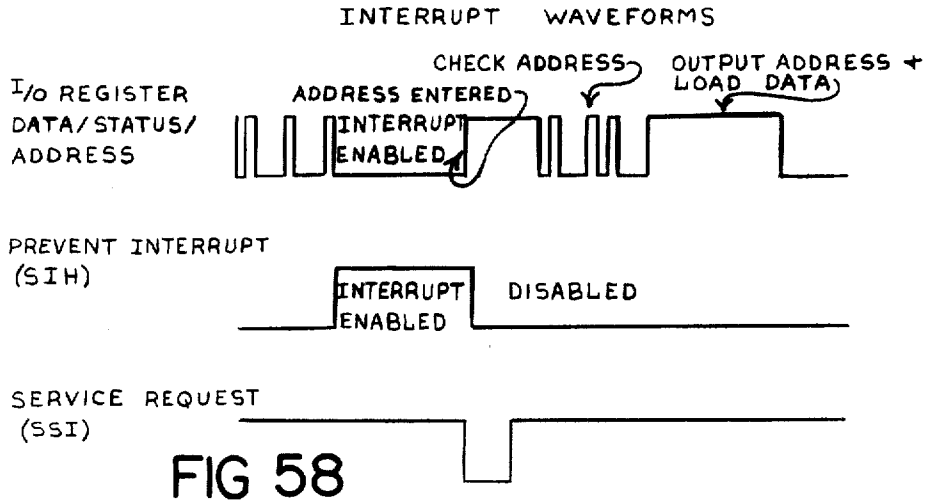


FIG 58

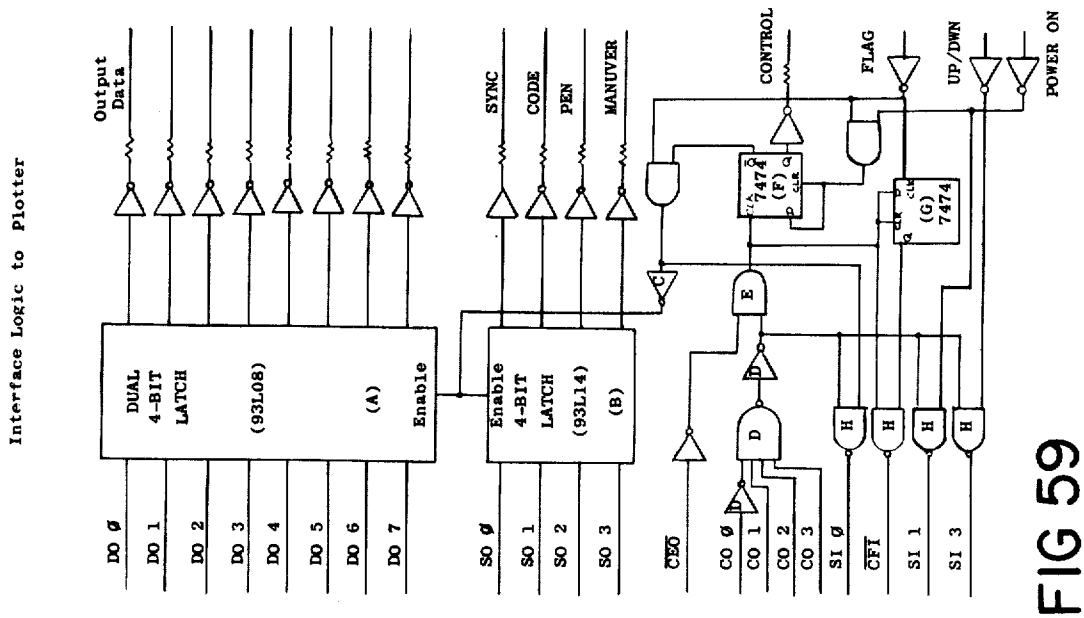

Interface Logic to Plotter
Output Data
DUAL 4-BIT LATCH (93L08) (A)
Enable
DO Ø
DO 1
DO 2
DO 3
DO 4
DO 5
DO 6
DO 7
SYNC
CODE
PEN
MANUVER
Enable 4-BIT LATCH (93L14) (B)
SO Ø
SO 1
SO 2
SO 3
CONTROL
FLAG
UP/DWN
POWER ON
7474 (F)
(G) 7474
CEO
CO Ø
CO 1
CO 2
CO 3
SI Ø
CFI
SI 1
SI 3

FIG 59

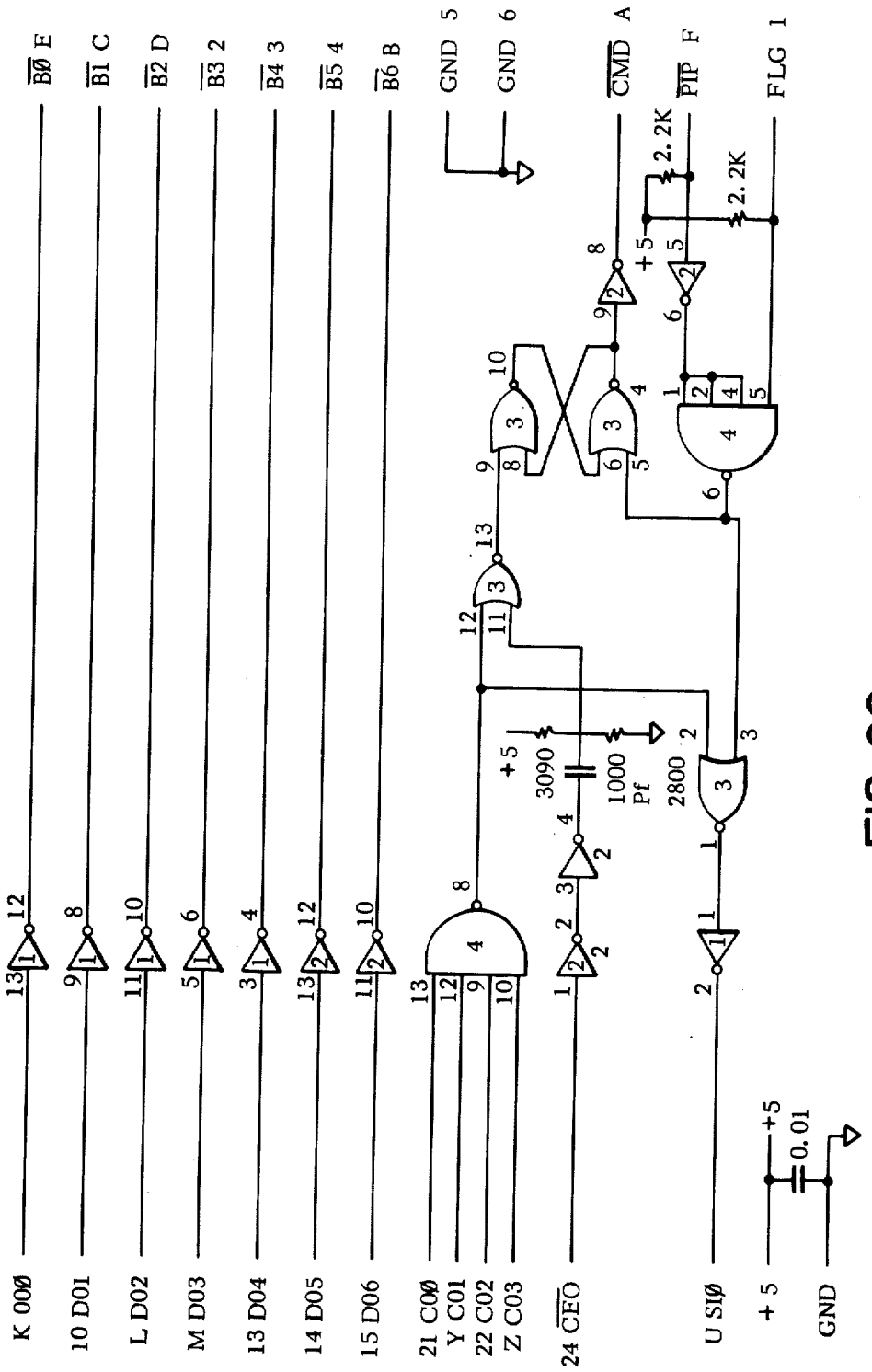

K 00
10 D01
L D02
M D03
13 D04
14 D05
15 D06
21 C00
Y C01
22 C02
Z C03
24 CEO
U SIO
+5
GND
0.01
B0 E
B1 C
B2 D
B3 2
B4 3
B5 4
B6 B
GND 5
GND 6
CMD A
PIP F
FLG 1
2.2K
2.2K
3090
1000 Pf
2800

FIG 60

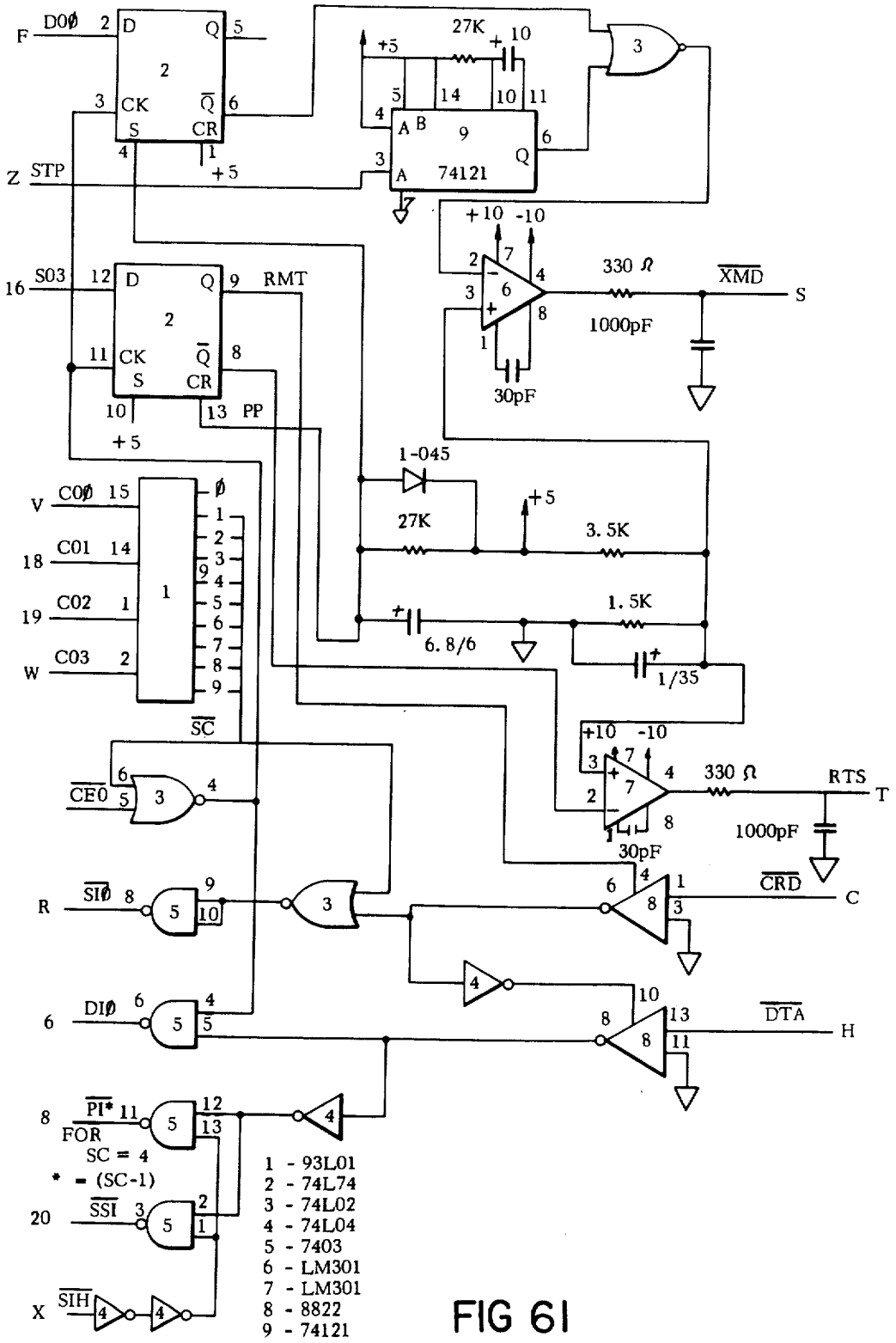

F
DØ
Z STP
S03
RMT
PP
+5
27K
+10
3
74121
+10 -10
330 Ω
XMD
S
1000pF
30pF
1-045
3.5K
1.5K
6.8/6
1/35
V CØ
18 C01
19 C02
W C03
SC
CE0
RTS
T
CRD
C
R SIØ
DIØ
6
DTA
H
8 PI*
FOR
SC = 4
* = (SC-1)
20 SSI
X SIH
1 - 93L01
2 - 74L74
3 - 74L02
4 - 74L04
5 - 7403
6 - LM301
7 - LM301
8 - 8822
9 - 74121

FIG 61

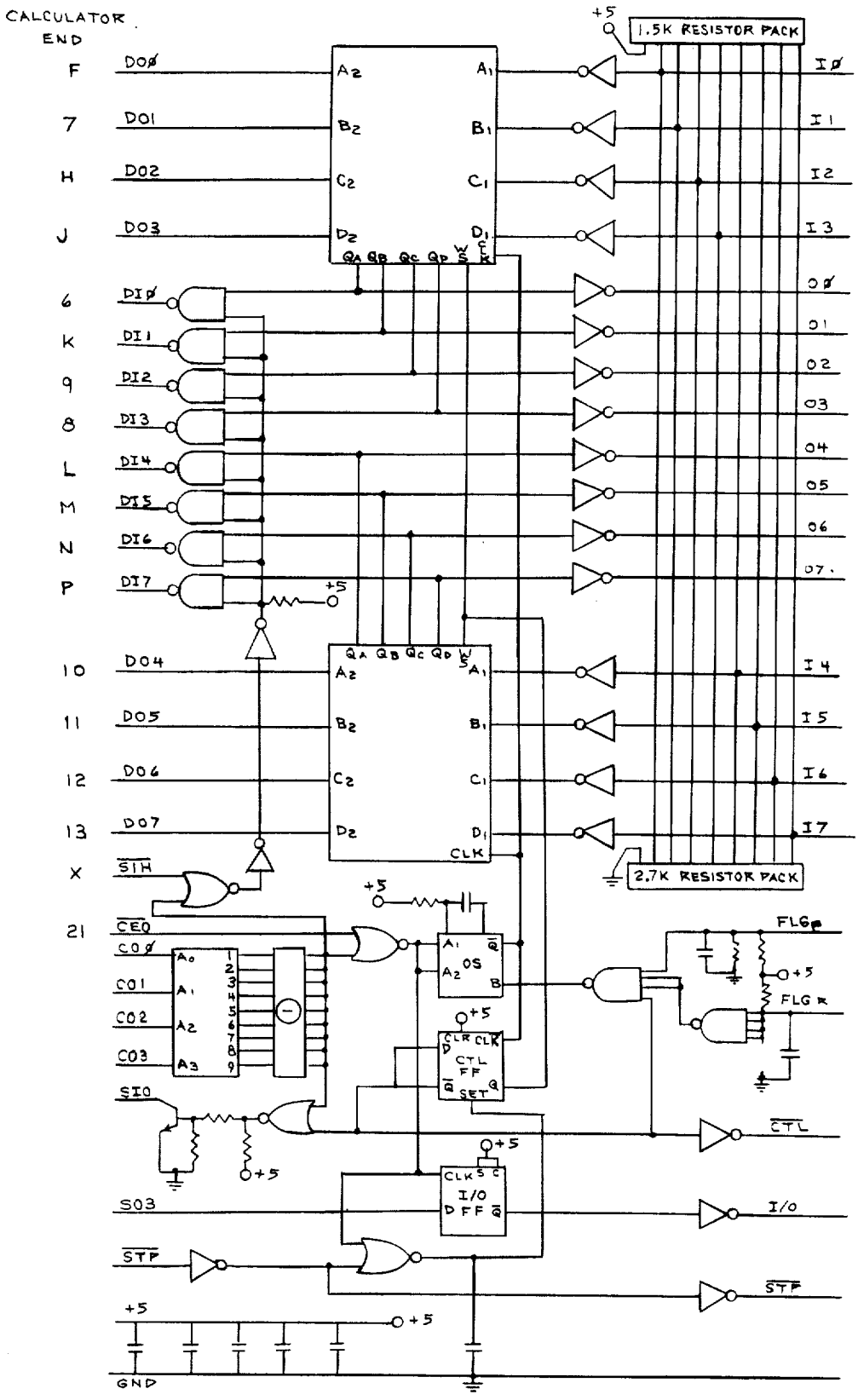

CALCULATOR END
F
7
H
J
6
K
9
8
L
M
N
P
10
11
12
13
X
21
DO0
DO1
DO2
DO3
DI0
DI1
DI2
DI3
DI4
DI5
DI6
DI7
DO4
DO5
DO6
DO7
SIH
CEO
CO0
CO1
CO2
CO3
SIO
SO3
STP
GND
+5
1.5K RESISTOR PACK
2.7K RESISTOR PACK
I0
I1
I2
I3
O0
O1
O2
O3
O4
O5
O6
O7
I4
I5
I6
I7
FLG
CTL
I/O
STF

FIG 62

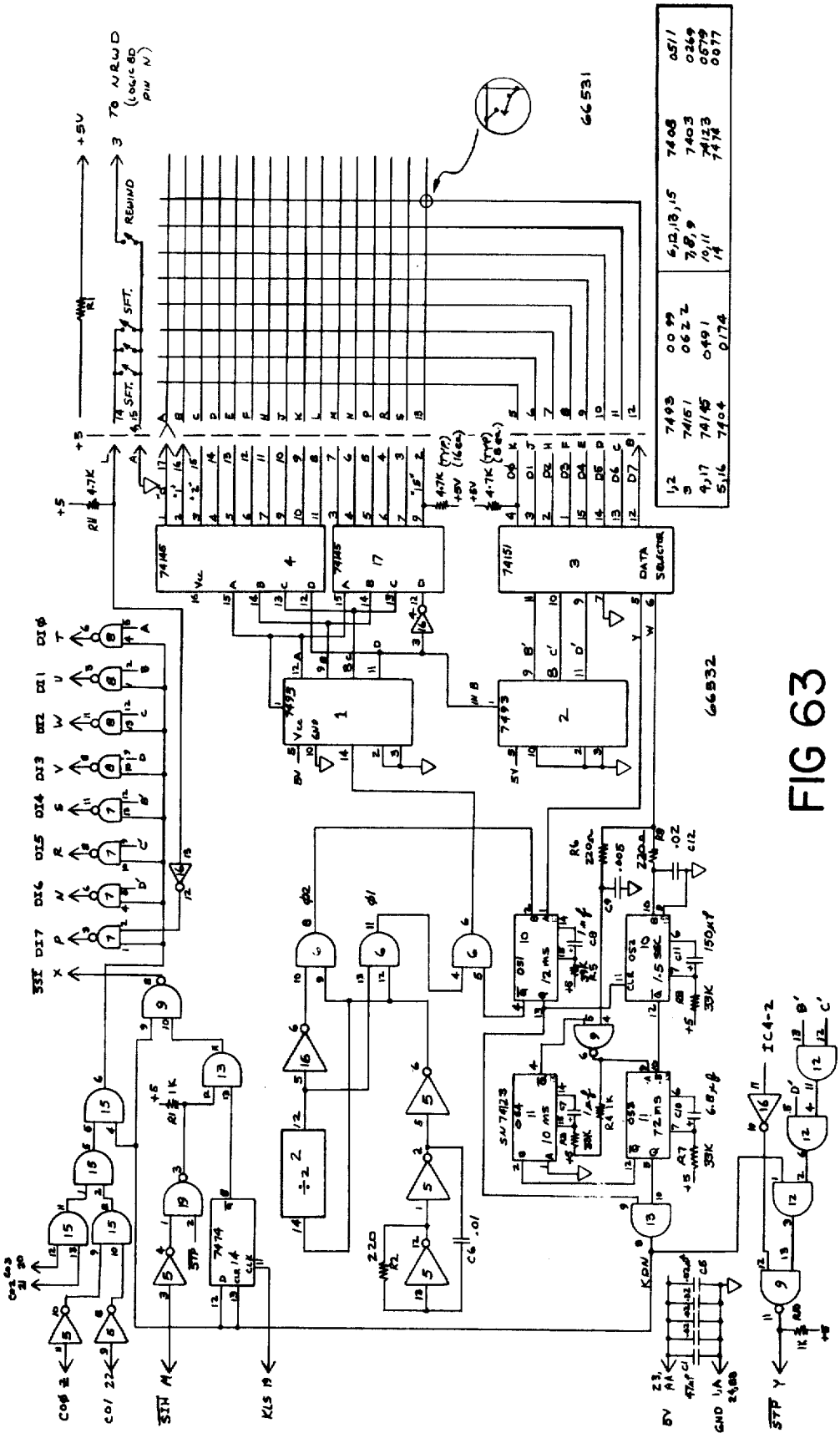


FIG 63

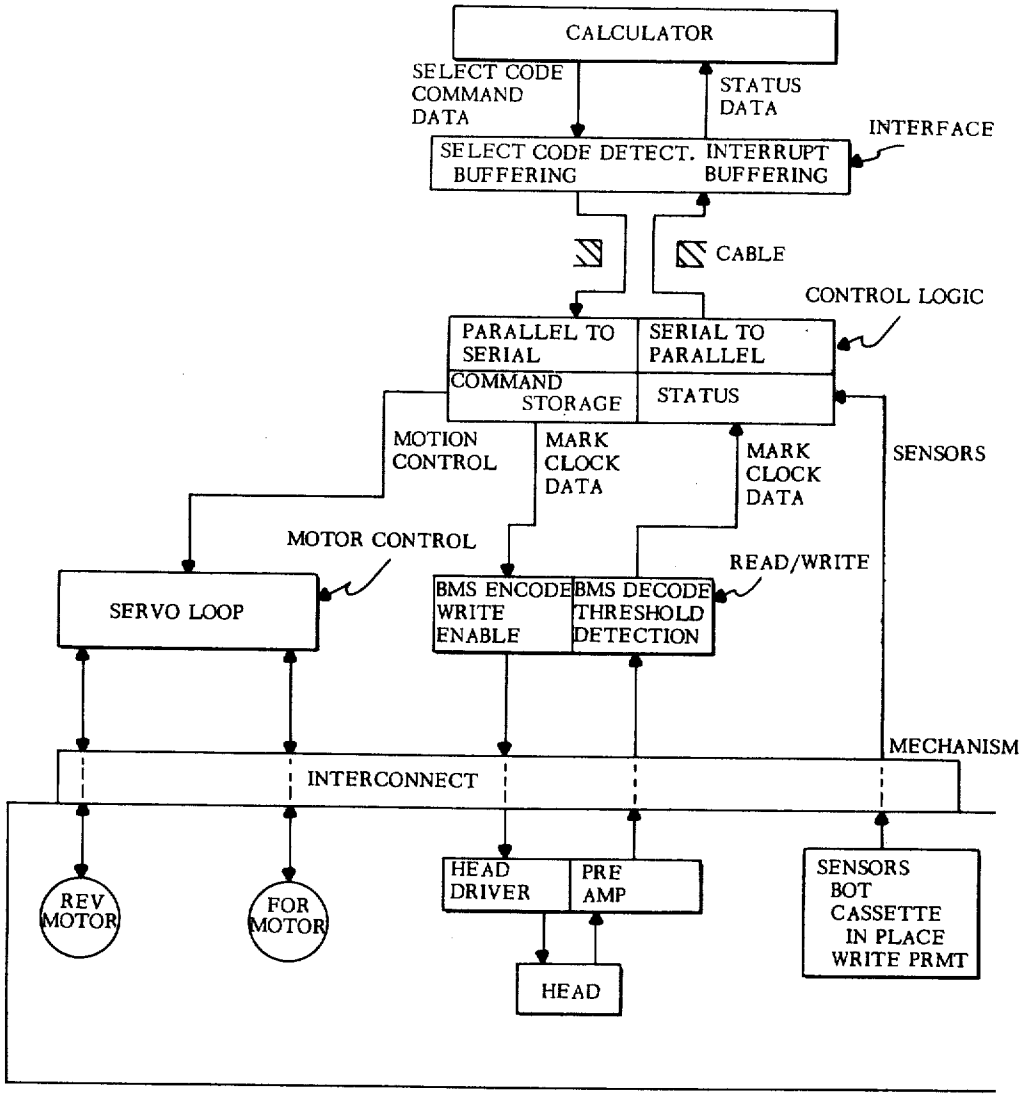

CALCULATOR
SELECT CODE COMMAND DATA
STATUS DATA
INTERFACE
SELECT CODE BUFFERING
DETECT. INTERRUPT BUFFERING
CABLE
CONTROL LOGIC
PARALLEL TO SERIAL
SERIAL TO PARALLEL
COMMAND STORAGE
STATUS
MOTION CONTROL
MARK CLOCK DATA
MARK CLOCK DATA
SENSORS
MOTOR CONTROL
READ/WRITE
SERVO LOOP
BMS ENCODE WRITE ENABLE
BMS DECODE THRESHOLD DETECTION
MECHANISM
INTERCONNECT
REV MOTOR
FOR MOTOR
HEAD DRIVER
PRE AMP
HEAD
SENSORS BOT CASSETTE IN PLACE WRITE PRMT

FIG 64

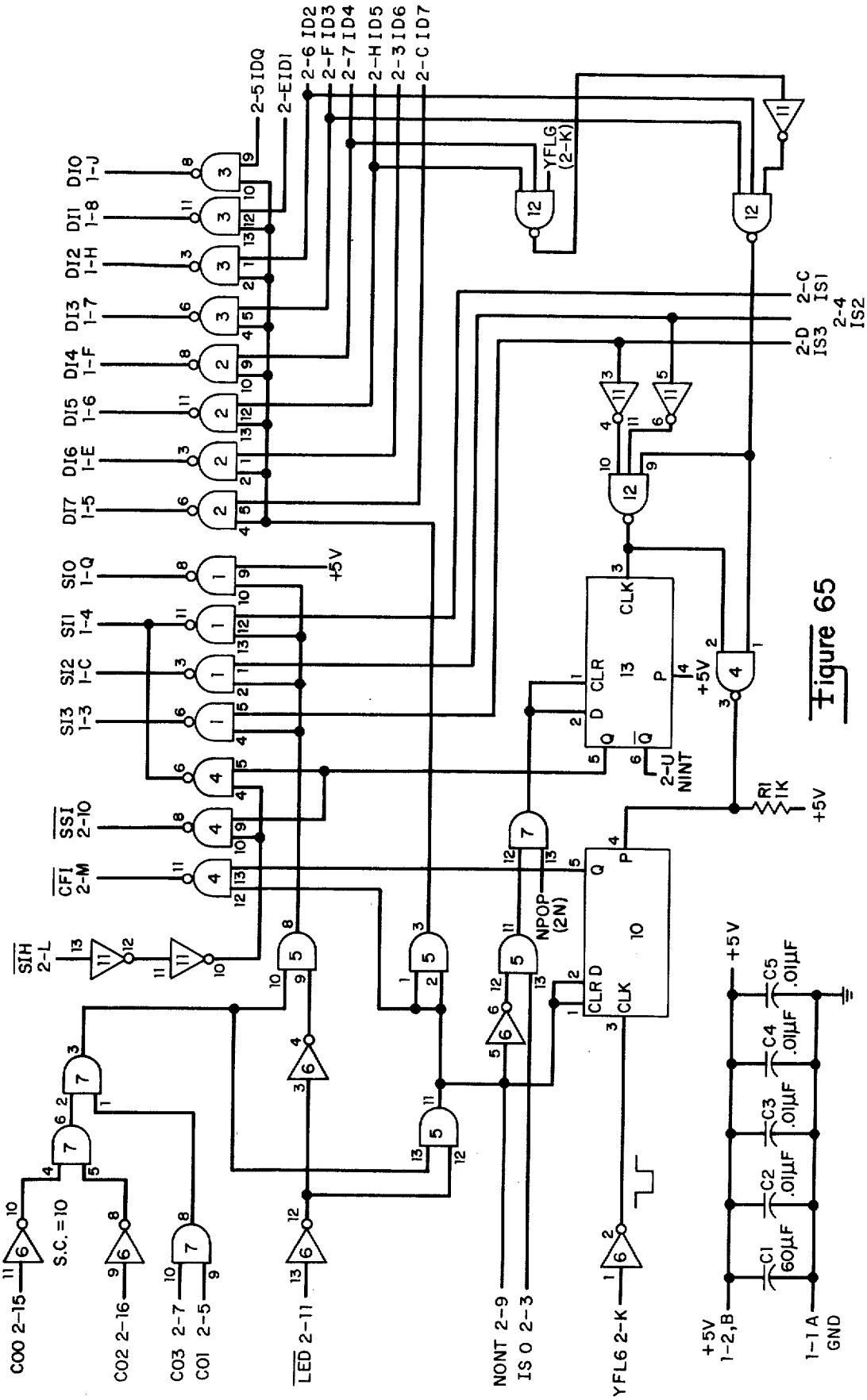

2-5 IDQ
2-E ID1
2-6 ID2
2-F ID3
2-7 ID4
2-H ID5
2-3 ID6
2-C ID7
DI0 1-J
DI1 1-8
DI2 1-H
DI3 1-7
DI4 1-F
DI5 1-6
DI6 1-E
DI7 1-5
SI0 1-Q
SI1 1-4
SI2 1-C
SI3 1-3
SSI 2-10
CFI 2-M
SIH 2-L
+5V
YFLG (2-K)
2-C IS1
2-4 IS2
2-D IS3
CLK
CLR
D
P
Q
2-U NINT
R1 1K
NPOP (2N)
S.C. = 10
C00 2-15
C02 2-16
C03 2-7
C01 2-5
LED 2-11
NONT 2-9
IS 0 2-3
YFL6 2-K
C1 60µF
C2 .01µF
C3 .01µF
C4 .01µF
C5 .01µF
+5V 1-2,B
1-1 A GND

Figure 65

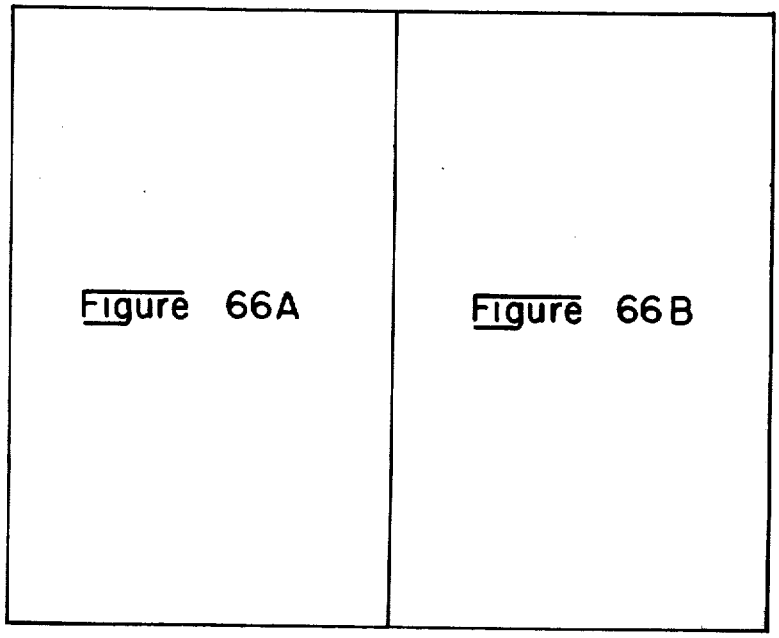


Figure 66'

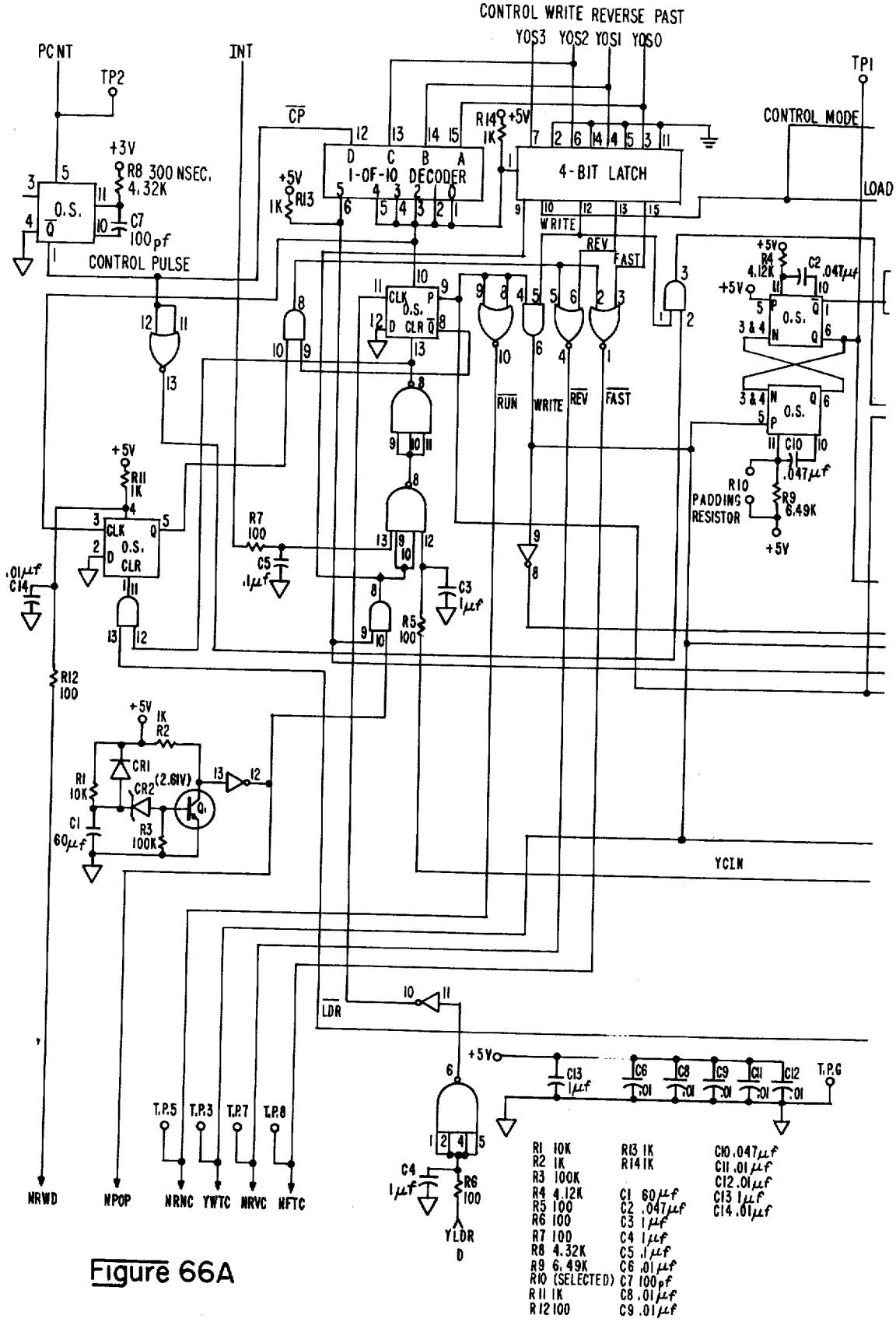

CONTROL WRITE REVERSE PAST
YOS3 YOS2 YOSI YOSO
PCNT
INT
TP2
TP1
CONTROL MODE
LOAD
+3V
R8 300 NSEC.
4.32K
O.S.
C7
100pf
CONTROL PULSE
CP
D C B A
1-OF-10 DECODER
R14 +5V
1K
4-BIT LATCH
+5V
R13
1K
WRITE
REV
FAST
RUN
WRITE
REV
FAST
+5V
R4
4.12K
C2 .047μf
O.S.
C10
.047μf
R10
PADDING
RESISTOR
R9
6.49K
+5V
+5V
R11
1K
CLK
D
O.S.
CLR
R7
100
C5
.1μf
.01μf
C14
R5
100
C3
1μf
R12
100
+5V
1K
R2
CR1
CR2
(2.61V)
R1
10K
R3
100K
C1
60μf
YCIN
LDR
+5V
C13
1μf
C6
.01
C8
.01
C9
.01
C11
.01
C12
.01
T.P.G
T.P.5
T.P.3
T.P.7
T.P.8
C4
1μf
R6
100
YLDR
D
MRWD
MPOP
MRNC
YWTC
MRVC
MFTC
R1 10K
R2 1K
R3 100K
R4 4.12K
R5 100
R6 100
R7 100
R8 4.32K
R9 6.49K
R10 (SELECTED)
R11 1K
R12 100
R13 1K
R14 1K
C1 60μf
C2 .047μf
C3 1μf
C4 1μf
C5 .1μf
C6 .01μf
C7 100pf
C8 .01μf
C9 .01μf
C10 .047μf
C11 .01μf
C12 .01μf
C13 1μf
C14 .01μf

Figure 66A

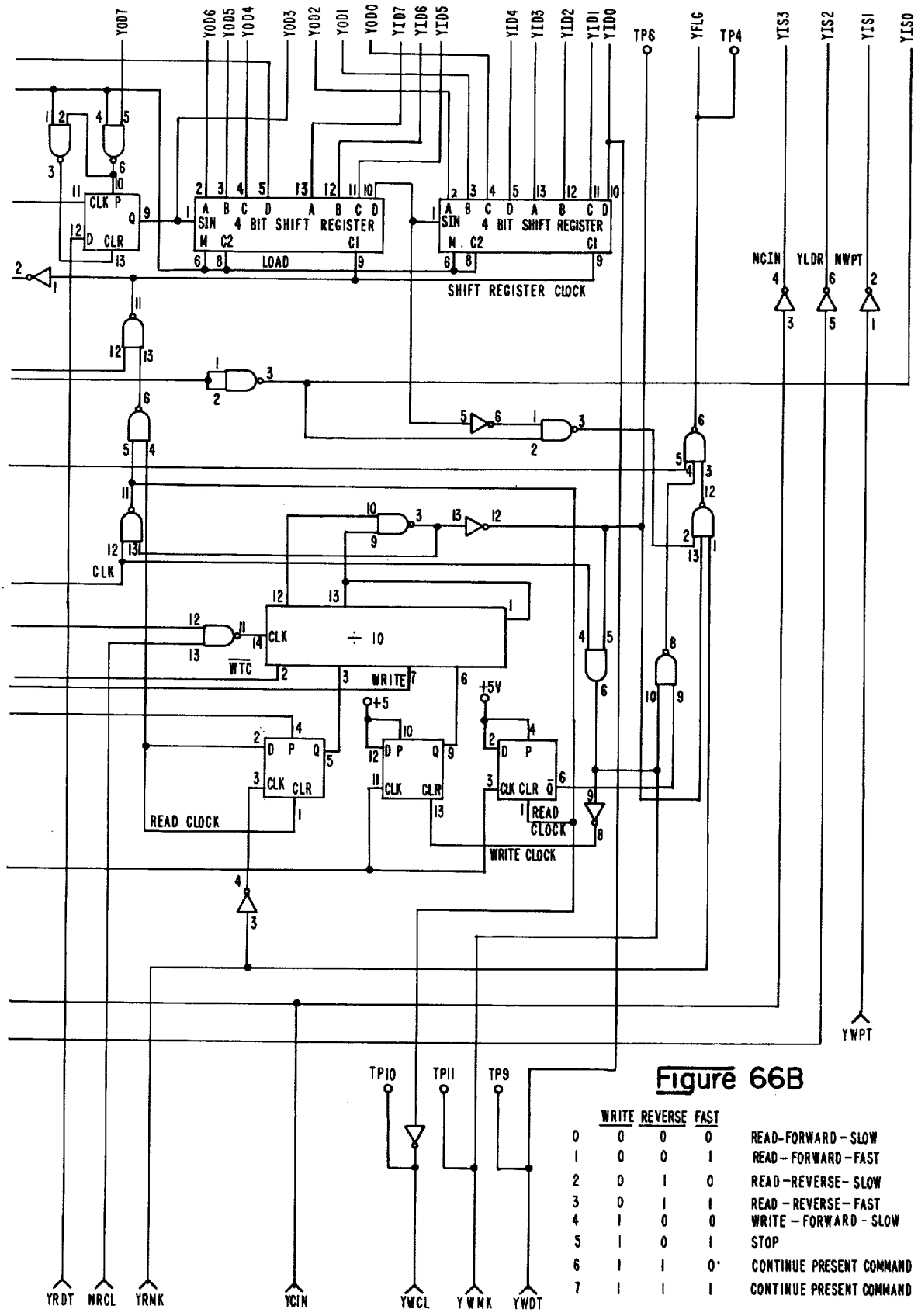


Figure 66B

	WRITE	REVERSE	FAST	
0	0	0	0	READ-FORWARD-SLOW
1	0	0	1	READ-FORWARD-FAST
2	0	1	0	READ-REVERSE-SLOW
3	0	1	1	READ-REVERSE-FAST
4	1	0	0	WRITE-FORWARD-SLOW
5	1	0	1	STOP
6	1	1	0	CONTINUE PRESENT COMMAND
7	1	1	1	CONTINUE PRESENT COMMAND

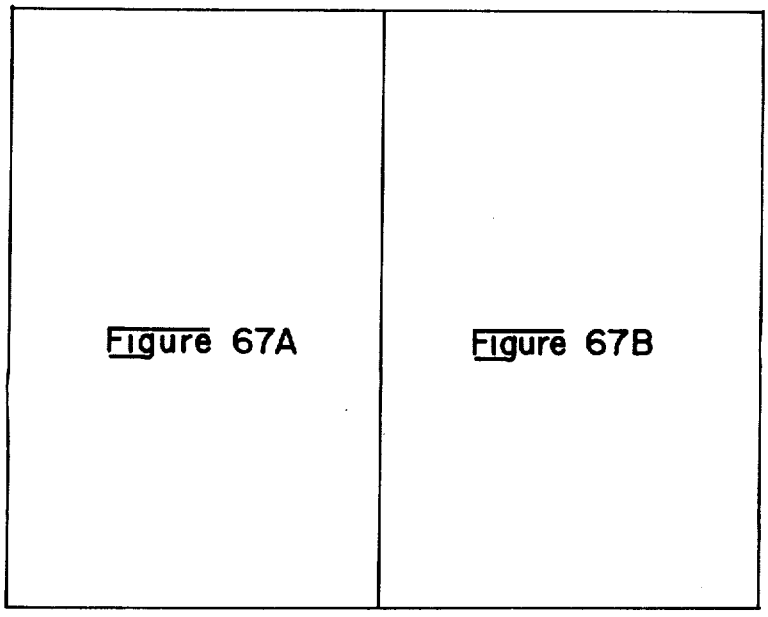


Figure 67'

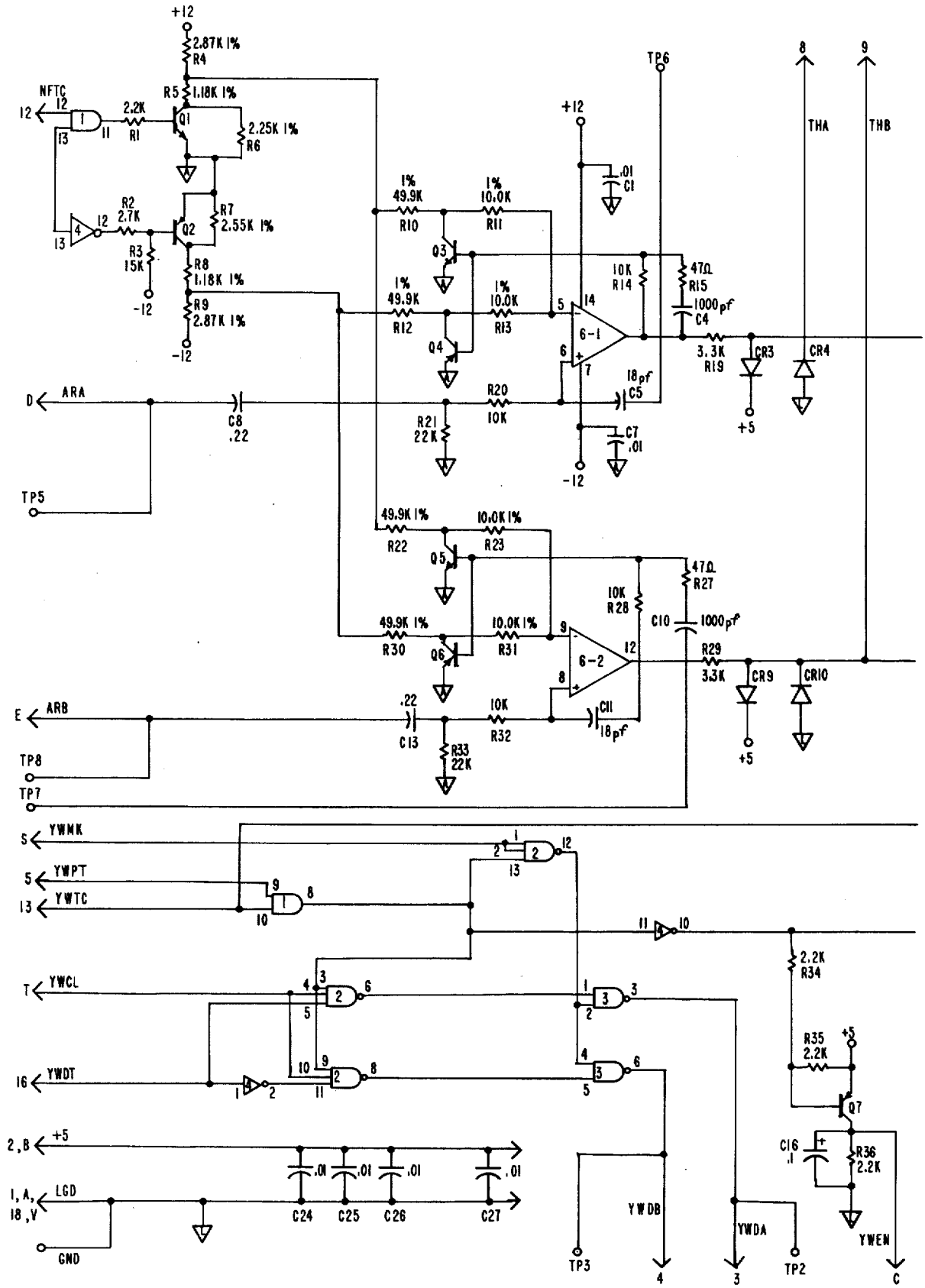

+12
2.87K 1%
R4
NFTC
12
13
11
2.2K
R1
R5 1.18K 1%
Q1
2.25K 1%
R6
R2
2.7K
Q2
R7
2.55K 1%
R3
15K
-12
R8
1.18K 1%
R9
2.87K 1%
-12
1%
49.9K
R10
1%
10.0K
R11
Q3
1%
49.9K
R12
1%
10.0K
R13
Q4
+12
.01
C1
TP6
8
9
THA
THB
10K
R14
47Ω
R15
1000pf
C4
3.3K
R19
CR3
CR4
+5
6-1
18pf
C5
R20
10K
R21
22K
C7
.01
-12
D
ARA
C8
.22
TP5
49.9K 1%
R22
10.0K 1%
R23
Q5
47Ω
R27
10K
R28
C10
1000pf
49.9K 1%
R30
10.0K 1%
R31
Q6
6-2
R29
3.3K
CR9
CR10
+5
E
ARB
.22
C13
10K
R32
C11
18pf
R33
22K
TP8
TP7
5
YWMK
5
YWPT
13
YWTC
T
YWCL
16
YWDT
2.2K
R34
R35
2.2K
+5
Q7
C16
.1
R36
2.2K
2,B
+5
1,A,
18,V
LGD
GND
.01
C24
.01
C25
.01
C26
.01
C27
YWDB
YWDA
YWEN
TP3
TP2
4
3
C

Figure 67A

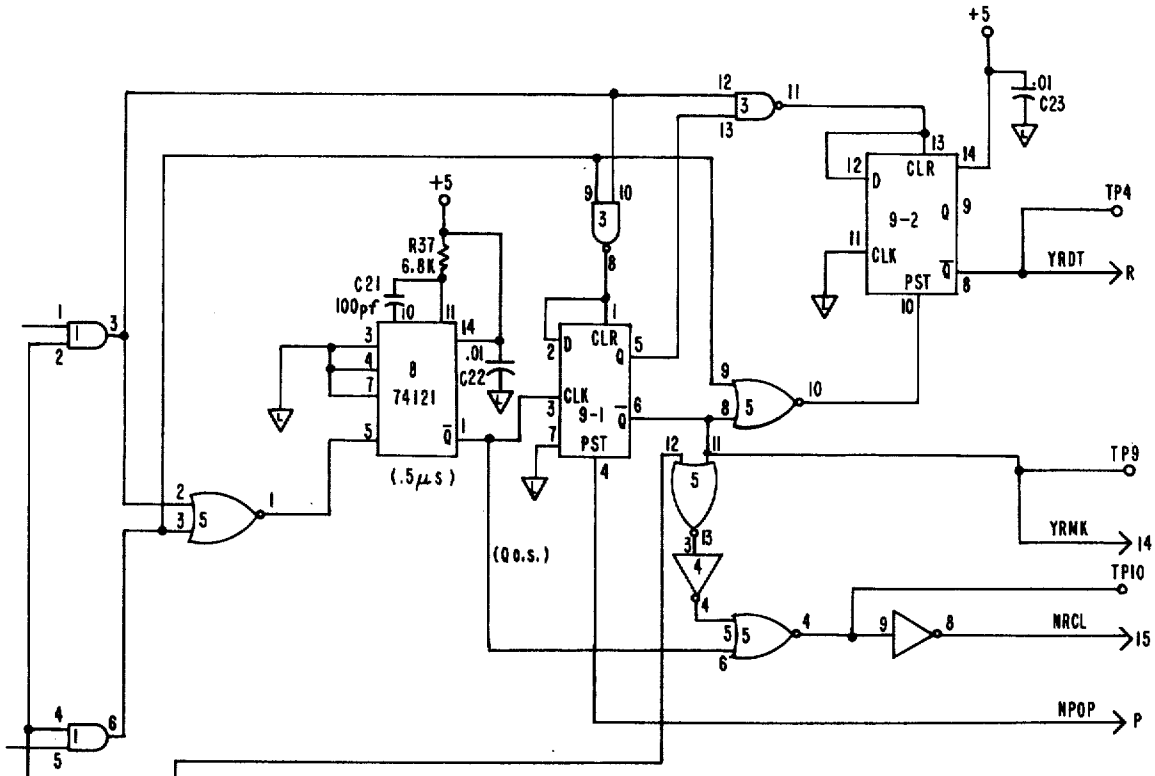
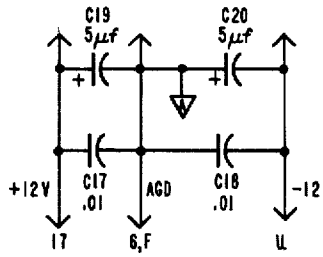


Figure 67B

DESIGNATORS USED: R1-R37, EXC. R16-18, R24-26
Q1-Q7
C1-C27, EXC. C6, 12, 14, 15, 2, 3, 9
CR3, 4, 9, 10

RESISTORS 1/4 WATT 10% UNLESS NOTED
DIODES 1901-0040
NPN: 1854-0071
PNP: 1853-0020



9	1	SN7474N
8	1	SN74121
6	1	MC1437
5	1	SN7402
4	1	SN7404N
3	1	SN7400N
2	1	SN7410N
1	1	SN7408N

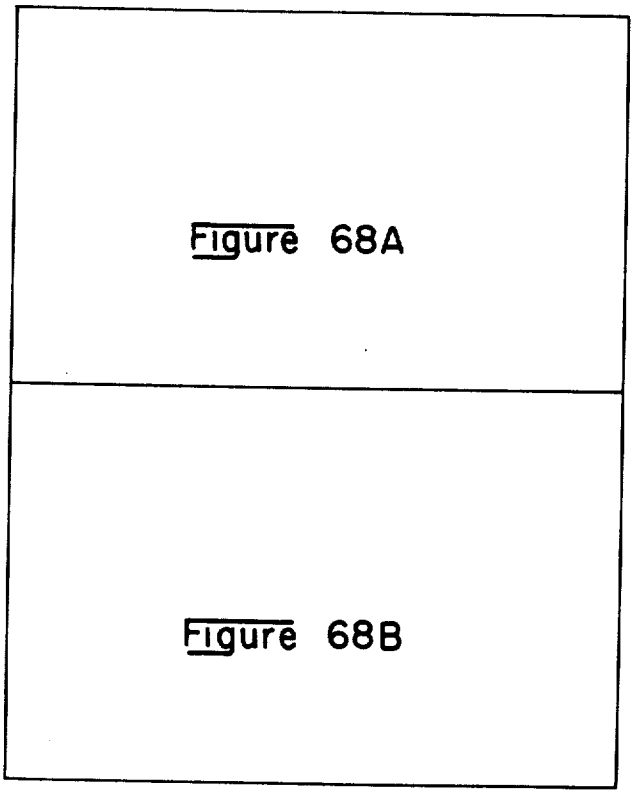


Figure 68'

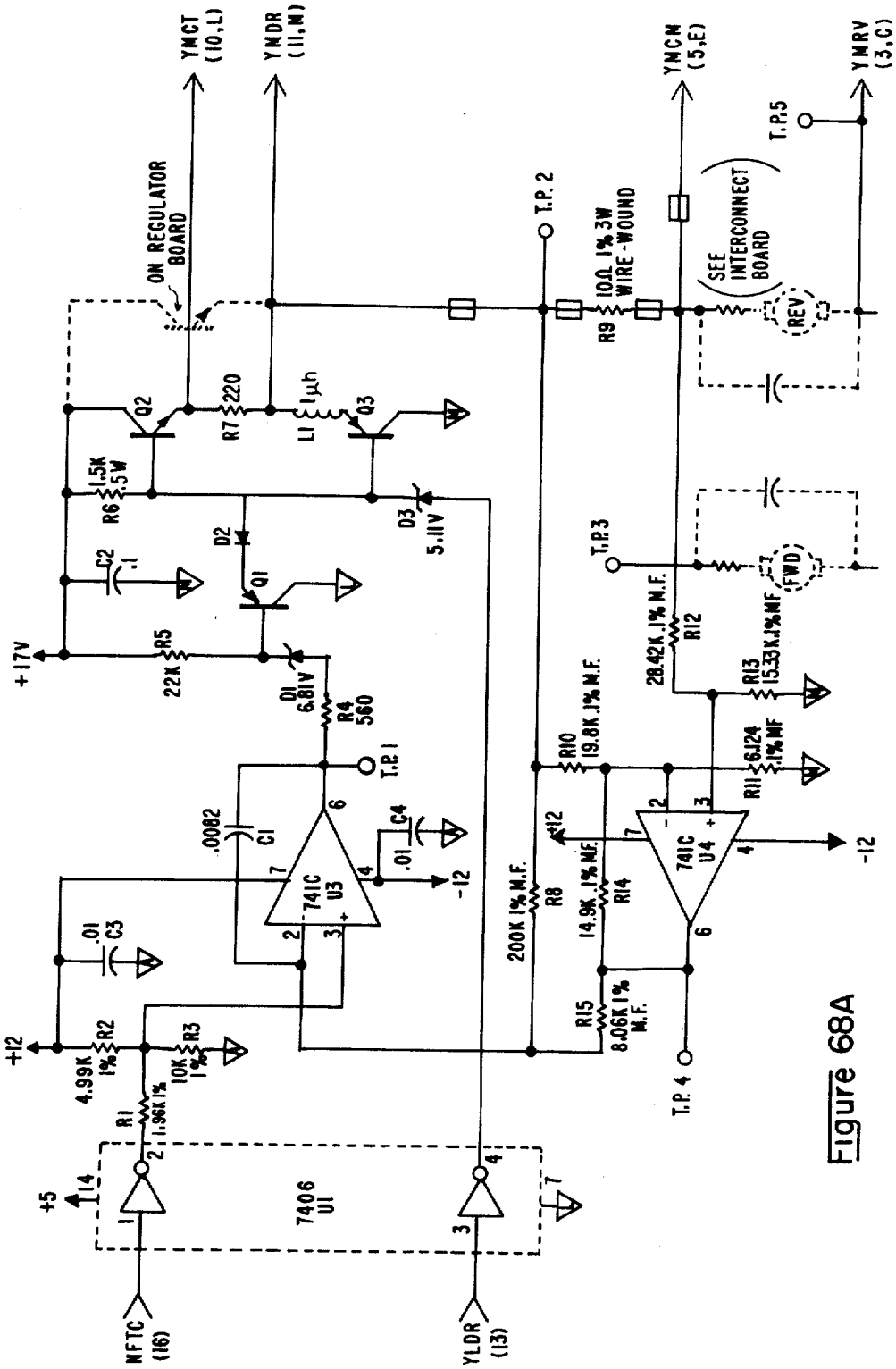

YMCT (10,L)
YMDR (11,M)
YMCM (5,E)
YMRV (3,C)
ON REGULATOR BOARD
T.P. 2
T.P.5
SEE INTERCONNECT BOARD
REV
FWD
R9 10Ω 1% 3W WIRE-WOUND
Q2
R7 220
L1 1μh
Q3
R6 1.5K .5W
D2
Q1
D3 5.11V
C2 .1
T.P.3
+17V
R5 22K
D1 6.81V
R4 560
T.P.1
R12 28.42K .1% N.F.
R13 15.33K .1% MF
R10 19.8K .1% N.F.
R11 6.124 .1% MF
.0082 C1
C4 .01
741C U3
741C U4
-12
+12
200K 1% N.F. R8
14.9K .1% MF R14
C3 .01
R15 8.06K 1% M.F.
T.P. 4
+12
4.99K 1% R2
10K 1% R3
R1 11.96K 1%
+5
14
7406 U1
7
NFTC (16)
YLDR (13)

Figure 68A

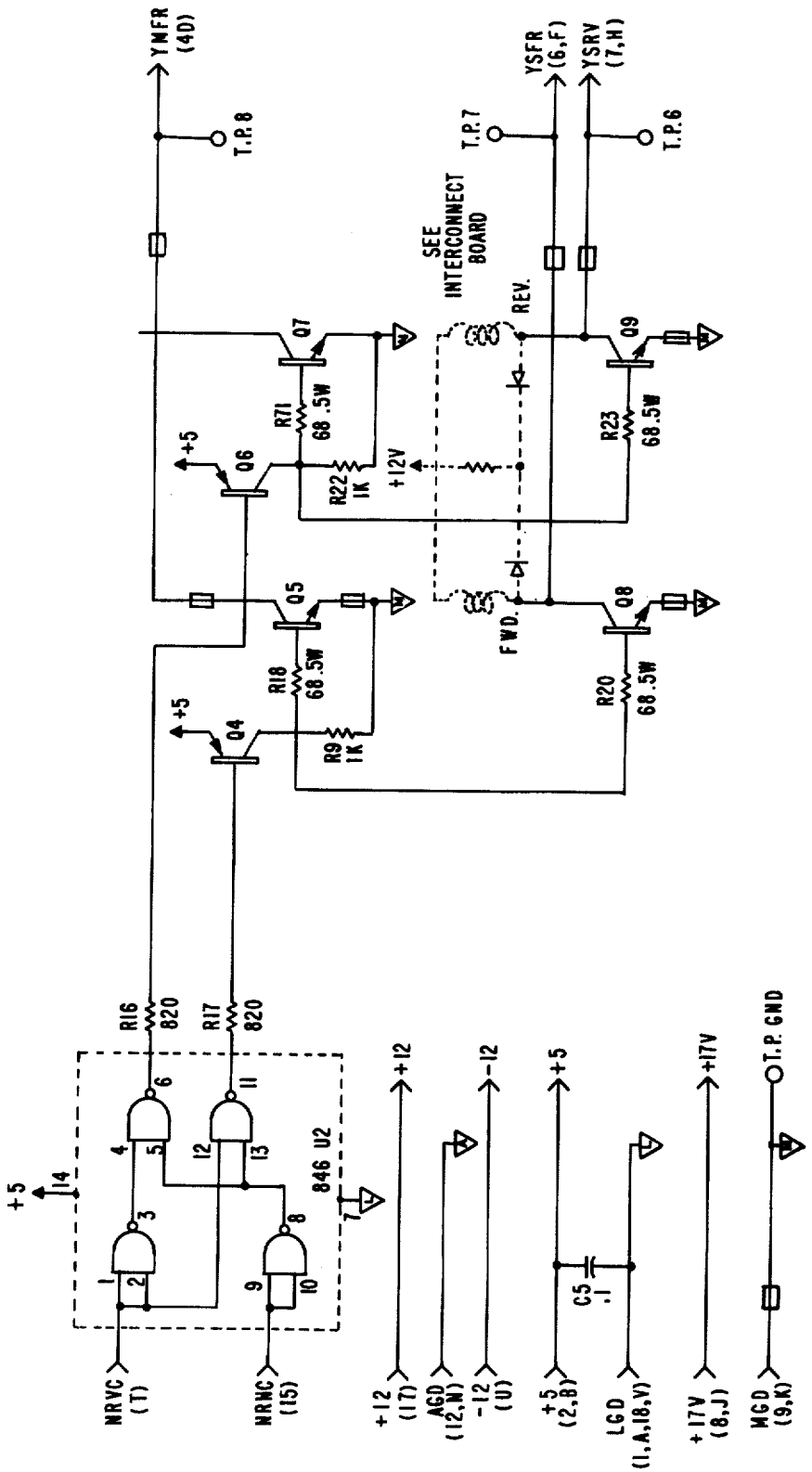
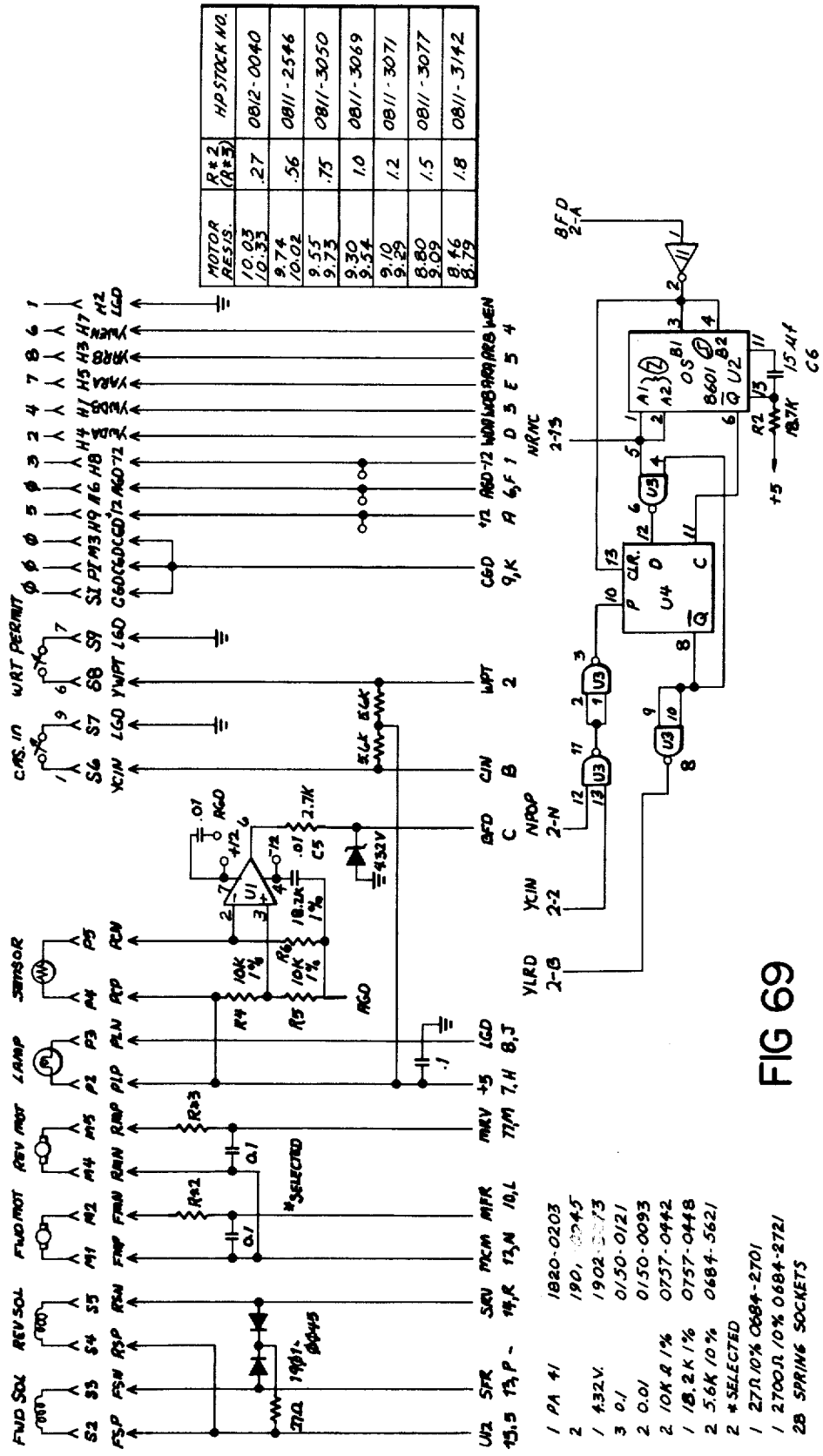

YMFR (4D)
T.P.8
YSFR (6,F)
YSRV (7,H)
T.P.7
T.P.6
SEE INTERCONNECT BOARD
REV.
FWD.
Q7
Q9
Q6
Q5
Q8
Q4
R71 68.5W
R22 1K
R23 68.5W
R18 68.5W
R9 1K
R20 68.5W
+5
+12V
R16 820
R17 820
846 U2
NRVC (T)
NRNC (15)
+12 (17)
AGD (12,N)
-12 (U)
+5 (2,B)
LGD (1,A,18,V)
+17V (8,J)
T.P. GND
MGD (9,K)
C5 .1

Figure 68B


MOTOR RESIS.
R*2 (R*3)
HP STOCK NO.
10.03 10.33
.27
0812-0040
9.74 10.02
.56
0811-2546
9.55 9.73
.75
0811-3050
9.30 9.54
1.0
0811-3069
9.10 9.29
1.2
0811-3071
8.80 9.09
1.5
0811-3077
8.46 8.79
1.8
0811-3142
FIG 69

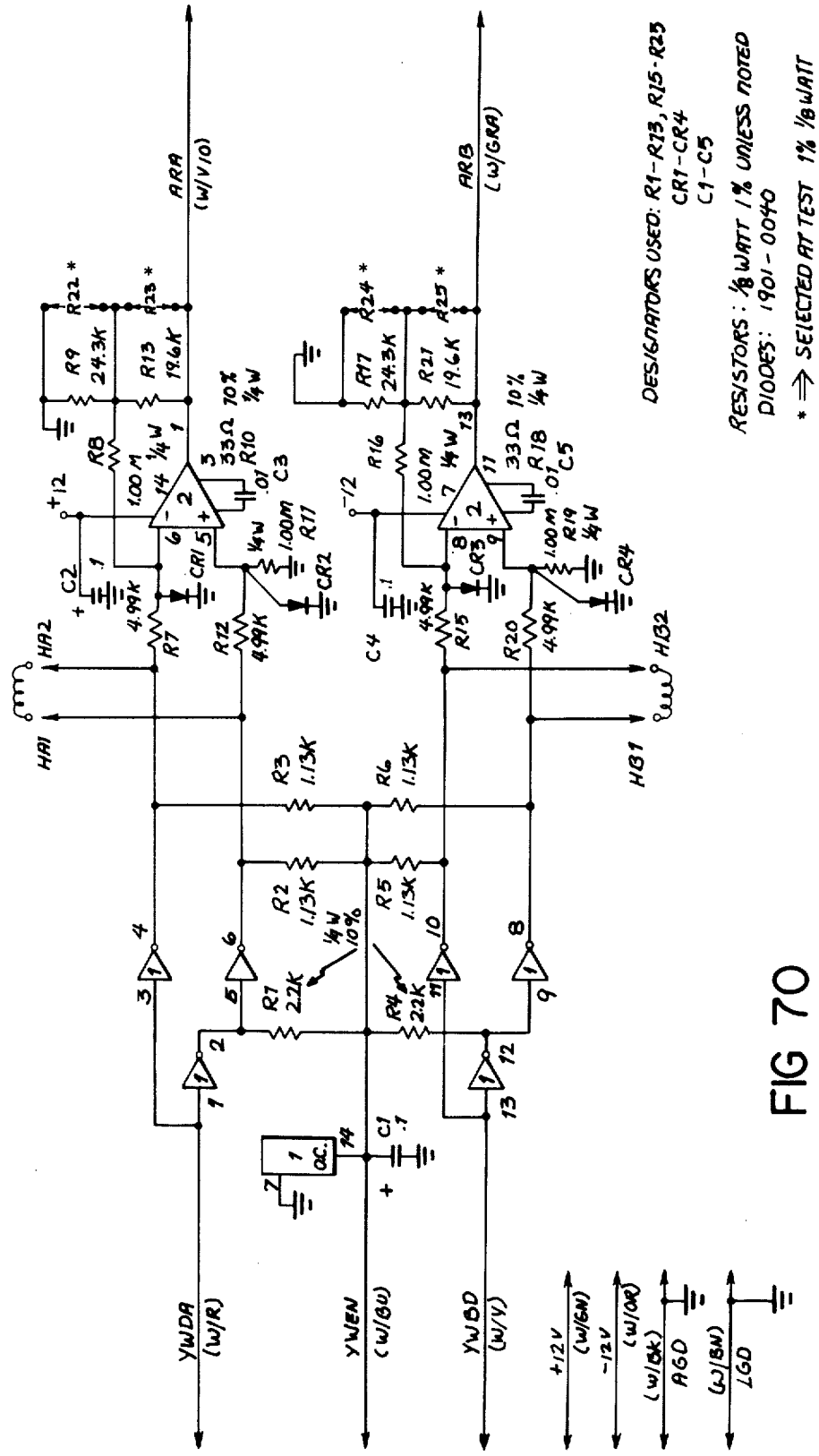

DESIGNATORS USED: R1-R13, R15-R25
CR1-CR4
C1-C5
RESISTORS: 1/8 WATT 1% UNLESS NOTED
DIODES: 1901-0040
* ⇒ SELECTED AT TEST 1% 1/8 WATT
ARA (W/V10)
ARB (W/GRA)
HA1
HA2
HB1
HB2
YWDA (W/R)
YWEN (W/BU)
YWBD (W/Y)
+12V (W/BN)
-12V (W/OR)
AGD (W/BK)
LGD (U/BN)

FIG 70

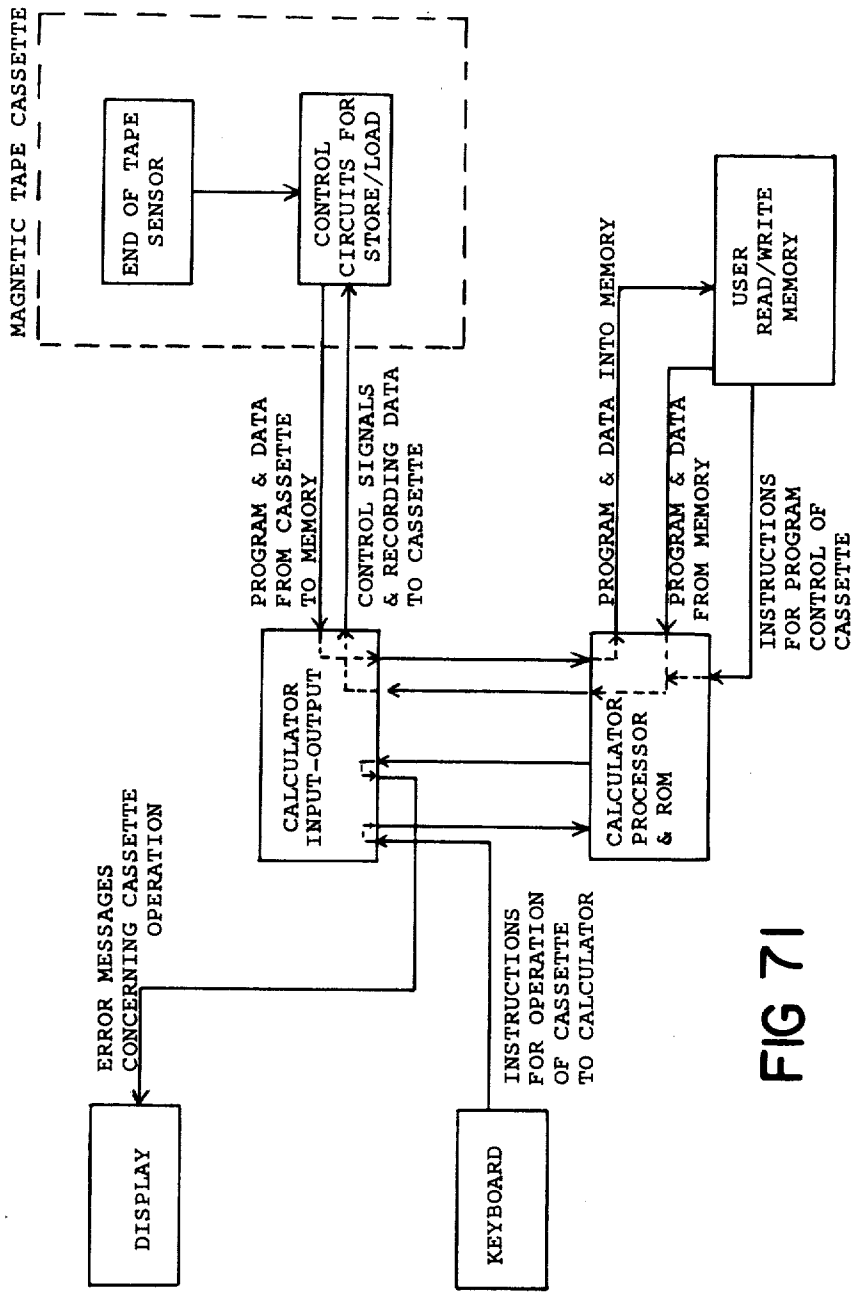

MAGNETIC TAPE CASSETTE
END OF TAPE SENSOR
CONTROL CIRCUITS FOR STORE/LOAD
PROGRAM & DATA FROM CASSETTE TO MEMORY
CONTROL SIGNALS & RECORDING DATA TO CASSETTE
PROGRAM & DATA INTO MEMORY
USER READ/WRITE MEMORY
PROGRAM & DATA FROM MEMORY
INSTRUCTIONS FOR PROGRAM CONTROL OF CASSETTE
CALCULATOR INPUT-OUTPUT
CALCULATOR PROCESSOR & ROM
ERROR MESSAGES CONCERNING CASSETTE OPERATION
INSTRUCTIONS FOR OPERATION OF CASSETTE TO CALCULATOR
DISPLAY
KEYBOARD

FIG 71

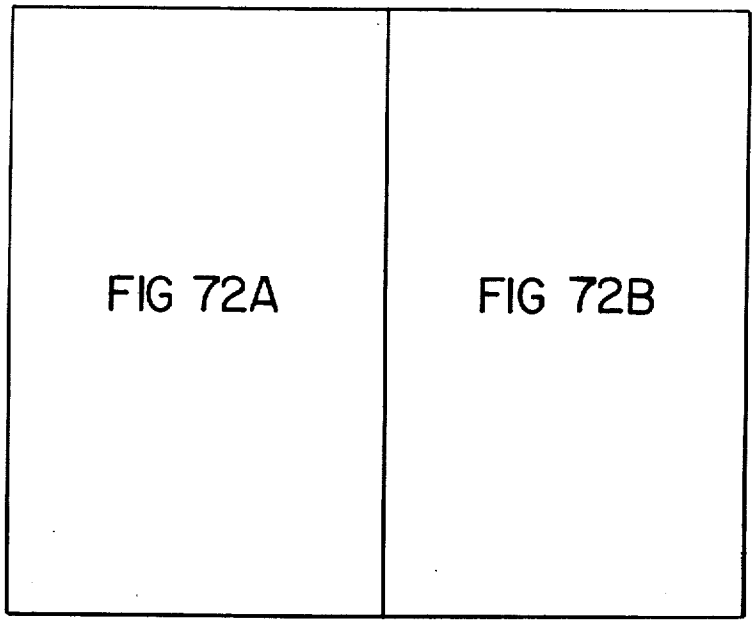


FIG 72'

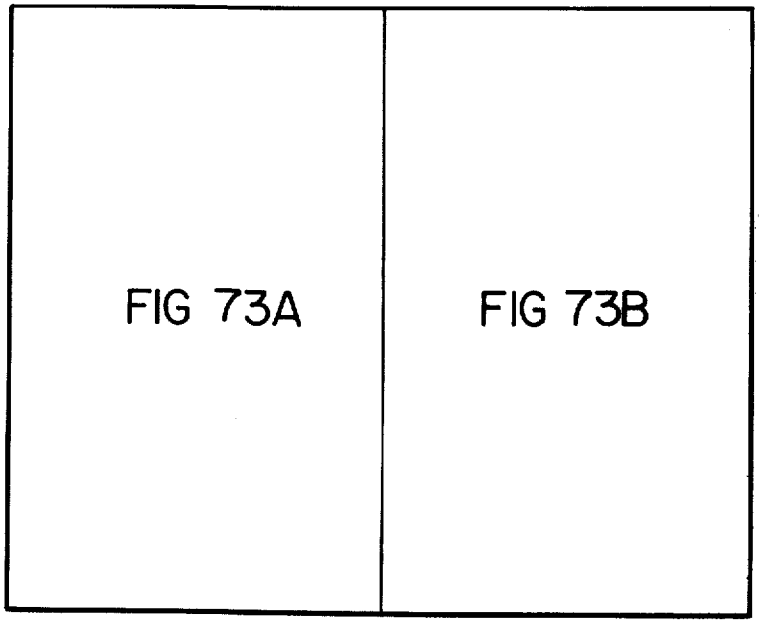


FIG 73'

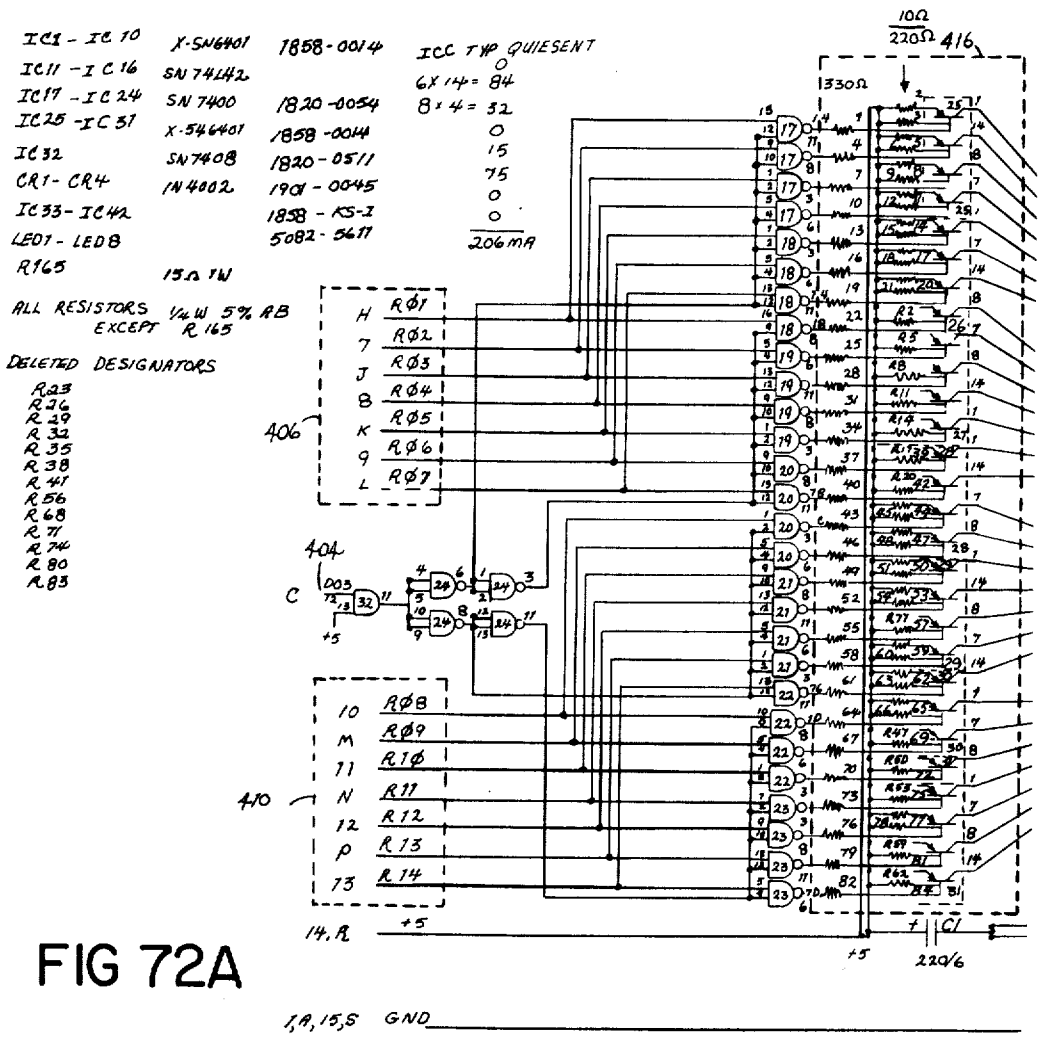

IC1 - IC 10 X-SN6401 1858-0014
IC11 - IC 16 SN 74L42
IC17 - IC 24 SN 7400 1820-0054
IC25 - IC 31 X-546401 1858-0014
IC 32 SN7408 1820-0511
CR1 - CR4 IN4002 1901-0045
IC33 - IC42 1858-KS-1
LED1 - LED8 5082-5611
R165 15Ω 1W
ALL RESISTORS 1/4W 5% AB EXCEPT R 165
DELETED DESIGNATORS
R23 R26 R29 R32 R35 R38 R41 R56 R68 R71 R74 R80 R83
ICC TYP QUIESENT
0
6X14= 84
8X4= 32
0
15
75
0
0
206MA
10Ω 220Ω 416
330Ω
406
404
410
14, R +5
1, A, 15, S GND
220/6

FIG 72A

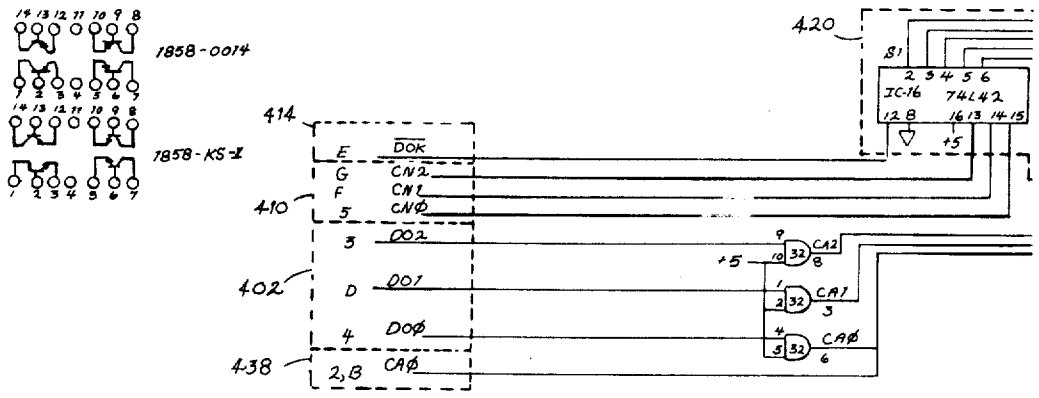
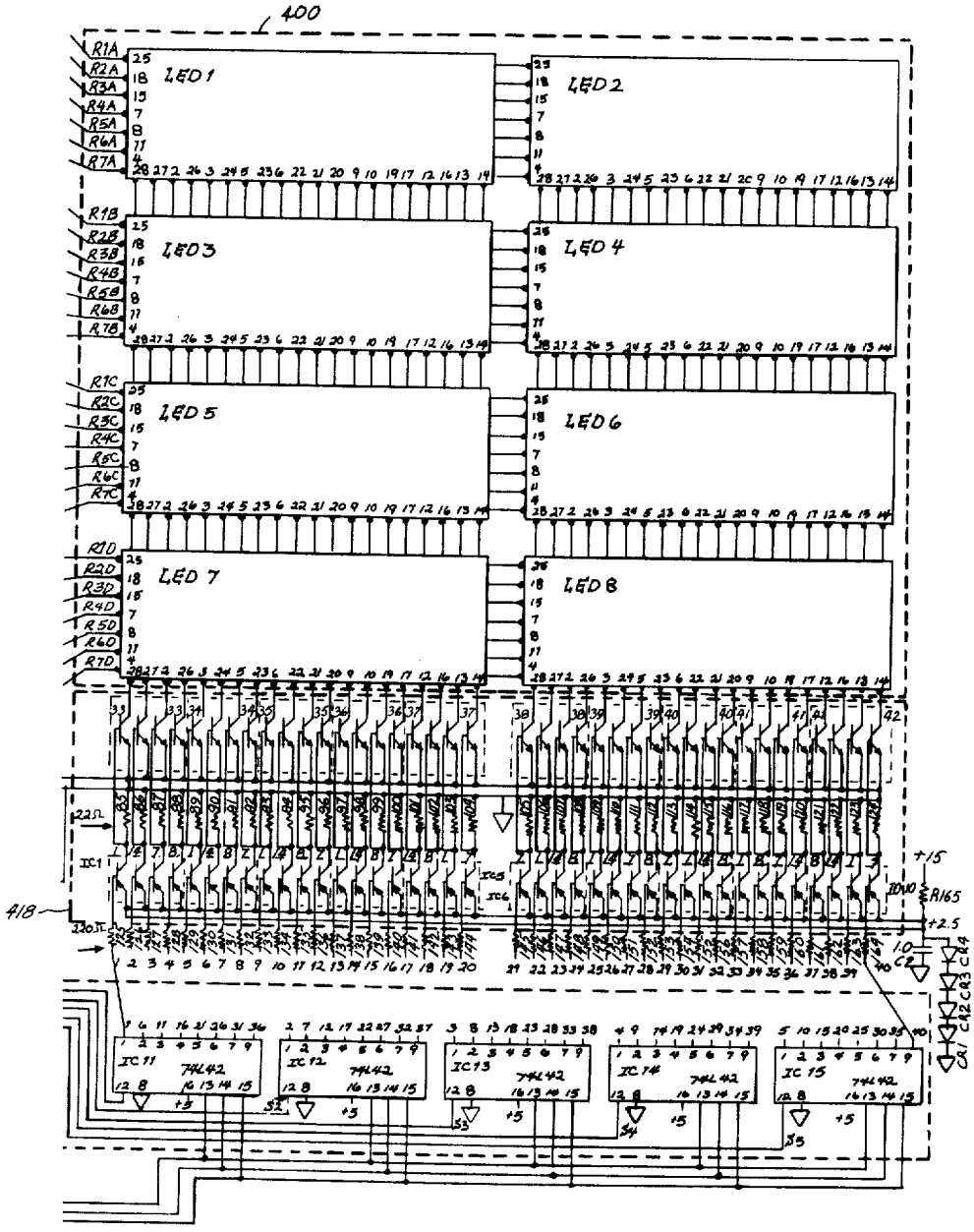

1858-0014
1858-KS-1
414
410
402
438
420
IC-16 74L42
+5

FIG 72B


400
LED 1
LED 2
LED 3
LED 4
LED 5
LED 6
LED 7
LED 8
R1A
R2A
R3A
R4A
R5A
R6A
R7A
R1B
R2B
R3B
R4B
R5B
R6B
R7B
R1C
R2C
R3C
R4C
R5C
R6C
R7C
R1D
R2D
R3D
R4D
R5D
R6D
R7D
22Ω
220Ω
418
IC1
IC5
IC6
IC 11
IC 12
IC 13
IC 14
IC 15
74L42
+5
+15
+2.5
R165
C1
C2
C12
C13
C14
52
53
54
55

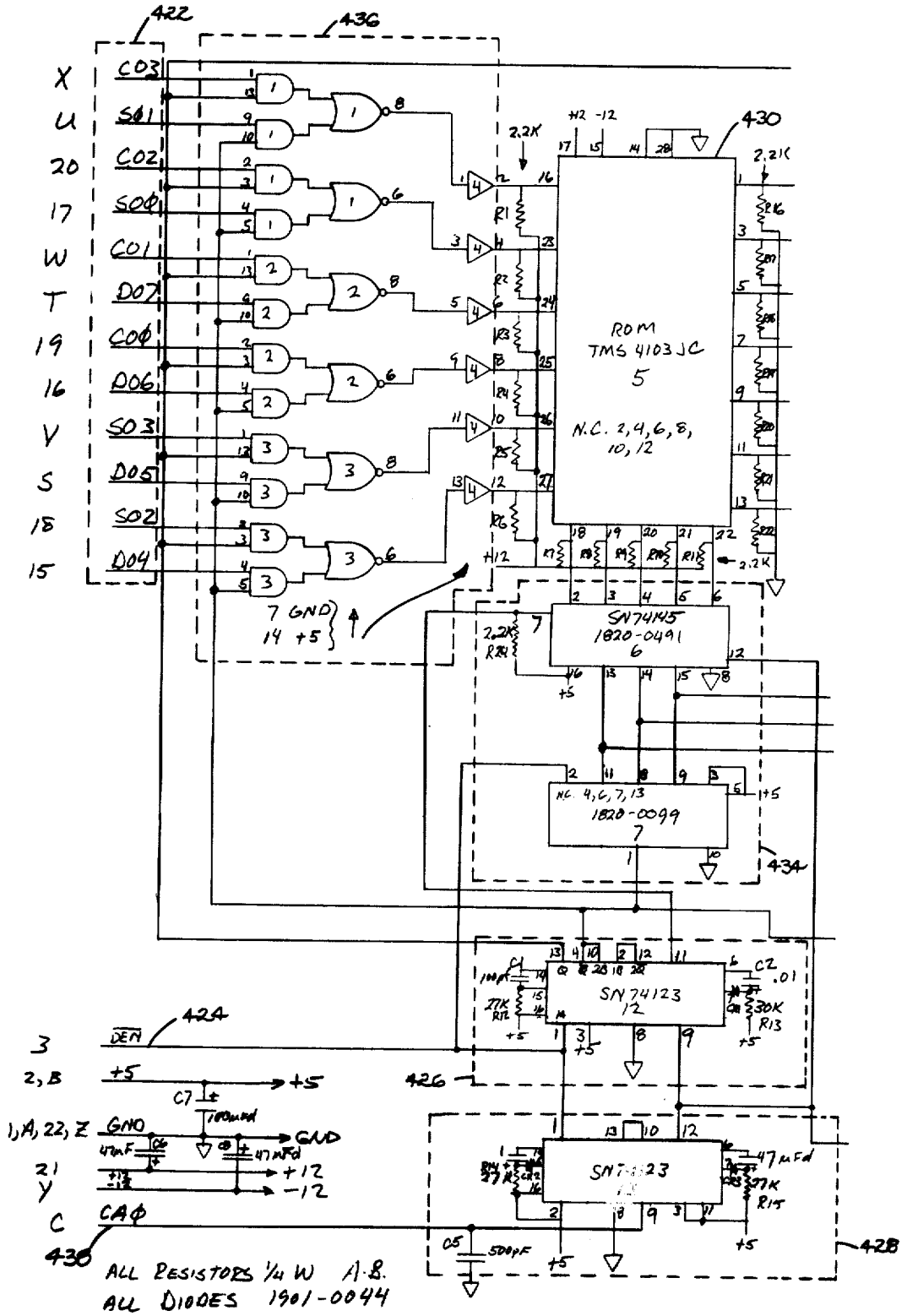
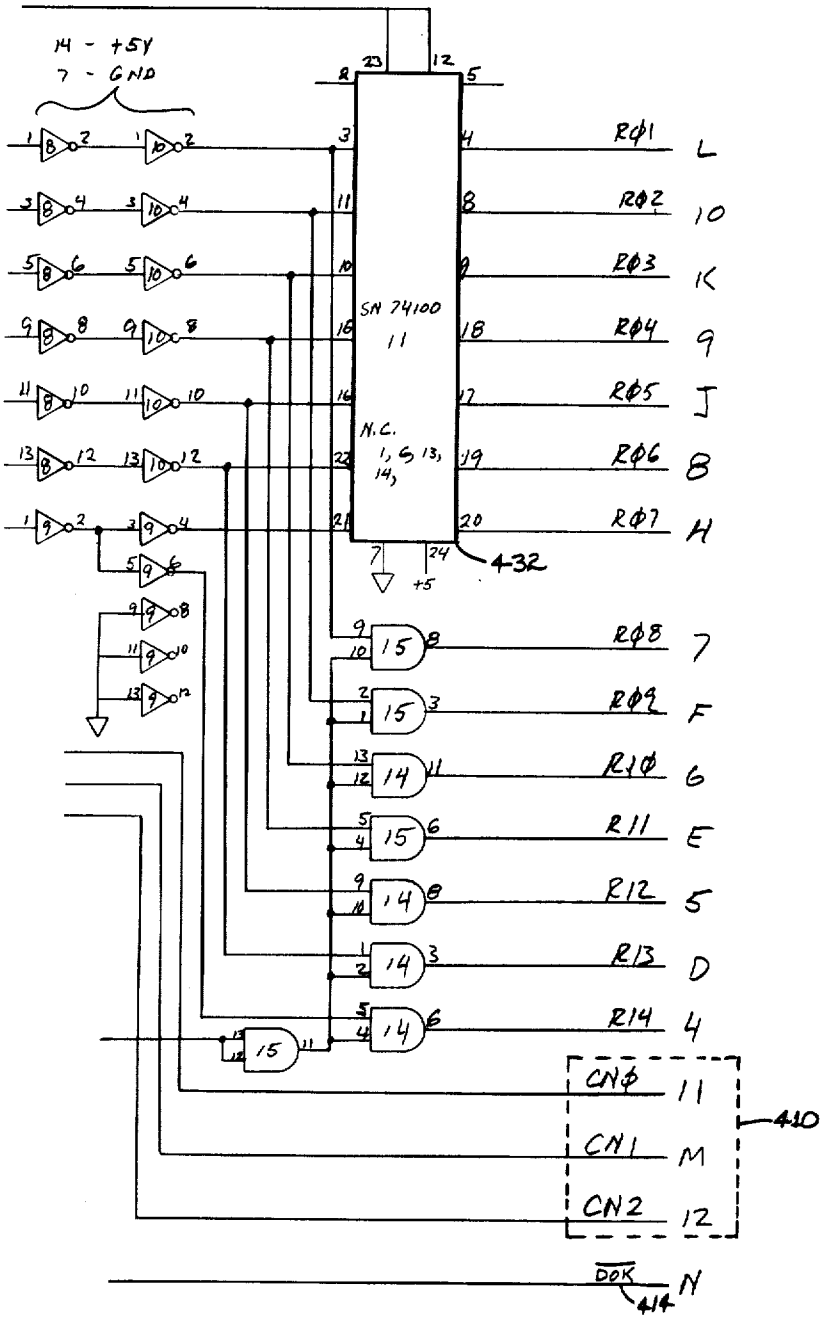

422
436
430
434
426
428
42A
ROM
TMS 4103 JC
5
N.C. 2,4,6,8,
10,12
SN74145
1820-0491
6
1820-0099
7
SN74123
12
7 GND
14 +5
+12 -12
2.2K
ALL RESISTORS 1/4 W 1.8.
ALL DIODES 1901-0044

FIG 73A



IC1, IC2, IC3 SN7451
IC4 SN7407
IC8, IC9 SN74L04
IC10 1820-0174
IC14, IC15 1820-0511

FIG 73B

CALCULATOR POWER SYSTEM BLOCK DIAGRAM

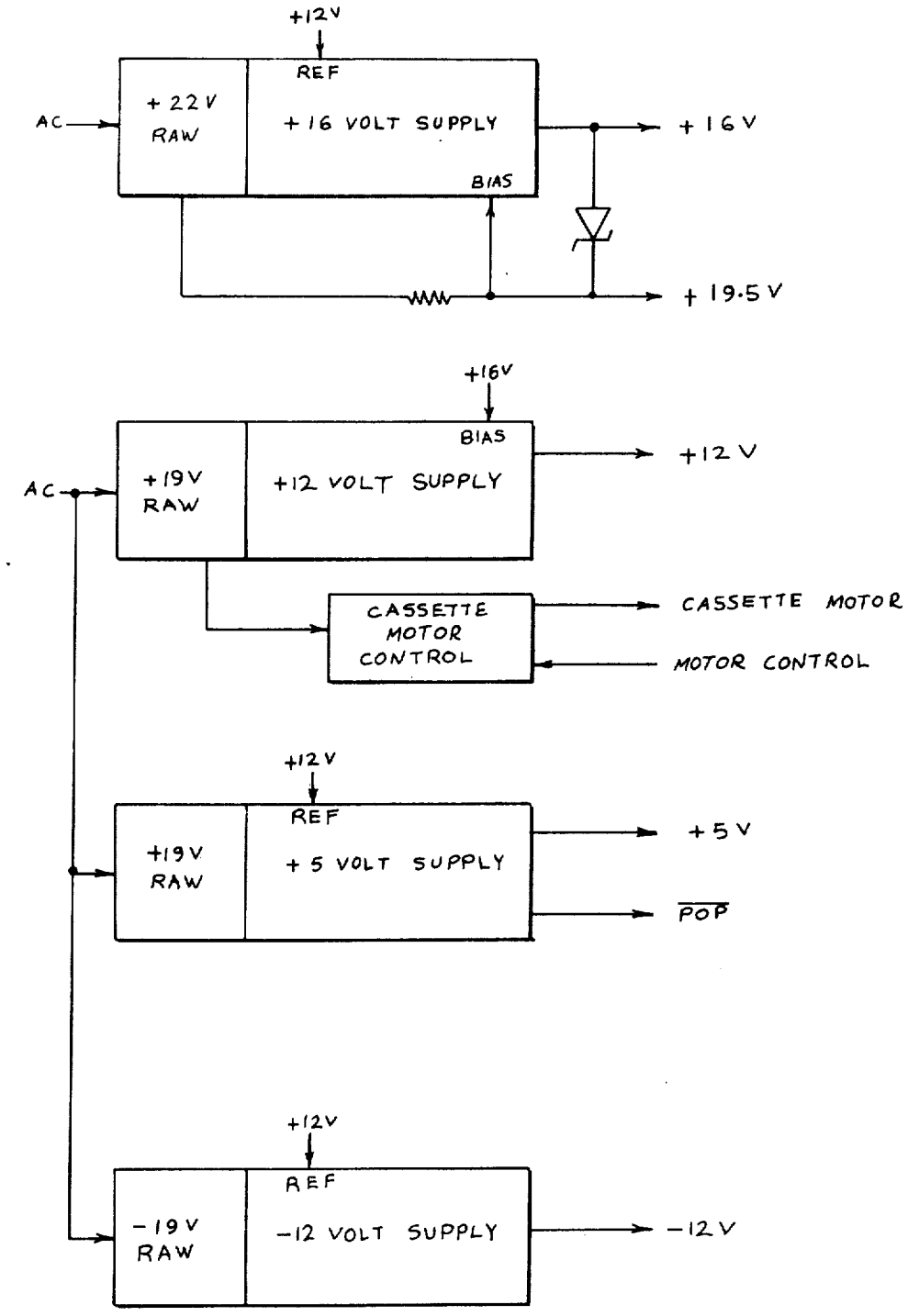


FIG 74

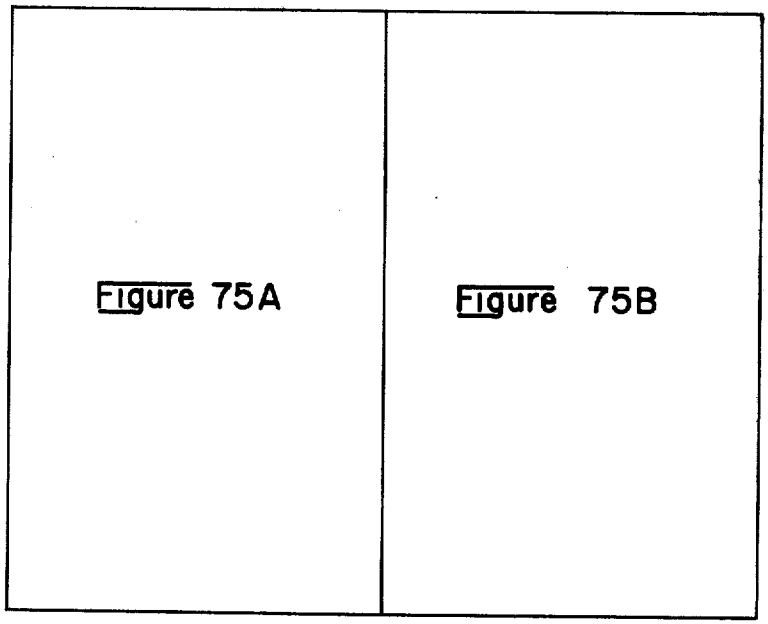


Figure 75'

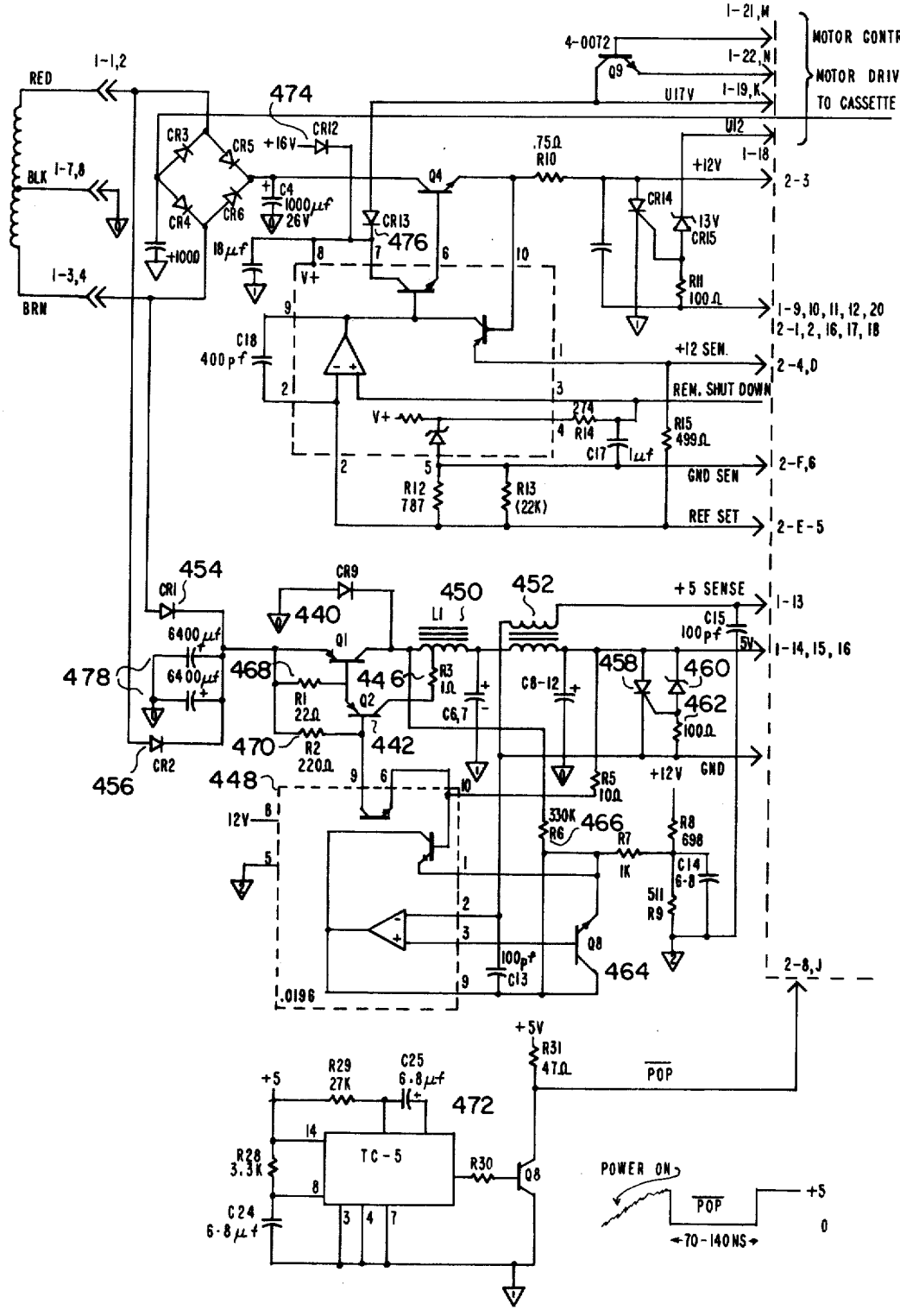

MOTOR CONTROL
MOTOR DRIVE TO CASSETTE
RED
BLK
BRN
+16V
+12V
+12 SEN.
REM. SHUT DOWN
GND SEN
REF SET
+5 SENSE
GND
POP
POWER ON
TC-5
474
476
478
440
442
446
448
450
452
454
456
458
460
462
464
466
468
470
472

Figure 75A

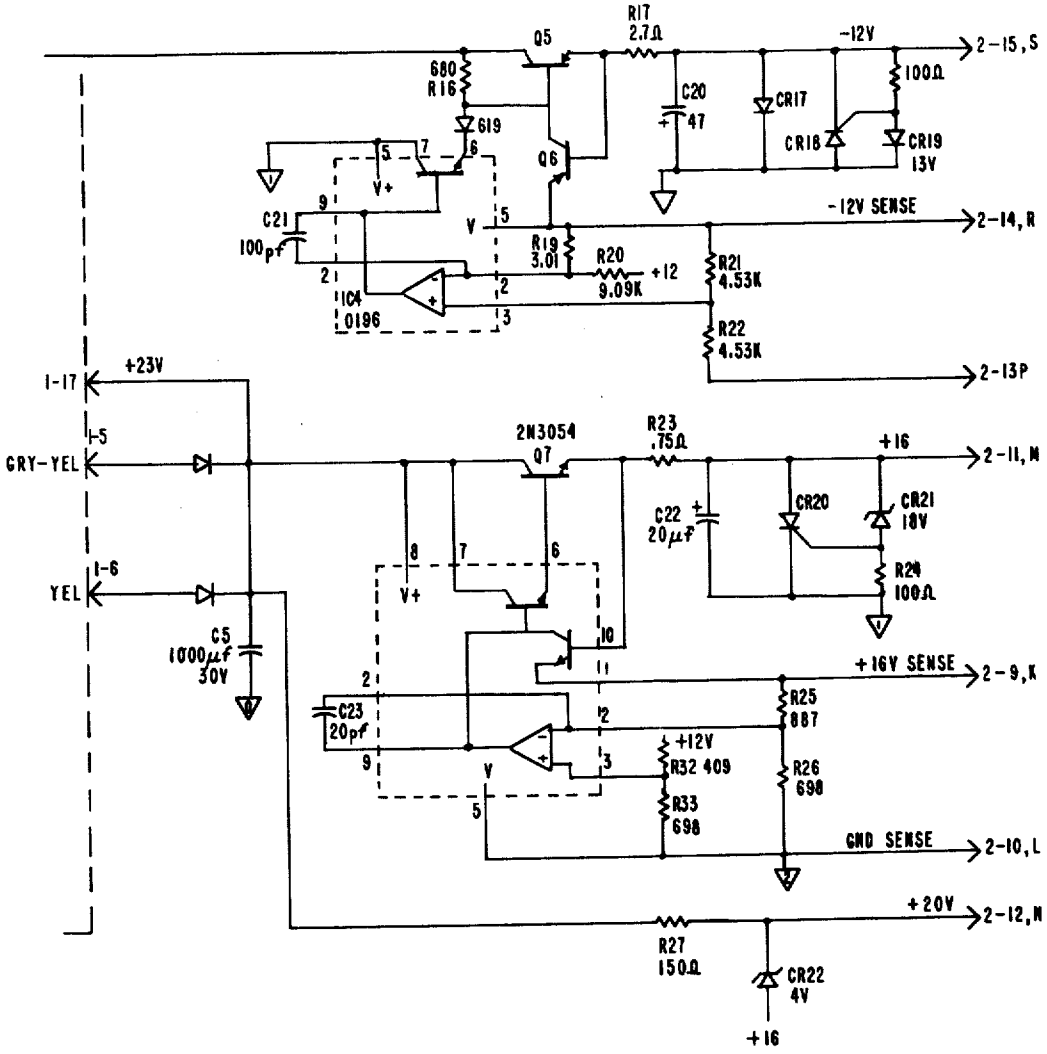

R17
2.7Ω
Q5
-12V
2-15,S
680
R16
C20
47
CR17
100Ω
619
CR18
CR19
13V
Q6
V+
C21
100pf
-12V SENSE
2-14,R
R19
3.01
R20
9.09K
+12
R21
4.53K
IC4
0196
R22
4.53K
2-13P
+23V
1-17
1-5
GRY-YEL
2N3054
Q7
R23
.75Ω
+16
2-11,N
C22
20μf
CR20
CR21
18V
1-6
YEL
R24
100Ω
C5
1000μf
30V
+16V SENSE
2-9,K
C23
20pf
R25
887
+12V
R32 409
R26
698
R33
698
GND SENSE
2-10,L
+20V
2-12,N
R27
150Ω
CR22
4V
+16

Figure 75B

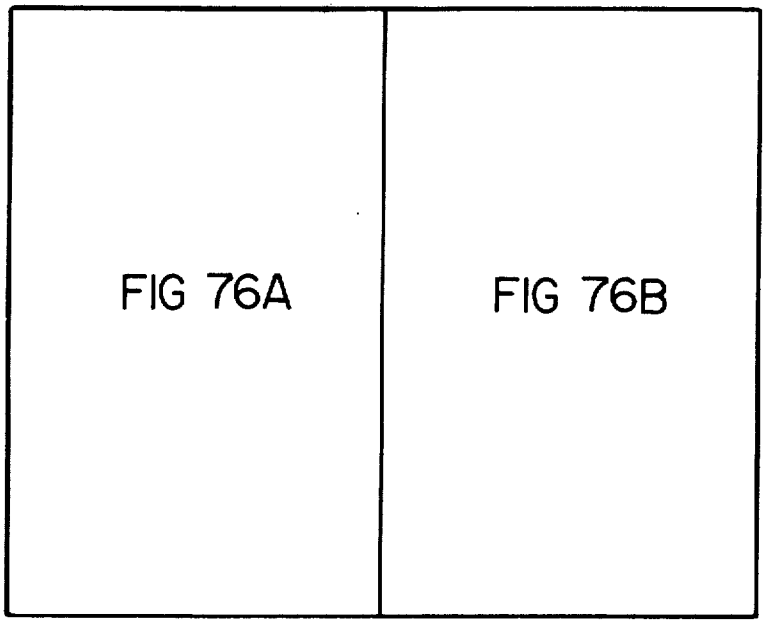


FIG 76'

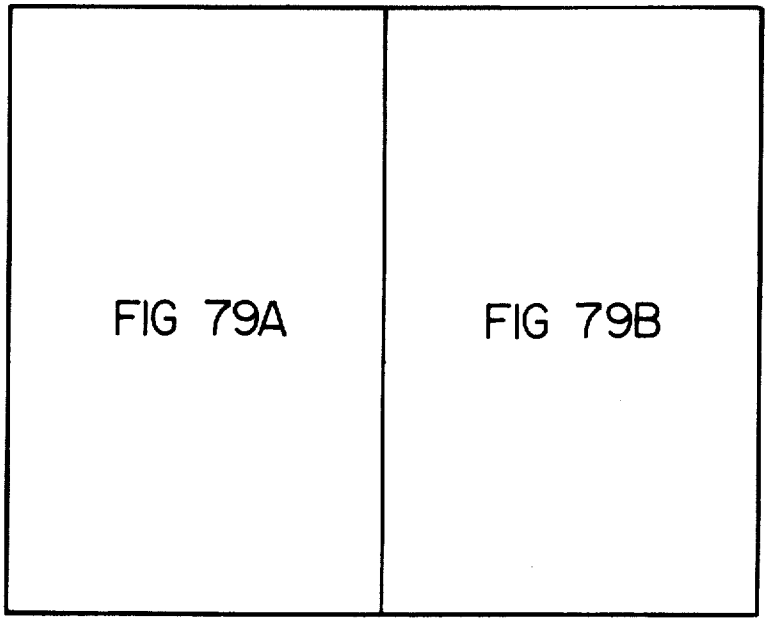


FIG 79'

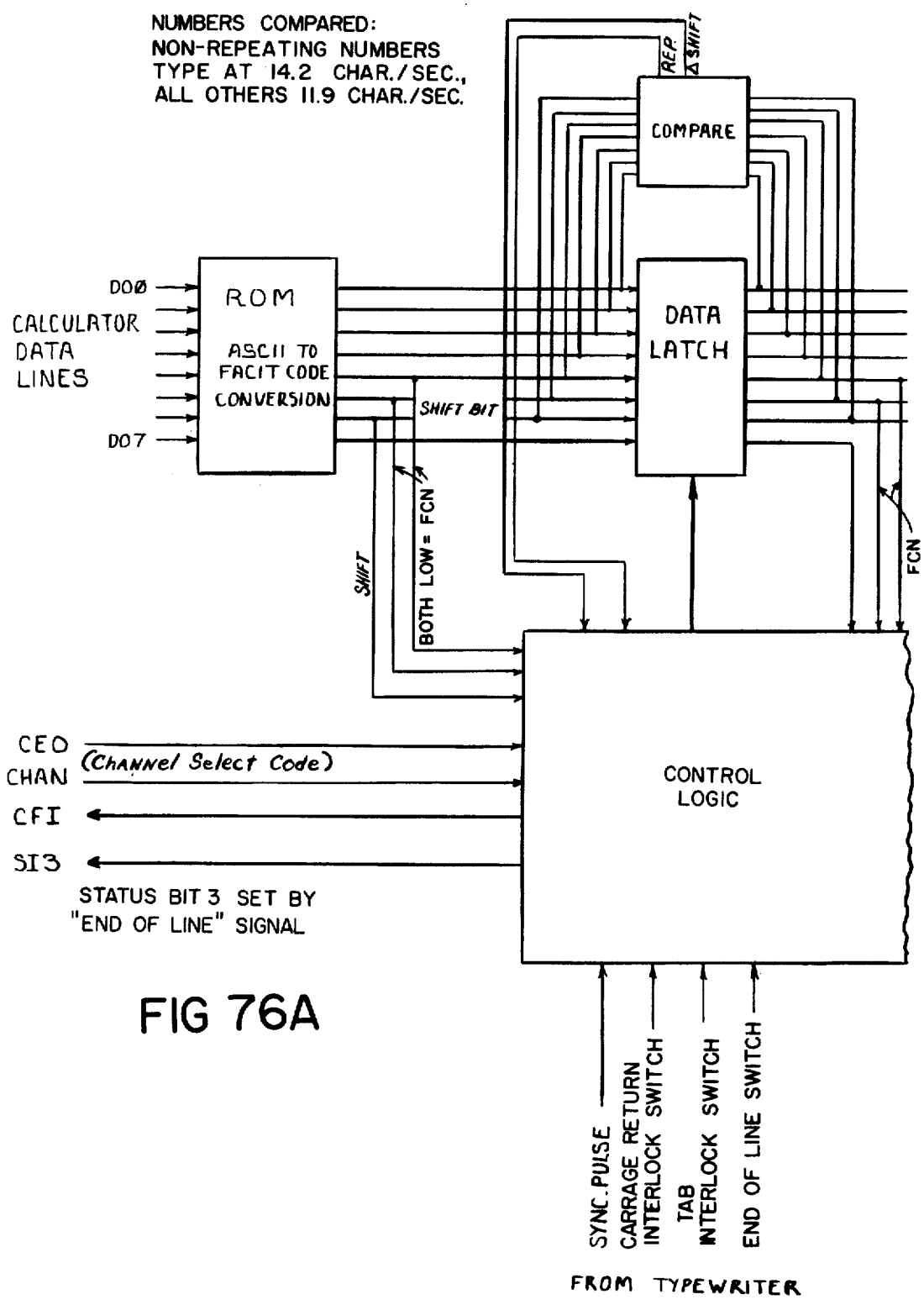

NUMBERS COMPARED:
NON-REPEATING NUMBERS
TYPE AT 14.2 CHAR./SEC.,
ALL OTHERS 11.9 CHAR./SEC.
REP
Δ SHIFT
COMPARE
DATA
LATCH
ROM
ASCII TO
FACIT CODE
CONVERSION
DOØ
DO7
CALCULATOR
DATA
LINES
SHIFT BIT
SHIFT
BOTH LOW = FCN
FCN
CEO
CHAN
(Channel Select Code)
CFI
SI3
STATUS BIT 3 SET BY
"END OF LINE" SIGNAL
CONTROL
LOGIC
SYNC. PULSE
CARRAGE RETURN
INTERLOCK SWITCH
TAB
INTERLOCK SWITCH
END OF LINE SWITCH
FROM TYPEWRITER

FIG 76A

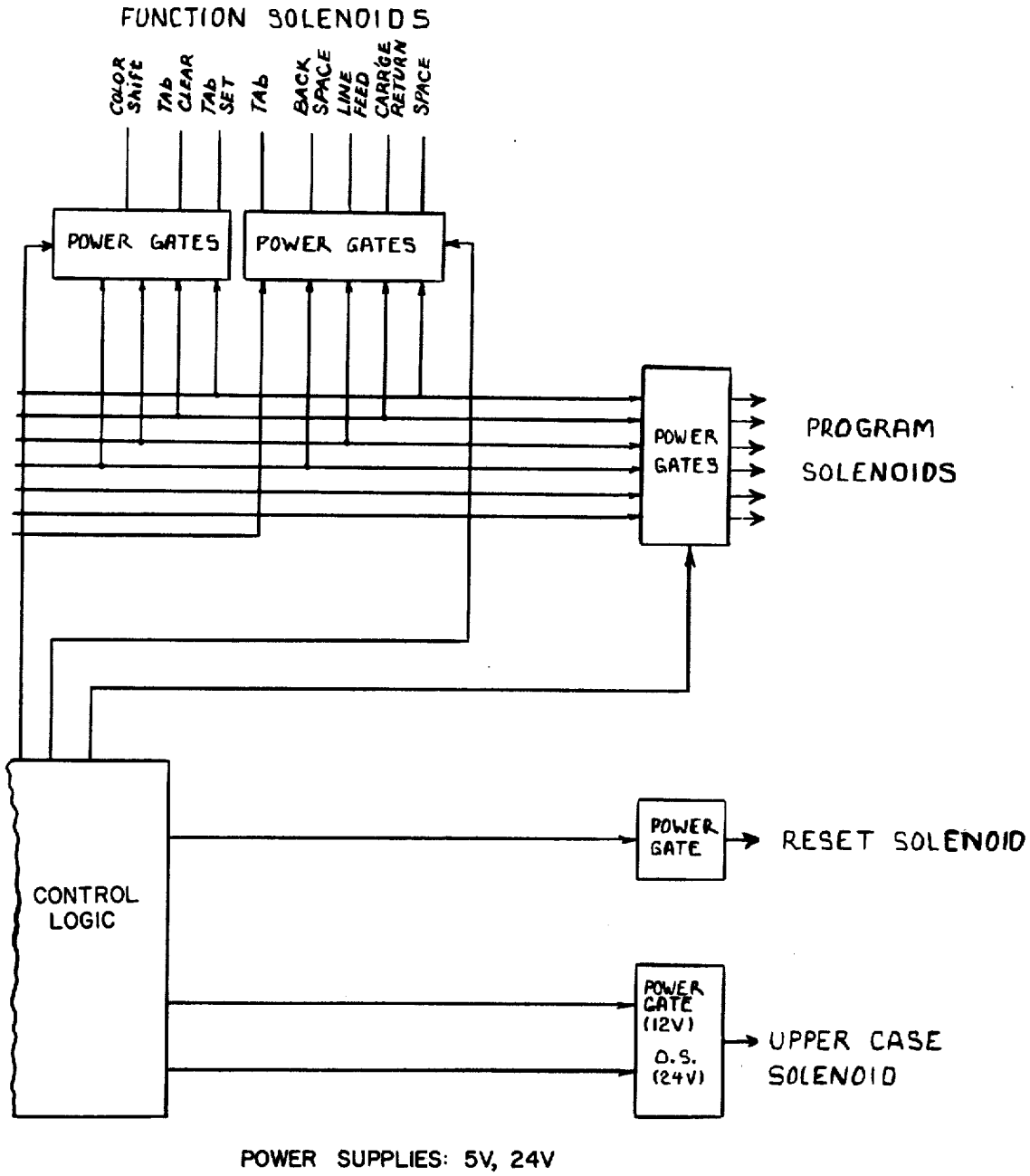

FUNCTION SOLENOIDS
COLOR SHIFT
TAb CLEAR
TAb SET
TAb
BACK SPACE
LINE FEED
CARRGE RETURN
SPACE
POWER GATES
POWER GATES
POWER GATES
PROGRAM SOLENOIDS
CONTROL LOGIC
POWER GATE
RESET SOLENOID
POWER GATE (12V)
O.S. (24V)
UPPER CASE SOLENOID
POWER SUPPLIES: 5V, 24V

FIG 76B

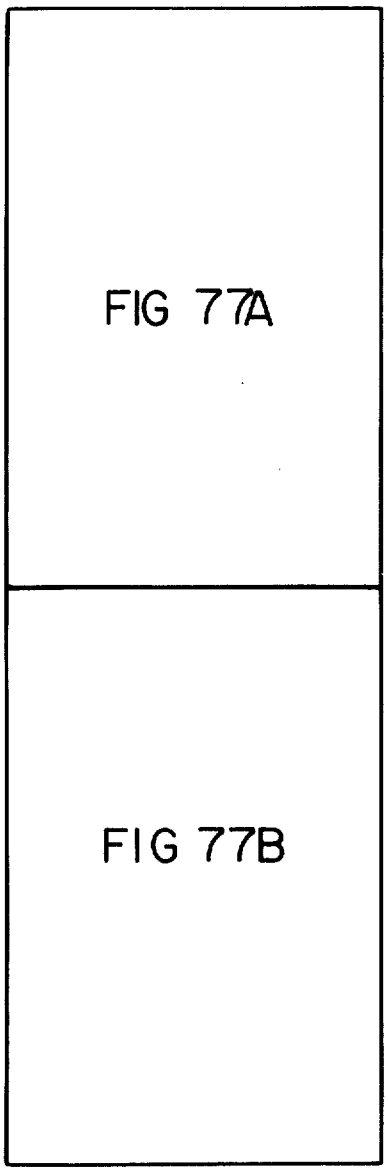


FIG 77'

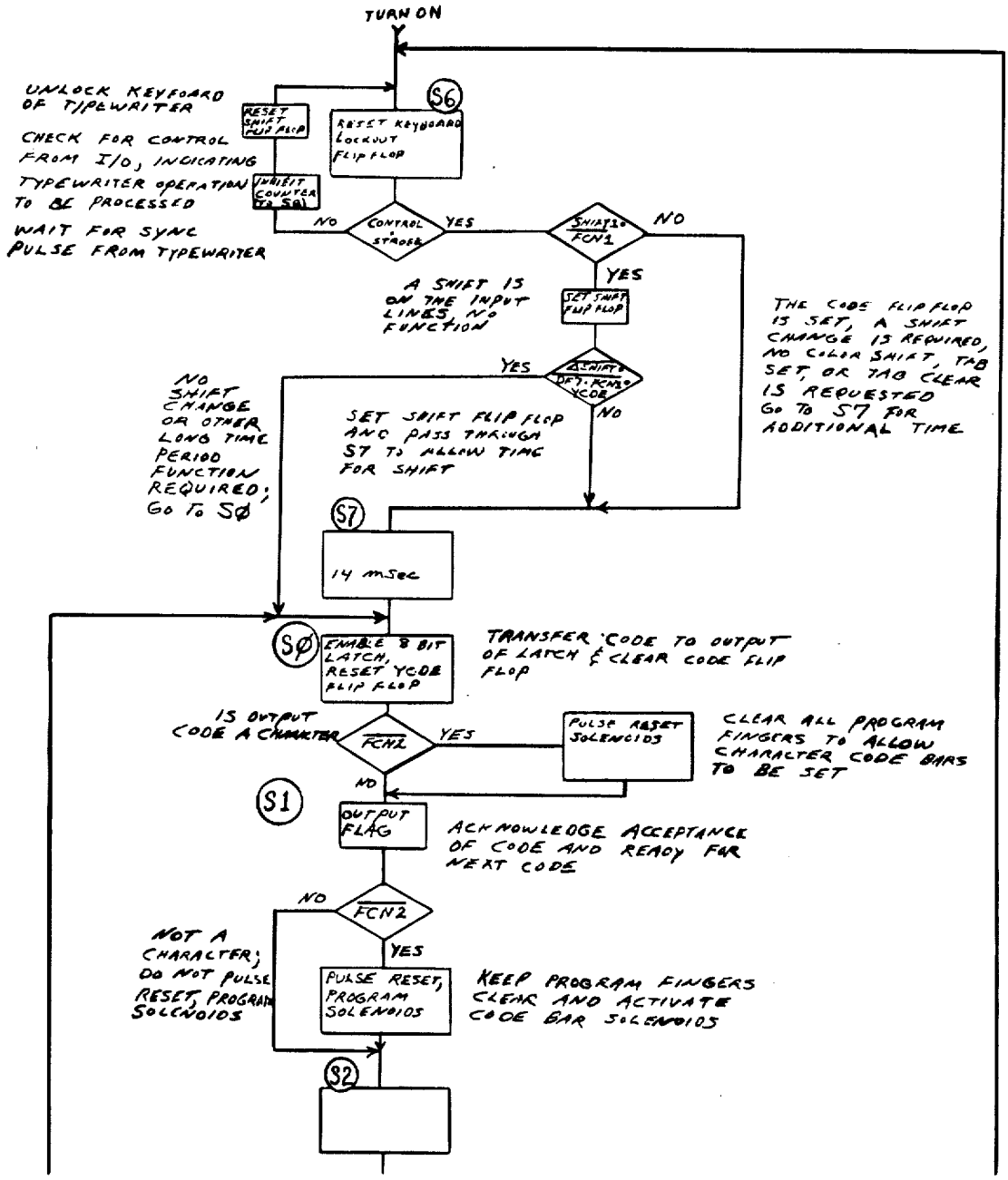

TURN ON
UNLOCK KEYBOARD OF TYPEWRITER
CHECK FOR CONTROL FROM I/O, INDICATING TYPEWRITER OPERATION TO BE PROCESSED
WAIT FOR SYNC PULSE FROM TYPEWRITER
RESET SHIFT FLIP FLOP
INHIBIT COUNTER TO S01
S6
RESET KEYBOARD LOCKOUT FLIP FLOP
CONTROL STROBE
NO
YES
SHIFT & FCN1
NO
YES
A SHIFT IS ON THE INPUT LINES, NO FUNCTION
SET SHIFT FLIP FLOP
THE CODE FLIP FLOP IS SET, A SHIFT CHANGE IS REQUIRED, NO COLOR SHIFT, TAB SET, OR TAB CLEAR IS REQUESTED GO TO S7 FOR ADDITIONAL TIME
YES
NO
NO SHIFT CHANGE OR OTHER LONG TIME PERIOD FUNCTION REQUIRED; GO TO SØ
SET SHIFT FLIP FLOP AND PASS THROUGH S7 TO ALLOW TIME FOR SHIFT
S7
14 mSec
SØ
ENABLE 8 BIT LATCH, RESET YCODE FLIP FLOP
TRANSFER CODE TO OUTPUT OF LATCH & CLEAR CODE FLIP FLOP
IS OUTPUT CODE A CHARACTER
FCN1
YES
PULSE RESET SOLENOIDS
CLEAR ALL PROGRAM FINGERS TO ALLOW CHARACTER CODE BARS TO BE SET
NO
S1
OUTPUT FLAG
ACKNOWLEDGE ACCEPTANCE OF CODE AND READY FOR NEXT CODE
NO
FCN2
YES
NOT A CHARACTER; DO NOT PULSE RESET, PROGRAM SOLENOIDS
PULSE RESET, PROGRAM SOLENOIDS
KEEP PROGRAM FINGERS CLEAR AND ACTIVATE CODE BAR SOLENOIDS
S2

FIG 77A

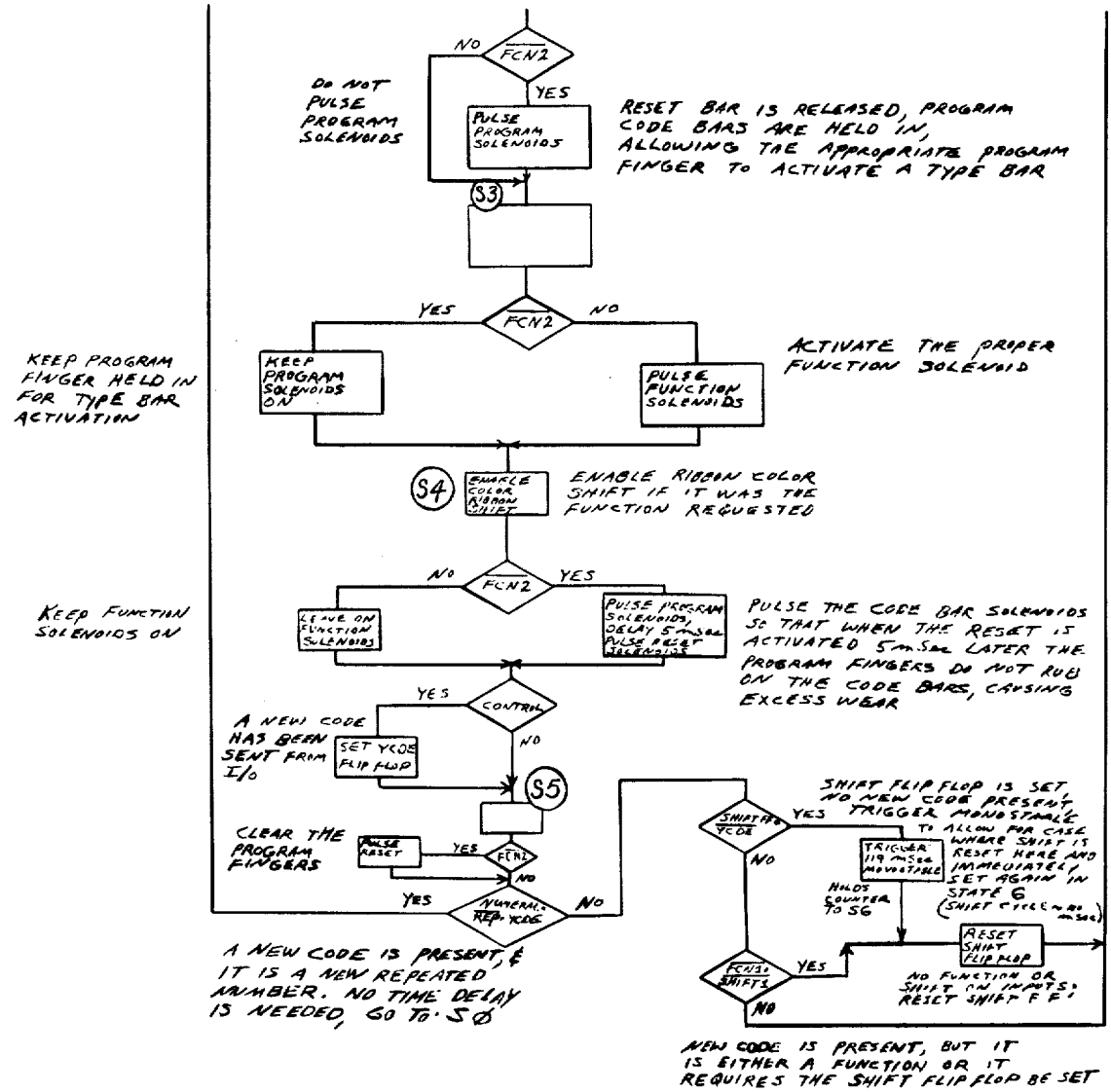

FCN2
NO
YES
DO NOT PULSE PROGRAM SOLENOIDS
PULSE PROGRAM SOLENOIDS
RESET BAR IS RELEASED, PROGRAM CODE BARS ARE HELD IN, ALLOWING THE APPROPRIATE PROGRAM FINGER TO ACTIVATE A TYPE BAR
S3
FCN2
YES
NO
KEEP PROGRAM FINGER HELD IN FOR TYPE BAR ACTIVATION
KEEP PROGRAM SOLENOIDS ON
PULSE FUNCTION SOLENOIDS
ACTIVATE THE PROPER FUNCTION SOLENOID
S4
ENABLE COLOR RIBBON SHIFT
ENABLE RIBBON COLOR SHIFT IF IT WAS THE FUNCTION REQUESTED
KEEP FUNCTION SOLENOIDS ON
FCN2
NO
YES
LEAVE ON FUNCTION SOLENOIDS
PULSE PROGRAM SOLENOIDS, DELAY 5 mSec, PULSE RESET SOLENOIDS
PULSE THE CODE BAR SOLENOIDS SO THAT WHEN THE RESET IS ACTIVATED 5 mSec LATER THE PROGRAM FINGERS DO NOT RUB ON THE CODE BARS, CAUSING EXCESS WEAR
CONTROL
YES
NO
A NEW CODE HAS BEEN SENT FROM I/O
SET YCODE FLIP FLOP
S5
CLEAR THE PROGRAM FINGERS
PULSE RESET
FCN2
YES
NO
NUMERAL REP. YCODE
YES
NO
A NEW CODE IS PRESENT, & IT IS A NEW REPEATED NUMBER. NO TIME DELAY IS NEEDED, GO TO S0
SHIFT FLIP FLOP IS SET, NO NEW CODE PRESENT, TRIGGER MONOSTABLE TO ALLOW FOR CASE WHERE SHIFT IS RESET HERE AND IMMEDIATELY SET AGAIN IN STATE 6 (SHIFT CYCLE = 80 mSec)
SHIFT & YCODE
YES
NO
TRIGGER 119 mSec MONOSTABLE
HOLDS COUNTER TO S6
RESET SHIFT FLIP FLOP
FCN2 & SHIFTS
YES
NO
NO FUNCTION OR SHIFT ON INPUTS: RESET SHIFT F F
NEW CODE IS PRESENT, BUT IT IS EITHER A FUNCTION OR IT REQUIRES THE SHIFT FLIP FLOP BE SET

FIG 77B

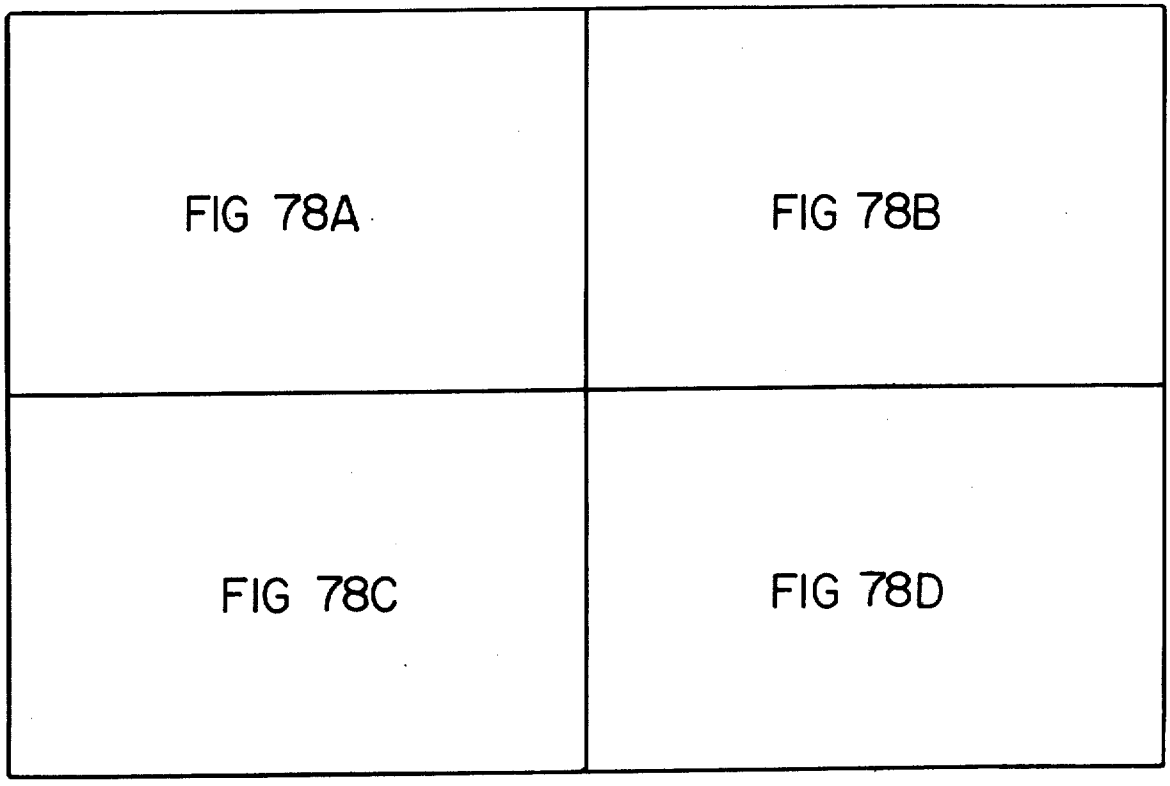


FIG 78'

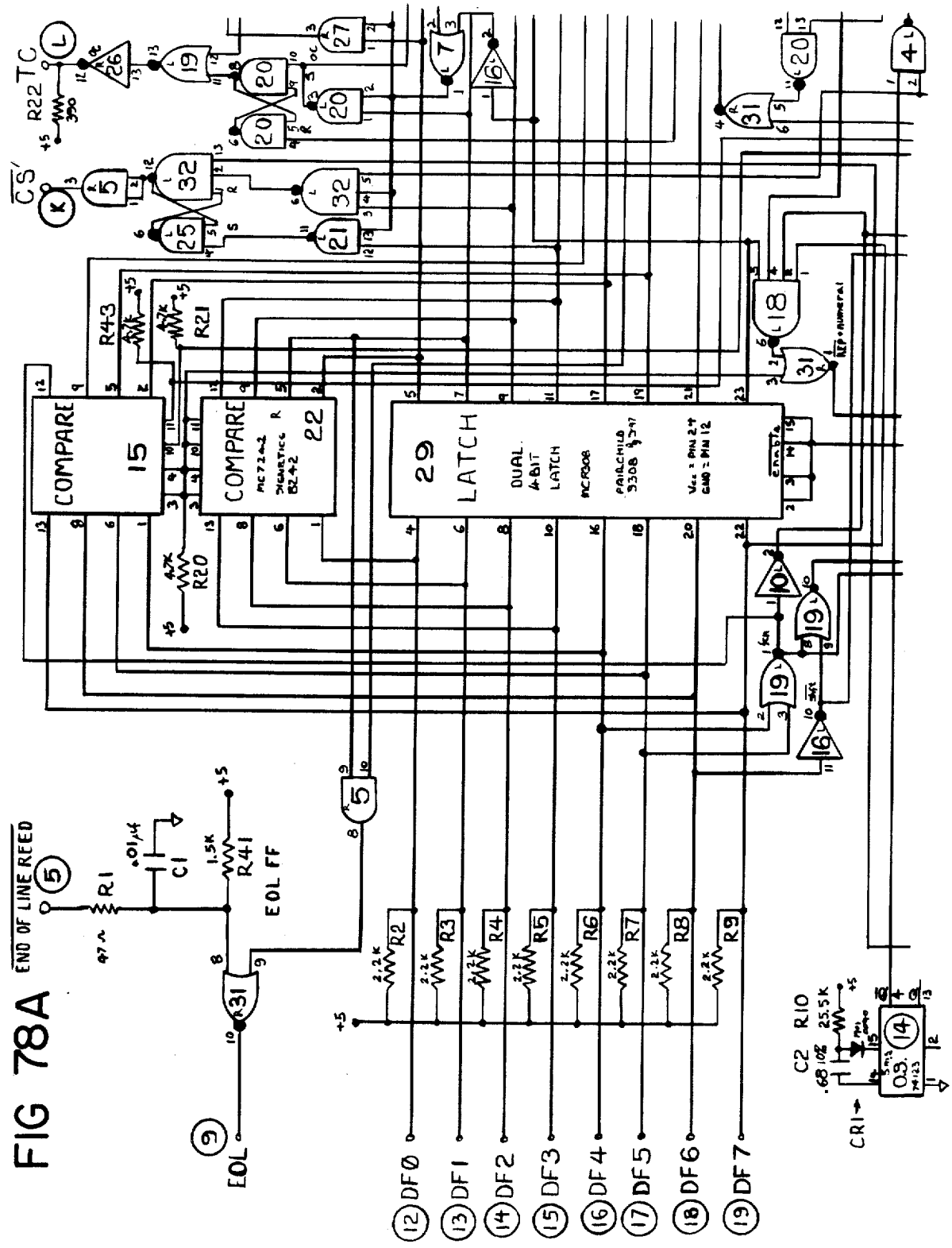

COMPARE 15
COMPARE 22
29 LATCH
DUAL 4-BIT LATCH
R22 TC
CS'
R43
R21
R20
R1
C1
R41
EOL FF
END OF LINE REED
EOL
R2
R3
R4
R5
R6
R7
R8
R9
DF0
DF1
DF2
DF3
DF4
DF5
DF6
DF7
C2 R10
CR1

FIG 78A

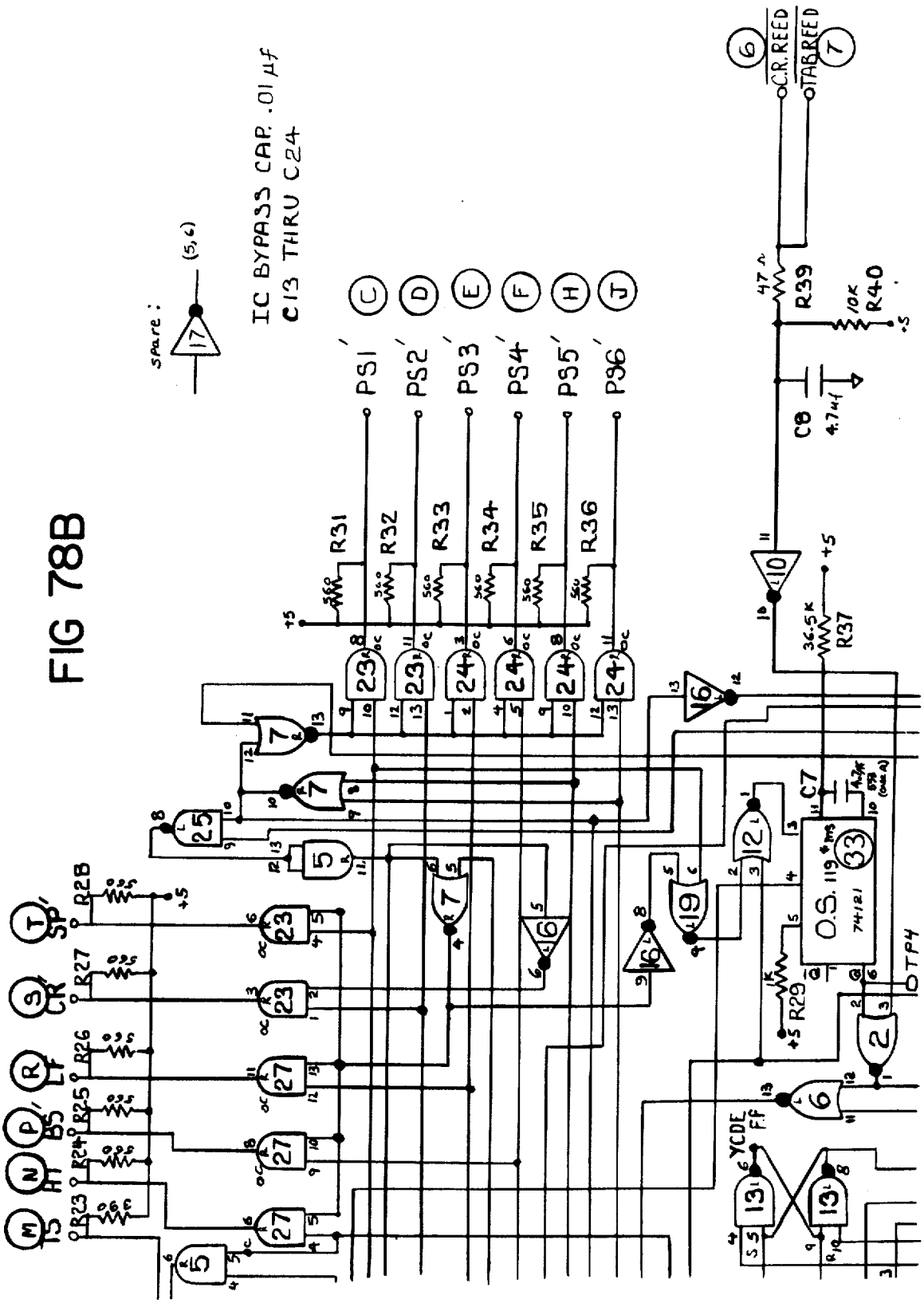

IC BYPASS CAP. .01 μf
C13 THRU C24
spare:
(5,4)
PS1'
PS2'
PS3'
PS4'
PS5'
PS6'
R31
R32
R33
R34
R35
R36
C.R. REED
TAB REED
R39
R40
C8
R37
C7
O.S. 119 ms
74121
R29
YCDE FF
TP4
R28
R27
R26
R25
R24
R23

FIG 78B

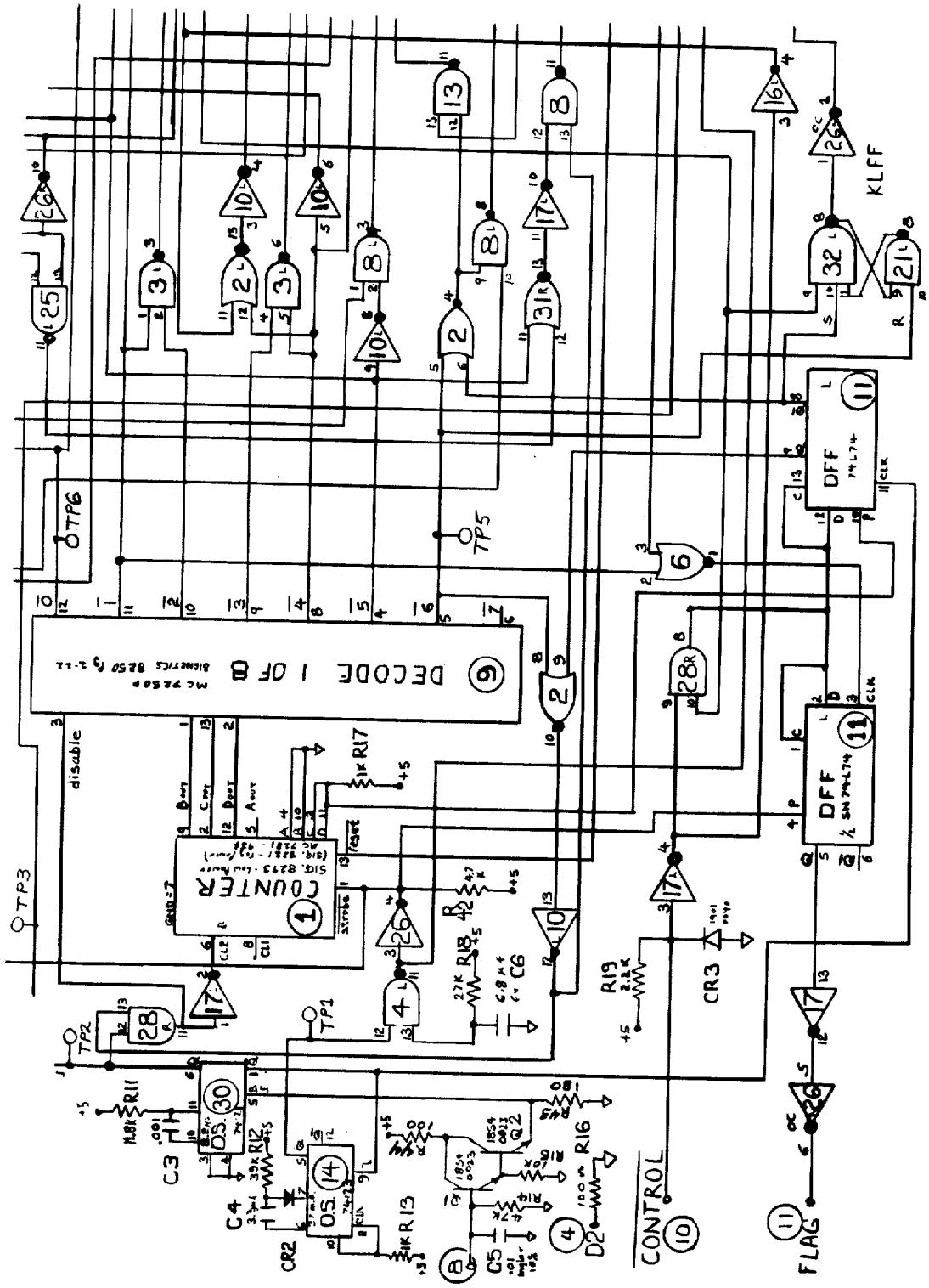

DECODE 1 OF 8
COUNTER
DFF
DFF
KLFF
disable
CONTROL
FLAG
TP1
TP2
TP3
TP5
TP6
R11
R12
R13
R16
R17
R18
R19
C3
C4
C5
C6
CR2
CR3
D2
Q2

FIG 78C

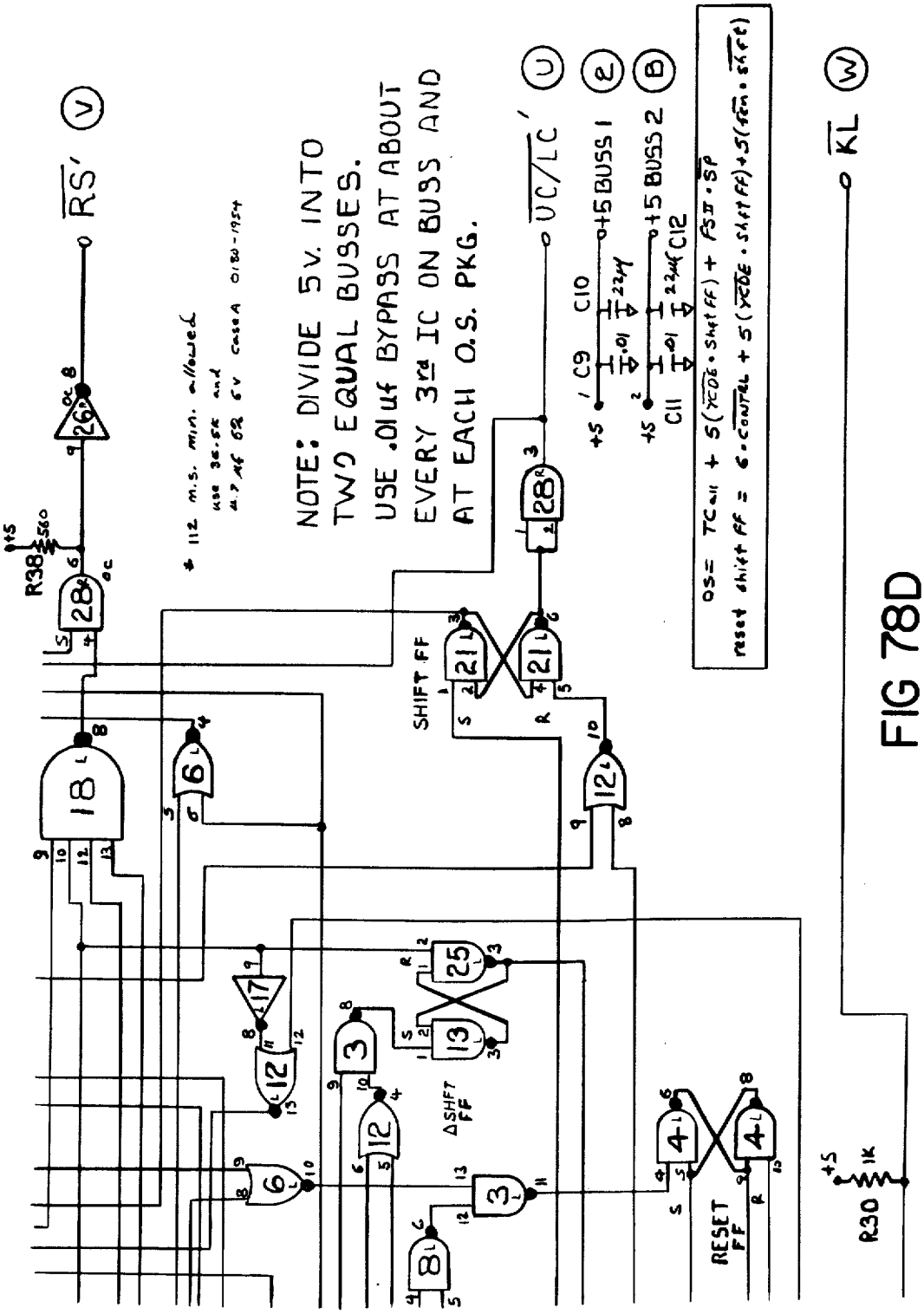

NOTE: DIVIDE 5v. INTO TWO EQUAL BUSSES.
USE .01uf BYPASS AT ABOUT EVERY 3rd IC ON BUSS AND AT EACH O.S. PKG.
RS'
UC/LC'
KL
+5 BUSS 1
+5 BUSS 2
C9
C10
C11
C12
R38
R30
SHIFT FF
RESET FF
ΔSHIFT FF

FIG 78D

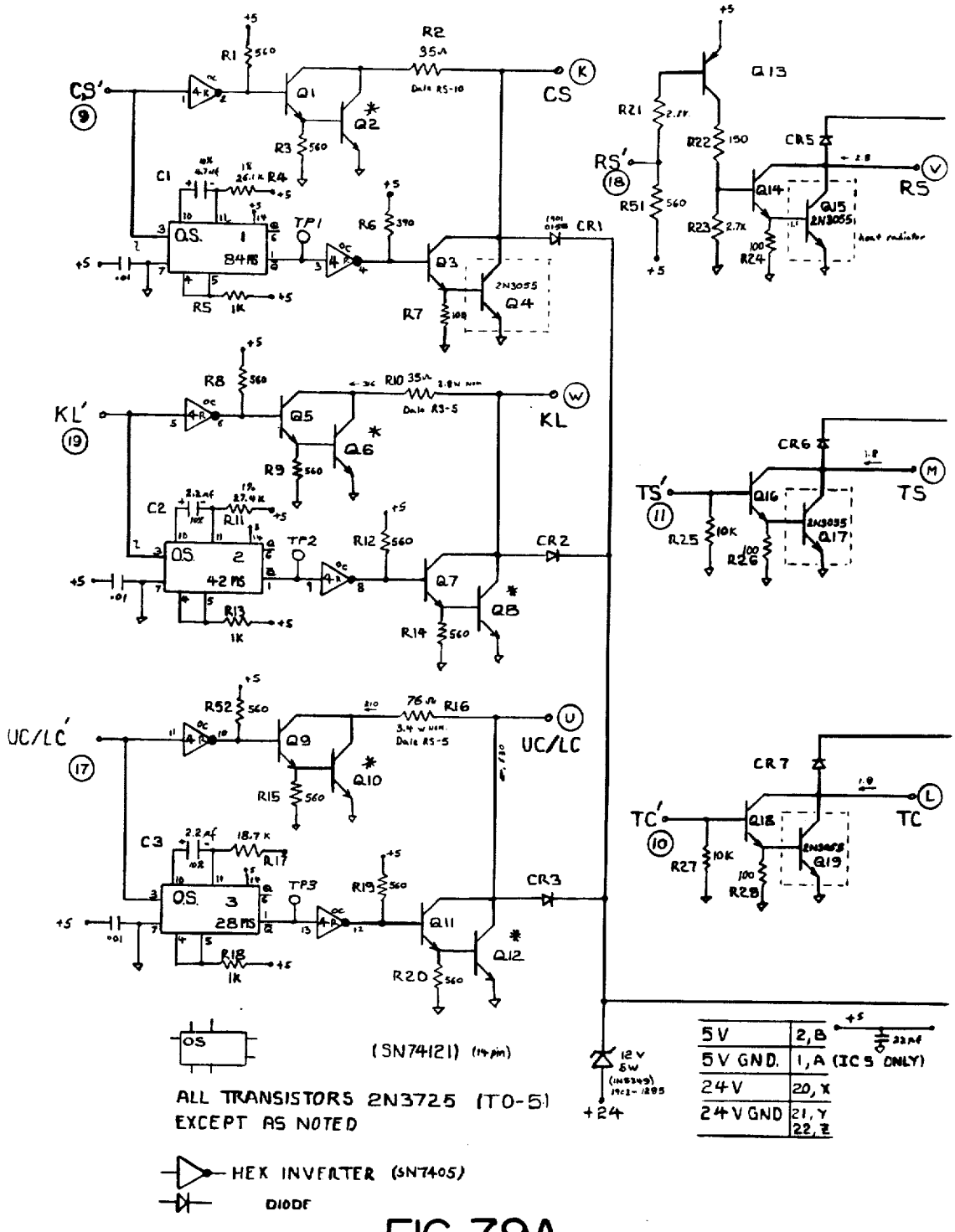

CS'
CS
KL'
KL
UC/LC'
UC/LC
RS'
RS
TS'
TS
TC'
TC
+24
+5
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
Q9
Q10
Q11
Q12
Q13
Q14
Q15
Q16
Q17
Q18
Q19
CR1
CR2
CR3
CR5
CR6
CR7
TP1
TP2
TP3
O.S. 1
O.S. 2
O.S. 3
84 MS
42 MS
28 MS
2N3055
(SN74121) (14 pin)
ALL TRANSISTORS 2N3725 (TO-5)
EXCEPT AS NOTED
HEX INVERTER (SN7405)
DIODE
5V 2,B
5V GND. 1,A (IC 5 ONLY)
24V 20, X
24V GND 21, Y
22, Z

FIG 79A

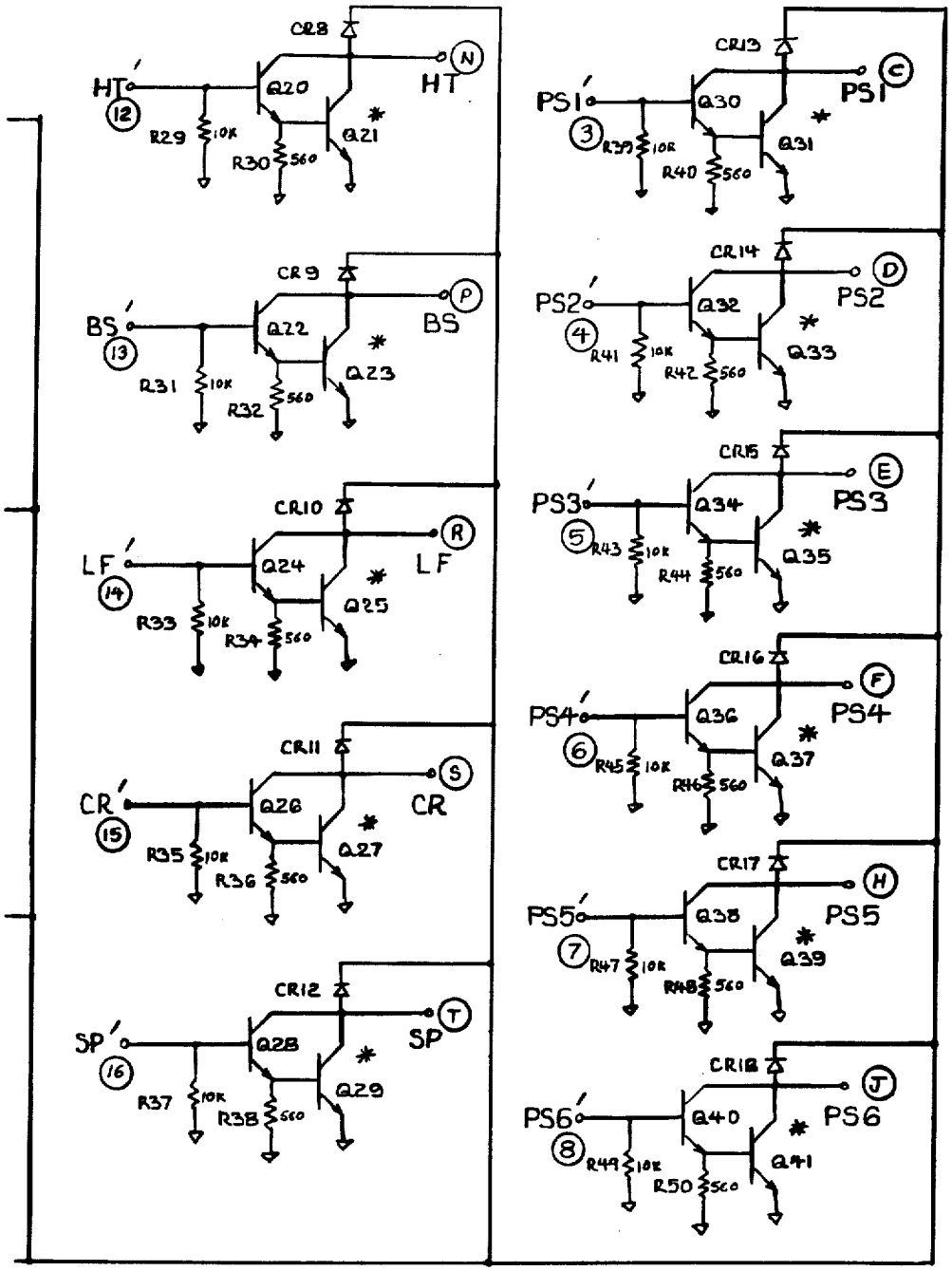

CR8
HT'
12
R29 10K
Q20
R30 560
Q21
HT
N
CR9
BS'
13
R31 10K
Q22
R32 560
Q23
BS
P
CR10
LF'
14
R33 10K
Q24
R34 560
Q25
LF
R
CR11
CR'
15
R35 10K
Q26
R36 560
Q27
CR
S
CR12
SP'
16
R37 10K
Q28
R38 560
Q29
SP
T
CR13
PS1'
3
R39 10K
Q30
R40 560
Q31
PS1
C
CR14
PS2'
4
R41 10K
Q32
R42 560
Q33
PS2
D
CR15
PS3'
5
R43 10K
Q34
R44 560
Q35
PS3
E
CR16
PS4'
6
R45 10K
Q36
R46 560
Q37
PS4
F
CR17
PS5'
7
R47 10K
Q38
R48 560
Q39
PS5
H
CR18
PS6'
8
R49 10K
Q40
R50 560
Q41
PS6
J
.01 bypass on each of
stuff on 5V.
NOTE: TRANSISTORS MARKED * USE HEAT SINK

FIG 79B

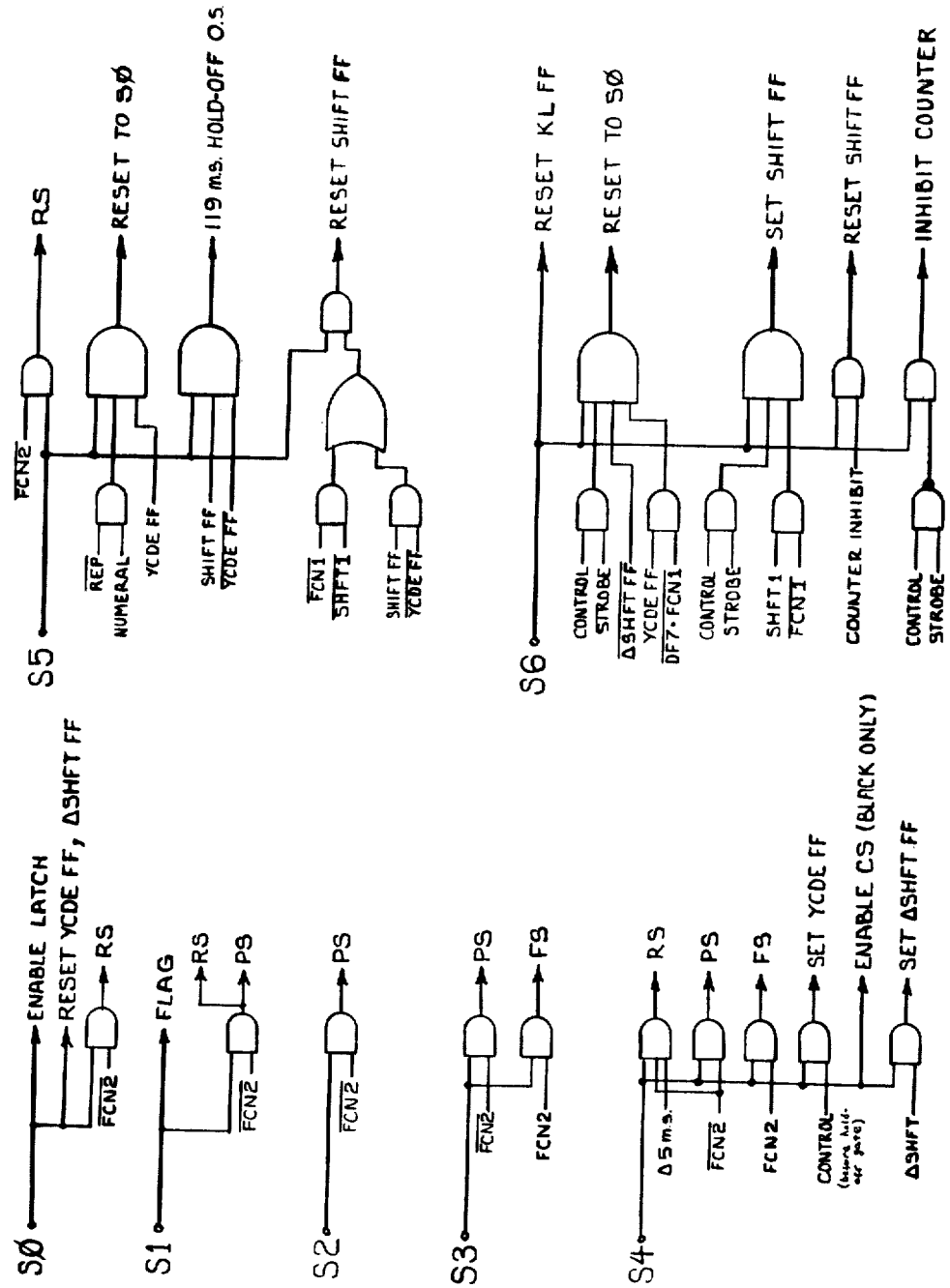

S0
ENABLE LATCH
RESET YCDE FF, ΔSHFT FF
RS
FCN2
S1
FLAG
RS
PS
FCN2
S2
PS
FCN2
S3
PS
FCN2
FS
FCN2
S4
RS
Δ5 m.s.
PS
FCN2
FS
FCN2
SET YCDE FF
CONTROL
ENABLE CS (BLACK ONLY)
ΔSHFT
SET ΔSHFT FF
S5
RS
RESET TO SØ
REP
NUMERAL
YCDE FF
119 m.s. HOLD-OFF O.S
SHIFT FF
YCDE FF
RESET SHIFT FF
FCN1
SHFT1
SHIFT FF
YCDE FF
S6
RESET KL FF
RESET TO SØ
CONTROL
STROBE
ΔSHFT FF
YCDE FF
DF7·FCN1
CONTROL
STROBE
SET SHIFT FF
SHFT1
FCN1
RESET SHIFT FF
COUNTER INHIBIT
INHIBIT COUNTER
CONTROL
STROBE
S7
NO INSTRUCTIONS
(NEXT STATE IS SØ)

FIG 80A

INPUTS

S0-S7 : 14 m.s. STATES
FCN 2 = CODE ON LATCH OUTPUTS IS A FUNCTION
FCN 1 = " " " INPUTS = " "
CONTROL = SAME AS CEO FROM 9810
REP = REPEATING CHARACTER
SHFT 1 = CODE ON LATCH INPUTS IS FOR UPPER CASE CHARACTER ON WRITER
STROBE = 8 μsec PULSE DURING STATE TRANSITION
ΔSHFT = SHIFT BIT AT INPUT AND STATE OF SHIFT FF NOT THE SAME
DF7 = DATA, FACIT CODE BIT 7
(USED IN CODING "LEFT SIDE" FUNCTIONS : COLOR SHIFT, TAB CLEAR, TAB SET)

OUTPUTS

RS = "RESET SOLENOIDS"
YCDE FF = "YES CODE FLIP FLOP
FLAG = REQUEST TO CALCULATOR FOR NEXT CODE
PS = "PROGRAM SOLENOIDS"
FS = "FUNCTION SOLENOIDS"
CS = RIBBON COLOR SHIFT
SHIFT FF = INSTRUCTS WRITER TO SHIFT TO UPPER CASE (also called 12VFF)
KLFF = KEYBOARD LOCK FF
INHIBIT COUNTER = STOP INPUT PULSES TO COUNTER

FIG 80B

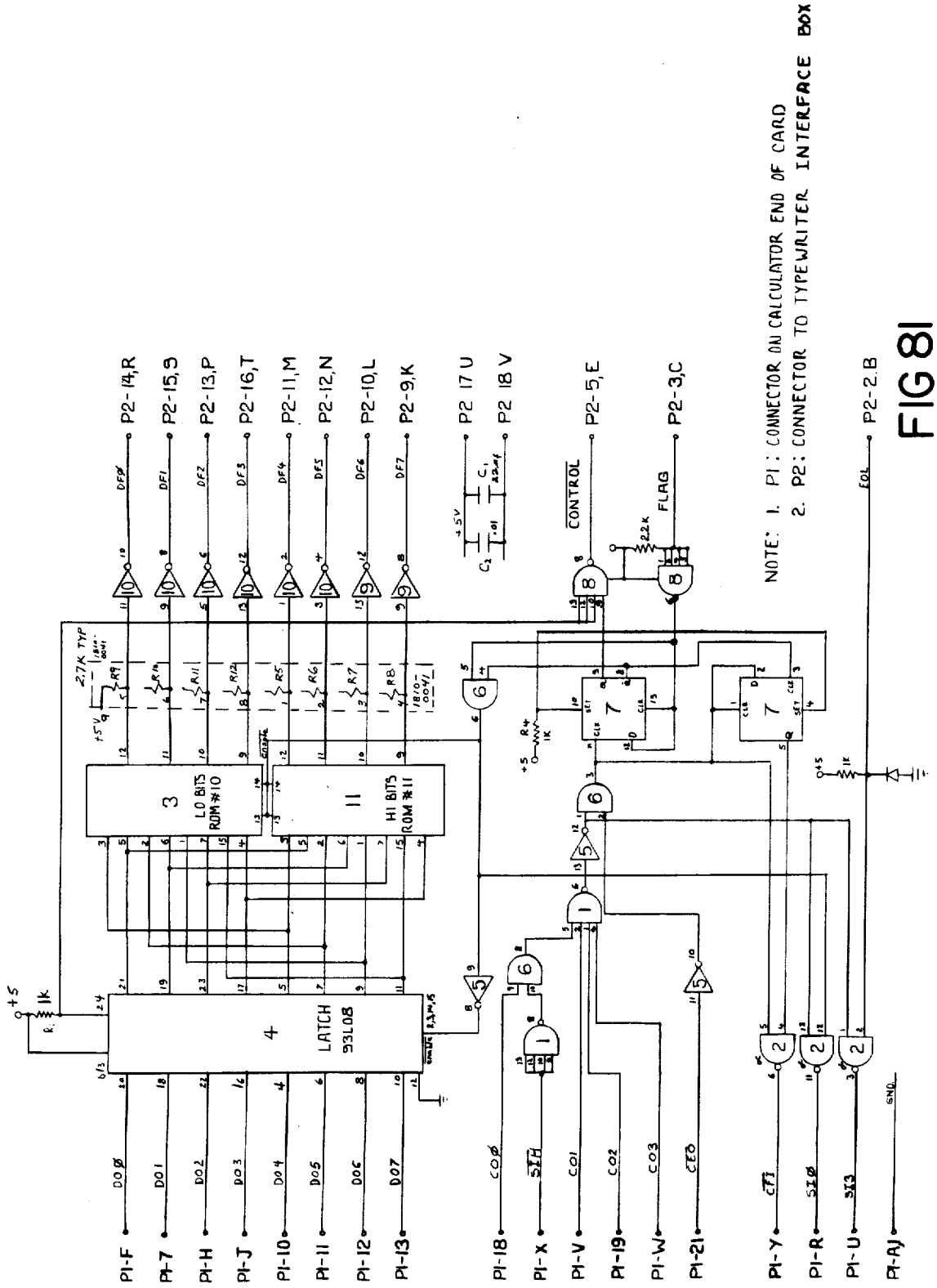

NOTE: 1. P1: CONNECTOR ON CALCULATOR END OF CARD
2. P2: CONNECTOR TO TYPEWRITER INTERFACE BOX
LO BITS ROM #10
HI BITS ROM #11
LATCH 93L08
CONTROL
FLAG
P2-14,R
P2-15,S
P2-13,P
P2-16,T
P2-11,M
P2-12,N
P2-10,L
P2-9,K
P2 17 U
P2 18 V
P2-5,E
P2-3,C
P2-2,B
P1-F
P1-7
P1-H
P1-J
P1-10
P1-11
P1-12
P1-13
P1-18
P1-X
P1-V
P1-19
P1-W
P1-21
P1-Y
P1-R
P1-U
P1-AJ

FIG 81

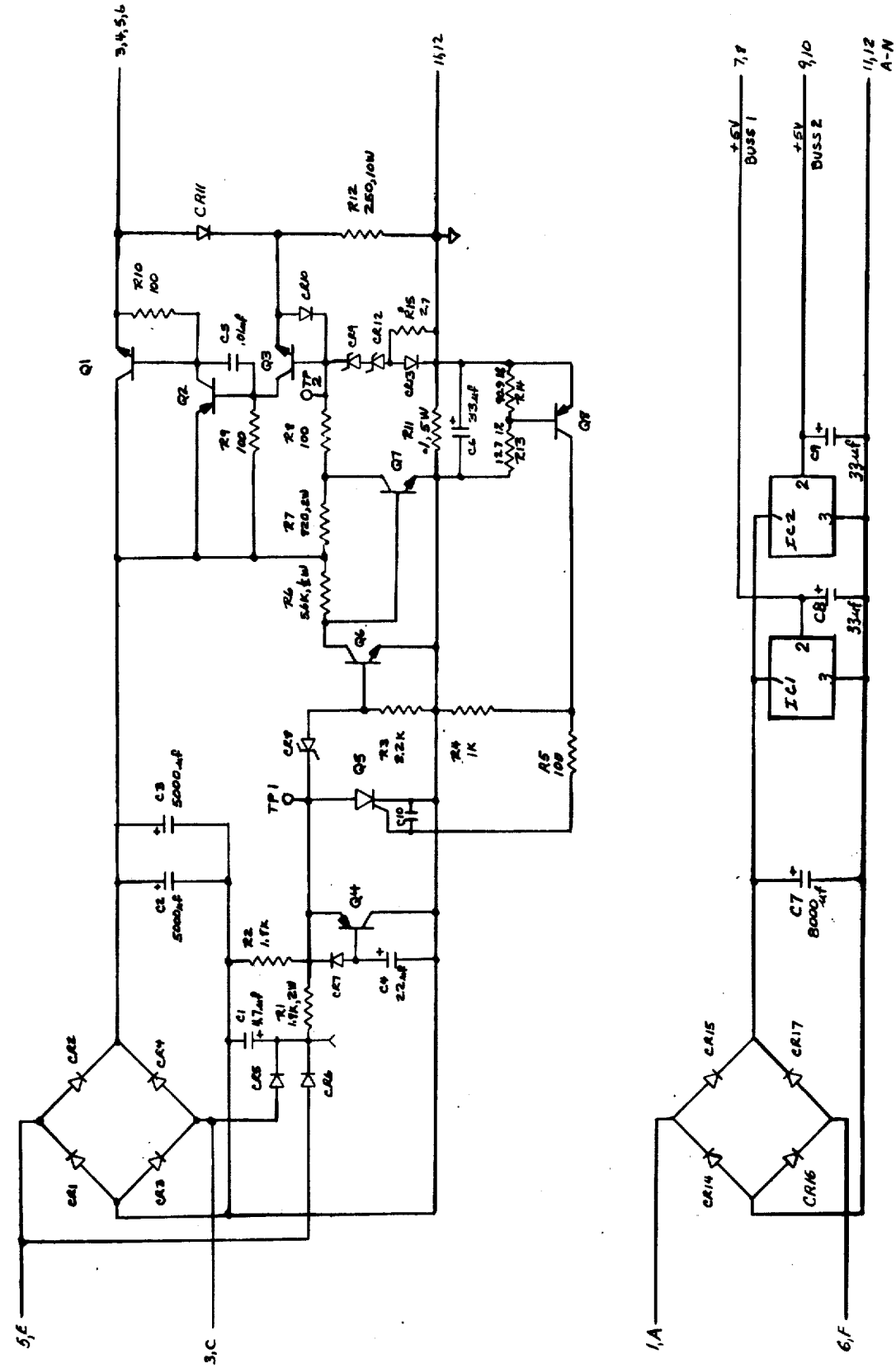

3,4,5,6
11,12
CR11
R12 250,10W
R10 100
C5 .01uf
CR10
CR9
CR12
R15 2.7
Q1
Q2
Q3
TP 2
R9 100
R8 100
CR13
C6 33uf
R13 15 2W
213
R14 1K
Q8
Q7
R11 .47,5W
R7 120,2W
R6 56K,1W
Q6
CR9
Q5
R3 2.2K
R4 1K
R5 100
TP 1
C10
C3 5000uf
C2 5000uf
Q4
R2 1.1K
C4 22uf
C1 .47uf
R1 1.1K,2W
CR7
CR8
CR5
CR6
CR1
CR2
CR3
CR4
5,F
3,C
7,8
9,10
11,12 A-N
+5V BUSS 1
+5V BUSS 2
C9 33uf
IC2
2
3
C8 33uf
IC1
2
3
C7 8000uf
CR15
CR17
CR14
CR16
1,A
6,F

FIG 82

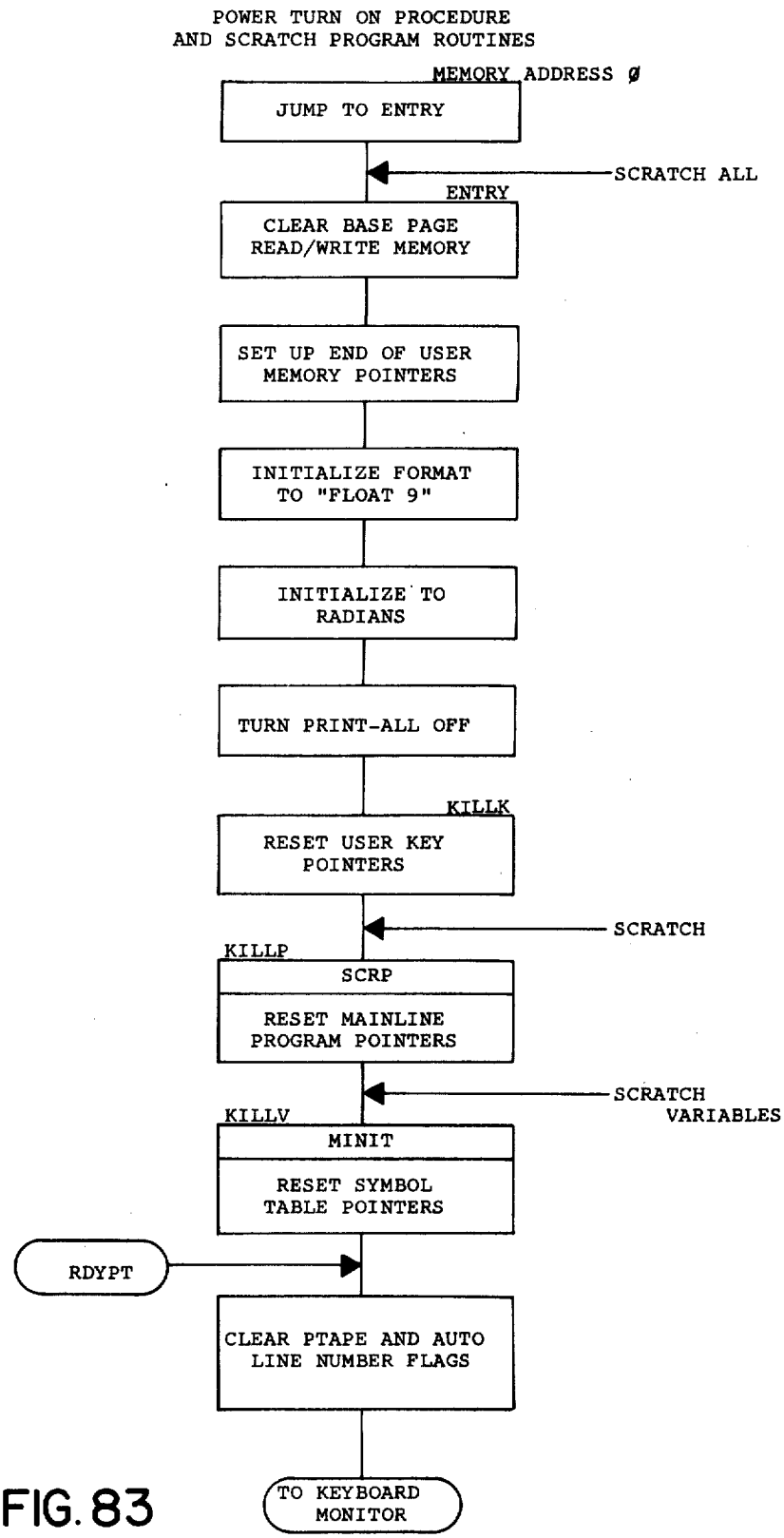

POWER TURN ON PROCEDURE
AND SCRATCH PROGRAM ROUTINES
MEMORY ADDRESS ø
JUMP TO ENTRY
SCRATCH ALL
ENTRY
CLEAR BASE PAGE
READ/WRITE MEMORY
SET UP END OF USER
MEMORY POINTERS
INITIALIZE FORMAT
TO "FLOAT 9"
INITIALIZE TO
RADIANS
TURN PRINT-ALL OFF
KILLK
RESET USER KEY
POINTERS
SCRATCH
KILLP
SCRP
RESET MAINLINE
PROGRAM POINTERS
SCRATCH
VARIABLES
KILLV
MINIT
RESET SYMBOL
TABLE POINTERS
RDYPT
CLEAR PTAPE AND AUTO
LINE NUMBER FLAGS
TO KEYBOARD
MONITOR

FIG. 83

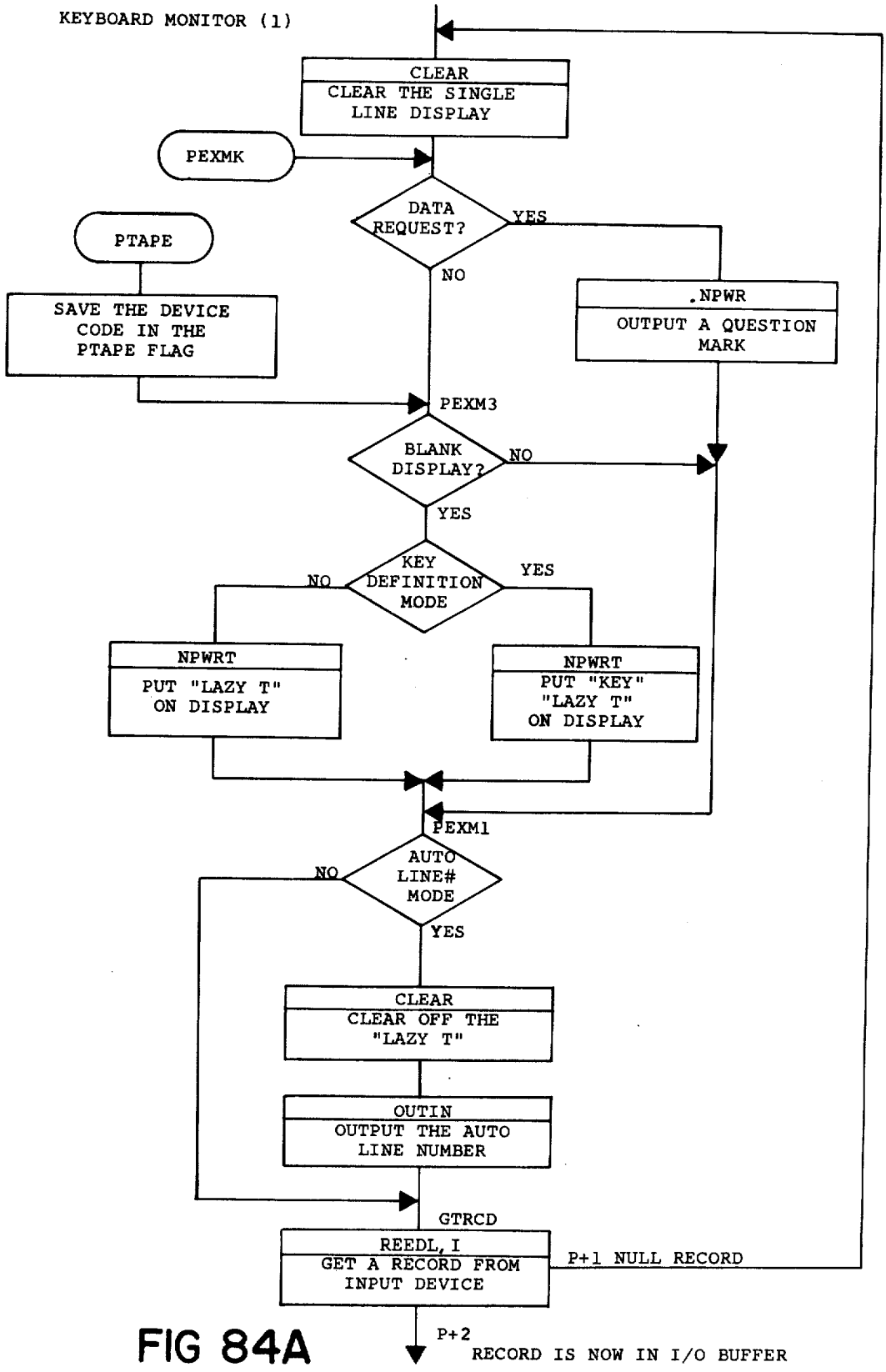

KEYBOARD MONITOR (1)
CLEAR
CLEAR THE SINGLE LINE DISPLAY
PEXMK
DATA REQUEST?
YES
NO
PTAPE
SAVE THE DEVICE CODE IN THE PTAPE FLAG
.NPWR
OUTPUT A QUESTION MARK
PEXM3
BLANK DISPLAY?
NO
YES
KEY DEFINITION MODE
NO
YES
NPWRT
PUT "LAZY T" ON DISPLAY
NPWRT
PUT "KEY" "LAZY T" ON DISPLAY
PEXM1
AUTO LINE# MODE
NO
YES
CLEAR
CLEAR OFF THE "LAZY T"
OUTIN
OUTPUT THE AUTO LINE NUMBER
GTRCD
REEDL, I
GET A RECORD FROM INPUT DEVICE
P+1 NULL RECORD
P+2
RECORD IS NOW IN I/O BUFFER

FIG 84A

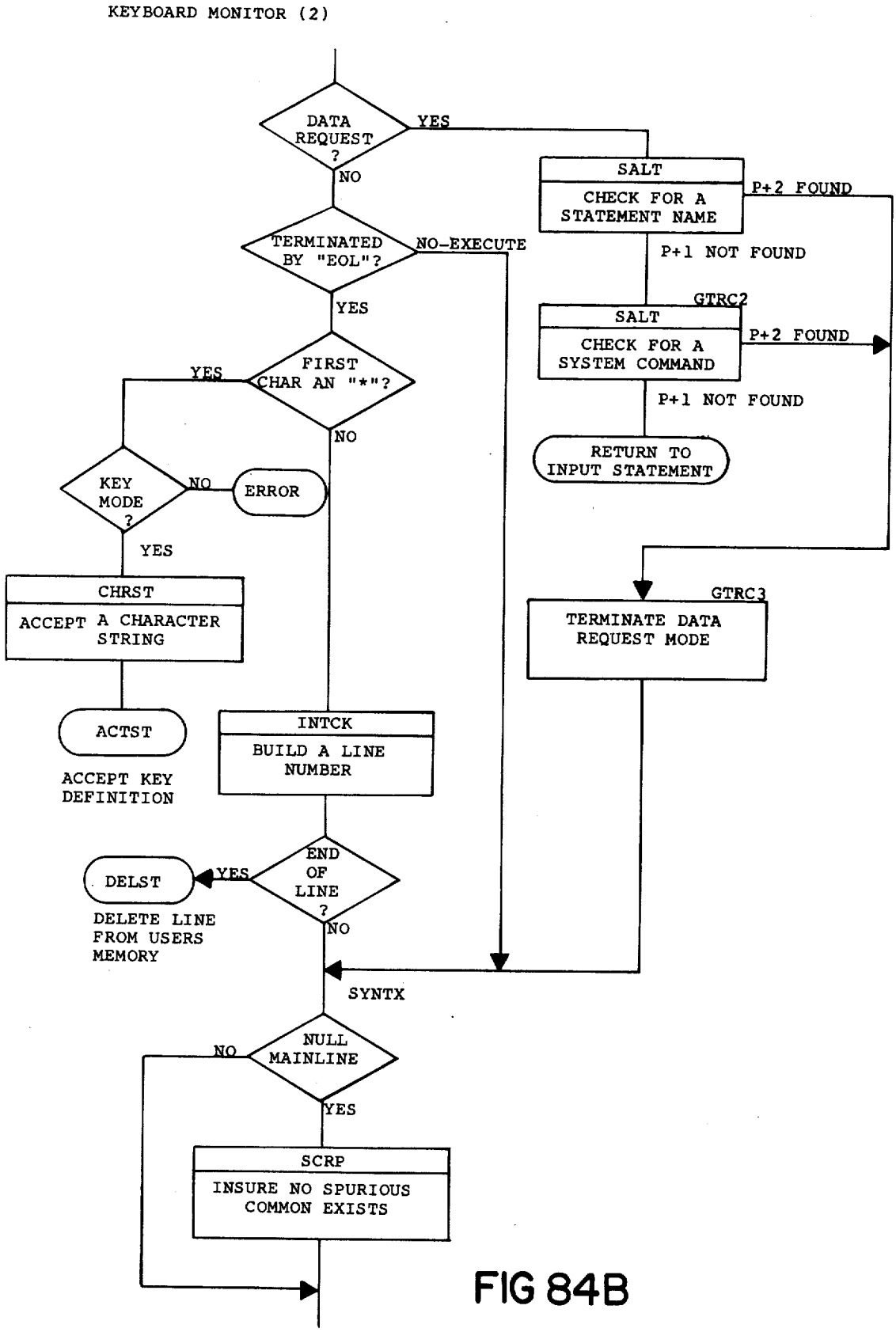

KEYBOARD MONITOR (2)
DATA REQUEST ?
YES
NO
SALT
CHECK FOR A STATEMENT NAME
P+2 FOUND
P+1 NOT FOUND
TERMINATED BY "EOL"?
NO-EXECUTE
YES
GTRC2
SALT
CHECK FOR A SYSTEM COMMAND
P+2 FOUND
P+1 NOT FOUND
FIRST CHAR AN "*"?
YES
NO
RETURN TO INPUT STATEMENT
KEY MODE ?
NO
ERROR
YES
GTRC3
TERMINATE DATA REQUEST MODE
CHRST
ACCEPT A CHARACTER STRING
ACTST
ACCEPT KEY DEFINITION
INTCK
BUILD A LINE NUMBER
END OF LINE ?
YES
DELST
DELETE LINE FROM USERS MEMORY
NO
SYNTX
NULL MAINLINE
NO
YES
SCRP
INSURE NO SPURIOUS COMMON EXISTS

FIG 84B

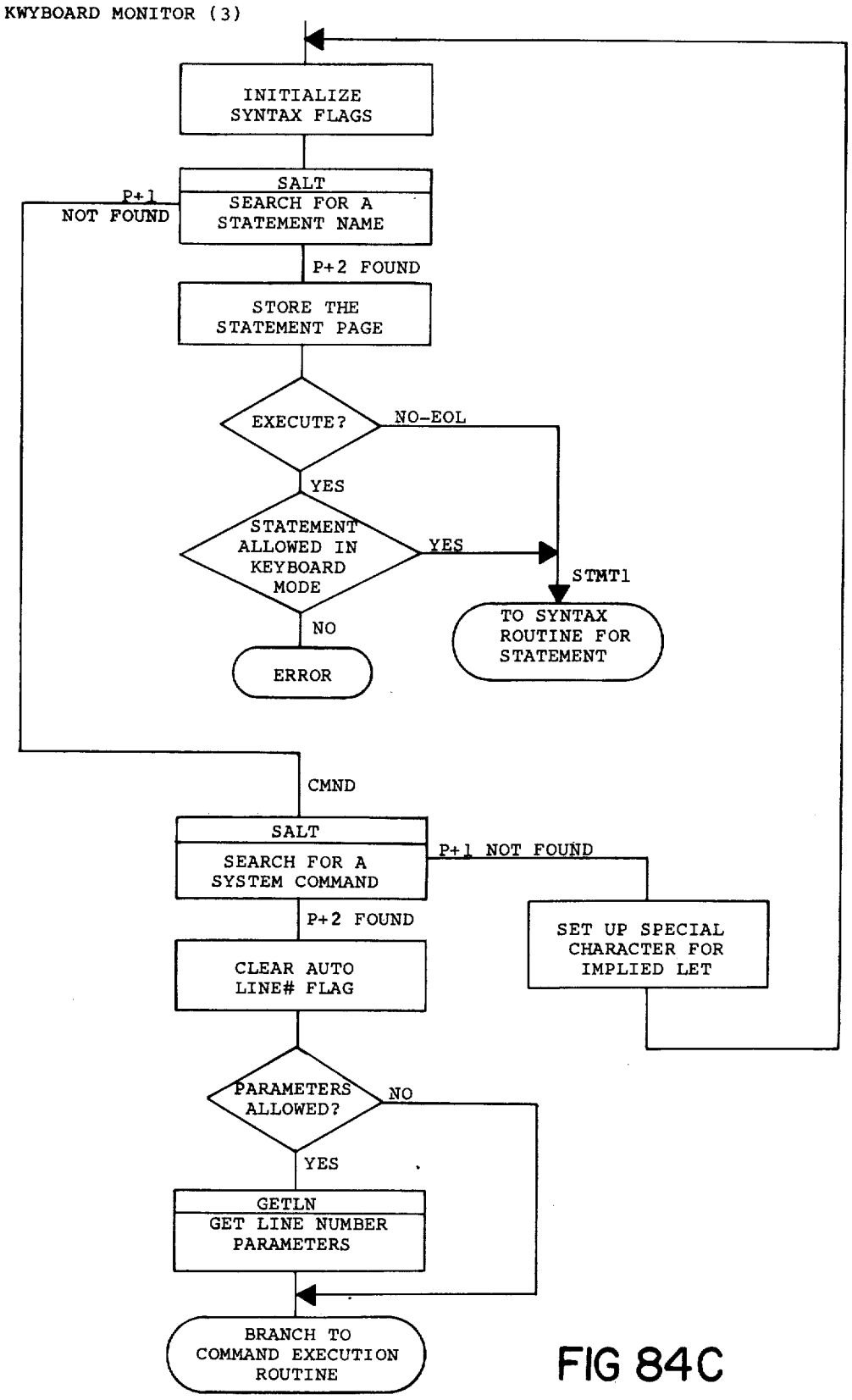

KWYBOARD MONITOR (3)
INITIALIZE SYNTAX FLAGS
SALT
SEARCH FOR A STATEMENT NAME
P+1 NOT FOUND
P+2 FOUND
STORE THE STATEMENT PAGE
EXECUTE?
NO-EOL
YES
STATEMENT ALLOWED IN KEYBOARD MODE
YES
STMT1
TO SYNTAX ROUTINE FOR STATEMENT
NO
ERROR
CMND
SALT
SEARCH FOR A SYSTEM COMMAND
P+1 NOT FOUND
P+2 FOUND
SET UP SPECIAL CHARACTER FOR IMPLIED LET
CLEAR AUTO LINE# FLAG
PARAMETERS ALLOWED?
NO
YES
GETLN
GET LINE NUMBER PARAMETERS
BRANCH TO COMMAND EXECUTION ROUTINE

FIG 84C

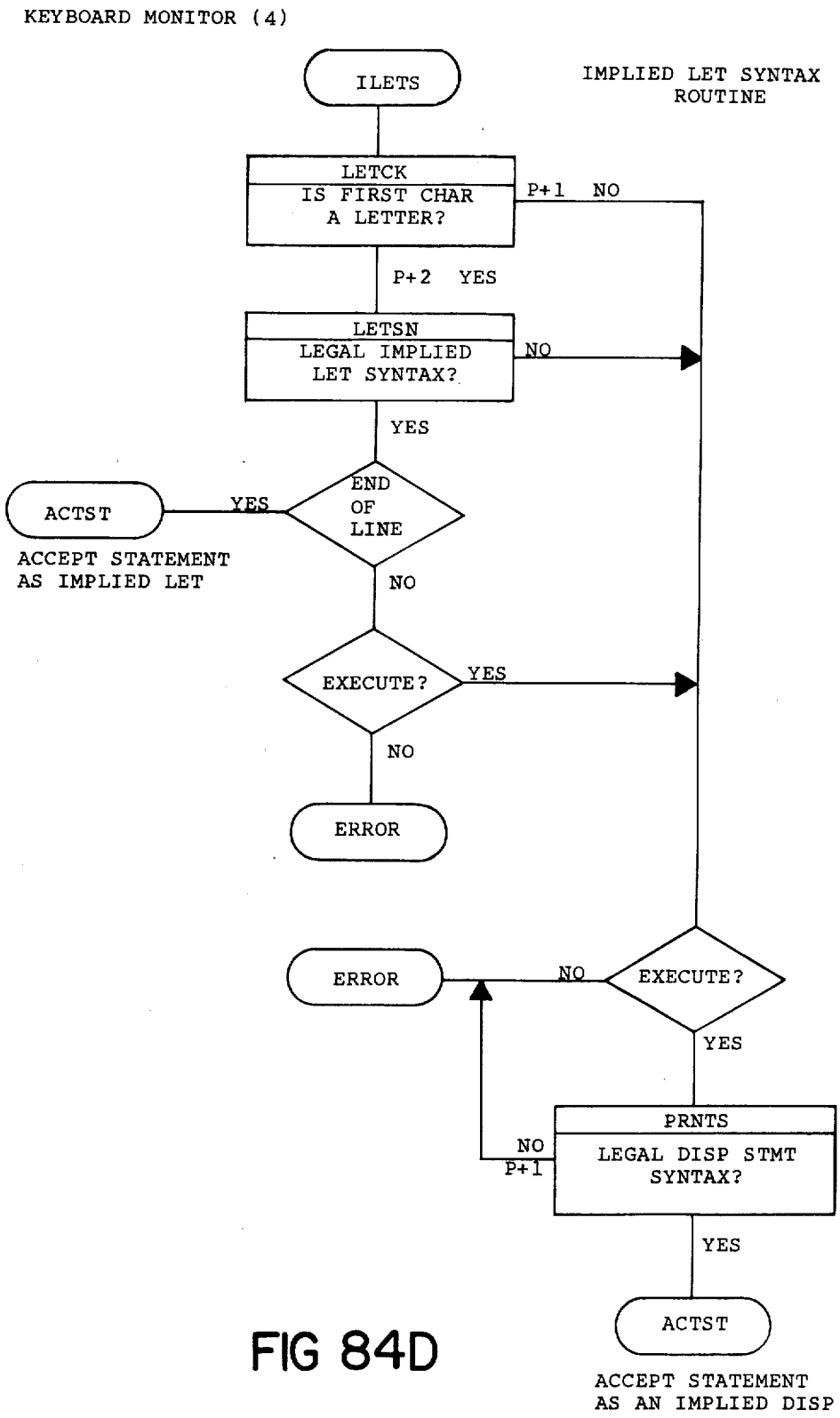
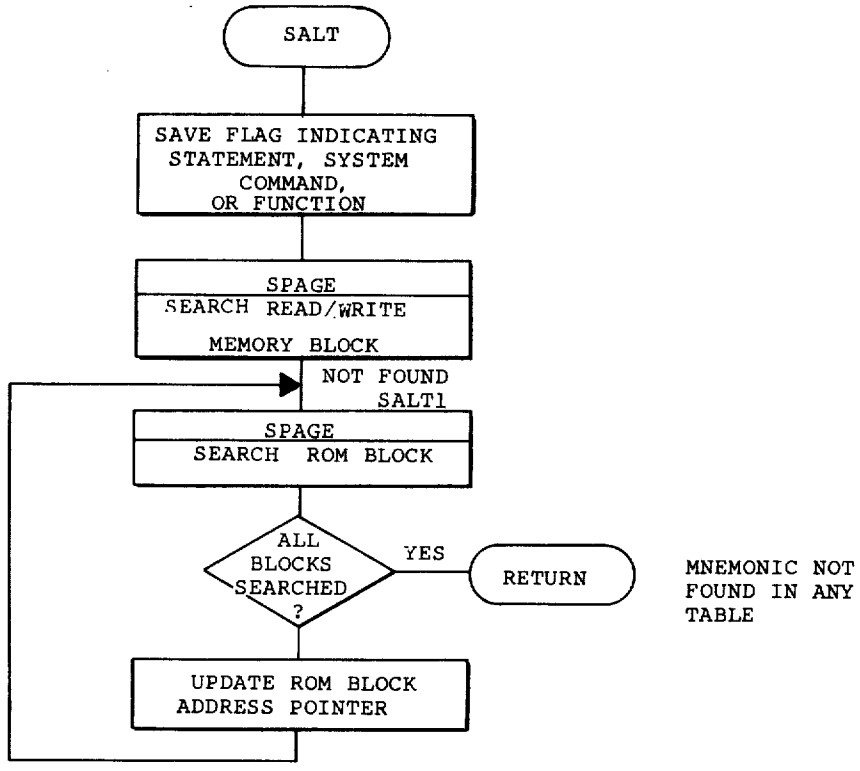

KEYBOARD MONITOR (4)
ILETS
IMPLIED LET SYNTAX ROUTINE
LETCK
IS FIRST CHAR A LETTER?
P+1 NO
P+2 YES
LETSN
LEGAL IMPLIED LET SYNTAX?
NO
YES
END OF LINE
YES
ACTST
ACCEPT STATEMENT AS IMPLIED LET
NO
EXECUTE?
YES
NO
ERROR
ERROR
NO
EXECUTE?
YES
PRNTS
LEGAL DISP STMT SYNTAX?
NO P+1
YES
ACTST
ACCEPT STATEMENT AS AN IMPLIED DISP

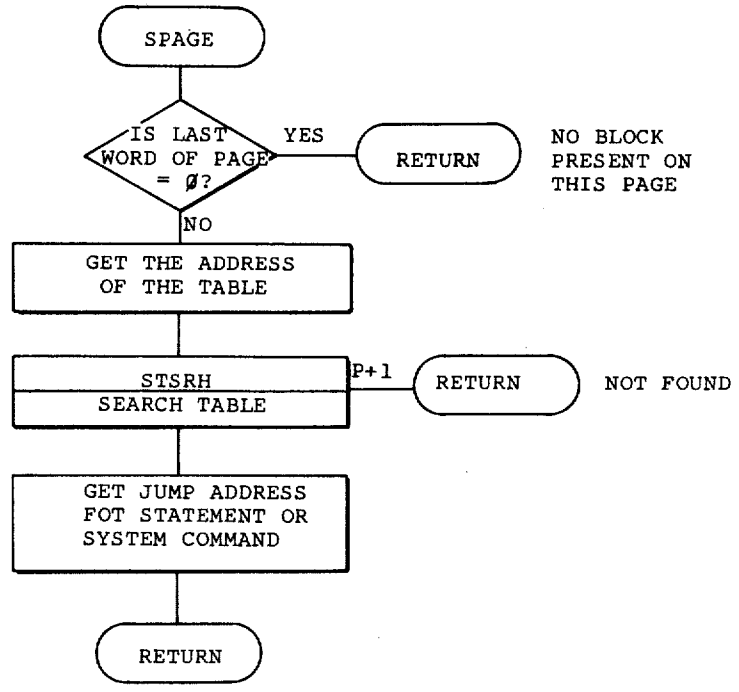
FIG 84D

SEARCH ALL TABLES



SEARCH A TABLE ON A GIVEN PAGE

FIG 85



RETURN TO P+2 OF ROUTINE WHICH CALLED SALT

FIG 86

TBSRH SUBROUTINE

NOTE 1: UPON ENTERING AT TBSRH THE FIRST CHARACTER OF THE STRING SEARCHED FOR IS IN THE A REGISTER. A POINTER TO THE TABLE BEING SEARCHED IS IN THE B REGISTER.

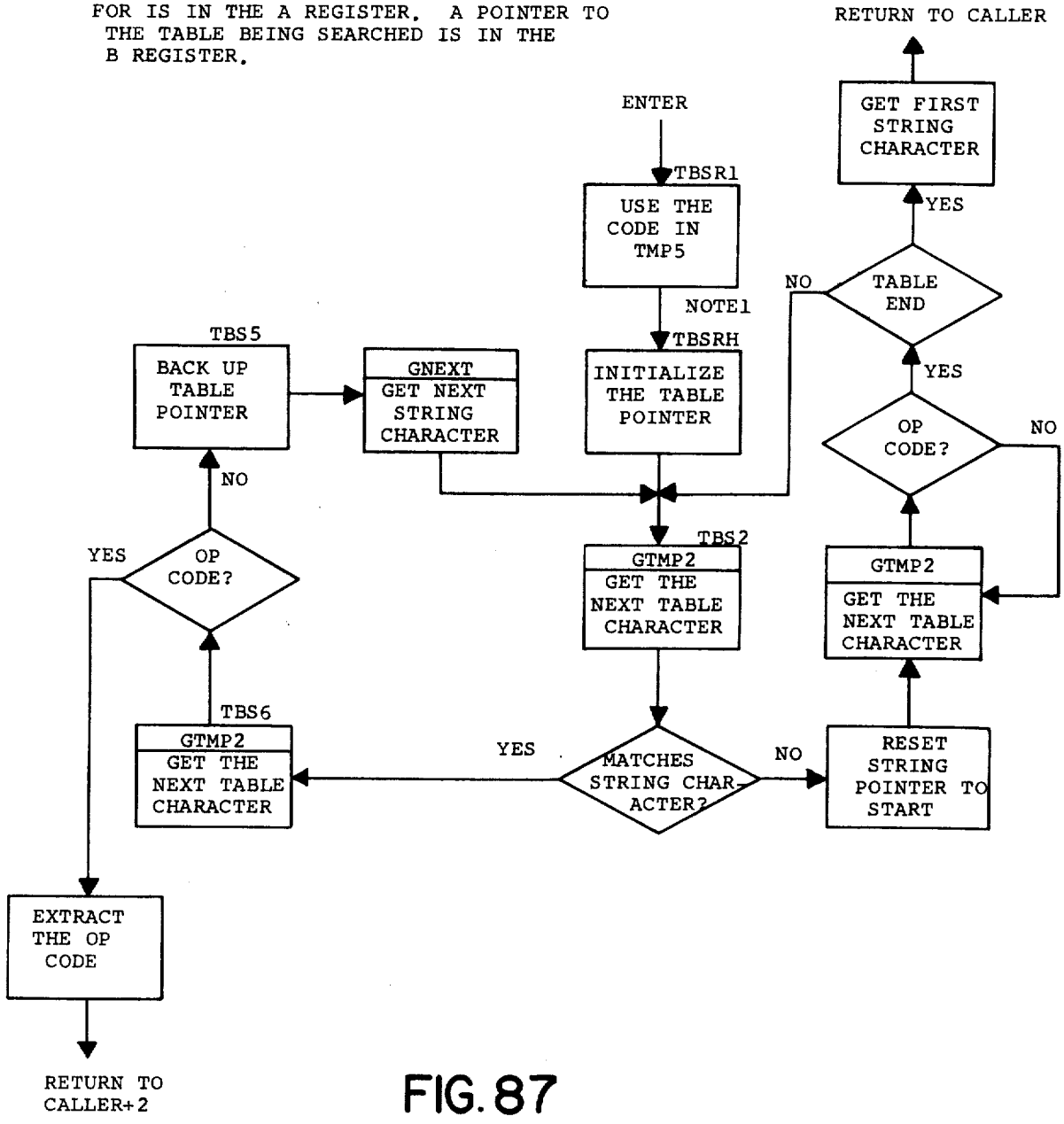
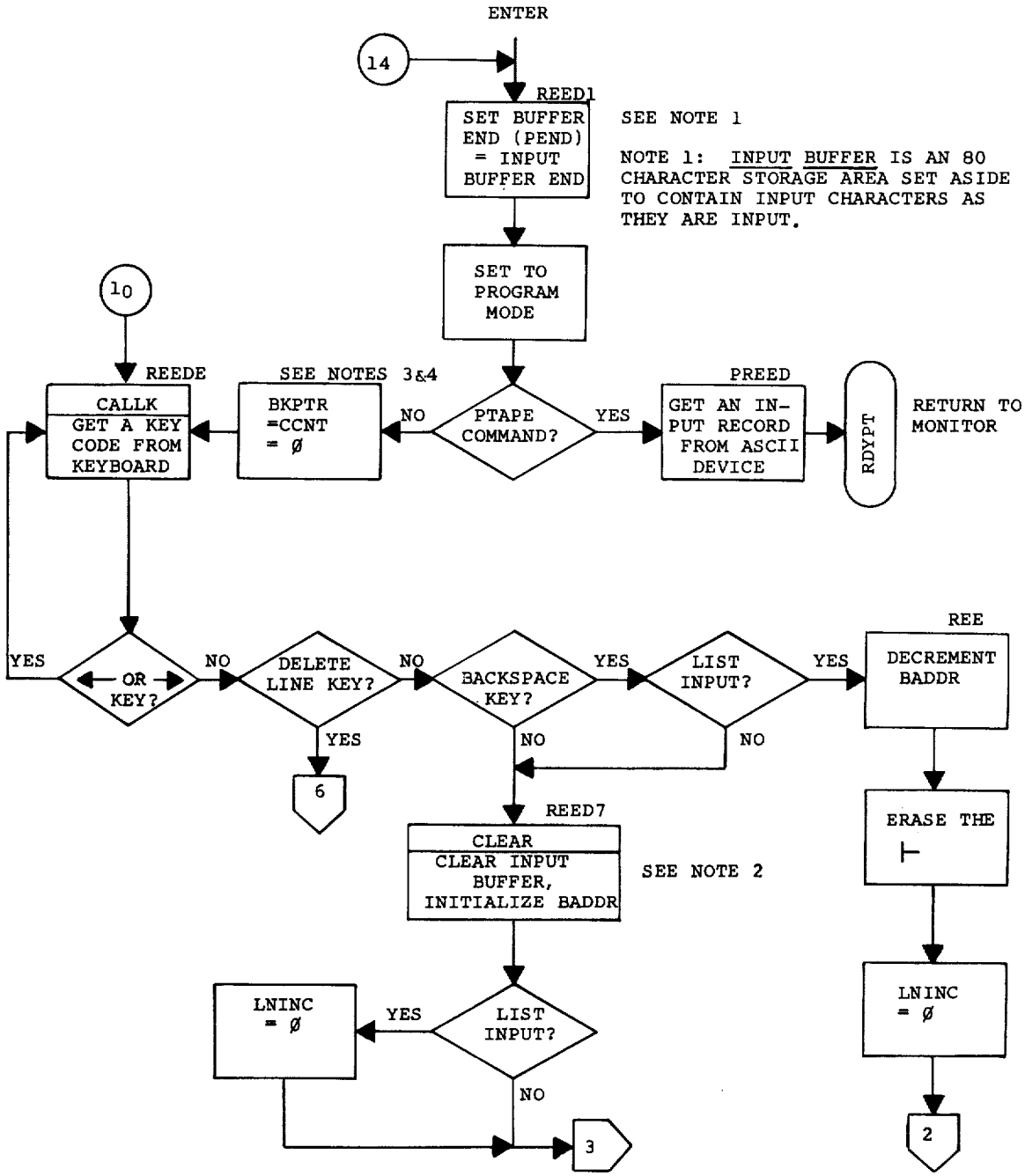


FIG. 87

REED SUBROUTINE (1)



NOTE 2: BADDR IS A CHARACTER POINTER WHICH POINTS TO THE NEXT CHARACTER POSITION IN THE INPUT BUFFER ALTHOUGH IT CAN BE USED WITH ANY BUFFER. IT CONSISTS OF A WORD POINTER SHIFTED LEFT ONE BIT POSITION. THE LEAST SIGNIFICANT BIT INDICATES UPPER OR LOWER HALF CHARACTER (0 OR 1). THIS TYPE OF POINTER ALLOWS DIRECT INCREMENTING AND DECREMENTING SINCE THE WORD ADDRESS PORTION AUTOMATICALLY INCREMENTS WHEN THE POINTER POINTS TO THE SECOND CHARACTER POSITION.
NOTE 3: BKPTR POINTS TO THE POSITION OF THE CURSOR CHARACTER IN THE INPUT BUFFER. IT IS ZERO IF THE CURSOR IS NOT VISABLE ON THE DISPLAY i.e. A BACKSPACE HAS NOT BEEN EXECUTED.
NOTE 4: CCNT IS A CHARACTER COUNT WHICH INDICATES THE NUMBER OF CHARACTERS IN A STRING TO BE TAKEN AS INPUT BEFORE A KEYCODE IS ENTERED.

FIG 88A

REED SUBROUTINE (2)

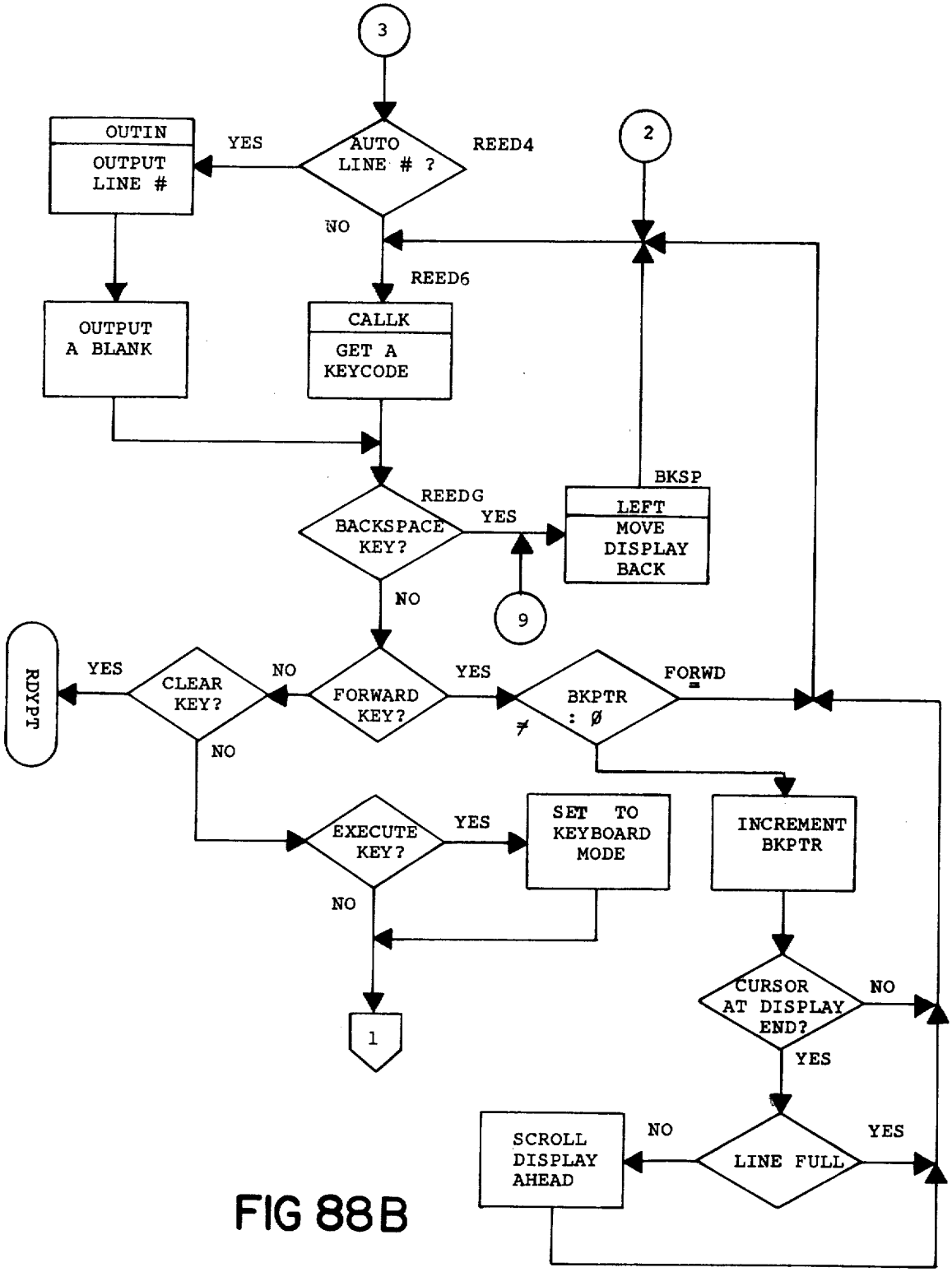


FIG 88B

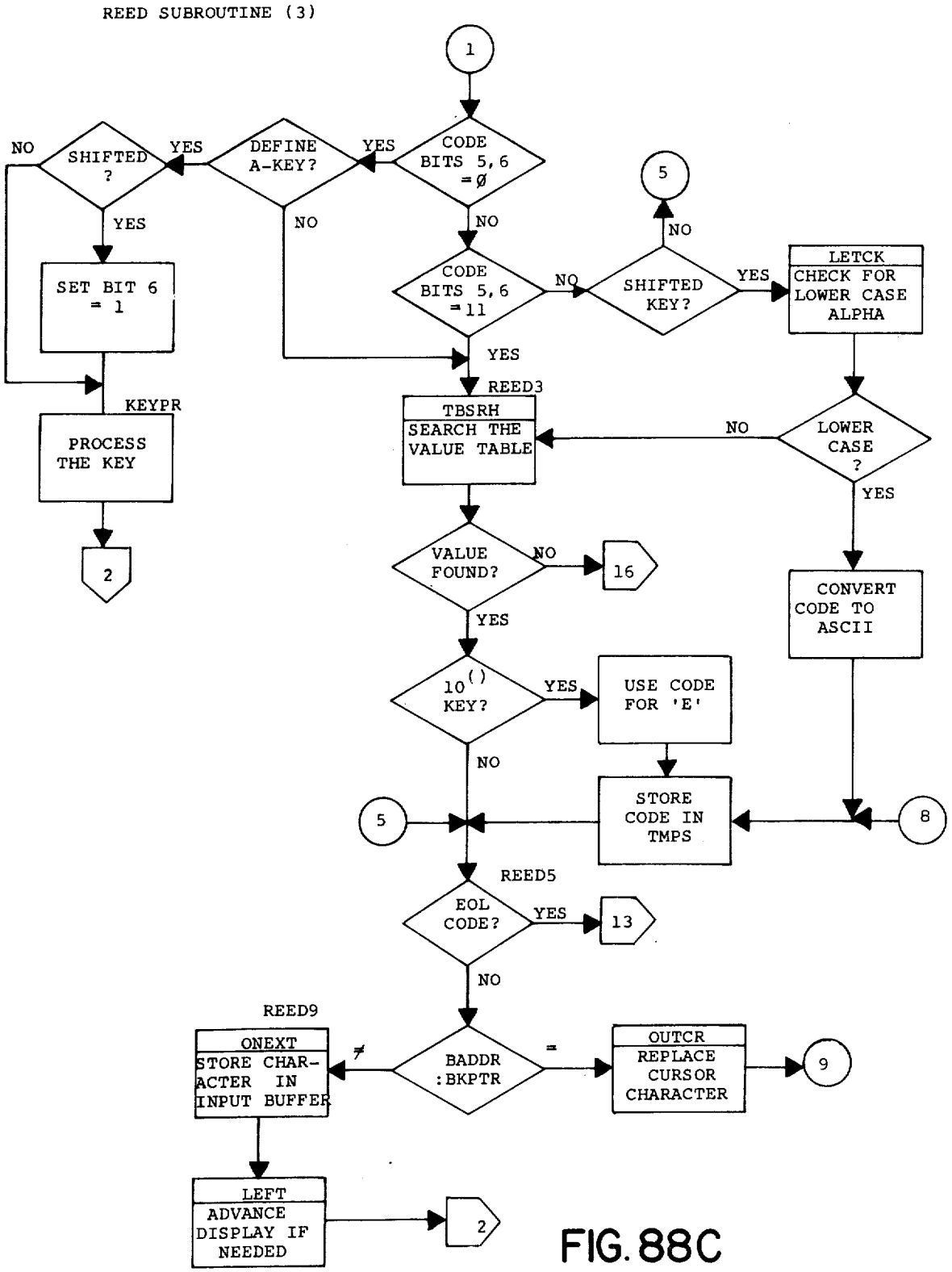

REED SUBROUTINE (3)
1
CODE BITS 5,6 =Ø
DEFINE A-KEY?
SHIFTED ?
SET BIT 6 = 1
KEYPR
PROCESS THE KEY
2
5
CODE BITS 5,6 =11
SHIFTED KEY?
LETCK
CHECK FOR LOWER CASE ALPHA
REED3
TBSRH
SEARCH THE VALUE TABLE
LOWER CASE ?
CONVERT CODE TO ASCII
VALUE FOUND?
16
10^( ) KEY?
USE CODE FOR 'E'
STORE CODE IN TMPS
8
REED5
EOL CODE?
13
REED9
ONEXT
STORE CHARACTER IN INPUT BUFFER
BADDR :BKPTR
OUTCR
REPLACE CURSOR CHARACTER
9
LEFT
ADVANCE DISPLAY IF NEEDED
YES
NO

FIG. 88C

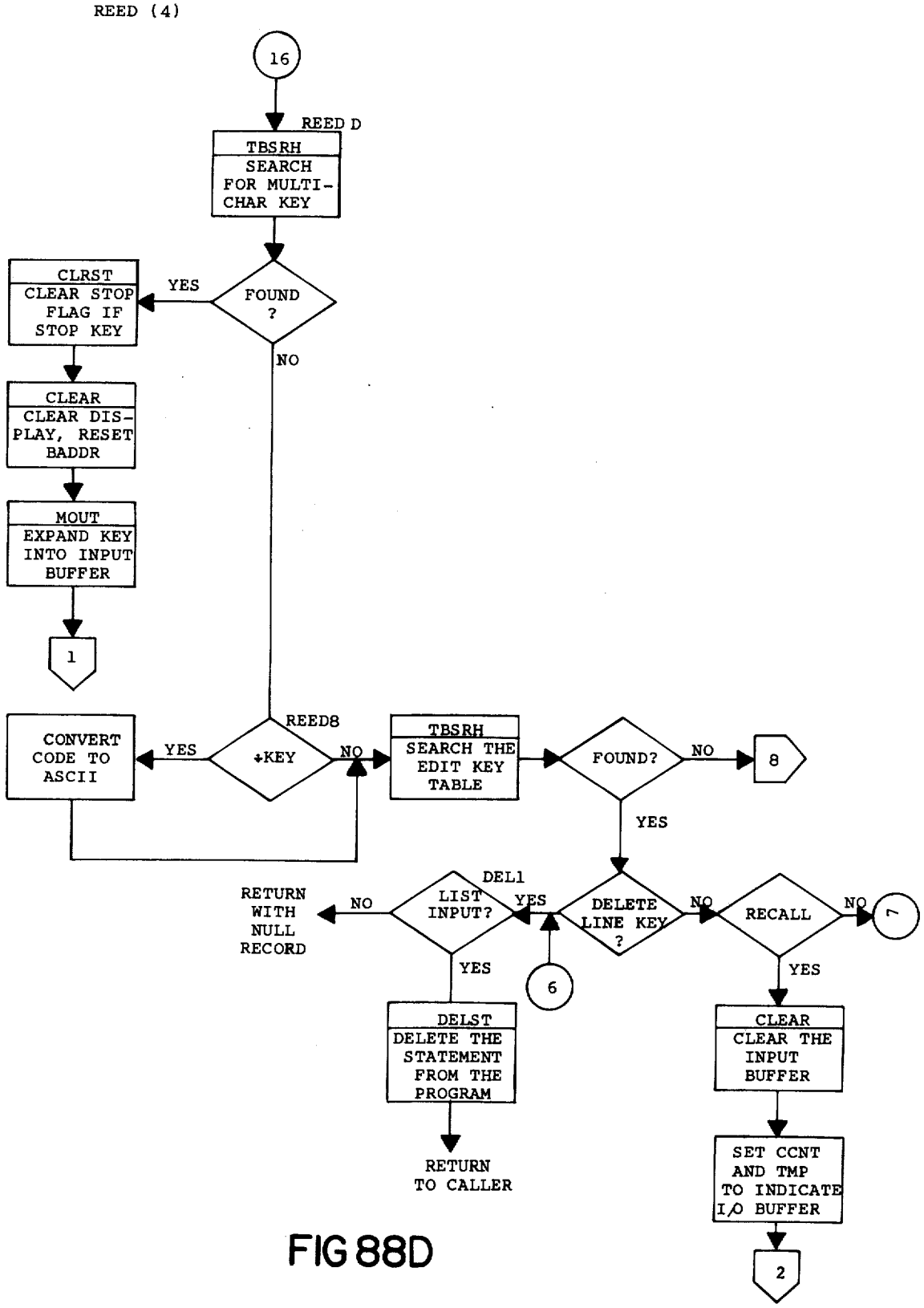

REED (4)
16
REED D
TBSRH
SEARCH FOR MULTI-CHAR KEY
FOUND ?
YES
CLRST
CLEAR STOP FLAG IF STOP KEY
NO
CLEAR
CLEAR DIS-PLAY, RESET BADDR
MOUT
EXPAND KEY INTO INPUT BUFFER
1
REED8
CONVERT CODE TO ASCII
YES
+KEY
NO
TBSRH
SEARCH THE EDIT KEY TABLE
FOUND?
NO
8
YES
DEL1
RETURN WITH NULL RECORD
NO
LIST INPUT?
YES
DELETE LINE KEY ?
NO
RECALL
NO
7
YES
6
YES
DELST
DELETE THE STATEMENT FROM THE PROGRAM
RETURN TO CALLER
CLEAR
CLEAR THE INPUT BUFFER
SET CCNT AND TMP TO INDICATE I/O BUFFER
2

FIG 88D

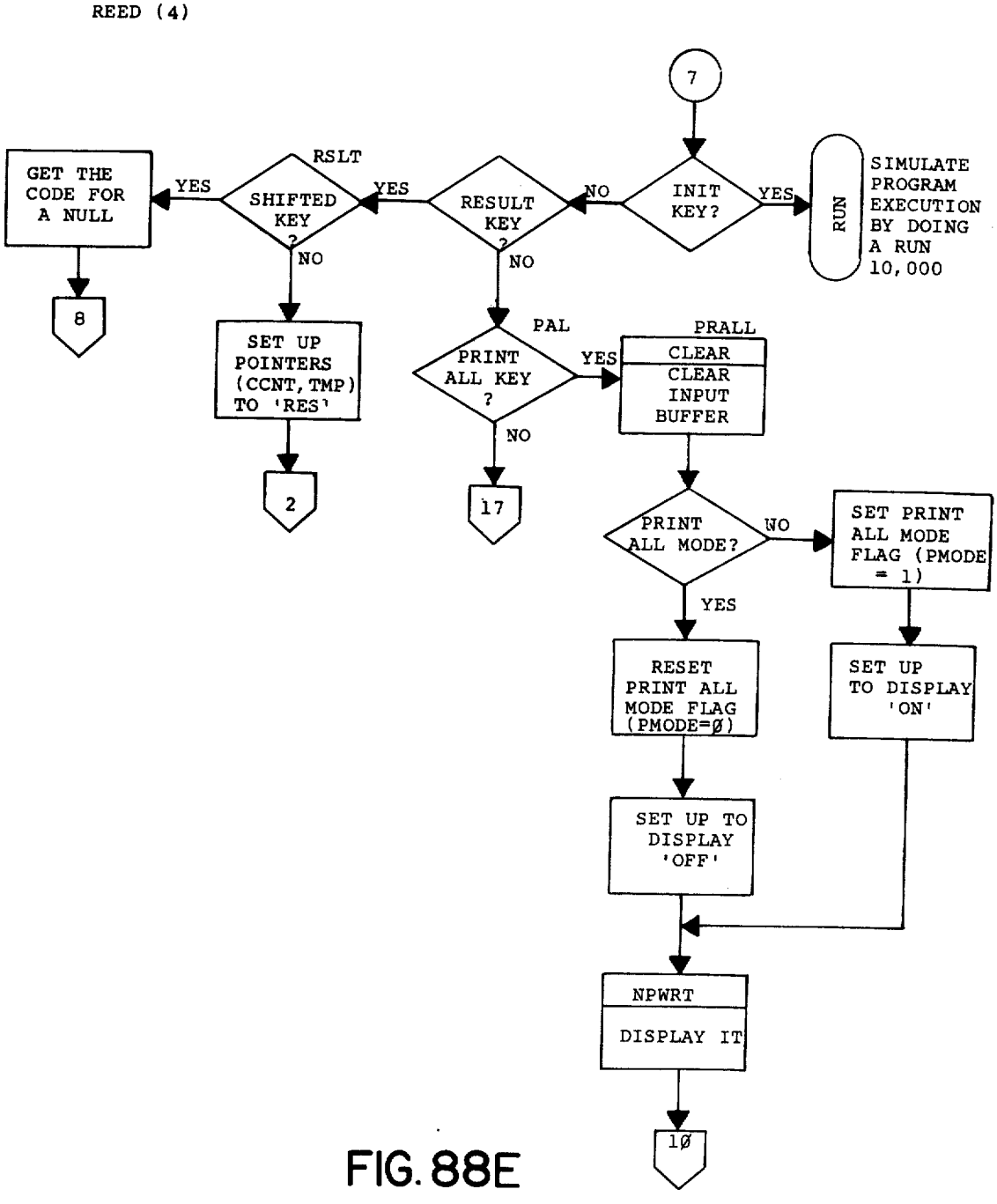

REED (4)
7
INIT KEY?
YES
RUN
SIMULATE PROGRAM EXECUTION BY DOING A RUN 10,000
NO
RESULT KEY ?
YES
RSLT
SHIFTED KEY ?
YES
GET THE CODE FOR A NULL
8
NO
SET UP POINTERS (CCNT, TMP) TO 'RES'
2
NO
PAL
PRINT ALL KEY ?
YES
PRALL
CLEAR
CLEAR INPUT BUFFER
NO
17
PRINT ALL MODE?
NO
SET PRINT ALL MODE FLAG (PMODE = 1)
YES
RESET PRINT ALL MODE FLAG (PMODE=Ø)
SET UP TO DISPLAY 'ON'
SET UP TO DISPLAY 'OFF'
NPWRT
DISPLAY IT
1Ø

FIG. 88E

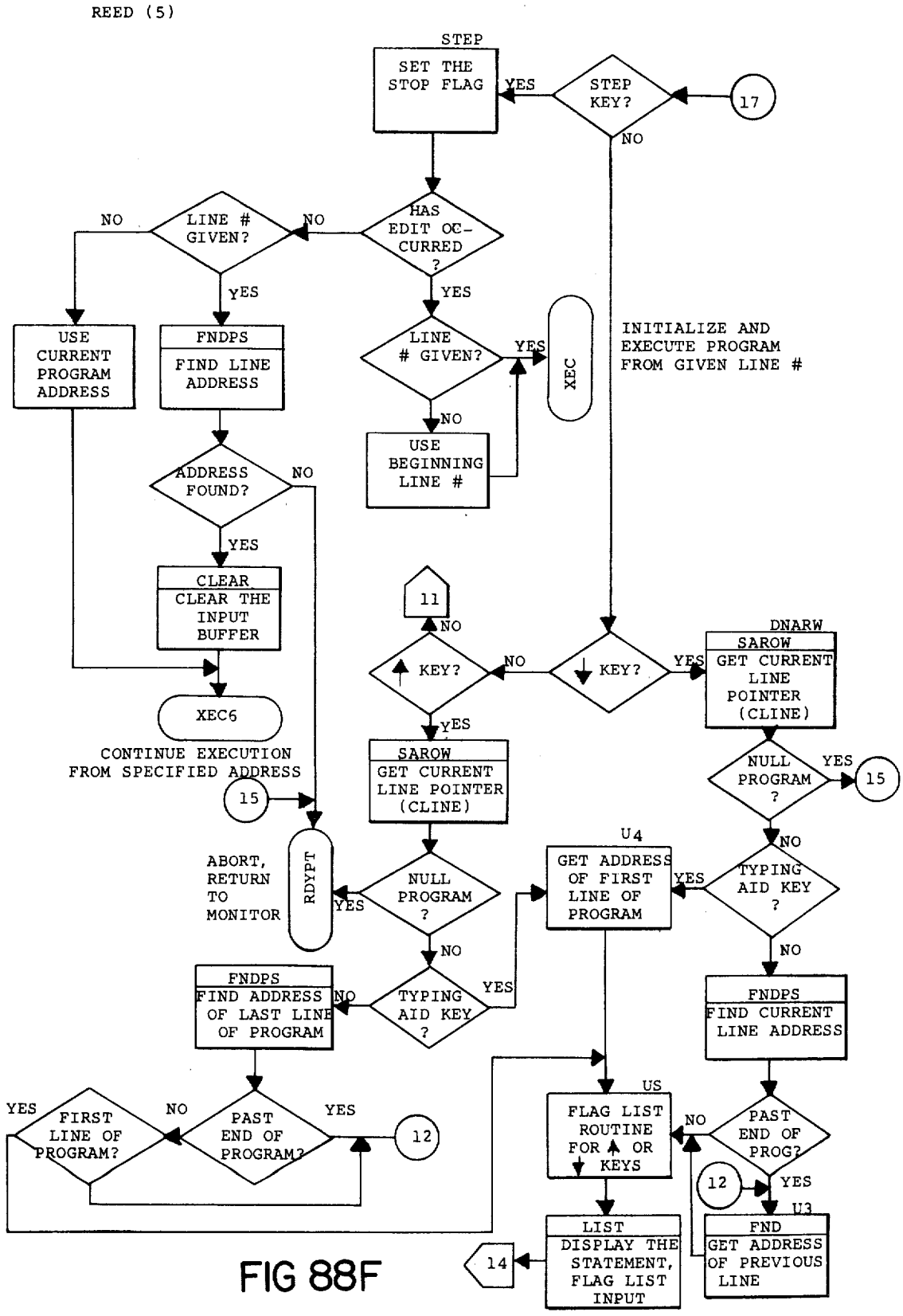

REED (5)
STEP
SET THE STOP FLAG
STEP KEY?
YES
NO
17
HAS EDIT OCCURRED ?
LINE # GIVEN?
NO
YES
USE CURRENT PROGRAM ADDRESS
FNDPS
FIND LINE ADDRESS
LINE # GIVEN?
YES
NO
XEC
USE BEGINNING LINE #
INITIALIZE AND EXECUTE PROGRAM FROM GIVEN LINE #
ADDRESS FOUND?
NO
YES
CLEAR
CLEAR THE INPUT BUFFER
XEC6
CONTINUE EXECUTION FROM SPECIFIED ADDRESS
15
11
KEY?
KEY?
DNARW
SAROW
GET CURRENT LINE POINTER (CLINE)
NULL PROGRAM ?
SAROW
GET CURRENT LINE POINTER (CLINE)
ABORT, RETURN TO MONITOR
RDYPT
NULL PROGRAM ?
U4
GET ADDRESS OF FIRST LINE OF PROGRAM
TYPING AID KEY ?
FNDPS
FIND ADDRESS OF LAST LINE OF PROGRAM
TYPING AID KEY ?
FNDPS
FIND CURRENT LINE ADDRESS
US
FLAG LIST ROUTINE FOR ↑ OR ↓ KEYS
PAST END OF PROG?
FIRST LINE OF PROGRAM?
PAST END OF PROGRAM?
12
U3
FND
GET ADDRESS OF PREVIOUS LINE
LIST
DISPLAY THE STATEMENT, FLAG LIST INPUT
14

FIG 88F

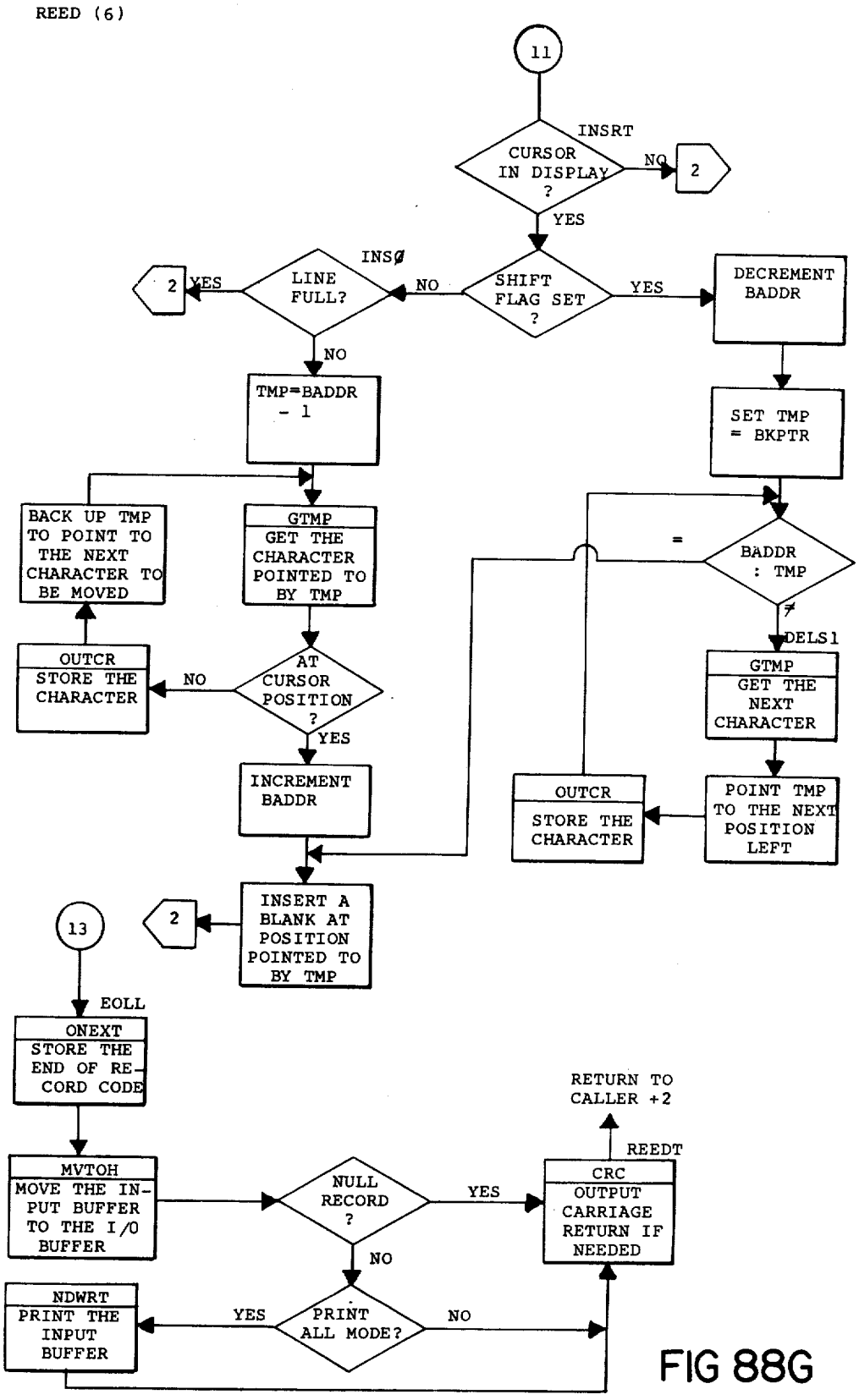

REED (6)
11
INSRT
CURSOR IN DISPLAY ?
NO
2
YES
INSØ
SHIFT FLAG SET ?
NO
YES
DECREMENT BADDR
2
YES
LINE FULL?
NO
TMP=BADDR - 1
SET TMP = BKPTR
BACK UP TMP TO POINT TO THE NEXT CHARACTER TO BE MOVED
GTMP
GET THE CHARACTER POINTED TO BY TMP
=
BADDR : TMP
≠
DELS1
OUTCR
STORE THE CHARACTER
NO
AT CURSOR POSITION ?
YES
GTMP
GET THE NEXT CHARACTER
INCREMENT BADDR
OUTCR
STORE THE CHARACTER
POINT TMP TO THE NEXT POSITION LEFT
INSERT A BLANK AT POSITION POINTED TO BY TMP
2
13
EOLL
ONEXT
STORE THE END OF RECORD CODE
RETURN TO CALLER +2
REEDT
MVTOH
MOVE THE INPUT BUFFER TO THE I/O BUFFER
NULL RECORD ?
YES
CRC
OUTPUT CARRIAGE RETURN IF NEEDED
NO
NDWRT
PRINT THE INPUT BUFFER
YES
PRINT ALL MODE?
NO

FIG 88G

LEFT SUBROUTINE

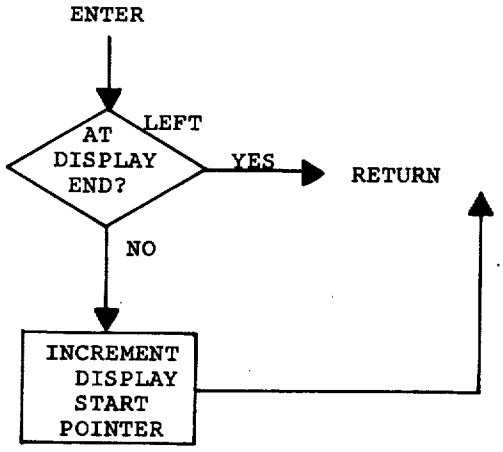


FIG 89A

RIGHT SUBROUTINE

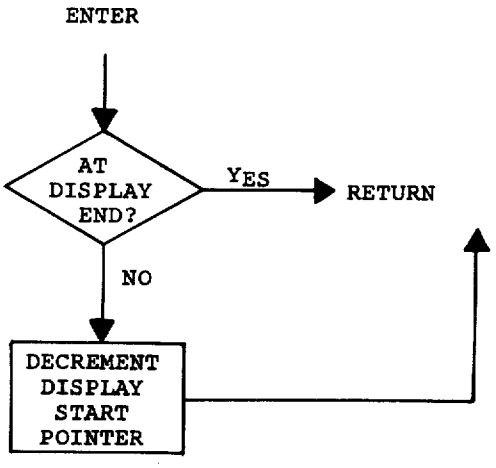


FIG 89B

CLEAR SUBROUTINE

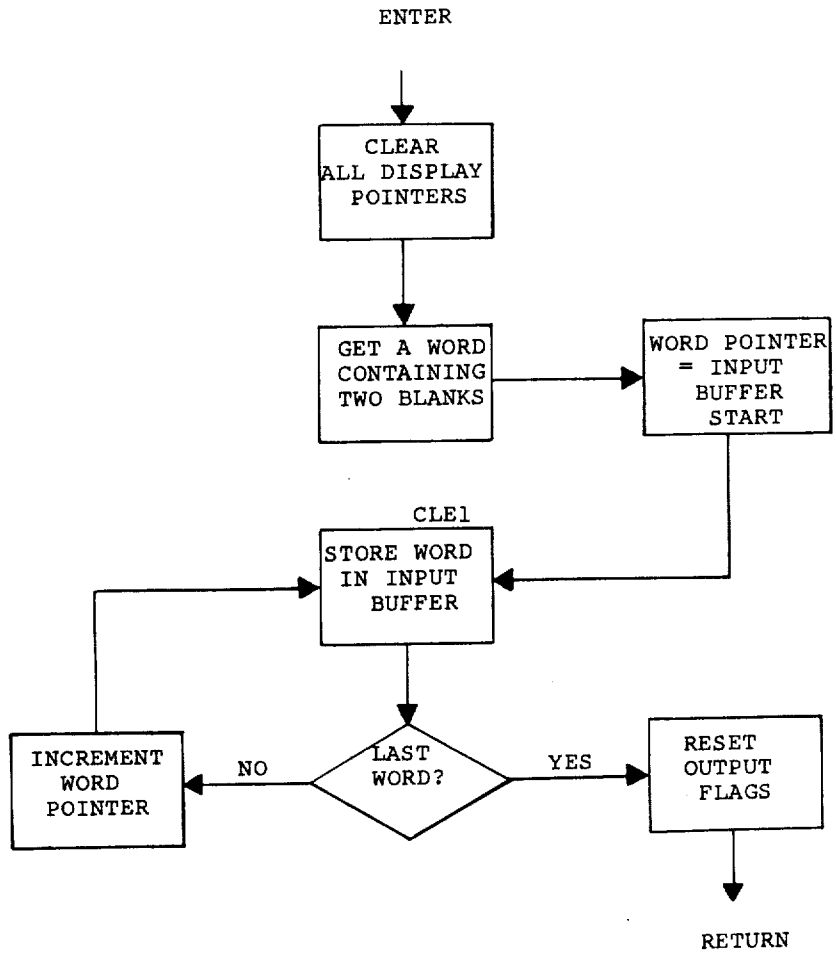


FIG 90

SAROW SUBROUTINE

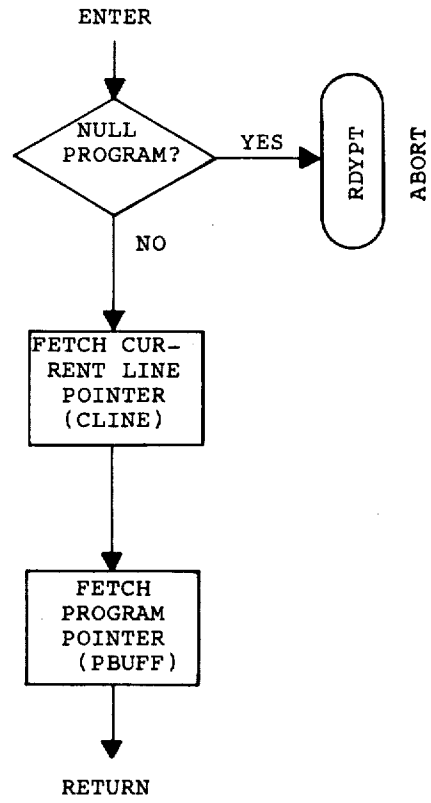


FIG 91

CALLK SUBROUTINE

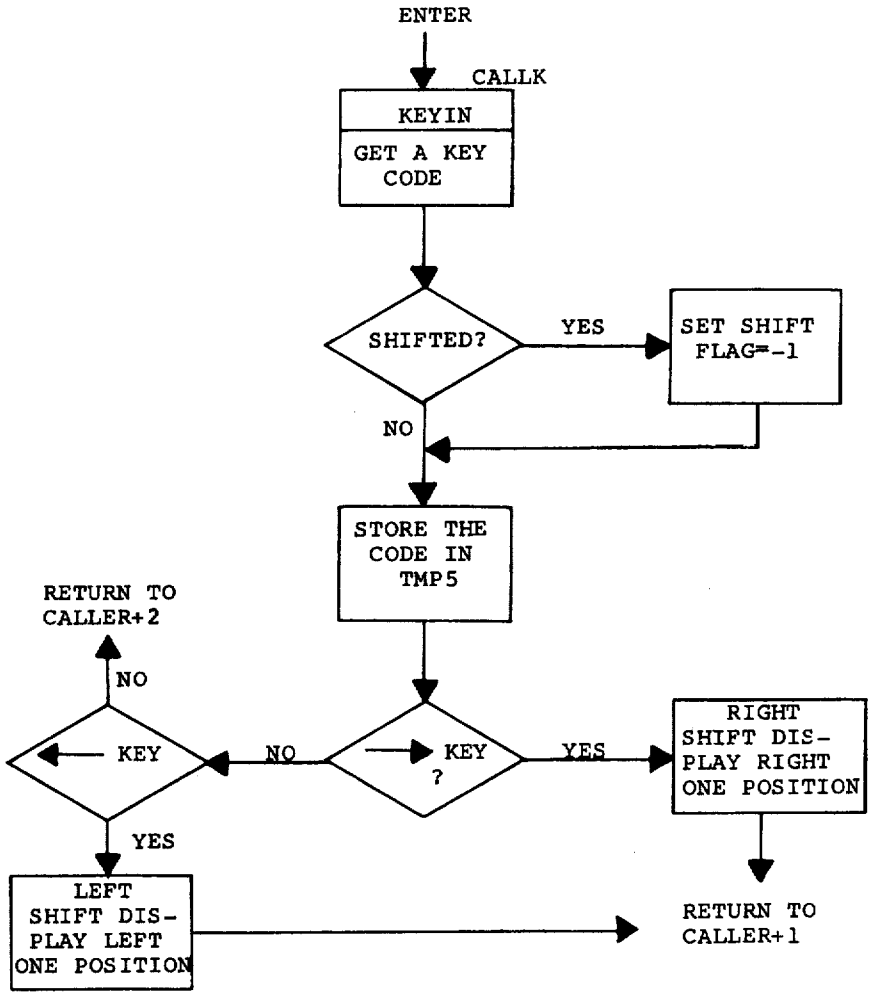


FIG. 92

KEYIN SUBROUTINE

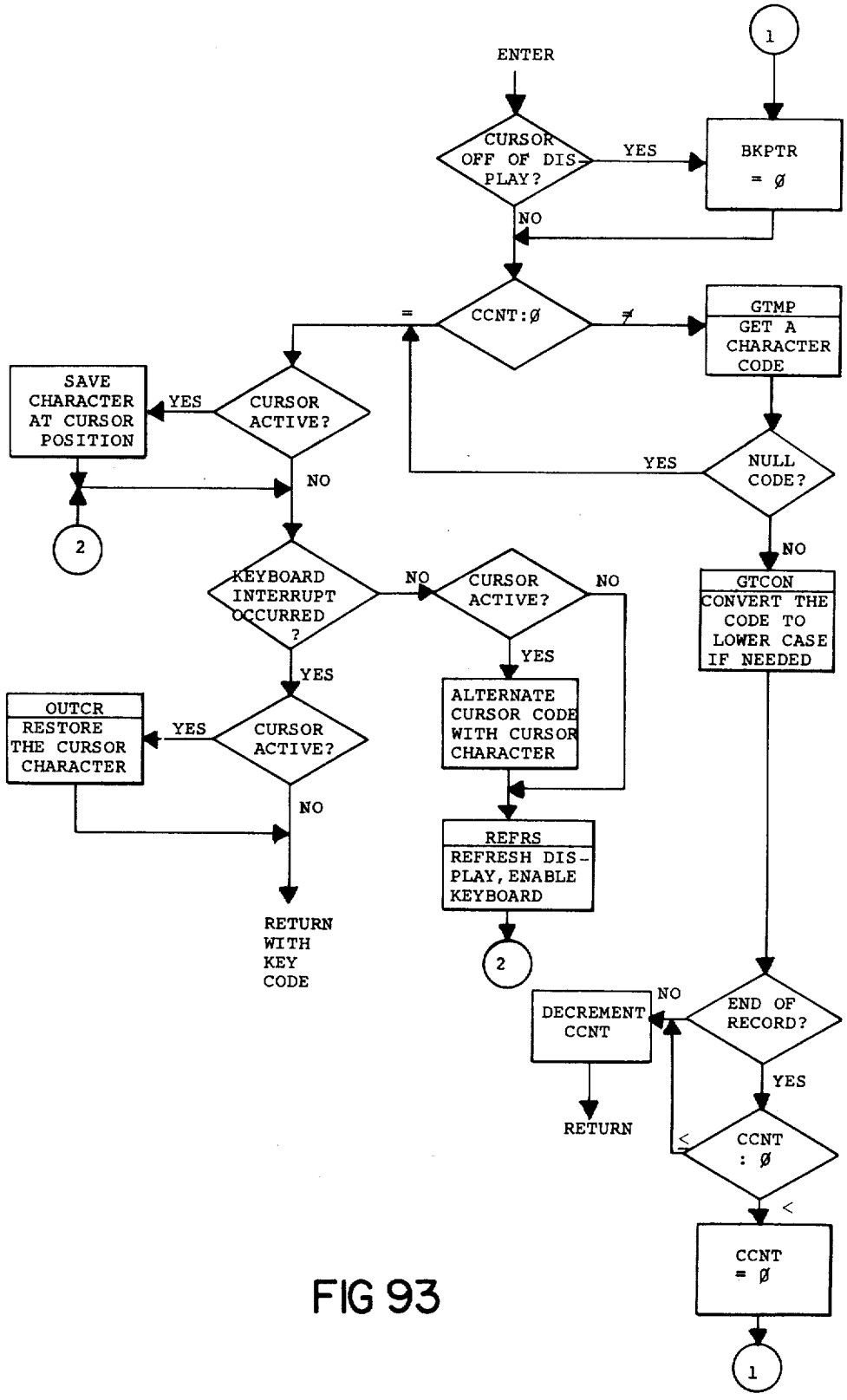
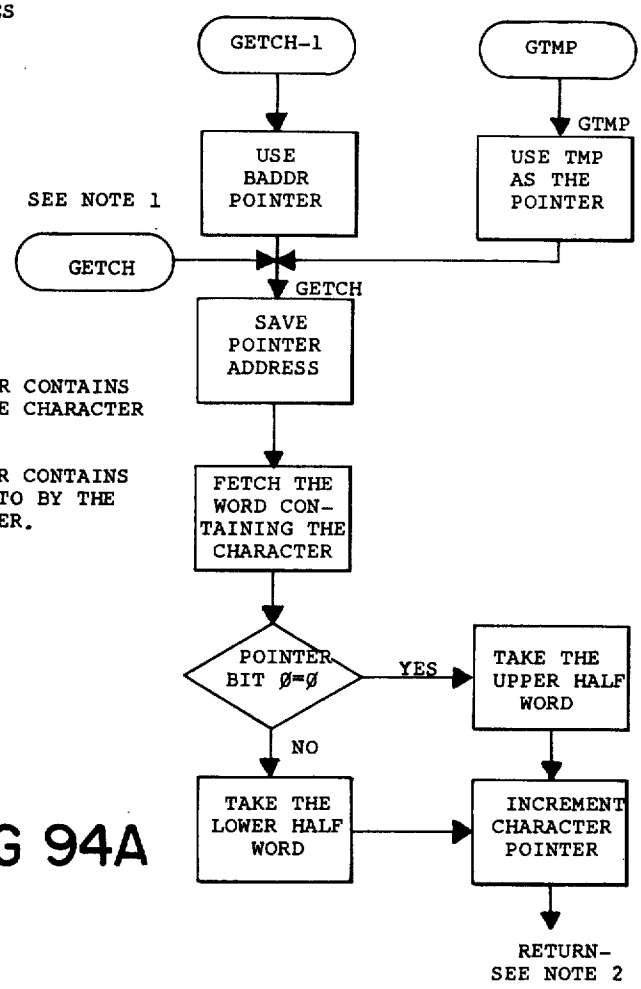


FIG 93

GETCH
GTMP GET CHARACTER
GNEXT SUBROUTINES
GTCON



NOTE 1: THE B REGISTER CONTAINS THE WORD ADDRESS OF THE CHARACTER POINTER UPON ENTRY.

NOTE 2: THE A REGISTER CONTAINS THE CHARACTER POINTED TO BY THE PASSED CHARACTER POINTER.

FIG 94A

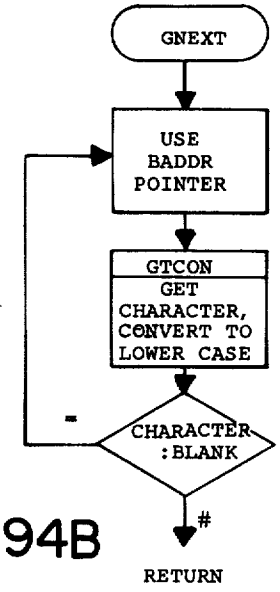


FIG 94B

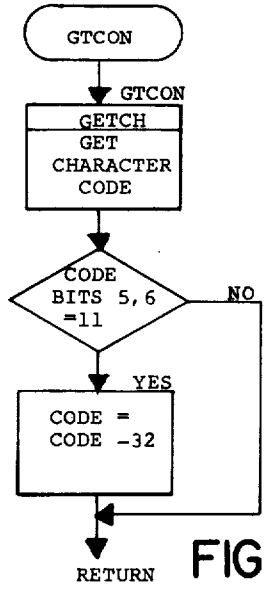


FIG 94C

OBLNK
ONEXT OUTPUT SUBROUTINES
OUTCR

NOTE 1: UPON ENTERING OUTCR THE A REGISTER HAS THE CHARACTER TO BE STORED, THE B REGISTER HAS THE ADDRESS OF THE CHARACTER POINTER WHERE THE CHARACTER IS TO BE STORED.

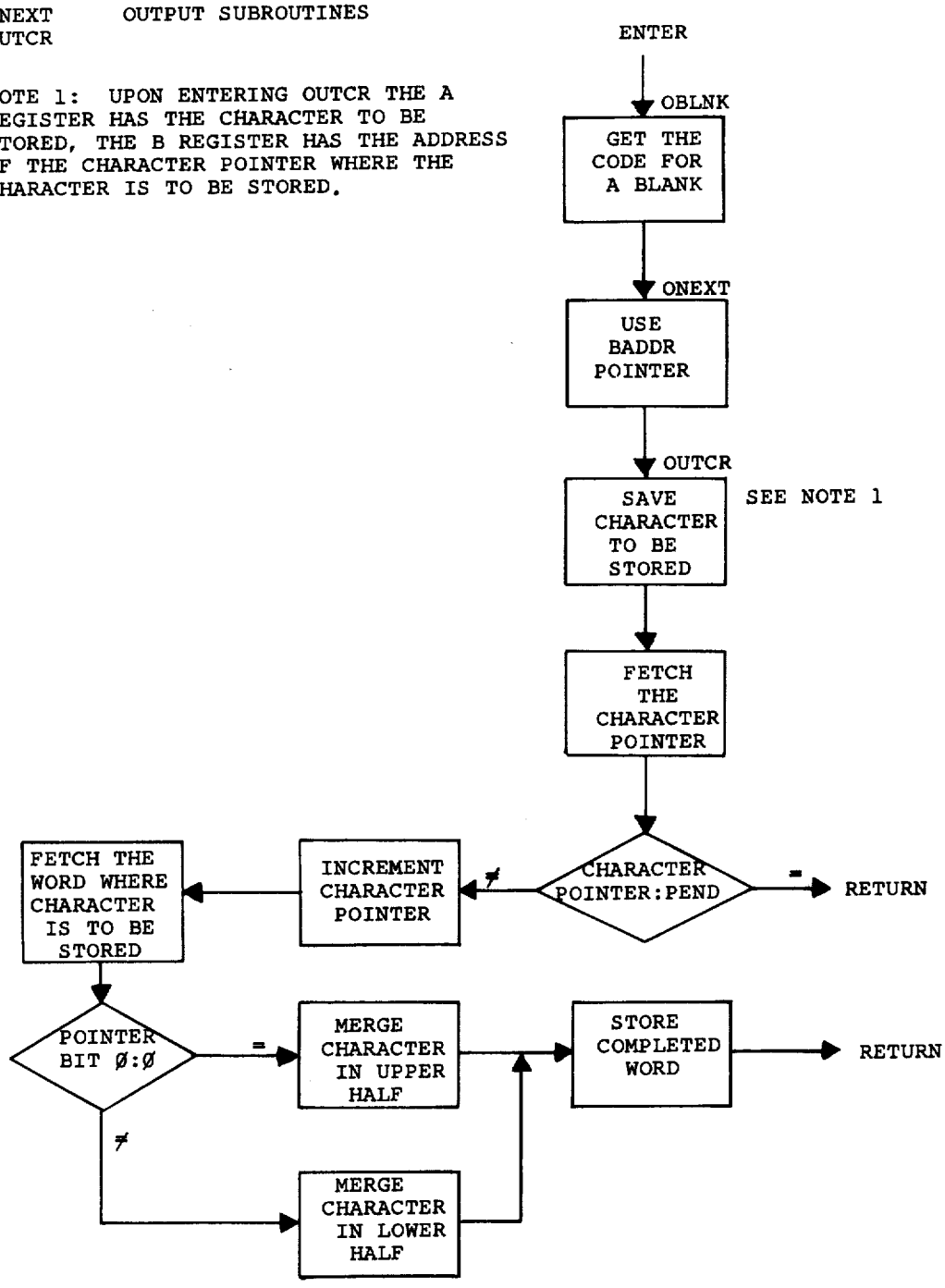


FIG. 94D

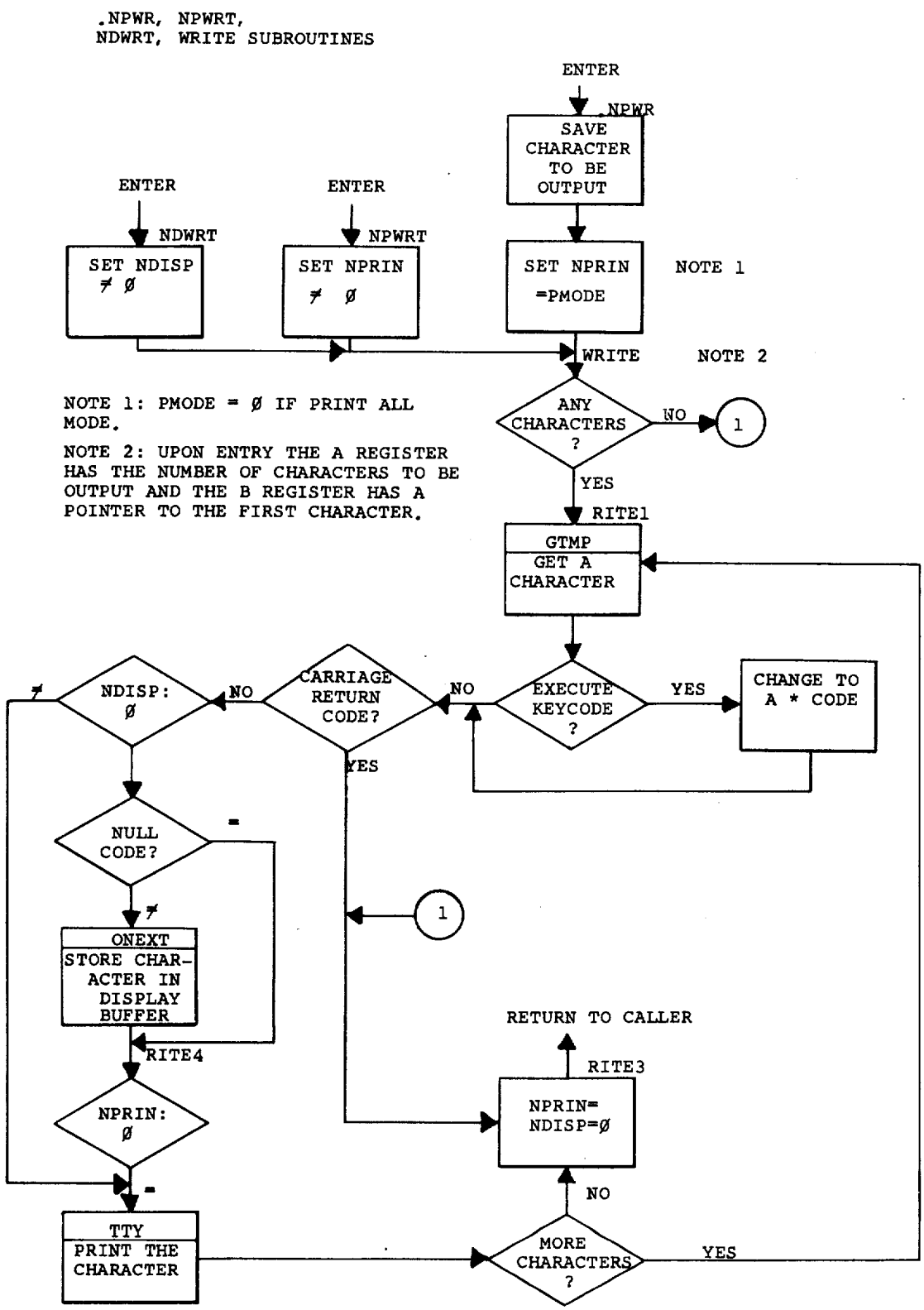

.NPWR, NPWRT,
NDWRT, WRITE SUBROUTINES
ENTER
.NPWR
SAVE CHARACTER TO BE OUTPUT
ENTER
NDWRT
SET NDISP ≠ Ø
ENTER
NPWRT
SET NPRIN ≠ Ø
SET NPRIN =PMODE
NOTE 1
WRITE
NOTE 2
NOTE 1: PMODE = Ø IF PRINT ALL MODE.
NOTE 2: UPON ENTRY THE A REGISTER HAS THE NUMBER OF CHARACTERS TO BE OUTPUT AND THE B REGISTER HAS A POINTER TO THE FIRST CHARACTER.
ANY CHARACTERS ?
NO
1
YES
RITE1
GTMP
GET A CHARACTER
EXECUTE KEYCODE ?
YES
CHANGE TO A * CODE
NO
CARRIAGE RETURN CODE?
NO
NDISP: Ø
≠
YES
NULL CODE?
=
≠
1
ONEXT
STORE CHARACTER IN DISPLAY BUFFER
RITE4
NPRIN: Ø
=
RETURN TO CALLER
RITE3
NPRIN= NDISP=Ø
NO
TTY
PRINT THE CHARACTER
MORE CHARACTERS ?
YES

FIG 94E

TTY SUBROUTINE

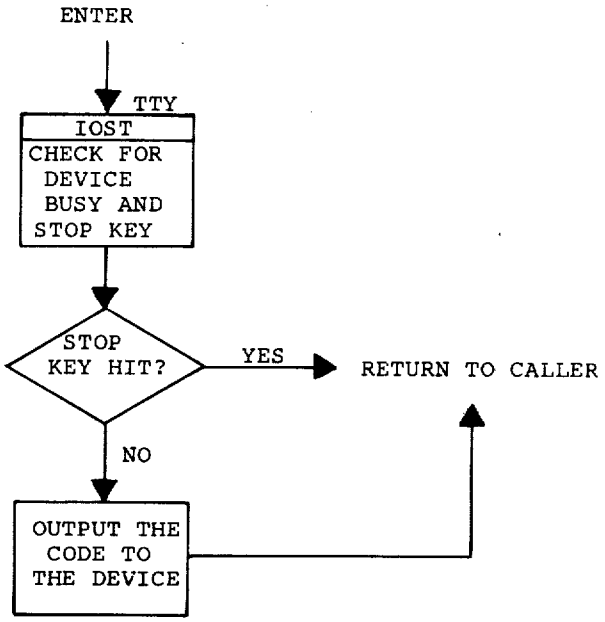


FIG 94F

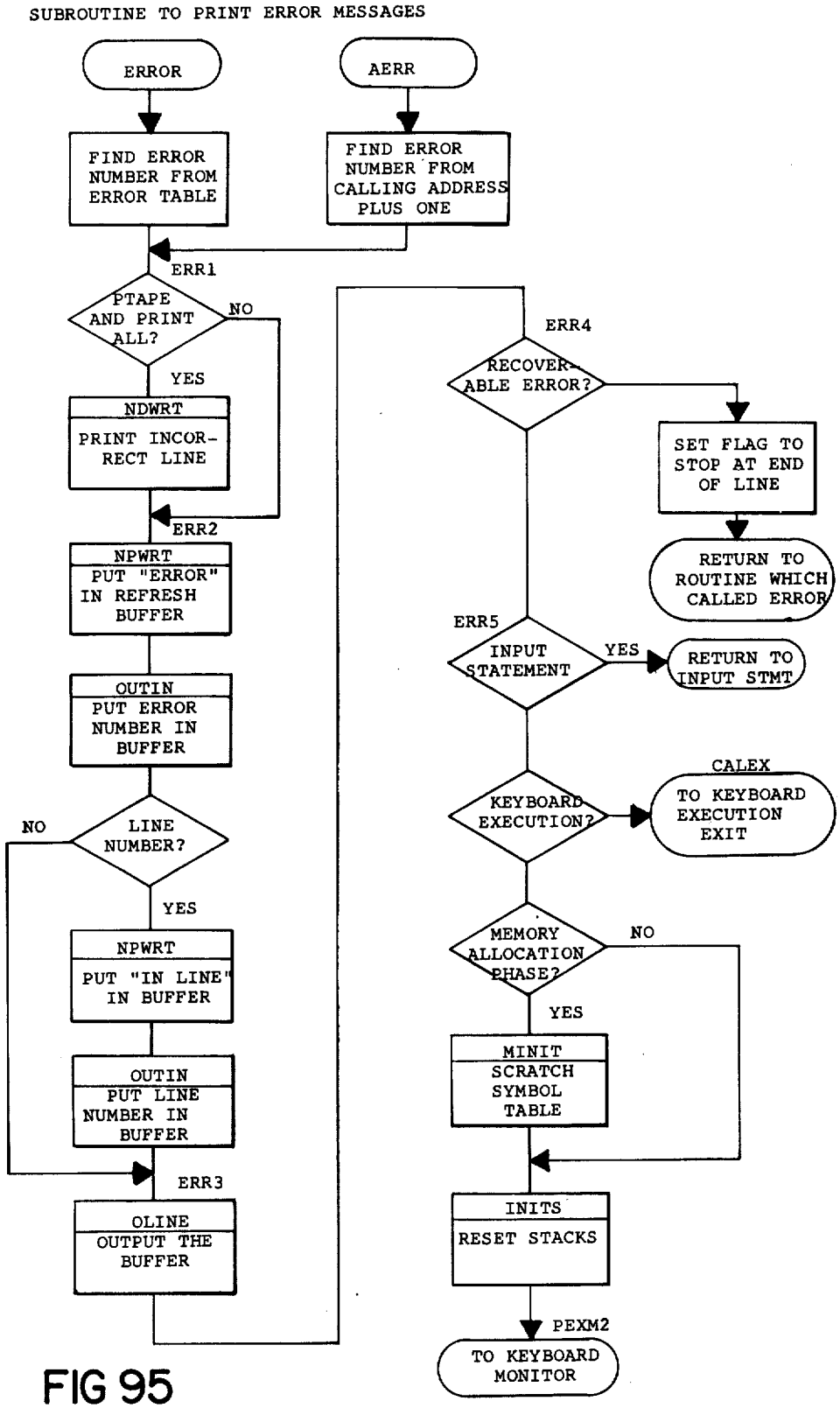

SUBROUTINE TO PRINT ERROR MESSAGES
ERROR
AERR
FIND ERROR NUMBER FROM ERROR TABLE
FIND ERROR NUMBER FROM CALLING ADDRESS PLUS ONE
ERR1
PTAPE AND PRINT ALL?
NO
YES
NDWRT
PRINT INCORRECT LINE
ERR2
NPWRT
PUT "ERROR" IN REFRESH BUFFER
OUTIN
PUT ERROR NUMBER IN BUFFER
NO
LINE NUMBER?
YES
NPWRT
PUT "IN LINE" IN BUFFER
OUTIN
PUT LINE NUMBER IN BUFFER
ERR3
OLINE
OUTPUT THE BUFFER
ERR4
RECOVERABLE ERROR?
SET FLAG TO STOP AT END OF LINE
RETURN TO ROUTINE WHICH CALLED ERROR
ERR5
INPUT STATEMENT
YES
RETURN TO INPUT STMT
CALEX
KEYBOARD EXECUTION?
TO KEYBOARD EXECUTION EXIT
MEMORY ALLOCATION PHASE?
NO
YES
MINIT
SCRATCH SYMBOL TABLE
INITS
RESET STACKS
PEXM2
TO KEYBOARD MONITOR

FIG 95

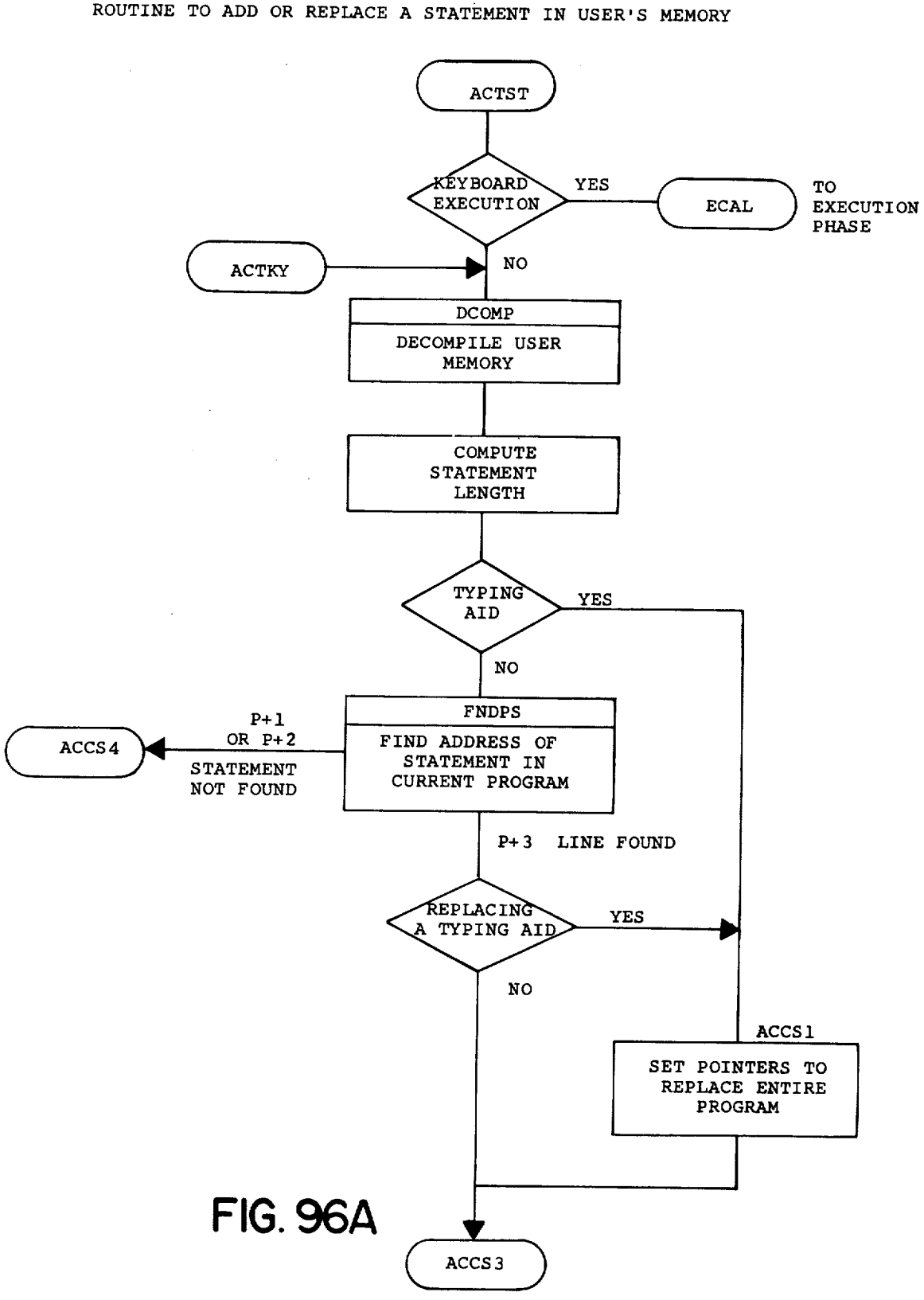

ROUTINE TO ADD OR REPLACE A STATEMENT IN USER'S MEMORY
ACTST
KEYBOARD EXECUTION
YES
ECAL
TO EXECUTION PHASE
ACTKY
NO
DCOMP
DECOMPILE USER MEMORY
COMPUTE STATEMENT LENGTH
TYPING AID
YES
NO
FNDPS
FIND ADDRESS OF STATEMENT IN CURRENT PROGRAM
P+1 OR P+2
STATEMENT NOT FOUND
ACCS4
P+3 LINE FOUND
REPLACING A TYPING AID
YES
NO
ACCS1
SET POINTERS TO REPLACE ENTIRE PROGRAM
ACCS3

FIG. 96A

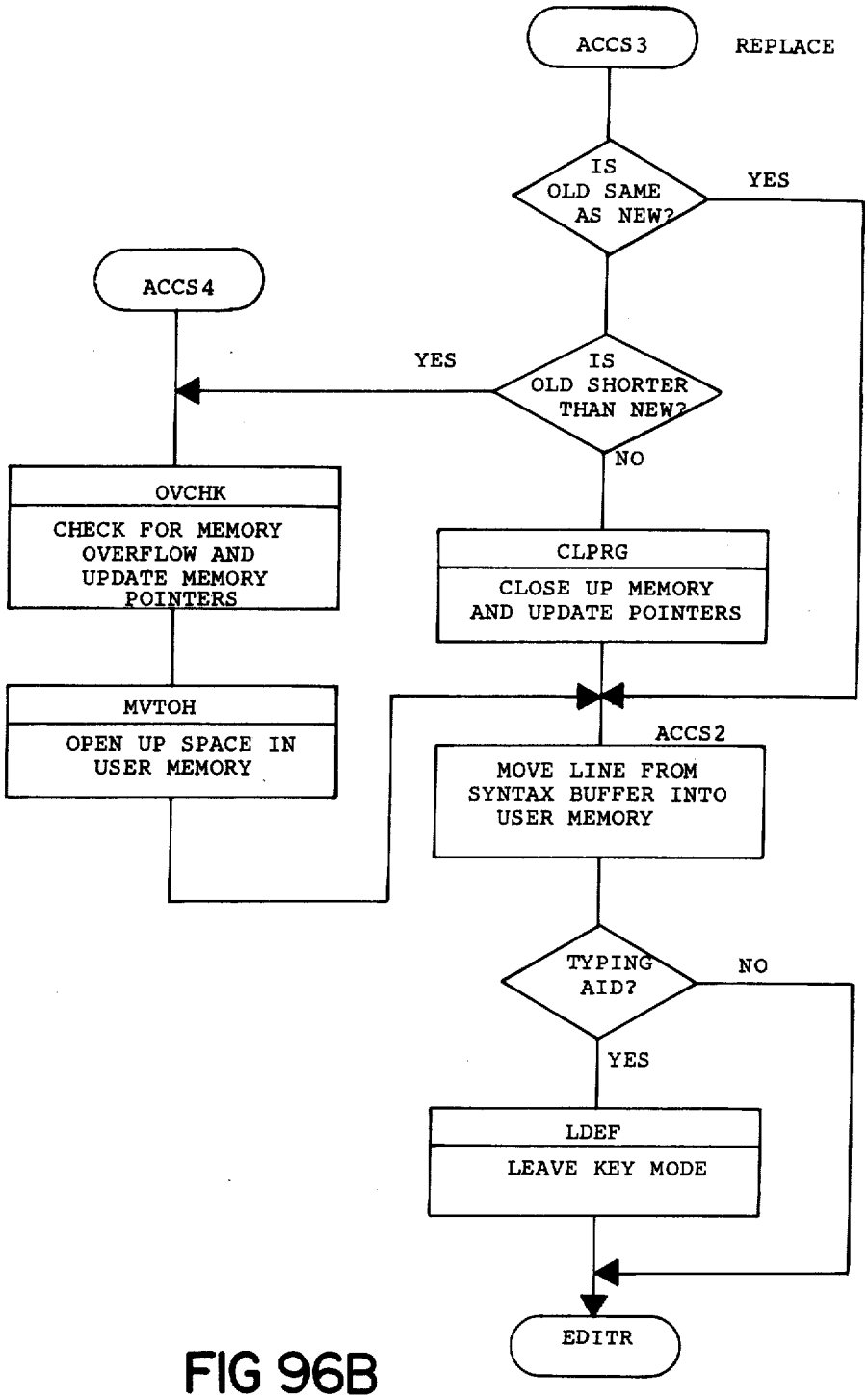

ACCS3
REPLACE
IS OLD SAME AS NEW?
YES
ACCS4
YES
IS OLD SHORTER THAN NEW?
NO
OVCHK
CHECK FOR MEMORY OVERFLOW AND UPDATE MEMORY POINTERS
CLPRG
CLOSE UP MEMORY AND UPDATE POINTERS
MVTOH
OPEN UP SPACE IN USER MEMORY
ACCS2
MOVE LINE FROM SYNTAX BUFFER INTO USER MEMORY
TYPING AID?
NO
YES
LDEF
LEAVE KEY MODE
EDITR

FIG 96B

ROUTINE TO DELETE A STATEMENT FROM USER'S MEMORY

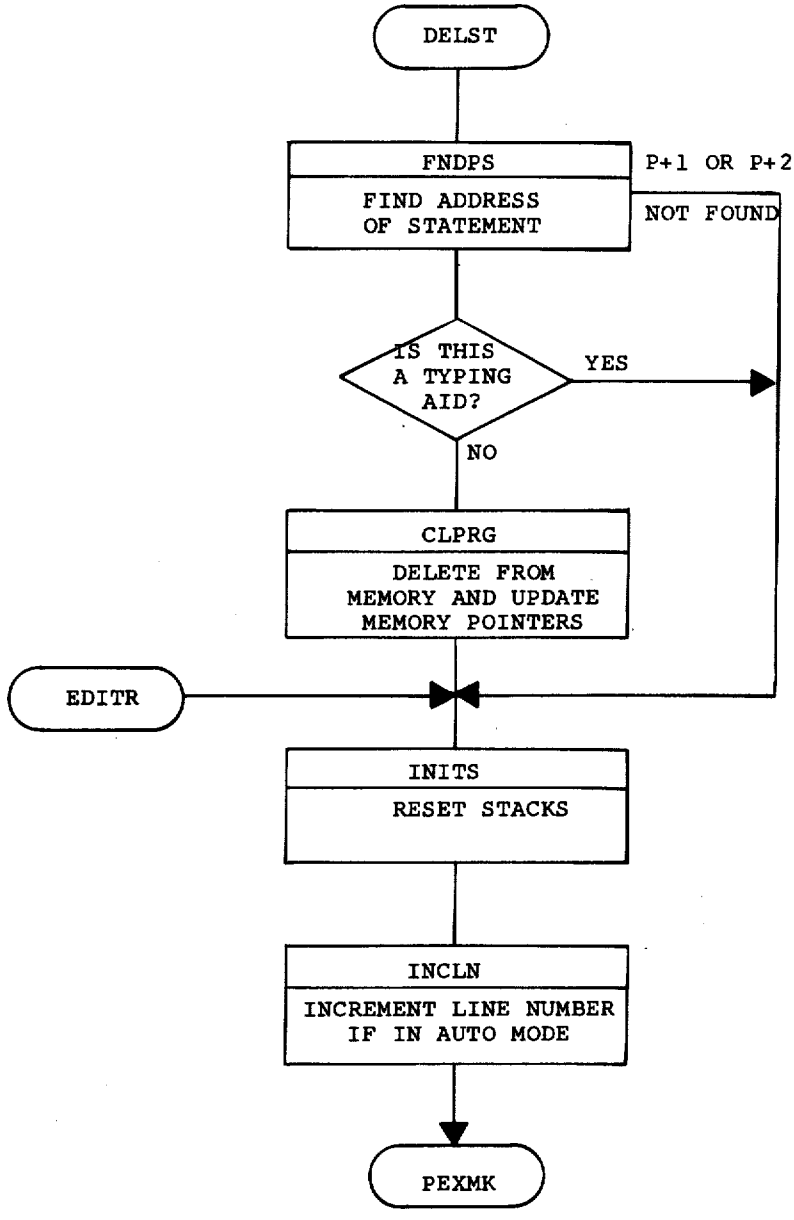


FIG 96C

SUBROUTINES TO ADJUST USER MEMORY AND UPDATE USER MEMORY POINTERS

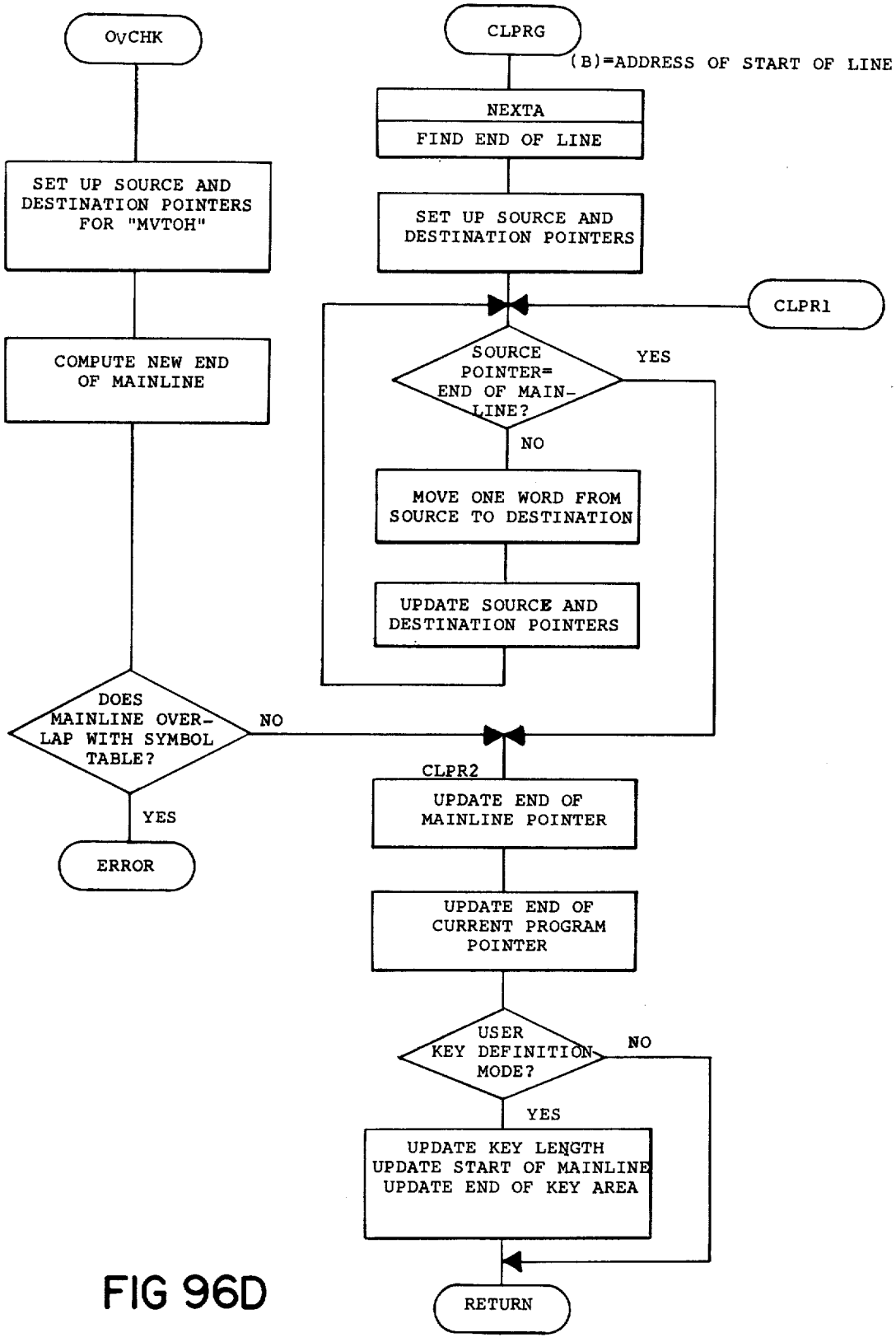


FIG 96D

MVTOH SUBROUTINE

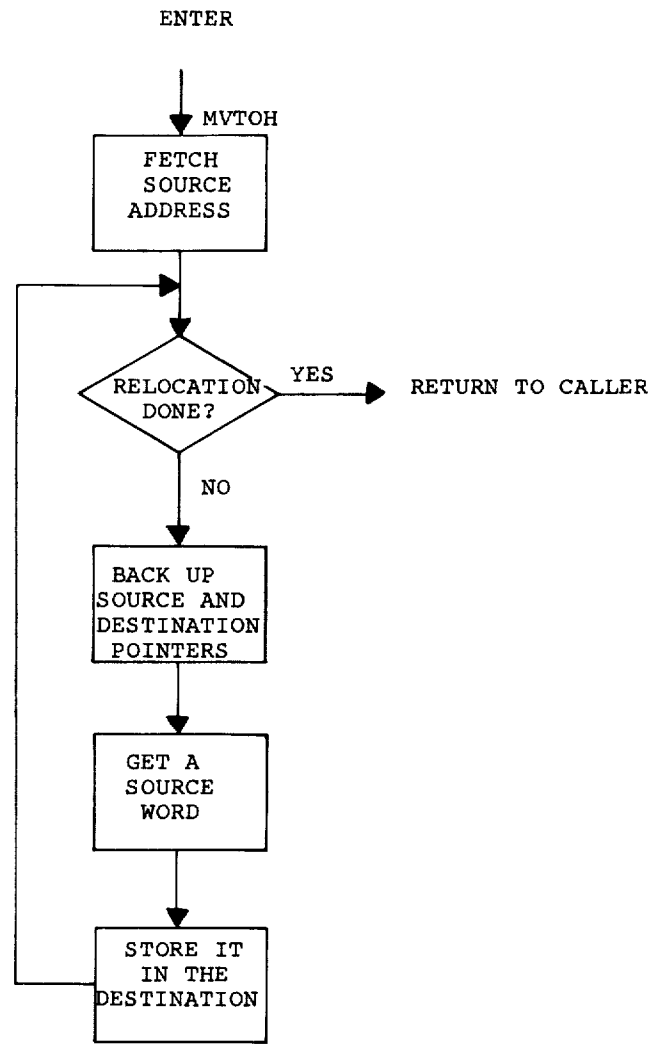


FIG. 97

FNDPS SUBROUTINE

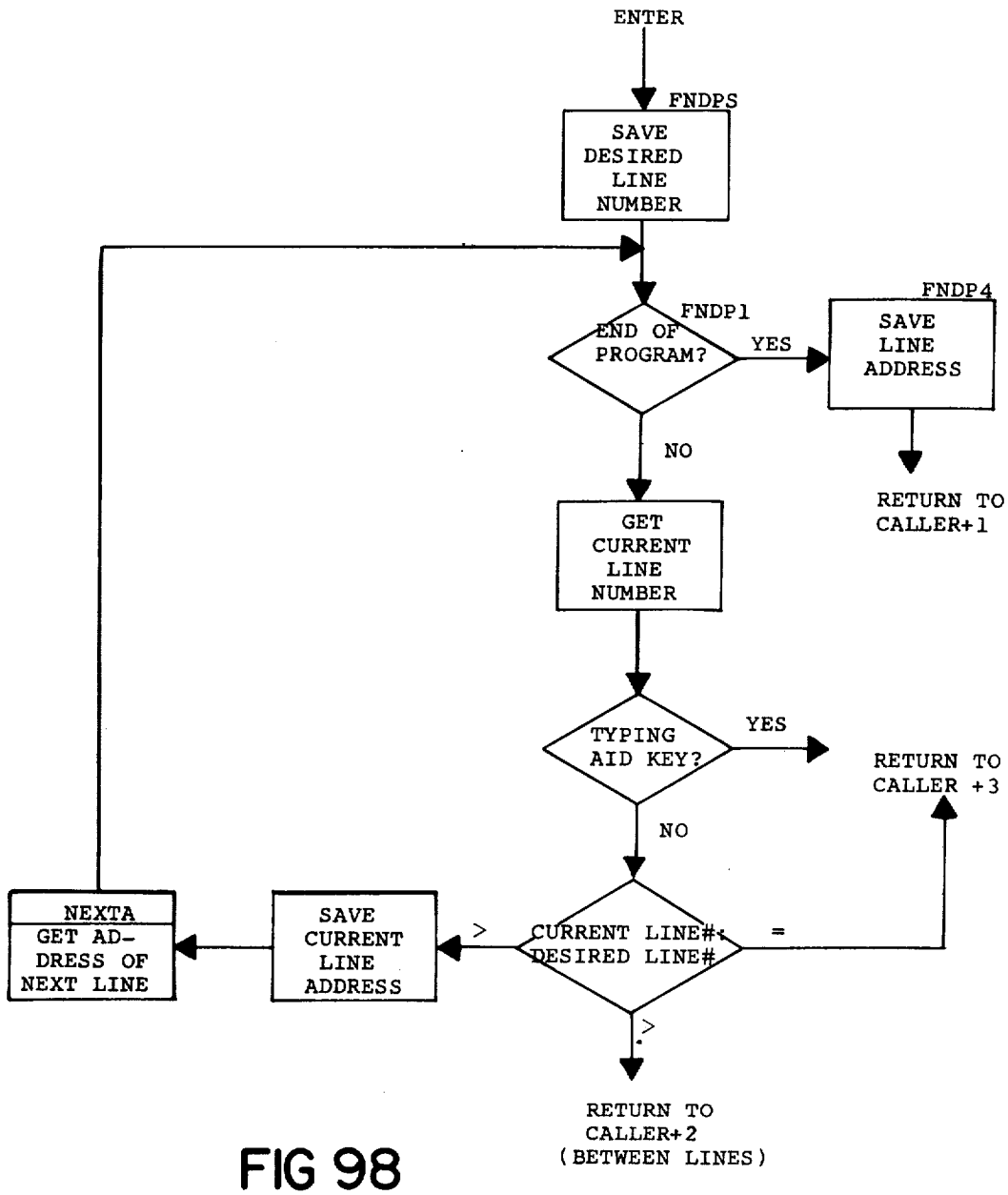


FIG 98

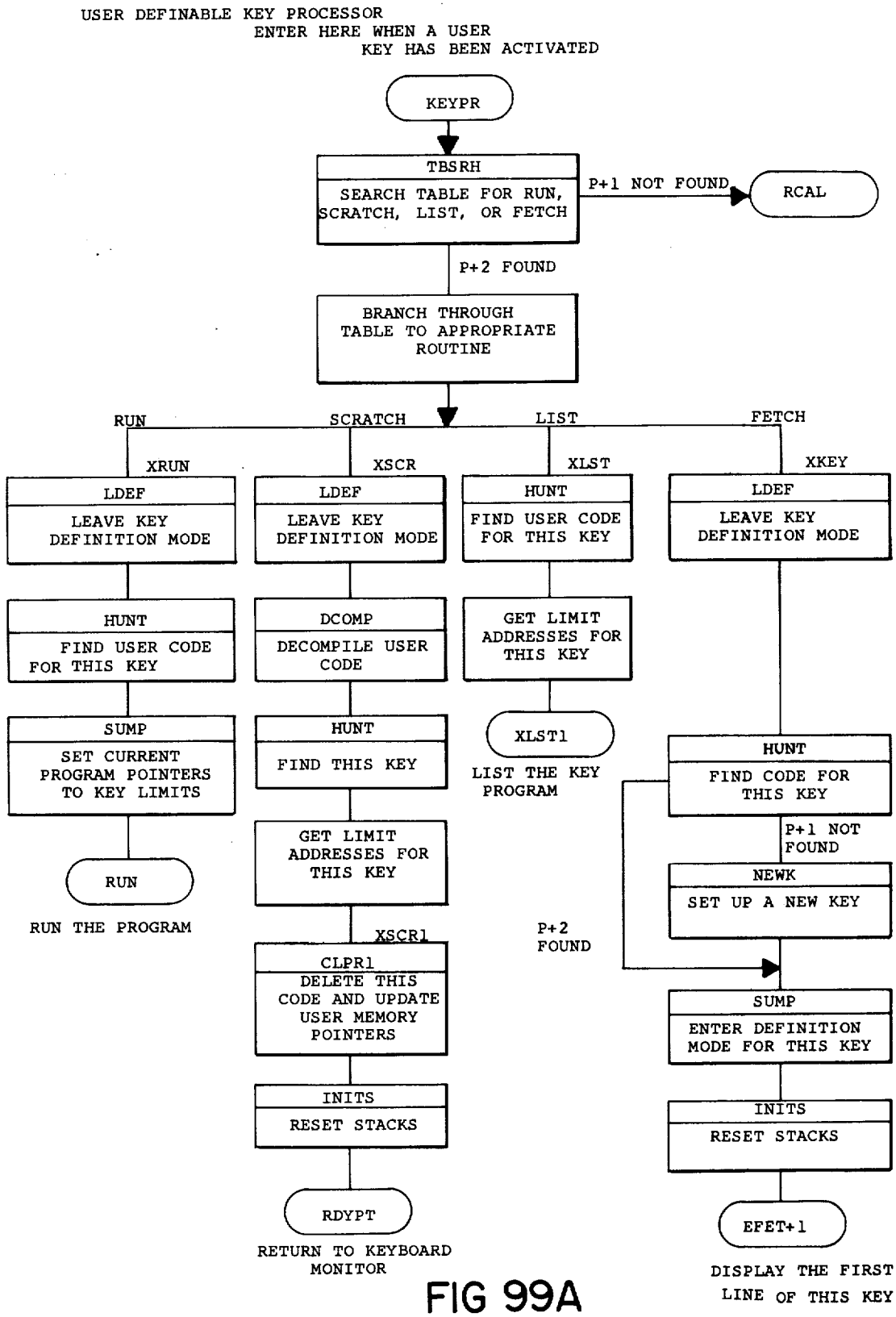

USER DEFINABLE KEY PROCESSOR
ENTER HERE WHEN A USER KEY HAS BEEN ACTIVATED
KEYPR
TBSRH
SEARCH TABLE FOR RUN, SCRATCH, LIST, OR FETCH
P+1 NOT FOUND
RCAL
P+2 FOUND
BRANCH THROUGH TABLE TO APPROPRIATE ROUTINE
RUN
SCRATCH
LIST
FETCH
XRUN
XSCR
XLST
XKEY
LDEF
LEAVE KEY DEFINITION MODE
LDEF
LEAVE KEY DEFINITION MODE
HUNT
FIND USER CODE FOR THIS KEY
LDEF
LEAVE KEY DEFINITION MODE
HUNT
FIND USER CODE FOR THIS KEY
DCOMP
DECOMPILE USER CODE
GET LIMIT ADDRESSES FOR THIS KEY
SUMP
SET CURRENT PROGRAM POINTERS TO KEY LIMITS
HUNT
FIND THIS KEY
XLST1
LIST THE KEY PROGRAM
HUNT
FIND CODE FOR THIS KEY
RUN
RUN THE PROGRAM
GET LIMIT ADDRESSES FOR THIS KEY
P+1 NOT FOUND
NEWK
SET UP A NEW KEY
XSCR1
P+2 FOUND
CLPR1
DELETE THIS CODE AND UPDATE USER MEMORY POINTERS
SUMP
ENTER DEFINITION MODE FOR THIS KEY
INITS
RESET STACKS
INITS
RESET STACKS
RDYPT
RETURN TO KEYBOARD MONITOR
EFET+1
DISPLAY THE FIRST LINE OF THIS KEY

FIG 99A

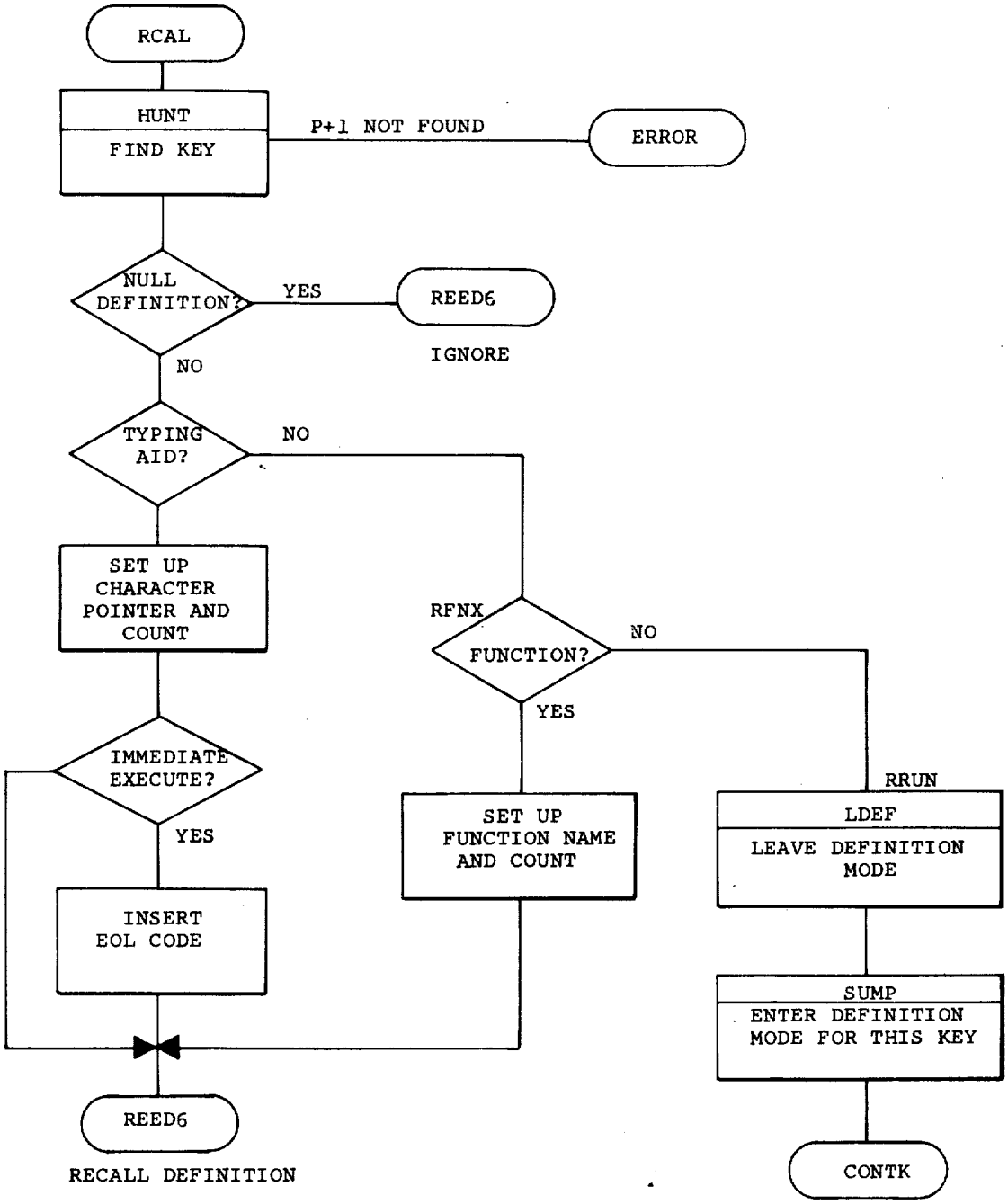

RCAL
HUNT
FIND KEY
P+1 NOT FOUND
ERROR
NULL DEFINITION?
YES
REED6
IGNORE
NO
TYPING AID?
NO
SET UP CHARACTER POINTER AND COUNT
RFNX
FUNCTION?
NO
YES
IMMEDIATE EXECUTE?
YES
INSERT EOL CODE
SET UP FUNCTION NAME AND COUNT
RRUN
LDEF
LEAVE DEFINITION MODE
SUMP
ENTER DEFINITION MODE FOR THIS KEY
REED6
RECALL DEFINITION
CONTK
CONTINUE EXECUTION AT BEGINNING OF THIS KEY DEFINITION

FIG 99B

USER DEFINABLE KEY SUBROUTINES

SET UP MEMORY POINTERS AND ENTER INTO DEFINITION MODE FOR A KEY

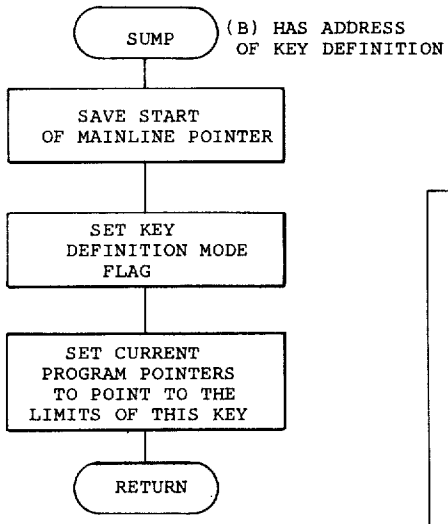


FIG 99C

FIND THE USER CODE FOR A PARTICULAR KEY

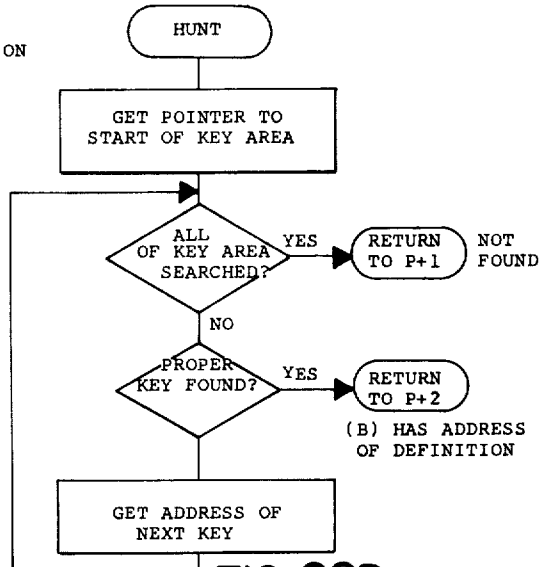


FIG 99D

SET UP A NEW KEY

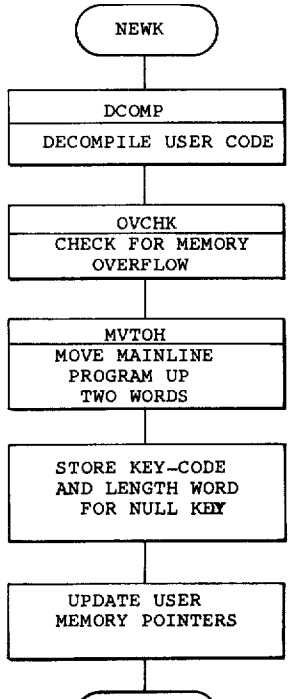


FIG 99E

LEAVE KEY DEFINITION MODE

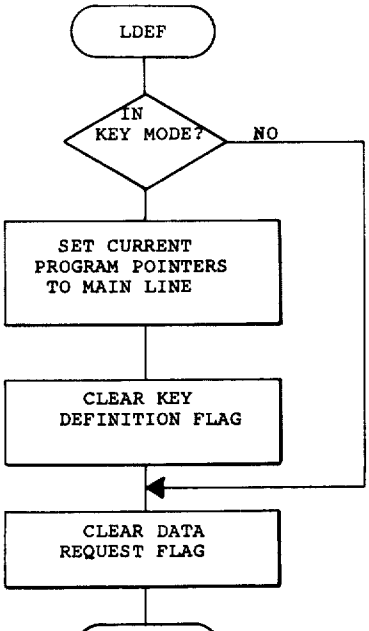


FIG 99F

EXECUTION MONITOR (1)

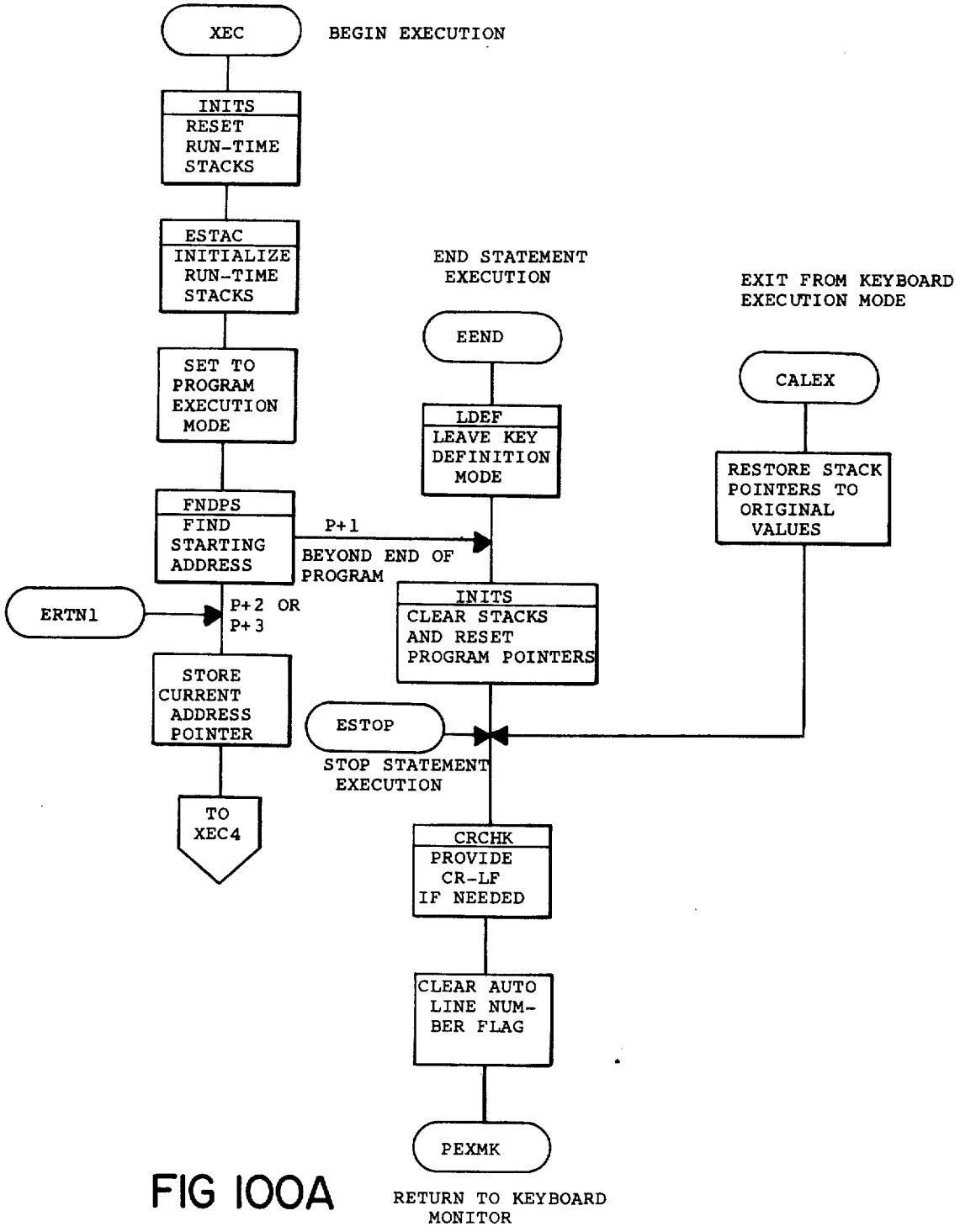


FIG 100A

EXECUTION MONITOR (2)

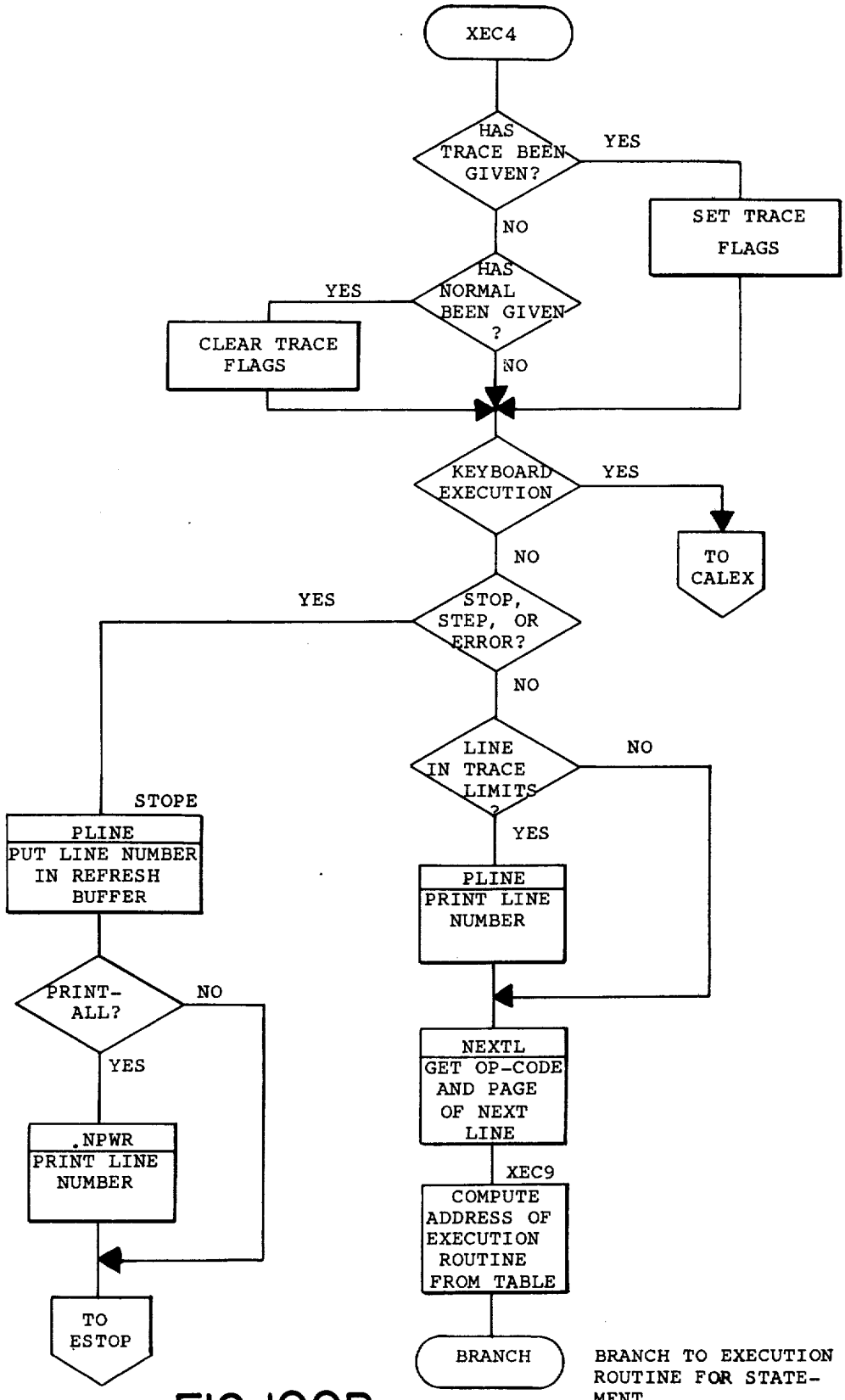


FIG 100B

INITIALIZATION FOR KEYBOARD EXECUTION

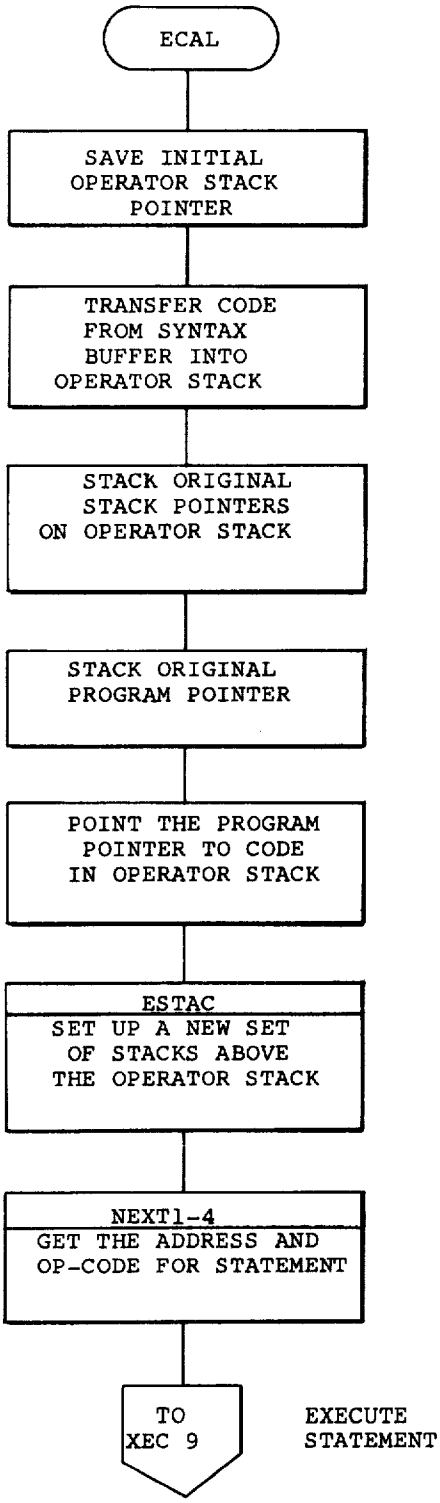


FIG 100C

SUBROUTINE TO SET UP RUN-TIME STACKS

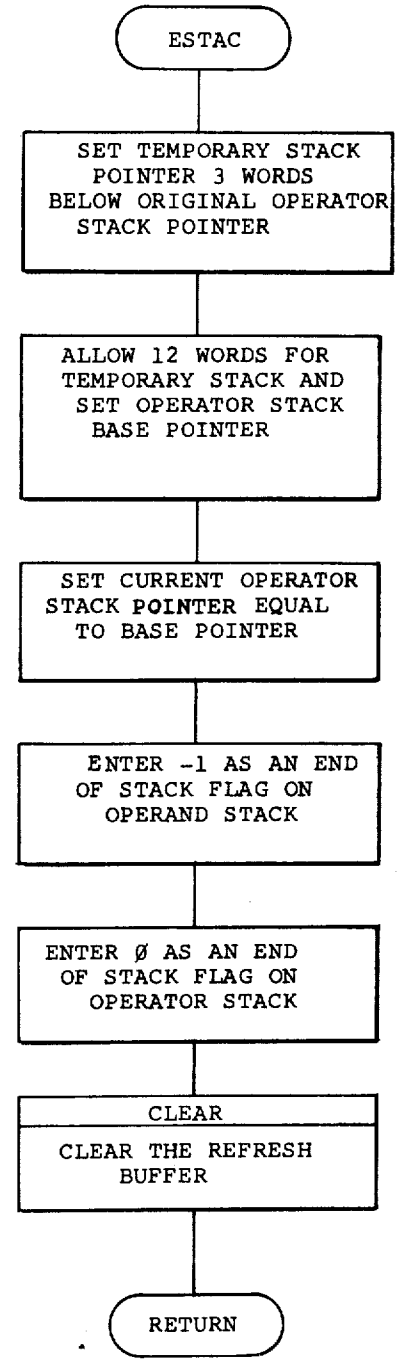


FIG 100D

LOAD A PROGRAM FROM THE CASSETTE

THE COMMAND IS: LOAD #UNIT, FILE NO., LNX1, LNX2

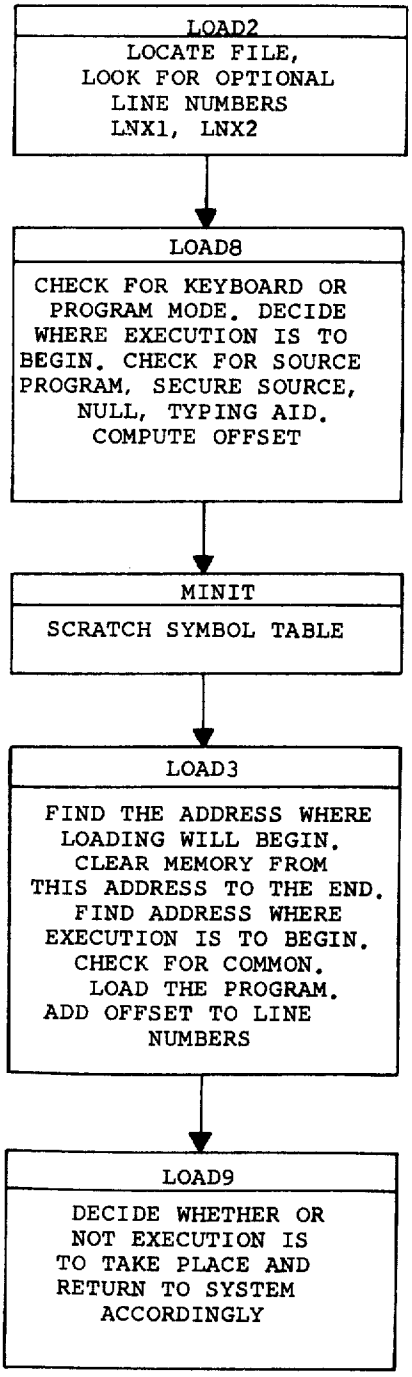


FIG. 101

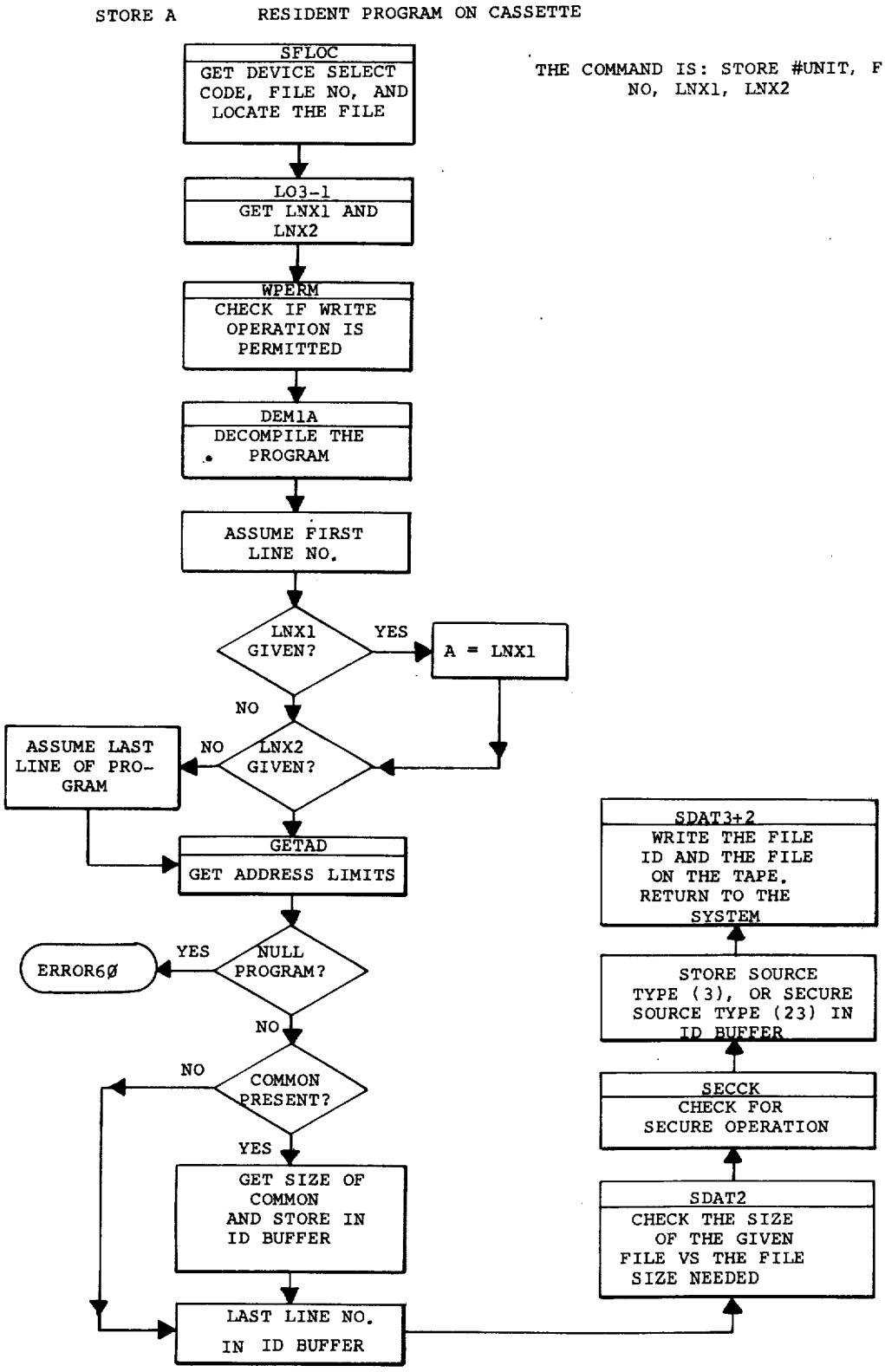

STORE A RESIDENT PROGRAM ON CASSETTE
SFLOC
GET DEVICE SELECT CODE, FILE NO, AND LOCATE THE FILE
THE COMMAND IS: STORE #UNIT, FILE NO, LNX1, LNX2
LO3-1
GET LNX1 AND LNX2
WPERM
CHECK IF WRITE OPERATION IS PERMITTED
DEM1A
DECOMPILE THE PROGRAM
ASSUME FIRST LINE NO.
LNX1 GIVEN?
YES
A = LNX1
NO
ASSUME LAST LINE OF PRO-GRAM
NO
LNX2 GIVEN?
GETAD
GET ADDRESS LIMITS
ERROR6Ø
YES
NULL PROGRAM?
NO
NO
COMMON PRESENT?
YES
GET SIZE OF COMMON AND STORE IN ID BUFFER
LAST LINE NO. IN ID BUFFER
SDAT2
CHECK THE SIZE OF THE GIVEN FILE VS THE FILE SIZE NEEDED
SECCK
CHECK FOR SECURE OPERATION
STORE SOURCE TYPE (3), OR SECURE SOURCE TYPE (23) IN ID BUFFER
SDAT3+2
WRITE THE FILE ID AND THE FILE ON THE TAPE. RETURN TO THE SYSTEM

FIG 102

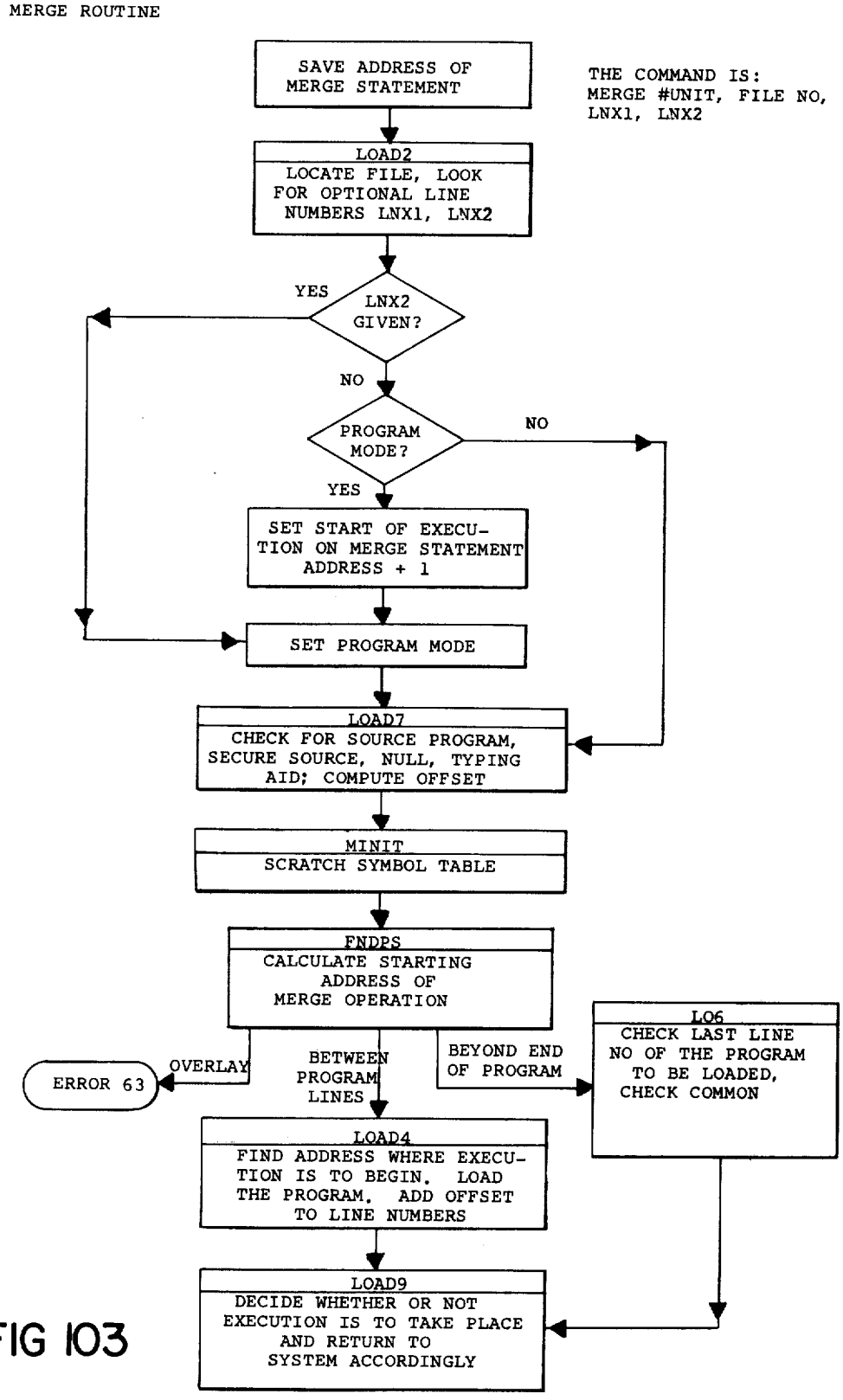

MERGE ROUTINE
SAVE ADDRESS OF MERGE STATEMENT
THE COMMAND IS: MERGE #UNIT, FILE NO, LNX1, LNX2
LOAD2
LOCATE FILE, LOOK FOR OPTIONAL LINE NUMBERS LNX1, LNX2
YES
LNX2 GIVEN?
NO
PROGRAM MODE?
NO
YES
SET START OF EXECUTION ON MERGE STATEMENT ADDRESS + 1
SET PROGRAM MODE
LOAD7
CHECK FOR SOURCE PROGRAM, SECURE SOURCE, NULL, TYPING AID; COMPUTE OFFSET
MINIT
SCRATCH SYMBOL TABLE
FNDPS
CALCULATE STARTING ADDRESS OF MERGE OPERATION
OVERLAY
ERROR 63
BETWEEN PROGRAM LINES
BEYOND END OF PROGRAM
LOAD4
FIND ADDRESS WHERE EXECUTION IS TO BEGIN. LOAD THE PROGRAM. ADD OFFSET TO LINE NUMBERS
LOAD9
DECIDE WHETHER OR NOT EXECUTION IS TO TAKE PLACE AND RETURN TO SYSTEM ACCORDINGLY
LOAD6
CHECK LAST LINE NO OF THE PROGRAM TO BE LOADED, CHECK COMMON

FIG 103

LINK ROUTINE

THE COMMAND IS: LINK #UNIT, FILE NO, LNX1, LNX2

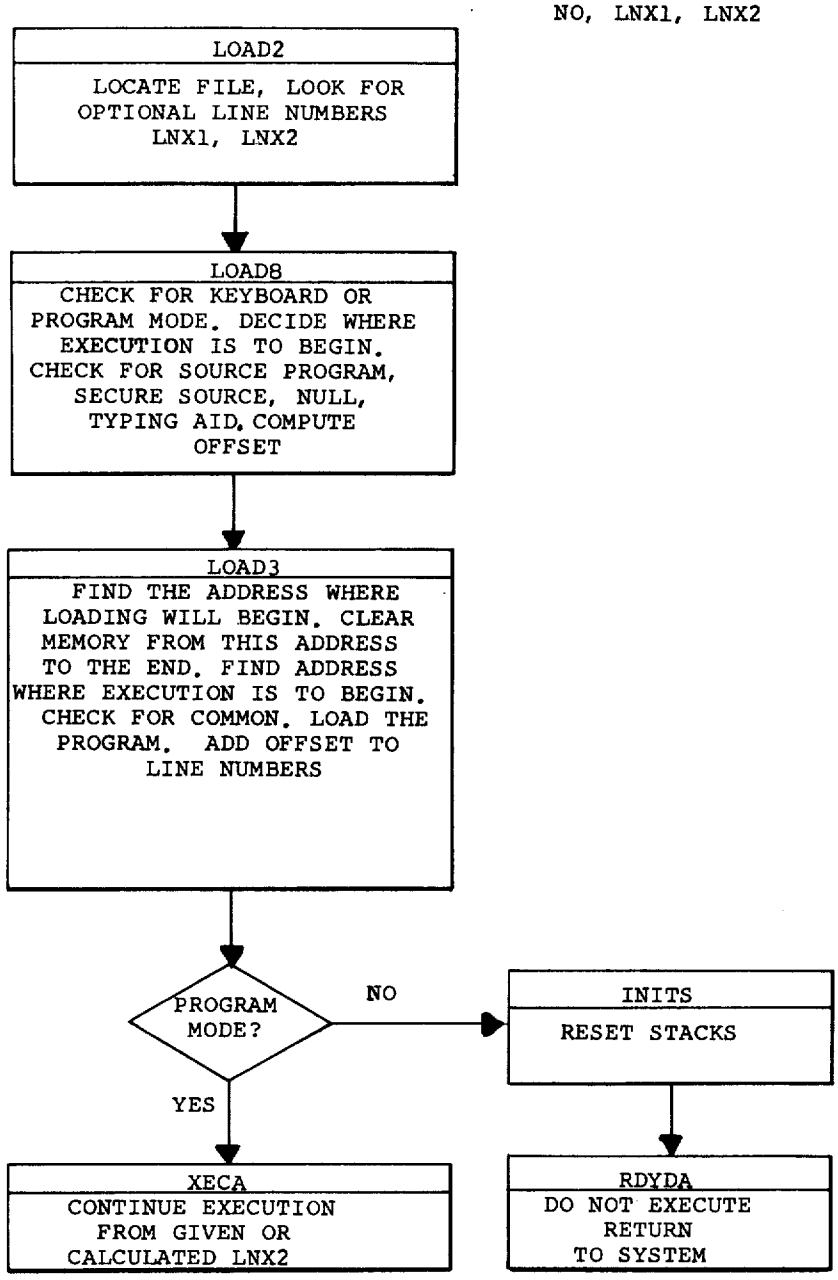


FIG 104

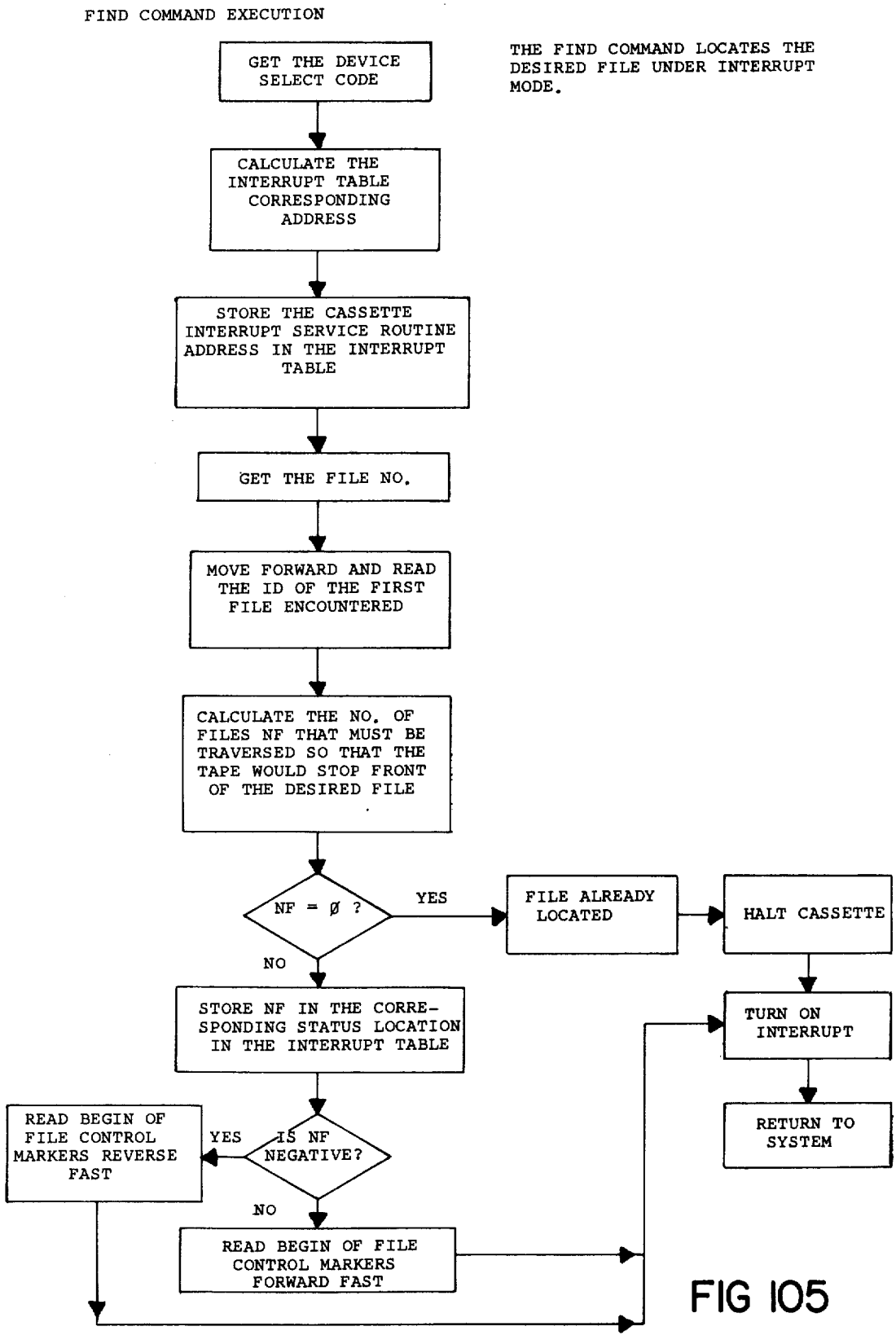

FIND COMMAND EXECUTION
THE FIND COMMAND LOCATES THE DESIRED FILE UNDER INTERRUPT MODE.
GET THE DEVICE SELECT CODE
CALCULATE THE INTERRUPT TABLE CORRESPONDING ADDRESS
STORE THE CASSETTE INTERRUPT SERVICE ROUTINE ADDRESS IN THE INTERRUPT TABLE
GET THE FILE NO.
MOVE FORWARD AND READ THE ID OF THE FIRST FILE ENCOUNTERED
CALCULATE THE NO. OF FILES NF THAT MUST BE TRAVERSED SO THAT THE TAPE WOULD STOP FRONT OF THE DESIRED FILE
NF = Ø ?
YES
FILE ALREADY LOCATED
HALT CASSETTE
NO
STORE NF IN THE CORRE-SPONDING STATUS LOCATION IN THE INTERRUPT TABLE
TURN ON INTERRUPT
RETURN TO SYSTEM
READ BEGIN OF FILE CONTROL MARKERS REVERSE FAST
YES
IS NF NEGATIVE?
NO
READ BEGIN OF FILE CONTROL MARKERS FORWARD FAST

FIG 105

LOAD2 SUBROUTINE

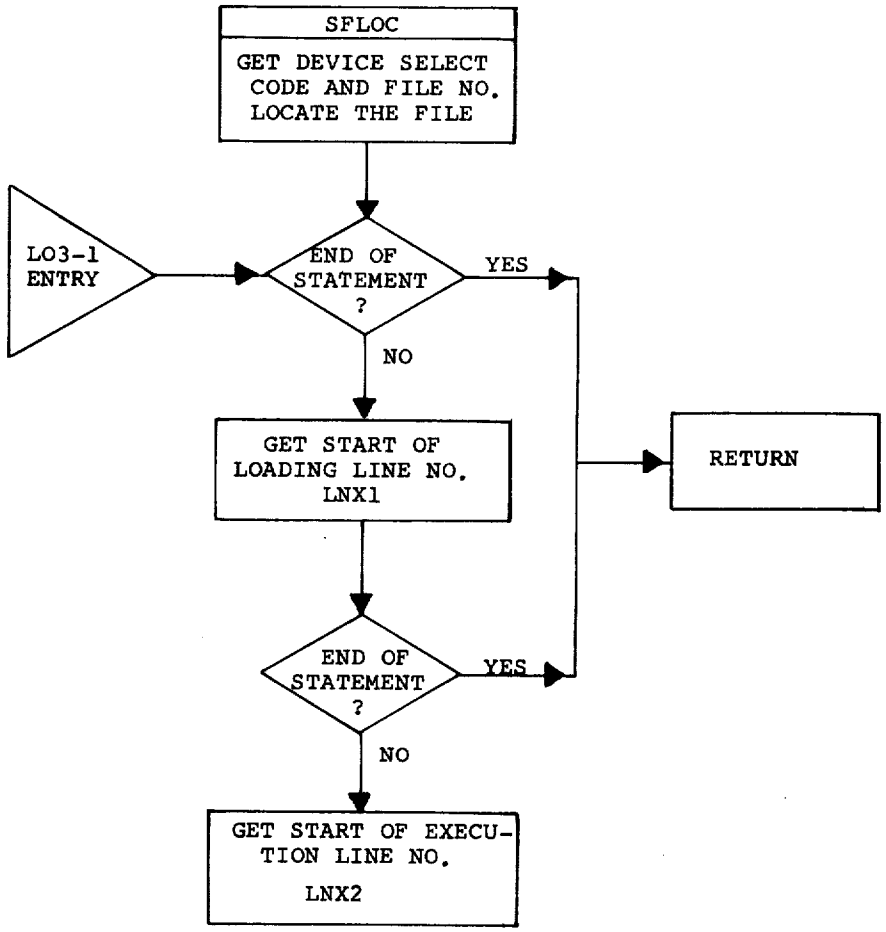


FIG 106A

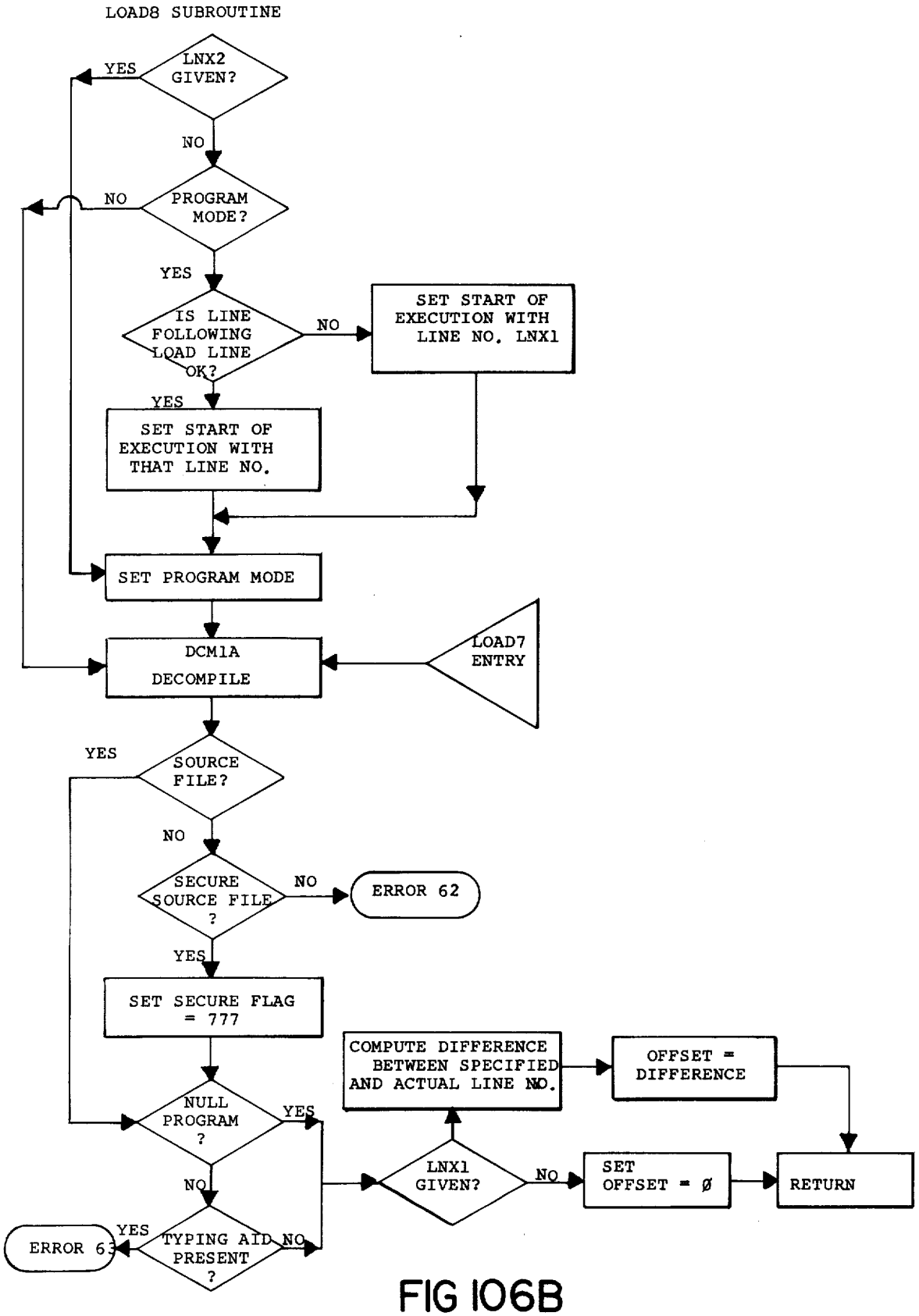

LOAD8 SUBROUTINE
LNX2 GIVEN?
YES
NO
PROGRAM MODE?
NO
YES
IS LINE FOLLOWING LOAD LINE OK?
NO
SET START OF EXECUTION WITH LINE NO. LNX1
YES
SET START OF EXECUTION WITH THAT LINE NO.
SET PROGRAM MODE
DCM1A DECOMPILE
LOAD7 ENTRY
SOURCE FILE?
YES
NO
SECURE SOURCE FILE ?
NO
ERROR 62
YES
SET SECURE FLAG = 777
NULL PROGRAM ?
YES
NO
TYPING AID PRESENT ?
YES
ERROR 63
NO
LNX1 GIVEN?
NO
SET OFFSET = ø
RETURN
COMPUTE DIFFERENCE BETWEEN SPECIFIED AND ACTUAL LINE NO.
OFFSET = DIFFERENCE

FIG 106B

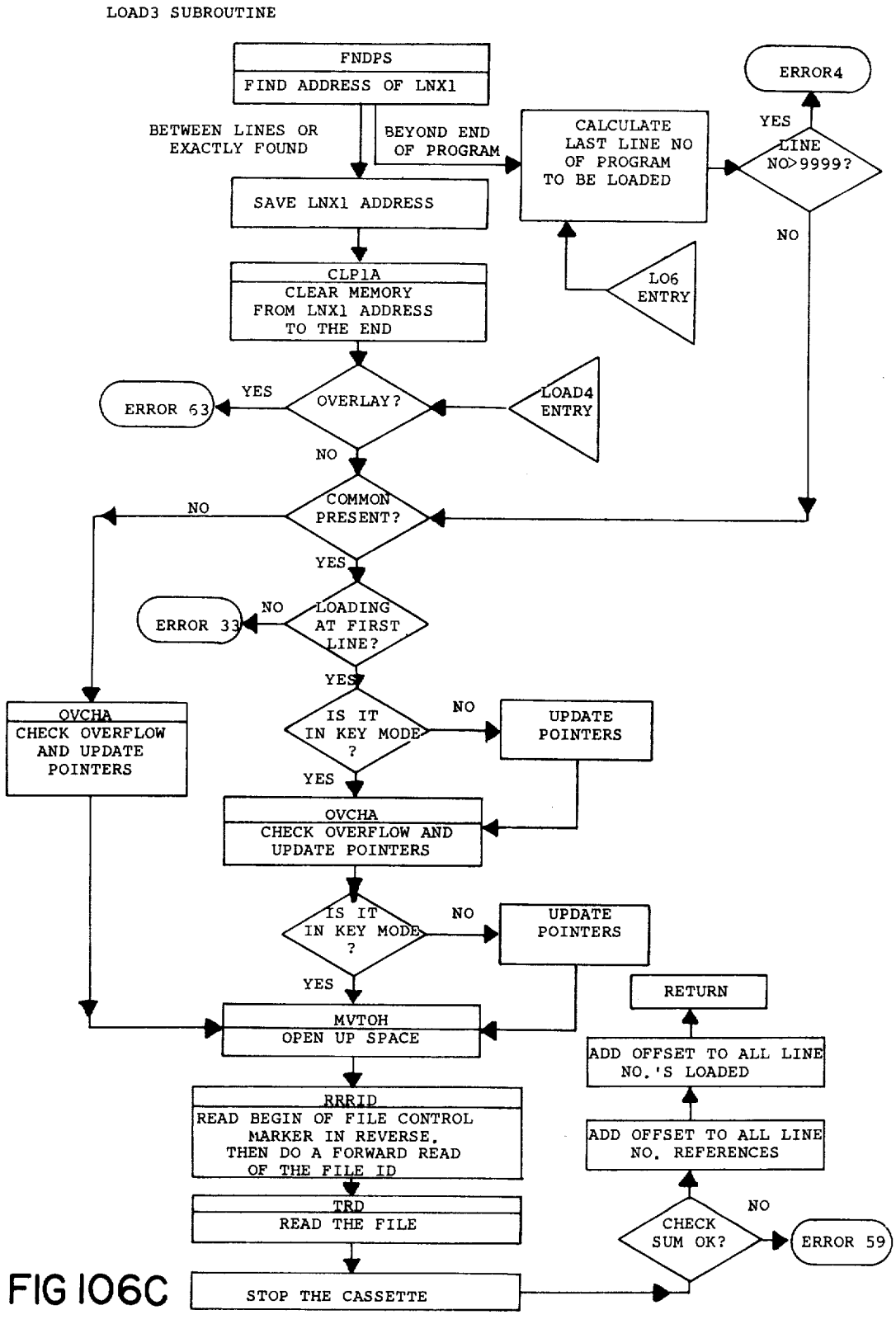

LOAD3 SUBROUTINE
FNDPS
FIND ADDRESS OF LNX1
BETWEEN LINES OR EXACTLY FOUND
BEYOND END OF PROGRAM
SAVE LNX1 ADDRESS
CLP1A
CLEAR MEMORY FROM LNX1 ADDRESS TO THE END
CALCULATE LAST LINE NO OF PROGRAM TO BE LOADED
LO6 ENTRY
ERROR4
YES
LINE NO>9999?
NO
ERROR 63
YES
OVERLAY?
LOAD4 ENTRY
NO
COMMON PRESENT?
NO
YES
ERROR 33
NO
LOADING AT FIRST LINE?
YES
OVCHA
CHECK OVERFLOW AND UPDATE POINTERS
IS IT IN KEY MODE ?
NO
UPDATE POINTERS
YES
OVCHA
CHECK OVERFLOW AND UPDATE POINTERS
IS IT IN KEY MODE ?
NO
UPDATE POINTERS
YES
MVTOH
OPEN UP SPACE
RRRID
READ BEGIN OF FILE CONTROL MARKER IN REVERSE, THEN DO A FORWARD READ OF THE FILE ID
TRD
READ THE FILE
STOP THE CASSETTE
RETURN
ADD OFFSET TO ALL LINE NO.'S LOADED
ADD OFFSET TO ALL LINE NO. REFERENCES
CHECK SUM OK?
NO
ERROR 59

FIG 106C

LOAD9 SUBROUTINE

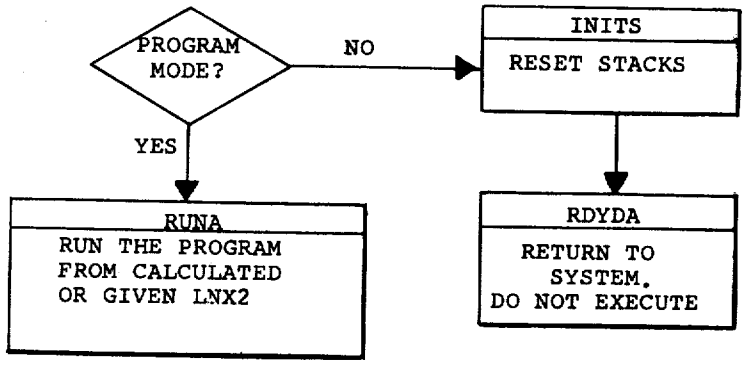


FIG 106D

LOAD DATA ROUTINE

THE COMMAND IS: LOAD DATA #UNIT, FILE NO, ARRAY NAME

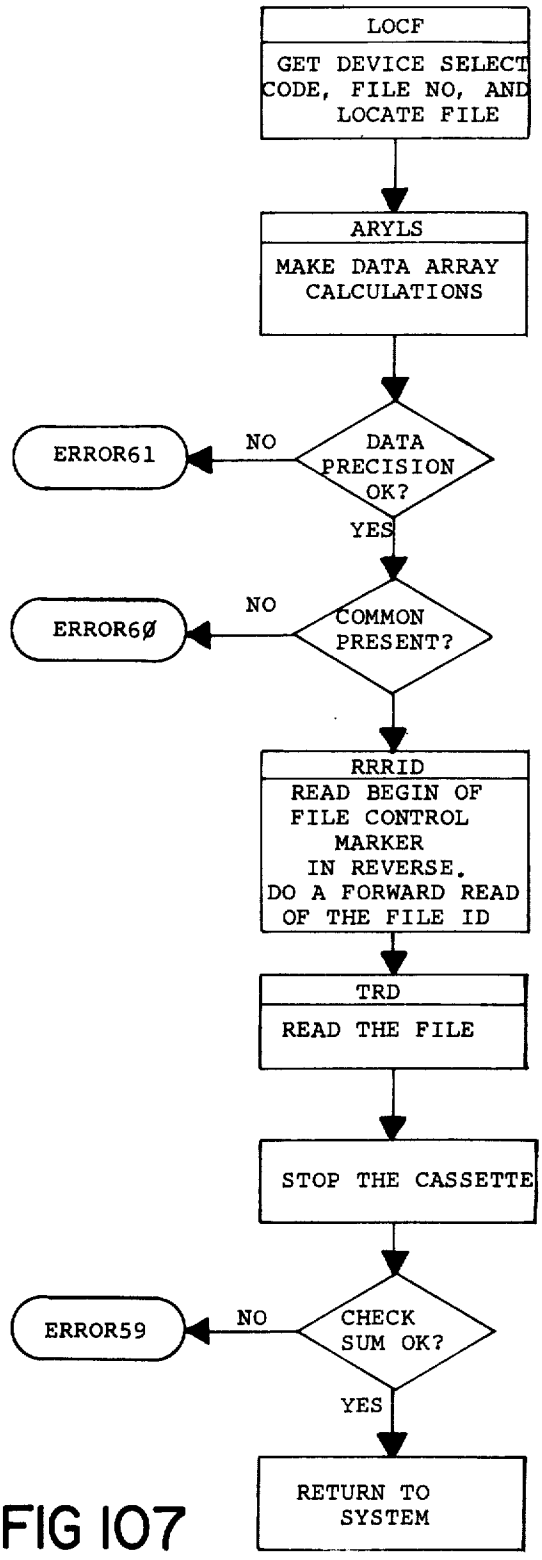


FIG 107

STORE DATA ROUTINE

THE COMMAND IS: STORE DATA #UNIT, FILE NO., ARRAY NAME

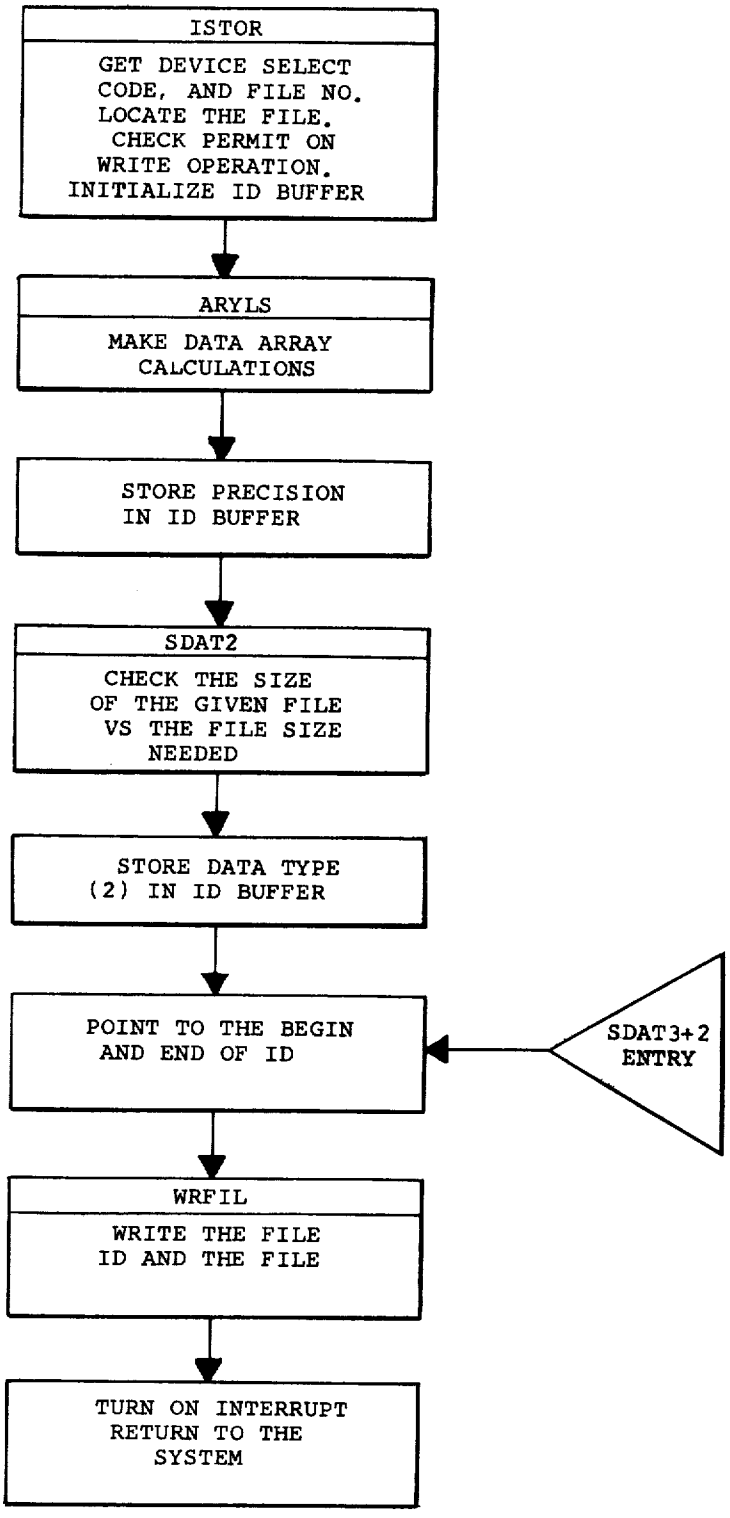


FIG 108

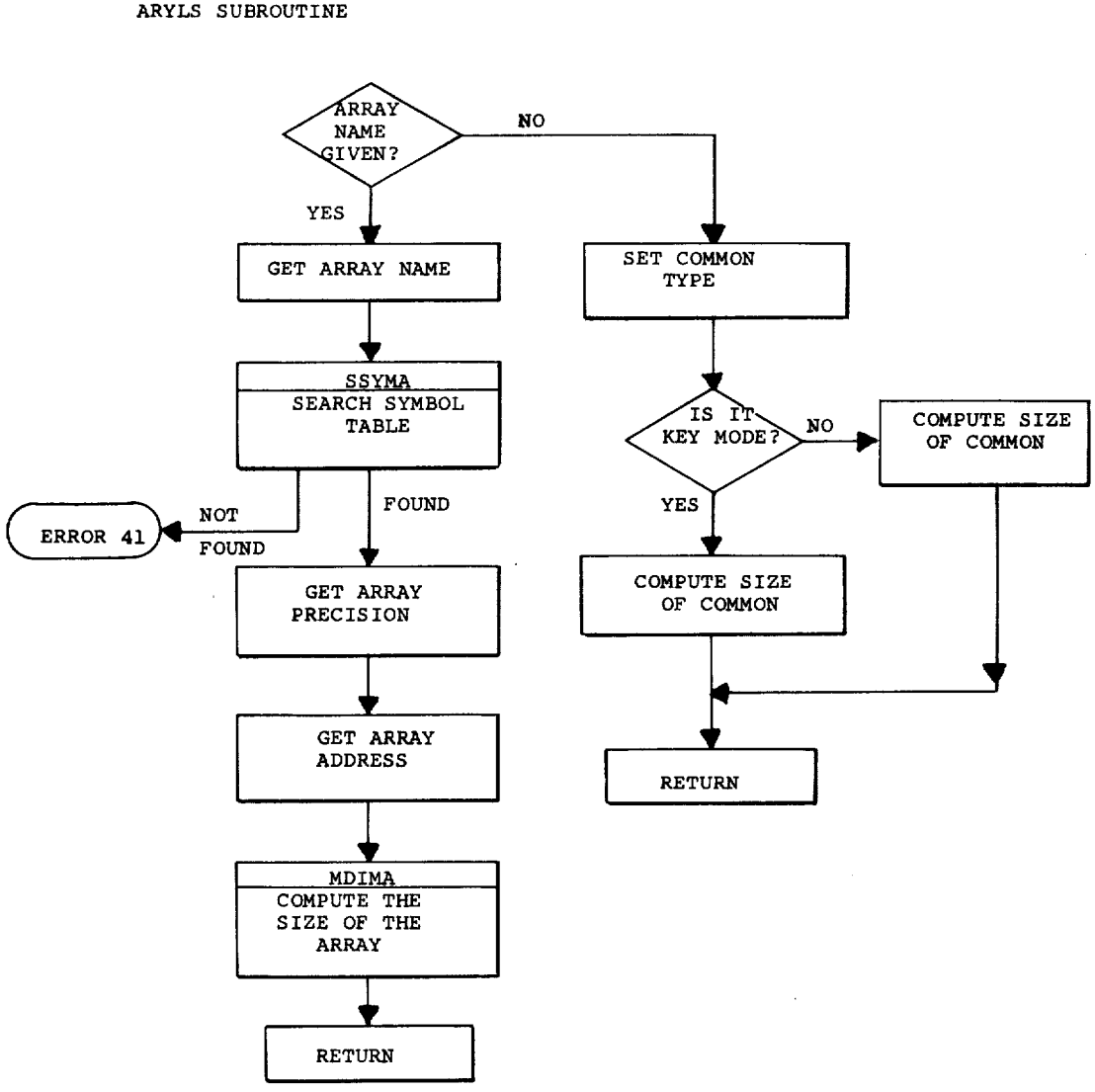

ARYLS SUBROUTINE
ARRAY NAME GIVEN?
NO
YES
GET ARRAY NAME
SSYMA
SEARCH SYMBOL TABLE
NOT FOUND
ERROR 41
FOUND
GET ARRAY PRECISION
GET ARRAY ADDRESS
MDIMA
COMPUTE THE SIZE OF THE ARRAY
RETURN
SET COMMON TYPE
IS IT KEY MODE?
NO
COMPUTE SIZE OF COMMON
YES
COMPUTE SIZE OF COMMON
RETURN

FIG 109

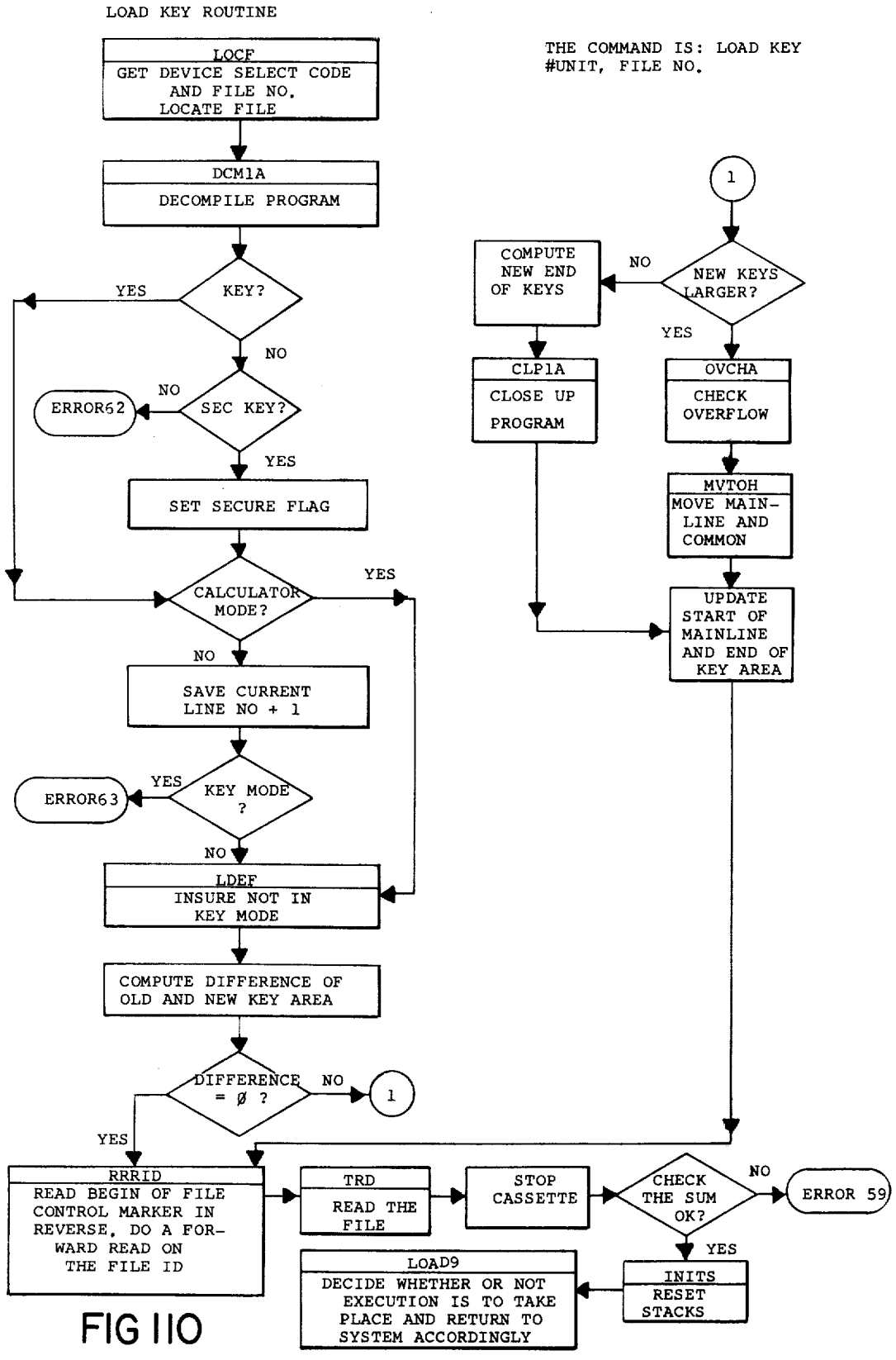

LOAD KEY ROUTINE
LOCF
GET DEVICE SELECT CODE AND FILE NO. LOCATE FILE
DCM1A
DECOMPILE PROGRAM
KEY?
YES
NO
SEC KEY?
NO
ERROR62
YES
SET SECURE FLAG
CALCULATOR MODE?
YES
NO
SAVE CURRENT LINE NO + 1
KEY MODE ?
YES
ERROR63
NO
LDEF
INSURE NOT IN KEY MODE
COMPUTE DIFFERENCE OF OLD AND NEW KEY AREA
DIFFERENCE = Ø ?
NO
1
YES
RRRID
READ BEGIN OF FILE CONTROL MARKER IN REVERSE, DO A FORWARD READ ON THE FILE ID
TRD
READ THE FILE
STOP CASSETTE
CHECK THE SUM OK?
NO
ERROR 59
YES
INITS
RESET STACKS
LOAD9
DECIDE WHETHER OR NOT EXECUTION IS TO TAKE PLACE AND RETURN TO SYSTEM ACCORDINGLY
THE COMMAND IS: LOAD KEY #UNIT, FILE NO.
1
NEW KEYS LARGER?
NO
COMPUTE NEW END OF KEYS
YES
CLP1A
CLOSE UP PROGRAM
OVCHA
CHECK OVERFLOW
MVTOH
MOVE MAINLINE AND COMMON
UPDATE START OF MAINLINE AND END OF KEY AREA

FIG 110

STORE KEY ROUTINE

THE COMMAND IS: STORE KEY #UNIT, FILE NO.

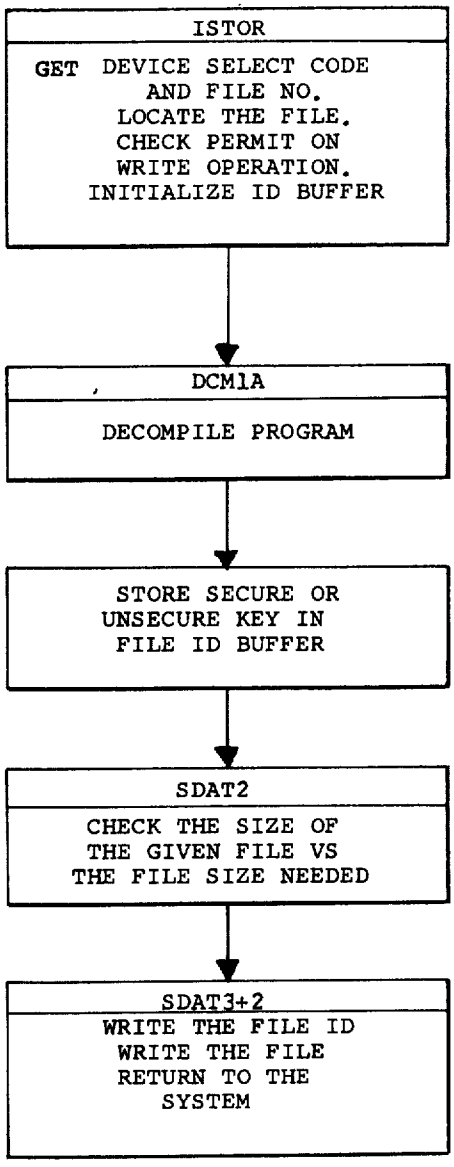


FIG III

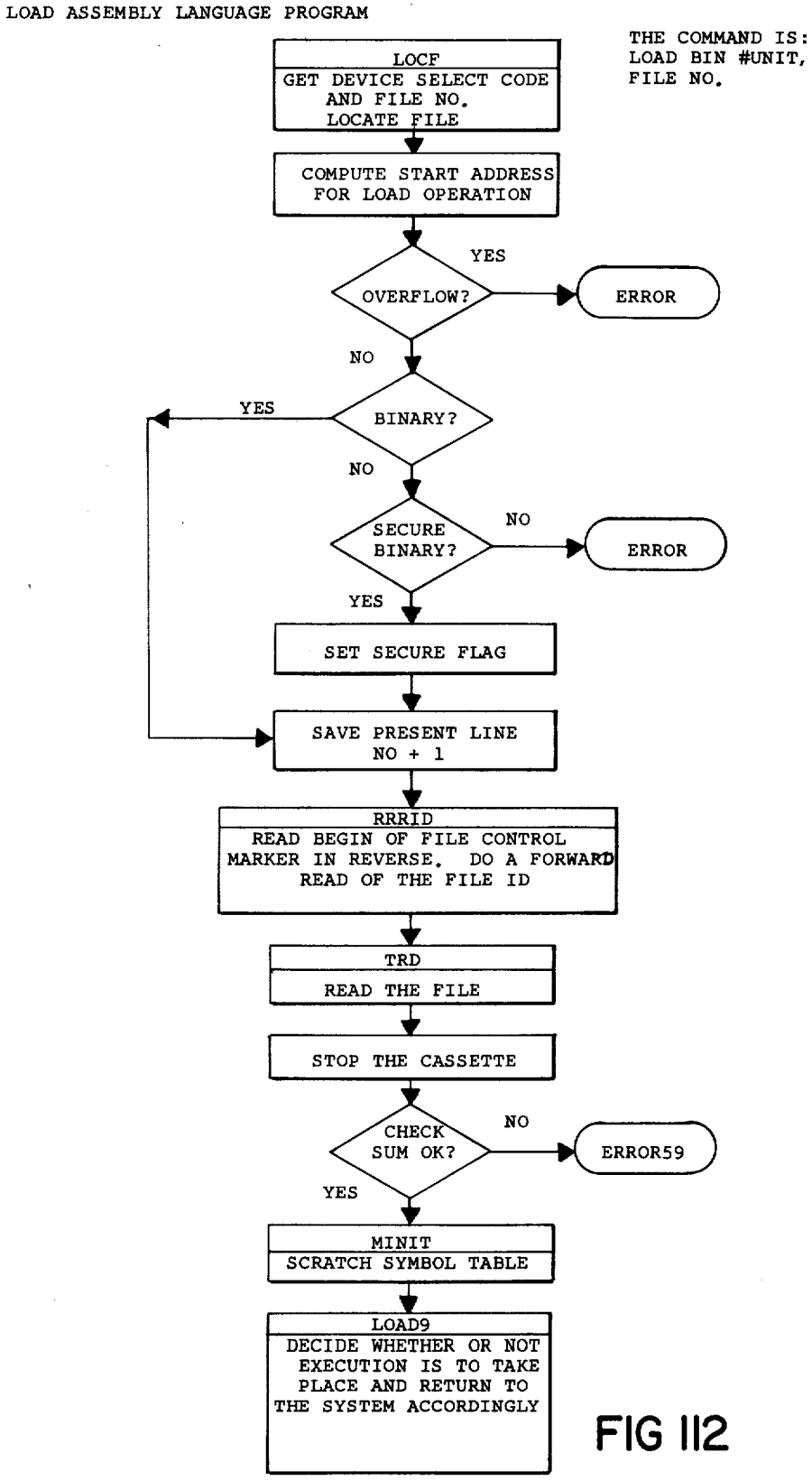

LOAD ASSEMBLY LANGUAGE PROGRAM
THE COMMAND IS:
LOAD BIN #UNIT,
FILE NO.
LOCF
GET DEVICE SELECT CODE AND FILE NO. LOCATE FILE
COMPUTE START ADDRESS FOR LOAD OPERATION
YES
OVERFLOW?
ERROR
NO
YES
BINARY?
NO
NO
SECURE BINARY?
ERROR
YES
SET SECURE FLAG
SAVE PRESENT LINE NO + 1
RRRID
READ BEGIN OF FILE CONTROL MARKER IN REVERSE. DO A FORWARD READ OF THE FILE ID
TRD
READ THE FILE
STOP THE CASSETTE
NO
CHECK SUM OK?
ERROR59
YES
MINIT
SCRATCH SYMBOL TABLE
LOAD9
DECIDE WHETHER OR NOT EXECUTION IS TO TAKE PLACE AND RETURN TO THE SYSTEM ACCORDINGLY

FIG 112

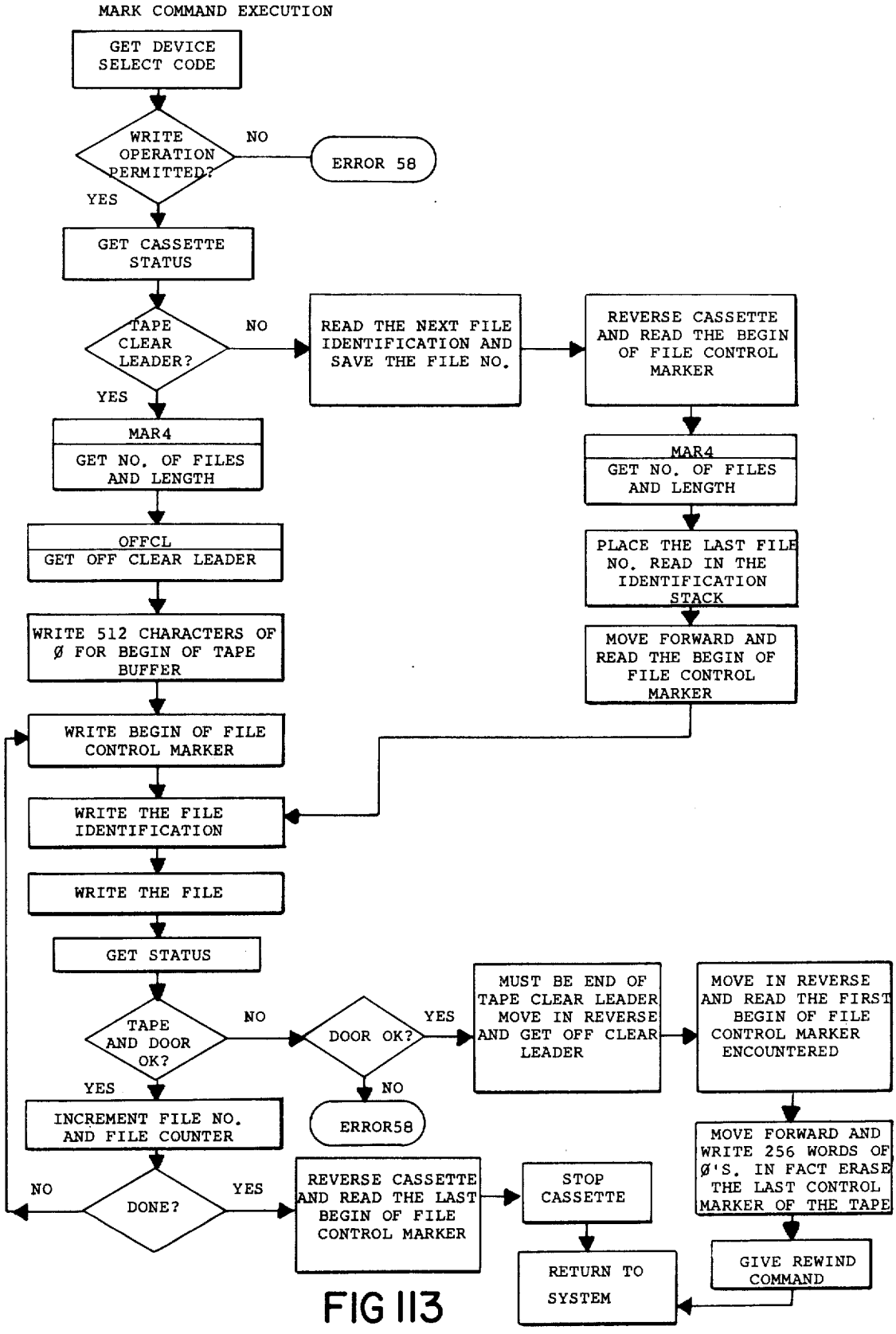

MARK COMMAND EXECUTION
GET DEVICE SELECT CODE
WRITE OPERATION PERMITTED?
NO
ERROR 58
YES
GET CASSETTE STATUS
TAPE CLEAR LEADER?
NO
READ THE NEXT FILE IDENTIFICATION AND SAVE THE FILE NO.
REVERSE CASSETTE AND READ THE BEGIN OF FILE CONTROL MARKER
YES
MAR4
GET NO. OF FILES AND LENGTH
MAR4
GET NO. OF FILES AND LENGTH
OFFCL
GET OFF CLEAR LEADER
PLACE THE LAST FILE NO. READ IN THE IDENTIFICATION STACK
WRITE 512 CHARACTERS OF Ø FOR BEGIN OF TAPE BUFFER
MOVE FORWARD AND READ THE BEGIN OF FILE CONTROL MARKER
WRITE BEGIN OF FILE CONTROL MARKER
WRITE THE FILE IDENTIFICATION
WRITE THE FILE
GET STATUS
TAPE AND DOOR OK?
NO
DOOR OK?
YES
MUST BE END OF TAPE CLEAR LEADER MOVE IN REVERSE AND GET OFF CLEAR LEADER
MOVE IN REVERSE AND READ THE FIRST BEGIN OF FILE CONTROL MARKER ENCOUNTERED
YES
NO
INCREMENT FILE NO. AND FILE COUNTER
ERROR58
MOVE FORWARD AND WRITE 256 WORDS OF Ø'S. IN FACT ERASE THE LAST CONTROL MARKER OF THE TAPE
NO
DONE?
YES
REVERSE CASSETTE AND READ THE LAST BEGIN OF FILE CONTROL MARKER
STOP CASSETTE
GIVE REWIND COMMAND
RETURN TO SYSTEM

FIG 113

MAR4 - SUBROUTINE USED BY THE MARK COMMAND

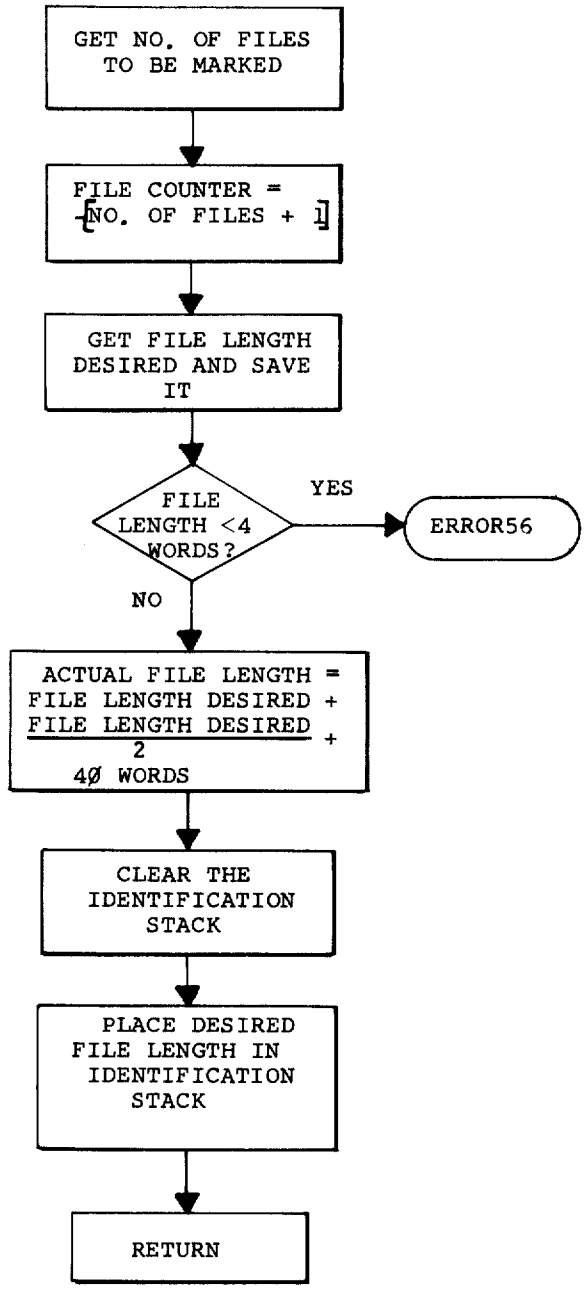
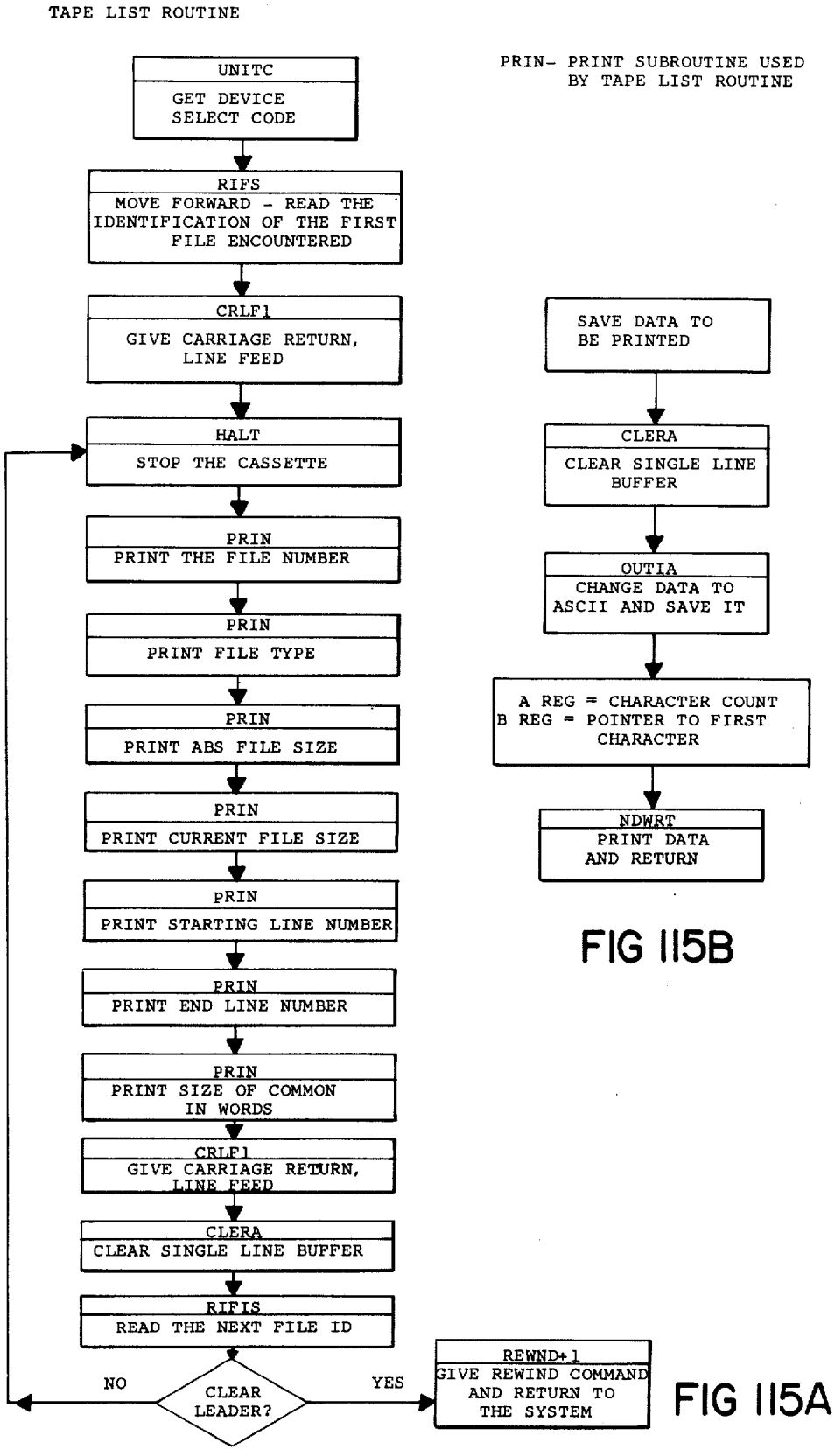


FIG 114


TAPE LIST ROUTINE
UNITC
GET DEVICE SELECT CODE
RIFS
MOVE FORWARD - READ THE IDENTIFICATION OF THE FIRST FILE ENCOUNTERED
CRLF1
GIVE CARRIAGE RETURN, LINE FEED
HALT
STOP THE CASSETTE
PRIN
PRINT THE FILE NUMBER
PRIN
PRINT FILE TYPE
PRIN
PRINT ABS FILE SIZE
PRIN
PRINT CURRENT FILE SIZE
PRIN
PRINT STARTING LINE NUMBER
PRIN
PRINT END LINE NUMBER
PRIN
PRINT SIZE OF COMMON IN WORDS
CRLF1
GIVE CARRIAGE RETURN, LINE FEED
CLERA
CLEAR SINGLE LINE BUFFER
RIFIS
READ THE NEXT FILE ID
CLEAR LEADER?
NO
YES
REWND+1
GIVE REWIND COMMAND AND RETURN TO THE SYSTEM
FIG 115A
PRIN- PRINT SUBROUTINE USED BY TAPE LIST ROUTINE
SAVE DATA TO BE PRINTED
CLERA
CLEAR SINGLE LINE BUFFER
OUTIA
CHANGE DATA TO ASCII AND SAVE IT
A REG = CHARACTER COUNT
B REG = POINTER TO FIRST CHARACTER
NDWRT
PRINT DATA AND RETURN
FIG 115B

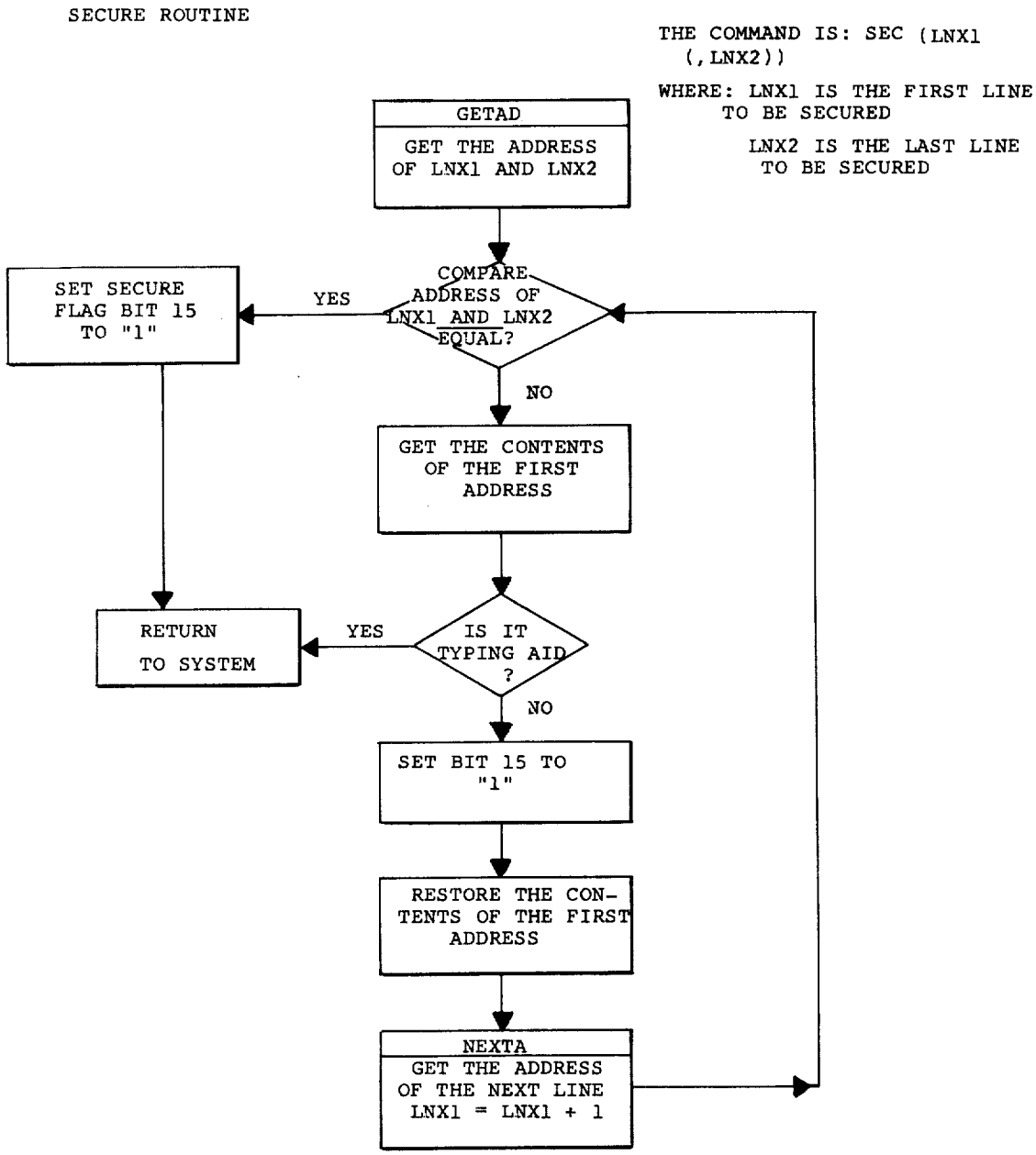

SECURE ROUTINE
THE COMMAND IS: SEC (LNX1 (,LNX2))
WHERE: LNX1 IS THE FIRST LINE TO BE SECURED
LNX2 IS THE LAST LINE TO BE SECURED
GETAD
GET THE ADDRESS OF LNX1 AND LNX2
COMPARE ADDRESS OF LNX1 AND LNX2 EQUAL?
YES
SET SECURE FLAG BIT 15 TO "1"
NO
GET THE CONTENTS OF THE FIRST ADDRESS
IS IT TYPING AID ?
YES
RETURN TO SYSTEM
NO
SET BIT 15 TO "1"
RESTORE THE CONTENTS OF THE FIRST ADDRESS
NEXTA
GET THE ADDRESS OF THE NEXT LINE LNX1 = LNX1 + 1

FIG 116

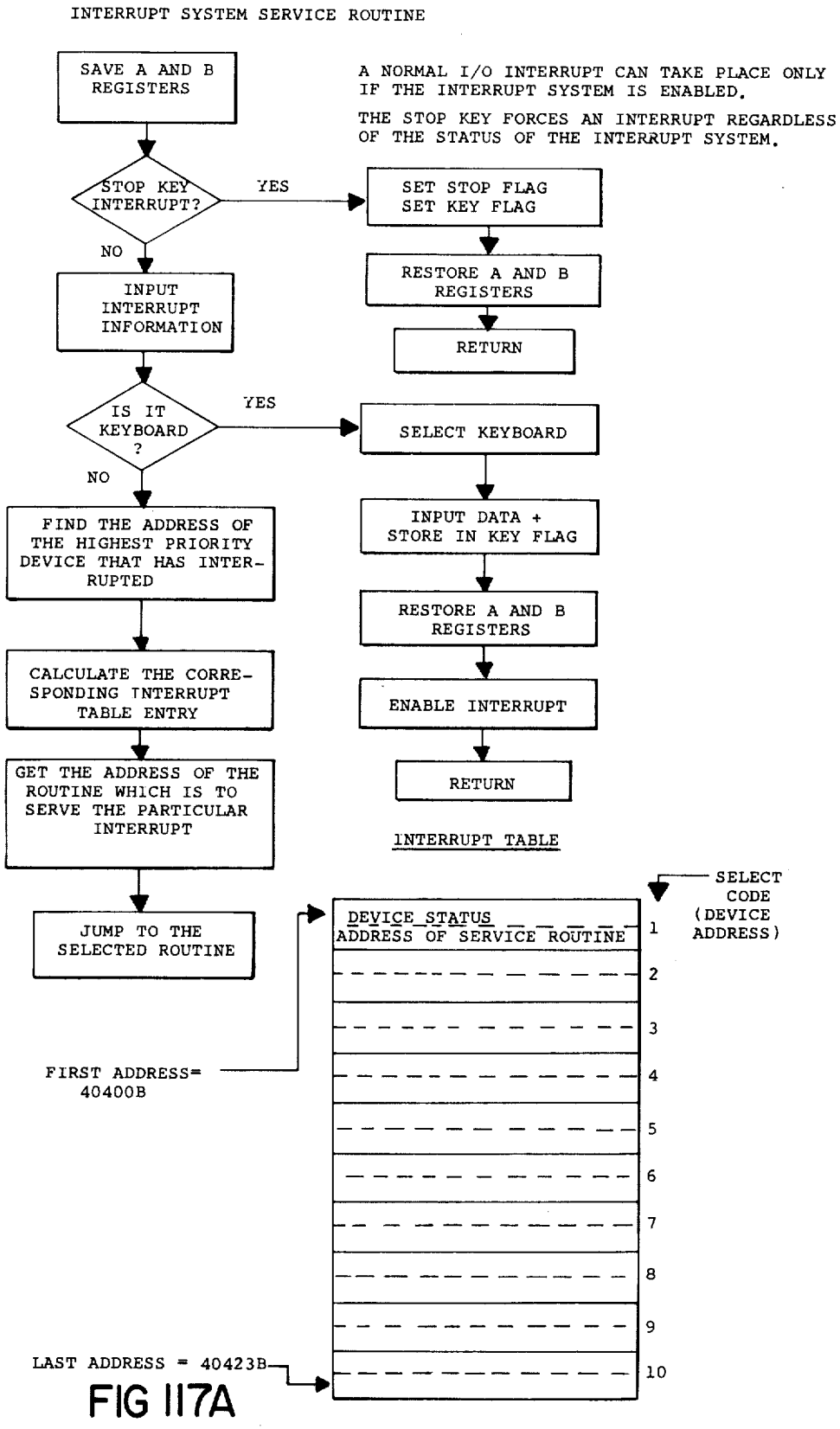

INTERRUPT SYSTEM SERVICE ROUTINE
SAVE A AND B REGISTERS
A NORMAL I/O INTERRUPT CAN TAKE PLACE ONLY IF THE INTERRUPT SYSTEM IS ENABLED.
THE STOP KEY FORCES AN INTERRUPT REGARDLESS OF THE STATUS OF THE INTERRUPT SYSTEM.
STOP KEY INTERRUPT?
YES
SET STOP FLAG
SET KEY FLAG
NO
RESTORE A AND B REGISTERS
INPUT INTERRUPT INFORMATION
RETURN
IS IT KEYBOARD ?
YES
SELECT KEYBOARD
NO
INPUT DATA + STORE IN KEY FLAG
FIND THE ADDRESS OF THE HIGHEST PRIORITY DEVICE THAT HAS INTERRUPTED
RESTORE A AND B REGISTERS
CALCULATE THE CORRESPONDING INTERRUPT TABLE ENTRY
ENABLE INTERRUPT
GET THE ADDRESS OF THE ROUTINE WHICH IS TO SERVE THE PARTICULAR INTERRUPT
RETURN
INTERRUPT TABLE
SELECT CODE (DEVICE ADDRESS)
JUMP TO THE SELECTED ROUTINE
DEVICE STATUS
ADDRESS OF SERVICE ROUTINE
1
2
3
4
5
6
7
8
9
10
FIRST ADDRESS= 40400B
LAST ADDRESS = 40423B

FIG 117A

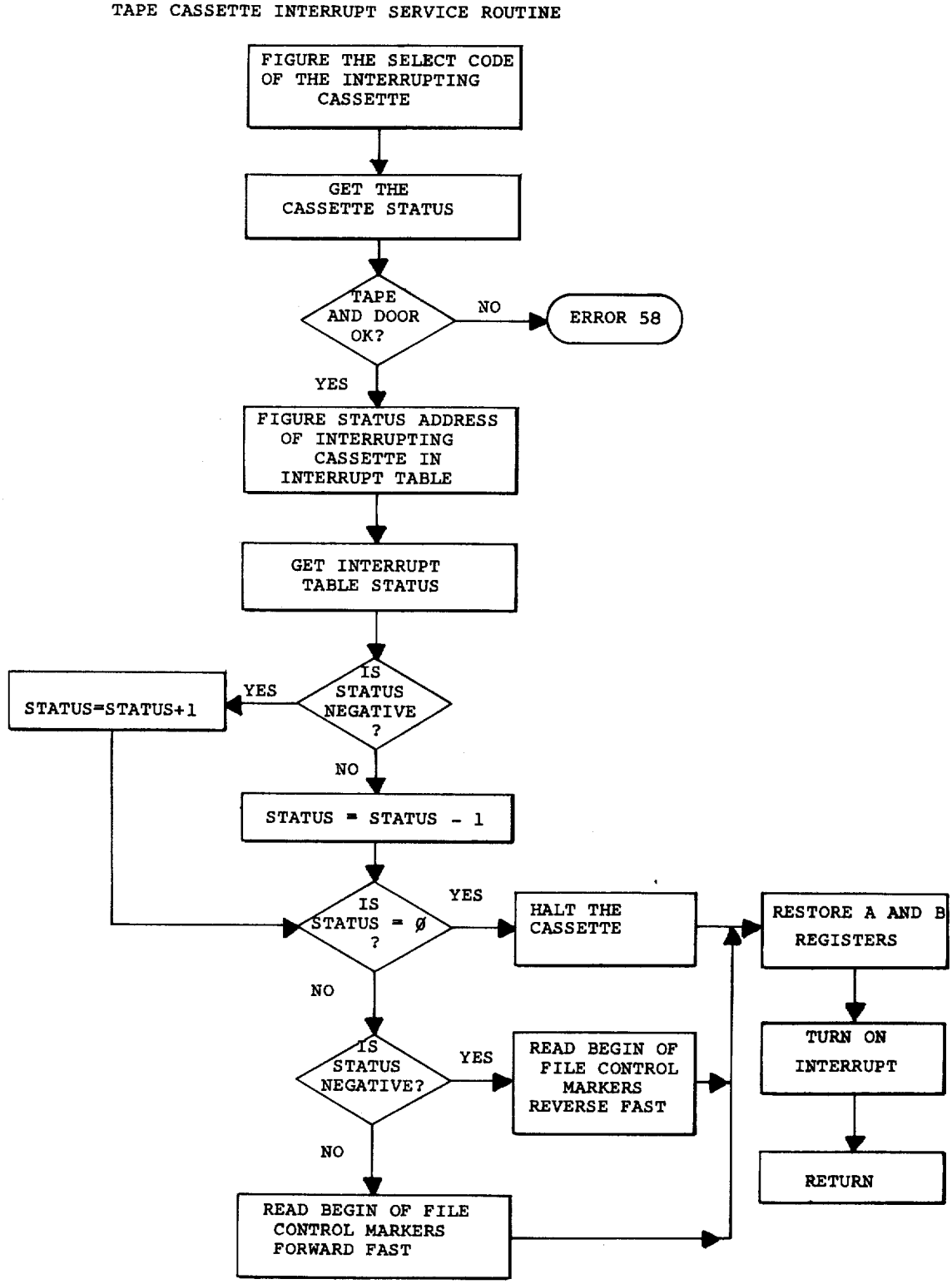

TAPE CASSETTE INTERRUPT SERVICE ROUTINE
FIGURE THE SELECT CODE OF THE INTERRUPTING CASSETTE
GET THE CASSETTE STATUS
TAPE AND DOOR OK?
NO
ERROR 58
YES
FIGURE STATUS ADDRESS OF INTERRUPTING CASSETTE IN INTERRUPT TABLE
GET INTERRUPT TABLE STATUS
IS STATUS NEGATIVE ?
YES
STATUS=STATUS+1
NO
STATUS = STATUS - 1
IS STATUS = ø ?
YES
HALT THE CASSETTE
RESTORE A AND B REGISTERS
NO
IS STATUS NEGATIVE?
YES
READ BEGIN OF FILE CONTROL MARKERS REVERSE FAST
TURN ON INTERRUPT
NO
RETURN
READ BEGIN OF FILE CONTROL MARKERS FORWARD FAST

FIG 117B

FLOWCHART OF OFFSET COMMAND FOR PLOTTER BLOCK

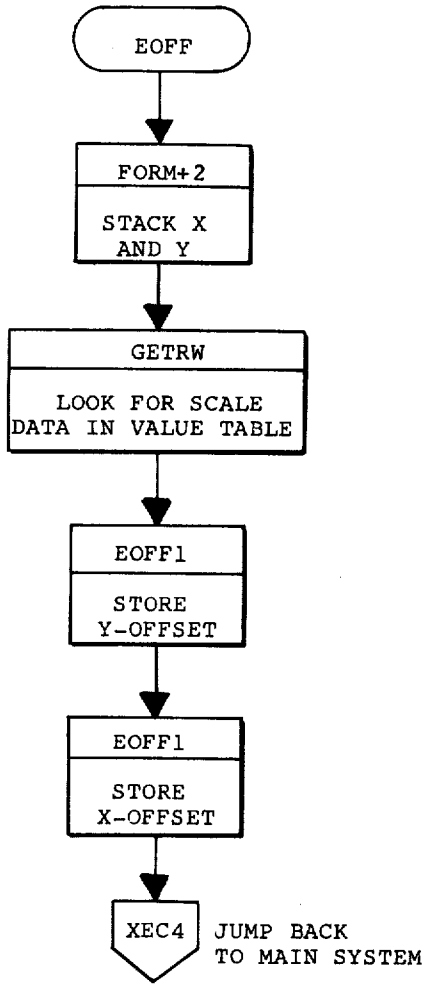


FIG 118A

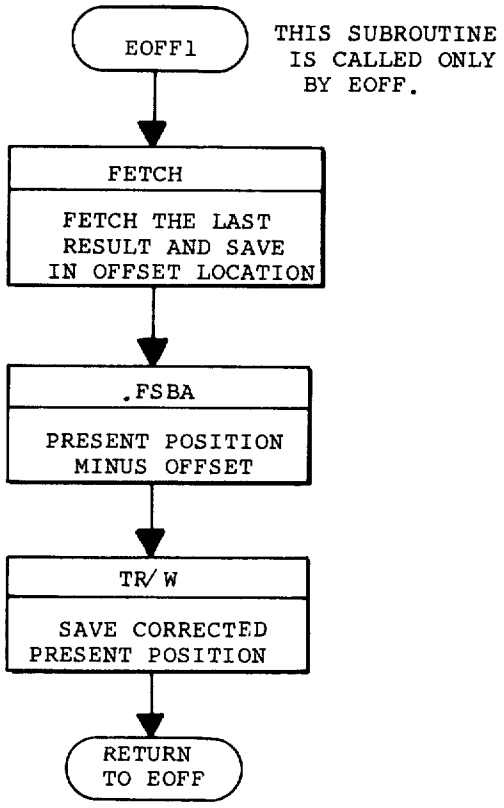


FIG 118B

FLOWCHART OF I PLOT COMMAND FOR PLOTTER BLOCK

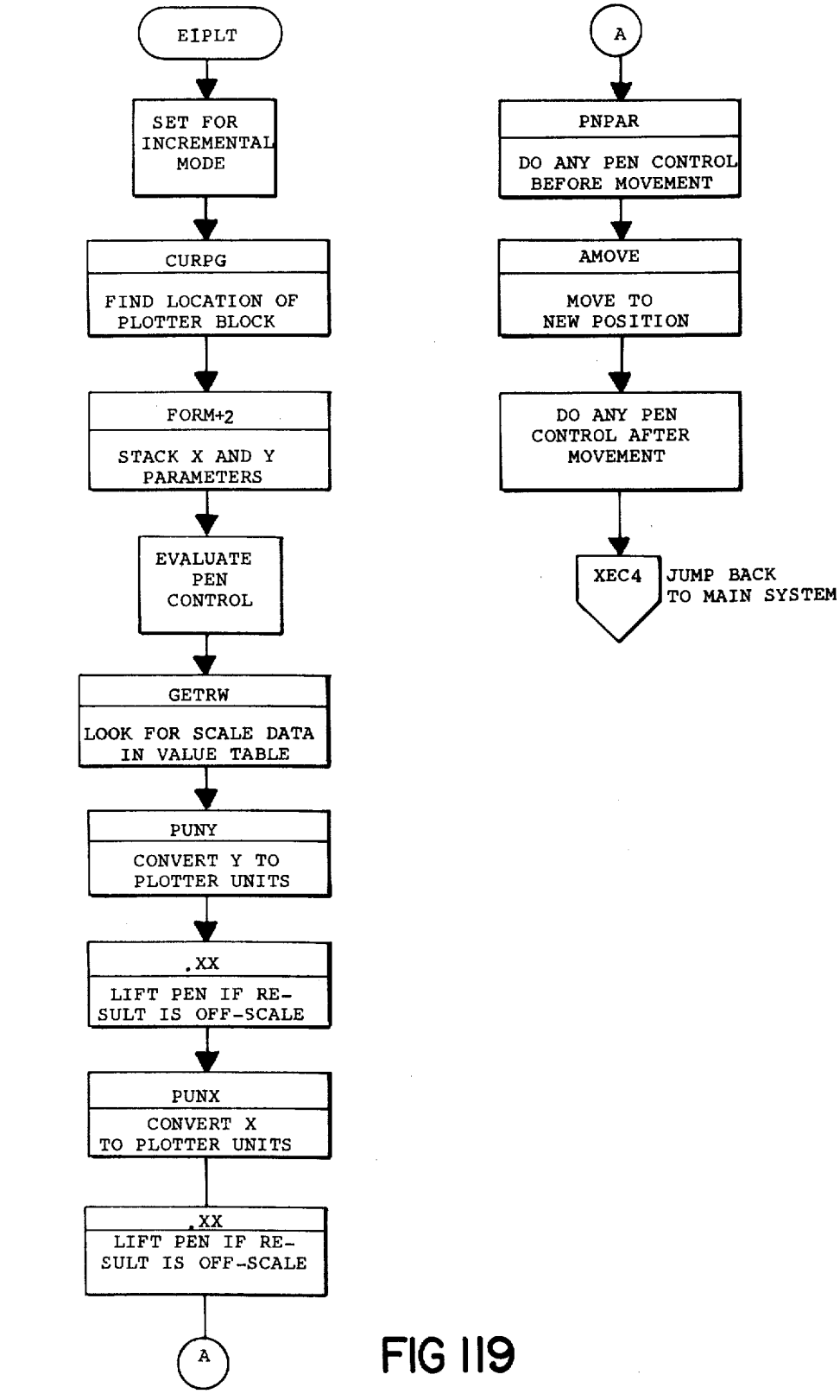


FIG 119

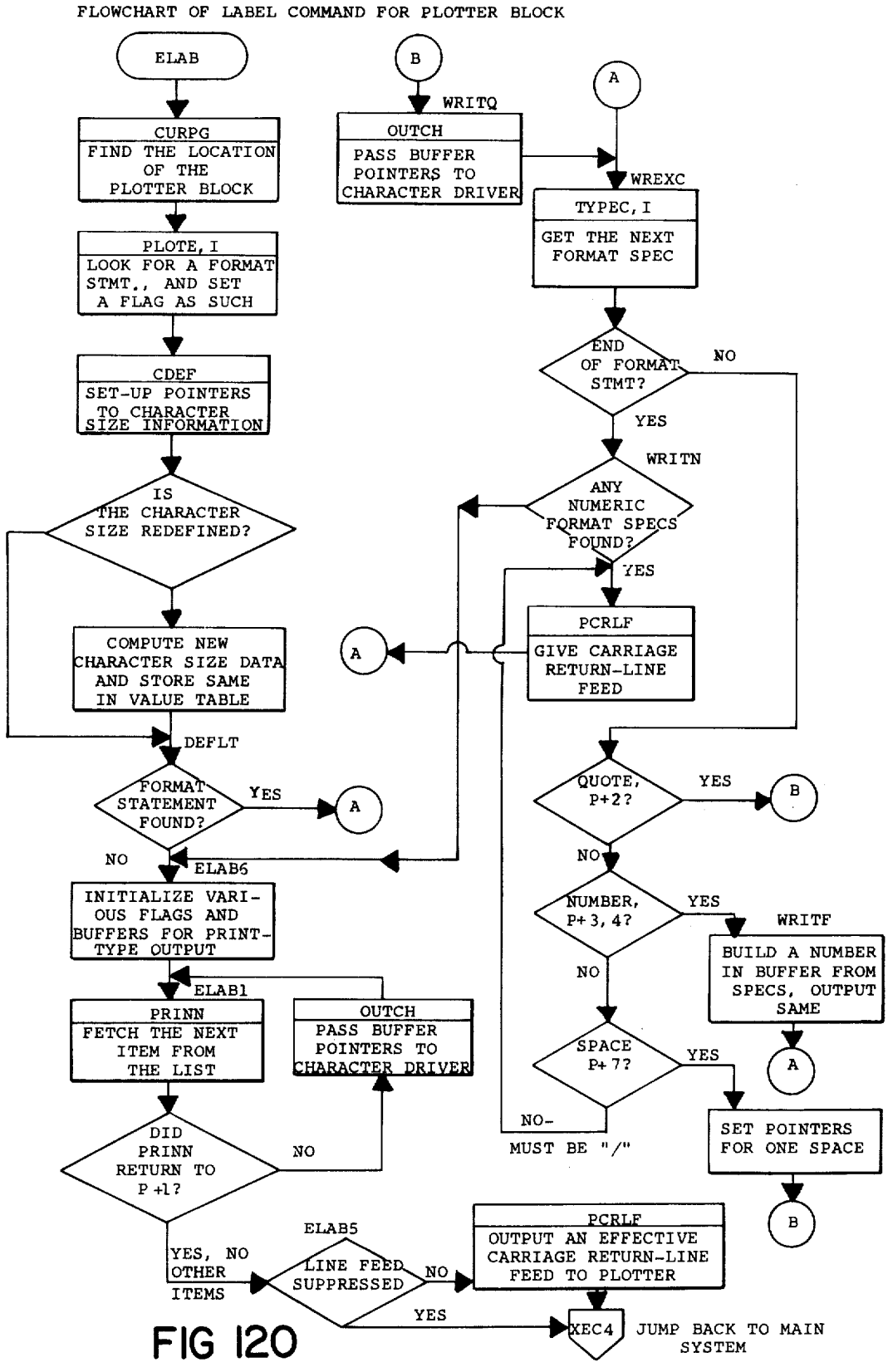

FLOWCHART OF LABEL COMMAND FOR PLOTTER BLOCK
ELAB
CURPG
FIND THE LOCATION OF THE PLOTTER BLOCK
PLOTE, I
LOOK FOR A FORMAT STMT., AND SET A FLAG AS SUCH
CDEF
SET-UP POINTERS TO CHARACTER SIZE INFORMATION
IS THE CHARACTER SIZE REDEFINED?
COMPUTE NEW CHARACTER SIZE DATA AND STORE SAME IN VALUE TABLE
DEFLT
FORMAT STATEMENT FOUND?
YES
A
NO
ELAB6
INITIALIZE VARIOUS FLAGS AND BUFFERS FOR PRINT-TYPE OUTPUT
ELAB1
PRINN
FETCH THE NEXT ITEM FROM THE LIST
OUTCH
PASS BUFFER POINTERS TO CHARACTER DRIVER
DID PRINN RETURN TO P+1?
NO
YES, NO OTHER ITEMS
ELAB5
LINE FEED SUPPRESSED
NO
YES
B
WRITQ
OUTCH
PASS BUFFER POINTERS TO CHARACTER DRIVER
A
WREXC
TYPEC, I
GET THE NEXT FORMAT SPEC
END OF FORMAT STMT?
NO
YES
WRITN
ANY NUMERIC FORMAT SPECS FOUND?
YES
PCRLF
GIVE CARRIAGE RETURN-LINE FEED
A
QUOTE, P+2?
YES
B
NO
NUMBER, P+3, 4?
YES
WRITF
BUILD A NUMBER IN BUFFER FROM SPECS, OUTPUT SAME
A
NO
SPACE P+7?
YES
SET POINTERS FOR ONE SPACE
B
NO-
MUST BE "/"
PCRLF
OUTPUT AN EFFECTIVE CARRIAGE RETURN-LINE FEED TO PLOTTER
XEC4
JUMP BACK TO MAIN SYSTEM

FIG 120

FLOWCHART OF LETTER COMMAND FOR PLOTTER BLOCK

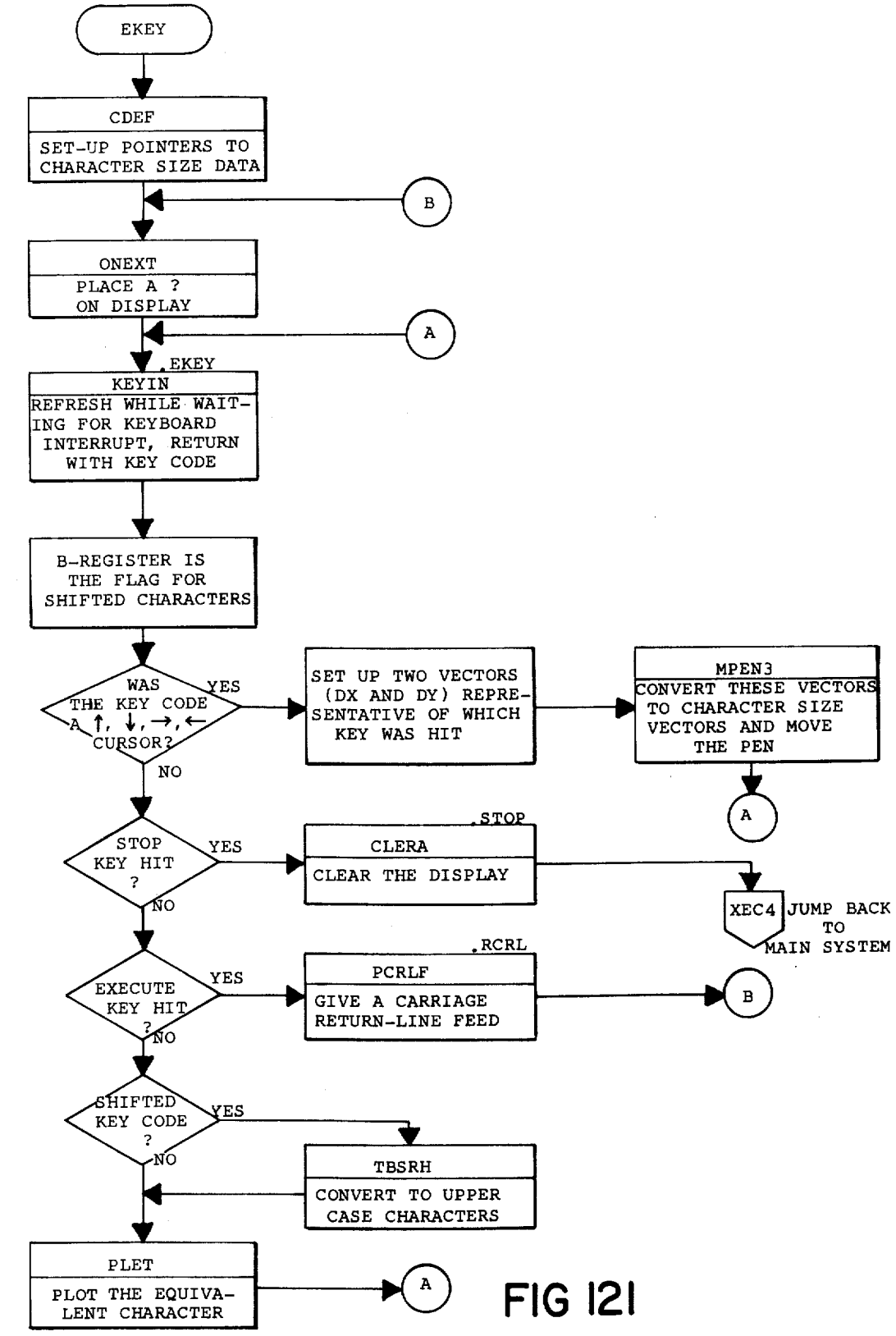


FIG 121

FLOWCHART OF CPLOT COMMAND FOR PLOTTER BLOCK

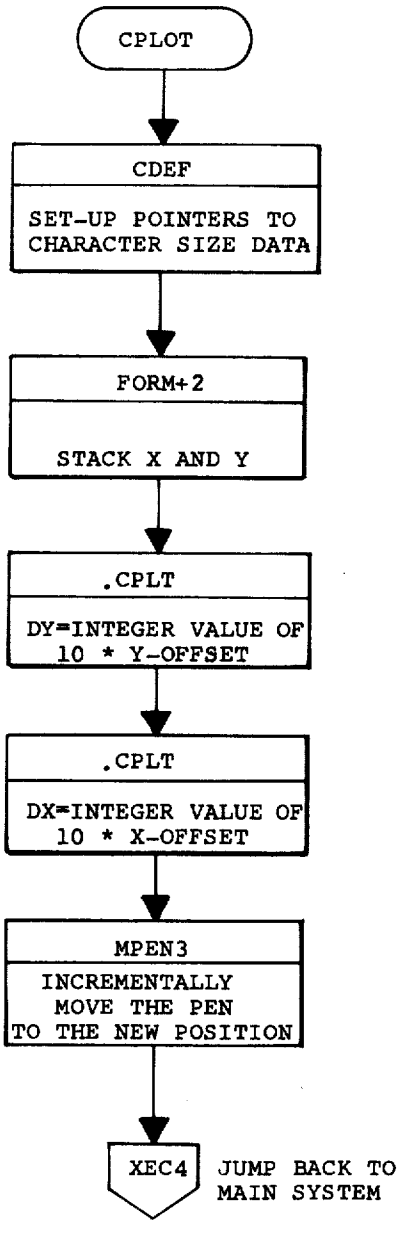


FIG 122

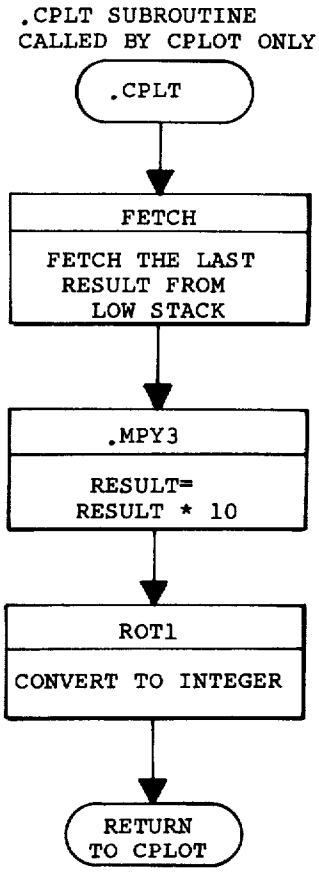


FIG 123

FLOWCHART OF XAXIS AND YAXIS COMMAND FOR PLOTTER BLOCK

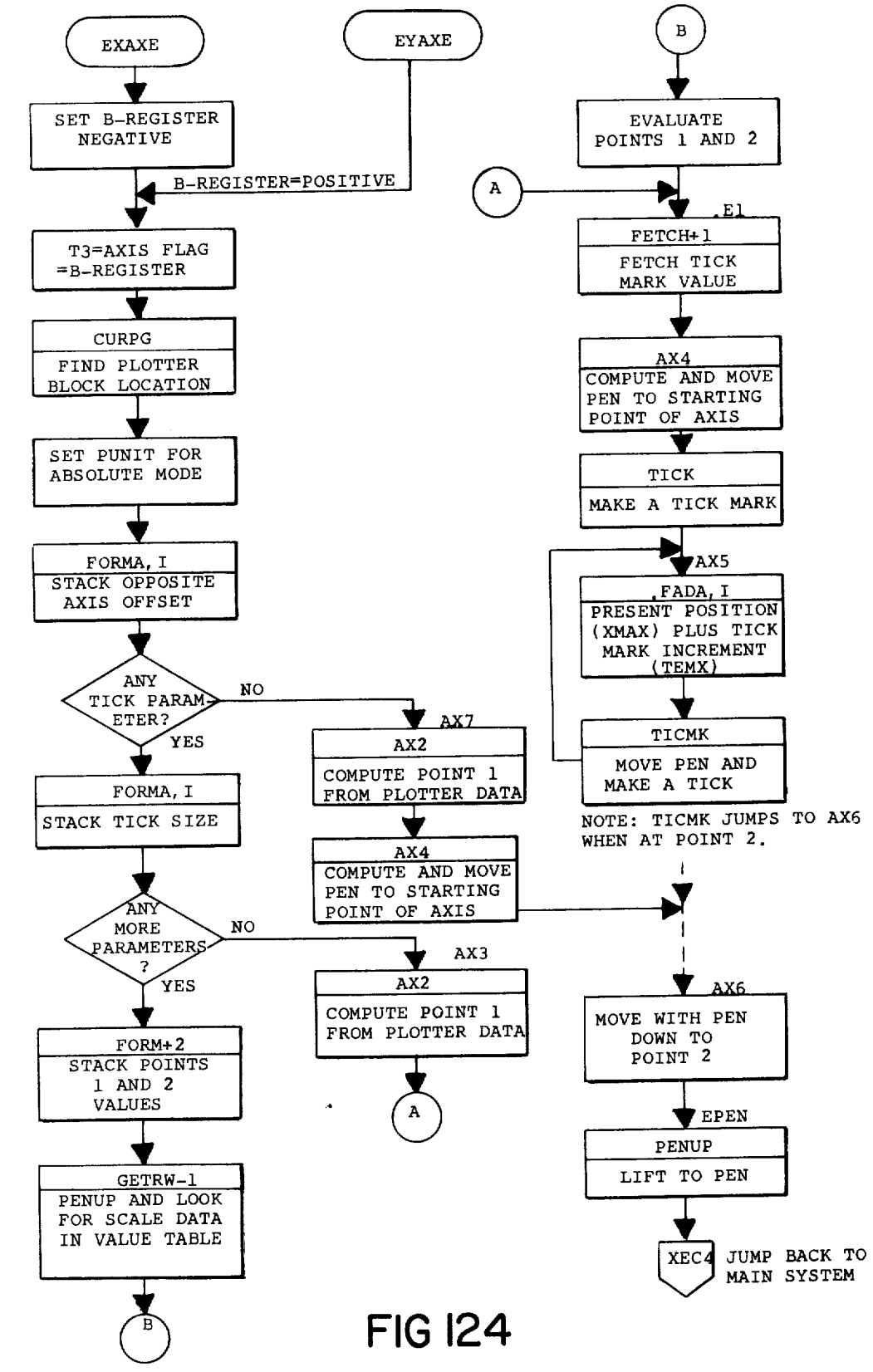


FIG 124

FLOWCHART OF AXIS SUBROUTINES FOR PLOTTER BLOCK

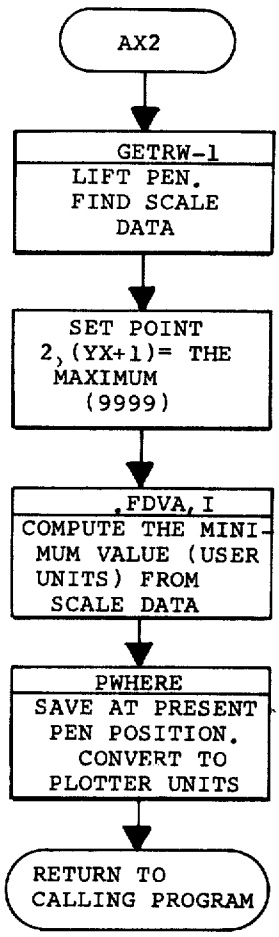


FIG 125A

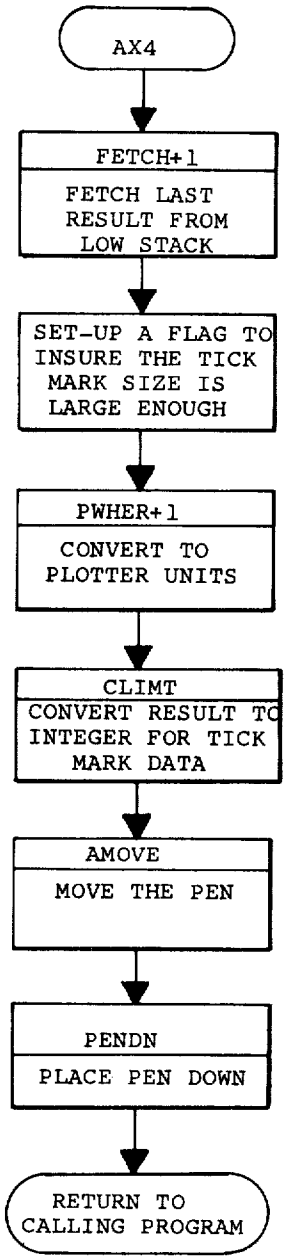


FIG 125B

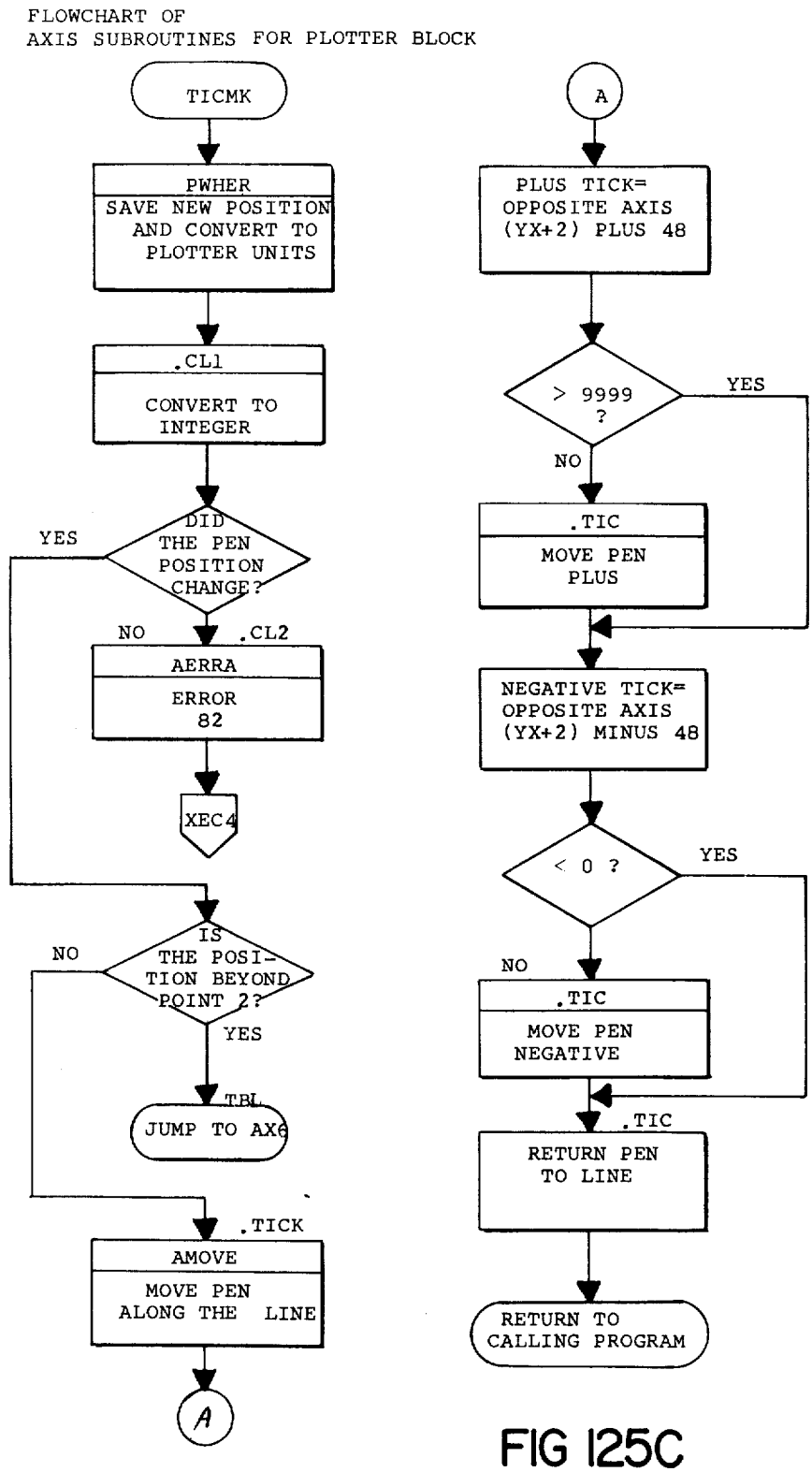

FLOWCHART OF
AXIS SUBROUTINES FOR PLOTTER BLOCK
TICMK
PWHER
SAVE NEW POSITION AND CONVERT TO PLOTTER UNITS
.CL1
CONVERT TO INTEGER
DID THE PEN POSITION CHANGE?
YES
NO
.CL2
AERRA
ERROR 82
XEC4
IS THE POSITION BEYOND POINT 2?
NO
YES
TBL
JUMP TO AXE
.TICK
AMOVE
MOVE PEN ALONG THE LINE
A
A
PLUS TICK= OPPOSITE AXIS (YX+2) PLUS 48
> 9999 ?
YES
NO
.TIC
MOVE PEN PLUS
NEGATIVE TICK= OPPOSITE AXIS (YX+2) MINUS 48
< 0 ?
YES
NO
.TIC
MOVE PEN NEGATIVE
.TIC
RETURN PEN TO LINE
RETURN TO CALLING PROGRAM

FIG 125C

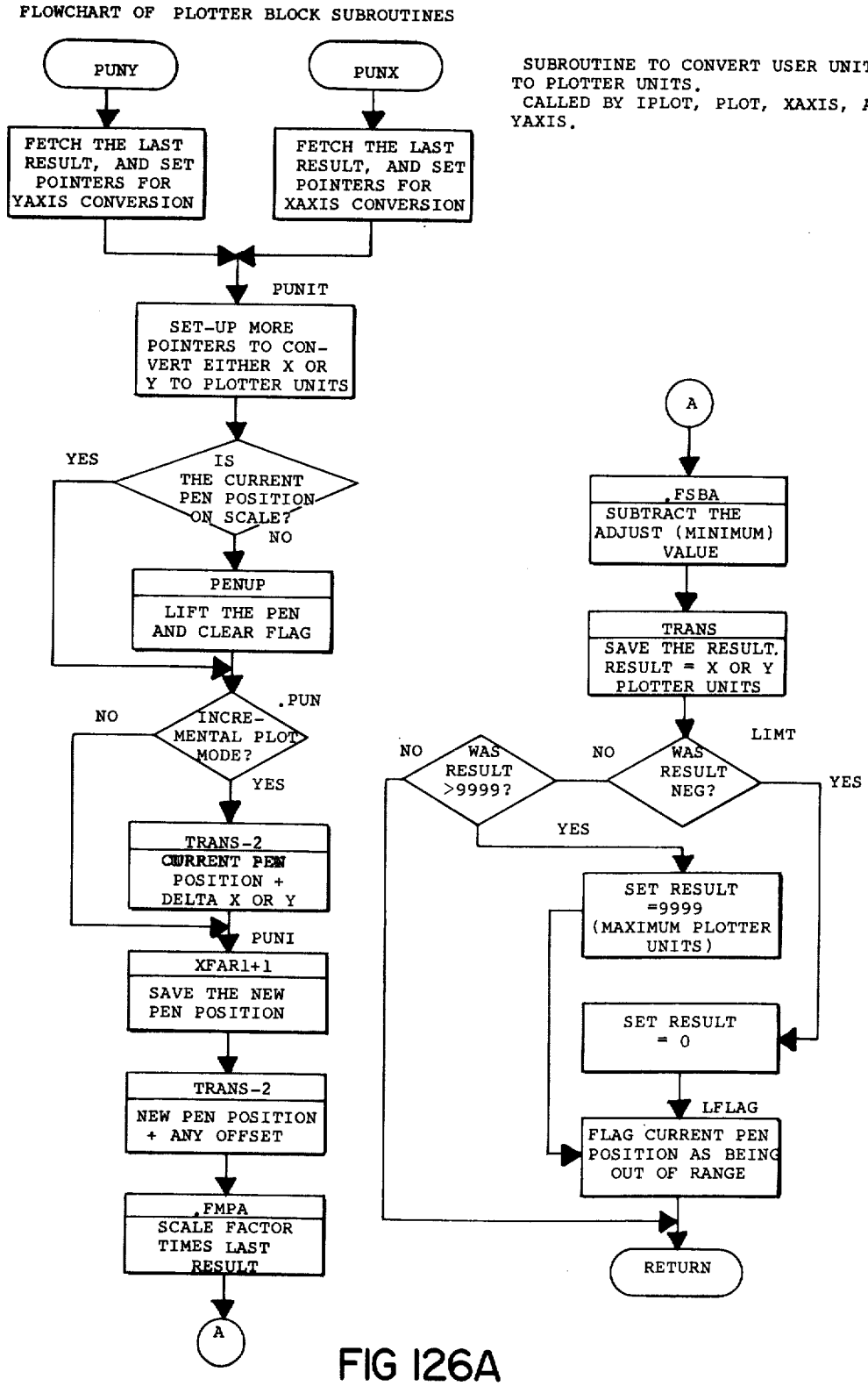

FLOWCHART OF PLOTTER BLOCK SUBROUTINES
PUNY
PUNX
SUBROUTINE TO CONVERT USER UNITS TO PLOTTER UNITS.
CALLED BY IPLOT, PLOT, XAXIS, AND YAXIS.
FETCH THE LAST RESULT, AND SET POINTERS FOR YAXIS CONVERSION
FETCH THE LAST RESULT, AND SET POINTERS FOR XAXIS CONVERSION
PUNIT
SET-UP MORE POINTERS TO CONVERT EITHER X OR Y TO PLOTTER UNITS
YES
IS THE CURRENT PEN POSITION ON SCALE?
NO
PENUP
LIFT THE PEN AND CLEAR FLAG
.PUN
NO
INCREMENTAL PLOT MODE?
YES
TRANS-2
CURRENT PEN POSITION + DELTA X OR Y
PUNI
XFAR1+1
SAVE THE NEW PEN POSITION
TRANS-2
NEW PEN POSITION + ANY OFFSET
.FMPA
SCALE FACTOR TIMES LAST RESULT
A
A
.FSBA
SUBTRACT THE ADJUST (MINIMUM) VALUE
TRANS
SAVE THE RESULT. RESULT = X OR Y PLOTTER UNITS
LIMT
NO
WAS RESULT >9999?
NO
WAS RESULT NEG?
YES
YES
SET RESULT =9999 (MAXIMUM PLOTTER UNITS)
SET RESULT = 0
LFLAG
FLAG CURRENT PEN POSITION AS BEING OUT OF RANGE
RETURN

FIG 126A

FLOWCHART OF PLOTTER BLOCK SUBROUTINES

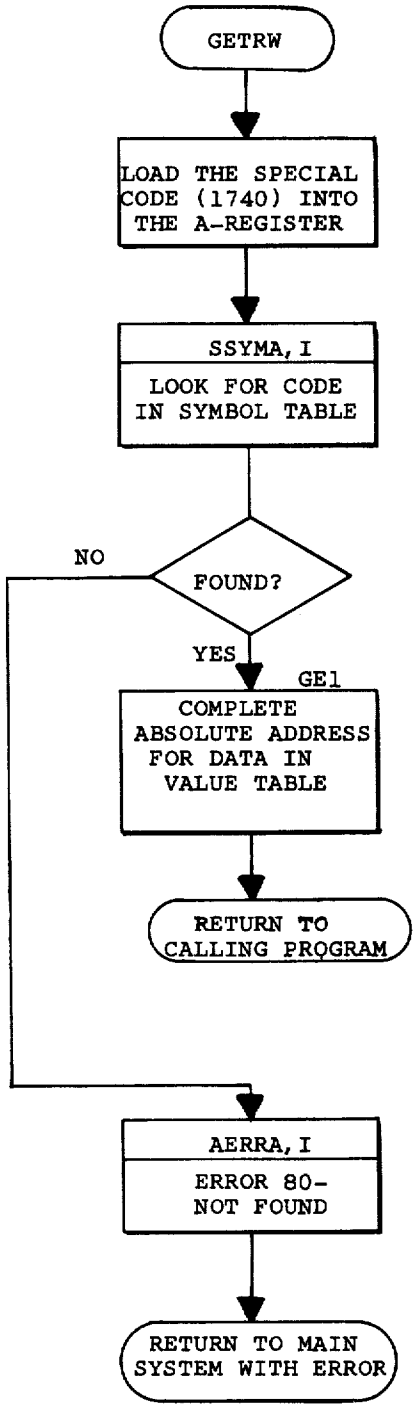


FIG 126B

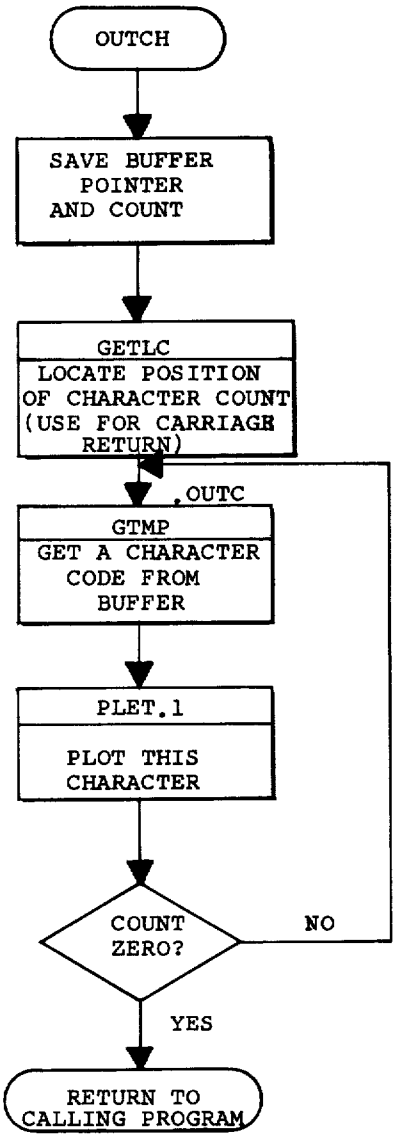


FIG 126C

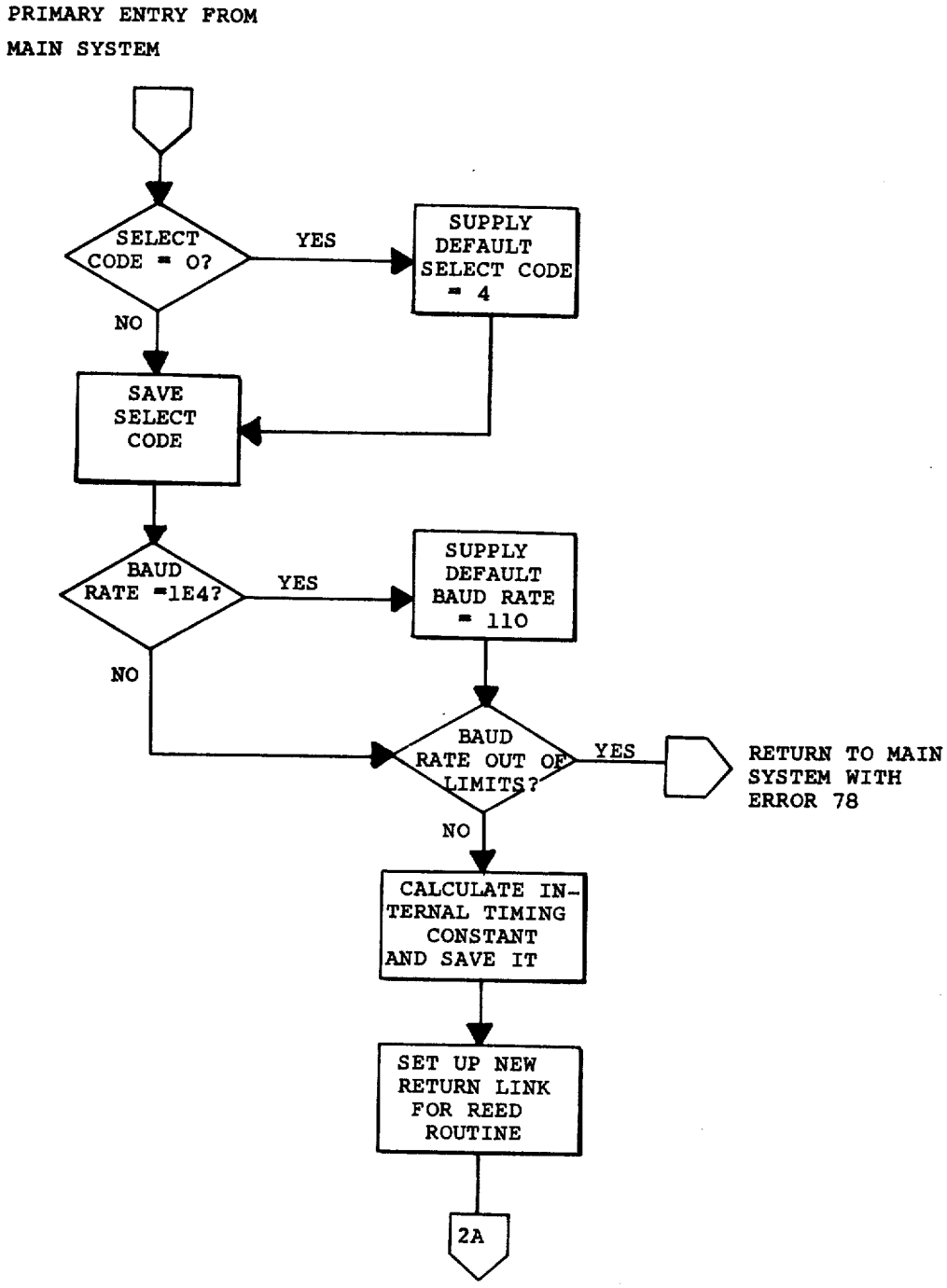

PRIMARY ENTRY FROM MAIN SYSTEM
SELECT CODE = 0?
YES
SUPPLY DEFAULT SELECT CODE = 4
NO
SAVE SELECT CODE
BAUD RATE =1E4?
YES
SUPPLY DEFAULT BAUD RATE = 110
NO
BAUD RATE OUT OF LIMITS?
YES
RETURN TO MAIN SYSTEM WITH ERROR 78
NO
CALCULATE INTERNAL TIMING CONSTANT AND SAVE IT
SET UP NEW RETURN LINK FOR REED ROUTINE
2A

FIG 127

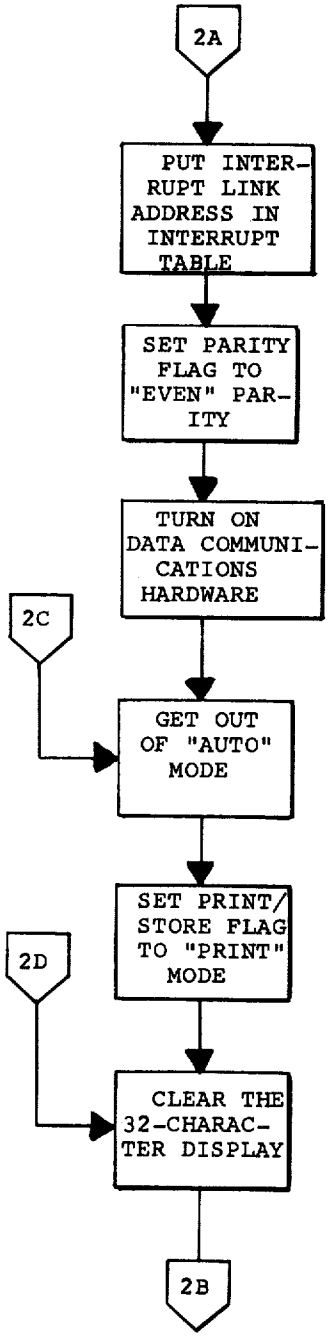


FIG 128A

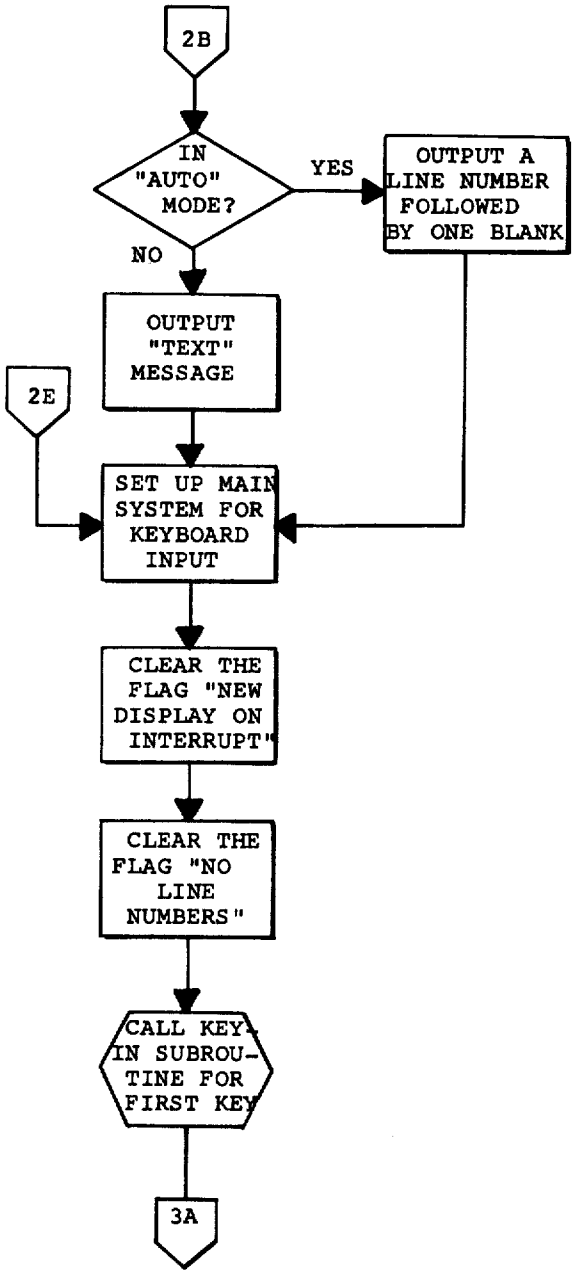


FIG 128B

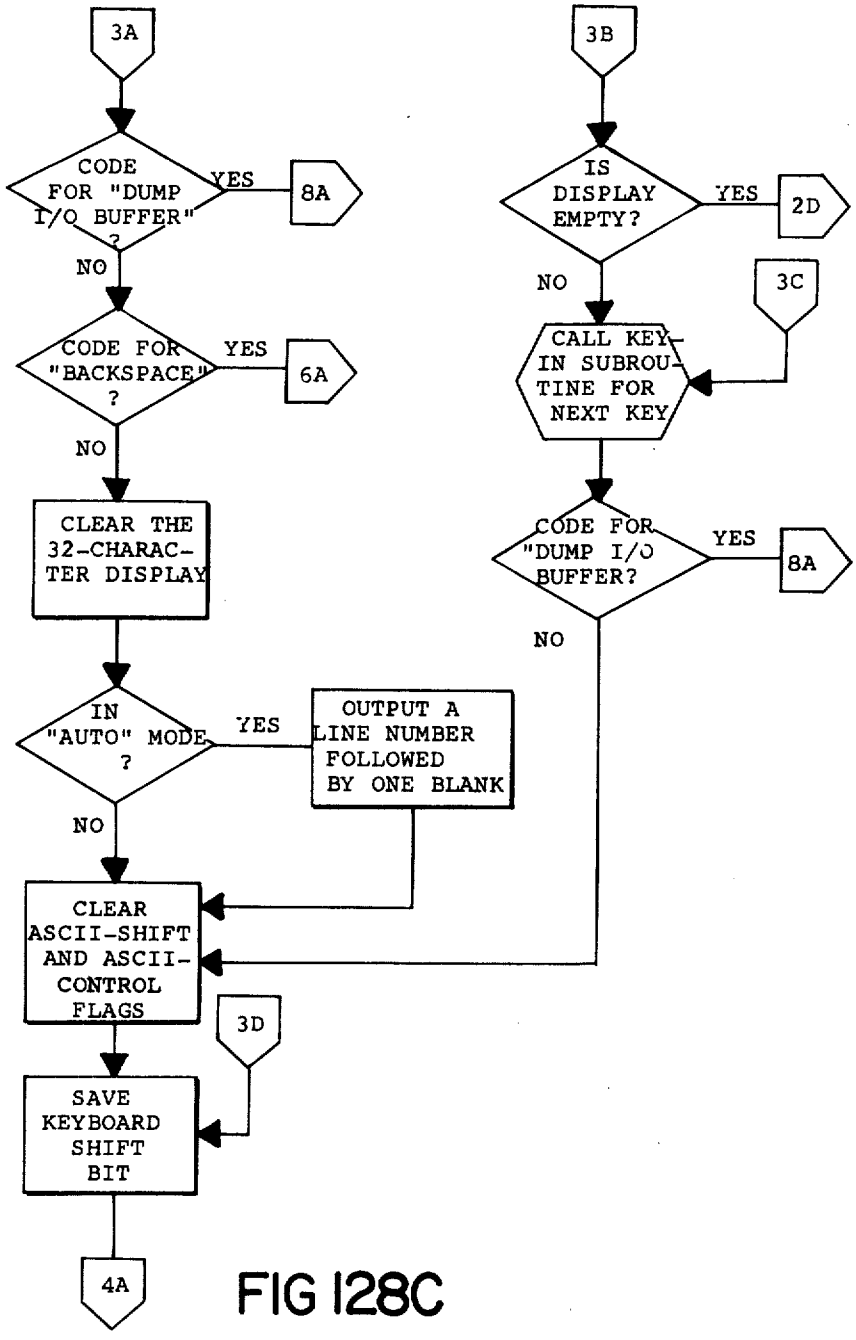

3A
CODE FOR "DUMP I/O BUFFER" ?
YES
8A
NO
CODE FOR "BACKSPACE" ?
YES
6A
NO
CLEAR THE 32-CHARACTER DISPLAY
IN "AUTO" MODE ?
YES
OUTPUT A LINE NUMBER FOLLOWED BY ONE BLANK
NO
CLEAR ASCII-SHIFT AND ASCII-CONTROL FLAGS
3D
SAVE KEYBOARD SHIFT BIT
4A
3B
IS DISPLAY EMPTY?
YES
2D
NO
3C
CALL KEY IN SUBROUTINE FOR NEXT KEY
CODE FOR "DUMP I/O BUFFER?
YES
8A
NO

FIG 128C

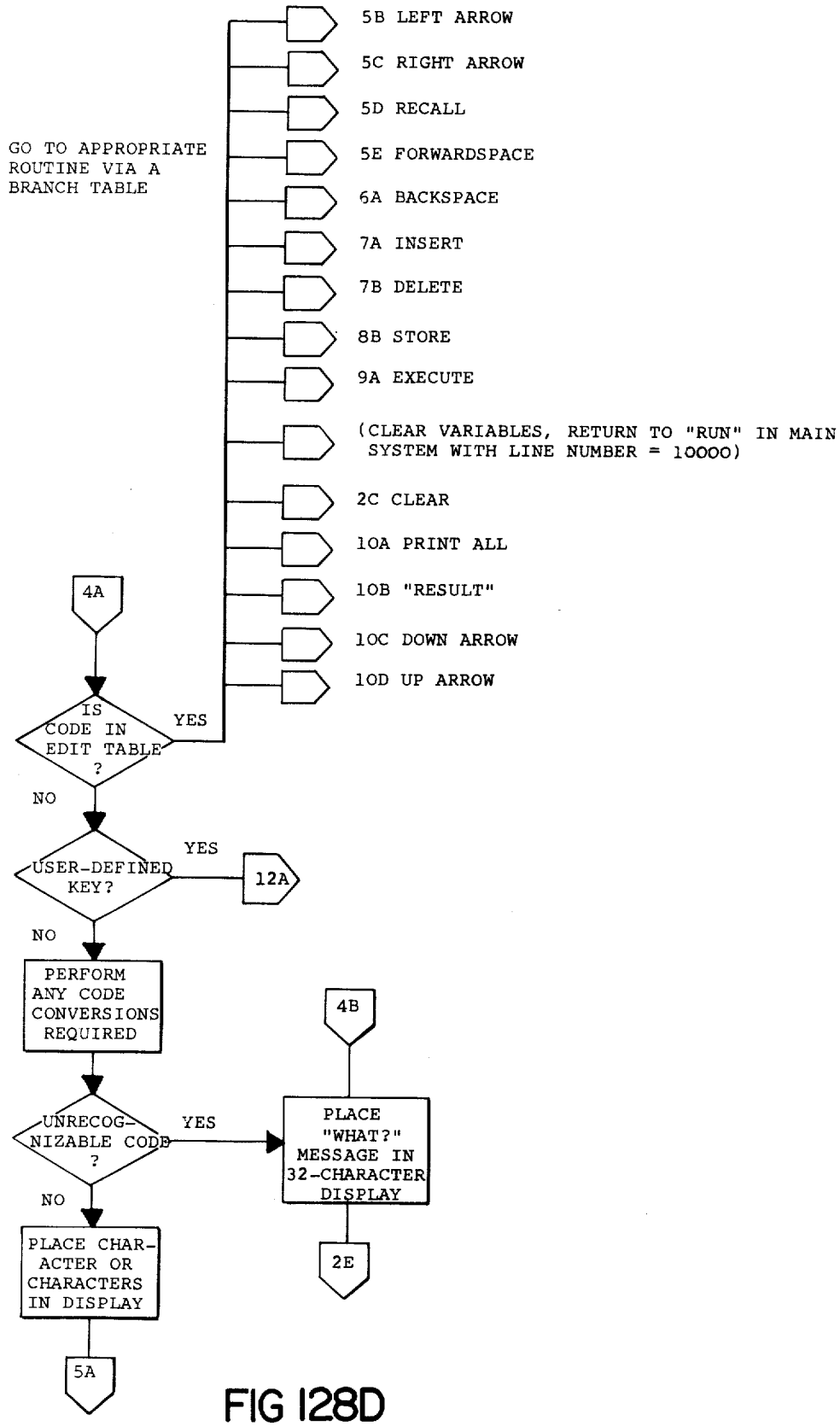

GO TO APPROPRIATE ROUTINE VIA A BRANCH TABLE
5B LEFT ARROW
5C RIGHT ARROW
5D RECALL
5E FORWARDSPACE
6A BACKSPACE
7A INSERT
7B DELETE
8B STORE
9A EXECUTE
(CLEAR VARIABLES, RETURN TO "RUN" IN MAIN SYSTEM WITH LINE NUMBER = 10000)
2C CLEAR
10A PRINT ALL
10B "RESULT"
10C DOWN ARROW
10D UP ARROW
4A
IS CODE IN EDIT TABLE ?
YES
NO
USER-DEFINED KEY?
YES
12A
NO
PERFORM ANY CODE CONVERSIONS REQUIRED
4B
UNRECOG-NIZABLE CODE ?
YES
PLACE "WHAT?" MESSAGE IN 32-CHARACTER DISPLAY
2E
NO
PLACE CHAR-ACTER OR CHARACTERS IN DISPLAY
5A

FIG 128D

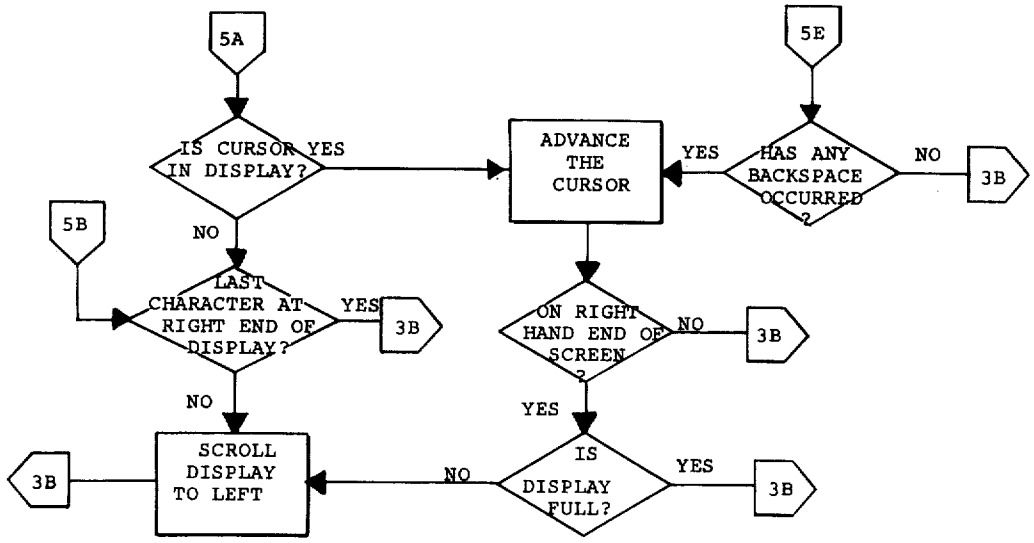


FIG 128E

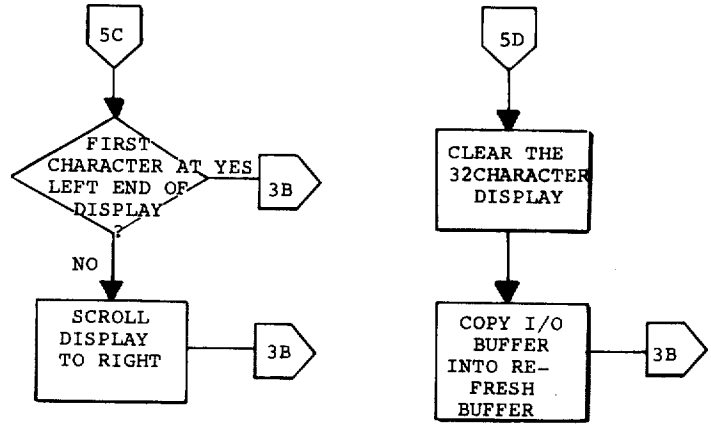


FIG 128F

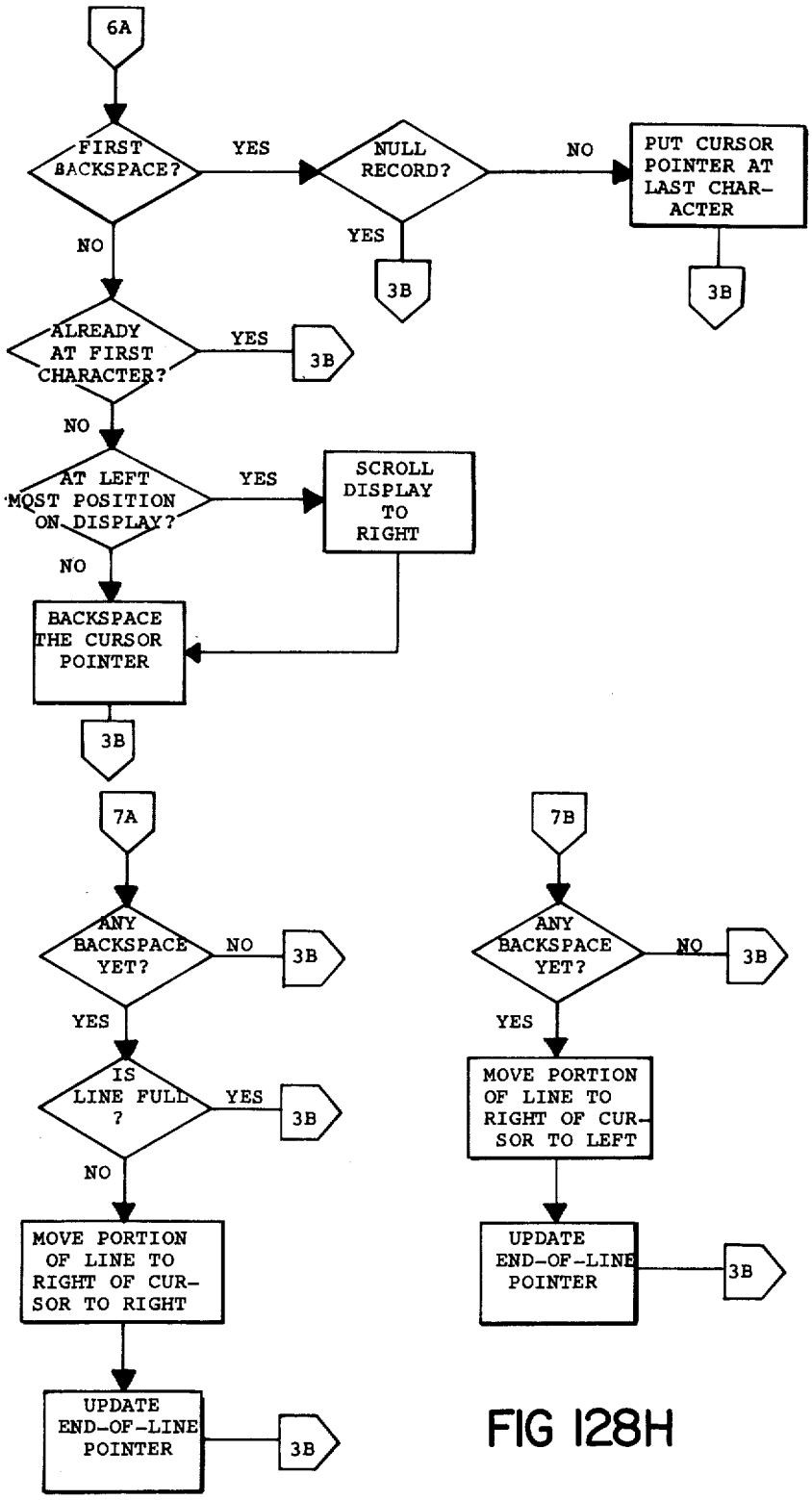


FIG 128G

FIG 128H

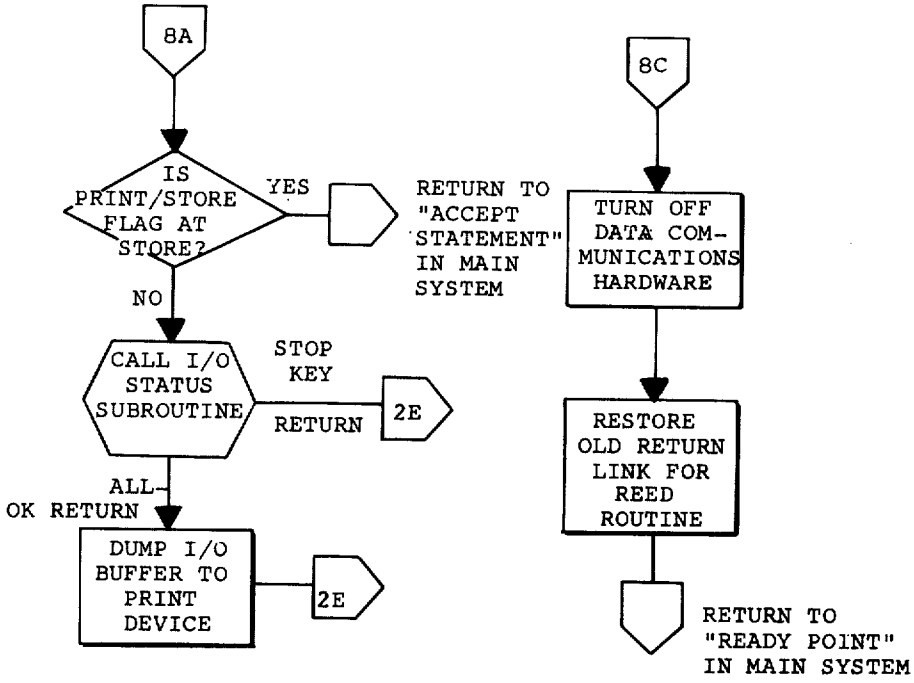


FIG 128I

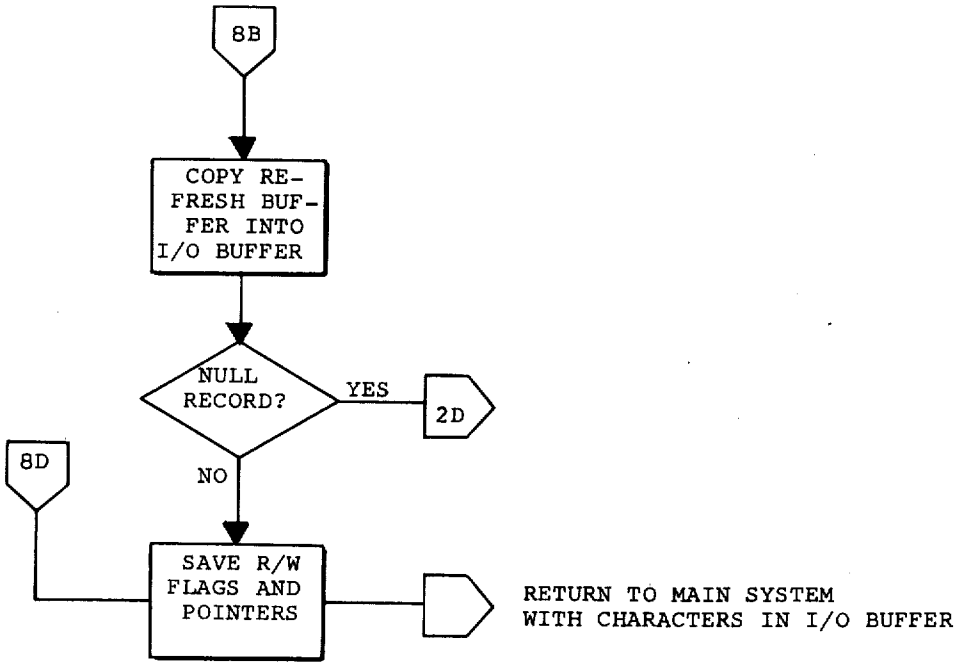


FIG 128J

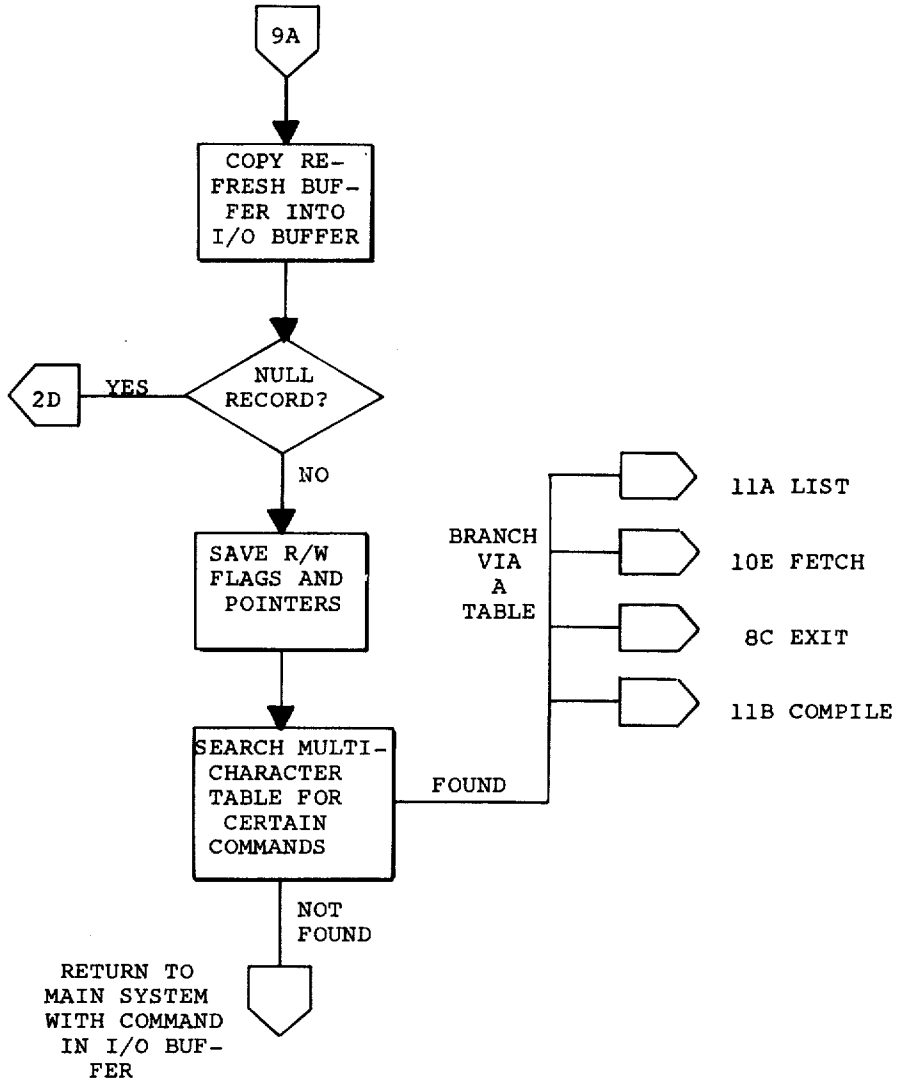

9A
COPY RE-FRESH BUF-FER INTO I/O BUFFER
2D
YES
NULL RECORD?
NO
SAVE R/W FLAGS AND POINTERS
SEARCH MULTI-CHARACTER TABLE FOR CERTAIN COMMANDS
FOUND
BRANCH VIA A TABLE
11A LIST
10E FETCH
8C EXIT
11B COMPILE
NOT FOUND
RETURN TO MAIN SYSTEM WITH COMMAND IN I/O BUF-FER

FIG 128K

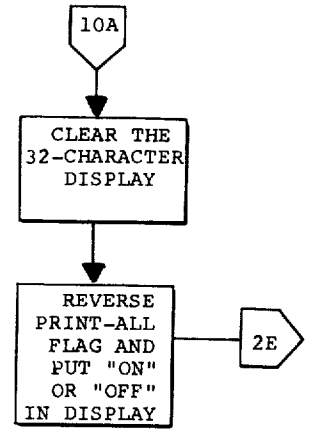


FIG 128L

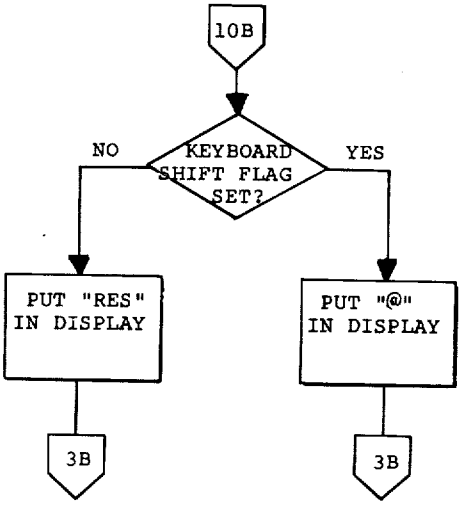


FIG 128M

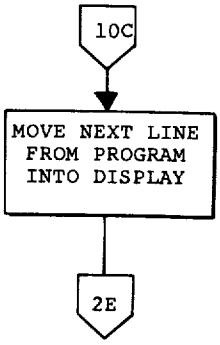


FIG 128N

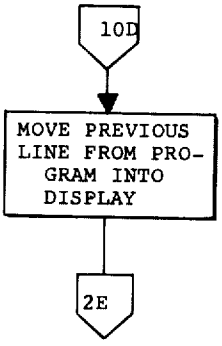


FIG 128O

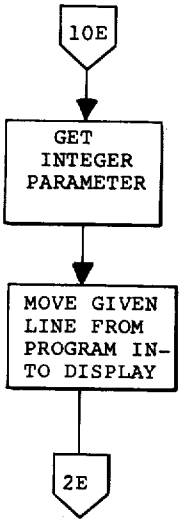


FIG 128P

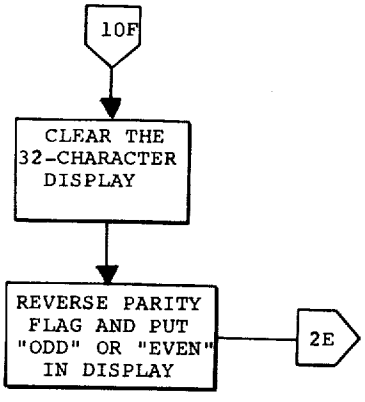


FIG 128Q

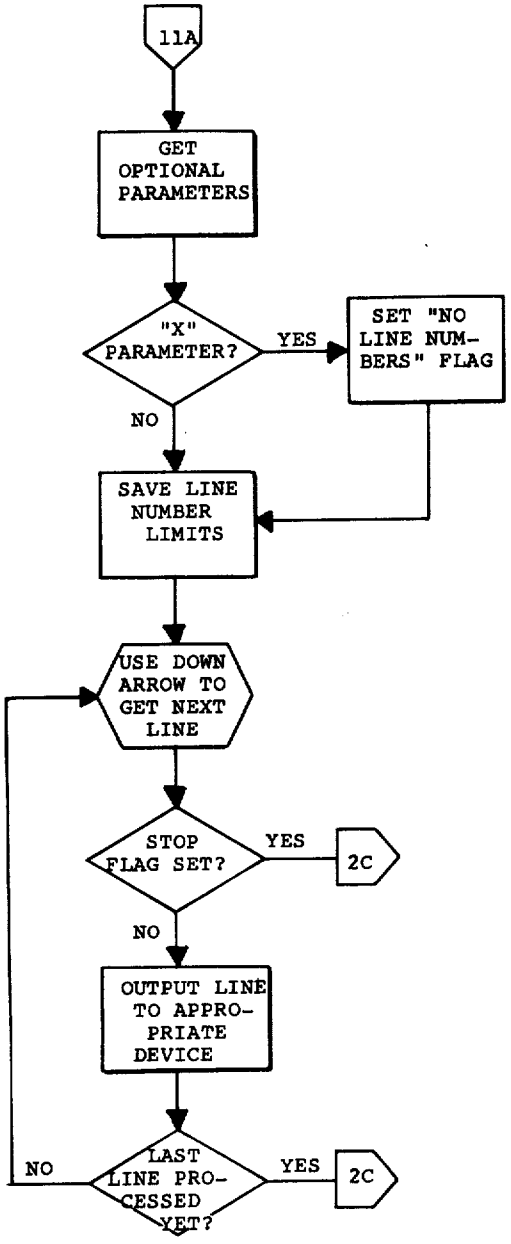


FIG 128R

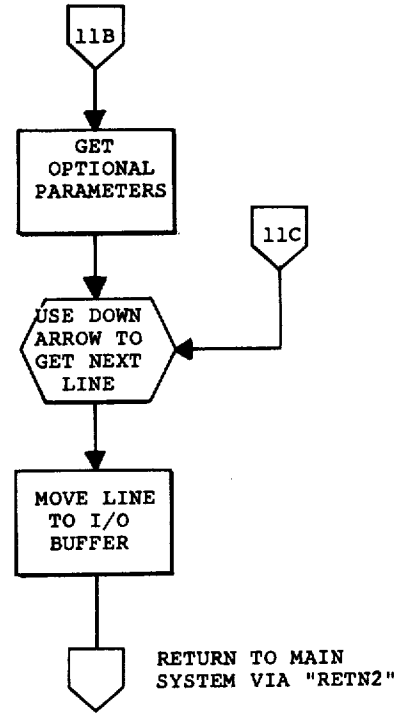


FIG 128S

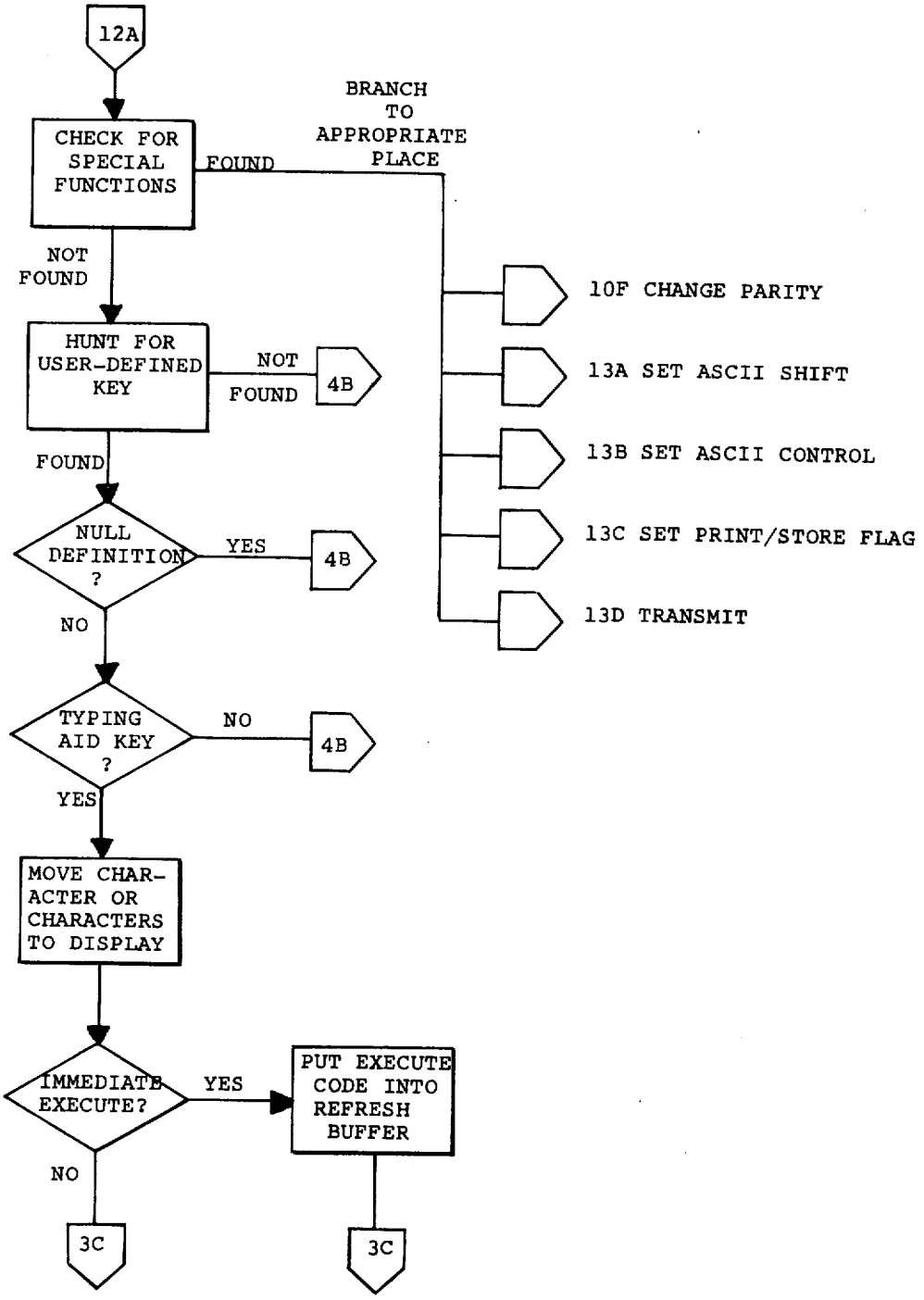

12A
CHECK FOR SPECIAL FUNCTIONS
FOUND
BRANCH TO APPROPRIATE PLACE
NOT FOUND
HUNT FOR USER-DEFINED KEY
NOT FOUND
4B
FOUND
NULL DEFINITION ?
YES
4B
NO
TYPING AID KEY ?
NO
4B
YES
MOVE CHARACTER OR CHARACTERS TO DISPLAY
IMMEDIATE EXECUTE?
YES
PUT EXECUTE CODE INTO REFRESH BUFFER
NO
3C
3C
10F CHANGE PARITY
13A SET ASCII SHIFT
13B SET ASCII CONTROL
13C SET PRINT/STORE FLAG
13D TRANSMIT

FIG 128T

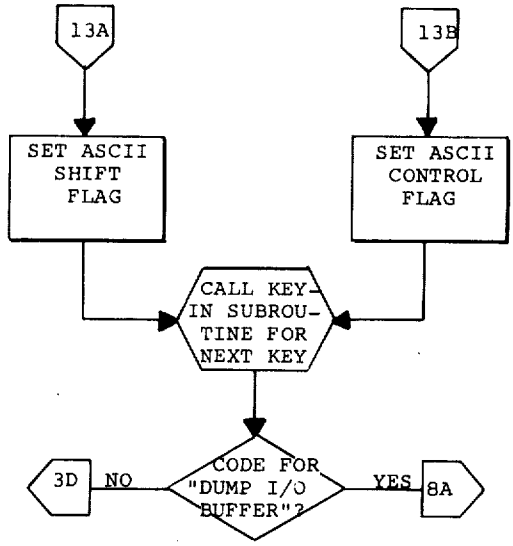


FIG 128U

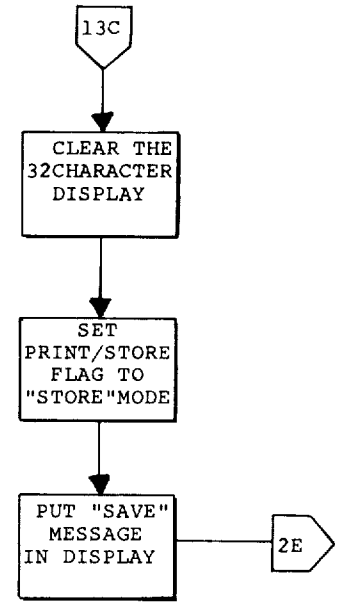


FIG 128V

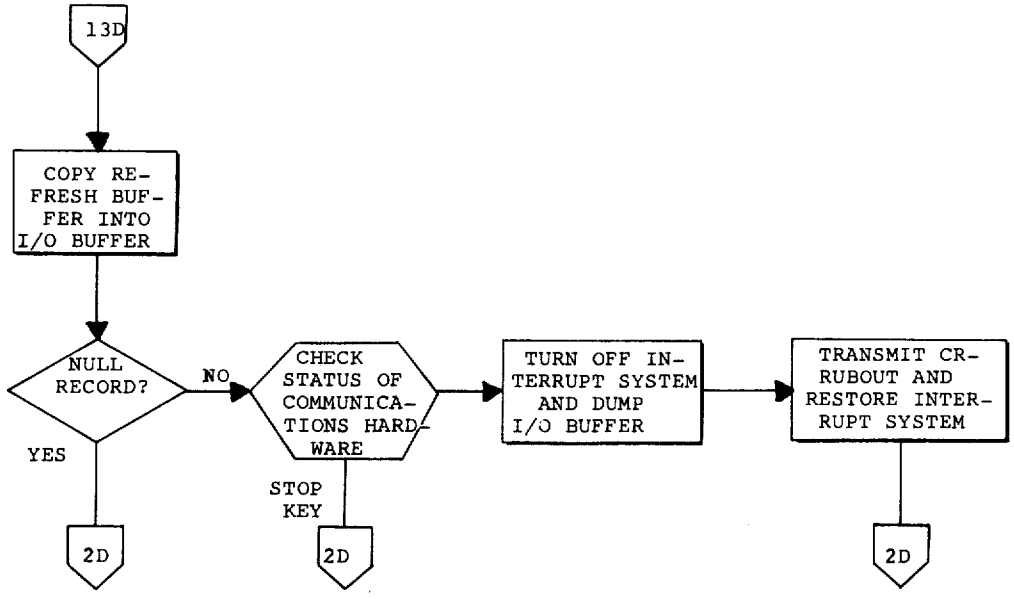


FIG 128W

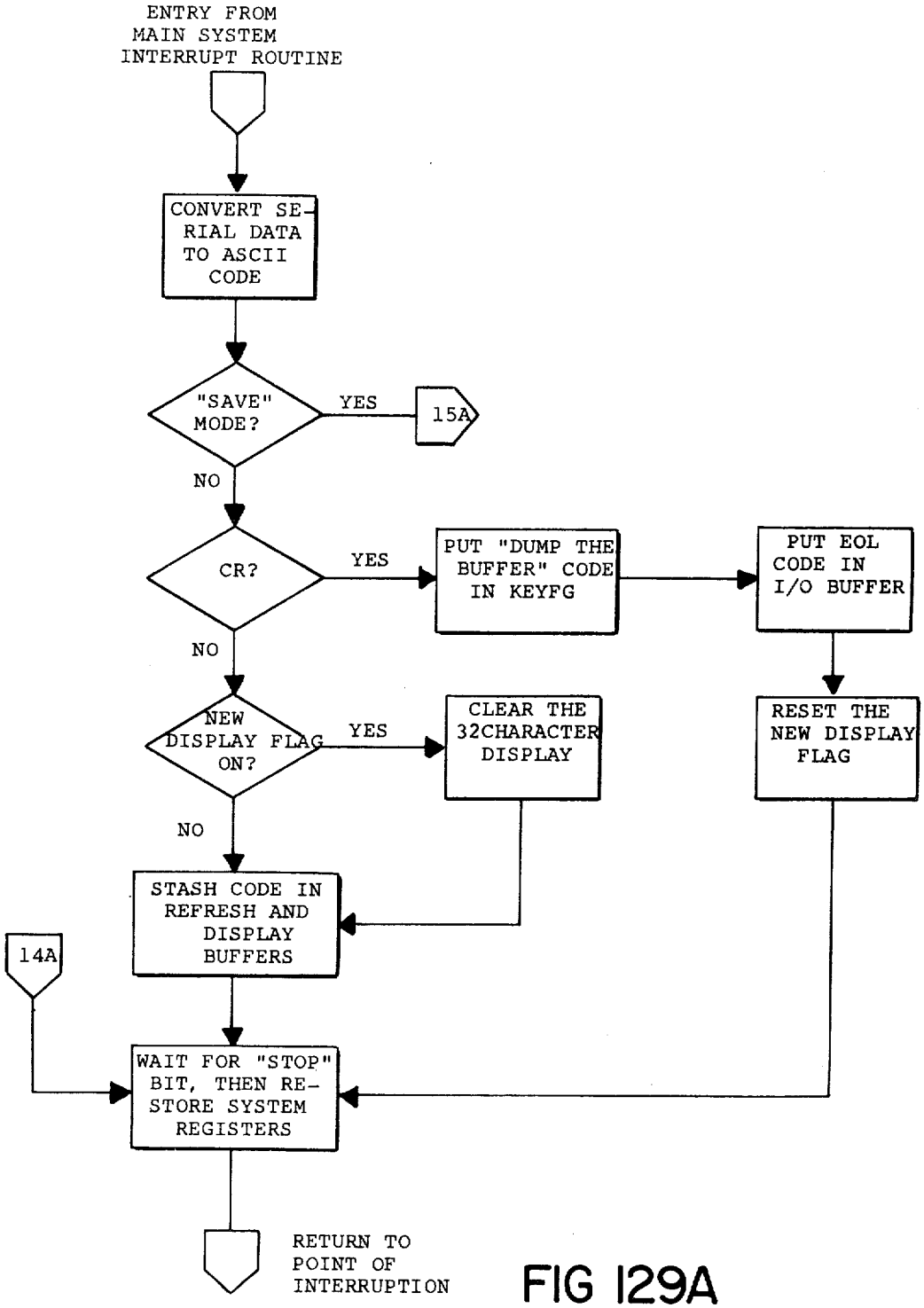

ENTRY FROM MAIN SYSTEM INTERRUPT ROUTINE
CONVERT SERIAL DATA TO ASCII CODE
"SAVE" MODE?
YES
15A
NO
CR?
YES
PUT "DUMP THE BUFFER" CODE IN KEYFG
PUT EOL CODE IN I/O BUFFER
NO
NEW DISPLAY FLAG ON?
YES
CLEAR THE 32CHARACTER DISPLAY
RESET THE NEW DISPLAY FLAG
NO
STASH CODE IN REFRESH AND DISPLAY BUFFERS
14A
WAIT FOR "STOP" BIT, THEN RESTORE SYSTEM REGISTERS
RETURN TO POINT OF INTERRUPTION

FIG 129A

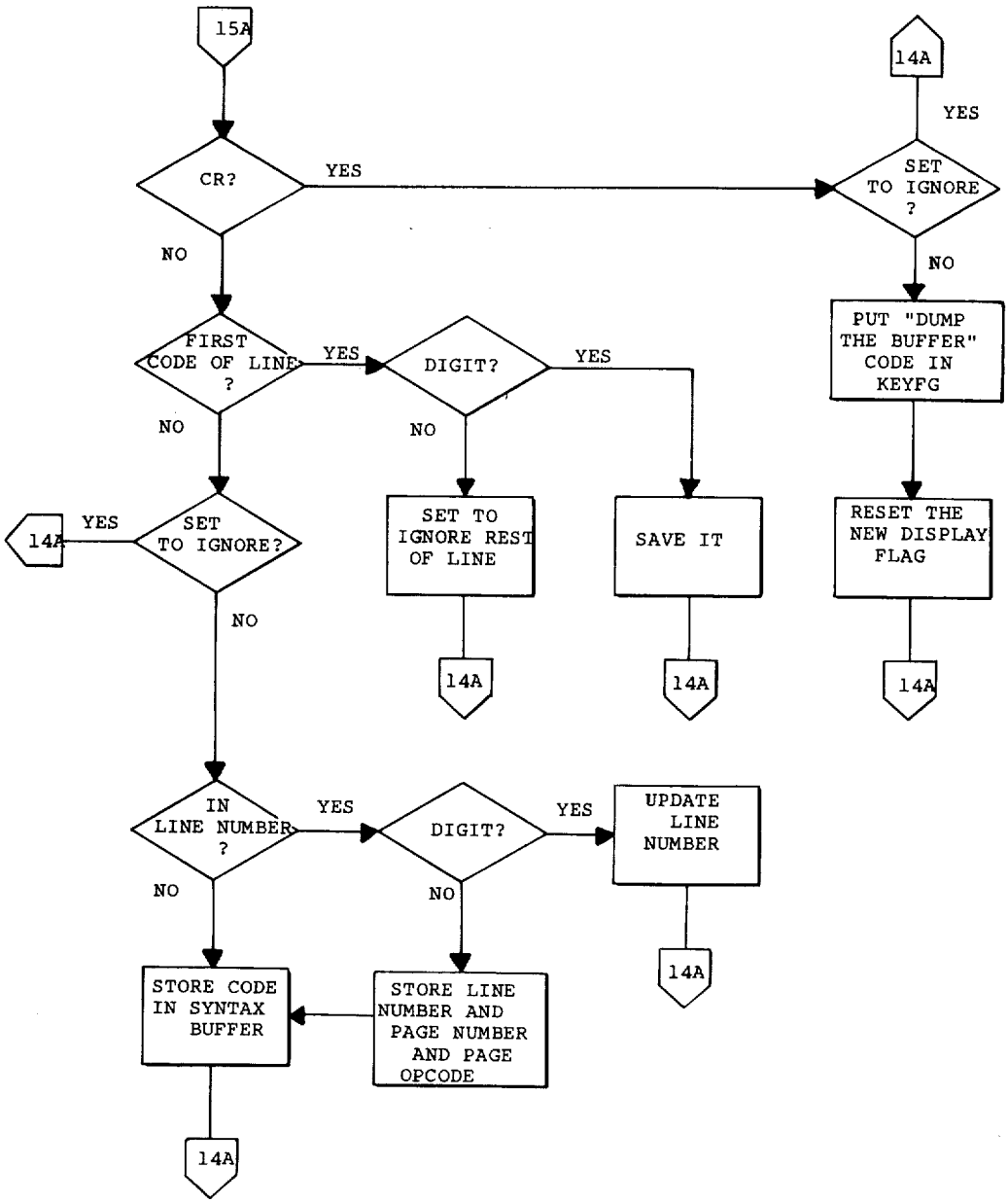

15A
CR?
YES
NO
14A
YES
SET TO IGNORE ?
NO
PUT "DUMP THE BUFFER" CODE IN KEYFG
RESET THE NEW DISPLAY FLAG
14A
FIRST CODE OF LINE ?
YES
DIGIT?
YES
NO
NO
SET TO IGNORE REST OF LINE
SAVE IT
14A
14A
14A
YES
14A
SET TO IGNORE?
NO
IN LINE NUMBER ?
YES
DIGIT?
YES
UPDATE LINE NUMBER
NO
NO
14A
STORE CODE IN SYNTAX BUFFER
STORE LINE NUMBER AND PAGE NUMBER AND PAGE OPCODE
14A

FIG 129B

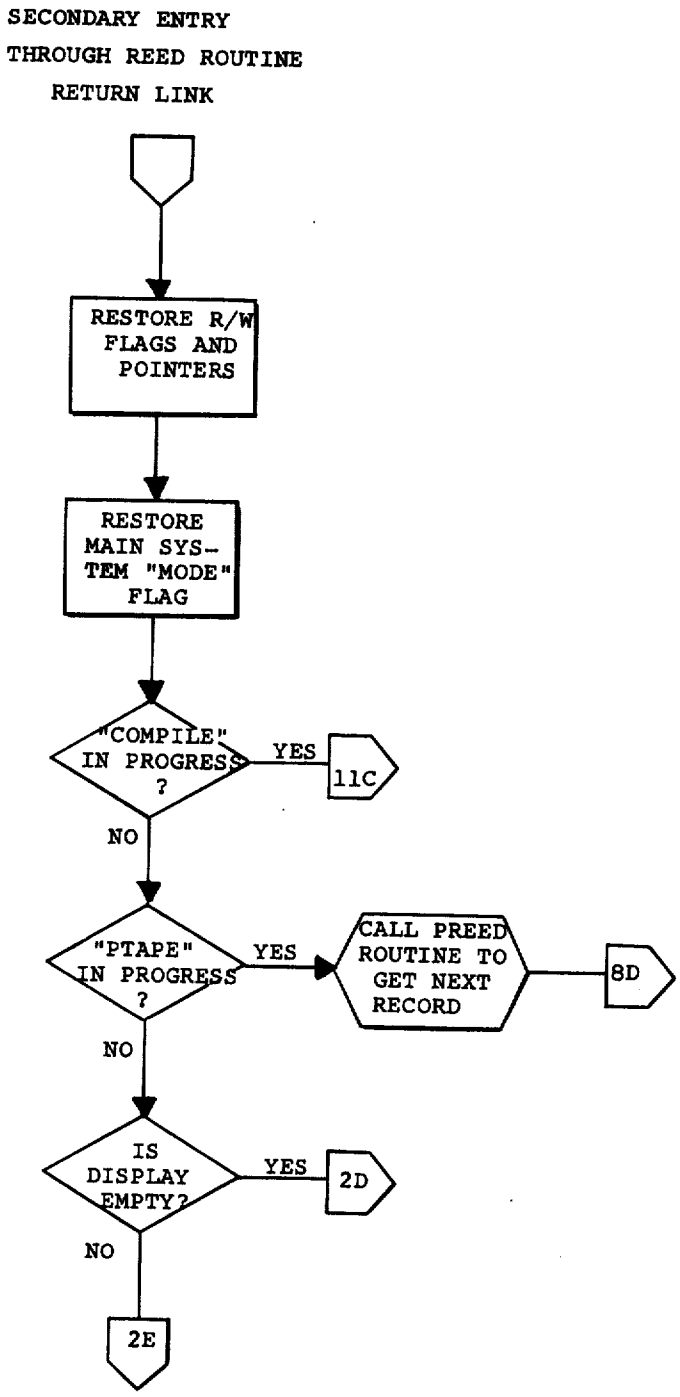

SECONDARY ENTRY
THROUGH REED ROUTINE
RETURN LINK
RESTORE R/W FLAGS AND POINTERS
RESTORE MAIN SYS-TEM "MODE" FLAG
"COMPILE" IN PROGRESS ?
YES
11C
NO
"PTAPE" IN PROGRESS ?
YES
CALL PREED ROUTINE TO GET NEXT RECORD
8D
NO
IS DISPLAY EMPTY?
YES
2D
NO
2E

FIG 130

FLOWCHART OF THE I/O BLOCK'S OUTPUT STATEMENT ILLUSTRATING INTERNAL CONVERSION AND CODE CONVERSION

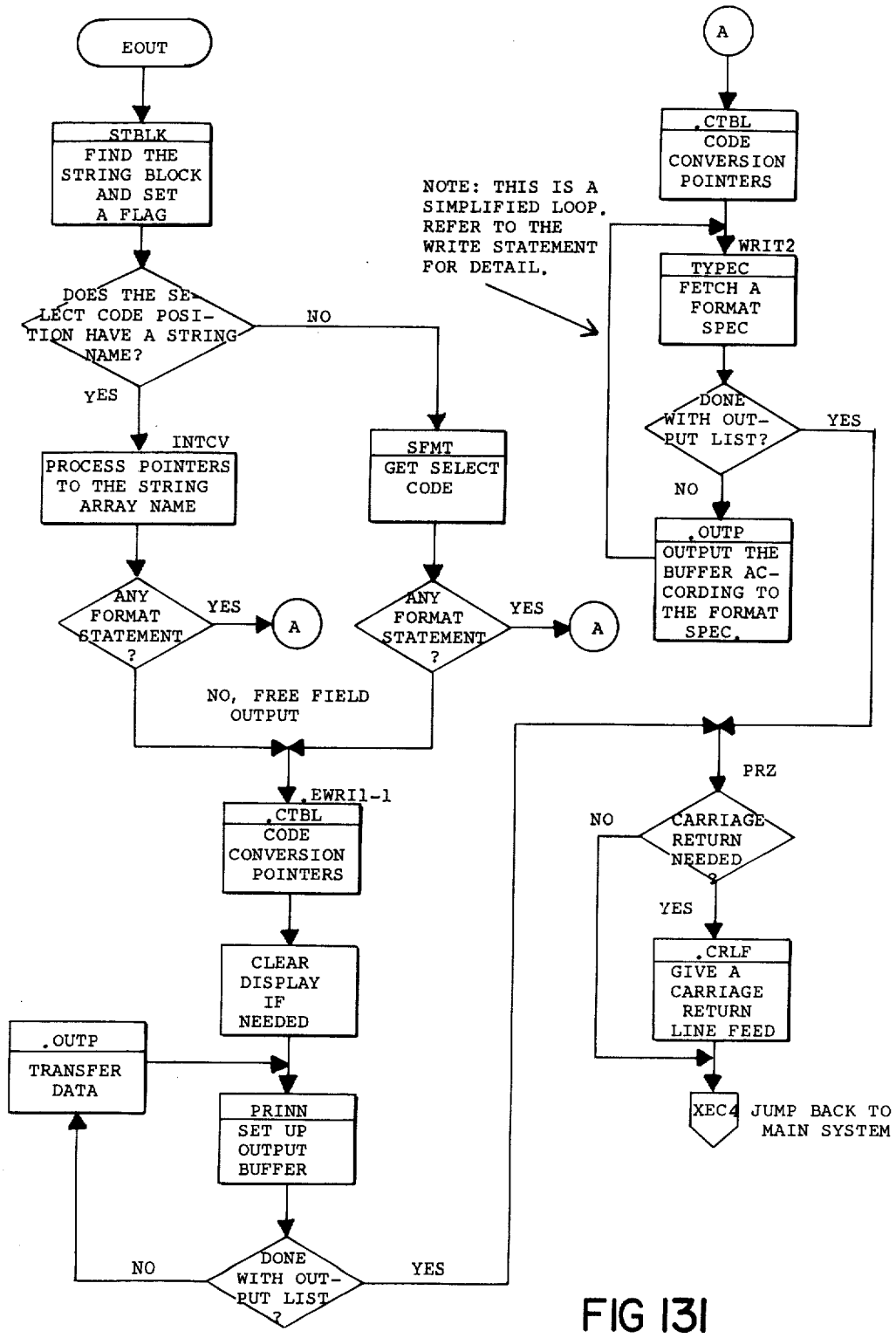


FIG 131

FLOWCHART OF THE I/O BLOCK'S OUTPUT ROUTINE
TO DO INTERNAL CONVERSION

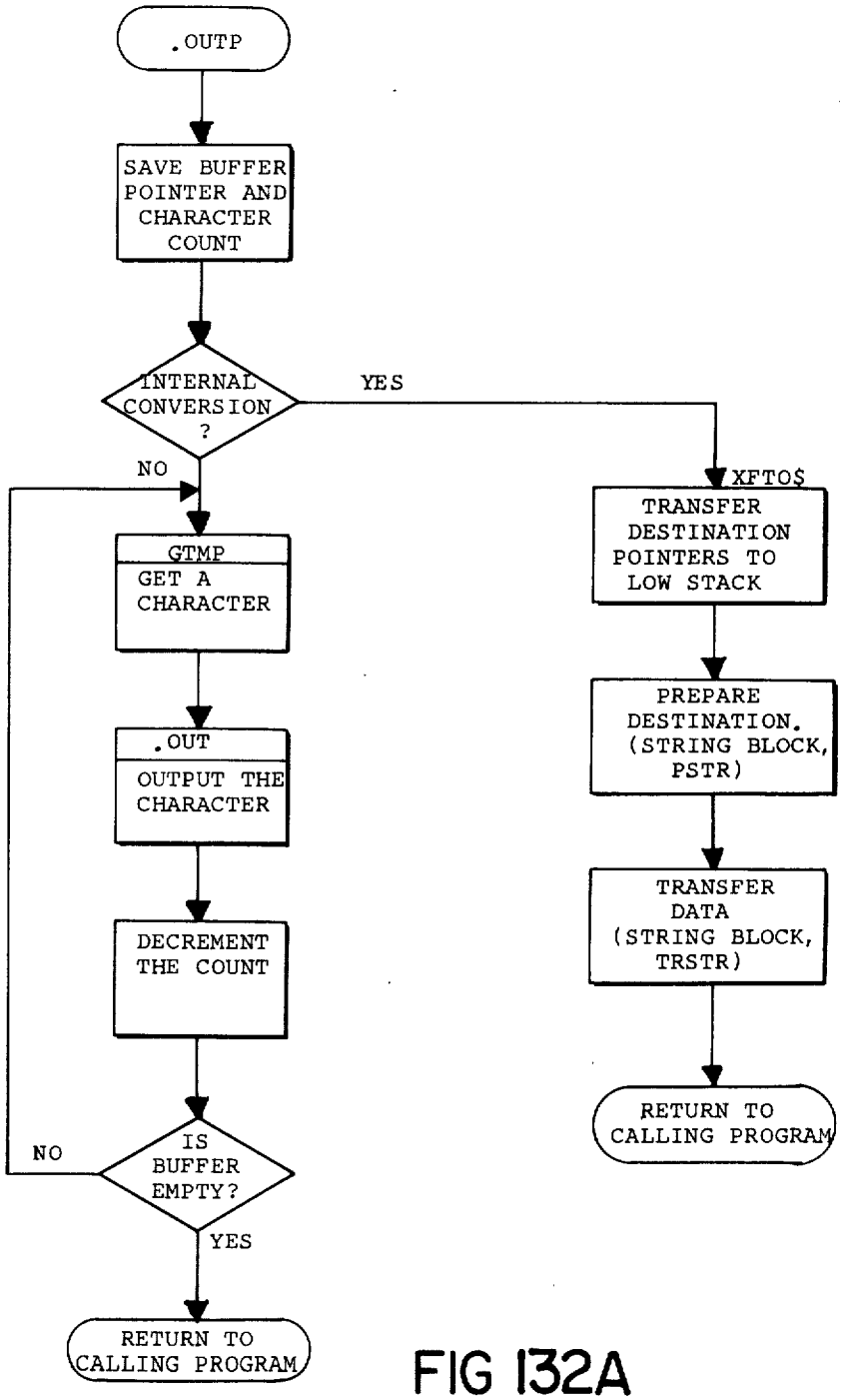


FIG 132A

FLOWCHART OF THE I/O BLOCKS CODE CONVERSION ROUTINES

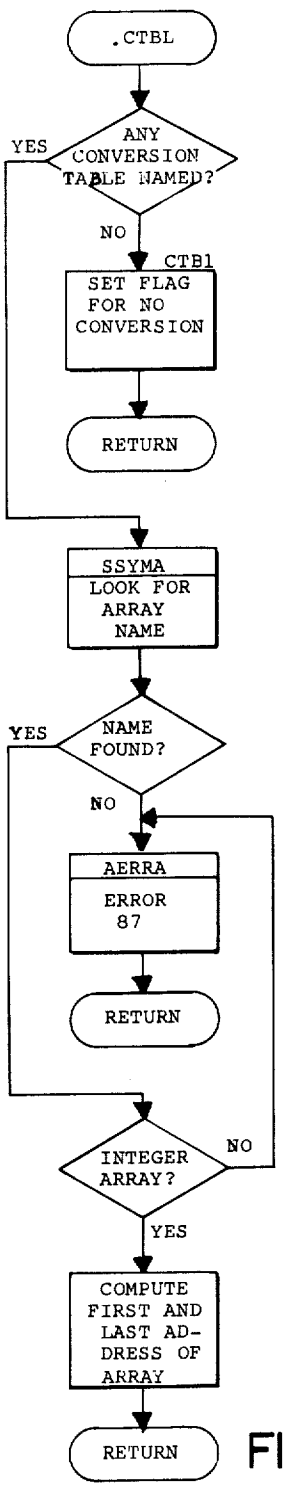


FIG 132B

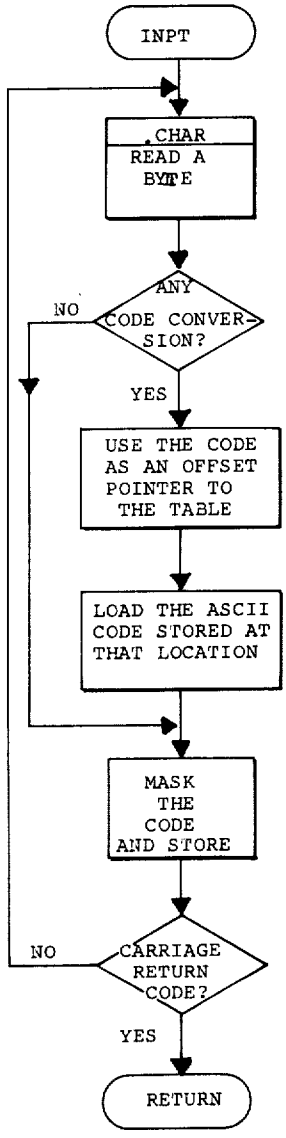


FIG 132C

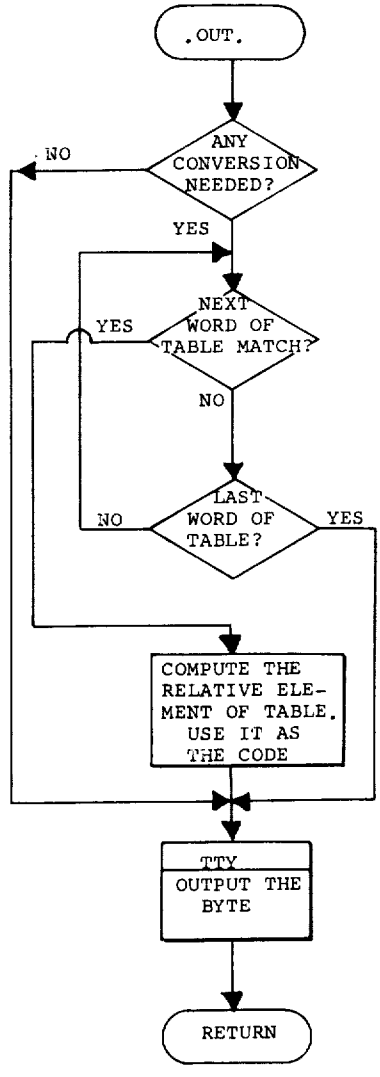


FIG 132D

FLOWCHART OF MISCELLANEOUS FUNCTIONS FOR I/O BLOCK

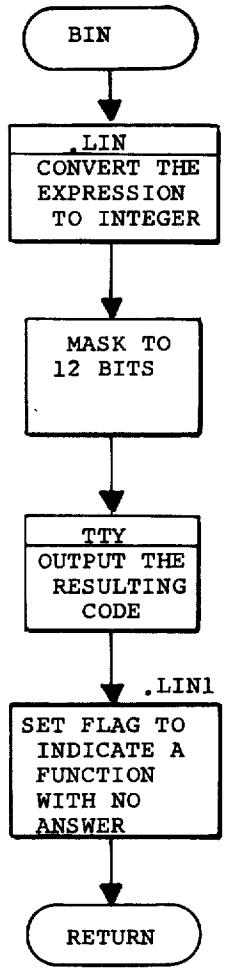


FIG 133

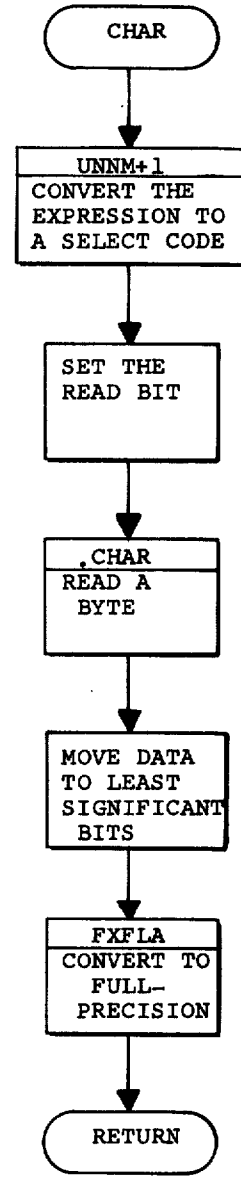


FIG 134

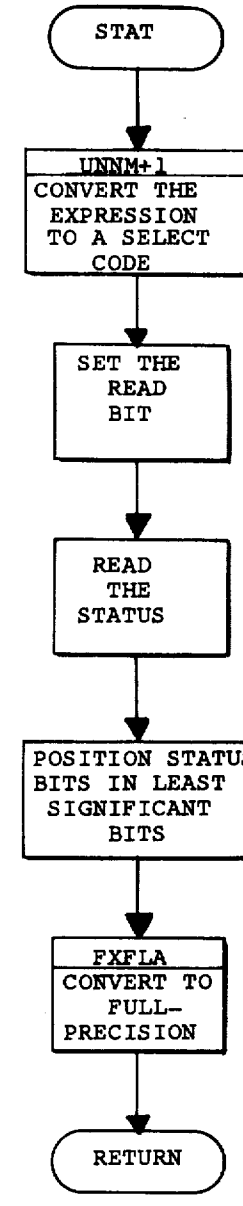


FIG 135

FLOWCHART OF MISCELLANEOUS FUNCTIONS FOR I/O BLOCK

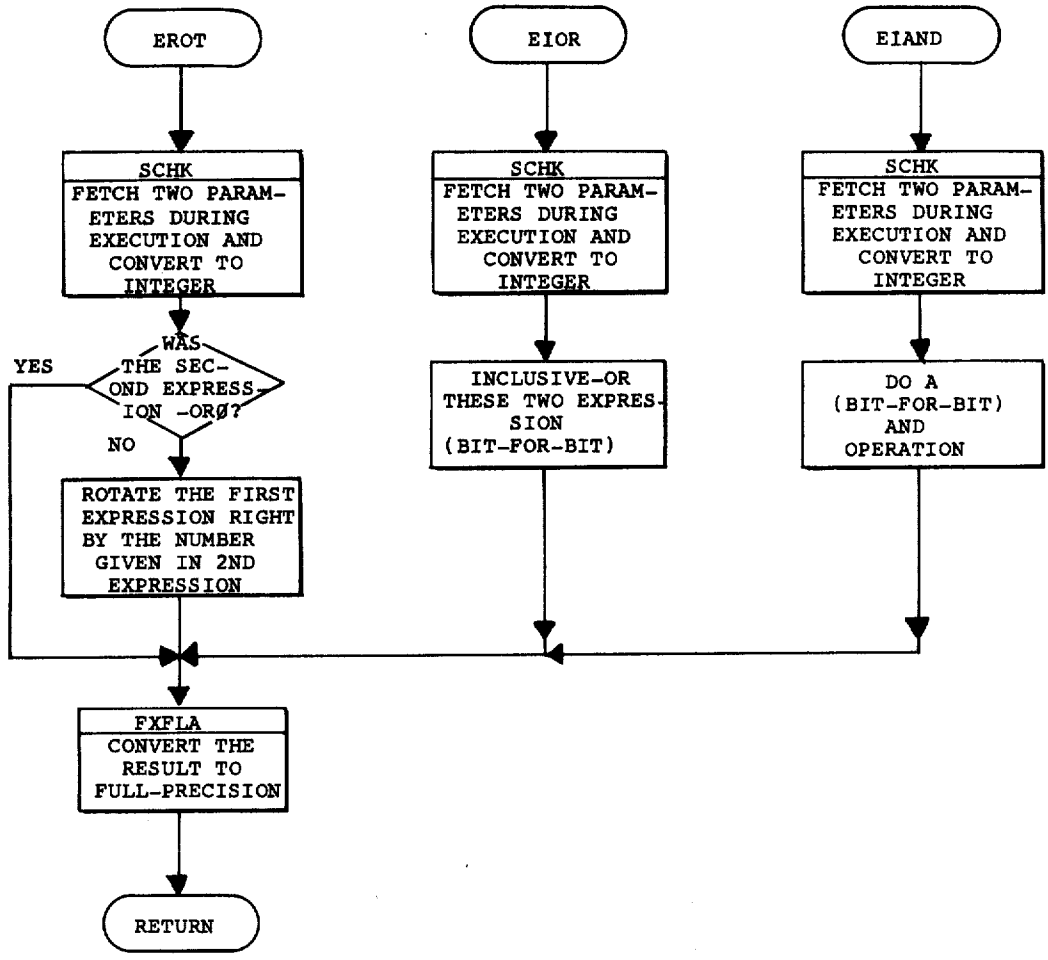


FIG 136

PROGRAMMABLE CALCULATOR

CROSS REFERENCE TO RELATED APPLICATION

This is a continuation of U.S. Patent application Ser. No. 269,899 filed July 7, 1972 and now abandoned.

BACKGROUND OF THE INVENTION

This invention relates generally to calculators and improvements therein and more particularly to programmable calculators that may be controlled both manually from the keyboard input unit and automatically by a stored program loaded into the calculator from the keyboard input unit or an external record member.

Computational problems may be solved manually, with the aid of a calculator (a dedicated computational keyboard-driven machine that may be either programmable or nonprogrammable), or a general purpose computer. Manual solution of computational problems is often very slow, so slow in many cases as to be an impractical, expensive, and ineffective use of the human resource, particularly when there are other alternatives for solution of the computational problems.

Nonprogrammable calculators may be employed to solve many relatively simple computational problems more efficiently than they could be solved by manual methods. However, the keyboard operations or language employed by these calculators is typically trivial in structure, thereby requiring many keyboard operations to solve more general arithmetic problems. Programmable calculators may be employed to solve many additional computational problems at rates hundreds of times faster than manual methods. However, the keyboard language employed by these calculators is also typically relatively simple in structure, thereby again requiring many keyboard operations to solve more general arithmetic problems.

Another basic problem with nearly all of the keyboard languages employed by conventional programmable and nonprogrammable calculators is that they allow the characteristics of the hardware of the calculator to show through to the user. Thus, the user must generally work with data movement at the hardware level, for example, by making sure that data is in certain storage registers before specifying the operations to be performed with that data and by performing other such housekeeping functions. In addition, these languages have been unique to a particular calculator and have not been generally familiar to those persons skilled in the computer and calculator arts.

In the past both programmable and nonprogrammable calculators have generally had very limited memories, thereby severely limiting the size of the computational problems they could be employed to solve. Because of these limitations, the relatively simple structure of the keyboard languages employed by these calculators and the housekeeping requirements associated with their languages have not heretofore been serious shortcomings. However, with advances in technology, the cost of memories has decreased to a point where larger memories could be economically included in programmable calculators. These larger memories have allowed larger and more sophisticated problems to be handled by programmable calculators. As a result the shortcomings of conventional calculator languages have become more critical, thereby creating the need for higher level keyboard languages.

In addition to the foregoing shortcomings, conventional programmable calculators generally have less capability and flexibility than is required to meet the needs of many users. For example, they typically cannot be readily expanded and adapted by the user to increase the amount of program and data storage memory or to perform many special keyboard functions oriented toward the environment of the user.

In some conventional programmable calculators a program stored within the calculator can be recorded onto an external magnetic record member and can later be reloaded back into the calculator from the magnetic record member. However, data and programs stored within these calculators typically cannot be separately recorded onto an external magnetic record member and later separately reloaded back into the calculator therefrom. Moreover, these calculators typically have no provision for making a program secure when it is recorded onto an external magnetic record member. Any user may therefore re-record the program or obtain an indication of the individual program steps once the program is reloaded into the calculator.

Conventional programmable calculators with self-contained output display units typically have little or no alpha capability and typically can only display the contents of one or more selected registers. They are therefore typically unable to display a line containing an alpha-numeric statement or an alphabetic message such as might be used, for example, to inform the user how to run programs with which he may be unfamiliar. Such features would be very helpful to the user both in editing programs and in simplifying their use.

Conventional programmable calculators typically have little or no capability for editing keyboard entries or programs stored within the calculator. For example, they typically have no provision for deleting, replacing, and inserting information included in or omitted from a keyboard entry or internally-stored program on a character-by-character or line-by-line basis. As another example, they typically have no provision for directly recalling any line of an internally-stored program. As a further example, they typically have no provision for automatically accommodating and sequencing program statements which are entered by the user in random order. Such features would be very helpful to the user in editing programs.

Conventional computers typically pose an interfacing problem between the user and the machine. This interface requirement takes the form of a machine-level operator with special abilities for maintaining the software system in operative condition for the user. Computer time sharing systems comprising a centrally located computer and a multiplicity of remotely located user terminals connected thereto by telephone lines have partially solved the user/machine interface problem. However, these systems lack the same flexibility as conventional computers in that they are only programmable and provide no convenient non-programmable method for performing relatively simple calculations. Both types of systems lack provision for editing a program statement from a keyboard without the necessity of retyping the entire statement.

SUMMARY OF THE INVENTION

The principal object of this invention is to provide an improved programmable calculator that has more ca-

pability and flexibility than conventional programmable calculators, that is smaller, less expensive and more efficient in calculating elementary mathematical functions than conventional computer systems, and that is easier to utilize than conventional programmable calculators or computer systems.

Another object of this invention is to provide a programmable calculator employing BASIC computer language implemented in a read-only memory (ROM) that completely eliminates the user/machine interface requirement of conventional computers and further eliminates the necessity of learning a non-universal language such as those typically associated with conventional programmable calculators.

Another object of this invention is to provide a programmable calculator in which the user may, be employing a BACK key or a FORWARD key of a keyboard, position a visual cursor over any character position of a line of information entered from the keyboard or recalled into the display from memory to indicate the character which may be edited by simply actuating a key representing the replacement character.

Another object of this invention is to provide a programmable calculator in which the user may employ an INSERT key of a keyboard to insert a blank character position immediately to the left of the cursor position, and may insert any character therein by actuating its representative key.

Another object of this invention is to provide a programmable calculator in which the user may employ a space bar of a keyboard when entering a line of information from the keyboard to introduce blank characters at desired character positions along the line.

Another object of this invention is to provide a programmable calculator in which the user may delete a character from a line of information entered from the keyboard or recalled into the display from memory by actuating the space bar after positioning a cursor over the character to be deleted.

Another object of this invention is to provide a programmable calculator in which the user may delete a character from a line of information entered from the keyboard or recalled into the display from memory and simultaneously move the right hand portion of that line one character position to the left to occupy the space resulting from the deleted character by positioning the cursor over the character to be deleted and then simultaneously actuating the INSERT key and a SHIFT key.

Another object of this invention is to provide a programmable calculator in which the user may move a displayed line of information either left or right across a display register independently of the location of a cursor at a particular character position thereof.

Another object of this invention is to provide a programmable calculator in which the user may display a statement located at a particular line in memory by actuating a FETCH key of a keyboard followed by the line designation.

Another object of this invention is to provide a programmable calculator in which the user may halt execution of a program, call various lines of information from memory into a display register for visual examination, and thereafter automatically resume program execution at the point at which the halt occurred by actuating a single key of a keyboard unit.

Another object of this invention is to provide a programmable calculator in which the user may call a particular line of information from memory into a display register and may then actuate one of two keys of a keyboard unit to step to and visually observe the contents of lines preceding or following the line originally displayed.

Another object of this invention is to provide a programmable calculator in which the user may store into memory a line of information residing in a display register irrespective of the position of that line in the display register and irrespective of the location of a cursor.

Another object of this invention is to provide a programmable calculator in which the user may, by actuating a RECALL key of a keyboard unit, recall into a display register the most recent line of information which was terminated by either an EXECUTE or END-OF-LINE command.

Another object of this invention is to provide a programmable calculator in which a group of keys of a keyboard unit, representing various editing commands, are provided for conditioning the calculator to perform the command represented without the necessity of literally spelling out the command by actuating a sequence of alphabetic keys.

Another object of this invention is to provide a programmable calculator in which the user may, by actuating a single key of a keyboard unit, delete a statement portion of a line of information being entered into a display register while retaining the line number of that line.

Another object of this invention is to provide a programmable calculator in which the user may, by actuating a single key of a keyboard unit, erase from memory a line of information recalled into a display register from the memory.

Another object of this invention is to provide a programmable calculator in which the user may selectively list on an external printing unit an entire file stored in memory or any portion thereof.

Another object of this invention is to provide a programmable calculator in which the user may designate an automatic line numbering mode for automatically providing a line number, according to a selected sequence involving a starting line number and an increment, at the beginning of each line of information entered into a display register.

Another object of this invention is to provide a programmable calculator in which the user may selectively scratch from memory a mainline program, non-common variables, a symbol table, and the definitions of a group of user-definable keys.

Another object of this invention is to provide a programmable calculator in which the user may, during execution of a program, designate a trace mode of operation whereby the line number of each statement executed is printed on an external output unit as it is executed.

Another object of this invention is to provide a programmable calculator in which the user may, prior to execution of a program and without altering the program, designate a line or lines prior to the execution of which program execution is to be halted.

Another object of this invention is to provide a programmable calculator in which the user may selectively commence execution of a program at either the beginning or at any intermediate line and may designate that at such time all variables are to be either unaltered or set to an undefined state.

Another object of this invention is to provide a programmable calculator in which the user may, by actuat-

ing a single key of a keyboard unit, instruct the calculator to set all program variables to an undefined state and to set up array storage for all arrays previously dimensioned in a program.

Another object of this invention is to provide a programmable calculator in which the user may designate any portion of a program as being secure so that a user may not obtain any indication of the contents of such portion and may not store onto a magnetic record member any secure program that has previously been entered into the calculator from a magnetic record member but may observe the contents of the unsecure portion of any program residing in memory.

Another object of this invention is to provide a programmable calculator in which a group of characters may be associated with each one of a plurality of definable keys of a keyboard unit, may later be displayed upon actuation of the associated key, and, if the group of characters represents an executable command, may be immediately executed upon actuation of the associated key.

Another object of this invention is to provide a programmable calculator in which each of a plurality of definable keys of a keyboard unit may be associated with a single or multiline function of one argument, which function may be employed as part of a program or immediately executed from the keyboard.

Another object of this inventon is to provide a programmable calculator in which each of a plurality of definable keys of a keyboard unit may be associated with a multiplicity of BASIC language program statements and the resulting program executed upon actuation of the associated key.

Another object of this invention is to provide a programmable calculator in which the user may, when employing a matrix plug-in read-only memory module, select as a function the determinant of a previously defined square matrix.

Another object of this invention is to provide a programmable calculator in which the user may, by actuating a SHIFT key of a keyboard unit, selectively cause to be printed on an external printing unit either lower or upper case alphabetic characters.

Another object of this invention is to provide a programmable calculator in which the user may, when the calculator is used in conjunction with an external printing unit, designate a print-all mode for printing all lines of information entered from a keyboard and terminated by EXECUTE or END-OF-LINE commands, all error messages, and all information displayed during execution.

Another object of this invention is to provide a programmable calculator in which the user may immediately execute any self-contained BASIC language program statement from a keyboard unit.

Another object of this invention is to provide a programmable calculator in which plug-in read-only memory modules are not uniquely associated with a particular group of keys of a keyboard unit.

Another object of this invention is to provide a programmable calculator in which functions available through the use of plug-in read-only memory modules are selected by actuating a series of alphanumeric keys of a keyboard unit rather than a single key, thereby allowing the number of functions available to be independent of the number of keys available.

Another object of this invention is to provide a programmable calculator in which the user may create a program using functions available from a plurality of plug-in read-only memory modules inserted into receptacles on the calculator in a particular sequence, may record such program onto an external magnetic record member, and may thereafter execute such program on another calculator of the same type configured with the same plug-in read-only memory modules but without observing the sequence in which they were placed in the receptacles of the first calculator.

Another object of this invention is to provide a programmable calculator in which the user may, when the calculator is configured with a plotter plug-in read-only memory module and an external X-Y plotter, specify an offset of X and Y coordinates to be applied to all subsequent coordinate specifications.

Another object of this invention is to provide a programmable calculator in which the user may, when the calculator is configured with a plotter plug-in read-only memory module, and an external X-Y plotter, incrementally plot, in user units, points relative to a current pen position.

Another object of this invention is to provide a programmable calculator in which the user may, when the calculator is configured with a plotter plug-in read-only memory module and external X-Y plotter, plot rectangular coordinate axes by specifying in user units starting and ending coordinates for each axis, a coordinate at which each axis intersects the other axis, and a tick mark spacing for tick marking along each axis in either direction from a starting point.

Another object of this invention is to provide a programmable calculator in which the user may, when the calculator is configured with a plotter plug-in read-only memory module and an external X-Y plotter, plot labels according to the format of a standard BASIC language print statement or according to a referenced format statement of a program, specify the height of characters contained in such labels, specify a character aspect ratio (ratio of height to width) for the characters contained in such labels, specify an angle of rotation in either degrees, radians or grads at which such labels are to be plotted, and specify a paper height-to-width ratio for assuring character uniformity regardless of plot angle.

Another object of this invention is to provide a programmable calculator in which the user may, when the calculator is configured with a plotter plug-in read-only memory module and an external X-Y plotter, incrementally plot characters according to character size units for facilitating pen placement at the beginning of plotter labelling operations.

Another object of this invention is to provide a programmable calculator in which the user may, when the calculator is configured with a plotter plug-in read-only memory module and an external X-Y plotter, plot characters as the corresponding keys of a keyboard input unit are actuated and use UP, DOWN, LEFT, and RIGHT ARROW keys of the keyboard input unit for controlling the position of the plotter pen.

Another object of this invention is to provide a programmable calculator in which the user may, when the calculator is configured with a terminal plug-in read-only memory module and an external modem for transmitting and receiving information over telephone lines, select from a keyboard unit any baud rate from a continuously variable range of baud rates.

Another object of this invention is to provide a programmable calculator in which the user may, when the

calculator is configured with a terminal plug-in read-only memory module and an external modem for transmitting and receiving data over telephone lines, select by actuating a single key of a keyboard unit either odd or even parity of transmitted data.

Another object of this invention is to provide a programmable calculator in which the user may when the calculator is configured with a terminal plug-in read-only memory module, enter free text information from a keyboard input unit, an external magnetic record member, or any peripheral input unit, edit such information on a line-by-line or character-by-character basis, store such information on an external magnetic record member, and thereafter transmit such information, for example, to a remotely located time-sharing computer system.

Another object of this invention is to provide a programmable calculator in which the user may, when the calculator is configured with a terminal plug-in read-only memory module, generate and transmit any ASCII codes.

Another object of this invention is to provide a programmable calculator in which the user may, when the calculator is configured with a terminal plug-in read-only memory module and an external modem for transmitting and receiving information over telephone lines, make calculations and run programs locally at the same time as, for example, the calculator is on-line with and running programs through a remotely located time-sharing computer system.

Another object of this invention is to provide a programmable calculator in which the user may, when the calculator is configured with a terminal plug-in read-only memory module, receive BASIC language programs from remote locations and thereafter run them locally on the calculator.

Another object of this invention is to provide a programmable calculator in which the user may, when the calculator is configured with an extended input-output plug-in read-only memory module, from a keyboard unit or under program control, generate and transmit to any of a plurality of input-output channels any twelve-bit code.

Another object of this invention is to provide a programmable calculator in which the user may, when the calculator is configured with an extended input-output plug-in read-only memory module, from a keyboard unit or under program control, read an eight-bit character from a designated input-output channel.

Another object of this invention is to provide a programmable calculator in which the user may, when the calculator is configured with an extended input-output plug-in read-only memory module, from a keyboard unit or under program control, read the status of a designated external input-output unit.

Another object of this invention is to provide a programmable calculator in which the user may, when the calculator is configured with an extended input-output plug-in read-only memory module, from a keyboard unit or under program control, perform bit manipulation on sixteen-bit integer data according to the functions of ROTATE, AND, and OR.

Another object of this invention is to provide a programmable calculator in which the user may, when the calculator is configured with an extended input-output plug-in read-only memory module, from a keyboard unit or under program control, convert between various multi-bit data codes and the ASCII code of the calculator for allowing the calculator to communicate with peripheral units having operating codes other than ASCII.

Another object of this invention is to provide a programmable calculator in which the user may, when the calculator is configured with an extended input-output plug-in read-only memory module, from a keyboard unit or under program control, write into a string variable any information which may be transmitted to an output unit.

Another object of this invention is to provide a programmable calculator in which the user may, from a keyboard or under program control, by means of a single command which may involve parameters, mark a magnetic tape, stored either on an internal tape cassette or on one of a plurality of external tape cassettes, into a designated number of files each being of a designated length.

Another object of this invention is to provide a programmable calculator in which the user may designate that a particular file stored on a magnetic tape cassette is to be used for either program storage or data storage and in which he is subsequently prevented from attempting to access data from a file designated for program storage and vice versa.

Another object of this invention is to provide a programmable calculator in which the user may, from a keyboard or under program control, access a file stored on a magnetic tape cassette, by means of a single command indicating a file number, which file number may be the result of an arithmetic expression.

Another object of this invention is to provide a programmable calculator in which only complete files of a designated length may be marked on a magnetic tape, thus preventing the existence of a partially complete file near the end of a tape because of an insufficient amount of tape.

Another object of this invention is to provide a programmable calculator in which the user may, by means of a single command entered either from a keyboard or encountered under program control, list on an external printing unit certain information with respect to each file stored on a magnetic tape cassette, such information being the file number, the amount of information currently stored therein, the file type, the maximum file length, the starting and ending line numbers of a program stored therein, the number of words of common data storage associated with a program stored therein, and the form of data stored therein.

Another object of this invention is to provide a programmable calculator in which the user may, from a keyboard unit or under program control, chain one or more program segments stored on a magnetic tape cassette to a program residing in the calculator memory and may use the same variables in each segment without declaring common storage for the variables.

Another object of this invention is to provide a programmable calculator in which the user may, from a keyboard unit or under program control, insert a program or a program segment stored on a magnetic tape cassette at any line number of a program residing in the calculator memory.

Another object of this invention is to provide a programmable calculator in which the user may specify a line number in memory at which a program or a program segment loaded from a magnetic tape cassette is to begin and in which, if the resulting line numbers differ from those of the program or program segment as

it resided on tape, references within such program or program segment are automatically modified to reflect the current line number sequence.

Another object of this invention is to provide a programmable calculator in which the user may, from a keyboard unit or under program control, selectively store any portion of a program residing in the calculator memory onto a magnetic tape cassette.

Another object of this invention is to provide a programmable calculator in which the user may, from a keyboard unit or under program control, store the definitions associated with all keys of a group of user-definable keys into a file on a magnetic tape cassette.

Another object of this invention is to provide a programmable calculator in which the user may, from a keyboard or under program control, load from a file on a magnetic tape cassette the definitions associated with all keys of a group of user-definable keys.

Another object of this invention is to provide a programmable calculator in which the user may, from a keyboard unit or under program control, load, store, merge, or chain programs or functions residing or to reside on a magnetic tape cassette and associated with or to be associated with any key of a group of user-definable keys.

Another object of this invention is to provide a programmable calculator in which the user may, from a keyboard unit or under program control, load into the calculator memory from a magnetic tape cassette or store onto a magnetic tape cassette from the calculator memory all common data defined in the program residing in the calculator memory without specifying individual variable names and by means of a single command.

Another object of this invention is to provide a programmable calculator in which the user may load prerecorded assembly language programs containing commands, statements, or functions which expand the capabilities of the calculator and which may be selected for execution by the user just as though the same commands, statements, or functions had been made available to the user by means of a plug-in read-only memory module.

Another object of this invention is to provide a programmable calculator in which the user may, from a keyboard unit or under program control, search a magnetic tape cassette in either forward or reverse directions for locating a particular file at the same time as the calculator is executing program statements or keyboard commands.

Other and incidental objects of this invention will become apparent from a reading of this specification and an inspection of the accompanying drawings.

These objects are accomplished according to the illustrated preferred embodiment of this invention by employing a keyboard input unit, a magnetic tape cassette reading and recording unit, a solid state output display unit, an optional external output printer unit, an input-output control unit, a memory unit, and a central processing unit to provide an adaptable programmable calculator having manual operating, automatic operating, program entering, magnetic tape reading, magnetic tape recording, and alphanumeric display and print modes. The keyboard input unit includes a group of data keys for entering numeric data into the calculator, a group of control keys for controlling the various modes and operations of the calculator and the format of the output display, a group of alphanumeric keys arranged as a typewriter keyboard for entering statements, and a group of user-definable keys. All of the data and alphanumeric keys and some of the control keys may also be employed for programming the calculator.

The magnetic tape cassette reading and recording unit includes a reading and recording head, a drive mechanism for driving a magnetic tape past the reading and recording head, and reading and recording drive circuits coupled to the reading and recording head for bidirectionally transferring information between the magnetic tape and the calculator as determined by keyboard commands or commands which are part of a stored program.

The input-output control unit includes a sixteen-bit universal shift register serving as an input-output register into which information may be transferred serially from the central processing unit or in parallel from the keyboard input and magnetic tape cassette reading and recording units and from which information may be transferred serially to the central processing unit or in parallel to the solid state output display, magnetic tape cassette reading and recording, and output printer units. It also includes control logic responsive to the central processing unit for controlling the transfer of information between these units. The input-output control unit may also be employed to perform the same functions between the central processing unit and peripheral units including, for example, an external printing unit, a digitizer, a marked card reader, an X-Y plotter, an external magnetic tape unit, a disc, a typewriter, and a modem. A plurality of peripheral units may be connected at the same time to the input-output control unit by simply plugging interface modules associated with the selected peripheral units into receptacles provided therefore in a rear panel of the calculator housing.

The memory unit includes a modular random-access read-write memory having a dedicated system area and a separate user area for storing program statements and/or data. The user portion of the read-write memory may be expanded without increasing the overall dimensions of the calculator by the addition of a program storage module. Additional read-write memory made vailable to the user is automaticlly accommodated by the calculator, and the user is automatically informed when the storage capacity of the read-write memory has been exceeded.

The memory unit also includes a modular read-only memory in which routines and subroutines of assembly language instructions for performing the various functions of the calculator are stored. These routines and subroutines of the read-only memory may be expanded and adapted by the user to perform additional functions oriented toward the specific needs of the user. This is accomplished by simply plugging additional read-only memory modules into receptacles provided therefore in a side panel of the calculator housing. Added read-only memory modules are automatically accommodated by the calculator and are accessed by the calculator through a series of mnemonic tables. These tables contain mnemonics which are additions to the calculator's programming language.

Plug-in read-only memory modules include, for example, a matrix module, a string variables module, a plotter module, an extended input-output module, and a terminal module. The matrix module makes available to the user standard BASIC language matrix functions

plus an additional function which returns the determinant of a previously defined square matrix. The string variables module makes available to the user standard BASIC language string variables operations. The plotter module enables the user to conveniently plot and label on an external X-Y plotter. The extended input-output module allows the calculator to be used with a wide variety of peripheral input-output units. The terminal module facilitates interfacing the calculator with a modem for communicating, for example, with remotely located time-sharing computer systems. It further allows free text editing and storage.

The memory unit further includes a pair of recirculating sixteen-bit serial shift registers. One of these registers serves as a memory address register for serially receiving information from an arithmetic-logic unit included in the central processing unit, for parallel addressing any memory location designated by the received information back to the arithmetic-logic unit. The other of these registers serves as a memory access register for serially receiving information from the arithmetic-logic unit for writing information in parallel into any addressed memory location, for reading information in parallel from any addressed memory location, and for serially transferring information to the arithmetic-logic unit. It also serves as a four-bit parallel shift register for transferring four bits of binary-coded-decimal information in parallel to the arithmetic-logic unit.

The central processing unit includes four recirculating sixteen-bit serial shift registers, a four-bit serial shift register, the arithmetic-logic unit, a programmable clock, and a microprocessor. Two of these sixteen-bit serial shift registers serve as accumulator registers for serially receiving information from and serially transferring information to the arithmetic-logic unit. The accumulator register employed is designated by a control flip-flop. One of the accumulator registers also serves as a four-bit parallel shift register for receiving four bits of binary-coded-decimal information in parallel from and transferring four bits of such information in parallel to the arithmetic-logic unit. The two remaining sixteen-bit serial shift registers serve as a program counter register and a qualifier register, respectively. They are also employed for serially receiving information from and serially transferring information to the arithmetic-logic unit. The four-bit serial shift register serves as an extend register for serially receiving information from either the memory access register or the arithmetic-logic unit and for serially transferring information to the arithmetic-logic unit.

The arithmetic-logic unit is employed for performing one-bit serial binary arithmetic, four-bit parallel binary-coded-decimal arithmetic, and logic operations. It may also be controlled by the microprocessor to perform bidirectional direct and indirect arithmetic between any of a plurality of the working registers and any of the registers of the read-write memory.

The programmable clock is employed to supply a variable number of shift clock pulses to the arithmetic-logic unit and to the serial shift registers of the input-output, memory, and central processing units. It is also employed to supply clock control signals to the input-output control logic and to the microprocessor.

The microprocessor includes a read-only memory in which a plurlaity of microinstructions and plurality are stored. These microinstructions and codes are employed to perform the basic instructions of the calculator. They include a plurality of coded and non-coded microinstructions for transferring control to the input-output control logic, for controlling the addressing and accessing of the memory unit, and for controlling the operation of the two accumulator registers, the program counter register, the extend register and the arithmetic-logic unit. They also include a plurality of clock codes for controlling the operation of the programmable clock, a plurality of qualifier selection codes for selecting qualifiers and serving as primary address codes for addressing the read-only memory of the microprocessor, and a plurality of secondary address codes for addressing the read-only memory of the microprocessor. In response to a control signal from a power supply provided for the calculator, control signals for the programmable clock, and qualifier control signals from the central processing and input-output control units, the microprocessor issues the microinstructions and codes stored in the read-only memory of the microprocessor as required to process either binary or binary-coded-decimal information entered into or stored in the calculator.

In the keyboard mode, the calculator is controlled by keycodes sequentially entered into the calculator from the keyboard input unit by the user. The solid state output display unit dispays either the alphanumeric representation of the keys as they are depressed or a numeric representation of output data or alphanumeric user instructions or program results. An external output printer unit may be controlled by the user to selectively print a numeric representation of any numeric data entered into the calculator from the keyboard input unit, a numeric representation of any result calculated by the calculator, or a program listing on a line-by-line basis of the statements entered.

When the calculator is in the keyboard mode, it may also be operated in a print-all printing mode. The output printer unit then prints out each program line as it is entered by the user.

In the program running mode, the calculator is controlled by automatically obtaining an internal representation of the program statements stored in the user storage section of the read-write memory. During automatic operation of the calculator, data may be obtained from the memory unit as designated by the program, from the keyboard input unit while the operation of the calculator is stopped for data either by the program or by the user, of from the magnetic tape cassette unit as designated by the program.

When the calculator is in the program running mode, the user may also selectively employ a trace mode to check the execution of the program line-by-line in order to determine whether the program, as entered into the calculator, does in fact carry out the desired sequence of statements.

In the program entering mode, statements are sequentially entered by the user into the calculator from the keyboard input unit and are translated into an internal stored format which consists of a series of operation codes and operand names and are thereafter stored as statements of a program in the user storage section of the readwrite memory.

The magnetic tape cassette reading and recording unit may be employed by the user to separately load either data, BASIC language programs, assembly language programs, or sets of user-definable key definitions into the calculator from an external magnetic tape cassette.

The magnetic tape cassette reading and recording unit may also be employed by the user to separately record either data, BASIC language programs, or sets of user-definable key definitions stored in the user section of the read-write memory onto an external magnetic tape cassette. Programs, or portions thereof, may be coded by the user as being secure when they are recorded onto an external magnetic tape cassette. The calculator detects such programs when they are reloaded into the calculator and prevents the user from re-recording them or obtaining any listing or other indication of the individual program steps contained in the secured portions of such programs.

DESCRIPTION OF THE DRAWINGS

FIG. **1** is a front perspective view of an adaptable programmable calculator according to the preferred embodiment of this invention.

FIG. **2** is a rear perspective view of the adaptable programmable calculator of FIG. **1**.

FIGS. **3A–B** are a simplified block diagram of the adaptable programmable calculator of FIGS. **1** and **2**.

FIGS. **4A–F** are a memory map of the memory unit employed in the adaptable programmable calculator of FIGS. **1–3B**.

FIG. **4'** is a diagram showing the arrangement of FIGS. **4A–F**.

FIGS. **5A–B** are a detailed memory map of the system read-write section of memory as shown in FIG. **4A**.

FIG. **5'** is a diagram showing the arrangement of FIGS. **5A–B**.

FIG. **6** is a detailed memory map of the user read/write section of memory as shown in FIG. **4F**.

FIG. **7** is a simplified operational logic flow chart illustrating the operation of the microprocessor employed in the central processing unit of FIGS. **3A–B**.

FIG **8** is a plan view of the keyboard input unit employed in the adaptable programmable calculator of FIGS. **1–3B**.

FIG. **9** is an overall firmware block diagram for the adaptable programmable calculator.

FIGS. **10A–C** are flow charts of floating point add and subtract routines selectable by the execution monitor of FIG. **9**.

FIG. **11** is a flow chart of a floating point multiply routine selectable by the execution monitor of FIG. **9**.

FIGS. **12A–B** are a flow chart of a floating point divide routine selectable by the execution monitor of FIG. **9**.

FIGS. **13A–C** are a flow chart of a floating point square root routine selectable by the execution monitor of FIG. **9**.

FIG. **14** is a flow chart of a store routine selectable by the execution monitor of FIG. **9**.

FIG. **15** is a flow chart of a rounding routine employed in connection with several of the routines selectable by the execution monitor of FIG. **9**.

FIGS. **16A–B** are a flow chart of a tangent X routine selectable by the execution monitor of FIG. **9**.

FIGS. **17A–B** are a flow chart of an arctangent X routine selectable by the execution monitor of FIG. **9**.

FIGS. **18A–B** are a flow chart of an e^X routine selectable by the execution monitor of FIG. **9**.

FIG. **19** is a flow chart of a natural logarithm X routine selectable by the execution monitor of FIG. **9**.

FIG. **20** is a flow chart of a subroutine employed by the tangent X and e^X routines of FIGS. **16A–B** and **18A–B**, respectively.

FIGS. **21A–B** are a flow chart of a subroutine employed by the tangent X and arctangent X routines of FIGS. **16A–B** and **17A–B**, respectively.

FIGS. **22A–B** are a flow chart of a subroutine employed by the e^X and natural logarithm X routines of FIGS. **18A–B** and **19**, respectively.

FIG. **23** is a flow chart of a subroutine employed by the arctangent X and natural logarithm X routines of FIGS. **18A–B** and **19**, respectively.

FIG. **24** is a flow chart of sine and cosine routines selectable by the execution monitor of FIG. **9**.

FIG. **25** is a flow chart of an X^Y power routine selectable by the execution monitor of FIG. **9**.

FIG. **26** is a flow chart of a logarithm to the base ten routine selectable by the execution monitor of FIG. **9**.

FIG. **27** is a block diagram of the microprocessor of FIGS. **3A–B**.

FIGS. **28A–D** are a detailed schematic diagram of the microprocessor of FIGS. **3A–B** and **27**.

FIG. **28'** is a diagram showing the arrangement of FIGS. **28A–D**.

FIGS. **29A–H** are detailed flow charts illustrating the operation of the microprocessor of FIGS. **3A–B**, **27**, and **28A–D**.

FIGS. **29'** and **29''** are diagrams showing the arrangement of FIGS. **29A-H**.

FIG. **30** is a block diagram of the programmable clock of FIGS. **3A–B**.

FIGS. **31A–C** are a detailed schematic diagram of the programmable clock of FIGS. **3A-B** and **30** and of a portion of the input-output control unit of FIGS. **3A–B**.

FIG. **31'** is a diagram showing the arrangement of FIGS. **31A–C**.

FIG. **32** is a waveform diagram illustrating the operation of the programmable clock of FIGS. **3A–B**, **30**, and **31A–C**.

FIGS. **33A–D** are a detailed schematic diagram of the shift register and arithmetic logic units of FIGS. **3A–B**.

FIG. **33'** is a diagram showing the arrangement of FIGS. **33A–D**.

FIG. **34** is a block diagram of the arithmetic logic unit of FIGS. **3A–B**.

FIG. **35** is a block diagram of the memory unit of FIGS. **3A–B**.

FIGS. **36A–B** are a schematic diagram of the read-write memory of FIGS. **3A–B**, **4A–F**, and **35**.

FIG. **36'** is a diagram showing the arrangement of FIGS. **36A–B**.

FIGS. **37A–B** are a schematic diagram of the optional add-on read-write memory of FIGS. **3A–B**, **4A–F**, and **35**.

FIG. **37'** is a diagram showing the arrangement of FIGS. **37A–B**.

FIG. **38** is a schematic diagram of the basic read-only memory of FIGS. **3A–B**, **4A–F**, and **35**.

FIG. **39** is a sehematic diagram of the optional add-on read-only memory modules of FIG. **35** that may be plugged into the calculator to increase the number of functions available to the user.

FIG. **40** is a detailed schematic diagram of the buffer circuitry associated with the read-only memory modules of FIG. **39**.

FIG. **41** is a block diagram of one of the read-only memory chips of FIGS. **38** and **39**.

FIGS. **42A–D** are a schematic diagram of one of the read-only memory chips of FIGS. **38** and **39**.

FIG. **42'** is a diagram showing the arrangement of FIGS. **42A–D**.

FIG. 43 is a memory map of the memory unit of FIGS. 3A–B and 4A–F illustrating how it is partitioned into the read-only and read-write memory chips of FIGS. 36A–B through 42A–D.

FIG. 44 is a flow chart illustrating how the row members of the lists stored in the read-only memory chips are computed.

FIG. 45 is a table of bit numbers and actual bits used in connection with the flow chart of FIG. 44.

FIGS. 46A–B are a detailed schematic diagram of the memory address register of FIGS. 3A–B and 35 with its associated control circuitry.

FIG. 46′ is a diagram showing the arrangement of FIGS. 46A–B.

FIGS. 47A–B are a waveform diagram and state sequence charts illustrating the operation of the control circuitry of FIGS. 46A–B.

FIG. 47′ is a diagram showing the arrangement of FIGS. 47A–B.

FIGS. 48A–B are a detailed schematic diagram of the memory access register of FIGS. 3A–B and 35.

FIG. 48′ is a diagram showing the arrangement of FIGS. 48A–B.

FIGS. 49A–D are a detailed schematic diagram of the input-output register and gating control circuits employed in the input-output control unit of FIGS. 3A–B.

FIG. 49′ is a diagram showing the arrangement of FIGS. 49A–D.

FIG. 50 is a schematic diagram of the source and relationship of the input-output party lines connected to the peripheral interface module receiving receptacles of FIG. 20.

FIG. 51 is a waveform diagram illustrating the operation of the control section of the input-output control unit of FIGS. 3A–B and 32.

FIG. 52 is a flow chart illustrating the operation of the control section of the input-output control unit of FIGS. 3A–B and 31A–C.

FIG. 53 is a diagram showing how a one-of-ten decoder is employed to address peripheral input-output units to the calculator through the input-output control unit of FIGS. 3A–B.

FIG. 54 is a waveform diagram of some of the input signals employed by the input-output control unit and associated interface modules of FIGS. 3A–B.

FIG. 55 is a waveform diagram of some of the output signals employed by the input-output control until and associated interface modules of FIGS. 3A–B.

FIG. 56 is a waveform diagram of some of the high speed input signals employed by the input-output control unit and associated interface modules of FIGS. 3A–B.

FIG. 57 is a waveform diagram of some of the high speed output signals employed by the input-output control unit and associated interface modules of FIGS. 3A–B.

FIG. 58 is a waveform diagram illustrating the operation of the interrupt mode of operation of the input-output control unit of FIGS. 3A–B.

FIG. 59 is a schematic diagram of logic that may be used to interface an X-Y plotter to the input-output control unit of FIGS. 3A–B.

FIG. 60 is a schematic diagram of logic that may be used to interface a printing unit to the input-output control print of FIGS. 3A–B.

FIG. 61 is a schematic diagram of logic that may be used to interface a modem to the input-output control unit of FIGS. 3A–B.

FIG. 62 is a schematic diagram of logic that may be used to transfer any eight-bit code into or out of the input-output control unit of FIGS. 3A–B.

FIG. 63 is a detailed schematic diagram of the keyboard input unit employed in the adaptable programmable calculator of FIGS. 1–3B.

FIG. 64 is a block diagram of the magnetic tape cassette reading and recording unit employed in the calculator of FIGS. 1—3B.

FIG. 65 is a detailed schematic diagram of the interface block of FIG. 64.

FIGS. 66A–B are a detailed schematic diagram of the control logic block of FIG. 64.

FIG. 66′ is a diagram showing the arrangement of FIGS. 66A–B.

FIGS. 67A–B are a detailed schematic diagram of the read-write block of FIG. 64.

FIG. 67′ is a diagram showing the arrangement of FIGS. 67A–B.

FIGS. 68A–B are a detailed schematic diagram of the motor control block of FIG. 64.

FIG. 68′ is a diagram showing the arrangement of FIGS. 68A–B.

FIG. 69 is a detailed schematic diagram of the interconnect block of FIG. 64.

FIG. 70 is a detailed schematic diagram of the head driver and preamp blocks of FIG. 64.

FIG. 71 is a block diagram illustrating how the magnetic tape cassette reading and recording unit of FIGS. 64–70 interacts with the calculator of FIGS. 1–3B.

FIGS. 72A–B are a detailed schematic diagram of the output display unit employed in the adaptable programmable calculator of FIGS. 1–3B.

FIG. 72′ is a diagram showing the arrangement of FIGS. 72A–B.

FIGS. 73A–B are a detailed schematic diagram of the control logic circuit associated with the output display unit of FIGS. 72A–B.

FIG. 73′ is a diagram showing the arrangement of FIGS. 73A–B.

FIG. 74 is a block diagram of the power supply system employed in the adaptable programmable calculator of FIGS. 1–3B.

FIGS. 75A–B are a detailed schematic diagram of the power supply system of FIG. 74.

FIG. 75′ is a diagram showing the arrangement of FIGS. 75A–B.

FIGS. 76A–B are a block diagram of an interface module that may be employed to interface a typewriter to the adaptable programmable calculator of FIG. 1–B.

FIG. 76′ is a diagram showing the arrangement of FIGS. 76A–B.

FIG. 77A–B are a flow chart of an input-output routine performed when a typewriter is employed with the programmable calculator of FIGS. 1–3B.

FIG. 77′ is a diagram showing the arrangement of FIGS. 77A–B.

FIGS. 78A–D are a detailed schematic diagram of the control logic block of FIGS. 76A–B.

FIG. 78′ is a diagram showing the arrangement of FIGS. 78A–D.

FIGS. 79A–B are a detailed schematic diagram of the power gates of FIGS. 76A–B.

FIG. 79′ is a diagram showing the arrangement of FIGS. 79A–B.

FIGS. 80A–B are a simplified logic diagram showing state qualifiers and instructions relating to the flow chart of FIGS. 77A–B.

FIG. 81 is a detailed schematic diagram of the ROM, data latch, and compare circuitry of FIGS. 76A–B.

FIG. 82 is a detailed schematic diagram of a power supply that may be employed to power the typewriter interface circuitry of FIGS. 78A–D, 79A–B, and 81.

FIG. 83 is a flow chart of the turn-on routine which is a portion of the system monitor of FIG. 4B.

FIGS. 84A–D are a flow chart of a routine comprising another portion of the system monitor of FIG. 4B.

FIG. 85 is a flow chart of the system table scan routine of FIG. 4D.

FIG 86 is a flow chart of a subroutine called by the routine of FIG. 85.

FIG. 87 is a flow chart of the table search routine of FIG. 4A.

FIGS. 88A–G are a flow chart of the keyboard input routine and special keyboard functions routines of FIG. 4A.

FIGS. 89A–B are flow charts of subroutines called by the routines of FIGS. 88A–G.

FIG. 90 is a flow chart of the clear subroutine called by various ones of the routines of FIGS. 4A–F.

FIG. 91 is a flow chart of a subroutine called by the routine of FIG. 88F.

FIG. 92 is a flow chart of a subroutine called by the keyboard input routine of FIGS. 4A and 88A–G.

FIG. 93 is a flow chart of a keyboard driver subroutine called by the subroutine of FIG. 92.

FIGS. 94A–F are flow charts of some of the general system subroutines of FIG. 4A.

FIG. 95 is a flow chart of the error routine of FIG. 4B.

FIGS. 96A–D are flow charts of the program memory manager routines of FIG. 4B.

FIG. 97 is a flow chart of another of the general system subroutines of FIG. 4A which is also called by the routines of FIGS. 96A–D.

FIG. 98 is a flow chart of another of the general system subroutines of FIG. 4A.

FIGS. 99A–F are flow charts of the user-definable key routines of FIG. 4A.

FIGS. 100A–D are flow charts of the execution monitor and keyboard execution control blocks of FIG. 4C.

FIG. 101 is a flow chart of the load execution routine of FIG. 4D.

FIG. 102 is a flow chart of the store execution routine of FIG. 4D.

FIG. 103 is a flow chart of the merge execution routine of FIG. 4D.

FIG. 104 is a flow chart of the link execution routine of FIG. 4D.

FIG. 105 is a flow chart of the find execution routine of FIG. 4D.

FIGS. 106A–D are flow charts of subroutines called by the execution routines of FIGS. 101–104.

FIG. 107 is a flow chart of the load data execution routine of FIG. 4D.

FIG. 108 is a flow chart of the store data execution routine of FIG. 4D.

FIG. 109 is a flow chart of a subroutine called by the routines of FIGS. 107 and 108.

FIG. 110 is a flow chart of the load key execution routine of FIG. 4D.

FIG. 111 is a flow chart of the store key execution routine of FIG. 4D.

FIG. 112 is a flow chart of the load bin execution routine of FIG. 4D.

FIG. 113 is a flow chart of the mark execution routine of FIG. 4D.

FIG. 114 is a flow chart of a subroutine called by the routine of FIG. 113.

FIG. 115A is a flow chart of the list execution routine of FIG. 4D.

FIG. 115B is a flow chart of a subroutine called by the routine of FIG. 115A.

FIG. 116 is a flow chart of the secure routine of FIG. 4D.

FIGS. 117A–B are a flow chart of the interrupt routine of FIG. 4D.

FIGS. 118A–B are a flow chart of a routine for performing an OFFSET command selectable when the plotter plug-in read-only memory module is employed with the calculator.

FIG. 119 is a flow chart of a routine for performing an IPLOT command selectable when the plotter plug-in read-only memory module is employed with the calculator.

FIG. 120 is a flow chart of a routine for performing a LABEL command selectable when the plotter plug-in read-only memory module is employed with the calculator.

FIG. 121 is a flow chart of a routine for performing a LETTER command selectable when the plotter plug-in read-only memory module is employed with the calculator.

FIG. 122 is a flow chart of a routine for performing a CPLOT command selectable when the plotter plug-in read-only memory module is employed with the calculator.

FIG. 123 is a flow chart of a subroutine employed by the routine of FIG. 122.

FIG. 124 is a flow chart of a routine for performing XAXIS and YAXIS commands selectable when the plotter plug-in read-only memory module is employed with the calculator.

FIGS. 125A–C are flow charts of subroutines called by the routine of FIG. 124.

FIGS. 126A–C are flow charts of subroutines called by the routines of FIGS. 118–124.

FIG. 127 is a flow chart of a primary entry routine employed when the terminal read-only memory module is plugged into the calculator.

FIGS. 128A–W are a flow chart of a keyboard input and editing routine employed when the terminal read-only memory module is plugged into the calculator.

FIGS. 129A–B are a flow chart of a modem interrupt service routine employed when the terminal read-only memory module is plugged into the calculator.

FIG. 130 is a flow chart of a secondary entry routine employed when the terminal read-only memory module is plugged into the calculator.

FIG. 131 is a flow chart of an output statment routine employed when the extended input-output read-only memory module is plugged into the calculator.

FIGS. 132A–D are flow charts of subroutines called by the routine of FIG. 131.

FIG. 133 is a flow chart of a BIN command selectable when the extended input-output plug-in read-only memory module is employed with the calculator.

FIG. 134 is a flow chart of a CHAR command selectable when the extended input-output plug-in read-only memory module is employed with the calculator.

FIG. 135 is a flow chart of a STAT command selectable when the extended input-output plug-in read-only memory module is employed with the calculator.

FIG. 136 is a flow chart of binary ROTATE, AND, and OR commands selectable when the extended input-output plug-in read-only memory module is employed with the calculator.

DESCRIPTION OF THE PREFERRED EMBODIMENT

GENERAL DESCRIPTION

Referring to FIGS. 1 and 2, there is shown an adaptable programmmable calculator 10 including both a keyboard input unit 12 for entering information into and controlling the operation of the calculator and the magnetic tape cassette reading and recording unit 14 for recording information stored within the calculator onto one or more external tape cassettes 16 and for subsequently loading the information recorded on these and other similar magnetic tape cassettes back into the calculator. The calculator also includes a solid state output display unit 18 for displaying alphameric information stored within the calculator. All of these input and output units are mounted within a single calculator housing 24 adjacent to a curved front panel 26 thereof.

As shown in FIG. 2, a plurality of peripheral input and output units including, for example, a line printer, a digitizer, a marked card reader, an X-Y plotter, a typewriter, a teletypewriter, an extended read-write memory unit, a magnetic disc reading and recording unit, and a modem for connecting the calculator via telephone lines to a remotely located computer, may be connected to the calculator at the same time by simply inserting interface modules 30 associated with the selected peripheral units into any of four receptacles 32 provided therefor in a rear panel 34 of the calculator housing. As each interface module 30 is inserted into one of these receptacles, a spring-loaded door 38 at the entrance of the receptacle swings down allowing passage of the interface module. Once the interface module is fully inserted, a printed-circuit terminal board 40 contained within the interface module plugs into a mating edge connector mounted inside the calculator. If any of the selected peripheral units require AC line power, their power cords may be plugged into either of two AC power outlets 42 provided therefor at the rear panel of calculator housing 24.

Referring to the simplified block diagram shown in FIGS. 3A–B, it may be seen that the calculator also includes an input-output control unit 44 (hereinafter referred to as the I/O control unit) for controlling the transfer of information to and from the input and output units, a memory unit 46 for storing and manipulating information entered into the calculator and for storing routines and subroutines of basic instructions performed by the calculator, and a central processing unit 48 (hereinafter referred to as the CPU) for controlling the execution of the routines and subroutines of basic instructions stored in the memory unit as required to process information entered into or stored within the calculator. The calculator also includes a bus system comprising an $\overline{S}$-bus 50, a T-bus 52, and an $\overline{R}$-bus 54 for transferring information from the memory and I/O control units to the CPU, from the CPU to the memory and I/O control units, and between different portions of the CPU. It further comprises a power supply for supplying DC power to the calculator and peripheral units employed therewith and for issuing a control signal $\overline{POP}$ when power is supplied to the calculator.

The I/O control unit 44 includes an input-output register 56 (hereinafter referred to as the I/O register), associated I/O gating control circuitry 58, and input-output control logic 60 (hereinafter referred to as the I/O control), I/O register 56 comprises a universal sixteen-bit shift register into which information may be transferred either bit-serially from CPU 48 via T-bus 52 or in parallel from keyboard input unit 12, magnetic tape cassette reading and recording unit 14, and peripheral input units 28 such as the marked card reader via twelve input party lines 62. Information may be transferred from I/O register 46 either bit-serially to CPU 48 via $\overline{S}$-bus 50 or in parallel to magnetic tape cassette reading and recording unit 14, solid state output display unit 18, output printer unit 20, and peripheral output units 28 such as the X-Y plotter or the typewriter via sixteen output party lines 64.

I/O gating control circuitry 58 includes control circuits for controlling the transfer of information into and out of I/O register 56 in response to selected I/O qualifier control signals from CPU 48 and selected I/O control instructions from I/O control 60. It also includes an interrupt control circuit 65, a peripheral control circuit 66, a printer control circuit 68, and a display control circuit 69 for variously controlling the input and output units and issuing control signals QFG and EBT to I/O control 60 via two output lines 71 and 72. These last mentioned control circuits variously perfrom their control functions in response to control signal $\overline{POP}$ from the power supply, I/O qualifier control signals from CPU 48, I/O control instructions from I/O control 60, and control signals from keyboard input unit 12. Interrupt control circuit 65 initiates the transfer of information into I/O register 56 from keyboard input unit 12 or interrupting peripheral input units 28 such as the marked card reader and issues a qualifier control signal QNR to CPU 48 via output lines 73. Peripheral control circuit 66 enables interface modules 30 plugged into the calculator to respond to information from I/O register 56, control associated peripheral units 28, transfer information to and/or receive information from associated peripheral units 28, and in some cases intiate the transfer of information to I/O register 56 from the interface modules themselves. Printer control circuit 68 and display control circuit 69 enable output display unit 18, and output printer unit 20, respectively, to respond to information from I/O register 56.

When a basic I/O instruction obtained from memory unit 46 is to be executed, CPU 48 transfers control to I/O control 60 by issuing a pair of I/O microinstructions $\overline{PTR}$ and $\overline{XTR}$ thereto. In response to these I/O microinstructions from CPU 48, control signal $\overline{POP}$ from the power supply, control signals QFG and EBT from I/O gating control circuitry 58, and I/O qualifier and clock control signals from CPU 48, I/O control 60 selectively issues one or more I/O control instructions to gating control circuitry 58 as required to execute the basic I/O instructions designated by CPU 48 and issues control signals, $\overline{TTX}$, $\overline{XTR}$, QRD, and SCT to CPU 48 via output lines 74-77. The I/O qualifier control signals issued to I/O control 60 and gating control circuitry 58 by CPU 48 are derived from the basic I/O instruction to be executed. Those qualifier control signals issued to I/O control 60 designate the specific I/O control instructions to be issued by I/O control 60, while those issued to gating control circuitry 58 designate selected

control circuits to be employed in executing the basic I/O instruction.

Memory unit **46** includes a modular random-access read-write memory **78** (hereinafter referred to as the RWM), a modular read-only memory **80** (hereinafter referred to as the ROM), a memory address register **82** (hereinafter referred to as the M-register), a memory access register **84** (hereinafter referred to as the T-register), and control circuitry **85** for these memories and registers. The RWM **78** and ROM **80** comprise MOS-type semiconductor memories. As shown in the memory map of FIGS. **4A–F** the basic RWM **78** contains a dedicated system storage section of 256 sixteen-bit words extending from address **1400** to address **1777** and a separate user program and/or data storage section of 1792 sixteen-bit words extending from address **40400** to address **43777**. All addresses on the memory map are represented in octal form.

An optional 2048 sixteen-bit words of RWM may be made available to the user at address **44000** to address **47777**. This is accomplished by removing a top panel **90** of the calculator housing shown in FIG. **1**, and inserting an additional printed circuit board containing the optional memory. The additional RWM is automatically accommodated by the calculator.

As shown in the more detailed memory map of FIGS. **5A–B**, the RWM dedicated system storage section includes 52 words (addresses **1400–1463**) containing information in the form of mnemonic variables which is employed by the firmware routines shown in FIG. **9**. A more detailed description of these mnemonic variables is given on page **21** of the calculator basic system firmware listing located elsewhere in this specification. Addresses **1466–1477** and **1701–1737** are used as temporary storage by the various routines shown in FIG. **9**. Addresses **1500–1550** comprise a 41-word buffer used to contain the input characters during syntax analysis. Addresses **1551–1621** comprise a 41-word buffer used by the single line display refresh routine of FIG. **9**. These 41 words along with 42 additional words (addresses **1622–1673**) are used as a syntax buffer during syntax analysis. Four of these words (addresses **1622–1625**) are used as temporary registers by several of the statement execution routines of FIG. **9**. Eight words (addresses **1630–1637**) are used as two temporary floating point number registers by the formula evaluation routines of FIG. **9**. Addresses **1640–1677** comprise 32 words which are used by the statement execution routines of FIG. **9**. Eight words (addresses **1744–1747** and **1754–1757**) are employed as AR1 and AR2 four-word working registers for performing binary-coded-decimal arithmetic. An additional eight words (addresses **1740–1743** and **1750–1753**) are employed as working data registers X_c and Y_c for implementation of the trigonometric functions. These sixteen words (addresses **1740–1757**) are used as temporary storage registers by all the routines of FIG. **9** except the statement execution and formula evaluation routines. A variable-length system subroutine stack (addresses **1760–1772**) is employed for storing return addresses required by programs stored in ROM **80**. Four words (addresses **1773–1776**) are used to store the result of the latest keyboard computation. The last word in the system RWM (address **1777**) is used to store a pointer indicating the next available location for the return address of the next subroutine call within the basic system. A complete assembly language description of the system RWM is included in pages **21–24** of the calculator basic system firmware listing.

As shown in the memory map of FIGS. **4A–F** and the more detailed memory map of FIG. **6**, user program and/or data storage section of RWM **78** contains 1760 words available to the user (as user addresses **40440–43777**) for storing programs and/or data, 20 words dedicated for use by the interrupt routine of FIGS. **9** and **117A–B**, and 12 words available for use by plug-in read-only memory modules. An additional 2048 sixteen-bit words may be available to the user (as user addresses **44000–47777**).

Also, as shown in the memory map of FIGS. **4A–B**, the basic ROM **80** contains 7680 sixteen-bit words extending from address **0000** to address **1377**, form address **2000** to address **16777**, and from address **40000** to address **40377**. Routines and subroutines of basic instructions for performing the basic functions of the calculator and constants employed by thse routines and subroutines are stored in these portions of ROM **80**. An additional 8192 sixteen-bit words of ROM may also be added at addresses **20000–37777** in steps of 512 and 1,024 words. This is accomplished by simply inserting plug-in ROM modules **92** into receptacles provided therefor within the calculator which are accessible through a door in the left panel of the calculator housing as illustrated in FIG. **1**. As each plug-in ROM module **92** is inserted into one of these receptacles a printed circuit terminal board **96** contained within the plug-in ROM module plugs into a mating edge connector mounted inside the calculator. A handle pivotally mounted at the top end of each plug-in ROM module **92** facilitates removal of the plug-in ROM module once it has been fully inserted into one of the receptacles.

Routines and subroutines of basic instructions (and any needed constants) for enabling the calculator to perform many additional functions are stored in each plug-in ROM module **92**. The user himself may therefore quickly and simply adapt the calculator to perform many additional functions oriented toward his specific needs by simply plugging ROM modules of his own choosing into the calculator. Added plug-in ROM modules are automatically accommodated by the calculator.

Referring again to FIGS. **3A–B**, M-register **82** of the memory unit comprises a recirculating sixteen-bit serial shift register into which information may be transferred bit-serially from CPU **48** via T-bus **52** and out of which information may be transferred bit-serially to CPU **48** via $\overline{S}$-bus **50**. Information shifted into M-register **82** may be employed to address any word in RWM **78** or ROM **80** via fifteen output lines **106**.

T-register **84** of the memory unit comprises a recirculating sixteen-bit serial shift register into which information may be transferred either bit-serially from CPU **48** via T-bus **52** or in parallel from any addressed word in RWM **78** and ROM **80** via sixteen parallel input lines **108**. Information may be transferred from T-register **84** either bit-serially to CPU **48** via $\overline{S}$-bus **50** or parallel to any addressed word in RWM **78** via sixteen parallel output lines **110**. The four least significant bits of information contained in T-register **84** may comprise binary-coded-decimal information and may be transferred from the T-register in parallel to CPU **48** via three parallel output lines **112** taken with $\overline{S}$-bus **50**.

The control circuitry **85** of the memory unit controls these transfers of information into and out of M-regis-

ter 82 and T-register 84, controls the addressing and accessing of RWM 78 and ROM 80, and refreshes RWM 78. It performs these functions in response to memory microinstructions, memory clock pulses, and shift clock pluses from CPU 48.

CPU 48 includes a register unit 114, an arithmetic-logic unit 116 (hereinafter referred to as the ALU), a programmable clock 118, and a microprocessor 120. Register unit 114 comprises four recirculating sixteen-bit shift registers 122, 124, 126, and 128 and one four-bit shift register 130. Shift registers 122 and 124 serve as sixteen-bit serial accumulator registers (hereinafter referred to as the A-register and the B-register, respectively) into which information may be transferred bit-serially from ALU 116 via T-bus 52 and out of which information may be transferred bit-serially to ALU 116 via $\overline{R}$-bus 54. The four least significant bit positions of A-register 122 also serve as a four-bit parallel accumulator register into which four bits of binary-coded-decimal information may be transferred in parallel from ALU 116 via four parallel input lines 132 and out of which four bits of binary-coded-decimal information may also be transferred in parallel to ALU 116 via three parallel output lines 134 taken with $\overline{R}$-bus 54.

Shift register 126 serves as a sixteen-bit system program counter (hereinafter referred to as the P-register) into which information may be transferred bit-serially from ALU 116 via T-bus 52 and out of which information may be transferred bit-serially to ALU 116 via $\overline{R}$-bus 54. Information contained in the least significant bit position of P-register 126 may also be transferred as a qualifier control signal QPO to microprocessor 120 via output line 135.

Shift register 128 serves as a sixteen-bit qualifier register (hereinafter referred to as the Q-register) into which information may be transferred bit-serially from ALU 116 via T-bus 52 and out of which information may be transferred bit-serially to ALU 116 via $\overline{R}$-bus 54. Information contained in the five least significant bit positions of Q-register 128 is transferred to I/O gating control circuitry 58 as five one-bit I/O qualifier control signals Q00–Q04 via five parallel output lines 136, and information contained in the six next least significant bit positions of the Q-register is transferred to I/O control 60 as six one-bit I/O qualifier control signals Q05-Q10 via six parallel output lines 138. Similarly, information contained in the seven least significant, the ninth and eleventh least significant, and the most signifcant bit positions of Q-register 128 and information derived from the thirteenth, fourteenth, and fifteenth bit positions of the Q-register may be transferred to microprocessor 120 as eleven one-bit microprocessor qualifier control slignals Q00–Q06, Q08, Q10, Q15, and QMR via eleven output lines 140. Information contained in the twelfth through the fifteenth least significant bit positions of Q-register 128 may be transferred to microprocessor 120 as a four-bit primary address code via four parallel output lines 142.

Shift register 130 serves as a four-bit serial extend register (hereinafter referred to as the E-register) into which information may be transferred bit-serially either from ALU 116 via T-bus 52 or from the least significant bit position of T-register 84 via input line 144. Information may also be transferred out of E-register 130 to ALU 116 via $\overline{R}$-bus 54.

Register unit 114 also includes control circuitry 146 for controlling the transfer of parallel binary-coded-decimal information into and out of A-register 122 and the transfer of serial binary information into and out of A-register 122, B-register 124, P-register 126, Q-register 128, and E-register 130. This is accomplished in response to register microinstructions from microprocessor 120, control signals $\overline{TTX}$ and $\overline{XTR}$ from I/O control 60, and shift clock control pulses from programmable clock 118. Control circuitry 146 includes a flip-flop 148 (hereinafter referred to as the A/B flip-flop) for enabling the transfer of information into and out of either the A-register 122 or the B-register 124 as determined by the state of the A/B flip-flop. The state of A/B flip-flop 148 is initially determined by information Q11 transferred to the A/B flip-flop from the twelfth least significant bit position of Q-register 128 but may be subsequently complemented one or more times by microinstruction CAB from microprocessor 120.

ALU 116 may perform either one-bit serial binary arithmetic on data received from T-register 84 of M-register 82 via $\overline{S}$-bus 50 and/or from any register of register unit 114 via $\overline{R}$-bus 54 or four-bit parallel binary-coded-decimal arithmetic on data received from T-register 84 via output lines 112 taken with $\overline{S}$-bus 50 and/or from A-register 122 via output lines 134 taken with $\overline{R}$-bus 54. It may also perform logic operations on data received from memory unit 46 and/or register unit 114 via any of these lines. The arithmetic and logic operations performed are designated by ALU microinstructions from microprocessor 120 and are carried out in response to these microinstructions, shift clock control pulses from programmable clock 118, and control signal SCB from I/O control 60. Information is also transferred from ALU 116 to A-register 122 via output lines 132 or to I/O register 56, M-register 82, T-register 84, or any register or register unit 114 via T-bus 52 in response to microinstructions and control signals applied to these registers. If a carry results while ALU 116 is performing either one-bit serial binary arithmetic or four-bit parallel binary-coded-decimal arithmetic, the ALU issues a corresponding qualifier control signal QBC or QDC to microprocessor 120 via one of two output lines 152 and 154.

Programmable clock 118 incudes a crystal-controlled system clock 156, a clock decoder and generator 158, and a control gate 160. System clock 156 issues regularly recurring clock pulses to clock decoder and generator 158 via output line 162. In response to these regularly recurring clock pulses from system clock 156 and to four-bit clock codes from microprocessor 120, clock decoder and generator 158 issues trains of n shift clock pulses to ALU 116, M-register 82, T-register 84, and all of the registers of register unit 114 via output line 164. These trains of n shift clock pulses are employed for shifting a corresponding number of bits of serial information into or out of any of these registers or for shifting a carry bit in the ALU. The number n of pulses in each of these trains may vary from one to sixteen as determined by the number of bits of serial information required during each operation to be performed. In response to a control signal $\overline{CCO}$ from microprocessor 120, control gate 160 prevents any shift clock pulses from being applied to the ALU or any of these registers. Upon completion of each train of n shift clock pulses, clock decoder and generator 158 issues a ROM clock pulse to microprocessor 120 via output line 166 and an I/O clock pulse to I/O control 60 via output line 168. In response to the regularly recurring clock signal from system

clock **56**, clock decoder and generator **158** also issues correspondingly regularly recurring memory clock pulses to memory unit **46** via output line **170.**

Microprocessor **120** selectively issues two I/O microinstructions to I/O control **60** via two output lines **172,** six memory microinstructions to memory unit **46** via six output lines **174,** thirteen register microinstructions to register unit **114** via thirteen output lines **176,** and five ALU microinstructions to ALU **116** via five output lines **178.** It also issues a four-bit clock code associated with each of these microinstructions to clock decoder **158** via four output lines **180.** These microinstructions and associated clock codes are issued as determined by the control signal $\overline{POP}$ from the power supply, the eleven microprocessor qualifier control singals from Q-register **128,** the four-bit primary address codes from Q-register **128,** and the five microprocessor qualifier control signals from I/O control **60,** interrupt control **65,** ALU **116,** and P-register **126.**

As shown in the simplified flow chart of FIG. **7,** microprocessor **120** executes a hardware diagnostic routine (stored within the microprocessor itself) in response to the control signal $\overline{POP}$. Upon completion of this diagnostic routine, ALU **116** issues the qualifier control signal QBC indicating whether or not the diagnostic routine was successful. Microprocessor **120** thereupon responds to this qualifier control signal by entering the basic machine operating loop and issuing microinstructions causing a sixteen-bit instruction stored in ROM **80** to be loaded into T-registter **84** and transferred from there to Q-register **128.** Microprocessor **120** thereupon sequentially responds to one or more additional qualifier control signals by issuing microinstructions and associated clock codes for executing the instruction then contained in Q-register **128.** and causing another sixteen-bit instruction stored in ROM **80** to be loaded into T-register **84** and transferred from there to the Q-register. When an instruction requiring multiple branching is contained in Q-register **128,** microprocessor **120** issues a pair of microinstructions $\overline{UTR}$ and $\overline{XTR}$ causing the microprocessor to respond to a four-bit primary address code from the Q-register by issuing additional microinstructions and associated clock codes for executing the instruction contained in the Q-register.

As illustrated by the basic machine operating loop shown in the flow chart of FIG. **7,** microprocessor **120** initially responds to the qualifier control signal QNR either by issuing microinstructions and associated clock codes for interrupting the basic machine operating loop and executing an I/O service routine or by issuing microinstructions and associated clock codes for loading A/B flip-flop **148** with the information Q11 contained in Q-register **128.** The manner in which microprocessor **120** responds is determined by the condition of the qualifier control signal QNR, which in turn indicates whether or not the basic machine operating loop should be interrupted.

Assuming the basic machine operating loop is not to be interrupted, microprocessor **120** loads the information Q11 into A/B flip-flop **148** and responds to the qualifier control signal QMR either by issuing microinstructions for transferring an address portion of the instruction contained in Q-register **128** from T-register **84** into M-register **82** or by responding to another qualifier control signal Q15. Again, the manner in which microprocessor **120** responds is determined by the condition of the qualifier control signal QMR, which in turn indicates whether or not the instruction contained in Q-register **128** is a memory reference instruction.

Assuming the instruction contained in Q-register **128** is a memory reference instruction, microprocessor **120** transfers the required address information into the M-register **82** and reponds to qualifier control signal Q10 either by issuing microinstructions and associated clock codes to select the base page of the memory (i.e. page **0**) or by issuing microinstructions and associated clock codes to select the current page of the memory (i.e. the page from which the instruction contained in Q-register **128** was obtained). In either case, the microprocessor then issues microinstructions as required to read data from the preset page of the memory at the address designated by the address information last transferred into M-register **82.** Upon completion of this operation, microprocessor **120** responds to qualifier control signal Q15 by issuing additional microinstructions and associated clock codes to execute an indirect memory access operation if the condition of this qualifier control signal indicates that the address information contained in M-register **82** is indirect.

Assuming the address information contained in M-register **82** is direct (or upon completion of the indirect memory access operation), microprocessor **120** issues microinstructions and associated clock codes causing the microprocessor itself to respond to a four-bit primary address code from the Q-register. The microprocessor responds by issuing additional microinstructions and associated clock codes for executing whichever one of ten possible memory reference instructions is contained in Q-register **128** and designated by the four-bit primary address code. Folowing execution of the designated memory reference instruction, microprocessor **120** issues microinstructions and associated clock codes causing another sixteen-bit instruction stored in ROM **80** to be loaded into T-register **84** and transferred from there to Q-register **128,** thereby beginning another cycle of the basic machine operating loop.

As illustrted by other possible paths of the basic machine operating loop shown in FIG. **7,** microprocessor **120** sequentially responds to other qualifier control signals when other types of instructions are contained in Q-register **128.** For example, when an I/O instruction is contained in Q-register **128,** microprocessor **120** sequentially responds to qualifier control signals QNR, QMR, Q15, Q10, and QRD by issuing microinstructions and associated clock codes to execute the I/O instruction. It should be noted that the microprocessor qualifier control signals not shown in the simplified flow chart of FIG. **7** are variously contained within those flow chart blocks requiring decisions as will hereinafter become apparent.

The calculator firmware operational diagram of FIG. **9** illustrates the basic components of the calculator firmware. These components comprise routines which reside in the calculator ROM **80** and serve to implement the definition of the calculator. Control information passing between routines is represented by solid lines on the drawing.

Referring to FIG. **9,** it is shown that the calculator hardware units are controlled by firmware routines contained in ROM **80.** These units comprise an on-off power switch **182,** an alphanumeric keyboard input unit **12,** a display unit **18,** and a magnetic tape cassette reading and recording unit **14.** The firmware routines also control an external printer **20,** external tape cas-

sette reading and recording units **14**, and various other external input-output devices **244**.

Operation of the calculator is begun by placing the on-off switch in the on position, thus forcing the hardware internal to the calculator to execute the instruction located at address **0000** of ROM **80**. This instruction directs control to the start-up routine **200**, which is shown in the flowchart of FIG. **83**, and described in detail on page 53 of the basic system firmware listing. The purposes of this routine are to initialize RWM **78**, set the stack pointer address at location **1777**, set the keyboard execution numeric output format to float **9**, initialize certain variables in the system RWM area for later use by other firmware routines, and initialize the various read-write pointers to the user read-write memory area shown in detail in the memory map of FIG. **6**.

After completion of the start-up routine, control is passed to the keyboard monitor routine **202** shown in the flowchart of FIGS. **84A-D** and detailed on pages 53–56 of the basic system firmware listing. This routine initializes certain variables in the system RWM **78** for use by the keyboard input routine **204**. It also outputs the automatic line number if necessary. It then calls for an input record from the keyboard input routine **204**. When the keyboard input routine returns with a record the keyboard monitor routine searches the mnemonic tables in the assembly language program area shown in FIG. **6**. It then searches the mnemonic tables in each of the plug-in read-only memory modules of FIG. **4E** and finally searches the mnemonic tables of the main system ROM **80**. A complete assembly language listing of each of the tables in read-only memory is given in the firmware listings. The subroutines which do the search of the mnemonic tables are detailed in the flowcharts of FIGS. **85, 86,** and **87**. If a match is found between the characters of the input record and any of the mnemonic tables, the keyboard monitor branches through a jump table to the appropriate syntax routine **210** for syntax analysis if the mnemonic is a statement, or branches through a jump table for execution if the mnemonic is a system command. A separate syntax analysis routine is provided for each statement and a separate execution routine is provided for each system command. Syntax and execution routines for statements and commands on an optional read-only memory module are contained in the firmware of that module. The assembly language program area is handled in the same fashion as the plug-in read-only memory module. If no mnemonic is found, control is passed to the implied LET syntax routine.

The keyboard input routine **204** is detailed in the flowcharts of FIGS. **88A–G, 89A–B,** and **90–93**. It calls on the display refresh routine **206** to refresh the 32 character single line display **18** between key entries. The display refresh routine is detailed on page 35 of the firmware listing. When a key is entered through the alphanumeric keyboard **12**, the interrupt circuitry causes the calculator to execute the instruction at address **00002**. This instruction causes a jump to the interrupt routine **208**, which is detailed in the flowchart of FIGS. **117A–B** and pages 257 and 258 of the firmware listing. The interrupt routine **208** saves the keycode in a memory location of the system RWM **78** and returns. The keyboard input routine **204** reads this memory word and decides what operations need to be performed for that particular keycode. The shift bit is stripped from the keycode and stored as a flag in a temporary location in system RWM **78**. If the key requires that a mnemonic name be displayed, the single line display buffer shown in FIGS. **5A–B** is cleared and the mnemonic name is entered. If an editing function is required, a routine is called to perform the editing function. If a user-definable key **f0–f9** has been given, the user-definable key routine **228** is called. If an alphanumeric key has been given, the shift flag in RWM **78** is tested, and, if the shift has been given, the keycode is converted to the code for the shifted key. Then the keycode is inserted into the single line display buffer shown in FIGS. **5A–B**, either at the end of the line or at the cursor position, if the cursor is within the line.

The user-definable key routines **228** perform the special operations for keys **f0–f9**. They are detailed in the flowcharts of FIGS. **99A–F** and pages 43–47 of the firmware listing.

The syntax routines **210** translate the characters of the input record into an internal format which is more easily handled by the execution routines. If a syntax error is encountered control is passed to the error routine **238** which outputs an error message. The error routine is shown in the flowchart of FIG. **95** and the firmware listing at pages 58 and 59. If no error is found, control is passed to either the memory management routines **236**, if the statement is to be stored in memory, or to the routine for initialization for keyboard execution if the statement is to be executed. The memory management routines are shown in the flowcharts of FIGS. **96A–D, 97,** and **98** and pages 16 and 60–62 of the firmware listings. The routine **230** for initialization for keyboard execution serves to initialize the run time stacks shown in the memory map of FIG. **6**. This routine is detailed in the flowchart of FIG. **100C** and pages 109–110 of the firmware listing.

When the RUN command is given, or if the INIT key is actuated, control is passed to the pre-execution processing routines **232**. These routines are detailed in pages 65–77 of the firmware listing. They serve to initialize the symbol table and non-common value table areas of the user read-write memory shown in FIG. **6**.

Control is next passed to the execution monitor **214**, which is detailed in the flowchart of FIGS. **100A–B** and pages 108–110 of the firmware listing. This routine initializes the run time stacks in the user read-write memory of FIG. **6** and initiates execution of a stored program beginning at the line number given by the user. After each statement is executed control is returned to the execution monitor, which prints the line number of the next line if the program is being executed in the trace mode. Step, stop, or error conditions are checked and program execution is terminated if any of these conditions exist. If program execution is to be continued, the jump address of the execution routine for the next statement of the program is computed, and control is passed to that routine.

Several of the statement execution routines **240** require evalution of arithmetic functions and expressions. This is done in the formula evaluation routines **242**. Several of the statement execution routines **240** require input from or output to various external input-output devices **20** and **244**. This is done by calling the standard output driver **224** or an optional special I/O driver **234**. Several of the statement execution routines require input from or output to an internal or external tape cassette unit **14**. This is done by calling the tape cassette drivers **226**. The statement execution routines for the statements that communicate with the tape cassette

units are detailed in the flowcharts of FIGS. **110–115** and pages 250–276 of the firmware listing.

The list routine **220** is used when listing stored programs on either the single line display **18** or an external ASCII output device **20.** The list routine is detailed on pages 78–82 of the firmware listing. Its function is to translate the stored program from the internal stored format into a string of ASCII characters which can be printed or displayed. To print a line of translated characters, the list routine calls on the standard output driver **242** which is detailed in FIG. **94E** and pages 20 and 33 of the firmware listing.

Detailed assembly language information relating to all of the firmware routines and subroutines herein described may be obtained by referring to the memory map of FIGS. **4A–F** and the basic system firmware listing located at a later point in this specification.

Communication with the routines in the various plug-in read-only memory modules is accomplished through a series of mnemonic tables and jump tables. The standard firmware, the cassette operating firmware, and each of the plug-in read-only memory modules all contain the following tables:

1. A statement mnemonic table
2. A statement syntax jump table
3. A statement execution jump table
4. A system command mnemonic table
5. A system command execution jump table
6. A function mnemonic table
7. A function execution jump table
8. A non-formula operator mnemonic table

All of these tables, with the exception of the statement execution jump table, may appear anywhere within a memory module. The last five words in each module are used by the table scan routines of FIGS. **85–87** to find the actual location of the tables. The last word of each module contains a unique operation code word for that particular module. The second from the last word contains a relative address of the statement mnemonic table. The third from the last word contains a relative address of the system command mnemonic table. The fourth from the last word contains a relative address to the function mnemonic table. The fifth from the last word contains a relative address to the non-formula operator table. The jump tables for statement syntax, system command execution, and function execution are located directly above their respective mnemonic tables. The statement execution jump table is located directly above the fifth from the last word of each module. The complete set of tables for the standard firmware is shown in the firmware listings at pages 48–49, 103–104, 131, and 141.

Each of the mnemonic tables consists of a string of seven-bit ASCII character and six-bit operation code characters packed two characters per sixteen-bit word. The eighth bit of each character is used to indicate whether that character is ASCII or an operator code. A zero in the eighth bit indicates ASCII and a one indicates an operation code. The seventh bit of each operation code character is used to indicate whether that operation code is the last character in that table. The jump table address for each mnemonic is found by subtracting the operation code for that mnemonic from the startng address of the associated mnemonic table. The internal stored format for program statements consists of a series of operation codes, operand codes, and other special codes. The first word of each statement contains the line number of that statement in binary format. The second word contains both the operation code for that particular statement mnemonic and also the length of the statement. The length information is used by various firmware routines to scan from one statement to the next. The third word contains the operation code for the table or optional read-only memory module, and it also contains the first operand code. The remainder of the statement is stored with one operator code and one operand code in each word.

Formula operation codes from the table on page 132 of the firmware listing and mnemonic operation codes are stored in a five-bit field, bits **10–14.** The operand codes are stored in two five-bit fields. Bits **5–9** are used to store the operand name. The name consists of an ASCII letter, A–Z, with its sixth and seventh bits removed. For example, the ASCII code for A is **1000001** and the five-bit operand code for A is **00001.** Bits **0–4** are used to store the operand type. Bit **15** is used as a special flag bit. When bit **15** is set, the operand field is interpreted differently than when it is not set. The following table shows the various operand types and the special codes.

OPERAND TYPE CODE	OPERAND TYPE (BIT 15=0)	MEANING IF BIT 15=1
	Full Precision Variables	Constant follows in next word
00000	Simple variable	
00001	Array of 1 dimension	Fixed point decimal
00010	Array of 2 dimensions	Floating point decimal
00011	Array of unknown dimension	Binary integer
	Split Precision Variables	
00100	Simple variable	Binary line number
00101	Array of 1 dimension	
00110	Array of 2 dimensions	
	Integer Precision Variables	
01000	Simple variable	
01001	Array of 1 dimension	
01010	Array of 2 dimensions	
	Full Precision Variables Letter followed by digit	
10000	0	
10001	1	
10010	2	
10011	3	
10100	4	
10101	5	
10110	6	
10111	7	
11000	8	
11001	9	
11110	String Variable	
11111	User defined function	Bits 5–9 contain the operation code of a function in ROM

As an example, the internal stored format for the following statements is shown in the table below:

```
10 DIM A[5]
20 LET B=C+FND(E)
30 GOTO 100
```

15	14–10	4–5	4–0	
0	00000	00000	01010	10
0	00010	00000	00111	DIM op-code – Length 7
0	01010	00001	00001	Table op-code – A [dimension
1	10100	00000	00011	[– integer follows
0	00000	00000	00101	5

-continued

0	00101	00000	00000	] – null operand
0	00000	00000	10100	20
0	10110	00000	00111	LET op-code – Length 7
0	01010	00010	00000	Table op-code – B
0	00111	00011	00000	= – C
0	01001	00100	11111	+ – FND
0	10101	00101	00000	(– E
0	00100	00000	00000	) – null operand
0	00000	00000	11110	30
0	00110	00000	00100	GOTO op-code – Length 4
1	01010	00000	00100	Table op-code – integer follows
0	00000	00011	00100	100

KEYBOARD OPERATIONS

All operations performed by the calculator may be controlled or initiated by the keyboard input unit and/or by keycodes entered into the calculator from the keyboard input unit, the magnetic tape cassette reading and recording unit, or peripheral input units such as the marked card reader and stored as program steps in the program storage section of the RWM. An operational description of the keyboard input unit is therefore now given with specific reference to the perspective view of the calculator as in FIG. **1** and the plan view of the keyboard as in FIG. **8**, except as otherwise indicated.

Line Switch

An on-off line switch **182**, which may be considered as part of the keyboard input unit, controls the application of power to the calculator and hence initiation of the control signal $\overline{POP}$ from the power supply.

As shown in FIG. **2**, the calculator may be operated at 240, 220, 120, **or** 100 volts +5%, −10% as determined by a pair of line voltage selector switches mounted at rear panel **34** of the calculator housing and at a line frequency within the range of 48 to 66 Hertz. The calculator is provided with a 6-amp fuse and either a 1-amp fuse for operation at a line voltage of 220 or 240 volts +5%, −10% or a 2-amp fuse for operation at a line voltage of 100 or 120 volts +5%, −10%. It is also provided with a three-conductor power cable **184** which, when plugged into an appropriate AC power outlet, grounds the calculator housing. The maximum power consumption of the calculator is 150 voltamps. No more than a total of 610 voltamps may be drawn from AC power outlets **42** provided for peripheral units.

EXECUTE

The EXECUTE (often referred to as EXEC) key, when pressed, will perform the indicated operations previously keyed in, if any, and display the result of any arithmetic statements on the 32-character display. (Although the display is only 32 characters, an 80-character line can be keyed in with automatic scrolling both for programs and for keyboard operations.) Most keys, when pressed, immediately cause their mnemonic to be displayed. However, pressing certain keys, such as PRT ALL (to be discussed later), allows a particular mode to be in effect till that mode is overridden.

FIXED N, FLOAT N

Immediately after either turn-on or *SCRATCH EXEC*, the user read/write memory area is cleared; numerical calculations that are executed will be displayed in float-nine notation. The values 2 and 12 in float-nine notation would appear as 2.000000000E+00 and 1.200000000E+01, respectively. Float 9 refers to the nine digits succeeding the decimal point; E symbolizes × 10 raised to the power of the two digits following the E.

NOTE

In this text, individual keyboard operations will be identified by being italicized; e.g., **3** + **2** *EXEC*. (On the display would appear 5.000000000E+00.)

You can specify the desired notation by pressing *FIXED N* or *FLOAT N* followed by the appropriate number from 0 through 11. The designated N indicates the number of digits to be displayed to the right of the decimal point after execution.

For example:

FIXED N **7** *EXEC*,

then *1 2 3 4 5 6 . 7 EXEC*, displays 123456.7000000

or, *1 2 3 4 5 6 7 8 9 . 1 2 3 4 5 6 7 8 9 EXEC*, displays 123456789.1230000*

FLOAT N **5** *EXEC*,

then **1 2 3** . **4** *EXEC* displays 1.23400E+02

or, . *1 2 3 4 5 6 7 EXEC* displays 1.23457E−01**

In fixed-*n* notation, a maximum of 12 digits will be displayed to the left of the decimal point; beyond that value the calculator reverts to float-*n* notation, with the number of digits displayed to the right of the decimal point determined by the particular fixed-*n*.

* The machine calculates to 12 significant digits regardless of the display length.

** Since only five digits can be displayed to the right of the decimal point the fifth digit is rounded.

The FIXED N and FLOAT N keys are not programmable; that is, they can be used only in keyboard operations. Output formatting under program control is accomplished by a format statement, which is discussed below.

The calculating range of the calculator is: $-9.99999999999 \times 10^{99}$ through -10^{-99}, 0, and 10^{-99} through $9.99999999999 \times 10^{99}$.

CLEAR, DELETE LINE

Pressing either *CLEAR* or *DELETE LINE* will erase whatever previously had been displayed during keyboard operations. Pressing either key will cause ⊢(the Lazy T) to appear on the far left of the display, indicating that the calculator is available for new inputs. The difference between the two keys occurs when a stored program line is displayed; *DELETE LINE* will erase the line from the stored program, whereas *CLEAR* will only erase the display and not affect the program line itself. (A program line can also be deleted by keying in the line number followed by *END OF LINE*.)

RECALL

Pressing *RECALL* during keyboard operations allows the last line that was executed to be recalled to the display; pressing *RECALL* during program-inputting operations allows the last program line that was input to be recalled to the display. If an error message appears on the display when an attempt is made either to execute a line in keyboard mode or to input a program line into memory, pressing *RECALL* will allow the original line to be reviewed for editing purposes.

RESULT

Pressing *CLEAR RESULT EXEC* displays the numerical value of the last arithmetic statement that was executed. The RESULT key not only allows the result of the previously executed statement to be reviewed, but also can function as an accumulator during arithmetic operations; e.g., by pressing

2 + 4 EXEC (in FIXED **2** notation) displays 6.00, then pressing

3 + *RESULT EXEC* displays 9.00, then pressing

4 + *RESULT* + *RESULT EXEC* displays 22.00.

Keying in either *R E S* or *R E S U L T* has the same effect as pressing the *RESULT* key.

PRINT ALL

By pressing *PRINT ALL*, you can determine whether or not the calculator is in print all mode. If ON appears, both the expression and the result of any executed calculation will be typed on whichever printing device is plugged into the calculator; if OFF appears, the information will appear only on the display. Pressing the PRINT ALL key a second time causes the alternative mode to be in effect. In the print all mode, each program statement is printed when *END OF LINE* is pressed. All error messages are also printed.

BACK, FORWARD, INSERT

The *BACK* and *FORWARD* keys can be used to edit expressions on the display. Successive presses of the *BACK* key will move a blinking cursor to the desired location within the display. Editing can then be performed at this location. The *FORWARD* key performs the same function as the back key, but in the opposite direction.

At the location of the blinking cursor, the following editing can be performed:

1. A character can be inserted by pressing *INSERT*. (This opens up a space to the left of the cursor, thereupon moving the cursor to the location of the space; additional presses of the INSERT key will open up more spaces, with the cursor always positioning itself in the left-most space, thus allowing for the immediate insertion of more than one character.)
2. A character can be changed by overscoring it with another character.
3. A character can be deleted by pressing either the space bar or *SHIFT INSERT*, the only difference being that pressing *SHIFT INSERT* will close up the space of the deleted character.

Holding either BACK or FORWARD down for approximately 1.5 seconds will cause the cursor to move in rapid succession in the chosen direction.

Once a line is appropriately edited, it can be immediately executed without having to move the cursor to the end of the line.

→ , ←

When a line that is greater than 32 characters (80 characters maximum) is being input, the characters to the left of the display are pushed out of the display region to make room for the additional characters. To view the beginning of the input, press → (the right arrow); this operation moves the characters in the display to the right. Pressing ← (the left arrow) performs the reverse operation. Either arrow key, when held down for approximately 1.5 seconds, will repeat its operation in rapid succession.

ARITHMETIC

There are five basic numerical operators: add (+), subtract (−), multiply (X), divide (/), and exponentiate (↑). The order of execution, known as the hierarchy, is identical to the BASIC hierarchy described below. The BASIC functions described below are available on the Calculator simply by keying in the appropriate mnemonics. In addition, the value of π can be obtained by keying in *P I*.

When operating on trigonometric functions, the calculator assumes the angle to be in radians unless otherwise stated. To express an angle in either degrees or grads, first clear the display, then press either *D E G EXEC* or *G R A D EXEC*, respectively; then key in the expression. To revert to radians, clear the display, then press *R A D EXEC*. Radians, degrees, and grads are also programmable commands.

VARIABLES

The simple scalar variables are A through Z and A0 through Z9 (286 total). Simple variables can be used in keyboard operations; e.g., pressing *A* **3** = **7** *EXEC* assigns the value of 7 to A3. Unless the value of A3 is then changed or erased from memory, pressing **4** * *A* **3** *EXEC* will display 28. If a variable is undefined, any attempt to use this variable in an expression (other than by assigning a value to it) will result in an error message.

An array is an ordered collection of numerical data. An array (subscripted) variable can have either one or two dimensions as indicated by the subscripts, which are presented as numbers within parentheses. A (m) is a one-dimensional array (or column vector) where m designates the row of the element; A(m,n) is a two-dimensional array where m designates the row and n designates the column of the element.

The maximum size of an array is limited by the available calculator memory. At normal 12-point precision, the effective memory limitation would be approximately a (30, 30) array.

Arrays should be referenced in either a common statement or a dimension statement before use in a program; if not, the program defaults to either a 10 element array for a singly subscripted array, or a 10 by 10 array for doubly subscripted arrays.

TAPE CASSETTES

The present Calculator is capable of operating with 10 tape cassettes: One cassette is available through the built-in (internal) cassette drive. Four peripheral cassette drives can be connected directly to the calculator through the four I/O slots in the rear panel. Five more peripheral cassette drives can be added if the user has an I/O expander box (in all, 11 peripheral devices can be added by having an expander box).

SYNTAX

The tape cassette commands have general syntactical rules which will be briefly described. The general command form, with minor variations, is:

COMMAND [UNIT] [FILE], or

COMMAND [UNIT] (FILE) [LNX$_1$ [LNX$_2$]]

Brackets [ ] indicate that the enclosed information is optional; parentheses ( ) indicate that the enclosed information is required with the particular command.

COMMAND — The various commands will be individually discussed.

UNIT — Individual units are referenced by the number sign, **#**, followed by the select code; the internal cassette is designated by **#10** (if no select code is given, the internal cassette is assumed); the nine peripheral cassettes are identified by **#1** through **#9**.

FILE — Files within individual cassettes are identified by file numbers; if no file is identified by number, the scratch-pad (or default) file, file **0**, is assumed.

LNX_1 — indicates the beginning line number to be affected by the command.

LNX_2 — indicates either the line number where program execution is to begin following the command implementation, or the ending line number to be accessed in the command. In any command whose syntax allows LNX_1 and LNX_2, in order for LNX_2 to be designated, LNX_1 must have been specified.

All information to be input following the command must be separated by commas.

In keyboard mode, press *EXEC* after fulfilling the syntax requirements of a particular command to implement the command.

PREPARING A FRESH CASSETTE

To prepare a fresh cassette (one which has no markings), first open the cassette door by pressing downward on the switch on the far right of the keyboard; then insert the cassette (print-side up) into the slot built into the door; be certain that the tape is wound around the left spindle — if it is not, but you wish to prepare that side of the cassette anyway, simply close the door and press *REWIND*.

To store information onto the cassette, first mark the files that are to be used.

PROGRAMMABLE COMMANDS

The following tape cassette commands are programmable: mark, store, load, merge, store key, load key, store date, load data, load binary, rewind, find, and tlist. These commands can also be used in the keyboard mode. The secure command, on the other hand, can be used only in the keyboard mode.

MARK

The mark command produces the designated number of files and defines the file lengths (by the number of 16-bit words per file).

Syntax: MARK [UNIT] (No. of FILES) (LENGTH)

e.g., *MARK #10, 5, 1000 EXEC* — since unit **#10** identifies the internal cassette, including it is superfluous; the 5 indicates the number of files to be made available; the 1000 signifies the number of 16-bit words per file.

Successive files on the tape can be marked with different word lengths as shown in the following example; *MARK* **3, 1000** *EXEC* then, *MARK* **2, 2000** *EXEC* will mark files **0, 1, 2** of the internal cassette with 1000-word lengths and will mark files **3, 4** of the same cassette with 2000-word lengths.

The length of a file can be changed; however, changing it will affect all the files following it by distorting their contents. To change the length of a particular file, first position the cassette at that file by using FIND (FIND will be discussed more thoroughly, later); then mark the file with the desired length. For Example: to mark file **12** of the internal cassette with 1000 words, press *FIND* **12** *EXEC* then press,

MARK **1, 1000** *EXEC* To mark files not previously marked (virgin files), first perform a TLIST; TLIST will be discussed thoroughly later; but for now, merely knowing that *TLIST [UNIT] EXEC* will list all the marked files, is sufficient. To mark virgin files, it is first necessary to mark the last file listed in TLIST*. As previously discussed, this file is located by the find command. Beginning with this file, successive virgin files can be marked by the methods previously discussed.

* The number of files referenced in TLIST is always one more than the number marked by the user.

To mark the beginning of a tape, be certain that the tape is completely rewound.

STORE

The store command will take the program line numbers in read/write memory, and put them in the designated tape location where they will be saved.

Syntax: STORE [UNIT] [FILE] or STORE [UNIT] (FILE) [LNX_1 [LNX_2]]

e.g., *STORE* **#2, 3** *EXEC*. select code **#2** designates which cassette is being accessed; the program will then be stored on that cassette in file number **3.**

In the store command, LNX_1 and LNX_2 are used to store a portion of the program: LNX_1 specifies the beginning line number to be stored; LNX_2 specifies the ending line number to be stored. If LNX_2 is not specified, the last line to be stored is assumed to be the highest-numbered program line.

For LNX_2 to be given, it is always necessary to have LNX_1; this is true for any cassette command. In addition, whenever LNX_1 is to be given, the file to be accessed must be identified; e.g.,

STORE **3, 60, 150** EXEC: file **3** of the internal cassette is located; program lines **60** through **150** in memory will then be stored into file **3.**

Note

Once a cassette has been marked, any file that has been marked can be accessed by giving its designated file number in a command; e.g., if five files are marked, information can be stored in file **4** even though file **3** is a virgin (empty) file, by pressing *STORE* **4** *EXEC.*

LOAD, LINK

The load common will take a program that is stored on a cassette and put it into the memory area.

Syntax: LOAD [UNIT] [FILE] or,

LOAD [UNIT] (FILE) [LNX_1 [LNX_2]]

e.g.,

LOAD EXEC assumes the internal cassette, default file (file **0**) is to be loaded into the calculator; any program previously in the **9830A** memory will be erased.

LOAD **5 , 40 , 10** *EXEC* locates file **5** on the internal cassette; the entire program on this file is renumbered beginning at line number **40,** and then the program is loaded into memory; program execution is initiated at line number **10.** All program line numbers beginning at line **40,** that were previously in memory, will be erased and replaced by the program being loaded; if the memory previously had line numbers **10, 20, 30,** it will retain them.

In both the load and the merge command, the use of LNX_1 and LNX_2 can have various results. Rules for predicting the results are given at the conclusion of the merge command discussion.

By substituting LINK for LOAD in the previous syntax, the user can implement the link command. This com-

mand operates identically to the load command with one exception:

During program execution, if a load statement is encountered, the calculator functions as though *RUN EXEC* were pressed — that is, the old symbol table is destroyed and a new symbol table is built; on the other hand, if a link command is encountered, the calculator functions as though *CONT EXEC* were pressed — that is, all variables retain their previous values.

MERGE

The merge command attempts to take program line numbers from the cassette and position them in read/write memory in front of the program currently there, between consecutive line numbers in the program currently there, or behind the program currently there. However, if any line number of the program to be entered matches a line number currently in the program, an error will result. In addition, if the line numbers of the two programs are interwoven, an error will occur; e.g., if the program currently in memory has line numbers **10, 20, 30, 40** and if the program to be merged has line numbers **15, 25, 35,** the merge command will cause an error message to occur even though no two line numbers matched.

Syntax: MERGE [UNIT] [FILE] or,
MERGE [UNIT] (FILE) [LNX_1 [LNX_2]]

e.g.,

MERGE #**2, 1, 200, 100** *EXEC* — file **1** of the cassette with select code #**2** is located; the entire program on this file is renumbered beginning at line number **200** (LNX_1) and then it is combined with the program currently in memory. Following implementation of the command, program execution will begin at line number **100** (LNX_2).

Either the merge or the load command can be used for stacking programs in read/write memory. The major difference between the two commands is as follows: LOAD will erase the line numbers previously in memory, beginning at the designated LNX_1; MERGE, on the other hand, will retain all line numbers previously in **9830A** memory.

In both the load and the merge command, LNX_1 and LNX_2 are predictable. Regardless of the mode, LNX_1 renumbers the program line numbers of the accessed file to begin at LNX_1; the spacing between consecutive line numbers remains the same; all GO TO statements, etc. are properly adjusted to reflect the new line numbers; this program is then loaded into user memory.

In program mode:

1. If LNX_2 is given, program execution will continue at LNX_2.
2. If LNX_2 is not given, program execution will continue either at the next higher line number of the original program or at LNX_1, whichever comes first.

In keyboard mode:

1. If LNX_2 is given, program execution will begin at LNX_2.
2. If LNX_2 is not given, the calculator will halt after loading in the program.

STORE KEY, LOAD KEY

The store key command will take all user definable keys (upper left-hand region of the keyboard) that have been defined and put them on a cassette file, which will be tagged as a key file.

The load key command takes the user definable information from the cassette and positions it in memory such that each user definable key performs the same operation that it previously did before being stored on tape.

Syntax: STORE KEY [UNIT] (FILE)
LOAD KEY [UNIT] (FILE)

Spacing may arbitrarily be left between the words STORE and KEY and the words LOAD and KEY. In general, the HP Basic language ignores blank spaces (except, of course, in a quote field where every character and space is duplicated).

STORE DATA, LOAD DATA

The store data command takes a block of data from memory and puts it on a cassette. Normally, only arrays can be stored using this command; however, if no array is specified in the command, all data in the common statement of the program can be stored. The calculator allows simple variables in the common statement, as well as arrays; in fact, the common statement can accept simple variables, array variables, integer arrays and variables, and split arrays and variables.

The common statement acts like a dimension statement with the additional feature that data in common is saved from program to program.

The load data command takes the data that was previously stored on a cassette file, and loads it into mainline memory, If an array has been stored, then LOADDATA must specify an array; if LOADDATA does not specify an array, an error will result. If, on the other hand, the common area will be retrieved by the load data command (in this case, no particular array can be specified in the command, lest and error occur).

Syntax: STOREDATA [UNIT] (FILE) [ARRAY]
LOAD DATA [UNIT] (FILE) [ARRAY]

e.g.,

STORE DATA **6,** *B EXEC* will locate file **6** on the internal cassette; then the B array in the current program will be stored in this file. (A simple variable cannot be stored in this manner.)

LOADDATA **6,** *B EXEC* can then retrieve the B array and load it into memory whenever it is needed; the following command could be given, also:

LOADDATA **6,** *C EXEC:* this would retrieve the B array and load it into memory in place of the C array, provided B and C are the same size and type.

Assume the common statement in a program looks like this: **1** COM A(**8**), B(**5,5**), D**3**, E.

Pressing: *STORE DATA* **2** *EXEC* will store all the common statement variables into file **2** of the internal cassette. Both the array variables and the simple variables will be retrieved by pressing: *LOAD DATA* **2** *EXEC.*

Pressing: *LOAD DATA* **2,** *A EXEC* is illegal and causes an error since no particular array can be retrieved from a file if it was stored in common.

In all store and load data commands, the file to be accessed must be identified, even if it is the default file (file **0**).

LOAD BIN

The load binary command will transfer binary information — assembly language program — from the cassette to the user memory. The assembly language program may be a system diagnostic, an I/O subroutine, or a

simulated option block designed to perform some specific function.

The assembly language program cannot be listed or displayed.

Syntax: LOAD BIN [UNIT] (FILE)

Note

Files on cassettes are tagged as: program files, key files, data files, or binary files. If an attempt is made to load from a particular file and the load command incorrectly identifies the file tag, an error will occur; e.g., pressing *LOAD KEY* **1** *EXEC* when file **1** is a program file, causes an error message to appear.

REWIND

Pressing the *REWIND* key, located on the right-hand side of the keyboard, immediately rewinds the internal tape cassette to the clear leader.

To rewind any other cassette, *R E W I N D* must be typed in, followed by the select code of the particular cassette. (The internal cassette can also be rewound by typing in *R E W I N D EXEC.*)

R E W I N D must be typed if the command is to be used in the programming mode.

Syntax: REWIND [UNIT]

e.g.,

R E W I N D **#2** *EXEC* will rewind the cassette with select code **#2.**

FIND

The find command (previously mentioned in conjunction with the mark command) is used to locate a particular file. While the cassestte is searching for the file number, a laxy T appears on the display. During this interval, the cassette is searching under interrupt control, thus returning control of the calculator keyboard to the user. This feature allows for the execution of one portion of a large program, while another portion is being found — thereby improving access time.

When the specified file is found, the cassette tape halts.

Syntax: FIND [UNIT] (FILE)

e.g.,

FIND **#3, 2** *EXEC* causes the cassette with select code **#3** to search until file number **2** is located.

TLIST

Beginning with the current tape location, this command reads all subsequent file identifiers and prints out information concerning each file.

Syntax: TLIST [UNIT]

The information for each file, on the designated cassette, is printed out on one line. There are no column headers identifying the information in each line; the assumed headers are:

File No.	File Type* (Code No.)	Absolute File Size (in words)	Actual File Size (in words)	Program Line Nos. (Beginning) (LNX_1)	Program Line Nos. (Ending) (LNX_2)	Common Area (in words)

*The code numbers identifying the file types are as follows:
1 binary
2 data
3 program (source)
4 key

In addition, if the file is secured, the number **2** appears in front of the code number (this applies only to binary, source, and key files); e.g., if **24** appears in the second column, the file is a secured key file.

If a file is a data file, LNX_1, is superfluous; in this case the data is described in this column as:

0 full precision
1 split precision
2 integer precision
3 common

If the file is not a program file, the last two columns will contain no information.

NON-PROGRAMMABLE COMMANDS

SECURE

This command has the capability of concealing program lines from potential users; that is, your program could be given to another, and that person could load and run it - however, he would not be able to fetch particular program lines for viewing purposes nor could he store the program on any other cassette file.

An attempt to fetch a secured program line will result in the line number appearing on the display, followed by an *; attempts to list the program will result in the secured line numbers appearing followed by an *.

If any lines in a program are secured, the entire program is considered to be secured; that is, even though certain program statements are visible, hone of the program can be reproduced onto another cassette file.

Syntax: SEC [LNX_1 [LNX_2]] or SECURE [LNX_1 [LNX_2]]

It is therefore, possible to secure specific lines within a program; e.g., pressing: *SEC* **30, 80** *EXEC*, followed by *STORE* **2** *EXEC* secures lines **30** through **80** of the program in memory and then stores both the secured and unsecured portions of the program into file **2** of the internal cassette.

When a program is initially secured, it can still be reproduced onto as many files as necessary; however, once the program is scratched from memory, (even though it can be loaded back into memory) it cannot be reporduced onto any cassette files.

When program lines are secured, the entire calculator is in the secured mode. Therefore, after the secured program is stored away, the user should press *SCRATCH A* before inputting other programs — thus, avoiding secured-program errors.

User definable keys (when not being used as typing aids) can be secured, too. Just press *FETCH* (*particular key*) *SEC EXEC* and the designated key will be secured.

Note

To protect all the information on a particular cassette, break one of the tabs on the top of the cassette; this makes the cassette inaccessible for further storage.

PROGRAMMING

The programming language of the Calculator is, with minor variations, BASIC as described below.

Instructions to the computer within a program are provided by program statements. Each statement in Basic has an associated line number which must appear in the left-most portion of the statement.* Statement line numbers appear in ascending order with 9999 being the largest possible line number.

*The length of a statement, including the line number and including appropriate spacing, can be up to **80** characters.

A program statement, which has been correctly keyed in, can be stored in read/write memory by pressing the *END OF LINE* (EOL) key. This key is situated in an area corresponding to the carriage return/line feed key on a teletype keyboard.

AUTO #

As previously mentioned, statement line numbers can be automatically input. In its simplest form, pressing *AUTO # EXEC* causes line number **10** to immediately appear on the display awaiting the program statement. The line numbers of additional statements will be in ascending order with a spacing of ten between consecutive line numbers. The AUTO # (AUTO) syntax and the examples to follow present some of the alternatives in automatic line numbering.

Syntax: AUTO # [LNX_1 [Spacing]]

LNX_1 is the beginning line number to be automatically input; the desired spacing between lines can then be input if LNX_1 is given. If no spacing is indicated, a spacing of 10 is assumed. E.g.,

AUTO #30 EXEC	AUTO #40 , 2 EXEC	AUTO # EXEC
30	40	10
40	42	20
50	44	30
⋮	⋮	⋮

When AUTO # is pressed, AUTO appears in the display.

Note

Although the CLEAR and DELETE LINE keys have previously been discussed, the following point should be made. If a program line currently being keyed in is found to be totally unacceptable (not worth salvaging by using the editing keys), it can be erased by using either the CLEAR or the DELETE LINE keys. However, if the pogram line numbers have been automatically input, pressing *CLEAR* not only erases the entire display but also eliminates the AUTO # mode, whereas pressing *DELETE LINE* erases only the program statement without affecting the line number itself.

PROGRAM VIEWING

There are two methods of viewing a program that is currently in memory:

1. List it on a printing device.
2. Bring it line-by-line to the display.

LIST

The list command has two specific applications: it can be used to provide a total listing of the programs in read/write memory, or it can be used to indicate the read/write memory available for inputting; e.g.,

LIST EXEC lists all program lines that are in memory on the user's standard printing device.

LIST #3 EXEC lists all program lines that are in memory on the peripheral with select code #3.

LIST **9999** *EXEC* causes the number of 16-bits words available in memory to be displayed.

↓, ↑

If a specific program line is in the display, pressing ↓ (down arrow) displays the next higher-numbered program line *; pressing ↑ (up arrow), on the other hand, displays the next lower-numbered program line. When no program line is currently in the display, pressing ↓ would display the successively higher line from the one most recently displayed, whereas ↑ would display the successively lower line.

*It should be noted that if a program line has more than 32 characters, the first 32 characters will be displayed; by pressing ←, the rest of the line can be viewed as the display is scrolled to the left. At the end of the line + appears.

FETCH

In addition to viewing a program line-by-line, specific program lines can be immediately brought to the display by using the fetch command.

Syntax: FETCH [LNX_1]

Where LNX_1 is the specific line number to be accessed; e.g.,

FETCH EXEC always displays the lowest-numbered program line.

FETCH **300** *EXEC* displays line **300** if it exists; if line **300** is not available (and there are other higher-numbered lines), the line immediately higher than **300** will be displayed; if there are no line numbers as high as **300,** the highest-numbered line available in memory will be displayed.

PROGRAM EDITING

Any displayed program line can be edited by using the BACK, FORWARD, and INSERT keys.

The RECALL key, previously discussed, need only be briefly mentioned. A program line is keyed in; if an error message appears on the display when *EOL* is pressed, the program line can be reviewed by pressing *RECALL*. Appropriate editing can then be performed on the program line. The previously input line can always be recalled by pressing RECALL whether or not an error appears.

CLEAR and *DELETE LINE* have already been thoroughly discussed. However, to reiterate, there are two methods of deleting a program line currently in memory:

1. If the line is currently in the display, pressing *DELETE LINE* will erase it from memory (*CLEAR* only clears the display).
2. Any line in memory can be immediately deleted by keying in the appropriate line number followed by *EOL*.

DELETE

The delete command (to be distinguished from DELETE LINE) can selectively delete program lines.

Syntax: DELETE [LNX_1 [LNX_2]] or DEL [LNX_1 [LNX_2]]

Where LNX_1 is the first line to be delected and LNX_2 is the last line to be deleted:
e.g.,
DELETE EXEC deletes all program lines;
DELETE **40** *EXEC* deletes all statements beginning at line number **40;**
DELETE **50, 80** *EXEC* deletes all lines numbered **50** through **80.**

SCRATCH

The scratch command can erase a variety of things from memory:
e.g.,
SCRATCH EXEC erases all program lines and variables;
SCRATCH A EXEC erases everthing from memory — program lines, user definable keys, variables (identical to turning the calculator off, then on again);
SCRATCH K EXEC erases all user definable keys (user definable keys will be discussed later);
SCRATCH V EXEC erases all variables;
SCRATCH (*particular UD key*) erases the particular user definable key that was pressed — pressing *EXEC* is not required in this case.
Scratch can be accessed either by keying in the seven letters or by pressing *SCRATCH.*

RENUMBER

The renumber command will take all the program line numbers and renumber them.
Syntax: RENUMBER [LNX_1 [SPACING]] or REN [LNX_1 [SPACING]]
Where LNX_1 will be the new line number of the first program statement, and SPACING will be the spacing between consecutive line numbers:
e.g.,
REN EXEC will renumber all statements by numbering the first statement **10** with a spacing of 10 between statements;
RENUMBER **30** *EXEC* renumbers the first statement **30** with a spacing of **10** between statements;
REN **45, 20** *EXEC* renumbers the first statement **45** with a spacing of 20 between statements.

All statements in the program that reference another line number are appropriately corrected with the renumber command; e.g., GO TO **80** would be corrected to reference the line that replaced **80.**

PROGRAM DEBUGGING

NORMAL, TRACE

The TRACE key can be used to determine the order of statement execution for a program that is currently running. Pressing *TRACE* during program execution causes the line numbers to be printed in the order in which they are accessed; then pressing *NORMAL* reverts the calculator to the normal mode. Thus, when a program is running, both *TRACE* and *NORMAL* are immediate execute keys.

When no program is running, to revert to either trace or normal mode requires pressing the appropriate key followed by *EXEC.* Trace mode can be set up to trace specific line numbers.
Syntax: TRACE [LNX_1 [LNX_2]]
Where LNX_1 is the first line number to be traced, and LNX_2 is the last line number to be traced;
e.g.,
TRACE **20** *EXEC* will trace beginning at line number **20** when the program is running.
TRACE **50, 60** *EXEC* will trace beginning at line number **50** and ending at line number **60,** each time these line numbers are executed in the program.

STOP

The stop command can be a statement within a program (to be discussed later) and can be used as a debugging tool.
As a debugging tool, STOP is extremely valuable. A program that is running can be stopped at any time by pressing *STOP* (the current line number of the program will be displayed). If any program lines are then edited, the program must be rerun from the beginning, using the RUN key. If no editing has been performed, the program can continue where it left off if the CONT key (to be discussed in detail later) is pressed.

While a program is stopped, the values of variables can be checked to determine if the program is doing what was intended; e.g., pressing *A EXEC* would determine the present value of the simple variable A.

While a program is stopped, STOP* , can have another funtion. Pressing *STOP* displays STOP; then keying in either one line number or two line numbers separated by a comma — indicates that the calculator should stop program execution at these line numbers. Pressing *CONT EXEC* will then start program execution; e.g., *STOP* **80** *EXEC* then *CONT EXEC* or *RUN EXEC* will cause the program to halt at line number **80.** Once a program is running under these conditions, there is one way to revert to normal program execution: After the program has halted, press *STOP EXEC,* then *CONT EXEC,* and the program will no longer stop at the given line numbers.
*When key is not being used as an immediate execute key, whether the key itself is pressed or whether the letters displayed on the key are individually keyed in, is generally arbitrary.

STEP

After a program is halted by a stop command, execution can continue by pressing *STEP.* STEP is always immediate execute; it causes the program to execute the appropriate statement and then to halt. Therefore, after each statement is executed, it can be checked to ensure that it performed the required function. When a particular function is considered satisfactory, either *CONT* or *STEP* can be pressed; CONT will execute the rest of the program while STEP will execute the next program statement only.

RUN, CONTINUE

Pressing RUN EXEC causes a program to begin execution at the first statement regardless of whether the program had previously been halted by a stop command; however, if the program had been halted by a stop command, pressing *CONT EXEC* will begin program execution where it had previously halted.
As previously mentioned if a program is edited after it has been halted, program execution is reinitialized at the first line by RUN EXEC*. However, if, during the halt, the values of variables are changed or other internally-programmed conditions are changed, then *CONT EXEC* must be pressed to keep these newly adjusted conditions intact. The following things can be done to the program while it is halted if, upon completion, the program is executed by pressing *CONT EXEC:*

1. Variables can be changed; e.g., *B* = 5 *EXEC* sets the simple variable B equal to 5.
2. Angular units in trigonometric functions can be changed to measure in radians, degrees, or grads,

depending on the user's requirements; e.g., *DEG EXEC* will assume all angles to be in degrees.

3. Write, print, and display statements can be input from the keyboard (these statements will be discussed later).
4. The data pointer, which indicates the next datum to be encountered, can be reset to the beginning of the data by typing *R E S T O R E EXEC.*
5. The program can go to a particular statement and be available for execution there, e.g., *GO TO* **80** EXEC sets the program line counter to **80** for either step-by-step or continuous execution. (If continuous execution is desired at line number **80**, the continue command can be employed, as discussed below.) IF ... THEN can also access a particular line number.
6. Any calculator-keyboard statements can be executed.

A halted program can be executed beginning at any line number by using the continue command; e.g., *CONT* **95** *EXEC* will continue execution starting at line number **95.**

*The initialize command, to be discussed later, is also legitimate.

A program can be run beginning at any line number; e.g., *RUN* **110** *EXEC* will begin program execution at line number **110.**

NOTE

The major difference between *RUN* and *CONT* is that *RUN* initializes all variables in the program and reverts to all normal program modes, while *CONT* neither affects any variables nor affects any current program modes.

PROGRAM STATEMENTS

LET

As in Basic LET A = 6 is a legitimate statement; however, the implied let statement is also allowed. Thus, A = 6 is the same as LET A = 6.

GO TO, GO SUB

The GO TO and GO SUB statements are the same as in Basic; however, each statement has one additional feature called respectively, the computed GO TO and the computed GO SUB. In either case an expression is evaluated and the rounded integer value of the expression is determined; the integer then acts as a pointer to a particular line number from a parameter listing in the statement. Some examples should explain this feature more clearly:

(In all examples, the present value of T will be 2.)

20 GO TO T↑2-3 OF **250, 350, 450**

Since the integer value of the expression is one, the first parameter following OF will be accessed (line number **250**).

80 GO SUB T+2.5 OF **130, 260, 330, 370, 490**

The rounded integer value of the expression is 5; therefore, the subroutine beginning at line number **490**, the fifth parameter following OF, will be accessed. (Decimal values of 0.5 and above are always rounded) to the next higher integer value.)

Any legitimate expression can be used; if the rounded value is either less than 1 or greater than the number of parameters following OF, then the line number following the GO TO or GO SUB statement is executed.

PRINT

The calculator has one feature in the print statement not generally available in BASIC. Alphabetic information in a quote field can be printed in either upper or lower case letters. Printing in lower case is just the opposite of that on a regular typewriter; with shift or shift lock pressed, letters inside the quote field will be printed in lower case (the display, however, will still appear in upper case). If the "at" symbol (@) is required, press *SHIFT RESULT.*

Quote fields in both the format and write statements also have this lower case feature available.

DISPLAY

The display statement performs the same function as a print statement; the difference is that the information appears on the display rather than on a printing device. Thus, when a permanent record of the information is desired, the print statement should be used. The syntax used is DISP.

FORMAT, WRITE

The format statement is a means of structuring program printouts in a specified manner. The write statement defines the variables, constants, etc. that will appear on the printout; it also determines the device to be printed upon and the particular format statement to be followed. The following examples should adequately explain the features in both the format and the write statements:

e.g.,

5 FORMAT F**10.2**

6 WRITE (**15, 5**) **.7**

printout will be ∧∧∧∧∧∧ **0.70** where the "∧" indicates a blank space.

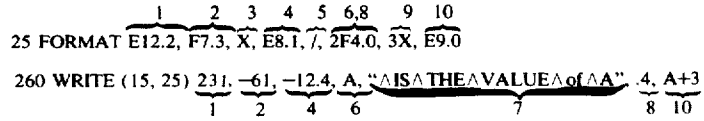
F**10.2** — the F refers to fixed-point format; **10** refers to the total field width reserved for the printout **2** refers to the number of digits to the right of the decimal point. Excess space to the right of the decimal point will be filled with zeros; one space is reserved for the decimal point; two spaces are reserved to the left of the decimal point, one for a digit preceding the decimal point, another for a sign (however, only minus signs are printed).

(**15, 5** — in the write statement, the information within parentheses is required. **15** refers to the printing device to be used (select code **15** refers to the standard printer); **5** refers to the line number of the format statement that is being accessed (in this case line number **5**). The format statement, which is being accessed, can appear anywhere in the program listing.

.7 — in the write statement, the information following the right parenthesis is to be printed according to the designated formats. The value is always right-justified within its field width.

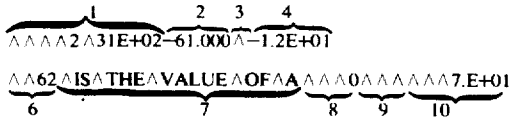
e.g.,

20 A = 62.4



printout will be:

-continued



The format statement, the write statement, and the printout all have individual fields referenced; interlinking fields are represented by corresponding reference numbers.

Since the write statement references the format statement numbered **25**, values in the write statement will be formatted as specified in this format statement.

REFERENCES

1 **E12.2** indicates exponential (floating-point) notation with a field width of 12 and two digits to the right of the decimal point. The printout displays the exact form. Remember the field width must be large enough to include a leading sign, the decimal point and E±XY. Since printouts from the write statement are right justified and since **231** takes up only 8 of the 12 character field width, the four blank spaces are to the left of the value.

2 The value −61 in the write statement totally fills the **F7.3** field, therefore there is no space between this value and the previous value in the printout. Since **F7.3** indicates three digits to the right of the decimal point, zeros are supplied in this case.

3 X indicates a space between values; therefore, the values supplied for the **F7.3** and **E8.1** formats will be separated in the printout by at least one space.

4 The value, −12.4 totally fills up **E8.1**, in fact, the last digit is suppressed since, with this field designation, only one digit can follow the decimal point.

5 / tells the printing device to skip one carriage return to the beginning of the next line.

6 **2F4.0** specifies two consecutive fixed-point formats of **F4.0** to be used for values in the write statement. The first format is for the value of A, which from program line number **20** if 62.4; when a fixed-point format specifies zero digits to the right of the decimal point, the value supplied is rounded to be an integer and the decimal point is suppressed — 62, in this case. Since the carriage return had previously been specified, this value is printed on the beginning of the next output line.

7 Quote fields are printed in the sequence in which they occur. Since this quote field is in the write statement, it is printed immediately after 62.4 is printed in **F4.0** format. Note the space in front of IS; without this space, the printout would read **62IS** instead of **62 IS**. Quote fields can appear in either the format or write statements.

8 The value, .4 is also to be printed in **F4.0** format. The rounded integer value of .4 is 0 is 0 — hence, the printout.

9 **3X** indicates there should be three spaces between the values supplied for the **F4.0** and the **E9.0** formats.

10 Expressions can be specified in write statements. The value of A+3 is 65.4; however, with this format, it will be rounded to **7.E+01**. The decimal point is not suppressed in floating-point notation when zero is specified as the number of digits to the right of the decimal point.

If there are more values presented in the write statement than there are formats in the referenced format statement, the formats will be repeated; e.g.,

10 FORMAT F6.2, E10.2
20 WRITE (15, 10) 18, 21, 19.3, 29.6, .71

printout will be:

^**18.00** ^^**2.10E+01**
^**19.30** ^^**2.96E+01**
^^**0.71**

After the first two values are printed according to the specified formats, the output printer's carriage return is activated, then two more values are printed according to the same two specified formats, etc.

Formatting Rules:

In fixed-point format, F*m.n*, *m* stipulates total field width and *n* stipulates the number of digits to the right of the decimal point. If $n>0$, the minimum field width allowable is $m = n + 3$; e.g., **F4.1** for a value of −.6 would print −0.6, which takes up the total field width of 4. If $n = 0$, the minimum field width is $m = 2$; e.g., **F2.0** for a value of −7 would print −7, thus taking up the allotted field width.

In floating-point format, E*m.n*, *m* and *n* are the same as in fixed-point format. However, the minimum field width allowable is always $m = n + 7$.

The following are all allowed in format statements:

F*m.n* — fixed-point formats;
E*m.n* — floating-point formats (often called exponential or scientific notation);
X — space;
/ — carriage return (for printing device);
**** — quote field;
B — binary format (where write statement could have octal number, the binary equivalent would be output).

All of the above can be duplicated any number of times by leading the symbol with the appropriate number.

The following are all allowed in write statements: constants, variables, expressions, and quote fields. It should be noted that a write statement can be input from keyboard mode; that is, the write statement can reference the line number of a format statement in memory without being in the program itself.

The maximum width of both the fixed-point and the floating-point fields in 9,999. However, the programmer is effectively restricted by the allowable characters per line of the printing device.

The information in the format statement must be separated by commas.

The information in the write statement must be separated either by commas or semicolons; generally it makes no difference. However, one additional feature of the write statement is that it can perform the identi-

cal operations as the print statement; the benefit is the ability of the write statement to select the device to be printed upon. To write on a punched tape photoreader with select code **2**, the write statement could be set up in the following manner:

30 WRITE (**2**, *) A; B; C, D

The * indicates that no format statement is referenced; thus, WRITE acts like a PRINT statement: data will be left justified, semicolons pack the output fields, commas spread out the fields, etc.

P TAPE

As in BASIC, PTAPE causes the computer to read in a program from the punched tape photoreader. If the photoreader select code is **5**, then pressing either *PTAPE # 5* or PTA # **5** will perform this task unless no photoreader is connected to the calculator. If a photoreader is not connected, the calculator will wait till one is hooked up to complete the command. During this time the display will be blank.

During the implementation of this command, lines being loaded into memory have their syntax checked; if a line is in error, it will be rejected — thus, only those lines with correct syntax are loaded into the calculator. To obtain a record of the rejected lines, it is necessary to put the calculator in the print-all mode prior to pressing PTA # **5**; in print-all mode, all rejected lines are printed.

Note

To punch information onto paper tape, use the list command as discussed earlier.

MULTILINE FUNCTIONS

Multiline functions serve the same purpose as single-line functions with the added capability of being able to describe more sophisticated functions. In single-line functions the general form of the defining function is:

statement number DEF FN single letter A to Z (simple variable) = expression

* The simple variable is a dummy variable which indicates where the actual argument of the function is used in the defining expression.

In multiline functions, the general form is the same aside from the equal sign and the expression; for in a multiline function, the expression can be spread out over many statement numbers. Thus, an extremely complicated expression can appear in a more simplified manner; additional flexibility is also gained in that the value of any variable within the expression can be computed for a given value of the argument of the function; e.g.,

```
10   W = .5
20   Y = 2
30   PRINT FNA (3)
40   STOP
50   DEF FNA (X)
60   Z = X ↑ 2 + Y ↑ 2
70   Q = Z + 3
80   D = Q/W
90   RETURN D
100  STOP
```

When this program is run, D will return a value of 32. The return statement returns the result of FNA (X) which, in this case, is equal to D. Line 40, the stop statement is needed to keep the calculator from trying to re-execute lines 50 through 90 after D is printed; without STOP in line 40, an error occurs in line 60 since a second pass beginning at line 50, would be made with X undefined.

Any variable evaluated in the expression can be returned by the return statement, and multiple return statements are allowed; e.g.,

-continued

```
20   INPUT Y                  110  Q = Z ↑ 2
30   WRITE (15,900) FNG (Y)   120  IF Q ≤ 100 THEN 150
40   END                      130  PRINT "Q ="
50   DEF FNG (Y)              140  RETURN Q
60   Y = Y + 1                150  PRINT "Z IS"
70   Z = Y ↑ 2*X              160  RETURN Z
80   IF Z ≤ 100 THEN 110      900  FORMAT F12.1
90   PRINT "Z ="              1000 END
```

In this example, a variable can be returned from three different lines (**100, 140, 160**) depending on the initial value of Y; Z can be returned from both lines **100** and **160** if the value of Y meets certain criteria.

Caution must be taken as to the placement of the statement that calls the function (in both examples, statement **30**), and it is generally advisable to put a stop statement immediately after this statement (as in line **40** of both examples); otherwise, an undesirable loop may develop.

If correctly entered, the function of a function can be evaluated.

STOP, END

STOP was previously discussed, with emphasis on its program debugging capabilities. Now it will be discussed as a program statement, emphasizing the differences between it and the end statement.

When the program encounters a stop statement, it halts and is waiting; if *CONT EXEC* is then pressed, the program will continue with the statement following the STOP. This is not true if an end statement is encountered; the program will halt, but if *CONT EXEC* is pressed, the calculator will revert back to the lowest-numbered statement in memory. Therefore, the stop statement should be used between stacked programs that are to be run sequentially. When STOP is used in this manner, the values of simple variables can be passed from program to program.

A program should be terminated by encountering either a STOP or an END. The highest-numbered program statement need not be an end statement.

INPUTTING DATA

Program data can be input in three ways: the input statement, the read and data statements and the initialize command. The input statement and the read and data statements are thoroughly discussed below.

INITIALIZE

The INITIALIZE key, when pressed, allocates storage space in memory for array variables. After the required

data is keyed in, the program can be executed by pressing *CONT EXEC;* remember — *RUN EXEC* erases the values of all variables, thereby requiring all variables to be defined in the program itself.

Simple variables can always be input in keyboard mode without using the INITIALIZE key, as long as *CONT EXEC* is pressed to run the program. Since array variables can be input in keyboard mode by using the initialize command, it is not necessary to define any variables in the program itself. It is still necessary, however, to identify arrays in either a dimension or a common statement.

USER—DEFINABLE KEYS

There are ten User-Definable Keys (UDK) in the upper left-hand block of the keyboard. There are, however, effectively 20 accessible UDK's since each key can be accessed normally or with the shift key held down.

To enter UDK mode, press *FETCH (particular UDK);* the display will then read, KEY indicating the mode. To exit from UDK mode, press *CLEAR E N D EXEC;* the scratch command can also be used to exit from UDK mode — this command will, of course, erase certain information in the process. UDK mode is automatically exited when certain sequences are followed; these cases will be discussed later.

The user-definable keys can be used effectively in three ways:

1. to represent text (where text can be used as a typing aid);
2. to represent functions (where different values can be passed to the function);
3. To represent programs.

REPRESENTING TEXT

If a key represents text, merely pressing the key will immediately display the text without erasing anything that was previously on the display. Thus, commonly used words and phrases can be put on keys to serve as typing aids. Text can be put on a key in the following manner.

First, access a key by pressing *FETCH (particular UDK).* Then, press * followed by a *character string* and finally *EOL.* Besides inputting the character string, pressing *EOL* in this sequence takes the user out of UDK mode. Any time the programmer wishes to use a character string, he must press the key into which the desired character string was input.

*The maximum length for the character string (including the *) is 80 characters.

If, for example, a key was accessed by *FETCH (particlar UDK);* then * *FORMAT F***10.2.***,X, E***10.1** *EOL* was input. If subsequently program line number **60** needed this format, pressing **60** *(particular UDK EOL* will put line number **60** into memory with the required format. A typing-aid key can be used as an immediate execute command if an * is placed both in front of the text and following the text;

e.g.,

FETCH (UDK) LOADDATA # **4, 6,** B * *EOL*

This command will be immediately executed whenever the UDK is pressed.

To use a key that has text, merely press the key. Pressing *FETCH (particular UDK)* will display the * with the text; however, text can be edited if the fetch command is used. Pressing *FETCH* and then * will erase the old character string and then wait for new text to be input.

REPRESENTING FUNCTIONS

A user definable key can be used to represent functions —either single or multiline. In either case, after the key is accessed, the function must be preceded by a line number — input either manually or automatically.

After a key has been accessed, the following function could be input:

10 DEF FNA (X) = 7 * X =**3** EOL

Whenever a value is to be passed to X (the argument of the function), first press the appropriate key; the display will read FNA. Then key in the appropiate value of the argument, which can be either a constant or an expression (e.g., 20), and press *EXEC;* the value of the function will then be displayed (in this case, 137). The same result could have been achieved by using the fetch command; however, FNA would not appear automatically on the display; it would have to be keyed in along with the argument; e.g., *FETCH (UDK) FNA* **20** *EXEC* would also display 137.

If a multiline function, DEF FND (Z), has been input, pressing the appropriate UDK causes FNB to be displayed; as before, passing a value to the argument, Z, and then pressing *EXEC* will compute and display the value of the function.

Functions in mainline memory and in a UDK can be called, regardless of the current operating mode.

If a function in the calculator is defined in more than one place, the first function found with the designated name will be accessed. If the user is in UDK mode, the calculator will search for the function in the following order:

1. The current UDK program will be checked.
2. The first line of each key (in the order defined) will be checked.
3. Mainline memory will be checked.

If the user is not in UDK mode, the order of the search will be steps **3** and **2,** respectively.

REPRESENTING PROGRAMS

A UDK can be used to represent an entire program. Programming rules in UDK mode are consistent with those discussed above. There is one restriction, however; if a common statement is used, its size must be less than or equal to the size of the common statement in the mainline program — for there is only one common area allocated to memory.

To run a program that is represented in a UDK, it is advisable to press *RUN (particular UDK) or FETCH (UDK),* then *INIT (particular UDK).* The program can be continued merely by pressing the (particular UDK), but if there are array variables in the program, pressi g only the key will cause these variables to be undefined (similar in this respect to the continue command, which neither destroys the old symbol table nor builds a new one). After executing the program, the calculator will exit from UDK mode.

Programs represented on a UDK generally use only simple variables for the obvious ease of handling.

To list program lines on a particular UDK, press: *LIST (particular UDK);* pressing *LIST EXEC* in the UDK mode will list the program lines on the key currently being accessed. To selectively list particular lines on the UDK, it is first necessary to *FETCH* a key; then use the list command as discussed below.

To load a program from a cassette file onto one particular key, first *FETCH* the key, then give the load command followed by *EXEC*. Program lines on a particular key can be stored onto the cassette in the same manner; text (as a typing aid), however, cannot be stored in this manner.

STORE KEY and LOAD KEY can be used for any UDK regardless of the information on the key. The use of these keys is discussed below.

STATEMENTS

This is a **BASIC statement:**
/10 INPUT A,B,C,D,E

Comments

A *statement* contains a maximum of 80 characters

A *statement* may also be called a line.

STATEMENT NUMBERS

Each BASIC statement begins with a **statement number** (in this example, 20):

20 LET S=(A+B+C+D+E)/5

Comments

The number is called a *statement number* or a *line number.*

The statement number is chosen by you, the programmer. It may be any integer from 1 to 9999 inclusive.

Each statement has a unique *statement number.* The computer uses the numbers to keep the statements in order.

Statements may be entered in any order; they are usually numbered by fives or tens so that additional statements can be easily inserted. The computer keeps them in numerical order no matter how they are entered. For example, if statements are input in the sequence 30,10,20; the computer arranges them in the order: 10,20,30.

INSTRUCTIONS

The statement then gives an **instruction** to the computer (in this example, PRINT):

30 PRINT S

Comments

Instructions are sometimes called *statement types* because they identify a type of statement. For example, the statement above is a "print" statement.

OPERANDS

If the instruction requires further details, **operands** (numeric details) are supplied (In this example, 10; on the previous page, "S"):

40 GO to 10

Comments

The *operands* specify what the instruction acts upon; for example, what is PRINTed, or where to GO.

A PROGRAM

The sequence of BASIC statements given on the previous pages is called a **program.** The last statement in a program, as shown here, is an END statement.

```
10 INPUT A,B,C,D,E
20 LET S=(A+B+C+D+E)/5
30 PRINT S
40 GO TO 10
50 END
```

Comments

The last (highest numbered) statement in a program must be an END statement.

The END statement informs the computer that the program is finished.

FREE-FORM LANGUAGE

BASIC is a "free format" language--the computer ignores extra blank spaces in a statement. For example, these three statements are equivalent:

```
30 PRINT S
30 PRINT   S
30 PRINTS
```

Comments

When possible, leave a space between words and numbers in a statement. This makes a program easier for people to read.

TERM: SIMPLE VARIABLE

Defined in Basic as:	A letter (from A to Z); or a letter immediately followed by a digit (from 0 to 9).
Examples:	A0 B M5 C2 Z9 D

Comments

Variables are used to represent numeric values. For instance, in the statement:

10 LET M5 = 96.7

M5 is a variable; 96.7 is the value of the variable **M5.**

There is one other type of variable in BASIC, the array (subscripted) variable; its use is explained in Section IV.

TERM: NUMBER

Defined in Basic as: A decimal number (the sign is optional) between 1E-99 and 9.999999999999E+99

Zero is included in this range.

Examples:

−10008	5	3.14159	10E+37
126.257	0	10 E37	10E−37
16.01	.06784	−10 E37	1.0E+2

TERM: E NOTATION

Defined in Basic as: A means of expressing numbers having more than six decimal digits, in the form of a decimal number raised to some power of 10 .

Examples: 1.00000E+06 is equal to 1,000,000 and is read: "1 times 10 to the sixth power" (1×10^6).
1.02000E+04 is equal to 10,200
1.02000E−04 is equal to .000102

Comments

E notation is used to print numbers having more than six significant digits. It may also be used for input of any number.

When entering numbers in E notation, leading and trailing zeroes may be omitted from the number; the + sign and leading zeroes may be omitted from the exponent.

The precision of numbers is 12 decimals digits

TERM EXPRESSION

Defined in Basic as: A combination of variables, constants and operators which evaluates to a numeric value.

Examples: (P + 5)/27
(where P has previously been assigned a numeric value.)
Q − (N + 4)
(where Q and N have previously been assigned numeric values.)

TERM ARITHMETIC EVALUATION

Defined in Basic as: The process of calculating the value of an expression.

The ASSIGNMENT OPERATOR

Symbol: =

Examples:
```
10 LET A = B2 = C = 0
20 LET A9 = C5
30 LET Y = (N-(R+5))/T
40 LET N5 = A + B2
50 LET P5 = P6=P7=A=B=98.6
```

General Form: **LET variable = expression**

Purpose

Assigns an arithmetic or logical value to a variable.

Comments

When used as an assignment operator, = is read "takes the value of," rather than "equals". It is, therefore, possible to use assignment statements such as:

```
LET X = X+2
```

This is interpreted by BASIC as: LET X take the value of (the present value of) X, plus two.

Several assignments may be made in the same statement, as in statements **10** and **50** above.

See Section V, Logical Operations for a description of logical assignments.

RELATIONAL OPERATORS

Symbols: = # <> > < >= <=

Examples:
```
100 IF A=B THEN 900
110 IF A+B >C THEN 910
120 IF A+B < C+E THEN 920
130 IF C>=D*E THEN 930
140 IF C9<= G*H THEN 940
150 IF P2#C9 THEN 950
160 IF J <> K THEN 950
```

Purpose

Determines the logical relationship between two expressions, as

equality:	=
inequality:	# or <>
greater than:	>
less than:	<
greater than or equal to:	>=
less than or equal to:	<=

Comments

Note: It is not necessary for the novice to understand the nature of logical evaluation of relational operators, at this point. The comments below are for the experienced programmer.

Expressions using relational operators are logically evaluated, and assigned a value of true or false (the numeric value is 1 for true, and 0 for false).

When the = symbol is used in such a way that it might have either an assignment or a relational function, BASIC assumes it is an assignment operator. For a description of the assignment statement using logical operators, see Section V, Logical Operations.

ARITHMETIC OPERATORS

Symbols: ↑ * / + −

Examples:
```
40 LET N1 = X-5
50 LET C2 = N↑3
```

-continued

ARITHMETIC OPERATORS

```
60 LET A = (B-C)/4

70 LET X = ((P↑2)-(Y*X))/N+Q
```

Purpose

Represents an arithmetic operation, as:

```
exponentiate: ↑
multiply: *
divide: /
add: +
subtract: -
```

Comments

The "-" symbol is also used as a sign for negative numbers. It is good practice to group arithmetic operations with parentheses when unsure of the exact order of precedence. The order of precedence (hierarchy) is:

```
↑
* /
+ -
```

with ↑ having the highest priority. Operators on the same level of priority are acted upon from left to right in a statement. See Order of Precedence in this Section for examples.

The symbols + and - are also used to indicate unary plus and unary minus. For example, negative numbers may be expressed in a statement without using parenthesis:

```
10 LET A1 = -B

20 LET C2 = D +-E

30 LET B5 = B --C
```

See Order of Precedence in this section for examples or how unary + and unary - are interpreted.

THE AND OPERATOR

```
Symbol:     AND
Examples:   60 IF A9<B1 AND C # 5 THEN 100
            70 IF T7   T AND J=27 THEN 150
            80 IF P1 AND R>1 AND N AND V2 THEN 10
            90 PRINT X AND Y
```

Purpose

Forms a logical conjunction between two expressions. If both are true, the conjunction is true; if one or both are false, the conjunction is false.

Note: It is not necessary for the novice to understand how this operator works. The comments below are for experienced programmers.

Comments

The numeric value of true is 1, of false is 0.

All non-zero values are true. For example, statement **90** would print either a 0 or a 1 (the logical value of the expression X AND Y) rather than the actual numeric values of X and Y.

Control is transferred in an IF statement using AND, only when all parts of the AND conjunction are true. For instance, example statement **80** requires four true conditions before control is transferred to statement **10.**

See Section V, Logical Operations for a more complete description of logical evaluation.

THE OR OPERATOR

```
Symbol:     OR
Examples:   100 IF A>1 OR B<5 THEN 500
            100 PRINT C OR D
            120 LET D = X OR Y
            130 IF (X AND Y) OR (P AND Q) THEN 600
```

Purpose

Forms the logical disjunction of two expressions. If either or both of the expressions are true, the OR disjunction is true; if both expressions are false, the OR disjunction is false.

NOTE: It is not necessary for the novice to understand how this operator works. The comments below are for experienced programmers.

Comments

The numeric values are: true = 1, false = 0.

All non-zero values are true; all zero values are false.

Control is transferred in an IF statement using OR, when either or both of the two expressions evaluate to true.

See Section V, Logical Operations for a more complete description of logical evaluation.

THE NOT OPERATOR

```
Symbol:     NOT
Examples:   30 LET X = Y = 0
            35 IF NOT A THEN 300
            45 IF (NOT C) AND A THEN 400
            55 LET B5 = NOT P
            65 PRINT NOT (X AND Y)
            70 IF NOT (A=B) THEN 500
```

Purpose

Logically evaluates the complement of a given expression.

NOTE: It is not necessary for the novice to understand how this operator works. The comments below are intended for experienced programmers.

Comments

If A = 0, then NOT A = 1; if A has a non-zero value, NOT A = 0.

The numeric values are: true = 1, false = 0; for example, statement **65** above would print 1, since the expression NOT (X AND Y) is true.

Note that the logical spsecifications of an expression may be changed by evaluating the complement. In statement **35** above, if A equals zero, the evaluation would be true (1); since A has a numeric value of 0, it has a logical value of false, making NOT A true.

See Section V, Logical Operations for a more complete description of logical evaluation.

ORDER OF PRECEDENCE
The order of performing operations is:

↑ **highest precedence**

NOT unary + unary
* /
+ −
Relational Operators
AND
OR **lowest precedence**

Comments

If two operators are on the same level, the order of execution is left to right, for example:

5 + 6*7 is evaluated as: 5 + (6×7)
7/14*2/5 is evaluated as: $\frac{(7/14)\times 2}{5}$

Parentheses override the order of precedence in all cases, for example:

5 + (6×3) is evaluated as: **5 + 18**

and

3 + (6+(2↑2)) is evaluated as: **3 + (6+4)**

Unary + and − may be used; the parentheses are assumed by BASIC. For example:

A + + B is interpreted: A + (+B)

C − + D −5 is interpreted: C − (+D)−5

Leading unary + signs are omitted from output by BASIC, but remain in program listings.

STATEMENTS

Statements are instructions to the calculator. They are contained in numbered lines within a program, and execute in the order of their line numbers. Statements cannot be executed without running a program. They tell the calculator what to do while a program is running.

Here are some examples mentioned in Section I:

LET
PRINT

INPUT

Do not attempt to memorize every detail in the Statements subsection; there is too much material to master in a single session. By experimenting with the sample programs and attempting to write your own programs, you will learn more quickly than by memorizing.

THE LET STATEMENT

Examples:
10 LET A = 5.02
20 LET X = Y7 = Z = 0
30 LET B9 = 5* (X↑2)
40 LET D = (3*C2↑N)/(A*(N/2))

General Form:
statement number LET variable = number or expression or variable . . .

Purpose

Used to assign or specify the value of a variable. The value may be an expression, a number, or a variable.

Comments

The assignment statement must contain:
1. A statement number,
2. LET is optional
3. The variable to be assigned a value (for example, **B9** in statement **30** above),
4. The assignment operator, an = sign,
5. The number, expression or variable to be assigned to the variable (for example, **5*(X↑2)** in statement **30** above).

Statement **20** in the example above shows the use of an assignment to give the same value (0) to several variables. This is a useful feature for initializing variables in the beginning of a program.

REM

Examples:
10 REM--THIS IS AN EXAMPLE
20 REM: OF REM STATEMENTS
30 REM-----////*****!!!!!
40 REM. STATEMENTS ARE NOT EXECUTED BY BASIC

General form: **statement number REM any remark or series of characters**

Purpose

Allows insertion of a line of remarks or comment in the listing of a program.

Comments

Must be preceded by a line number. Any series of characters may follow REM.

REM lines are part of a BASIC program and are printed when the program is listed or punched; however, they are ignored when the program is executing.

Remarks are easier to read if REM is followed by a punctuation mark, as in the example statements.

PRINT

```
This sample program gives a variety of examples of the PRINT statement.
The results are shown below.
 10 LET A=B=C=10
 20 LET D1=E9=20
 30 PRINT A,B,C,D1,E9
 40 PRINT A/B,B/C/D1+E9
 50 PRINT "NOTE THE POWER TO EVALUATE AN EXPRESSION AND PRINT THE"
 60 PRINT "VALUE IN THE SAME STATEMENT."
 70 PRINT
```

-continued

PRINT

```
 80 PRINT
 90 REM* "PRINT" WITH NO OPERAND CAUSES THE TELEPRINTER TO SKIP A LINE.
100 PRINT "'A' DIVIDED BY 'E9' =";A/E9
110 PRINT "11111", "22222", "33333", "44444", "55555", "66666"
120 PRINT "11111"; "22222"; "33333"; "44444"; "55555"; "66666"
130 END
```

RESULTS

```
RUN
10    10          10    20    20
 1    20.05
NOTE THE POWER TO EVALUATE AN EXPRESSION AND PRINT THE
VALUE IN THE SAME STATEMENT.
'A' DIVIDED BY 'E9' = .5
11111     22222     33333     44444     55555
66666
1111122222333334444455555566666
```

NOTE: The "," and ";" used in statements 110 and 120 have very different effects on the format.

General Form:

statement number PRINT expression , expression , . . .
or
statement number PRINT "any text" ; expression ; . . .
or
statement number PRINT "text" ; expression ; "text" , "text" , . . .
or
statement number PRINT any combination of text and/or expressions
or
statement number PRINT

Purpose

Causes the *expressions* or "*text*" to be output to the Printer

Causes the printer to skip a line when used without an operand.

Comments

Note the effects of, and; on the output of the sample program. If a comma is used to separate PRINT operands, five fields are printed per printer line. If semicolon is used, up to twelve packed numeric fields are output per printer line (72 characters). Text in quotes is printed literally.

NOTE: A variable name is considered as a simple expression by BASIC. For example, a statement for the first general form shown above might be:

100 PRINT A1, B2, C3

or

110 PRINT A, Z, X, T9

where the variables represent numeric expressions.

Remember that variable values must be defined in an assignment, INPUT, READ or FOR statement before being used in a PRINT statement.

Ending a PRINT statement with a semicolon causes the output to be printed on the same line, rather than generating a *return linefeed* after the statement is executed. For example, the sequence:

```
20 LET X = 1

30 PRINT X;

40 LET X=X+1

50 GO TO 30
```

produces output in this format:

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

Similarly, ending a PRINT statement with a comma causes output to fill all five fields on a line before moving to the next line. The trailing comma in statement **30** in the sequence:

```
20 LET X = 1

30 PRINT X,

40 LET X=X+1

50 GO TO 30
```

produces output in this format:

```
 1       2       3       4       5
 6       7       8       9       10
 11      12      13      14      15
```

A PRINT statement without an operand (statements **70** and **80** in the sample program) generates a *return linefeed.*

GO TO AND MULTIBRANCH GO TO

Examples:

```
10 LET X = 20

.
.

40 GO TO X+Y OF 410,420,430
50 GOTO 100
80 GOTO 10
90 GO TO N OF 100,150,180,190
```

-continued

GO TO AND MULTIBRANCH GO TO

General Form:
statement number GO TO statement number
statement number GO TO expression OF sequence of statement numbers

Purpose

GO TO transfers control to the statement specified.

GO TO *expression*...rounds the *expression* to an integer *n* and transfers control to the *n*th statement number following OF.

Comments

GO TO may be written: GOTO or GO TO.

Must be followed by the statement number to which control is transferred, or *expression* OF, and a sequence of statement numbers.

GO TO overrides the normal execution sequence of statements in a program.

If there is no statement number corresponding to the value of the *expression*, the GO TO is ignored.

Useful for repeating a task infinitely, or jumping (GOing TO) another part of a program if certain conditions are present.

GO TO should not be used to enter FOR-NEXT loops; doing so may produce unpredictable results or fatal errors.

GOSUB...RETURN

Example:
```
50 READ A2
60 IF A2<100 THEN 80
70 GOSUB 400
.
.
380 STOP (STOP frequently precedes the first statement of
a subroutine, to prevent accidental entry.)
390 REM--THIS SUBROUTINE ASKS FOR A 1 OR 0 REPLY.
400 PRINT "A2 IS>100"
410 PRINT "DO YOU WANT TO CONTINUE";
420 INPUT N
430 IF N # 0 THEN 450
440 LET A2 = 0
450 RETURN
.
.
600 END
```

General Form:
statement number GOSUB statement number starting subroutine
.
.
.
statement number RETURN

Purpose

GOSUB transfers control to the specified statement number.

RETURN transfers control to the statement following the GOSUB statement which transferred control.

GOSUB...RETURN eliminates the need to repeat frequently used groups of statements in a program.

MULTIBRANCH GOSUB

Examples:
```
20 GOSUB 3 OF 100,200,300,400,500
60 GOSUB N+1 OF 200,210,220
70 GOSUB N OF 80,180,280,380,480,580
```

General Form:
statement number GOSUB expression OF sequence of statement numbers . . .

Purpose

GOSUB *espression* rounds the *expression* to an integer *n* and transfers control to the *n*th statement number following OF.

Comments

Subroutines should be exited only with a RETURN statement.

The *expression* indicates which of the specified subroutines will be executed. For example, statement **20,** above transfers control to the subroutine beginning with statement **300.** The *expression* specifies which statement in the sequence of five statements is used as the starting one in the subroutine.

The *expression* is evaluated as an integer. Non-integer values are rounded to the nearest integer.

If the *expression* evaluates to a number greater than the number of statements specified, or less than 1, the GOSUB is ignored.

Statement numbers in the sequence following OF must be separated by commas.

IF...THEN

Sample Program:
```
10 LET N = 10
```

-continued

```
IF...THEN
20 READ X
30 IF X <=N THEN 60
40 PRINT "X IS OVER"; N
50 GO TO 100
60 PRINT "X IS LESS THAN OR EQUAL TO"; N
70 GO TO 20
80 STOP
.
.
.
```

General Form:
statement number IF expression relational op expression THEN statement number

Purpose

Transfers control to a specified statement if a specified condition is true.

Comments

Sometimes described as a conditional transfer; GO TO is implied by IF...THEN, if the condition is true. In the example above, if X<=10, the message in statement **60** is printed (statement **60** is executed).

Since numbers are not always represented exactly in the computer, the = operator should be used carefully in IF...THEN statements. Limits, such as <=,>=, etc. should be used in an IF expression, rather than =, whenever possible.

If the specified condition for transfer is not true, the program will continue executing in sequence. In the example above, if X>10, the message in statement **40** prints.

The relational operator is optional in logical evaluations.

See Section V, Logical Operations, for a more complete description of logical evaluation.

Purpose

Allows controlled repetition of a group of statements within a program.

Comments

Initial value, *final value* and *step value* may be any expression.

STEP and *step value* are optional; if no step value is specified, the computer will automatically increment by one each time it executes the loop.

How the loop works:
The simple variable is assigned the value of the *initial value;* the value of the simple variable is increased by 1 (or by the *step value*) each time the loop executes. When the value of the *simple variable* passes the *final value*, control is transferred to the statement following the NEXT statement.

The *initial*, *final*, and *step values* are all evaluated upon entry to the loop and remain unchanged after entry. For example,

```
FOR I = 1 TO I + 5
```

goes from 1 to 6; that is, the *final value* does not move as I increases with each pass through the loop.

```
FOR...NEXT
Examples:  100 FOR P1 = 1 TO 5
           110 FOR Q1 = N TO X
           120 FOR R2 = N TO X STEP 2.5
           130 FOR S = 1 TO X STEP Y
           140 NEXT S
           150 NEXT R2
           160 NEXT Q1
           170 NEXT P1
Sample Program - Variable Number Of Loops
 40 PRINT "HOW MANY TIMES DO YOU WAT TO LOOP";
 50 INPUT A
 60 FOR J = 1 TO A
 70 PRINT "THIS IS LOOP"; J
 80 READ N1, N2, N3
 90 PRINT "THESE DATA ITEMS WERE READ:" N1; N2; N3
100 PRINT "SUM ="; (N1+N2+N3)
110 NEXT J
120 DATA 5, 6, 7, 8, 9, 10, 11, 12
130 DATA 13, 14, 15, 16, 17, 18, 19, 20, 21
140 DATA 22, 23, 24, 25, 26, 27, 28, 29, 30
150 DATA 31, 32, 33, 34
160 END
```

General Form:
statement number FOR simple variable = initial value TO final value
or
statement no. FOR simple var. = initial value TO final value STEP step value

```
.
.
```

statement number NEXT simple variable
NOTE: The same simple variable must be used in both the FOR and NEXT statements of a loop.

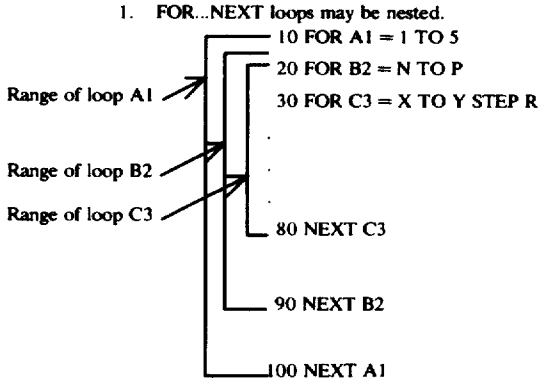
For further details on the STEP feature, see FOR...NEXT with STEP in Section III.

Try running the sample program if you are not sure what happens when FOR...NEXT loops are used in a program.

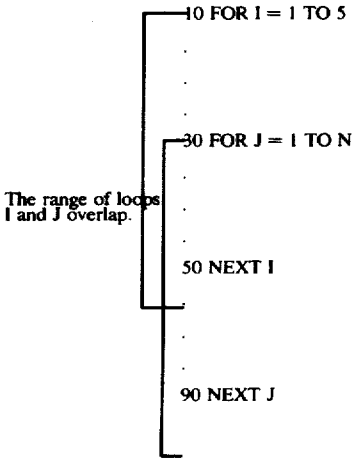
NESTING FOR...NEXT LOOPS

Several FOR...NEXT loops may be used in the same program; they may also be nested (placed inside one another). There are two important features of FOR...NEXT loops:

1. FOR...NEXT loops may be nested.



2. The range of FOR...NEXT loops may not overlap. The loops in the example above are nested correctly. This example shows improper nesting.



READ; DATA AND RESTORE
Sample Program using READ and DATA

```
15 FOR I=1 TO 5
20 READ A
40 LET X=A↑2
```

-continued
READ; DATA AND RESTORE

```
45 PRINT A;"SQUARED =";X
50 NEXT I
55 DATA 5.24,6.75,30.8,72.65,89.72
60 END
```

Each data item may be read only once in this program. TSB keeps track of data with a "pointer." When the first READ statement is encountered, the pointer indicates that the first item in the first DATA statement is to be read; the pointer is then moved to the second item of data, and so on.

In this example, after the loop has executed five times, the pointer remains at the end of the data list. To reread the data, it is necessary to react the pointer. A RESTORE statement moves the pointer back to the first data item.

Sample Program Using READ, DATA and RESTORE

```
20 FOR I=1 TO 5
30 READ A
40 LET X=A↑2
50 PRINT A; "SQUARED =";X
```

-continued

```
 60 NEXT I
 80 RESTORE
100 FOR J=1 TO 5
110 READ B
120 LET Y=B↑4
130 PRINT B; "TO READ FOR FOURTH POWER =";Y
140 NEXT J
150 DATA 5.24;6.75;30.8;72.65;89.72
160 END
```

General Form:

statement number READ variable , variable , . . .
statement number DATA number or string , number or string , . . .
statement number RESTORE
statement number RESTORE statement number

Purpose

The READ statement instructs TSB to read an item from a DATA statement.

The DATA statement is used for specifying data in a program. The data is read in sequence from first to last DATA statements, and from left to right within the DATA statement.

The RESTORE statement resets the pointer to the first data item, allowing data to be re-read.

RESTORE followed by a statement number resets the pointer to the first data item, beginning at the specified statement.

Comments

READ statements require at least one DATA statement in the same program.

Items in a DATA statement must be separated by commas. String and numeric data may be mixed.

DATA statements may be placed anywhere in a program. The data items will be read in sequence as required.

DATA statements do not execute; they merely specify data.

The RUN command automatically sets the pointer to the first data item.

If you are not sure of the effects of READ, DATA, and RESTORE, try running the sample programs.

WAIT

Example:
```
900 WAIT (1000)
990 WAIT (3000)
```
General Form: **statement number WAIT (expression max. value of 32767)**

Purpose

Introduces delays into a program. WAIT causes the program to wait the specified number of milliseconds (maximum 32,767 milliseconds) before continuing execution.

Comments

The time delay produced by WAIT is not precisely the number of milliseconds specified because there is no provision to account for time elapsed during calculation or terminal-computer communication.

One millisecond = 1/1000 second.

TERM: ROUTINE

Defined in Basic as: A sequence of program statements which produces a certain result.

Purpose

Routines are used for frequently performed operations, saving the programmer the work of defining an operation each time he uses it, and saving computer memory space.

Comments

A routine may also be called a program, subroutine, or sub-program.

The task performed by a routine is defined by the programmer.

TERM: STRING

Defined in Basic as: 0 to 255 printer characters enclosed by quotation marks (one line on a teleprinter terminal).

Comments

Sample strings:
```
ANY CHARACTERS!?*/---
TEXT 1234567...
```

Quotation marks may not be used within a string. Strings are used only in PRINT statements.

The statement number PRINT, and quotation marks are not included in the 65 character count. Each statement may contain up to 72 characters. Maximum string length is 72 characters minus 6 characters for PRINT, two for the quotation marks, and the number of characters in the statement number.

TERM: FUNCTION

Defined in Basic as: The mathematical relationship between two variables (X and Y, for example) such that for each value of X there is one and only one value of Y.

Comments

The independent variable in a function is called an argument; the dependent variable is the function value. For instance, if X is the argument, the function value is the square root of X, and Y takes the value of the function.

TERM: ARRAY OR MATRIX

Defined in Basic as: An ordered collection of numeric data (numbers).

Comments

Arrays are divided into columns (vertical) and rows (horizontal):

C ROWS
O
L
U
M
N
S

Arrays may have one or two dimensions. For example,

1.0
2.1
3.2
4.3

is a one-dimensional array, while

6, 5, 4
3, 2, 1
0, 9, 8

is a two-dimensional array.

Array elements are referenced by their row and column position For instance, if the two examples above were arrays A and Z respectively, 2.1 would be A(2); similarly, 0 would be Z(3,1). The references to array elements are called subscripts, and set apart with parentheses. For example, P(1,5) references the fifth element of the first row of array P; 1 and 5 are the subscripts. In X(M,N) M and N are the subscripts.

TERM: WORD

Defined in Basic as: The amount of computer memory space occupied by two teleprinter characters.

Comments

Numbers require two words of memory space when stored as numbers. When used within a string, numbers require one-half word of space per character in the number.

SUBROUTINES AND FUNCTIONS

The following pages explain BASIC features useful for repetitive operations — subroutines, programmer-defined functions and standard functions.

The programmer-defined features, such as GOSUB, FOR...NEXT with STEP, and DEF FN become more useful as the user gains experience and learns to use them as shortcuts.

Standard mathematical and trigonometric functions are convenient timesavers for programmers at any level. They are treated as numeric expressions by BASIC.

FOR...NEXT WITH STEP

Examples:
```
20 FOR 15 = 1 TO 20 STEP 2
40 FOR N2 = 0 TO -10 STEP -2
80 FOR P = 1 TO N STEP X5

90 FOR X = N TO W STEP (N↑2-V)
```

General Form:

statement no. FOR simple var. = expression TO expression STEP expression

Purpose

Allows the user to specify the size of the increment of the FOR variable.

Comments

The step size need not be an integer. For instance,

```
100 FOR N = 1 TO 2 STEP .01
```

is a valud statement which produces approximately 100 loop executions, incrementing N by .01 each time.

A step size of 1 is assumed if STEP is omitted from a FOR statement.

A negative step size may be used, as shown in statement **40** above.

GENERAL MATHEMATICAL FUNCTIONS

Examples:
```
642 PRINT EXP(N); ABS(N)
652 IF RND (0)>=.5 THEN 900
662 IF INT (R) # 5 THEN 910
672 PRINT SQR (X); LOG (X)
```

General Form: The general mathematical functions may be used as expressions, or as parts of an expression.

Purpose

Facilitates the use of common mathematical functions by pre-defining them as:

ABS (*expression*) the absolute value of the expression;
EXP (*expression*) the constant *e* raised to the power of the expression value (in statement **642** above, e↑N)
INT (*expression*) the largest integer ≤ the expression;
LOG (*expression*) the logarithm of the positively valued expression to the base *e;*
RND (*expression*) a random number between 1 and 0; the expression is a dummy argument;
SQR (*expression*) the square root of the positively valued expression.

Comments

The RND function is restartable; the sequence of random numbers using RND is identical each time a program is RUN.

TRIGONOMETRIC FUNCTIONS

Examples:
```
500 PRINT SIN(X); COS(Y)
510 PRINT 3*SIN(B); TAN (C2)
520 PRINT ATN (22.3)
530 IF SIN (A2) <1 THEN 800
540 IF SIN (B3) = 1 AND SIN(X) <1 THEN 90
```

Purpose

Facilitates the use of common trigonometric functions by pre-defining them, as:

SIN (*expression*) the sine of the expression
COS (*expression*) the cosine of the expression
TAN (*expression*) the tangent of the expression
ATN (*expression*) the arctangent of the expression

Comments

The function is of the value of the expression (the value in parentheses, also called the argument).

The trigonometric functions may be used as expressions or parts of an expression.

The angle of the trigonometric functions can be specified as radians, degrees, or grads by executing a RAD, DEG, or GRAD statement. The calculator assumes radians if not specified.

THE TAB AND SGN FUNCTIONS

Examples:
```
500 IF SGN (X) # 0 THEN 800
510 LET Y = SGN(X)
520 PRINT TAB (5); A2; TAB (20)"TEXT"
530 PRINT TAB (N),X,Y,Z2
540 PRINT TAB (X+2) "HEADING"; R5
```

General Form: The TAB and SGN may be used as expressions, or parts of an expression. The function forms are:

TAB (expression indicating number of spaces to be moved)
SGN (expression)

Purpose

TAB (*expression*) is used only in a PRINT statement, and causes the terminal typeface to move to the space number specified by the expression (**0** to **71**). The *expression* value after TAB is rounded to the nearest integer. Expression values greater that **71** cause a *return linefeed* to be generated.

SGN (*expression*) returns a 1 if the expression is greater than 0, returns a 0 if the expression equals 0, returns a −1 if the expression is less than 0.

SECTION IV

MATRICES

This section explains matrix manipulation. It is intended to show the matrix capabilities of BASIC and assumes that the programmer has some knowledge of matrix theory.

TERM: MATRIX (ARRAY)

Defined in Basic as: An ordered collection of numeric data (numbers).

Matrix elements are referenced by subscripts following the matrix variable, indicating the row and column of the element. For example, if matrix A is:

1 2 3
4 5 6
7 8 9

the element **5** is referenced by A(2,2); likewise, **8** is A(3,2).

See Section III, Vocabulary for a more complete description of matrices

DIM

Examples:
```
110 DIM A (50), B(20,20)
120 DIM Z (5,20)
130 DIM S (5,25)
140 DIM R (4,4)
```

General Form:
statement number DIM matrix variable (integer) . . .
or
statement number DIM matrix variable (integer , integer) . . .

Purpose

Reserves working space in memory for a matrix.

The maximum integer value (matrix bound) is 255.

Comments

The integers refer to the number of matrix elements if only one dimension is supplied, or to the number of rows and columns respectively, if two dimensions are given.

A matrix (array) variable is any single letter from A to Z.

Arrays not mentioned in a DIM statement are assumed to have 10 elements if one-dimensional, or 10 rows and columns if two-dimensional.

The working size of a matrix may be smaller than its physical size. For example, an array declared 9 × 9 in a DIM statement may be used to store fewer than 81 elements; the DIM statement supplies only an upper bound on the number of elements.

The absolute maximum matrix size depends on the memory size of the computer.

MAT...ZER

Examples:
```
305 MAT A = ZER
310 MAT Z = ZER (N)
315 MAT X = ZER (30, 10)
320 MAT R = ZER (N, P)
```

General Form:
statement number MAT matrix variable = ZER
or
statement number MAT matrix variable = ZER (expression)
or
statement number MAT matrix variable = ZER (expression , expression)

Purpose

Sets all elements of the specified matrix equal to 0; a new working size may be established.

Comments

The new working size in a MAT...ZER is an implicit DIM statement, and may not exceed the limit set by the DIM statement on the total number of elements in an array.

Since 0 has a logical value of false, MAT...ZER is useful in logical initialization.

MAT...CON

Examples:
```
205 MAT C = CON
210 MAT A = CON (N,N)
220 MAT Z = CON (5,20)
```

-continued
MAT...CON

230 MAT Y = CON (50)

General Form:
statement number MAT matrix variable = CON
or
statement number MAT matrix variable = CON (expression)
or
statement number MAT matrix variable = CON (expression , expression)

Purpose

Sets up a matrix with all elements equal to 1; a new working size may be specified, within the limits of the original DIM statement on the total number of elements.

Comments

The new working size (an implicit DIM statement) may be omitted as in example statement **205.**

Note that since 1 has a logical value of true, the MAT...CON statement is useful for logical initialization.

The expressions in new size specificatons should evaluate to integers. Non-integers are rounded to the nearest integer value.

PRINTING SINGLE MATRIX ELEMENTS

Examples:
800 PRINT A(3)
810 PRINT A(3,3);
820 PRINT F(X);E; C5;R(N)
830 PRINT G(X,Y)
840 PRINT Z(X,Y), Z(1,5), Z(X+N), Z(Y+M)

General Form:
statement number PRINT matrix variable (expression) . . .
or
statement number PRINT matrix variable (expression, expression) . . .

Purpose

Causes the specified matrix element(s) to be printed.

Comments

Expressions used as subscripts should evaluate to integers. Non-integers are rounded to the nearest integer value.

A trailing semicolon packs output into twelve elements per teleprinter line, if possible (statement **810** above). A trailing comma or *return* prints five elements per line.

Expressions (or subscripts) following the matrix variable designate the row and column of the matrix element. Do not confuse these with new working size specifications, such as those following a MAT IDN statement.

INPUTTING SINGLE MATRIX ELEMENTS

Examples:
600 INPUT A(5)
610 INPUT B(5,8)
620 INPUT R(X), N, A(3,3),S,T
630 INPUT Z(X,Y), P3, W
640 INPUT Z(X,Y), Z(X+1,Y+1), Z(X+R3,Y+S2)

General Form:
statement number INPUT matrix variable (expression) . . .
or

-continued
INPUTTING SINGLE MATRIX ELEMENTS

statement number INPUT matrix variable (expression, expression) . . .

Purpose

Allows input of a specified matrix element from the keyboard.

Comments

The subscripts (in *expressions*) used after the matrix variable designate the row and column of the matrix element. Do not confuse these expressions with working size specifications, such as those following a MAT READ statement.

Expression used as subscripts should evaluate to integers. Non-integers are rounded to the nearest integer value.

Inputting, printing, and reading individual array elements are logically equivalent to simple variables and may be intermixed in INPUT, PRINT, and READ statements.

MAT PRINT

Examples:
500 MAT PRINT A
505 MAT PRINT A;
515 MAT PRINT A,B,C
520 MAT PRINT A,B,C;

General Form:
statement number MAT PRINT matrix variable
or
statement number MAT PRINT matrix variable, matrix variable . . .

Purpose

Causes an entire matrix to be printed, row by row, with double spacing between rows.

Comments

Matrices may be printed in packed rows up to 12 elements wide by using the ; separator, as in example statement **505.**

READING MATRIX ELEMENTS

EXAMPLES:
900 READ A(6)
910 READ A(9,9)
920 READ C(X); P; R7
930 READ C(X,Y)
940 READ Z(X,Y), P(R2, S5), X(4)

General Form:
statement number READ matrix variable (expression)
or
statement number READ matrix variable (expression, expression) . . .

Purpose

Causes the specified matrix element to be read from the current DATA statement.

Comments

Expressions (used as subscripts) should evaluate to integers. Non-integers are rounded to the nearest integer.

Expressions following the matrix variable designate the row and column of the matrix element. Do not confuse

these with working size specifications, such as those following MAT READ statement.

The MAT READ statement is used to read an entire matrix from DATA statements. See details in this section.

MAT READ

Examples:
350 MAT READ A
370 MAT READ B(5),C,D
380 MAT READ Z (5,8)
390 MAT READ N (P3,Q/)

General Form:
statement number MAT READ matrix variable
or
statement number MAT READ matrix variable (expression) . . .
or
statement number MAT READ matrix variable (expression, expression) . . .

Purpose

Reads an entire matrix from DATA statements. A new workking size may be specified, within the limits of the original DIM statement.

Comments

MAT READ causes the entire matrix to be filled from the current DATA statement in the row, column order 1,1; 1,2; 1,3; etc. In this case, the DIM statement controls the number of elements read.

MATRIX ADDITION

Examples:
310 MAT C = B + A
320 MAT X = X + Y
330 MAT P = N + M

General Form:
statement number MAT matrix variable = matrix variable + matrix variable

Purpose

Establishes a matrix equal to the sum of two matrices of identical dimensions; addition is performed element-by-element.

Comments

The resulting matrix must be previously mentioned in a DIM statement if it has more than 10 elements, or 10 × 10 elements if two-dimensional. Dimensions must be the same as the operand matrices.

The same matrix may appear on both sides of the = sign, as in example statement **320.**

MATRIX SUBTRACTION

Examples:
550 MAT C = A − B
560 MAT B = B − Z
570 MAT X = X − A

General Form:
statement number MAT matrix variable = matrix variable − matrix variable

Purpose

Establishes a matrix equal to the difference of two matrices of identical dimensions; subtraction is performed element-by-element.

Comments

The resulting matrix must be previously mentioned in a DIM statement if it has more than 10 elements, or 10 × 10 elements if two-dimensional. Its dimension must be the same as the operand matrices.

The same matrix may appear on both sides of the = sign, as in example statement **560.**

MATRIX MULTIPLICATION

Examples:
930 MAT Z = B * C
940 MAT X = A * A
950 MAT C = Z * B

General Form:
statement number MAT matrix variable = matrix variable * matrix variable

Purpose

Establishes a matrix equal to the product of the two specified matrices.

Comments

Following the rules of matrix multiplication, if the dimensions of matrix B = (P,N) and matrix C = (N,Q), multiplying matrix B by matrix C results in a matrix of dimensions (P,Q).

Note that the product matrix must have an appropriate working size.

The same matrix variable may not appear on both sides of the = sign.

SCALAR MULTIPLICATION

Examples:
110 MAT A = (5) * B
115 MAT C = (10) * C
120 MAT C = (N/3) * X
130 MAT P = (Q7*N5) * R

General Form:
statement number MAT matrix variable = (expression) * matrix variable

Purpose

Establishes a matrix equal to the product of a matrix multiplied by a specified expression (number); that is, each element of the original matrix is multiplied by the number.

Comments

The resulting matrix must be previously mentioned in a DIM statement if it contains more than 10 elements (10 × 10 if two-dimensional).

The same matrix variable may appear on both sides of the = sign.

Both matrices must have the same working size.

COPYING A MATRIX

Examples:
405 MAT B = A
410 MAT X = Y
420 MAT Z = B

General Form:
statement number MAT variable = matrix variable

Purpose

Copies a specified matrix into a matrix of the same dimensions; copying is performed element-by-element.

Comments

The resulting matrix must be previously mentioned in a DIM statement if it has more than 10 elements, or 10 × 10 if two-dimensional. It must have the same dimensions as the copied matrix.

IDENTITY MATRIX

Examples:
205 MAT A = IDN
210 MAT B = IDN (3,3)
215 MAT Z = IDN (Q5, Q5)
220 MAT S = IDN (6, 6)

General Form:
statement number MAT array variable = IDN
or
statement number MAT array variable = IDN (expression, expression)

Purpose

Establishes an identity matrix (all 0's, with a diagonal from left to right of all 1's); a new working size may be specified.

Comments

The IDN matrix must be two-dimensional and square.

Specifying a new working size has the effect of a DIM statement.
Sample identity matrix:
1 0 0 0 1 0 0 0 1

MATRIX TRANSPOSITION

Examples:
959 MAT Z = TRN (A)
969 MAT X = TRN (B)
979 MAT Z = TRN (C)

General Form:

-continued

MATRIX TRANSPOSITION
statement number MAT matrix variable = TRN (matrix variable)

Purpose

Establishes a matrix as the transposition of a specified matrix (transposes rows and columns).

Comments

Sample transposition:

Original	Transposed
1 2 3 4 5 6 7 8 9	1 4 7 2 5 8 3 6 9

Note that the dimensions of the resulting matrix must be the reverse of the original matrix. For instance, if A has dimensions of 6,5 and MAT C = TRN (A), C must have dimensions of 5,6.

Matrices cannot be transposed or inverted into themselves.

MATRIX INVERSION

Examples:
380 MAT A = INV(B)
390 MAT C = INV(A)
400 MAT Z = INV(Z)

General Form:
statement number MAT matrix variable = INV (matrix variable)

Purpose

Establishes a square matrix as the inverse of the specified square matrix of the same dimensions.

Comments

The inverse is the matrix by which you multiply the original matrix to obtain an identity matrix.

For example,

Original		Inverse		Indentity
$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$	×	$\begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix}$	=	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Number representaton in BASIC is accurate to 6–7 decimal digits; matrix elements are rounded accordingly.

SECTION V

LOGICAL OPERATIONS

LOGICAL VALUES AND NUMERIC VALUES

A distinction should be made between logical values and the numeric values produced by logical evaluation, when using the logical capability of BASIC.

The *logical value* of an expression is determined by definitions established in the user's program.

The *numeric values produced by logical evaluation* are assigned by BASIC. The user may not assign these values.

Logical value is the value of an expression or statement, using the criteria:
any nonzero expression value = true
any expression value of zero = false

When an expression or statement is logically evaluated, it is assigned one of two numeric values, either:
1, meaning the expression or statement is true,
or
0, meaning the expression or statement is false.

RELATIONAL OPERATORS

There are two ways to use the relational operators in logical evaluations:

1. As a simple check on the numeric value of an expression.

Examples:
```
150 IF B=7 THEN 600
200 IF A9 # 27.65 THEN 700
300 IF (Z/10)>0 THEN 800
```

When a statement is evaluated, if the IF condition is currently true (for example, B = 7 in statement **150**), then control is transferred to the specified statement; if it is not true, control passes to the next statement in the program.

Note that the numeric value produced by the logical evaluation is unimportant when the relational operators are used in this way. The user is concerned only with the presence or absence of the condition indicated in the IF statement.

2. As a check on the numeric value produced by logically evaluating an expression, that is: true = 1, false = 0.

Examples:
```
610 LET X=27
615 PRINT X=27
620 PRINT X # 27
630 PRINT X>=27
```

The example PRINT statements give the numeric values produced by logical evaluation. For instance, statement **615** is interpreted by BASIC as "Print 1 if X equals 27, 0 if X does not equal 27." There are only two logical alternatives; 1 is used to represent true, and 0 false.

The numeric value of the logical evaluation is dependent on, but distinct from, the value of the expression. In the example above, X equals 27, but the numeric value of the logical expression X=27 is 1 since it describes a true condition.

BOOLEAN OPERATORS

There are two ways to use the Boolean Operators.
1. As logical checks on the value of an expression or expressions.

Examples:
```
510 IF A1 OR B THEN 670
520 IF B3 AND C9 THEN 680
530 IF NOT C9 THEN 690
540 IF X THEN 700
```

Statement **510** is interpreted: "If either A1 is true (has a non-zero value) or B is true (has a non-zero value), then transfer control to statement **670**"
Similarly, statement **540** is interpreted: "If X is true (has a non-zero value), then transfer control to statement **700.**"

The Boolean operators evaluate expressions for their logical values only: these are true = any non-zero value, false = zero. For example, if B3 = **9** and C9 = **−5**, statement **520** would evaluate to true, since both B3 and C9 have a non-zero value.

2. As a check on the numeric value produced by logically evaluating an expression, that is: true = 1, false = 0.

Examples:
```
490 LET B = C = 7
500 PRINT B AND C
510 PRINT C OR B
520 PRINT NOT B
```

Statements **500 – 520** return a numeric value of either 1, indicating that the statement has a logical value of true, or 0, indicating a logical value of false.

Note that the criteria for determining the logical values are:
true = any non-zero expression value
false = an expression value of 0.
The numeric value 1 or 0 is assigned accordingly.

SECTION VI

SYNTAX REQUIREMENTS OF BASIC

Legend

::= "is defined as..."
| "or"
< > enclose an element of BASIC

Language Rules

1. The <com statement>, if any exists, must be the first statement presented and have the lowest sequence number; the last statement must be an <END statement>.
2. A sequence number may not exceed 9999 and must be non-zero.
3. Exponent integers may not have more than two digits.
4. A formal bound may not exceed **255** and must be non-zero.
5. A subroutine number must lie between 1 and 63, inclusive.
6. Strings may not contain the quote character (").

7. A <bound part> for an IDN must be doubly subscripted.

8. An array may not be inverted or transposed into itself.

9. An array may not be replaced by itself multiplied by another array.

SYNTAX REQUIREMENTS

<basic program> ::= <program statement> |<basic program> <program statement>(1)

<program statement> ::= <sequence number> <basic statement> carriage return

<sequence number> ::= <integer>(2)

<basic statement> ::= <let statement> |<dim statement> |<com statement> |
<def statement> |<rem statement> |<go to statement> |
<if statement> |<for statement> |<next statement> |
<gosub statement> |<return statement> |<end statement> |
<stop statement> |<wait statement> |<call statement> |
<data statement> |<read statement> |<restore statement> |
<input statement> |<print statement> |<mat statement>

<let statement> ::= <let head> <formula>

<let head> ::= LET <variable>= |<let head> <variable>=

<formula> ::= <conjunction> |<formula> OR <conjunction>

<conjunction> ::= >boolean primary> |<conjunction> AND <boolean primary>

<boolean primary> ::= <arithmetic expression> |<boolean primary>
<relational operator> <arithmetic expression>

<arithmetic expression> ::= <arithmetic expression> − <term>
<term> | <arithmetic expression> + <term> |

<term> ::= <factor> |<term> * <factor> |<term> / <factor>

<factor> ::= <primary> |<sign> <primary> |NOT <primary>

<primary> ::= <operand> |<primary> ↑ <operand>

<relational operator> ::= > |< |>= |<= |= | # |<>

<operand> ::= <variable> |<unsigned number> |<system function> |
<function> |<formula operand>

<variable> ::= <simple variable> |<subscripted variable>

<simple variable> ::= <letter> |<letter> <digit>

<subscripted variable> ::= <array identifier> <subscript head> <subscript>
<right bracet>

<array dentifier> ::= <letter>

<subscript head> ::= <left bracket> |<left bracket> <subscript>

<subscript> ::= <formula>

<letter> ::= A |B |C |D |E |F |G |H |I |J |K |L |M |N |O |P |Q |R |S |T |U |V |W |X |Y |Z

<digit> ::= 0 |1 |2 |3 |4 |5 |7 |8 |9

<left bracet> ::= (|[

<right bracket> ::=) |]

<sign> ::= + |−

-continued

SYNTAX REQUIREMENTS

<unsigned number > ::= <decimal part> |<decimal part> <exponent>

<decimal part> ::= <integer> |<integer> . <integer> |. <integer>

<integer> ::= <digit> |<integer> <digit>

<exponent< ::= E <integer> |E<sign> <integer>[3]

<system function> ::= <system function name> <parameterpart>

<system function name> ::= SIN |COS |TAN |ATN |EXP |LOG |ABS |SQR |INT |RND |SGN

<parameter part> ::= <left bracket> <actual parameter> <right bracket>

<actual parameter> ::= <formula>

<function> ::= FN <letter> <parameter part>

<formula operand> ::= <left bracket> <formula> <right bracket>

<dim statement> ::= DIM <formal array list>

<formal array list> ::= <formal array> |<formal array list> , <formal array>

<formal array> ::= <array identifier> <formal bound head> <formal bound> <right bracket>

<formal bound head> ::= <left bracket> |<left bracket> <formal bound> ,

<formal bound> ::= <integer>[4]

<com statement> ::= COM <formal array list>

<def statement> ::= DEF FN <letter> <left bracket> <formal parameter><right bracket>=<formula>

<formal parameter> ::= <simple variable>

<rem statement> ::= REM <character string>

<character string> ::= any teletype character except carriage return, alt mode, escape, rubout, or line feed, or null, control B, control C, left arrow

<goto statement> ::= GO TO <sequence number>

<if statement> ::= IF <formula> THEN <sequence number>

<for statement> ::= <for head> |<for head> STEP <step size>

<for head> ::= FOR <for variable>=<initial value> TO <limit value>

<for variable> ::= <simple variable>

<initial value> ::= <formula>

<limit value> ::= <formula>

<step size> ::= <formula>

<next statement> ::= NEXT <for variable>

<gosub statement ::= GOSUB <sequence number>

<return statement> ::= RETURN

<end statement> ::= END

<stop statement> ::= STOP

<wait statement> ::= WAIT <parameter part>

<call statement> ::= CALL <call head> <right bracket>

<call head> ::= <left bracket> <subroutine number> |<call head> , <actual parameter>

<subroutine number> ::= <integer>[5]

-continued

SYNTAX REQUIREMENTS	
<data statement>	::= DATA <constant> |<data statement> , <constant>
<contstant>	::= <unsigned number> |<sign> <unsigned number>
<read statement>	::= READ <variable list>
<variable list>	::= <variable> |<variable list> , <variable>
<rstore statement>	::= RESTORE
<input statement>	::= INPUT <variable list>
<print statement>	::= <print head> |<print head> <print formula>
<print head>	::= PRINT |<print head> <print part>
<print part>	::= <string> |<string> <delimiter> |<print formula> <delimiter> |<print formula> <string> | <print formula> <string> <delimiter>
<string>	::= "<character string>"[6]
<delimiter>	::= , |;
<print formula>	::= <formula> |TAB <parameter part>
<mat statement>	::= MAT <mat body>
<mat body>	::= <mat read> |<mat print> |<mat replacement>
<mat read>	::= READ <actual array> |,5 <mat read> , <actual ar ray>
<actual array>	::= <array identifier> |<array identifier> <bound part>
<bound part>	::= <actual bound hed> <actual bound> <right bracet>
<actual bound head>	::= <left bracket> |<left bracket> <actual bound> ,
<actual bound>	::= <formula>
<mat print>	::= PRINT <mat print part> |PRINT <mat print part> <delimiter>
<mat print part>	::= <array identifier> |<mat print part> <delimiter> <array identifier>
<mat replacement>	::= >array identifier>=<mat formula>
<mat formula>	::= <array identifier> |<mat funtion> |<array identifier> <mat operator> <array identifier> |<formula operand>* <array identifier>
<mat function>	::= <mat°initialization> |<mat initialization> <bound part> | INV <parameter>|TRN <array parameter>
<mat initialization>	::= ZER |CON |IDN[7]
<array parameter>	::= <left bracket> <array identifier> <right bracket>[8]
<mat operator>	::= + |- |*[9]

STRINGS PLUG-IN READ-ONLY MEMORY MODULE

The strings plug-in read-only memory module makes available to the user all of the string variables functions and operations associated with standard BASIC programming language. Two additional functions not usually provided in most versions of BASIC language have been implemented. These include a POS function for determining the position of a substring within a string

and a VAL function for determining the position of a substring within a string and a VAL function for determining the numeric value of a string. A discussion of these and other string functions and operations follows.

String

A set of 1 to 255 characters or the null string (no characters).

e.g. "ABCDEF"
"12345"
""

String Variable

A variable used to store strings; consists of a single letter (A to Z) followed by a $.

e.g. A$; B$; Z$

Substring Variable

A single character or a set of contiguous characters from within a string variable. The substring is defined by a subscripted string variable. A single subscript specifies the first character of the substring and implies that all characters following are part of the substring. Two subscripts specify the first and last characters of the substring.

e.g. A$ = "ABCDEF"
A$(4) = DEF
A$(1,3) = ABC

DIM Statement

General Form:

stmt # DIM string var (# chars in string)

Purpose:

Reserves storage space for strings longer than 1 character.

Comments:

The number of characters specified for a string in its DIM statement must be expressed as an integer from 1 to 255. Strings not mentioned in a DIM statement are assumed to have length 1. The length mentioned in the DIM statement specifies the maximum number of characters which may be assigned and is known as the physical length. The actual length is the actual number of characters which has been assigned.

e.g. DIM A$(10), B(5,5), B$(255)

Source String Semantic Description

A source string is an entity from which a string value is extracted. If no substring designator is specified for a string variable, the value of the source string is the entire logical string currently assigned to it. Substring designator expressions must be at least 1 in value and the second may be no smaller than one less than the first. If one subscript is given, the value is the entire logical string beginning at the character specified by the subscript. The subscript may be no larger than the actual length of the string plus 1. If two substring expressions are given, the source string value is the substring whose first and last characters are designated. If the specified substring extends beyond the logical length of the string, spaces are used to fill out the substring.

Assignment Statement

General Form:

line # [LET] destination string = source string

Destination String:

A destination string is an entire string variable or part of a string variable into which a source string is to be copied. The definition of the action depends on the number of substring subscripts specified in the destination string.

If no subscript qualifier is specified, the entire destination variable is replaced by the source string. The physical length of the destination string must be large enough to accommodate the entire source string.

If one substring subscript is specified, the entire value of the destination variable, beginning at the designated character, is replaced by the source string. That part of the destination variable preceding the subscript value is unchanged. The subscript value must be no more than one greater than the actual length of the destination variable.

If two subscripts are specified, the first substring subscript must be no more than one greater than the logical length of the destination variable, and the second subscript must be no greater than the physical length of the destination variable. The specified section of the destination string is replaced by the source string. If the source string is longer than the destination, it is truncated on the right. If the source string is shorter, as many trailing blanks are appended as necessary. The new actual length of the destination string is the larger of the old actual length and the second subscript.

String Input Statement

General Form:

stmt # INPUT string or substring variable

Purpose:

Allows string values to be entered from the keyboard.

Comments:

Numeric variables may be used in the same input statement as string variables. Placing a single string variable in an input statement allows the string value to be entered without enclosing it in quotation marks. If multiple string variables are used each string value must be enclosed in quotation marks, and the values separated by commas.

e.g. **10** INPUT A$, B$, C, Al, A$(**1,5**)

String Print Statement

General Form:

stmt # PRINT string or substring variable,

Purpose:

Causes the current value of the specified string or substring variable to be output on the standard output device.

Comments:

Strings and numeric values may be mixed in a print statement. Sring variables are specified identically to numeric variables. They are printed under the same format rules as quote fields. String and substring variables are printed as source strings.

A maximum of 72 characters can be printed using the print statement, i.e., all strings >72 characters are truncated at 72 characters.

String Read Statement

General Form:

stmt # READ string or substring variable

Purpose:

Causes the value of a specified string or substring variable to be read from a data statement.

Comments:

Mixed string and numeric values may be read. If the wrong data type is given in the data statement an error is given.

e.g. **10** READ A, B$(**1, 10**), C$, B

String If Statement

General Form:

stmt # IF str.var. rel.oper. string THEN stmt #

Purpose:

Compares two strings. If the specified condition is true, control is transferred to the specified statement.

Comments:

Strings are compared one character at a time, from left to right; the first difference determines the relation. If one string ends before a difference is found, the shortest string is considered the smaller one.

Characters are compared by their ASCII representations.

The relational operators allowed are:

=, ≠, <, >, <=, >=, <>

e.g. **10** IF A$ = "SAM" THEN **20**

String Data Statement

General Form:

stmt # DATA "string text", "string text",

Purpose:

Specifies data in a program (string or/and numeric)

Comments:

String values must be enclosed by quotation marks and separated by commas.

String and numeric values may be mixed in a single data statement.

e.g. **10** DATA "ABC", **1.2**, "DEF"

String Write Statement

General Form:

stmt # WRITE (device #, format stat #) string or substring variable

Purpose:

Similar to the print statement but allows the specification of the output device and format statement. No field specifications are made for string variables in the format statement. They are treated identically to string constants (quote fields).

String Display Statement

General Form:

stmt # DISP string or substring variable

Purpose:

Identical to print statement except output device is 32 character display.

LEN Function

General Form:

LEN string

or

LEN (string)

Purpose:

Obtain the length of a string for use in an arithmetic expression.

Comment:

The actual length is found which is not necessarily the same as the physical length reserved in the DIM statement.

e.g. LEN ("ABCD") = **4**

POS Function

General Form:

POS string, string

or

POS (string, string)

Purpose:

Determine the position of a substring within a string.

Comments:

If the second string argument is a part of the first, the value of the function is the position in the first string at which the second string starts. If the second string in the argument is not a part of the first, the value of the function is zero.

e.g. POS ("ABCD", "C") = **3**

VAL Function

General Form:

VAL string

or

VAL (string)

Purpose:

Determine the numeric value of a string.

Comments:

The VAL function converts a string of digits into a number. The string is converted into a number by the same rules used in a numeric input statement.

e.g. VAL ("**123**") = **123**

ERROR MESSAGES FOR STRING OPTION BLOCK

70 — string IF error
71 — string function syntax error
72 — negative string length
73 — non-contiguous string
74 — string overflow
75 — data is of wrong type
76 — VAL function argument not numeric

BNF SYNTAX DESCRIPTION

<literal string>	::="<character string 22 "
<character string>	::=<character> |character string> <character>
<character>	::=any ASCII character except NULL, LINE FEED, RETURN
<letter>	::=A |B |X |Y |Z
<sublist>	::=expression> |expression>,<expression>
<string variable>	::=<simple string variable> | <simple string variable>(<sublist>)
<simple string variable>	::=<letter>$
21 relational operator>	::=< |<=|=|# |< >|>=|>
<assignment statement>	::=LET <destination string>=<source string> |<destination string>=<source string>
<destination string>	::=<string variable>
<source string>	::=<string variable> |<literal string>
<IF statement>	::=IF <destination string><relational operator><sourcestring> THEN <line number>
<data statement>	::=DATA <constant> |<data statement>, <constant>
<constant>	::=<numeric constant> |<literal string>
<read statement>	::=INPUT <variable list>
<variable list>	::<read variable> |<variable list>, <read variable>
<read variable>	::=<numeric variable> |<string variable>
<input statement>	::=INPUT <variable list>
<print statement>	::=<print> |<print 2>
<print 1>	::=PRINT |<print 2>, |<print 2>; |<print 3>
<print 2>	::=<print 1><print expression> | <print 3>

-continued

BNF SYNTAX DESCRIPTION	
<print 3'	::=(type statement><literal string>
<print expression>	::=<expression> |<source string>
WRITE and disp statements have the same syntax as the PRINT statement.	
<LEN function>	::=LEN<source string> |LEN(<source string>)
<VAL function>	::=VAL<source string> |VAL(<source string>)
<POS function>	::=POS<source string>,<source string> | POS(<source string>,<source string>)

EXTENDED INPUT/OUTPUT PLUG-IN READ-ONLY MEMORY MODULE

The extended I/O read-only memory module (hereinafter referred to as the extended I/O ROM) provides additional functions and statements so that the calculator can be made compatible with a wide variety of peripheral devices. Added functions include decimal-to-binary and octal-to-decimal conversion, status code inquiry for peripheral devices, control of spacing and line feeds in output records, and others. Use of these functions requires no special programming techniques; once the ROM is plugged in, its functions and statements become a part of the calculator, in the same way as, for example, the square root function is part of the calculator.

Some read-only memory units decrease the amount of programmable memory avilable to the user by automatically requiring a portion of that memory for their own internal usage, but the extended I/O ROM has no such requirement and does not affect memory availability.

The following table describes the extended I/O ROM functions and statements. Parameters shown underlined in the table are explained immediately following the table. Parameters shown in brackets may or may not be included as parts of a statement.

FUNCTION MNEMONIC	DESCRIPTION OF FUNCTION	SYNTAX
BIN	Converts decimal expression to its binary equivalent. For use in output-to-binary storage device; also provides increased control of print format.	BIN exp
OCT	Converts octal expression to decimal equivalent. For use in construction of code conversion tables; see "Conversion Tables".	OCT exp
STAT	Returns code of operational status (on, off, wait, etc.) for the device specified by the select code.	STAT select code
CHAR	Returns one byte of data from the device specified by the select code regardless of the data structure.	CHAR select code
LIN	Advances printer or typewriter the number of lines represented by the expression.	LIN exp
SPA	Advances printer or typewriter carriage the number of spaces represented by the expression.	SPA exp
ROT	Converts expression 1 to binary equivalent; performs rotation right the number of positions represented by expression 2; returns decimal equivalent. For use in special input or output code translation.	ROT (exp 1, exp 2)
INOR	Combines binary equivalents of expression 1 and expression 2 in an "inclusive or" logic operation. Returns decimal equivalent. For use in special input or output code translation.	INOR (exp 1, exp 2)
BIAND	Combines binary equivalents of expression 1 and expression 2 in an "and" logic operation. Returns decimal equivalent. For use in special input or output translation.	BIAND (exp 1, exp 2)

STATEMENT SYNTAX	DESCRIPTION OF STATEMENT
ENTER (select code or string name, format [, conversion table]) list [, FOR function]	Inputs data frm named device or string with optional conversion to ASCII code; includes capability for iteration.
OUTPUT (select code or string name, format [, converison table]) list	Outputs data to named device or string with optional conversion to ASCII code.

PARAMATER	EXPLANATION
exp	Expression
select code	A numeric code, from 1 to 15, uniquely representing the input or output device, as follows: select code 1-9 User assignable. select code 10 Cassette Memory. select code 11-13 reserved. select code 14 Plotter. select code 15 Typewriter or Printer.
string name	A single letter followed by a "$". Valid only when String ROM is also plugged in.
format	To reference a FORMAT statement, the line number of that statement is shown; for free-form data, an asterisk (*) is shown.
conversion table	The variable or array name given to a conversion table. See heading "Conversion Tables".
list	A list of variables, literals, expressions or numerics separated by commas.
FOR functions	To input multiple data items from one record into an array. Syntax as follows:

(FOR var 1=exp TO exp, [(FOR var 2= exp TO exp,] array name(var 1 [, var 2]) [])

CONVERSION TABLES

Using a pre-established conversion table, a string of characters or an array of data can be converted from one code to another. The calculator makes use of standard ASCII* codes. Let us refer to all non-ASCII representation codes used by printers, card readers, paper tape readers, punches, typewriters, etc. as foreign codes.

* American Standard Code for Information Interchange.

A conversion table is defined in the BASIC language program DIM statement. Only single-dimensioned integer arrays are considered valid for use as conversion tables. In the following DIM statements, A and B are valid array structures for conversion tables, but C, D, and E are not.

DIM AI (150), CI(20, 30)

DIM BI (80), D(200), E(15, 15)

In order to insert conversion table information in the array, the foreign code must first be known. Suppose your paper tape reader uses EIA** coded tape. The chart following shows symbols and the equivalent tape punches for EIA code, with the octal code equivalent for each symbol.

** Electronic Industries Association Standard Code.

STANDARD TAPE CHANNEL NUMBERS 8	7	6	5	4	FEED	3	2	1	SYMBOL	EIA OCTAL CODE
		•			•				(ZERO) 0 —)	040
					•			•	1 — _	001
					•		•		2 — @	002
			•		•		•	•	3 — #	023
					•	•			4 — $	004
			•		•	•		•	5 — %	025
			•		•	•	•		6 — '	026
					•	•	•	•	7 — &	007
				•	•				8 — *	010
			•	•	•			•	9 — (	031
	•	•			•			•	a — A	141
	•	•			•		•		b — B	142
	•	•	•		•		•	•	c — C	163
	•	•			•	•			d — D	144
	•	•	•		•	•		•	e — E	165
	•	•	•		•	•	•		f — F	166
	•	•			•	•	•	•	g — G	147
	•	•		•	•				h — H	150
	•	•	•	•	•			•	i — I	171

STANDARD TAPE CHANNEL NUMBERS 8	7	6	5	4	FEED	3	2	1	SYMBOL	EIA OCTAL CODE
	•		•		•			•	j — J	121
	•		•		•		•		k — K	122
	•				•		•	•	l — L	103
	•		•		•	•			m — M	124
	•				•	•		•	n — N	105
	•				•	•	•		o — O	106
	•		•		•	•	•	•	p — P	127
	•		•	•	•				q — Q	130
	•			•	•			•	r — R	111
		•	•		•		•		s — S	062
		•			•		•	•	t — T	043
		•	•		•	•			u — U	064
		•			•	•		•	v — V	045
		•			•	•	•		w — W	046
		•	•		•	•	•	•	x — X	067
		•	•	•	•				y — Y	070
		•		•	•			•	z — Z	051
			•		•				SPACE	020
	•				•				- _ "	100
		•	•		•			•	/ — ?	061
				•	•		•	•	STOP	013
	•			•	•		•	•		133
		•	•	•	•		•	•	, — <	073
	•	•		•	•		•	•	. — >	153
				•	•	•				
		•		•	•	•				
	•			•	•	•			PUNCH ON	114
	•	•		•	•	•			UPPER CASE	174
	•	•			•				; — :	160
		•		•	•	•	•		TAB	076
				•	•	•	•		CONTROL	016
		•		•	•	•	•	•	PUNCH OFF	067
				•	•	•	•	•	DATA SELECTOR(AUX.3)	037
	•			•	•	•	•	•	FORM FEED (AUX. 4)	117
				•	•		•			
	•			•	•		•			
		•		•	•		•		BACK SPACE	062
		•		•	•	•		•		
	•	•		•	•	•		•		
	•			•	•	•		•	ADDRESS IDEN.(AUX.8)	135
				•	•	•		•		
	•	•		•	•		•		LOWER CASE	172
	•	•		•	•	•	•			
	•			•	•	•	•			
	•	•		•	•	•	•	•	TAPE FEED	177
•					•				CAR. RET.	200

The following chart shows the ASCII symbols with their octal code equivalents.

Symbol	ASCII Equivalent (octal) code)
(Space)	040
!	041
#	043
$	044
%	045
&	046
'	047
(	050
)	051
*	052
+	053
,	054
—	055
.	056
/	057
0	060
1	061
2	062
3	063
4	064
5	065
6	066
7	067
8	070
9	071
:	072
;	073
<	074
=	075
>	076
?	077
@	100
A	101
B	102
C	103
D	104
E	105
F	106
G	107
H	110
I	111
J	112
K	113
L	114
M	115
N	117
O	117
P	120
Q	121
R	122
S	123
T	124
U	125
V	126
W	127
X	130
Y	131
Z	132
[	133
\	134
]	135

The following chart shows the information to be programmed into the conversion table.

Symbol	EIA (octal equiv)	ASCII (octal equiv)	EIA Symbol	EIA (octal equiv)	ASCII (octal equiv)
Sp	020	040	A	141	101
!		041	B	142	102
#		043	C	163	103
$		045	E	165	105
&		046	F	166	106
'		047	G	147	107
			H	150	110
(		050	I	171	111
)		051	J	121	112
*		052	K	122	113
+		053	L	103	114
,	073	054	M	124	115
—	100	055	~N	105	116
.	153	056	0	106	117
/	061	057	P	127	120
			Q	130	121
0	040	060	R	111	122
1	001	061	S	62	123
2	002	062	T	43	124
3	023	063	U	64	125
4	004	064	V	45	126
5	025	065	W	46	127
6	026	066	X	67	130
7	007	067	Y	70	131
8	010	070	Z	51	132
9	031	071			
:		072	[		133
;		073	\		134
<		074	]		135
=		075			
>		076			
?		077			
@		100			
C+RR	200	012			

The conversion table portion of a BASIC program is shown on the chart below. Each statement defines one element in the conversion Table A with the use of the OCT function: the octal notation does not need to be translated to decimal. If decimal notation was supplied in the symbol tables, the OCT function could have been omitted; however, it is common to obtain symbol tables in the octal form rather than in decimal form.

```
10 DIM AI[128]
20 A[OCT20]=OCT40
30 A[OCT73]=OCT54
40 A[OCT100]=OCT55
50 A[OCT153]=OCT56
60 A[OCT61]=OCT57
70 A[OCT40]=OCT60
80 A[OCT1]=OCT61
90 A[OCT2]=OCT62
100 A[OCT23]=OCT63
110 A[OCT4]=OCT64
120 A[OCT25]=OCT65
```

```
130 A[OCT26]=OCT66
140 A[OCT7]=OCT67
150 A[OCT10]=OCT70
160 A[OCT31]=OCT71
170 A[OCT200]=OCT12
180 A[OCT141]=OCT101
190 A[OCT142]=OCT102
200 A[OCT163]=OCT103
210 A[OCT144]=OCT104
220 A[OCT165]=OCT105
230 A[OCT166]=OCT106
240 A[OCT147]=OCT107
250 A[OCT150]=OCT110
260 A[OCT171]=OCT111
270 A[OCT121]=OCT112
280 A[OCT122]=OCT113
290 A[OCT103]=OCT114
300 A[OCT124]=OCT115
310 A[OCT105]=OCT116
320 A[OCT106]=OCT117
330 A[OCT127]=OCT120
340 A[OCT130]=OCT121
350 A[OCT111]=OCT122
360 A[OCT62]=OCT123
370 A[OCT43]=OCT124
380 A[OCT64]=OCT125
390 A[OCT45]=OCT126
400 A[OCT46]=OCT127
410 A[OCT67]=OCT130
420 A[OCT70]=OCT131
430 A[OCT51]=OCT132
```

ENTER

When a CHAR request is keyed into the calculator for the paper tape reader, conversion is not done and the calculator will display the decimal equivalent of the EIA octal code for the symbol taken from the paper tape. In order to have automatic code conversion, the program must contain an ENTER statement.

```
410 ENTER (9, 420, A) B
```

B will be read and its ASCII equivalent found in the conversion table A. If data contained in a string is to be converted to ASCII code, the string name is used instead of the select code, and in this way conversion can be done internally as well as at time of input or output.

OUTPUT

For output of data in a foreign code, automatic code conversion is invoked by use of the OUTPUT statement.

```
500 OUTPUT (8, 510, A) B
```

The ASCII B will be found in the conversion table A and changed to the foreign code equivalent before output. Notice that the same conversion table A is used for input and output. The ENTER statement causes the calculator to look for ASCII code, (on the right of the equals (=) signs in the table), and assume that it is receiving foreign code; whereas the OUTPUT statement causes the calculator to assume it has ASCII code, and to look for the fogeign code (on the left of the equals (=) signs in the table above.

ERROR CODES

Program diagnostic or error conditions found in the Extended I/O ROM:

NUMBER	EXPLANATION
ERROR 83	End of data reached or data contains more than ten (10) blanks in a row.
ERROR 84	Invalid format specification: format must be free format (*), E format or F format.
ERROR 85	Numeric input syntax error: multiple decimal points, more than one E in E format, etc.
ERROR 86	Conversion table not found. Check for integer initialization in DIM statement.

TERMINAL PLUG-IN READ-ONLY MEMORY MODULE

The terminal plug-in read-only memory module that is available with the calculator allows the user to enter, store, and edit free-text. It also allows the user to communicate, either directly or through an external modem, with another calculator, a computer or a time-sharing computer system. The calculator may transmit or receive BASIC language programs or free-text.

To put the calculator into terminal mode, the user types in TERM and actuates the EXECUTE key. If he wants to use the calculator for data transmission, he can specify one or two optional parameters following the TERM mnemonic. The first parameter is the select code of the modem interface module. If another select code is not specified the calculator assumes select code four. The second parameter is the baud rate of the transmitted data. If not specified, the calculator will assume 110 baud which is the same rate as a standard ASR232 teletypewriter. For example, to transmit or receive on select code six at 300 baud, the user enters TERM, **6,300** followed by actuation of the EXECUTE key. The selectable baud rate is continuous in integer increments from 3 to 300. The conversion of characters from parallel to serial format is done automatically within the calculator firmware, so the modem interface circuitry is simplified and no switches are required to change baud rate.

While the calculator is operating in the terminal mode, lines of text may be entered from the keyboard terminated with the END-OF-LINE key. These lines must be preceded by a line number, but there is no syntax requirement for the remainder of the line. All of the line-by-line and character-by-character editing features of the calculator are available in this mode. These include listing, backspace, forward space, insert character, delete character, and display shift control. Automatic line numbering is also available. In addition, the tape cassette commands, the PTAPE command, the PRINT ALL command, and the LIST command operate normally. The LIST command syntax has been expanded to include LISTX, which means list without line numbers, as well as LIST # SC or LISTX # SC where SC is the select code of the modem interface, which means transmit the information through the modem to the remote system.

When the calculator is operating in the terminal mode, five of the user-definable keys take on special meaning. Key *f5* becomes a teletype shift key and *f6* becomes a teletype control key. These keys and the lower case shift key allow the user to generate any seven-bit ASCII code. To generate a teletype shift c. control character, the user first actuates the *f5* or *f6* keys and then actuates the appropriate key on the alpha section of the keyboard. For example, to generate a control C, the *f6* key and the C key are actuated sequentially. No character is entered into the display until after the chosen alpha key is actuated. The char-

acter entered into the display may or may not be the same as the alpha character. For instance a shift O generates the symbol ⊣.

Key *f8* is used in the terminal mode to select even or odd parity for transmitted characters. When the terminal mode is first entered, even parity is assumed. The user may then convert to odd parity by actuating key *f8*. The display will then indicate ODD. To revert back to even parity the user actuates key *f8* again, and the display indicates EVEN.

Key *f9* is used in the terminal mode as a transmit key. To transmit a message through the modem interface, the message is typed into the display from the keyboard, and then the line is terminated by actuating key *f9*. The calculator then serializes the characters entered, and transmits them at the selected baud rate. For example, to obtain a listing of a program from a remote time-sharing computer service, the user enters LIST *f9*. The transmit key may also be used to enter the responses in a sign-on procedure for a time-sharing service.

Key *f7* is used to place the calculator into a mode for saving, in memory, an incoming program. For example, to receive and store a program from a time-sharing computer service, the user actuates key *f7* followed by LIST *f9*. As the program is listed from the time-sharing service, the lines are stored in the calculator's user memory as free-text. If the *f7* key is not actuated just prior to entering LIST *f9*, the program will be printed on the external line printer.

Once a program has been entered as free-text, either from a remote source or from the keyboard, and if it is a BASIC program, it may be checked for syntax errors and converted to BASIC program format in memory. This is done by typing COMP followed by actuation of the EXECUTE key. Any syntax errors will be listed, and only the lines which have correct BASIC syntax will be translated. Incorrect lines will remain in free-text format. After a program has been translated by the COMP command, it may be executed locally using any of the normal execution commands, i.e. RUN, CONT, etc. An attempt to execute an untranslated line will cause an error message (ERROR 79) to be displayed.

The receiving section of the modem driver routine operates under interrupt control which allows the user to execute programs locally and remotely at the same time. For example, the user may want to run a program on a time-sharing computer system that may take several minutes to complete. He may start that program by transmitting a RUN command. While that program is being executed, the calculator is free for normal keyboard operation or program execution. The only limitation in the mode is that the calculator may not use the display or printer at this same time as it is receiving information from the remote program.

PLOTTER PLUG-IN READ-ONLY MEMORY MODULE

The plotter plug-in read-only memory module enables the calculator to control an Hewlett-Packard **9862A** calculator plotter, providing permanent graphic solutions to problems solved by the calculator.

In general, the plotter command set can be considered as consisting of two groups: plotting commands and writing commands. The user can specify any plotting units he pleases, the calculator then automatically scales those units to fit the chosen plotting area. Also, the calculator keyboard characters can be drawn in different sizes and directions.

The plotting commands enable the system to automatically scale user-units; draw X and Y axes of any length, anywhere in the plotting area; make any desired tic-marks on the axes; plot points or functions; lower or raise the pen, either before or after movement; temporarily translate the established origin to any point within the plotting area and then plot, still in user-units, with respect to the new origin; plot in increments (that is, in user-units, plot any point with respect to the current pen position).

The writing commands enable most calculator keyboard characters to be printed on the plotter. The user can specify the position, height and width of the characters, and the direction in which they will be printed. A centering command (CPLOT) enables labels to be centered on some particular point, thus simplifying labelling of axes and of specific points on the graph. The format of labels and numbers — field width, fixed or floating point, the number of digits following the decimal point, etc. — is specified by standard FORMAT statements. In addition, a unique LETTER command establishes a typewriter mode enabling the plotter to be controlled and positioned from the calculator keyboard, on a character-by-character basis; this allows the user to add extra labelling or individual comments to his graphs.

INITIALIZING THE PLOTTER

Before plotting, the plotter must be prepared and the physical limits of the plotting area must be established. The front-panel controls on the plotter are used for this purpose.

LINE AND CHART HOLD

The LINE pushbutton is the power switch for the plotter; press it to apply power, and press it again to remove power; the white LINE lamp lights whenever the plotter is ON.

Pressing CHART HOLD activates the electro-static paper hold-down mechanism. Pressing CHART HOLD again deactivates it. The plotter will not plot or letter, and the pen holder and arm will move freely in all directions when CHART HOLD is deactivated.

LOADING PAPER

To load paper, release CHART HOLD and manually move the pen arm all the way to one side of the plotter. Lay a sheet of paper on the plotting surface and smooth out any irregularities in the paper (you may also wish to ensure that the paper is squarely against the ridge at the bottom of the plotting surface); then activate CHART HOLD.

GRAPH LIMITS

The graph limit controls are used to determine the physical size of the plot.

LOWER LEFT and the two knobs to its left are used to determine the physical location of the lower left hand corner of the plotting area.

UPPER RIGHT and the two knobs to its right are used to determine the physical location of the upper right-hand corner of the plotting area. Together, the upper right-hand corner and the lower left-hand corner determine the size of the plotting area.

Also, altering the lower left-hand setting will translate the upper right-hand setting by the same direction and amount.

To specify the lower left-hand corner of the plotting area, press LOWER LEFT; the pen will move (without touching the paper) to the lower left-hand corner of the

plotting area. This point can be set anywhere within the lower left-hand quarter of the plotting surface (platen) by adjusting the two knobs associated with LOWER LEFT. Once the lower left-hand corner has been set, the upper right-hand corner is set in the same general way by pressing UPPER RIGHT and adjusting the two knobs associated with it. Once the plotting area has been determined, it can be relocated by moving the position of the lower left-hand corner – the upper right-hand corner will track the change.

PLOTTING COMMANDS

NOTES:

1. All commands can be activated either from the keyboard or from a program except where noted.
2. All values in the following statements can be numbers, variables or expressions except where noted.
3. Any parameter enclosed in square brackets is optional as far as the statement containing it is concerned. However, program sense may dictate that the parameter be present in specific cases.

THE SCALE STATEMENT

SCALE Xmin, Xmax, Ymin, Ymax

Examples:

SCALE **–10, 10, –5, 5**

SCALE **–4PI, 4PI, .3, 1.1**

establishes the full-scale units for the plot. Xmin to Xmax and Ymin to Ymax correspond exactly to the limits of the horizontal and vertical edges, respectively, of the plotting area (the area is established mechanically as previously described). This also establishes the point, on or off the plotting area, where the original of the graph (**0, 0**) is located.

A SCALE statement must be executed before any plotting can occur. Once established the scale remains established until one of the following occurs:

A new SCALE statement is executed.

The program is initialized.

A SCRATCH OR SCRATCH A or SCRATCH V is executed.

The calculator is switched off.

The parameters (X, Y, etc.) in the SCALE statement must be given in the correct order. If the minimum or maximum values are switched no ERROR message will occur; however, subsequent plotting commands may not be executed properly.

The SCALE statement has no effect on the position of the pen.

THE PEN STATEMENT

PEN

The PEN statement is a stand-alone instruction requiring no parameter. It raises the pen without otherwise changing its position relative to the plotting area.

Instructions to raise or lower the pen, either before or after movement, can be easily included in several other statements (see PLOT and IPLOT) so there is no special lower pen instruction.

THE OFFSET STATEMENT

OFFSET X, Y

Example: OFFSET **3, –3**

Temporarily offsets the origin (point **0, 0** established by the previous SCALE statement) by an amount, and in the direction, determined by the values (in user-units) of X and Y. All future plotting commands are then made with respect to the new origin until such time as that origin is again changed by means of, for example, a new OFFSET or a new SCALE statement.

The OFFSET statements are not accumulative; that is, a new offset is with respect to the original origin and not with respect to the last offset origin.

Offsetting greatly simplifies plotting, from the user's point of view, when it becomes necessary to divide the plotting area into several smaller segments and then make a separate plot in each segment. As the plot is made in each segment it is not necessary for the user to correct each point before plotting; instead OFFSET statement moves the origin to some convenient point within that segment so that the calculator automatically makes the necessary corrections for each point plotted.

THE AXIS STATEMENT

X AXIS Y-offset [, ±tic [, start point, end point]]

or

Y AXIS X-offset [, ±tic [, starting point, end point]]

X AXIS **3, 1, –4, 4**

DRAWS an X-(or Y-) axis according to the parameters given in the AXIS statement. The pen is automatically raised both before and after drawing the axis.

(NOTE: The following describes the X-axis; the same information is applicable to the Y-axis if left and right for the X-axis are read as, respectively, bottom and top for the Y-axis.)

1. If no optional parameters are given, draws a straight line from left to right across the complete plotting area (from Xmin to Xmax). The line crosses the Y-axis at a point determined by the value of y-offset.
2. If a tic parameter is included then tic marks are made along the axis as it is drawn; the value for tic determines the spacing, in user-units, between tics. The first tic is drawn at the starting point of the line. The tic parameter is usually positive (the sign is not required), but occasionally a negative tic spacing is useful — see 4, below.
3. If the start point/end point parameters are given, then the axis is drawn only between the points specified — from the start point to the end point.
4. a. A negative tic spacing when no start point/end point parameters are given results in a tic only at the left end (Xmin) of the axis.
   b. If the start point parameter is more positive than (i.e., to the right of) the end point parameter, then the axis is drawn from right to left; in this case, negative tic spacing results in normal tic marks being drawn along the axis.
   c. With the start point/end point parameters the same as in *b* above, a positive tic spacing results in a tic only at the right end (Xmax) of the axis.

THE PLOT STATEMENT

PLOT X, Y [, control pen]

PLOT SIN(X), COS(X), **–2**

Moves the pen to the co-ordinate specified by the value of X and Y.

When no optional control pen parameter is given:

If the pen was raised, it moves to the point specified and then lowers.

If the pen was lowered, it remains lowered while moving to the point specified, thus drawing a line on the plotting surface.

The Control Pen Parameter

The value and sign of this parameter in the PLOT (and IPLOT) statements determines whether the pen will be raised or lowered before or after it moves to the specified point.

If the parameter is:

negative — control occurs after movement;

positive — control occurs before movement;

odd — raises pen if it was lowered;
even — lowers pen if it was raised.

The value of the control parameter can be any number in the range ±32,767. If the value is not an integer then it is automatically rounded up or down according to the value of the fractional part of the number; that is up for .5 or greater, or down for less then .5. (Rounding is the same as the standard rounding in the calculator; it is not the same as the INT function, where the value becomes that of the next lower integer.)

THE IPLOT STATEMENT

IPLOT deltaX, deltaY [, control pen]

IPLOT 2, −3A/4, 1

Moves the pen (from its current position) in the X direction and in the Y direction, by the amounts specified by deltaX and deltaY, respectively.

The control pen parameter is optional and operates exactly as described previously — see the PLOT STATEMENT.

Notice that the action of the IPLOT statement is such that it is as if, during the execution of that statement only, the origin (**0, 0**) of the graph is offset to the current position of the pen. Pen movement is then related to that offset origin.

The IPLOT statement is most useful when drawing regular geometric shapes such as, for example, a swastika (卍). In this case each point is more easily plotted with respect to the previous point, rather than with respect to the origin of the graph.

THE LABEL STATEMENT

LABEL (FORMAT statement number or *[, character heigth in %, Aspect ratio, angle of rotation [, paper heigth/paper width]]) print list

Examples: LABEL (x, **2, 2, 9, 0, 8.5/11**) "PLOTTER"

LABEL (**100**) **1**, A, SIN(X)

The LABEL statement is used to write alpha and numeric characters with the plotter. Several parameters are allowed which can be used to control the size, shape, and angle of rotation of the character printed. The character height can be specified in percent of the paper height. The aspect ratio is the ratio of the character height to the character width before any rotation. The angle of rotation of characters printed can be given. This parameter can be given in degrees, radians, or grads and is dependent on a previously given DEG, RAD, or GRAD statement. A fourth parameter can be specified. This is the ratio of the actual measured height of the plot paper to the measured width as set by the upper right and lower left positions. This parameter is necessary to keep proper aspect ratio of characters printed on an angle on a non-square plot. If not specified, the calculator will assume character height = 2.5%, aspect ratio = 2, rotation = 0, and paper ratio = 1.

The format specification and the print list are the same as a WRITE statement. If a FORMAT statement number is specified, the print list is written on the plotter using the specification given in the FORMAT statement. All specifications are allowed except B. If an * is used, the print list will be written according to standard format. This includes the normal definition of comma and semi-colon spacing, string fields, TAB, etc.

The LABEL statement will start printing characters at the current pen position. Anytime an end of line is needed, the plotter pen will return to the character position directly below the first character of the current line, simulating a carriage return, line feed.

If the string variables option block is also plugged into the memory, then string variables are allowed in the print list.

THE LETTER STATEMENT

LETTER

When the LETTER statement is executed, the calculator enters a unique typewriter mode with the plotter as the printing device. While in this mode when the user hits any printing character on the keyboard, that character is immediately printed on the plotter at the current pen position. An EOL or EXECUTE will cause a carriage return, line-feed to be simulated. In addition, while in this mode the ↑, ↓, ←, and → keys can be used to position the pen. The ↑ and ↓ keys can cause the pen to move up or down one character position. The ← and → keys cause the pen to move left or right one character position. If the shift key is held down at the same time as a pen control key, the pen will move one-tenth of a character position. Character size, aspect ratio, and rotation can be specified by giving a LABEL statement prior to the LETTER statement.

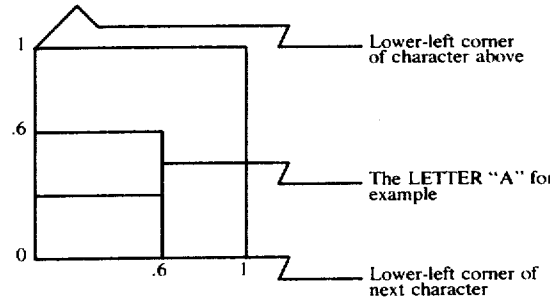
THE CPLOT STATEMENT

CPLOT delta X characters, delta Y characters

Example: CPLOT **5**, −.3

The CPLOT statement is similar to the IPLOT statement in that it moves the pen to a position relative to the current pen position. The difference is that the delta X and delta Y values are specified in character size units. In the example above the pen would move five character position down. This statement is particularly useful in positioning the pen when labeling a plot using the LABEL statement.

One character space is defined as follows:



The error messages given by the plotter module are:

ERROR **80** — No scale statement executed before PLOT, IPLOT, OFFSET, or AXIS.

ERROR **81** — Character size too large (limited to = 20%) or binary mode not allowed.

ERROR **82** — OFFSET, POINT **1**, or POINT **2** out of range during axis execution or tic increment in axis statement is too small.

MATRIX PLUG-IN READ-ONLY MEMORY MODULE

The Matrix read-only-memory module enables the calculator to understand the MAT statements of BASIC. These statements facilitate the matrix operations of addition, subtraction, initialization, scalar multiplication, matrix multiplication transposition, and inversion. A function additional to standard BASIC language matrix operations is provided for computing the determinate of a square matrix. The MAT READ and MAT PRINT commands facilitate entering data into an

array and the printing thereof. The MAT INPUT command which appears in some versions of BASIC is not allowed. The matrix operations are allowed on split precision or integer arrays as well as full floating point arrays.

The operations performed by this matrix module are summarized below and additional information is provided above in Section IV, Matrices.

A. MAT READ
reads numeric information from DATA statements into an array
MAT READ A
MAT READ A(3,5)
MAT READ B, W(8),X,Y

B. MAT PRINT
prints complete arrays
close packing specified with a semicolon
MAT PRINT A
MAT PRINT A;
MAT PRINT R,S;V;W

C. ZER, CON, and IDN operations
ZER: initialize an array to all zeros
CON: initialize an array to all ones
IDN: initialize an array to the identity matrix
MAT A = ZER
MAT Q = CON(9)
MAT I — IDN(4,4)

D. MAT assignment statements
assign to a matrix the result of a MAT operation
1. MAT A = B
assigns elements of A from array B
2. MAT A = B ± C
performs indicated addition or subtraction and assigns result to A
any of A, B, and C may be the same matrix
3. MAT A = B * C
performs indicated multiplication and assigns result to A array A must be distinct from B or C
4. MAT A = (expression) * B
performs the indicated scalar multiply (each element of B is multiplied by the value of the expression) and assigns the result to A
5. MAT A = TRN(B)
assigns the transpose of B to A
array A must be distinct from B
6. MAT A = INV(B)
assigns the inverse of B to A
A and B may be the same matrix

E. DET operation
calculates the determinant of a square matrix
MAT D4 = DET(A)

KEY CODES AND MNEMONICS

All of the keys of the keyboard input unit and their associated mnemonics and binary keycodes are listed in the Table below. Every key has one mnemonic and two keycodes, (namely, a shifted keycode and an unshifted keycode). Keycodes are applied to the CPU in eight-bit binary form. The first four bits of each keycode are given in the left-most vertical column of the table below and the next three bits of each keycode are given in the uppermost row of the table below. The eight bit of each keycode is the shift bit and is determined by whether or not the shift key of the keyboard input unit is depressed. Keycodes entered into the CPU from the keyboard input unit or from the program storage section of the memory unit are processed by the keyboard input routine **204** as generally described above in connection with FIG. **9** and as shown in detail in FIGS. **88A–G**.

KEYCODE AND MNEMONIC TABLE

	D0	D1	D2	D3	D4	D5	D6	D7
b6	0	0	0	0	1	1	1	1
b5	0	0	1	1	0	0	1	1
b4	0	1	0	1	0	1	0	1
b3b2b1b0								
0 0 0 0	f0	RECALL	SPACE	0		P		PA
0 0 0 1	f1	FETCH		1	A	Q		STOP
0 0 1 0	f2	BACK		2	B	R		EOL
0 0 1 1	f3	FWD		3	C	S		DL
0 1 0 0	f4	↓		4	D	T		FXD
0 1 0 1	f5	↑		5	E	U		FLT
0 1 1 0	f6	←		6	F	V		SCRATCH
0 1 1 1	f7	→		7	G	W		AUTO
1 0 0 0	f8	LOAD	(	8	H	X		
1 0 0 1	f9	STORE	)	9	I	Y		
1 0 1 0	LIST	INIT	X	* :	J	Z	CLR	
1 0 1 1	EXEC	/	+	+ ;	K		RESULT	
1 1 0 0	CONT		,		L			
1 1 0 1	STEP	STD	−	=	M			

KEYCODE AND MNEMONIC TABLE-continued

	D0	D1	D2	D3	D4	D5	D6	D7
b6	0	0	0	0	1	1	1	1
b5	0	0	1	1	0	0	1	1
b4	0	1	0	1	0	1	0	1
1 1 1 0	TRACE	NORMAL	.		N	↑		
1 1 1 1	RUN	INSERT	? /		O		ENTER EXP	

BASIC INSTRUCTION SET

Every routine and subroutine of the calculator comprises a sequence of one or more of 71 basic sixteen-bit instructions listed below. These 71 instructions are all implemented serially by the micro-processor in a time period which varies according to the specific instruction, to whether or not it is indirect, and to whether or not the skip condition has been met.

Upon completion of the execution of each instruction, the program counter (P register) has been incremented by one except for instructions JMP, JSM, and the skip instructions in which the skip condition has been met. The M-register is left with contents identical to the P-register. The contents of the addressed memory location and the A and B registers are left unchanged unless specified otherwise.

memory Reference Group

The 14 memory reference instructions refer to the specific address in memory determined by the address field <*m*>, by the ZERO/CURRENT page bit, and by the DIRECT/INDIRECT bit. Page addressing and indirect addressing are both described in detail in the reference manuals for the Hewlett-Packard Model 2116 computer (hereinafter referred to as the HP 2116).

The address field <*m*> is a 10 bit field consisting of bits 0 through 9. The ZERO/CURRENT page bit is bit **10** and the DIRECT/INDIRECT bit is bit **15,** except for reference to the A or B register in which case bit **8** becomes the DIRECT/INDIRECT bit. An indirect reference is denoted by a <, I> following the address <*m*>.

REGISTER REFERENCE OF A OR B REGISTER: If the location <A> or <B> is used in place of <*m*> for any memory reference instruction, the instruction will treat the contents of A or B exactly as it would be contents of location <*m*>. See the note below on the special restriction for direct register reference of A or B.

ADA *m*,I Add to A. The contents of the addressed memory location *m* are added (binary add) to contents of the A register, and the sum remains in the A register. If carry occurs from bit **15,** the E register is loaded with **0001,** otherwise E is left unchanged.

ADB *m*,I Add to B. Otherwise identical to ADA.

CPA *m*,I compare to A and skip if unequal. The contents of the addressed memory location are compared with the contents of the A register. If the two 16-bit words are different, the next instruction is skipped; that is, the P and M registers are advanced by two instead of one. Otherwise, the next instruction will be executed in normal sequence.

CPB *m*,I Compare to B and skip is unequal. Otherwise identical to CPA.

LDA *m*,I Load into A. The A register is loaded with the contents of the addressed memory location.

LDB *m*,I Load into B. The B register is loaded with the contents of the addressed memory location.

STA *m*,I Store A. The contents of the A register are stored into the addressed memory location. The previous contents of the addressed memory location are lost.

STB *m*,I Store B. Otherwise identical to STA.

IOR *m*,I Inclusive OR to A. The contents of the addressed location are combined with the contents of the A register as an INCLUSIVE OR logic operation.

ISZ *m*,I Increment and Skip if Zero. The ISZ instruction adds ONE to the contents of the addressed memory location. If the result of this operaion is ZERO, the next instruction is skipped, that is, the P and M registers are advanced by TWO instead of ONE. The incremental value is written back into the addressed memory location. Use of ISZ with the A or B register is limited to indirect reference; see footnote on restrictions.

AND *m*,I Logical AND to A. The contents of the addressed location are combined with the contents of the A register as an AND logic operation.

DSZ *m*,I Decrement and Skip if Zero. The DSZ instruction subtracts ONE from the contents of the addressed memory location. If the result of this operation is zero, the next instruction is skipped. The decremented value is written back into the addressed memory location. Use of DSZ with the A or B register is limited to indirect reference; see footnote on restrictions.

JSM *m*,I Jump to Subroutine. The JSM instruction permits jumping to a subroutine in either ROM or R/W memory. The contents of the P register is stored at the address contained in location **1777** (stack pointer). The contents of the stack pointer is incremented by one, and both M and P are loaded with the referenced memory location.

JMP *m*,I Jump. This instruction transfers control to the contents of the addressed location. That is, the referenced memory location is loaded into both M and P registers, effecting a jump to that location.

Shift-Rotate Group

The eight shift-rotate instructions all contain a 4 bit variable shift field <*n*> which permits a shift of one through 16 bits; that is, $1 \leq n \leq 16$. If <*n*> is omitted, the shift will be treated as a one bit shift. The shift code appearing in bits **8,7,6,5** is the binary code for $n-1$, except for SAL and SBL, in which cases the complementary code for $n-1$ is used.

AAR *n* Arithmethic right shift of A. The A register is shifted right *n* places with the sign bit (bit **15**) filling all vacated bit positions. That is, the $n+1$ most significant bits become equal to the sign bit.

ABR *n* Arithmetic right shift of B. Otherwise identical to AAR.

SAR *n* Shift A right. The A register is shifted right *n* places with all vacated bit positions cleared. That is, the *n* most significant bits become equal to zero.

SBR *n* Shift B right. Otherwise identical to SAR.

SAL *n* Shift A left. The A register is shifted left *n* places with the *n* least significant bits equal to zero.

SBL *n* Shift B left. Otherwise identical to SAL.

RAR *n* Rotate A right. The A register is rotated right *n* places, with bit **0** rotated around to bit **15.**

RBR *n* Rotate B right. Otherwise identical to RAR.

Alter-Skip Group

The sixteen alter-skip instructions all contain a 5-bit variable skip field <*n*> which, upon meeting the skip condition, permits a relative branch to any one of 32 locations. Bits **9,8,7,6,5** are coded for positive or negative relative branching in which the number <*n*> is the number to be added to the current address, (skip in forward direction), and the number <−*n*> is the number to be subtracted from the current address, (skip in negative direction). If <*n*> is omitted, it will be interpreted as a ONE.

<n>=0	CODE=00000	REPEAT SAME INSTRUCTION
<n>=1	CODE=00001	DO NEXT INSTRUCTION
<n>=2	CODE=00010	SKIP ONE INSTRUCTION
<n>=15	CODE=01111	ADD 15 TO ADDRESS
<n>=−1	CODE=11111	DO PREVIOUS INSTRUCTION
<n>=−16	CODE=10000	SUBTRACT 16 FROM ADDRESS
<n>=nothing	CODE=00001	DO NEXT INSTRUCTION

The alter bits consist of bits **10** and bits **4.** The letter <S> following the instruction places a ONE in bit **10** which causes the tested bit to be set after the test. Similarly the letter <C> will place a ONE In bit **4** to clear the test bit. If both a set and clear bit are given, the set will take precedence. Alter bits do not apply to SZA, SZB, SIA, and SIB.

SZA *n* Skip if A zero. If all 16 bits of the A register are zero, skip to location defined by *n*.

SZB *n* Skip if B zero. Otherwise identical to SZA.

RZA *n* Skip if A not zero. This is a Reverse Sense skip of SZA

RZB *n* Skip if B not zero. Otherwise identical to RZA.

SIA *n* Skip if A zero; then increment A. The A register is tested for zero, then incremented by one. If all 16 bits of A were zero before incrementing, skip to location defined by *n*.

SIB *n* Skip if B zero; then increment B. Otherwise identical to SIA.

RIA *n* Skip if A not zero; then increment A. This is a Reverse Sense skip of SIA.

RIB *n* Skip if B not zero; then increment B. Otherwise identical to RIA.

SLA *n*,S/C Skip if Least Significant bit of A is zero. If the least significant bit (bit **0**) of the A register is zero, skip to location defined by n. If either S or C is present, the test bit is altered accordingly after test.

SLB *n*,S/C Skip if Least Significant bit of B is zero. Otherwise identical to SLA.

SAM *n*,S/C Skip if A is Minus. If the sign bit (bit **15**) of the A register is a ONE, skip to location defined by *n*. If either S or C is present, bit **15** is altered after the test.

SBM *n*,S/C Skip if B is Minus. Otherwise identical to SAM.

SAP *n*,S/C Skip if A is Positive. If the sign bit (bit **15**) of the A register is a ZERO, skip to location defined by *n*. If either S or C is present, bit **15** is altered after the test.

SBP *n*,S/C Skip if B is Positive. Otherwise identical to SAP.

SES *n*,S/C Skip if Least Significant bit of E is Set. If bit O of the E register is a ONE, skip to location defined by *n*. If either S or C is present, the entire E register is set or cleared respectively.

SEC *n*,S/C Skip if Least Significant bit of E is Clear. If bit **0** of the E register is a ZERO, skip to location defined by *n*. If either S or C is present, the entire E register is set or cleared respectively.

Complement-Execute-DMA Group.

These seven instructions include complement operations and several special-purpose instructions chosen to speed up printing and extended memory operations.

CMA Complement A. The A register is replaced by its One's complement.

CMB Complement B. The B register is replaced by its One's complement.

TCA Two's Complement A. The A register is replaced by its One's Complement and incremented by one.

TCB Two's complement B. The B register is replaced by its One's Complement and incremented by one.

EXA Execute A. The contents of the A register are treated as the current instruction, and excuted in the normal manner. The A register is left unchanged unless the instruction code causes A to be altered.

EXB Execute B. Otherwise identical to EXA.

DMA Direct Memory Access. The DMA control in Extended Memory is enabled by setting the indirect bit in M and giving a WTM instruction. The next ROM clock transfers A → M and the following two cycles transfer B → M. ROM clock then remains inhibited until relased by DMA control.

Note: Special Restriction for Direct Register Reference of A or B

For the five register reference instructions which involve a write operation during execution, a register reference to A or B must be restricted to an INDIRECT reference. These instructions are STA, STB, ISZ, DSZ, and JSM. A DIRECT register reference to A or B with these instructions may result in program modification. (This is different from the hp **2116** in which a memory reference to the A or B register is treated as a reference to locations **0** or **1** respectively.) A reference to location **0** or **1** will actually refer to locations **0** or **1** in Read Only Memory.

Input/Output Group (IOG)

The eleven IOG instructions, when given with a select code, are used for the purpose of checking flags, setting or clearing flag and control flip-flops, and transferring data between the A/B registers and the I/O register.

STF <SC> Set the flag. Set the flag flip-flop of the channel indicated by select code <SC>.

CLF <SC> Clear the flag flip-flop of the channel indicated by select code <SC>.

SFC <SC> Skip if flag clear. If the flag flip-flop is clear in the channel indicated by <SC>, skip the next instruction.

SFS <SC> H/C Skip if flag set. If the flag flip-flop is set in the channel indicated by <SC>, skip the next instruction. H/C indicates if the flag flip-flop should be held or cleared after executing SFS.

CLC <SC> H/C Clear control. Clear the control flip-flop in the channel indicated by <SC>. H/C indi-

cates if the flag flip-flop should be held or cleared after executing CLC.

STC <SC> H/C Set Control. Set the control flip-flop in the channel indicated by <SC>. H/C indicates if the flag flip-flop should be held or cleared after executing STC.

OT* <SC> H/C Output A or B. Sixteen bits from the A/B register are output to the I/O register. H/C allows holding or clearing the flag flop after execution of OT*. The different select codes allow different functions to take place after loading the I/O register.

SC=00 Data from the A or B register is output eight bits at a time for each OT* instruction given. The A or B register is rotated right eight bits.

SC=01 The I/O register is loaded with 16 bits from the A/B registers.

SC=02 Data from the A/B register is output one bit at a time for each OT* instruction for the purpose of giving data to the Magnetic Card Reader. The I/O register is unchanged.

SC=04 The I/O register is loaded with 16 bits from the A/B register and the control flip flop for the printer is then set.

SC=08 The I/O register is loaded with 16 bits from the A/B register and the control flip flop for the display is then set.

SC=16 The I/O register is loaded with 16 bits from the A/B register and then data in the I/O register is transferred to the switch latches.

LI* <01> H/C Load into A or B. Load 16 bits of data into the A/B register from the I/O register. H/C allows holding or clearing the flag flop after L1* has been executed.

LI* <00> The least significant 8 bits of the I/O register are loaded into the most significant locations in the A or B register.

MI* <01> H/C Merge into A or B. Merge 16 bits of data into the A/B register from the I/O register by inclusive or. H/C allows holding or clearing the flag flop after MT* has been executed.

MI* <00> The least significant 8 bits of the I/O register are combined by inclusive OR with the least significant 8 bits of the A or B register, and rotated to the most significant bit locations of the A or B register.

MAC Instruction Group

A total of 16 MAC instructions are available for operation

a. with the whole floating-point data (like transfer, shifts, etc), or

b. with two floating-point data words to speed up digit and word loops in arithmethic routines.

NOTE: $<A_{0-3}>$ means: contents of A-register bit 0 to 3

AR 1 is a mnemonix for arithmetic pseudo-register located in R/W memory on addresses 1744 to 1747 (octal)

AR 2 is a mnemonix for arithmethic pseudo-register located in R/W memory on addresses 1754 to 1757 (octal)

D_i means: mantissas i-th decimal digit;
most significant digit is D1
least significant digit is D12
decimal point is located between D1 and D2

Every operation with mantissa means BCD-coded decimal operation.

RET Return
16-bit-number stored at highest occupied address in stack is transferred to P- and M-registers. Stack pointer (=next free address in stack) is decremented by one. <A>, <B>, <E> unchanged.

MOV Move overflow
The contents of E-register is transferred to A_{0-3}. Rest of A-register and E-register are filled by zeros. <B> unchanged.

CLR Clear a floating-point data register in R/W memory on location <A>
ZERO → <A>, <A>+1, <A>+2, <A>+3
<A>, >B>, <E> unchanged

EXF Floating-point data transfer in R/W memory from location <A> to location <B>.
Routine starts with exponent word transfer.
Data on location <A> is unchanged.
<E> unchanged.

MRX AR1 mantissa is shifted to right n-times. Exponent word remains unchanged.

$<B_{0-3}> = n$ (binary coded)

1st shift: $<A_{0-3}> \rightarrow D_1$; $D_i \rightarrow D_{i+1}$; D_{12} is lost
jth shift: $\theta \rightarrow D_1$; $D_i \rightarrow D_{i+1}$; D_{12} is lost
nth shift: $\theta \rightarrow D_1$; $D_i \rightarrow D_{i+1}$; $D_{12} \rightarrow A_{0-3}$
$\theta \rightarrow E, A_{4-15}$
each shift: $<B_{0-3}> - 1 \rightarrow B_{0-3}$
$<B_{4-15}>$ unchanged

MRY AR2 mantissa is shifted to right n-times. Otherwise identical to MRX

MLS AR2 mantissa is shifted to left once. Exponent word remains unchanged.
$\theta \rightarrow D_{12}$; $D_i \rightarrow D_{i-1}$; $D_1 \rightarrow A_{0-3}$
<B> unchanged

DRS AR1 mantissa is shifted to right once Exponent word remains unchanged
$\theta \rightarrow D_1$; $D_1 \rightarrow D_{i+1}$; $D_{12} \rightarrow A_{0-3}$
ZERO → E and A_{4-15}
<B> unchanged

DLS AR1 mantissa is shifted to left once. Exponent word remains unchanged.
$<A_{0-3}> \rightarrow D_{12}$; $D_i \rightarrow D_{i-1}$; $D_1 \rightarrow A_{0-3}$
$\theta \rightarrow E, A_{4-15}$
<B> unchanged

FXA Fixed-point addition Mantissas in pseudo-registers AR2 and AR1 are added together and result in placed into AR2. Both exponent words remain unchanged. When overflow occurs 0001 is set into E-reg., in opposite case <E> will be zero.
<AR2> + <AR1> + DC → AR2
DC = θ if <E> was 0000 before routine execution
DC = 1 if <E> was 1111 before routine execution
<B>, <AR1> unchanged

FMP Fast multiply
Mantissas in pseudo-registers AR2 and AR1 are added together $<B_{0-3}>$-times and result is placed into AR2. Total decimal overflow is placed to A_{0-3}. Both exponent words remain unchanged.
<AR2> + <AR1> * $<B_{0-3}>$+DC → AR2
DC = 0 if <E> was 0000 before routine execution
DC = 1 if <E> was 1111 before routine execution
ZERO → E, A_{4-15}
<AR1> unchanged

FDV Fast divide
Mantissas in pseudo-registers AR2 and AR1 are added together so many times until first decimal overflow occurs. Result is placed into AR2. Both

exponent words remain unchanged. Each addition without overflow causes +1 increment of <B>.

1st addition: <AR2> + <AR1> + DC → AR2

DC = 0 if <E> was 0000 before routine execution

DC 32 1 if <E> was 1111 before routine execution

next additions: <AR2> + <AR1> → AR2

ZERO → E

<AR1> unchanged

MX 10's complement of AR1 mantissa is placed back to AR1, and ZERO is set into E-register. Exponent word remains unchanged

<B> unchanged

MY 10's complement of AR2 mantissa.

Otherwise identical to CMY

IDI Mantissa decimal increment.

Mantissa on location <A> is incremented by decimal ONE on D_{12} level, result is placed back into the same location, and zero is set into E-reg.

Exponent word is unchanged.

When overflow occurs, result mantissa will be 1,000 0000 0000 (dec)

and 0001 (bin) will be set into E-reg.

<B> unchanged.

NRM Normalization

Mantissa in pseudo-register AR2 is rotated to the left to get $D_1 \neq 0$. Number of these 4-bit left shifts is stored in B_{0-3} in binary form ($<B_{4-15}>=0$)

when $<B_{0-3}> = 0,1,2,\ldots,11$ (dec) ⇒ <E> = 0000

When $<B_{0-3}> = 12$ (dec) ⇒ mantissa is zero, and <E>= 0001

Exponent word remains unchanged

<A> unchanged.

The binary codes of all of the above instructions are listed in the following coding table, where * implies the A or B register, D/I means direct/indirect, A/B means A register/B register, Z/C means zero page (base page) (current page, H/S means hold test bit/set test bit, and H/C means hold test bit/clear test bit. D/I, A/B, Z/C, H/S and H/C are all coded as 0/1.

CODING TABLE

GROUP	OCTAL	INSTR	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEMORY	-0----	AD*	D/I	0	0	0	A/B	Z/C	← MEMORY ADDRESS →									
REFERENCE	-1----	CP*	D/I	0	0	1	A/B	Z/C										
GROUP	-2----	LO*	D/I	0	1	0	A/B	Z/C										
	-3----	ST*	D/I	0	1	1	A/B	Z/C										
	-4----	IOR	D/I	1	0	0	0	Z/C										
	-4----	ISZ	D/I	1	0	0	1	Z/C										
	-5----	AND	D/I	1	0	1	0	Z/C										
	-5----	DSZ	D/I	1	0	1	1	Z/C										
	-6----	JSM	D/I	1	1	0	0	Z/C										
	-6----	JMP	D/I	1	1	0	1	Z/C										
SHIFT-	07---0	A*R	0	1	1		1	A/B	-		-	← SHIFT CODE →			0	0	0	0
ROTATE	07---2	S*R	0	1	1		1	A/B	-		-				-	0	0	1 0
GROUP	07---4	S*L	0	1	1		1	A/B	-		-				-	0	1	0 0
	07---6	R*R	0	1	1		1	A/B	-		-				-	0	1	1 0
ALTER-	07---0	SZ*	0	1	1	1	A/B	0	← SKIP CODE →					0	1	0	0	0
SKIP	07---0	RZ*	0	1	1	1	A/B	1						0	1	0	0	0
GROUP	07---0	SI*	0	1	1	1	A/B	0						1	1	0	0	0
	07---0	RI*	0	1	1	1	A/B	1						1	1	0	0	0
	07---1	SL*	0	1	1	1	A/B	H/S						H/C	1	0	0	1
	07---2	S*M	0	1	1	1	A/B	H/S						H/C	1	0	1	0
	07---3	S*P	0	1	1	1	A/B	H/S						H/C	1	0	1	1
	07---4	SES	0	1	1	1	A/B	H/S						H/C	1	1	0	0
	07---5	SEC	0	1	1	1	A/B	H/S						H/C	1	1	0	1
D/I	07--17	ADA	0	1	1	1	A/B	-	-	D/I	0	0	0	0	1	1	1	1
REFERENCE	07--37	ADB	0	1	1	1	A/B	-	-	D/I	0	0	0	1	1	1	1	1
GROUP	07--57	CPA	0	1	1	1	A/B	-	-	D/I	0	0	1	0	1	1	1	1
	07--77	CPB	0	1	1	1	A/B	-	-	O/I	0	0	1	1	1	1	1	1
	07--17	LDA	0	1	1	1	A/B	-	-	D/I	0	1	0	0	1	1	1	1
	07--37	LDB	0	1	1	1	A/B	-	-	D/I	0	1	0	1	1	1	1	1
	07-557	STA	0	1	1	1	A/B	-	-	1	0	1	1	0	1	1	1	1
	07-577	STB	0	1	1	1	A/B	-	-	1	0	1	1	1	1	1	1	1
	07--17	IOR	0	1	1	1	A/B	-	-	D/I	1	0	0	0	1	1	1	1
	07-637	ISZ	0	1	1	1	A/B	-	-	1	1	0	0	1	1	1	1	1
	07--57	AND	0	1	1	1	A/B	-	-	D/I	1	0	1	0	1	1	1	1
	07-677	DSZ	0	1	1	1	A/B	-	-	1	1	0	1	1	1	1	1	1
	07-717	JSM	0	1	1	1	A/B	-	-	1	1	1	0	0	1	1	1	1
	07--37	JMP	0	1	1	1	A/B	-	-	D/I	1	1	0	1	1	1	1	1
COMP	07-016	EX*	0	1	1	1	A/B	-	-	-	-	-	0	0	1	1	1	0
EXECUTE	070036	DMA	0	1	1	1	0	-	-	-	-	-	0	1	1	1	1	0
DMA	07-056	CM*	0	1	1	1	A/B	-	-	-	-	-	1	0	1	1	1	0
	07-076	TC*	0	1	1	1	A/B	-	-	-	-	-	1	1	1	1	1	0
INPUT	1727--	STF	1	1	1	1	-	1	0	1	1	1	1	← SELECT CODE →				
OUTPUT	1737--	CLF	1	1	1	1	-	1	1	1	1	1	1					
GROUP	17-7--	SFC	1	1	1	1	-	1	H/C	1	1	1	0					
	17-5--	SFS	1	1	1	1	-	1	H/C	1	0	1	0					
	17-5--	CLC	1	1	1	1		1	H/C	1	0	1	1					
	17-6--	STC	1	1	1	1	-	1	H/C	1	1	0	0					
	17-1--	OT*	1	1	1	1	A/B	1	H/C	0	0	1	1					
	17-2--	LI*	1	1	1	1	A/B	1	H/C	0	1	0	1					
	17-0--	MI*	1	1	1	1	A/B	1	H/C	0	0	0	1					
MAC	170402	RET	1	1	1	1	0	0	0	1	0	0	0	0	0	0	1	0
GROUP	170002	MOV	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0

CODING TABLE-continued

GROUP	OCTAL	INSTR	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	170000	CLR	1	1	1	1	0	0	0	00	0	0	0	0	0	0	0	
	170004	XFR	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0
	174430	MRX	1	1	1	1	1	0	0	1	0	0	0	1	1	0	0	0
	174470	MRY	1	1	1	1	1	0	0	1	0	0	1	1	1	0	0	0
	171400	MLS	1	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0
	170410	DRS	1	1	1	1	0	0	0	1	0	0	0	0	1	0	0	0
	175400	DLS	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0
	170560	FXA	1	1	1	1	0	0	0	1	0	1	1	1	0	0	0	0
	171460	FMP	1	1	1	1	0	0	1	1	0	0	1	1	0	0	0	0
	170420	FDV	1	1	1	1	0	0	0	1	0	0	0	1	0	0	0	0
	174400	CMX	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0
	170400	CMY	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0
	170540	MDI	1	1	1	1	0	0	0	1	0	1	1	0	0	0	0	0
	171450	NRM	1	1	1	1	0	0	1	1	0	0	1	0	1	0	0	0

DETAILED LISTING OF ROUTINES AND SUBROUTINES OF BASIC INSTRUCTIONS

A complete listing of all of the routines and subroutines of basic instructions employed by the calculator and of all of the constants employed by these routines and subroutines is given below. All of these routines, subroutines, and constants are stored either in the basic ROM or in the plug-in ROM modules employed therewith. Each page within the listing is numbered at the upper left-hand corner, and its number within the specification as a whole is indicated at the bottom of the page. Each line of each page is separately numbered in the first column from the left-hand side of the page. This facilitates reference to different parts of the listing. Descriptive headings are also provided throughout the listing to identify routines, subroutines, groups of constants, different portions of the ROM, the plug-in ROM modules, etc. Each instruction of each routine or subroutine and each constant stored in the ROM or plug-in ROM modules is represented in octal form by six digits in the third column from the left-hand side of the page, and the address of the ROM location in which each such instruction or constant is stored is represented in octal form by five digits in the second column from the left-hand side of the page.

Mnemonic labels serving as symbolic addresses or names are given in the fourth column from the left-hand side of the page for most of the constants and many of the instructions to facilitate references to these constants and instructions and associated instructions. the mnemonic code of each basic instruction and of each pseudo instruction is given in the fifth column from the left-hand side of the page. As noted above, each basic instruction is employed as a step in a routine or subroutine of one or more basic instructions and therefore has an address in the ROM. Pseudo instructions such as ORG, EQU, etc. which appear (and are recognizable as not being one of the 71 basic machine instructions listed above) are used for control of the Assembler, which translates the symbolic/mnemonic coding of the fourth, fifth, and sixth columns into the address and contents of ROM registers which appear in the second and third columns. (See chapter 4 of the Hewlett-Packard "Assembler Programmer's Reference Manual" of April, 1970.) They are not employed as steps in the routines and subroutines performed by the calculator and therefore have no addresses in the ROM. Mnemonic operand codes are given in the sixth column from the left-hand side of the page, and descriptive comments are given to the right of the sixth column. The format, assembly, and use of the listing is explained in greater detail in the above-mentioned Hewlett-Packard "Assembler Programmer's Reference Manual".

In addition, cross-reference symbol tables are included directly below each group of routines and subroutines for providing an alphabetical listing of each of the mnemonic labels and operands. The number contained in the first column to the right of each mnemonic in the cross-reference table represents the line in the associated routine or subroutine listing at which that mnemonic appears as a label in column four. Subsequent columns beyond the first column to the right of each mnemonic in the cross-reference table represent the line in the associated routine or subroutine listing at which that mnemonic appears as an operand in column six.

```
#01  9830 MAIN SYSTEM SOFTWARE

#001                    ASMB,A,L,C
#003*  MAIN SYSTEM FIRMWARE LISTING  3-27-72
#004*
#005  00000                    ORG 0
#006*
#007  00000 164121             JMP START,I    START OF MONITOR
#008  00001 043060             OCT 43060      CONSTANT NEEDED BY PROCESSOR
#009  00002 161463             JSM INTFL,I    INTERRUPT
#010*
#011  00003 000000 C6          OCT 0          ONE
#012  00004 #10000             OCT #10000     0000
#013  00005 000000             OCT 0          0000
#014*
#015  00006 000000 C5          OCT 0000000    P1/2
#016  00007 012560             OCT 12560      1570
#017  00010 074543             OCT 074543     7963
#018  00011 #23200             OCT #23200     2680
#019*
#020  00012 000001 .1          DEC 1
#021*
#022  00013 000400 C7          OCT 000400     1RM/PT
#023  00014 #53451             OCT 053451     5729
```

#01 9838 MAIN SYSTEM SOFTWARE

```
0024  00015 053571        OCT 053571     5779
0025  00016 050450        OCT 050450     5128
0026*
0027  00017 000002 .2      DEC 2
0028  00020 000003 .3      DEC 3
0029  00021 000004 .4      DEC 4
0030  00022 000005 .5      DEC 5
0031  00023 000006 .6      DEC 6
0032  00024 000007 .7      DEC 7
0033  00025 000010 .8      DEC 8
0034  00026 000011 .9      DEC 9
0035  00027 000013 .11     DEC 11
0036  00030 000014 .12     DEC 12
0037  00031 000015 CRTN    OCT 15
0038  00031         EOL    EQU CRTN
0039  00031         .13    EQU CRTN
0040*******************
0041* POINTERS TO FUNCTIONS CONSTANTS
0042*
0043  00027         OPERR  EQU .11        POINTER TO 180/PI
0044  00023         PI/2   EQU .6         POINTER  TO PI/2
0045  00020         ONE    EQU .3         POINTER TO THE CONSTANT ONE
0046*
0047*******************
0048  00032 000017 .15     DEC 15
0049  00033 000020 .16     DEC 16
0050  00034 000025 .21     DEC 21
0051  00035 000026 .22     DEC 22
0052  00036 000027 .23     DEC 23
0053  00037 000034 .28     DEC 28
0054  00040 000037 .31     DEC 31
0055  00041 000040 .32     DEC 32
0056  00042 000041 .33     DEC 33
0057  00043 000042 .34     DEC 34
0058  00044 000045 .37     DEC 37
0059  00045 000050 .40     DEC 40
0060  00046 000051 .41     DEC 41
0061  00047 000053 .43     DEC 43
0062  00050 000055 .45     DEC 45
0063  00051 000056 .46     DEC 46
0064  00052 000060 .48     DEC 48
0065  00053 000072 .58     DEC 58
0066  00054 000077 .63     DEC 63
0067  00055 000111 I       OCT 111
0068  00056 023420 .1E4    DEC 10000
0069  00057 031750 .1000   DEC 1000
0070  00060 000144 .100    DEC 100
0071  00061 000012 .10     DEC 10
0072  00061         LNFD   EQU .10
0073  00062 100000 FLGBT   OCT 100000
0074  00063 000143 .99     DEC 99
0075  00064 100004 INTFL   OCT 100004
0076  00065 076000 OPMSK   OCT 76000
0077  00066 077777 INF     OCT 77777
0078  00067 100037 TYPFL   OCT 100037
0079  00070 101777 OPDMK   OCT 101777
0080  00071 000105 E       OCT 105
0081  00072 000106 F       OCT 106
0082  00073 000123 S       OCT 123
0083  00032         A17    EQU .15
0084  00033         A20    EQU .16
0085  00074 000022 A22     OCT 22
0086  00075 000023 A23     OCT 23
0087  00035         A26    EQU .22
0088  00036         A27    EQU .23
0089  00076 000036 A36     OCT 36
0090  00040         A37    EQU .31
0091  00043         A42    EQU .34
0092  00277 000044 A44     OCT 44
0093  00100 000052 A52     OCT 52
0094  00053         A72    EQU .58
0095  00101 000100 A100    OCT 100
0096  00102 000133 A133    OCT 133
0097  00103 000140 A140    OCT 140
0098  00104 000177 A177    OCT 177
0099  00105 000377 A377    OCT 377
0100  00013         A400   EQU C7
0101  00106 000777 A777    OCT 777
0102  00107 001000 A1000   OCT 1000
0103  00110 001400 A1400   OCT 1400
0104  00111 001740 A1740   OCT 1740
0105  00112 001760 A1760   OCT 1760
0106  00113 001774 A1774   OCT 1774
0107  00114 001775 A1775   OCT 1775
0108  00115 001777 A1777   OCT 1777
0109  00116 002000 QTOP    OCT 2000
0110  00117 002400 A2400   OCT 2400
0111  00120 003000 A3000   OCT 3000
0112  00121 004000 A4000   OCT 4000
0113  00122 006000 HAK     OCT 6000
```

#01 9830 MAIN SYSTEM SOFTWARE

```
#114  #0004          B10K  EQU C6+1
#115  #0004          RPOP  EQU B10K
#116  #0123 #12000#  B12K  OCT 12000
#117  #0123          R80P  EQU B12K
#118  #0124 #32000   EXPOP OCT 32000
#119  #0125 #50000   L80P  OCT 50000
#120  #0126 #52000   LPOP  OCT 52000
#121  #0127 170000   B170K OCT 170000
#122  #0130 #13777   FROMA OCT 13777
#123  #0131  37777   LROMA OCT 37777B    LAST ROM BLOCK
#124  #0132 177777   M1    DEC -1
#125  #0133 177776   M2    DEC -2
#126  #0134 177775   M3    DEC -3
#127  #0135 177774   M4    DEC -4
#128  #0136 177773   M5    DEC -5
#129  #0137 177772   M6    DEC -6
#130  #0140 177767   M9    DEC -9
#131  #0141 177766   M10   DEC -10
#132  #0142 177765   M11   DEC -11
#133  #0143 177764   M12   DEC -12
#134  #0144 177763   M13   DEC -13
#135  #0145 177762   M14   DEC -14
#136  #0146 177761   M15   DEC -15
#137  #0147 177751   M23   DEC -23
#138  #0150 177747   M25   DEC -25
#139  #0151 177740   M32   DEC -32
#140  #0152 177734   M36   DEC -36
#141  #0153 177720   M48   DEC -48
#142  #0154 177700   M64   DEC -64
#143  #0155 177660   M80   DEC -80
#144  #0156 177634   M100  DEC -100
#145  #0157 177400   M256  DEC -256
#146  #0162 176000   M1024 DEC -1024
#147  #0161 154360   MAXSN DEC -10000     -(MAXIMUM ALLOWABLE LINE NUMBER)
#148  #0162 #43116   EN    ASC 1,EN
#149*
#150*  MATH CONSTANTS THAT ARE NEEDED ON BASE PAGE
#151*
#152  #0163 177401   ZAP   OCT 177401
#153  #0164 #63440   MASK1 OCT 063440
#154  #0165 #61400   MASK5 OCT 061400
#156*
#157  #0166 40440    FWAM  DEF 40440B    FIRST WORD R/W
#158  #0112          RSTKA EQU B1760      START OF RETURN STACK
#159  #0167 #02454   WRITE DEF WRIT       PRINTER AND DISPLAY DRIVER
#160  #0170 #02533   CLERA DEF CLEAR      CLEAR REFRESH BUFFER
#161  #0116          REED  EQU QTOP       KEYBOARD DRIVER QTOP IS = 2000B
#162  #0121          START EQU B4000
#163  #0171 #04741   RUNA  DEF RUN        PHASE 2: SYMBOL TABLE
#164  #0172 #10012   XEC4A DEF XEC4       STATEMENT EXECUTION EXIT
#165  #0173 #04044   PEXMA DEF PEXMK
#166  #0174 #04037   RDYDA DEF RDYPT
#167  #0122          LISTA EQU B6K
#168  #0175 #03072   TSRHA DEF THSRH      TABLE SEARCH
#169  #0176 #07450   ELINA DEF OLINE      OUTPUT I/O BUFFER
#170  #0177 #05305   SSYMA DEF SSYMT
#171  #0200 #11433   OPCHA DEF OPCHK      CHECK FOR UNDEFINED OPERAND
#172  #0201 #10563   FORMA DEF FORMX      EVALUATE FORMULA
#173  #0202 #12040   FSCA  DEF FSC
#174  #0203 #04511   ACTSA DEF ACTST
#175  #0204 #04500   DELSA DEF DELST
#176  #0205 #04355   AERRA DEF AERR
#177  #0206 #04360   ERRA  DEF ERROR
#178  #0207 #00000   INTFA BSS 1          KEYBOARD INTERRUPT LINK
#179  #0210 #00000   OUTIA BSS 1          OUTPUT AN INTEGER
#180  #0211 #03200   .BUFA DEF IOBUF+IOBUF   FIRST CHAR OF I/O BUFFER
#181  #0212 #03322   .SLBF DEF RBUFF+RBUFF   FIRST CHAR OF REFRESH BUFFER
#182  #0213 #01744   AR1A  DEF AR1
#183  #0214 #01754   AR2A  DEF AR2
#184  #0215 #01773   TMPOP DEF RES        RESULT REGISTER POINTER
#185  #0216 #00000   .FADA BSS 1          ADD
#186  #0217 #00000   .FSBA BSS 1          SUBTRACT
#187  #0220 #00000   .FMPA BSS 1          MULTPY
#188  #0221 #00000   .FDVA BSS 1          DIVIDE
#189  #0222 #05447   ENTFA DEF ENTFN
#190  #0223 #00000   .FSR  BSS 1          SQUARE ROOT
#191  #0224 #00000   UPARA BSS 1          UP ARROW
#192  #0225 #00000   FLTRA BSS 1
#193  #0226 #00000   FLTFA BSS 1
#194  #0227 #00000   FXFLA BSS 1
#195  #0230 #00000   FLTSA BSS 1
#196  #0231 #00000   SPLFA BSS 1
#197  #0232 #00000   SNFLA BSS 1
#198  #0233 #01551   BEND  DEF RBUFF      I/O BUFFER END
#199  #0233          SBUFA EQU BEND
#200  #0234 #06353   NMOTA DEF NMOUT      NUMBER OUTPUT ROUTINE
#201  #0235 #07027   FILLA DEF FILL       OUTPUT BLANKS
#202  #0236 #03442   SLEND DEF RBUFF+RBUFF+80   REFRESH BUFFER END
#203  #0111          TMPA  EQU B1740
#204  #0110          BADRP EQU B1400      POINTER POINTER TO I/O BUFFER
```

#01 9830 MAIN SYSTEM SOFTWARE

```
0205*
0206  00123          EQUS EQU B12K
0207  00237 016516 SALTA DEF SALT
0208  00240 005257 MDIMA DEF MDIM
0209  00241 010413 SYTSA DEF SYTSH
0210  00242 005711 DCOMA DEF DCOMP
0211  00243 000000 NUMCA BSS 1
0212  00244 000000 NUMOA BSS 1
0213  00245 007656 GTSTA DEF GETST
0214  00246 012231 VAROA DEF VAROP
0215  00247 010112 ECALA DEF ECAL       EXECUTE CALC MODE
0216  00250 004625 OVCHA DEF OVCHK
0217  00251 004575 CLPIA DEF CLPRI
0218  00252 005416 ENTEA DEF ENTER
0219  00253 007672 CHRSA DEF CHRST
0220  00254 003317 XSCRA DEF XSCRI      SCRATCH KEYS
0221  00255 010040 XEC6A DEF XEC6
0222  00256 010425 FDAAA DEF FDATA
0223  00257 005660 FNDAA DEF FNDAD
0224  00260 005671 FNDIA DEF FNDAI
0225  00261 002547 IOSTA DEF IOST       I/O STATUS CHECK
0226  00262 010467 INRTA DEF INRET
0227  00263 003336 SUMPA DEF SUMP
0228  00264 002504 CRCKA DEF CRCHK      OUTPUT SUPRESSED CRLF
0229  00265 016501 IOHEA DEF IOHED
0230  00266 007240 QTCHA DEF QTCHK      CHECK FOR STRING
0231**********************************
0232*
0233  00267 000000       BSS 1          CHECKSUM WORD****************
```

#01 9830--ERROR TABLE

```
0235  00270 000625 E1    DEF SYSER      ROM CONFIGURATION ERROR
0236  00271 000561       DEF FSCE4      MEMORY OVERFLOW
0237  00272 004177       DEF NCALC      STMT NOT ALLOWED IN CALC MODE
0238  00273 000726       DEF SYE25      MISSING OR BAD INTEGER
0239  00274 004243 E5    DEF NSTMT      NO STATEMENT OR COMMAND RECOGNISED
0240  00275 012134       DEF FSCE3      UNRECOGNISED OPERAND
0241  00276 001152       DEF NOEOF      CHARS FOLLOW STMT LOGICAL END
0242  00277 004307       DEF SYE15      MISSING PRINT DELIMETER
0243  00300 004253       DEF UKERR      IMPROPER TYPING AID ASSIGNMENT
0244  00301 007155 E10   DEF KERR       KEY UNDEFINED
0245  00302 000000       BSS 1
0246  00303 000000       BSS 1
0247  00304 001145       DEF SYE12      SIGN GIVEN BUT NO NUMBER
0248  00305 000511       DEF COMCE+1    MISSING COMMA
0249  00306 000522 E15   DEF FSCE1      NO LEFT PAREN
0250  00307 000535       DEF FSCE2      NO RIGHT PAREN
0251  00310 016432       DEF ARRE1      MISSING SUBSCRIPT
0252  00311 012452       DEF STRNP      STRING NOT PERMITTED
0253  00312 007660       DEF NOST       MISSING OR BAD STRING
0254  00313 007665 E20   DEF SYE14      NO CLOSING QUOTE
0255  00314 012433       DEF SYNE4      MISSING OR BAD FUNCTION I.D.
0256  00315 012515       DEF SYNE5      MISSING PARAMETER
0257  00316 003504       DEF NODATA     MISSING OR BAD DATA ITEM
0258  00317 007723       DEF SYNE7      MISSING THEN
0259  00320 003565 E25   DEF MISOF      MISSING OF
0260  00321 016461       DEF SYE13      MISSING VARIABLE
0261  00322 012444       DEF SYNE8      MISSING FOR VARIABLE
0262  00323 012473       DEF SYNE9      MISSING TO
0263  00324 012500       DEF SYE10      MISSING STEP
0264  00325 012522 E30   DEF SYNE6      MISSING ASSIGNMENT IN FOR STMT
0265  00326 004273       DEF SYNE2      LET STMT HAS NO STORE
0266  00327 003531       DEF FMTER      IMPROPER FORMAT SPEC
0267  00330 016436       DEF SYNE3      MISPLACED COM STATEMENT
0268  00331 005067       DEF CMERR      IMPROPER COM DECLARATION
0269  00332 005037 E35   DEF MERS       ARRAY DOUBLY DEFINED
0270  00333 005372       DEF SSYM1      PRECISION DOUBLY DEFINED
0271  00334 005353       DEF SYME2      INCONSISTENT DIMENSIONS
0272  00335 005157       DEF MER10      ARRAY HAS UNKNOWN DIMENSIONS
0273  00336 005276       DEF MER9       DIMENSIONS TOO LARGE
0274  00337 011440 E40   DEF ER8        OPERAND IS UNDEFINED
0275  00340 005421       DEF NINIT      ARRAY OR STRING NOT INITIALIZED
0276  00341 011240       DEF ER6        SUBSCRIPT EXCEEDS BOUNDS
0277  00342 007372       DEF SCERR      SELECT CODE LESS THAN 1 OR GTR THAN 15
0278  00343 010224       DEF NGOTO      LINE NOT FOUND OR IMPROPER STMT FOUND
0279  00344 007064 E45   DEF TABER      STATEMENTS USING FORMAT CANNOT USE TAB
0280  00345 007414       DEF FNEST      STATEMENTS USING FORMAT CANNOT BE NESTED
0281  00346 010231       DEF ER3        IMPROPER RETURN
0282  00347 010335       DEF BERR       IMPROPER FOR LOOP
0283  00350 010444       DEF ER60       OUT OF DATA
0284  00351 010041 E50   DEF MER11      LAST STMT IS NOT "END"
0285  00352 000000       BSS 1
0286  00353 000000       BSS 1
0287  00354 000000       BSS 1
0288  00355 000000       BSS 1
```

#01 BASE PAGE ROUTINES

```
0290*
0291*  MULTI EXIT POINT RETURNS
0292*
0293  00356 060365 RETN3 JSM RET2       RETURN TO P+3
0294  00357 031401 RETN2 STA RSAVE      RETURN TO P+2
0295  00360 021777       LDA RSPTR
0296  00361 000132       ADA M1
0297  00362 070637       ISZ A,I         INCREMENT RETURN ADDRESS
0298  00363 021401       LDA RSAVE       RESTORE (A)
0299  00364 170402       RET
0300*
0301  00365 031401 RET2  STA RSAVE
0302  00366 021777       LDA RSPTR
0303  00367 000133       ADA M2
0304  00370 070637       ISZ A,I         INCREMENT RETURN ADDRESS
0305  00371 021401       LDA RSAVE       RESTORE (A)
0306  00372 170402       RET
0307*
0308  00373 024212 ICNT  LDB .SLBF       GET SINGLE LINE CHARACTER COUNT
0309  00374 074117 WBUFF LDA B
0310  00375 070076       TCA
0311  00376 001400       ALA HADDR       COMPUTE CHAR COUNT
0312  00377 170402       RET
0313*
0314*  GET A PARAMETER IN (B) FROM P+1
0315*
0316  00400 025777 GPARM LDB RSPTR
0317  00401 000133       ADB M2
0318  00402 074637       ISZ B,I
0319  00403 074537       LDB B,I
0320  00404 065375       JMP FIX,4
0321*
0322*  TRANSFER FROM AR2 TO AR1
0323*
0324  00405 024213 XFAR1 LDB AR1A
0325  00406 020214       LDA AR2A
0326  00407 170004       XFR
0327  00410 000134       ADB M1          RESTORE (B)
0328  00411 170402       RET
0329*
0330*  CONVERT AND STORE AN OPERAND VALUE
0331*
0332*    AR2= SOURCE
0333*    (A)= DESTINATION TYPE
0334*    (B)= DESTINATION ADDRESS
0335*
0336  00412 125413 RSULT LDB HSTPT,I
0337  00413 045413       ISZ HSTPT
0338  00414 121413       LDA HSTPT,I
0339  00415 045413       ISZ HSTPT
0340  00416 070206 STORE RAR 5           IF BIT 4=1
0341  00417 071352       SAM XFAR1,I       IT IS FULL PRECISION
0342  00420 070602       SAR 13          NO, GET PRECISION BITS
0343  00421 071250       SZA XFAR1,I       FULL?
0344  00422 035727       STB MBIN2       SAVE ADDRESS
0345  00423 010012       CPA .1          NO, SPLIT?
0346  00424 064441       JMP STOR2       YES
0347  00425 060435 STOR1 JSM XFAR1       INTEGER: SAVE IN AR1
0348  00426 160225       JSM FLTRA,I     CONVERT INTEGER
0349  00427 064432       JMP INTOV       OVERFLOW EXIT
0350  00430 135727       STB MBIN2,I     STORE INTEGER
0351  00431 064444       JMP XFAR2       RESTORE AR2
0352  00432 160235 INTOV JSM AERRA,I     EMIT OVERFLOW
0353  00433 177627       DEC -105        INTEGER OVERFLOW
0354  00434 024066       LDB INF         SET MAXIMUM POSITIVE INTEGER
0355  00435 021744       LDA AR1         GET MANTISSA SIGN
0356  00436 071511       SLA STOR1,3     POSITIVE?
0357  00437 074076       TCA             NO
0358  00440 064430       JMP STOR1,3
0359  00441 060435 STOR2 JSM XFAR1       SAVE IN AR1
0360  00442 025727       LDB MBIN2       RECALL ADDRESS
0361  00443 160230       JSM FLTSA,I     CONVERT TO SPLIT AND STORE
0362*
0363  00444 020213 XFAR2 LDA AR1A
0364  00445 024214       LDB AR2A
0365  00446 170004       XFR
0366  00447 170402       RET
0367*  STORE AN OPERAND NAME AND TYPE
0368*
0369*    (A)= ASCII CODE FOR LETTER
0370*    ENTER AT STROP+1 WITH PRE:ADJUSTED OPERAND TYPE
0371*    EXIT* (A)= COMPLETED OPERATOR-OPERAND PAIR
0372*
0373  00450 004151 STROP ADB M32         NUMERICALLY ADJUST OPERAND TYPE
0374  00451 000154       ADA M64         GET 5 BIT CODE FOR OPERAND NAME
0375  00452 070544       SAL 5           COMBINE THE
0376  00453 074217       IOR B           TWO OPERAND PARTS AND THE
0377  00454 141706       IOR SHPTR,I       OPERATOR-OPERAND PAIR
0378*
0379*  UPDATE SYNTAX BUFFER POINTER AND CHECK FOR OVERFLOW
```

```
          #01  BASE PAGE ROUTINES


#380*
#381  00455 131706        STA SBPTR,I   STORE (A) BEFORE UPDATE
#382  00456 045706 SBPUD  ISZ SBPTR     UPDATE POINTER
#383  00457 131706        STA SBPTR,I   SAVE (A)
#384  00460 170402        RET
#385*
#386*  FIND AND STORE ONE-CHARACTER OPERATORS
#387*    (A)=SOURCE CHARACTER
#388*    (B)=NUMBER OF ENTRIES TO BE SEARCHED
#389*    P+1=STARTING TABLE ADDRESS-1
#390*
#391  00461 024017 SYM2 LDB .2         ALTERNATE ENTRY TO SEARCH TWO SYMBOLS
#392  00462 064464       JMP SYMCK
#393  00463 024012 SYM1 LDB .1         ALTERNATE ENTRY TO SEARCH ONE SYMBOL
#394*
#395  00464 035466 SYMCK STB LOCL1     SAVE # TO BE SEARCHED
#396  00465 031467       STA LOCL2     SAVE SOURCE CHARACTER
#397  00466 060400       JSM GPARM     GET STARTING ADDRESS
#398  00467 074070 SYM1  SIR **1       INCREMENT TABLE ADDRESS
#399  00470 074517       LDA B*I       GET TABLE ENTRY
#400  00471 070342       SAR B         POSITION IT
#401  00472 011467       CPA LOCL2     MATCH SOURCE CHAR?
#402  00473 064500       JMP SYM2      YES
#403  00474 055466       DSZ LOCL1     NO. CONTINUE SEARCH?
#404  00475 064467       JMP SYM1      YES
#405  00476 021467       LDA LOCL2     NO. RESTORE CHARACTER
#406  00477 170402       RET           RETURN TO P+2- NOT FOUND
#407  00500 020123 SYM2  LDA FOPRS     COMPUTE
#408  00501 070076       TCA
#409  00502 074017       ADA B           OP-CODE
#410  00503 064543       JMP STSRH+4   SHIFT-STORE-RETURN TO P+3
#411*
#412*  CHECK FOR A COMMA
#413*
#414  00504 060463 COMCK JSM SYM1      CHECK FOR
#415  00505 012001       DEF 12001B    COMMA-1
#416  00506 170402       RET           NOT FOUND - RETURN TO P+1
#417  00507 064357       JMP RETN2     FOUND- RETURN TO P+2
#418*
#419  00510 060504 COMCE JSM COMCK     CHECK FOR COMMA
#420  00511 160206       JSM ERRA,I    ERROR IF NOT FOUND
#421  00512 170402       RET
#422*
#423*  CHECK FOR LEFT PARENTHESIS
#424*
#425  00513 060461 LPCK  JSM SYM2      LEFT PAREN
#426  00514 012023       DEF 12023B    LBRAC-1
#427  00515 170402       RET           NO. RETURN TO P+1
#428  00516 020126       LDA LPOP      YES. RECODE A LEFT
#429  00517 064544       JMP STSRH+5     PAREN AND RETURN TO P+2
#430*
#431  00520 061055       JSM GNEXT     GET NEXT CHAR
#432  00521 060513 LPCKE JSM LPCK      DEMAND A
#433  00522 160206 ESCE1 JSM ERRA,I      LEFT PAREN OR BRACKET
#434  00523 170402       RET
#435*
#436*  CHECK FOR RIGHT PARENTHESIS
#437*
#438  00524 060461 RPCK  JSM SYM2      RIGHT PAREN
#439  00525 012003       DEF 12003B    PRAN-1
#440  00526 170402       RET           NO. RETURN TO P+1
#441  00527 020004       LDA RPOP      YES. RECODE A RIGHT
#442  00530 060455       JSM SBPUD-1     PAREN AND INC SBPTR
#443  00531 061055       JSM GNEXT     GET NEXT CHAR
#444  00532 064357       JMP RETN2     RETURN TO P+2
#445*
#446  00533 061055       JSM GNEXT     GET NEXT CHAR
#447  00534 060524 RPCKE JSM RPCK      DEMAND A RIGHT
#448  00535 160206 ESCE2 JSM ERRA,I      LEFT PAREN OR BRACKET
#449  00536 170402       RET
#450*
#451*  SEARCH TABLE AND STORE OP CODE IF FOUND
#452*
#453  00537 160175 STSRH JSM TSRHA,I   SEARCH TABLE
#454  00540 170402       RET           NOT FOUND
#455  00541 031466       STA LOCL1     SAVE OP CODE
#456  00542 050040       AND B37       RET 5 BIT CODE
#457  00543 070304       SAL 10        SHIFT AND
#458  00544 131706       STA SBPTR,I     STORE OP CODE
#459  00545 064357       JMP RETN2     RETURN TO P+2
#460*
#461*  SEARCH TABLE AND STORE EXPANDED OP CODE
#462*
#463  00546 160175 STEXP JSM TSRHA,I   SEARCH TABLE
#464  00547 170402       RET
#465  00550 070544       SAL 5         ASSEMBLE AND
#466  00551 040065       IOR OPMSK       STORE
#467  00552 064544       JMP STSRH+5
#468*
```

```
          #01  BASE PAGE ROUTINES


0469*  BUMP H-STACK POINTER AND CHECK FOR OVERFLOW
0470*
0471  00553 055413 BMSTP ISZ HSTPT      BINC POINTER
0472  00554 025413       LDB HSTPT
0473  00555 074076       TCB
0474  00556 005412       ADB LSTPT
0475  00557 004017       ADB .2
0476  00560 075012       SBM STSRM+1    OVERLAP?  RETURN IF NOT
0477  00561 160206 MERB  JSM ERRA,I
0478*
0479  00561        FSCE4 EQU MERB
0480*
0481*  STACK (B) ON L-STACK AND CHECK OVERFLOW
0482*
0483  00562 025421       LDB PBADD
0484  00563 045412 SLWST ISZ LSTPT      INC POINTER
0485  00564 135412       STB LSTPT,I    STORE VALUE
0486  00565 064554       JMP BMSTP+1    CHECK FOR OVERFLOW
0487*
0488*   FETCH TOP OF L-STACK
0489*
0490  00566 031421       STA PBADD
0491  00567 121412 UNSTK LDA LSTPT,I
0492  00570 055412       DSZ LSTPT
0493  00571 170402       RET
0494*
0495*  PREPARE FOR BRANCH THROUGH TABLE
0496*
0497  00572 025421       LDB PBADD
0498  00573 035472       STB TEMPS
0499  00574 125472 NEXTL LDB TEMPS,I    READ AND SAVE
0500  00575 074073       SBP *+1,C      CLEAR SECURITY BIT
0501  00576 035427       STB .LNUM        CURRENT LINE NUMBER
0502  00577 025472       LDB TEMPS      FIND ADDRESS
0503  00600 061007       JSM NEXTA        OF NEXT STATEMENT
0504  00601 035421       STB PBADD      SAVE IT
0505  00602 045472       ISZ TEMPS
0506  00603 125472       LDB TEMPS,I    READ OP CODE
0507  00604 074073       SBP *+1,C
0508  00605 045472       ISZ TEMPS
0509  00606 074442 NEXTI SBR 10
0510  00607 074076       TCB            GET NEGATIVE OF OP CODE
0511  00610 035466       STB LOCL1      SAVE -(OP CODE)
0512  00611 121472       LDA TEMPS,I
0513  00612 050065       AND OPMSK      GET PAGE
0514  00613 024130 FPAGE LDB FROMA      GET FIRST TABLE ADDRESS
0515  00614 074457       CPA B,I        IS THIS THE PAGE?
0516  00615 170402       RET            YES, (B) = LAST WORD OF PAGE
0517  00616 014131       CPB LROMA      ALL ROM'S SEARCHED?
0518  00617 064622       JMP FPAGI      YES, CHECK BINARY AREA
0519  00620 004116       ADB WTOP       NO, CHECK NEXT ROM
0520  00621 064614       JMP FPAGE+1
0521  00622 025403 FPAGI LDB MAW
0522  00623 111403       CPA MAW,I      IS CODE IN R/W?
0523  00624 170402       RET            YES, (B) = MAW
0524  00625 160206 SYSER JSM ERRA,I     NO, THE PROPER BLOCK IS NOT IN
0525*
0526*
0527*
0528*  SAVE BADDR AND SBPTR
0529*
0530  00626 025400 SAVBP LDB BADDR
0531  00627 035715       STB SBADR
0532  00630 025706       LDB SBPTR
0533  00631 035716       STB SSBPT
0534  00632 170402       RET
0535*
0536*  RESTORE BADDR AND SBPTR
0537*
0538  00633 021715 RESBI LDA SBADR
0539  00634 031400       STA BADDR
0540  00635 021716       LDA SSBPT
0541  00636 031706       STA SBPTR
0542  00637 170402       RET
0543*
0544  00640 060633 RESBP JSM RESBI
0545  00641 121706       LDA SBPTR,I    ERASE
0546  00642 050065       AND OPMSK        OPERAND
0547  00643 131706       STA SBPTR,I
0548  00644 170402       RET
0549*
0550*  SCRATCH SYMBOL TABLE AND INITIALIZE STACKS
0551*
0552  00645 025404 MINIT LDB LWAM
0553  00646 035407       STB SYMTP
0554  00647 035406       STB VALUF
0555  00650 064653       JMP INITS+2
0556*
0557*  INITIALIZE STACKS FOR SYNTAX
0558*
```

```
        #01  BASE PAGE ROUTINES


#559  00651 020662 INITS LDA HFLGA
#560  00652 170000       CLR            CLEAR HALT AND TRACE FLAGS
#561  00653 025426       LDB PBPTO      SCRATCH L-STACK
#562  00654 004132       ADB M1
#563  00655 035412       STB LSTPT
#564  00656 025407 INIT1 LDB SYMTP      SCRATCH H-STACK
#565  00657 074070       STB *+1
#566  00660 035413       STB HSTPT
#567  00661 164242       JMP UCOMA,I    RESET DATA POINTERS
#568  00662 001436 HFLGA DEF HFLG1
#569*
#570*  SUBROUTINE TO BUILD A PROGRAM INTEGER LESS THAN 10000
#571*
#572  00663 055400       DSZ HADDR      BACK UP POINTER
#573  00664 024161 LNUM  LDB MAXSN      CALL PGINT WITH -10000
#574  00665 145706       ISZ SBPTR,I    FLAG FOR RESEQUENCABLE INTEGER
#575*
#576*  SUBROUTINE TO BUILD A PROGRAM INTEGER AND STORE IT
#577*    IN THE S-BUFFER AND FLAG THE PREVIOUS WORD
#578*    (B)=-(MAXIMUM ALLOWABLE INTEGER)
#579*
#580  00666 121706 PGINT LDA SBPTR,I    SET
#581  00667 072053       SAP *+1,S        "INTEGER
#582  00670 000020       ADA .3             FOLLOWS"
#583  00671 060455       JSM SBPUD-1          FLAG
#584  00672 035477       STB TEMP4      SAVE UPPER LIMIT
#585  00673 061055       JSM GNEXT
#586  00674 064677       JMP INTCK+1
#587*
#588*  SUBROUTINE TO BUILD AN INTEGER AND STORE IT IN SBUFFER
#589*    (A)=FIRST CHARACTER
#590*    (B)=-(MAXIMUM ALLOWABLE INTEGER)
#591*
#592  00675 024161       LDB MAXSN      BUILD A LINE #
#593*
#594  00676 035477 INTCK STB TEMP4      SAVE UPPER LIMIT
#595  00677 074742       SBR 16         INITIALIZE
#596  00700 035722       STB INTGR        PARTIAL RESULT
#597  00701 060750       JSM DIGCK      1ST CHAR A DIGIT?
#598  00702 064725       JMP SYE25-1    NO
#599  00703 060750 INTC1 JSM DIGCK      DIGIT?
#600  00704 064716       JMP INTC2      NO
#601  00705 021722       LDA INTGR      RECALL PARTIAL RESULT
#602  00706 035722       STB INTGR      SAVE LATEST DIGIT
#603  00707 024061       LDB .10        MULTIPLY PARTIAL
#604  00710 061201       JSM IMPY       RESULT BY 10
#605  00711 001722       ADA INTGR      ADD LATEST DIGIT
#606  00712 070552       SAM SYE25-1    SKIP IF OVERFLOW
#607  00713 031722       STA INTGR      STORE PARTIAL RESULT
#608  00714 061055       JSM GNEXT      GET NEXT CHARACTER
#609  00715 064703       JMP INTC1
#610  00716 025722 INTC2 LDB INTGR      SKIP THROUGH LINK
#611  00717 135706       STB SBPTR,I    STORE INTEGER
#612  00720 074252       SZB SYE25-1    IF ZERO
#613  00721 005477       ADB TEMP4      SUBTRACT LIMIT
#614  00722 076153       SBP SYE25-1,S    SKIP IF TOO LARGE
#615  00723 025722       LDB INTGR      RETURN WITH
#616  00724 064456       JMP SBPUD        INTEGER IN (B)1 NEXT CHAR IN (A)
#617  00725 045724       ISZ GFLAG      RETURN ON ERROR FLAG SET?
#618  00726 160206 SYE25 JSM ERRA,I     NO
#619  00727 064357       JMP RETN2      YES, RETURN TO P+2
#620*
#621*  GET ADDRESSES FOR SYSTEM COMMANDS
#622*
#623*    RETURN: TEMP4= ADDRESS OF FIRST LINE
#624*            (B)= ADDRESS OF LAST LINE+1
#625*
#626  00730 074070 GETAD STB *+1        ADD 1 TO SECOND TO
#627  00731 035477       STB TEMP4        MAKE ADDRESS INCLUSIVE
#628  00732 074076       ICB            IS FIRST
#629  00733 070037       ADB A            GREATER
#630  00734 075513       SBP SYE25      THAN SECOND
#631  00735 061075       JSM FNDPS      FIND ADDRESS OF FIRST
#632  00736 064740       JMP *+2
#633  00737 064740       JMP *+1
#634*
#635*  CALL FNDPS AND RETURN TO P+1
#636*
#637  00740 021477       LDA TEMP4
#638  00741 035477       STB TEMP4
#639  00742 061075 FND    JSM FNDPS
#640  00743 170402       RET
#641  00744 170402       RET
#642  00745 170402       RET
#643*
#644*  CHECK IF CHAR IN (A) IS A DIGIT
#645*    RETURN TO P+1 IF NOT FOUND
#646*    RETURN TO P+2 IF FOUND: ASCII IN (A): DIGIT IN (B)
#647*
```

```
          #01  BASE PAGE ROUTINES


#648  #0746 #30071 DLIMT OCT 30071      0-9
#649  #0747 #40532 LLIMT OCT 40532      A-Z
#650  #0750 #24746 DIGCK LDB DLIMT      LODA LIMITS FOR DIGITS
#651  #0751 #64753       JMP LIMIT     CHECK LIMITS AND RETRUN
#652*
#653*  CHECK IF CHAR IN (A) IS A LETTER
#654*    RETURN TO P+1 IF NOT FOUND
#655*    RETURN TOP+2 IF FOUND
#656*
#657  #0752 #24747 LETCK LDB LLIMT      LOAD LIMITS FOR LETTER
#658*                    ENTER INTO LIMIT DIRECTLY
#659*
#660  #0753 #35466 LIMIT STB LOCL1      SAVE LIMITS
#661  #0754 #74346       RBR B
#662  #0755 #74342       SBR B          GET UPPER LIMITS
#663  #0756 #74056       CMB
#664  #0757 #70037       ADB A          CODE EXCEEDS
#665  #0760 #75253       SBP END+3      UPPER LIMIT?
#666  #0761 #25466       LDB LOCL1      NO
#667  #0762 #74342       SBR B          GET LOWER LIMIT
#668  #0763 #74076       TCB
#669  #0764 #70037       ADB A          CODE LESS THAN
#670  #0765 #75012       SBM END+3      LOWER LIMIT
#671  #0766 #70137       LDB A          NO. CODE IN LIMITS
#672  #0767 #64153       ADB M48       YES. RETURN TO P+2
#673  #0770 #64357       JMP RETN2
#674*
#675*    GET OPERAND
#676*
#677  #0771 121472 GTOPP LDA TEMP5,I
#678  #0772 #50070       AND OPDMK
#679  #0773 170402       RET
#680***
#681*
#682  #0774 #00000       BSS 10
#683*
#684***
#685*
#686*
#687*
#688*  COMPUTE ADDRESS OF NEXT LINE
#689*
#690  #1006 #35466       STB LOCL1
#691  #1007 #74117 NEXTA LDA B          SAVE ADDRESS OF CURRENT LINE
#692  #1010 #70070       STA **1       GET
#693  #1011 #70517       LDA A+I          STATEMENT
#694  #1012 #50184       AND B177           LENGTH
#695  #1013 #70037       ADB A          COMPUTE ADDRESS OF NEXT LINE
#696  #1014 170402       RET
#697*
#698*
#699*    THIS SUBROUTINE STORES A CHARACTER
#700*    IN A BUFFER.  THE BUFFER POINTER
#701*    IS USED TO INDICATE WHERE IN THE
#702*    BUFFER THE CHARACTER IS TO BE STORED
#703*    THE BUFFER IS UNDISTURBED EXCEPT
#704*    WHERE THE POINTER POINTS.
#705*
#706*
#707*    ENTRY CONDITIONS:
#708*        (B)=BUFFER POINTER POINTER
#709*        (A)=CHARACTER TO BE STORED
#710*            RIGHT JUSTIFIED WITH LEADING 0+S
#711*
#712*
#713*
#714*
#715  #1015 #20041 ORLNK LDA .32
#716  #1016 #24110 ONEXT LDB BADRP       OPTIONAL ENTRY POINT
#717  #1017 #31467 OUTCR STA LOCL2       SAVE CHARACTER TO BE STORED
#718  #1020 #74517       LDA B+I        FETCH BUFFER POINTER
#719  #1021 #11623       CPA PEND       AT I/O END?
#720  #1022 170402       RET            YES
#721  #1023 #74637       ISZ B+I        BUMP BUFFER POINTER
#722  #1024 #70137       LDB A
#723  #1025 #74006       RBR 1
#724  #1026 #70002       SAR 1
#725  #1027 #70517       LDA A+I        FETCH BUFFER WORD
#726  #1030 #74112       SBM **2
#727  #1031 #70346       RAR B          SET A TO STORE IN LEFT HALF
#728  #1032 #50157       AND M256
#729  #1033 #41467       IOR LOCL2
#730  #1034 #74132       SBM **2+C
#731  #1035 #70346       RAR B          ROTATE IF FLAG WAS SET
#732  #1036 #74557       STA B+I        STORE NEW BUFFER WORD
#733  #1037 170402       RET
#734*
#735*
#736*    THIS SUBROUTINE FETCHES
#737*    A CHARACTER FROM A BUFFER
```

```
        #01  BASE PAGE ROUTINES


#738*
#739*
#740*    ENTRY CONDITIONS:
#741*        (B)=BUFFER POINTER POINTER
#742*
#743*
#744*    EXIT CONDITIONS:
#745*        (A)=8-BIT CHARACTER
#746*        (B)=BUFFER POINTER POINTER
#747*
#748*
#749*
#750  01040 024110        LDB BADRP       GET NEXT CHAR FROM I/O BUFFER
#751  01041 035466 GETCH  STB LOCLI       SAVE POINTER POINTER
#752  01042 074517        LDA B,I         FETCH BUFFER POINTER
#753  01043 070002        SAR 1
#754  01044 070517        LDA A,I         FETCH TWO CHARACTERS
#755  01045 074537        LDB B,I
#756  01046 074111        SLB GET1        TOP HALF WORD?
#757  01047 070346        PAR B           NO, MOVE RIGHT HLAF TO LEFT
#758  01050 070342 GET1   SAR B           TAKE TOP HALF
#759  01051 145466        ISZ LOCLI,I     BUMP POINTER
#760  01052 170402        RET             YES
#761*
#762  01053 024111 GTMP   LDB TMPA
#763  01054 065041        JMP GETCH
#764*
#765*
#766  01055 024110 GNEXT  LDB BADRP       GET CHAR, AND CONVERT
#767  01056 061362        JSM GTCON
#768  01057 011707        CPA BLANK       BLANK?
#769  01060 065055        JMP *-3         YES, IGNORE IT
#770  01061 170402        RET             YES, RETURN TO P+1
#771*
#772*  GET CHARACTER ROUTINES
#773*
#774  01062 020233 GFRST  LDA SBUFA
#775  01063 031706        STA SBPTR       INITIALIZE SYNTAX BUFFER POINTER
#776  01064 020211        LDA .BUFA       SET BUFFER POINTER TO START
#777  01065 031400        STA BADDR       OF I/O BUFFER
#778  01066 061055 GETSK  JSM GNEXT       GET A CHARACTER
#779  01067 010031        CPA EOL         END OF LINE?
#780  01070 170402        RET             YES, RETURN TO P+1
#781  01071 064357        JMP RETN2       NO, RETURN TO P+2
#782*
#783*  FIND SEQUENTIAL POSITION OF LINE # GIVEN IN (A)
#784*
#785*    RETURN P+1:BEYOND END OF PROGRAM
#786*           P+2:BETWEEN LINES
#787*           P+3:FOUND
#788*           TEMP3=(B)=ADDRESS OF LINE OR NEXT LINE IN SEQUENCE
#789*           LSTLN=LINE NUMBER OF PREVIOUS LINE
#790*
#791  01072 120233        LDA SBUFA,I     GET SYNTAX SEQUENCE #
#792  01073 031431        STA CLINE       SAVE FOR LIST ROUTINE
#793  01074 031427        STA .LNUM
#794  01075 031476 FNDPS  STA TEMP3       SAVE SEQUENCE #
#795  01076 025424        LDB PBUFF
#796  01077 015423 FNDP1  CPB PBPTR       END OF PROGRAM?
#797  01100 065117        JMP FNDP4       YES, RETURN TO P+1
#798  01101 074517        LDA B,I         GET CURRENT #
#799  01102 070073        SAP *+1,C       CLEAR SECURE BIT
#800  01103 070510        SZA FNDP2       EXIT P+3 IF TYPING AID KEY
#801  01104 070076        TCA             SUBTRACT CURRENT # FROM
#802  01105 001476        ADA TEMP3           DESIRED #
#803  01106 070412        SAM FNDP3       CURRENT>DESIRED? IF YES, P+2
#804  01107 070330        SIA FNDP2       EQUAL?  IF YES,P+3 (A)=1
#805  01110 074517        LDA B,I         SAVE
#806  01111 070073        SAP *+1,C
#807  01112 031720        STA LASTN         CURRENT LINE
#808  01113 061007        JSM NEXTA       COMPUTE ADDRESS OF NEXT LINE
#809  01114 065077        JMP FNDP1
#810  01115 060365 FNDP2  JSM RET2        P+3
#811  01116 060365 FNDP3  JSM RET2        P+2
#812  01117 035476 FNDP4  STB TEMP3       P+1, SAVE STATEMENT ADDRESS
#813  01120 170402        RET
#814*
#815*  MOVE WORDS TO HIGHER MEMORY
#816*
#817*    TEMP2=LAST WORD+1 OF SOURCE
#818*    TEMP3=FIRST WORD OF SOURCE
#819*    TEMP4=LAST WORD+1 OF DESTINATION
#820*
#821  01121 025475 MVTOH  LDB TEMP2       FETCH SOURCE ADDRESS
#822  01122 015476        CPB TEMP3       ALL RELOCATION DONE?
#823  01123 170402        RET
#824  01124 055477        ISZ TEMP4       BACK UP DESTINATION POINTER
#825  01125 004132        ADB M1          BACK UP SOURCE POINTER
#826  01126 074517        LDA B,I         MOVE
#827  01127 131477        STA TEMP4,I       WORD
```

#01 BASE PAGE ROUTINES

```
#828  #1130 #65122        JMP MVTOH+1
#829*
#830*  FETCH AN INTEGER ALLOWING ZERO
#831*
#832  #1131 #20132 PGIN0 LDA M1         FLAG TO RETURN
#833  #1132 #31724        STA GFLAG       ON OVERFLOW OR 0
#834  #1133 #60666        JSM PGINT      FETCH INTEGER
#835  #1134 170402        RET            WITHIN LIMITS
#836  #1135 #45706        ISZ SAPTR      OVERFLOW OR ZERO; UPDATE POINTER
#837  #1136 #75710        SZB *-2        SKIP IF ZERO
#838  #1137 #64726        JMP SYE25      MUST BE OVERFLOW
#839*
#840*  FETCH A LETTER
#841*    EXIT:P+1 IF  NEXT CHR IS NOT A LETTER;CHAR IN (A)
#842*         P+2 IF LETTER; LETTER IN TEMP1; NEXT CHAR IN (A) AND TEMP2
#843*
#844  #1140 #61055 LTR   JSM GNEXT
#845  #1141 #60752        JSM LETCK      LETTER?
#846  #1142 170402        RET            NO. RETURN TO P+1
#847  #1143 #31474        STA TEMP1      YES. SAVE IT
#848  #1144 #61055        JSM GNEXT
#849  #1145 #31475        STA TEMP2      SAVE SECOND CHARACTER
#850  #1146 #64357        JMP RETN2      RETURN TO P+2
#851*
#852*  END AND RESTORE STATEMENT SYNTAX
#853*
#854  #1147 #61055 ENDS  JSM GNEXT      CHECK FOR ADDITIONAL CHARACTERS
#855  #1150 #45706        ISZ SAPTR
#856  #1151 #61153 EOST  JSM EOLCK
#857  #1152 160206 NOEOF JSM ERRA,I
#858*
#859*  SUBROUTINE TO CHECK FOR EOL
#860*
#861  #1153 #10031 EOLCK CPA EOL         EOL?
#862  #1154 164203        JMP ACTSA,I    YES. JUMP DIRECTLY TO ACTST
#863  #1155 170402        RET
#864*
#865*  INPUT A CONSTANT
#866*
#867  #1156 #61166 CONST JSM GSIGN       GET SIGN
#868  #1157 #35721        STB SIGN       STORE IT
#869  #1160 160243        JSM NUMCA,I    FETCH UNSIGNED NUMBER
#870  #1161 #65163        JMP CONS3      NONE FOUND
#871  #1162 #64357        JMP RETN2      FOUND. RETURN TO P+2
#872  #1163 #25721 CONS3 LDB SIGN
#873  #1164 #74610        SZB GSIGR      SKIP IF NO SIGN GIVEN
#874  #1165 160206 SYE12 JSM ERRA,I     SIGN GIVEN WITHOUT NUMBER
#875*
#876*  GET THE SIGN OF A NUMBER
#877*
#878  #1166 #61055 GSIGN JSM GNEXT
#879  #1167 #74742        SBR 16
#880  #1170 #10047        CPA .43        "+"?
#881  #1171 #74230        SIB GSIG1+1    YES
#882  #1172 #10050        CPA .45        "-"?
#883  #1173 #24132        LDB M1         YES
#884  #1174 #74210 GSIG1 SZB GSIGR      SKIP IF NO SIGN GIVEN
#885  #1175 #35470        STB L2T1       SAVE SIGN
#886  #1176 #61055        JSM GNEXT      GET CHARACTER FOLLOWING SIGN
#887  #1177 #25470        LDB L2T1       RECALL SIGN
#888  #1200 170402 GSIGR RET
#889*
#890*  MULTIPLY 2 INTEGERS IN (A) AND (B) AND RETURN
#891*    RESULT IN (A); IF PRODUCT > 32K. (A)<0
#892*
#893  #1201 #31466 IMPY  STA MCAND      SAVE MULTIPLICAND
#894  #1202 #70742        SAR 16         INITIALIZE PRODUCT
#895  #1203 #74151 IMPY1 SLB *+3        SKIP IF MULTIPLIER BIT IS ZERO
#896  #1204 #01466        ADA MCAND      ADD MULTIPLICAND TO PRODUCT
#897  #1205 #71552        SAM IMPYR      SKIP IF RESULT >32K
#898  #1206 #74002        SBR 1          SHIFT MULTIPLIER RIGHT
#899  #1207 #75450        SZB IMPYR      RETURN IF FINISHED
#900  #1210 #31467        STA PROD       SAVE PRODUCT
#901  #1211 #21466        LDA MCAND      MULTIPLY
#902  #1212 #01466        ADA MCAND        MULTIPLICAND
#903  #1213 #71252        SAM IMPYR      SKIP IF RESULT >32K
#904  #1214 #31466        STA MCAND          BY 2
#905  #1215 #21467        LDA PROD       RECALL PRODUCT
#906  #1216 #65203        JMP IMPY1
#907  #1200          IMPYR EQU GSIGR
#908*
#909*  EVALUATE FORMULA AND FETCH RESULT TO AR2
#910*
#911  #1217 160201 FETCH JSM FORMA,I    EVALUATE FORMULA
#912  #1220 160200        JSM OPCHA,I    GET ADDRESS OF RESULT
#913  #1221 #45413        ISZ HSTPT
#914  #1222 #45413        ISZ HSTPT      REMOVE FROM STACK
#915  #1223 #74117        LDA B
#916  #1224 #64445        JMP XFAR2+1    TRANSFER TO AR2
#917*
```

```
          #01  BASE PAGE ROUTINES


#918* SUBROUTINE TO LEAVE KEY DEFINITION MODE
#919*
#920  #1225 #21422 LDEF  LDA UKPTR
#921  #1226 #70410       SZA **R
#922  #1227 #21425       LDA PBUF0
#923  #1230 #31424       STA PBUFF       RESTORE PBUFF
#924  #1231 #21426       LDA PBPT0
#925  #1232 #31423       STA PBPTR       RESTORE PBPTR
#926  #1233 #70742       SAR 16
#927  #1234 #31422       STA UKPTR       CLEAR USER KEY FLAG
#928  #1235 #31432       STA UROST     CLEAR DATA REQUEST FLAG
#929  #1236 170402       RET
#930*
#931*
#932**********************************
#933*
#934*  MATH ROUTINES
#935*
#936*
#937* STORE ROUTINES
#938*
#939  #1237 #74117 STAR2 LDA B          ADD THE EXPONENT OF
#940  #1240 #01744       ADA AR1E        AR1 AND THE A REGISTER,
#941  #1241 #50163       AND ZAP           CLEAR OUT BITS 1-7 AND
#942  #1242 #31754       STA AR2E      STORE IN AR2
#943*
#944  #1243 #74040       ARR 2         OFFSET THE TWO ABOVE
#945  #1244 #21744       LDA AR1E        QUANTITIES AND RE-ADD
#946  #1245 #70040       AAR 2           TO CHECK FOR A
#947  #1246 #74017       ADA B           MAXIMUM EXPONENT
#948  #1247 #70137       LDB A         SAVE THE SIGN OF THE EXPONENT
#949  #1250 #70113       SAP **2       IF IT IS POSITIVE
#950  #1251 #70076       TCA             COMPLEMENT IT
#951  #1252 #00164       ADA MASK1     ADD A POSITIVE 100 TO IT
#952  #1253 #70312       SAM OF        SKIP IF AN OVERFLOW HAS OCCURED
#953*
#954  #1254 #25755 ZONK  LDA AR2E+1
#955  #1255 #76150       RZB ZONK1
#956  #1256 #20214 CLAR2 LDA AR2A      CLEAR AR2
#957  #1257 170000       CLR
#958  #1260 170402 ZONK1 RET
#959*
#960*
#961  #1261 #21754 OF    LDA AR2E      OTHERWISE SET UP BITS 0 AND 15 OF B
#962  #1262 #70111       SLA **2         FOR MAX OR MIN ANSWER
#963  #1263 #76051       SLB **1,S       AND ITS SIGN
#964*
#965  #1264 #74213       SHP OVFW
#966*
#967  #1265 160205 UNFW  JSM AERRA,I
#968  #1266 177633       DEC -101
#969  #1267 #65256       JMP CLAR2
#970*
#971  #1270 #61274 OVFW  JSM STMAX
#972  #1271 160205       JSM AERRA,I
#973  #1272 177634       DEC -100
#974  #1273 170402       RET
#975*
#976  #1274 #61256 STMAX JSM CLAR2
#977  #1275 #74352       SHM .OVER
#978*
#979  #1276 #45757       ISZ AR2E+1
#980  #1277 170400       CMY
#981  #1300 #20165       LDA MASK5
#982  #1301 #31754       STA AR2E
#983  #1302 #74111       SLR **2
#984  #1303 #45754       ISZ AR2E
#985  #1304 170402 .OVER RET
#986*
#987  #1305 #70742 ROUND SAR 16        CLEAR THE A REGISTER AND
#988  #1306 174470       MRY             PERFORM A RIGHT SHIFT B TIMES
#989  #1307 #00136       ADA M5        IF THE LAST SHIFTED
#990  #1310 #70175       SEC **3,C     OUT DIGIT IS =>5 INCREMENT THE
#991  #1311 #20214       LDA AR2A
#992  #1312 170540       MBI             MANTISSA
#993  #1313 170402       RET           THEN RETURN
#994*
#995*
#996**********************************
#997*
#998*
#999*
1000  #1314 171400 ODGIT MLS
1001  #1315 #00052       ADA .4R
1002  #1316 #65016       JMP UNEXT
1003*
1004  #1317 #70537 SGNS  LDB A,I       GET EXPONENT WORD
1005  #1320 #70070       STA **1
1006  #1321 #70517       LDA A,I       GET MANTISSA
1007  #1322 #70210       SZA **4       CHECK IF MANTISSA WAS ZERO
```

```
          #01  BASE PAGE ROUTINES


1008  01323 020012        LDA .1
1009  01324 074111        SLB *+2         SKIP IF SIGN IS 0
1010  01325 020132        LDA M1          OTHERWISE SET A = -1
1011  01326 070137        LDB A           TRANSFER A INTO B
1012  01327 170402        RET
1013*
1014  01330 031466 .NPWR  STA LOCL1
1015  01331 021450        LDA PMODE
1016  01332 031702        STA NPRIN
1017  01333 021466        LDA LOCL1
1018  01334 164167        JMP WRITE,I
1019*
1020  01335 035702 NPWRT  STB NPRIN
1021  01336 164167        JMP WRITE,I
1022*
1023  01337 035701 NDWRT  STB NDISP
1024  01340 164167        JMP WRITE,I
1025*
1026  01341 160170        JSM CLERA,I
1027  01342 021450 CRLF   LDA PMODE
1028  01343 072450        RZA CR1
1029  01344 020031 CRLF1  LDA CRTN
1030  01345 061353        JSM TTY         OUTPUT CRTN
1031  01346 070742        SAR 16
1032  01347 160261        JSM IOSTA,I     STOP KEY?
1033  01350 020040        LDA .31         YES, SET CRLF SUPRESSED FLAG
1034  01351 031622        STA TERM
1035  01352 020061        LDA LNFD        OUTPUT LINE FEED
1036*
1037* GENERAL DRIVER
1038*
1039  01353 160261 TTY    JSM IOSTA,I     STOP KEY?
1040  01354 170402 CR1    RET             YES
1041  01355 041447        IOR SLCOD
1042  01356 172741        SIF 1
1043  01357 172141        OTA 1
1044  01360 173601        SIC 1,C
1045  01361 170402        RET
1046*
1047* CONVERT FROM LOWER CASE
1048*
1049  01362 061041 GTCON  JSM GETCH
1050  01363 070137        LDB A
1051  01364 050103        AND B140
1052  01365 010103        CPA B140
1053  01366 004151        ADB M32
1054  01367 074117        LDA B
1055  01370 170402        RET
1056*
1057*       TEMP1 IS POINTING AT ADDRESS OF THE VARIABLE IN SYMBOL TABLE
1058*       ON RETURN (A) = TYPE CODE
1059*                 (B) = ADDRESS
1060*
1061  01371 025474 FIX    LDB TEMP1
1062  01372 074517        LDA B,I
1063  01373 050040        AND B37
1064  01374 004132        ADB M1
1065  01375 074537        LDB B,I
1066  01376 170402        RET
1067*
1068*****************************************
1069*
1070  01377 000000        BSS 1       CHECKSUM WORD FOR ADD.'S 1000 TO 1377
1071*
1072*****************************************
1073*


              9830--BASE PAGE READ-WRITE MEMORY

1075  01400               ORG 1400B
1076  01400 000000 BADDR  BSS 1     CURRENT CHAR POINTER
1077  01401 000000 RSAVE  BSS 1      SAVE (A) IN RETN?
1078  01402 000000 OTPTR  BSS 1     REFRESH POINTER
1079  01403 000000 MAW    BSS 1     LAST WORD OF R/W MEM
1080  01404 000000 LWAM   BSS 1     LAST WORD AVAILABLE FOR USER
1081  01405 000000 FWUP   BSS 1     FIRST WORD OF COMMON OR MAINLINE
1082  01406 000000 VALUE  BSS 1     FIRST WORD -1 OF VALUE TABLE
1083  01407 000000 SYMTP  BSS 1     FIRST WORD -1 OF SYMBOL TABLE
1084  01410 000000 LSTAK  BSS 1     FIRST WORD OF OPERATOR STACK
1085  01411 000000 TSTPT  BSS 1     TEMPORARY STACK POINTER
1086  01412 000000 LSTPT  BSS 1     OPERATOR STACK POINTER
1087  01413 000000 HSTPT  BSS 1     OPERAND STACK POINTER
1088  01414 000000 DSTAT  BSS 1     DATA STATEMENT
1089  01415 000000 NXTDT  BSS 1       POINTERS
1090  01416 000000 MODE   BSS 1     PROGRAM=01 KEYBOARD=1
1091  01417 000000 PARMA  BSS 1     PARAMETER ADDRESS
1092  01420 000000 PARMN  BSS 1     PARAMETER NAME
1093  01421 000000 PRADD  BSS 1     LAST WORD+1 OF CURRENT STMT
```

```
#01  9836--BASE PAGE READ-WRITE MEMORY


1094  01422 000000 UKPTR BSS 1     USER KEY POINTER
1095  01423 000000 PBPTR BSS 1     LAST WORD OF CURRENT PROGRAM
1096  01424 000000 PBUFF BSS 1     FIRST WORD OF CURRENT PROGRAM
1097  01425 000000 PBUFO BSS 1     FIRST WORD OF MAINLINE
1098  01426 000000 PBPTO BSS 1     LAST WORD+1 OF MAINLINE
1099  01427 000000 .LNUM BSS 1     CURRENT LINE NUMBER
1100  01430 000000 LNINC BSS 1     AUTO LINE INCREMENT
1101  01431 000000 CLINE BSS 1     CURRENT LINE FOR LIST
1102  01432 000000 DROST BSS 1     INPUT STMT FLAG
1103  01433 000000 TFLAG BSS 1     PTAPE FLAG
1104  01434 000000 UNITS BSS 1     TRIG UNITS
1105  01435 000000 SEC   BSS 1     SECURE PROGRAM FLAG
1106  01436 000000 HFLG1 BSS 2     STOP ADDRESSES
1107  01440 000000 TFLG1 BSS 2     TRACE LIMIT ADDRESSES
1108  01442 000000 STPFL BSS 1     STOP FLAG
1109  01443 000000 KEYFG BSS 1     KEYBOARD FLAG
1110  01444 000000 SAVEA BSS 1     INTERRUPT
1111  01445 000000 SAVER BSS 1
1112  01446 000000 ADRSS BSS 1       TEMPORARIES
1113  01447 000000 SLCOD BSS 1     SELECT CODE
1114  01450 000000 PMODE BSS 1     PRINT-ALL FLAG
1115  01451 000000 CTYPE BSS 1     KEYBOARD
1116  01452 000000 CPLAC BSS 1       FORMAT FLAGS
1117  01453 000000 PRFLG BSS 1     PRINT/DISP FLAG
1118  01454 000000 RCOUT BSS 1     FORMAT
1119  01455 000000 FORMT BSS 1
1120  01456 000000 FORMS BSS 1
1121  01457 000000 FORME BSS 1       STATEMENT FLAGS
1122  01460 000000 FTMP1 BSS 1     CURRENT OPERATOR CODE
1123  01461 000000 FTMP2 BSS 1     CURRENT OPERATOR PRECEDENCE
1124  01462 000000 REFDL BSS 1      KEYBOARD DRIVER LINK
1125  01463 000000 INTFL BSS 1      INTERRUPT LINK
1126  01464 000000       BSS 2      UNUSED
1127*
1128*  TEMPORARIES
1129*
1130  01466 000000 LOCL1 BSS 1
1131  01467 000000 LOCL2 BSS 1
1132  01470 000000 L2T1  BSS 1
1133  01471 000000 L2T2  BSS 1
1134  01472 000000 TEMPS BSS 1
1135  01473 000000 TEMP  BSS 1
1136  01473        MROX1 EQU TEMP
1137  01474 000000 TEMP1 BSS 1
1138  01475 000000 TEMP2 BSS 1
1139  01476 000000 TEMP3 BSS 1
1140  01477 000000 TEMP4 BSS 1
1141*
1142*  BUFFERS
1143*
1144  01500 000000 IOBUF BSS 41         INPUT AND RECALL BUFFER
1145  01551 000000 RBUFF BSS 41         SINGLE LINE REFRESH BUFFER
1146*
1147*  PRINT ROUTINE FLAGS
1148*
1149  01622 000000 TERM  BSS 1     CRLF FLAG
1150  01623 000000 PEND  BSS 1     BUFFER OVERFLOW FLAG
1151  01624 000000 PPFF  BSS 1     BUFFER POINTER OR TAB FLAG
1152  01625 000000 FIRST BSS 1
1153  01626 000000       BSS 2      UNUSED
1154*
1155*  MATH TEMPORARIES
1156*
1157  01630 000000 TREGF BSS 4
1158  01634 000000 RSLTF BSS 4
1159*
1160*  STATEMENT SYNTAX AND EXECUTION TEMPORARIES
1161*
1162  01640 000000 ST    BSS 32
1163*
1164*  SYSTEM SYNTAX AND EXECUTION TEMPORARIES
1165*
1166  01700 000000       BSS 1          SAVED FOR PROCESSOR
1167  01701 000000 NOISP BSS 1
1168  01702 000000 NPRIN BSS 1
1169  01703 000000 PLACE BSS 1
1170  01704 000000 WIDTH BSS 1
1171  01705 000000 FLOAT BSS 1
1172  01706 000000 SRPTR BSS 1
1173  01707 000000 BLANK BSS 1
1174  01710 000000 MSFLG BSS 1
1175  01711 000000 DFLAG BSS 1
1176  01712 000000 PFLAG BSS 1
1177  01713 000000 UFLAG BSS 1
1178  01714 000000 SFLAG BSS 1
1179  01715 000000 SRADR BSS 1
1180  01716 000000 SSHPT BSS 1
1181  01717 000000 NUM   BSS 1
1182  01717        OPTYP EQU NUM
1183  01720 000000 LASTN BSS 1
```

```
          #@1  9830--BASE PAGE READ-WRITE MEMORY


1184  @1721 @@@@@@ SIGN  BSS 1
1185  @1721        MAXIN EQU SIGN
1186  @1722 @@@@@@ EXP   BSS 1
1187  @1722        INTGR EQU EXP
1188  @1723 @@@@@@ ARYAD BSS 1
1189  @1724 @@@@@@ GFLAG BSS 1
1190  @1725 @@@@@@ NXTST BSS 1
1191  @1726 @@@@@@ MBIN1 BSS 1
1192  @1727 @@@@@@ MBIN2 BSS 1
1193*
1194*  ARITHMETIC REGISTERS AND RETURN STACK
1195*
1196  @1740              ORG 1740@
1197  @1740 @@@@@@ XC    BSS 4
1198  @1744 @@@@@@ AR1   BSS 4
1199  @1750 @@@@@@ YC    BSS 4
1200  @1754 @@@@@@ AR2   BSS 4
1201  @1760 @@@@@@ RSTAK BSS 11      RETURN STACK
1202  @1773 @@@@@@ RES   BSS 4       RESULT REGISTER
1203  @1777 @@@@@@ RSPTR BSS 1     RETURN STACK POINTER


          #@1  EQUATES

1205  @1466        MCAND EQU LOCL1
1206  @1467        PROD  EQU LOCL2
1207  @1744        ARIE  EQU AR1
1208  @1754        AR2E  EQU AR2
1209  @1730        COML  EQU MBIN2+1
1210  @1731        TYPE  EQU MBIN2+2
1211  @1640        TEPPP EQU ST
1212  @1641        SPROD EQU ST+1
1213  @1642        SAVE  EQU ST+2
1214  @1453        ARW   EQU PRFLG
1215  @1740        TMP   EQU XC
1216  @1741        TMP1  EQU XC+1
1217  @1742        TMP2  EQU XC+2
1218  @1743        TMP3  EQU XC+3
1219  @1750        TMP4  EQU YC
1220  @1751        TMP5  EQU YC+1
1221  @1706        TMP6  EQU SBPTR
1222  @1752        TMP6A EQU YC+2
1223  @1460        TMP7  EQU FTMP1
1224  @1474        TMP9  EQU TEMPS+2
1225  @1475        TMP10 EQU TEMPS+3
1226  @1476        TMP11 EQU TEMPS+4
1227  @1477        TMP12 EQU TEMPS+5
1228  @1461        TMP13 EQU FTMP2
1229  @1473        CCNT  EQU TEMPS+1
1230  @1704        BKPTR EQU #IDTH
1231  @1741        R     EQU TMP1
1232  @1742        L     EQU TMP2
1233  @1751        DIGIT EQU TMP5
1234  @1743        INDEX EQU TMP3
1235  @1750        RBASE EQU TMP4
1236*
1237*
1238  @@047        PLUS  EQU .43
1239  @@@5@        MINUS EQU .45
1240  @@@51        POINT EQU .46
1241  @@@52        @60   EQU .48
1242  @@@53        COLON EQU .58
1243  @@@77        DOLAR EQU @44
1244  @@116        R2K   EQU @TOP
1245  @@@@4        XFCA  EQU @10K


          #@1  KEYBOARD ENTRY AND DISPLAY

1248*
1249*    REED SUBROUTINE
1250*
1251*    ENTRY CONDITIONS: NONE
1252*
1253*    EXIT CONDITIONS:
1254*
1255*    AN INPUT RECORD IS IN THE I/O
1256*    BUFFER.  THE LAST CHAR IS AN EOL.
1257*
1258*
1259*    IF A DELETE LINE IS HIT
1260*    RETURN TO P+1, OTHERWISE
1261*    RETURN TO P+2.
1262*
1263*
1264  @2@@@          ORG 2@@@@
1265*
```

#01 KEYBOARD ENTRY AND DISPLAY

```
1266*
1267  #2000 #20236 REED1 LDA SLEND
1268  #2001 #31623       STA PEND
1269  #2002 #20062       LDA FLGAT
1270  #2003 #31416       STA MODE      SET TO PROGRAM SYNTAX
1271  #2004 #21433       LDA TFLAG
1272  #2005 #70110       SZA *+2       PTAPE COMMAND?
1273  #2006 #66617       JMP PREED
1274  #2007 #31473       STA CCNT
1275  #2010 #31704       STA BKPTR
1276*  GET FIRST KEY
1277  #2011 #63046 REED0 JSM CALLK     GET FIRST KEY
1278  #2012 #66011       JMP *-1
1279  #2013 #13127       CPA DELIN
1280  #2014 #66223       JMP DEL1
1281  #2015 #25430       LDB LNINC
1282  #2016 #10074       CPA B22       BACKSPACE?
1283  #2017 #74112       SBM REE       YES. SKIP IF LIST INPUT
1284  #2020 #66032       JMP REED7
1285  #2021 #55400 REE   DSZ BADDR     YES. BACKUP POINTER
1286  #2022 #45623       ISZ PEND
1287  #2023 #20041       LDA .32
1288  #2024 #61016       JSM ONEXT
1289  #2025 #55400       DSZ BADDR     WIPE OUT LAZY-T
1290  #2026 #55623       DSZ PEND      RESTORE END OF BUFFER POINTER
1291  #2027 #70742       SAR 16
1292  #2030 #31430       STA LNINC     GET OUT OF AUTO
1293  #2031 #66045       JMP REED6
1294  #2032 #62533 REED7 JSM CLEAR     CLEAR THE DISPLAY
1295  #2033 #21430       LDA LNINC
1296  #2034 #70153       SAP REED4      LIST WAS LAST?
1297  #2035 #70742       SAR 16
1298  #2036 #31430       STA LNINC     GET OUT OF AUTO
1299  #2037 #70210 REED4 SZA REED6-2   AUTO LINE?
1300  #2040 #25427       LDB .LNUM     YES OUTPUT #
1301  #2041 #60210       JSM OUTIA,I
1302  #2042 #45400       ISZ BADDR     OUTPUT A BLANK
1303  #2043 #21751       LDA TMP5
1304  #2044 #66047       JMP REED6
1305*  GET NEXT KEY
1306  #2045 #63046 REED6 JSM CALLK     GET NEXT KEY
1307  #2046 #66045       JMP *-1
1308  #2047 #10074 REED6 CPA B22       BACKSPACE?
1309  #2050 #66707       JMP BKSP
1310  #2051 #10075       CPA B23       FORWARD?
1311  #2052 #66725       JMP FORWD     YES
1312  #2053 #12436       CPA H152      CLEAR?
1313  #2054 #67665       JMP ECLR      YES
1314  #2055 #10027       CPA .11       EXECUTE KEY?
1315  #2056 #45416       ISZ MODE      YES
1316  #2057 #50103       AND MSK65
1317  #2060 #72510       RZA REED2     BITS 5&6=0?
1318  #2061 #21751       LDA TMP5      YES
1319  #2062 #00141       ADA M10
1320  #2063 #70413       SAP REED2+1   DEF KEY?
1321  #2064 #25706       LDB TMP6
1322  #2065 #74210       SZA *+4
1323  #2066 #21751       LDA TMP5
1324  #2067 #40101       IOR B100      SET SHIFT BIT FOR DEF-KEYS
1325  #2070 #31751       STA TMP5
1326  #2071 #67131       JMP KEYPR
1327  #2072 #10103 REED2 CPA MSK65     BITS 5&6=1?
1328  #2073 #66103       JMP REED3     YES
1329  #2074 #21706       LDA TMP6      NO
1330  #2075 #70610       SZA REED5     SHIFTED?
1331  #2076 #21751       LDA TMP5
1332  #2077 #60752       JSM LETCK     YES. LOWER CASE LETTER?
1333  #2100 #66103       JMP REED3     NO
1334  #2101 #40041       IOR .32       CONVERT CODE
1335  #2102 #66110       JMP REED5-1
1336  #2103 #26412 REED3 LDB VALUP
1337  #2104 #63071       JSM THSRI     CODE IN VALUE TABLE?
1338  #2105 #66124       JMP REED0     NO
1339  #2106 #10022       CPA .5        ENTER EXPONENT?
1340  #2107 #00101       ADA B100      YES. FORM ASCII E
1341  #2110 #31751       STA TMP5
1342*
1343*  ALL CONVERSIONS COMPLETE
1344  #2111 #21751 REED5 LDA TMP5
1345  #2112 #13031       CPA CRTN
1346  #2113 #66172       JMP EOLL      EXECUTE OR STORE
1347  #2114 #25704       LDB BKPTR
1348  #2115 #74210       SZB REED9     CURSOR?
1349  #2116 #27122       LDB BKPTA     YES
1350  #2117 #61017       JSM OUTCR     REPLACE CHARACTER
1351  #2120 #66730       JMP FORWD+3   SCROLL IF NECESSARY
1352  #2121 #61016 REED9 JSM ONEXT     PUT CHAR AT END OF LINE
1353  #2122 #62737       JSM LEFT      SCROLL IF NECESSARY
1354  #2123 #66045       JMP REED6
1355*
```

#01 KEYBOARD ENTRY AND DISPLAY

```
1356  #2124 #26373 REEDD LDB EXPTR      SEARCH FOR MULTI-CHAR KEY
1357  #2125 #63071       JSM TBSR1      FOUND?
1358  #2126 #66213       JMP REEDR      NO
1359  #2127 #31751       STA TMP5
1360  #2130 #62521       JSM CLRST      CLEAR FLAG IF "STOP"
1361  #2131 #62533       JSM CLEAR
1362  #2132 #26443       LDB COMTA
1363  #2133 #21751       LDA TMP5
1364  #2134 162441       JSM MOUTA,I    OUTPUT FROM SYSTEM COMMAND TABLE
1365  #2135 #66045       JMP REED6
1366  #2136 #62533 PRALL JSM CLEAR      CLEAR BUFFER
1367  #2137 #21450       LDA PMODE
1368  #2140 #70130       SIA *+2
1369  #2141 #70742       SAR 16
1370  #2142 #31450       STA PMODE
1371  #2143 #26444       LDB OFFA       GET ADDRESS OF "OFF"
1372  #2144 #72110       RZA *+2        ON?
1373  #2145 #04020       ADA .3         YES, GET ADDRESS OF "ON "
1374  #2146 #20020       LDA .3
1375  #2147 #61335       JSM NPWRT
1376  #2150 #66011       JMP REEDE      GET FIRST KEY AGAIN
1377*
1378*  STEP AND CONT EXECUTION
1379*
1380  #2151 #31442 STEP  STA STPFL      SET POSITIVE NON-ZERO
1381  #2152 #70742       SAR 16
1382  #2153 #25421 CONT  LDB PRADD
1383  #2154 #74273       SBP CONT1,C    HAS AN EDIT OCCURRED?
1384  #2155 #72110       RZA CONTK      YES, LINE # GIVEN?
1385  #2156 #74517       LDA B,I         NO, EXECUTE FROM BEGINNING
1386  #2157 #31725 CONTK STA NXTST      YES, EXECUTE FROM BEGINNING
1387  #2160 164004       JMP XEQA,I
1388  #2161 #70250 CONT1 SZA *+5        LINE # GIVEN?
1389  #2162 #61075       JSM FNDPS      YES, FIND ADDRESS
1390  #2163 164174       JMP RDYDA,I
1391  #2164 #64165       JMP *+1
1392  #2165 #35421       STB PRADD      STORE ADDRESS
1393  #2166 #62533       JSM CLEAR
1394  #2167 #31416       STA MODE       SET TO PROGRAM EXECUTION
1395  #2170 #25421       LDB PRADD
1396  #2171 164255       JMP XEQ6A,I    CONTINUE EXECUTION
1397*
1398*  PROCESS EXEC OR STORE
1399  #2172 #45623 EOLL  ISZ PEND       GUARANTEE ROOM FOR EOL
1400  #2173 #61016       JSM ONEXT
1401  #2174 #24233       LDB BEND
1402  #2175 #35477       STB TEMP4
1403  #2176 #24233       LDB SHUFA
1404  #2177 #35476       STB TEMP3
1405  #2200 #04046       ADA .41
1406  #2201 #61122       JSM MVTOH+1    MOVE INPUT RECORD TO I/O BUFF
1407  #2202 #60373       JSM ICNT
1408  #2203 #00132       ADA M1
1409  #2204 #70250       SZA REEDT      SKIP IF NULL RECORD
1410  #2205 #25450       LDB PMODE
1411  #2206 #76150       RZB REEDT
1412  #2207 #24211       LDB .BUFA
1413  #2210 #61337       JSM NDWRT
1414  #2211 #62502 REEDT JSM CRC
1415  #2212 #64357       JMP RETN2
1416*
1417  #2213 #13124 REEDR CPA B33        DIVIDE?
1418  #2214 #22437       LDA BS7        YES, GET ASCII
1419  #2215 #26361       LDB EDIT2
1420  #2216 #63072       JSM TBSRH      CODE IN EDIT TABLE?
1421  #2217 #66110       JMP REED5-1    NO
1422  #2220 #02440       ADA BEEP
1423  #2221 #70737       JMP A,I
1424*
1425*  DELETE LINE
1426  #2222 170402 DEL2  RET
1427  #2223 #21430 DEL1  LDA LNINC
1428  #2224 #71713       SAP DEL2
1429  #2225 #21431       LDA CLINF
1430  #2226 130233       STA SHUFA,I
1431  #2227 #70742       SAR 16
1432  #2230 #31430       STA LNINC
1433  #2231 164204       JMP DELSA,I
1434*
1435*  RECALL
1436*
1437  #2232 #62533 EREC  JSM CLEAR
1438  #2233 #55473       DSZ CONT       SET RECALL FLAG
1439  #2234 #24211       LDB .BUFA
1440  #2235 #35740 RSLT1 STB IMP        GET STARTING ADDRESS  OF IO BUFF
1441  #2236 #66045       JMP REED6
1442*
1443*  INITIALIZE VARIABLES
1444*
1445  #2237 #20056 INITV LDA .1E4
```

```
          #81  KEYBOARD ENTRY AND DISPLAY


1446  02240 164171        JMP RUNA,I     RUN 10000
1447*  RESULT KEY
1448  02241 020101 RSLT  LDA B100       CODE FOR @
1449  02242 025706        LDB TMP6
1450  02243 074110        SZB *+2        SHIFTED?
1451  02244 066110        JMP KEEDS-1    YES
1452  02245 026445        LDB KESA
1453  02246 020020 RENX1 LDA .3
1454  02247 031473        STA CCNT       COUNT=3
1455  02250 066235        JMP RSLT1
1456*
1457*
1458  02251 066136 PAL    JMP PHALL
1459  02252 066151 STPL   JMP STEP
1460*
1461*  FETCH EXECUTION
1462*
1463  02253 000132 FFET   ADA M1
1464  02254 031431        STA CLINF
1465*
1466*     THIS ROUTINE LISTS ONE LINE
1467*     OF THE PROGRAM. THE FIRST
1468*     LINE TO BE LISTED IS THE FIRST
1469*     LINE OF THE PROGRAM.
1470*
1471*
1472  02255 062353 ONARW JSM SAROW      CHECK FOR NULL PROGRAM
1473  02256 074310        SZB U4         SKIP IF TYPING AID KEY
1474  02257 070070        STA *+1
1475  02260 061075        JSM FNDPS      FIND LINE
1476  02261 066277        JMP U3         PAST END OF PROGRAM
1477  02262 066301        JMP U5-1       BETWEEN LINES
1478  02263 066301        JMP U5-1
1479*
1480*     THIS ROUTINE LISTS ONE LINE
1481*     OF THE PROGRAM. THE FIRST LINE
1482*     TO BE LISTED IS THE LAST LINE
1483*     OF THE PROGRAM.
1484*
1485*
1486  02264 021424 U4     LDA PBUFF      GET FIRST LINE
1487  02265 066302        JMP U5
1488  02266 062353 UPARW JSM SAROW      CHECK FOR NULL PROG
1489  02267 075650        SZB U4         SKIP IF TYPING AID
1490  02270 072110        RZA U1         CLINF =0?
1491  02271 020056        LDA .1F4       YES, FIND LAST LINE OF PROG
1492  02272 061075 U1     JSM FNDPS      FIND LINE
1493  02273 066277        JMP U3         PAST END OF PROG
1494  02274 066275        JMP *+1        BETWEEN LINES
1495  02275 015424        CPB PBUFF      FIRST LINE?
1496  02276 066301        JMP U5-1        YES
1497  02277 021720 U3     LDA LASTN      GET PREVIOUS LINE
1498  02300 060742        JSM FND
1499  02301 074117        LDA B          GET ADDRESS IN (A)
1500  02302 074742 U5     SBR 16
1501  02303 035453        STB ARW        FLAG LIST ROUTINE
1502  02304 035460        STB TMP7
1503  02305 167120        JMP LISTR,I
1504*
1505*  INSERT AND DELETE CHARACTER
1506*
1507  02306 066045        JMP KEED6
1508  02307 025704 INSRT LDB BKPTR
1509  02310 075710        SZB INSRT-1    ANY BACKSPACE YET?
1510  02311 025400        LDB BADDR
1511  02312 021706        LDA TMP6
1512  02313 070710        SZA INS0       SHIFT FLAG SET?
1513*  DELETE
1514  02314 055400        DSZ BADDR      YES, DINC END OF LINE POINTER
1515  02315 025704        LDB BKPTR
1516  02316 035740        STB TMP        SET POINTER TO CURSOR CHARACTER
1517  02317 025740 DELC1 LDB TMP
1518  02320 015400        CPB BADDR      LAST CHAR?
1519  02321 066347        JMP INS2+1     YES, OUTPUT A BLANK
1520  02322 045740        ISZ TMP
1521  02323 061053        JSM GTMP       GET CHARACTER
1522  02324 055740        DSZ TMP
1523  02325 055740        DSZ TMP
1524  02326 024111        LDB TMPA       STORE SHIFTED LEFT ONE
1525  02327 061017        JSM OUTCR
1526  02330 066317        JMP DELC1
1527*  INSERT
1528  02331 014216 INS0   CPB SLEND
1529  02332 066045        JMP KEED6       LINE FULL
1530  02333 004132        ADB M1
1531  02334 035740 INS1   STB TMP
1532  02335 061053        JSM GTMP       GET CHARACTER
1533  02336 025740        LDB TMP
1534  02337 015704        CPB BKPTR      AT CURSOR POSITION?
1535  02340 066346        JMP INS2       YES
```

#01 KEYBOARD ENTRY AND DISPLAY

```
1537  02341 024111        LDB TMPA     NO. MOVE RIGHT 1 PLACE
1538  02342 061017        JSM OUTCR
1539  02343 025740        LDB TMP
1540  02344 004134        ADB M3         BACK UP POINTER TO NEXT
1541  02345 066334        JMP INS1         CHAR LEFT
1542  02346 045400 INS2   ISZ HADDR      UPDATE END OF LINE POINTER
1543  02347 020041        LDA .32
1544  02350 024111        LDB TMPA
1545  02351 061017        JSM OUTCR      INSERT A BLANK
1546  02352 066045        JMP REEDS
1547*
1548*  SUBROUTINE FOR UP AND DOWN ARROW ROUTINES
1549*
1550  02353 021424 SAROW LDA PBUFF
1551  02354 011423        CPA PAPTR
1552  02355 164174        JMP RDYDA,I    NULL PROGRAM ABORT
1553  02356 021431        LDA CLINE
1554  02357 125424        LDB PBUFF,I
1555  02360 170402        RET
1557*
1558*    THIS SECTION CONTAINS TABLES,
1559*    POINTERS USED IN THE ABOVE ROUTINE
1560*
1561*
1562*
1563*
1564  02361 004744 EDIT2 DEF *+*+2
1565  02362 012643        DEF 12600B-EEB+UPARW
1566  02363 012232        DEF 12200B-EEB+DNARW
1567  02364 015214        DEF 15200B-EEB+INITV
1568  02365 065616        DEF 65600B-EEB+RSLT
1569  02366 006627        DEF 6600B-EEB+STPL
1570  02367 017664        DEF 17600B-EEB+INSRT
1571  02370 071600        DEF 71600B-EEB+DELI
1572  02371 070226        DEF 70200B-EEB+PAL
1573  02372 010307        DEF 10300B-EEB+EREC
1574  02223        EEB  EQU DELI
1575*
1576*
1577*    EXPANDING DEY TABLE
1578*
1579*
1580  02373 004770 EXPTB DEF *+*+2
1581  02374 007601  OCT 7601       RUN
1582  02375 014600  OCT 14600       STORE
1583  02376 014220  OCT 14220      LOAD
1584  02377 017207  OCT 17207      NORMAL
1585  02400 006212  OCT 6212       CONTINUE
1586  02401 007226  OCT 7226       TRACE
1587  02402 077637  OCT 77637      STOP
1588  02403 017216  OCT 17216      NORMAL
1589  02404 010604  OCT 10604      FETCH
1590  02405 072214  OCT 72214      FIXED
1591  02406 072615  OCT 72615      FLOAT
1592  02407 005203  OCT 5203       LIST
1593  02410 073202  OCT 73202      SCRATCH
1594  02411 073705  OCT 73705      AUTO
1595*
1596*
1597*    KEY VALUE TABLE
1598*
1599  02412 005026 VALUP DEF *+*+2
1600  02413 005615  OCT 5615       EOL
1601  02414 071215  OCT 71215      STORE
1602  02415 020240  OCT 20240      BLANK
1603  02416 030641  OCT 30641      !
1604  02417 032244  OCT 32244      %
1605  02420 032645  OCT 32645      &
1606  02421 067605  OCT 67605      F
1607  02422 031643  OCT 31643      #
1608  02423 031242  OCT 31242      "
1609  02424 033246  OCT 33246      &
1610  02425 033647  OCT 33647      '
1611  02426 034250  OCT 34250      (
1612  02427 034651  OCT 34651      )
1613  02430 027677  OCT 27677     ?
1614  02431 026274        OCT 26274      <
1615  02432 027276        OCT 27276      >
1616  02433 035252  OCT 35252      *
1617  02434 035653  OCT 35653      +
1618  02435 030355  OCT 30355      -
1619*
1620*
1621*
1622  00103        MSK65 EQU 9140
1623  02436 000152 B152  OCT 152
1624  02437 000057 B57   OCT 57
1625  02440 002223 BEEB  DEF EEB
1626  02441 006306 MOUTA DEF MCOUT
1627  02442 003402 LPTR  DEF RBUFF+RBUFF+4R
```

#01 KEYBOARD ENTRY AND DISPLAY

```
1628  02443 007062 COMTA DEF LHL1+LBL1+20
1629  02444 005117 OFFA  DEF *+*+7
1630  02445 005114 RFSA  DEF *+*+2
1631  02446 051105       ASC 4,RESOFFON
      02447 051517
      02450 043106
      02451 047516
1632  02452 020040 TOBLN OCT 20040
1633*
1634*
1635  02453 000000       BSS 1            ***** CHECKSUM*****
1637*    WRITE SUBROUTINE
1638*
1639*    ENTRY CONDITIONS:
1640*
1641*         (A)=CHARACTER COUNT
1642*         (B)=POINTER TO FIRST CHAR
1643*         NPRIN#0 FOR PRINT ABORT
1644*         NDISP#0   PRINT ONLY
1645*
1646*
1647  02454 070410 WRIT  SZA RIT3L      SKIP IF NULL COUNT
1648  02455 031741       STA TMP1
1649  02456 035740       STB TMP        INITIALIZE POINTER
1650  02457 061053 RITF1 JSM GTMP       GET CHARACTER
1651  02460 010027       CPA .11        EXECUTE?
1652  02461 020100       LDA B52        YES, CHANGE TO * FOR KEY
1653  02462 025701       LDB NDISP
1654  02463 010031       CPA CRTN       CRTN CHARACTER?
1655  02464 066476 RIT3L JMP RITF3      YES, ABORT
1656  02465 076310       RZB RITE2-1    DISPLAY DISABLED?
1657  02466 070150       SZA RITE4      NO, NULL?
1658  02467 061016       JSM UNEXT      OUTPUT TO DISPLAY BUFFER
1659  02470 021467       LDA LOCL2      RECALL CHARACTER
1660  02471 025702 RITE4 LDB NPRIN
1661  02472 076110       RZB RITF2      PRINT DISABLED?
1662  02473 061353       JSM TTY        NO, PRINT CHAR
1663  02474 055741 RITF2 DSZ TMP1       MORE CHARACTERS?
1664  02475 066457       JMP RITF1      YES
1665  02476 070742 RITE3 SAR 16         RESET FLAGS
1666  02477 031702       STA NPRIN
1667  02500 031701       STA NDISP
1668  02501 170402 RETA  RET
1669*
1670  02502 025450 CRC   LDB PMODF
1671  02503 074350       SZB CRC1       PRINT ALL?
1672  02504 025450 CRCHK LDB PMODF
1673  02505 074150       SZB CRC2       PRINT ALL?
1674  02506 025453       LDB PRFLG
1675  02507 074352       SBM CTERM
1676  02510 025622 CRC2  LDB TERM
1677  02511 075410       SZB RETA
1678  02512 172700 CRC1  SFC 0          WAIT UNTIL "STOP" KEY
1679  02513 066512       JMP *-1           IS RELEASED
1680  02514 062521       JSM CLRST      CLEAR "STOP" FLAG
1681  02515 061344       JSM CRLF1
1682*
1683  02516 074742 CTERM SBR 16
1684  02517 035622       STB TERM
1685  02520 170402       RET
1686*
1687  02521 025442 CLRST LDB STPFL
1688  02522 014040       CPA .31        SET FROM KEYBOARD?
1689  02523 066525       JMP **2        YES, CLEAR IT
1690  02524 170402       RET
1691  02525 074742       SBR 16
1692  02526 035442       STB STPFL
1693  02527 035433       STB TFLAG
1694  02530 170402       RET
1695*
1696*
1697  02531 027117 GTMP2 LDB TMP2A
1698  02532 065041       JMP GETCH
1699*
1700*
1701*    THIS SUBROUTINE CLEARS THE DISPLAY
1702*
1703  02533 070742 CLEAR SAR 16
1704  02534 031704       STA BKPTR
1705  02535 020212       LDA .SLAF
1706  02536 031402       STA OTPTR
1707  02537 031400       STA BADDR
1708  02540 031625       STA FIRST
1709  02541 024233       LDB BEND
1710  02542 022452       LDA TOBLN
1711  02543 074557 CLE1  STA B,I
1712  02544 017123       CPB ENDW
1713  02545 066476       JMP RITE3
1714  02546 077670       RIB CLE1       NO
1715*
```

#01 KEYBOARD ENTRY AND DISPLAY

```
1716*    I/O STATUS CHECK ROUTINE
1717*
1718  #2547 #25442 IOST  LDB STPFL
1719  #2550 #14040       CPB .31        STOP FLAG?
1720  #2551 170402       RET            YES
1721  #2552 #25447       LDB SLCOD      NO
1722  #2553 172741       STF 1          TURN INTERRUPT OFF
1723  #2554 176141       OTB 1
1724  #2555 172741       STF 1          GET STATUS IN I/O REG
1725  #2556 177241       LIB 1,C        TURN INTERRUPT ON
1726  #2557 #74406       RBR 9
1727  #2560 #75353       SHP IOST       DEVICE BUSY?
1728  #2561 #64357       JMP RETN2      NO
1730*
1731*
1732*    THIS SUBROUTINE REFRESHES
1733*    THE SINGLE LINE DISPLAY
1734*
1735  #2562 #21402 REFRS LDA DTPTR
1736  #2563 #31740       STA TMP
1737  #2564 #00033       ADA .16
1738  #2565 #31742       STA TMP2
1739  #2566 #74742       SBR 16
1740  #2567 #35741 RE1   STB TMP1
1741  #2570 #24111       LDB TMPA
1742  #2571 #61362       JSM GTCON
1743  #2572 #70304 RE3   SAL 10
1744  #2573 #31743       STA TMP3
1745  #2574 #27117       LDB TMP2A
1746  #2575 #61362       JSM GTCON
1747  #2576 #50054 RE4   AND .63
1748  #2577 #70604       SAL 4
1749  #2600 #41743       IOR TMP3
1750  #2601 #41741       IOR TMP1       FORM OUTPUT WORD
1751  #2602 173750       CLF 8          TURN DISPLAY OFF
1752  #2603 173741       CLF 1          INTERRUPT ON
1753  #2604 #25443       LDB KEYFG
1754  #2605 172741       STF 1          INTERRUPT OFF
1755  #2606 #74353       SBP RE2        INTERRUPT OCCURRED?
1756  #2607 #25741       LDB TMP1
1757  #2610 #14033       CPB .16
1758  #2611 #66615       JMP RE2
1759  #2612 #74070       SIB *+1
1760  #2613 172150       OTA 8
1761  #2614 #66567       JMP RE1
1762  #2615 173741 RE2   CLF 1
1763  #2616 170402       RET            TURN INTERRUPT ON
1765*
1766*    PTAPE COMMAND EXECUTION
1767*
1768*    THIS ROUTINE READS PROGRAMS FROM
1769*    THE PHOTOREADER. 10 OR MORE NULLS
1770*    CONSTITUTE LEADER OR TRAILER WHICH
1771*    ARE THE PROGRAM DELIMITERS.
1772*
1773*    INPUT CODES:
1774*
1775*      DATA  64 ASCII SET
1776*
1777*      CONTROL
1778*
1779*        176  ALT MODE=DELETE LINE
1780*        33   ESC MODE=DELETE LINE
1781*        137  LEFT ARROW=DELETE CHAR
1782*        177  RUBOUT= IGNORE
1783*        15   CRTN= IGNORE
1784*        12   LNFD= END OF RECORD
1785*    RETURN TO:
1786*
1787*        P+1 IF NULL RECORD
1788*        P+2 OTHERWISE
1789  #2617 #70204 PREFD SAL 12
1790  #2620 #31447       STA SLCOD
1791  #2621 #20211       LDA .BUFA
1792  #2622 #31400       STA BADDR
1793  #2623 #20132       LDA M1
1794  #2624 #31751       STA TMP5       CLEAR DELETE LINE FLAG
1795  #2625 #20141       LDA M10
1796  #2626 #31741       STA TMP1       SET FEED HOLE COUNTER
1797*
1798*
1799*    GENERAL INPUT ROUTINE
1800*
1801  #2627 #62547 P1    JSM IOST       STOP KEY?
1802  #2630 #66704       JMP P7         YES
1803  #2631 #20121       LDA B4000      GET INPUT STATUS BIT
1804  #2632 #61355       JSM CR1+1      CALL FOR INPUT
1805  #2633 #62547       JSM IOST
1806  #2634 #66704 P4    JMP P7         STOP
1807  #2635 #74117       LDA B
```

```
        #01  KEYBOARD ENTRY AND DISPLAY


1808  02636 070306        XAR 7
1809  02637 050104        AND B177
1810  02640 072310        SZA P3          NULL?
1811  02641 045741        ISZ TMP1        10 FEED HOLES?
1812  02642 066627        JMP P1          NO
1813  02643 021742        LDA TMP2
1814  02644 071413        SAP P4          TRAILER?
1815  02645 066625        JMP P1-2        NO, LEADER
1816  02646 031742 P3     STA TMP2        RESET LEADER FLAG
1817  02647 010104        CPA B177        RUBOUT?
1818  02650 066625        JMP P1-2        YES
1819  02651 010031        CPA CRTN        CRTN?
1820  02652 066625        JMP P1-2        YES, IGNORE
1821  02653 013126        CPA B137        LEFT ARROW?
1822  02654 066700        JMP P5          YES, BACKSPACE
1823  02655 010061        CPA LNFD        LINE FEED?
1824  02656 020031        LDA EOL         YES, CHANGE TO END OF LINE CHARACTER
1825  02657 013124        CPA B33         ESCAPE MODE?
1826  02660 066624        JMP P1-3        YES, SET DELETE FLAG
1827  02661 013125        CPA B176        ALT MODE?
1828  02662 066624        JMP P1-3        YES, DELETE
1829  02663 061016        JSM ONEXT
1830  02664 021467        LDA LOCL2
1831  02665 010031        CPA EOL         END OF RECORD?
1832  02666 066673        JMP P2          YES
1833  02667 025400        LDB BADDR
1834  02670 017130        CPB IOEND       FULL?
1835  02671 055400 P6     DSZ BADDR
1836  02672 066625        JMP P1-2
1837  02673 020127 P2     LDA B170K
1838  02674 031447        STA SLCOD       SET TO STANDARD PRINTER
1839  02675 045751        ISZ TMP5        CHECK DELETE FLAG
1840  02676 170402        RET             DELETE
1841  02677 064357        JMP RETN2       ACCEPT
1842  02700 025400 P5     LDB BADDR
1843  02701 014211        CPB .BUFA       BUFFER EMPTY?
1844  02702 066625        JMP P1-2        YES, IGNORE
1845  02703 066671        JMP P6          NO, BACKSPACE
1846  02704 020031 P7     LDA EOL
1847  02705 061016        JSM ONEXT       PUT EOL CHAR IN BUFFER
1848  02706 164174        JMP RDYDA,I     EXIT PTAPE MODE
1850*
1851*     BACKSPACE
1852*
1853  02707 021704 BKSP   LDA BKPTR
1854  02710 072350        SZA BK2         FIRST BACKSPACE?
1855  02711 021400        LDA BADDR       YES
1856  02712 010212        CPA .SLBF       NULL RECORD?
1857  02713 066045        JMP REED6       YES
1858  02714 021400        LDA BADDR
1859  02715 031704        STA BKPTR
1860  02716 066723        JMP BK1-1       PUT POINTER AT LAST CHAR
1861  02717 010212 BK2    CPA .SLBF       ALREADY AT FIRST CHAR?
1862  02720 066045        JMP REED6       YES
1863  02721 011402        CPA OTPTR       LEFTMOST ON SCREEN?
1864  02722 055402        DSZ OTPTR       YES, SCROLL
1865  02723 055704        DSZ BKPTR       BACKSPACE CURSOR
1866  02724 066045 BK1    JMP REED6
1867*
1868*     FORWARD SPACE
1869*
1870  02725 021704 FORWD  LDA BKPTR
1871  02726 070310        SZA FW1         SKIP IF NO BACKSPACE HAS OCCURRED
1872  02727 045704        ISZ BKPTR
1873  02730 021402        LDA OTPTR
1874  02731 000041        ADA .32
1875  02732 011704        CPA BKPTR       ON RIGHT OF SCREEN?
1876  02733 011400        CPA BADDR       YES, LINE FULL?
1877  02734 066045 FW1    JMP REED6
1878  02735 045402        ISZ OTPTR       NO, SCROLL
1879  02736 066045        JMP REED6
1880*
1881*     LEFT SUBROUTINE
1882*
1883*     SHIFT THE 32 CHARACTER DISPLAY LEFT
1884*
1885  02737 021400 LEFT   LDA BADDR       IS THE LAST CHAR
1886  02740 000151        ADA M32             ALREADY TO THE LEFT OF
1887  02741 070076        TCA
1888  02742 001402        ADA OTPTR           THE END OF DISPLAY?
1889  02743 070113        SAP **2
1890  02744 045402        ISZ OTPTR       NO, SCROLL
1891  02745 170402        RET
1892*
1893*     RIGHT SUBROUTINE
1894*
1895*     SHIFT THE 32 CHARACTER DISPLAY RIGHT
1896*     ONE POSITION. IF THE FIRST CHARACTER
1897*     IS IN THE LEFTMOST POSITION, ABORT.
```

#01 KEYBOARD ENTRY AND DISPLAY

```
1898*
1899  02746 021402 RIGHT LDA OTPTR
1900  02747 010212       CPA .SLAF
1901  02750 170402       RET
1902  02751 055402       DSZ OTPTR      NO
1903  02752 170402       RET
1905*    KEYIN SUBROUTINE
1906*
1907*    THIS SUBROUTINE SCANS ALL
1908*    INPUT DEVICES SEARCHING FOR
1909*    AN INPUT CODE, IT REFRESHES
1910*    THE DISPLAY WHILE WAITING
1911*    EXIT WITH CODE IN (A)
1912*
1913  02753 074742 KEYIN SHR 16
1914  02754 021400       LDA HADDR
1915  02755 011704       CPA BKPTR      CURSOR PAST END?
1916  02756 035704       STB BKPTR      YES, CLEAR POINTER
1917  02757 021473 KEYN7 LDA CCNT
1918  02760 072110       SZA **2
1919  02761 067002       JMP KEYN1
1920  02762 061053       JSM GTMP       GET CHARACTER
1921  02763 031751       STA TMP5       SAVE UNCONVERTED CHAR
1922  02764 070550       SZA KEYN1-3    EMPTY BUFFER?
1923  02765 061363       JSM GTCON+1    CONVERT
1924  02766 011751       CPA TMP5       DID CONVERSION OCCURR?
1925  02767 066771       JMP **2        NO
1926  02770 043121       IOR B200       YES, SET SHIFT BIT
1927  02771 025473       LDB CCNT
1928  02772 010031       CPA EOL        END OF LINE?
1929  02773 074212       SHM **4        RECALL MODE?
1930  02774 055473       DSZ CCNT       NO, DINC POINTER
1931  02775 170402       RET
1932  02776 170402       RET            RETURN WITH CHAR IN (A)
1933  02777 074742       SHR 16
1934  03000 035473       STB CCNT       RESET POINTER
1935  03001 066756       JMP KEYN7-1
1936  03002 031750 KEYN1 STA TMP4       SET CURSOR COUNTER TO ZERO
1937  03003 063024       JSM KEYN6
1938  03004 025704       LDB BKPTR
1939  03005 074250       SZB KEYN3      CURSOR?
1940  03006 027122       LDB BKPTA      YES
1941  03007 061041       JSM GETCH
1942  03010 031751       STA TMP5       SAVE CHAR AT CURSOR POSITION
1943  03011 055704       DSZ BKPTR
1944  03012 025443 KEYN3 LDB KEYFG
1945  03013 074612       SHM KEYN2      HAS KEY INPUT OCCURRED?
1946  03014 021704       LDA BKPTR      YES
1947  03015 070250       SZA KEYN5      CURSOR?
1948  03016 021751       LDA TMP5       YES, RESTORE CHARACTER
1949  03017 027122       LDB BKPTA
1950  03020 061017       JSM OUTCR
1951  03021 055704       DSZ BKPTR
1952  03022 021443 KEYN5 LDA KEYFG
1953  03023 050105       AND B377
1954  03024 024132 KEYN6 LDB M1
1955  03025 035443       STB KEYFG      RESET KEY OCCURRED FLAG
1956  03026 170402       RET
1957  03027 025704 KEYN2 LDB BKPTR
1958  03030 074610       SZB KEYN4      CURSOR?
1959  03031 025750       LDB TMP4       YES
1960  03032 004132       ADB M1
1961  03033 014136       CPB M5
1962  03034 024061       LDB .10
1963  03035 035750       STB TMP4
1964  03036 021751       LDA TMP5
1965  03037 074113       SBP **2
1966  03040 020037       LDA .28
1967  03041 027122       LDB BKPTA
1968  03042 061017       JSM OUTCR      REPLACE CHARACTER WITH CURSOR
1969  03043 055704       DSZ BKPTR
1970  03044 062562 KEYN4 JSM REFRS
1971  03045 067012       JMP KEYN3
1972*
1973*  CALL KEYIN AND EXECUTE RIGHT AND LEFT ARROWS
1974*
1975*  P+1=ARROW1  P+2=OTHERWISE
1976  03046 062753 CALLK JSM KEYIN
1977  03047 074742       SHR 16
1978  03050 070346       RAR 8
1979  03051 070133       SAP **2,C      SHIFTED?
1980  03052 024132       LDB M1         YES
1981  03053 035706       STB TMP6
1982  03054 070346       RAR 8
1983  03055 031751       STA TMP5
1984  03056 010036       CPA B27        RIGHT ARROW
1985  03057 066746       JMP RIGHT      YES
1986  03060 010035       CPA B26
1987  03061 066737       JMP LEFT       LEFT ARROW
1988  03062 064357       JMP RETN2
```

#01 KEYBOARD ENTRY AND DISPLAY

```
1990*
1991*     TABLE SEARCH ROUTINE
1992*
1993*
1994*     THIS SUBROUTINE COMPARES THE
1995*     INPUT STRING WITH STRINGS IN
1996*     A TABLE.  IF A SUCCESSFUL COMPARE
1997*     IS MADE AN 6-BIT CODE IS RETURNED
1998*
1999*     ENTRY CONDITIONS:
2000*         (A)=FIRST CHARACTER OF INPUT STRING
2001*         (B)=POINTER TO FIRST CHAR OF TABLE
2002*
2003*     EXIT CONDITIONS:
2004*        IF MATCH, RETURN TO P+2 WITH CODE
2005*        IN A OTHERWISE RETURN TO P+1
2006*
2007*
2008  #3063 #62531 TBS6 JSM GTMP2
2009  #3064 #70306       RAR 7
2010  #3065 #70511       SLA TBS5       OP CODE?
2011  #3066 #70402       SAR 9          YES, RIGHT JUSTIFY
2012  #3067 #50054       AND .63
2013  #3070 #64357       JMP RETN2
2014  #3071 #21751 TBSR1 LDA TMP5
2015  #3072 #31750 TBSRH STA TMP4        SAVE INPUT CHAR
2016  #3073 #35742       STB TMP2       INITIALIZE TABLE POINTER
2017  #3074 #25400       LDB BADDR
2018  #3075 #35752       STB TMP6A      SAVE INPUT POINTER
2019  #3076 #67101       JMP TBS2
2020  #3077 #55742 TBS5  DSZ TMP2
2021  #3100 #61055       JSM GNEXT      FETCH NEXT INPUT CHAR
2022  #3101 #31743 TBS2  STA TMP3       STORE CHARACTER
2023  #3102 #62531       JSM GTMP2
2024  #3103 #11743       CPA TMP3       CHARACTERS MATCH?
2025  #3104 #67063       JMP TBS6       YES, CONTINUE
2026  #3105 #25752       LDB TMP6A      NO, RESTORE BADDR
2027  #3106 #35400       STB BADDR
2028  #3107 #62531       JSM GTMP2
2029  #3110 #70306       RAR 7
2030  #3111 #71711       SLA *-2       CHAR IS OP CODE?
2031  #3112 #70153       SAP TBS4       LAST ENTRY?
2032  #3113 #21750       LDA TMP4       RESTORE FIRST CHARACTER
2033  #3114 #170402      RET             RETURN TO P+1
2034  #3115 #21750 TBS4  LDA TMP4
2035  #3116 #67101       JMP TBS2       CONTINUE
2037*
2038*    CONSTANTS, TEMPORARIES, ETC.
2039*    USED IN ABOVE ROUTINES
2040*
2041*
2042  #3117 #01742 TMP2A DEF TMP+2
2043  #3120 #06006 LISTR DEF LIST1
2044  #3121 #00200 B200  OCT 200
2045  #3122 #01704 BKPTA DEF BKPTR
2046  #3123 #01621 ENDW  DEF KBUFF+40
2047  #3124 #00033 B33   OCT 33
2048  #3125 #00176 B176  OCT 176
2049  #3126 #00137 B137  OCT 137
2050  #3127 #00163 DELIN OCT 163
2051  #3130 #03321 IOEND DEF IOBUF+IOBUF+81
```

#01 DEFINE-A-KEY-ROUTINES

```
2053*
2054*    DEFINE-A-KEY-ROUTINES
2055*
2056*  THE FIRST WORD OF EACH DEFINITION CONTAINS THE
2057*  KEYCODE OF THAT KEY.  THE SECOND WORD CONTAINS THE LENGTH
2058*  OF THE DEFINITION PLUS 1.  WHILE IN DEFINITION MODE
2059*  "BKPTR" POINTS THE THE LENGTH WORD.
2060*
2061  #3131 #60626 KEYPR JSM SAVBP       SAVE BADDR
2062  #3132 #20212       LDA .SLRF
2063  #3133 #31400       STA BADDR
2064  #3134 #61055       JSM GNEXT       SET UP FOR TABLE SEARCH
2065  #3135 #27152       LDB KTBA
2066  #3136 #160175      JSM TSRHA,I     SEARCH TABLE
2067  #3137 #67153       JMP RCAL        NOT FOUND
2068  #3140 #74742       SBR 16
2069  #3141 #35432       STB DKQST
2070  #3142 #50024       AND .7
2071  #3143 #03145       ADA JTBA        A-REG HAS OPCODE
2072  #3144 #70737       JMP A,I
2073*
2074  #3145 103145 JTBA  DEF JTBL-1,I
2075  #3146 #03253 JTBL  DEF XRUN   RUN
2076  #3147 #03310       DEF XSCR   SCRATCH
```

```
2077  03150 003330        DEF XLST   LIST
2078  03151 003261        DEF XKEY   FETCH
2079*
2080  03152 007155 KTRA   DEF KTRL*KTRL*1
2082  03153 060633 RCAL   JSM RESR1        RESTORE RADDR
2083  03154 063346        JSM HUNT         FIND THE LOCATION
2084  03155 160206 KERR   JSM ERRA,I       NOT FOUND, ERROR
2085  03156 074517        LDA B,I
2086  03157 031466        STA LOCL1        SAVE LENGTH WORD
2087  03160 070744        SAL 1
2088  03161 000136        ACA M5           ASSUME ODD COUNT
2089  03162 070113        SAP *+2        NULL DEFINITION?
2090  03163 066045        JMP REED6      YES, IGNORE
2091  03164 031473        STA CCNT       SAVE CHARACTER COUNT
2092  03165 035470        STB L2T1         SAVE LOCATION
2093  03166 074070        SIB *+1
2094  03167 074517        LDA B,I          LOOK AT LINE #
2095  03170 074070        SIB *+1
2096  03171 070110        SZA *+2          IS THIS A TYPING AID?
2097  03172 067222        JMP RFNX         NO
2098  03173 074744        SBL 1            YES
2099  03174 035740        STB TMP        SAVE CHARACTER POINTER
2100  03175 074002        SBR 1
2101  03176 005466        ADB LOCL1
2102  03177 004134        ADB M3
2103  03200 074517        LDA B,I          LOOK AT LAST WORD
2104  03201 050105        AND B377
2105  03202 070450        SZA *+9
2106*
2107  03203 045473        ISZ CCNT       CORRECT COUNT
2108  03204 010100        CPA ASTR1        CHECK FOR *
2109  03205 067207        JMP *+2
2110  03206 066045        JMP REED6
2111  03207 074517        LDA B,I
2112  03210 050157        AND M256
2113  03211 040027        IOR AIEX1        IEX CODE
2114  03212 067220        JMP *+6
2115*
2116  03213 074517        LDA B,I
2117  03214 013362        CPA ASTR2        CHECK FOR *
2118  03215 067217        JMP *+2
2119  03216 066045        JMP REED6
2120  03217 023363        LDA AIEX2        IEX CODE
2121  03220 074557        STA B,I
2122  03221 066045        JMP REED6
2123*
2124  03222 074517 RFNX   LDA B,I          CHECK FOR LEGITIMATE FUNCTION
2125  03223 050065        AND K1           (076000)
2126  03224 010004        CPA K2           (010000)
2127  03225 076130        RIB *+2
2128  03226 067246        JMP RRUN         NOT DEF
2129  03227 074517        LDA B,I
2130  03230 053357        AND K3           (176037)
2131  03231 013360        CPA K4           (012037)
2132  03232 067234        JMP *+2
2133  03233 067246        JMP RRUN         NOT DEF
2134  03234 020162        LDA FN
2135  03235 031754        STA AR2          "FN"
2136  03236 074517        LDA B,I
2137  03237 070202        SAR 5            EXTRACT THE LETTER
2138  03240 050040        AND .31
2139  03241 040101        IOR B100
2140  03242 070404        SAL 8
2141  03243 031755        STA AR2+1        "X("
2142  03244 027361        LDB ALCL1      GET POINTER
2143  03245 066246        JMP RFNX1      SET UP COUNT
2144*
2145  03246 061225 RRUN   JSM LDEF       LEAVE DEF MODE
2146  03247 025470        LDB L2T1         YES, CAN RUN
2147  03250 063336        JSM SUMP
2148  03251 070742        SAR 16
2149  03252 066157        JMP CONTK      CONTINUE FROM BEGINNING OF KEY
2150*
2151  03253 061225 XRUN   JSM LDEF       INSURE NOT IN KEY MODE
2152  03254 063346        JSM HUNT
2153  03255 067155        JMP KERR       KEY UNDEFINED
2154  03256 063336        JSM SUMP       SET UP PROGRAM POINTERS
2155  03257 070742        SAR 16
2156  03260 164171        JMP RUNA,I     RUN
2157*
2158*
2159  03261 061225 XKEY   JSM LDEF         INSURE NOT IN DEFINITION MODE
2160  03262 063346        JSM HUNT         FIND THE LOCATION
2161  03263 063270        JSM NEWK         NOT FOUND, NEW KEY REQUIRED
2162  03264 063336        JSM SUMP
2163  03265 060651        JSM INITS        INITIALIZE STACKS
2164  03266 070742        SAR 16
2165  03267 066254        JMP EFET+1     FETCH FIRST LINE
2166*
2167  03270 160242 NEWK   JSM DCOMA,I    DECOMPILE PROGRAM
```

```
2168  #3271 #21405        LDA FWUP       ADD A NEW KEY TO DEFINITION AREA
2169  #3272 #31476        STA TEMP3
2170  #3273 #20#17        LDA .2
2171  #3274 160250        JSM OVCHA,I      CHECK FOR OVERFLOW
2172  #3275 #61121        JSM MVTOH        MOVE CORE
2173  #3276 #21751        LDA TMP5
2174  #3277 131405        STA FWUP,I       STORE NEW KEY
2175  #3300 #45405        ISZ FWUP
2176  #3301 #25405        LDB FWUP
2177  #3302 #45405        ISZ FWUP         UPDATE FWUP
2178  #3303 #20#12        LDA .1
2179  #3304 #74557        STA B,I          INITIALIZE LENGTH WORD
2180  #3305 #45424        ISZ PBUFF
2181  #3306 #45424 NEWK1  ISZ PBUFF        UPDATE PBUFF
2182  #3307 170402        RET
2184*
2185* SCRATCH A KEY
2186*
2187  #3310 #61225 XSCR   JSM LDEF         INSURE NOT IN DEFINITION MODE
2188  #3311 160242        JSM DCOMA,I    DECOMPILE
2189  #3312 #63346        JSM HUNT         FIND THE LOCATION
2190  #3313 164174        JMP RDYDA,I      NOT FOUND, IGNORE
2191  #3314 #74117        LDA B
2192  #3315 #70437        ADA A,I          END OF AREA
2193  #3316 #00132        ADA M1           START OF AREA
2194  #3317 #31477 XSCR1  STA TEMP4
2195  #3320 160251        JSM CLPIA,I      CLOSE UP AREA
2196  #3321 #21405        LDA FWUP
2197  #3322 #74017        ADA B
2198  #3323 #31405        STA FWUP         UPDATE FWUP
2199  #3324 #05424        ADB PBUFF
2200  #3325 #35424        STB PBUFF        UPDATE PBUFF
2201  #3326 #64651        JSM INITS      INITIALIZE STACKS
2202  #3327 164174        JMP RDYDA,I      RETURN TO MONITOR
2203*
2204* LIST A DEFINITION
2205*
2206  #3330 #63346 XLST   JSM HUNT         FIND THE LOCATION
2207  #3331 #67615        JMP XLST1+1    NOT FOUND
2208  #3332 #74117        LDA B
2209  #3333 #70437        ADA A,I          END OF AREA
2210  #3334 #70070        STA *+1          START OF AREA
2211  #3335 #67614        JMP XLST1      LIST DEFINITION
2212*
2213* SET UP MISCELLANEOUS POINTERS
2214*
2215  #3336 #21424 SUMP   LDA PBUFF
2216  #3337 #31425        STA PBUF0        SAVE OLD PBUFF
2217  #3340 #35422        STB UKPTR
2218  #3341 #35424        STB PBUFF
2219  #3342 #74517        LDA B,I
2220  #3343 #70037        ADB A
2221  #3344 #35423        STB PBPTR        ESTABLISH NEW PBPTR
2222  #3345 #67306        JMP NEWK1
2224*
2225* SUBROUTINE TO FIND A DEFINED KEY AREA
2226*
2227* RETURN TO P+1 IF NOT FOUND
2228* RETURN TO P+2 IF FOUND WITH B-REG = ADDRESS OF LENGTH WORD
2229*
2230  #3346 #24166 HUNT   LDB FWAM       START SEARCHING AT FWAM
2231  #3347 #15405 HUN1   CPB FWUP
2232  #3350 170402        RET                UNDEFINED
2233  #3351 #74517        LDA B,I
2234  #3352 #74070        SIB *+1          EXTRACT THIS KEYCODE
2235  #3353 #11751        CPA TMP5         AND COMPARE WITH DESIRED
2236  #3354 #64357        JMP RETN2        FOUND IT
2237  #3355 #74437        ADB B,I
2238  #3356 #67347        JMP HUN1
2239*
2240* CONSTANTS
2241*
2242  #0#65         K1    EQU UPMSK        (076000)
2243  #0004         K2    EQU B10K         (010000)
2244  #3357 176037 K3     OCT 176037
2245  #3360 #12037 K4     OCT #12037
2246  #3361 #03730 ALCL1  DEF AR2+AR2
2247  #0100         ASTR1 EQU B52
2248  #3362 #25000 ASTR2  OCT #25000
2249  #0027         AIEX1 EQU .11
2250  #3363 #05400 AIEX2  OCT #05400

          #01  SYSTEM TABLES


2252*
2253*  NON-FORMULA OPERATORS
2254*
2255  #3364 #52110 FOP13  ASC 2,THEN
      #3365 #42516
```

#01 SYSTEM TABLES

```
2256  @3366 150117        OCT 150117      THEN: 20
2257  @3367 @43321        OCT @43321      OF: 21
2258  @3370 @43203 FTABL  OCT 43203      F: 3
2259  @3371 @42704        OCT 42704       E: 4: EOT
2260  @3372 @41206        OCT 41206       B: 6
2261  @3373 @54207        OCT 54207       X: 7
2262  @3374 @27710        OCT 27710       /: 10: EOT
2263  @3375 @27311        OCT 27311       .: 11: EOT
2264  @3376 @51524 FOPST  ASC 2,STEP
      @3377 @42520
2265  @3400 156524        OCT 156524      STEP  35
2266  @3401 @47736        OCT @47736      TO: 36
2267  @3402 @47522 OR     ASC 1,OR
2268  @3403 113101        OCT 113101      OR: 26
2269  @3404 @47104        ASC 1,ND
2270  @3405 153516        OCT 153516      AND: 27
2271  @3406 @47524        ASC 1,OT
2272  @3407 154076 GTE    OCT 154076      NOT: 30
2273  @3410 @36631        OCT @36631      >=: 31
2274  @3411 @36475        ASC 1,<=
2275  @3412 115074        OCT 115074      <=: 32
2276  @3413 @37320        OCT @37320      <>: 20
2277*
2278*  SYSTEM COMMAND JUMP TABLE
2279*
2280  @3414 @04322        ABS FLT-*
2281  @3415 @04322        ABS FIXED-*
2282  @3416 @00437        ABS PTAPE-*
2283  @3417 176534 LHLI   ABS CONT-*
2284  @3420 @01237        ABS DELET-*
2285  @3421 @01244        ABS RENUM-*
2286  @3422 @04335        ABS NORML-*
2287  @3423 @04331        ABS TRACE-*
2288  @3424 @00151        ABS AUTO-*
2289  @3425 176626        ABS EFET-*
2290  @3426 @00155        ABS LISTC-*
2291  @3427 @01212        ABS SCRAT-*
2292  @3430 @01311        ABS RUN-*
2293*
2294*  SYSTEM COMMAND NAME TABLE
2295*
2296  @3431 @51105 CMD13  ASC 1,RE
2297  @3432 @47210        OCT 47210       REN: 001000
2298  @3433 @42105        ASC 1,DE
2299  @3434 @46271        OCT @46271      DELETE: 111001
2300  @3435 @50124        ASC 1,PT
2301  @3436 @40653        OCT 40653       PTAPE: 101011
2302  @3437 @43111        ASC 1,FI
2303  @3440 @54214        OCT 54214       FIXED: 001100
2304  @3441 @43114        ASC 1,FL
2305  @3442 @52215        OCT 52215       FLOAT: 001101
2306  @3443 @52122        ASC 2,TRAC
      @3444 @40503
2307  @3445 @42666        OCT 42666       TRACE: 110110
2308  @3446 @47117        ASC 3,NORMAL
      @3447 @51115
      @3450 @40514
2309  @3451 103503        OCT 103503      NORMAL: 000111
2310  @3452 @47516        ASC 1,ON
2311  @3453 @52252        OCT 52252       CONT: 101010
2312  @3454 @51524        ASC 2,STOP
      @3455 @47520
2313  @3456 137501        OCT 137501      STOP: 111111 (NO JUMP)
2314  @3457 @52524        ASC 1,UT
2315  @3460 @47605        OCT 47605       AUTO 000101
2316  @3461 @51524        ASC 2,STOR
      @3462 @47522
2317  @3463 @42600        OCT 42600       STORE: 000000 (NO JUMP)
2318  @3464 @46117        ASC 2,LOAD
      @3465 @40504
2319  @3466 110114 KTBL   OCT 110114      LOAD: 010000 (NO JUMP)
2320  @3467 @44523        ASC 1,IS
2321  @3470 @52203        OCT 52203       LIST: 000011
2322  @3471 @51125        ASC 1,RU
2323  @3472 @47241        OCT 047241      RUN: 100001
2324  @3473 @51503        ASC 3,SCRATC
      @3474 @51101
      @3475 @52103
2325  @3476 @44202        OCT 44202       SCRATCH: 000010
2326  @3477 @43105        ASC 2,FETC
      @3500 @52103
2327  @3501 @44344        OCT 44344       FETCH: 100100
```

#01 SYNTAX

```
2329*
2330*  DATA STATEMENT SYNTAX
2331*
```

#01 SYNTAX

```
2332  #3502 #45706 DATAS ISZ SBPTR
2333  #3503 #61156       JSM CONST     LOOK FOR A NUMBER
2334  #3504 160206 NDATA JSM ERRA,I    NOT FOUND
2335  #3505 160244       JSM NUM0A,I   FOUND; STORE IN SYNTAX BUFFER
2336  #3506 #60504 DATA2 JSM COMCK     COMMA?
2337  #3507 #65151       JMP EOST      NO
2338  #3510 #67502       JMP DATAS
2339*
2340*  FORMAT STATEMENT SYNTAX
2341*
2342  #3511 #61055 FMTS  JSM GNEXT     GET FIRST CHAR OF NEXT SPEC
2343  #3512 #60750       JSM DIGCK     DIGIT?
2344  #3513 #67521       JMP FMTF      NO
2345  #3514 #35724       STB GFLAG     YES, SET FOR ERROR ON OVERFLOW
2346  #3515 #55400       DSZ BADDR     FETCH AND
2347  #3516 #24161       LDB MAXSN       STORE
2348  #3517 #60666       JSM PGINT     REPEAT FIELD
2349  #3520 #67522       JMP *+2
2350  #3521 #45706 FMTF  ISZ SBPTR     MOVE TO NEW WORD
2351  #3522 #27552       LDB FADD      LOOK FOR
2352  #3523 #60546       JSM STEXP       "F" OR "E"
2353  #3524 #67537       JMP FMTB      NOT FOUND
2354  #3525 #24161       LDB MAXSN     FOUND; FETCH AND
2355  #3526 #60666       JSM PGINT     STORE FIELD WIDTH
2356  #3527 #27553       LDB .ADD      LOOK FOR
2357  #3530 #60546       JSM STEXP
2358  #3531 160206 FMTER JSM ERRA,I    NOT FOUND
2359  #3532 #24161 FMTD  LDB MAXSN
2360  #3533 #61131       JSM PGIN0     FETCH DIGIT FIELD, ALLOW 0
2361  #3534 #61153 FMTC  JSM EOLCK     EOL?
2362  #3535 #60510       JSM COMCF     NO, DEMAND A COMMA
2363  #3536 #67511       JMP FMTS      GET NEXT SPEC
2364  #3537 #27554 FMTB  LDB BADD
2365  #3540 #60546       JSM STEXP     SEARCH FOR "B", "X", OR "/"
2366  #3541 #67545       JMP FMTQT     NOT FOUND
2367  #3542 #61055       JSM GNEXT     FOUND
2368  #3543 #45706       ISZ SBPTR
2369  #3544 #67534       JMP FMTC      DEMAND COMMA OR EOL
2370  #3545 #10043 FMTQT CPA H42       QUOTE?
2371  #3546 #67550       JMP *+2
2372  #3547 #67531       JMP FMTER     NO
2373  #3550 160245       JSM GTSTA,I   YES, GET QUOTE FIELD
2374  #3551 #67534       JMP FMTC
2375  #3552 #06760 FADD  DEF FTABL+FTABL
2376  #3553 #06772 .ADD  DEF FTABL+FTABL+10
2377  #3554 #06764 BADD  DEF FTABL+FTABL+4
2378*
2379*  GOTO AND GOSUB STATEMENT SYNTAX
2380*
2381  #3555 #60626 GOTOS JSM SAVRP     SAVE BADDR AND SBPTR
2382  #3556 #55724       DSZ GFLAG     SET TO RETURN ON ERROR FROM INTCK
2383  #3557 #60664       JSM LNUM      LOOK FOR A LINE NUMBER
2384  #3560 #67572       JMP GOTO3     FOUND
2385  #3561 #60640 GOTO1 JSM RESRP     RESTORE POINTERS
2386  #3562 160202       JSM FSCA,I    FETCH FORMULA
2387  #3563 #27574       LDB OFA
2388  #3564 #60546       JSM STEXP     STORE "OF" AS EXPANDED OPERATOR
2389  #3565 160206 MISOF JSM ERRA,I    NOT FOUND
2390  #3566 #60664 GOTO2 JSM LNUM      FETCH LINE NUMBER
2391  #3567 #61153       JSM EOLCK     EOL?
2392  #3570 #60510       JSM COMCF     DEMAND A COMMA
2393  #3571 #67566       JMP GOTO2
2394  #3572 #61153 GOTO3 JSM EOLCK     EOL?
2395  #3573 #67561       JMP GOTO1     NO, TRY FOR A FORMULA INSTEAD
2396  #3574 #06755 OFA   DEF FOP13+FOP13+5
2397*
2398*  AUTO SYSTEM COMMAND
2399*
2400  #3575 #67647 AUTO  JSM GETLN     GET OPTIONAL PARAMETERS
2401  #3576 #20061       LDA .10
2402  #3577 #24061       LDB .10
2403  #3600 #31427       STA .LNUM     SET INITIAL LINE NUMBER
2404  #3601 #35430       STB LNINC     SET INCREMENT
2405  #3602 164173       JMP PEXMA,I
2406*
2407*  LIST EXECUTION
2408*
2409  #3603 #61055 LISTC JSM GNEXT     GET NEXT CHARACTER
2410  #3604 #17627       CPA H43       "#"?
2411  #3605 #67623       JMP LSTC2     YES
2412  #3606 #55400       DSZ BADDR     RESET POINTER
2413  #3607 #67647 LSTC1 JSM GETLN     GET OPTIONAL LINE #'S
2414  #3610 #70742       SAR 16
2415  #3611 #24056       LDB .1E4
2416  #3612 #60730       JSM GETAD     GET ADDRESSES
2417  #3613 #21477       LDA TEMP4
2418  #3614 160122 XLST1 JSM LISTA,I   LIST
2419  #3615 160170       JSM CLERA,I   CLEAR BUFFER
2420  #3616 #25412       LDB LSTPT
2421  #3617 #74476       TCB
```

#01 SYNTAX

```
2422  03620 005413        ADB HSTPT      COMPUTE WORDS BETWEEN STACKS
2423  03621 160210        JSM OUTIA,I    PUT IN BUFFER
2424  03622 164173        JMP PEXMA,I
2425  03623 060664 LSTC2  JSM LNUM       FETCH SELECT CODE
2426  03624 074204        SBL 12
2427  03625 035447        STB SLCOD
2428  03626 067607        JMP LSTC1
2429  03627 000043 043    OCT 43
2430*
2431*  STOP STATEMENT SYNTAX
2432*
2433  03630 065147        JMP ENDS
2434  03631 021416 STOPS  LDA MODE
2435  03632 071711        SLA STOPS-1    SKIP IF PROG
2436  03633 063647        JSM GETLN      GET PARAMETERS
2437  03634 020056        LDA .1E4
2438  03635 024056        LDB .1E4       SET DEFAULT PARAMETERS
2439  03636 060741        JSM FND-1      FIND ADDRESS OF FIRST
2440  03637 015423        CPB PBPTR      PAST END OF PROG?
2441  03640 074742        SBR 16         YES, CLEAR FLAG
2442  03641 035436        STB MFLG1
2443  03642 060740        JSM FND-2      FIND ADDRESS OF SECOND
2444  03643 015423        CPB PBPTR
2445  03644 074742        SBR 16
2446  03645 035437        STB MFLG1+1
2447  03646 164174        JMP RDYDA,I
2448*
2449*
2450*  SUBROUTINE TO BUILD AND STORE N>0 LINE NUMBERS
2451*    RETURN TO: P+1=NONE GIVEN
2452*               P+2:1 GIVEN (A)=NUMBER
2453*               P+3:2 GIVEN (A)=FIRST, (B)=SECOND
2454*    NUMBERS ARE STORED SEPARATED BY COMMAS IN S-BUFFER
2455*
2456  03647 061066 GETLN  JSM GETSK      GET NEXT CHAR
2457  03650 170402        RET            NONE GIVEN
2458  03651 060750        JSM DIGCK      DIGIT?
2459  03652 067647        JMP GETLN      NO, IGNOR IT
2460  03653 060663        JSM LNUM-1     BUILD FIRST INTEGER
2461  03654 035720        STB LASTN      SAVE 1ST #
2462  03655 010031        CPA EOL        END OF LINE?
2463  03656 067662        JMP GETL2      YES,RETURN TO P+2
2464  03657 060365        JSM RET2       INCREMENT RETURN ADDRESS
2465  03660 060510        JSM COMCF      FIND AND RECORD A COMMA
2466  03661 060664        JSM LNUM       GET NEXT LINE NUMBER
2467  03662 021720 GETL2  LDA LASTN      MOVE TO (A)
2468  03663 064357 GETLR  JMP RETN2      RETURN TO P+2 OR P+3
2469*
2470  03664 000000        BSS 1          *******CHECKSUM*******
2471*
2472*  CLEAR ROUTINE
2473*
2474  03665 070742 ECLR   SAR 16
2475  03666 031430        STA LNINC      CLEAR AUTO MODE
2476  03667 170402        RET            RETURN TO P+1
2477*  3677 IS LAST
2478*  MATH TABLES HERE
```

#01 9830--MONITOR

```
2480*
2481  04000               ORG 4000B
2482*
2483  04000 020116 ENTRY  LDA QTOP       CLEAR THE BASE PAGE
2484  04001 170000        CLR
2485  04002 000135        ADA M4
2486  04003 073710        RZA *-2
2487  04004 024066        LDB INF        DETERMINE THE
2488  04005 004160 MAWCK  ADB M1024
2489  04006 074577        STB B,I
2490  04007 074477        CPB B,I        LAST WORD OF R/W
2491  04010 066012        JMP **2        YES , IT WAS
2492  04011 066005        JMP MAWCK      CHECK NEXT PAGE
2493*
2494*
2495  04012 074557        STA B,I        INITIALIZE LAST WORD
2496  04013 035403        STB MAW
2497  04014 004132        ADB M1         SET THE POINTER
2498  04015 035404        STB LWAM         TO THE LAST AVAILABLE WORD
2499  04016 024116        LDB REED
2500  04017 035462        STB REEDL      INITIALIZE KEYBOARD LINK
2501  04020 024207        LDB INTEA
2502  04021 035463        STB INTFL      INITIALIZE INTERRUPT LINK
2503  04022 020026        LDA .9
2504  04023 031452        STA CPLAC      SET DECIMAL SETTING TO 9
2505  04024 031451        STA CTYPE      SET MODE TO "FLOAT"
2506  04025 045434        ISZ UNITS      INITIALIZE TO RADIANS
2507  04026 045450        ISZ PMODE      INITIALIZE TO NORMAL
2508  04027 020166 KILLK  LDA FWAM       INITIALIZE START OF
```

```
2509  04030 031405        STA FWUP        USER PROGRAM(SCRATCH KEYS)
2510  04031 020112 KILLP  LDA RSTKA       INITIALIZE RETURN
2511  04032 031777        STA RSPTR       POINTER
2512  04033 062734        JSM SCRP        INITIALIZE PROGRAM POINTERS
2513  04034 061233        JSM LOFF,C      RESET USER KEY POINTER
2514  04035 060645 KILLV  JSM MINIT       INITIALIZE SYMBOL TABLE
2515  04036 060651        JSM INITS       INITIALIZE STACKS AND FLAGS
2516  04037 070742 RDYPT  SAR 16
2517  04040 031433        STA TFLAG       CLEAR PTAPE FLAG
2518  04041 031430        STA LNINC       SET TO MANUAL
2519*
2520  04042 160170        JSM CLERA,I     CLEAR DISPLAY
2521  04043 031622        STA TERM        RESET CRLF FLAG
2522  04044 070742 PEXMK  SAR 16
2523  04045 031701        STA NOISP
2524  04046 031702        STA NPRIN
2525  04047 021432 PEXM2  LDA DRQST
2526  04050 070510        SZA PEXM3       SKIP IF NOT DATA REQUEST
2527  04051 020012        LDA .1
2528  04052 026267        LDB QMRKA         PRINT A ?
2529  04053 061330        JSM .NPWR
2530  04054 066100        JMP PEXM1
2531*  PTAPE SYSTEM COMMAND
2532  04055 031433 PTAPE  STA TFLAG       STORE DEVICE CODE
2533  04056 072110        RZA **2
2534  04057 064726        JMP SYE25       SELECT CODE NOT GIVEN
2535  04060 020132        LDA M1
2536  04061 031742        STA TMP2        SET FLAG FOR LEADER
2537  04062 070742 PEXM3  SAR 16
2538  04063 031442        STA STPFL       CLEAR THE STEP-STOP FLAG
2539  04064 020127        LDA H170K
2540  04065 031447        STA SLCOD       SET TO STANDARD PRINTER
2541  04066 021430        LDA BADDR
2542  04067 010212        CPA .SLBF       BLANK SCREEN?
2543  04070 066072        JMP **2         YES
2544  04071 066100        JMP PEXM1
2545  04072 026264        LDB LTADD       GET ADDRESS OF LAZY-T
2546  04073 021422        LDA UKPTR
2547  04074 070170        SIA **3         SKIP IF NOT KEY MODE (A)=1
2548  04075 004134        ADB M3          GET ADDRESS OF KEY-LAZY-T
2549  04076 020021        LDA .4
2550  04077 061335        JSM NPWRT       WRITE ON DISPLAY
2551*
2552  04100 024112 PEXM1  LDB RSTKA
2553  04101 035777        STB RSPTR       RESET STACK
2554  04102 020041 ALNUM  LDA .32
2555  04103 031707        STA BLANK       SET TO IGNOR BLANKS
2556  04104 021430        LDA LNINC
2557  04105 070410        SZA GTRC1       SKIP IF MANUAL
2558  04106 021701        LDA NOISP
2559  04107 072350        RZA GTRC1+1     SKIP IF MESSAGE ON SCREEN
2560  04110 160170        JSM CLERA,I     CLEAR OFF LAZY-T
2561  04111 025427        LDB .LNUM
2562  04112 160210        JSM OUTIA,I     PUT LINE # IN BUFFER
2563  04113 045400        ISZ BADDR       OUTPUT A SPACE
2564  04114 066116        JMP **2
2565  04115 031427 GTRC1  STA .LNUM       SET LINE # IF MANUAL
2566  04116 161462 GTRCD  JSM KEEDL,I     READ RECORD
2567  04117 066042        JMP PEXMK-2     NULL RECORD
2568  04120 061062        JSM GFRST       GET FIRST CHAR IN (A)
2569  04121 066042        JMP PEXMK-2     NULL RECORD
2570  04122 074742        SBR 16
2571  04123 035724        STB GFLAG       SET FLAG FOR INTEGER ERROR
2572  04124 025432        LDB DRQST
2573  04125 074730        SIB CKRCD       SKIP IF NOT DATA REQUEST
2574  04126 035642        STB SAVE        SAVE RETURN ADDRESS
2575  04127 074742        SBR 16
2576  04130 035432        STB DRQST       CLEAR THE INPUT FLAG
2577  04131 035416        STB MODE        SET TO PROGRAM EXECUTION MODE
2578  04132 160237        JSM SALTA,I     STATEMENT GIVEN?
2579  04133 066260        JMP GTRC2       NO
2580  04134 020127 GTRC3  LDA H170K       YES
2581  04135 031447        STA SLCOD       SET TO STANDARD PRINTER
2582  04136 022270        LDA CALSN
2583  04137 031416        STA MODE        SET TO CALC SYNTAX MODE
2584  04140 061062        JSM GFRST
2585  04141 000000        NOP             *** CHECKSUM
2586  04142 066154        JMP SYNTX
2587  04143 025416 CKRCD  LDB MODE
2588  04144 074111        SLB **2         SKIP IF EOL
2589  04145 066154        JMP SYNTX       EXECUTE: DONT ALLOW LINE #
2590  04146 010130        CPA STAR        FIRST CHAR AN "*"?
2591  04147 066251        JMP UKDEF       YES
2592  04150 060675        JSM INTCK-1     BUILD A LINE #
2593  04151 035427        STB .LNUM       STORE LINE #
2594*                     (A) CONTAINS NEXT CHAR
2595  04152 010031        CPA EOL         END OF LINE?
2596  04153 066500        JMP DELST       YES, DELETE LINE FROM CODE
2597*
2598  04154 025426 SYNTX  LDB PBPTO       GET MAINLINE END POINTER
```

```
2599  #4155 #15424        CPB PBUFF      NULL PROGRAM?
2600  #4156 #62734        JSM SCRP       YES. INSURE NO SPURIOUS COMMON EXISTS
2601  #4157 #24026        LDB .9         SET STORE FLAG FALSE
2602  #4160 #35710        STB MSFLG        (DONT RECOGNISE ASSIGNMENT)
2603  #4161 #25413        LDB MSTPT      INITIALIZE
2604  #4162 #04132        ADB M1           THE S-STACK
2605  #4163 #35472        STB TEMPS          POINTER
2606  #4164 #74742        SBR 16
2607  #4165 #35711        STB DFLAG      CLEAR DEFINE FLAG
2608  #4166 #35714        STB SFLAG
2609  #4167 160237 STMT   JSM SALTA,I    SEARCH TABLES FOR STATEMENT
2610  #4170 #66201        JMP CMND       NOT FOUND
2611  #4171 #45706        ISZ SBPTR
2612  #4172 135706        STB SBPTR,I    STORE THE PAGE
2613  #4173 #25416        LDB MODE
2614  #4174 #74211        SLB STMT1      SKIP IF PROGRAM MODE
2615  #4175 #70304        SAL 1#         SHIFT OP CODE
2616  #4176 #70112        SAM STMT1      SKIP IF STMT TYPE IS ALLOWED
2617  #4177 #62360 NCALC  JSM ERROR      STMT NOT ALLOWED IN CALC
2618  #4200 165473 STMT1  JMP TEMP,I     BRANCH TO CODE FOR STATEMENT
2619  #4201 #24132 CMND   LDB M1         SEARCH TABLES FOR
2620  #4202 160237        JSM SALTA,I      SYSTEM COMMAND
2621  #4203 #66221        JMP ILET       NOT FOUND
2622  #4204 #74742 CMNDF  SBR 16
2623  #4205 #35431        STB CLINE      RESET POINTER FOR LIST
2624  #4206 #35430        STB LNINC      RESET AUTO
2625  #4207 #35416        STB MODE       SET TO PROG
2626  #4210 #35442        STB STPFL      CLEAR "STEP" FLAG
2627  #4211 #70142        SAR 4          GET # OF PARAMETERS ALLOWED
2628  #4212 #70210        SZA JCMND      BRANCH IMMEDIATELY
2629  #4213 162733        JSM GETLA,I    GET LINE # PARAMETERS
2630  #4214 #70742        SAR 16         NONE GIVEN.ASSUME #
2631  #4215 #24056        LDB .1E4       ONLY ONE GIVEN; ASSUME INFINITY
2632  #4216 161473 JCMND  JSM TEMP,I     BRANCH TO COMMAND EXECUTION
2633*  ALL COMMAND JUMP TO PEXMK OR RDYPT EXCEPT FETCH
2634*     WHICH JUMPS TO READ; THE FOLLOWING TWO INSTRUCTIONS
2635*     ARE RETURN EXITS FOR READ
2636  #4217 #66042        JMP PEXMK-2
2637  #4220 #66120        JMP GTRCD+2
2638*
2639  #4221 #70742 ILET   SAR 16        GET SPEICAL CHAR FOR IMPLIED LET
2640  #4222 #66164        JMP STMT-3     FIND BLOCK FOR IMPLIED LET
2641*
2642  #4223 #55400 ILETS  ISZ BADDR
2643  #4224 #61055        JSM GNEXT      RECALL CHARACTER
2644  #4225 #55400        ISZ BADDR
2645  #4226 #60752        JSM LETCK      LETTER?
2646  #4227 #66234        JMP ILET1+1     NO. TRY IMPLIED DISP
2647  #4230 #60626        JSM SAVRP      SAVE POINTERS
2648  #4231 #62274        JSM LETSN      TRY "LET" SNTAX
2649  #4232 #66244        JMP NSTMT+1    OK. CHECK FOR EOL
2650  #4233 #60640 ILET1  JSM RESRP      RESTORE POINTERS
2651  #4234 #20125        LDA PRTOP      STORE CODE FOR
2652  #4235 #55706        ISZ SBPTR
2653  #4236 131706        STA SBPTR,I      IMPLIED LET
2654  #4237 #45706        ISZ SBPTR
2655  #4240 #25416        LDB MODE
2656  #4241 #74111        SLB NSTMT      SKIP IF PROG
2657  #4242 #62315        JSM PRNTS      TRY "PRINT" SYNTAX
2658  #4243 #62360 NSTMT  JSM ERROR      NO STATEMENT TYPE CAN BE FOUND
2659  #4244 #61153        JSM EOLCK      END OF LINE?
2660  #4245 #21416        LDA MODE       NO
2661  #4246 #70111        SLA **2
2662  #4247 #66233        JMP ILET1      TRY PRINT SYNTAX IF CALC MODE
2663  #4250 #65152        JMP NOEOF
2664*
2665  #4251 #21422 UKDEF  LDA UKPTR      IN KEY DEFINITION MODE?
2666  #4252 #72110        RZA **2
2667  #4253 160206 UKERR  JSM ERRA,I     NO. DEFINITION NOT ALLOWED
2668  #4254 #74742        SBR 16
2669  #4255 135706        STB SBPTR,I    SET A # LINE NUMBER
2670  #4256 160253        JSM CHRSA,I    ACCEPT AN ASCII STRING
2671  #4257 #66520        JMP ACTKY      ACCEPT DEFINITION
2672*
2673  #4260 #24132 GTRC2  LDB M1
2674  #4261 160237        JSM SALTA,I    SYSTEM COMMAND IN "INPUT"?
2675  #4262 165642        JMP SAVE,I     NO. RETURN DATA TO INPUT STMT
2676  #4263 #66134        JMP GTRC1      YES. ABORT INPUT
2677*
2678  #4264 #10555 LTADD  DEF *-*-5
2679  #4265 #45505        ASC 1,KE
2680  #4266 #54437        OCT 54437      Y.LAZY-T
2681  #4267 #00131 QMRKA  DEF .63*.63+1
2682  #4270 100001 CALSN  OCT 100001
2683  #0116        IMPOP  EQU GTOP       OP CODE 1
2684  #0125        PRTOP  EQU LBOP       OP CODE 24
2685  #0100        STAR   EQU B52
2686  #0123        STOPG  EQU RBOP
2687*
2688*  LET STATEMENT SYNTAX
```

#01 9830--MONITOR

```
2689*
2690  04271 062274 LETS  JSM LETSN
2691  04272 065151       JMP LOST
2692  04273 160206 SYNE2 JSM ERRA,I
2693*
2694*  LET SYNTAX SUBROUTINE
2695*  EXIT P+1: STORE OCCURRED
2696*       P+2: NO STORE OCCURRED
2697*
2698  04274 055710 LETSN DSZ MSFLG     ENABLE ASSIGNMENT
2699  04275 160202       JSM FSCA,I    GET FORMULA
2700  04276 025714       LDA SFLAG
2701  04277 014133       CPB M2        DID A STORE OCCUR
2702  04300 170402       RET           YES
2703  04301 064357       JMP RETN2     NO
2704*
2705*  WRITE AND PRINT STATEMENT SYNTAX
2706*
2707  04302 160265 RITE  JSM IOHEA,I   GET DEVICE AND FORMAT
2708  04303 060534       JSM RPCKE     GET RIGHT PAREN
2709  04304 055400       DSZ BADDR
2710  04305 055706       DSZ SBPTR
2711*
2712  04306 062315 PRINS JSM PRNTS     CHECK SYNTAX (TO ACTST IF OK)
2713  04307 160206 SYE1S JSM ERRA,I    IMPROPER SYNTAX
2714*
2715*  PRINT STATEMENT SYNTAX SUBROUTINE
2716*
2717  04310 010043 PRIN1 CPA B42       QUOTE?
2718  04311 066325       JMP PRIN2+4   YES
2719  04312 060461       JSM SYMC?
2720  04313 012021       DEF COMMA-1   COMMA OR SEMICOLON?
2721  04314 066321       JMP PRIN2     NO
2722  04315 024132 PRNTS LDB M1
2723  04316 035712       STB PFLAG     ENABLE FORMULA AND TAB
2724  04317 061066       JSM GETSK
2725  04320 065150       JMP EOST-1
2726  04321 010043 PRIN2 CPA B42       QUOTE?
2727  04322 066324       JMP *+2       YES
2728  04323 066334       JMP PRIN3     NO
2729  04324 045706       ISZ SBPTR
2730  04325 160245       JSM GTSTA,I   RECORD STRING CONSTANT
2731  04326 061153       JSM EOLCK     EOL?
2732  04327 074742       SBR 16
2733  04330 135706       STB SBPTR,I   CLEAR OUT NEXT WORD
2734  04331 024132       LDB M1
2735  04332 035712       STB PFLAG     ENABLE FORMULA
2736  04333 066310       JMP PRIN1
2737  04334 045712 PRIN3 ISZ PFLAG     TAB OR FORMULA PERMITTED?
2738  04335 170402       RET           NO. RETURN. IMPROPER SYNTAX
2739  04336 026354       LDB TABA
2740  04337 160175       JSM ISRHA,I   LOOK FOR "TAB"
2741  04340 066350       JMP PRIN4     NO
2742  04341 070544       SAL 5
2743  04342 040067       IOR TYPEI     SET PRE-DEFINED FUNCTION FLAGS
2744  04343 060454       JSM SBPUD-2   STORE CODE FOR TAB
2745  04344 024123       LDB RHOP
2746  04345 135706       STB SBPTR,I
2747  04346 160202       JSM FSCA,I    GET TAB FORMULA
2748  04347 066352       JMP PRIN5
2749  04350 055400 PRIN4 DSZ BADDR     BACK UP POINTER
2750  04351 160202       JSM FSCA,I    FETCH FORMULA
2751  04352 061153 PRIN5 JSM EOLCK     EOL?
2752  04353 066310       JMP PRIN1     NO
2753  04354 021773 TABA  DEF TAB+TAB+1
2754*
2755*  OUTPUT ERROR # GIVEN IN P+1
2756*
2757  04355 060405 AERR  JSM XFAR1     SAVE AR2 IN AR1
2758  04356 060400       JSM GPARM     GET ERROR # IN (B)
2759  04357 066370       JMP ERR1
2760*
2761*  OUTPUT ERROR # FROM ERROR TABLE
2762*
2763  04360 055777 ERROR DSZ RSPTR
2764  04361 121777       LDA RSPTR,I   ERROR SOURCE IN (A)
2765  04362 026476       LDB ERRS      TABLE ADDRESS IN (B)
2766  04363 074070       SIB *+1       INC ADDRESS
2767  04364 074457       CPA B,I       MATCH?
2768  04365 074076       TCB           YES
2769  04366 075653       SBP *-3       CONTINUE IF NO MATCH
2770  04367 006476       AUB ERRS      COMPUTE NEGATIVE OF ERROR #
2771  04370 035706 ERR1  STB SBPTR     SAVE -(ERROR #)
2772  04371 021450       LDA PMODE
2773  04372 072350       RZA ERR2
2774  04373 021433       LDA TFLAG
2775  04374 070250       SZA ERR2      SKIP IF NOT PTAPE
2776  04375 020060       LDA .100
2777  04376 024211       LDB .BUFA
2778  04377 061337       JSM NDWRT     PRINT LINE IF PTAPE AND P-ALL
```

#01 9830--MONITOR

```
2780  #4400 #61344        JSM CRLF1
2781  #4401 #21416 ERR2   LDA MODE
2782  #4402 #70112        SAM *+2         SKIP IF SYNTAX MODE
2783  #4403 160264        JSM CRCKA,I     TERMINATE CURRENT LINE
2784  #4404 160170        JSM CLERA,I     CLEAR BUFFER
2785  #4405 #35622        STB TERM        SET UNTERMINATED FLAG
2786  #4406 #20127        LDA B17#K
2787  #4407 #31447        STA SLCOD       SET TO STANDARD PRINTER
2788  #4410 #20023        LDA .6
2789  #4411 #26464        LDB EMSGA
2790  #4412 #61335        JSM NPWRT       PUT "ERROR" IN BUFFER
2791  #4413 #25706        LDB SBPTR
2792  #4414 #74076        TCB
2793  #4415 160210        JSM OUTIA,I     OUTPUT POSITIVE ERROR #
2794  #4416 #25427        LDB .LNUM       IS A LINE NUMBER
2795  #4417 #74310        SZB ERR3          APPLICABLE
2796  #4420 #20026        LDA .9
2797  #4421 #26465        LDB LMSGA
2798  #4422 #61335        JSM NPWRT       PUT "IN LINE " IN BUFFER
2799  #4423 #25427        LDB .LNUM
2800  #4424 160210        JSM OUTIA,I     OUTPUT LINE #
2801  #4425 #45400 ERR3   ISZ BADDR
2802  #4426 #45400        ISZ BADDR       OUTPUT 2 BLANKS
2803  #4427 #21433        LDA TFLAG
2804  #4430 #72110        RZA *+2
2805  #4431 #72053        SAP *+1,S       SET TO "PRINT" IF PTAPE
2806  #4432 #31453        STA PRFLG       SET "DISP"FLAG
2807  #4433 160176        JSM ELINA,I     OUTPUT ERROR
2808  #4434 160264        JSM CRCKA,I     GIVE CRLF IF P-ALL OR PTAPE
2809  #4435 #25706 ERR4   LDB SBPTR       RECALL -(ERROR #)
2810  #4436 #14132        CPB M1          ERROR 1?
2811  #4437 #66445        JMP ERN1        YES
2812  #4440 #04063        ADB .99
2813  #4441 #74313        SBP ERR5        SKIP IF NON-RECOVERABLE
2814  #4442 #35442        STB STPFL       SET TO STOP AT END OF LINE
2815  #4443 #35622        STB TERM        SET UNTERMINATED LINE FLAG
2816  #4444 #64444        JMP XFAR2       RESTORE AR2 AND RETURN
2817*
2818  #4445 #35421 ERN1   STB PRADD       SET TO ABORT DECOMPILE
2819  #4446 #35416        STB MODE        SET TO SCRATCH SYMTAB
2820  #4447 #21432 ERR5   LDA DROST
2821  #4450 #70110        SZA *+2         IN "INPUT" STATEMENT?
2822  #4451 165432        JMP DROST,I     YES, RETURN
2823  #4452 #21416        LDA MODE        FIND OUT WHERE ERROR OCCURRED
2824  #4453 #10132        CPA M1          PHASE2
2825  #4454 #66460        JMP ERET-2      YES
2826  #4455 #70210        SZA ERET-1      SKIP IF PROGRAM EXECUTION
2827  #4456 #70212        SAM ERET        SKIP IF SYNTAX PHASE
2828  #4457 164477        JMP CALXA,I     MUST HAVE BEEN CALC EXECUTION
2829  #4460 #60645        JSM MINIT       SCRATCH INCOMPLETE SYMTAB
2830  #4461 #60653        JSM INITS+2     SCRATCH STACKS AND RESET PROGRAM POINTERS
2831  #4462 #45701 ERET   ISZ NDISP       SET TO NOT DISPLAY AUTO #
2832  #4463 #66047        JMP PEXM2
2833  #4464 #11154 EMSGA  DEF ERRM+ERRM      POINTER TO "ERROR"
2834  #4465 #11162 LMSGA  DEF ERRM+ERRM+6    POINTER TO " IN LINE "
2835  #4466 #42522 ERRM   ASC 8,ERROR  IN LINE
      #4467 #51117
      #4470 #51040
      #4471 #20111
      #4472 #47040
      #4473 #46111
      #4474 #47105
      #4475 #20040
2836  #4476 #00267 ERR5   DEF E1-1
2837  #4360        MERR   EQU ERROR       PHASE ERROR
2838  #4477 #10070 CALXA  DEF CALEX
```

#01 9830--UPDATE PROGRAM

```
2840*  DELETE STATEMENT FROM PROGRAM
2841*
2842  #4500 160242 DELST  JSM DCOMA,I    DECOMPILE
2843  #4501 #61072        JSM FNDPS-3    FIND STATEMENT TO BE DELETED
2844  #4502 #66506        JMP EDITR      DOESNT EXIST
2845  #4503 #66506        JMP EDITR
2846  #4504 #70110        SZA EDITR      SKIP IF TYPING AID KEY
2847  #4505 #62571        JSM CLPRG      CLOSE UP PROGRAM
2848  #4506 #60651 EDITR  JSM INITS       INITIALIZE STACK POINTERS
2849  #4507 #62723        JSM INCLN      INCREMENT LINE NUMBER
2850  #4510 #66042        JMP PEXMK-2
2851*
2852*  ACCEPT STATEMENT
2853*
2854  #4511 #20112 ACTST  LDA RSTKA
2855  #4512 #31777        STA RSPTR      RESET RETURN STACK
2856  #4513 #21416        LDA MODE
2857  #4514 #70073        SAP *+1,C      SWITCH OUT OF
2858  #4515 #31416        STA MODE         SYNTAX MODE
```

```
#01  9830--UPDATE PROGRAM


2859  #4516 #70110        SZA *+2
2860  #4517 164247        JMP ECALA,I    JUMP TO CALC EXECUTION
2861  #4520 160242 ACTKY  JSM DCOMA,I    DECOMPILE
2862  #4521 #20233        LDA SBUFA
2863  #4522 #70076        TCA            COMPUTE LENGTH
2864  #4523 #01706        ADA SBPTR        OF STATEMENT
2865  #4524 #31473        STA TEMP       SAVE LENGTH IN TEMP
2866  #4525 #25551        LDB RBUFF      READ LINE #
2867  #4526 #74450        SZB ACCS1      SKIP IF TYPING AID DEFINITION
2868  #4527 #41552        IOR RBUFF+1    MERGE INTO OP-CODE WORD
2869  #4530 #31552        STA RBUFF+1
2870  #4531 #61072        JSM FNOPS-3    FIND ADDRESS IN PROGRAM
2871  #4532 #66565        JMP ACCS4      APPEND
2872  #4533 #66565        JMP ACCS4      INSERT
2873  #4534 #70150        SZA ACCS1      SKIP IF REPLACING TYPING AID KEY
2874  #4535 #61007        JSM NEXTA      NO. COMPUTE LENGTH AND END OF STATEMENT
2875  #4536 #66544        JMP ACCS3
2876  #4537 #21424 ACCS1  LDA PBUFF      SET POINTERS TO
2877  #4540 #31476        STA TEMP3        REPLACE
2878  #4541 #70076        TCA
2879  #4542 #01423        ADA PBPTR          ENTIRE PROGRAM
2880  #4543 #25423        LDB PBPTR
2881  #4544 #70076 ACCS3  TCA            COMPARE
2882  #4545 #01473        ADA TEMP           LENGTHS
2883  #4546 #70250        SZA ACCS2      EQUAL
2884  #4547 #70112        SAM *+2
2885  #4550 #66566        JMP ACCS4+1    OLD LINE IS SHORTER
2886  #4551 #21473        LDA TEMP       LONGER
2887  #4552 #62573        JSM CLPRG+2      CLOSE UP PROGRAM
2888*   MOVE FROM S-BUFFER INTO PROGRAM
2889  #4553 #24233 ACCS2  LDB SBUFA      INITIALIZE POINTER
2890  #4554 #74517        LDA 0,I
2891  #4555 131476        STA TEMP3,I      WORD
2892  #4556 #45476        ISZ TEMP3
2893  #4557 #55473        DSZ TEMP       FINISHED?
2894  #4560 #77630        RIB ACCS2+1    NO
2895  #4561 121424        LDA PBUFF,I    YES
2896  #4562 #72110        RZA *+2        0 LINE NUMBER?
2897  #4563 #61225        JSM LDEF       YES. END KEY DEFINITION
2898  #4564 #66506        JMP EDITR      NO
2899*    INSERT STATEMENT
2900  #4565 #21473 ACCS4  LDA TEMP
2901  #4566 #62625        JSM OVCHK      CHECK FOR OVERFLOW
2902  #4567 #61121        JSM MVTOH      MOVE PROGRAM UP
2903  #4570 #66553        JMP ACCS2
2904*
2905*   CLOSE UP PORGRAM
2906*     ADDRESS OF LINE BEING CHANGED IS IN TEMP3
2907*
2908  #4571 #61007 CLPRG  JSM NEXTA      GET ADDRESS OF FIRST WORD MOVED
2909  #4572 #70742        SAR 16
2910  #4573 #01476        ADA TEMP3      SKIP (A) LOCATIONS FORM TEMP3
2911  #4574 #31477        STA TEMP4        AND SAVE DESTINATION ADDRESS
2912*  TEMP4= DESTINATION ADDR OF FIRST WORD
2913*  PBPTO= SOURCE ADDRESS OF LAST WORD +1
2914*   (B)= SOURCE ADDRESS OF FIRST WORD
2915  #4575 #15426 CLPR1  CPB PBPTO      ALL OF PROGRAM MOVED?
2916  #4576 #66603        JMP CLPR2      YES
2917  #4577 #74517        LDA 0,I        MOVE WORD FROM SOURCE
2918  #4600 131477        STA TEMP4,I      TO DESTINATION ADDRESS
2919  #4601 #45477        ISZ TEMP4      INC DESTINATION AND
2920  #4602 #77570        RIB CLPR1        SOURCE ADDRESSES THEN JUMP
2921  #4603 #21477 CLPR2  LDA TEMP4      RECALL NEW PBPTO
2922  #4604 #74076        TCB            COMPUTE NET CHANGE
2923  #4605 #70037        ADB A            IN PROGRAM LENGTH
2924  #4606 #31426        STA PBPTO      STORE NEW END OF MAINLINE
2925  #4607 #21423        LDA PBPTR
2926  #4610 #74017        ADA B
2927  #4611 #31423        STA PBPTR      UPDATE END OF CURRENT PROGRAM
2928  #4612 #21422        LDA UKPTR      GET KEY DEFINITION FLAG
2929  #4613 #70450        SZA OVCHK-1    RETURN IF NOT IN KEY DEFINITION
2930  #4614 #70517        LDA A,I
2931  #4615 #74017        ADA B
2932  #4616 131422        STA UKPTR,I    UPDATE KEY LENGTH
2933  #4617 #21425        LDA PBUFO
2934  #4620 #74017        ADA B
2935  #4621 #31425        STA PBUFO      UPDATE MAINLINE PBUFF
2936  #4622 #05405        ADB FWUP
2937  #4623 #35405        STB FWUP       UPDATE END OF KEY AREA POINTER
2938  #4624 170402        RET
2939*
2940*   CHECK FOR PROGRAM SPACE OVERFLOW
2941*     (A)= NEW WORD REQUIREMENT
2942  #4625 #25426 OVCHK  LDB PBPTO      SET SOURCE ADDRESS
2943  #4626 #35475        STB TEMP2        FRO PROGRAM RELOCATION
2944  #4627 #31711        STA OFLAG      SAVE COUNT
2945  #4630 #70037        ADB A          SET DESTINATION
2946  #4631 #35477        STB TEMP4        ADDRESS
2947  #4632 #01426        ADA PBPTO      COMPUTE NEW END OF MAINLINE
2948  #4633 #74076        TCB            ENOUGH SPACE FOR
```

#01 9830--UPDATE PROGRAM

```
2949  04634 005407        ADB SYMTP       PROGRAM
2950  04635 074113        SBP *+2         BUFFER: SKIP IF YES
2951  04636 064561        JMP FSCF4       NO. SYNTAX OVERFLOW
2952  04637 025711        LDB DFLAG       RECALL COUNT
2953  04640 066606        JMP CLPR2+3
```

#01 9830--SYSTEM COMMANDS

```
2955*
2956*  SCRATCH SYSTEM COMMAND
2957*
2958  04641 061055 SCRAT JSM GNEXT      GET NEXT CHAR
2959  04642 000154        ADA M64
2960  04643 010012        CPA .1          "A"?
2961  04644 066020        JMP ENTRY       YES. SCRATCH ALL
2962  04645 010027        CPA .11         "K"?
2963  04646 066652        JMP SCRK        YES. SCRATCH KEYS
2964  04647 010035        CPA .22         "V"?
2965  04650 066035        JMP KILLV       YES. SCRATCH VARIABLES
2966  04651 066031        JMP KILLP       OTHERWISE SCRATCH PROGRAM
2967  04652 061225 SCRK   JSM LDEF       INSURE NOT IN KEY MODE
2968  04653 063711        JSM DCOMP
2969  04654 020166        LDA FWAM
2970  04655 025405        LDB FWUP
2971  04656 164254        JMP XSCRA,I     SCRATCH ALL KEYS
2972*
2973*  DELETE SYSTEM COMMAND
2974*
2975  04657 060730 DELET JSM GETAD      GET ADDRESSES
2976  04660 035476        STB TEMP3       SAVE SOURCE ADDRESS
2977  04661 160242        JSM DCOMA,I     DECOMPILE
2978  04662 025476        LDB TEMP3
2979  04663 062575        JSM CLPR1       CLOSE UP PROGRAM
2980  04664 066036        JMP RDYPT-1     RE INITIALIZE
2981*
2982*  RENUMBER SYSTEM COMMAND
2983*
2984  04665 162733 RENUM JSM GETLA,I     GET OPTIONAL PARAMETERS
2985  04666 020061        LDA .10
2986  04667 024061        LDB .10
2987  04670 031470        STA L2T1        SAVE STARTING LINE NUMBER
2988  04671 035471        STB L2T2        SAVE INCREMENT
2989  04672 063675        JSM COMPL       CONVERT OT ADDRESS FORMAT
2990  04673 025471        LDB L2T2
2991  04674 035430        STB LNINC       SET LINE INCREMENT
2992  04675 024133        LDB M2          SET FLAG
2993  04676 035471        STB L2T2          FOR 1ST PASS
2994  04677 021470 RENM1  LDA L2T1        SET STARTING
2995  04700 031427        STA .LNUM         LINE NUMBER
2996  04701 025424        LDB PBUFF       INITIALIZE PROGRAM
2997  04702 035472 RENM2  STB TEMPS         POINTER
2998  04703 015423        CPB PBPTR       END OF PROGRAM
2999  04704 066720        JMP RENM4       YES
3000  04705 121472        LDA TEMPS,I
3001  04706 070610        SZA RENM4+2     SKIP IF TYPING AID
3002  04707 050062        AND FLGRT       GET SECURITY BIT
3003  04710 041427        IOR .LNUM       MERGE IN NEW LINE #.
3004  04711 024132        LDB M1
3005  04712 015471        CPB L2T2        SECOND PASS?
3006  04713 131472        STA TEMPS,I     YES. STORE NEW LINE #.
3007  04714 025472 RENM3  LDB TEMPS       COMPUTE ADDRESS
3008  04715 061007        JSM NEXTA         OF NEXT LINE
3009  04716 062723        JSM INCLN       NO. INCREMENT .LNUM
3010  04717 066702        JMP RENM2
3011  04720 045471 RENM4  ISZ L2T2        FIRST PASS?
3012  04721 066677        JMP RENM1       YES. NOW DO SECOND PASS
3013  04722 066036        JMP RDYPT-1     DECOMPILE AND RETURN
3014*
3015*  INCREMENT LINE NUMBER
3016*
3017  04723 021430 INCLN LDA LNINC
3018  04724 001427        ADA .LNUM
3019  04725 031427        STA .LNUM       UPDATE LINE NUMBER
3020  04726 000161        ADA MAXSN       GREATER THAN 9999?
3021  04727 070452        SAM INCLR       RETURN IF LESS THAN 9999
3022  04730 070742        SAR 16
3023  04731 031430        STA LNINC       SET TO MANUAL
3024  04732 064726        JMP SYE25       JUMP TO IMPROPER INTEGER ERROR
3025  04733 003647 GETLA  DEF GETLN
3026*
3027*  INITIALIZE PROGRAM POINTERS
3028*
3029  04734 025405 SCRP   LDB FWUP
3030  04735 035424        STB PBUFF
3031  04736 035423        STB PBPTR
3032  04737 035426        STB PBPTO
3033  04740 170402 INCLR  RET
```

#01 9830- PRE-EXECUTION PROCESSING

```
3035*
3036*  RUN SYSTEM COMMAND
3037*    (A)= STARTING LINE NUMBER
3038*
3039  04741 031725 RUN   STA NXTST      SAVE STARTING LINE NUMBER
3040  04742 060645       JSM MINIT      SCRATCH OLD SYMTAB
3041  04743 070742       SAR 16
3042  04744 031455       STA FORMT      CLEAR THE FORMAT FLAG
3043  04745 031432       STA DRGST
3044  04746 020012       LDA .1
3045  04747 031434       STA UNITS      SET TO RADIANS
3046  04750 021424       LDA PBUFF
3047  04751 011423       CPA PBPTR      NULL PROGRAM?
3048  04752 164174       JMP RDYDA,I    YES
3049  04753 125424       LDB PBUFF,I
3050  04754 075710       SZA *-2        TYPING AID KEY?
3051  04755 031421       STA PBADD      INITIALIZE PROGRAM POINTER
3052  04756 021405       LDA FWUP
3053  04757 031730       STA COML       INITIALIZE COMMON POINTER
3054  04760 045412       ISZ LSTPT      SET L-STACK POINTER = PBPTR
3055  04761 020132       LDA M1         FLAG ERROR ROUTINE
3056  04762 031416       STA MODE         TO SCRATCH SYMTAB
3057*
3058*  SKIP TO BEGINNING OF NEXT STATEMENT
3059*
3060  04763 025421 MSKIP LDB PBADD
3061  04764 021407       LDA SYMTP      GET POINTER FOR MILOP
3062  04765 015423       CPB PBPTR      END OF PROGRAM?
3063  04766 067122       JMP MILOP      YES
3064  04767 063601       JSM NXTOP      FETCH 1ST OPERAND IN NEXT STATEMENT
3065  04770 066763       JMP MSKIP      END OF STATEMENT
3066  04771 067106       JMP MLOP2      INTEGER, SKIP IT
3067  04772 031473 MSTND STA MBOX1      VARIABLE OPERAND, SAVE IT
3068  04773 024132       LDB M1
3069  04774 035474       STB MBOX1+1    INITIALIZE DIMENSIONS TO UNDEFINED
3070  04775 063234       JSM FNTYP      FIND OPERAND TYPE
3071  04776 067106       JMP MLOP2      FUNCTION
3072  04777 020132       LDA M1         STRING
3073  05000 025731 MLOP7 LDB TYPE
3074  05001 014020       CPB MDMA       DIM?
3075  05002 074742 MLP72 SBR 16         YES, SET (B)=0
3076  05003 014017       CPB MCOMA      COM?
3077  05004 067112       JMP MLP71
3078  05005 070153       SAP *+3        SKIP IF SIMPLE VARIABLE
3079  05006 076150       RZB MLOP5      ARRAY, SKIP IF NOT DIM OR COM
3080  05007 067017       JMP MLOP8
3081  05010 076310       RZB ML2L       SKIP IF NOT DIM OR COM
3082*
3083*  PROCESS VARIABLE
3084  05011 063306 MLOP5 JSM SSYMT+1    SEARCH SYMBOL TABLE
3085  05012 067043       JMP MLOP9      NOT FOUND
3086  05013 021731       LDA TYPE       FOUND
3087  05014 010017       CPA MCOMA      COM STMT?
3088  05015 067037       JMP MERS       YES, DOUBLE DEFINITION
3089  05016 067106 ML2L  JMP MLOP2      NO, PROCESS NEXT WORD
3090*  COM OR DIM
3091*  (A)=-1 FOR 1 DIM OR STRING, -2 FOR 2 DIM
3092  05017 045472 MLOP8 ISZ TEMPS
3093  05020 045472       ISZ TEMPS
3094  05021 125472       LDB TEMPS,I    READ FIRST DIMENSION
3095  05022 004132       ADB M1         OFFSET BY 1
3096  05023 074404       SBL 8          PLACE IT IN LEFT HALF WORD
3097  05024 010132       CPA M1         2 DIMENSIONS
3098  05025 067032       JMP *+5        NO, SET SECOND DIM TO 1
3099  05026 045472       ISZ TEMPS
3100  05027 045472       ISZ TEMPS
3101  05030 105472       ADB TEMPS,I    ADD SECOND DIMENSION INTO RIGHT HALF
3102  05031 004132       ADB M1         OFFSET SECOND DIM
3103  05032 035474       STB MBOX1+1    STORE PACKED DIMENSIONS
3104  05033 063306       JSM SSYMT+1    SEARCH SYMBOL TABLE
3105  05034 067043       JMP MLOP9      NOT FOUND
3106  05035 004132       ADB M1         FOUND, (B)=ADDRESS
3107  05036 074637       ISZ B,I        ALREADY DIMENSIONED?
3108  05037 062360 MERS  JSM MERR       YES
3109  05040 021474       LDA MBOX1+1    STORE DIMENSIONS
3110  05041 074557       STA B,I
3111  05042 067106       JMP MLOP2
3112*  ENTER NEW SYMBOL INTO TABLE
3113  05043 063404 MLOP9 JSM ESYMT      ENTER INTO SYMBOL TABLE
3114  05044 021731       LDA TYPE
3115  05045 010017       CPA MCOMA      COM STMT?
3116  05046 067050       JMP MCOM1      YES
3117  05047 067106       JMP MLOP2      NO
3118*  ALLOCATE SPACE FOR COMMON VARIABLE
3119  05050 021473 MCOM1 LDA MBOX1      SET SIGN BIT IN
3120  05051 072053       SAP **1,S        SYMBOL TABLE TO
3121  05052 074557       STA B,I             INDICATE COMMON
3122  05053 025474       LDB MBOX1+1    RECALL DIMENSIONS
3123  05054 063257       JSM MDIM       COMPUTE NUMBER OF WORDS NEEDED
3124  05055 025730       LDB COML       GET OLD COMMON POINTER
```

#01 9830- PRE-EXECUTION PROCESSING

```
3125  #5056 #74017        ADA B
3126  #5057 #31730        STA COML       UPDATE COMMON POINTER
3127  #5060 #21424        LDA PHUFF
3128  #5061 #45422        ISZ UKPTR
3129  #5062 #55422        DSZ UKPTR      KEY MODE?
3130  #5063 #21425        LDA PHUF0      YES, GET FIRST WORD OF MAINLINE
3131  #5064 #70056        CMA            MAXIMUM COMMON EXCEEDED?
3132  #5065 #31730        ADA COML
3133  #5066 #70112        SAM *+2
3134  #5067 #62360 CMERR  JSM MERR       YES, TOO MUCH COMMON IN KEY
3135  #5070 #21474        LDA MBOX1+1    READ DIMENSIONS
3136  #5071 #10132        CPA M1         SIMPLE VARIABLE?
3137  #5072 #67103        JMP MCOM2      YES
3138  #5073 #74557        STA B,I        STORE FORMAL DIMENSIONS
3139  #5074 #74070        SIB *+1
3140  #5075 #21473        LDA MBOX1
3141  #5076 #70206        RAR 5          IF BIT 4=1, IT MUST
3142  #5077 #70212        SAM MCOM2        BE A STRING VARIABLE
3143  #5100 #21474        LDA MBOX1+1    NO, MUST BE AN ARRAY
3144  #5101 #74557        STA B,I        STORE ACTUAL DIMENSIONS
3145  #5102 #74070        SIB *+1        POINT TO FIRST VARIABLE OF ARRAY
3146  #5103 #21407 MCOM2  LDA SYMTP
3147  #5104 #70070        SIA *+1
3148  #5105 #70577        STB A,I        STORE ADDRESS IN SYMTAB
3149  #5106 #61620 MLOP2  JSM NXTP1      GET NEXT OPERAND OF STMT
3150  #5107 #66763        JMP MSKIP      END OF STMT
3151  #5110 #67106        JMP MLOP2      INTEGER, SKIP IT
3152  #5111 #66772        JMP MSTNO      VARIABLE OPERAND
3153*  CHECK FOR LEGAL COM
3154  #5112 125424 MLP71  LDB PHUFF,I
3155  #5113 #74073        SBP *+1,C
3156  #5114 #15427        CPB .LNUM      IS THIS THE FIRST LINE?
3157  #5115 #67002        JMP MLP72      YES
3158  #5116 164330        JMP E30+3,I    NO, MISPLACED COM
3159*
3160*  MEASE PART TWO
3161*
3162*  MOVE SYMTAB TO LOW MEMORY
3163  #5117 #70537 M5LOP  LDB A,I        MOVE SYMBOL
3164  #5120 135412        STB LSTPT,I      TABLE WORD
3165  #5121 #45412        ISZ LSTPT      INC DESTINATION POINTER
3166  #5122 #11406 M1LOP  CPA VALUE      ALL OF TABLE MOVED?
3167  #5123 #67125        JMP *+2        YES
3168  #5124 #73570        RIA M5LOP      INC SOURCE POINTER
3169  #5125 #70742        SAR 16
3170  #5126 #31427        STA .LNUM      CLEAR LINE #
3171  #5127 #25412        LDB LSTPT
3172  #5130 #20062        LDA FLGAT      SET ALL PROSPECTIVE
3173  #5131 #74557        STA B,I
3174  #5132 #15406        CPB VALUE        VALUE TABLE WORDS
3175  #5133 #67135        JMP *+2
3176  #5134 #77670        RIB *-3            TO UNDEFINED VALUE
3177*  DURING THIS SECTION USE LSTAK AS SYMTAB POINTER
3178*    AND HSTPT AS VALUE TABLE POINTER
3179  #5135 #35413        STB HSTPT      INITIALIZE VALUE TABLE POINTER
3180  #5136 #21412        LDA LSTPT
3181  #5137 #31410        STA LSTAK      INITIALIZE SYMBOL TABLE POINTER
3182*
3183  #5140 #25410 M2LOP  LDB LSTAK
3184  #5141 #15426        CPB PBPTO      MORE SYMBOLS?
3185  #5142 #67216        JMP M4LOP      NO
3186  #5143 #55410        DSZ LSTAK      YES, MOVE TO NEXT SYMBOL
3187  #5144 121410        LDA LSTAK,I    READ SYMBOL
3188  #5145 #55410        DSZ LSTAK
3189  #5146 #70233        SAP **4,C      DEFINED IN COM?
3190  #5147 #04132        ADB M1
3191  #5150 #74557        STA B,I        YES, CLEAR FLAG
3192  #5151 #67140        JMP M2LOP      BYPASS ALLOCATION PHASE
3193  #5152 #31473        STA MBOX1      NO, SAVE SYMBOL
3194  #5153 #63234        JSM FNTYP      FIND VARIABLE TYPE
3195  #5154 #67140        JMP M2LOP      FUNCTION
3196  #5155 #67160        JMP *+3        STRING
3197  #5156 #10134        CPA M3         ARRAY OF UNKNOWN DIMENSIONS?
3198  #5157 #62360 MER10  JSM MERR       YES
3199  #5160 #31474        STA MBOX1+1    SAVE TYPE FLAG
3200  #5161 125410        LDB LSTAK,I    READ DIMENSIONS
3201  #5162 #14132        CPB M1         DIMENSIONED?
3202  #5163 #67165        JMP **2        NO
3203  #5164 #67173        JMP M3LOP      YES
3204  #5165 #74742        SBR 16         LOAD STD DIM FOR STRING
3205  #5166 #70213        SAP M3LOP-1    SKIP IF SIMPLE VARIABLE OR STRING
3206  #5167 #27233        LDB B4400      LOAD STD DIMENSION FOR 1 DIM
3207  #5170 #10133        CPA M2         2 DIMENSIONS?
3208  #5171 #04026        ADB .9         YES, CORRECT FOR 2 DIMS
3209  #5172 135410        STB LSTAK,I
3210  #5173 #21473 M3LOP  LDA MBOX1      RECALL SYMBOL
3211  #5174 #63257        JSM MDIM       COMPUTE STORAGE REQUIRED
3212  #5175 #70076        TCA
3213  #5176 #25413        LDB HSTPT      MOVE VALUE TABLE POINTER
3214  #5177 #63556        JSM MOVCK+1      AND CHECK FOR OVERFLOW
```

#01 9830- PRE-EXECUTION PROCESSING

```
3215  05200 021474          LDA MBOX1+1   RECALL TYPE FLAG
3216  05201 025413          LDB HSTPT     GET NEW VALUE TABLE POINTER
3217  05202 074070          SIB *+1
3218  05203 010076          CPA B36       STRING?
3219  05204 067211          JMP M6LOP     YES
3220  05205 070353          SAP M6LOP+3   SKIP IF SIMPLE VARIABLE
3221  05206 121410          LDA LSTAK,I   GET DIMENSIONS AND STORE
3222  05207 074557          STA B,I         IN VALUE TABLE
3223  05210 074130          SIB *+2       INC POINTER
3224  05211 121410 M6LOP    LDA LSTAK,I   GET STRING DIMENSIONS
3225  05212 074557          STA B,I       STORE DIMENSION
3226  05213 074070          SIB *+1         IN VALUE TABLE
3227  05214 135410          STB LSTAK,I   STORE VALUE ADDRESS IN SYMTAB
3228  05215 067140          JMP M2LOP
3229*  CLEAN UP FOR EXECUTION
3230  05216 021413 M4LOP    LDA HSTPT
3231  05217 031406          STA VALUE     SET VALUE TABLE POINTER
3232  05220 025412          LDB LSTPT     MOVE SYMBOL
3233  05221 015426          CPB PRPTO
3234  05222 067230          JMP *+6
3235  05223 004132          ADB M1
3236  05224 074517          LDA B,I         TABLE BACK
3237  05225 131413          STA HSTPT,I
3238  05226 055413          DSZ HSTPT         TO HIGH
3239  05227 067221          JMP M4LOP+3         MEMORY
3240  05230 021413          LDA HSTPT
3241  05231 031407          STA SYMTP     SET SYMBOL TABLE POINTER
3242  05232 164004          JMP XECA,I    PHASE COMPLETE
3243*
3244  05233 004400 B4400 OCT 4400
3245*  DETERMINE THE TYPE OF AN OPERAND GIVEN IN (A)
3246*
3247*    RETURN P+1: FUNCTION
3248*           P+2: STRING
3249*           P+3: NUMERIC VARIABLE
3250*                (A)= POSITIVE IF SIMPLE VARIABLE
3251*                   = -1 IF SINGLE DIM ARRAY
3252*                   = -2 IF DOUBLE DIM
3253*                   = -3 IF UNKNOWN DIM
3254*
3255*                (B)= 4 IF FULL PRECISION
3256*                   = 2 IF SPLIT PRECISION
3257*                   = 1 IF INTEGER
3258*
3259  05234 050040 FNTYP    AND B37       ISOLATE TYPE
3260  05235 010076          CPA B36       STRING?
3261  05236 064357          JMP RETN2     YES
3262  05237 074742          SBR 16
3263  05240 010040          CPA B37       FUNCTION?
3264  05241 170402          RET           YES
3265  05242 024021          LDB .4        SET TO FULL PRECISION
3266  05243 070146          RAR 4
3267  05244 070171          SLA *+3,C     NUMBERED VARIABLE
3268  05245 070546          RAR 12        YES
3269  05246 064356          JMP RETN3
3270  05247 070133          SAP *+2,C     INTEGER?
3271  05250 004134          ADB M3        YES
3272  05251 070646          RAR 14
3273  05252 070131          SLA *+2,C     SPLIT PRECISION?
3274  05253 004133          ADB M2        YES
3275  05254 070646          RAR 14        RECALL NUMBER OF DIMENSIONS
3276  05255 070076          TCA           NEGATE
3277  05256 064356          JMP RETN3
3278*
3279*  COMPUTE STORAGE NEEDED BY A VARIABLE
3280*
3281*    (A)= SYMBOL
3282*    (B)= PACKED DIMENSIONS FOR ARRAY OR STRING
3283*
3284*    STORAGE NEEDED: STRING = (DIM1+1)/2 +1
3285*                    ARRAY= DIM1*DIM2* # OF WORDS/ELEM +2
3286*                    SIMPLE VARIABLE: # OF WORDS
3287*
3288*    EXIT (A)= STORAGE NEEDED
3289  05257 035470 MDIM     STB L2T1
3290  05260 063234          JSM FNTYP
3291  05261 067277          JMP MDIM2     FUNCTION (B)=0
3292  05262 067301          JMP MDIM1     STRING
3293  05263 070613          SAP MDIM2     ARRAY?
3294  05264 021470          LDA L2T1
3295  05265 050105          AND B377      MULTIPLY SECOND DIMENSION
3296  05266 070070          STA *+1
3297  05267 061201          JSM IMPY      BY THE NUMBER OF WORDS PER ELEMENT
3298  05270 025470          LDB L2T1
3299  05271 074342          SBR B
3300  05272 074070          SIB *+1
3301  05273 061201          JSM IMPY      MULTIPLY BY SECOND DIMENSION
3302  05274 000017          ADA .2        ADD WORDS FOR DIMENSIONS
3303  05275 070153          SAP MDIM2+1   GREATER THAN 32K?
3304  05276 062360 MER9     JSM MERR      YES, DIMENSIONS TOO LARGE
```

#01 9830- PRE-EXECUTION PROCESSING

```
3305  05277 074117 MDIM2 LDA B          MOVE RESULT TO (A)
3306  05300 170402       RET
3307  05301 074342 MDIM1 SBR B
3308  05302 004017       ADB .2         ADD 2 TO FORMAL DIMENSION
3309  05303 074002       SBR 1          DIVIDE BY 2
3310  05304 077570       MIB MDIM2      INCREMENT AND JUMP
3312*
3313*  SEARCH SYMBOL TABLE
3314*
3315*    RETURN P+1 IF NOT FOUND
3316*           P+2 IF FOUND; ADDRESS IN B
3317*
3318*    IF BIT 4=1 THEN SYMBOL AND TYPE MUST MATCH EXACTLY
3319*
3320*    AN UNSPECIFIED PRECISION VARIABLE MATCHES A SPECIFIED PRECISION
3321*      AND THE SPECIFIED PRECISION IS INSERTED IN THE TABLE
3322*
3323*    AN ARRAY WITH UNSPECIFIED NUMBER OF DIMENSIONS MATCHES A
3324*       SPECIFIED ARRAY AND THE DIMENSION NUMBER IS PUT IN THE TABLE
3325*
3326*    AN INCONSISTENCY IN DIMENSIONS OR PRECISION CAUSES AN ERROR
3327*
3328  05305 031473 SSYMT STA MBOX1
3329  05306 021473       LDA MBOX1      (A)=SYMBOL TO BE FOUND
3330  05307 050020       AND .3
3331  05310 031466       STA STMP2      SAVE SIMPLE/ARRAY TYPE
3332  05311 021473       LDA MBOX1
3333  05312 050030       AND .12
3334  05313 031467       STA STMP3      SAVE PRECISION TYPE
3335  05314 021473       LDA MBOX1      RECALL VARIABLE
3336  05315 024112       LDB B1760      GET MASK TO MASK OFF PRECISION
3337  05316 070146       RAR 4
3338  05317 070111       SLA **2        BIT 4=1?
3339  05320 024115       LDB B1777      YES, DONT MASK OFF ANY BITS
3340  05321 070546       RAR 14         RESTORE VARIABLE
3341  05322 074257       AND B          MASK
3342  05323 035470       STB MASK       STORE MASK
3343  05324 031471       STA STMP1      STORE PART OF SYMBOL TO MATCH
3344  05325 014115       CPB B1777      MUST MATCH WHOLE SYMBOL?
3345  05326 067373       JMP SVALU-1    YES, SEARCH AND RETURN FROM SVALU
3346  05327 063373       JSM SVALU-1    NO, SEARCH FOR NAME ONLY
3347  05330 170402       RET            NOT FOUND
3348  05331 021466 SSYM1 LDA STMP2      SEARCHING FOR
3349  05332 072350       RZA SSYM4      SIMPLE VARIABLE?
3350  05333 074517       LDA B,I        YES, SIMPLE
3351  05334 050020       AND .3           VARIABLE
3352  05335 070410       SZA SM2L       FOUND?
3353  05336 063402 SSYM6 JSM SVAL1      NO, CONTINUE SEARCH
3354  05337 170402       RET
3355  05340 067331       JMP SSYM1
3356*  SEARCHING FOR AN ARRAY
3357*
3358  05341 074517 SSYM4 LDA B,I
3359  05342 050020       AND .3
3360  05343 071550       SZA SSYM6      ARRAY FOUND?  SKIP IF NO
3361  05344 011466       CPA STMP2      DO DIMENSIONS MATCH?
3362  05345 067361 SM2L  JMP SSYM2      YES, NOW CHECK PRECISION
3363  05346 010020       CPA .3         NO, TABLE VARIABLE UNSPECIFIED?
3364  05347 067354       JMP SSYM5      YES
3365  05350 021466       LDA STMP2      NO, SEARCH
3366  05351 010020       CPA .3           VARIABLE UNSPECIFIED?
3367  05352 067361       JMP SSYM2      YES, OK
3368  05353 062360 SYMF2 JSM MERR       NO, INCONSISTENT DIMENSIONS
3369  05354 074517 SSYM5 LDA B,I        INSERT
3370  05355 050135       AND M4
3371  05356 041466       IOR STMP2        DIMENSION NUMBER
3372  05357 074557       STA B,I
3373  05360 067361       JMP SSYM2          INTO TABLE
3374*  PROPER TYPE FOUND, NOW CHECK FOR PROPER PRECISION
3375*
3376  05361 021467 SSYM2 LDA STMP3
3377  05362 070350       SZA SSYM3-1    SEARCH VARIABLE UNSPECIFIED?
3378  05363 074517       LDA B,I        NO
3379  05364 050030       AND .12
3380  05365 072250       RZA SSYM3      TABLE VARIABLE UNSPECIFIED?
3381  05366 021467       LDA STMP3      YES
3382  05367 074617       IOR B,I
3383  05370 074557       STA B,I        INSERT SPECIFICATION IN TABLE
3384  05371 064357       JMP RETN2      FOUND
3385  05372 062360 SSYM3 JSM MERR       PRECISION DOUBLY DEFINED
3386*
3387*  SEARCH SYMBOL TABLE FOR A MATCH - (B),I AND MASK= STMP1
3388*
3389*    RET P+1 IF NO MATCH FOUND
3390*        P+2 IF MATCH FOUND, (B)= ADDRESS
3391  05373 025406       LDB VALUF
3392  05374 015407 SVALU CPB SYMTP      ALL TABLE SEARCHED?
3393  05375 170402       RET            YES, NO MATCH FOUND
3394  05376 074517       LDA B,I        NO, READ TABLE ENTRY
3395  05377 051470       AND MASK       GET THE BITS TO BE COMPARED
```

```
3396  05400 011471        CPA STMP1      MATCH?
3397  05401 064357        JMP RETN2      YES, RET P+2, (B)= ADDRESS
3398  05402 004133 SVAL1  ADB M2         UPDATE TABLE POINTER
3399  05403 067374        JMP SVALU
3400  01470         MASK  EQU L2T1
3401  01471         STMP1 EQU L2T2
3402  01466         STMP2 EQU LOCL1
3403  01467         STMP3 EQU LOCL2
3404*
3405*  ENTER 2 WORDS GIVEN IN MHOX1 AND MHOX1+1 INTO SYMBOL TABLE
3406*
3407  05404 020133 ESYMT LDA M2          MOVE H-STACK
3408  05405 025407        LDB SYMTP
3409  05406 063555        JSM MOVCK        DOWN BY 2 WORDS
3410  05407 074070        SIB *+1        ((B)= NEW SYMTP)
3411  05410 021474        LDA MHOX1+1    TRANSFER WORDS INTO SYMTAB
3412  05411 074557 ESYM1  STA B,I
3413  05412 074070        SIB *+1
3414  05413 021473        LDA MHOX1
3415  05414 074557        STA B,I
3416  05415 170402        RET
3417*
3418*  ENTER A SIMPLE VARIABLE INTO SYMTAB AND VALUE TABLE
3419*
3420  05416 021473 ENTER  LDA MHOX1
3421  05417 063234        JSM FNTYP      FIND OPERAND TYPE
3422  05420 067437        JMP ENTE1      FUNCTION
3423  05421 160206 NINIT  JSM ERRA,I     STRING, NOT INITIALIZED
3424  05422 071752        SAM *-1        SIMPLE VARIABLE?
3425  05423 025406        LDB VALUF      YES
3426  05424 020137        LDA M6         NO, MOVE SYMTAB AND H-STACK
3427  05425 063555        JSM MOVCK        DOWN BY 6 WORDS
3428  05426 004135        ADA M4        COMPUTE ADDRESS OF VARIABLE
3429  05427 031474        STA MHOX1+1
3430  05430 070137        LDB A
3431  05431 020062        LDA FLGAT
3432  05432 074557        STA B,I        SET TO UNDEFINED
3433  05433 004133        ADB M2         COMPUTE ADDRESS OF
3434  05434 063410        JSM ESYMT+4      SYMTAB ENTRY AND ENTER
3435  05435 035406        STB VALUF      UPDATE VALUE TABLE POINTER
3436  05436 170402        RET
3437*  SEARCH FOR A FUNCTION NOT FOUND IN SYMTAB
3438*
3439  05437 025406 ENTF1  LDB VALUF      OPEN UP 2 WORD SPACE
3440  05440 020133        LDA M2
3441  05441 063555        JSM MOVCK
3442  05442 025406        LDB VALUF
3443  05443 004132        ADB M1
3444  05444 035750        STB YC         SET UP ADDRESS POINTER
3445  05445 070742        SAR 16         INITIALIZE ADDRESS WORD TO 0
3446  05446 063411        JSM ESYM1      MAKE SYMTAB ENTRY
3447*
3448*  SEARCH FOR FUNCTION WHICH HAS 0 ADDRESS
3449  05447 063513 ENTFN  JSM SSTT       SAVE PROGRAM POINTERS
3450  05450 025424        LDB PBUFF
3451  05451 063524        JSM SERFN      SEARCH CURRENT PROGRAM FIRST
3452  05452 024166        LDB FWAM      NOT FOUND
3453*  SEARCH FIRST LINE OF EACH KEY
3454  05453 015405 ENTF3  CPB FWUP       END OF KEY AREA?
3455  05454 067503        JMP ENTFS      YES, NOT FOUND
3456  05455 074070        SIB *+1
3457  05456 074517        LDA B,I
3458  05457 074070        SIB *+1
3459  05460 010012        CPA .1         NULL KEY?
3460  05461 067453        JMP ENTF3      YES
3461  05462 035472        STB TEMPS
3462  05463 121472        LDA TEMPS,I
3463  05464 004132        ADB M1
3464  05465 070610        SZA ENTF4+2    TYPING AID KEY?
3465  05466 060602        JSM NEXT1-4    NO, GET PAGE AND OP CODE
3466  05467 001466        ADA LOCL1
3467  05470 013554        CPA DEFOP      DEF STMT?
3468  05471 067473        JMP *+2        YES
3469  05472 067477        JMP ENTF4      NO
3470  05473 121472        LDA TEMPS,I
3471  05474 050115        AND B1777
3472  05475 011473        CPA MHOX1      IS IT THE FUNCTION WE WANT?
3473  05476 067536        JMP SERF1      YES
3474  05477 025472 ENTF4  LDB TEMPS      NO
3475  05500 004134        ADB M3
3476  05501 074437        ADB B,I        FIND NEXT KEY
3477  05502 067453        JMP ENTF3
3478*  SEARCH MAINLINE LAST
3479  05503 021422 ENTFS  LDA UKPTR
3480  05504 070250        SZA ENTF6      KEY MODE?
3481  05505 025426        LDB PBPTO
3482  05506 035423        STB PBPTR
3483  05507 025425        LDB PBUFO      YES, SEARCH MAINLINE
3484  05510 063524        JSM SERFN
3485  05511 063543 ENTF6  JSM RSTT      NOT FOUND, UNDEFINED
```

```
        #01  9830- PRE-EXECUTION PROCESSING


3486  #5512 164337        JMP E40,I
3487*
3488*  SUBROUTINE TO SAVE PROGRAM POINTERS
3489*
3490  #5513 #25421 SSTT   LDB PRADD
3491  #5514 #35741        STB XC+1
3492  #5515 #25472        LDB TEMPS
3493  #5516 #35740        STA XC
3494  #5517 #25427        LDA .LNUM
3495  #5520 #35742        STB XC+2
3496  #5521 #25423        LDA PRPTR
3497  #5522 #35743        STB XC+3
3498  #5523 170402 SERFR RET
3499*
3500*  SUBROUTINE TO SEARCH A PROGRAM FOR A FUNCTION DEF
3501*
3502*  (A)=ADDRESS OF FIRST WORD OF PROGRAM
3503*  RETURN IF NOT FOUND
3504*  RETURN TO SECOND RETURN ON STACK IF FOUND; STORE ADDRESS IN XC+3,I
3505*
3506  #5524 #35421 SERFN STB PRADD
3507  #5525 #23554        LDA DEFOP
3508  #5526 160241        JSM SYTSA,I    SEARCH FOR DEF STMT
3509  #5527 #75613        SBP SERFR      FOUND?
3510  #5530 121472        LDA TEMPS,I    YES
3511  #5531 #50115        AND B1777
3512  #5532 #11473        CPA MBOX1      IS IT THE FN WE WANT?
3513  #5533 #67535        JMP *+2        YES
3514  #5534 #67525        JMP SERFN+1
3515  #5535 #55777        ISZ RSPTR      FOUND, BINK RETURN OFF STACK
3516  #5536 #25472 SERF1  LDB TEMPS
3517  #5537 #04133        ADB M2
3518  #5540 135750        STB YC,I       STORE ADDRESS IN SYMTAB
3519  #5541 #25750        LDB YC
3520  #5542 #74070        SIB *+1        GET SYMBOL TABLE ADDRESS
3521*
3522  #5543 #21740 RSTT   LDA XC         RESTORE PROGRAM POINTERS
3523  #5544 #31472        STA TEMPS
3524  #5545 #21741        LDA XC+1
3525  #5546 #31421        STA PRADD
3526  #5547 #21742        LDA XC+2
3527  #5550 #31427        STA .LNUM
3528  #5551 #21743        LDA XC+3
3529  #5552 #31423        STA PRPTR
3530  #5553 170402        RET
3531*
3532  #5554 #11774 DEFOP OCT 11774
3533*
3534*  MOVE H-STACK DOWN AND CHECK FOR OVERFLOW
3535*
3536*     MOVE DOWN -(A) WORDS
3537*     (B)= ADDRESS OF LAST WORD MOVED
3538*
3539  #5555 #74070 MOVCK SIB *+1
3540  #5556 #31726        STA MBIN1      SAVE SPACE REQUIREMENT
3541  #5557 #35466        STB LOCL1      SAVE LAST WORD + 1
3542  #5560 #01413        ADA HSTPT      COMPUTE NEW HSTPT
3543  #5561 #70137        LDB A
3544  #5562 #60555        JSM BHSTP+2    CHECK FOR OVERFLOW
3545  #5563 #70137        LDB A
3546  #5564 #21413        LDA HSTPT      GET OLD HSTPT
3547  #5565 #35413        STB HSTPT      STORE NEW HSTPT
3548  #5566 #35727        STB MBIN2      STORE SOURCE POINTER
3549  #5567 #11466 MOVC1  CPA LOCL1      ALL WORDS MOVED?
3550  #5570 #67575        JMP MOVC2      YES
3551  #5571 #70537        LDB A,I        MOVE
3552  #5572 135727        STB MBIN2,I      WORD
3553  #5573 #45727        ISZ MBIN2
3554  #5574 #73570        BIA MOVC1      UPDATE POINTERS
3555  #5575 #25407 MOVC2  LDB SYMTP
3556  #5576 #05726        ADB MBIN1
3557  #5577 #35407        STB SYMTP      UPDATE SYMBOL TABLE POINTER
3558  #5600 170402 NXTPR RET
3559*  ROUTINE TO FETCH THE NEXT OPERAND FROM THE PROGRAM
3560*     TEMPS POINTS TO CURRENT WORD OF CODE; PRADD POINTS TO END OF LINE
3561*     CONSTANTS; STRING CONSTANTS; NON-RESEQUENCEABLE INTEGERS ARE SKIPPE
3562*     EXIT P+1: END OF LINE
3563*          P+2: RESEQUENCABLE INTEGER FOUND
3564*          P+3: VARIABLE OPERAND FOUND
3565  #5601 #60572 NXTOP JSM NEXTL-2      COMPUTE END OF CURRENT LINE
3566  #5602 #05466        ADB LOCL1      ADD OP-CODE AND
3567  #5603 #07772        AUR MREM       CONSTANT TO PAGE TO GET
3568  #5604 #35731        STB TYPE         UNIQUE VALUE FOR STATEMENT TYPE
3569  #5605 #55472        ISZ TEMPS
3570  #5606 121472        LDA TEMPS,I
3571  #5607 #50107        AND B1000      GET ASCII FLAG
3572  #5610 #73410        SZA NXTPR      ASCII?
3573  #5611 #67620        JMP NXTP1
3574*  PROCESS A STRING CONSTANT
3575  #5612 125472 NXTP2 LDB TEMPS,I   COMPUTE
```

```
3576  @5613 @74070        SIR *+1
3577  @5614 @74404        SBL 8             LENGTH OF
3578  @5615 @74402        SBR 4               STRING
3579  @5616 @05472        ADB TEMPS       SKIP TO
3580  @5617 @35472        STB TEMPS         END OF CONSTANT
3581*  PROCESS NEXT WORD: RE-ENTER ROUTINE HERE AFTER
3582*     PROCESSING A VARIABLE OPERAND
3583  @5620 @45472 NXTP1 ISZ TEMPS       SET TO NEXT WORD
3584  @5621 @21472        LDA TEMPS
3585  @5622 @11421        CPA PRADD       END OF CURRENT STMT?
3586  @5623 170402        RET             YES
3587  @5624 @70517        LDA A,I         READ CURRENT WORD
3588  @5625 @70452        SAM NXTP3       SKIP IF SIGN BIT SET
3589  @5626 @50065        AND OPMSK
3590  @5627 @10116        CPA QTOP        QUOTE OPERATOR?
3591  @5630 @67612        JMP NXTP2       YES. PROCESS STRING CONSTANT
3592  @5631 @10065        CPA OPMSK       MULTI-CHARACTER-NON-FORMULA OPERATOR?
3593  @5632 @67620 NXTIL JMP NXTP1       YES. NO OPERAND IN THIS WORD
3594  @5633 @60771        JSM GTOPP       GET OPERAND
3595  @5634 @71210        SZA NXTP1       SKIP IF NULL
3596  @5635 @64356        JMP RETN3       RETURN TO P+3 WITH OPERAND IN (A)
3597*  PROCESS WORD WITH SIGN BIT SET
3598  @5636 @50106 NXTP3 AND B777        GET TYPE AND CHARACTER COUNT IF CONSTANT
3599  @5637 @70137        LDB A
3600  @5640 @50040        AND B37         ISOLATE THE TYPE
3601  @5641 @10021        CPA .4          RESEQUENCABLE INTEGER?
3602  @5642 @67654        JMP NXTP4       YES
3603  @5643 @71350        SZA NXTIL       SKIP IF PARAMETER
3604  @5644 @00135        ADA M4
3605  @5645 @71253        SAP NXTIL       SKIP IF PARAMETER OR FUNCTION
3606  @5646 @74202        SBR 5           COMPUTE NUMBER OF
3607  @5647 @04022        ADR .5            WORDS TO SKIP IF
3608  @5650 @74042        SBR 2               CONSTANT
3609  @5651 @10132        CPA M1          INTEGER?
3610  @5652 @24012        LDB .1          YES. SKIP 1 WORD
3611  @5653 @67616        JMP NXTP1-2
3612  @5654 @45472 NXTP4 ISZ TEMPS       READ THE
3613  @5655 121472        LDA TEMPS,I       INTEGER
3614  @5656 @64357        JMP RETN2       RETURN TO P+2 WITH INTEGER IN (A)
3615*
3616*  ROUTINE TO FIND ALL RESEQUENCABLE LINE NUMBER REFERENCES IN A PROGRAM
3617*     EXIT: P+1  END OF PROGRAM FOUND
3618  @5657 @67620        JMP NXTP1       NO
3619*           (A)= LINE NUMBER
3620*           (TEMPS)= ADDRESS OF LINE NUMBER
3621*
3622  @5660 @31421 FNDAD STA PRADD        SAVE STARTING ADDRESS
3623  @5661 @70517        LDA A,I       TYPING AID KEY?
3624  @5662 @70150        SZA FNDA2+2     RETURN IF YES
3625  @5663 @21421 FNDA2 LDA PRADD
3626  @5664 @11423        CPA PRPTR       END OF PROGRAM?
3627  @5665 170402        RET             YES
3628  @5666 @63601        JSM NXTOP       NO. CONTINUE SEARCH
3629  @5667 @67663        JMP FNDA2       END OF STATEMENT
3630  @5670 @64357        JMP RETN2       LINE NUMBER FOUND
3631  @5671 @63620 FNDA1 JSM NXTP1       CONTINUE SEARCH
3632  @5672 @67663        JMP FNDA2
3633  @5673 @64357        JMP RETN2
3634  @5674 @67671        JMP FNDA1
3635*
3636*  ROUTINE TO CONVERT ALL STATEMENT REFERENCES INTO ADDRESS FORMAT
3637*     THE ADDRESS OF A STATEMENT REPLACES THE LINE NUMBER AND
3638*     THE SIGN BIT IS SET
3639*
3640  @5675 @21424 COMPL LDA PBUFF      INITIALIZE TO START OF PROGRAM
3641  @5676 @63660        JSM FNDAD       FIND FIRST LINE #
3642  @5677 170402        RET
3643  @5700 @70312        SAM COMP1
3644  @5701 @61075        JSM FNDPS
3645  @5702 @67703        JMP *+1
3646  @5703 164343        JMP E45-1,I     NOT IN PROGRAM
3647  @5704 @76053        SBP *+1,S       FOUND SET SIGN BIT
3648  @5705 135472        STB TEMPS,I     REPLACE LINE NUMBER
3649  @5706 @63671 COMP1 JSM FNDA1       CONTINUE SEARCH
3650  @5707 170402        RET
3651  @5710 @67700        JMP COMPL+3
3652*
3653*  ROUTINE TO CONVERT ALL STATEMENT REFERENCES INTO LINE NUMBER FORMAT
3654*
3655  @5711 @25424 DCOMP LDB PBUFF
3656  @5712 @35415        STB NXTDT       RESET DATA POINTERS
3657  @5713 @35414        STB DSTRT
3658  @5714 @70742        SAR 16
3659  @5715 @31420        STA PARMN       CLEAR PARAMETER NAME
3660  @5716 @76053        SBP *+1,S       SET DECOMPILE FLAG
3661  @5717 @63514 DCOM1 JSM SSTT+1       SAVE POINTERS
3662  @5720 @21421        LDA PRADD
3663  @5721 @70113        SAP *+2         ALREADY DECOMPLIED?
3664  @5722 @67750        JMP DCOMF       YES. DECOMPILE FUNCTIONS ONLY
3665  @5723 @20166        LDA FWAM
```

```
          #01  9830- PRE-EXECUTION PROCESSING


3666  05724 011405 DCMP5 CPA FWUP       END OF KEY AREA?
3667  05725 067764        JMP DCMP4      YES, NOW DECOMPILE MAINLINE
3668  05726 070070        STA **1
3669  05727 070137        LDB A
3670  05730 070437        ADB A,I
3671  05731 070070        STA **1        COMPUTE START AND END OF KEY
3672  05732 035423 DCMP6 STB PRPTR
3673  05733 063660        JSM FNDAD      FIND ADDRESS WORDS
3674  05734 067744        JMP DCMP2      END OF THIS PROGRAM
3675  05735 070233 DCMP1 SAP DCMP3,C    SKIP IF ALREADY DECOMPILED
3676  05736 070517        LDA A,I        READ LINE NUMBER
3677  05737 070073        SAP **1,C      CLEAR SECURE BIT
3678  05740 131472        STA TEMP5,I    STORE LINE #
3679  05741 063671 DCMP3 JSM FNDA1      FIND NEXT ADDRESS
3680  05742 067744        JMP DCMP2
3681  05743 067735        JMP DCMP1
3682*  END OF CURRENT PROGRAM
3683  05744 021423 DCMP2 LDA PRPTR
3684  05745 011426        CPA PRPTO      END OF MAINLINE?
3685  05746 067750        JMP DCOMF      YES
3686  05747 067724        JMP DCMP5      NO, CONTINUE KEYS
3687*  DECOMPILE FUNCTIONS IN SYMTAB BY SETTING THEIR ADDRESSES TO 0
3688  05750 063543 DCOMF JSM RSTT       RESTORE PROGRAM POINTERS
3689  05751 024040 DCMF1 LDB B37
3690  05752 035470        STB MASK      SET UP MASK FOR FUNCTION TYPE
3691  05753 035471        STB STMP1
3692  05754 025406        LDB VALUF
3693  05755 063374 DCMF2 JSM SVALU      SEARCH SYMTAB FOR FUNCTION
3694  05756 170402        RET           END OF TABLE
3695  05757 004132        ADB M1
3696  05760 070742        SAR 16
3697  05761 074557        STA 0,I       SET ADDRESS TO ZERO
3698  05762 004132        ADB M1
3699  05763 067755        JMP DCMF2     FIND NEXT FUNCTION
3700*
3701*  END OF KEY AREA
3702  05764 021425 DCMP4 LDA PBUF0
3703  05765 025422        LDB UKPTR
3704  05766 076110        RZB **2       KEY MODE?
3705  05767 021424        LDA PBUFF     NO, START OF MAINLINE IS IN PBUFF
3706  05770 025426        LDB PRPTO     GET END OF MAINLINE
3707  05771 067732        JMP DCMP6
3708*
3709*
3710  00020         MDMA  EQU .3        TYPE CODE FOR DTM
3711  00017         MCOMA EQU .2
3712  00012         MDEFA EQU .1
3713  05772 164006 MREM  ABS 5-137778   OP CODE 5 ON STANDARD PAGE
3714*
3715  05773 000000        BSS 1         CHECKSUM WORD**************


          #01  LIST ROUTINE


3717*
3718*     THIS SUBROUTINE LISTS ALL OR
3719*     A PORTION OF THE PROGRAM.
3720*
3721*     ENTRY CONDITIONS:
3722*          (A)=FWA OF LIST
3723*          (B)=LWA+1 OF LIST
3724*          PBUFF=FWA OF PROGRAM
3725*          PRPTR=LWA+1 OF PROGRAM
3726*
3727  06000         ORG 60008
3728  06000 031453 LIST  STA ARW        SET FLAG FOR LIST
3729  06001 035460        STB TMP7      SAVE ENDING ADDRESS
3730  06002 074057        CPA B         NULL LIST?
3731  06003 066005        JMP **2       YES
3732  06004 061344        JSM CRLF1
3733  06005 021453        LDA ARW
3734  06006 031472 LIST1 STA TEMP5      SAVE STARTING ADDRESS
3735  06007 031476        STA TMP11     SET FLAG FOR BLANK AFTER MCOUT
3736  06010 070742        SAR 16
3737  06011 031432        STA DRQST     CLEAR INPUT MODE
3738*  LIST NEXT LINE
3739  06012 160170 L1    JSM CLFRA,I    CLEAR BUFFER
3740  06013 025472        LDB TEMP5
3741  06014 015460        CPB TMP7      FINISHED?
3742  06015 170402        RET           YES
3743  06016 062036        JSM LISTS     NO, PUT NEXT LINE IN BUFFER
3744  06017 025453        LDB ARW
3745  06020 076350        RZB L100      UP OR DOWN ARROW?
3746  06021 020040        LDA .31       YES
3747  06022 045623        ISZ PEND      GUARANTEE ROOM FOR LAZY-T
3748  06023 061016        JSM ONEXT     OUTPUT LAZY-T
3749  06024 020132        LDA M1
3750  06025 031430        STA LNINC     FLAG NEED FOR LIST
3751  06026 165462        JMP NEEDL,I
```

```
        #01  LIST ROUTINE


3752  #6027 #60373 L100  JSM ICNT       COMPUTE COUNT
3753  #6030 #61337       JSM NOWRT      OUTPUT TO PRINTER
3754  #6031 #61344       JSM CRLFI
3755  #6032 #21442       LDA STPFL
3756  #6033 #10040       CPA .31        STOP KEY HIT?
3757  #6034 167517       JMP CRC1A,I    YES. ABORT
3758  #6035 #66012       JMP L1
3759*
3760*  SUBROUTINE TO PUT NEXT LINE IN BUFFER
3761*
3762  #6036 #61007 LISTS JSM NEXTA      FIND END OF LINE
3763  #6037 #35475       STB TMP10      SAVE END OF STMT
3764  #6040 #20236       LDA SLEND
3765  #6041 #31523       STA PEND       SET END OF BUFFER POINTER
3766  #6042 125472       LDB TEMPS,I
3767  #6043 #74073       SBP **1,C      CLEAR SECURE BIT
3768  #6044 #35431       STB CLINF
3769  #6045 #76650       R/B LLL        TYPING AID KEY?
3770  #6046 #20100       LDA H52        YES. OUTPUT AN *
3771  #6047 #61016       JSM ONEXT
3772  #6050 #25460       LDB TMP7
3773  #6051 #35475       STB TMP10      SET UP END OF LINE POINTER
3774  #6052 #55472       DSZ TEMPS
3775  #6053 121472       LDA TEMPS,I
3776  #6054 #00133       ADA M2
3777  #6055 #70744       SAL 1          GET CHARACTER COUNT
3778  #6056 #25472       LDB TEMPS
3779  #6057 #04017       ADB .2
3780  #6060 #74744       SBL 1          GET STARTING ADDRESS
3781  #6061 #66276       JMP L28-3      OUTPUT STRING
3782*  OUTPUT STMT NUMBER AND TYPE
3783  #6062 160210 LLL   JSM OUTTA,I
3784  #6063 #61015       JSM OBLNK
3785  #6064 121472       LDA TEMPS,I
3786  #6065 #70213       SAP **4        SECURE?
3787  #6066 #20100       LDA B52        YES. OUTPUT A "*"
3788  #6067 #61016       JSM ONEXT
3789  #6070 #66277       JMP L28-2
3790  #6071 #60602       JSM NEXT1-4    NO. GET PAGE AND TYPE
3791  #6072 #35474       STB TMP9       STORE LAST ADDRESS OF PAGE
3792  #6073 #55472       DSZ TEMPS
3793  #6074 121472       LDA TEMPS,I
3794  #6075 #45472       ISZ TEMPS
3795  #6076 #50107       AND B1000      GET ASCII FLAG
3796  #6077 #31461       STA TMP13
3797  #6100 #21466       LDA LUCL1      RECALL OP CODE
3798  #6101 #10132       CPA M1         IMPLIED LET?
3799  #6102 #66117       JMP L23        YES
3800  #6103 #70076       ICA            GET OP-CODE
3801  #6104 #04132       ADA M1         GET STMT NAME TABLE ADDRESS
3802  #6105 #74437       ADA H,I
3803  #6106 #74744       SBL 1          COMPUTE TABLE ADDRESS
3804*  OUTPUT MULTI-CHARACTER OPERATOR
3805  #6107 #62306 L6    JSM MCOUT      OUTPUT NAME
3806  #6110 #21461       LDA TMP13
3807  #6111 #70110       SZA **2        ASCII FORMAT?
3808  #6112 #66266       JMP L25        YES
3809  #6113 #21476       LDA TMP11
3810  #6114 #70110       SZA **2
3811  #6115 #61015       JSM OBLNK
3812  #6116 #35476       STB TMP11      RESET BLANK FLAG
3813*  OUTPUT NEXT OPERAND
3814  #6117 121472 L23   LDA TEMPS,I
3815  #6120 #50067       AND TYPEL      ISOLATE OPERAND TYPE
3816  #6121 #70472       SAM L4,C       SKIP IF CONSTANT OR FUNCTION
3817  #6122 #31477       STA TMP12      SAVE TYPE
3818  #6123 121472       LDA TEMPS,I
3819  #6124 #50065       AND OPMSK      WAS THE PREVIOUS OPERATOR
3820  #6125 #10065       CPA OPMSK        EXPANDED?
3821  #6126 #66152       JMP L11        YES SKIP THIS FIELD
3822  #6127 #62302       JSM GTOPD
3823  #6130 #71710       SZA *-2        SKIP IF NULL OPERAND
3824  #6131 #66221       JMP L15
3825*  OUTPUT CONSTANT
3826  #6132 #10040 L4    CPA .31        PREDEFINED FUNCTION?
3827  #6133 #66255       JMP L9         YES
3828  #6134 #00132       ADA M1
3829  #6135 #31705       STA FLOAT      SAVE FIX/FLOAT FLAG (0=FIX)
3830  #6136 #00133       ADA M2
3831  #6137 #70313       SAP L12        SKIP IF INTEGER
3832  #6140 160232       JSM SNFLA,I    CONVERT
3833  #6141 #20133       LDA M2
3834  #6142 #31703       STA PLACE      DECIMAL PLACES NOT SPECIFIED
3835  #6143 #62353       JSM NMOUT      OUTPUT NUMBER FROM AR2
3836  #6144 #66152       JMP L11
3837*  OUTPUT INTEGER
3838  #6145 #45472 L12   ISZ TEMPS
3839  #6146 125472       LDB TEMPS,I    GET INTEGER
3840  #6147 #74133       SBP **2,C      COMPILED FORMAT?
```

```
          #01  LIST ROUTINE


3841  #6150 #74537        LDB B,I        YES, READ LINE #
3842  #6151 160210        JSM OUTIA,I    OUTPUT INTEGER
3843*  CHECK FOR END OF STATEMENT
3844  #6152 #45472 L11    ISZ TEMPS
3845  #6153 #21472        LDA TEMPS
3846  #6154 #11475        CPA TMP10
3847  #6155 170402        RET
3848*  OUTPUT NEXT OPERATOR
3849  #6156 121472        LDA TEMPS,I
3850  #6157 #50065        AND OPMSK
3851  #6160 #70442        SAR 10
3852  #6161 #70510        SZA L13        SKIP IF NULL OPERATOR
3853  #6162 #10012        CPA .1         QUOTE?
3854  #6163 #66213        JMP L29        YES
3855  #6164 #24034        LDB .21
3856  #6165 #60753        JSM LIMIT      SINGLE CHARACTER OPERATOR?
3857  #6166 #66174        JMP L3         NO
3858  #6167 #00123        ADA FUPRS
3859  #6170 #70517        LDA A,I
3860  #6171 #70342        SAR 8          GET ASCII FROM TABLE
3861  #6172 #61016        JSM ONEXT
3862  #6173 #66117 L13    JMP L23        GO OUTPUT NEXT OPERAND
3863*  OUTPUT MULTI-CHAR OPERATOR
3864  #6174 #31703 L3     STA PLACE      SAVE OPERATOR IN TEMP
3865  #6175 #10040        CPA .31        EXPANDED OP?
3866  #6176 #62302        JSM GTOPD      YES, GET OP CODE
3867  #6177 #31704        STA WIDTH      SAVE
3868  #6200 #70142        SAR 4
3869  #6201 #31476        STA TMP11      SET FLAG FOR BLANK
3870  #6202 #70110        SZA *+2        OUTPUT BLANK IF
3871  #6203 #61015        JSM OBLNK        OP > 17B
3872  #6204 #21703        LDA PLACE
3873  #6205 #24130        LDB FROMA      GET STANDARD PAGE
3874  #6206 #10040        CPA .31        EXPANDED OP?
3875  #6207 #25474        LDB TMP9       YES, GET PAGE FOR STMT
3876  #6210 #21704        LDA WIDTH
3877  #6211 #04135        ADB M4
3878  #6212 #66105        JMP L6-2       COMPUTE TABLE ADDR AND OUTPUT
3879*  OUTPUT QUOTE FIELD
3880  #6213 #20043 L29    LDA .34
3881  #6214 #61016        JSM ONEXT
3882  #6215 #62647        JSM QUOTE
3883  #6216 #20043        LDA .34
3884  #6217 #61016        JSM ONEXT
3885  #6220 #66153        JMP L11+1
3886*  OUTPUT OPERAND NAME AND TYPE
3887  #6221 #21477 L15    LDA TMP12
3888  #6222 #10040        CPA .31        FUNCTION?
3889  #6223 #62751        JSM L16        YES
3890  #6224 #62302 L8     JSM GTOPD
3891  #6225 #00101        ADA B100
3892  #6226 #61016        JSM ONEXT      OUTPUT LETTER NAME
3893  #6227 #21477        LDB TMP12
3894  #6230 #14040        CPB .31        FUNCTION?
3895  #6231 #66152        JMP L11        YES
3896  #6232 #20077        LDA DOLAR
3897  #6233 #14076        CPB B36        STRING
3898  #6234 #66241        JMP L19        YES
3899  #6235 #74146        RBR 4
3900  #6236 #74251        SLB L20        LETTER AND DIGIT?
3901  #6237 #21477        LDA TMP12      YES, OUTPUT DIGIT
3902  #6240 #40052        IOR .48
3903  #6241 #61016 L19    JSM ONEXT
3904  #6242 #66152        JMP L11
3905  #6243 #74642 L20    SBR 14
3906  #6244 #75710        SZB *-2        LETTER ONLY?
3907  #6245 #20055        LDA I          NO
3908  #6246 #75551        SLB L19        INTEGER?
3909  #6247 #20073        LDA S          NO - SPLIT
3910  #6250 #66241        JMP L19
3911*  OUTPUT "FN"
3912  #6251 #20072 L16    LDA F
3913  #6252 #61016        JSM ONEXT      OUTPUT F
3914  #6253 #23522        LDA N
3915  #6254 #65016        JMP ONEXT      OUTPUT N AND RETURN TO L8
3916*  OUTPUT PRE-DEFINED FUNCTION NAME
3917  #6255 #62302 L9     JSM GTOPD      GET OP-CODE
3918  #6256 #31447        STA LOCL2
3919  #6257 #60605        JSM NEXT1-1    GET PAGE
3920  #6260 #04134        ADB M3
3921  #6261 #74437        ADB d,I
3922  #6262 #21467        LDA LOCL2
3923  #6263 #74744        SBL 1          COMPUTE TABLE ADDRESS
3924  #6264 #62306        JSM MCOUT
3925  #6265 #66117        JMP L23
3926*  OUTPUT REM STATEMENT
3927  #6266 #21472 L25    LDA TEMPS
3928  #6267 #70076        TCA
3929  #6270 #01475        ADA TMP10
3930  #6271 #70744        SAL 1
```

```
          #01  LIST ROUTINE


3931  #6272 #00132        ADA M1          COMPUTE CHARACTER COUNT
3932  #6273 #25472        LDB TEMPS
3933  #6274 #74744        SHL 1
3934  #6275 #74070        SIR *+1         COMPUTE STARTING ADDRESS
3935  #6276 #62651        JSM STOUT       OUTPUT STRING TO BUFFER
3936  #6277 #25475        LDB TMP10
3937  #6300 #35472        STA TEMPS
3938  #6301 170402 L2R    RET             LINE COMPLETE
3939*
3940*  GET OPERAND NAME
3941*
3942  #6302 121472 GTOPD LDA TEMPS,I
3943  #6303 #50115        AND H1777
3944  #6304 #70202        SAR 5
3945  #6305 170402        RET
          #01  OUTPUT TO BUFFER ROUTINES


3947*
3948*     THIS SUBROUTINE SEARCHES THE
3949*     MULTIPLE CHARACTER OP CODE
3950*     TABLE. IT RETURNS WITH THE
3951*     ASCII STRING IN THE SYNTAX
3952*     BUFFER.
3953*
3954*     ENTRY CONDITIONS:
3955*
3956*          (A)= OP CODE
3957*          (B)= POINTER TO FIRST
3958*               CHARACTER OF TABLE
3959*
3960*
3961  #6306 #31741 MCOUT STA TMP1       SAVE CODE
3962  #6307 #35740        STA TMP
3963  #6310 #70742 MC1    SAR 16
3964  #6311 #31743        STA TMP3      COUNT=0
3965  #6312 #61053 MC4    JSM GTMP
3966  #6313 #70306        RAR 7
3967  #6314 #70331        SLA MC2,C     OP CODE?
3968  #6315 #70402        SAR 9         YES
3969  #6316 #50040        AND .31
3970  #6317 #11741        CPA TMP1      MATCH?
3971  #6320 #66324        JMP MC3       YES
3972  #6321 #66310        JMP MC1
3973  #6322 #45743 MC2    ISZ TMP3
3974  #6323 #66312        JMP MC4
3975  #6324 #21743 MC3    LDA TMP3      GET COUNT
3976  #6325 #25743        LDB TMP3
3977  #6326 #74056        CMB
3978  #6327 #05740        ADB TMP       COMPUTE STARTING ADDRESS
3979  #6330 #66651        JMP STOUT
3981*
3982*     THIS SUBROUTINE ROUNDS THE NUMBER
3983*     IN AR2.  IF OVERFLOW OCCURSBUMP THE
3984*     EXPONENT AND SHIFT A ONE IN AS THE
3985*     MSD.  B HAS THE LEADING ZERO COUNT
3986*     FOR THE ROUNDING CONSTANT.
3987*
3988*
3989  #6331 #21744 RND2   LDA AR1
3990  #6332 #31754        STA AR2
3991  #6333 170402        RET
3992  #6334 #61256 RND    JSM CLAR2
3993  #6335 #20022        LDA .5
3994  #6336 174470        MRY           ROUND OFF LSD-1
3995  #6337 170560        EXA
3996  #6340 170002        NOV           SET A=E
3997  #6341 #71410        SZA RND2
3998  #6342 #24012        LDB .1
3999  #6343 174470        MRY           SHIFT IN A ONE
4000  #6344 #20013        LDA B400
4001  #6345 #01744        ADA AR1
4002  #6346 #31754        STA AR2       BUMP EXPONENT
4003  #6347 #70340        AAR 8
4004  #6350 #10060        CPA .100      EXPONENT OVERFLOW?
4005  #6351 #64444        JMP XFAR2
4006  #6352 #55777        ISZ RSPTR
4007*
4008*
4009*
4010*     NUMOUT ROUTINE
4011*
4012*     THIS ROUTINE OUTPUTS A NORMALIZED
4013*     NUMBER FROM AR2 INTO A BUFFER/
4014*
4015*     PLACE = DIGITS FOLLOWING DECIMAL POINT
4016*     PLACE = -1  STANDARD FORMAT
4017*     PLACE = -2  LIST NUMBER
4018*     WIDTH = FIELD WIDTH
4019*     HADDR = OUTPUT BUFFER POINTER
```

```
          #01  OUTPUT TO BUFFER ROUTINES


4020*     FLOAT = 1 SPECIFIES FLOATING POINT
4021*
4022*
4023  @6353 @60435 NMOUT JSM XFAR1
4024  @6354 @24012       LDB .1
4025  @6355 @35751       STB DIGIT      INITIALIZE
4026  @6356 171400 NM3   MLS            SHIFT NEXT DIGIT
4027  @6357 @73450       SZA NM1        DIGIT IS ZERO?
4028  @6360 @21703       LDA PLACE
4029  @6361 @70070       STA *+1
4030  @6362 @72250       RZA NM2        STANDARD FORMAT?
4031  @6363 @14027       CPB .11
4032  @6364 @66374       JMP NM4
4033  @6365 @14030       CPB .12
4034  @6366 @66374       JMP NM4
4035  @6367 @35751 NM2   STB DIGIT
4036  @6370 @74070 NM1   SIB *+1
4037  @6371 @14031       CPB .13        LAST DIGIT?
4038  @6372 @66374       JMP *+2        YES
4039  @6373 @66356       JMP NM3        NO
4040  @6374 @60444 NM4   JSM XFAR2
4041  @6375 @21705       LDA FLOAT
4042  @6376 @72110       RZA *+2        FIXED OR FLOAT?
4043  @6377 @66467       JMP OFXD       FIXED
4044*
4045*
4046*     OUTPUT A FLOATING POINT NUMBER
4047*     THE NUMBER WILL HAVE THE FORM:
4048*
4049*     SX.XXXXXXXXESXX
4050*
4051  @6400 @25703 OFLT  LDB PLACE
4052  @6401 @21416       LDA MODE
4053  @6402 @70151       SLA *+3        SKIP IF PROG OR LIST
4054  @6403 @35741       STB R
4055  @6404 @66433       JMP FLT2+1
4056  @6405 @74513       SBP FLT1       STANDARD FORMAT?
4057  @6406 @14132       CPB M1         LIST NUMBER?
4058  @6407 @66414       JMP FLT7       NO
4059  @6410 @21751       LDA DIGIT      YES
4060  @6411 @00132       ADA M1
4061  @6412 @31741       STA R          R = DIGIT-1
4062  @6413 @66435       JMP FLT5
4063  @6414 @24022 FLT7  LDB .5         STD PRINTOUT--FLOAT 5
4064  @6415 @35741 FLT11 STB R
4065  @6416 @66432       JMP FLT2
4066  @6417 @35741 FLT1  STB R
4067  @6420 @76110       RZB *+2        ANY RIGHT DIGITS?
4068  @6421 @04132       ADB M1         NO. ALLOW ONE LESS CHARACTER
4069  @6422 @04024       ADB .7
4070  @6423 @74076       TCB
4071  @6424 @05704       ADA WIDTH
4072  @6425 @74213       SBP FLT4
4073  @6426 @20077 FLT3  LDA DOLAR      NO
4074  @6427 @25704       LDB WIDTH
4075  @6430 @67475       JMP MOUT
4076  @6431 @67474 FLT4  JSM MOUT-1
4077  @6432 @25741 FLT2  LDB R
4078  @6433 @04017       ADB .2
4079  @6434 @62334       JSM RND        ROUND THE NUMBER
4080  @6435 @62637 FLT5  JSM OSIGN
4081  @6436 @61314       JSM ODGIT      OUTPUT A DIGIT FROM AR2
4082  @6437 @21741       LDA R
4083  @6440 @70310       SZA FLT8       ANY RIGHT DIGITS?
4084  @6441 @20051       LDA POINT
4085  @6442 @61016       JSM ONEXT      YES. OUTPUT DECIMAL POINT
4086  @6443 @61314 FLT6  JSM ODGIT
4087  @6444 @55741       DSZ R          DINC R
4088  @6445 @66443       JMP FLT6
4089  @6446 @20071 FLT8  LDA E
4090  @6447 @61016       JSM ONEXT
4091  @6450 @25754       LDB AR2
4092  @6451 @74340       AHR B          GET EXPONENT
4093  @6452 @20047       LDA PLUS
4094  @6453 @74153       SBP FLT10      NEGATIVE EXPONENT?
4095  @6454 @20050       LDA MINUS      YES
4096  @6455 @74076       TCA
4097  @6456 @35743 FLT10 STB INDEX      SAVE EXPONENT
4098  @6457 @61016       JSM ONEXT
4099  @6460 @25743       LDB INDEX
4100  @6461 @04141       ADB M10
4101  @6462 @74153       SBP FLT9       EXPONENT LT 10?
4102  @6463 @20052       LDA B60
4103  @6464 @61016       JSM ONEXT
4104  @6465 @25743 FLT9  LDB INDEX
4105  @6466 164210       JMP OUTIA,I    OUTPUT EXPONENT
```

#01 OUTPUT TO BUFFER ROUTINES

```
4107*
4108*    OUTPUT A FIXED POINT NUMBER
4109*    IF IT WILL FIT IN XE SPECIFIED
4110*    FIELD. THE NORMALIZED NUMBER IS
4111*    IN REGISTER AR1.
4112*
4113*
4114  @6467 @21754 OFXO  LDA AR2
4115  @6470 @70340       AAR 8
4116  @6471 @70070       STA *+1
4117  @6472 @31742       STA L
4118  @6473 @74742       SBR 16
4119  @6474 @35750       STB RBASE
4120  @6475 @70076       TCA
4121  @6476 @70152       SAM RL4
4122  @6477 @35742       STB L
4123  @6500 @31750       STA RBASE
4124  @6501 @01751 RL4   ADA DIGIT
4125  @6502 @70113       SAP *+2          DIGIT GE EXP?
4126  @6503 @70742       SAR 16           NO. R=0
4127  @6504 @31741       STA R            R=DIGIT*RBASE-L
4128  @6505 @25703 RL3   LDB PLACE
4129  @6506 @14133       CPB M2           LIST NUMBER?
4130  @6507 @66567       JMP NN2          YES
4131  @6510 @74313       SBP N2           STANDARD FORMAT?
4132  @6511 @25741       LDB R            YES
4133  @6512 @14027       CPB .11          11 RIGHT DIGITS?
4134  @6513 @04133       ADB M2           YES. USE ONLY 9
4135  @6514 @14061       CPB .10          10 RIGHT DIGITS?
4136  @6515 @04132       ADB M1           YES. USE ONLY 9
4137  @6516 @35741 N2    STB R
4138  @6517 @74076       TCB
4139  @6520 @05750       ADB RBASE
4140  @6521 @74076       TCB
4141  @6522 @05742       ADB L
4142  @6523 @04143       ADB M12
4143  @6524 @74353       SBP N1
4144  @6525 @04031       ADB .13
4145  @6526 @62334 N4    JSM RND          ROUND THE NUMBER
4146  @6527 @66533       JMP N1
4147  @6530 @25754 N6    LDB AR2
4148  @6531 @74112       SBM N1         L GE 0?
4149  @6532 @45742       ISZ L            YES. BUMP L
4150  @6533 @21416 N1    LDA MODF
4151  @6534 @25754       LDB AR2
4152  @6535 @74340       ABR 8
4153  @6536 @70311       SLA NN3          CALC MODE?
4154  @6537 @04143       ADB M12          YES
4155  @6540 @74513       SBP NN           OVERFLOW?
4156  @6541 @05703       ADB PLACE        NO
4157  @6542 @04030       ADB .12
4158  @6543 @74313       SBP N7           UNDERFLOW?
4159  @6544 @21703 NN3   LDA PLACE
4160  @6545 @70313       SAP NN1          STANDARD FORMAT?
4161  @6546 @04141       ADB M10
4162  @6547 @74153       SBP NN           EXPONENT GT 9?
4163  @6550 @04030       ADB .12
4164  @6551 @74713 N7    SBP NN2          OVERFLOW?
4165  @6552 @66400 NN    JMP OFLT         YES
4166  @6553 @25741 NN1   LDB R
4167  @6554 @74130       SIB *+2          R=0?
4168  @6555 @04012       ADB .1           NO. R=R+2
4169  @6556 @21742       LDA L            YES. R=R+1
4170  @6557 @70130       SIA *+2          L=0?
4171  @6560 @00132       ADA M1           NO
4172  @6561 @70037       ADB A            YES. USE L=1
4173  @6562 @74076       TCB
4174  @6563 @05704       ADB WIDTH
4175  @6564 @75312       SBM NN           NUMBER FITS IN FIELD?
4176  @6565 @35461       STB TMP13
4177  @6566 @63474       JSM MOUT-1
4178  @6567 @62637 NN2   JSM OSIGN
4179  @6570 @21742       LDA L
4180  @6571 @25742       LDB L
4181  @6572 @04144       ADB M13
4182  @6573 @74112       SBM NN10         MORE THAN 12 DIGITS?
4183  @6574 @20030       LDA .12          YES. USE ONLY 12
4184  @6575 @31461 NN10  STA TMP13
4185  @6576 @70510       SZA NN4
4186  @6577 @61314 NN5   JSM ODGIT
4187  @6600 @55461       DSZ TMP13        LAST LEFT DIGIT?
4188  @6601 @66577       JMP NN5          NO
4189  @6602 @25742       LDB L
4190  @6603 @04143       ADB M12
4191  @6604 @74312       SBM NN4+2
4192  @6605 @74250       SZB NN4+2
4193  @6606 @20052       LDA B60
4194  @6607 @63475       JSM MOUT         FILL WITH ZEROES
4195  @6610 @20052 NN4   LDA B60
4196  @6611 @61016       JSM ONEXT
```

```
        #01  OUTPUT TO BUFFER ROUTINES


4197  @6612 @21741        LDA R
4198  @6613 @72110        SZA **2
4199  @6614 @66636 NN11   JMP NN6
4200  @6615 @20051        LDA POINT
4201  @6616 @61016        JSM ONEXT
4202  @6617 @25750        LDB RBASE
4203  @6620 @74117        LDA B
4204  @6621 @70076        TCA
4205  @6622 @01741        ADA R
4206  @6623 @70153        SAP **3
4207  @6624 @70742        SAR 16
4208  @6625 @25741        LDB R
4209  @6626 @31741        STA R           R=R-RBASE
4210  @6627 @20052        LDA H60
4211  @6630 @63475        JSM MOUT        FILL WITH ZEROES
4212  @6631 @21741 NN7    LDA R
4213  @6632 @70210        SZA NN6
4214  @6633 @61314        JSM GDGIT
4215  @6634 @55741        DSZ R           LAST RIGHT DIGIT?
4216  @6635 @66631        JMP NN7         NO
4217  @6636 170402 NN6    RET             YES*DONE
4218*
4220*
4221*    THIS SUBROUTINE OUTPUTS THE MANTISSA SIGN
4222*
4223*    MANTISSA NEGATIVE----UNCONDITIONALLY OUTPUT -
4224*
4225*    MANTISSA POSITIVE
4226*        LIST NUMBER------OUTPUT NOTHING
4227*        OTHERWISE--------OUTPUT BLANK
4228*
4229*
4230  @6637 @21754 OSIGN  LDA AR2
4231  @6640 @70151        SLA OS1         MANTISSA POSITIVE?
4232  @6641 @20050        LDA MINUS       NO
4233  @6642 @65016        JMP ONEXT
4234  @6643 @21703 OS1    LDA PLACE
4235  @6644 @10133        CPA M2          LIST NUMBER?
4236  @6645 170402        RET             YES
4237  @6646 @65015        JMP OBLNK
4239*
4240*    OUTPUT A QUOTE STRING
4241*
4242  @6647 @63261 QUOTE  JSM QTCH1       GET CHAR COUNT AND ADDRESS
4243  @6650 @00000        BSS 1           CHECKSUM WORD****(QTCH1 RETURNS TO P+2)
4244*
4245*
4246*    OUTPUT ANY STRING
4247*
4248*    ENTRY CONDITIONS:
4249*
4250*        A=NUMBER OF CHARACTERS
4251*        B=POINTER FIRST CHAR
4252*
4253  @6651 @31466 STOUT  STA LOCL1       SAVE COUNT
4254  @6652 @70742        SAR 16
4255  @6653 @31701        STA NOISP       ENABLE DISPLAY
4256  @6654 @20012        LDA .1
4257  @6655 @65332        JMP .NPWR+2     DISABLE PRINT


        #01  PRINT DISP AND WRITE COMMAND EXECUTION


4259*
4260*  SUBROUTINE TO PLACE NEXT PRINT FIELD IN BUFFER
4261*
4262*    RETURN P+1: END OF LINE (A)=0 FOR CRLF NOT SUPPRESSED
4263*           P+2: ITEM IN BUFFER (HADDR)=BUFFER POINTER
4264*
4265*
4266  @6656 @63227 PRINN  JSM CKEOL       END OF LINE?
4267  @6657 @74742        SBR 16          YES CRLF NOT SUPPRESSED
4268  @6660 170402        RET             YES CRLF SUPPRESSED
4269  @6661 @63004        JSM PFOR1       SET UP 72 CHAR LINE
4270  @6662 @63240        JSM QTCHK       QUOTE FIELD NEXT?
4271  @6663 @66674        JMP PR5         NO
4272  @6664 @21471 QOV    LDA L2T2        RECALL COUNT
4273  @6665 @03523        ADA M72
4274  @6666 @70113        SAP **2         MORE THAN 72 CHARS?
4275  @6667 @70742        SAR 16          NO
4276  @6670 @70076        TCA
4277  @6671 @01471        ADA L2T2        TRUNCATE TO 72
4278  @6672 @62651        JSM STOUT       OUTPUT STRING
4279  @6673 @66731        JMP PR7
4280  @6674 121472 PR5    LDA TEMPS,I
4281  @6675 @50070        AND OPDMK
4282  @6676 @12732        CPA TABOP       TAB?
4283  @6677 @67040        JMP ETAB        YES
4284  @6700 @21453        LDA PRFLG
```

```
          #01  PRINT: DISP# AND WRITE COMMAND EXECUTION


4285  #6701 #10132        CPA M1
4286  #6702 #66706        JMP *+4         KEYBOARD "LET"
4287  #6703 #62773        JSM PFORM
4288  #6704 #61220        JSM FETCH+1     EVALUATE EXPRESSION
4289  #6705 #66707        JMP *+2
4290  #6706 #62773        JSM PFORM       EVALUATE "LET"
4291  #6707 #25622        LDA TERM
4292  #6710 #76150        SZB *+3         CRLF GIVEN IN FUNCTION?
4293  #6711 #60170        JSM CLERA,I     YES
4294  #6712 #35622        STB TERM
4295  #6713 #66721        JMP PR6         NO
4296* BUFFER OVERFLOW
4297  #6714 #25742 POV    LDB TMP2
4298  #6715 #74110        SZB *+2
4299  #6716 #66664        JMP NOV         OVERFLOW DURING STRING
4300  #6717 #20215        LDA TMPOP       NO, RESTORE AR2
4301  #6720 #60445        JSM XFAR2,I
4302  #6721 #24132 PR6    LDB M1
4303  #6722 #21416        LDA MODE
4304  #6723 #70150        SZA PR11        SKIP IF PROG MODE
4305  #6724 #25452        LDB CPLAC
4306  #6725 #21451        LDA CTYPE       SET UP CALC FORMAT FLAGS
4307  #6726 #63065 PR11   JSM GNUM1       PUT NUMBER IN BUFFER
4308  #6727 #74742        SBR 16
4309  #6730 #35742        STB TMP2        SET FLAG FOR NUMBER
4310  #6731 #63052 PR7    JSM OVCK        CHECK FOR OVERFLOW
4311  #6732 #00077 TAROP  OCT 100077      (OVCK RETURNS TO P+2)
4312  #6733 #75752        SBM POV         SKIP IF OVERFLOW
4313  #6734 #21472        LDA TEMPS
4314  #6735 #11421        CPA PRADD       END OF LINE?
4315  #6736 #64357        JMP RETN2       YES RETURN TO P+2
4316  #6737 #21472        LDA TEMPS,I     NO, GET NEXT OPERATOR
4317  #6740 #70073        SAP **1,C
4318  #6741 #07#442       SAR 1#
4319  #6742 #00133        ADA ,M2         SET UP COMMA:SEMICOLON FLAG
4320  #6743 #63027        JSM FILL        OUTPUT TRAILING BLANKS
4321  #6744 #64357        JMP RETN2
4322*
4323*
4324*  PRINT DISP AND WRITE(N,*) ROUTINE
4325*
4326  #6745 #20146 EPRIN LDA M15          SET UP SELECT CODE 15
4327  #6746 #70076 EDISP CCA              NEG= DISP, -1= LET
4328  #6747 #31453        STA PRFLG       SELECT CODE=PRINT OR WRITE
4329  #6750 #25416 EWRI1  LDB MODE
4330  #6751 #76150        SZB *+3         SKIP IF KEYBOARD MODE
4331  #6752 #21622        LDA TERM
4332  #6753 #72150        SZA *+3         SKIP IF BUFFER IS PARTIALLY FILLED
4333  #6754 #60170        JSM CLERA,I     INITIALIZE BUFFER
4334  #6755 #35622        STB TERM        INITIALIZE CRLF SUPPRESSED FLAG
4335  #6756 #63227        JSM CKEOL       NULL LIST?
4336  #6757 #66765        JMP PR2-1
4337  #6760 #66765        JMP PR2-1       YES, GIVE CRLF
4338  #6761 #62656 PR3    JSM PRINN       OUTPUT NEXT FIELD TO BUFFER
4339  #6762 #66766        JMP PR2         END OF LINE
4340  #6763 #63458        JSM OLINE       PRINT THE LATEST FIELD
4341  #6764 #66761        JMP PR3
4342*
4343  #6765 #74742        SBR 16
4344  #6766 #35622 PR2    STB TERM        SET CRLF SUPPRESSED FLAG
4345  #6767 #76150        SZB PR17        SKIP IF SUPPRESSED
4346  #6770 #63007        JSM PRCHK       DISP?
4347  #6771 #61344        JSM CRLF1       OUTPUT A CRLF
4348  #6772 #64172 PR17   JMP XEC4A,I
4349*
4350*  SAVE PRINT FLAGS AND CALL FORMX
4351*
4352  #6773 #25453 PFORM LDB PRFLG
4353  #6774 #60563        JSM SLWST
4354  #6775 #25447        LDB SLCOD
4355  #6776 #60563        JSM SLWST
4356  #6777 #60201        JSM FORMA,I     EVALUATE FORMULA
4357  #7000 #60567        JSM UNSTK
4358  #7001 #31447        STA SLCOD       RESTORE SELECT CODE
4359  #7002 #60567        JSM UNSTK
4360  #7003 #31453        STA PRFLG       RESTORE PRINT FLAG
4361  #7004 #23521 PFOR1  LDA LN72
4362  #7005 #31623        STA PEND      SET UP 72 CHAR LINE
4363  #7006 #170402       RET
4365  #7007 #25450 PRCHK  LDB PMODE
4366  #7010 #74310        SZB PRCHR       PRINT ALL?
4367  #7011 #25453        LDB PRFLG       NO
4368  #7012 #74213        SBP PRCHR       PRINT STMT?
4369  #7013 #64357        JMP RETN2       NO
4370*    THIS SUBROUTINE FILLS A COMMA
4371*    OR SEMICOLON FIELD WITH BLANKS/
4372*    IF THE CURRENT LINE IS OVERFLOWED
4373*    IT IS OUTPUT AND A CRLF IS DONE/
4374*
```

```
        #01  PRINT# DISP# AND WRITE COMMAND EXECUTION

4375*    ENTRY CONDITIONS:
4376*
4377*         A=0       IF COMMA FIELD
4378*         TMP2#0 IF QUOTE FIELD WAS PRINTED
4379*
4380  07014 021742 FILL1 LDA TMP2
4381  07015 070110       SZA *+2        SKIP IF NUMERIC
4382  07016 170402 PRCHR RET
4383  07017 061015       JSM OBLNK      INSURE AT LEAST 1 BLANK
4384  07020 025625       LDB FIRST      NO
4385  07021 074076       TCB
4386  07022 005400       ADB BADDR      GET OUTPUT COUNT
4387  07023 004137       ADB M6
4388  07024 074512       SBM FILL2
4389  07025 004134       ADB M3
4390  07026 067024       JMP *-2
4391  07027 073250 FILL  SZA FILL1      COMMA FIELD?
4392  07030 063014       JSM FILL1      YES. FIRST OUTPUT "#" SPACING
4393  07031 025400       LDB BADDR
4394  07032 007520       ADB MSLPT      GET OUTPUT COUNT
4395  07033 075150       SZB PRCHR      SKIP IF EXACT
4396  07034 004146       ADB M15
4397  07035 075713       SBP *-2
4398  07036 074076 FILL2 TCB
4399  07037 067474       JMP MOUT-1     OUTPUT BLANKS
4400*
4401*
4402*    TAB EXECUTION
4403*
4404*
4405  07040 045472 ETAB  ISZ TEMPS
4406  07041 062773       JSM PFORM
4407  07042 061220       JSM FETCH+1
4408  07043 160226       JSM FLTFA,I    TRUNCATE TO INTEGER
4409  07044 064357       JMP RETN?      OVERFLOW
4410  07045 075752       SBM *-1        NEGATIVE NUMBER?
4411  07046 074076       TCB
4412  07047 005400       ADB BADDR      COMPUTE # OF BLANKS NEEDED
4413  07050 007520       ADB MSLPT
4414  07051 063036       JSM FILL2
4415  07052 025400 OVCK  LDB BADDR
4416  07053 015623       CPB PEND       OVERFLOW?
4417  07054 067056       JMP ET1        YES
4418  07055 064357       JMP RETN?
4419  07056 160170 ET1   JSM CLERA,I
4420  07057 063007       JSM PRCHK      PRINT?
4421  07060 061344       JSM CRLF1      YES
4422  07061 024062       LDB FLGRT      SET (B) NEGATIVE FOR OVERFLOW
4423  07062 035622       STB TERM       SET CRLF SUPRESSED FLAG
4424  07063 064357       JMP RETN?
4425*
4426  07064 160206 TABER JSM ERRA,I     ERROR IF CALLED FROM WRITE STMT
4427*
4428*
4429  07065 035703 GNUM1 STB PLACE
4430  07066 031705       STA FLOAT
4431  07067 024215 GNUM  LDB TMPOP
4432  07070 060406       JSM XFAR1+1    SAVE RESULT
4433  07071 066353       JMP NMOUT      PUT NUMBER IN BUFFER
4491  07152 074073       SBP *+1,C
4492  07153 035456       STB FORMS      SET NUMERIC SPEC FOUND FLAG
4493  07154 125466       LDB LOCL1,I    GET DECIMAL SPEC
4494  07155 063356       JSM FSPR       UPDATE FORMAT POINTERS
4495  07156 031704       STA WIDTH
4496  07157 070353       SAP WRIT7      SKIP IF F OR F
4497*  OUTPUT A FIELD CHARACTER
4498  07160 160225       JSM FLTRA,I    CONVERT TO ROUNDED INTEGER
4499  07161 074742       SBR 16         OVERFLOW? USE 0
4500  07162 074117       LDA B
4501  07163 050105       AND B377       MASK TO 8 BITS
4502  07164 061353       JSM TTY        OUTPUT SINGLE CHAR
4503  07165 067216       JMP WRIT6
4504*  OUTPUT E OR F NUMBER
4505  07166 035703 WRIT7 STB PLACE
4506  07167 021412       LDA LSTPT      START TEMP BUFFER
4507  07170 070070       STA *+1
4508  07171 070744       SAL 1
4509  07172 031400       STA BADDR        ABOVE LOW STACK
4510  07173 031706       STA SBPTR
4511  07174 021413       LDA HSTPT
4512  07175 070744       SAL 1
4513  07176 031623       STA PEND       SET END OF BUFFER AT H-STACK
4514  07177 021416       LDA MODE
4515  07200 031641       STA ST+1
4516  07201 070742       SAR 16
4517  07202 031416       STA MODE       SET FOR PROGRAM FORMAT
4518  07203 063067       JSM GNUM       OUTPUT NUMBER TO BUFFER
4519  07204 021641       LDA ST+1
4520  07205 031416       STA MODE
4521  07206 021400       LDA BADDR
```

```
        #P1  PRINT: DISP: AND WRITE COMMAND EXECUTION

4522  07207 011623        CPA PEND      OVERFLOW?
4523  07210 064561        JMP MERR
4524  07211 025736        LDB SBPTR
4525  07212 063374        JSM WBUFF     COMPUTE CHAR COUNT
4526  07213 061337        JSM NOWRT     PRINT THE BUFFER
4527  07214 025625        LDB FIRST
4528  07215 035400        STB BADDR     RESET BADDR
4529  07216 063227 WRITA  JSM CKEOL     END OF STMT?
4530  07217 074742        SBR 16        YES. CRLF NOT SUPRESSED
4531  07220 067224        JMP WRITB     YES
4532  07221 067100        JMP WRIT2     NO
4533* END OF WRITE STMT
4534  07222 055412        DSZ LSTPT
4535  07223 055412        DSZ LSTPT     UNSTACK FLAGS
4536  07224 070742 WRITB  SAR 16
4537  07225 031455        STA FORMT     SET FORMAT COMPLETE FLAG
4538  07226 066767        JMP PR2+1
4540*
4541* CHECK FOR END OF OUTPUT LIST
4542*   P+1:  TEMPS=PRADD
4543*   P+2:  NULL OPERAND AND TEMPS+1=PRADD
4544*   P+3:  ALL OTHER
4545*
4546  07227 025472 CKEOL  LDB TEMPS
4547  07230 015421        CPB PRADD     END OF STMT?
4548  07231 170402        RET
4549  07232 060771        JSM GTOPP
4550  07233 072210        RZA CKEO1     NULL OPERAND?
4551  07234 074070        SIB **1       YES
4552  07235 015421        CPB PRADD     TEMPS+1=PRADD?
4553  07236 064357        JMP RETN2      YES
4554  07237 064356 CKEO1  JMP RETN3     NO
4555*
4556* CHECK TO SEE IF NEXT OPERATOR IS A QUOTE FIELD
4557*   P+1:  QUOTE NOT NEXT
4558*   P+2:  QUOTE NEXT:  (A)=LENGTH: (B)=POINTER: TEMPS UPDATED
4559*
4560  07240 121472 QTCHK  LDA TEMPS,I
4561  07241 050065        AND OPMSK     CURRENT OPERATOR
4562  07242 010116        CPA QTOP        A QUOTE?
4563  07243 067261        JMP QTCH1     YES
4564  07244 060771        JSM GTOPP
4565  07245 072410        RZA QTCH3     NULL OPERAND?
4566  07246 045472        ISZ TEMPS     YES. GET NEXT OPERATOR
4567  07247 121472        LDA TEMPS,I
4568  07250 050065        AND OPMSK
4569  07251 010116        CPA QTOP      QUOTE?
4570  07252 067261        JMP QTCH1     YES
4571  07253 055472        DSZ TEMPS     NO. RESTORE TEMPS
4572  07254 170402        RET           RETURN P+1
4573  07255 050040 QTCH3  AND B37
4574  07256 010076        CPA B36       STRING VARIABLE?
4575  07257 067276        JMP QTCH2     YES
4576  07260 170402        RET           NO. RET P+1
4577  07261 121472 QTCH1  LDA TEMPS,I
4578  07262 050104        AND B177      GET CHAR COUNT
4579  07263 031471        STA L2T2      SAVE
4580  07264 045472        ISZ TEMPS
4581  07265 025472        LDB TEMPS
4582  07266 074744        SBL 1         GET STARTING CHARADDRESS
4583  07267 035742        STB IMP2      SAVE
4584  07270 070037        ADB A
4585  07271 074070        SIB **1
4586  07272 074002        SBR 1
4587  07273 035472        STB TEMPS     POINT TEMPS PAST END OF STRING
4588  07274 025742        LDB IMP2
4589  07275 064357        JMP RETN2
4590  07276 020121 QTCH2  LDA SBLOC
4591  07277 060613        JSM FPAGE     FIND STRING BLOCK PAGE
4592  07300 074117        LDA B
4593  07301 050065        AND OPMSK     GET STARTING ADDRESS OF PAGE
4594  07302 070737        JMP A,I       JUMP TO STRING PAGE
4595  00121         SBLOC EQU B4000     OP CODE 2
4597*
4598* SUBROUTINE TO GET NEXT FORMAT SPEC
4599*
4600  07303 021454 TYPEC  LDA RCOUT
4601  07304 072450        RZA FSP1
4602  07305 025455        LDB FORMT
4603  07306 015457        CPB FORMF
4604  07307 067353        JMP FSPN
4605*  SET UP NEW REPEAT COUNT
4606  07310 121455        LDA FORMT,I
4607  07311 045455        ISZ FORMT
4608  07312 070253        SAP FSP2
4609  07313 121455        LDA FORMT,I
4610  07314 045455        ISZ FORMT
4611  07315 000132 FSP1   ADA M1
4612  07316 031454        STA RCOUT
4613* GET FORMAT SPEC: (FORMT) POINTS TO TYPE WORD
```

```
          #01  PRINT DISP AND WRITE COMMAND EXECUTION

4614  07317 121455 FSP2  LDA FORMT,I
4615  07320 050065       AND OPMSK      READ TYPE
4616  07321 025777       LDB RSPTR
4617  07322 004132       ADB M1
4618  07323 035401       STB RSAVE      SAVE ADDRESS OF RETURN
4619  07324 025455       LDB FORMT
4620  07325 035466       STB LOCL1
4621  07326 010116       CPA QTOP       QUOTE FIELD?
4622  07327 067340       JMP FSPQ       YES
4623  07330 121455       LDA FORMT,I
4624  07331 070202       SAR 5
4625  07332 050040       AND B37
4626  07333 070137       LDB A
4627  07334 105401       ADB RSAVE,I
4628  07335 004132       ADB M1
4629  07336 135401 BASE  STB RSAVE,I
4630  07337 170402       RET            RETURN TO P+(OP CODE)
4631  07340 121466 FSPQ  LDA LOCL1,I
4632  07341 050104       AND B177       GET CHAR COUNT
4633  07342 031467       STA LOCL2      SAVE
4634  07343 074070       SIB *+1
4635  07344 074744       SBL 1          COMPUTE STARTING CHARACTER ADDRESS
4636  07345 074017       ADA B
4637  07346 000132       ADA M1
4638  07347 070002       SAR 1
4639  07350 031466       STA LOCL1      COMPUTE AND SAVE ENDING WORD ADDRESS
4640  07351 145401       ISZ RSAVE,I    SET TO RETURN TO P+2
4641  07352 067357       JMP FSPR+1
4642*  END OF FORMAT STMT
4643  07353 025456 FSPN  LDB FORMS      RESET TO START OF STMT
4644  07354 035455       STB FORMT
4645  07355 170402       RET
4646*
4647*
4648*
4649  07356 031467 FSPR  STA LOCL2      SAVE (A)
4650  07357 021454       LDA RCOUT
4651  07360 072210       RZA FSPR1      END OF REPEAT COUNT?
4652  07361 021466       LDA LOCL1      YES, UPDATE
4653  07362 070070       SIA *+1
4654  07363 031455       STA FORMT        FORMAT POINTER
4655  07364 021467 FSPR1 LDA LOCL2      RESTORE (A)
4656  07365 170402       RET
4658*
4659*  SET UP FORMAT STMT POINTERS FOR WRITE OR ENTER
4660*
4661  07366 031640 SFMT  STA ST         SAVE FORMAT OP CODE
4662  07367 045472       ISZ TEMPS
4663  07370 061217       JSM FETCH      EVALUATE SELECT CODE
4664  07371 160225       JSM FLTRA,I    CONVERT TO INTEGER
4665  07372 160206 SCERR JSM ERRA,I     OVERFLOW
4666  07373 074117       LDA B
4667  07374 075710       SZB *-2
4668  07375 074142       SBR 4
4669  07376 077610       RZB *-4        CHECK FOR 1<SC<15
4670  07377 031453       STA PRFLG      SAVE SELECT CODE
4671  07400 070204       SAL 12
4672  07401 031447       STA SLCOD
4673  07402 035454 PLOTE STB RCOUT      SET REPEAT COUNT TO 0
4674  07403 125472       LDB TEMPS,I
4675  07404 045472       ISZ TEMPS
4676  07405 074152       SBM *+3        FORMAT SPECIFIED
4677  07406 045472       ISZ TEMPS
4678  07407 170402       RET            NO, RETURN P+1
4679  07410 121472       LDA TEMPS,I    YES
4680  07411 070673       SAP SFMT2,C
4681  07412 025455 SFMT1 LDB FORMT      ERROR IF STATEMENTS USING
4682  07413 074110       SZB *+2        FORMATS NESTED?
4683  07414 160206 FNEST JSM ERRA,I     YES
4684  07415 070137       LDB A
4685  07416 000017       ADA .2
4686  07417 031455       STA FORMT      SET CURRENT FMT SPEC POINTER
4687  07420 072053       SAP *+1,S      SET FLAG = - NO NUMERIC SPEC YET
4688  07421 031456       STA FORMS
4689  07422 061007       JSM NEXTA
4690  07423 035457       STB FORMF      SET POINTER TO END OF FMT STMT
4691  07424 045472       ISZ TEMPS
4692  07425 064357       JMP RETN2
4693*  FIND AND VERIFY FORMAT STMT
4694  07426 061075 SFMT2 JSM FNDPS      FIND STATEMENT
4695  07427 067430       JMP *+1
4696  07430 164343 NOFMT JMP E45-1,I    NOT FOUND
4697  07431 076053       SBP *+1,S      SET FLAG
4698  07432 135472       STB TEMPS,I    STORE ADDRESS IN PLACE OF STMT #
4699  07433 021476       LDA TEMP3
4700  07434 070070       SIA *+1
4701  07435 070537       LDB A,I        GET OP CODE
4702  07436 070070       SIA *+1
4703  07437 070517       LDA A,I        GET PAGE
```

```
          #01  PRINT: DISP AND WRITE COMMAND EXECUTION


4704  @7440 @50065        AND OPMSK
4705  @7441 @74442        SHR 10
4706  @7442 @70037        ADB A           COMBINE
4707  @7443 @21476        LDA TEMP3       RECALL STMT ADDRESS
4708  @7444 @15640        CPB ST          IS THIS A FORMAT STMT?
4709  @7445 @67412        JMP SFMT1       YES
4710  @7446 @67430        JMP NOFMT       NO. ERROR
4711*
4712  @7447 @12025 FMTOP OCT 12025

4714*    THIS ROUTINE TAKES A RECORD FROM
4715*    THE REFRESH BUFFER AND OUTPUTS IT
4716*    ON THE PRINTER.
4717*
4718*
4719  @7450 @21625 OLINE LDA FIRST
4720  @7451 @70076        TCA
4721  @7452 @01400        ADA BADDR       GET CHAR COUNT
4722  @7453 @70410        SZA OL1         BUFFER EMPTY?
4723  @7454 @63007        JSM PRCHK       PRINT?
4724  @7455 @67457        JMP *+2         YES
4725  @7456 @67464        JMP OL2         NO
4726  @7457 @25625        LDB FIRST       YES
4727  @7460 @61337        JSM NOWRT
4728  @7461 @21400 OL3    LDA BADDR
4729  @7462 @31625        STA FIRST
4730  @7463 170402 OL1    RET
4731  @7464 @24030 OL2    LDB .12
4732  @7465 @35705        STB FLOAT       SAVE COUNT
4733  @7466 163516 EW2    JSM REFRA,I
4734  @7467 @25443        LDB KEYFG       HAS A KEY BEEN ENTERED?
4735  @7470 @75453        SBP OL3
4736  @7471 @55705        DSZ FLOAT
4737  @7472 @67466        JMP EW2
4738  @7473 @67461        JMP OL3
4739*
4740*     THIS ROUTINE OUTPUTS THE CODE
4741*     IN A B TIMES
4742*
4743*
4744  @7474 @23041        LDA .42
4745  @7475 @74412 MOUT   SBM M01         COUNT NEGATIVE
4746  @7476 @74350        SZA M01         OR ZERO?
4747  @7477 @35740        STA TMP         NO. SAVE COUNT
4748  @7500 @31752        STA TMP&A
4749  @7501 @21752        LDA TMP&A
4750  @7502 @61016 M02    JSM ONEXT
4751  @7503 @55740        DSZ TMP
4752  @7504 @67501        JMP M02-1
4753  @7505 170402 M01    RET             YES
4755*
4756*  WAIT EXECUTION
4757*
4758  @7506 @61217 EWAIT  JSM FETCH       GET COUNT
4759  @7507 160226        JSM FLTFA,I     CONVERT TO INTEGER
4760  @7510 164172 EW1    JMP AEC4A,I     OVERFLOW
4761  @7511 @75752        SBM EW1
4762  @7512 @74142        SHR 4           DIVIDE BY 16
4763  @7513 @75650        SZB EW1
4764  @7514 @63465        JSM EW2-1       REFRESH DISPLAY
4765  @7515 164172        JMP AEC4A,I
4767  @7516 @02562 REFRA  DEF REFRS
4768  @7517 @02512 CRC1A  DEF CRC1
4769  @7520 174456 MSLPT  ABS -RBUFF-RBUFF
4770  @7521 @03433 LN72   DEF RBUFF+RBUFF+73
4771  @7522 @00116 N      OCT 116
4772  @7523 177670 M72    DEC -72
          #01  SYNTAX TABLES


4774*
4775*  STATEMENT SYNTAX JUMP TABLE
4776*
4777  @7524 174556        ABS RITE-*
4778  @7525 171422        ABS ENDS-*      GRAD
4779  @7526 171421        ABS ENDS-*      DEG
4780  @7527 171420        ABS ENDS-*      RAD
4781  @7530 174541        ABS LETS-*
4782  @7531 173760        ABS FMTS-*
4783  @7532 174554        ABS PRINS-*     DISP
4784  @7533 @06725        ABS READS-*     INPUT
4785  @7534 @06732        ABS RSTRS-*
4786  @7535 174551        ABS PRINS-*
4787  @7536 @06722        ABS READS-*
4788  @7537 173743        ABS DATAS-*
4789  @7540 @02741        ABS WAITS-*
4790  @7541 174070        ABS STOPS-*
4791  @7542 171405        ABS ENDS-*
4792  @7543 @02761        ABS RETNS-*
```

```
          #01  SYNTAX TABLES


4793  #7544 174011        ABS GOTOS-*   GOSUB
4794  #7545 #00147        ABS NEXTS-*
4795  #7546 #02714        ABS FORS-*
4796  #7547 #00151        ABS IFS-*
4797  #7550 174005        ABS GOTOS-*
4798  #7551 #06721        ABS REMS-*
4799  #7552 #02731        ABS DEFS-*
4800  #7553 #06660        ABS COMS-*
4801  #7554 #06702        ABS DIMS-*    DIM
4802  #7555 174446        ABS ILETS-*
4803*
4804*  STATEMENT NAME TABLE
4805*
4806  #7556 #00241 SYN13 OCT #0241      IMPLIED LET OP CODE 100001
4807  #7557 #46105       ASC 1,LE
4808  #7560 #52266       OCT 52266     LET: OP CODE 110110
4809  #7561 #42111       ASC 1,DI
4810  #7562 #46602       OCT #46602    DIM:#00010
4811  #7563 #41517       ASC 1,CO
4812  #7564 #46603       OCT #46603    COM:#00011
4813  #7565 #42105       ASC 1,DE
4814  #7566 #43204       OCT #43204    DEF:#00100
4815  #7567 #51105       ASC 1,RE
4816  #7570 #46645       OCT #46645    REM:100101
4817  #7571 #43517       ASC 2,GOTO
      #7572 #52117
4818  #7573 123111       OCT 123111    GOTO:100110
4819  #7574 #43247       OCT #43247    IF:100111
4820  #7575 #47105       ASC 2,NEXT
      #7576 #54124
4821  #7577 104507       OCT 104507    NEXT:001001
4822  #7600 #47523       ASC 2,OSUB
      #7601 #52502
4823  #7602 105122       OCT 105122    GOSUB: 001010
4824  #7603 #42524       ASC 2,ETUR
      #7604 #52522
4825  #7605 #47213       OCT #47213    RETURN: 001011
4826  #7606 #42516       ASC 1,EN
4827  #7607 #42254       OCT #42254    END:101100
4828  #7610 #51524       ASC 2,STOP
      #7611 #47520
4829  #7612 126527       OCT 126527    STOP:101101
4830  #7613 #40511       ASC 1,AI
4831  #7614 #52216       OCT #52216    WAIT:001110
4832  #7615 #42101       ASC 2,DATA
      #7616 #52101
4833  #7617 107522       OCT 107522    DATA:001111
4834  #7620 #42501       ASC 1,EA
4835  #7621 #42220       OCT #42220    READ: 010000
4836  #7622 #50122       ASC 2,PRIN
      #7623 #44516
4837  #7624 #52261       OCT #52261    PRINT:111001
4838  #7625 #42111       ASC 2,DISP
      #7626 #51520
4839  #7627 132111       OCT 132111    DISP: 110100
4840  #7630 #47120       ASC 2,NPUT
      #7631 #52524
4841  #7632 111506       OCT 111506    INPUT: 010011
4842  #7633 #47522       ASC 2,ORMA
      #7634 #46501
4843  #7635 #52225       OCT 52225     FORMAT: 010101
4844  #7636 #43117       ASC 1,FO
4845  #7637 #51210       OCT 51210     FOR: 001000
4846  #7640 #51105       ASC 3,RESTOR
      #7641 #51524
      #7642 #47522
4847  #7643 #42662       OCT 42662     RESTORE: 110010
4848  #7644 #51101       ASC 1,RA
4849  #7645 #42267       OCT 42267     RAD: 110111
4850  #7646 #42105       ASC 1,DE
4851  #7647 #43670       OCT 43670     DEG: 111000
4852  #7650 #43522       ASC 2,GRAD
      #7651 #40504
4853  #7652 134527       OCT 134527    GRAD: 111001
4854  #7653 #51111       ASC 2,RITE
      #7654 #52105
4855  #7655 175000       OCT 175000    WRITE: 111010

4857*  FETCH A STRING CONSTANT
4858*
4859*    START STRING IN NEXT WORD OF SYNTAX BUFFER, THEN RECORD
4860*      OPENING QUOTE AND THE STRING LENGTH.  EXIT WITH NEXT CHAR IN (A)
4861*    EXIT TO ERROR IF NO OPENING QUOTE
4862*    EXIT TO ERROR IF NO CLOSING QUOTE
4863*    ENTER AT GETST+3, IF OPENING QUOTE ALREADY FOUND
4864*
4865  #7656 #10043 GETST CPA #42       OPENING QUOTE?
4866  #7657 #67661       JMP *+2       YES
4867  #7660 160206 NOST  JSM ERRA,I    NO
4868  #7661 #25706       LDR SAPTR     SAVE INITIAL SYNTAX
```

```
        #01  SYNTAX


4869  @7662 @35470        STA L2T1        BUFFER POINTER
4870  @7663 @74742        SBR 16          SET TO START STORING IN LEFT HALF
4871  @7664 @63672        JSM CHRST       GET STRING
4872  @7665 160206 SYE14  JSM ERRA,I       NO CLOSING QUOTE
4873  @7666 @20116        LDA QTOP        STORE OPENING QUOTE
4874  @7667 @01474        ADA TEMP1         AND THE
4875  @7670 131470        STA L2T1,I          STRING LENGTH
4876  @7671 @65055        JMP GNEXT       GET NEXT CHAR AND RETURN
4877*
4878*  PROCESS A CHARACTER STRING
4879*
4880*     (A)=STRING TERMINATOR
4881*     (B)= CHARACTER POINTER; 1=START IN RIGHT HALF OF CURRENT WORD
4882*          0= START IN LEFT HALF OF NEXT WORD
4883*    EXIT:P+1 ON END OF LINE
4884*         P+2 ON TERMINATOR CHAR FOUND
4885*         (A)= NEXT CHARACTER
4886*         TEMP1= CHAR COUNT BIASED BY (B) ON ENTRY
4887*         SBPTR= NEXT WORD NOT IN STRING
4888*
4889  @7672 @31475 CHRST STA TEMP2      RECORD TERMINATOR CHARACTER
4890  @7673 @35474 CHRS1 STB TEMP1      SAVE CHARACTER FLAG AND COUNTER
4891  @7674 @61040        JSM GETCH-1   GET CHARACTER
4892  @7675 @10031        CPA EOL
4893  @7676 @67713        JMP CHRS3+1   EOL
4894  @7677 @11475        CPA TEMP2     TERMINATOR?
4895  @7700 @67712        JMP CHRS3     YES
4896  @7701 @25474        LDB TEMP1
4897  @7702 @74251        SLB CHRS2     SELECT WORD HALF
4898  @7703 141706        IOR SBPTR,I   RIGHT HALF, MERGE
4899  @7704 131706        STA SBPTR,I     WITH PREVIOUS CHAR
4900  @7705 @77330        RIB CHRS1     ADVANCE COUNT
4901  @7706 @67673        JMP CHRS1     JUMP ON @ COUNT
4902  @7707 @45706 CHRS2  ISZ SBPTR     LEFT HALF, ADVANCE POINTER
4903  @7710 @70404        SAL 8         POSITION CHAR
4904  @7711 @67704        JMP *-5
4905  @7712 @60365 CHRS3  JSM RET2      SET TO RETURN TO P+2
4906  @7713 @64456        JMP SBPUD     ADVANCE POINTER
4907*
4908*  NEXT STATEMENT SYNTAX
4909*
4910  @7714 160246 NEXTS  JSM VAR0A,I   FETCH SIMPLE VARIABLE
4911  @7715 164322        JMP E25+2,I
4912  @7716 164322        JMP E25+2,I   NOT FOUND
4913  @7717 @65151        JMP EOST
4914*
4915*  IF STATEMENT SYNTAX
4916*
4917  @7720 160202 IFS    JSM FSCA,I    GET DECISION FORMULA
4918  @7721 @27726        LDB THENA     DEMAND "THEN"
4919  @7722 @60546        JSM STEXP       AND STORE AS EXPANDED OPERATOR
4920  @7723 160206 SYNE7  JSM ERRA,I    NOT FOUND
4921  @7724 @60664        JSM LNUM      DEMAND A SEQUENCE NUMBER
4922  @7725 @65151        JMP EOST
4923  @7726 @06750 THENA  DEF FOP13+FOP13
4924*
4925*  INITIALIZE THE TRIG UNITS
4926*
4927  @7727 @24012 ERAD   LDB .1
4928  @7730 @35434        STB UNITS
4929  @7731 164172        JMP XEC4A,I
4930  @7732 @74742 EDEG   SBR 16
4931  @7733 @67730        JMP ERAD+1
4932  @7734 @24017 EGRAD  LDB .2
4933  @7735 @67730        JMP ERAD+1
4934*
4935*  FLOAT AND FIXED SYSTEM COMMAND EXECUTION
4936*  (A)=@
4937*
4938  @7736 @72051 FLT    SLA *+1,S     SET TO NON ZERO FOR FLOAT
4939  @7737 @31451 FIXED  STA CTYPE     SET TO ZERO FOR FIXED
4940  @7740 @61066        JSM GETSK     PARAMETER GIVEN?
4941  @7741 164174        JMP RDYDA,I   NO
4942  @7742 @60750        JSM DIGCK     DIGIT?
4943  @7743 @67740        JMP FIXED+1   NO, IGNOR IT
4944  @7744 @55400        DSZ BADDR     YES, RESET POINTER
4945  @7745 @24143        LDB M12
4946  @7746 @61131        JSM PGIN@     GET DIGIT FIELD, ALLOW @
4947  @7747 @35452        STB CPLAC
4948  @7750 164174        JMP RDYDA,I
4949*
4950*  TRACE SYSTEM COMMAND EXECUTION
4951*
4952  @7751 @24062 TRAC2  LDB FLGRT
4953  @7752 @35441        STB IFLG1+1   SET -(MAXIMUM UPPER LIMIT)
4954  @7753 @76330        RIB TRAC3     SET MAXIMUM (UPPER-LOWER LIMIT)
4955  @7754 @71650 TRACE  SZA TRAC2     ANY PARAMETERS?
4956  @7755 @60730        JSM GETAD     FIND ADDRESSES
4957  @7756 @74076        TCB
```

```
        #01  SYNTAX


4958*  NORMAL:  (B)=0; CLEAR TRACE FLAG
4959  07757 035441 NORM  STB TFLG1+1    STORE -(UPPER LIMIT)
4960  07760 005477       ADB TEMP4
4961  07761 074076 TRAC3 TCB
4962  07762 035440       STB TFLG1      STORE ( UPPER LIMIT - LOWER LIMIT)
4963  07763 164174       JMP RDYDA,I
4964*
4965*
4966*  EXECUTE READ
4967*
4968  07764 160201 EREAD JSM FORMA,I    GET VARIABLE ADDRESS IN HSTPT
4969  07765 160256       JSM FDAAA,I    FIND DATA AND SAVE IN AR2
4970  07766 060412       JSM RSULT      STORE DATA
4971  07767 021421       LDA PRADD
4972  07770 011472       CPA TEMPS      MORE VARIABLES TO COME?
4973  07771 164172       JMP XEC4A,I    NO: RETURN
4974  07772 067764       JMP EREAD      YES: LOOP

        #01  9830- PROGRAM EXECUTION


4976*
4977  10000              ORG 10000B
4978*
4979*  INITIALIZATION
4980*
4981  10000 020112 XEC   LDA RSTKA
4982  10001 031777       STA RSPTR      GUARANTEE EMPTY STACK
4983  10002 060653       JSM INITS+2    RESET STACK POINTERS
4984  10003 062162       JSM ESTAC      START NEW STACKS
4985  10004 031416       STA MODE       SET TO PROG (0)
4986  10005 021725 XEC3  LDA NXTST      GET STARTING STATEMENT NUMBER
4987  10006 061075       JSM FNDPS      FIND STARTING ADDRESS
4988  10007 066066       JMP EEND+1      BEYOND END OF PROG
4989  10010 066011       JMP *+1        BETWEEN LINES
4990  10011 035421 ERTN1 STB PRADD
4991*  PROCEED TO XEC4
4992*
4993*  EXECUTE NEXT PROGRAM STATEMENT
4994*
4995  10012 024062 XEC4  LDB FLGAT
4996  10013 021443       LDA KEYFG
4997  10014 050104       AND B177       MASK OFF SHIFT BIT
4998  10015 035443       STB KEYFG      RESET KEY FLAG
4999  10016 000145       ADA M14
5000  10017 072210       RZA *+4        TRACE KEY?
5001  10020 020066       LDA INF        YES, SET ALL TRACE
5002  10021 031440       STA TFLG1
5003  10022 066025       JMP XEC7
5004  10023 074742       SBR 16
5005  10024 010033       CPA B20        NORMAL KEY?
5006  10025 035441 XEC7  STB TFLG1+1    YES, CLEAR TRACE
5007  10026 025416       LDB MODE
5008  10027 074110       SZB *+2        KEYBOARD EXECUTION?
5009  10030 066070       JMP CALEX      YES, EXIT
5010  10031 025442       LDB STPFL
5011  10032 076150       RZB XEC5       SKIP IF STOP: STEP: OR ERROR
5012  10033 025421       LDB PRADD
5013  10034 015436       CPB HFLG1
5014  10035 066151 XEC5  JMP STOPF
5015  10036 015437       CPB HFLG1+1
5016  10037 066151       JMP STOPF      HALT IF ADDRESS = FLAG
5017  10040 015423 XEC6  CPB PBPTR      PAST END OF PROGRAM?
5018  10041 160206 MER11 JSM ERRA,I     YES, IMPROPER TERMINATION
5019  10042 005441       ADB TFLG1+1    ADD -(UPPER LIMIT)
5020  10043 074313       SBP XEC8       SKIP IF NOT IN TRACE LIMITS
5021  10044 005440       ADB TFLG1      ADD (UPPER LIMIT -(LOWER LIMIT))
5022  10045 074212       SBM XEC8
5023  10046 062136       JSM PLINE      OUTPUT LINE # TO BUFFER
5024  10047 061337       JSM NDWRT      PRINT ON PRINTER
5025  10050 061344       JSM CRLF1
5026  10051 060572 XEC8  JSM NEXTL-2    FIND OP-CODE AND PAGE
5027  10052 004135 XEC9  ADA M4      GET JUMP TABLE STARTING ADDRESS
5028  10053 005466       ADB LOCL1      ADD IN -(OP CODE)
5029  10054 074437       ADA B,I        ADD RELATIVE JUMP ADDRESS
5030  10055 074737       JMP B,I        JUMP TO STATEMENT EXECUTION
5031*
5032*  EXECUTE LET OR IMPLIED LET STATEMENT
5033*
5034  10056 026063 ELET  LDB DISPA      GET JUMP ADDRESS FOR DISP EXEC
5035  10057 021416       LDA MODE       GET CALC/PROG FLAG
5036  10060 073610       RZA ELET-2     SKIP IF CALC MODE
5037  10061 062563       JSM FORMX      PROG-  EVALUATE FORMULA
5038  10062 066012       JMP XEC4
5039  10063 013747 DISPA DEF DISP
5040*
5041*********************************
5042*
```

```
        #01  9830- PROGRAM EXECUTION


5043  10064 000000        OSS 1          CHECKSUM WORD
5044*
5045*********************************
5046*
5047*
5048*  END EXECUTION
5049*
5050  10065 061225 EEND  JSM LDEF
5051  10066 060653       JSM INITS+2    RESET POINTERS AND STACKS
5052  10067 066106       JMP ESTOP
5053*
5054*  EXIT FOR CALC MODE
5055*
5056  10070 021421 CALEX LDA PRADD
5057  10071 071652       SAM EEND+1     SKIP IF DECOMPILED
5058  10072 000020       ADA .3
5059  10073 031412       STA LSTPT      RESET STACK POINTER
5060  10074 060567       JSM UNSTK
5061  10075 060566       JSM UNSTK-1    RESTORE PROGRAM POINTER
5062  10076 031411       STA TSTPT      RESTORE T-STACK POINTER
5063  10077 060567       JSM UNSTK
5064  10100 031410       STA LSTAK      RESTORE
5065  10101 060567       JSM UNSTK
5066  10102 031412       STA LSTPT        L-STACK POINTERS
5067  10103 121413       LDA MSTPT,I
5068  10104 045413       ISZ MSTPT
5069  10105 071713       SAP *-2
5070  10106        ESTOP EQU *
5071  10106 160264 CALX2 JSM CRCKA,I    YES, PROVIDE CR-LF IF NEEDED
5072  10107 070742       SAR 16
5073  10110 031430       STA LNINC      GET OUT OF AUTO
5074  10111 164173       JMP PEXMA,I
5075*
5076*  INITIALIZATION FOR KEYBOARD EXECUTION
5077*
5078  10112 025412 ECAL  LDB LSTPT
5079  10113 035472       STB TEMPS      SAVE ORIGINAL STACK POINTER
5080  10114 020233       LDA SHUFA
5081  10115 111706 ECAL1 CPA SHPTR      TRANSFER SYNTAX
5082  10116 066122       JMP ECAL2        BUFFER INTO
5083  10117 070537       LDB A,I            STACK AREA
5084  10120 060563       JSM SLWST
5085  10121 073630       RIA ECAL1
5086  10122 025472 ECAL2 LDB TEMPS
5087  10123 060563       JSM SLWST      STACK OLD LSTPT
5088  10124 021412       LDA LSTPT      GET NEW END OF LINE POINTER
5089  10125 025410       LDB LSTAK
5090  10126 060563       JSM SLWST      STACK LSTPT
5091  10127 025411       LDB TSTPT
5092  10130 060563       JSM SLWST      STACK TSTPT
5093  10131 060562       JSM SLWST-1    STACK OLD PRADD
5094  10132 031421       STA PRADD      SET PRADD TO END OF BUFFER
5095  10133 062162       JSM ESTAC      SET UP NEW STACKS
5096  10134 060602       JSM NEXT1-4    GET PAGE AND OP CODE
5097  10135 066052       JMP XEC9
5098*
5099*  PRINT CURRENT LINE NUMBER FOR TRACE AND STOP
5100*
5101  10136 160264 PLINE JSM CRCKA,I    GIVE CRLF IF PRINT
5102  10137 160170       JSM CLEBA,I    CLEAR BUFFER
5103  10140 025421       LDB PRADD
5104  10141 015423       CPB PMPTR      PAST END OF PROG?
5105  10142 066146       JMP *+4        YES, DONT PRINT LINE #
5106  10143 125421       LDB PRADD,I
5107  10144 074073       SBP *+1,C
5108  10145 160210       JSM OUTTA,I    OUTPUT CURRENT LINE #
5109  10146 020053       LDA COLON
5110  10147 061416       JSM OUEXT      OUTPUT A :
5111  10150 064373       JMP ICNT       COMPUTE CHAR COUNT
5112*
5113*  STOP EXECUTION
5114*
5115  10151 074412 STOPE SHM STP1       SKIP IF ERROR MESSAGE
5116  10152 062136       JSM PLINE      OUTPUT LINE # AND COMPUTE COUNT
5117  10153 031466       STA LOCL1      SAVE COUNT
5118  10154 021450       LDA PMODE
5119  10155 072210       RZA STP1       PRINT IF PRINT-ALL
5120  10156 035701       STB NDISP
5121  10157 061332       JSM .NPWR+2
5122  10160 061344       JSM CRLF1
5123  10161 066106 STP1  JMP ESTOP      RETURN TO MONITOR
5124*
5125*
5126*  INITIALIZE STACKS FOR EXECUTION
5127*
5128  10162 025412 ESTAC LDB LSTPT      START AT OLD LSTPT
5129  10163 004134       ADB MJ
5130  10164 035411       STB TSTPT      OFFSET T-STACK POINTER BY 4
5131  10165 004033       ADB .16
```

#01 9830- PROGRAM EXECUTION

```
5132  10166 #35410        STB LSTAK      ALLOW 12 WORDS FOR T-STACK
5133  10167 #35412        STB LSTPT
5134  10170 #60553        JSM BHSTP
5135  10171 #74742        SBR 16
5136  10172 135410        STB LSTAK,I
5137  10173 035432        STB DROST
5138  10174 #24132        LDB M1
5139  10175 135413        STB HSTPT,I
5140  10176 164170        JMP CLERA,I    INITIALIZE THE PRINT BUFFER
5141*
5142*  EXECUTE  IF STATEMENT
5143*
5144  10177 #61217 EIF    JSM FETCH      EVALUATE FORMULA AND RETURN RESULT
5145  10200 #21755        LDA AR2+1      GET FIRST WORD OF MANTISSA
5146  10201 #70250        SZA EGOS4      SKIP IF FALSE
5147  10202 #62217 EGOS1  JSM EADD       GET ADDRESS
5148  10203 #25416        LDB MODE
5149  10204 #74150        RZB EGOS4+1    SKIP IF KEYBOARD MODE
5150  10205 #31421        STA PRADD      STORE IN PROGRAM POINTER
5151  10206 #66012 EGOS4  JMP XEC4
5152  10207 #25421        LDB PRADD
5153  10210 #04020        ADB .3         GET ADDRESS OF OLD PRADD
5154  10211 #35466        STB LOCL1
5155  10212 125466        LDB LOCL1,I    GET OLD PRADD
5156  10213 #74113        SBP *+2
5157  10214 #72053        SAP *+1,S      SAVE DECOMPILE FLAG
5158  10215 131466        STA LOCL1,I
5159  10216 #66012        JMP XEC4
5160*
5161*  SUBROUTINE TO GET ADDRESS
5162*
5163  10217 #45472 EADD   ISZ TEMPS
5164  10220 121472        LDA TEMPS,I    GET LINE # OR ADDRESS
5165  10221 #70372        SAM EADD1,C     ADDRESS FORMAT?
5166  10222 #61075        JSM FNDPS      NO, GET ADDRESS
5167  10223 #66224        JMP *+1
5168  10224 160206 NGOTO  JSM ERRA,I     NOT FOUND
5169  10225 #74117        LDA B          MOVE ADDRESS TO (A)
5170  10226 #76053        SBP *+1,S
5171  10227 135472        STB TEMPS,I    STORE COMPILED FORMAT
5172  10230 170402 EADD1  RET            ADDRESS IN (A)
5173*
5174*  EXECUTE RETURN
5175*
5176  10231 160206 ER3    JSM ERRA,I    INCORRECT RETURN STATEMENT
5177  10232 125412 ERTRN  LDB LSTPT,I   GET TOP OF RETURN STACK
5178  10233 #55412        ISZ LSTPT
5179  10234 #75650        SZB ER3        SKIP IF NO RETURN
5180  10235 #21472        LDA TEMPS
5181  10236 #70070        SIA *+1
5182  10237 #11421        CPA PRADD      RETURN A FORMULA?
5183  10240 #66243        JMP *+3        NO
5184  10241 #75413        SBP ER3        YES, FUNCTION RETURN ALLOWED?
5185  10242 #67106        JMP RETNN      YES, EXECUTE
5186  10243 #60771        JSM GTOPP      GET NEXT OPERAND
5187  10244 #73650        RZA *-3        RETURN AND SINGLE VARIABLE?
5188  10245 #75212        SBM ER3        NO, GOSUB RETURN ALLOWED?
5189  10246 #66011        JMP ERTN1      YES, SET RETURN ADDRESS
5190*
5191*      FOR STATEMENT
5192*
5193  10247 #11767 NOPCD  OCT 11767    ENCODED OP CODE FOR "NEXT"
5194*
5195  10250 #60771 EFOR   JSM GTOPP      GET FOR-VARIABLE
5196  10251 160177        JSM SSYMA,I        FIND IT IN SYMBOL TABLE
5197  10252 160252        JSM ENTEA,I     NOT FOUND - ENTER IT
5198  10253 #35474        STB TEMP1      SAVE ABS ADDRESS ALSO
5199  10254 #21407        LDA SYMTP
5200  10255 #70076        TCA
5201  10256 #70037        ADB A          COMPUTE RELATIVE ADDRESS
5202  10257 #20141        LDA M10        MOVE HSTPT TO LOWER
5203  10260 #01413        ADA HSTPT      ADDRESS TO ACCOMODATE
5204  10261 #31413        STA HSTPT      THE NEW FOR VAR.
5205  10262 #31476        STA TEMP3      SAVE IT FOR MVTOH IF NEEDED.
5206  10263 135413        STB HSTPT,I    SAVE RELATIVE SYMBOL TABLE POINTER
5207*
5208  10264 #60554        JSM BHSTP+1    OVERLAY?
5209*
5210  10265 #00061 EFOR2  ADA .10
5211  10266 #70537        LDB A,I       LOAD THE RELATIVE SYMBOL TABLE POINTER
5212  10267 #74512        SBM EFOR1      IF MINUS MEANS BOOTTOM OF HIGH STACK
5213  10270 #31475        STA TEMP2      SAVE THIS ADDRESS FOR MVTOH IF NEEDED
5214  10271 115413        CPB HSTPT,I    IS THIS F.V. THE SAME ?
5215  10272 #66274        JMP *+2        YES - GO DELETE IT
5216  10273 #66265        JMP EFOR2      NO - LOOP BACK FOR NEXT ONE
5217*
5218  10274 #00061        ADA .10        SET UP LAST WORD OF DESTINATION
5219  10275 #31477        STA TEMP4
5220  10276 #61121        JSM MVTOH      DELETE IT AND MOVE REST TO HIGHER MEMORY
```

```
        #01  9830- PROGRAM EXECUTION


5221  10277 025477        LDB TEMP4
5222  10300 035413        STB HSTPT     RESET HSTPT
5223*
5224  10301 160201 EFOR1  JSM FORMA,I
5225*
5226  10302 061371        JSM FIX       GET (A) AND (B) FIXED UP
5227  10303 060416        JSM STORE       STORE AR2 IN INITIAL VALUE TABLE
5228*
5229  10304 061217        JSM FETCH     GET ENDING LIMIT
5230  10305 025413        LDB HSTPT     TRANSFER END LIMIT
5231  10306 074070        STB *+1       TO HSTPT+1 THUR +4
5232  10307 060406        JSM XFAR1+1
5233*
5234  10310 020020        LDA ONE
5235  10311 060445        JSM XFAR2+1   SET AR2=1
5236  10312 021472        LDA TEMP5       IS THERE A STEP SIZE?
5237  10313 011421        CPA PRADD
5238  10314 066316        JMP *+2       NO, LEAVE 1 IN AR2
5239  10315 061217        JSM FETCH     YES - GET IT
5240  10316 025413        LDB HSTPT     TRANSFER STEP
5241  10317 004022        ADB .5        SIZE
5242  10320 060406        JSM XFAR1+1       TO HSTPT+4
5243*
5244  10321 004021        ADB .4        STORE NEXT
5245  10322 021421        LDA PRADD           EXECUTABLE
5246  10323 074557        STA B,I           LINE ADDRESS IN HSTPT+4
5247*
5248*   FIND THE CORRESPONDING NEXT STATEMENT (TEMP5 AND PRADD
5249*   WILL BE UP-DATED ACCORDINGLY). THEN CHECK IF THE "FOR"
5250*   LIMITS ARE CORRECT FOR LOOPING. IF NOT - ERASE FOR INFORMATION
5251*   AND JMP TO XEC4 TO CONTINUE AFTER THE "NEXT" STATEMENT.
5252*   IF THE "FOR" LOOP WAS VALID, LOAD HSTPT+4(THE ADDRESS FOLLOWING
5253*   THE "FOR" STATEMENT) AND EXCUTE FROM THERE(XEC4+1).
5254*
5255*
5256  10324 022247        LDA NUPCD       LOAD SERACH CODE
5257  10325 062413        JSM SYTSH       FIND A NEXT STATEMENT
5258  10326 074373        SBP BERR,C      B IS NEGATIVE IF FOUND
5259  10327 062405        JSM TEST        GO CHECK IF SAME VARIABLE
5260  10330 075613        SBP *-4       LOOP BACK UNLESS FOUND
5261*
5262  10331 061371        JSM FIX       GET A AND B REG FIXED FOR
5263  10332 063444        JSM OPCH1       FOR CONVERTING INT, SPLIT, OR FULL TO FULL
5264  10333 035475        STB TEMP2
5265  10334 066361        JMP FTEST       CHECK IF LOOP IS VALID
5266*
5267  10335 160206 BERR   JSM ERRA,I    NO CORRESPONDING "NEXT" OR INCORRECT NESTING
5268*
5269*     NEXT STATEMENT
5270*
5271*
5272  10336 125413 ENEXT  LDB HSTPT,I     RECOVER SYMBOL TABLE ADDRESS
5273  10337 075712        SBM BERR      ERROR BOTTOM OF STACK
5274  10340 005407        ADB SYMTP
5275  10341 035474        STB TEMP1     COMPUTE ARS ADDRESS OF VARIABLE
5276  10342 062405        JSM TEST
5277  10343 074252        SBM *+5       IF B IS -, GET OUT OF LOOP
5278  10344 020061        LDA .10       IF + THE VARIABLE WASN'T FOUND
5279  10345 001413        ADA HSTPT     SO DINC THE HSTPT
5280  10346 031413        STA HSTPT     AND LOOP BACK
5281  10347 066336        JMP ENEXT
5282*
5283*  TEMP1 IS POINTING AT SYMBOL IN SYMBLAOL TABLE
5284*  HSTPT+1 = ENDING LIMIT
5285*  HSTPT+5 = STEP SIZE
5286*
5287*
5288  10350 061371        JSM FIX       FIX A AND B FOR
5289  10351 063444        JSM OPCH1     CONVERT INT OR SPLIT TO FULL
5290  10352 021413        LDA HSTPT      ON RETURN (B) IS POINTING AT NUMBER
5291  10353 000022        ADA .5         LOAD (A) WITH STEP SIZE ADDRESS
5292  10354 160216        JSM .FADA,I   *
5293*
5294  10355 063454        JSM OPC1        SAVE AR2 AT 1470B
5295  10356 035475        STB TEMP2     SAVE 1470B IN TEMP2
5296*
5297  10357 061371        JSM FIX     FIX (A) AND (B)  FOR .XER
5298  10360 060416        JSM STORE       STORE ANSWER BACK IN VALUE TABLE
5299*
5300  10361 025413 FTEST  LDB HSTPT
5301  10362 074070        STB *+1         (B) = ADDRESS OF END LIMIT
5302  10363 021413        LDA HSTPT       GET SIGN
5303  10364 000022        ADA .5             OF STEP
5304  10365 070517        LDA A,I              SIZE
5305  10366 070151        SLA *+3
5306  10367 021475        LDA TEMP2     STEP SIZE IS -, F.V.-END LIMIT = AR2
5307  10370 066373        JMP *+3
5308  10371 074117        LDA B         STEP SIZE IS +, END LIMIT-F.V. = AR2
5309  10372 025475        LDB TEMP2
5310  10373 160217        JSM .FSBA,I
5311*
```

```
        #01  9830- PROGRAM EXECUTION


5312  10374 021413        LDA MSTPT      (A) IS POINTING AT NEXT
5313  10375 000026        ADA .9         EXECUTABLE LINE ADDRESS
5314  10376 025754        LDB AR2        TEST AR2 FOR + RESULT
5315  10377 074211        SLB *+4
5316*
5317  10400 070070        STA *,I       AR2 WAS NEGATIVE, ERASE
5318  10401 031413        STA MSTPT      "FOR" LOOP INFORMATION
5319  10402 066012        JMP XEC4         AND EXECUTE LINE AFTER "NEXT"
5320*
5321  10403 070537        LDB A,I        AR2 WAS POSITIVE OR 0,
5322  10404 066011        JMP EXTN1      EXECUTE STMT AFTER "FOR"
5324*
5325*    SUBROUTINES FOR "FOR" AND "NEXT" STATEMENTS.
5326*
5327  10405 061371 TEST   JSM FIX          RETURN WITH TYPE CODE OF F.V.
5328  10406 141472        IOR TEMPS,I      IN A.  IOR IN THE F.V. AND PAGE
5329  10407 050115        AND B1777        REMOVE PAGE.
5330  10410 111474        CPA TEMP1,I      DO THE VARIABLES AGREE?
5331  10411 076053        SOP **1,S       THEY WERE THE SAME, FLAG B
5332  10412 170402        RET              DID NOT AGREE B IS +.
5333*
5334*    CALLING SEQUENCE FOR SYTSH
5335*
5336*      LDA OP CODE     (PAGE-OP CODE)
5337*      JSM SYTSH     (TEMPS IS POINTING AT LINE #.)
5338*  ON RETURN
5339*         (B) = -, OP CODE FOUND  PRADD IS POINTING AT NEXT LINE #.
5340*                                   TEMPS IS POINTING AT PRADD+2
5341*             (B) =+    ERROR RETURN = OP CODE NOT FOUND
5342*
5343  10413 031475 SYTSH STA TEMP2      SAVE SEARCH CODE
5344  10414 025421        LDB PRADD
5345  10415 015423        CPB PHPTR      CHECK IF END OF PROGRAM
5346  10416 170402        RET            YES - RETURN WITH (B) +.
5347  10417 035472        STB TEMPS      NO - SET TEMPS = PRADD
5348  10420 060577        JSM NEXTL+3    LOOK AT THE NEXT STATEMENT.
5349  10421 001466        ADA LOCL1      ENCODE PAGE AND OP CODE
5350  10422 011475        CPA TEMP2      WAS IT = TO TEMP2(SEARCH CODE)?
5351  10423 077553        SOP *-S,S      YES, RETURN WITH (B) NEGATIVE
5352  10424 066414        JMP SYTSH+1    NO - LOOP BACK.

        #01  FORMULA EVALUATION ROUTINES


5354*
5355*
5356*  FIND THE NEXT DATA ITEM AND MOVE IT TO AR2
5357*    NXTDT MUST BE POINTING AT THE NEXT DATA ITEM OR
5358*    EQUAL TO DSTRT.  IF EQUAL TO DSTRT FDATA WILL SEARCH
5359*    FOR THE NEXT DATA STATEMENT.
5360*
5361  10425 062437 FDATA JSM SWITH     SAVE TEMPS, TEMPS = NXTDT
5362  10426 021421        LDA PRADD
5363  10427 031642        STA SAVE
5364  10430 025472        LDB TEMPS
5365  10431 015414        CPB DSTRT      IF NXTDT(TEMPS) AND DSTRT ARE =?
5366  10432 066445        JMP .FDA1      GO FIND NEXT DATA STATEMENT
5367  10433 160232        JSM SNELA,I    OR CONVERT SYNTAX TO FULL
5368  10434 045472        ISZ TEMPS
5369  10435 021642        LDA SAVE
5370  10436 031421        STA PRADD
5371  10437 021472 SWITH  LDA TEMPS
5372  10440 025415        LDB NXTDT
5373  10441 031415        STA NXTDT
5374  10442 035472        STB TEMPS
5375  10443 170402        RET
5376*
5377  10444 160206 ER60   JSM ERRA,I     OUT OF DATA
5378  10445 035421 .FDA1  STB PRADD
5379  10446 022454        LDA DATCD      LOAD "DATA" STATEMENT SEARCH CODE
5380  10447 062413        JSM SYTSH      FIND NEXT DATA STATEMENT
5381  10450 075613        SOP ER60       IF (B) WAS +, NO MORE DATA FOUND
5382  10451 025421        LDB PRADD
5383  10452 035414        STB DSTRT
5384  10453 066430        JMP FDATA+3  DATA WAS FOUND GO CHECK IF ANY DATA WITH IT
5385*
5386  10454 011761 DATCD  OCT 11761
5388*
5389*  EXCUTE INPUT   *
5390*
5391  10455 021622 FINPT  LDA TERM
5392  10456 072110        RZA **2
5393  10457 160170        JSM CLERA,I
5394  10460 021472        LDA TEMPS
5395  10461 000133        ADA M2
5396  10462 031641        STA SPRDD      SAVE START OF INPUT STATEMENT
5397  10463 021641        LDA SPRDD      SET PRADD TO START OF
5398  10464 031421        STA PRADD        INPUT STATEMENT
5399  10465 020262        LDA INREA      SET UP RETURN
5400  10466 031432        STA DRQST      SET DATA REQUEST FLAG
5401  10467 164173 INRET  JMP PEXMA,I    CALL FOR DATA
5402  10470 055400        DSZ BADDR
```

#01 FORMULA EVALUATION ROUTINES

```
5403  10471 025641        LDB SPRDD
5404  10472 061007        JSM NEXTA       SET PRADD BACK TO END
5405  10473 035421        STB PRADD         OF INPUT STATEMENT
5406  10474 021472        LDA TEMPS
5407  10475 031640        STA TEPPP       SAVE START OF VARIABLE
5408  10476 160201 .EIN1  JSM FORMA,I     GET THE ADDRESS OF THE VARIABLE IN HSTPT
5409  10477 024017        LDB .2
5410  10500 035706        STB SBPTR       AIM SBPTR AT ROM
5411  10501 026532        LDB INCHA
5412  10502 035432        STB DROST       SET FLAG IN CASE OF ERROR
5413  10503 061156        JSM CONST       BUILD A NUMBER IN AR2
5414  10504 066520        JMP INCHT-1     NO NUMBER FOUND, RESTORE H-STACK
5415  10505 070742        SAR 16
5416  10506 031432        STA DROST       CLEAR FLAG
5417  10507 060412        JSM RSULT       SAVE AR2 AT HSTPT,I
5418  10510 025472        LDB TEMPS
5419  10511 015421        CPB PRADD       END OF STATEMENT?
5420  10512 066530        JMP CXEC4       YES
5421  10513 035640        STB TEPPP       RESET THE TEMPORARY POINTER
5422  10514 055400        DSZ BADDR       NO - DINC BADDR TO LOOK AT THE LAST CHAR IN BUFF.
5423  10515 061066        JSM GETSK       GET LAST CHAR.
5424  10516 066526        JMP INCH1
5425  10517 066476        JMP .EIN1       P+2 RETURN - OTHER CHARS
5426*
5427  10520 160170        JSM CLERA,I
5428  10521 021640 INCHT  LDA TEPPP
5429  10522 031472        STA TEMPS       RESTORE VARIABLE POINTER
5430  10523 045413        ISZ HSTPT
5431  10524 045413        ISZ HSTPT
5432  10525 066463        JMP INRET-4
5433*
5434  10526 160170 INCH1  JSM CLERA,I
5435  10527 066463        JMP INRET-4
5436*
5437  10530 160170 CXEC4  JSM CLERA,I
5438  10531 066012        JMP XEC4
5439*
5440  10532 010521 INCHA  DEF INCHT
5441*
5442*  EXECUTE RESTORE STATEMENT
5443*
5444  10533 125472 ERSTR  LDB TEMPS,I
5445  10534 021424        LDA PBUFF
5446  10535 074113        SBP *+2         LINE GIVEN?
5447  10536 062217        JSM EADD        YES, GET ADDRESS
5448  10537 031415        STA NXTDT       SET UP DATA POINTERS
5449  10540 031414        STA DSTRT
5450  10541 066012        JMP XEC4
5452*
5453*  EVALUATE ARGUEMENT OF FUNCTION
5454*
5455  10542 024032 EVARG  LDB B17         INITIALIZE OPERATOR STACK
5456  10543 066564        JMP FORMX+1       FOR FUNCTION
5457*
5458*  EVALUATE A FORMULA
5459*
5460*
5461*  STACK A CONSTANT
5462  10544 050040 FORM6  AND B37
5463  10545 010040        CPA B37         PRE-DEFINED FUNCTION?
5464  10546 066664        JMP FORM7       YES
5465  10547 055413        DSZ HSTPT       BUMP STACK POINTER
5466  10550 063406        JSM HSCHK       ALLOT SPACE ON T-STACK
5467  10551 160232        JSM SNFLA,I     CONVERT TO FLOATING
5468  10552 045472 FORST  ISZ TEMPS
5469  10553 074742        SBR 16
5470  10554 135413        STB HSTPT,I     STORE FULL PRECISION TYPE
5471  10555 060553        JSM BHSTP
5472  10556 025411        LDB TSTPT
5473  10557 135413        STB HSTPT,I     STORE ADDRESS OF T-STACK
5474  10560 020214        LDA AR2A
5475  10561 170004        XFR             TRANSFER RESULT TO T-STACK
5476  10562 066616 FOR2L  JMP FORM2
5477*
5478*
5479  10563 074742 FORMX  SBR 16          INITIALIZE
5480  10564 060563        JSM SLWST         OPERATOR STACK
5481*  PROCESS NEXT OPERAND
5482  10565 121472 FORM1  LDA TEMPS,I     FETCH OPERAND
5483  10566 050070        AND OPDMK
5484  10567 070113        SAP *+2
5485  10570 066544        JMP FORM6       CONSTANT OR FUNCTION
5486  10571 045472        ISZ TEMPS       SET FOR NEXT WORD OF FORMULA
5487  10572 071410        SZA FOR2L       SKIP IF NULL OPERAND
5488  10573 055413        DSZ HSTPT       BUMP POINTER
5489  10574 025420        LDB PARMN       GET PARAMETER NAME
5490  10575 074504        SBL 6
5491  10576 074242        SBR 6           MASK OFF PRECISION
5492  10577 074057        CPA B           IS THIS OPERAND THE PARAMETER
5493  10600 067116        JMP FPARM       YES
```

#01 FORMULA EVALUATION ROUTINES

```
5494*  STACK NON-FUNCTION VARIABLE OPERAND ADDRESS
5495  10601 160177        JSM SSYMA,I    SEARCH FOR VARIABLE IN SYMTAB
5496  10602 160252        JSM ENTEA,I    NOT FOUND, ENTER IT
5497  10603 074517        LDA B,I
5498  10604 050040        AND B37        FETCH TYPE
5499  10605 004132        ADB M1
5500  10606 035750        STB YC         SAVE ADDRESS-ADDRESS
5501  10607 125750        LDB YC,I       READ ADDRESS
5502  10610 010040        CPA B37
5503  10611 066725        JMP FORM8      IT WAS A USER DEFINED FUNCTION!
5504  10612 131413 FORMP  STA HSTPT,I    SAVE TYPE ON H-STACK
5505  10613 074117        LDA B          GET ADDRESS OF OPERAND
5506  10614 060553        JSM BHSTP      BUMP POINTER
5507  10615 131413        STA HSTPT,I    STORE ADDRESS ON OPERAND STACK
5508*  PROCESS NEXT OPERATOR
5509  10616 021472 FORM2  LDA TEMPS      END OF
5510  10617 011421        CPA PRADD        STATEMENT?
5511  10620 066631        JMP FORM3      YES, FORCE END OF FORMULA
5512  10621 121472        LDA TEMPS,I    EXTRACT
5513  10622 070073        SAP **1,C
5514  10623 070442        SAR 10             NEXT
5515  10624 070137        LDB A                OPERATOR
5516  10625 000135        ADA M4         OP CODE LESS THAN 4?
5517  10626 070152        SAM FORM3      IF YES, NON FORMULA OPERATOR
5518  10627 000147        ADA M23        NO, GREATER THAN 32?
5519  10630 070112        SAM **2        NON-FORMULA OPERATOR?
5520  10631 024012 FORM3  LDB .1         YES, USE CODE FOR QUOTE
5521  10632 035460        STB FTMP1      SAVE CURRENT OP-CODE
5522  10633 004123        ADB FOPBS      ADD OP CODE TO TABLE ADDRESS
5523  10634 074517        LDA B,I        GET AND MASK
5524  10635 050040        AND B37          PRECEDENCE
5525  10636 031461        STA FTMP2      SAVE PRECEDENCE OF CURRENT OP
5526*  STACK PRESENT OR EXECUTE PREVIOUS OPERATOR
5527  10637 121412 FORM4  LDA LSTPT,I    DOES OPERATOR ON
5528  10640 050040        AND B17          TOP OF OPERATOR
5529  10641 070056        CMA                STACK HAVE
5530  10642 001461        ADA FTMP2            HIGHER
5531  10643 070412        SAM FORM5              PRECEDENCE?
5532  10644 025461 FORM9  LDB FTMP2      NO, STACK CURRENT OPERATOR
5533  10645 014033        CPB B20        LEFT PAREN OR BRACKET?
5534  10646 024012        LDB .1         YES, STACK PRECEDENCE OF 1
5535  10647 021460        LDA FTMP1      COMBINE
5536  10650 070304        SAL 10           PRECEDENCE
5537  10651 070037        ADB A              WITH OP-CODE
5538  10652 066564        JMP FORM1-1      AND STACK IT
5539  10653 121412 FORM5  LDA LSTPT,I
5540  10654 055412        DSZ LSTPT
5541  10655 070442        SAR 10         COMPUTE JUMP ADDRESS
5542  10656 003135        ADA ARBAS        FOR OPERATOR
5543*  AT THIS POINT (HSTPT,I)= ADDRESS OF THE TOP OPERAND
5544*    (HSTPT+1,I)= THE TYPE OF THE TOP OPERAND
5545  10657 070737        JMP A,I        BRANCH TO ROUTINE
5546*
5547*  EVALUATE NON-ARGUEMENT FUNCTION
5548  10660 055413 FOR72  DSZ HSTPT
5549  10661 060553        JSM BHSTP      PLACE DUMMY ON STACK
5550  10662 045472        ISZ TEMPS      POINT TO NEXT OPERATOR
5551  10663 066707        JMP FOR71
5552*  EVALUATE PRE-DEFINED FUNCTION
5553  10664 025777 FORM7  LDB RSPTR
5554  10665 004132        ADB M1         TAKE RETURN ADDRESS OF
5555  10666 074537        LDB B,I          FORMX OFF STACK AND
5556  10667 055777        DSZ RSPTR
5557  10670 060563        JSM SLWST          SAVE ON LOW STACK
5558  10671 125472        LDB TEMPS,I    GET OP CODE
5559  10672 074544        SHL 5
5560  10673 060604        JSM NEXTI-2    CONFIGURE OP CODE AND GET PAGE
5561  10674 004134        ADB M3
5562  10675 074437        ADB B,I        GET TABLE STARTING ADDRESS
5563  10676 005466        ADB LOCL1      ADD -(OP CODE)
5564  10677 074437        ADB B,I        GET ACTUAL JUMP ADDRESS
5565  10700 060563        JSM SLWST      STACK JUMP ADDRESS
5566  10701 021466        LDA LOCL1
5567  10702 000036        ADA .23
5568  10703 070113        SAP **2        ARGUEMENT REQUIRED?
5569  10704 066660        JMP FOR72      NO
5570  10705 062542        JSM EVARG      EVALUATE ARGUEMENT
5571  10706 063433        JSM OPCHK      VERIFY RESULT
5572  10707 063405 FOR71  JSM HSCHK-1    ALLOT SPACE ON T-STACK
5573  10710 125412        LDB LSTPT,I
5574  10711 055412        DSZ LSTPT      REMOVE ADDRESS FROM STACK
5575  10712 074717        JSM B,I        BRANCH TO ROUTINE
5576*  RECORD RESULT OF FUNCTION
5577*
5578*  FUNCTIONS RETURN HERE WITH RESULT IN AR2
5579  10713 045413 FOR10  ISZ HSTPT      POP ARGUEMENT BUT LEAVE STACK BUMPED 1
5580  10714 121412        LDA LSTPT,I    UNSTACK RETURN ADDRESS
5581  10715 055412        DSZ LSTPT
5582  10716 025777        LDB RSPTR      GET RETURN ADDRESS
```

```
        #01  FORMULA EVALUATION ROUTINES

5583  10717 @45777          ISZ RSPTR
5584  10720 @70173          SAP **3,C       MORE ADDRESSES TO COME?
5585  10721 @74557          STA B,I         YES
5586  10722 @66714          JMP FOR10+1     GET NEXT RETURN ADDRESS
5587  10723 @74557          STA B,I         NO
5588  10724 @66553          JMP FORST+1
5589*  EVALUATE PROGRAMMER DEFINED FUNCTION
5590  10725 @76110 FORMA    RZB **2         ADDRESS =0?
5591  10726 160222          JSM ENTFA,I     YES, FIND FUNCTION
5592  10727 @74742          SBR 16          FLAG FIRST RETURN, SIGN=0
5593  10730 @21777 FMXA0    LDA RSPTR
5594  10731 @00132          ADA M1
5595  10732 @74437          ADB A,I       COMBINE RETURN ADDRESS WITH FLAG
5596  10733 @55777          DSZ RSPTR
5597  10734 @60563          JSM SLWST       STACK ON L-STACK
5598  10735 @21777          LDA RSPTR
5599  10736 @10112          CPA RSTKA       ALL OF STACK MOVED?
5600  10737 @66742          JMP **3         YES
5601  10740 @24062          LDB FLGBT
5602  10741 @66730          JMP FMXA0
5603  10742 @20132          LDA M1
5604  10743 131413          STA HSTPT,I     FLAG STACK FOR "FORM LOOPS
5605  10744 @25411          LDB TSTPT
5606  10745 @60563          JSM SLWST       SAVE T-STACK POINTER
5607  10746 125750          LDB TC,I
5608  10747 @60563          JSM SLWST       SAVE FUNCTION ADDRESS
5609  10750 @62542          JSM EVARG       EVALUATE ARGUEMENT
5610  10751 @21472          LDA TEMPS       SWITCH FORMULA
5611  10752 125412          LDB LSTPT,I       POINTER TO FUNCTION FORMULA
5612  10753 @35472          STB TEMPS
5613  10754 131412          STA LSTPT,I
5614  10755 @23100          LDA DSTRA       MOVE LSTPT UP 9 WORDS
5615  10756 @24026          LDB .9           AND TRANSFER DSTRT
5616  10757 @63122          JSM MOVST            THROUGH PBUFF
5617  10760 170004          XFR                    TO L-STACK
5618  10761 121413          LDA HSTPT,I     GET PARAMETER ADDRESS
5619  10762 @31417          STA PARMA
5620  10763 @11411          CPA TSTPT       IS PARAMETER ON T-STACK?
5621  10764 @63406          JSM RSCHK        YES, PROTECT IT
5622  10765 @25423          LDB PHPTR
5623  10766 @74076          TCB
5624  10767 @05472          ADB TEMPS
5625  10770 @74313          SBP FMXA2       SKIP IF OUTSIDE PRESENT PROGRAM
5626  10771 @25424          LDB PBUFF
5627  10772 @74076          TCB
5628  10773 @05472          ADB TEMPS
5629  10774 @74112          SBM FMXA2       SKIP IF OUTSIDE PRESENT PROGRAM
5630  10775 @67016          JMP FMXA1
5631  10776 @61225 FMXA2    JSM LUFF        INSURE NOT IN KEY MODE
5632  10777 @24166          LDB FWAM        FIND PROGRAM CONTAINING FUNCTION
5633  11000 @15405 FMXA4    CPB FWUP        END OF KEYS?
5634  11001 @67013          JMP FMXA3       YES, MUST BE MAINLINE
5635  11002 @74070          SIB **1         POINT TO LENGTH WORD
5636  11003 @35474          STB TEMP1       SAVE START
5637  11004 @74437          ADB B,I         COMPUTE END
5638  11005 @74117          LDA B
5639  11006 @70076          TCA
5640  11007 @01472          ADA TEMPS
5641  11010 @71413          SAP FMXA4       FUNCTION IN THIS KEY?
5642  11011 @25474          LDB TEMP1       YES
5643  11012 160263          JSM SUMPA,I     YES, SET UP PROGRAM POINTERS
5644  11013 @25424 FMXA3    LDB PBUFF       RESET DATA STMT POINTERS
5645  11014 @35414          STB DSTRT
5646  11015 @35415          STB NXTDT
5647  11016 @70742 FMXA1    SAR 16
5648  11017 @31416          STA MODE        SET TO PROGRAM MODE
5649  11020 @25472          LDB TEMPS
5650  11021 @61007          JSM NEXTA
5651  11022 @35421          STB PRADD       SET POINTER TO END OF LINE
5652  11023 @45413          ISZ HSTPT
5653  11024 125413          LDB HSTPT,I
5654  11025 @74244          SHL 11          GET PRECISION OF PARAMETER
5655  11026 @45413          ISZ HSTPT       DINK PARAMETER OFF STACK
5656  11027 @21472          LDA TEMPS
5657  11030 @00020          ADA .3
5658  11031 @70517          LDA A,I         FETCH PARAMETER NAME
5659  11032 @50115          AND B1777
5660  11033 @74217          IOR B           MERGE WITH PRECISION
5661  11034 @31420          STA PARMN
5662  11035 @25472          LDB TEMPS
5663  11036 @04022          ADB .5        POINT PAST PARAMETER
5664  11037 @15421          CPB PRADD       MULTI-LINE FUNCTION?
5665  11040 @67102          JMP EVMLF       YES
5666  11041 @35472          STB TEMPS       NO
5667*
5668  11042 @62563 FMX9     JSM FORMX       EVALUATE FUNCTION FORMULA
5669  11043 121412          LDA LSTPT,I     FETCH OLD PBUFF
5670  11044 @27076          LDB PBUFA
5671  11045 @35477          STB TEMP4       INITIALIZE TRANSFER POINTER
5672  11046 @25412          LDB LSTPT
```

#01 FORMULA EVALUATION ROUTINES

```
5673  11047 074070        SIB *+1
5674  11050 035475        STB TEMP2
5675  11051 004141        ADB M10
5676  11052 035412        STB LSTPT      RESET STACK POINTER
5677  11053 074070        SIB *+1
5678  11054 011424        CPA PBUFF      WAS A KEY ENTERED?
5679  11055 004017        ADB .2         NO, DONT RESTORE DATA POINTERS
5680  11056 035476        STB TEMP3
5681  11057 061121        JSM MVTOH      RESTORE PROGRAM POINTERS
5682  11060 121412        LDA LSTPT,I
5683  11061 055412        DSZ LSTPT
5684  11062 031472        STA TEMP5      RESTORE PROGRAM POINTER
5685  11063 121412        LDA LSTPT,I
5686  11064 055412        DSZ LSTPT
5687  11065 031411        STA TSTPT      RESTORE T-STACK POINTER
5688  11066 063404        JSM SITOP      GET ADDRESS OF RESULT
5689  11067 024214        LDB AR2A
5690  11070 170004        XFR            TRANSFER
5691  11071 121413        LDA HSTPT,I    REMOVE ANY UNTERMINATED
5692  11072 010132        CPA M1         "FOR" INFORMATION
5693  11073 066714        JMP FOR10+1
5694  11074 045413        ISZ HSTPT
5695  11075 067071        JMP *-4
5696  11076 001425 PBUFA  DEF PBUFF+1
5697  11077 177771 M7     DEC -7
5698  11100 001414 DSTRA  DEF DSTRT
5699  11101 001640 STADD  DEF ST
5700*
5701*EVALUATE MULTI-LINE FUNCTION
5702  11102 027101 EVMLF  LDA STADD      TRANSFER FIRST 7 WORDS
5703  11103 024025        LDB .8           OF EXECUTION TEMPRORARY AREA
5704  11104 063122        JSM MOVST          TO L-STACK
5705  11105 066012        JMP XEC4
5706*
5707  11106 021412 RETNN  LDA LSTPT
5708  11107 003077        ADA M7         RESTORE EXECUTION
5709  11110 031412        STA LSTPT
5710  11111 074070        SIA *+1
5711  11112 027101        LDB STADD        TEMPRORARIES
5712  11113 170004        XFR
5713  11114 170004        XFR
5714  11115 067042        JMP FMX9
5715*
5716*  SET UP ADDRESS AND TYPE FOR PARAMETER
5717  11116 021420 FPARM  LDA PARMN      PRECISION IS STORED IN
5718  11117 074502        SAR 11           UPPER BITS
5719  11120 025417        LDB PARMA
5720  11121 066612        JMP FORMP
5721*
5722*  TRANSFER WORDS TO LOW STACK
5723*  (A)= TRANSFER STARTING ADDRESS
5724*  (B)= # OF WORDS
5725*
5726  11122 035467 MOVST  STB LOCL2      SAVE COUNT
5727  11123 025412        LDB LSTPT      SAVE OLD LSTPT
5728  11124 035466        STB LOCL1
5729  11125 005467        ADB LOCL2      ADD COUNT TO POINTER
5730  11126 035412        STB LSTPT
5731  11127 060554        JSM BHSTP+1    CHECK FOR OVERFLOW
5732  11130 135412        STB LSTPT,I    STACK NEGATIVE FLAG FOR "RETURN" STMT
5733  11131 025466        LDB LOCL1
5734  11132 074070        SIB *+1
5735  11133 170004        XFR            TRANSFER 4 WORDS
5736  11134 064446        JMP XFAR2+2    TRANSFER 3 MORE AND RETURN
5738*
5739*  FORMULA OPERATOR JUMP TABLE
5740*
5741  11135 111136 ARRAS  DEF *+1,I
5742  11136 011432        DEF FORMR      EXECUTE END OF FORMULA
5743*
5744*  THE FOLLOWING 3 TABLE WORDS ARE NOT USED
5745*
5746*  QUOTE, COMMA, AND SEMI-COLON ARE END OF FORMULA OPERATORS
5747*
5748*  CALL SUBTRACT
5749*
5750  11137 063376 EFSB   JSM BINOP
5751  11140 160217        JSM .FSBA,I
5752  11141 067260        JMP FOR14
5753*
5754  11142 011432        DEF FORMR      *)*
5755  11143 011432        DEF FORMR      *)*
5756  11144 011202        DEF ESCMA      SUBSCRIPT COMMA
5757  11145 011171        DEF ESTR       ASSIGNMENT EQUAL
5758  11146 000000   NOP   ************CHECKSUM WORD ************
5759  11147 011256        DEF EFAD
5760  11150 011137        DEF EFSB
5761  11151 011272        DEF EFMP
5762  11152 011275        DEF EFDV
5763  11153 011300        DEF EPWR
```

```
        #01  FORMULA EVALUATION ROUTINES


5764  11154 @11333        DEF EGTRT
5765  11155 @11336        DEF ELST
5766  11156 @11352        DEF ENEQL
5767  11157 @11344        DEF EEQI        LOGICAL EQUAL
5768  11160 @11383        DEF EUMIN       UNARY MINUS
5769  11161 @10637        DEF FORM4       UNARY PLUS
5770  11162 @11317        DEF ELARC       *[*
5771  11163 @10565        DEF FORM1       *(*
5772  11164 @11357        DEF EOR
5773  11165 @11346        DEF EAND
5774  11166 @11716        DEF ENOT
5775  11167 @11341        DEF EGORF
5776  11170 @11347        DEF ELORF       *<=*
5778*
5779*  EXECUTE ASSIGNMENT EQUAL
5780*
5781  11171 @25461 ESTR   LDB FTMP2       IS NEXT OPERATOR AN
5782  11172 @74150        SZB *+3           END OF FORMULA?
5783  11173 @45412        ISZ LSTPT       NO* MULTIPLE STORE--RESTORE
5784  11174 @66644        JMP FORM9         STORE OPERATOR AND DEFER STORE
5785  11175 @55460        DSZ FTMP1       FIRST STORE EXECUTED?
5786  11176 @67200        JMP ESTR1       NO SOURCE IS ALREADY IN AR2
5787  11177 @61220        JSM FETCH*I     POP STACK AND MOVE SOURCE TO AR2
5788  11200 @60412 ESTR1  JSM RSULT       POP DESTINATION AND STORE
5789  11201 @66637        JMP FORM4
5790*
5791* EXECUTE SUBSCRIPT COMMA
5792*
5793  11202 @61220 ESCMA  JSM FETCH*I     CHECK COLUMN SUBSCRIPT
5794  11203 @60225        JSM FLTRA,I     ROUND TO INTEGER
5795  11204 @67240        JMP ER6         OVERFLOW
5796  11205 @04132        ADB M1          ADD OFFSET
5797  11206 @75712        SBM *-2         UNDERFLOW?
5798  11207 @35412        STB LSTPT,I     SAVE COLUMN SUBSCRIPT
5799  11210 @61220        JSM FETCH*I     CHECK ROW SUBSCRIPT
5800  11211 @60225        JSM FLTRA,I     ROUND TO INTEGER
5801  11212 @67240        JMP ER6
5802  11213 @04132        ADB M1
5803  11214 @75712        SBM *-2
5804  11215 @35471        STB L2T2      SAVE ROW SUBSCRIPT
5805  11216 @25413        LDB HSTPT,I       GET ACTUAL BOUNDS
5806  11217 @04132        ADB M1
5807  11220 @74517        LDA B,I
5808  11221 @50105        AND B377          EXTRACT COLUMN BOUND
5809  11222 @70070        SIA *+1
5810  11223 @31470        STA L2T1          SAVE IT
5811  11224 @74537        LDB B,I         EXTRACT ROW BOUND
5812  11225 @74342        SBR 8
5813  11226 @74070        SIB *+1
5814  11227 @74076        ICB
5815  11230 @05471        ADB L2T2
5816  11231 @74353        SBP ER6         ROW SUBSCRIPT LEGAL?
5817  11232 @25471        LDB L2T2        YES* RECALL ROW SUBSCRIPT
5818  11233 @61201        JSM IMPY        COMPUTE ROW DISPLACEMENT
5819  11234 @25470        LDB L2T1
5820  11235 @74076        ICB
5821  11236 @05412        ADB LSTPT,I       COLUMN SUBSCRIPT LEGAL?
5822  11237 @74112        SBM *+2
5823  11240 @60206 ER6    JSM ERRA,I      NO*ERROR
5824  11241 @01412        ADA LSTPT,I       YES* ADD IN COLUMN DISPLACEMENT
5825  11242 @25413        LDB HSTPT         FIND TYPE OF ARRAY
5826  11243 @74070        SIB *+1
5827  11244 @74537        LDB B,I
5828  11245 @74042        SBR 2           FIND # OF WORDS PER ELEMENT
5829  11246 @76110        RZB *+2         FULL PRECISION?
5830  11247 @70704        SAL 2           YES* MULTIPLY BY 4
5831  11250 @14012        CPB ,1           SPLIT?
5832  11251 @70744        SAL 1           YES* MULTIPLY BY 2
5833  11252 @01413        ADA HSTPT,I
5834  11253 @31413        STA HSTPT,I     STORE ACTUAL ADDRESS OF ELEMENT
5835  11254 @55412        DSZ LSTPT         UNSTACK I
5836  11255 @66565        JMP FORM1
5837*
5838*  CALL ADD
5839*
5840  11256 @63376 EFAD   JSM BINOP
5841  11257 @60216        JSM .FADA,I
5842*  PROCEED TO FOR14
5843*
5844*  OPERATORS CREATING INTERMEDIATE RESULTS RETURN HERE
5845*    WITH RESULT IN AR2
5846*
5847  11260 @20214 FOR14  LDA AR2A
5848  11261 @25411        LDB TSTPT       TRANSFER TO T-STACK
5849  11262 @70004        XFR
5850  11263 @21413        LDA HSTPT
5851  11264 @70070        SIA *+1         POINT TO PRECISION WORD
5852  11265 @74742        SBR 16
5853  11266 @70577        STB A,I         STORE FULL PRECISION
5854  11267 @25411        LDB TSTPT
```

```
          #01  FORMULA EVALUATION ROUTINES


5855  11270 135413        STB HSTPT,I   STORE ADDRESS OF T-STACK
5856  11271 066637        JMP FORM4
5857*
5858* CALL MULTIPLY
5859*
5860  11272 063376 EFMP   JSM BINOP
5861  11273 160220        JSM .FMPA,I
5862  11274 067260        JMP FOR14
5863*
5864* CALL DIVIDE
5865*
5866  11275 063376 EFDV   JSM BINOP
5867  11276 160221        JSM .FDVA,I
5868  11277 067260        JMP FOR14
5869*
5870*  CALL EXPONENTIAL ARROW
5871*
5872  11300 063376 EPWR   JSM BINOP
5873  11301 160224        JSM UPARA,I
5874  11302 067260        JMP FOR14
5875*
5876*  CALL UNARY MINUS
5877*
5878  11303 063404 EUMIN  JSM STTOP     FETCH TOP ELEMENT OF STACK
5879  11304 025411        LDB TSTPT
5880  11305 170004        XFR           TRANSFER TO T-STACK
5881  11306 000133        ADA M2
5882  11307 070517        LDA A,I     READ FIRST WORD OF MANTISSA
5883  11310 070250        SZA *+5       SKIP IF ZERO
5884  11311 121411        LDA TSTPT,I
5885  11312 072111        SLA *+2,S   REVERSE THE SIGN
5886  11313 070071        SLA *+1,C
5887  11314 131411        STA TSTPT,I
5888  11315 067263        JMP FOR14+3
5889*
5890*  EXECUTE LEFT BRACKET
5891*
5892  11316 014002 SCCNT  OCT 14002
5893*
5894  11317 045412 ELBRC  ISZ LSTPT     STACK A
5895  11320 027316        LDB SCCNT       SUBSCRIPT COMMA
5896  11321 060563        JSM SLWST         ON OPERATOR STACK
5897  11322 063406        JSM RSCHK     STACK A 1 ON
5898  11323 055413        DSZ HSTPT     OPERATOR
5899  11324 060553        JSM RHSTP
5900  11325 020020 TRUE   LDA ONE
5901  11326 067261        JMP FOR14+1    STACK A 1 ON T-STACK
5902*
5903* COMPARE TOP OPERANDS OF STACK
5904*
5905  11327 063376 COMPR  JSM BINOP       GET OPERAND ADDRESSES
5906  11330 160217        JSM .FSBA,I    SUBTRACT ONE FROM ANOTHER
5907  11331 020214        LDA AR2A      POINT TO RESULT IN AR2
5908  11332 065317        JMP SGNS     GET SGN IN (B)
5909*
5910* EXECUTE >
5911*
5912  11333 063327 EGTRT  JSM COMPR       COMPARE OPERANDS
5913  11334 074753        SBP ENEQL+1   <?
5914  11335 067354        JMP FALSE       YES
5915*
5916* EXECUTE <
5917*
5918  11336 063327 ELST   JSM COMPR       COMPARE OPERANDS
5919  11337 074056        CMB             REVERSE COMPARISON SENSE
5920  11340 067342        JMP *+2
5921*
5922* EXECUTE >=
5923*
5924  11341 063327 EGORE  JSM COMPR       COMPARE OPERANDS
5925  11342 075153        SBP TRUE      <?
5926  11343 067354        JMP FALSE       YES
5927*
5928* EXECUTE =
5929*
5930  11344 063327 EEQL   JSM COMPR       COMPARE OPERANDS
5931  11345 075010        SZB TRUE      =?
5932  11346 067354        JMP FALSE       NO
5933*
5934* EXECUTE <=
5935*
5936  11347 063327 ELORE  JSM COMPR       COMPARE OPERANDS
5937  11350 075653        SBP EEQL+1    >=?
5938  11351 067325 LTRUE  JMP TRUE      NO
5939*
5940* EXECUTE # OR <>
5941*
5942  11352 063327 ENEQL  JSM COMPR       COMPARE OPERANDS
5943  11353 077710        NZB LTRUE     #?
5944  11354 021411 FALSE  LDA TSTPT     CLEAR T-STACK ENTRY
5945  11355 170000        CLR
```

```
        #P1  FORMULA EVALUATION ROUTINES


5946  11356 @67263        JMP FOR14+3
5947*
5948* EXECUTE "OR"
5949*
5950  11357 @63376 EOR    JSM BINOP
5951  11360 @70070        STA **1
5952  11361 @70517        LDA A,I          LOOK AT FIRST OPERAND
5953  11362 @73350        SZA LTRUE      IF NON-ZERO, RESULT IS TRUE
5954  11363 @74070 EOR2   SLB **1          LOOK AT SECOND OPERAND
5955  11364 @74537        LDB B,I
5956  11365 @67353        JMP ENEQL+1
5957*
5958* EXECUTE "AND"
5959*
5960  11366 @63376 EAND   JSM BINOP
5961  11367 @70070        STA **1
5962  11370 @70517        LDA A,I          LOOK AT FIRST OPERAND
5963  11371 @71150        SZA FALSE      ZERO?
5964  11372 @67363        JMP EOR2         NO




5966*
5967* PREPARE TO EXECUTE A BINARY OPERATOR
5968*
5969  11373 @24057 BINO1  LDB .1000
5970  11374 @60406        JSM XFAR1+1    TRANSFER TO SECOND TEMP(YC)
5971  11375 @67401        JMP BINO2
5972  11376 @63433 BINOP  JSM OPCHK
5973  11377 @14111        CPB B1740      CONVERSION TEMPORARY?
5974  11400 @67373        JMP BINO1      YES
5975  11401 @35726 BINO2  STB MBIN1      SAVE ADDRESS IN TEMPORARY
5976  11402 @45413        ISZ MSTPT        MOVE HIGH STACK POINTER
5977  11403 @45413        ISZ MSTPT          TO DELETE OPERAND
5978*  PROCEED TO STTOP
5979*
5980*  FETCH TOP OF STACK
5981*
5982  11404 @63433 STTOP  JSM OPCHK
5983  11405 @35471        STB L2T2       SAVE SECOND ADDRESS
5984*  PROCEED TO RSCHK
5985*
5986* ALLOT SPACE FOR INTERMEDIATE RESULT
5987*
5988  11406 @21411 RSCHK  LDA TSTPT        MOVE TEMPORARY POINTER
5989  11407 @00021        ADA .4
5990  11410 @31411        STA TSTPT
5991  11411 @11410        CPA LSTAK      OVERFLOW INTO LOW STACK?
5992  11412 @67414        JMP **2        YES
5993  11413 @67430        JMP RSCH2      NO
5994  11414 @31476        STA TEMP3
5995  11415 @00025        ADA .8
5996  11416 @31410        STA LSTAK
5997  11417 @21412        LDA LSTPT
5998  11420 @70070        STA **1
5999  11421 @31475        STA TEMP2
6000  11422 @00024        ADA .7
6001  11423 @31412 RSCH1  STA LSTPT
6002  11424 @70070        STA **1
6003  11425 @31477        STA TEMP4
6004  11426 @60554        JSM BMSTP+1      TEST FOR OVERLAP WITH HIGH STACK
6005  11427 @61121        JSM MVTOH      MOVE LOW STACK UP
6006  11430 @21471 RSCH2  LDA L2T2       RECALL SECOND ADDRESS
6007  11431 @25726        LDB MBIN1      RECALL FIRST ADDRESS
6008  11432 170402 FORMR  RET
6009*
6010* VERIFY LEGITIMACY OF OPERAND
6011*
6012  11433 125413 OPCHK  LDB MSTPT,I      GET OPERAND ADDRESS
6013  11434 @15411        CPB TSTPT        IS IT TEMPORARY?
6014  11435 @67440        JMP OPC3         YES
6015  11436 @74517        LDA B,I          NO, GET EXPONENT WORD
6016  11437 @10062        CPA FLGBT        IS OPERAND DEFINED?
6017  11440 160206 ER8    JSM ERRA,I     YES,ERROR
6018  11441 @21413        LDA MSTPT          YES, GET TYPE
6019  11442 @70070        STA **1
6020  11443 @70517        LDA A,I
6021  11444 @50037 OPCH1  AND .28         GET PRECISION BITS
6022  11445 @71250        SZA FORMR      FULL PRECISION?
6023  11446 @10021        CPA .4         NO, SPLIT?
6024  11447 @67456        JMP OPC2       YES
6025  11450 @70206        MAR 5
6026  11451 @71052        SAM FORMR      NUMBERED VARIABLE?
6027  11452 @74537        LDB B,I          NO, IT MUST BE INTEGER
6028  11453 160227        JSM FXFLA,I    RESULT IS IN AR2 NOW
6029  11454 @24111 OPC1   LDB B1740
6030  11455 @64406        JMP XFAR1+1    TRANSFER TO XC REGISTER
6031  11456 160231 OPC2   JSM SPLFA,I    CONVERT SPLIT TO FULL PRECISION
```

```
          #01  FORMULA EVALUATION ROUTINES


6032  11457 @67454        JMP OPC1
6033  11460 @21411 OPC3   LDA TSTPT       DELETE TEMPORARY
6034  11461 @00135        ADA M4
6035  11462 @31411        STA TSTPT
6036  11463 170402        RET
6037*  11464 IS LAST
6038*  MATH ROUTINES HERE

          #01  FUNCTION TABLES


6040  11707               ORG 11707B
6041*
6042*  FUNCTION SYNTAX AND EXECUTION JUMP TABLE
6043*
6044  11707 @30002        ABS EPI-*
6045  11710 @00003        ABS ERES-*
6046  11711 @23715 EPI    LDA PIA
6047  11712 @64445        JMP XFAR2+1
6048  11713 @20215 ERES   LDA TMPOP
6049  11714 @64445        JMP XFAR2+1
6050  11715 @03764 PIA    DEF 3764B
6051*
6052*  EXECUTE "NOT"
6053*
6054  11716 @67404 ENOT   JSM SITOP
6055  11717 @70070        SIA *+1
6056  11720 @70537        LDB A+I
6057  11721 @67345        JMP EEQL+1
6058*
6059  11722 @00000        BSS 13           FUNCTION JUMP TABLE
6060  11737 175125        ABS TABER-*      TAB
6061*
6062*  FUNCTION NAME TABLE
6063*
6064  11740 @42530 FTN13  ASC 1,EX
6065  11741 @50202        OCT 050202       EXP!OP CODE 2
6066  11742 @46117        ASC 1,LO
6067  11743 @43603        OCT 43603        LOG! OP CODE 3
6068  11744 @40502        ASC 1,AB
6069  11745 @51604        OCT 051604       ABS!OP CODE 4
6070  11746 @51521        ASC 1,SQ
6071  11747 @51205        OCT 051205       SQR!OP CODE 5
6072  11750 @44516        ASC 1,IN
6073  11751 @52206        OCT 052206       INT!OP CODE 6
6074  11752 @51507        ASC 1,SG
6075  11753 @47207        OCT 47207        SGN! 7
6076  11754 @51511        ASC 1,SI
6077  11755 @47210        OCT 047210       SIN! 10
6078  11756 @41517        ASC 1,CO
6079  11757 @51611        OCT 051611       COS! 11
6080  11760 @52101        ASC 1,TA
6081  11761 @47212        OCT 047212       TAN! 12
6082  11762 @40524        ASC 1,AT
6083  11763 @47213        OCT 047213       ATN! 13
6084  11764 @46107        ASC 1,LG
6085  11765 @52214        OCT 52214        LGT!14
6086  11766 @40523        ASC 1,AS
6087  11767 @47215        OCT 47215        ASN! 15
6088  11770 @40503        ASC 1,AC
6089  11771 @51616        OCT 51616        ACS! 16
6090  11772 @51105        ASC 1,PE
6091  11773 @51630        OCT 51630        RES! 30 (NO ARGUEMENT)
6092  11774 @50111        ASC 1,PI
6093  11775 154524 TAB    OCT 154524       PI! 31 (NO ARGUEMENT)
6094  11776 @40502        OCT 40502
6095  11777 140777        OCT 140777       TAB! 1

          #01  FORMULA OPERATOR TABLE

6097  12000               ORG 12000B
6098  12000 @06774 STEPA  DEF FOPST+FOPST
6099*
6100*  FORMULA OPERATOR SEARCH AND PRECEDENCE TABEL
6101*  LEFT CHAR IS ASCII! RIGHT CHAR IS PRECEDENCE
6102*
6103  12001 @21000 QUOT   OCT 21000        OP CODE 1
6104  12002 @26000 COMMA  OCT 26000        2
6105  12003 @35400 SMCLN  OCT 35400        3
6106  12004 @24400 RPARN  OCT 24400        4
6107  12005 @56400 RBRAC  OCT 56400        5
6108  12006 @26002 SCMMA  OCT 26002        6
6109  12007 @36402 ASSOP  OCT 36402        7
6110  12010 177377 M257   DEC -257         THIS IS NOT USED AS TABLE ENTRY
6111  12011 @25407 PLUSS  OCT 25407        11
6112  12012 @26407 MINS   OCT 26407        12
6113  12013 @25010 TIMES  OCT 25010        13
6114  12014 @27410 DIV    OCT 27410        14
6115  12015 @57013 EXPS   OCT 57013        15
6116  12016 @37005 GTR    OCT 37005        16
```

```
          #01  SYNTAX SUBROUTINES


6117  12017 036005 LSS    OCT 36005      17
6118  12020 021405 UNEQL  OCT 21405      20
6119  12021 036405 EQUAL  OCT 36405      21
6120  12022 026412 UNMIN  OCT 26412      22
6121  12023 025412 UPLUS  OCT 25412      23
6122  12024 055420 LBRAC  OCT 55420      24
6123  12025 024020 LPARN  OCT 24020      25
6124  12026 000003 OROP   OCT 00003      26
6125  12027 000004 ANDOP  OCT 00004      27
6126  12030 000012 NOTOP  OCT 00012      30
6127  12031 000005 GTREQ  OCT 00005      31
6128  12032 000005 LSSEQ  OCT 00005      32
6129*
6130  12033 000000        BSS 1           *******CHECKSUM*****
6132                LST
6133  12034 007001 TOA    DEF FOPST+FOPST+5
6134  12035 007004 ORA    DEF OR+OR
6135  12036 007013 NOTA   DEF OR+OR+7
6136  12037 007017 GTEA   DEF GTE+GTE+1
6137*
6138*  CHECK SYNTAX OF A FORMULA
6139*
6140*     EXIT TO ERROR ON UNRECOGNISABLE OR INCORRECT INPUT
6141*     ON EXIT (A)= NEXT CHARACTER NOT IN FORMULA
6142*             (B)= 0
6143*     IF FIRST OPERAND  IS A STRING VARIABLE, EXIT TO ERROR
6144*       UNLESS STRING IS ENABLED BY SFLAG=SAPTH, THEN EXIT (B)=SFLAG=-1
6145*
6146*     MSFLG=8: LOOKING FOR "=" IN LET OR FOR STATEMENT
6147*     MSFLG=9: INTERPRET "=" AS A LOGICAL OPERATOR
6148*     IF A STORE IS FOUND, SFLAG=-2
6149*
6150  12040 020012 FSC    LDA .1          SET LEFT PARENTHESIS
6151  12041 131472        STA TEMPS,I       COUNT TO 0 (OFFSET BY 1)
6152  12042 020132 FSC1   LDA M1          ENABLE UNARY
6153  12043 031713        STA UFLAG         OPERATORS
6154*  LOOK FOR AN OPERAND
6155  12044 062231 FSC2   JSM VAROP       LOOK FOR VARIABLE OPERAND
6156  12045 066114        JMP FSC7        FIRST CHAR NOT A LETTER
6157  12046 066164        JMP FSC13-3     SUBSCRIPTED OR STRING VARIABLE
6158  12047 060752        JSM LETCK       SIMPLE VARIABLE: FOLLOWED BY LETTER?
6159  12050 066167        JMP FSC13       NO
6160*  DOES "AND" OR "OR" FOLLOW SIMPLE VARIABLE?
6161  12051 062217        JSM MCHCK       CHECK FOR "AND"OR"OR"ONLY:(B)#-1
6162*  CHECK FOR FUNCTION INSTEAD OF SIMPLE VARIABLE
6163  12052 062424        JSM FNCK        "FN"?
6164  12053 066061        JMP FSC4        NO
6165  12054 055400        DSZ BADDR       YES
6166  12055 000154        ADA M64         GET FUNCTION NAME
6167  12056 074742        SBR 16
6168  12057 135706        STB SBPTR,I     CLEAR OUT NEXT WORD
6169  12060 066074        JMP FSC3-1
6170  12061 070342 FSC4   SAR 8           RECALL LAST CHAR
6171  12062 055400        DSZ BADDR       BACK UP TO POINT AT SECOND CHAR
6172  12063 024133        LDB M2          SEARCH ALL TABLES FOR
6173  12064 160237        JSM SALTA,I       A PREDEFINED FUNCTION
6174  12065 066143        JMP FSC11       NOT FOUND
6175  12066 135706        STB SBPTR,I     STORE TABLE CODE
6176  12067 070137        LDB A
6177  12070 024147        ADB M23
6178  12071 074076        TCB
6179  12072 035474        STB TEMP1       SET NEGATIVE IF NO ARGUEMENT
6180  12073 040116        IOR UTOP        SET FLAG FOR PRE-DEF FUNCTION
6181  12074 070544        SAL 5           POSITION OP CODE
6182*  ASSEMBLE AND STORE FUNCTION
6183  12075 040040 FSC3   IOR B37         SET OPERAND TYPE FOR FUNCTION
6184  12076 070137        LDB A
6185  12077 055706        DSZ SBPTR       REPLACE
6186  12100 060641        JSM RESBP,I       PREVIOUS OPERAND
6187  12101 074217        IOR B             WITH FUNCTION
6188  12102 131706        STA SBPTR,I
6189  12103 045706        ISZ SBPTR
6190  12104 021474        LDA TEMP1       RECALL ARGUEMENT FLAG
6191  12105 070253        SAP FSC31        NORMAL ARITHMETIC ARGUEMENT?
6192  12106 161473        JSM TEMP,I      NO, JUMP THROUGH SYNTAX/EXEC TABLE
6193  12107 045706        ISZ SBPTR       RETURN HERE IF NO ARGUEMENT
6194  12110 061055        JSM GNEXT
6195  12111 066130        JMP FSC8        NO, CHECK FOR EOL NEXT
6196  12112 031713 FSC31  STA UFLAG       DISABLE UNARY OPERATORS
6197  12113 066140        JMP FSC10
6198*  CHECK FOR LEFT PARENTHESIS
6199  12114 045706 FSC7   ISZ SBPTR
6200  12115 060513        JSM LPCK        CHECK FOR PAREN
6201  12116 066123        JMP *+5         NOT FOUND
6202  12117 145472 FSC5   ISZ TEMPS,I
6203  12120 024026 FSC6   LDB .9          DISABLE ASSIGNMENT
6204  12121 035710        STB MSFLG
6205  12122 066042        JMP FSC1
```

#01 SYNTAX SUBROUTINES

```
6206*  CHECK FOR A NUMBER
6207   12123 074742         SBR 16          INITIALIZE
6208   12124 035721         STB SIGN          TO POSITIVE
6209   12125 160243         JSM NUMCA,I     FETCH AND BUILD CONSTANT
6210   12126 066133         JMP FSC9        NOT FOUND
6211   12127 160244         JSM NUMOA,I     STORE IN SYNTAX BUFFER
6212   12130 024026 FSCA    LDB .9
6213   12131 035710         STB MSFLG       DISABLE ASSIGNMENT
6214   12132 066167         JMP FSC13
6215*  CHECK FOR A UNARY OPERATOR
6216   12133 045713 FSC9    ISZ UFLAG       UNARY OPERATORS PERMITTED?
6217   12134 160206 FSCE3   JSM ERRA,I      NO, UNRECOGNISED OPERNAD
6218   12135 060461         JSM SYMC?       UNARY MINUS
6219   12136 012021         DEF UNMIN-1       OR UNARY PLUS?
6220   12137 066134         JMP FSCE3       NO
6221   12140 024026 FSC10   LDB .9
6222   12141 035710         STB MSFLG       DISABLE ASSIGNMENT
6223   12142 066044         JMP FSC?
6224*  CHECK FOR "NOT" RATHER THAN SIMPLE VARIABLE
6225   12143 045713 FSC11   ISZ UFLAG       "NOT" PERMITTED?
6226   12144 066212         JMP FSC14-1     NO
6227   12145 026036         LDB NOTA
6228   12146 060537         JSM STSRH       SEARCH FOR "NOT"
6229   12147 066212         JMP FSC14-1     NOT FOUND
6230   12150 055706         DSZ SBPTR       ERASE
6231   12151 060641         JSM RESBP+1       SPURIOUS
6232   12152 045706         ISZ SBPTR           OPERAND
6233   12153 066140         JMP FSC10
6234*
6235*  CHECK FOR RIGHT PARENTHESIS
6236*
6237   12154 060524 FSC12   JSM RPCK        RIGHT PAREN?
6238   12155 066213         JMP FSC14       NO
6239   12156 155472         DSZ TEMPS,I
6240   12157 066167         JMP FSC13       ((A)=NEXT CHAR)
6241   12160 055706         DSZ SBPTR       NO MATCHING LEFT PAREN
6242   12161 055400         DSZ BADDR       RESTORE
6243   12162 020046         LDA .41           PREVIOUS CONDITIONS
6244   12163 066215         JMP FSC14+2         AND RETURN
6245*  CHECK FOR A BINARY OPERATOR
6246   12164 024132         LDB M1          STRING VARIABLE
6247   12165 015714         CPB SFLAG         JUST FOUND?
6248   12166 170402         RET             YES
6249   12167 010031 FSC13   CPA EOL         END OF STMT?
6250   12170 066213         JMP FSC14       YES
6251   12171 031713         STA UFLAG       DISABLE UNARY OPERATORS
6252   12172 031474         STA TEMP1       SAVE CHAR
6253   12173 024132         LDB M1
6254   12174 062217         JSM MCHCK       CHECK FOR AND, OR, >=,<=,<>
6255   12175 025710         LDB MSFLG
6256   12176 060464         JSM SYMCK       SEARCH FOR
6257   12177 012010         DEF PLUSS-1       SINGLE CHAR OPERATOR
6258   12200 066204         JMP *+4         NOT FOUND
6259   12201 010124         CPA EXPOP       EXPONENTIATION?
6260   12202 066140         JMP FSC10       YES
6261   12203 066122         JMP FSC6        NO
6262   12204 060463         JSM SYMC1       NO, "=" ASSIGNMENT?
6263   12205 012006         DEF ASSOP-1
6264   12206 066154         JMP FSC12       NO
6265   12207 020133         LDA M2          YES, SET
6266   12210 031714         STA SFLAG         STORE OCCURRED FLAG
6267   12211 066042         JMP FSC1
6268*  END OF FORMULA
6269   12212 061055         JSM GNEXT       RECALL CHAR
6270   12213 155472 FSC14   DSZ TEMPS,I     ALL PARENS MATCHED?
6271   12214 064535         JMP FSCE?       MISSING RIGHT PAREN
6272   12215 074742         SBR 16
6273   12216 170402         RET             RETURN WITH (B)=0
6274*
6275*  CHECK FOR A MULTI-CHARACTER BINARY OPERATOR
6276*
6277*     (B)=-1: CHECK >=,<=,<> AS WELL AS AND,OR
6278*     EXIT TO FSC6 IF FOUND
6279*
6280   12217 135706 MCHCK   STB SBPTR,I     SAVE FLAG
6281   12220 026435         LDB ORA         SEARCH FOR "AND" OR "OR"
6282   12221 060537         JSM STSRH
6283   12222 066225         JMP *+3         NOT FOUND
6284   12223 055777         DSZ RSPTR       DINC STACK POINTER
6285   12224 066120         JMP FSC6        RETURN
6286   12225 145706         ISZ SBPTR,I     CHECK REST OR OPERATORS?
6287   12226 170402         RET             NO
6288   12227 026037         LDB GTEA        CHECK FOR >=,<=,<>
6289   12230 066221         JMP MCHCK+2
6291*
6292*  SEEK A VARIABLE OPERAND
6293*         P+1: FIRST CHAR NOT A LETTER
6294*         P+2: ARRAY OR STRING VARIABLE FOUND
6295*         P+3: SIMPLE VARIABLE
6296*
```

```
        #01  SYNTAX SUBROUTINES


6297*    (A)= NEXT CHAR FOLLOWING OPERAND
6298*    IF STRING IS FOUND AND SFLAG=SBPTR THEN SET SFLAG=-1
6299*         IF SFLAG#SBPTR THEN STRING IS NOT PERMITTED! ERROR
6300*
6301  12231 061140 VAROP JSM LTR       LETTER?
6302  12232 170402       RET           NO. RETURN TO P+1
6303  12233 074742       SBR 16        INITIALIZE FOR
6304  12234 035717       STB OPTYP       FULL PRECISION
6305  12235 010077       CPA D44       IS NEXT CHAR "$"?
6306  12236 066275       JMP VAR04     YES
6307  12237 060750       JSM DIGCK     DIGIT?
6308  12240 066250       JMP VAR01     NO
6309  12241 070137       LDB A         YES. RESTORE ASCII
6310  12242 021474       LDA TEMP1     RECALL LETTER
6311  12243 035474       STB TEMP1     SAVE DIGIT
6312  12244 060450       JSM STROP     COMBINE AND STORE OPERAND
6313  12245 061055       JSM GNEXT
6314  12246 031475       STA TEMP2     SAVE IT
6315  12247 064356       JMP RETN3
6316  12250 074742 VAR01 SBR 16
6317  12251 010073       CPA S
6318  12252 024021       LDB .4        OPTYP=4 FOR SPLIT PRECISION
6319  12253 010055       CPA I
6320  12254 024225       LDB .8        OPTYP=8 FOR INTEGER
6321  12255 074350       SZB VAR06     SKIP IF FULL PRECISION
6322  12256 055711       DSZ DFLAG     DIM OR COM?
6323  12257 066261       JMP *+2       NO. SPLIT OR INTEGER INT ALLOWED
6324  12260 035717       STB OPTYP     YES. STORE OPERAND TYPE
6325  12261 045711       ISZ DFLAG     DIM OR COM?
6326  12262 061055       JSM GNEXT     YES GET CHAR FOLLOWING PRECISION
6327  12263 031475       STA TEMP2     STORE CHAR
6328  12264 010045 VAR06 CPA .40       "("
6329  12265 066275       JMP VAR04     YES
6330  12266 010102       CPA D133      "["?
6331  12267 066275       JMP VAR04     YES
6332  12270 021474       LDA TEMP1     RECALL LETTER
6333  12271 025717       LDB OPTYP
6334  12272 060451       JSM STROP+1   COMBINE AND STORE SIMPLE VARIABLE
6335  12273 021475 VAR03 LDA TEMP2     NO. RECALL NEXT CHAR
6336  12274 064356       JMP RETN3     RETURN TO P+3
6337  12275 062436 VAR04 JSM STRID     STORE STRING OR ARRAY OPERAND
6338  12276 062301       JSM SBSCK     GET SUBSCRIPT
6339  12277 074742       SBR 16        NO SUBSCRIPT. SET B=0
6340  12300 064357       JMP RETN2
6342*
6343*  SUBSCRIPT SYNTAX
6344*
6345*    ARYAD POINTS TO ARRAY OPERAND
6346*    EXIT TO P+1 IF FIRST CHAR ISNOT "(" OR "[" OTHERWISE P+2
6347*    DFLAG=1 IF ANALYSING DIM OR COM STMT; 0 OTHERWISE
6348*    SFLAG=-1 FOR STRING OPERAND
6349*
6350*    ARRAY OPERANDS ARE MARKED AS ONE OR TWO DIMENITIONAL
6351*    MEMORY REQUIREMENT IS COMPUTED FOR COM STMT
6352*
6353  12301 060513 SBSCK JSM LPCK      FIRST CHAR A "(" OR "["?
6354  12302 170402       RET           NO
6355  12303 020125       LDA LBOP      YES. RECORD
6356  12304 131706       STA SBPTR,I     A LEFT BRACKET
6357  12305 155723       DSZ ARYAD,I   SET ARRAY
6358  12306 155723       DSZ ARYAD,I     TO 7 DIMENSION
6359  12307 021711       LDA DFLAG     DIM OR COM?
6360  12310 072110       RZA *+2
6361  12311 066363       JMP SBSC3
6362  12312 024157       LDB M256      ALLOW 255
6363  12313 045714       ISZ SFLAG     STRING?
6364  12314 004132       ADB M1        NO. ALLOW UP TO 256
6365  12315 060666       JSM PGINT     GET AN INTEGER
6366  12316 004132       ADB M1        OFFSET BY -1
6367  12317 074404       SBL 8
6368  12320 035474       STB TEMP1     SAVE DIMENSION
6369  12321 025714       LDB SFLAG
6370  12322 074450       SZB SBSC1     SKIP IF STRING
6371  12323 060463       JSM SYMC1     LOOK FOR A COMMA
6372  12324 012005       DEF SCMMA-1     A COMMA
6373  12325 066333       JMP SBSC1     NOT FOUND
6374  12326 145723       ISZ ARYAD,I   SET TO 2 DIMENSIONS
6375  12327 026010       LDB M257      GET SECOND
6376  12330 060666       JSM PGINT     DIMENSION
6377  12331 004132       ADB M1        OFFSET BY -1
6378  12332 066334       JMP SBSC1+1
6379  12333 074742 SBSC1 SBR 16        SECOND DIM=1 OFFSET BY -1
6380  12334 035474       ADB TEMP1     COMBINE DIMENSIONS
6381  12335 035714       STB SFLAG     SAVE PACKED DIMENSIONS
6382  12336 060534 SBSC2 JSM RPCKF     DEMAND ")" OR "]"
6383  12337 024123       LDB RBOP      STORE
6384  12340 055706       DSZ SBPTR
6385  12341 135706       STB SBPTR,I     A RIGIT BRACKET
```

```
        #01  SYNTAX SUBROUTINES

6396  12342 045706        ISZ SBPTR
6397  12343 025711        LDB UFLAG      DIM OR COM?
6398  12344 076710        RZB SBSCR
6399* RESTORE LOCAL QUANTITIES FOR FSC
6400  12345 045472        ISZ TEMPS
6401  12346 024112        LDB RSTKA      RESET
6402  12347 035777        STB RSPTR        STACK POINTER
6403  12350 066353        JMP FPOP1
6404  12351 135777 FPOP2  STB RSPTR,I     STORE ON RETURN STACK
6405  12352 045777        ISZ RSPTR
6406  12353 125472 FPOP1  LDB TEMPS,I    GET WORD FROM S-STACK
6407  12354 045472        ISZ TEMPS
6408  12355 075633        SAP FPOP2,C    IF NEGATIVE, IT MUST BE MSFLG
6409  12356 035710        STB MSFLG      RESTORE MSFLG
6410  12357 125472        LDB TEMPS,I
6411  12360 045472        ISZ TEMPS
6412  12361 035713        STB UFLAG      RESTORE UFLAG
6413  12362 064357 SBSCR  JMP RETN2      RETURN TO P+2
6414* SAVE LOCAL QUANTITIES FOR FSC
6415  12363 021713 SBSC3  LDA UFLAG
6416  12364 062416        JSM DINCT      SAVE UFLAG ON THE S-STACK
6417  12365 021710        LDA MSFLG
6418  12366 072053        SAP *+1,S      SET FLAG
6419  12367 062416        JSM DINCT      SAVE MSFLG
6420  12370 055777 FRCU1  DSZ RSPTR
6421  12371 121777        LDA RSPTR,I    GET RETURN ADDRESS
6422  12372 062416        JSM DINCT      SAVE IT
6423  12373 021777        LDA RSPTR
6424  12374 020112        CPA RSTKA      ALL OF STACK MOVED?
6425  12375 066377        JMP *+2        YES
6426  12376 066370        JMP FRCU1
6427  12377 021723        LDA ARYAD
6428  12400 062416        JSM DINCT      SAVE ARRAY ADDRESS
6429  12401 062416        JSM DINCT      DINK AGAIN AND CHECK OVERFLOW
6430  12402 020026        LDA .9         DISABLE STORE MODE
6431  12403 031710        STA MSFLG
6432  12404 062040        JSM FSC        GET INDEX FORMULA
6433  12405 045472        ISZ TEMPS      RESTORE
6434  12406 125472        LDB TEMPS,I      ARRAY
6435  12407 035723        STB ARYAD          ADDRESS
6436  12410 060463        JSM SYMC1
6437  12411 012005        DEF SCMMA-1    COMMA?
6438  12412 066336        JMP SBSC2      NO
6439  12413 145723        ISZ ARYAD,I    SET TO 2 DIMENSIONS
6440  12414 062040        JSM FSC        GET SECOND INDEX FORMULA
6441  12415 066336        JMP SBSC2
6442*
6443* DECREMENT TEMPS AND CHECK FOR OVERFLOW
6444*
6445  12416 055472 DINCT  DSZ TEMPS
6446  12417 025472        LDB TEMPS
6447  12420 015412        CPB LSTPT      OVERFLOW?
6448  12421 064561        JMP FSCE4      YES
6449  12422 131472        STA TEMPS,I    NO, SAVE (A) ON STACK
6450  12423 170402        RET
6451*
6452* CHECK FOR "FN" FOLLOWED BY A LETTER
6453*   EXIT P+1: NOT FOUND
6454*        P+2: FOUND, (A)=LETTER
6455*
6456  12424 025474 FNCK   LDB TEMP1
6457  12425 074404        SBL 8
6458  12426 074217        IOR B
6459  12427 010162        CPA FN         "FN"?
6460  12430 066432        JMP *+2        YES
6461  12431 170402        RET
6462  12432 061140        JSM LTR        NEXT CHAR A LETTER?
6463  12433 160206 SYNE4  JSM ERRA,I     NO
6464  12434 021474        LDA TEMP1      YES, RECALL LETTER
6465  12435 064357        JMP RETN2
6466*
6467* RECORD A STRING OR ARRAY IDENTIFIER
6468*
6469  12436 025706 STRID  LDB SBPTR      SET POINTER TO
6470  12437 035723        STB ARYAD        ARRAY OR STRING OPERAND
6471  12440 010077        CPA B44        NEXT CHAR "$"?
6472  12441 066450        JMP STRI1      YES
6473  12442 021474        LDA TEMP1      GET OPERAND LETTER
6474  12443 025717        LDB OPTYP      SET THE TYPE TO FULL,SPLIT,OR INTEGER
6475  12444 040020        ADB .3           WITH UNKNOWN DIMENSIONS
6476  12445 060451        JSM STROP+1    COMBINE AND STORE
6477  12446 021475        LDA TEMP2      RECALL FOLLOWING CHAR
6478  12447 170402        RET
6479  12450 015714 STRI1  CPB SFLAG      STRING VARIABLE ALLOWED?
6480  12451 066453        JMP *+2        YES
6481  12452 160206 STRNP  JSM ERRA,I       NO
6482  12453 020132        LDA MI         SET FLAG
6483  12454 031714        STA SFLAG        TO "STRING OCCURRED"
6484  12455 021474        LDA TEMP1
6485  12456 024076        LDB B36
6486  12457 035723        STB ARYAD      POINT ADDRESS AT ROM
6487  12460 060451        JSM STROP+1    STORE STRING OPERAND
6488  12461 065055        JMP GNEXT      GET NEXT CHAR AND RETURN
```

#01 SYNTAX

```
6481*
6482*  FOR STATEMENT SYNTAX
6483*
6484  12462 062231 FORS  JSM VAROP      SEEK A SIMPLE VARIABLE
6485  12463 066464       JMP SYNE8
6486  12464 160206 SYNE8 JSM ERRA,I     NOT FOUND
6487  12465 060463       JSM SYMC1      SEEK "="
6488  12466 012006       DEF ASSOP-1
6489  12467 066522       JMP SYNE6      NOT FOUND
6490  12470 062040       JSM FSC        GET REST OF INITIAL FORMULA
6491  12471 026034       LDB TOA        SEEK "TO"
6492  12472 060537       JSM STSRH
6493  12473 160206 SYNE9 JSM ERRA,I     NOT FOUND
6494  12474 062040       JSM FSC        GET ENDING FORMULA
6495  12475 061153       JSM EOLCK      EOL?
6496  12476 026000 FORS1 LDA STEPA      SEEK "STEP"
6497  12477 060537       JSM STSRH
6498  12500 160206 SYE1A JSM ERRA,I     NOT FOUND
6499*
6500*  WAIT STATEMENT SYNTAX
6501*
6502  12501 062040 WAITS JSM FSC        GET FORMULA
6503  12502 065151       JMP EOST       DEMAND END OF STMT
6504*
6505*  DEF STATEMENT SYNTAX
6506*
6507  12503 061140 DEFS  JSM LTR
6508  12504 066433       JMP SYNF4
6509  12505 062424       JSM FNCK       "FN" FOLLOWED BY LETTER?
6510  12506 066433       JMP SYNF4      NO
6511  12507 024040       LDB B17        YES RECORD
6512  12510 060451       JSM STROP,I      FUNCTION NAME
6513  12511 021475       LDA TEMP2
6514  12512 060521       JSM LPCKF      DEMAND LEFT PAREN
6515  12513 062231       JSM VAROP      FETCH SIMPLE VARIABLE
6516  12514 066515       JMP SYNF5
6517  12515 160206 SYNF5 JSM ERRA,I     NOT FOUND
6518  12516 060534       JSM RPCKF      DEMAND RIGHT PAREN
6519  12517 061153       JSM EOLCK      EOL?
6520  12520 060463       JSM SYMC1      DEMAND
6521  12521 012006       DEF ASSOP-1      AN "="
6522  12522 160206 SYNF6 JSM ERRA,I
6523  12523 066501       JMP WAITS      FETCH ASSIGNING FORMULA
6524*
6525*  RETURN STATEMENT SYNTAX
6526*
6527  12524 061066 RETNS JSM GETSK      RETURN A  VALUE?
6528  12525 065150       JMP EOST-1     NO
6529  12526 055400       DST HADDR
6530  12527 066501       JMP WAITS      YES, GET A FORMULA
6531*  12533  IS LAST
6532*  MATH ROUTINES HERE
```

#01 STATEMENT EXECUTION JUMP TABLE

```
6534*
6535*  EXECUTION JUMP TABLE
6536*
6537  13741              ORG 13741B
6538  13741 173131       ABS EWRIT-*
6539  13742 173772       ABS EGRAD-*
6540  13743 173767       ABS EDEG-*
6541  13744 173763       ABS ERAD-*
6542  13745 174111       ABS ELET-*
6543  13746 174044       ABS XEC4-*     FORMAT
6544  13747 172777 DISP  ABS EDISP-*
6545  13750 174505       ABS EINPT-*
6546  13751 174562       ABS ENSTR-*
6547  13752 172773       ABS EPRIN-*
6548  13753 174011       ABS EREAD-*
6549  13754 174036       ABS XEC4-*     DATA
6550  13755 173531       ABS EWAIT-*
6551  13756 174130       ABS ESTOP-*
6552  13757 174106       ABS EEND-*
6553  13760 174252       ABS ERTRN-*
6554  13761 002577       ABS EGOSB-*
6555  13762 174354       ABS ENEXT-*
6556  13763 174265       ABS EFOR-*
6557  13764 174213       ABS EIF-*
6558  13765 002574       ABS EGOTO-*
6559  13766 174024       ABS XEC4-*     REM
6560  13767 174023       ABS XEC4-*     DEF
6561  13770 174022       ABS XEC4-*     COM
6562  13771 174021       ABS XEC4-*     DIM
6563  13772 174064       ABS ELET-*     IMPLIED LET
6565  13773 167371       ABS POP13-*    NON-FORMULA OPERATOR TABLE
6566  13774 175744       ABS FTN13-*    FUNCTION TABLE
6567  13775 167434       ABS CMD13-*    SYSTEM COMMAND TABLE
6568  13776 173560       ABS SYN13-*    STATEMENT TABLE
6569  13777 012000       OCT 12000      TABLE OP-CODE
```

#P1 SYNTAX

```
6571  16410              ORG 16410B
6572*
6573*  CHECK SYNTAX OF DIM AND COM STATEMENTS
6574*
6575  16410 045711 ARRYS ISZ DFLAG     SET DIM MODE FOR SBSCK
6576  16411 021706       LDA SBPTR
6577  16412 031710       STA MSFLG     SAVE POINTER
6578  16413 031714       STA SFLAG     ENABLE STRING VARIABLES
6579  16414 160246       JSM VAR0A,I   FETCH I.D.
6580  16415 164321       JMP E25+1,I   MISSING VARIABLE
6581  16416 074610       SZB ARRE1     ARRAY OR STRING; DIMENSIONED?
6582  16417 031475       STA TEMP2
6583  16420 121710       LDA MSFLG,I   FETCH IDENTIFIER
6584  16421 025714       LDB SFLAG     FETCH PACKED DIMENSIONS
6585  16422 160240       JSM MDIM4,I   COMPUTE SPACE REQUIRED
6586  16423 001461       ADA FTMP2     UPDATE
6587  16424 031461       STA FTMP2       COMMON POINTER
6588  16425 021475       LDA TEMP2
6589  16426 010031 ARRY1 CPA EOL       END OF STMT?
6590  16427 170402       RET           YES
6591  16430 060510       JSM COMCF     DEMAND A COMMA
6592  16431 066411       JMP ARRYS+1   CONTINUE
6593  16432 160206 ARRE1 JSM ERRA,I    NO DIMENSION ON STRING
6594*
6595*  COM STMT SYNTAX AND STORAGE ALLOCATION
6596*
6597  16433 025423 COMS  LDB PBPTR
6598  16434 015424       CPB PBUFF     NULL PROGRAM?
6599  16435 066437       JMP *+2
6600  16436 160206 SYNE3 JSM ERRA,I    NO. ILLEGAL COM
6601  16437 021422       LDA UKPTR
6602  16440 072710       RZA DIMS      SKIP IF KEY: NO STORAGE ALLOCATION
6603  16441 035461       STB FTMP2     INITIALIZE COMMON POINTER
6604  16442 062410       JSM ARRYS      CHECK SYNTAX
6605  16443 025404       LDB LWAM
6606  16444 004156       ADB M102
6607  16445 074076       TCB
6608  16446 005461       ADB FTMP2     EXCEEDED MAXIMUM
6609  16447 074112       SBM *+2         COMMON AREA?
6610  16450 064561       JMP ESCF4     YES
6611  16451 025461       LDB FTMP2     MOVE PROGRAM
6612  16452 035424       STB PBUFF       POINTERS
6613  16453 035423       STB PBPTR
6614  16454 035426       STB PBPTO
6615  16455 164203       JMP ACTSA,I
6616*
6617*  DIM STMT SYNTAX
6618*
6619  16456 062410 DIMS  JSM ARRYS     CHECK SYNTAX
6620  16457 164203       JMP ACTSA,I
6621*
6622*  READ AND INPUT STATEMENT SYNTAX
6623*
6624  16460 160246 READS JSM VAR0A,I   RECORD VARIABLE OPERAND
6625  16461 160206 SYE13 JSM ERRA,I    NONE FOUND
6626  16462 066463       JMP *+1
6627  16463 060504       JSM COMCK     COMMA?
6628  16464 065151       JMP EOST      NO
6629  16465 066460       JMP READS     YES
6630*
6631*  RESTORE STMT SYNTAX
6632*
6633  16466 061056 RSTRS JSM GETSK     LINE # GIVEN?
6634  16467 065150       JMP ENDS+1    NO
6635  16470 060663       JSM LNUM-1    YES. GET LINE #
6636  16471 065151       JMP EOST
6637*
6638*  REM STATEMENT SYNTAX
6639*
6640  16472 020107 REMS  LDA B1000
6641  16473 041552       IOR RBUFF+1
6642  16474 031552       STA RBUFF+1   SET ASCII BIT
6643  16475 020031       LDA EOL       SET STATEMENT TERMINATOR
6644  16476 024012       LDB .1        SET TO STORE IN LOW HALF
6645  16477 160253       JSM CHRSA,I   RECORD REST TO STMT AS ASCII
6646  16500 164203       JMP ACTSA,I
6647*
6648*  GET SELECT CODE AND FORMAT #
6649*
6650  16501 045706 IOHED ISZ SBPTR
6651  16502 060520       JSM LPCKF-1     DEMAND LEFT BRACKET
6652  16503 160202       JSM FSCA,I      GET SELECT CODE
6653  16504 060510       JSM COMCF       DEMAND COMMA
6654  16505 061055       JSM GNEXT
6655  16506 045706       ISZ SBPTR
6656  16507 060463       JSM SYMC1       LOOK FOR "#"
6657  16510 012012       DEF TIMFS-1
6658  16511 066514       JMP *+3         NOT FOUND
6659  16512 045706       ISZ SBPTR
6660  16513 065055       JMP GNEXT
6661  16514 055706       DSZ SBPTR
6662  16515 064663       JMP LNUM-1
```

```
          #01  SYNTAX


6664*
6665*  SEARCH ALL TABLES
6666*     (A)=FIRST CHAR   BADDR POINTS TO REST OF STRING
6667*     (B)=0     FOR STATEMENTS
6668*        =-1    FOR SYSTEM COMMANDS
6669*        =-2    FOR ARITHMETIC FUNCTIONS
6670*
6671  16516 004132 SALT  ADB M1          SAVE RELATIVE ADDRESS OF TABLE
6672  16517 035473       STB TEMP        GET ADDRESS OF LAST
6673  16520 025403       LDB MAW         SEARCH TABLE ON R/W PAGE
6674  16521 062531       JSM SPAGE       GET ADDRESS OF LAST ROM PAGE
6675  16522 024131       LDB LROMA       SEARCH TABLE IN ROM
6676  16523 062531 SALT1 JSM SPAGE       ALL TABLES SEARCHED?
6677  16524 025717       LDB NUM         YES
6678  16525 014130       CPB FROMA       NO, SEARCH NEXT PAGE
6679  16526 170402       RET
6680  16527 004160       ADB M1024
6681  16530 066523       JMP SALT1
6682*
6683*  SEARCH TABLE ON A GIVEN PAGE
6684*     RET IF NOT FOUND
6685*     RET TO P+2 OF ROUTINE WHICH CALLED SALT IF FOUND
6686*        (A)= OP CODE
6687*        (B)= PAGE
6688*        (SRPTR,I)=SHIFTED OP CODE
6689*        (TEMP)=JUMP ADDRESS
6690*
6691  16531 035717 SPAGE STB NUM         SAVE PAGE ADDRESS
6692  16532 074537       LDB B,I         READ LAST WORD OF PAGE
6693  16533 074310       SZB ISRET       RETURN IF ZERO
6694  16534 025717       LDB NUM         GET TABLE
6695  16535 005473       ADB TEMP          STARTING
6696  16536 074437       ADB B,I             ADDRESS
6697  16537 074744       SBL 1           POINT TO FIRST CHARACTER
6698  16540 060537       JSM STSRH       SEARCH TABLE
6699  16541 170402 ISRET RET             NOT FOUND, RETURN
6700  16542 070442       SAR 1W
6701  16543 025473       LDB TEMP
6702  16544 014133       CPB M2          GET 4 BIT CODE
6703  16545 050032       AND B17           FOR SYSTEM COMMANDS
6704  16546 005717       ADB NUM         GET TABLE
6705  16547 074437       ADB B,I           ADDRESS
6706  16550 070076       TCA             COMPUTE AND
6707  16551 074017       ADA B           STORE
6708  16552 070417       ADA A,I           ABSOLUTE
6709  16553 031473       STA TEMP            JUMP ADDRESS
6710  16554 125717       LDB NUM,I       GET OP CODE FOR THIS PAGE
6711  16555 021466       LDA LOCL1       RECALL OP CODE
6712  16556 055777       DSZ RSPTR       RETURN TO ROUTINE
6713  16557 064357       JMP RETN2         WHICH CALLED SALT

          #01  EXECUTION


6715*
6716*  EXECUTE GOTO AND GOSUB
6717*
6718  16560 025421 EGOSB LDB PRADD       STACK RETURN ADDRESS
6719  16561 060563 EGOTO JSM SLWST       STACK GARBAGE IF GOTO
6720  16562 060771       JSM GTOPP       GET FIRST OPERAND
6721  16563 010064       CPA INTFL
6722  16564 066602       JMP EGOS2       YES, SIMPLE BRANCH
6723  16565 061217       JSM FETCH       NO, COMPUTE BRANCH INDEX
6724  16566 160225       JSM FLTRA,I     CONVERT TO INTEGER
6725  16567 066606       JMP EGOS3       OVERFLOW
6726  16570 004154       ADB M64
6727  16571 074653       SBP EGOS3       SKIP IF GREATER THAN 64
6728  16572 004054       ADB .63           RESTORE OFFSET BY -1
6729  16573 074552       SBM EGOS3       SKIP IF RESULT WAS NEGATIVE OR 0
6730  16574 074744       SBL 1           MULTIPLY BY 2
6731  16575 005472       ADB TEMPS       COMPUTE
6732  16576 035472       STB TEMPS         AND SAVE ADDRESS
6733  16577 074056       CMB
6734  16600 005421       ADB PRADD
6735  16601 074252       SBM EGOS3       SKIP IF BEYOND END OF STATEMENT
6736  16602 060567 EGOS2 JSM UNSTK       UNSTACK FLAG
6737  16603 011421       CPA PRADD       WAS IT A GOSUB?
6738  16604 045412       ISZ LSTPT       YES, RESTORE RETURN ADDRESS ON STACK
6739  16605 164610       JMP EGS1A,I     EXECUTE BRANCH
6740  16606 055412 EGOS3 DSZ LSTPT       OUT OF BOUNDS, DELETE FROM STACK
6741  16607 164172       JMP XEC4A,I
6742  16610 010202 EGS1A DEF EGOS1
6743*  166148 IS LAST
6744*  TAPE COMMAND SYNTAX FOLLOWS HERE
6745*
6746                     END
*  NO ERRORS*
```

CROSS-REFERENCE TABLE

.1 0020 0345 0393 1008 2178 2527 2960 3044 3459 3610 3712
3853 3998 4024 4168 4256 4462 4927 5520 5534 5831 6150
6644
.10 0071 0072 0603 1962 2401 2402 2985 2986 4135 5210 5218
5278
.100 0070 2777 4004
.1000 0069 5969
.11 0035 0043 1314 1651 2249 2962 4031 4133
.12 0036 3333 3379 4033 4157 4163 4183 4731
.13 0039 4037 4144
.15 0048 0083
.16 0049 0084 1737 1757 5131
.1F4 006A 1445 1492 2415 2437 2438 2631
.2 0027 0391 0475 2170 3302 3308 3711 3779 4478 4685 4932
5409 5679
.21 0050 3855
.22 0051 0087 2964
.23 0052 0088 5567
.28 0053 1966 6021
.3 0028 0045 0582 1373 1374 1453 3330 3351 3359 3363 3366
3710 5058 5153 5657 6466
.31 0054 0090 1033 1688 1719 2138 3746 3756 3826 3865 3874
3888 3894 3969
.32 0055 0715 1287 1334 1543 1874 2554 4467 4467 4744
.33 0056
.34 0057 0091 3880 3883
.37 0058
.4 0029 2549 3265 3601 5244 5989 6023 6318
.48 0059 6328
.41 0060 1405 0243
.43 0061 0880 1238
.45 0062 0882 1239
.46 0063 1240
.48 0064 1001 1241 3902
.5 0030 1339 3607 3993 4063 5241 5291 5303 5663
.58 0065 0094 1242
.6 0031 0044 2788
.63 0066 1747 2012 2681 2681 6728
.7 0032 2070 4069 6000
.8 0033 5703 5495 6320
.9 0034 2503 2601 2796 3208 5313 5615 6203 6212 6221 6420
.94 0074 2812
.ADD 2376 2356
.BUFA 0180 0776 1412 1439 1791 1843 2778
.EIN1 5408 5425
.FADA 0185 5292 5841
.FDA1 5378 5366
.FDVA 0188 5867
.FMPA 0187 5861
.FSBA 0186 5310 5751 5906
.FSR 0190
.LNUM 1099 0501 0793 1300 2403 2561 2565 2593 2794 2799 2995
3083 3018 3019 3156 3170 3494 3527
.NPWR 1014 2529 4257 5121
.OVER 0985 0977
.SLRF 0181 0308 1705 1856 1861 1900 2062 2542
A 0000 0297 0304 0629 0664 0669 0671 0693 0695 0722 0725
0754 0948 1004 1006 1011 1050 1423 2072 2192 2209 2220
2923 2930 2945 3148 3163 3430 3543 3545 3551 3587 3599
3623 3669 3670 3676 3859 4172 4584 4594 4626 4684 4701
4703 4706 5083 5201 5211 5304 5321 5515 5537 5545 5595
5658 5853 5882 5952 5962 6020 6056 6176 6184 6309 6708
ACCS1 2876 2867 2873
ACCS2 2889 2883 2894 2903
ACCS3 2881 2875
ACCS4 2900 2871 2872 2885
ACTKY 2861 2671
ACTSA 0174 0862 6615 6620 6646
ACTST 2854 0174
ADRSS 1112
AERR 2758 0176
ALRRA 0176 0352 0967 0972
AIEX1 2249 2113
AIEX2 2250 2120
ALCL1 2246 2142
ALNUM 2554
ANDOP 6125
AR1 1198 0182 0355 1207 3989 4001
AR1A 0182 0324 0363
AR1E 1207 0940 0945
AR2 1200 0183 1208 2135 2141 2246 2246 3990 4002 4091 4114
4147 4151 4230 5145 5314
AR2A 0183 0325 0364 0956 0991 5474 5689 5847 5907
AR2E 1208 0942 0954 0961 0979 0982 0984
ARRAS 5741 5542
ARRE1 6593 0251 6581
ARRY1 6589
ARRYS 6575 6592 6604 6619
ARW 1214 1502 3728 3733 3744

CROSS-REFERENCE TABLE

INYAD	1198	6357	6358	6374	6417	6425	6429	6461	6477	
ISSOP	6109	6263	6488	6521						
ISTR1	2247	2108								
ISTR2	2248	2117								
IUTD	2400	2288								
	0000	0309	0318	0313	0376	0399	0409	0515	0691	0718	0721
	0732	0752	0755	0798	0805	0826	0915	0939	0947	1054	1062
	1065	1385	1508	1711	1807	2085	2094	2103	2111	2116	2121
	2124	2129	2136	2179	2191	2197	2208	2213	2233	2237	2489
	2490	2495	2768	2890	2917	2926	2931	2934	3107	3110	3121
	3125	3138	3144	3173	3191	3222	3225	3236	3305	3341	3350
	335A	3369	3372	3378	3382	3383	3394	3412	3415	3432	3457
	3476	3697	3730	3802	3841	3921	4203	4500	4592	4636	4666
	5029	5030	5169	5246	530A	5492	5497	5505	5523	5555	5562
	5564	5575	5585	5587	5637	5674	5660	5807	5811	5827	5955
	6015	6027	6187	6449	6692	6696	6705	6707			
1100	0095	1324	1340	144A	2139	3891					
11000	0102	3571	3795	6664							
110K	0114	0115	1245	2243							
112K	0116	0117	2206								
1133	0096	6330									
1137	2049	1821									
140	0097	1051	1052	1622							
1400	0103	0204									
152	1623	1312									
17	0083	5455	6703								
170K	0121	1837	2539	2584	2786						
1740	0104	0203	5973	6029							
176	2049	1827									
1760	0105	0158	3336								
177	0098	0694	1809	1817	457A	4632	4997				
1774	0106										
1775	0107										
1777	010A	3339	3344	3471	3511	3943	5329	5659			
20	0084	5005	5533								
200	2044	1926									
22	0085	1282	130A								
23	0086	1310									
2400	0110										
26	0087	1986									
27	0088	1984									
2K	1244										
3000	0111										
33	2047	1417	1425								
36	0089	3218	3260	3897	4574	6476					
37	0090	0456	1063	3259	3263	3600	3689	4573	4625	5462	5463
	5498	5502	5524	5528	6183	6511					
377	0099	1953	2104	3295	4501	5808					
400	0100	4000									
4000	0112	0162	1803	4595							
42	0091	2370	2717	2726	4865						
43	2429	2410									
44	0092	1243	6305	6462							
4400	3244	3206									
52	0093	1652	2247	2685	3770	3787					
57	1624	1419									
60	1241	4102	4193	4195	4210						
6K	0113	0167									
72	0094										
777	0101	359A									
ADD	2377	2364									
ADDR	1076	0311	0530	0539	0572	0777	1285	1289	1302	1511	1515
	1519	1542	1707	1792	1931	1835	1842	1955	1958	1976	1985
	1914	2017	2027	2063	2366	2412	2541	2563	2642	2644	2709
	2749	2801	2802	4386	4393	4412	4415	4509	4521	452A	4721
	4728	4944	5402	5422	6165	6171	6242	6529			
ADRP	0204	0716	0750	0766							
ASE	4629										
EFB	1625	1422									
END	0198	0199	1401	1709							
ERR	5267	0282	5258	5273							
HSTP	0471	0486	3544	5134	520A	5471	5506	5549	5731	5899	6004
IND1	5969	5974									
IND2	5975	5971									
INDP	5972	5758	5840	5860	5866	5872	5905	5950	5964		
K1	1866	1860									
K2	1861	1854									
KPTA	2845	1349	1940	1949	1967						
KPTR	1230	1275	1347	1509	1516	1535	1704	1953	1959	1865	1870
	1872	1875	1915	1918	193A	1943	1946	1951	1957	1969	2045
KSP	1853	1309									
LANK	1173	0768	2555								
LNKA	4467	4463									
S	0015										
6	0011	0114									
7	0022	0100									
ALEX	5856	283A	5809								
ALLK	1976	1277	1306								
ALSN	2482	25A2									
ALX2	5871										

CROSS-REFERENCE TABLE

CALXA	2838	2828									
CCNT	1229	1274	1438	1454	1917	1927	1930	1934	2091	2107	
CHRS1	4890	4900	4901								
CHRS2	4902	4897									
CHRS3	4905	4893	4895								
CHRSA	0219	2670	6645								
CHRST	4889	0219	4871								
CKEO1	4554	4550									
CKEOL	4546	4266	4335	4441	4472	4529					
CKRCD	2587	2573									
CLAR2	0956	0969	0976	3992							
CLF1	1711	1714									
CLEAR	1703	0160	1294	1361	1366	1393	1437				
CLERA	0160	1026	2419	2520	2560	2784	3739	4293	4333	4414	5102
	5140	5393	5427	5434	5437						
CLINE	1101	0792	1429	1465	1553	2623	3768				
CLP1A	0217	2195									
CLPR1	2915	0217	2920	2979							
CLPR2	2921	2916	2953								
CLPRG	2908	2847	2887								
CLRST	1687	1360	1680								
CMD13	2296	6567									
CMERR	3134	0268									
CMND	2619	2610									
CMNDF	2622										
COLON	1242	5109									
COMCE	0419	0248	2362	2392	2465	6591	6653				
COMCK	0414	0419	2336	6627							
COML	1209	3053	3124	3126	3132						
COMMA	6104	2720									
COMP1	3649	3643									
COMPL	3640	2989	3651								
COMPR	5905	5912	5918	5924	5930	5936	5942				
COMS	6597	4800									
COMTA	1628	1362									
CONS3	0872	0870									
CONST	0867	2333	5413								
CONT	1382	2283									
CONT1	1388	1383									
CONTK	1386	1384	2149								
CPLAC	1116	2504	4305	4947							
CR1	1040	1028	1804								
CRC	1670	1414									
CRC1	1678	1671	4768								
CRC1A	4768	3757									
CRC2	1676	1673									
CRCHK	1672	0228									
CRCKA	0228	2783	2808	5071	5101						
CRLF	1027										
CRLF1	1029	1681	2780	3732	3754	4347	4421	4454	5025	5122	
CRTN	0037	0038	0039	1029	1345	1654	1819				
CTERM	1683	1675									
CTYPE	1115	2505	4306	4939							
CXFC4	5437	5420									
DATA2	2336										
DATAS	2332	2338	4798								
DATCD	5386	5379									
DCMF1	3689										
DCMF2	3693	3699									
DCMP1	3675	3681									
DCMP2	3683	3674	3680								
DCMP3	3679	3675									
DCMP4	3702	3667									
DCMP5	3666	3686									
DCMP6	3672	3707									
DCOM1	3661										
DCOMA	0210	0567	2167	2188	2842	2861	2977				
DCOMF	3688	3664	3685								
DCOMP	3655	0210	2968								
DEFOP	3532	3467	3507								
DEFS	6507	4799									
DEL1	1427	1280	1571	1574							
DEL2	1426	1428									
DELC1	1518	1527									
DELET	2975	2284									
DELIN	2050	1279									
DELSA	0175	1433									
DELST	2842	0175	2596								
DFLAG	1175	2607	2944	2952	6322	6325	6359	6387	6575		
DIGCK	0650	0597	0599	2343	2458	4942	6307				
DIGIT	1233	4025	4035	4059	4124						
DIMS	6619	4801	6602								
DINCT	6435	6406	6409	6412	6418	6419					
DISP	6544	5039									
DISPA	5039	5034									
DIV	6114										
DLIMT	0648	0650									
DNARW	1473	1566									
DOLAR	1243	3896	4073								
DPERR	0043										

CROSS-REFERENCE TABLE

DROST	1102	0928	2069	2525	2572	2576	2820	2822	3443	3737	5137
	5400	5412	5416								
DSTRA	5698	5614									
DSTRT	1088	3657	5365	5383	5449	5645	5698				
E	0080	4089									
E1	0235	2836									
E10	0244										
E15	0249										
E20	0254										
E25	0259	4911	4912	6584							
E30	0264	3158									
E35	0269										
E40	0274	3486									
E45	0279	3646	4696								
E5	0239										
E50	0284										
EADD	5163	5147	5447								
EADD1	5172	5165									
EAND	5960	5773									
ECAL	5078	0215									
ECAL1	5081	5085									
ECAL2	5086	5082									
ECALA	0215	2860									
ECLR	2474	1313									
EDEG	4930	6540									
EDISP	4327	6544									
EDIT2	1564	1419									
EDITR	2848	2844	2845	2846	2898						
EER	1574	1565	1566	1567	1568	1569	1570	1571	1572	1573	1625
EEND	5050	4988	5057	6552							
EEQL	5930	5767	5937	6057							
EFAD	5840	5759									
EFDV	5866	5762									
EFET	1464	2165	2289								
EFMP	5860	5761									
EFOR	5195	6556									
EFOR1	5224	5212									
EFOR2	5210	5216									
EFSB	5750	5760									
EGORE	5924	5775									
EGOS1	5147	6742									
EGOS2	6736	6722									
EGOS3	6740	6725	6727	6729	6735						
EGOS4	5151	5146	5149								
EGOSB	6718	6554									
EGOTO	6719	6558									
EGRAD	4932	6539									
EGS1A	6742	6739									
EGTRT	5912	5764									
EIF	5144	6557									
EINPT	5391	6545									
ELARC	5894	5770									
ELET	5034	5036	6542	6563							
ELINA	0169	2807									
ELORE	5936	5776									
ELST	5918	5765									
EMSGA	2833	2789									
ENDS	0854	2433	4778	4779	4780	4791	6634				
ENDW	2046	1712									
ENEQL	5942	5766	5913	5956							
ENEXT	5272	5281	6555								
ENOT	6054	5774									
ENTE1	3439	3422									
ENTE3	3454	3460	3477								
ENTE4	3474	3464	3469								
ENTE5	3479	3455									
ENTE6	3485	3480									
ENTEA	0218	5197	5496								
ENTER	3420	0218									
ENTFA	0189	5591									
ENTFN	3449	0189									
ENTRY	2483	2961									
EOL	0038	0779	0861	1824	1831	1846	1928	2462	2595	4892	6249
	6589	6643									
EOLCK	0861	0856	2361	2391	2394	2659	2731	2751	6495	6519	
EOLL	1399	1346									
EOR	5950	5772									
EOR2	5954	5964									
EOST	0856	2337	2691	2725	4913	4922	6503	6528	6628	6636	
EPI	6046	6044									
EPRIN	4326	6547									
EPWR	5872	5763									
EQUAL	6119										
ER3	5176	0281	5179	5184	5188						
ER6	5823	0276	5795	5801	5816						
ER60	5377	0283	5381								
ER8	6017	0274									
ERAD	4927	4931	4933	6541							
ERRS	2836	2766	2771								
ERFAD	4968	4974	6548								

CROSS-REFERENCE TABLE

EREC	1437	1573								
ERES	604A	6045								
ERET	2831	2825	2826	2827						
ERN1	2818	2811								
ERR1	2772	2760								
ERR2	2781	2774	2776							
ERR3	2801	2795								
ERR4	2809									
ERR5	2820	2813								
ERRA	0177	0420	0433	0448	0477	0524	061A	0857	0A74	2084	2134
	2358	2389	2667	2692	2713	3423	4426	4665	4683	4867	4872
	4920	501A	5168	5176	5267	5377	5823	6017	6217	6454	6472
	64A6	6493	6498	6517	6522	6593	6600	6625			
ERRM	2835	2833	2833	2834	2834						
ERROR	2764	0177	2617	2658	2837						
ERSTR	5444	6546									
ERTN1	4990	5189	5322								
FRTRN	5177	6553									
ESCMA	5793	5756									
ESTAC	5128	4984	5095								
ESTOP	5070	5052	5123	6551							
ESTR	5781	5757									
ESTR1	57AA	5786									
ESYM1	3417	3446									
ESYMT	3407	3113	3434								
ET1	4419	4417									
ETAB	4405	4283									
EUMIN	5A7A	5768									
EVARG	5455	5570	5609								
EVMLF	5702	5665									
EW1	4760	4761	4763								
EW2	4733	4737	4764								
EWAIT	475A	6550									
EWRI1	4329	4440	4459								
EWRIT	443A	653A									
EXP	11A6	1187									
EXPOP	011A	6259									
EXPS	6115										
EXPTB	1580	1356									
F	00A1	3912									
FADD	2375	2351									
FALSE	5944	5914	5926	5932	5963						
FDAAA	0222	4969									
FDATA	5361	0222	5384								
FETCH	0911	4288	4407	4480	4663	4758	5144	5229	5234	5787	5793
	5799	6723									
FILL	4391	0201	4320								
FILL1	43A0	4391	4392								
FILL2	439A	43AA	4414								
FILLA	0201										
FIRST	1152	170A	4384	4527	4719	4726	4729				
FIX	1061	0320	5226	5262	528A	5297	5327				
FIXED	4939	2281	4943								
FLGRT	0073	1269	1002	3172	3431	4422	4952	4995	5601	6016	
FLOAT	1171	3829	4041	4430	44A4	4732	4736				
FLT	493A	2280									
FLT1	4066	4056									
FLT10	4097	4094									
FLT11	4064										
FLT2	4077	4055	4065								
FLT3	4073										
FLT4	4076	4072									
FLT5	40A0	4062									
FLT6	40A6	40AA									
FLT7	4063	405A									
FLT8	40A9	4083									
FLT9	4104	4101									
FLTFA	0193	4408	4759								
FLTRA	0192	034A	4498	4664	5794	5800	6724				
FLTSA	0195	0361									
FMTB	2364	2353									
FMTC	2361	2369	2374								
FMTD	2359										
FMTER	235A	0266	2372								
FMTF	2350	2344									
FMTOP	4712	443A									
FMTQT	2370	2366									
FMTS	2342	2363	47A2								
FMX80	5593	5602									
FMX81	5647	5630									
FMX82	5631	5625	5629								
FMX83	5644	5634									
FMX84	5633	5641									
FMX9	566A	5714									
FN	014A	2134	6450								
FNCK	6447	6163	6509								
FND	0639	0665	0670	1494	2439	2443					
FND1A	0224										
FSC3	6183	6169									
FSC31	6196	6191									
FSC4	6170	6164									

CROSS-REFERENCE TABLE

FSC5	6202										
FSC6	6203	6261	6285								
FSC7	6199	6156									
FSC8	6212	6195									
FSC9	6216	6210									
FSCA	0173	2386	2699	2747	2750	4917	6652				
FSCE1	0433	0249									
FSCE2	044A	0250	6271								
FSCE3	6217	0240	6220								
FSCE4	0479	0236	2951	6438	6610						
FSP1	4611	4601									
FSP2	4614	4608									
FSPN	4643	4604									
FSPO	4631	4622									
FSPR	4649	4453	4461	4494	4641						
FSPR1	4655	4651									
FTABL	2258	2375	2375	2376	2376	2377	2377				
FTEST	5300	5265									
FTMP1	1122	1223	5521	5535	5785						
FTMP2	1123	1228	5525	5530	5532	5781	6586	6587	6603	6608	6611
FTN13	6064	6566									
FWI	1877	1871									
FWAM	0157	2230	2508	2969	3452	3645	5632				
FWUP	1081	2168	2174	2175	2176	2177	2196	2198	2231	2509	2936
	2937	2970	3029	3052	3454	3666	5633				
FXFLA	0194	6028									
GET1	0758	0756									
GETAD	0626	2416	2975	4956							
GETCH	0751	0763	1049	1698	1941	4891					
GETL2	2467	2463									
GETLA	3025	2629	2984								
GETLN	2456	2400	2413	2436	2459	3025					
GETLR	2468										
GETSK	0778	2456	2724	4940	5423	6527	6633				
GETST	4865	0213									
GFLAG	1189	0617	0833	2345	2382	2571					
GFRST	0774	2568	2584								
GNEXT	0766	0431	0443	0446	0585	0608	0778	0844	0848	0854	0878
	0886	2021	2064	2342	2367	2409	2643	2958	4876	6194	6269
	6313	6326	6479	6654	6660						
GNUM	4431	451A									
GNUM1	4429	4307									
GOTO1	2385	2395									
GOTO2	2390	2393									
GOTO3	2394	2384									
GOTOS	2381	4793	4797								
GPARM	0316	0397	2759								
GSIG1	0884	0881									
GSIGN	0878	0867									
GSIGR	0888	0873	0884	0907							
GTCON	1049	0767	1742	1746	1923						
GTE	2272	6136	6136								
GTEA	6136	6288									
GTMP	0762	1522	1533	1650	1920	3965					
GTMP2	1697	2008	2023	2028							
GTOPD	3942	3822	3866	3890	3917						
GTOPP	0677	3594	4549	4564	5186	5195	6720				
GTR	6116										
GTRC1	2565	2557	2559								
GTRC2	2673	2579									
GTRC3	2580	2676									
GTRCD	2566	2637									
GTREQ	6127										
GTSTA	0213	2373	2730								
HELG1	1106	0568	2442	2446	5013	5015					
HELGA	056A	0559									
HSTPT	1087	0336	0337	0338	0339	0471	0472	0566	0913	0914	2422
	2603	3179	3213	3216	3230	3237	3238	3240	3542	3546	3547
	4511	5067	5068	5139	5203	5204	5206	5214	5222	5230	5240
	5272	5279	5280	5290	5300	5302	5312	5318	5430	5431	5465
	5470	5473	5488	5504	5507	554A	5579	5604	561A	5652	5653
	5655	5691	5694	5805	5825	5833	5834	5850	5855	5898	5976
	5977	6012	6018								
HUN1	2231	2238									
HUNT	2230	2083	2152	2160	2189	2206					
I	0067	3907	6319								
ICNT	030A	1407	1752	5111							
IFS	4917	4796									
ILFT	2639	2621									
ILET1	2650	2646	2662								
ILETS	2642	4802									
IMPOP	2683										
IMPY	0893	0604	3297	3301	5818						
IMPY1	0895	0906									
IMPYR	0907	0897	0899	0903							
INCH1	5434	5424									
INCHA	5440	5411									
INCHT	542A	5414	5440								
INCLN	3017	2849	3009								
INCLR	3033	3021									

CROSS-REFERENCE TABLE

INDEX	1234	4097	4099	4104						
INF	0077	0354	2487	5001						
INIT1	0564									
INITS	0559	0555	2163	2201	2515	2830	2848	4983	5051	
INITV	1445	1567								
INREA	0226	5399								
INRET	5401	0226	5432	5435						
INS0	1529	1513								
INS1	1532	1541								
INS2	1542	1520	1536							
INSRT	1509	1510	1570							
INTC1	0599	0609								
INTC2	0610	0600								
INTCK	0594	0586	2592							
INTEA	0178	2501								
INTEL	1125	0009	2502							
INTFL	0075	6721								
INTGR	1187	0596	0601	0602	0605	0607	0610	0615		
INTOV	0352	0349								
IOBUF	1144	0180	0180	2051	2051					
IOEND	2051	1834								
IOHEA	0229	2707								
IOHED	6650	0229								
IOST	1718	0225	1727	1801	1805					
IOSTA	0225	1032	1039							
JCMND	2632	2628								
JTRA	2074	2071								
JTRL	2075	2074								
K1	2242	2125								
K2	2243	2126								
K3	2244	2130								
K4	2245	2131								
KERR	2084	0244	2153							
KEYFG	1109	1753	1944	1952	1955	4734	4996	4998		
KEYIN	1913	1976								
KEYN1	1936	1919	1922							
KEYN2	1957	1945								
KEYN3	1944	1939	1971							
KEYN4	1970	1958								
KEYN5	1952	1947								
KEYN6	1954	1937								
KEYN7	1917	1935								
KEYPR	2061	1326								
KILLK	250A									
KILLP	2510	2966								
KILLV	2514	2965								
KTRA	2080	2065								
KTRL	2319	2080	2080							
L	1232	4117	4122	4141	4149	4169	4179	4180	4189	
L1	3739	3758								
L100	3752	3745								
L11	3844	3821	3836	3885	3895	3904				
L12	3838	3831								
L13	3862	3852								
L15	3887	3824								
L16	3912	3889								
L19	3903	3898	3908	3914						
L20	3905	3900								
L23	3814	3799	3862	3925						
L25	3927	3808								
L28	393A	3781	3789							
L29	3880	3854								
L2T1	1132	0885	0887	2092	2146	2997	2994	3289	3294	329A 3400
	4869	4875	5810	5819						
L2T2	1133	2988	2990	2993	3005	3011	3401	4272	4277	4579 5804
	5815	5817	5983	6005						
L3	3864	3857								
L4	3826	3816								
L6	3905	3878								
L8	3890									
L9	3917	3827								
LASTN	1183	0807	1498	2461	2467					
LBL1	2283	162A	1628							
LBOP	0119	2684	6355							
LHRAC	6122									
LOFF	0920	2145	2151	2159	2187	2513	2897	2967	5050	5631
LEFT	1885	1353	1987							
LETCK	0657	0845	1332	2645	6158					
LETS	2690	4781								
LETSN	269A	2648	2690							
LIMIT	0660	0651	4856							
LIST	372A									
LIST1	3734	2043								
LISTA	0167	2418								
LISTC	2409	2290								
LISTR	2043	1504								
LISTS	3762	3743								
LLIMT	0649	0657								
LLL	3783	3769								
LMSGA	2834	2797								

CROSS-REFERENCE TABLE

LN72	4770	4361									
LNFD	0072	1035	1823								
LNINC	1100	1281	1692	1295	1298	1427	1432	2404	2475	2518	2556
	2624	2991	3017	3023	3750	5073					
LNUM	0573	2383	2390	2425	2460	2466	4921	6635	6662		
LOCL1	1130	0395	0403	0455	0511	0660	0666	0690	0751	0759	1014
	1017	1205	2086	2101	3402	3466	3541	3549	3566	3797	4253
	4470	4482	4486	4487	4488	4489	4493	4620	4631	4639	4652
	5028	5117	5154	5155	5158	5349	5563	5566	5728	5733	6711
LOCL2	1131	0396	0401	0405	0717	0729	1206	1659	1830	3403	3918
	3922	4633	4649	4655	5726	5729					
LPARN	6123										
LPCK	0425	0432	5200	6353							
LPCKE	0432	6514	6551								
LPOP	0120	0428									
LPTR	1627										
LROMA	0123	0517	5675								
LSS	6117										
LSSEQ	6128										
LSTAK	1084	3181	3183	3186	3187	3188	3200	3209	3221	3224	3227
	5064	5089	5132	5136	5991	5996					
LSTC1	2413	2428									
LSTC2	2425	2411									
LSTPT	1086	0474	0484	0485	0491	0492	0563	2420	3054	3164	3165
	3171	3180	3232	4506	4534	4535	5059	5066	5078	5088	5128
	5133	5177	5178	5527	5539	5540	5573	5574	5580	5581	5611
	5613	5669	5672	5676	5682	5683	5685	5686	5707	5709	5727
	5730	5732	5783	5798	5821	5824	5835	5894	5997	6001	6437
	6738	6740									
LTADD	2678	2545									
LTR	0944	6301	6453	6507							
LTRUE	5938	5943	5953								
LWAM	1080	0552	2498	6605							
M1	0124	2296	4562	0825	0832	0883	1010	1064	1408	1464	1531
	1793	1954	1960	1980	2193	2497	2535	2604	2619	2673	2722
	2734	2810	2824	3004	3055	3068	3072	3095	3097	3102	3104
	3136	3190	3201	3235	3443	3463	3609	3695	3698	3749	3798
	3801	3828	3931	4057	4060	4068	4136	4171	4285	4302	4611
	4617	4628	4637	5138	5499	5554	5594	5603	5692	5796	5802
	5806	6152	6246	6253	6364	6366	6377	6473	6671		
M10	0131	1319	1795	4100	4161	5202	5675				
MDIM	3249	0208	3123	3211							
MDIM1	3307	3292									
MDIM2	3305	3291	3293	3303	3310						
MDMA	0208	6585									
MDMA	3710	3074									
MER10	3198	0272									
MER11	5018	0284									
MER5	3108	0269	3088								
MER8	0477	0479	4523								
MER9	3304	0273									
MERR	2837	3108	3134	3198	3304	3368	3385				
MINIT	0552	2514	2829	3040							
MINS	6112										
MINUS	1239	4095	4232								
MISOF	2389	0259									
ML2L	3089	3081									
MLOP2	3149	3066	3071	3089	3111	3117	3151				
MLOP5	3084	3079									
MLOP7	3073										
MLOP8	3092	3080									
MLOP9	3113	3085	3105								
MLP71	3154	3077									
MLP72	3075	3157									
MO1	4753	4745	4746								
MO2	4750	4752									
MODE	1090	1270	1315	1394	2434	2577	2583	2587	2613	2625	2655
	2660	2781	2819	2823	2856	2858	3056	4052	4150	4303	4329
	4514	4517	4520	4985	5007	5035	5148	5648			
MOUT	4745	4075	4076	4177	4194	4211	4399				
MOUTA	1626	1364									
MOVC1	3549	3554									
MOVC2	3555	3550									
MOVCK	3539	3214	3409	3427	3441						
MOVST	5726	5616	5704								
MREM	3713	3567									
MSFLG	1174	2602	2698	6204	6213	6222	6255	6399	6407	6421	6577
	6583										
MSK65	1622	1316	1327								
MSKIP	3060	3065	3150								
MSLPT	4769	4394	4413								
MSTND	3067	3152									
MVTOH	0821	0828	1406	2172	2902	5220	5681	6005			
N	4771	3914									
N1	4150	4143	4146	4148							
N2	4137	4131									
N4	4145										
N6	4147										
N7	4164	4158									
NCALC	2617	0237									

CROSS-REFERENCE TABLE

NDATA	2334	0257								
NDISP	1167	1023	1653	1667	2523	2558	2831	4255	5120	
NDWRT	1023	1413	2779	3753	4465	4477	4526	4727	5024	
NEWK	2167	2161								
NEWK1	2181	2222								
NEXT1	0509	3465	3790	3919	5096	5560				
NEXTA	0691	0503	0808	2874	2908	3008	3762	4689	5404	5650
NEXTL	0499	3565	5026	5348						
NEXTS	4910	4794								
NGOTO	5168	0278								
NINIT	3423	0275								
NM1	4036	4027								
NM2	4035	4030								
NM3	4026	4039								
NM4	4040	4032	4034							
NMOTA	0200									
NMOUT	4023	0200	3835	4433						
NN	4165	4155	4162	4175						
NN1	4166	4160								
NN10	4184	4182								
NN11	4199									
NN2	4178	4130	4164							
NN3	4159	4153								
NN4	4195	4185	4191	4192						
NN5	4186	4188								
NN6	4217	4199	4213							
NN7	4212	4216								
NOFOF	0857	0241	2663							
NOFMT	4696	4710								
NOPCD	5193	5256								
NORML	4959	2286								
NOST	4867	0253								
NOTA	6135	6227								
NOTOP	6126									
NPRIN	1168	1016	1020	1660	1666	2524				
NPWRT	1020	1375	2550	2790	2798					
NSTMT	2658	0239	2649	2656						
NUM	1181	1182	6677	6691	6694	6704	6710			
NUMCA	0211	0869	6209							
NUMOA	0212	2335	6211							
NXTIL	3593	3603	3605							
NXTDT	1089	3656	5372	5373	5448	5646				
NXTOP	3565	3064	3628							
NXTP1	3583	3149	3573	3593	3595	3611	3618	3631		
NXTP2	3575	3591								
NXTP3	3598	3588								
NXTP4	3612	3602								
NXTPR	3558	3572								
NXTST	1190	1386	3039	4986						
OBLNK	0715	3784	3811	3871	4237	4383				
ODGIT	1000	4081	4086	4186	4214					
OF	0961	0952								
OFA	2396	2387								
OFFA	1629	1371								
OFLT	4051	4165								
OFXD	4114	4043								
OL1	4730	4722								
OL2	4731	4725								
OL3	4728	4735	4738							
OLINE	4719	0169	4340							
ONE	0045	5234	5900							
ONEXT	0716	1002	1288	1352	1400	1658	1829	1847	3748	3771	3789
	3861	3881	3884	3892	3903	3913	3915	4085	4090	4098	4103
	4196	4201	4233	4750	5110						
OPC1	6029	5294	6032								
OPC2	6031	6024									
OPC3	6033	6014									
OPCH1	6021	5263	5289								
OPCHA	0171	0912									
OPCHK	6012	0171	5571	5972	5982						
OPDMK	0079	0678	4281	5483							
OPMSK	0076	0466	0513	0546	2242	3589	3592	3819	3820	3858	4561
	4568	4593	4615	4704							
OPTYP	1182	6304	6324	6333	6465						
OR	2267	6134	6134	6135	6135						
ORA	6134	6281									
OROP	6124										
OS1	4234	4231									
OSIGN	4230	4080	4178								
OTPTR	1078	1706	1735	1863	1864	1873	1878	1888	1890	1899	1902
OUTCR	0717	1350	1526	1538	1545	1950	1968				
OUTIA	0179	1301	2423	2562	2793	2800	3783	3842	4105	5108	
OVCHA	0216	2171									
OVCHK	2942	0216	2901	2929							
OVCK	4415	4310									
OVFW	0971	0965									
P1	1801	1812	1815	1818	1820	1826	1828	1836	1844		
P2	1837	1832									
P3	1816	1810									
P4	1806	1814									

CROSS-REFERENCE TABLE

P5 1842 1822
P6 1835 1845
P7 1846 1802 1806
PAL 1458 1572
PARMA 1091 5619 5719
PARMN 1092 3659 5489 5661 5717
PBPTO 1098 0561 0524 2598 2915 2924 2942 2947 3032 3184 3233
3481 3684 4706 6614
PBPTR 1095 0796 0925 1551 2221 2440 2444 2879 2880 2925 2927
2998 3031 4047 3062 3482 3496 3529 3626 3672 3683 5017
5104 5345 5622 6597 6613
PBUFA 5696 5670
PBUFF 1096 0795 0923 1447 1496 1550 1554 2180 2181 2199 2200
2215 2218 2599 2876 2895 2996 3030 3046 3049 3127 3154
3450 3640 4655 3705 5445 5626 5644 5678 5696 6598 6612
PBUFO 1097 0922 2216 2933 2935 3130 3483 3702
PEND 1150 0719 1268 1286 1290 1399 3747 3765 4362 4416 4513
4522
PEXM1 2552 2530 2544
PEXM2 2525 2832
PEXM3 2537 2526
PEXMA 0165 2405 2424 5074 5401
PEXMK 2522 0165 2567 2569 2636 2850
PFLAG 1176 2723 2735 2737
PFOR1 4361 4269
PFORM 4352 4287 4290 4406 4479
PGIN0 0832 2360 4946
PGINT 0580 0834 2348 2355 6365 6376
PIZZ 0044
PIA 6050 6046
PLACE 1169 3834 3864 3872 4028 4051 4128 4156 4159 4234 4429
4505
PLINE 5101 5023 5116
PLOTE 4673
PLUS 1238 4093
PLUSS 6111 6257
PMODE 1114 1015 1027 1367 1370 1410 1670 1672 2507 2773 4365
5118
POINT 1240 4084 4200
POV 4297 4312
PPEF 1151
PR11 4307 4304
PR17 4348 4345
PR2 4344 4336 4337 4339 4539
PR3 4338 4341
PR5 4280 4271
PR6 4302 4295
PR7 4310 4279
PRADD 1093 0483 0490 0497 0504 1382 1392 1395 2818 3051 3060
3490 3506 3525 3585 3622 3625 3662 4314 4547 4552 4971
4990 5012 5056 5094 5103 5106 5150 5152 5182 5237 5245
5344 5362 5370 5378 5382 5398 5405 5419 5510 5651 5664
6718 6734 6737
PRALL 1366 1458
PRCHK 4365 4346 4420 4723
PRCHR 4382 4366 4368 4395
PREED 1789 1273
PRFLG 1117 1214 1674 2805 4284 4328 4352 4368 4367 4670
PRIN1 2717 2736 2752
PRIN2 2726 2718 2721
PRIN3 2737 2728
PRIN4 2749 2741
PRIN5 2751 2748
PRINN 4266 4338
PRINS 2712 4783 4786
PRNTS 2722 2657 2712
PROD 1206 0900 0905
PRTOP 2684 2651
PTAPE 2532 2282
QMARK 2681 2528
QOV 4272 4299
QTCH1 4577 4242 4563 4570
QTCH2 4590 4575
QTCH3 4573 4565
QTCHA 0230
QTCHK 4560 0230 4270 4475
QTOP 0109 0161 0519 1244 2483 2683 3590 4562 4569 4621 4473
6180
QUOT 6103
QUOTE 4242 3882
R 1231 4054 4061 4064 4066 4077 4082 4087 4127 4132 4137
4166 4197 4205 4208 4209 4212 4215
RBASE 1235 4119 4123 4139 4202
RBOP 0117 2686 2745 6383
RBRAC 6107
RBUFF 1145 0181 0181 0198 0202 0202 1627 1627 2046 2866 2868
2869 4769 4769 4770 4770 6641 6642
RCAL 2082 2067
RCOUT 1118 4600 4612 4650 4673
RDYDA 0166 1390 1552 1848 2190 2202 2447 3048 4941 4948 4963

CROSS-REFERENCE TABLE

```
RDYPT   2516   0166   2980   3013
RE1     1740   1761
RE2     1762   1755   1758
RE3     1743
RE4     1747
READS   6624   4784   4787   6629
REF     1285   1283
REFD    0161   2499
REFD1   1267
REFD2   1327   1317   1320
REFD3   1336   1328   1333
REFD4   1299   1296
REFD5   1344   1330   1335   1421   1451
REFD6   1306   1293   1299   1354   1365   1441   1508   1530   1546   1857   1862
        1866   1877   1879   2090   2110   2119   2122
REFD7   1294   1284
REFD8   1417   1358
REFD9   1352   1348
REFDD   1356   1338
REFDE   1277   1376
REFDG   1308   1304
REFDL   1124   2500   2566   3751
REFDT   1414   1409   1411
REFRA   4767   4733
REFRS   1735   1970   4767
REMS    6640   4798
RENM1   2994   3012
RENM2   2997   3010
RENM3   3007
RENM4   3011   2999   3001
RENUM   2984   2285
RES     1202   0184
RESA    1638   1452
RESB1   0538   0544   2082
RESBP   0544   2385   2650   6185   6231
RET2    0301   0293   0810   0811   2464   4905
RETA    1668   1677
RETN2   0294   0417   0444   0459   0619   0673   0781   0850   0871   1415   1728
        1841   1988   2013   2236   2468   2703   3261   3384   3397   3614   3634
        3633   4315   4321   4369   4409   4418   4424   4553   4589   4692   6340
        6403   6456   6713
RETN3   0293   3269   3277   3596   4554   6315   6336
RETNN   5707   5185
RETNS   6527   4792
RFNX    2124   2097
RFNX1   1453   2143
RIGHT   1899   1985
RIT3L   1655   1647
RITE    2707   4777
RITE1   1650   1664
RITE2   1663   1656   1661
RITE3   1665   1655   1713
RITE4   1660   1657
RL3     4128
RL4     4124   4121
RND     3992   4079   4145
RND2    3989   3997
ROUND   0987
RPARN   6106
RPCK    0438   0447   6237
RPCKE   0447   2708   6382   6518
RPOP    0115   0441
RRUN    2145   2128   2133
RSAVE   1077   0294   0298   0301   0305   4618   4627   4629   4640
RSCH1   6301
RSCH2   6006   5993
RSCHK   5988   5466   5572   5621   5897
RSLT    1448   1568
RSLT1   1440   1455
RSLTE   1158
RSPTR   1203   0295   0302   0316   2511   2553   2764   2765   2855   3515   4006
        4616   4982   5553   5556   5582   5583   5593   5596   5598   6284   6392
        6394   6395   6410   6411   6413   6712
RSTAK   1201
RSTKA   0158   2510   2552   2854   4981   5599   6391   6414
RSTRS   6633   4785
RSTT    3522   3485   3688
RSULT   0336   4970   5417   5788
RUN     3039   0163   2292
RUNA    0163   1446   2156
S       0382   3909   6317
SALT    6671   0207
SALT1   6676   6681
SALTA   0207   2578   2609   2620   2674   6173
SAROW   1550   1473   1489
SAVBP   0538   2061   2381   2647
SAVE    1213   2574   2675   5363   5369
SAVEA   1110
SAVEB   1111
SBADR   1179   0531   0538
```

CROSS-REFERENCE TABLE

SHLOC	4595	4590								
SHPTR	1172	0377	0381	0382	0383	0458	0532	0541	0545	0547	0574
	0580	0611	2775	0836	0855	1221	2332	2350	2368	2611	2612
	2652	2653	2654	2669	2710	2729	2733	2746	2772	2791	2809
	2864	4510	4524	4868	4898	4899	4902	5081	5410	6168	6175
	6185	6188	6189	6193	6199	6230	6232	6241	6280	6286	6356
	6384	6385	6386	6460	6576	6650	6655	6659	6661		
SHPUD	0382	0442	0583	0616	2744	4906					
SHSC1	6379	6370	6373	6378							
SHSC2	6382	6428	6431								
SHSC3	6405	6361									
SHSCK	6353	6338									
SHSCM	6403	6388									
SHUFA	0199	0774	0791	1403	1430	2862	2889	5080			
SCCNT	5892	5895									
SCERR	4665	0277									
SCMMA	6128	6372	6427								
SCRAT	2958	2291									
SCRK	2967	2963									
SCRP	3029	2512	2600								
SEC	1105										
SERF1	3516	3473									
SERFN	3506	3451	3484	3514							
SERFR	3498	3509									
SFLAG	1178	2608	2700	6247	6266	6363	6369	6391	6470	6474	6578
	6584										
SFMT	4661	4439									
SFMT1	4681	4709									
SFMT2	4694	4680									
SGNS	1004	5908									
SIGN	1184	0868	0872	1185	6208						
SLCOD	1113	1041	1721	1790	1838	2427	2540	2581	2787	4354	4358
	4672										
SLEND	0202	1267	1529	3764							
SLWST	0484	4353	4355	4448	4471	5084	5087	5090	5092	5093	5480
	5557	5565	5597	5606	5608	5896	6719				
SM2L	3362	3352									
SMCLN	6105										
SNFLA	0197	3832	5367	5467							
SPAGE	6691	6674	6676								
SPLFA	0196	6031									
SPROD	1212	5396	5397	5403							
SSRPT	1180	0513	0540								
SSTT	3490	3449	3661								
SSYM1	334B	3355									
SSYM2	3376	3362	3367	3373							
SSYM3	3385	0270	3377	3380							
SSYM4	335B	3349									
SSYM5	3369	3364									
SSYM6	3353	3360									
SSYMA	0170	5196	5495								
SSYMT	332B	0170	3084	3104							
ST	1162	1211	1212	1213	4515	4519	4661	4708	5499		
STADD	5699	5702	5711								
STAR	2685	2590									
STAR2	0939										
START	0162	0007									
STOPG	2686										
STEP	1380	1459									
STEPA	6098	6496									
STEXP	0463	2352	2357	2365	2388	4919					
STMAX	0976	0971									
STMP1	3401	3343	3396	3691							
STMP2	3402	3331	3348	3361	3365	3371					
STMP3	3403	3334	3376	3381							
STMT	2609	2640									
STMTI	261A	2614	2616								
STOPE	5115	5014	5016								
STOPS	2434	2435	4790								
STOR1	0347	0356	0358								
STOR2	0359	0346									
STORE	0340	5227	5298								
STOUT	4253	3935	3979	4278							
STP1	5123	5115	5119								
STPFL	1108	1380	1687	1692	1718	2538	2626	2814	3755	5010	
STPL	1459	1569									
STRI1	6470	6463									
STRID	6460	6337									
STRNP	6472	0252									
STROP	0373	6312	6334	6467	6478	6512					
STSRH	0453	0410	0429	0467	0476	6228	6282	6492	6497	6698	
STTOP	5982	5688	5878	6054							
SUMP	2215	0227	2147	2154	2162						
SUMPA	0227	5643									
SVAL1	3398	3353									
SVALU	3392	3345	3346	3399	3693						
SWITH	5371	5361									
SYF10	6498	0263									
SYF12	0874	0247									
SYF13	6625	0260									

CROSS-REFERENCE TABLE

SYE14 4A72 0254
SYE15 2713 0242
SYE25 0618 0238 059A 0606 0612 0614 0630 0838 2534 3024
SYM1 039A 0404
SYM2 0407 0402
SYMC1 0393 0414 6262 6371 6426 6487 6520 6656
SYMC2 0391 3425 0438 2719 6218
SYMCK 0395 0392 6256
SYME2 336A 0271
SYMTP 1083 0553 0564 2949 3061 3146 3241 3392 3408 3555 3557
5199 5274
SYN13 4A06 6568
SYNE2 2692 0265
SYNE3 6600 0267
SYNE4 6454 0255 6508 6510
SYNE5 6517 0256 6516
SYNE6 6522 0264 6489
SYNE7 4920 025A
SYNE8 6486 0261 6485
SYNE9 6493 0262
SYNTX 259A 2586 2589
SYSER 0524 0235
SYTSA 0209 3508
SYTSH 5343 0209 5257 5352 5380
TA9 6093 2753 2753
TABA 2753 2739
TABER 4426 0279 6060
TABOP 4311 4282
TBS2 2022 2019 2035
TBS4 2034 2031
TBS5 2020 2010
TBS6 200A 2025
TBSR1 2014 1337 1357
TBSRH 2015 016A 1420
TEMP 1135 1136 261A 2632 2865 2882 2886 2893 2900 6192 6672
6695 6701 6709
TEMP1 1137 0847 1061 4874 4890 4896 5198 5275 5330 5636 5642
6179 6190 6252 6310 6311 6332 6368 6380 6447 6455 6464
6475
TEMP2 113A 0821 0849 2943 4889 4894 5213 5264 5295 5306 5309
5343 5350 5674 5999 6314 6327 6335 6468 6513 6582 6589
TEMP3 1139 0794 0802 0812 0822 1404 2169 2877 2891 2892 2910
2976 2978 4699 4707 5205 5680 5994
TEMP4 1140 0584 0594 0613 0627 0637 063A 0824 0827 1402 2194
2417 2911 2918 2919 2921 2946 4960 5219 5221 5671 6003
TEMPS 1134 0498 0499 0502 0505 0506 0508 0512 0677 1224 1225
1226 1227 1229 2605 2997 3000 3006 3007 3092 3093 3094
3099 3100 3101 3461 3462 3470 3474 3492 3510 3516 3523
3569 3570 3575 3579 3580 3583 3584 3612 3613 3648 3678
3734 3740 3766 3774 3775 3778 3785 3792 3793 3794 3814
381A 383A 3839 3844 3845 3849 3927 3932 3937 3942 4280
4313 4316 4405 4443 4546 4560 4566 4567 4571 4577 4580
4581 4587 4662 4674 4675 4677 4679 4691 4698 4972 5079
5096 5163 5164 5171 5180 5236 532A 5347 5364 536A 5371
5374 5394 5406 541A 5429 5444 546A 5482 5486 5509 5512
555A 555A 5610 5612 5624 5628 5640 5649 5656 5662 5666
5684 6151 6202 6239 6270 6390 6396 6397 6400 6401 6423
6424 6435 6436 6439 6731 6732
TEPPP 1211 5407 5421 5428
TERM 1149 1034 1676 1684 2521 2785 2815 4291 4294 4331 4334
4344 4423 5391
TEST 5327 5259 5276
TFLAG 1103 1271 1693 2517 2532 2775 2803
TFLG1 1107 4953 4959 4962 5002 5006 5019 5021
TMENA 4923 491A
TIMES 6113 6657
TMP 1215 1440 1517 1518 1521 1523 1524 1532 1534 1539 1649
1736 2042 2099 3962 397A 4747 4751
TMP1 1216 1231 1648 1663 1740 1750 1756 1796 1811 3961 3974
TMP10 1225 3763 3773 3846 3929 3936
TMP11 1226 3735 3809 3812 3869
TMP12 1227 3817 3887 3893 3901
TMP13 122A 3796 3806 4176 4184 4187
TMP2 1217 1232 1738 1813 1816 2016 2020 2536 4297 4309 4380
4583 4588
TMP2A 2042 1697 1745
TMP3 121A 1234 1744 1749 2022 2024 3964 3973 3975 3976
TMP4 1219 1235 1936 1959 1963 2015 2032 2034
TMP5 1220 1233 1303 1318 1323 1325 1331 1341 1344 1359 1363
1794 1839 1921 1924 1942 1948 1964 1983 2014 2173 2235
TMP6 1221 1321 1329 1449 1512 1981
TMP6A 1222 201A 2026 474A 4749
TMP7 1223 1503 3729 3741 3772
TMP9 1224 3791 3875
TMPA 0203 0762 1525 1537 1544 1741
TMPOP 0184 4300 4431 4648
TOA 6133 6491
TORLN 1632 1710
TRAC2 4952 4955
TRAC3 4961 4954

CROSS-REFERENCE TABLE

```
TRACE   4955    2287
TREGE   1157
TRUE    5900    5925    5931    5938
TSRET   6699    6693
TSRHA   0168    0453    0463    2066    2740
TSTPT   1085    5062    5091    5130    5472    5605    562M    5687    584A    5854    5879
        5884    5887    5944    5988    599M    6013    6033    6035
TTY     1039    1030    1662    4502
TYPE    1210    3073    1086    3114    356A
TYPEC   460M    4444
TYPEL   007A    2743    3815
U1      1493    1491
U3      1498    1477    1494
U4      1487    1474    1490
U5      1501    1478    1479    1488    1497
UFLAG   1177    6153    6196    6216    6225    6251    6402    6405
UKDEF   2665    2591
UKERR   2667    0243
UKPTR   1094    0920    0927    2217    2546    2665    2928    2932    3128    3129    3479
        3703    6601
UNFOL   611A
UNFW    0967
UNITS   1104    2506    3045    4928
UNMIN   6120    6219
UNSTK   0491    4357    4359    4481    4483    5060    5061    5063    5065    6736
UPARA   0191    5873
UPARW   1489    1565
UPLUS   6121
VALUE   1082    0554    1166    3174    3231    3391    3425    3435    3439    3442    3692
VALUP   1599    1336
VAR01   6316    630A
VAR03   6335
VAR04   6337    6306    6329    6331
VAR06   632A    6321
VARDA   0214    4910    6579    6624
VAROP   6301    0214    6155    6484    6515
WAITS   6502    47A9    6523    653M
WBUFF   0309    4525
WIDTH   1170    1230    3867    3876    4071    4074    4174    4495
WRIT    1647    0159
WRIT2   4444    4455    4466    4532
WRIT4   4472    4478
WRITS   4479    4476
WRIT6   4529    4503
WRIT7   4505    4496
WRITA   4536    4474    4531
WRITB   4490    44A5
WRITE   0159    1019    1021    1024
WRITF   4470    4449
WRITN   4456    4445
WRITQ   4465    4446
WRITS   4454    4456
WRITX   4461    4451
XC      1197    1215    1216    1217    1219    3491    3493    3495    3497    3522    3524
        3526    352A
XEC     49A1
XEC3    49A6
XEC4    4995    0164    503A    5151    5159    5319    5438    5450    5705    6543    6549
        6559    6560    6561    6562
XEC4A   0164    434A    4760    4765    4929    4973    6741
XEC5    5014    5011
XEC6    5017    0221
XEC6A   0221    1396
XEC7    5006    5003
XEC8    5026    5020    5022
XEC9    5027    5097
XECA    1245    13A7    3242
XFAR1   0324    0341    0343    0347    0359    2758    4023    4432    5232    5242    597A
        6030
XFAR2   0363    0351    4916    2816    4005    404M    4301    5235    5736    6047    6049
XKEY    2159    207A
XLST    2206    2077
XLST1   2419    2207    2211
XRUN    2151    2075
XSCR    21A7    2076
XSCR1   2194    0220
XSCRA   0220    2971
YC      1199    1219    1220    1222    3444    351A    3519    5500    5501    5607
ZAP     0152    0941
ZONK    0954
ZONK1   095A    0955
```

#01 9830 MATH ROUTINES

```
0001                ASMB,A,L,C
0002*  MATH ROUTINES 3-20-72
0004*
0005  00000                 ORG 0
```

#01 9830 MATH ROUTINES

```
0006*
0007  00000 164121        JMP START,I   START OF MONITOR
0008  00001 043060        OCT 43060     CONSTANT NEEDED BY PROCESSOR
0009  00002 000000        BSS 1
0010*
0011  00003 000000 C6      OCT 0         ONE
0012  00004 010000         OCT 010000    0000
0013  00005 000000         OCT 0         0000
0014*
0015  00006 000000 C5      OCT 0000000   PI/2
0016  00007 012560         OCT 12560     1574
0017  00010 074543         OCT 074543    7963
0018  00011 023200         OCT 023200    2680
0019*
0020  00012 000001 .1      DEC 1
0021*
0022  00013 000400 C7      OCT 000400    180/PI
0023  00014 053451         OCT 053451    5729
0024  00015 053571         OCT 053571    5779
0025  00016 050450         OCT 050450    5128
0026*
0027  00017 000002 .2      DEC 2
0028  00020 000003 .3      DEC 3
0029  00021 000004 .4      DEC 4
0030  00022 000005 .5      DEC 5
0031  00023 000006 .6      DEC 6
0032  00024 000007 .7      DEC 7
0033  00025 000010 .8      DEC 8
0034  00026 000011 .9      DEC 9
0035  00027 000013 .11     DEC 11
0036  00030 000014 .12     DEC 12
0037  00031 000015 CRTN    OCT 15
0038  00031         EOL     EQU CRTN
0039  00031         .13     EQU CRTN
0040*
0041* POINTERS TO FUNCTIONS CONSTANTS
0042*
0043  00027         DPERR EQU .11       POINTER TO 180/PI
0044  00023         PI/2  EQU .6        POINTER  TO PI/2
0045  00020         ONE   EQU .3        POINTER TO THE CONSTANT ONE
0046*
0047*
0048  00032 000017 .15     DEC 15
0049  00033 000020 .16     DEC 16
0050  00034 000025 .21     DEC 21
0051  00035 000026 .22     DEC 22
0052  00036 000027 .23     DEC 23
0053  00037 000034 .28     DEC 28
0054  00040 000037 .31     DEC 31
0055  00041 000040 .32     DEC 32
0056  00042 000041 .33     DEC 33
0057  00043 000042 .34     DEC 34
0058  00044 000045 .37     DEC 37
0059  00045 000050 .40     DEC 40
0060  00046 000051 .41     DEC 41
0061  00047 000053 .43     DEC 43
0062  00050 000055 .45     DEC 45
0063  00051 000056 .46     DEC 46
0064  00052 000060 .48     DEC 48
0065  00053 000072 .58     DEC 58
0066  00054 000077 .63     DEC 63
0067  00055 000111 I       OCT 111
0068  00056 023420 .1E4    DEC 10000
0069  00057 001750 .1000   DEC 1000
0070  00060 000144 .100    DEC 100
0071  00061 000012 .10     DEC 10
0072  00061         LNFD    EQU .10
0073  00062 100000 FLGBT OCT 100000
0074  00063 000143 .99     DEC 99
0075  00064 100004 INTFL OCT 100004
0076  00065 076000 OPMSK OCT 76000
0077  00066 077777 INF    OCT 77777
0078  00067 100037 TYPFL OCT 100037
0079  00070 101777 OPDMK OCT 101777
0080  00071 000105 E      OCT 105
0081  00072 000106 F      OCT 106
0082  00073 000123 S      OCT 123
0083  00032         A17   EQU .15
0084  00033         A20   EQU .16
0085  00074 000022 A22    OCT 22
0086  00075 000023 A23    OCT 23
0087  00035         A26   EQU .22
0088  00036         A27   EQU .23
0089  00076 000036 A36    OCT 36
0090  00040         A37   EQU .31
0091  00043         A42   EQU .34
0092  00077 000044 A44    OCT 44
0093  00100 000052 A52    OCT 52
0094  00053         A72   EQU .58
0095  00101 000100 A100   OCT 100
```

```
          #01  4K30 MATH ROUTINES


0096  00102 000133 B133   OCT 133
0097  00103 000140 B144   OCT 144
0098  00104 000177 B177   OCT 177
0099  00105 000377 B377   OCT 377
0100  00013          B400   EQU C7
0101  00106 000777 B777   OCT 777
0102  00107 001000 B1000  OCT 1000
0103  00110 001400 B1400  OCT 1400
0104  00111 001740 B1740  OCT 1740
0105  00112 001760 B1760  OCT 1760
0106  00113 001774 B1774  OCT 1774
0107  00114 001775 B1775  OCT 1775
0108  00115 001777 B1777  OCT 1777
0109  00116 002000 QTOP   OCT 2000
0110  00117 002400 B2400  OCT 2400
0111  00120 003000 B3000  OCT 3000
0112  00121 004000 B4000  OCT 4000
0113  00122 006000 B6K    OCT 6000
0114  00004          B10K   EQU C6+1
0115  00004          BPOP   EQU B10K
0116  00123 012000 B12K   OCT 12000
0117  00123          BPOP   EQU B12K
0118  00124 032000 EXPOP  OCT 32000
0119  00125 050000 LBOP   OCT 50000
0120  00126 052000 LPOP   OCT 52000
0121  00127 170000 B17AK  OCT 170000
0122  00130 013777 FROMA  OCT 13777
0123  00131 000000 LROMA  BSS 1
0124  00132 177777 M1     DEC -1
0125  00133 177776 M2     DEC -2
0126  00134 177775 M3     DEC -3
0127  00135 177774 M4     DEC -4
0128  00136 177773 M5     DEC -5
0129  00137 177772 M6     DEC -6
0130  00140 177767 M9     DEC -9
0131  00141 177766 M10    DEC -10
0132  00142 177765 M11    DEC -11
0133  00143 177764 M12    DEC -12
0134  00144 177763 M13    DEC -13
0135  00145 177762 M14    DEC -14
0136  00146 177761 M15    DEC -15
0137  00147 177751 M23    DEC -23
0138  00150 177747 M25    DEC -25
0139  00151 177740 M32    DEC -32
0140  00152 177734 M36    DEC -36
0141  00153 177720 M48    DEC -48
0142  00154 177700 M64    DEC -64
0143  00155 177640 M80    DEC -80
0144  00156 177634 M100   DEC -100
0145  00157 177400 M256   DEC -256
0146  00160 176000 M1024  DEC -1024
0147  00161 154360 MAXSN  DEC -10000     -(MAXIMUM ALLOWABLE LINE NUMBER)
0148  00162 043116 FN     ASC 1•FN
0149*
0150*  MATH CONSTANTS THAT ARE NEEDED ON BASE PAGE
0151*
0152  00163 177401 ZAP    OCT 177401
0153  00164 063440 MASK1  OCT 063440
0154  00165 061400 MASK5  OCT 061400
0155*
0156  00166 000000 FWAM   BSS 1
0157  00117          RSTKA  EQU B1760      START OF RETURN STACK
0158  00167 000000 WRITE  BSS 1           PRINTER AND DISPLAY DRIVER
0159  00170 000000 CLERA  BSS 1           CLEAR SCREEN
0160  00116          KEED   EQU QTOP       KEYBOARD DRIVER QTOP IS = 2000B
0161  00121          START  EQU B4000
0162  00171 000000 RUNA   BSS 1           PHASE Z:SYMBOL TABLE AND STORAGE
0163  00172 000000 XEC4A  BSS 1
0164  00173 000000 PEXMA  BSS 1
0165  00174 000000 RDYDA  BSS 1
0166  00122          LISTA  EQU B6K
0167  00175 000000 TSRHA  BSS 1           TABLE SEARCH
0168  00176 000000 FLINA  BSS 1           OUTPUT I/O BUFFER
0169  00177 000000 SSYMA  BSS 1           SEARCH SYMBOL TABLE FOR SYMBOL
0170  00200 000000 OPCHA  BSS 1
0171  00201 000000 FORMA  BSS 1
0172  00202 000000 FSCA   BSS 1
0173  00203 000000 ACTSA  BSS 1
0174  00204 000000 DELSA  BSS 1
0175  00205 000000 AERRA  BSS 1           ALTERNATE ENTRY FOR ERROR
0176  00206 000000 ERRA   BSS 1
0177  00207 000000 INTEA  BSS 1
0178  00210 017036 OUTIA  DEF OUTIN       OUTPUT AN INTEGER
0179  00211 000000 .BUFA  BSS 1           FIRST CHAR OF I/O BUFFER
0180  00212 000000 .SLBF  BSS 1            REFRESH BUFFER POINTER
0181  00213 001744 AR1A   DEF AR1
0182  00214 001754 AR2A   DEF AR2
0183  00215 000000 TMPOP  BSS 1
0184  00216 011500 .FADA  DEF FAD
```

```
          #01  9830 MATH ROUTINES


0186  00217 011466 .FSBA DEF FSB      SUBTRACT
0187  00220 016311 .FMPA DEF FMP      MULTPLY
0188  00221 011576 .FDVA DEF FDV      DIVIDE
0189  00222 000000 ENTFA BSS 1
0190  00223 016162 .FSB  DEF FSB
0191  00224 016054 UPARA DEF UPARO    UP ARROW
0192  00225 017021 FLTRA DEF FLTRF
0193  00226 017001 FLTFA DEF FLTFX
0194  00227 017055 FXFLA DEF FXFLT
0195  00230 017121 FLTSA DEF FLTSP
0196  00231 016365 SPLFA DEF SPLFL
0197  00232 017331 SNFLA DEF SNFLT    SYNTAX FORMAT TO FLOATING
0198  00233 000000 BEND   BSS 1        I/O BUFFER END
0199  00233        SBUFA EQU BEND
0200  00234 000000 NMOTA BSS 1        NUMBER OUTPUT ROUTINE LINK
0201  00235 000000 FILLA BSS 1          SUPPLY NEEDED BLANKS ON OUTPUT
0202  00236 000000 SLEND BSS 1        REFRESH BUFFER END
0203  00111        TMPA  EQU B1740
0204  00110        BADRP EQU B1400     POINTER POINTER TO I/O BUFFER
0205*
0206  00123        FOPRS EQU B12K
0207  00237 000000 SALTA BSS 1
0208  00240 000000 MDIMA BSS 1
0209  00241 000000 SYTSA BSS 1
0210  00242 000000 DCOMA BSS 1
0211  00243 017154 NUMCA DEF NUMCK
0212  00244 017266 NUMOA DEF NUMOP
0213  00245 000000 GTSTA BSS 1
0214  00246 000000 VARDA BSS 1
0215  00247 000000 FCALA BSS 1
0216  00250 000000 OVCHA BSS 1
0217  00251 000000 CLPIA BSS 1
0218  00252 000000 ENTFA BSS 1
0219  00253 000000 CHRSA BSS 1
0220  00254 000000 XSCRA BSS 1
0221  00255 000000 XFCKA BSS 1
0222  00256 000000 FDAAA BSS 1
0223  00257 000000 FNDAA BSS 1
0224  00260 000000 FNDIA BSS 1
0225  00261 000000 IOSTA BSS 1
0226  00262 000000 INREA BSS 1
0227  00263 000000 SUMPA BSS 1
0228  00264 000000 CRCKA BSS 1
0229  00265 000000 IOMEA BSS 1
0230  00266 000000 QTCHA BSS 1
0231**********************************************
0232*
0233  00267 000000        BSS 1        CHECKSUM WORD FOR FIRST 512 BLOCK


          #01  9830--ERROR TABLE


0235  00270 000000 E1     BSS 1
0236  00271 000000        BSS 1
0237  00272 000000        BSS 1
0238  00273 000000        BSS 1
0239  00274 000000 E5     BSS 1
0240  00275 000000        BSS 1
0241  00276 000000        BSS 1
0242  00277 000000        BSS 1
0243  00300 000000        BSS 1
0244  00301 000000 E10    BSS 1
0245  00302 017255        DEF NUMER
0246  00303 017172        DEF NUMCP+1
0247  00304 000000        BSS 1
0248  00305 000000        BSS 1
0249  00306 000000 E15    BSS 1
0250  00307 000000        BSS 1
0251  00310 000000        BSS 1
0252  00311 000000        BSS 1
0253  00312 000000        BSS 1
0254  00313 000000 E20    BSS 1
0255  00314 000000        BSS 1
0256  00315 000000        BSS 1
0257  00316 000000        BSS 1
0258  00317 000000        BSS 1
0259  00320 000000 E25    BSS 1
0260  00321 000000        BSS 1
0261  00322 000000        BSS 1
0262  00323 000000        BSS 1
0263  00324 000000        BSS 1
0264  00325 000000 E30    BSS 1
0265  00326 000000        BSS 1
0266  00327 000000        BSS 1
0267  00330 000000        BSS 1
0268  00331 000000        BSS 1
0269  00332 000000 E35    BSS 1
0270  00333 000000        BSS 1
0271  00334 000000        BSS 1
```

```
          #01  9830--ERROR TABLE


0272  00335 000000          BSS 1
0273  00336 000000          BSS 1
0274  00337 000000 E44      BSS 1
0275  00340 000000          BSS 1
0276  00341 000000          BSS 1
0277  00342 000000          BSS 1
0278  00343 000000          BSS 1
0279  00344 000000 E45      BSS 1
0280  00345 000000          BSS 1
0281  00346 000000          BSS 1
0282  00347 000000          BSS 1
0283  00350 000000          BSS 1
0284  00351 000000 E50      BSS 1
0285  00352 013355          DEF ER47      LOG OF A NEGATIVE NUMBER
0286  00353 016207          DEF ER48      SQUARE ROOT OF A NEGATIVE NUMBER
0287  00354 016053          DEF ER49      ZERO TO THE ZERO POWER
0288  00355 016141          DEF ER50      NEGATIVE NUMBER RAISED TO A NON-INTEGER POWER


          #01  9830--BASE PAGE READ-WRITE MEMORY

0290  01400                 ORG 1400B
0291  01400 000000 RADDR    BSS 1     CURRENT CHAR POINTER
0292  01401 000000 RSAVE    BSS 1          SAVE (A) FOR RETN2
0293  01402 000000 OTPTR    BSS 1     REFRESH POINTER
0294  01403 000000 MAW      BSS 1     LAST WORD OF R/W MEM
0295  01404 000000 LWAM     BSS 1     LAST WORD AVAILABLE FOR USER
0296  01405 000000 FWUP     BSS 1     FIRST WORD OF COMMON OR MAINLINE
0297  01406 000000 VALUE    BSS 1     FIRST WORD -1 OF VALUE TABLE
0298  01407 000000 SYMTP    BSS 1     FIRST WORD -1 OF SYMBOL TABLE
0299  01410 000020 LSTAK    BSS 1     FIRST WORD OF OPERATOR STACK
0300  01411 000000 TSTPT    BSS 1     TEMPORARY STACK POINTER
0301  01412 000000 LSTPT    BSS 1     OPERATOR STACK POINTER
0302  01413 000000 HSTPT    BSS 1     OPERAND STACK POINTER
0303  01414 000000 DSTRT    BSS 1     DATA STATEMENT
0304  01415 000000 NXTDT    BSS 1       POINTERS
0305  01416 000000 MODE     BSS 1     PROGRAM=0: KEYBOARD=1
0306  01417 000000 PARMA    BSS 1     PARAMETER ADDRESS
0307  01420 000000 PARMN    BSS 1     PARAMETER NAME
0308  01421 000000 PRADD    BSS 1     LAST WORD+1 OF CURRENT STMT
0309  01422 000000 UKPTR    BSS 1     USER KEY POINTER
0310  01423 000000 PRPTR    BSS 1     LAST WORD OF CURRENT PROGRAM
0311  01424 000000 PRUFF    BSS 1     FIRST WORD OF CURRENT PROGRAM
0312  01425 000000 PRUFO    BSS 1     FIRST WORD OF MAINLINE
0313  01426 000000 PRPTO    BSS 1     LAST WORD+1 OF MAINLINE
0314  01427 000000 .LNUM    BSS 1     CURRENT LINE NUMBER
0315  01430 000000 LNINC    BSS 1     AUTO LINE INCREMENT
0316  01431 000000 CLINE    BSS 1     CURRENT LINE FOR LIST
0317  01432 000000 DRQST    BSS 1     INPUT STMT FLAG
0318  01433 000000 TFLAG    BSS 1     PTAPE FLAG
0319  01434 000000 UNITS    BSS 1     TRIG UNITS
0320  01435 000000 SEC      BSS 1     SECURE PROGRAM FLAG
0321  01436 000000 BFLG1    BSS 2     STOP ADDRESSES
0322  01440 000000 TFLG1    BSS 2     TRACE LIMIT ADDRESSES
0323  01442 000000 STPFL    BSS 1     STOP FLAG
0324  01443 000000 KEYFG    BSS 1     KEYBOARD FLAG
0325  01444 000000 SAVEA    BSS 1     INTERRUPT
0326  01445 000000 SAVER    BSS 1
0327  01446 000000 ADRSS    BSS 1       TEMPORARIES
0328  01447 000000 SLCOD    BSS 1     SELECT CODE
0329  01450 000000 PMODE    BSS 1     PRINT-ALL FLAG
0330  01451 000000 CTYPE    BSS 1     KEYBOARD
0331  01452 000000 CPLAC    BSS 1       FORMAT FLAGS
0332  01453 000000 PRFLG    BSS 1     PRINT/DISP FLAG
0333  01454 000000 RCOUT    BSS 1     FORMAT
0334  01455 000000 FORMT    BSS 1
0335  01456 000000 FORMS    BSS 1
0336  01457 000000 FORMF    BSS 1       STATEMENT FLAGS
0337  01460 000000 FTMP1    BSS 1     CURRENT OPERATOR CODE
0338  01461 000000 FTMP2    BSS 1     CURRENT OPERATOR PRECEDENCE
0339  01462 000000 REFDL    BSS 1
0340  01463 000000 INTFL    BSS 1
0341  01464 000000          BSS 2           UNUSED
0342*
0343*  TEMPORARIES
0344*
0345  01466 000000 LOCL1    BSS 1
0346  01467 000000 LOCL2    BSS 1
0347  01470 000000 L2T1     BSS 1
0348  01471 000000 L2T2     BSS 1
0349  01472 000000 TEMPS    BSS 1
0350  01473 000000 .TEMP    BSS 1
0351  01473        MAOX1    EQU .TEMP
0352  01474 000000 TEMP1    BSS 1
0353  01475 000000 TEMP2    BSS 1
0354  01476 000000 TEMP3    BSS 1
0355  01477 000000 TEMP4    BSS 1
0356*
0357*  BUFFERS
0358*
```

```
          #01  9830--BASE PAGE READ-WRITE MEMORY


0359  01500 000000 IOBUF BSS 41         INPUT AND RECALL BUFFER
0360  01551 000000 RBUFF BSS 41         SINGLE LINE REFRESH BUFFER
0361*
0362*  PRINT ROUTINE FLAGS
0363*
0364  01622 000000 TERM  BSS 1     CRLF FLAG
0365  01623 000000 PEND  BSS 1     BUFFER OVERFLOW FLAG
0366  01624 000020 PPFF  BSS 1     BUFFER POINTER OR TAB FLAG
0367  01625 000000 FIRST BSS 1
0368  01626 000000       BSS 2         UNUSED
0369*
0370*       MATH TEMPORARIES
0371*
0372  01630 000000 TREGE BSS 4
0373  01634 000000 RSLTE BSS 4
0374*
0375*  STATEMENT SYNTAX AND EXECUTION TEMPORARIES
0376*
0377  01640 000000 ST    BSS 32
0378*
0379*  SYSTEM SYNTAX AND EXECUTION TEMPORARIES
0380*
0381  01700 000000       BSS 1          SAVED FOR PROCESSOR
0382  01701 000000 NOISP BSS 1
0383  01702 000000 NPRIN BSS 1
0384  01703 000000 PLACE BSS 1
0385  01704 000000 WIDTH BSS 1
0386  01705 000020 FLOAT BSS 1
0387  01706 000000 SAPTR BSS 1
0388  01707 000000 BLANK BSS 1
0389  01710 000000 MSFLG BSS 1
0390  01711 000000 DFLAG BSS 1
0391  01712 000000 PFLAG BSS 1
0392  01713 000000 UFLAG BSS 1
0393  01714 000000 SFLAG BSS 1
0394  01715 000000 SRADR BSS 1
0395  01716 000000 SSHPT BSS 1
0396  01717 000000 NUM   BSS 1
0397  01717        OPTYP EQU NUM
0398  01720 000000 LASTN BSS 1
0399  01721 000000 .SIGN BSS 1
0400  01721        MAXIN EQU .SIGN
0401  01722 000000 EXP   BSS 1
0402  01722        INTGR EQU EXP
0403  01723 000000 ARYAD BSS 1
0404  01724 000000 GFLAG BSS 1
0405  01725 000000 NXTST BSS 1
0406  01726 000000 MAIN1 BSS 1
0407  01727 000000 MAIN2 BSS 1
0408*
0409*  ARITHMETIC REGISTERS AND RETURN STACK
0410*
0411  01740              ORG 1740B
0412  01740 000000 XC    BSS 4
0413  01744 000000 AR1   BSS 4
0414  01750 000000 YC    BSS 4
0415  01754 000000 AR2   BSS 4
0416  01760 000000 RSTAK BSS 11         RETURN STACK
0417  01773 000000 RES   BSS 4          RESULT REGISTOR
0418  01777 000000 RSPTR BSS 1     RETURN STACK POINTER
0419*


          #01  NON-COMMON BASE PAGE ROM


0421  01744        AR1E  EQU AR1
0422  01754        AR2E  EQU AR2
0423  00136        N5    EQU N5
0424  01237        STAR2 EQU 1237B
0425  01305        ROUND EQU 1305B
0426  01261        OF    EQU 1261B
0427  01274        STMAX EQU 1274B
0428*
0429  00356        RETN3 EQU 356B
0430  00357        RETN2 EQU 357B
0431  00365        RET2  EQU 365B
0432  00373        ICNT  EQU 373B
0433  00374        WBUFF EQU 374B
0434  00400        GPARM EQU 400B
0435  00405        XFAR1 EQU 405B
0436  00412        RSULT EQU 412B
0437  00416        STORE EQU 416B
0438  00425        STOR1 EQU 425B
0439  00444        XFAR2 EQU 444B
0440  00450        STROP EQU 450B
0441  00456        SRPUD EQU 456B
0442  00461        SYMC2 EQU 461B
0443  00463        SYMC1 EQU 463B
0444  00464        SYMCK EQU 464B
```

```
          #01  NON-COMMON BASE PAGE ROM


0445  00504         COMCK EQU 504B
0446  00512         COMCF EQU 514B
0447  00513         LPCK  EQU 513B
0448  00521         LPCKF EQU 521B
0449  00524         RPCK  EQU 524B
0450  00534         RPCKF EQU 534B
0451  00535         FSCF2 EQU 535B
0452  00537         STSPH EQU 537B
0453  00546         STFXP EQU 546B
0454  00553         RHSTP EQU 553B
0455  00561         MERR  EQU 561B
0456  00561         FSCF4 EQU 561B
0457  00563         SLWST EQU 563B
0458  00567         UNSTK EQU 567B
0459  00574         NEXTL EQU 574B
0460  00606         NEXT1 EQU 606B
0461  00613         FPAGE EQU 613B
0462  00626         SAVRP EQU 626B
0463  00633         RESR1 EQU 633B
0464  00640         RESRP EQU 640B
0465  00645         MINIT EQU 645B
0466  00651         INITS EQU 651B
0467  00664         LNUM  EQU 664B
0468  00666         PGINT EQU 666B
0469  00676         INTCK EQU 676B
0470  00726         SYE25 EQU 726B
0471  00730         GETAD EQU 730B
0472  00742         END   EQU 742B
0473  00750         DIGCK EQU 750B
0474  00752         LETCK EQU 752B
0475  00753         LIMIT EQU 753B
0476  00771         GTOPP EQU 771B
0477  01007         NEXTA EQU 1007B
0478  01015         ORLNK EQU 1015B
0479  01016         ONEXT EQU 1016B
0480  01017         OUTCR EQU 1017B
0481  01041         GETCH EQU 1041B
0482  01053         GTMP  EQU 1053B
0483  01055         GNEXT EQU 1055B
0484  01062         GERST EQU 1062B
0485  01066         GETSK EQU 1066B
0486  01075         ENDPS EQU 1075B
0487  01121         MVTOH EQU 1121B
0488  01131         PGIN0 EQU 1131B
0489  01140         LTR   EQU 1140B
0490  01147         ENDS  EQU 1147B
0491  01151         EOST  EQU 1151B
0492  01152         NOEOF EQU 1152B
0493  01153         EOLCK EQU 1153B
0494  01156         CONST EQU 1156B
0495  01165         SYE12 EQU 1165B
0496  01166         GSIGN EQU 1166B
0497  01201         IMPY  EQU 1201B
0498  01217         FETCH EQU 1217B
0499  01225         LOFF  EQU 1225B
0500  01254         ZONK  EQU 1254B
0501  01256         CLAR2 EQU 1256B
0502  01314         OOGIT EQU 1314B
0503  01317         SGNS  EQU 1317B
0504  01330         .NPWR EQU 1330B
0505  01335         NPWRT EQU 1335B
0506  01337         NOWRT EQU 1337B
0507  01342         CRLF  EQU 1342B
0508  01344         CRLF1 EQU 1344B
0509  01351         TTY   EQU 1351B
0510  01362         GTCON EQU 1362B
0511  01371         FIX   EQU 1371B

          #01  CONSTANTS AND EQUATES FOR BOTH


0513                    LST
0514*       MIKE WINGERT
0515*       AND GENE BURMEISTER 8-24-71
0516*
0517*
0518**********************************
0519*
0520* BASE PAGE CONSTANTS
0521*
0522  00012         P1    EQU .1
0523  00017         P2    EQU .2
0524  00021         P4    EQU .4
0525  00022         P5    EQU .5
0526  00025         P8    EQU .8
0527  00030         P12   EQU .12
0528  00031         P13   EQU CRTN
0529  00032         P15   EQU .15
0530  00013         P256  EQU B400
```

```
          #01  CONSTANTS AND EQUATES FOR BOTH


#531  00132          N1     EQU M1
#532  00141          N10    EQU M10
#533  00157          N256   EQU M256
#534  00057          ATEMP  EQU .1000      POINTER TO YC REGISTOR
#535  00004          D1000  EQU B10K
#536*
#537**********************************
#538*
#539* SHARED READ/WRITE
#540*
#541  01701          TEMPA  EQU 17018
#542  01466          TEMP5  EQU LOCL1
#543*
#544*
#545**********************************
#546*
#547* R/W MEMORY AREA
#548*
#549  01702          ADR1   EQU 17028
#550  01703          ADR2   EQU 17038
#551  01703          ADR3   EQU ADR2
#552  01704          TEMP   EQU 17048      TEMPORARY
#553  01705          TABL0  EQU TEMP+1     CORDIC CONSTANT
#554  01706          GFG    EQU TEMP+2     G FLAG
#555  01707          SHIFT  EQU TEMP+3     SHIFT FACTOR
#556  01710          COUNT  EQU TEMP+4     BCD DATA STORAGE
#557  01711          RCPFG  EQU TEMP+5     RECIPROCAL FLAG
#558  01712          FNCFG  EQU TEMP+6     FUNCTION FLAG
#559  01713          RY1    EQU TEMP+7     REGISTER ADDRESS STORAGE
#560  01714          SAVE   EQU TEMP+8     TEMPORARY
#561  01715          RY2    EQU TEMP+9     REGISTER ADDRESS STORAGE
#562  01716          SIGN   EQU TEMP+10    EXPONENT WORD STORAGE
#563  01717          DIGIT  EQU TEMP+11    DIGIT STACK POINTER
#564  01720          WRDN   EQU TEMP+12    START OF DIGIT STACK
#565  01734          ADRYE  EQU 17348
#566*
#567*
#568**********************************
#569*
#570***      LABELS
#571*
#572  00023          P6     EQU .6         +6
#573  00024          P7     EQU .7         +7
#574  00061          P10    EQU .10            +10
#575  00133          N2     EQU M2         -2
#576  00135          N4     EQU M4         -4
#577  00143          N12    EQU M12        -12
#578  00062          D8000  EQU FLGAT          BCD 8000
#579*
#580* LINKAGE TO ARITHMETIC
#581*
#582  00216          ADD    EQU .FADA       FLOATING ADD ROUTINE
#583  00220          MLT    EQU .FMPA       FLOATING MULTIPLY ROUTINE
#584  00221          DVD    EQU .FDVA       FLOATING DIVIDE ROUTINE
#585*
#586*
#587**********************************
#588*


          #01  LOG CONSTANTS AT 11600B

#591  03700               ORG 3700B
#592*
#593*
#594**********************************
#595*
#596* ROM AREA
#597*
#598  03700 000000 C1     OCT 0000000     LN(2)
#599  03701 003223         OCT 003223      0693
#600  03702 012161         OCT 012161      1471
#601  03703 100126         OCT 100126      8056
#602*
#603  03704 000000         DEC 0000        LN(1.1)
#604  03705 004523         OCT 004523      0953
#605  03706 010027         OCT 010027      1017
#606  03707 114004         OCT 114004      9804
#607*
#608  03710 000000         DEC 0000        LN(1.01)
#609  03711 004625         OCT 004625      0995
#610  03712 001460         OCT 001460      0330
#611  03713 102462         OCT 102462      8532
#612*
#613  03714 000000         DEC 0000        LN(1.001)
#614  03715 004631         OCT 004631      0999
#615  03716 050003         OCT 050003      5003
#616  03717 031410         OCT 031410      3308
#617*
#618  03720 000000         DEC 0000        LN(1.0001)
```

```
          #21  LOG CONSTANTS AT 11600B

0619  03721 004631        OCT 004631     0999
0620  03722 112430        OCT 112400     9500
0621  03723 001463        OCT 001463     0333
0622*
0623  03724 000000        OCT 000000     LN(1.00001)
0624  03725 004631        OCT 004631     0999
0625  03726 114520        OCT 114520     9950
0626  03727 000003        OCT 000003     0003
0627*
0628  03730 030000 C2     OCT 00000000     LN(10)
0629  03731 021402        OCT 021402     2342
0630  03732 054120        OCT 054120     5850
0631  03733 111400        OCT 111400     9300
0632*
0633***
0634*
0635  03734 000000        DEC 0000       ARCTAN(1)
0636  03735 003605        OCT 003605     0785
0637  03736 034601        OCT 034601     3981
0638  03737 061500        OCT 061500     6340
0639*
0640  03740 000000        DEC 0000       ARCTAN(.1)
0641  03741 004626        OCT 004626     0996
0642  03742 064145        OCT 064145     6865
0643  03743 022221        OCT 022221     2491
0644*
0645  03744 000000        DEC 0000       ARC TAN (.01)
0646  03745 004631        OCT 004631     09999
0647  03746 113146        OCT 113146     9666
0648  03747 064147        OCT 064147     6867
0649*
0650  03750 000000        DEC 0000       ARCTAN(.001)
0651  03751 004631        OCT 004631     0999
0652  03752 114626        OCT 114626     9996
0653  03753 063147        OCT 063147     6667
0654*
0655  03754 000000        OCT 000000     ARCTAN(.0001)
0656  03755 004631        OCT 004631     0999
0657  03756 114631        OCT 114631     9999
0658  03757 113147        OCT 113147     9667
0659*
0660  03760 000000        OCT 000000     ARCTAN(.00001)
0661  03761 004631        OCT 004631     0999
0662  03762 114631        OCT 114631     9999
0663  03763 114627        OCT 114627     9997
0664*
0665  03764 000000 C3     OCT 00000000     PI
0666  03765 030501        OCT 030501     3141
0667  03766 054446        OCT 054446     5926
0668  03767 051540        OCT 051540     5360
0669*
0670  03770 000002 C4     OCT 00000000     2PI
0671  03771 061203        OCT 061203     6283
0672  03772 014123        OCT 014123     1853
0673  03773 007440        OCT 007440     0720
0674*
0675  03774 000400 C8     OCT 000400     200/PI
0676  03775 061546        OCT 061546     6366
0677  03776 014567        OCT 014567     1977
0678  03777 021545        OCT 021545     2365


          #01  FUNCTIONS FOR 9830

                                                    50
06P1  12533               UNG 12533B
0682*
0683  12533 003774 GPERR  DEF C8         ADDRESS OF 200/PI
0684  00035       P22     EQU .22
0685  00060       P100    EQU .100
0686  00144       N13     EQU M13
0687  00156       N100    EQU M100
0688  12534 003730 LN10   DEF C2         ADDRESS OF LN10
0689  12535 003764 PI     DEF C3         ADDRESS OF PI
0690  12536 003770 TWOPI  DEF C4         ADDRESS OF 2PI
0691  12537 001720 AWRDN  DEF WRDN       TOP OF THE DIGIT STACK
0692  00111       ADRX    EQU B1740      ADDRESS OF REGISTER X
0693  12540 001734 ADRY   OCT 1734       ADDRESS OF REGISTER Y
0694  12540       ADR0    EQU ADRY       ADDRESS OF REGISTOR THETE
0695  12541 001630 TREG   DEF TREGE      ADDRESS OF REGISTOR T
0696  12542 001634 RSLT   DEF RSLTE      ADDRESS OF THE ORIGNAL ARGUMENT
0697  00110       EP3     EQU B1400       EXPONENT POSITIVE 3
0698  00013       D01W    EQU B400        BCD 0100
0699  00033       D0010   EQU B20         BCD 0010
0700  00121       GRAF    EQU B4000       EXPONENT POSITIVE 8
0701  12543 176400 EN3    OCT 176400     EXPONENT NEGATIVE 3
0702  12544 172000 EN12   OCT 172000     EXPONENT NEGATIVE 12
0703  12545 003674 THETL  DEF C1-4       ADDRESS OF LOG CONSTANT TARLE
0704  12546 003724 THETT  DEF C2-4       ADDRESS OF TRIG. CONSTANT TALE
0705  00012       RAD     EQU P1         RADIAN CODE
```

```
          #01  FUNCTIONS FOR 9830

 0706  00017         GRA   EQU P2         GRAD CODE
 0707*
 0708*  ADDITITONAL CONSTANTS FOR MATH
 0709*
 0710  00117         AD5   EQU B2400
 0711  00145         N14   EQU M14
 0713*
 0714*
 0715**********************************
 0716*
 0717***      PROGRAM SUBROUTINES
 0718*
 0719*
 0720**********************************
 0721*
 0722* INITIALIZE ROUTINE
 0723*
 0724  12547 035712 INIT  STB FNCFG      STORE THE FUNCTION FLAG
 0725*
 0726  12550 031703       STA ADR2       STORE THE ADDRESS OF THE ARGUMENT
 0727  12551 121703       LDA ADR2,I     SAVE THE ORIGINAL EXPONENT
 0728  12552 031716       STA SIGN         WORD IN SIGN
 0729*
 0730  12553 021703       LDA ADR2       SAVE THE ANSWER OF THE RESULT
 0731  12554 026542       LDB RSLT
 0732  12555 035703       STB AUR2
 0733  12556 170004       XFR              SO IT IS SURELY IN A R/W TEMPORARY
 0734*
 0735  12557 022542 RTT   LDA RSLT       LOAD THE ARGUMENT
 0736  12560 024214       LDB AR2A       INTO THE AR2 WORKING REGISTER
 0737  12561 170004       XFR
 0738*
 0739  12562 066635       JMP ZERO       JUMP TO DETERMINE IF IT IS ZERO
 0740*
 0741*
 0742**********************************
 0743*
 0744* NORMALIZE TWO ARGUMENTS
 0745*
 0746  12563 035702 DIV   STB ADR1       SAVE THE ADDRESS IN B
 0747  12564 062575       JSM NORM       NORMALIZE THE NO. SPECIFIED IN A
 0748  12565 026541       LDB TREG       TRANSFER IT INTO
 0749  12566 170004       XFR              A TEMPORARY
 0750  12567 021702       LDA ADR1       LOAD THE OTHER ADDRESS
 0751  12570 062575       JSM NORM       NORMALIZE IT
 0752  12571 026542 ITR   LDB RSLT       TRANSFER IT INTO THE
 0753  12572 020214       LDA AR2A       RESULT REGISTOR AND RET
 0754  12573 170004       XFR
 0755  12574 170402       RET
 0756*
 0757*
 0758**********************************
 0759*
 0760* NORMALIZE A SINGLE ARGUMENT
 0761*
 0762  12575 062560 NORM  JSM RTT+1      TRANSFER THE ARGUMENT SPECIFIED IN A INTO AR2
 0763  12576 171450       NRM            NORMALIZE IT
 0764  12577 074404       SBL 8          ALLIGN THE SHIFT COUNT IN B
 0765  12600 074076       TCB            COMPLEMENT IT
 0766  12601 025754       ADB AR2E       ADD THE EXPONENT WORD TO IT
 0767  12602 035754       STB AR2E       RESTORE THE EXPONENT WORD
 0768  12603 020214       LDA AR2A     LOAD THE ADD. OF THE ARGUMENT
 0769  12604 170402       RET              AND RETURN
 0770*
 0771*
 0772**********************************
 0773*
 0774* REDUCE ONE NUMBER TO A FRACTIONAL PART OF ANOTHER
 0775*
 0776  12605 062727 FRAC  JSM FDV1       DIVIDE THE ARGUMENT BY THE SPECIFIED CONSTANT
 0777*
 0778  12606 025754       LDB AR2E       LOAD THE EXPONENT WORD OF THE RESULT
 0779  12607 070742       SAR 16         CLEAR THE COUNT WORD
 0780  12610 031710       STA COUNT        FOR STORING THE INTEGER PART
 0781  12611 074252       SBM *+5        SKIP IF THE EXPONENT IS NEGATIVE
 0782*
 0783  12612 004157       ADB N256       SUBTRACT 1 FROM THE EXPONENT
 0784  12613 171400       MLS            LEFT SHIFT THE ARGUMENT
 0785  12614 001710       ADA COUNT      STORE THE LEFT
 0786  12615 070604       SAL 4            SHIFTED DIGIT
 0787  12616 031710       STA COUNT        IN COUNT
 0788  12617 075553       SBP *-5        SKIP IF THE EXPONENT IS STILL POSITIVE
 0789  12620 035754       STB AR2E       STORE THE NEW EXPONENT WORD
 0790*
 0791  12621 062576       JSM NORM+1     JUMP TO NORMALIZE THE ARGUMENT
 0792*
 0793  12622 021702       LDA AUR1       MULTIPLY THIS FRACTIONAL
 0794  12623 066750       JMP .FMP-1      PART OF THE ORIGINAL SPECIFIED CONSTANT
 0795*
 0796*
```

```
          #01  FUNCTIONS FOR 9830

0797**********************************
0798*
0799* UNNORMALIZE AN ARGUMENT
0800*
0801  12624 025754 UNRM  LDB AR2E       LOAD THE EXPONENT WORD
0802  12625 074413       SHP ZERO       SKIP IF THE EXPONENT IS POSITIVE
0803  12626 074340       ABR B          ALLIGN AND COMPLEMENT
0804  12627 074076       TCB              THE EXPONENT
0805  12630 074117       LDA B          DUPLICATE IT AND
0806  12631 000143       ADA N12          CHECK TO SEE IF IT
0807  12632 070112       SAM *+2          IS GREATER THAN 12
0808  12633 024031       LDB P13        SET B = 13 IF IT IS
0809*
0810  12634 061305       JSM ROUND SHIFT AND COUNT  THE ARGUMENT
0812*
0813*
0814**********************************
0815*
0816* DETERMINE WHETHER AN ARGUMENT IS ZERO
0817*
0818  12635 070075 ZERO  SEC *+1,C      CLEAR THE E REGISTER
0819  12636 021755       LDA AR2E+1     ADD THE 12 DIGITS
0820  12637 001756       ADA AR2E+2       OF AR2 TOGETHER
0821  12640 001757       ADA AR2E+3       IN A
0822  12641 072154       SES *+3,S      SKIP IF THE REGISTER OVERFLOWED AND SET F
0823  12642 072110       CLA *+2        CLEAR E
0824  12643 070075       SEC *+1,C      SKIP IF A IS NOT ZERO
0825  12644 025754       LDB AR2E       LOAD THE EXPONENT WORD INTO B
0826  12645 170402       RET              AND RETURN
0827*
0828*
0829**********************************
0830*
0831* RECIPROCATE AN ARGUMENT
0832*
0833  12646 062571 RECIP JSM ITR
0834  12647 020020       LDA ONE        SET THE DIVIDEND = 1
0835  12650 025703       LDB ADR3
0836  12651 066731       JMP .FDV+1
0893*
0894*
0895**********************************
0896*
0897* INITIALIZE ROUTINE FOR THE TRIG FUNCTIONS
0898*
0899*
0900  12714 074742 STN   SBR 16         CLEAR THE G FLAG
0901  12715 035706       STB GFG          AND THE TEMP
0902  12716 035704       STB TEMP         LOCATION
0903  12717 062547       JSM INIT       FETCH THE ARGUMENT AND INITIALIZE OTHER FLAGS
0904*
0905  12720 025434       LDB UNITS      LOAD THE UNITS WORD
0906  12721 074250       SZB DTOR       IF IT IS SET TO DEGREES JUMP TO CONVERT TO RAD.
0907  12722 014012       CPB RAD        IF IT SAYS NOT RADIANS CONVERT
0908  12723 170402       RET              IT FROM GRADS TO RADIANS
0909*
0910  12724 026533 GTOR  LDB GPERR      LOAD THE ADDRESS OF 200/PI
0911  12725 066727       JMP DTOR+1       DIVIDE IT INTO THE ARGUMENT
0912*
0913*
0914**********************************
0915*
0916* LINK ROUTINE TO FLOATING DIVIDE
0917*
0918  12726 024027 DTOR  LDB UPERR      LOAD THE ADDRESS OF 180/PI
0919  12727 021703 FDV1  LDA ADR3
0920  12730 031703 .FDV  STA ADR2
0921  12731 160221       JSM DVD,I      DO THE DIVIDE
0922  12732 066737       JMP SANS       STORE THE RESULT
0923*
0924*
0925**********************************
0926*
0927* LINK ROUTINE TO FLOATING ADD
0928*
0929  12733 022542 FAD1  LDA RSLT
0930  12734 035702 .FAD  STB ADR1
0931  12735 031703       STA ADR2
0932  12736 160216       JSM ADD,I      DO THE ADD
0933  12737 020214 SANS  LDA AR2A
0934  12740 025703       LDB ADR3
0935  12741 170004       XFR
0936  12742 170402       RET
0938*
0939*
0940**********************************
0941*
0942* LINK ROUTINE TO FLOATING MULTIPLY
0943*
```

```
          #01  FUNCTIONS FOR 9830

0944  12743 020027 RTOD  LDA DPERR      LOAD THE ADDRESS OF 180/PI
0945  12744 026541       LDB TREG       TRANSFER THIS QUANTITY INTO A
0946  12745 170004       XFR            TEMPORARY IN ORDER TO
0947  12746 045633       ISZ TREGE+3    INCREMENT DIGIT 12 BY 1
0948  12747 022541       LDA TREG       SET (A) TO THE POINTER TO TREG
0949  12750 024214       LDB AR2A       SET B = ADD. OF AR2
0950*
0951*
0952  12751 160220 .FMP  JSM MLT,I
0953  12752 066737       JMP SANS       STORE THE ANSWER
0954*
0955*
0956**********************************
0957*
0958* REGISTER TRANSFER SUBROUTINES
0959*
0960  12753 020020 OTO   LDA ONE        SET A = THE ADDRESS OF FLOATING ONE
0961  12754 024213 XAR1  LDB AR1A       TRANSFER IT INTO AR1 AND RETURN
0962  12755 170004       XFR
0963  12756 173402       RET
0964*
0965  12757 020020 OTT   LDA ONE        SET A = THE ADDRESS OF FLOATING ONE
0966  12760 066560       JMP RTT+1      TRANSFER FLOATING ONE INTO AR2
0968**********************************
0969*
0970***      MAIN PROGRAMS
0971*
0972**********************
0973*
0974*      FIND THE SINE OF AN ARGUMENT
0975*
0976  12761 062714 SIN   JSM STN        FETCH THE ARGUMENT AND INITIALIZE FLAGS
0977  12762 045704       ISZ TEMP       SET TEMP = 1
0978  12763 063054       JSM TAN+1      CALCULATE THE TANGENT OF THE ARGUMENT
0979*
0980  12764 072150       RZA *+3        SKIP IF OVERFLOW DID NOT OCCUR
0981  12765 062757       JSM OTT        IF TAN WAS INFINITY
0982  12766 066773       JMP .CS        SET + OR - ONE INTO AR2
0983  12767 062737       JSM SANS
0984  12770 070742       SAR 16         LET THE FUNCTION FLAG BE 0
0985  12771 062652       JSM SCRT       JUMP TO CALCULATE SQR(1+XX)
0986*
0987  12772 062730       JSM .FDV       DIVIDE THE ORIGINAL NO. BY THIS QUANTITY
0988*
0989  12773 021754 .CS   LDA AR2E       LOAD THE EXPONENT WORD WORD OF THE RESULT
0990  12774 072051       SLA *+1,S      SET THE MANTISSA SIGN BIT
0991  12775 025711       LDB RCPFG      LOAD THE RECIPROCAL FLAG INTO B
0992  12776 004133       ADB N2         SUBTRACT 2 FROM IT
0993  12777 074702       SHR 15         LET BIT 0 = 1 IF THE RESULT WAS NEGATIVE
0994  13000 005716       ADB SIGN       ADD THE ORIGINAL MANTISSA SIGN BIT TO THIS
0995  13001 074111       SLB **2        IF THE RESULTANT LOW BIT IS
0996  13002 070071       SLA **1,C        ZERO CLEAR THE MANTISSA SIGN BIT OF THE ANSWER
0997  13003 074742       SBR 1b         SET B = 0 AND JUMP
0998  13004 070112       SAM **2
0999  13005 035757       STB AR2+3
1000  13006 067202       JMP OUT          TO THE STORE ROUTINE
1001*
1002**********************************
1003*
1004* FIND THE COSINE OF AN ARGUMENT
1005*
1006  13007 062714 COS   JSM STN        FETCH THE ARGUMENT IN RADIANS
1007  13010 021754       LDA AR2E       CHECK THE EXPONENT
1008  13011 070340       AAR 8
1009  13012 000142       ADA M11        IF LESS THAN 11
1010  13013 070112       SAM **2        SKIP
1011  13014 066757       JMP OTT        OTHERWISE ASSUME A FULL CIRCLE
1012  13015 024023       LDB PI/2
1013  13016 062733       JSM FAD1       ADD PI/2 TO THE ARGUMENT
1014  13017 066762       JMP SIN+1      JUMP TO SINE ROUTINE
1015*
1016**********************************
1017  13020 000000       oSS 1       CHECKSUM WORD
1018*
1019****************************************
1020*
1022*
1023*
1024**********************************
1025*
1026* FIND THE ARCCOSINE OF AN ARGUMENT
1027*
1028  13021 024132 ACS   LDB N1       LET THE FUNCTION FLAG BE -1
1029  13022 063032       JSM ASNS+1     JUMP TO THE ARCSINE ROUTINE
1030  13023 021706       LDA GFG        LOAD THE ORIGINAL EXPONENT WORD
1031  13024 070211       SLA ACSC       SKIP IF THE MANTISSA SIGN BIT IS 0
1032  13025 062571       JSM ITR     OTHERWISE FETCH ANSWER
1033  13026 026535       LDB PI
1034  13027 062733       JSM FAD1
1035  13030 067256 ACSC  JMP EATN       JUMP TO CONVERT TO THE PROPER UNITS
```

```
          #01  FUNCTIONS FOR 9830

1036*
1037*
1038**********************************
1039*
1040* SUBROUTINE FOR THE ARC ROUTINES
1041*
1042  13031 024012 ASNS  LDB P1       LET THE FUNCTION FLAG BE 1
1043  13032 035710       STB COUNT    STORE THE FUNCTION FLAG
1044  13033 070537       LDB A,I      SAVE THE EXPONENT WORD
1045  13034 035706       STB GFG        FOR FUTURE REFERENCES
1046  13035 062550       JSM INIT+1   FETCH THE ARGUMENT AND INITIALIZE THE FLAGS
1047  13036 062653       JSM SQRT+1   CALCULATE SQR (1+OR-XX)
1048  13037 074070       SIB *+1      LOAD THE FIRST 4 DIGITS
1049  13040 074537       LDB B,I        OF THE NUMBER SPECIFIED BY THE ADDRESS IN B
1050  13041 076210       HZB *+4      SKIP IF THEY ARE NOT ZERO
1051  13042 061256       JSM CLAR2    * CLEAR AR2
1052  13043 021716       LDA SIGN     * LOAD A WITH THE ORIGNAL SIGN
1053  13044 067244       JMP ROOT+3   JUMP TO ATNS ROUTINE TO STORE +- PI/2
1054*
1055  13045 062674       JSM SQRT1    JUMP TO FETCH THE DIVIDE ADDRESSES OF SQRT
1056  13046 062731       JSM .FDV+1   DIVIDE  THE RESULTS OF SQRT
1057  13047 021703       LDA ADR3     SET THE ADDRESS OF THE RESULT
1058  13050 067206       JMP ATNS     JUMP TO THE ARCTANGENT ROUTINE
1059*
1060*
1061**********************************
1062*
1063* FIND THE ARCSINE OF AN ARGUMENT
1064*
1065  13051 063031 ASN   JSM ASNS     CALCULATE THE ANSWER IN RADIANS
1066  13052 067256       JMP EATN     CONVERT IT TO THE PROPER UNITS
1068*
1069*
1070**********************************
1071*
1072* FIND THE TANGENT OF AN ARGUMENT
1073*
1074  13053 062714 TAN   JSM SIN      FETCH THE ARGUMENT IN RADIANS AND INITIALIZE FLAGS
1075  13054 024012       LDB P1       SET THE RECIPROCAL
1076  13055 035711       STB RCPFG      FLAG TO 1
1077  13056 025754       LDB AR2E     LOAD THE EXPONENT WORD
1078  13057 035716       STB SIGN
1079  13060 074153       SBP *+3        SKIP IF THE EXPONENT IS POSITIVE
1080  13061 045706 GET3  ISZ GFG    OTHERWISE SET GFG=1 AND
1081  13062 067116       JMP GET        DO NOT SCALE
1082  13063 026536       LDB TWOPI    SCALE THE ARGUMENT TO
1083  13064 062605       JSM FRAC       WITHIN ONE CIRCLE
1084  13065 062635       JSM ZERO     DETERMINE IF THE RESULT IS ZERO
1085  13066 070635       SEC FAL1,C   STORE ZERO IF IT IS
1086  13067 075512       SBM GET3   SKIP IF THE EXPONENT IS NOW NONZERO
1087*
1088  13070 020023       LDA PI/2     LOAD PI/2
1089  13071 062754       JSM XAR1     INTO AR1
1090  13072 170400       CMY          COMPLEMENT IT
1091  13073 074742       SBR 16       SUBTRACT AR 1 FROM AR2
1092  13074 170420       FDV            UNTIL AR2 GOES NEGATIVE
1093  13075 035711       STB RCPFG    STORE THE NO. OF SHIFTS BEFORE OVERFLOW
1094  13076 062635       JSM ZERO     DETERMINE IF AR2 IS NOW ZERO
1095  13077 070434       SES *+B,C    SKIP IF IT IS NOT ZERO
1096*
1097  13100 025711       LDB RCPFG    SKIP IF THE NO. OF SUBTRACTS
1098  13101 074111       SLR *+2        ABOVE WAS EVEN
1099  13102 065256 FAL1  JMP CLAR2    STORE ZERO IF ODD - EXIT HERE FOR SIN 0, COS 90,TAN
1100*    EXIT BELOW FOR SIN 90, COS 0, AND TAN 90
1101  13103 025754       LDB AR2E     LOAD THE EXPONENT WORD
1102  13104 055704       DSZ TEMP     STORE MAX NO. IF THE SIN
1103  13105 065274       JMP STMAX    ROUTINE WAS NOT CALLED
1104  13106 170402       RET          OTHERWISE RETURN WITH A=0
1105*
1106  13107 062576       JSM NORM+1   NORMALIZE AR2
1107*
1108  13110 074312       SBM GET      SKIP IF THE NEW EXPONENT IS NEGATIVE
1109  13111 020157       LDA N256     MAKE THE EXPONENT
1110  13112 041716       IOR SIGN       MINUS ONE AND
1111  13113 031754       STA AR2E       ORIGINAL MANTISSA SIGN BIT
1112  13114 031716       STA SIGN     STORE THIS ALSO OVER THE ORIGINAL EXPONENT
1113*
1114  13115 067121       JMP GET+3    JUMP TO AVOID THE OFFSET BELOW
1115  13116 035716 GET   STB SIGN     STORE THE NEW EXPONENT WORD
1116  13117 024012       LDB P1       SET B=1 AND
1117  13120 061305       JSM ROUND    JUMP TO RIGHT SHIFT AR2
1118*
1119  13121 063523       JSM ATD      INITIALIZE THE DIGIT STACK POINTER
1120*
1121  13122 021754       LDA AR2E     LOAD THE EXPONENT WORD
1122  13123 070240       AAR 6        MULTIPLY THE EXPONENT WORD BY 4
1123  13124 170376       TCA          COMPLEMENT IT
1124  13125 002546       ADA THETT    ADD IT TO THE TRIG TABLE LINK IN A
1125  13126 025754       LDB AR2E     LOAD THE EXPONENT WORD
1126  13127 074343       ABR 8        ALLIGN THE EXPONENT WORD
```

```
          #01  FUNCTIONS FOR 9830

1127  13130 #004024        ADB P7          AND ADD 7 TO IT
1128  13131 #74453         SBP .T.        SKIP IF THE RESULT IS POSITIVE
1129*
1130  13132 171450         NRM            NORMALIZE AR2
1131  13133 #62571         JSM ITR        TRANSFER AR2 INTO THE RESULT LOCATION
1132  13134 #25711         LDB RCPFG      LOAD THE NO. OF SUBTRACTS FROM ABOVE
1133  13135 #74111         SLB *+2        SKIP IF IT IS EVEN
1134  13136 #67200         JMP OUT-2      IF ODD JUMP TO THE STORE ROUTINE
1135  13137 #62646         JSM RECIP      RECIPROCATE THE RESULT
1136  13140 #25711         LDB RCPFG      LOAD THE NO. OF SUBTRACTS
1137  13141 #67200         JMP OUT-2      JUMP TO THE STORE ROUTINE
1138  13142 #75410 .T.     SZB *-8        SKIP BACK IF THE EXPONENT WAS -7
1139*
1140  13143 #63431         JSM IN         EXECUTE PHASE I
1141*
1142  13144 #21754         LDA AR2E       LOAD THE EXPONENT WORD
1143  13145 #00013         ADA P256       INCREMENT THE EXPONENT BY 1
1144  13146 #70340         AAR B          SHIFT TO THE LOW 8 BITS IN A
1145  13147 #74742         SBR 16         SET THE EXPONENT
1146  13150 #35754         STB AR2E         WORD OF AR2 TO.0
1147  13151 #70137         LDB A          SET B EQUAL TO 6
1148  13152 #70076         TCA              MINUS THE
1149  13153 #004023        ADB P6           6 REGISTER
1150  13154 #63457         JSM IN2        EXECUTE PHASE II
1151*
1152  13155 #21711         LDA RCPFG      LOAD THE RECIPROCAL FLAG
1153  13156 #70211         SLA *+4        SKIP IF IT IS EVEN
1154  13157 #20213         LDA AR1A       SET A TO THE ADD. OF CORDIC X
1155  13160 #25713         LDB KY1        SET B TO THE ADDRESS OF CORDIC Y
1156  13161 #67164         JMP *+3        JUMP TO NORMALIZE THESE
1157  13162 #21713         LDA KY1        SET A TO THE ADDRESS OF CORDIC Y
1158  13163 #24213         LDB AR1A       SET B TO THE ADD. OF CORDIC X
1159  13164 #62563         JSM DIV        JUMP TO NORMALIZE THESE QUANTITIES
1160  13165 #22542         LDA RSLT       LOAD THE ADDRESS OF THE DIVISOR
1161  13166 #26541         LDB IREG       LOAD THE ADDRESS OF THE DIVIDEND
1162  13167 #62730         JSM .FDV        JUMP TO THE DIVIDE ROUTINE
1163*
1164  13170 #25716         LDB SIGN       LOAD THE EXPONENT ORIGINAL EXPONENT WORD INTO A
1165  13171 #05711         ADB RCPFG      ADD THE RECIPROCAL FLAG TO THE MANTISSA SIGN BIT
1166  13172 #04013         ADB P256       INCREMENT THE EXPONENT BY 1
1167  13173 #21711         LDA RCPFG      LOAD THE RECIPROCAL FLAG INTO B
1168  13174 #70111         SLA *+2        SKIP IF IT IS EVEN
1169  13175 #67200         JMP OUT-2      JUMP TO THE STORE ROUTINE
1170  13176 #74076         TCB            COMPLEMENT B AND
1171  13177 #04033         ADB D0010        PROPOGATE BIT 7 INTO  THE EXPONENT
1172  13200 #05706         ADB GFG        ADD THE G FLAG TO THE MANTISSA SIGN INB
1173  13201 #21754         LDA AR2E       STORE THE EXPONENT WORD OF AR2
1174  13202 #31744 OUT     STA AR1E         IN AR1 FOR THE STORE ROUTINE
1175*
1176  13203 #22542         LDA RSLT       RESET THE RETURN
1177  13204 #31703         STA ADR3         LOCATION ADDRESS
1178*
1179  13205 #65237 ZOR     JMP STAR2      JUMP TO THE STORE ROUTINE
1181*
1182*
1183**********************************
1184*
1185* ARCTANGENT SUBROUTINE
1186*
1187  13206 #24062 ATNS    LDB D8000      LOAD THE ARCTANGENT FLAG - BCD 8000
1188  13207 #62547         JSM INIT       JUMP TO FETCH THE ARGUMENT AND INITIALIZE THE FLAGS
1189*
1190  13210 #71655         SEC ZOR        RETURN IF IT IS ZERO
1191  13211 #74312         SAM OK+1       SKIP IF THE EXPONENT IS NEGATIVE
1192  13212 #06544         ADB EN12       IF THE EXPONENT IS LESS THAN 12
1193  13213 #74152         SBM OK           JUMP TO CONTINUE
1194  13214 #61256 FAP     JSM CLAR2     OTHERWISE SET AR2 = 0 AND JUMP TO
1195  13215 #67241         JMP POOT       STORE AN ANSWER OF PI/2
1196*
1197  13216 #62646 OK      JSM RECIP      RECIPROCATE THE ARGUMENT
1198*
1199  13217 #62704         JSM CLER       CLEAR THE DIGIT STACK
1200  13220 #21754         LDA AR2E       SET THE A REGISTER EQUAL
1201  13221 #70300         AAR 7            TO 22 - 2* THE EXPONENT
1202  13222 #70076         TCA              TO BE USED AS A SHIFT
1203  13223 #00035         ADA P22          FACTOR IN PHASE II
1204  13224 #24030         LDB P12        SET B = 12 TO BE USED AS A LOOP COUNT
1205*
1206  13225 #63457         JSM IN2        EXECUTE PHASE II
1207  13226 #21754         LDA AR2E       SAVE THE EXPONENT
1208  13227 #31714         STA SAVE         OF AR2 IN SAVE
1209  13230 #70340         AAR B          SET A EQUAL TO
1210  13231 #70076         TCA              6 MINUS THE EXPONENT
1211  13232 #00023         ADA P6           TO BE USED AS A LOOP COUNT
1212*
1213  13233 #00144         ADA N13          IN PHASE III
1214  13234 #70112         SAM *+2        IF THIS QUANTITY IS GREATER
1215  13235 #20132         LDA N1           THAN 12 SET THE
1216  13236 #00031         ADA P13          A REGISTER TO 12
1217  13237 #26535         LDB PI         LET B = THE ADDRESS OF THE TRIG CORDIC TABLE
```

```
          #01  FUNCTIONS FOR 9830

1218  13240 #63654        JSM IN3       EXECUTE PHASE III
1219*
1220  13241 #21716 POOT   LDA SIGN      LOAD THE ORIGINAL EXPONENT WORD
1221  13242 #31754        STA AR2F        AND STORE IT IN AR2
1222  13243 #70452        SAM GONO      SKIP IF THE EXPONENT IS NEGATIVE
1223*
1224  13244 #50012        AND P1        OTHERWISE CLEAR ALL BUT THE MANTISSA
1225  13245 #31754        STA AR2F        SIGN BIT IN THE EXPONENT WORD
1226  13246 #25714        LDB SAVE      LOAD THE EXPONENT FROM AFTER THE RECIPROCATION
1227  13247 #62625        JSM UNRM+1    UNNORMALIZE AR2
1228*
1229  13250 #20023        LDA PI/2      SET PI/2 IN THE
1230  13251 #62754        JSM XAR1      AR1 TEMPOARY
1231*
1232  13252 170400        CMY           CALCULATE
1233  13253 170560        FAA             AR2=AR1-AR2
1234*
1235  13254 #66576 GONO   JMP NORM+1    NORMALIZE THE RESULT
1236*
1237*
1238**********************************
1239*
1240* FIND THE ARCTANGENT OF AN ARGUMENT
1241*
1242  13255 #63296 ATN    JSM ATNS      JUMP TO CALCULATE THE ANSWER IN RADIANS
1243*
1244  13256 #25434 EATN   LDB UNITS     LOAD THE UNITS MODE WORD
1245  13257 #76110        RZB *+2       IF IT IS SET TO DEGREES
1246  13260 #62743        JSM RTOD        CONVERT THE ANSWER TO DEGREES
1247  13261 #22533        LDA GPERR     IF GRADS ARE SET
1248  13262 #14017        CPB GRA         CONVERT THE ANSWER
1249  13263 #62744        JSM RTOD+1      TO GRADS
1250*
1251  13264 #74742 SOUT   SBR 16        CLEAR B AND RETURN
1252  13265 #67201        JMP OUT-1       THRU THE STORE ROUTINE
1253*
1254*
1255**********************************
1257* RAISE NAPERIAN E TO A POWER
1258*
1259  13266 #21703        LDA ADR2
1260  13267 #62751        JSM .FMP
1261  13270 #21703        LDA ADR3
1262  13271 #24012 .EXP   LDB P1        LET THE FUNCTION FLAG BE 1
1263  13272 #62547        JSM INIT      FETCH THE ARGUMENT AND INITIALIZE THE FLAGS
1264  13273 #06543        ADB EN3       IF THE EXPONENT IS 3 OR GREATER
1265  13274 #74413        SHP STM1-1      JUMP TO THE STORE MAX ROUTINE
1266*
1267  13275 #26534        LDB LN10      SUBTRACT AS MANY WHOLE LN10'S
1268  13276 #62605        JSM FRAC        FROM THE NO. AS POSSIBLE
1269  13277 #62624        JSM UNRM      UNNORMALIZE THE RESULT
1270*
1271  13300 #21710        LDA COUNT     LOAD THE NO. OF SUBTRACTS FROM ABOVE
1272  13301 #70542        SAR 12        ISOLATE THE FIRST DIGIT
1273  13302 #70210        SZA STM1+1    IF IT IS NOT ZERO
1274  13303 #25716        LDB SIGN      LOAD THE EXPWORD OF THE ORIGNAL ARGUMENT
1275  13304 #74044        SBL 15        POSITION THE SIGN BIT AND JUMP TO
1276  13305 #65264 STM1   JMP OF+3      THE OVERFLOW/UNDERFLOW ROUTINE
1277*
1278  13306 #25710        LDB COUNT     LOAD THE NO. OF LN10'S
1279  13307 #74404        SBL 8         ISOLATE THE UNITS DIGIT
1280  13310 #74142        SBR 4          OF IT IN B
1281  13311 #21710        LDA COUNT     ISOLATE THE TENS DIGIT
1282  13312 #70342        SAR 8           OF IT IN A
1283  13313 #70344        SAL 9         MULTIPLY THE A REGISTER
1284  13314 #70037        ADB A           BY 10 AND ADD IT
1285  13315 #70704        SAL 2           TO B
1286  13316 #70037        ADB A         THIS QUANTITY WILL BE THE EXPONENT
1287  13317 #35711        STB RCPEG       OF THE UNNORMALIZED ANSWER
1288*
1289  13320 #62704        JSM CLER      CLEAR THE DIGIT STACK
1290  13321 #63523        JSM ATD       INITIALIZE THE DIGIT
1291  13322 #55717        ISZ DIGIT       STACK POINTER
1292  13323 #22545        LDA THETL     PUT THE ADDRESS OF THE LOG CONSTANT STACK IN A
1293  13324 #24023        LDB P6        SET B = 6 LOOPS
1294  13325 #63431        JSM IN        EXECUTE PHASEI
1295*
1296  13326 #24136        LDB N5        TRANSFER THE LAST
1297  13327 #45717        ISZ DIGIT       SIX DIGITS OF THE
1298  13330 171400        MLS             REDUCED ARGUMENT
1299  13331 131717        STA DIGIT,I     INTO THE DIGIT STACK
1300  13332 #77767#       MIB *-3         FROM AR2
1301*
1302  13333 #62753        JSM UTO       SET AR1 = 1
1303  13334 #63603        JSM PTJ       EXECUTE PHASE IV
1304*
1305  13335 #21711 BACK   LDA RCPEG     STORE THE EXPONENT CALCULATED
1306  13336 #31754        STA AR2F        ABOVE IN AR2
1307  13337 171450        NRM           NORMALIZE THE ANSWER
1308*
```

```
          #01  FUNCTIONS FOR 9830

1309  13340 021716        LDA SIGN       IF THE ORIGINAL ARGUMENT
1310  13341 070111        SLA *+2          WAS NEGATIVE RECIPROCATE
1311  13342 062646        JSM RECIP        THE ANSWER
1312  13343 067264        JMP SOUT       JUMP TO THE STORE ROUTINE
1314*
1315*
1316**********************************
1317*
1318* FIND THE NATURAL LOGARITHM OF AN ARGUMENT
1319*
1320  13344 024132 LNX    LDB N1         SET THE FUNCTION FLAG TO -1
1321  13345 062547        JSM INIT       FETCH THE ARGUMENT AND INITIALIZE THE FLAGS
1322*
1323  13346 070314        SES *+6
1324  13347 160205        JSM AERRA,I
1325  13350 177632        DEC -102
1326  13351 024012        LDB P1       SET THE FUNCTION FLAG TO 1
1327  13352 035712        STB FNCFG      FNCFG IS FLAGED TO RETURN -9.9E99 WHEN LOG10 CALL.
1328  13353 065274        JMP SIMAX
1329  13354 074111        SLB *+2
1330  13355 160206 ER47   JSM ERRA,I
1331*
1332  13356 062704        JSM CLER       CLEAR THE DIGIT STACK
1333  13357 063603        JSM PTJ        EXECUTE PHASE IV
1334  13360 055717        DSZ DIGIT      DECREMENT THE DIGIT STACK POINTER
1335*
1336  13361 020023        LDA P6         LET 1 = 6 LOOPS
1337  13362 026534        LDB LN10       LET B = THE ADDRESS OF THE LOG CONSTANT STACK
1338*
1339  13363 063654        JSM IN3        EXECUTE PHASE III
1340*
1341  13364 026540        LDB ADR0       SAVE THE MANTISSA OF THE ANSWER IN
1342  13365 020214        LDA AR2A       IN A TEMPORARY
1343  13366 170004        XFR
1344*
1345  13367 062757        JSM OTT        SET AR2 = 1
1346*
1347  13370 021716        LDA SIGN       LOAD THE ORIGINAL EXPONENT
1348  13371 000013        ADA P256       INCREMENT IT
1349  13372 070153        SAP *+3        IF IT IS NEGATIVE
1350  13373 070071        SLA *+1,C        CLEAR THE MANTISSA SIGN
1351  13374 070076        TCA              AND COMPLEMENT IT
1352  13375 070342        SAR 8          SHIFT IT TO THE LOW 8 BITS
1353*
1354  13376 000156        ADA N100       SUBTRACT 100 FROM IT
1355  13377 024013        LDB D0100      SET B = BCD 0100
1356  13400 070153        SAP *+3        SKIP IF THE EXPONENT IS POSITIVE
1357  13401 000060        ADA P100       OTHERWISE RESTORE IT AND
1358  13402 074742        SBR 16           CLEAR B
1359*
1360  13403 000141        ADA N10        SUBTRACT 10 FROM IT
1361  13404 070152        SAM *+3        SKIP IF IT IS NEGATIVE
1362  13405 004033        ADB D0010      OTHERWISE ADD BCD 0010 TO B
1363  13406 067403        JMP *-3        JUMP BACK TO ADD AGAIN
1364  13407 000061        ADA P10        RESTORE A
1365*
1366  13410 070037        ADB A          ADD THE ONES LEFT TO B
1367  13411 035755        STB AR2E+1     STORE THIS WORD AS THE FIRST 4 DIGITS OF AR2
1368*
1369  13412 020110        LDA EP3        SET THE EXPONENT
1370  13413 031754        STA AR2E         OF AR2 TO +3
1371  13414 062576        JSM NORM+1     NORMALIZE AR2
1372*
1373  13415 022534        LDA LN10       MULTIPLY AR2 BY
1374  13416 062750        JSM .FMP-1      LOG OF 10
1375*
1376  13417 021703        LDA ADR3       NORMALIZE THIS AND THE
1377*
1378  13420 026540        LDB ADR0         RESULT OF THE
1379  13421 062563        JSM DIV          CORDIC ITERATION
1380*
1381  13422 146542        ISZ RSLT,I     NEGATE THE MANTISSA QUANTITY
1382*
1383  13423 021716        LDA SIGN       IF THE EXPONENT OF THE
1384  13424 070113        SAP *+2          ORIGINAL WAS NEGATIVE
1385  13425 045630        ISZ TREGE        NEGATE THE LN10 MULTIPLE
1386  13426 022541        LDA TREG       ADD THESE TWO
1387  13427 026542        LDB RSLT         QUANTITIES FOR THE
1388  13430 066734        JMP .FAD       FINAL ANSWER
1390*
1391*
1392**********************************
1393*
1394* CORDIC LOOP
1395*
1396*
1397**********************************
1398*
1399* PHASE I
1400*
```

#01 FUNCTIONS FOR 9830

```
1401  13431 #31705 IN    STA TABLØ     STORE THE TABLE ADDRESS
1402  13432 #35707       STB SHIFT     STORE THE NO. OF LOOPS
1403*
1404  13433 #62635 PTA   JSM ZERO      DETERMINE IF THE ARGUMENT IS ZERO
1405  13434 #70154       SES *+3       IF IT IS INCREMENT THE
1406  13435 145717       ISZ DIGIT,I     LAST DIGIT IN THE STACK
1407  13436 170402       RET             AND RETURN
1408*
1409  13437 170400       CMY           COMPLEMENT THE ARGUMENT
1410*
1411  13440 #45717       ISZ DIGIT     INCREMENT THE DIGIT STACK POINTER
1412*
1413  13441 #20021       LDA P4        FETCH THE NEXT CONSTANT
1414  13442 #01705       ADA TABLØ     FETCH THE NEXT CONSTANT
1415  13443 #31705       STA TABLØ     INTO AR1
1416  13444 #24213       LDB AR1A
1417  13445 170004       XFR
1418*
1419  13446 #74004       SBL 16        ADD AR1 TO AR2
1420  13447 170420       FDV             UNTIL OVERFLOW
1421*
1422  13450 135717       STB DIGIT,I   STORE THE NO. OF SUBTRACTS BEFORE OVERFLOW
1423*
1424  13451 170400       CMY           RESTORE THE
1425  13452 170560       FXA             ARGUMENT TO ITS LAST NEGATIVE VALUE
1426*
1427  13453 171400       MLS           LEFT SHIFT THE ARGUMENT
1428*
1429  13454 #55707       DSZ SHIFT     DECREMENT THE LOOP COUNTER
1430  13455 #67433       JMP PTA       CONTINUE IF IT IS NONZERO
1431  13456 170402       RET           OTHERWISE RETURN
1433*
1434*
1435**********************************
1436*
1437* PHASE II
1438*
1439  13457 #31714 IN2   STA SAVE      STORE THE SHIFT FACTOR
1440  13460 #35707       STB SHIFT     STORE THE LOOP COUNT
1441  13461 #62753       JSM OTO       SET AR1 = 1
1442*
1443  13462 #22540       LDA AURY      INITIALIZE RY1 AND RY2
1444  13463 #31713       STA RY1         TO THE ADDRESSES OF
1445  13464 #20111       LDA AURX        TWO TEMPORARIES FOR
1446  13465 #31715       STA RY2         INTERMEDIATE STORAGE
1447*
1448  13466 #25712       LDB FNCFG     SKIP TO PTE IF THE ATN
1449  13467 #76650       RZB PTE         IS BEING CALCULATED
1450  13470 #67525 PTL1  JMP PTL       OTHERWISE GO TO PTL
1451*
1452  13471 #76710 PTB   RZB PTN
1453  13472 #67534       JMP PTC
1454*
1455  13473 #55717 PTD   DSZ DIGIT     DECREMENT THE DIGIT STACK POINTER
1456  13474 #25712       LDB FNCFG     SKIP TO PTL IF THE
1457  13475 #75550       SZB PTL1        TAN IS BEING CALCULATED
1458*
1459  13476 #21755 PTO   LDA AR2E+1    LOAD THE FIRST FOUR DIGITS OF AR2
1460  13477 #70542       SAR 12        ISOLATE THE FIRST ONE
1461  13500 #70150       SZA *+3         AND SKIP IF IT IS ZERO
1462  13501 170410       ORS           OTHERWISE RIGHT SHIFT
1463  13502 #67507       JMP PTN         AR2 AND JUMP TO PTN
1464  13503 171400       MLS           LEFT SHIFT AR2 ONCE
1465  13504 #20214 PTE   LDA AR2A      STORE THE NEW VALUE
1466  13505 #25713       LDB RY1         OF Y IN
1467  13506 170004       XFR             RY1
1468*
1469  13507 174400 PTN   CMX           NEGATE AR1 = X
1470  13510 170560       FXA           AR2 = AR2-AR1   (Y-X)
1471  13511 #70314       SES *+6       SKIP ON OVERFLOW
1472  13512 #21713       LDA RY1       OTHERWISE RESTORE
1473  13513 #24214       LDB AR2A      AR2 WITH ITS
1474  13514 170004       XFR             LAST VALUE
1475  13515 174400       CMX           RECOMPLEMENT AR1 = X
1476  13516 #67600 PTF1  JMP PTF       JUMP TO PTF
1477*
1478  13517 174400       CMX           RECOMPLEMENT AR1
1479*
1480  13520 #62635       JSM ZERO      DETERMINE IF AR2 = 0
1481  13521 #70634       SES PTM,C     SKIP IF IT IS NOT
1482  13522 155717       DSZ DIGIT,I   IF IT IS ZERO DECREMENT THE
1483  13523 #22537 ATD   LDA AWRDN       LAST DIGIT AND FINALIZE THE
1484  13524 #66712       JMP CK-1        DIGIT STACK POINTER AND RETURN
1485  13525 #24012 PTL   LDB P1        RIGHT SHIFT AND ROUND UP
1486  13526 #61305       JSM ROUND     AR1 ONCE
1487*
1488  13527 #20214       LDA AR2A      STORE THIS
1489  13530 #25713       LDB RY1         NEW Y IN
1490  13531 170004       XFR             RY1
1491*
```

#01 FUNCTIONS FOR 9830

```
1492  13532 121717        LDA DIGIT,I    IF THE PRESENT DIGIT IS ZERO FOR
1493  13533 071150        SZA PTF1         THE TAN JUMP TO PTF
1494*
1495  13534 170560 PTC    FXA            OTHERWISE CALCULATE AR2=AR2+ARY    (Y=Y+X)
1496*
1497  13535 021713 PTM    LDA RY1        EXCHANGE THE NEW
1498  13536 025715        LDB RY2          AND OLD VALUES
1499  13537 031715        STA RY2          OF Y BY EXCHANGING
1500  13540 035713        STB RY1          THEIR ADDRESSES
1501  13541 020214        LDA AR2A      STORE THE NEW
1502  13542 170004        XFR              Y INTO AR2
1503*
1504  13543 021715        LDA RY2        FETCH THE VALUE
1505  13544 024214        LDB AR2A      OF Y USED IN THE LAST LOOP
1506  13545 170004        XFR
1507*
1508  13546 021712        LDA FNCFG      LOAD THE FUNCTION FLAG
1509  13547 025707        LDB SHIFT      LOAD THE LOOP COUNTER
1510  13550 004132        ADB N1         SUBTRACT 1 FROM IT
1511  13551 074744        SHL 1          MULTIPLY IT BY 2
1512*
1513  13552 070150        SZA *+3        SKIP IF TAN
1514  13553 074076        TCB            COMPLEMENT IT AND
1515  13554 067556        JMP *+2          JUMP
1516  13555 005714        ADB SAVE       ADD THE SHIFT FACTOR TO IT
1517  13556 005714        ADB SAVE         TWICE IF TAN AND ONCE IF ATN
1518*
1519  13557 074117        LDA B          IF THE RESULTING
1520  13560 000143        ADA N12          VALUE IS 12 OR
1521  13561 070453        SAP PTK          MORE SKIP TO PTK
1522*
1523  13562 061305        JSM ROUND      RIGHT SHIFT AR2 B TIMES AND ROUND UP
1524*
1525  13563 025712        LDB FNCFG      LOAD THE FUNCTION FLAG
1526  13564 076110        RZB *+2          AND COMPLEMENT AR2 IF
1527  13565 170400        CMY              THE TAN IS BEING CALCULATED
1528  13566 170560        FXA            ADD THE SHIFTED Y TO X FOR A NEW X
1529*
1530  13567 020214        LDA AR2A       STORE THE NEW VALUE OF X
1531  13570 024213        LDB AR1A       BACK IN AR1
1532  13571 170004        XFR
1533*
1534  13572 021713 PTK    LDA RY1        FETCH THE LAST
1535  13573 024214        LDB AR2A      CALCULATED VALUE OF Y INTO AR2
1536  13574 170004        XFR
1537*
1538  13575 025712        LDB FNCFG      LOAD THE FUNCTION FLAG
1539  13576 155717        DSZ DIGIT,I    DECREMENT THE PRESENT DIGIT
1540  13577 067471        JMP PTB        JUMP TO PTB IF IT IS NOT ZERO
1541*
1542  13600 055707 PTF    DSZ SHIFT      OTHERWISE DECREMENT THE LOOP COUNTER
1543  13601 067473        JMP PTD        JUMP TO PTD IF NOT FINISHED
1544  13602 170402        RET            OTHERWISE RETURN
1546*
1547*
1548**********************************
1549*
1550* PHASE III
1551*
1552  13603 022537 PTJ    LDA AWRDN      INITIALIZE THE
1553  13604 031717        STA DIGIT        DIGIT STACK POINTER
1554*
1555  13605 020030        LDA P12        INITIALIZE THE
1556  13606 031707        STA SHIFT        LOOP COUNT TO 12
1557  13607 025712        LDB FNCFG      JUMP TO PTS IF THE
1558  13610 074312        SBM PTS          LN IS BEING CALCULATED
1559  13611 062757        JSM OTT        DUPLICATE AR1 IN AR2
1560*
1561  13612 025712 PTR    LDB FNCFG      JUMP TO PTI IF THE
1562  13613 074353        SBP PTI          EXP IS BEING CALCULATED
1563  13614 174400        CMX            SUBTRACT
1564  13615 170560        FXA              AR1 FROM AR2
1565  13616 020214 PTS    LDA AR2A       DUPLICATE AR2, THE NEW X, IN AR1
1566  13617 024213        LDB AR1A
1567  13620 170004        XFR
1568  13621 067624        JMP PTH-2      JUMP
1569*
1570  13622 121717 PTI    LDA DIGIT,I    SKIP IF THE PRESENT
1571  13623 070750        SZA GOON-1       DIGIT IN THE STACK IS ZERO
1572*
1573  13624 020061        LDA P10        INITIALIZE THE INNER
1574  13625 031705        STA TABL0        LOOP COUNT TO 10
1575  13626 025707 PTH    LDB SHIFT      SUBTRACT THE OUTER
1576  13627 074076        TCB              LOOP COUNT FROM 12
1577  13630 004030        ADB P12          FOR THE SHIFT FACTOR
1578  13631 070742        SAR 16         RIGHT SHIFT AR1
1579  13632 174430        MRX              THIS MANY TIMES
1580*
1581  13633 025712        LDB FNCFG      IF THE EXP IS BEING CALCULATED
1582  13634 014012        CPB P1           SET THE OVERFLOW BIT IF THE
```

```
        #01  FUNCTIONS FOR 9830

1583  13635 000136        ADA N5          LAST SHIFTED DIGIT IS >=5
1584*
1585  13636 170560        FAA             CALCULATE THE NEW X
1586*
1587  13637 070454        SES PTG         SKIP ON OVERFLOW
1588  13640 055705        DSZ TABL0       OTHERWISE DECREMENT THE INNER LOOP COUNTER
1589  13641 067643        JMP GOON        JUMP BACK IF THIS DIGIT COUNT
1590  13642 067650        JMP PTG           IS LESS THAN 10
1591*
1592  13643 020214 GOON   LDA AR2A        DUPLICATE THE NEW VALUE
1593  13644 024213        LDB AR1A        OF X IN AR1
1594  13645 170004        XFR
1595*
1596  13646 155717        DSZ DIGIT,I     DECREMENT THE STACK DIGIT
1597  13647 067626        JMP PTH         JUMP TO PTH IF IT IS NOT ZERO
1598  13650 045717 PTG    ISZ DIGIT       OTHERWISE INCREMENT THE DIGIT STACK POINTER
1599  13651 055707        DSZ SHIFT       DECREMENT THE MAIN LOOP COUNTER
1600  13652 067612        JMP PTR         JUMP TO PTR IF NOT FINISHED
1601  13653 170402        RET             OTHERWISE RETURN
1603*
1604*
1605**********************************
1606*
1607* PHASE IV
1608*
1609  13654 035705 IN3    STB TABL0       STORE THE CONSTANT TABLE ADDRESS
1610  13655 031704        STA TEMP        STORE THE SHIFT FACTOR
1611  13656 031707        STA SHIFT       STORE THE LOOP COUNT
1612*
1613  13657 020214        LDA AR2A
1614  13660 170000        CLR           CLEAR AR2
1615*
1616  13661 121717 PTP    LDA DIGIT,I     LOAD THE PRESENT DIGIT FROM THE STACK
1617  13662 070076        TCA             COMPLEMENT IT
1618  13663 024012        LDB P1          SHIFT IT INTO
1619  13664 174470        MRY               AR2
1620  13665 025712        LDB FNCFG       LOAD THE FUNCTION FLAG
1621  13666 074151        SLB *+3         SKIP IF LN
1622  13667 055717        DSZ DIGIT       ESTABLISH WHERE THE NEXT DIGIT
1623  13670 067672        JMP *+2
1624  13671 045717        ISZ DIGIT       IS.
1625  13672 055707        DSZ SHIFT       DECREMENT THE LOOP COUNTER
1626  13673 067661        JMP PTP         JUMP BACK IF IT IS NONZERO
1627*
1628  13674 070742        SAR 16          ESTABLISH THE NEW LOOP
1629  13675 025704        LDB TEMP          COUNTER AS 12
1630  13676 074076        TCB             MINUS THE LAST
1631  13677 004030        ADB P12           INITIAL ONE AND
1632  13700 035707        STB SHIFT         SHIFT AR2 RIGHT
1633  13701 174470        MRY               THIS MANY TIMES
1634*
1635  13702 020135 PTQ    LDA N4          FETCH THE NEXT CONSTANT
1636  13703 001705        ADA TABL0       INTO AR1
1637  13704 031705        STA TABL0
1638  13705 024213        LDB AR1A
1639  13706 170004        XFR
1640  13707 070742        SAR 16          CLEAR A
1641  13710 025707        LDB SHIFT       SKIP IF THE LOOP COUNT
1642  13711 076110        RZB *+2         IS NOT ZERO
1643  13712 170402        RET             OTHERWISE RETURN
1644  13713 004132        ADB N1          SHIFT AR1 THE LOOP COUNT
1645  13714 174430        MRX               MINUS 1 TIMES
1646*
1647  13715 000136        ADA N5          IF THE LAST SHIFTED
1648  13716 070175        SEC *+3,C         OUT DIGIT IS = 75
1649  13717 020213        LDA AR1A        INCREMENT THE
1650  13720 170540        MDI               CONSTANT IN AR1
1651*
1652  13721 125717        LDB DIGIT,I     LOAD THE PRESENT DIGIT
1653  13722 074076        TCB             COMPLEMENT IT
1654  13723 171460        FMP             ADD AR1 INTO AR2 THIS MANY TIMES
1655*
1656  13724 025712        LDB FNCFG
1657  13725 074151        SLB *+3
1658  13726 055717        DSZ DIGIT
1659  13727 067731        JMP *+2
1660  13730 045717        ISZ DIGIT
1661  13731 055707        DSZ SHIFT       DECREMENT THE LOOP COUNTER
1662  13732 067702        JMP PTQ         JUMP BACK IF IT IS NOT ZERO
1663  13733 170402        RET             OTHERWISE RETURN
1664*
1665**********************************
1666*
1667* FIND THE LOG BASE TEN OF AN ARGUMENT
1668*
1669  13734 063344 LOG    JSM LNX         TAKE THE NATURAL LOG OF THE ARGUMENT
1670  13735 026534        LDB LN10
1671  13736 055712        DSZ FNCFG         ZERO DIVIDE THE RESULT
1672  13737 066727        JMP DTOR+1        BY LN10
1673  13740 170402        RET             OTHERWISE RETURN WITH THE MAX NO.
1674*
1675*  LAST WORD AVAILABLE 13740B
```

```
        #01  MATH ROUTINED FOR 9830* CALLED *MATH*

1678*
1679* MATH ROUTINES BEGIN HERE
1680*
1681*
1682  11465                ORG 11465B
1683*
1684  11465 #77764 MASK2 OCT 77764
1685*
1686**********************************
1687*
1688* FLOATING POINT SUBTRACT
1689*
1690*
1691  11466 #31703 FSB    STA ADR2
1692  11467 #74117        LDA B
1693  11470 #24057        LDB ATEMP
1694  11471 #35702        STB ADR1
1695  11472 170004        XFR
1696  11473 120057        LDA ATEMP,I
1697  11474 #72111        SLA *+2,S       * CHANGE THE SIGN
1698  11475 #70071        SLA *+1,C       * AND ADD
1699  11476 130057        STA ATEMP,I
1700  11477 #67502        JMP FAD+2
1701*
1702*
1703**********************************
1704*
1705* FLOATING POINT ADDITION
1706*
1707*
1708  11500 #31703 FAD    STA ADR2
1709  11501 #35702        STB ADR1
1710  11502 125702        LDB ADR1,I      "SUBTRACT" ENTRY POINT
1711  11503 #74340        ABR B             SUBTRACT THE
1712  11504 #74076        TCB               EXPONENT OF THE
1713  11505 121703        LDA ADR2,I        SECOND ARGUMENT FROM
1714  11506 #70340        AAR B             THE FIRST ARGUMENT
1715  11507 #70037        ADB A             AND STORE THIS OFFSET
1716  11510 #35701        STB TEMPA         IN A TEMPORARY
1717*
1718  11511 #21703        LDA ADR2        LOAD THE ADDRESS OF THE FIRST ARGUMENT
1719  11512 #74253        SBP *+5         JUMP IF THE 1ST ARI. IS LARGER
1720  11513 #24214        LDB AR2A
1721  11514 170004        XFR             TRANSFER THE SMALLER NUMBER INTO AR2
1722  11515 #24213        LDB AR1A        LOAD THE ADDRESS OF
1723  11516 #67522        JMP *+4         AR1 AND JUMP
1724  11517 #24213        LDB AR1A        TRANSFER THE LARGER NUMBER INTO AR1
1725  11520 170004        XFR
1726  11521 #24214        LDB AR2A        TRANSFER THE OTHER
1727  11522 #21702        LDA ADR1          ARGUMENT INTO THE
1728  11523 170004        XFR               OTHER WORKING REGISTER
1729*
1730*
1731  11524 #21755        LDA AR2E+1      JUMP IF THE SMALLER
1732  11525 #70510        SZA BNZ           ARGUMENT IS ZERO TO STORE THE LARGER
1733*
1734  11526 #21745        LDA AR1E+1      JUMP IF THE LARGER
1735  11527 #72110        RZA *+2           NUMBER IS NON-ZERO
1736  11530 #65254        JMP ZONK        STORE THE LARGER ARGUMENT
1737*
1738*
1739  11531 #21701        LDA TEMPA       LOAD THE OFFSET
1740  11532 #70113        SAP *+2           AND COMPLEMENT IT
1741  11533 #70076        TCA               IF ITS NEGATIVE
1742  11534 #70137        LDB A           IF THE OFFSET IS GREATER
1743  11535 #03465        ADA MASK2         THAN ELEVEN THEN
1744  11536 #70253        SAP *+5         JUMP TO STORE THE
1745  11537 #20213 BNZ    LDA AR1A        LARGER NUMBER
1746  11540 #24214        LDB AR2A
1747  11541 170004        XFR
1748  11542 #65254        JMP ZONK
1749*
1750*
1751  11543 #61305        JSM ROUND       RIGHT JUSTIFY AND ROUND THE SMALLER ARGUMENT
1752*
1753  11544 #21744        LDA AR1E        ADD THE SIGNS OF THE
1754  11545 #01754        ADA AR2E          TWO MANTISSAS AND
1755  11546 #70511        SLA INTLK       JUMP IF SAME
1756*
1757*
1758*
1759  11547 170400        CMY             THEY HAVE ULIKE SIGNS SO COMPLEMENT THE LARGER
1760  11550 170540        FXA             ADD THE MANTISSAS
1761  11551 #70154        SES *+3         IF A CARRY DIGIT IS GENERATED
1762  11552 170400        CMY               COMPLEMENT THE ANSWER AND
1763  11553 #45744        ISZ AR1E          CHANGE THE SIGN OF THE LARGER ARGUMENT
1764  11554 171450        NRM             NORMALIZE THE ANSWER;PUT THE NO. OF SHIFTS IN B
1765  11555 #74076        TCB             COMPLEMENT AND ALLIGN THE
1766  11556 #74404        SBL B             NO. OF SHIFTS
1767  11557 #67572        JMP .FAD.       JUMP TO STORE ROUTINE IF NOT ZERO
```

```
#01  MATH ROUTINED FOR 9830, CALLED 'MATH'

1768*
1769  11560 070074 INTLK SES *+1,C      CLEAR THE E REGISTER
1770  11561 170560       FXA            ADD THE MANTISSAS
1771  11562 070214       SES *+4        IF NO CARRY IS GENERATED
1772  11563 021744       LDA AR1E         LET THE EXPONENT WORD OF THE
1773  11564 031754       STA AR2E       LARGER ARGUMENT
1774  11565 065254       JMP ZONK       JUMP TO THE STORE ROUTINE
1775  11566 170002       MOV            ELSE SHIFT A ONE INTO
1776  11567 070137       LDB A            THE MOST SIGNIFICANT
1777  11570 174470       MRY              LOCATION OF THE ANSWER
1778  11571 024013       LDB P256       LOAD B WITH THE EXPONENT CORRECTION
1779*
1780  11572 021755 .FAD. LDA AR2E+1
1781  11573 070110       SZA *+2        IF THE MANTISSA WAS 0 CLEAR AR2
1782  11574 065237       JMP STAR2
1783  11575 065256       JMP CLAR2
1785*
1786*
1787**********************************
1788*
1789* FLOATING POINT DIVISION
1790*
1791*
1792  11576 035702 FDV   STB ADR1
1793  11577 031703       STA ADR2
1794*
1795  11600 024214       LDB AR2A       PUT THE DIVIDEND INTO AR2
1796  11601 170004       XFR
1797*
1798  11602 021702       LDA ADR1       PLACE THE DIVISOR INTO AR1
1799  11603 024213       LDB AR1A
1800  11604 170004       XFR
1801*
1802  11605 020057       LDA ATEMP      INITIALIZE THE WORD
1803  11606 031475       STA TEMP2        POINTER FOR THE QUOTIENT
1804  11607 170000       CLR            CLEAR THE QUOTIENT REGISTER
1805*
1806*
1807*
1808  11610 021745       LDA AR1F+1
1809  11611 072310       SZA GOOD
1810  11612 164205       JSM AERRA,I
1811  11613 177631       DEC -103
1812  11614 025754       LDB AR2F       LOAD THE DIVIDEND WORD
1813  11615 074073       SBP *+1,C        AND COPAR THE SIGN BIT
1814  11616 065274       JMP STMAX
1815*
1816  11617 021755 GOOD  LDA AR2E+1
1817  11620 072110       SZA *+2
1818  11621 065254       JMP ZONK
1819*
1820  11622 170400       CMY            COMPLEMENT THE DIVISOR
1821  11623 020132       LDA N1         INITIAL FIRST DIVIDEND DIGIT TO -1
1822  11624 024144       LDB M13        INITIALIZE THE DIGIT
1823  11625 035477       STB TEMP4        COUNT TO 13
1824*
1825  11626 024135 NWWD  LDB M4         SET THE WORD DIGIT
1826  11627 035701       STB TEMPA        COUNT TO 4
1827*
1828  11630 074004 NXDG  SBL 4          CLEAR LOCATION FOR NEXT CALCULATED DIGIT AND ALIGN
1829  11631 031476       STA TEMP3      STORE FIRST DIVIDEND DIGIT
1830*
1831  11632 170420 DINC  FDV            SUBTRACT DIVISOR FROM DIVIDEND UNTIL OVERFLOW
1832  11633 074070       SIB *+1        INCREMENT THIS COUNT
1833  11634 045476       ISZ TEMP3      INCREMENT FIRST DIVIDEND DIGIT
1834  11635 067632       JMP DINC       JUMP BACK IF NOT ZERO
1835*
1836  11636 021755       LDA AR2E+1     JUMP OUT OF
1837  11637 001756       ADA AR2E+2       MAIN LOOP
1838  11640 001757       ADA AR2E+3       IF DIVIDEND
1839  11641 072150       SZA NO           IS ZERO (I.E. AN
1840  11642 070134       SES NO,C         EXACT QUOTIENT
1841  11643 067663       JMP YES          WAS FOUND)
1842*
1843  11644 170400 NO    CMY            OTHERWISE RESTORE THE
1844  11645 170560       FXA              DIVIDEND TO ITS LAST POSITIVE VALUE
1845  11646 004132       ADB N1         SUBTRACT ONE FROM THE LAST CALCULATED DIGIT
1846  11647 170400       CMY            COMPLEMENT THE DIVIDEND AGAIN
1847*
1848  11650 171400       MLS            SHIFT THE DIVIDEND LEFT ONE DIGIT
1849  11651 000141       ADA N10        LET THE NEW FIRST DIGIT BE THE ONE SHIFTED OUT-10
1850*
1851  11652 045477       ISZ TEMP4      INCREMENT THE DIGIT COUNT
1852  11653 070112       SAM *+2        JUMP OUT OF LOOP IF DONE OR
1853  11654 067665       JMP DONE         IF DIVISOR HAS A NON BCD DIGIT
1854  11655 045701       ISZ TEMPA      INCREMENT WORD DIGIT COUNT
1855  11656 067630       JMP NXDG       JUMP BACK THRU LOOP IF NON ZERO
1856  11657 135475       STB TEMP2,I    ELSE STORE LAST 4 DIGITS IN QUOTIENT
1857  11660 045475       ISZ TEMP2      INCREMENT THE QUOTIENT WORD POINTER
1858  11661 067626       JMP NWWD         AND JUMP BACK
```

```
          #01  MATH ROUTINES FOR 9830, CALLED 'MATH'


1859*
1860  11662 074604        SBL 4           ALLIGN THE LAST DIGIT
1861  11663 045701 YES    ISZ TEMPA       IF A PERFECT QUOTIENT
1862  11664 067662        JMP *-2           WAS FOUND
1863*
1864  11665 135475 DONE   STB TEMP2,I     STORE THE LAST 4 DIGITS
1865  11666 020057        LDA ATEMP       TRANSFER
1866  11667 000132        ADA N1            THE QUOTIENT
1867  11670 024214        LDB AR2A        INTO A
1868  11671 170004        XFR             WORKING REGISTOR
1869*
1870  11672 171450        NRM             NORMALIZE THE ANSWER
1871  11673 074310        SZB .GOON        SKIP IF IT WAS NORMALIZED  (B=0)
1872*
1873  11674 021753        LDA AR2E-1      LOAD LAST DIGIT
1874  11675 050032        AND P15           AND STORE IT
1875  11676 031757        ADA AR2E+3        IN THE
1876  11677 031757        STA AR2E+3        QUOTIENT
1877  11700 024157        LDB N256        LOAD AN EXP OF -1 INTO B
1878*
1879  11701 105703 .GOON   ADB ADR2,I
1880  11702 021744        LDA AR1E
1881  11703 070076        TCA               EXPONENT OF
1882  11704 000017        ADA P2            THE DIVISOR AND PASS INTO
1883  11705 031744        STA AR1E          THE STORE ROUTINE
1884  11706 065237        JMP STAR2
1885*
1887*
1888**********************************
1889*
1890  11707               ORG 11707B
1891  11707 000000        BSS 11          RESERVED FOR OTHER POINTERS
1892  11722 001077        ABS ACS-*
1893  11723 001126        ABS ASN-*
1894**********************************
1895*
1896* LINKAGE
1897*
1898  11724 002010        ABS LOG-*     LOG BASE 10
1899  11725 001330        ABS ATN-*     ARC TANGENT
1900  11726 001125        ABS TAN-*     TANGENT
1901  11727 001060        ABS COS-*     COSINE
1902  11730 001031        ABS SIN-*     SINE
1903  11731 005123        ABS SGN-*     SIGN
1904  11732 004046        ABS INT-*     INTEGER VALUE
1905  11733 004227        ABS FSR-*     SQUARE ROOT
1906  11734 005155        ABS ABS-*     ABSOLUTE VALUE
1907  11735 001407        ABS LNX-*     NATURAL LOG
1908  11736 001333        ABS .EXP-*    E TO THE POWER
1909*
1910*


          #01  NUMBER CONVERSION ROUTINES

1912  17000               ORG 17000B
1913*
1914  01256         CLR2 EQU CLAR2
1915  00051         LDVSR EQU .46         POINTER TO .1E4
1916  01471         DIVSR EQU L2T2
1917  17000 037400 B37.4 OCT 37400
1918*
1919* CONVERT FULL-PRECISION FLOATING NUMBER TO INTEGER
1920*
1921* FLOATING NUMBER IS IN AR2
1922*
1923*      JSM FLTFX
1924*      ---        ERROR EXIT (OVERFLOW)
1925*      ---        NORMAL RETURN WITH RESULT IN (B)
1926*
1927* NEGATIVE INTEGER IN 2'S COMPLEMENT FORM
1928*
1929  17001 074742 FLTFX SBR 16         CLEAR B
1930  17002 021754       LDA AR2        LOOK AT EXPONENT
1931  17003 070652       SAM PP2        DONE IF FRACTIONAL
1932  17004 000157 FLT1  ADA M256        ELSE DECREMENT 10'S EXPONENT
1933  17005 031471       STA L2T2       AND SAVE IT.
1934  17006 020061       LDA .10         *10
1935  17007 061201       JSM IMPY
1936*
1937*    ON RETURN THE A-REG IS SET NEGATIVE ON OVERFLOW. TESTING
1938*    IS NOT MADE UNTIL THE NEXT DIGIT IS ADDED IN.
1939*
1940  17010 070137       LDB A
1941  17011 171400       MLS              NEXT DIGIT
1942  17012 070037       ADB A           ADD NEW DIGIT
1943  17013 074352 FLT2  SBM RET1          OVERFLOW?
1944  17014 021471       LDA L2T2        RECALL EXPONENT
1945  17015 071353       SAP FLT1         LOOP IF POSITIVE
1946  17016 070111       SLA *+2          TEST MANTISSA SIGN
1947  17017 074076       TCB
1948  17020 064357 PP2   JMP RETN2
1949*
```

```
          #01  NUMBER CONVERSION ROUTINES

1950*    CONVERT FULL-PRECISION FLOATING NUMBER TO ROUNDED INTEGER
1951*
1952*     FLOATING NUMBER IN AR2
1953*       JSM FLTRF
1954*     --- ERROR EXIT
1955*       ---   NORMAL RETURN WITH RESULT IN B
1956*
1957  17021 #63001 FLTRF JSM FLTFX      GO GET INTEGER
1958  17022 #64444 RET1  JMP XFAR2      P+1 RETURN FOR OVERFLOW ERRORS
1959  17023 #76210       RZB **4        IS (B) =0?
1960  17024 #31471       STA L2T2       YES, SAVE AR2 EXP AT L2T2
1961  17025 #00013       ADA B400       ADD EXP OF 1 TO CHECK IF<.1
1962  17026 #71512       SAM PP2        IF EXP IS STILL NEGATIVE RETURN
1963  17027 171400       MLS           GET MS FRACTIONAL DIGIT
1964  17030 #00136       ADA M5        ADD -5
1965  17031 #71352       SAM PP2       DONE IF .LT. 5
1966  17032 #74113       SBP **2        IS B POSITIVE ?
1967  17033 #74076       TCB           NO - MAKE IT SO
1968  17034 #74070       SIB **1       ADD 1 FOR ROUNDING
1969  17035 #67013       JMP FLT2      USE FLTFX TO RESTORE SIGN,
1970*                    AND OVERFLOW TEST
1971*
1972*
1973*
1974*  OUTPUT AN INTEGER GIVEN IN (B) TO THE I/O BUFFER
1975*
1976  17036 #63055 OUTIN JSM FXFLT
1977  17037 #35470       STB L2T1         (B) = EXPONENT WORD OF AR2
1978  17040 #61314       JSM OUGIT
1979  17041 #25470       LOB L2T1        LOOK AT EXPONENT
1980  17042 #04157       ADB M256        DECREMENT 10'S EXPONENT
1981  17043 #75613       SBP *-4        LOOP IF POSITIVE
1982  17044 170402 .R1   RET
1983*
1984*  TRANSFER A BCD DIGIT INTO D12 OF AR2
1985*
1986  17045 #31470       STA L2T1       ALTERNATE ENTRY POINT
1987  17046 171400 XFER  MLS
1988  17047 #21757       LDA AR2+3
1989  17050 #41470       IOR L2T1
1990  17051 #31757       STA AR2+3
1991  17052 170402       RET
1992*
1993*************************************************
1994*
1995  17053 #00000       BSS 1        CHECKSUM WORD FOR THIS 256 WORD BLOCK
1996*
1997*************************************************
1999*
2000*    SIGN ROUTINE
2001*
2002*    ON RETURN AR2 = 1 IF ARG WAS +, 0 IF 0, AND -1 IF NEGATIVE
2003*
2004  17054 #61317 SGN   JSM SGNS       SGN FLOWS INTO FXFLT
2005*
2006* CONVERT INTEGER TO FULL-PRECISION FLOATING NUMBER
2007* FLOATING NUMBER BUILT IN AR2
2008*
2009*     LOB N
2010*     JSM FXFLT
2011*
2012  17055 #61256 FXFLT JSM CLRA2       CLEAR AR2
2013  17056 #75310       SZB .R1        RETURN IF B IS ZERO
2014  17057 #74153       SBP **3        AND GET ABSOLUTE VALUE
2015  17060 #74076       TCB           MAKE B POSITIVE
2016  17061 #45754       ISZ AR2       NOTE THE SIGN OF THE INTEGER
2017  17062 #20051       LDA LDVSR        INITIALIZE THE DIVISOR POINTER
2018  17063 #31471       STA DIVSR
2019  17064 121471 FXFL  LDA DIVSR,I
2020  17065 #70513       SAP .IK
2021  17066 #35470       STB L2T1
2022  17067 #63046      JSM XFER      UNITS POSITION
2023  17070 171450       NRM
2024  17071 #04142       ADB M11        COMPUTE EXPONENT
2025  17072 #74076       TCB               PLACEMENT
2026  17073 #74404       SBL 8          SHIFT B LEFT 8
2027  17074 #05754       ADB AR2       MERGE IN THE SIGN BIT
2028  17075 #35754       STB AR2        STORE IT IN AR2
2029  17076 170432       RET
2030  17077 #70076 .IK   TCA
2031  17100 #31467       STA LOCL2        SAVE NEGATIVE DIVISOR
2032  17101 #20132       LDA M1
2033  17102 #70070       SIA **1          INCREMENT INTEGER COUNTER
2034  17103 #05447       ADB LOCL2
2035  17104 #75713       SBP *-2
2036  17105 105471       ADB DIVSR,I      RESTORE DIVIDEND
2037  17106 #63045       JSM XFER-1     MERGE A INTO D12 SLOT
2038  17107 #45471       ISZ DIVSR        ADVANCE TO NEXT DIVISOR IN TABLE
2039  17110 #67044       JMP FXFL
2040*****************************
```

```
        #01  NUMBER CONVERSION ROUTINES


2041* ARSOLUTE VALUE
2042*
2043  17111 #60445 ARS   JSM XFAR2+1
2044  17112 #25754       LDB AR2
2045  17113 #74071       SLB **1,C
2046  17114 #67075       JMP .IK-2
2047*
2048*****************************
2049*
2050* CONVERT FULL-PRECISION FLOATING NUMBER TO SPLIT
2051*
2052* FLOATING NUMBER IS IN AR2
2053*     LDB ADDRESS FOR SPLIT
2054*     JSM FLTSP
2055*
2056  17115 160205 SPLUF JSM AERRA,I     THE EXPONENT = 10XXXXXX
2057  17116 177625       DEC -107       THIS IS AN UNDERFLOW CONDITION
2058  17117 #61256       JSM CLRA2        CONDITION SO CLEAR AR2
2059  17120 #67122       JMP FLTSP+1      AR2 IS ZERO - PROCEED
2060*
2061  17121 #35727 FLTSP STB MBIN2       SAVE STARTING ADD. OF SPLIT
2062  17122 #21754       LDA AR2        GET SIGN
2063  17123 #50012       AND .1           OF MANTISSA
2064  17124 #31726       STA MBIN1      AND SAVE
2065  17125 #25754       LDB AR2        LOAD EXPONENT FOR OVERFLOW CHECK
2066  17126 #74706       RBR 15         POSITION BIT 15 IN LSB
2067  17127 #74111       SLB **2        IS BIT 15 = 1?
2068  17130 #75273       SBP SPLUF,C    YES - IF BIT 14 =1; CONTINUE
2069  17131 #74333       SBP OK1,C      NO - IF BIT 14 = #; JUMP OVER ERROR
2070*
2071  17132 160205       JSM AERRA,I    OVERFLOW ERROR
2072  17133 177626       DEC -106
2073  17134 #61274       JSM STMAX      SET AR2 = THE MAX NUMBER
2074  17135 #23000       LDA B37,4       SET EXPONENT
2075  17136 #31754       STA AR2           = E63
2076*
2077  17137 #24017 OK1   LDB .2
2078  17140 #70742       SAR 16
2079  17141 174470       MRY              SHIFT AR2 MANTISSA RIGHT
2080  17142 #21754       LDA AR2          PICK UP EXPONENT WORD
2081  17143 #70300       AAR 7            SHIFT OFF 7 BITS
2082  17144 #41726       IOR MBIN1       ADD IN MANTISSA SIGN
2083  17145 #70404       SAL 8
2084  17146 #41755       IOR AR2+1        ADD IN FIRST TWO DIGITS
2085  17147 131727       STA MBIN2,I    STASH FIRST WORD
2086  17150 #45727       ISZ MBIN2
2087  17151 #21756       LDA AR2+2
2088  17152 131727       STA MBIN2,I      STASH SECOND WORD
2089  17153 170402       RET
2090*
2092*
2093*   BUILD A NUMBER IN AR2
2094*     FIRST CHARACTER IN A
2095*
2096  17154 #70137 NUMCK LDB A
2097  17155 #61256       JSM CLRA2          CLEAR AR2
2098  17156 #20132       LDA M1
2099  17157 #31745       STA EXCNT      SET EXPONENT COUNT TO -1
2100  17160 #31746       STA DPFLG      SET DECIMAL POINT FLAG TO -1
2101  17161 #20143       LDA M12
2102  17162 #31744       STA SIGDG      SET SIGNIFICANT DIGIT COUNT TO -12
2103  17163 #74117       LDA B          RECALL CHARACTER
2104  17164 #67175       JMP NUMC1
2105*
2106*  PROCESS A DECIMAL POINT
2107*
2108  17165 #21746 .ANUM LDA DPFLG       DECIMAL POINT YET ?
2109  17166 #72250       RZA NUMC1-2     NO - SET NUMBER FOUND AND GET NEXT CHAR.
2110  17167 #55745       DSZ EXCNT        YES - DECR. EXPONENT COUNT
2111  17170 #67174       JMP NUMC1-1
2112  17171 #45746 NUMCP ISZ DPFLG      SET FLAG
2113  17172 160206       JSM ERRA,I     FLAG ALREADY SET, 2 DECIMAL POINTS
2114  17173 #45754       ISZ AR2        SET EXPONENT NON-ZERO; NUMBER FOUND
2115  17174 #61055       JSM GNEXT      GET NEXT CHAR.
2116  17175 #60750 NUMC1 JSM DIGCK      DIGIT?
2117  17176 #67215       JMP NUMC2        NO - IT'S NOT A DIGIT
2118  17177 #76210       RZB NUMCD      ZERO?
2119  17200 #21744       LDA SIGDG      YES,
2120  17201 #10143       CPA M12        ANY SIGNIFICANT DIGITS YET?
2121  17202 #67165       JMP .ANUM       NO - NON-SIGNIFICANT ZERO
2122*  PROCESS A SIGNIFICANT DIGIT
2123  17203 #21746 NUMCD LDA DPFLG      DECIMAL POINT FOUND YET?
2124  17204 #70150       SZA **3
2125  17205 #45745       ISZ EXCNT      NO, INC COUNT
2126  17206 #67207       JMP **1
2127  17207 #21744       LDA SIGDG      12 DIGITS
2128  17210 #70230       SIA NUML1        FOUND YET?  IF YES SKIP
2129  17211 #31744       STA SIGDG      UPDATE COUNT
2130  17212 #35470       STB L2T1          MERGE LATEST DIGIT
2131  17213 #63046       JSM XFER           INTO D12 OF AR2
2132  17214 #67173 NUML1 JMP NUMC1-2
2133*
```

```
        #01  NUMBER CONVERSION ROUTINES


2134*  PROCESS NON-DIGIT MANTISSA CHAR
2135*
2136*
2137  17215 #10051 NUMC2 CPA .46        DECIMAL POINT?
2138  17216 #67171       JMP NUMCP      YES
2139  17217 #25754       LDB AR2
2140  17220 #76110       RZB *+2        ANY MANTISSA FOUND?
2141  17221 170402       RET            NO. RETURN TO P+1
2142  17222 #10071       CPA E          "E"?
2143  17223 #67250       JMP NUMCE      YES
2144  17224 #74742       SBR 16         NO. SET 0 EXPONENT
2145  17225 #72053       SAP *+1,S      SET FIXED POINT FLAG
2146*  CHARACTER IS NOT PART OF NUMBER
2147  17226 #45746 NUMC3 ISZ DPFLG      COMPLEMENT EXPONENT
2148  17227 #67231       JMP *+2
2149  17230 #74076       TCB              IF NEGATIVE
2150  17231 #05745       ADB EXCNT      ADD EXPONENT DUE TO DECIMAL PLACEMENT
2151  17232 #35745       STB EXCNT      STORE EXPONENT
2152  17233 171450       NRM            LEFT JUSTIFY MANTISSA
2153  17234 #25745       LDB EXCNT      RECALL EXPONENT
2154  17235 #74404       SBL 8          LEFT JUSTIFY EXPONENT
2155  17236 #45721       ISZ .SIGN      .SIGN WAS SET IN "CONST" ROUTINE.
2156  17237 #67241       JMP *+2
2157  17240 #74070       SIB *+1        ADD IN SIGN OF MANTISSA
2158  17241 #70115       SEC *+2        IF MANTISSA IS 0
2159  17242 #74742       SBR 16           THEN SET EXPONENT TO 0
2160  17243 #35754       STB AR2        STORE EXPONENT WORD
2161  17244 #04021       ADR .4         SET FLOATING FLAG (BIT 2)
2162  17245 #70133       SAP *+2,C      LEAVE (B) AT .4 IF THERE WAS AN EXPONENT
2163  17246 #04133       ADB M2         NO. SET FIXED FLAG (BIT1)
2164  17247 #64357       JMP RETN2      RETURN TO P+2
2165*  PROCESS EXPONENT
2166  17250 #24132 NUMCE LDB M1
2167  17251 #35724       STB GFLAG      SET INTCK TO RETURN ON ERROR
2168  17252 #61166       JSM GSIGN      GET EXPONENT SIGN
2169  17253 #35746       STB DPFLG      SAVE IT
2170  17254 #60750       JSM DIGCK      DIGIT?
2171  17255 160206 NUMER JSM ERRA,I     NO. MISSING EXPONENT
2172  17256 #25745       LDB EXCNT      COMPUTE -(ABSOLUTE VALUE
2173  17257 #04156       ADB M100         OF MAX. ALLOWALE EXPONENT)
2174  17260 #60676       JSM INTCK      GET ABSOLUTE VALUE OF EXPONENT
2175  17261 #67264       JMP *+3
2176  17262 #77550       RZB NUMER        EXPONENT OVERFLOW
2177  17263 #67226       JMP NUMC3        EXPONENT OF ZERO GIVEN
2178  17264 #55706       DSZ SBPTR      RESET BUFFER POINTER
2179  17265 #67226       JMP NUMC3      EXPONENT OF ZERO GIVEN
2180  01744         SIGDG EQU AR1
2181  01745         EXCNT EQU AR1+1
2182  01746         DPFLG EQU AR1+2
2184*
2185*  CONVERT A NUMBER IN AR2 INTO SYNTAX FORMAT AND STORE IT
2186*     (B)= EXPONENT WORD
2187*        (A) = NEXT CHARCTER. MUST BE SAVED.
2188*
2189  17266 #31474 NUMOP STA TEMP1
2190  17267 #74117       LDA B
2191  17270 #50105       AND B377       GET MANTISSA SIGN AND FLOAT FLAG
2192  17271 #70006       RAR 1          POSITION FLAG AND MOVE SIGN TO BIT 15
2193  17272 #72113       SAP *+2,S
2194  17273 #40107       IOR B1000      SET MANTISSA SIGN IN BIT 9
2195  17274 #24030       LDB .12
2196  17275 #05744       ADB SIGDG      GET SIGNIFICANT DIGIT COUNT
2197  17276 #76110       RZB *+2
2198  17277 #74070       SIB *+1        IF 0. SET TO 1
2199  17300 #74544       SBL 5          MOVE TO OPERAND FIELD
2200  17301 #74217       IOR B          MERGE
2201  17302 #55706       DSZ SBPTR
2202  17303 141706       IOR SBPTR,I    COMBINE WITH OPERAND
2203  17304 131706       STA SBPTR,I      AND STORE
2204  17305 #74202 NUMO1 SBR 5          COMPUTE NUMBER OF
2205  17306 #04022       ADR .5
2206  17307 #74042       SBR 2            WORDS TO BE TRANSFERRED +1
2207  17310 #35477       STB TEMP4      SAVE COUNT
2208  17311 #21754       LDA AR2
2209  17312 #50157       AND M256       GET EXPONENT
2210  17313 #25755       LDB AR2+1
2211  17314 #74342       SBR B
2212  17315 #74217       IOR B          COMBINE WITH D1 AND D2
2213  17316 #60456       JSM SBPUD      UPDATE POINTER
2214  17317 171400       MLS
2215  17320 171400       MLS            SHIFT MANTISSA
2216  17321 #24214       LDB AR2A       GET MANTISSA ADDRESS
2217  17322 #74070 NUMO2 SIB *+1        UPDATE MANTISSA ADDRESS
2218  17323 #74517       LDA B,I    GET READY TO STORE WORD IN SYN. BUFF.
2219  17324 #60456       JSM SBPUD      UPDATE SYNTAX ADDRESS
2220  17325 #55477       DSZ TEMP4      MORE WORDS TO BE TRANSFERRED?
2221  17326 #67322       JMP NUMO2        YES
2222  17327 #21474       LDA TEMP1      NO. RECALL CHAR AND RETURN
2223  17330 170402       RET
```

#01 NUMBER CONVERSION ROUTINES

```
2225*      SNFLT   ROUTINE
2226*
2227*     THIS ROUTINE TRANSFERS A
2228*     NUMBER IN SYNTAX FORMAT TO
2229*     AR2 AND STORES IT IN STANDARD
2230*     FORMAT
2231*
2232*     ENTRY CONDITIONS:
2233*
2234*         TEMPS=WORD POINTER TO
2235*               START OF NUMBER
2236*
2237*     EXIT CONDITIONS:
2238*
2239*         TEMPS=WORD POINTER TO FIRST
2240*               WORD FOLLOWING NUMBER
2241*           AR2=NUMBER IN STANDARD FORM
2242*
2243*
2244  17331 021472 SNFLT LDA TEMPS
2245  17332 070070       SIA *+1        TRANSFER THE NUMBER
2246  17333 024214       LDB AR2A       INTO AR2
2247  17334 170004       XFR
2248*
2249  17335 121472       LDA TEMPS,I
2250  17336 031470       STA L2T1
2251  17337 070202       SAR 5
2252  17340 050032       AND B17        GET THE # OF DIGITS
2253  17341 000022       ADA .5
2254  17342 070042       SAR 2          COMPUTE HOW MANY WORDS THIS IS
2255  17343 070137       LDB A
2256  17344 001472       ADA TEMPS
2257  17345 031472       STA TEMPS      MOVE TEMPS TO THE LAST WORD OF THE NUMBER
2258  17346 004214       ADB AR2A
2259  17347 070742       SAR 16
2260  17350 014112       CPB B1760
2261  17351 067355       JMP *+4
2262  17352 074557       STA B,I        CLEAR THE REMAINDER OF
2263  17353 074070       SIB *+1        THE MANTISSA DIGITS
2264  17354 067350       JMP *-4
2265*
2266  17355 024017       LDB .2         RIGHT SHIFT AR2
2267  17356 174470       MRY            TWO PLACES
2268  17357 021754       LDA AR2
2269  17360 070404       SAL 8
2270  17361 041755       IOR AR2+1      MERGE INTO AR2+1 DIGITS ONE AND TWO
2271  17362 031755       STA AR2+1
2272  17363 021754       LDA AR2        LOAD THE EXPONENT
2273  17364 050157       AND M256       AND MASK
2274  17365 025470       LDB L2T1
2275  17366 074402       SBR 9          INCLUDE THE SIGN BIT
2276  17367 074111       SLB *+2
2277  17370 070070       SIA *+1
2278  17371 031754       STA AR2
2279  17372 170402       RET
2280**********************************
```

#01 MORE 9830 FUNCTIONS

```
2282****************************************
2283  16000              ORG 16000B
2284*
2285*
2286* INTEGER ROUTINE
2287*
2288  16000 062015 INT   JSM FDFRC      TRANSFER ARG. INTO AR1. AND TEST FOR FRACTIONAL
2289  16001 066005       JMP *+4        P+1, FRACTIONAL ARGUMENT
2290*
2291  16002 060444       JSM XFAR2      P+2, NON FRACTIONAL ARG.
2292  16003 171450       NRM            TRANSFER TO AR2, AND NORMALIZE
2293  16004 065254       JMP ZONK
2294*
2295  16005 062002       JSM INT+2
2296  16006 025744       LDB AR1        LOAD (B) WITH THE EXPONENT
2297  16007 074113       SBP *+2        IF THE EXPONENT WAS NEGATIVE
2298  16010 061256       JSM CLAR2      CLEAR AR2
2299  16011 075551       SLB INT+4      RETURN IF THE ARG WAS +.
2300  16012 020214       LDA AR2A       THE ARGUMENT WAS - AND IT HAD A
2301  16013 024020       LDB ONE        FRACTIONAL PART
2302  16014 164217       JMP .FSBA,I    SO SUBTRACT ONE FROM IT
2303*
2304**********************************************
2305*
2306*
2307*    SUBROUTINE FOR INT AND UPARROW
2308*
2309  16015 024213 FDFRC LDB AR1A
2310  16016 170004       XFR
2311  16017 020012       LDA .1         SET THE FLAG
```

```
        #01  MORE 9830 FUNCTIONS

2312  16020 #31714        STA SAVE
2313  16021 #25744        LDB AR1         LOAD THE EXPONENT
2314  16022 #74512        SHM UP1+7       EXIT IF EXP IS FRACTIONAL
2315  16023 #74340        ABR B
2316  16024 #04142        ADR M11
2317*
2318  16025 #74313 UP1    SBP *+6         MUST BE AN INTEGER IF (B) IS +
2319  16026 170410        CRS             RIGHT SHIFT AR1
2320  16027 #74070        SIB *+1
2321  16030 #71650        SZA UP1         SKIP BACK IS SHIFTED DIGIT WAS ZERO
2322  16031 #45714        ISZ SAVE        OTHERWISE INCR. THE FLAG FOR FRACTIONAL PART
2323  16032 #66025        JMP UP1         SKIP BACK TO FINISH INTEGERATION
2324  16033 #55714        DSZ SAVE        SKIP IF THERE WAS NO FRACTIONAL PART
2325  16034 170402        RET             P+1 RETURN FOR FRACTIONAL PART
2326  16035 #64357        JMP RETN2       P+2 RETURN IF IT WAS AN INTEGER
2328*
2329  16036 #01734 ADRYY  OCT 1734        ADDRESS OF REGISTOR Y
2330  16037 #13344 ILNX   DEF LNX
2331  16040 #12550 IINIT  DEF INIT+1
2332  16041 #13266 I.EXP  DEF .EXP-3
2333*
2334*  RET4 IS USED WHEN THE NUMBER IS ZERO
2335*
2336  16042 #21741 RET4   LDA XC+1        LOAD DIGITS 1-4 OF THE POWER
2337  16043 #70410        SZA ER49        IF THE POWER IS ZERO ERROR 49
2338  16044 #21740        LDA XC          LOAD THE EXPONENT OF THE POWER
2339  16045 #74742        SBR 16          CLEAR (B) FOR STMAX ROUTINE
2340  16046 #70211        SLA *+4         RETURN IF THE POWER IS POSITIVE; AR2 = #
2341  16047 #61274        JSM STMAX       SET AR2 = +9.9999E99; NEGATIVE POWER
2342  16050 160205        JSM AERRA,I
2343  16051 177630        DEC -104
2344  16052 170402        RET
2345*
2346  16053 160206 ER49   JSM ERRA,I      0+0 ERROR
2347*
2348**********************
2349*
2350* RAISE A NUMBER TO A POWER
2351*
2352**********************
2353*
2354  16054 #35702 UPARO  STB ADR1        STORE THE ADDRES OF THE POWER,(A) IS SAVED BY INIT
2355*
2356  16055 162040        JSM IINIT,I
2357  16056 #21702        LDA ADR1        TRANSFER THE POWER INTO
2358  16057 #24111        LDB R1740       XC REGISTOR
2359  16060 170004        XFR
2360  16061 #71055        SEC RET4        STORE ZERO IF IT IS ZERO
2361*
2362  16062 #21702        LDA ADR1        LOAD THE ADDRESS OF THE POWER
2363  16063 262015        JSM FDERC       GO CHECK IF THERE IS A FRACTIONAL PART
2364  16064 #66137        JMP .UPA1       YES THERE WAS
2365*
2366  16065 #25747 UP3    LDB AR1+3       SAVE THE D12 DIGIT IN CASE THERE
2367  16066 #35714        STB SAVE        IS AN OVERFLOW IN FLTFA
2368  16067 #21702        LDA ADR1
2369  16070 #60445        JSM XFAR2+1
2370  16071 160226        JSM FLTFA,I     CONVERT AR2 TO AN INTEGER
2371  16072 #66153        JMP UP4         IT'S AN INTEGER BUT TO LARGE
2372*
2373*     RAISE AN ARGUMENT TO AN INTEGER POWER
2374*
2375  16073 #35714 IPWR   STB SAVE
2376  16074 #74113        SBP *+2
2377  16075 #74076        ICB
2378  16076 #35710        STB COUNT       SAVE THE ABS(POWER)
2379  16077 #20020        LDA ONE
2380  16100 #26036        LDB ADRYY       INITIALIZE THE RESULT TO ONE
2381  16101 170004        XFR
2382  16102 #21703        LDA AUR2
2383  16103 #24057        LDB ATEMP       PLACE THE BASE IN REGISTOR Y
2384  16104 170004        XFR
2385*
2386  16105 #21710 IPW1   LDA COUNT       LOAD THE POWER
2387  16106 #70411        SLA IPW2        IS THE POWER ODD?
2388*
2389  16107 #20057        LDA ATEMP       YES
2390  16110 #26036        LDB ADRYY
2391  16111 160220        JSM .FMPA,I     MULTIPY BY PARTIAL RESULT
2392  16112 #26036        LDB ADRYY
2393  16113 #20214        LDA AR2A        AND PLACE IT BACK
2394  16114 170004        XFR
2395  16115 #21710        LDA COUNT
2396*
2397  16116 #70002 IPW2   SAR 1           EVEN POWER ENTRY POINT, DIVIDE POWER BY 2
2398  16117 #31710        STA COUNT       AND RESTORE
2399*
2400  16120 #70410        SZA UPW         IF THE POWER IS ZERO, JUMP TO EXIT PART
2401*
2402  16121 #20057        LDA ATEMP
```

```
          #01  MORE 9830 FUNCTIONS

2403  16122 024057        LDB ATEMP
2404  16123 160220        JSM .FMPA,I    THE POWER WAS EVEN SO MULTIPY
2405  16124 024057        LDB ATEMP      IT BY ITSELF
2406  16125 020214        LDA AR2A       AND PLACE IT BACK
2407  16126 170004        XFR
2408  16127 066105        JMP IPW1
2409*
2410  16130 021714 DPW    LDA SAVE       ALMOST DONE
2411  16131 070152        SAM *+3        WAS THE SIGN OF THE EXPONENT NEG?
2412  16132 022036        LDA ADRYY      NO, TRANSFER THE ANSWER INTO AR2
2413  16133 064445        JMP XFAR2,1
2414  16134 020020        LDA ONE        YES, TAKE THE RECIPICAL OF THE ANSWER
2415  16135 026036        LDB ADRYY
2416  16136 164221        JMP .FDVA,I
2417*
2418*
2419*
2420*
2421  16137 125703 .UPA1  LDB ADR2,I     LOAD THE EXPONENT WORD OF THE BASE
2422  16140 074131        SLB UP2,C      SKIP IF IT IS POSITIVE
2423*
2424*
2425  16141 160206 ERR0   JSM ERRA,I     (-BASE) * (FRACTIONAL EXPOENT)
2426*
2427*
2428  16142 035714 UP2    STB SAVE
2429  16143 021703        LDA ADR2       LOAD THE ADDRESS OF THE BASE
2430  16144 162037        JSM ILNX,I     TAKE NATURAL LOG OF THE BASE
2431  16145 024111        LDB B1740      LOAD THE ADDRESS OF THE EXPONENT(SAVED IN XC)
2432  16146 162041        JSM I.EXP,I    EXP ( POWER * LOG ( BASE ))
2433*
2434  16147 021714        LDA SAVE       LOAD THE LAST DIGIT OF THE ORIGNAL EXPOENET
2435  16150 070451        SLA OVER
2436  16151 045754        ISZ AR2        NEGATE THE ANSWER IF IT IS ODD
2437  16152 065254        JMP ZONK       STORE ANSWER
2438*
2439*
2440*     INTEGER POWER, BUT GREATER THAN 32676
2441*
2442  16153 125703 UP4    LDB ADR2,I
2443  16154 074131        SLB *+2,C
2444  16155 066157        JMP *+2
2445  16156 035714        STB SAVE       THE BASE WAS +, SO CLEAR BIT 0
2446  16157 135703        STB ADR2,I     TAKE THE ABS(BASE)
2447  16160 066143        JMP UP2+1
2448*
2450*
2451  16161 170402 OVER   RET
2452*
2453* FLOATING SQUARE ROOT
2454*
2455*
2456  16162 024213 FSR    LDB AR1A
2457  16163 170004        XFR
2458  16164 020214        LDA AR2A
2459  16165 170000        CLR
2460*
2461  16166 021745        LDA AR1F+1     IF THE ARGUMENT
2462  16167 071510        SZA OVER         IS ZERO RETURN
2463*
2464*
2465  16170 024022        LDB P5         MULTIPLY THE ARGUMENT
2466  16171 171460        FMP              BY 5
2467  16172 025744        LDB AR1E       RIGHT SHIFT THE ARGUMENT
2468  16173 074444        SHL 7            ONCE IF THE EXPONENT IS
2469  16174 074112        SBM **2          ODD AND TWICE IF ITS GVEN
2470  16175 074070        SIB **1     AND ONCE MOR IF A CARRY
2471  16176 074070        SIB **1     EXISTS FROM THE MULTIPLY
2472*
2473  16177 061306        JSM ROUND+1    ROUND IF THE LAST SHIFTED DIGIT IS >=5
2474*
2475  16200 020213        LDA AR1A       INITIALIZE THE ANSWER
2476  16201 070070        STA **1        ANSWER WORD POINTER 1 AND 2
2477  16202 031701        STA TEMPA        TO THE SECOND WORD OF THE
2478  16203 031476        STA TEMP3        WORKING REGISTER
2479  16204 170000        CLR            CLEAR THE WORKING REGISTER
2480*
2481  16205 021744        LDA AR1F       IF THE RADICAND IS
2482  16206 070131        SLA *+2,C      NEGATIVE - RETURN.
2483  16207 160206 ER48     JSM ERRA,I
2484*
2495  16210 070000        AAR 1          DIVIDE THE EXPONENT WORD BY 2,
2486  16211 050163        AND ZAP          CLEAR BITS 1-7
2487  16212 031744        STA AR1E         AND STORE
2488*
2489  16213 020117        LDA AD5        INITIALIZE THE
2490  16214 031475        STA TEMP2        'FIVE' WORD   TO 0500
2491  16215 020004        LDA D1000      INITIALIZE THE
2492  16216 031477        STA TEMP4        'N'  WORD     TO 0100
2493*
```

#01 MORE 9830 FUNCTIONS

```
2494  16217 021477 CONT  LDA TEMP4     IF THE LOW DIGIT OF THE
2495  16220 070111       SLA *+2         'N'    WORD IS A 1 THEN
2496  16221 045476       ISZ TEMP3       INCREMENT POINTER 2
2497  16222 070146       XAR 4          ROTATE THE 'N'    WORD ONE
2498  16223 031477       STA TEMP4       BCD AND REPLACE
2499*
2500  16224 021475       LDA TEMP2      MASK OUT THE LAST
2501  16225 070056       CMA             FIVE FROM THE
2502  16226 151701       AND TEMPA,I     WORKING
2503  16227 131701       STA TEMPA,I     REGISTER
2504*
2505  16230 021475       LDA TEMP2      IF THE LOW DIGIT OF THE
2506  16231 070111       SLA *+2         'FIVE' WORD IS A 5 THEN
2507  16232 045701       ISZ TEMPA       INCREMENT POINTER 1
2508  16233 070146       XAR 4          ROTATE THE 'FIVE' WORD
2509  16234 031475       STA TEMP2       ONE BCD AND REPLACE
2510  16235 101701       ADA TEMPA,I    LOCATE THE NEXT FIVE INTO
2511  16236 131701       STA TEMPA,I     THE WORKING REGISTER
2512*
2513  16237 125476       LDB TEMP3,I    LOAD THE WORKING REGISTER WORD TO BE INCREMENTED INT
2514  16240 170400       CMY            COMPLEMENT THE ARGUMENT
2515*
2516  16241 020061       LDA M10        SET THE INCREMENT
2517  16242 031466       STA TEMP5       COUNTER = 10
2518*
2519  16243 135476       STB TEMP3,I    STORE B INTO THE WORKING REGISTER
2520  16244 170560       FXA            ADD THE WORKING REGISTER TO THE ARGUMENT
2521  16245 005477       ADB TEMP4      INCREMENT THE PROPER DIGIT OF B
2522  16246 055466       DSZ TEMP5      DECREMENT THE INCREMENT COUNTER AND EXIT LOOP IF ZER
2523  16247 071635       SEC *-4,C      IF NO OVERFLOW OCCURRED JUMP BACK INTO LOOP
2524*
2525  16250 170400       CMY            COMPLEMENT THE ARGUMENT
2526*
2527  16251 021755       LDA AR2E+1     JUMP OUT OF THE
2528  16252 001756       ADA AR2E+2      MAIN LOOP IF
2529  16253 001757       ADA AR2E+3      AN PERFECT SQUARE
2530  16254 070134       SES *+2,C       HAS BEEN
2531  16255 070350       SZA PFSQ        FOUND
2532*
2533  16256 170560       FXA            RESTORE THE ARGUMENT
2534  16257 171400       MLS            SHIFT THE ARGUMENT LEFT ONCE
2535*
2536  16260 021476       LDA TEMP3      IF 12 DIGITS HAVE
2537  16261 010057       CPA ATEMP       BEN CALCULATED
2538  16262 066302       JMP LSTLP       EXIT FROM MAIN LOOP
2539  16263 066217       JMP CONT        ELSE CONTINUE
2540*
2541  16264 025701 PFSQ  LDB TEMPA      A PERFECT SQUARE HAS BEEN FOUND
2542  16265 014057       CPB .1000
2543  16266 066272       JMP *+4         DO NOT STORE THE LAST DIGIT
2544  16267 004025       ADB P8           SO INCREMENT THE LEAST SIGNIFICANT
2545  16270 021475       LDA TEMP2        DIGIT (A5) OF THE WORKING REGISTER
2546  16271 074557       STA B,I          BY 5 AND
2547  16272 170560       FXA              THEN LEFT SHIFT
2548  16273 171400       MLS              IT ONCE
2549*
2550  16274 070250       SZA *+5       IF A DIGIT WAS SHIFTED OU
2551  16275 070137       LDB A
2552  16276 174470       MRY           SHIFT IT BACK IN AND INCREMENT THE EXPONENT
2553  16277 024013       LDB P256
2554  16300 065237       JMP STAR2
2555  16301 065240       JMP STAR2+1   OTHERWISE STORE THE ANSWER AND RETURN
2556*
2557  16302 021750 LSTLP LDA ARIF+4     LOCATE THE LAST DIGIT
2558  16303 070542       SAR 12          ALLIGN IT
2559  16304 175400       DLS             AND SHIFT IT INTO THE ANSWER
2560  16305 020213       LDA AR1A
2561  16306 024214       LDB AR2A
2562  16307 170004       XFR
2563  16310 065254       JMP ZONK
2564*
2565*
2566**********************************
2567*
2568* FLOATING POINT MULTIPLY
2569*
2570*
2571  16311 031702 FMP   STA ADR1
2572  16312 074117       LDA B
2573  16313 024213       LDB AR1A       TRANSFER THE MULTIPLICAND INTO A
2574  16314 170004       XFR              TEMPORARY
2575*
2576  16315 021702       LDA ADR1       TRANSFER THE MULTIPLIER
2577  16316 024057       LDB ATEMP      INTO A TEMPORARY
2578  16317 170004       XFR
2579  16320 020214       LDA AR2A
2580  16321 170000       CLR            INITIALIZE WORD POINTER FOR THE
2581  16322 031475       STA TEMP2      MULTIPLIER AND CLEAR AR2
2582*
```

```
        #01  MORE 9830 FUNCTIONS


2583  16323 #20134        LDA M3          INITIALIZE THE
2584  16324 #31476        STA TEMP3         WORD COUNT TO 3
2585*
2586  16325 #55475 NXWD   DSZ TEMP2       DECREMENT THE WORD POINTER
2587  16326 125475        LDB TEMP2,I     LOAD 4 DIGITS OF THE MULTIPLIER
2588*
2589  16327 171460        FMP             ACCUMULATE THE PRESENT DIGIT MULTIPLICAND
2590  16330 #04145        ADB N14         SET BITS #-3 =#001 IN B
2591  16331 174470        MRY             ALLIGN THE TEMP ANSWER FOR NEXT DIGIT
2592  16332 #74142        SBR 4           SHIFT IN THE NEXT MULTIPLIER DIGIT
2593*
2594  16333 171460        FMP
2595  16334 #04145        ADB N14
2596  16335 174470        MRY
2597  16336 #74142        SBR 4
2598*
2599  16337 171460        FMP
2600  16340 #04145        ADB N14
2601  16341 174470        MRY
2602  16342 #74142        SBR 4
2603*
2604  16343 171460        FMP
2605  16344 #04145        ADB N14
2606  16345 174470        MRY
2607*
2608  16346 #45476        ISZ TEMP3       INCREMENT THE WORD COUNT AND
2609  16347 #66325        JMP NXWD          JUMP IF NON ZERO
2610*
2611  16350 #25755        LDB AR2E+1      SKIP IF THE FIRST
2612  16351 #74542        SBR 12            DIGIT OF THE ANSWER
2613  16352 #76350        SZB *+7           IS NOT ZERO
2614*
2615  16353 #70137        LDB A           IT IS ZERO SO
2616  16354 171400        MLS               NORMALIZE ANSWER
2617  16355 #05757        ADB AR2E+3      PICK UP THE 12TH DIGIT
2618  16356 #35757        STB AR2E+3        WHICH WAS SHIFTED OUT ABOVE
2619*
2620  16357 124057        LDB ATEMP,I     LOAD THE EXPONENT WORD OF THE
2621  16360 #65237        JMP STAR2         MULTIPLIER AND EXIT TO RETURN ANSWER
2622*
2623  16361 #61307        JSM ROUND+2     ROUND UP ANSWER IF 13TH DIGIT IS >=5
2624  16362 124057        LDB ATEMP,I     LOAD EXPONENT OF THE MULTIPLIER
2625  16363 #04013        ADB P256        INCREMENT IT
2626  16364 #65237        JMP STAR2       EXIT TO RETURN ANSWER
2627*
2628*
2629*  CONVERT SPLIT TO FULL-PRECISION FLOATING POINT NUMBER
2630*
2631*  LDB ADDRESS OF SPLIT
2632*  JSM SPLFL
2633*
2634*  FULL FLOATING POINT NUMBER IN AR2 ON RETURN
2635*
2636  16365 #61256 SPLFL  JSM CLAR2       CLEAR AR2
2637  16366 #74517        LDA B,I         LOAD EXPONENT , SIGN, D1 AND D2
2638  16367 #31754        STA AR2         SAVE TEMPORARLY IN AR2
2639  16370 #50105        AND B377        MASK D1 AND D2
2640  16371 #31755        STA AR2+1       AND STORE
2641  16372 #74070        SIB *+1
2642  16373 #74517        LDA B,I
2643  16374 #31756        STA AR2+2       STORE D3 THUR D6
2644  16375 171450        NRM
2645  16376 #74742        SBR 16          FIX
2646  16377 #21754        LDA AR2          UP
2647  16400 #70340        AAR 8             THE
2648  16401 #70131        SLA *+2,C          FULL
2649  16402 #74070        SIB *+1             EXPONENT
2650  16403 #73444        SAL 7
2651  16404 #74217        IOR B
2652  16405 #31754        STA AR2
2653  16406 170402        RET
2654*
2655  16407 #20000        BSS 1           *** CHECKSUM WORD ***
2656*
2657                      END
*  NO ERRORS*


       CROSS-REFERENCE TABLE


.1     0020     0522     2063     2311
.10    0071     0072     0574     1934
.100   0070     0685
.1000  0069     0534     2542
.11    0035     0043
.12    0036     0527     2195
.13    0039
.15    0048     0083     0529
.16    0049     0084
.1F4   006B
```

CROSS-REFERENCE TABLE

.2	0027	0523	2077	2266						
.21	0050									
.22	0051	0087	0684							
.23	0052	0088								
.28	0053									
.3	0028	0045								
.31	0054	0090								
.32	0055									
.33	0056									
.34	0057	0091								
.37	0058									
.4	0029	0524	2161							
.40	0059									
.41	0060									
.43	0061									
.45	0062									
.46	0063	1915	2137							
.48	0064									
.5	0030	0525	2205	2253						
.58	0065	0094								
.6	0031	0044	0572							
.63	0066									
.7	0032	0573								
.8	0033	0526								
.9	0034									
.99	0074									
.ANUM	2108	2121								
.BUFA	0180									
.CS	0989	0982								
.EXP	1262	1908	2332							
.FAD	0930	1388								
.FAD.	1780	1767								
.FADA	0185	0582								
.FDV	0920	0836	0987	1056	1162					
.FDVA	0188	0584	2416							
.FMP	0952	0794	0856	1260	1374					
.FMPA	0187	0583	2391	2404						
.FSBA	0186	2302								
.FSR	0190	0866								
.GOON	1879	1871								
.IK	2070	2020	2046							
.LNUM	0314									
.NPWR	0504									
.R1	1982	2013								
.SIGN	0399	0400	2155							
.SLBF	0181									
.T.	1138	1128								
.TEMP	0350	0351								
.UPA1	2421	2364								
A	0000	1044	1147	1284	1286	1366	1715	1742	1776	1940	1942
	2096	2255	2551	2615							
ABS	2043	1906									
ACS	1028	1892									
ACSC	1035	1031									
ACTSA	0174										
ADS	0710	2489									
ADD	0582	0932									
ADP0	0694	1341	1378								
ADR1	0549	0746	0750	0793	0930	1694	1709	1710	1727	1792	1798
	2354	2357	2362	2368	2571	2576					
ADR2	0550	0551	0726	0727	0730	0732	0920	0931	1259	1691	1708
	1713	1718	1793	1879	2382	2421	2429	2442	2446		
ADR3	0551	0835	0919	0934	1057	1177	1261	1376			
ADRSS	0327										
ADRX	0692	0848	0851	0872	0876	1445					
ADRY	0693	0694	1443								
ADRYE	0565										
ADRYY	2329	2380	2390	2392	2412	2415					
AERRA	0176	1324	1810	2056	2071	2342					
AR1	0413	0182	0421	2180	2181	2182	2296	2313	2366		
AR1A	0182	0961	1154	1158	1416	1531	1566	1593	1638	1649	1722
	1724	1745	1799	2309	2456	2475	2560	2573			
AR1E	0421	1174	1734	1753	1763	1772	1808	1880	1883	2461	2467
	2481	2487	2557								
AR2	0415	0183	0422	0999	1930	1988	1990	2016	2027	2028	2044
	2062	2065	2075	2080	2084	2087	2114	2139	2160	2208	2210
	2268	2270	2271	2272	2278	2436	2638	2640	2643	2646	2652
AR2A	0183	0736	0753	0768	0847	0855	0865	0933	0949	1342	1465
	1473	1488	1501	1505	1530	1535	1565	1592	1613	1720	1726
	1746	1795	1867	2216	2246	2258	2300	2393	2406	2458	2561
	2579										
AR2E	0422	0766	0767	0778	0789	0801	0819	0820	0821	0825	0989
	1007	1077	1101	1111	1121	1125	1142	1146	1173	1200	1207
	1221	1225	1306	1367	1370	1459	1731	1754	1773	1780	1812
	1816	1836	1837	1838	1873	1875	1876	2527	2528	2529	2611
	2617	2618									
ARYAD	0403										
ASN	1065	1893									
ASNS	1042	1029	1065								
ATN	1483	1119	1290								

CROSS-REFERENCE TABLE

ATFMP	0534	1693	1696	1699	1802	1865	2383	2389	2402	2403	2405
	2537	2577	2620	2624							
ATN	1242	1899									
ATNS	1187	1058	1242								
AWRDN	0691	0884	1483	1552							
B	0000	0805	1049	1519	1692	2103	2190	2200	2212	2218	2262
	2546	2572	2637	2642	2651						
B100	0095										
B1000	0102	2194									
B10K	0114	0115	0535								
B12K	0116	0117	0206								
B133	0096										
B140	0097										
B1400	0103	0204	0697								
B17	0083	2252									
B170K	0121										
B1740	0104	0203	0692	2358	2431						
B1760	0105	0158	2260								
B177	0098										
B1774	0106										
B1775	0107										
B1777	0108										
B20	0084	0699									
B22	0085										
B23	0086										
B2400	0110	0710									
B26	0087										
B27	0088										
B3000	0111										
B36	0089										
B37	0090										
B37.4	1917	2074									
B377	0099	2191	2639								
B400	0100	0530	0698	1961							
B4000	0112	0162	0700								
B42	0091										
B44	0092										
B52	0093										
B6K	0113	0167									
B72	0094										
B777	0101										
BACK	1305										
BADDR	0291										
BADRP	0204										
BEND	0198	0199									
BHSTP	0454										
BLANK	0388										
BNZ	1745	1732									
C1	0598	0703									
C2	0628	0688	0704								
C3	0665	0689									
C4	0670	0690									
C5	0015										
C6	0011	0114									
C7	0022	0100									
C8	0675	0683									
CHRSA	0219										
CK	0891	1484									
CLAR2	0501	0854	1051	1099	1194	1783	1914	2298	2636		
CLER	0884	1199	1249	1332							
CLERA	0160										
CLINE	0316										
CLP1A	0217										
CLRA2	1914	2012	2058	2097							
COMCE	0446										
COMCK	0445										
CONST	0494										
CONT	2494	2539									
COS	1006	1901									
COUNT	0556	0780	0785	0787	0845	0858	0870	1043	1271	1278	1281
	2378	2386	2395	2398							
CPLAC	0331										
CRCKA	0228										
CRLF	0507										
CRLF1	0508										
CRTN	0037	0038	0039	0528							
CTYPE	0330										
D0010	0699	1171	1362								
D0100	069A	1355									
D1000	0535	2491									
D8000	057A	1187									
DCOMA	0210										
DELSA	0175										
DFLAG	0390										
DIGCK	0473	2116	2170								
DIGIT	0563	0890	1291	1297	1299	1334	1406	1411	1422	1455	1482
	1492	1539	1553	1570	1596	1598	1616	1622	1624	1652	1658
	1660										
DINC	1831	1834									
DIV	0746	1159	1379								

CROSS-REFERENCE TABLE

DIVSR	1916	2018	2019	2036	203A						
DONE	1864	1853									
DPERR	0043	091A	2944								
DPFLG	2182	2100	210A	2112	2123	2147	2169				
DPW	2410	2400									
DROST	0317										
DSTRT	0303										
DTOR	0918	0906	0911	1672							
DVD	0584	0921									
E	0080	2142									
E1	0235										
E10	0244										
E15	0249										
E20	0254										
E25	0259										
E30	0264										
E35	0269										
E40	0274										
E45	0279										
E5	0239										
E50	0284										
EATN	1244	1035	1066								
ECALA	0215										
ELTNA	0169										
EN12	0702	1192									
EN3	0701	1264									
ENDS	0490										
ENTFA	021A										
ENTFA	0189										
EOL	003A										
EOLCK	0493										
EOST	0491										
EP3	0697	1369									
ER47	1330	0285									
ER48	2483	0286									
ER49	2346	0287	2337								
ER50	2425	028A									
ERRA	0177	1330	2113	2171	2346	2425	2483				
EXCNT	2181	2099	2110	2125	2150	2151	2153	2172			
EXP	0401	0402									
EXPOP	0118										
F	0081										
FAD	170A	0185	1700								
FAD1	0929	0863	1013	1034							
FAL1	1099	1045									
FAP	1194										
FDAAA	0222										
FDERC	2309	2288	2363								
FDV	1792	0188									
FDV1	0919	0776									
FETCH	049A										
FILLA	0201										
FIRST	0367										
FIX	0511										
FLGBT	0073	0578									
FLOAT	0346										
FLT1	1932	1945									
FLT2	1943	1949									
FLTFA	0193	2370									
FLTFX	1929	0193	1957								
FLTRA	0192										
FLTRF	1957	0192									
FLTSA	0195										
FLTSP	2061	0195	2059								
FMP	2571	0187									
FN	014A										
FNCFG	055A	0724	1327	1448	1456	1508	1525	1538	1557	1561	151
	1620	1656	1671								
FND	0472										
FND1A	0224										
FNDAA	0223										
FNDPS	0486										
FOPBS	0206										
FORMA	0172										
FORMF	0336										
FORMS	0335										
FORMT	0334										
FPAGE	0461										
FRAC	0776	1083	1268								
FROMA	0122										
FSB	1691	0186									
FSCA	0173										
FSCE2	0451										
FSCE4	0456										
FSR	2456	0190	1405								
FTMP1	0337										
FTMP2	0338										
FWAM	0157										
FWUP	0296										

CROSS-REFERENCE TABLE

```
EXFL    2019    2039
EXFLA   0194
EXFLT   2012    0194    1976
GET     1115    1081    1108    1114
GET3    1080    1086
GETAD   0471
GETCH   0481
GETSK   0485
GFG     0554    0901    1030    1045    1080    1172
GFLAG   0404    2167
GFRST   0484
GNEXT   0483    2115
GONO    1235    1222
GOOD    1816    1809
GOON    1592    1571    1589
GPARM   0434
GPERR   0683    2910    1247
GRA     0706    1248
GRAF    0700    0852
GSIGN   0496    2168
GTCON   0510
GTMP    0482
GTOPP   0476
GTOR    0910
GTSTA   0213
HFLG1   0321
HSTPT   0302
I       0067
I.EXP   2332    2432
ICNT    0432
IINIT   2331    2356
ILNX    2330    2430
IMPY    0497    1935
IN      1401    1140    1294
IN2     1439    1150    1206
IN3     1609    1218    1339
INF     0077
INIT    0724    0903    1046    1188    1263    1321    2331
INITS   0466
INREA   0226
INT     2288    1904    2295    2299
INTCK   0469    2174
INTEA   0178
INTEL   0340
INTFL   0375
INTGR   0402
INTLK   1769    1755
IOBUF   0359
IOHEA   0229
IOSTA   0225
IPW1    2386    2408
IPW2    2397    2387
IPWR    2375
ITR     0752    0833    0868    1032    1131
KEYFG   0324
L2T1    0347    1977    1979    1986    1989    2021    2130    2250    2274
L2T2    0348    1916    1933    1944    1960
LASTN   0398
LBOP    0119
LDEF    0499
LOVSR   1915    2017
LETCK   0474
LIMIT   0475
LISTA   0167
LN10    0688    1267    1337    1373    1670
LNFD    0072
LNINC   0315
LNUM    0467
LNX     1320    1669    1907    2330
LOCL1   0345    0542
LOCL2   0346    2031    2034
LOG     1669    1898
LPCK    0447
LPCKE   0448
LPOP    0120
LROMA   0123
LSTAK   0299
LSTLP   2557    2538
LSTPT   0301
LTR     0489
LWAM    0295
M1      0124    0531    2032    2098    2164
M10     0131    0532
M100    0144    0687    2173
M1024   0146
M11     0132    1009    2024    2316
M12     0133    0577    2101    2120
M13     0134    0686    1822
M14     0135    0711
M15     0136
```

CROSS-REFERENCE TABLE

M2	0125	0575	2163							
M23	0137									
M25	0138									
M256	0145	0533	1932	1980	2209	2273				
M3	0126	2583								
M32	0139									
M36	0140									
M4	0127	0576	1825							
M48	0141									
M5	0128	0423	1964							
M6	0129									
M64	0142									
M80	0143									
M9	0130									
MASK1	0153									
MASK2	1684	1743								
MASK5	0154									
MAW	0294									
MAXIN	0400									
MAXSN	0147									
MBIN1	0406	2064	2082							
MBIN2	0407	2061	2085	2086	2088					
MBOX1	0351									
MDIMA	0209									
MER8	0455									
MINIT	0465									
MLT	0583	0952								
MODE	0305									
MSFLG	0389									
MVTOH	0487									
N1	0531	0889	1028	1215	1320	1510	1644	1821	1845	1866
N10	0532	1360	1849							
N100	0687	1354								
N12	0577	0806	1520							
N13	0686	1213								
N14	0711	2590	2595	2600	2605					
N2	0575	0885	0992							
N256	0533	0783	1109	1877						
N4	0576	1635								
N5	0423	1296	1583	1647						
NOISP	0382									
NOWRT	0506									
NEXT1	0460									
NEXTA	0477									
NEXTL	0459									
NMOTA	0200									
NO	1843	1839	1840							
NOEOF	0492									
NORM	0762	0747	0751	0791	1106	1235	1371			
NPRIN	0383									
NPWRT	0505									
NUM	0396	0397								
NUMC1	2116	2104	2109	2111	2132					
NUMC2	2137	2117								
NUMC3	2147	2177	2179							
NUMCA	0211									
NUMCD	2123	2118								
NUMCE	2166	2143								
NUMCK	2096	0211								
NUMCP	2112	0246	2138							
NUMER	2171	0245	2176							
NUML1	2132	2128								
NUMO1	2204									
NUMO2	2217	2221								
NUMOA	0212									
NUMOP	2189	0212								
NWWD	1825	1858								
NXDG	1828	1855								
NXTDT	0304									
NXTST	0405									
NXWD	2586	2609								
OBLNK	047A									
ODGIT	0502	197A								
OF	0426	1276								
OK	1197	1191	1193							
OK1	2077	2069								
ONE	0045	0834	0862	0960	0965	2301	2379	2414		
ONEXT	0479									
OPCHA	0171									
OPDMK	0079									
OPMSK	0076									
OPTYP	0397									
OTO	0960	1302	1441							
OTPTR	0293									
OTT	0965	0981	1011	1345	1559					
OUT	1174	1000	1134	1137	1169	1252				
OUTCR	0480									
OUTIA	0179									
OUTIN	1976	0179								
OVCHA	0216									

CROSS-REFERENCE TABLE

OVER	2451	2435	2462								
P1	0522	0705	1042	1075	1116	1224	1262	1326	1485	1582	1618
P10	0574	1364	1573	2516							
P100	0685	1357									
P12	0527	1204	1555	1577	1631						
P13	0528	0808	1216								
P15	0529	1874									
P2	0523	0706	1882								
P22	0684	1203									
P256	0530	1143	1166	1348	1778	2553	2625				
P4	0524	0887	1413								
P5	0525	2465									
P6	0572	1149	1211	1293	1336						
P7	0573	1127									
P8	0526	2544									
PARMA	0306										
PARMN	0307										
PBPTO	0313										
PBPTP	0310										
PBUFF	0311										
PBUFO	0312										
PEND	0365										
PEXMA	0165										
PFLAG	0391										
PFSQ	2541	2531									
PGIN0	0488										
PGINT	0468										
PI	0689	1033	1217								
PI/2	0044	1012	1088	1229							
PLACE	0384										
PMODE	0329										
POOT	1220	1053	1195								
PP2	1948	1931	1962	1965							
PPFF	0366										
PRADD	0308										
PRFLG	0332										
PTA	1404	1430									
PTB	1452	1540									
PTC	1495	1453									
PTD	1455	1543									
PTE	1465	1449									
PTF	1542	1476									
PTF1	1476	1493									
PTG	1598	1587	1590								
PTH	1575	1568	1597								
PTI	1570	1562									
PTJ	1552	1303	1333								
PTK	1534	1521									
PTL	1485	1450									
PTL1	1450	1457									
PTM	1497	1481									
PTN	1469	1452	1463								
PTO	1459										
PTP	1616	1626									
PTQ	1635	1662									
PTR	1561	1600									
PTS	1565	1558									
QTCHA	0230										
QTOP	0109	0161									
RAD	0705	0907									
RAX	0875	0871									
RBOP	0117										
RBUFF	0360										
RCOUT	0333										
RCPFG	0557	0991	1076	1093	1097	1132	1136	1152	1165	1167	1287
	1305										
RDYDA	0166										
RECIP	0833	1135	1197	1311							
REFD	0161										
REFDL	0339										
RES	0417										
RESB1	0463										
RESBP	0464										
RET1	1958	1943									
RET2	0431										
RET4	2336	2360									
RETN2	0430	1948	2164	2326							
RETN3	0429										
ROUND	0425	0810	1117	1486	1523	1751	2473	2623			
RPCK	0449										
RPCKE	0450										
RPOP	0115										
RSAVE	0292										
RSLT	0696	0731	0735	0752	0860	0873	0875	0929	1160	1176	1381
	1387										
RSLTE	0373	0696									
RSPTR	0418										
RSTAK	0416										
RSTKA	0158										
RSULT	0436										

CROSS-REFERENCE TABLE

RTOD	0944	1246	1249							
RTT	0735	0762	0966							
RUNA	0163									
RY1	0559	1155	1157	1444	1466	1472	1489	1497	1500	1534
RY2	0561	1446	1498	1499	1504					
S	0082									
SALTA	0207									
SANS	0933	0922	0953	0983						
SAVBP	0462									
SAVE	0560	1208	1226	1439	1516	1517	2312	2322	2324	2367	2375
	2410	2428	2434	2445						
SAVEA	0325									
SAVER	0326									
SBADR	0394									
SBPTR	0387	2178	2201	2202	2203					
SBPUD	0441	2213	2219							
SHUFA	0199									
SCRT	0845	0985	1047							
SCRT1	0870	1055								
SEC	0320									
SFLAG	0393									
SGN	2004	1903								
SGNS	0503	2004								
SHIFT	0555	1402	1429	1440	1509	1542	1556	1575	1599	1611	1625
	1632	1641	1661							
SIGDG	2180	2102	2119	2127	2129	2196				
SIGN	0562	0728	0994	1052	1078	1110	1112	1115	1164	1220	1274
	1389	1347	1383							
SIN	0976	1014	1902							
SLCOD	0328									
SLEND	0202									
SLWST	0457									
SNFLA	0197									
SNFLT	2244	0197								
SOUT	1251	1312								
SPLFA	0196									
SPLFL	2636	0196								
SPLUF	2656	2068								
SSRPT	0395									
SSYMA	0170									
ST	0377									
STAR2	0424	1179	1782	1884	2554	2555	2621	2626		
START	0162	0007								
STEXP	0453									
STM1	1276	1265	1273							
STMAX	0427	1103	1328	1814	2073	2341				
STN	0900	0976	1006	1074						
STOR1	0438									
STORE	0437									
STPFL	0323									
STROP	0440									
STSRH	0452									
SUMPA	0227									
SYE12	0495									
SYE25	0470									
SYMC1	0443									
SYMC2	0442									
SYMCR	0444									
SYMTP	0298									
SYTSA	0209									
TABLE	0553	1401	1414	1415	1574	1588	1609	1636	1637	
TAN	1074	0978	1900							
TEMP	0552	0553	0554	0555	0556	0557	0558	0559	0560	0561	0562
	0563	0564	0902	0977	1102	1610	1629				
TEMP1	0352	2189	2222								
TEMP2	0353	1803	1856	1857	1864	2490	2500	2505	2509	2545	2581
	2586	2587									
TEMP3	0354	1829	1833	2478	2496	2513	2519	2536	2584	2608	
TEMP4	0355	1823	1851	2207	2220	2492	2494	2498	2521		
TEMP5	0542	2517	2522								
TEMPA	0541	1716	1739	1826	1854	1861	2477	2502	2503	2507	2510
	2511	2541									
TEMPS	0349	2244	2249	2256	2257						
TERM	0364										
TFLAG	0318										
TFLG1	0322										
THETL	0703	1292									
THETT	0704	1124									
TMPA	0203										
TMPOP	0184										
TREG	0695	0748	0945	0948	1161	1386					
TREGE	0372	0695	0947	1385							
TSRHA	0168										
TSTPT	0300										
TTY	0509										
TWOPI	0698	1082									
TYPFL	0078										
UFLAG	0392										
UKPTR	0309										
UNITS	0319	0905	1244								

CROSS-REFERENCE TABLE

```
UNRM   0801    1227    1269
UNSTK  0458
UP1    2318    2314    2321    2323
UP2    2428    2422    2447
UP3    2366
UP4    2442    2371
UPARA  0191
UPARO  2354    0191
VALUE  0297
VAROA  0214
WBUFF  0433
WIDTH  0385
WRDN   0564    0691
WRITE  0159
XAR1   0961    1089    1230
XC     0412    2336    2338
XEC4A  0164
XEC6A  0221
XFAR1  0435
XFAR2  0439    1958    2043    2291    2369    2413
XFER   1987    2022    2037    2131
XSCRA  0220
YC     0414
YES    1961    1841
ZAP    0152    2486
ZERO   0818    0739    0802    1084    1094    1404    1480
ZONK   0500    1736    1748    1774    1818    2293    2437    2563
ZOR    1179    1190
```

#01 9830

```
#001              ASMB,A,B,L
#003*
#004*
#005*
#006*
#007*             ****    9830A TAPE CASSETTE SYSTEM   ****
#008*
#009*
#010*
#011  00000               ORG 0
#012*
#013  00000 164121        JMP START,I   START OF MONITOR
#014  00001 043060        OCT 43060     CONSTANT NEEDED BY PROCESSOR
#015  00002 000000        BSS 1        FOR INTERPT
#016*
#017  00003 000000 C6     OCT 0         ONE
#018  00004 010000        OCT 010000    0000
#019  00005 000000        OCT 0         0000
#020*
#021  00006 000300 C5     OCT 0000000   PI/2
#022  00007 012560        OCT 12560     1570
#023  00010 074543        OCT 074543    7963
#024  00011 023200        OCT 023200    2680
#025*
#026  00012 000001 .1     DEC 1
#027*
#028  00013 000400 C7     OCT 000400    180/PI
#029  00014 053451        OCT 053451    5729
#030  00015 053571        OCT 053571    5779
#031  00016 050450        OCT 050450    5128
#032*
#033  00017 000002 .2     DEC 2
#034  00020 000003 .3     DEC 3
#035  00021 000004 .4     DEC 4
#036  00022 000005 .5     DEC 5
#037  00023 000006 .6     DEC 6
#038  00024 000007 .7     DEC 7
#039  00025 000010 .8     DEC 8
#040  00026 000011 .9     DEC 9
#041  00027 000013 .11    DEC 11
#042  00030 000014 .12    DEC 12
#043  00031 000015 CRTN   OCT 15
#044  00031        EOL    EQU CRTN
#045  00031        .13    EQU CRTN
#046*****************
#047* POINTERS TO FUNCTIONS CONSTANTS
#048*
#049  00027        DPERR EQU .11       POINTER TO 180/PI
#050  00023        PI/2  EQU .6        POINTER  TO PI/2
#051  00020        ONE   EQU .3        POINTER TO THE CONSTANT ONE
#052*
#053*****************
#054  00032 000017 .15    DEC 15
#055  00033 000020 .16    DEC 16
#056  00034 000025 .21    DEC 21
#057  00035 000026 .22    DEC 22
```

#01 9830

```
0058  00036 000027 .23    DEC 23
0059  00037 000034 .28    DEC 28
0060  00040 000037 .31    DEC 31
0061  00041 000040 .32    DEC 32
0062  00042 000041 .33    DEC 33
0063  00043 000042 .34    DEC 34
0064  00044 000045 .37    DEC 37
0065  00045 000050 .40    DEC 40
0066  00046 000051 .41    DEC 41
0067  00047 000053 .43    DEC 43
0068  00050 000055 .45    DEC 45
0069  00051 000056 .46    DEC 46
0070  00052 000060 .48    DEC 48
0071  00053 000072 .58    DEC 58
0072  00054 000077 .63    DEC 63
0073  00055 000111 I       OCT 111
0074  00056 023420 .1E4    DEC 10000
0075  00057 001750 .1000   DEC 1000
0076  00060 000144 .100    DEC 100
0077  00061 000012 .10     DEC 10
0078  00061        LNFD    EQU .10
0079  00062 100000 FLGBT   OCT 100000
0080  00063 000143 .99     DEC 99
0081  00064 100004 INTFL   OCT 100004
0082  00065 076000 OPMSK   OCT 76000
0083  00066 077777 INF     OCT 77777
0084  00067 100037 TYPFL   OCT 100037
0085  00070 101777 OPDMK   OCT 101777
0086  00071 000105 E       OCT 105
0087  00072 000106 F       OCT 106
0088  00073 000123 S       OCT 123
0089  00032        B17     EQU .15
0090  00033        B20     EQU .16
0091  00074 000022 B22     OCT 22
0092  00075 000023 B23     OCT 23
0093  00035        B26     EQU .22
0094  00036        B27     EQU .23
0095  00076 000036 B36     OCT 36
0096  00040        B37     EQU .31
0097  00043        B42     EQU .34
0098  00077 000044 B44     OCT 44
0099  00100 000052 B52     OCT 52
0100  00053        B72     EQU .58
0101  00101 000100 B100    OCT 100
0102  00102 000133 B133    OCT 133
0103  00103 000140 B140    OCT 140
0104  00104 000177 B177    OCT 177
0105  00105 000377 B377    OCT 377
0106  00013        B400    EQU C7
0107  00106 000777 B777    OCT 777
0108  00107 001000 B1000   OCT 1000
0109  00110 001400 B1400   OCT 1400
0110  00111 001740 B1740   OCT 1740
0111  00112 001760 B1760   OCT 1760
0112  00113 001774 B1774   OCT 1774
0113  00114 001775 B1775   OCT 1775
0114  00115 001777 B1777   OCT 1777
0115  00116 002000 QTOP    OCT 2000
0116  00117 002400 B2400   OCT 2400
0117  00120 003000 B3000   OCT 3000
0118  00121 004000 B4000   OCT 4000
0119  00122 006000 B6K     OCT 6000
0120  00004        B10K    EQU C6+1
0121  00004        RPOP    EQU B10K
0122  00123 012000 B12K    OCT 12000
0123  00123        RBOP    EQU B12K
0124  00124 032000 EXPOP   OCT 32000
0125  00125 050000 LROP    OCT 50000
0126  00126 052000 LPOP    OCT 52000
0127  00127 170000 B170K   OCT 170000
0128  00130 013777 FROMA   OCT 13777
0129  00131 000000 LROMA   HSS 1
0130  00132 177777 M1      DEC -1
0131  00133 177776 M2      DEC -2
0132  00134 177775 M3      DEC -3
0133  00135 177774 M4      DEC -4
0134  00136 177773 M5      DEC -5
0135  00137 177772 M6      DEC -6
0136  00140 177767 M9      DEC -9
0137  00141 177766 M10     DEC -10
0138  00142 177765 M11     DEC -11
0139  00143 177764 M12     DEC -12
0140  00144 177763 M13     DEC -13
0141  00145 177762 M14     DEC -14
0142  00146 177761 M15     DEC -15
0143  00147 177751 M23     DEC -23
0144  00150 177747 M25     DEC -25
0145  00151 177740 M32     DEC -32
0146  00152 177734 M36     DEC -36
0147  00153 177720 M48     DEC -48
```

```
          #01  9830

#148  00154 177700 M64    DEC -64
#149  00155 177660 M80    DEC -80
#150  00156 177634 M100   DEC -100
#151  00157 177400 M256   DEC -256
#152  00160 176000 M1024  DEC -1024
#153  00161 154360 MAXSN  DEC -10000     -(MAXIMUM ALLOWABLE LINE NUMBER)
#154  00162 043116 FN     ASC 1,FN
#155*
#156*  MATH CONSTANTS THAT ARE NEEDED ON BASE PAGE
#157*
#158  00163 177401 ZAP    OCT 177401
#159  00164 063440 MASK1  OCT 063440
#160  00165 061400 MASK5  OCT 061400
#162*
#163  00166 000000 FWAM   BSS 1           FIRST WORD OF AVAILABLE MEM
#164  00112        RSTKA  EQU B1760       START OF RETURN STACK
#165  00167 000000 WRITE  BSS 1           PRINTER AND DISPLAY DRIVER
#166  00170 000000 CLERA  BSS 1           CLEAR SCREEN
#167  00116        REED   EQU UTOP        KEYBOARD DRIVER UTOP IS = 20000
#168  00121        START  EQU B4000
#169  00171 000000 RUNA   BSS 1
#170  00172 000000 XEC4A  BSS 1
#171  00173 000000 PEXMA  BSS 1
#172  00174 000000 RDYDA  BSS 1
#173  00122        LISTA  EQU B6K
#174  00175 000000 TSRHA  BSS 1           TABLE SEARCH
#175  00176 000000 ELINA  BSS 1           OUTPUT I/O BUFFER
#176  00177 000000 SSYMA  BSS 1           SEARCH SYMBOL TABLE FOR SYMBOL
#177  00200 000000 OPCHA  BSS 1
#178  00201 000000 FORMA  BSS 1
#179  00202 000000 FSCA   BSS 1
#180  00203 000000 ACTSA  BSS 1
#181  00204 000000 DELSA  BSS 1
#182  00205 000000 AERRA  BSS 1           ALTERNATE ENTRY FOR ERROR
#183  00206 000000 ERRA   BSS 1
#184  00207 014560 INTEA  DEF ISERV
#185  00210 000000 OUTIA  BSS 1           OUTPUT AN INTEGER
#186  00211 000000 .BUFA  BSS 1           FIRST CHAR OF I/O BUFFER
#187  00212 000000 .SLBF  BSS 1            REFRESH BUFFER POINTER
#188  00213 001744 AR1A   DEF AR1
#189  00214 001754 AR2A   DEF AR2
#190  00215 000000 TMPOP  BSS 1
#191  00216 000000 .FADA  BSS 1           ADD
#192  00217 000000 .FSBA  BSS 1           SUBTRACT
#193  00220 000000 .FMPA  BSS 1           MULTPY
#194  00221 000000 .FDVA  BSS 1           DIVIDE
#195  00222 000000 FNTFA  BSS 1
#196  00223 000000 .FSR   BSS 1           SQUARE ROOT
#197  00224 000000 UPARA  BSS 1           UP ARROW
#198  00225 000000 FLTRA  BSS 1
#199  00226 000000 FLTFA  BSS 1
#200  00227 000000 FXFLA  BSS 1
#201  00230 000000 FLTSA  BSS 1
#202  00231 000000 SPLFA  BSS 1
#203  00232 000000 SNFLA  BSS 1
#204  00233 000000 BEND   BSS 1           I/O BUFFER END
#205  00233        SBUFA  EQU BEND
#206  00234 000000 NMOTA  BSS 1           NUMBER OUTPUT ROUTINE LINK
#207  00235 000000 FILLA  BSS 1             SUPPLY NEEDED BLANKS ON OUTPUT
#208  00236 000000 SLEND  BSS 1           REFRESH BUFFER END
#209  00111        TMPA   EQU B1740
#210  00110        BADRP  EQU B1400       POINTER POINTER TO I/O BUFFER
#211*
#212  00123        FOPRS  EQU B12K
#213  00237 000000 SALTA  BSS 1
#214  00240 000000 MDIMA  BSS 1
#215  00241 000000 SYTSA  BSS 1
#216  00242 000000 DCOMA  BSS 1
#217  00243 000000 NUMCA  BSS 1
#218  00244 000000 NUMOA  BSS 1
#219  00245 000000 GTSTA  BSS 1
#220  00246 000000 VAROA  BSS 1
#221  00247 000000 ECALA  BSS 1
#222  00250 000000 OVCHA  BSS 1
#223  00251 000000 CLPIA  BSS 1
#224  00252 000000 ENTEA  BSS 1
#225  00253 000000 CHRSA  BSS 1
#226  00254 000000 XSCRA  BSS 1
#227  00255 000000 XEC6A  BSS 1
#228  00256 000000 FDAAA  BSS 1
#229  00257 000000 FNDAA  BSS 1
#230  00260 000000 FNDIA  BSS 1
#231  00261 000000 IOSTA  BSS 1
#232  00262 000000 INREA  BSS 1
#233  00263 000000 SUMPA  BSS 1
#234  00264 000000 CRCKA  BSS 1
#235  00265 000000 IOHEA  BSS 1
#236  00266 000000 QTCHA  BSS 1
#237*********************************
#238*
#239  00267 000000        BSS 1          CHECKSUM WORD*****************
```

```
          #01  9830--ERROR TABLE

0241  00270 000000 F1     HSS 1
0242  00271 000000        HSS 1
0243  00272 000000        HSS 1
0244  00273 000000        HSS 1
0245  00274 000000 F5     HSS 1
0246  00275 000000        HSS 1
0247  00276 000000        HSS 1
0248  00277 000000        HSS 1
0249  00300 000000        HSS 1
0250  00301 000000 E10    HSS 1
0251  00302 000000        HSS 1
0252  00303 000000        HSS 1
0253  00304 000000        HSS 1
0254  00305 000000        HSS 1
0255  00306 000000 E15    HSS 1
0256  00307 000000        HSS 1
0257  00310 000000        HSS 1
0258  00311 000000        HSS 1
0259  00312 000000        HSS 1
0260  00313 000000 E20    HSS 1
0261  00314 000000        HSS 1
0262  00315 000000        HSS 1
0263  00316 000000        HSS 1
0264  00317 000000        HSS 1
0265  00320 000000 E25    HSS 1
0266  00321 000000        HSS 1
0267  00322 000000        HSS 1
0268  00323 000000        HSS 1
0269  00324 000000        HSS 1
0270  00325 000000 E30    HSS 1
0271  00326 000000        HSS 1
0272  00327 000000        HSS 1
0273  00330 000000        HSS 1
0274  00331 000000        HSS 1
0275  00332 000000 E35    HSS 1
0276  00333 000000        HSS 1
0277  00334 000000        HSS 1
0278  00335 000000        HSS 1
0279  00336 000000        HSS 1
0280  00337 000000 E40    HSS 1
0281  00340 000000        HSS 1
0282  00341 000000        HSS 1
0283  00342 000000        HSS 1
0284  00343 000000        HSS 1
0285  00344 000000 E45    HSS 1
0286  00345 000000        HSS 1
0287  00346 000000        HSS 1
0288  00347 000000        HSS 1
0289  00350 000000        HSS 1
0290  00351 000000 E50    HSS 1
0291  00352 000000        HSS 1
0292  00353 000000        HSS 1
0293  00354 000000        HSS 1
0294  00355 000000        HSS 1

          #01  9830--BASE PAGE READ-WRITE MEMORY

0296  01400               ORG 1400B
0297  01400 000000 BADDR  HSS 1     CURRENT CHAR POINTER
0298  01401 000000 RSAVE  HSS 1          SAVE (A) FOR RETN?
0299  01402 000000 OTPTR  HSS 1     REFRESH POINTER
0300  01403 000000 MAW    HSS 1     LAST WORD OF R/W MEM
0301  01404 000000 LWAM   HSS 1     LAST WORD AVAILABLE FOR USER
0302  01405 000000 FWUP   HSS 1     FIRST WORD OF COMMON OR MAINLINE
0303  01406 000000 VALUE  HSS 1     FIRST WORD -1 OF VALUE TABLE
0304  01407 000000 SYMTP  HSS 1     FIRST WORD -1 OF SYMBOL TABLE
0305  01410 000000 LSTAK  HSS 1     FIRST WORD OF OPERATOR STACK
0306  01411 000000 TSTPT  HSS 1     TEMPORARY STACK POINTER
0307  01412 000000 LSTPT  HSS 1     OPERATOR STACK POINTER
0308  01413 000000 HSTPT  HSS 1     OPERAND STACK POINTER
0309  01414 000000 DSTRT  HSS 1     DATA STATEMENT
0310  01415 000000 NXTDT  HSS 1       POINTERS
0311  01416 000000 MODE   HSS 1     PROGRAM=0; KEYBOARD=1
0312  01417 000000 PARMA  HSS 1     PARAMETER ADDRESS
0313  01420 000000 PARMN  HSS 1     PARAMETER NAME
0314  01421 000000 PRADD  HSS 1     LAST WORD+1 OF CURRENT STMT
0315  01422 000000 UKPTR  HSS 1     USER KEY POINTER
0316  01423 000000 PAPTR  HSS 1     LAST WORD OF CURRENT PROGRAM
0317  01424 000000 PAUFF  HSS 1     FIRST WORD OF CURRENT PROGRAM
0318  01425 000000 PAUFO  HSS 1     FIRST WORD OF MAINLINE
0319  01426 000000 PAPTO  HSS 1     LAST WORD+1 OF MAINLINE
0320  01427 000000 .LNUM  HSS 1     CURRENT LINE NUMBER
0321  01430 000000 LNINC  HSS 1     AUTO LINE INCREMENT
0322  01431 000000 CLINE  HSS 1     CURRENT LINE FOR LIST
0323  01432 000000 DRQST  HSS 1     INPUT STMT FLAG
0324  01433 000000 TFLAG  HSS 1     PTAPE FLAG
0325  01434 000000 UNITS  HSS 1     TRIG UNITS
0326  01435 000000 SEC    HSS 1     SECURE PROGRAM FLAG
```

#01 9830--BASE PAGE READ-WRITE MEMORY

```
0327  01436 000000 HFLG1 BSS 2     STOP ADDRESSES
0328  01440 000000 TFLG1 BSS 2     TRACE LIMIT ADDRESSES
0329  01442 000000 STPFL BSS 1     STOP FLAG
0330  01443 000000 KEYFG BSS 1     KEYBOARD FLAG
0331  01444 000000 SAVEA BSS 1     INTERRUPT
0332  01445 000000 SAVEH BSS 1
0333  01446 000000 ADRSS BSS 1       TEMPORARIES
0334  01447 000000 SLCOD BSS 1     SELECT CODE
0335  01450 000000 PMODF BSS 1     PRINT-ALL FLAG
0336  01451 000000 CTYPF BSS 1     KEYBOARD
0337  01452 000000 CPLAC BSS 1       FORMAT FLAGS
0338  01453 000000 PRFLG BSS 1     PRINT/DISP FLAG
0339  01454 000000 RCOUT BSS 1     FORMAT
0340  01455 000000 FORMT BSS 1
0341  01456 000000 FORMS BSS 1
0342  01457 000000 FORME BSS 1       STATEMENT FLAGS
0343  01460 000000 FTMP1 BSS 1     CURRENT OPERATOR CODE
0344  01461 000000 FTMP2 BSS 1     CURRENT OPERATOR PRECEDENCE
0345  01462 000000 REEDL BSS 1
0346  01463 000000 INTEL BSS 1
0347  01464 000000       BSS 2       UNUSED
0348*
0349*  TEMPORARIES
0350*
0351  01466 000000 LOCL1 BSS 1
0352  01467 000000 LOCL2 BSS 1
0353  01470 000000 L2T1  BSS 1
0354  01471 000000 L2T2  BSS 1
0355  01472 000000 TEMPS BSS 1
0356  01473 000000 TEMP  BSS 1
0357  01473       MROX1 EQU TEMP
0358  01474 000000 TEMP1 BSS 1
0359  01475 000000 TEMP2 BSS 1
0360  01476 000000 TEMP3 BSS 1
0361  01477 000000 TEMP4 BSS 1
0362*
0363*  BUFFERS
0364*
0365  01500 000000 IOBUF BSS 41          INPUT AND RECALL BUFFER
0366  01551 000000 RBUFF BSS 41          SINGLE LINE REFRESH BUFFER
0367*
0368*  PRINT ROUTINE FLAGS
0369*
0370  01622 000000 TERM  BSS 1     CRLF FLAG
0371  01623 000000 PEND  BSS 1     BUFFER OVERFLOW FLAG
0372  01624 000000 PPFF  BSS 1     BUFFER POINTER OR TAB FLAG
0373  01625 000000 FIRST BSS 1
0374  01626 000000       BSS 2      UNUSED
0375*
0376*  MATH TEMPORARIES
0377*
0378  01630 000000 TREGE BSS 4
0379  01634 000000 RSLTE BSS 4
0380*
0381*  STATEMENT SYNTAX AND EXECUTION TEMPORARIES
0382*
0383  01640 000000 ST    BSS 32
0384*
0385*  SYSTEM SYNTAX AND EXECUTION TEMPORARIES
0386*
0387  01700 000000       BSS 1           SAVED FOR PROCESSOR
0388  01701 000000 NDISP BSS 1
0389  01702 000000 NPRIN BSS 1
0390  01703 000000 PLACE BSS 1
0391  01704 000000 WIDTH BSS 1
0392  01705 000000 FLOAT BSS 1
0393  01706 000000 SRPTR BSS 1
0394  01707 000000 BLANK BSS 1
0395  01710 000000 MSFLG BSS 1
0396  01711 000000 DFLAG BSS 1
0397  01712 000000 PFLAG BSS 1
0398  01713 000000 UFLAG BSS 1
0399  01714 000000 SFLAG BSS 1
0400  01715 000000 SRADR BSS 1
0401  01716 000000 SSBPT BSS 1
0402  01717 000000 NUM   BSS 1
0403  01717       OPTYP EQU NUM
0404  01720 000000 LASTN BSS 1
0405  01721 000000 SIGN  BSS 1
0406  01721       MAXIN EQU SIGN
0407  01722 000000 EXP   BSS 1
0408  01722       INTGR EQU EXP
0409  01723 000000 ARYAD BSS 1
0410  01724 000000 GFLAG BSS 1
0411  01725 000000 NXTST BSS 1
0412  01726 000000 MRIN1 BSS 1
0413  01727 000000 MRIN2 BSS 1
0414*
0415*  ARITHMETIC REGISTERS AND RETURN STACK
0416*
```

#01 9930--BASE PAGE READ-WRITE MEMORY

```
0417  01740              ORG 1740B
0418  01740 000000 XC    BSS 4
0419  01744 000000 AR1   BSS 4
0420  01750 000000 YC    BSS 4
0421  01754 000000 AR2   BSS 4
0422  01760 000000 RSTAK BSS 11      RETURN STACK
0423  01773 000000 RFS   BSS 4       RESULT REGISTER
0424  01777 000000 RSPTR BSS 1    RETURN STACK POINTER
0425  00356        RETN3 EQU 356B
0426  00357        RETN2 EQU 357B
0427  00365        RET2  EQU 365B
0428  00373        ICNT  EQU 373B
0429  00374        WBUFF EQU 374B
0430  00400        GPARM EQU 400B
0431  00405        XFAR1 EQU 405B
0432  00412        RSULT EQU 412B
0433  00416        STORE EQU 416B
0434  00425        STOR1 EQU 425B
0435  00444        XFAR2 EQU 444B
0436  00450        STROP EQU 450B
0437  00456        SRPUD EQU 456B
0438  00461        SYMC2 EQU 461B
0439  00463        SYMC1 EQU 463B
0440  00464        SYMCK EQU 464B
0441  00504        COMCK EQU 504B
0442  00510        COMCF EQU 510B
0443  00513        LPCK  EQU 513B
0444  00521        LPCKF EQU 521B
0445  00524        RPCK  EQU 524B
0446  00534        RPCKE EQU 534B
0447  00535        FSCF2 EQU 535B
0448  00537        STSRH EQU 537B
0449  00546        STEXP EQU 546B
0450  00553        BHSTP EQU 553B
0451  00561        MERR  EQU 561B
0452  00561        FSCF4 EQU 561B
0453  00563        SLWST EQU 563B
0454  00567        UNSTK EQU 567B
0455  00574        NEXTL EQU 574B
0456  00606        NEXT1 EQU 606B
0457  00613        FPAGE EQU 613B
0458  00626        SAVRP EQU 626B
0459  00633        RESR1 EQU 633B
0460  00640        RESRP EQU 640B
0461  00645        MINIT EQU 645B
0462  00651        INITS EQU 651B
0463  00664        LNUM  EQU 664B
0464  00666        PGINT EQU 666B
0465  00676        INTCK EQU 676B
0466  00726        SYE25 EQU 726B
0467  00730        GETAD EQU 730B
0468  00742        FND   EQU 742B
0469  00750        DIGCK EQU 750B
0470  00752        LFTCK EQU 752B
0471  00753        LIMIT EQU 753B
0472  00771        GTOPP EQU 771B
0473  01007        NEXTA EQU 1007B
0474  01015        ORLNK EQU 1015B
0475  01016        ONEXT EQU 1016B
0476  01017        OUTCR EQU 1017B
0477  01041        GETCH EQU 1041B
0478  01053        GTMP  EQU 1053B
0479  01055        GNEXT EQU 1055B
0480  01062        GFRST EQU 1062B
0481  01066        GETSK EQU 1066B
0482  01075        FNDPS EQU 1075B
0483  01121        MVTOH EQU 1121B
0484  01131        PGIN0 EQU 1131B
0485  01140        LTR   EQU 1140B
0486  01147        ENDS  EQU 1147B
0487  01151        EOST  EQU 1151B
0488  01152        NOEOF EQU 1152B
0489  01153        EOLCK EQU 1153B
0490  01156        CONST EQU 1156B
0491  01165        SYE12 EQU 1165B
0492  01166        GSIGN EQU 1166B
0493  01201        IMPY  EQU 1201B
0494  01217        FETCH EQU 1217B
0495  01225        LDEF  EQU 1225B
0496  01254        ZONK  EQU 1254B
0497  01256        CLAR2 EQU 1256B
0498  01314        ODGIT EQU 1314B
0499  01317        SGNS  EQU 1317B
0500  01330        .NPWR EQU 1330B
0501  01335        NPWRT EQU 1335B
0502  01337        NDWRT EQU 1337B
0503  01342        CRLF  EQU 1342B
0504  01344        CRLF1 EQU 1344B
0505  01351        TTY   EQU 1351B
0506  01362        GTCON EQU 1362B
0507  01371        FIX   EQU 1371B
```

```
        #01  9830A TAPE CASSETTE SYSTEM

0510                    LST
0511*             CTAPE  3-20-72-9:30
0512  14000             ORG 14000B
0513*
0514*  SYSTEM COMMAND TABLE
0515*
0516  14000 000003      ABS ESEC-*
0517  14001 051505 CMNDT ASC 1,SE       SECURE, OPCODE 1
0518  14002 041761      OCT 41761       C, OPCODE
0519  14003 060730 ESEC  JSM GETAD      A=LNX1,B=LNX2
0520  14004 035647      STB TAP1        TEMPOR, LNX2 ADDR
0521  14005 025477      LDB TEMP4       LNX1 ADDR
0522  14006 015647 ESEC3 CPB TAP1        DONE ?
0523  14007 066016      JMP ESEC6
0524  14010 074517      LDA 5,I
0525  14011 070410      SZA ESEC4       SKIP IF TYPING AID
0526  14012 072053      SAP *+1,S       SET BIT 15
0527  14013 074557      STA 8,I         RESTORE LINE
0528  14014 061007      JSM NEXTA       NO, GET NEXT LINE
0529  14015 066006      JMP ESEC3
0530  14016 021435 ESEC6 LDA SECFL
0531  14017 040062      IOR FLGAT       SET BIT 15
0532  14020 031435      STA SECFL       CUSTOMER SECURING PROG
0533  14021 164174 ESEC4 JMP RDYDA,I      ET TO SYSTEM
0534*
0535*
0536  14022 000000      HSS 1             ********** C.S.**********
0537*
0538*
0539*     SUB-TABLE FOR STORE AND LOAD
0540*
0541*
0542  14023 041111 SUBT  ASC 1,RI        BIN OPCODE 24
0543  14024 047264       OCT 47264       N, OPCODE
0544  14025 045505       ASC 1,KE        KEY, OPCODE 20
0545  14026 054660       OCT 54660       Y, OPCODE
0546  14027 042101       ASC 2,DATA      DATA, OPCODE 21
      14030 052101
0547  14031 170400       OCT 170400      OPCODE, EOT
0548*
0549*
0550*    SYNTAX SERVICE SUBR
0551*
0552*
0553  14032 061055 SERV  JSM GNEXT
0554  14033 060456       JSM SBPUD
0555  14034 010031       CPA EOL         EOL ?
0556  14035 170402       RET
0557  14036 055706       DSZ SBPTR
0558  14037 055400       DSZ BADDR
0559  14040 160202       JSM FSCA,I      GEET NO. OR EXPRESSION
0560  14041 064357       JMP RETN2
0561*
0562*
0563*
0564*
0565*                    EXECUTION
0566*
0567*
0568*
0569*
0570*   ALOAD - ALL LOAD EXECT ENTRY
0571*
0572*
0573  14042 060771 ALOAD JSM GTOPP      GET OPERAND
0574  14043 072510       RZA LOADP       SKIP IF OPERAND GIVEN
0575  14044 045472       ISZ TEMPS
0576  14045 121472       LDA TEMPS,I
0577  14046 013732       CPA DATOP       DATA OPCODE?
0578  14047 067172       JMP LDATA       YES
0579  14050 013733       CPA KEYOP       KEY OPCODE?
0580  14051 067315       JMP LKEY        YES
0581  14052 013734       CPA BINOP       BINARY OPCODE?
0582  14053 067421       JMP LBIN        YES
0583  14054 055472       DSZ TEMPS
0584*
0585*
0586*  LOADP - LOAD PROGRAM
0587*
0588*
0589  14055 062070 LOADP JSM LOAD2
0590  14056 062102       JSM LOADR
0591  14057 060645       JSM MINIT       SRATCH SYMBOL TABLE
0592  14060 021660       LDA TAP7
0593  14061 062160       JSM LOAD3
0594  14062 021416 LOAD9 LDA MODE
0595  14063 070150       SZA *+3
0596  14064 060651       JSM INITS       RESET STACKS
0597  14065 164174       JMP RDYDA,I     KEYB-RET TOSYSTEM-DO NOT EXECUTE
0598  14066 021661       LDA TAPA        SEC LINE #
```

#01 9830A TAPE CASSETTE SYSTEM

```
0599  14067 164171        JMP RUNA,I
0600*
0601*
0602*
0603  14070 062300 LOAD2  JSM SFLOC
0604  14071 027745        LDB TAPAD       TEMPOR. ADDR
0605  14072 021472 LO3    LDA TEMPS       STATEMENT POINTER
0606  14073 011421        CPA PRADD       EOL FOLLOWS?
0607  14074 170402        RET             YES. PROCESS
0608  14075 045472        ISZ TEMPS       POINT TO LINE #
0609  14076 121472        LDA TEMPS,I     LINE #
0610  14077 074557        STA B,I         TAP7-TAP8. LINE #S
0611  14100 045472        ISZ TEMPS
0612  14101 077470        RIB LO3         LOOK FOR SECOND LINE #
0613*
0614*
0615*
0616  14102 021661 LOADR  LDA TAP8
0617  14103 070110        SZA *+2
0618  14104 066126        JMP LOAD7-2       LNX2 GIVEN
0619  14105 021416        LDA MODE
0620  14106 070110        SZA *+2         SKIP ON PROGRAM
0621  14107 066130        JMP LOAD7       DO NOT EXECUTE
0622  14110 121421        LDA PRADD,I     GET NEXT LINE #
0623  14111 070072        SAM *+1,C        CLEAR SEC BIT
0624  14112 025421        LDB PRADD
0625  14113 015423        CPB PBPTR
0626  14114 020056        LDA .1E4          10000
0627  14115 070076        TCA
0628  14116 001660        ADA TAP7        LNX1 - LINE # FOR START LOAD
0629  14117 070213        SAP *+4         NEXT LINE OK
0630  14120 021660        LDA TAP7         START AT FIRST LINE OF LOADED PROG.
0631  14121 031661        STA TAP8        FOR START OF EXECUTION
0632  14122 066126        JMP *+4
0633  14123 021427        LDA .LNUM        PRESENT L.N.
0634  14124 070070        SIA *+1
0635  14125 031661        STA TAP8         START AT NEXT N+N+LINE
0636  14126 070742        SAR 16
0637  14127 031416        STA MODE         EXECUTE
0638  14130 025421 LOAD7  LDB PRADD
0639  14131 163756        JSM DCM1A,I       DECOMPILE
0640  14132 021666        LDA IDB0+2      FILE TYPE
0641  14133 010020        CPA .3          SOURCE?
0642  14134 066142        JMP *+6
0643  14135 010036        CPA .23        SEC SOURCE ?
0644  14136 066140        JMP *+2          YES
0645  14137 067325        JMP ERR7        TYPE ERROR
0646  14140 024106        LDB B777
0647  14141 035435        STB SECFL       SET SECURE FLAG
0648  14142 021424        LDA PBUFF
0649  14143 011423        CPA PBPTR       NULL PROG?
0650  14144 066151        JMP LO12         YES
0651  14145 121424        LDA PBUFF,I     TYPING AID RESIDENT?
0652  14146 072150        RZA LO12
0653  14147 160205 ERR8   JSM AERRA,I     OVERLAY ERROR
0654  14150 177701        DEC -63
0655  14151 025660 LO12   LDB TAP7
0656  14152 074210        SZB *+4        START LINE SPECIFIED ?
0657  14153 025670        LDB IDB0+4      YES. COMPUTE DIFF
0658  14154 074076        TCB             BETWEEN SPECIFIED
0659  14155 005660        ADB TAP7        AND ACTUAL LINE #
0660  14156 035663        STB OFSET       TEMPOR
0661  14157 172402        RET
0662*
0663*
0664*
0665  14160 061075 LOAD3  JSM FNDPS       FIND ADDR OF START LINE
0666  14161 066271        JMP LO6       L.N. BEYOND END OF PROGR
0667  14162 066163        JMP *+1        L.N. BETWEEN PROGRAM LINES
0668  14163 035650        STB TAP2    L. N.    EXACTLY FOUND - STORE START ADDR
0669  14164 035477        STB TEMP4      SAVE FOR CLP1A
0670  14165 025423        LDB PBPTR
0671  14166 160251        JSM CLP1A,I       DEL FROM HERE TO END OF PROGR
0672*
0673*
0674*
0675  14167 021671 LOAD4  LDA IDB0+5     END LINE #
0676  14170 001663        ADA OFSET
0677  14171 000012        ADA .1
0678  14172 063742        JSM FND       FIND ADDR OF END LINE
0679  14173 015650        CPB TAP2      SAVE AS START ADDR
0680  14174 066176        JMP *+2        YES
0681  14175 066147        JMP ERR8       NO. OVERLAY
0682  14176 035660        STB TAP7       TEMPOR
0683*
0684*    CHECK FOR LEGAL COMMON AND OVERFLOW
0685*            UPDATE POINTERS
0686*
0687  14177 021672 COMON  LDA IDB0+6     LOADING COMMON?
0688  14200 072210        RZA YCOMN
0689  14201 021665        LDA IDB0+1     NO COMMON
```

```
#01  9830A TAPE CASSETTE SYSTEM

0690  14202 160250        JSM OVCHA,I        UPDATE PROGRAM POINTERS
0691  14203 066231        JMP KCOMN
0692  14204 025660 YCOMN LDB TAP7       YES
0693  14205 015424        CPB PBUFF      LOADING AT FIRST LINE?
0694  14206 066210        JMP *+2        YES
0695  14207 164330        JMP E30+3,I      NO. MISPLACED COM
0696  14210 025422        LDB UKPTR
0697  14211 076350        RZB COM1
0698  14212 025405        LDB FWUP
0699  14213 035424        STB PBUFF
0700  14214 035426        STB PBPT0
0701  14215 035423        STB PBPTR
0702  14216 035476        STB TEMP3
0703  14217 066221        JMP *+2
0704  14220 070742 COM1   SAR 16         DO NOT ALLOCATE SPACE FOR COMMON
0705  14221 001665        ADA IDB0+1     ADD PROGR SIZE TO COMMON SIZE
0706  14222 160250        JSM OVCHA,I    CHECK OVERFLOW AND UPDATE POINTERS
0707  14223 021422        LDA UKPTR
0708  14224 072250        RZA KCOMN      SKIP IF KEY
0709  14225 021424        LDA PBUFF
0710  14226 001672        ADA IDB0+6     UPDATE PBUFF
0711  14227 031424        STA PBUFF
0712  14230 031660        STA TAP7       UPDATE LOAD ADDR
0713  14231 061121 KCOMN  JSM MVTOH      OPEN UP SPACE
0714*
0715*    LOAD THE FILE
0716*
0717  14232 063105        JSM RRRID       REV READ HOF; READ IMMDT FILE ID
0718  14233 021660        LDA TAP7        START ADDR
0719  14234 025665        LDB IDB0+1     CURR FILE SIZE
0720  14235 070037        ADB A          END ADDR
0721  14236 062717        JSM IRD        READ FILE
0722  14237 063031        JSM HALT
0723  14240 070115        SEC *+2
0724  14241 067100        JMP ERR4       CHECK SUM ERROR
0725*
0726*    ADD OFFSET TO ALL LINE NUMBER REFERENCES
0727*
0728  14242 021423        LDA PBPTR
0729  14243 031647        STA TAP1       SAVE END OF PROG POINTER
0730  14244 021654        LDA TAP5
0731  14245 031423        STA PBPTR      SET TO END OF LOADED AREA
0732  14246 021653        LDA TAP4       POINTER TO START OF LOADED AREA
0733  14247 160257        JSM FNDAA,I    FIND FIRST LINE /# REF
0734  14250 066256        JMP L04
0735  14251 001663        ADA OFSET      ADD OFFSET
0736  14252 131472        STA TEMPS,I    RESTORE
0737  14253 160260        JSM FND1A,I    FIND NEXT REF
0738  14254 066256        JMP L04        END OF LOADED AREA
0739  14255 066251        JMP *-4
0740*
0741*
0742*    ADD OFFSET TO ALL LINE NUMBERS LOADED
0743*
0744  14256 025653 L04    LDB TAP4       POINT TO START OF LOADED AREA
0745  14257 015423        CPB PBPTR      END OF LOADED AREA?
0746  14260 066266        JMP L05        YES
0747  14261 074517        LDA B,I        GET LOADED LINE NUMBER
0748  14262 001663        ADA OFSET      ADD OFFSET
0749  14263 074557        STA B,I        STORE NEW LINE #
0750  14264 061007        JSM NEXTA      COMPUTE ADDR OF NEXT LINE
0751  14265 066257        JMP L04+1
0752  14266 025647 L05    LDB TAP1
0753  14267 035423        STB PBPTR      RESTORE END OF PROGR POINTER
0754  14270 170402        RET
0755*
0756*    APPEND
0757*
0758  14271 035660 L06    STB TAP7        START ADDR
0759  14272 025671        LDB IDB0+5        END LINE #
0760  14273 005643        ADB OFSET
0761  14274 004161        ADB MAXSN      FINAL LINE #
0762  14275 074112        SBM *+2          EXCEED 9999?
0763  14276 164273        JMP E5-1,I     YES
0764  14277 066177        JMP COMON
0765*
0766*
0767*     SUBR FOR FILE LOCATING
0768*
0769*
0770  14300 062457 SFLOC JSM UNITC
0771  14301 021472        LDA TEMPS
0772  14302 011421        CPA PRADD      END OF STATEMENT?
0773  14303 066312        JMP SF2        YES. PROCESS
0774  14304 062510        JSM FILEC      NO
0775  14305 031644        STA TFILE
0776  14306 074742        SBR 16
0777  14307 035660        STB TAP7
0778  14310 035661        STB TAP8
0779  14311 067123        JMP LOCF2        LOC  FILE AND RET
0780  14312 074742 SF2    SBR 16
```

#01 9830A TAPE CASSETTE SYSTEM

```
0781  14313 035660        STB TAP7
0782  14314 035661        STB TAP8
0783  14315 035644        STB TFILE       FILE 0
0784  14316 067123        JMP LOCF2
0785*
0786*
0787*
0788*
0789*
0790*  ALSTOR - ALL STORE EXECUTION ENTRY
0791*
0792*
0793  14317 060771 ASTOR JSM GTOPP      GET OPERAND
0794  14320 072510        RZA STORP     SKIP IF OPERAND GIVEN
0795  14321 045472        ISZ TEMPS
0796  14322 121472        LDA TEMPS,I
0797  14323 013732        CPA DATOP     DATA OPCODE?
0798  14324 067216        JMP SDATA     YES
0799  14325 013733        CPA KEYOP     KEY OPCODE?
0800  14326 067403        JMP SKEY      YES
0801  14327 013734        CPA BINOP     BINARY OPCODE?
0802  14330 167755        JMP ERR0,I
0803  14331 055472        DSZ TEMPS
0804*
0805*
0806*    STORP - STORE PROGRAM
0807*
0808*
0809  14332 062300 STORP JSM SFLOC
0810  14333 062071        JSM LO3-1
0811  14334 063025        JSM WPERM
0812  14335 025421        LDB PRADD
0813  14336 163756        JSM DCM1A,I       DECOMPILE
0814  14337 121424        LDA PBUFF,I
0815  14340 070072        SAM **1,C         ASSUME FIRST LINE #
0816  14341 025660        LDB TAP7          FIRST LINE #
0817  14342 074110        SZB *+2
0818  14343 021660        LDA TAP7
0819  14344 025661        LDB TAP8        SECOND LINE #
0820  14345 076110        RZB *+2         SKIP IF GIVEN
0821  14346 024056        LDB .1E4        ASSUME LAST LINE IN PROGRAM
0822  14347 060730        JSM GETAD       GET ADDR LIMITS
0823  14350 015477        CPB TEMP4
0824  14351 067242        JMP ERR5          NULL PROGRAM
0825  14352 121477        LDA TEMP4,I
0826  14353 070072        SAM **1,C
0827  14354 031670        STA IDB0+4          IN FILE ID
0828  14355 021477        LDA TEMP4       START OF STORE AREA
0829  14356 031472        STA TEMPS       LOW LIMIT
0830  14357 031650        STA TAP2        LOW LIMIT
0831  14360 035660        STB TAP7        HIGH LIMIT
0832  14361 060602        JSM NEXTI-4
0833  14362 001466        ADA LOCL1
0834  14363 013746        CPA TCOM        IS IT COMMON ?
0835  14364 066366        JMP *+2         YES
0836  14365 066373        JMP *+6         NO
0837  14366 025405        LDB FWUP        START OF USER AREA
0838  14367 074076        TCB
0839  14370 005424        ADB PBUFF       FIND SIZE OF COMMON
0840  14371 035672        STB IDB0+6      IN FILE ID BUFF
0841  14372 066375        JMP *+3
0842  14373 074742        SBR 16
0843  14374 035672        STB IDB0+6      ZERO COMMON
0844  14375 025720        LDB LASTN       LAST LINE #
0845  14376 035671        STB IDB0+5      IN FILE ID BUFF
0846  14377 025660        LDB TAP7        TEMPR, END ADDR
0847  14400 021650        LDA TAP2        TEMPR, START ADDR
0848  14401 063237        JSM SDAT2
0849  14402 063463        JSM SECCK
0850  14403 024020        LDB .3        UNSEC SOURCE
0851  14404 070113        SAP *+2
0852  14405 024036        LDB .23       SEC SOURCE
0853  14406 035666        STB IDB0+2
0854  14407 067231        JMP SDAT3+2
0855*
0856*
0857*  MERGE EXECUTION
0858*
0859*
0860  14410 025472 MERGE LDB TEMPS
0861  14411 004133        ADB M2          ADDR OF MERGE LINE
0862  14412 074517        LDA B,I         MERGE LINE NO.
0863  14413 070072        SAM **1,C        CLEAR SEC BIT
0864  14414 031630        STA TREGE          TEMPOR
0865  14415 062070        JSM LOAD2
0866  14416 021661        LDA TAP8
0867  14417 072310        RZA LO7-2       LNX2 GIVEN
0868  14420 021416        LDA MODE
0869  14421 072310        RZA LO7         DO NOT EXECUTE
0870  14422 021630        LDA TREGE
```

```
        #01  9830A TAPE CASSETTE SYSTEM

0871  14423 000012        ADA .1          POINT TO NEXT LINE
0872  14424 031661        STA TAPA        LINE FOR START EXECUTION
0873  14425 070742        SAR 16
0874  14426 031416        STA MODE        EXECUTE
0875  14427 062130 LO7    JSM LOAD7
0876  14430 060645        JSM MINIT
0877  14431 021670        LDA IDB0+4      START L.N.
0878  14432 001663        ADA OFSET
0879  14433 061075        JSM FNDPS       FIND START ADDR
0880  14434 066442        JMP MERG1
0881  14435 066437        JMP *+2
0882  14436 066147        JMP ERRA        OVERLAY ERROR
0883  14437 035650        STB TAP2        SAVE ADDR
0884  14440 062167        JSM LOAD4
0885  14441 066062        JMP LOAD9
0886  14442 062271 MERG1  JSM L06
0887  14443 066062        JMP LOAD9
0888*
0889*
0890*       LINK  EXECUTION
0891*
0892*
0893  14444 062070 LINK   JSM LOAD2
0894  14445 062102        JSM LOADR
0895  14446 021660        LDA TAP7
0896  14447 062160        JSM LOAD3
0897  14450 021416        LDA MODE
0898  14451 070150        SZA *+3
0899  14452 060651        JSM INITS       RESET STACKS
0900  14453 164174        JMP ROYDA,I     KEYB-RET TO SYSTEM- DONOT EXECUTE
0901  14454 021661        LDA TAPA        SEC LINE #
0902  14455 031725        STA NXTST
0903  14456 164004        JMP XECA,I
0904*
0905*
0906*  UNIT NO CONFIGURATION
0907*
0908*
0909  14457 060771 UNITC  JSM GTOPP         GET OPERAND
0910  14460 072510        RZA FS          NO # PRESENT
0911  14461 045472        ISZ TEMPS
0912  14462 021472        LDA TEMPS
0913  14463 011421        CPA PRADD
0914  14464 066472        JMP FS
0915  14465 121472        LDA TEMPS,I
0916  14466 050065        AND OPMSK
0917  14467 013742        CPA NUMS        IS IT #?
0918  14470 066475        JMP GUNT        GET UNIT
0919  14471 055472        DSZ TEMPS
0920  14472 023735 FS     LDA B120K       ASSUME INTERNAL
0921  14473 031645        STA TUNIT       UNIT TEMPOR
0922  14474 170402        RET
0923  14475 061217 GUNT   JSM FETCH       EVALUATE FORMULA
0924  14476 160225        JSM FLTRA,I     CONVERT TO INTEGER
0925  14477 164342 UOVF   JMP E40+3,I
0926  14500 075752        SBM *-1
0927  14501 074117        LDA B         SAVE UNIT IN A REG
0928  14502 004142        ADB M11       UNIT<11 ?
0929  14503 075613        SBP UOVF
0930  14504 070204        SAL 12          POSITION UNIT NO.
0931  14505 072110        RZA *+2
0932  14506 066477        JMP UOVF
0933  14507 066473        JMP FS+1
0934*
0935*
0936*  FILE NO. CONFIGURATION
0937*
0938*
0939  14510 061217 FILEC  JSM FETCH         EVALUATE FORMULA
0940  14511 160225        JSM FLTRA,I     CONVERT TO INTEGER
0941  14512 066514        JMP *+2
0942  14513 074153        SBP *+3         RET ON POSITIVE
0943  14514 160205 ERR1   JSM AERRA,I     WRONG FILE NO.
0944  14515 177710        DEC -56
0945  14516 074117        LDA B
0946  14517 170402        RET             WITH FILE NO IN A REG
0947*
0948*
0949*  FINDE - FIND EXECUTION
0950*
0951*
0952  14520 062457 FINDE  JSM UNITC         GET UNIT NO.
0953  14521 070542        SAR 12          POSITUON UNIT
0954  14522 062554        JSM FITAS
0955  14523 000012        ADA .1        FIG. TABLE ADDR OF SERV SUBR
0956  14524 027744        LDB CASIN     CASST INTRPT SUBRT ADDR
0957  14525 070577        STB A,I
0958  14526 062510        JSM FILEC       GET FILE
0959  14527 031644        STA TFILE
0960  14530 063106 FIND3  JSM RIFS
```

#01 9830A TAPE CASSETTE SYSTEM

```
0961*
0962*
0963*
0964  14531 025664 DIRFC LDB FILEN      FILE NO.+1 FROM ID BUFF
0965  14532 074076       TCB           MAKE FILE NO READ NEG.
0966  14533 005644       ADB TFILE       FILE WANTED
0967  14534 004132       ADB M1        SUBTR. ONE TO STOP FRONT OF CORRECT FILE
0968  14535 062552       JSM FITA3
0969  14536 070577       STB A,I       STORE FILE DIFF IN STATUS
0970  14537 076210       RZB DI3
0971  14540 063031       JSM HALT       FILE LOCATED
0972  14541 173741 DI2   CLF 1         TURN ON INTERPT
0973  14542 164172       JMP XEC4A,I         BACK TO SYSTEM
0974  14543 074212 DI3   SBM *+4
0975  14544 023747       LDA RCFF      READ CONTRL FORW FAST,I
0976  14545 063033       JSM COUTP,1
0977  14546 066541       JMP DI2
0978  14547 023750       LDA RCRF      READ CONTRL REV FAST
0979  14550 063033       JSM COUTP,1
0980  14551 066541       JMP DI2
0981*
0982*   FITAS - FIG. INTRPT TABLE ADDR FOR STATUS
0983*
0984  14552 021645 FITA3 LDA TUNIT
0985  14553 070542       SAR 12
0986  14554 000132 FITAS ADA M1
0987  14555 070744       SAL 1
0988  14556 003743       ADA ITBAS     INTRPT TABLE BASE
0989  14557 170402       RET
0990*
0991*
0992*  ISERV - INTERRUPT SERVICE SUBR
0993*
0994*
0995  14560 031444 ISERV STA SAVEA
0996  14561 035445       STB SAVEB
0997  14562 172700       SFC 0
0998  14563 066611       JMP IS2       MUST BE STOP
0999  14564 172241       LIA 1         INPUT INTERPT ADDRESSES
1000  14565 070504       SAL 6
1001  14566 070242       SAR 6         CLEAR 6 MSB
1002  14567 072410       RZA ISE4
1003  14570 023736       LDA KEYBD
1004  14571 172160       OTA 16
1005  14572 172741       STF 1
1006  14573 172240       LIA 0
1007  14574 070342       SAR 8
1008  14575 031443       STA KEYFG
1009  14576 066651       JMP RESTR
1010  14577 074742 ISE4  SBR 16
1011  14600 035446       STB IADDR      TEMPR INTRPT ADDR
1012  14601 070006       RAR 1         POSITION NEXT H.P. BIT
1013  14602 045446       ISZ IADDR     INCRM COUNT
1014  14603 071713       SAP *-2       CONTINUE
1015  14604 021446       LDA IADDR
1016  14605 062554       JSM FITAS
1017  14606 000012       ADA .1         FIG. TABLE ADDR OF SERV SUBR
1018  14607 070537       LDB A,I
1019  14610 074737       JMP B,I       GO TO SERVICE SUBR
1020  14611 020040 IS2   LDA .31
1021  14612 031442       STA STPFL
1022  14613 020104       LDA B177
1023  14614 031443       STA KEYFG
1024  14615 021444 IS3   LDA SAVEA
1025  14616 025445       LDB SAVEB
1026  14617 170402       RET
1027*
1028*
1029*  IST0 - STATUS OF FIND COMMAND
1030*
1031*
1032*    POSITIVE FILE DIFF FOR FORW  AND NEG FOR REV DIRECTION
1033*
1034*
1035  14620 021446 IST0  LDA IADDR
1036  14621 070204       SAL 12          FIGURE SELECT CODE
1037  14622 063003       JSM STAT+2     GET STATUS
1038  14623 063027       JSM STACK      STATUS CHECK
1039  14624 021446       LDA IADDR
1040  14625 062554       JSM FITAS
1041  14626 070517       LDB A,I        LOAD FILE STATUS
1042  14627 074152       SBM *+3
1043  14630 004132       ADB M1        ADD -1 FOR FORW DIRECTIN
1044  14631 066633       JMP IST1
1045  14632 004012       ADB .1        ADD +1 FOR REV DIRECTION
1046  14633 070577 IST1  STB A,I         RESTORE UPDATED STATUS
1047  14634 076110       RZB *+2
1048  14635 067020       JMP INSTP       FILE LOCATED
1049  14636 021446       LDA IADDR
1050  14637 070204       SAL 12         FIG. SELECT. CODE
1051  14640 074252       SBM *+5
```

#01 9830A TAPE CASSETTE SYSTEM

```
1052  14641 043747        IOR RCFF       READ-CONTROL-FORW-FAST
1053  14642 063034        JSM COUTP+2
1054  14643 172741        STF 1
1055  14644 066650        JMP COINT      COMMAND OUTPUT .  INTRPT TURN ON
1056  14645 043750        IOR RCRF       READ-CONTROL-REV-FAST
1057  14646 063034        JSM COUTP+2
1058  14647 172741        STF 1          CLEAR CONTROL
1059*
1060*
1061*
1062*
1063*
1064  14650 063034 COINT JSM COUTP+2
1065*
1066*
1067*       RESTORE REGISTERS
1068*
1069*
1070  14651 062615 RESTR JSM IS3
1071  14652 173741       CLF 1
1072  14653 170402       RET
1073*
1074*
1075*  BASIC DATA WRITER
1076*
1077*
1078  14654 031653 TWR   STA TAP4       STORE LOW ADDRESS LIMIT
1079  14655 035654       STB TAP5       STORE THE HIGH LIMIT
1080  14656 021645 TWR1  LDA TUNIT      UNIT NO.
1081  14657 040116       IOR WDFS       WRITE DATA FORW SLOW
1082  14660 070075       SEC *+1,C      CLEAR E
1083  14661 074742 TWR4  SBR 16
1084  14662 035655       STB TAP6       CLEAR CHECK:SUM REG
1085  14663 025653       LDB TAP4       LOAD INITIAL POINTER VALUE
1086  14664 172501       SFS 1
1087  14665 066664       JMP *-1
1088  14666 172141       OTA 1
1089  14667 121653       LDA TAP4,I
1090  14670 066674       JMP NWL2
1091  14671 074517 NWL   LDA B,I        LOAD A TRANSFER WORD
1092  14672 172501 NWL1  SFS 1          SKIP IF READY
1093  14673 066672       JMP *-1
1094  14674 172140 NWL2  OTA 0          WRITE 8 BITS
1095  14675 073655       SEC NWL1,S     WRITE 8 MORE BITS IF E=0
1096  14676 001655       ADA TAP6       UPDATE CHECK-SUM
1097  14677 031655       STA TAP6
1098  14700 070075       SEC *+1,C      CLEAR E
1099  14701 015654       CPB TAP5       AREA TRANSFERRED?
1100  14702 066705       JMP COUT       YES, OUTPUT CHECK-SUM
1101  14703 077330       RIB NWL        WRITE MORE
1102  14704 066671       JMP NWL        JUMP IF B=0
1103  14705 021655 COUT  LDA TAP6       LOAD CHECK-SUM
1104  14706 172501       SFS 1
1105  14707 066706       JMP *-1
1106  14710 172140       OTA 0          OUTPUT 8 BITS
1107  14711 073655       SEC *-3,S
1108  14712 025442       LDB STOP
1109  14713 074150       SZB *+3
1110  14714 063031       JSM HALT
1111  14715 067016       JMP ERR3       IMPROPER STOP
1112  14716 170402       RET
1113*
1114*
1115*  BASIC DATA READER
1116*
1117*
1118  14717 031653 TRD   STA TAP4       STORE START OF BUFFER AREA
1119  14720 035654       STB TAP5       STORE THE HIGH LIMIT
1120  14721 021645 TRD1  LDA TUNIT      UNIT NO.
1121  14722 172141       OTA 1            READ DATA FORW SLOW
1122  14723 172601       STC 1          ENABLE CASST
1123  14724 070742       SAR 16
1124  14725 031655       STA TAP6       CLEAR CHECK SUM REG
1125  14726 021653       LDA TAP4       LOAD INITIAL POINTER VALUE
1126  14727 070075 TRL   SEC *+1,C      CLEAR ERROR FLAG
1127  14730 172501       SFS 1          SKIP IF DONE
1128  14731 066730       JMP *-1
1129  14732 176240       LIB 0          INPUT 8 BITS
1130  14733 073655       SEC *-3,S      GET NEXT 8 BBITS
1131  14734 011654       CPA TAP5       AREA TRANSFERED?
1132  14735 066742       JMP CHCK       YES, RETURN AFTER CHECKING RESULT
1133  14736 070577       STB A,I        STORE THE READ WORD
1134  14737 005655       ADB TAP6       UPDATE CHECK SUM WORD
1135  14740 035655       STB TAP6
1136  14741 073330       RIA TRL        READ MORE
1137  14742 015655 CHCK  CPB TAP6       CHECK-SUM OK?
1138  14743 070075       SEC *+1,C      CLEAR ERROR FLAG
1139  14744 172741       STF 1          TURN OFF CEO
1140  14745 170402       RET
1141*
1142*
```

#01 9830A TAPE CASSETTE SYSTEM

```
1143*  WRFIL - WRITE A FILE
1144*  THE FILE HAS BEEN LOCATED
1145*  BACK UP AND WRITE NEW ID AND FILE
1146*
1147  14746 063610 WRFIL JSM RRBOF
1148  14747 062776       JSM CSTOP
1149  14750 020121 WRF3  LDA RCFS      READ CONTRL FORW SLOW
1150  14751 063037       JSM BOFCK
1151  14752 017740       CPB MARK
1152  14753 066755       JMP *+2
1153  14754 066750       JMP WRF3      KEEP LOOKING
1154  14755 062656       JSM TWR1       WRITE ID
1155  14756 021650       LDA TAP2      START ADDR
1156  14757 025660       LDB TAP7      END ADDR
1157  14760 004132       ADB M1       ADJUST COUNTER FOR WRITE ROUTINE
1158  14761 062654       JSM TWR       WRITE FILE BODY
1159  14762 172501       SFS 1
1160  14763 066762       JMP *-1       WAIT
1161  14764 067031       JMP HALT
1162*
1163*
1164*  STATE - GET STATUS
1165*
1166*
1167  14765 063001 STATE JSM STAT
1168  14766 074406       RBR 9         POSITION POWER ON BIT
1169  14767 074153       SBP STAT5      CAST NOT ON OR NOT PRESENT
1170  14770 074046       RBR 2          POSITION CLR + OPEN DOOR BITS
1171  14771 074111       SLB *+2
1172  14772 067014 STAT5 JMP STAC3
1173  14773 074112       SBM *+2
1174  14774 066776       JMP *+2
1175  14775 064357       JMP RETN2     CLEAR LEADER
1176  14776 025442 CSTOP LDB STOP      CHECK STOP
1177  14777 074350       SZB STAT4
1178  15000 067606       JMP GBACK     STOP CASST + GO BACK TO SYSTM
1179*
1180*
1181*
1182  15001 021645 STAT  LDA TUNIT
1183  15002 172741       STF 1          TURN OFF INTERPT
1184  15003 172141       OTA 1          LOAD I/O REG WITH SELECT CODE
1185  15004 172741       STF 1          LOAD I/O REG WITH STATUS
1186  15005 176241       LIB 1          INPUT STATUS INFO
1187  15006 170402 STAT4 RET
1188*
1189*
1190*  STACK - STATUS CHECK
1191*
1192*
1193  15007 074506 STACK RBR 11       POSITION CLEAR LEADER + OPEN DOOR BITS
1194  15010 074112       SBM *+2     CHECK CLEAR LEADER
1195  15011 075651       SLB STAT4        CHECK OPEN DOOR
1196  15012 063020       JSM INSTP
1197  15013 067016       JMP ERR3
1198  15014 063031 STAC3 JSM HALT          STOP CASST
1199  15015 173741       CLF 1         TURN ON INTERPT
1200  15016 160205 ERR3  JSM AERRA,I
1201  15017 177706       DEC -58
1202*
1203*
1204*      INSTP - STOP FROM INTRPT ROUTINE
1205*
1206*
1207  15020 021446 INSTP LDA IADDR
1208  15021 070204       SAL 12         FIG. SEL CODE
1209  15022 040117       IOR STOPC
1210  15023 063034       JSM COUTP+2
1211  15024 066651       JMP RESTR
1212*
1213*
1214*  WPERM - IS WRITE PERMITTED?
1215*
1216*
1217  15025 063001 WPERM JSM STAT
1218  15026 074504       SBL 6          POSITION WRITE PERMIT
1219  15027 074353       SBP COUT1
1220  15030 067014       JMP STAC3          NOT PERMITTED
1221*
1222*
1223*        HALT SUBR
1224*
1225*
1226  15031 020117 HALT  LDA STOPC         CONSTANT
1227*
1228*
1229*  COMMAND OUTPUT
1230*
1231*  ENTRY: COMMAND CONSTANT IN A REG.
1232*
```

#01 9830A TAPE CASSETTE SYSTEM

```
1233*
1234  15032 172741 COUTP STF 1          TURN OFF INTERPT
1235  15033 041645       IOR IUNIT
1236  15034 172141       OTA 1          OUTPUT COMMAND
1237  15035 172601       STC 1          ENABLE CAST
1238  15036 170402 COUT1 RET
1239*
1240*
1241*   BOFCK - CHECK FOR BOF
1242*
1243*
1244  15037 063032 BOFCK JSM COUTP
1245  15040 172501       SFS 1
1246  15041 067046       JMP BSTP
1247  15042 176241       LIB 1          INPUT CONTRL CHAR
1248  15043 074404       SBL 8
1249  15044 074342       SBR 8          CLEAR B MSR
1250  15045 170402       RET
1251  15046 025442 BSTP  LDB STOP
1252  15047 075450       SZB BOFCK+1        WAIT FOR BOF
1253  15050 067606       JMP GBACK          STOP CASST + GO BACK TO SYSTEM
1254*
1255*
1256*  RIFID - READ IMMDT FILE ID
1257*
1258*
1259  15051 063031 RIFID JSM HALT
1260  15052 062765       JSM STATE
1261  15053 067055       JMP *+2
1262  15054 063656       JSM OFFCL        GET OFF CLR LOR
1263  15055 020133 RIFI5 LDA M2
1264  15056 031643       STA TAP0
1265  15057 023751       LDA IDBA       ID BUFF ADDR
1266  15060 031653       STA TAP4
1267  15061 023753       LDA EOBID      END OF BUFF FOR ID
1268  15062 031654       STA TAP5
1269  15063 062776 RIF4  JSM CSTOP
1270  15064 020121       LDA RCFS       READ CONTRL FORW SLOW
1271  15065 063037       JSM BOFCK
1272  15066 017740       CPB MARK       BOF?
1273  15067 067073       JMP RD1        READ ID
1274  15070 062765       JSM STATE
1275  15071 067063       JMP RIF4
1276  15072 170402       RET            CLEAR LEADER
1277  15073 062721 RD1   JSM TRD1           READ FILE ID
1278  15074 070114       SES *+2        ERROR?
1279  15075 064357       JMP RETN2       NO
1280  15076 045643       ISZ TAP0
1281  15077 067103       JMP RIF6
1282  15100 063031 ERR4  JSM HALT
1283  15101 160205       JSM AERRA,I        CHECK SUM ERROR
1284  15102 177705       DEC -59
1285  15103 063610 RIF6  JSM RRBOF
1286  15104 067057       JMP RIFI5+2          TRY AGAIN
1287*
1288*
1289*
1290  15105 063610 RRRID JSM RRBOF      REV READ BOF
1291  15106 063051 RIFS  JSM RIFID       READ IMMDT FILE ID
1292  15107 067016       JMP ERR3       CLR LOR
1293  15110 170402       RET
1294*
1295*
1296*
1297*  LOCF - LOCATE A FILE
1298*
1299  15111 045472 LOCF  ISZ TEMPS
1300  15112 121472       LDA TEMPS,I
1301  15113 050065       AND OPMSK
1302  15114 013742       CPA NUMS       IS IT  # ?
1303  15115 067120       JMP *+3
1304  15116 062472       JSM FS         ASSUME INTERNAL
1305  15117 067121       JMP *+2
1306  15120 062475       JSM GUNT       GET UNIT NO.
1307  15121 062510       JSM FILEC      GET FILE
1308  15122 031644       STA TFILE
1309  15123 063106 LOCF2 JSM RIFS
1310  15124 020133       LDA M2
1311  15125 031647       STA TAP1       FLAG FOR FILE LOCATION
1312  15126 025664 LOCF1 LDB FILEN      FILE NO. FROM ID BUFF
1313  15127 015644       CPB TFILE
1314  15130 067031 LOCF3 JMP HALT       FILE LOCATED, RET
1315  15131 074076       TCB            MAKE FIL NO. READ NEG.
1316  15132 005644       ADB TFILE        FILE WANTED
1317  15133 004132       ADB M1
1318  15134 035653       STB TAP4         SAVE DIFF
1319  15135 076110       SZB *+2
1320  15136 067162       JMP FVERF        FILE LOCATED
1321  15137 074153       SBP *+3
1322  15140 023750       LDA RCRF       READ CONTRL REV FAST
```

```
1323  15141 #67145        JMP GDIR-1
1324  15142 #74076        TCB             MAKE DIFF NEG.
1325  15143 #35653        STB TAP4
1326  15144 #21747        LDA RCFF        READ CNTRL FORW FAST
1327  15145 #31654        STA TAP5
1328  15146 #21654 GDIR   LDA TAP5
1329  15147 #62776        JSM CSTOP
1330  15150 #63037        JSM BOFCK
1331  15151 #17740        CPB MARK
1332  15152 #67156        JMP GDIR2
1333  15153 #62765        JSM STATF
1334  15154 #67146        JMP GDIR    LOK=OK FOR BOF
1335  15155 #67016        JMP ERR3     CLR LOR
1336  15156 #45653 GDIR2  ISZ TAP4
1337  15157 #67146        JMP GDIR     CONTINUE LOOKING
1338  15160 #20160        LDA DEL2         15.6 MS DELAY
1339  15161 #72030        RIA *
1340*
1341*
1342*
1343  15162 #63106 FVERF  JSM RIFS     READ IMMDT FILE ID
1344  15163 #21644        LDA TFILE
1345  15164 #11664        CPA IDB0
1346  15165 #67031        JMP HALT    FILE LOCATED, T=RETURN
1347  15166 #45647        ISZ TAP1
1348  15167 #67126        JMP LOCF1    TRY AGAIN
1349  15170 #63031        JSM HALT
1350  15171 #66514        JMP ERR1      FILE NOT FOUND
1351*
1352*
1353*  LDATA- LOAD DATA EXECUTION
1354*
1355*
1356  15172 #63111 LDATA  JSM LOCF
1357  15173 #63256        JSM ARYLS
1358  15174 #25652        LDB TAP3     TEMPR, PRECISION
1359  15175 #15670        CPB IDB0+4
1360  15176 #67201        JMP *+3      PRECISION OK
1361  15177 #60205 ERR6   JSM AERRA,I   WORNG PRECISION
1362  15200 #77703        DEC -61
1363  15201 #70056        CMA
1364  15202 #01665        ADA IDB0+1     CHECK SIZE
1365  15203 #70112        SAM *+2
1366  15204 #67242        JMP ERR5
1367  15205 #63105        JSM RWRID    REV READ BOF; READ IMMDT FILE ID
1368  15206 #21650        LDA TAP2     POINT TOSTART OF LOAD AREA
1369  15207 #25665        LDB IDB0+1   FILE SIZE
1370  15210 #70037        ADB A         POINT TO END OF LOAD AREA
1371  15211 #62717        JSM TRD     READ THE FILE
1372  15212 #63031        JSM HALT
1373  15213 #70115        SEC *+2
1374  15214 #67100        JMP ERR4     CHECK SUM ERROR
1375  15215 #67654        JMP SYSRF     SYSTEM RET
1376*
1377*
1378*       SDATA - STORE DATA
1379*
1380*
1381  15216 #63252 SDATA  JSM ISTOR
1382  15217 #63256        JSM ARYLS
1383  15220 #25650        LDB TAP2     TEMPOR, ARRAY START ADDR
1384  15221 #70037        ADB A       COMPUTE END ADDR
1385  15222 #21652        LDA TAP3     TEMPOR, PRECISION
1386  15223 #31670        STA IDB0+4    STORE IN ID TABLE
1387  15224 #35660        STB TAP7     TEMPOR, END ADDR
1388  15225 #21650        LDA TAP2
1389  15226 #63237        JSM SDAT2
1390*
1391*
1392*
1393  15227 #20017 SDAT3  LDA .2      TYPE = DATA
1394  15230 #31666        STA IDB0+2     IN FILE ID BUFF
1395  15231 #21751        LDA IDBA     POINT TO REG. OF ID
1396  15232 #31653        STA TAP4
1397  15233 #23754        LDA EBIOW     POINT TO END OF ID
1398  15234 #31654        STA TAP5
1399  15235 #62746        JSM WRFIL    FRITE THE FILE
1400  15236 #67654        JMP SYSRF     SYSTEM RET
1401*
1402*     ENTER: AREG=START ADDR, BREG=END ADDR
1403*
1404  15237 #70076 SDAT2  TCA
1405  15240 #70037        ADB A         FIND FILE FIZE NEEDED
1406  15241 #76150        RZB *+3       SKIP IF NON-ZERO
1407  15242 #60205 ERR5   JSM AERRA,I   WRONG FILE SIZE
1408  15243 #77704        DEC -60
1409  15244 #75712        SSM *-2       ERROR IF NEG
1410  15245 #35665        STB IDB0+1
1411  15246 #74076        TCB
1412  15247 #05667        ADB IDB0+3     ABS FILE SIZE ON TAPE
```

```
#01  9830A TAPE CASSETTE SYSTEM

1413  15250 #75512        SBM ERR5      SIZE ERROR
1414  15251 170402        RET
1415*
1416*
1417*   ISTOR - INITLZ FOR STORE
1418*
1419*
1420  15252 #63111 ISTOR  JSM LOCF
1421  15253 #63025        JSM WPERM
1422  15254 #23752        LDA IDB4A
1423  15255 #67647        JMP RE3-1      CLEAR 4 CENTER WORDS + RET
1424*
1425*
1426*ARYLS - DATA ARRAY CALCULATIONS
1427*
1428*
1429  15256 #21472 ARYLS  LDA TEMPS
1430  15257 #11421        CPA PRADD      END OF STMT?
1431  15260 #67302        JMP ARCOM      YES, MUST BE COMMON
1432  15261 #60771        JSM GTOPP      NO, GET ARRAY NAME
1433  15262 160177        JSM SSYMA,I    SEARCH SYMBOL TABLE
1434  15263 164340        JMP E40+1,I      NOT FOUND
1435  15264 #74517        LDA B,I        READ NAME AND TYPE
1436  15265 #31661        STA TAPB       TEMPR
1437  15266 #50040        AND B37
1438  15267 #70042        SAP 2
1439  15270 #31652        STA TAP3       TEMPR, PRECISION
1440  15271 #04132        ADB M1
1441  15272 #74517        LDA B,I        GET ARRAY ADDR
1442  15273 #31650        STA TAP2
1443  15274 #00132        ADA M1
1444  15275 #70537        LDB A,I        READ CURR DIMENSION
1445  15276 #21661        LDA TAPB
1446  15277 160240        JSM MDIMA,I    COMPUTE THE SIZE
1447  15300 #00133        ADA M2
1448  15301 170402        RET
1449  15302 #20020 ARCOM  LDA .3
1450  15303 #31652        STA TAP3       SET COMMON TYPE
1451  15304 #21405        LDA FWUP
1452  15305 #31650        STA TAP2       SET START ADDR
1453  15306 #70076        TCA
1454  15307 #25422        LDB UKPTR
1455  15310 #74150        SZB *+3        KEY?
1456  15311 #01425        ADA PBUFO      YES
1457  15312 170402        RET
1458  15313 #01424        ADA PBUFF      COMPUTE SIZE OF COMMON
1459  15314 170402        RET
1460*
1461*
1462*  LKEY - LOAD KEYS
1463*
1464*
1465  15315 #63111 LKEY   JSM LOCF
1466  15316 #25421        LDB PRADD
1467  15317 163756        JSM DCM1A,I       DECOMPILE
1468  15320 #21666        LDA IDB0+2     FILE TYPE
1469  15321 #10021        CPA .4         KEY?
1470  15322 #67331        JMP *+7
1471  15323 #13741        CPA .24           SEC KEY ?
1472  15324 #67327        JMP *+3
1473  15325 160205 ERR7   JSM AERRA,I    TYPE ERROR
1474  15326 177702        DEC -62
1475  15327 #24106        LDB B777
1476  15330 #35435        STB SECFL      SET SEC FLAG
1477  15331 #21416        LDA MODE
1478  15332 #72350        RZA LKEY4      CALC MODE?
1479  15333 #21427        LDA .LNUM      NO
1480  15334 #70070        SIA *+1
1481  15335 #31661        STA TAPB        SAVE CURRENT LINE #+1
1482  15336 #21422        LDA UKPTR
1483  15337 #70110        SZA *+2        KEY MODE
1484  15340 #66147        JMP ERRA       YES, OVERLAY ERROR
1485  15341 #61225 LKEY4  JSM LOEF       INSURE NOT IN KEY MODE
1486  15342 #21405        LDA FWUP
1487  15343 #70076        TCA
1488  15344 #00166        ADA FWAM       COMPUTE - CURR KEY SIZE
1489  15345 #01665        ADA IDB0+1     COMPUTE DIFFERENCE (NEW-OLD)
1490  15346 #72110        RZA *+2
1491  15347 #67371        JMP LKEY9
1492  15350 #70412        SAM LKEY5      NEW KEYS LARGER?
1493  15351 160250        JSM OVCHA,I       YES CHECK FOR OVERFLOW
1494  15352 #21405        LDA FWUP
1495  15353 #31476        STA TEMP3      SET PROGRAM RELOCATION POINTER
1496  15354 #35470        STB L2T1
1497  15355 #61121        JSM MVTOH      MOVE MAINLINE AND COMMON
1498  15356 #25470        LDB L2T1
1499  15357 #67364        JMP LKEY6
1500  15360 #01405 LKEY5  ADA FWUP
1501  15361 #31477        STA TEMP4      COMPUTE NEW END OF KEYS
1502  15362 #25405        LDB FWUP
1503  15363 160251        JSM CLP1A,I    CLOSE UP PROGRAM
```

```
#01  9R30A TAPE CASSETTE SYSTEM

1504  15364 @21424 LKEY6 LDA PBUFF
1505  15365 @74017       ADA B
1506  15366 @31424       STA PBUFF      UPDATE START OF MAINLINE
1507  15367 @05405       ADB FWUP
1508  15370 @35405       STB FWUP         UPDATE END OF KEY AREA
1509  15371 @63105 LKEY9 JSM PRRID      REV READ BOF; READ IMMDT FIELD
1510  15372 @20166       LDA FWAM       START OF KEY AREA
1511  15373 @25665       LDB IDB0+1     FILE SIZE
1512  15374 @70437       ADB A          END OF KEY AREA
1513  15375 @62717       JSM TRD        READ THE FILE
1514  15376 @63031       JSM HALT
1515  15377 @70115       SEC *+2
1516  15400 @67100       JMP ERR4       CHECK SUM ERROR
1517  15401 @60651       JSM INITS      RESET STACKS
1518  15402 @66062       JMP LOAD9
1519*
1520*
1521*  SKEY - STORE KEYS
1522*
1523*
1524  15403 @63252 SKEY  JSM ISTOR
1525  15404 @25421       LDB PRADD
1526  15405 163756       JSM DCM1A,I
1527  15406 @63463       JSM SECCK
1528  15407 @24021       LDB .4         UNSEC KEY
1529  15410 @70113       SAP *+2
1530  15411 @27741       LDB .24         SEC KEY
1531  15412 @35666       STB IDB0+2     INFILE ID BUFF
1532  15413 @20166       LDA FWAM       START OF KEY AREA
1533  15414 @31650       STA TAP2       TEMPOR
1534  15415 @25405       LDB FWUP       END OF KEY AREA+1
1535  15416 @35660       STB TAP7       END OF KEY AREA
1536  15417 @63237       JSM SDAT2
1537  15420 @67231       JMP SDAT3+2
1538*
1539*
1540*     LBIN - LOAD BINARY
1541*
1542*
1543  15421 @63111 LBIN  JSM LOCF
1544  15422 @21403       LDA MAW        POINTER
1545  15423 @25665       LDB IDB0+1     FILE SIZE
1546  15424 @74076       TCB
1547  15425 @74017       ADA B          FIG. MEMORY START ADDR
1548  15426 @70117       LDB A       SAVE A
1549  15427 @70076       TCA
1550  15430 @01426       ADA PBPTR      POINTER
1551  15431 @70112       SAM *+2
1552  15432 @64561       JMP MERR       OVERFLOW
1553  15433 @21666       LDA IDB0+2     FILE TYPE
1554  15434 @14012       CPA .1         BINARY?
1555  15435 @67443       JMP LBI2        YES .  UNSEC
1556  15436 @10034       CPA .21         SEC BIN ?
1557  15437 @67441       JMP *+2
1558  15440 @67325       JMP ERR7
1559  15441 @20106       LDA B777
1560  15442 @31435       STA SECFL      SET SEC FLAG
1561  15443 @21427 LBI2  LDA .LNUM       PRESENT L.N.
1562  15444 @70070       STA *+1
1563  15445 @31661       STA TAPR
1564  15446 @76070       RIB *+1
1565  15447 @35404       STB LWAM         POINTER
1566  15450 @63105       JSM RRRID      REV READ BOF; READ IMMDT FILE ID
1567  15451 @21404       LDA LWAM
1568  15452 @25665       LDB IDB0+1     FILE SIZE
1569  15453 @70037       ADB A          END OF BIN
1570  15454 @62717       JSM TRD        READ THE FILE
1571  15455 @63031       JSM HALT
1572  15456 @70115       SEC *+2
1573  15457 @67100       JMP ERR4       CHECK SUM ERROR
1574  15460 @55404       DSZ LWAM
1575  15461 @60645       JSM MINIT      SCRATCH SYMBOL TABLE
1576  15462 @66062       JMP LOAD9
1577*
1578*
1579*     SECCK - SECURE CHECK
1580*
1581*
1582  15463 @21435 SECCK LDA SECFL
1583  15464 @50106       AND B777
1584  15465 @70150       SZA *+3
1585  15466 160205       JSM AERRA,I        SEC NOT ALLOWED
1586  15467 177707       DEC -57
1587  15470 @21435       LDA SECFL      RET WITH SEC FLG IN A
1588  15471 170402       RET
1589*
1590*
1591*     WRZE - WRITE ZEROS
1592*
1593*
```

```
        #01  9830A TAPE CASSETTE SYSTEM

1594  15472 024013 WRZF  LDB B400
1595  15473 074744       SBL 1          CONVERT WORD COUNT TO CH. COUNT
1596  15474 074076       TCB            NEG. COUNT
1597  15475 021645       LDA TUNIT
1598  15476 040116       IOR WDFS       WRITE DATA FORW SLOW
1599  15477 172501       SFS 1
1600  15500 067477       JMP *-1
1601  15501 172141       OTA 1
1602  15502 070742 WRZE1 SAR 16
1603  15503 172501       SFS 1
1604  15504 067503       JMP *-1
1605  15505 172140       OTA 0
1606  15506 077630       MIB WRZE1
1607  15507 170402       RET
1608*
1609*
1610*  MARKF - MARK TAPE EXECUTION
1611*
1612*
1613  15510 062457 MARKF JSM UNITC      GET UNIT NO.
1614  15511 063025       JSM WPERM
1615  15512 062765       JSM STATE      GET STATUS
1616  15513 067631       JMP MAR3       NO CLEAR LEADER
1617  15514 063524       JSM MAR4
1618  15515 063656 MAR2  JSM OFFCL      GET OFF CLEAR LEADER
1619  15516 020116       LDA WDFS       WRITE - DATA - FORW - SLOW
1620  15517 063032       JSM COUTP      COMMAND OUTPUT
1621  15520 063472       JSM WRZF       WRITE 256 WORDS OF 0
1622  15521 172501       SFS 1
1623  15522 067521       JMP *-1        STILL WRITING
1624  15523 067547       JMP MCONT
1625*
1626*
1627*
1628  15524 062510 MAR4  JSM FILEC      GET NO. OF FILES TO BE MARKED
1629  15525 000012       ADA .1         WRITE N+1 FILES
1630  15526 070076       TCA            MAKE FILE COUNT NEG.
1631  15527 031644       STA TFILE
1632  15530 062510       JSM FILEC      GET LENGTH OF FILE
1633  15531 031653       STA TAP4       ABS SIZE
1634  15532 070137       LDB A
1635  15533 074002       SBR 1          FIND LENGTH/2
1636  15534 000135       ADA M4
1637  15535 070113       SAP *+2
1638  15536 066514       JMP ERR1       FILE TOO SMALL
1639  15537 021653       LDA TAP4
1640  15540 074017       ADA B
1641  15541 000045       ADA .40          40 WORDS ADDITIONAL BUFF
1642  15542 031657       STA TLENG      LENGTH TEMPOR
1643  15543 063644       JSM IDCLR
1644  15544 021653       LDA TAP4        ABS SIZE
1645  15545 031667       STA IDB0+3
1646  15546 170402       RET
1647*
1648*
1649*
1650  15547 023730 MCONT LDA WCFS       WRITE -CONTROL -FORW-SLOW
1651  15550 041645       IOR TUNIT
1652  15551 172141       OTA 1
1653  15552 172601       STC 1          ENABLE CASRT
1654  15553 023751 MCON2 LDA IDBA       POINTER TO BEGIN OF ID
1655  15554 027754       LDB EBIOW       POINT TO END OF ID
1656  15555 062654       JSM TWR        WRITE THE FILE ID
1657  15556 025657       LDB TLENG      LOAD ACTUAL FILE LENGTH
1658  15557 063473       JSM WRZF+1       WRITE THE FILE
1659  15560 172501       SFS 1
1660  15561 067560       JMP *-1         WAIT .  IF STILL WRITING
1661  15562 172741       STF 1
1662  15563 172241       LIA 1           INPUT STATUS
1663  15564 050122       AND B6K         ISOLATE OPEN DOOR+CLR LLP
1664  15565 070610       SZA MC2          SKIP IF OK
1665  15566 070506       RAR 11          POSITION STATUS BITS
1666  15567 070151       SLA *+3
1667  15570 063031 ER3   JSM HALT
1668  15571 067016       JMP ERR3
1669  15572 020107 MC3   LDA RDRS       READ DAR*TA REV SLOW
1670  15573 063657       JSM OFFCL+1       GET OFF CLEAR LEADER-EOT
1671  15574 063610       JSM RRBOF
1672  15575 063472       JSM WRZF        ERASE LAST BOF
1673  15576 172501       SFS 1
1674  15577 067576       JMP *-1         WAIT IF STILL WRITING
1675  15600 063652       JSM REWND+1    GIVE COMMAND AND RETURN TO SYSTEM
1676  15601 045664 MC2   ISZ IDB0       INCRM FILE NO.
1677  15602 045644       ISZ TFILE      INCRM FILE COUNTER
1678  15603 067547       JMP MCONT      CONTINUE
1679  15604 024151       LDB M32
1680  15605 063611       JSM RRBOF+1        BYPASS 32 CH. + REV. READ BOF
1681  15606 063031 GRACK JSM HALT
1682  15607 067654       JMP SYSRE        SYSTEM RET
1683*
1684*
```

```
1685*  RRBOF - REVERS READ BOF
1686*
1687*
1688  15610 @24146 RRBOF LDB M15
1689  15611 @20107       LDA RDRS     READ-DATA-REV-SLOW
1690  15612 @63032       JSM COUTP
1691  15613 172501       SFS 1
1692  15614 @67613       JMP *-1
1693  15615 172240       LIA 0
1694  15616 @77670       RIB *-3
1695  15617 @23731       LDA RCRS     READ-CONTRL-REV-SLOW
1696  15620 @63037       JSM BOFCK
1697  15621 @17740       CPB MARK     BOF?
1698  15622 @67626       JMP DEL65
1699  15623 @62765       JSM STATE    GET STATUS
1700  15624 @67610       JMP RRBOF    KEEP LOOKING
1701  15625 @67016       JMP ERR3    BOT - CLR LOK
1702  15626 @23737 DEL65 LDA DEL1          75 MS DELAY
1703  15627 @72030       RIA *
1704  15630 170402       RET
1705*
1706*
1707*
1708  15631 @63106 MAR3  JSM RIFS       READ IMMDT FILE ID
1709  15632 @63610       JSM RRBOF    REV. READ BOF
1710  15633 @25644       LDB IDA0     SAVE LAST FILE NO.
1711  15634 @35643       STB TAP0      TEMPOR
1712  15635 @63524       JSM MAR4
1713  15636 @25643       LDB TAP0
1714  15637 @35644       STB IDB0
1715  15640 @20121       LDA RCFS     READ CONTROL FORW SLOWW
1716  15641 @63037       JSM BOFCK
1717  15642 @17740       CPB MARK     BOF ?
1718  15643 @67553       JMP MCON2     WRITE MORE FILES
1719*
1720*
1721*  IDCLR - CLEAR ID BUFF
1722*
1723*
1724  15644 @23751 IDCLR LDA IDBA
1725  15645 170000       CLR          CLEAR FILE NO. - ABS SIZE
1726  15646 @00021       ADA .4
1727  15647 170000       CLR       CLEAR STARTING LN + 3 WORDS
1728  15650 170402 RE3   RET
1729*
1730*
1731*  REWND - GIVE COMMAND AND RETURN TO SYSTEM
1732*
1733*
1734  15651 @62457 REWND JSM UNITC     GET UNIT NO.
1735  15652 @20110       LDA RDRF     READ-DATA-REV-FAST
1736  15653 @63032       JSM COUTP
1737  15654 173741 SYSRE CLF 1        TURN ON INTRPT
1738  15655 164172       JMP XEC4A,I   RETURN TO SYSTEM
1739*
1740*
1741*  OFFCL - GET OFF CLEAR LEADER
1742*
1743*
1744  15656 @20003 OFFCL LDA RDFS    READ DATA FORW SLOW
1745  15657 @24124       LDB DLCNT    DELAY COUNT
1746  15660 @35652       STB TAP3      TEMPR
1747  15661 @63032       JSM COUTP     COMMAND OUTPUT
1748  15662 @63001 OF1   JSM STAT     GET STATUS
1749  15663 @74544       SBL 5         POSITION CLEAR LEADER BIT
1750  15664 @75213       SEP RE3
1751  15665 @45652       ISZ TAP3      INCRM DELAY
1752  15666 @67642       JMP OF1       WAIT
1753  15667 @67016       JMP ERR3      MUST BE EOT
1754*
1755*
1756*  LISTF - TAPE LIST EXECUTION
1757*
1758*
1759  15670 @62457 LISTE JSM UNITC
1760  15671 @63106       JSM RIFS
1761  15672 @61344       JSM CRLF1
1762  15673 @63031 LI2   JSM HALT
1763  15674 @25664       LDB IDB0
1764  15675 @63720       JSM PRIN      PRINT FILE NO.
1765  15676 @25666       LDB IDB0+2
1766  15677 @63720       JSM PRIN      FILE TYPE
1767  15700 @25667       LDB IDB0+3
1768  15701 @63720       JSM PRIN      RPINT ABS FILE SIZE
1769  15702 @25665       LDB IDB0+1
1770  15703 @63720       JSM PRIN      CUR SIZE
1771  15704 @25670       LDB IDB0+4
1772  15705 @63720       JSM PRIN      START LINE NO.
1773  15706 @25671       LDB IDB0+5
1774  15707 @63720       JSM PRIN      END LINE NO.
```

```
#01  9830A TAPE CASSETTE SYSTEM


1775  15710 025672        LDB IDB0+6
1776  15711 063720        JSM PRIN        NO. OF WORDS IN COMON
1777  15712 061344        JSM CRLF1
1778  15713 160170        JSM CLERA,I
1779  15714 031453        STA PRFLG           CL. FLAG FO CRLF
1780  15715 063055        JSM RIFIS
1781  15716 067652        JMP REWND+1          RET TO SYSTEM + REWIND
1782  15717 067673        JMP LI2             CONTINUE LIST
1783  15720 035647 PRIN   STB TAP1
1784  15721 160170        JSM CLERA,I   CLEAR SINGLE LINE BUFF
1785  15722 025647        LDB TAP1
1786  15723 045400        ISZ BADDR
1787  15724 160210        JSM OUTIA,I   MAKE ASCII AND STORE
1788  15725 020022        LDA CCNT      CHARACTER COUNT
1789  15726 024212        LDB .SLBF     POINT TO FIRST CHAR.
1790  15727 065337        JMP NOWPT     PRINT
1791*
1792*
1793*
1794*
1795*
1796*
1797*
1798*  DEFINITIONS
1799*
1800*
1801  00003         RDFS  EQU C6        ZERO
1802  15730 006074  WCFS  OCT 6074
1803  15731 005000  RCRS  OCT 5000
1804  00110         RDRF  EQU B1400
1805  00107         RDRS  EQU B1000
1806  01657         TLENG EQU 1657B
1807  00124         DLCNT EQU EXPOP          32000B
1808  15732 077040  DATOP OCT 77040
1809  15733 077000  KEYOP OCT 77000
1810  15734 077200  BINOP OCT 77200
1811  15735 120000  B120K OCT 120000
1812  15736 140000  KEYRD OCT 140000
1813  00004         XECA  EQU B10K
1814  00160         B176K EQU M1024
1815  15737 166272  DEL1  DEC -4934
1816  00160         DEL2  EQU M1024
1817  15740 000074  MARK  OCT 74
1818  15741 000030  .24   DEC 24
1819  00022         CCNT  EQU .5
1820  15742 040000  NUMS  OCT 40000
1821  01645         TUNIT EQU 1645B
1822  01644         TFILE EQU 1644B
1823  01446         IADDR EQU AURSS
1824  15743 040400  ITRAS DEF 40400B
1825  01643         TAP0  EQU 1643B
1826  01647         TAP1  EQU 1647B
1827  01650         TAP2  EQU 1650B
1828  15744 014620  CASIN DEF IST0
1829  01664         FILEN EQU 1664B
1830  01652         TAP3  EQU 1652B
1831  01653         TAP4  EQU 1653B
1832  01654         TAP5  EQU 1654B
1833  01655         TAP6  EQU 1655B
1834  01660         TAP7  EQU 1660B
1835  01661         TAPA  EQU 1661B
1836  15745 001660  TAPAD DEF 1660B
1837  15746 011775  TCOM  OCT 11775
1838  01435         SECFL EQU SEC
1839  01663         OFSET EQU 1663B
1840  15747 004400  RCFF  OCT 4400
1841  00121         RCFS  EQU B4000
1842  15750 005400  RCRF  OCT 5400
1843  15751 001664  IDBA  DEF 1664B
1844  01664         IDB0  EQU 1664B
1845  15752 001670  IDB4A DEF 1670B
1846  15753 001700  EOBID DEF 1700B
1847  15754 001677  EBIDW DEF 1677B       END OF ID BUFF FOR WRITE
1848  00116         WDFS  EQU QTOP
1849  00013         RDFF  EQU B400
1850  00245         GETSA EQU GTSTA
1851  01442         STOP  EQU STPFL
1852  00117         STOPC EQU B2400
1853  15755 016747  ERR0  DEF ERROR
1854  15756 005717  DCM1A DEF 5717B
1855*
1856*  TABLE LINKS
1857*
1858  15762               ORG 15762B
1859  15762 176462        ABS LINK-*
1860  15763 176334        ABS ASTOR-*
1861  15764 176534        ABS FINDE-*
1862  15765 177703        ABS LISTE-*
1863  15766 177663        ABS REWND-*
1864  15767 177521        ABS MARKE-*
```

```
        #01  9830A TAPE CASSETTE SYSTEM

1865  15770 174052        ABS ALOAD-*
1866  15771 176417        ABS MERGE-*
1867  15772 000000        HSS 1                    ******* CHECK SUM ***********
1868  15773 176030        ABS SUBT-*
1869  15774 162111        ABS B377-*
1870  15775 176004        ABS CMNDT-*
1871  15776 000630        ABS SNMT-*
1872  15777 014000        OCT 14000
1873*
1874*
1875*     SYNTAX
1876*
1877  16615               ORG 16615B
1878*  SYNTAX JUMP TABLE
1879  16615 000041        ABS MLOST-*
1880  16616 000064        ABS ELOST-*
1881  16617 000051        ABS FIMAR-*
1882  16620 000124        ABS REWLT-*
1883  16621 000123        ABS REWLT-*
1884  16622 000046        ABS FIMAR-*
1885  16623 000057        ABS ELOST-*
1886  16624 000032        ABS MLOST-*
1887  16625 000031 B31    OCT 31
1888*  SYNTAX TABLE
1889  16626 043111 SNMT   ASC 2,FIND     FIND, OPCODE+DE 7
      16627 047104
1890  16630 123522        OCT 123522     OPCODE, R
1891  16631 042527        ASC 2,EWIN     REWIND, OPCODE 5
      16632 044516
1892  16633 042245        OCT 42245      D,OPCODE
1893  16634 046505        ASC 2,MERG     MERGE, OPCODE 2
      16635 051107
1894  16636 042642        OCT 42642      F,OPCODE
1895  16637 051524        ASC 2,STOR          STORE,  OPCODE 10
      16640 047522
1896  16641 042650        OCT 42650          F, OPCODE
1897  16642 046117        ASC 2,LOAD     LOAD OPCODE 3
      16643 040504
1898  16644 121515        OCT 121515     OPCODE, M
1899  16645 040522        ASC 1,AR       MARK, OPCODE 4
1900  16646 045644        OCT 45644      K, OPCODE
1901  16647 052114        ASC 2,TLIS     TLIST, OPCODE 6
      16650 044523
1902  16651 052246        OCT 52246         T, OPCODE
1903  16652 046111        ASC 2,LINK        LINK ,  OPCODE 11
      16653 047113
1904  16654 164400        OCT 164400        OPCODE, ET
1905*
1906*
1907*  MERGE, LOAD, STORE SYNTAX CHECK
1908*
1909*
1910  16655 062764 ML     JSM UNIT1      DECRM POINTERS
1911  16656 062751 MLOST  JSM UNIT
1912  16657 066661        JMP MA         NO UNIT GIVEN
1913  16660 062770        JSM COMEL      COMA? NO, EOL
1914  16661 162773 MA     JSM SERVA,I
1915  16662 065153        JMP EOLCK      0=EOL? ACCEPT
1916  16663 062770        JSM COMEL      FILE NO. GIVEN
1917  16664 060664        JSM LNUM        GET LINE #1
1918  16665 062770        JSM COMEL
1919  16666 060664        JSM LNUM          GET LINE #2
1920  16667 065151        JMP EOST
1921*
1922*
1923*  FIND AND MARK SYNTAX CHECK
1924*
1925*
1926  16670 062751 FIMAR  JSM UNIT
1927  16671 066673        JMP AA         NO UNIT GIVEN
1928  16672 060510        JSM COMCF      COMA? NO, ERROR
1929  16673 160202 AA     JSM FSCA,I     GET FILE
1930  16674 025473        LDB TEMP
1931  16675 014024        CPB .7         FIND?
1932  16676 065151        JMP EOST       ACCEPT STATEMENT
1933  16677 060510        JSM COMCF
1934  16700 160202        JSM FSCA,I
1935  16701 065151        JMP EOST
1936*
1937*
1938*  ENTRY FOR ALL LOAD AND STORE
1939*
1940*
1941  16702 021466 ELOST  LDA LOCL1
1942  16703 050040        AND B37
1943  16704 031473        STA TEMP       TEMPOR, OPCODE
1944  16705 060456        JSM SBPUD      INCRM POINTER SRPTR
1945  16706 061055        JSM GNEXT      GET NEXT CHARACTER
1946  16707 026774        LDB SUBTA      ADDR OF SUB-TABLE
1947  16710 060546        JSM STEXP
```

```
          #01  9830A TAPE CASSETTE SYSTEM

1948  16711 #66655        JMP ML          LOAD, STORE
1949  16712 #70202        SAR 5           POSITION KEY OR DATA OPCODE
1950  16713 #50040        AND B37
1951  16714 #01473        ADA TEMP        FIGURE EXACT OPCODE
1952  16715 #31473        STA TEMP
1953*
1954*
1955*      LOADK,STOREK,LOADD,STORED,LOADB,STORER,SYNTAX CHECK
1956*
1957*
1958  16716 #62754        JSM UNIT2
1959  16717 #66741        JMP LA          NO UNIT GIVEN
1960  16720 #60510        JSM COMCE       COMA? NO, ERROR
1961  16721 162773 LA1    JSM SERVA,I
1962  16722 #66747        JMP ERROR       EOL AFTER COMA
1963  16723 #25473        LDB TEMP
1964  16724 #16625        CPB B31           STORE DATA ?
1965  16725 #66731        JMP *+4         YES
1966  16726 #16775        CPB BB24         LOAD DATA ?
1967  16727 #66731        JMP *+2         YES
1968  16730 #65151        JMP EOST        DEMAND EOL
1969  16731 #62770        JSM COMEL       COMA? NO, EOL
1970  16732 #61140        JSM LTR         CHECK FOR ARRAY
1971  16733 #66747        JMP ERROR
1972  16734 #21474        LDA TEMP1       LETTER: LOAD VARIABLE NAME
1973  16735 #24020        LDB .3          ASSUME ARRAY
1974  16736 #60451        JSM STROP+1     PUT IN MEMORY
1975  16737 #21475        LDA TEMP2       LOAD FOLLOWING CHARACTER
1976  16740 #65151        JMP EOST
1977  16741 #70742 LA     SAR 16
1978  16742 #60456        JSM SBPUD
1979  16743 #66721        JMP LA1
1980*
1981*
1982*  REWIND AND TLIST SYNTAX CHECK
1983*
1984*
1985  16744 #62751 REWLI  JSM UNIT
1986  16745 #65147        JMP ENDS
1987  16746 #65151        JMP EOST        ACCEPT STATEMENT
1988  16747 160205 ERROR  JSM AERRA,I
1989  16750 177711        DEC -55
1990*
1991*
1992*    UNIT SYNTAX CHECK
1993*
1994*
1995  16751 #21466 UNIT   LDA LOCL1
1996  16752 #50040        AND B37         RECALL OPCODE
1997  16753 #31473        STA TEMP        OPCODE TEMPORARY
1998  16754 #60456 UNIT2  JSM SBPUD       GET NEW SYNTAX WORD
1999  16755 #61055        JSM GNEXT       GET NEW CHARACTER
2000  16756 #60463        JSM SYMC1       CHECK FOR #
2001  16757 #12017       OCT 12017
2002  16760 #66764        JMP UNIT1       NO
2003  16761 #66762        JMP FS          YES
2004  16762 160202 FS     JSM FSCA,I      GET UNIT
2005  16763 #64357        JMP RETN2
2006  16764 #55706 UNIT1  DSZ SBPTR
2007  16765 #55400        DSZ BADDR
2008  16766 170402        RET
2009*
2010*
2011*  COMA-EOL CHECK
2012*
2013*
2014  16767 #61055 COMA   JSM GNEXT
2015  16770 #60504 COMEL  JSM COMCK       COMA?
2016  16771 #65151        JMP EOST        NO
2017  16772 170402        RET
2018*
2019*      DEFINITIONS
2020*
2021  16773 #14032 SERVA  DEF SERV
2022  16774 #30046 SUBTA  DEF SUBT+SUBT
2023  16775 #00024 BB24   OCT 24
2024                      END
*  NO ERRORS*


          CROSS-REFERENCE TABLE

.1      0026      0677      0871      0955      1017      1045      1554      1629
.10     0077      0078
.100    0076
.1000   0075
.11     0041      0049
.12     0042
.13     0045
```

CROSS-REFERENCE TABLE

.15	0054	0089								
.16	0055	0090								
.1F4	0074	0626	0821							
.2	0033	1393								
.21	0056	1556								
.22	0057	0093								
.23	0058	0094	0643	0852						
.24	1818	1471	1530							
.28	0059									
.3	0034	0051	0641	0850	1449	1973				
.31	0060	0096	1020							
.32	0061									
.33	0062									
.34	0063	0097								
.37	0064									
.4	0035	1469	1528	1726						
.40	0065	1641								
.41	0066									
.43	0067									
.45	0068									
.46	0069									
.48	0070									
.5	0036	1819								
.58	0071	0100								
.6	0037	0050								
.63	0072									
.7	0038	1931								
.8	0039									
.9	0040									
.99	0080									
.BUFA	0186									
.FADA	0191									
.FDVA	0194									
.FMPA	0193									
.FSBA	0192									
.FSR	0196									
.LNUM	0320	0633	1479	1561						
.NPWR	0500									
.SLBF	0187	1789								
A	0000	0720	0957	0969	1018	1041	1046	1133	1370	1384	1405
	1444	1512	1548	1569	1634						
AA	1929	1927									
ACTSA	0180									
ADRSS	0333	1823									
AERRA	0182	0653	0943	1200	1283	1361	1407	1473	1585	1988	
ALOAD	0573	1865									
AR1	0419	0188									
AR1A	0188										
AR2	0421	0189									
AR2A	0189										
ARCOM	1449	1431									
ARYAD	0409										
ARYLS	1429	1357	1382								
ASTOR	0793	1860									
B	0000	0524	0527	0610	0747	0749	0862	0927	0945	1019	1091
	1435	1441	1505	1547	1640						
B100	0101										
B1000	0108	1805									
B10K	0120	0121	1813								
B120K	1811	0920									
B12K	0122	0123	0212								
B133	0102										
B140	0103										
B1400	0109	0210	1804								
B17	0089										
B170K	0127										
B1740	0110	0209									
B1760	0111	0164									
B176K	1814										
B177	0104	1022									
B1774	0112										
B1775	0113										
B1777	0114										
B20	0090										
B22	0091										
B23	0092										
B2400	0116	1852									
B26	0093										
B27	0094										
B3000	0117										
B31	1887	1964									
B36	0095										
B37	0096	1437	1942	1950	1996						
B377	0105	1869									
B400	0106	1594	1849								
B4000	0118	0168	1841								
B42	0097										
B44	0098										
B52	0099										
B6K	0119	0173	1663								

CROSS-REFERENCE TABLE

B72	0100									
B777	0107	0646	1475	1559	1583					
BADDR	0297	0558	1786	2007						
BADRP	0210									
BB24	2023	1966								
BEND	0204	0205								
BHSTP	0450									
BINOP	1810	0581	0801							
BLANK	0394									
BOFCK	1244	1150	1252	1271	1330	1696	1716			
BSTP	1251	1246								
C5	0021									
C6	0017	0120	1801							
C7	0028	0106								
CASIN	1828	0956								
CCNT	1819	1788								
CHCK	1137	1132								
CHRSA	0225									
CLAR2	0497									
CLERA	0166	1778	1784							
CLINE	0322									
CLP1A	0223	0671	1503							
CMNDT	0517	1870								
COINT	1064	1055								
COM1	0704	0697								
COMA	2014									
COMCE	0442	1928	1933	1960						
COMCK	0441	2015								
COMEL	2015	1913	1916	1918	1969					
COMON	0687	0764								
CONST	0490									
COUT	1103	1100								
COUT1	1238	1219								
COUTP	1234	0976	0979	1053	1057	1064	1210	1244	1620	1690	1736
	1747									
CPLAC	0337									
CRCKA	0234									
CRLF	0503									
CRLF1	0504	1761	1777							
CRTN	0043	0044	0045							
CSTOP	1176	1148	1269	1329						
CTYPE	0336									
DATOP	1808	0577	0797							
DCM1A	1854	0639	0813	1467	1526					
DCOMA	0216									
DEL1	1815	1702								
DEL2	1816	1338								
DEL65	1702	1698								
DELSA	0181									
DFLAG	0396									
DI2	0972	0977	0980							
DI3	0974	0970								
DIGCK	0469									
DIREC	0964									
DLCNT	1807	1745								
DPERR	0049									
DRQST	0323									
DSTRT	0309									
E	0086									
E1	0241									
E10	0250									
E15	0255									
E20	0260									
E25	0265									
E30	0270	0695								
E35	0275									
E40	0280	0925	1434							
E45	0285									
E5	0245	0763								
E50	0290									
EBIDW	1847	1397	1655							
ECALA	0221									
ELINA	0175									
ELOST	1941	1880	1885							
ENDS	0486	1986								
ENTEA	0224									
ENTFA	0195									
EORID	1846	1267								
EOL	0044	0555								
EOLCK	0489	1915								
EOST	0487	1920	1932	1935	1968	1976	1987	2016		
ER3	1667									
ERR0	1853	0802								
ERR1	0943	1350	1638							
ERR3	1200	1111	1197	1292	1335	1668	1701	1753		
ERR4	1282	0724	1374	1516	1573					
ERR5	1407	0824	1366	1413						
ERR6	1361									
ERR7	1473	0645	1558							
ERR8	0653	0681	0882	1484						
ERRA	0183									

CROSS-REFERENCE TABLE

ERROR 1988 1853 1962 1971
ESEC 0519 0516
ESEC3 0522 0529
ESEC4 0533 0525
ESEC6 0530 0523
EXP 0407 0408
EXPOP 0124 1807
F 00A7
F5 0920 0910 0914 0933 1304
FDAAA 0228
FETCH 0494 0923 0939
FILEC 0939 0774 0958 1307 1628 1632
FILEN 1829 0964 1312
FILLA 0207
FIMAR 1926 1881 1884
FIND3 0960
FINDE 0952 1861
FIRST 0373
FITA3 0984 0968
FITAS 0986 0954 1016 1040
FIX 0507
FLGBT 0079 0531
FLOAT 0392
FLTFA 0199
FLTRA 0198 0924 0940
FLTSA 0201
FN 0154
FND 0468 0678
FNDIA 0230 0737
FNDAA 0229 0733
FNDPS 0482 0665 0879
FOPBS 0212
FORMA 0178
FORME 0342
FORMS 0341
FORMT 0340
FPAGE 0457
FWDMA 0128
FS 2004 2003
FSCA 0179 0559 1929 1934 2004
FSCE2 0447
FSCE4 0452
FTMP1 0343
FTMP2 0344
FVERF 1343 1320
FWAM 0163 1488 1510 1532
FWUP 0302 0698 0837 1451 1486 1494 1500 1502 1507 1508 1534
FXFLA 0200
GBACK 1681 1178 1253
GDIR 1328 1323 1334 1337
GDIR2 1336 1332
GETAD 0467 0519 0822
GETCH 0477
GETSA 1850
GETSK 0481
GFLAG 0410
GFRST 0480
GNEXT 0479 0553 1945 1999 2014
GPARM 0438
GSIGN 0492
GTCON 0506
GTMP 0478
GTOPP 0472 0573 0793 0909 1432
GTSTA 0219 1850
GUNT 0923 0918 1306
HALT 1226 0722 0971 1110 1161 1198 1259 1282 1314 1346 1349
1372 1514 1571 1667 1681 1762
HFLG1 0327
HSTPT 030A
I 0073
IADDR 1823 1011 1013 1015 1035 1039 1049 1207
ICNT 042A
IDR0 1844 0640 0657 0675 0687 0689 0705 0710 0719 0759 0827
0840 0843 0845 0853 0877 1345 1359 1364 1369 1386 1394
1410 1412 1468 1489 1511 1531 1545 1553 1568 1645 1676
1710 1714 1763 1765 1767 1769 1771 1773 1775
IDR4A 1845 1422
IDRA 1843 1265 1395 1654 1724
IDCLR 1724 1643
IMPY 0493
INF 0083
INITS 0462 0596 0899 1517
INREA 0232
INSTP 1207 1048 1196
INTCK 0465
INTEA 0184
INTEL 0346
INTFL 0081
INTGR 040A
IORUF 0365
IOREA 0235

CROSS-REFERENCE TABLE

```
IOSTA  0231
IS2    1020    0998
IS3    1024    1070
ISF4   1010    1002
ISERV  0995    0184
IST0   1035    1828
IST1   1046    1044
ISTOR  1420    1381    1524
ITBAS  1824    0988
KCOMN  0713    0691    0708
KEYBD  1812    1003
KEYFG  0330    1008    1023
KEYOP  1809    0579    0799
L2T1   0353    1496    1498
L2T2   0354
LA     1977    1959
LA1    1961    1979
LASTN  0404    0844
LBI2   1561    1555
LBIN   1543    0582
LBOP   0125
LDATA  1356    0578
LDEF   0495    1485
LETCK  0470
LI2    1762    1782
LIMIT  0471
LINK   0893    1859
LISTA  0173
LISTE  1759    1862
LKEY   1465    0580
LKEY4  1485    1478
LKEY5  1500    1492
LKEY6  1504    1499
LKEY9  1509    1491
LNFD   0078
LNINC  0321
LNUM   0463    1917    1919
LO12   0655    0650    0652
LO3    0605    0612    0810
LO4    0744    0734    0738    0751
LO5    0752    0746
LO6    0758    0666    0886
LO7    0875    0867    0869
LOAD2  0603    0589    0865    0893
LOAD3  0665    0593    0896
LOAD4  0675    0884
LOAD7  0638    0618    0621    0875
LOAD8  0616    0590    0894
LOAD9  0594    0885    0887    1518    1576
LOADP  0589    0574
LOCF   1299    1356    1420    1465    1543
LOCF1  1312    1348
LOCF2  1309    0779    0784
LOCF3  1314
LOCL1  0351    0833    1941    1995
LOCL2  0352
LPCK   0443
LPCKE  0444
LPOP   0126
LPOMA  0129
LSTAK  0305
LSTPT  0307
LTR    0485    1970
LWAM   0301    1565    1567    1574
M1     0130    0967    0986    1043    1157    1317    1440    1443
M10    0137
M100   0150
M1024  0152    1814    1816
M11    0138    0928
M12    0139
M13    0140
M14    0141
M15    0142    1688
M2     0131    0861    1263    1310    1447
M23    0143
M25    0144
M256   0151
M3     0132
M32    0145    1679
M36    0146
M4     0133    1636
M48    0147
M5     0134
M6     0135
M64    0148
M80    0149
M9     0136
MA     1914    1912
MAR2   1618
MAR3   1708    1616
MAR4   1628    1617    1712
```

CROSS-REFERENCE TABLE

MARK	1817	1151	1272	1331	1697	1717					
MARKE	1613	1864									
MASK1	0159										
MASK5	0160										
MAW	0300	1544									
MAXIN	0406										
MAXSN	0153	0761									
MBIN1	0412										
MBIN2	0413										
MBOX1	0357										
MC2	1676	1664									
MC3	1669										
MCON2	1654	1718									
MCONT	1650	1624	1678								
MDIMA	0214	1446									
MER8	0451	1552									
MERG1	0886	0880									
MERGE	0860	1866									
MINIT	0461	0591	0876	1575							
ML	1910	1948									
MLOST	1911	1879	1886								
MODE	0311	0594	0619	0637	0868	0874	0897	1477			
MSFLG	0395										
MVTOH	0483	0713	1497								
NDISP	0388										
NDWRT	0502	1790									
NEXT1	0456	0832									
NEXTA	0473	0528	0750								
NEXTL	0455										
NMOTA	0206										
NOEOF	0488										
NPRIN	0389										
NPWRT	0501										
NUM	0402	0403									
NUMCA	0217										
NUMOA	0218										
NUMS	1820	0917	1302								
NWL	1091	1101	1102								
NWL1	1092	1095									
NWL2	1094	1090									
NXTOT	0310										
NXTST	0411	0902									
OBLNK	0474										
ODGIT	0498										
OF1	1748	1752									
OFFCL	1744	1262	1618	1670							
OFSET	1839	0660	0676	0735	0748	0760	0878				
ONF	0851										
ONFXT	0475										
OPCHA	0177										
OPDMK	0085										
OPMSK	0082	0916	1301								
OPTYP	0403										
OTPTR	0299										
OUTCR	0476										
OUTIA	0185	1787									
OVCHA	0222	0690	0706	1493							
PARMA	0312										
PARMN	0313										
PHPTO	0319	0700	1550								
PHPTR	0316	0625	0649	0670	0701	0728	0731	0745	0753		
PHUFF	0317	0648	0651	0693	0699	0709	0711	0814	0839	1458	1504
	1506										
PHUFO	0318	1456									
PEND	0371										
PEXMA	0171										
PFLAG	0397										
PGIN0	0484										
PGINT	0464										
PI/2	0050										
PLACE	0390										
PMODE	0335										
PPFF	0372										
PRADD	0314	0606	0622	0624	0638	0772	0812	0913	1430	1466	1525
PRFLG	0338	1779									
PRIN	1783	1764	1766	176H	1770	1772	1774	1776			
QTCHA	0236										
QTOP	0115	0167	1848								
RBOP	0123										
RBUFF	0366										
RCFF	1840	0975	1052	1326							
RCFS	1841	1149	1270	1715							
RCOUT	0339										
RCRF	1842	0978	1056	1322							
RCRS	1803	1695									
RD1	1277	1273									
RDFF	1849										
RDFS	1801	1744									
RDRF	1804	1735									
RDRS	1805	1669	1689								

CROSS-REFERENCE TABLE

```
RDYDA   0172    0533    0597    0900
RE3     1728    1423    1750
REED    0167
REEDL   0345
RES     0423
RESB1   0459
RESBP   0460
RESTR   1070    1009    1211
RET2    0427
RETN2   0426    0560    1175    1279    2005
RETN3   0425
REWLI   1985    1882    1883
REWND   1734    1675    1781    1863
RIF4    1269    1275
RIF6    1285    1281
RIFIS   1263    1286    1780
RIFID   1259    1291
RIFS    1291    0960    1309    1343    1708    1760
RPCK    0445
RPCKE   0446
RPOP    0121
RRROF   1688    1147    1285    1294    1671    1680    1700    1709
RRRID   1290    0717    1367    1509    1566
RSAVE   0298
RSLTE   0379
RSPTR   0424
RSTAK   0422
RSTKA   0164
RSULT   0432
RUNA    0169    0599
S       0088
SALTA   0213
SAVBP   0458
SAVEA   0331    0995    1024
SAVEB   0332    0996    1025
SBADR   0400
SBPTR   0393    0557    2006
SBPUD   0437    0554    1944    1978    1998
SBUFA   0205
SDAT2   1404    0848    1389    1536
SDAT3   1393    0854    1537
SDATA   1381    0798
SEC     0326    1838
SECCK   1582    0849    1527
SECFL   1838    0530    0532    0647    1476    1560    1582    1587
SERV    0553    2021
SERVA   2021    1914    1961
SF2     0780    0773
SFLAG   0399
SFLOC   0770    0603    0809
SGNS    0499
SIGN    0405    0406
SKFY    1524    0800
SLCOD   0334
SLEND   0208
SLWST   0453
SNFLA   0203
SNMT    1889    1871
SPLFA   0202
SSRPT   0401
SSYMA   0176    1433
ST      0383
STAC3   1198    1172    1220
STACK   1193    1038
START   0168    0013
STAT    1182    1037    1167    1217    1748
STAT4   1187    1177    1195
STAT5   1172    1169
STATE   1167    1260    1274    1333    1615    1699
STEXP   0449    1947
STOP    1851    1108    1176    1251
STOPC   1852    1209    1226
STOR1   0434
STORE   0433
STORP   0809    0794
STPFL   0329    1021    1851
STROP   0436    1974
STSRH   0448
SUBT    0542    1868    2022    2022
SUBTA   2022    1946
SUMPA   0233
SYF12   0491
SYF25   0466
SYMC1   0439    2000
SYMC2   0438
SYMCK   0440
SYMTP   0304
SYSRE   1737    1375    1400    1682
SYTSA   0215
TAP0    1825    1264    1290    1711    1713
```

CROSS-REFERENCE TABLE

```
TAP1    1826    0520    0522    0729    0752    1311    1347    1783    1785
TAP2    1827    0668    0679    0830    0847    0883    1155    1368    1383    1398    1442
        1452    1533
TAP3    1838    1358    1385    1439    1450    1746    1751
TAP4    1831    0732    0744    1078    1085    1089    1118    1125    1266    1318    1325
        1336    1396    1633    1639    1644
TAP5    1832    0730    1079    1099    1119    1131    1268    1327    1328    1398
TAP6    1833    1084    1096    1097    1103    1124    1134    1135    1137
TAP7    1834    0592    0628    0630    0655    0659    0682    0692    0712    071A    075A
        0777    0781    0816    081B    0831    0846    0895    1156    1387    1535
TAP8    1835    0598    0616    0631    0635    0778    0782    0819    0866    0872    0901
        1436    1445    1481    1563
TAPAD   1836    0604
TCOM    1837    0834
TEMP    0356    0357    1430    1943    1951    1952    1963    1997
TEMP1   0358    1972
TEMP2   0359    1975
TEMP3   0360    0702    1495
TEMP4   0361    0521    0669    0823    0825    0828    1501
TEMP5   0355    0575    0576    0583    0605    0608    0609    0611    0736    0771    0795
        0796    0803    0829    0860    0911    0912    0915    0919    1299    1300    1429
TERM    0370
TFILE   1822    0775    0783    0959    0966    1308    1313    1316    1344    1631    1677
TFLAG   0324
TFLG1   032B
TLENG   1806    1642    1657
TMPA    0209
TMPOP   0190
TRD     111B    0721    1371    1513    1570
TRD1    1120    1277
TREGE   037B    0864    0870
TRL     1126    1136
TSRHA   0174
TSTPT   0306
TTY     0505
TUNIT   1821    0921    0984    1080    1120    1182    1235    1597    1651
TWR     107B    115B    1656
TWR1    1080    1154
TWR4    1083
TYPFL   0084
UFLAG   039B
UKPTR   0315    0696    0707    1454    1482
UNIT    1995    1911    1926    1985
UNIT1   2006    1910    2002
UNIT2   199B    1958
UNITC   0909    0770    0952    1613    1734    1759
UNITS   0325
UNSTK   0454
UOVF    0925    0929    0932
UPARA   0197
VALUE   0303
VAROA   0220
WBUFF   0429
WCFS    1802    1650
WDFS    1848    1081    1598    1619
WIDTH   0391
WPERM   1217    0811    1421    1614
WRF3    1149    1153
WRFIL   1147    1399
WRITE   0165
WRZE    1594    1621    1658    1672
WRZE1   1602    1606
XC      041B
XEC4A   0170    0973    1738
XEC6A   0227
XECA    1813    0903
XFAR1   0431
XFAR2   0435
XSCRA   0226
YC      0420
YCOMN   0692    0688
ZAP     015B
ZONK    0496
```

#01 MATRIX BLOCK

```
0501                    LST
0502  20000             ORG 20000B
0503*
0504* MATRIX BLOCK
0505*
0506  12001     QUOTF EQU 12001B      TABLE OF EQUATES FOR SYNTAX
0507  12002     COMMA EQU QUOTF+1
0508  12007     ASSOP EQU QUOTF+6
0509  12011     PLUS  EQU QUOTF+8
0510  12013     TIMES EQU QUOTF+10
0511*
0512*
0513  20000 000256 ID   ARS DETT-*     DETERMINANT JUMP TABLE
```

```
        #01  MATRIX BLOCK

0514*
0515  20001 000100        ABS REDS-*       SYNTAX JUMP TABLE
0516  20002 000041        ABS MATS-*
0517  20003 000000        NOP              ** CHECKSUM WORD **
0518*
0519  20004 046501 SYSNT  OCT 046501       MA       STATEMENT-NAME TABLE
0520  20005 052242        OCT 052242       T (2)
0521  20006 051105        OCT 051105       RE
0522  20007 042111        OCT 042111       DI
0523  20010 046003        OCT 046003       M (3)
0524*
0525  20011 046501        OCT 046501       MA
0526  20012 052040        OCT 052040       T
0527  20013 050122        OCT 050122       PR
0528  20014 044516        OCT 044516       IN
0529  20015 052204        OCT 052204       T (4)
0530  20016 046501        OCT 046501       MA
0531  20017 052040        OCT 052040       T
0532  20020 051105        OCT 051105       RE
0533  20021 040504        OCT 040504       AD
0534  20022 142400        OCT 142400         (5)   END
0535*
0536  20023 052122 NFOPT  OCT 052122       TR       NON-FORMULA OPERATOR TABLE
0537  20024 047201        OCT 047201       N (1)
0538  20025 044516        OCT 044516       IN
0539  20026 053202        OCT 053202       V (2)
0540  20027 055105        OCT 055105       ZE
0541  20030 051203        OCT 051203       R (3)
0542  20031 041517        OCT 041517       CO
0543  20032 047204        OCT 047204       N (4)
0544  20033 044504        OCT 044504       ID
0545  20034 047305        OCT 047305       N (5)   END
0546*
0547  20035 042105 FUNCT  OCT 042105       DE       FUNCTION-NAME TABLE
0548  20036 052335        OCT 052335       T (29)  END
0549*
0550  20037 000777 RN777  OCT 777          NULL TABLE
0551*
0552  20040 177771 M7     DEC -7
0553  20041 026000 R26K   OCT 26000
0554  20042 000046 NFOPA  DEF NFOPT+NFOPT-ID-ID
0556*
0557* MAT STATEMENT SYNTAX
0558*
0559  20043 021717 MATS   LDA NUM          GET PAGE INFO
0560  20044 070442        SAR 10
0561  20045 070244        SAL 11
0562  20046 031660        STA T8
0563*
0564  20047 055706        DSZ SBPTR        MOVE S-BUFFER POINTER
0565  20050 061055        JSM GNEXT        GET FIRST CHARACTER
0566  20051 026220        LDB SUBTA        GET SUBSIDIARY TABLE ADDRESS
0567  20052 035660        ADB T8
0568  20053 060537        JSM SISRH        DO SEARCH FOR "PRINT" OR "READ"
0569  20054 066107        JMP MATS2        NOT FOUND
0570  20055 045706        ISZ SBPTR
0571  20056 010004        CPA 010K         PRINT?
0572  20057 066066        JMP MATS0        YES
0573*
0574* MAT READ SYNTAX
0575*
0576  20060 062227 MATS1  JSM ARRID        RECORD ARRAY
0577  20061 062240        JSM MATSR        IF SUBSCRIPT, RECORD IT
0578  20062 066064        JMP *+2          NOP
0579  20063 061153        JSM EOLCK        END-OF-STATEMENT?
0580  20064 060510        JSM COMCF        NO, DEMAND COMMA
0581  20065 066060        JMP MATS1
0582*
0583* MAT PRINT SYNTAX
0584*
0585  20066 062227 MATS0  JSM ARRID        RECORD ARRAY
0586  20067 061153        JSM EOLCK        END-OF-STATEMENT?
0587  20070 060461        JSM SYMC?        NO, LOOK FOR COMMA OR SEMICOLON
0588  20071 012001        DEF COMMA-1
0589  20072 164277        JMP E10-2,I      NOT FOUND, ERROR 8
0590  20073 061066        JSM GETSK        END-OF-STATEMENT?
0591  20074 066077        JMP *+3          YES
0592  20075 055400        DSZ BADDR        NO
0593  20076 066066        JMP MATS0
0594  20077 045706        ISZ SBPTR
0595  20100 164203        JMP ACTSA,I
0596*
0597* REDIM SYNTAX
0598*
0599  20101 062227 REDS   JSM ARRID        RECORD ARRAY
0600  20102 062240        JSM MATSR        RECORD SUBSCRIPT
0601  20103 066105        JMP *+2          NOP
0602  20104 164306        JMP E15,I        NONE FOUND, ERROR 15
0603  20105 060510        JSM COMCF        DEMAND COMMA
0604  20106 066101        JMP REDS
```

#01 MATRIX BLOCK

```
0606*
0607* OTHER SYNTAX
0608*
0609  20107 045706 MAT$2 ISZ SBPTR
0610  20110 061141       JSM LTR+1        LETTER?
0611  20111 164321       JMP E25+1,I      NO, ERROR 26
0612  20112 021706       LDA SBPTR        SAVE CHARACTER ADDRESS
0613  20113 031723       STA ARYAD
0614  20114 062231       JSM ARRI         RECORD ARRAY
0615  20115 060463       JSM SYMC1        LOOK FOR "="
0616  20116 012006       DEF ASSOP-1
0617  20117 164325       JMP E30+1        NOT FOUND, ERROR
0618  20120 061055       JSM GNEXT
0619  20121 045706       ISZ SBPTR
0620  20122 026042       LDB NFOPA        SEARCH NON-FORMULA OPERATOR TABLE
0621  20123 005660       ADB T8
0622  20124 060546       JSM STEXP
0623  20125 066147       JMP MAT$8        NOT FOUND, MUST BE ARITHMETIC
0624  20126 045706       ISZ SBPTR
0625  20127 070202       SAR 5
0626  20130 050040       AND B37          RETRIEVE OPCODE
0627  20131 031661       STA T9
0628  20132 000134       ADA M3           INV OR TRN?
0629  20133 070252       SAM *+3          YES
0630  20134 061066       JSM GETSK        NO, END-OF-STATEMENT?
0631  20135 164203       JMP ACTSA,I      YES
0632  20136 062240       JSM MAT$8        NO, GO RECORD SUBSCRIPT
0633  20137 065152       JMP NOEOF
0634*
0635* INV & TRN SYNTAX
0636*
0637  20140 061055 MAT$3 JSM GNEXT
0638  20141 060521       JSM LPCKF        DEMAND LEFT PARENTHESIS
0639  20142 062227       JSM ARRID        RECORD ARRAY
0640  20143 060534       JSM RPCKF        DEMAND RIGHT PARENTHESIS
0641  20144 055661       DSZ T9           TRN?
0642  20145 065151       JMP EOST         NO
0643  20146 066205       JMP LHSC         GO CHECK PREVIOUS ARRAY IDENTIFIER
0644*
0645  20147 061141 MAT$8 JSM LTR+1        LETTER?
0646  20150 066154       JMP MAT$4        NO, IT MUST BE SCALAR MULTIPLY
0647  20151 060752       JSM LETCK        LETTER?
0648  20152 066164       JMP MAT$5        NO, IT MUST BE MAT ARITHMETIC
0649  20153 164326       JMP E30+1,I      YES, ERROR 31
0650*
0652* SCALAR MULTIPLY SYNTAX
0653*
0654  20154 060521 MAT$4 JSM LPCKF        DEMAND LEFT PARENTHESIS
0655  20155 160202       JSM FSCA,I       FETCH FORMULA
0656  20156 060534       JSM RPCKF        DEMAND RIGHT PARENTHESIS
0657  20157 060463       JSM SYMC1        LOOK FOR "*"
0658  20160 012012       DEF TIMES-1
0659  20161 164326       JMP E30+1,I      NOT FOUND, ERROR 31
0660  20162 062227       JSM ARRID        RECORD ARRAY
0661  20163 065151       JMP EOST
0662*
0663* MAT ARITHMETIC SYNTAX
0664*
0665  20164 056706 MAT$5 DSZ SBPTR
0666  20165 062231       JSM ARRI         RECORD ARRAY
0667  20166 061153       JSM EOLCK        END-OF-STATEMENT?
0668  20167 024020       LDB .3           NO, SEARCH FOR "+" "-" OR "*"
0669  20170 060464       JSM SYMCK
0670  20171 012010       DEF PLUS-1
0671  20172 164326       JMP E30+1,I      NOT FOUND, ERROR 31
0672  20173 074742       SHR 16           SET B FOR PFLAG
0673  20174 012041       CPA B26K         "*"?
0674  20175 066212       JMP MAT$7        YES
0675*
0676  20176 035712 MAT$6 STB PFLAG        SET PFLAG
0677  20177 062227       JSM ARRID        RECORD SECOND ARRAY
0678  20200 010031       CPA EOL          END-OF-STATEMENT?
0679  20201 066203       JMP *+2          YES
0680  20202 065152       JMP NOEOF        NO, ERROR
0681  20203 045712       ISZ PFLAG        WAS OPERATOR A "*"?
0682  20204 164203       JMP ACTSA,I      NO
0683  20205 125723 LHSC  LDB ARYAD,I      YES, RETRIEVE PREVIOUS ARRAY
0684  20206 074504       SHL 6
0685  20207 015473       CPB TEMP         SAME AS LEFT HAND SIDE?
0686  20210 164326       JMP E30+1,I      YES, ERROR (IMPROPER ASSIGNMENT)
0687  20211 065151       JMP EOST         NO
0688*
0689  20212 121723 MAT$7 LDA ARYAD,I      RETRIEVE ARRAY IDENTIFIER
0690  20213 070504       SAL 6
0691  20214 024132       LDB M1           SET B FOR PFLAG
0692  20215 011473       CPA TEMP         SAME AS LEFT HAND SIDE?
0693  20216 164326       JMP E30+1,I      YES, ERROR (IMPROPER ASSIGNMENT)
0694  20217 066176       JMP MAT$6        NO
```

```
          #01  MATRIX BLOCK

 0696  20220 000442 SUBTA DEF SUBTB+SUBTB-ID-ID
 0697  20221 052122 SUBTB OCT 052122       PR
 0698  20222 044516       OCT 044516       IN
 0699  20223 052204       OCT 052204       T (4)
 0700  20224 051105       OCT 051105       RE
 0701  20225 040504       OCT 040504       AD
 0702  20226 142400       OCT 142400         (5)   END
 0703*
 0704* SUBROUTINE TO FETCH ARRAY IDENTIFIER AND RECORD IT
 0705*
 0706  20227 061140 ARRID JSM LTR          LETTER?
 0707  20230 164321       JMP E25+1,I      NO. ERROR 26
 0708  20231 021474 ARRI  LDA TEMP1        RECORD ARRAY IDENTIFIER
 0709  20232 024020       LDB .3           OF UNKNOWN DIMENSIONS
 0710  20233 060451       JSM STROP+1
 0711  20234 070504       SAL 6
 0712  20235 031473       STA TEMP
 0713  20236 021475       LDA TEMP2        RETRIEVE FOLLOWING CHARACTER
 0714  20237 170402       RET
 0715*
 0716* SUBROUTINE TO FETCH MAT SUBSCRIPT AND RECORD IT
 0717*
 0718  20240 060513 MATSB JSM LPCK         LOOK FOR LEFT PARENTHESIS
 0719  20241 064357       JMP RETN2        NOT FOUND. RETURN TO P+2
 0720  20242 023125       LDA LBOP         CHANGE ( TO [
 0721  20243 131706       STA SBPTR,I
 0722  20244 160202       JSM FSCA,I       FETCH SUBSCRIPT
 0723  20245 060504       JSM COMCK        LOOK FOR COMMA
 0724  20246 066250       JMP *+2          NOT FOUND
 0725  20247 160202       JSM FSCA,I       FOUND. FETCH SECOND SUBSCRIPT
 0726  20250 060534       JSM RPCKE        DEMAND RIGHT PARENTHESIS
 0727  20251 055706       DSZ SBPTR
 0728  20252 024123       LDB RBOP         CHANGE ) TO ]
 0729  20253 135706       STB SBPTR,I
 0730  20254 045706       ISZ SBPTR
 0731  20255 065153       JMP EOLCK        CHECK FOR END-OF-STATEMENT
 0733*
 0734* DETERMINANT
 0735*
 0736  20256 021416 DETT  LDA MODE
 0737  20257 070453       SAP DETX         SYNTAX OR EXECUTION?
 0738*
 0739  20260 023777       LDA OPCOD        GET MAT OPCODE
 0740  20261 131706       STA SBPTR,I
 0741  20262 045706       ISZ SBPTR
 0742  20263 061055       JSM GNEXT
 0743  20264 060521       JSM LPCKE        DEMAND LEFT PARENTHESIS
 0744  20265 062227       JSM ARRID        RECORD ARRAY
 0745  20266 060534       JSM RPCKE        DEMAND RIGHT PARENTHESIS
 0746  20267 064356       JMP RETN3
 0747*
 0748*
 0749  20270 023726 DETX  LDA B1A          SAVE FIRST SEVEN TEMPORARIES
 0750  20271 024024       LDB .7
 0751  20272 162335       JSM MOVST,I
 0752  20273 021412       LDA LSTPT        SET UP FOR DETERMINANT EXECUTION
 0753  20274 031656       STA T6
 0754  20275 121472       LDA TEMPS,I
 0755  20276 045472       ISZ TEMPS
 0756  20277 045472       ISZ TEMPS
 0757  20300 063541       JSM INFO1        ESTABLISH INFORMATION VECTOR
 0758  20301 021642       LDA B1+2
 0759  20302 031645       STA B3+2
 0760  20303 063564       JSM COMP1        EXTRACT DIMENSIONS
 0761  20304 011653       CPA T3
 0762  20305 066307       JMP *+2
 0763  20306 067120       JMP MERR5        MATRIX NOT SQUARE
 0764*
 0765  20307 061201       JSM IMPY         N*N
 0766  20310 025640       LDB B1
 0767  20311 035643       STB B3
 0768  20312 076110       MPB *+2          GET # OF TEMPORARY WORDS NEEDED
 0769  20313 070704       SAL 2
 0770  20314 014012       CPB .1
 0771  20315 070744       SAL 1
 0772  20316 025412       LDB LSTPT
 0773  20317 074070       STB *+1
 0774  20320 035644       STB B3+1
 0775  20321 074017       ADA B
 0776  20322 063130       JSM XIN2+1       GO CALCULATE DETERMINANT
 0777  20323 066324       JMP *+1          DET=0 RETURN
 0778  20324 021656       LDA T6
 0779  20325 002040       ADA M7
 0780  20326 031412       STA LSTPT
 0781  20327 070070       STA *+1
 0782  20330 027726       LDB B1A          RESTORE FIRST SEVEN TEMPORARIES
 0783  20331 170004       XFR
 0784  20332 170004       XFR
 0785  20333 023731       LDA DELA
 0786  20334 064445       JMP XFAR2+1
 0787*
 0788  20335 011122 MOVST DEF 111228
```

#01 MATRIX BLOCK

```
0790*
0791* EXECUTE MAT PRINT
0792*
0793  20336 021416 EMATP LDA MODE        SAVE MODE FLAG
0794  20337 031656       STA T6
0795  20340 121472 MTP3  LDA TEMPS,I
0796  20341 045472       ISZ TEMPS
0797  20342 063541       JSM INFO1       ESTABLISH INFORMATION VECTOR
0798  20343 121472       LDA TEMPS,I
0799  20344 070442       SAR 10          GET FORMAT INDICATOR
0800  20345 074742       SBR 16
0801  20346 035453       STB PRFLG       INDICATE "PRINT"
0802  20347 010020       CPA .3
0803  20350 024020       LDB .3          3 IS ,    0 IS .
0804  20351 021472       LDA TEMPS
0805  20352 011421       CPA PRADD       FALSE ALARM?
0806  20353 074742       SBR 16          YES, IT WAS PART OF NEXT STATEMENT
0807  20354 035655       STB T5
0808  20355 021642       LDA B1+2        SET UP ROUNDS INFO
0809  20356 025642       LDB B1+2
0810  20357 063565       JSM COMPR
0811*
0812  20360 020132       LDA M1
0813  20361 031703       STA PLACE
0814  20362 022442       LDA LN72
0815  20363 031623       STA PEND
0816*
0817  20364 020012       LDA .1          START THE PRINT OPERATION
0818  20365 031646       STA T1
0819  20366 020012 MTP1  LDA .1
0820  20367 031652       STA T2
0821  20370 160170       JSM CLERA,I     CLEAR BUFFER
0822  20371 063600 MTP2  JSM GETTI       GET AN ARRAY ELEMENT
0823  20372 070742       SAR 16
0824  20373 031705       STA FLOAT       INDICATE FIXED-POINT FORMAT
0825  20374 031416       STA MODE
0826  20375 160234       JSM NMOTA,I
0827  20376 025400       LDB BADDR
0828  20377 015623       CPB PEND        CHECK FOR OVERFLOW
0829  20400 066435       JMP OVFW
0830  20401 070742       SAR 16          SIGNAL NUMERIC (NOT QUOTE) TO FILL ROUTINE
0831  20402 031742       STA XC+2
0832  20403 021655       LDA I5          CHECK THE FORMAT INDICATOR
0833  20404 160235       JSM FILLA,I     SUPPLY NEEDED BLANKS
0834  20405 160176       JSM ELINA,I     OUTPUT ONE FIELD
0835  20406 021442       LDA STPFL       CHECK STOP FLAG
0836  20407 010040       CPA .31
0837  20410 066437       JMP MTPX
0838  20411 021652       LDA T2
0839  20412 011654       CPA T4
0840  20413 066416       JMP *+3
0841  20414 045652       ISZ T2
0842  20415 066371       JMP MTP2
0843  20416 061344       JSM CRLF1       DOUBLE-SPACE BETWEEN ROWS
0844  20417 061344       JSM CRLF1
0845
0846  20420 021646       LDA T1
0847  20421 011653       CPA T3
0848  20422 066425       JMP *+3
0849  20423 045646       ISZ T1
0850  20424 066366       JMP MTP1
0851  20425 061344       JSM CRLF1       EXTRA SPACE AFTER MATRICES
0852  20426 025472       LDB TEMPS       CHECK FOR MORE ARRAYS TO BE PRINTED
0853  20427 015421       CPB PRADD
0854  20430 066437       JMP MTPX
0855  20431 121472       LDA TEMPS,I
0856  20432 050115       AND B1777       CHECK FOR TRAILING FORMAT INDICATOR
0857  20433 070210       SZA MTPX
0858  20434 066340       JMP MTP3
0859*
0860  20435 061344 OVFW  JSM CRLF1
0861  20436 066370       JMP MTP2-1
0862*
0863  20437 021656 MTPX  LDA T6          RESTORE MODE
0864  20440 031416       STA MODE
0865  20441 164172       JMP XEC4A,I
0866*
0867  20442 003433 LN72  DEF RBUFF+RBUFF+73
0868*
0869*
0870* EXECUTE MAT READ
0871*
0872  20443 121472 EMATR LDA TEMPS,I
0873  20444 045472       ISZ TEMPS
0874  20445 027730       LDB B3A
0875  20446 063542       JSM INFOV       ESTABLISH INFORMATION VECTOR
0876  20447 121472       LDA TEMPS,I
0877  20450 050065       AND OPMSK
0878  20451 010125       CPA LBOP        IS IT ( ?
0879  20452 063625       JSM REDIR       YES, GO REDIMENSION
0880  20453 063563       JSM COMP2
0881  20454 022462       LDA EMRX        GO EXECUTE DATA TRANSFERS
```

```
        #01  MATRIX BLOCK


0882  20455 062617        JSM COMM
0883  20456 025472        LDB TEMPS
0884  20457 015421        CPB PRADD
0885  20460 164172        JMP XEC4A,I      DONE
0886  20461 066443        JMP EMATR
0887*
0888  20462 062463 EMRX   JSM *+1
0889  20463 160256        JSM FDAAA,I
0890*

0891* SUBROUTINE TO STASH AN ARRAY ELEMENT
0892*

0893* ELEMENT IS IN AR2
0894*

0895* ROW & COLUMN VALUES ARE IN T1 AND T2
0896*
0897  20464 027730 STASH LDB B3A          FIND TARGET ADDRESS
0898  20465 063705        JSM WHERE
0899  20466 021643        LDA B3           FIND TYPE OF ARRAY
0900  20467 064421        JMP STORE+3
0901*
0902*

0903* EXECUTE REDIM
0904*
0905  20470 121472 EREDM LDA TEMPS,I
0906  20471 045472        ISZ TEMPS
0907  20472 027730        LDB B3A
0908  20473 063542        JSM INFOV        ESTABLISH INFORMATION VECTOR
0909  20474 063625        JSM REDIR        GO REDIMENSION
0910  20475 025472        LDB TEMPS
0911  20476 015421        CPB PRADD
0912  20477 164172        JMP XEC4A,I      DONE
0913  20500 066470        JMP EREDM
0915  20501 121472 EMATX  LDA TEMPS,I      GET DESTINATION ARRAY
0916  20502 045472        ISZ TEMPS
0917  20503 027730        LDB B3A
0918  20504 063542        JSM INFOV        ESTABLISH INFORMATION VECTOR
0919  20505 121472        LDA TEMPS,I
0920  20506 045472        ISZ TEMPS
0921  20507 050115        AND B1777
0922  20510 072310        SZA MATX1        REGULAR MAT ASSIGNMENT?
0923  20511 121472        LDA TEMPS,I      NO
0924  20512 070442        SAR 10
0925  20513 010040        CPA .31          SCALAR MULTIPLY?
0926  20514 066531        JMP MATX0        NO
0927  20515 066644        JMP ESMY         YES
0928*
0929  20516 063541 MATX1  JSM INFO1        ESTABLISH INFORMATION VECTOR
0930  20517 025472        LDB TEMPS
0931  20520 015421        CPB PRADD        STATEMENT EXHAUSTED?
0932  20521 066605        JMP EMRP         YES, GO DO MAT REPLACEMENT
0933  20522 121472        LDA TEMPS,I      NO, GET SECOND OPERAND
0934  20523 027727        LDB B2A
0935  20524 063542        JSM INFOV        ESTABLISH INFORMATION VECTOR
0936  20525 121472        LDA TEMPS,I      THEN GET MAT OPERATOR
0937  20526 070442        SAR 10
0938  20527 000134        ADA M3
0939  20530 066535        JMP MATJ
0940*
0941  20531 121472 MATX0  LDA TEMPS,I      IDENTIFY SPECIAL NON-FORMULA OPERATOR
0942  20532 045472        ISZ TEMPS
0943  20533 070202        SAR 5
0944  20534 050040        AND .31
0945  20535 000537 MATJ   ADA XTBL1
0946  20536 170016        PAA              BRANCH TO PROPER ROUTINE
0947  20537 066537 XTBL1  JMP *
0948  20540 066722        JMP ETRN   TRN
0949  20541 067110        JMP EINV   INV
0950  20542 066553        JMP EZER   ZER
0951  20543 066550        JMP ECON   CON
0952  20544 066562        JMP EIDN   IDN
0953*
0954  20545 066671        JMP EADD   +
0955  20546 066673        JMP ESUB   -
0956  20547 066760        JMP EMUL   *
```

#01 MATRIX BLOCK

```
0958* EXECUTE ZER AND CON
0959*
0960  20550 063622 ECON  JSM REDIM       REDIMENSION IF NECESSARY
0961  20551 024004       LDB B10K
0962  20552 066555       JMP *+3
0963  20553 063622 EZER  JSM REDIM       REDIMENSION IF NECESSARY
0964  20554 074742       SBR 16
0965  20555 061256       JSM CLAR2       SET UP AR2
0966  20556 035755       STB AR2+1
0967  20557 063563       JSM COMP2       SET UP BOUNDS INFO
0968  20560 062611       JSM CMR1        DO THE TRANSFERS
0969  20561 164172       JMP XEC4A,I     DONE
0970*


0971* EXECUTE IDN
0972*
0973  20562 063622 EIDN  JSM REDIM       REDIMENSION IF NECESSARY
0974  20563 061256       JSM CLAR2       SET UP AR2
0975  20564 063563       JSM COMP2       SET UP BOUNDS INFO
0976  20565 011653       CPA I3          CHECK FOR SQUARE MATRIX
0977  20566 066570       JMP *+2
0978  20567 067120       JMP MERR5
0979  20570 062611       JSM CMR1        TRANSFER THE ZEROS
0980  20571 024004       LDB B10K
0981  20572 035755       STB AR2+1
0982  20573 020012       LDA .1          TRANSFER THE ONES
0983  20574 031646       STA T1
0984  20575 031652       STA T2
0985  20576 062464 EID3  JSM STASH
0986  20577 021646       LDA T1
0987  20600 011653       CPA I3
0988  20601 164172       JMP XEC4A,I     DONE
0989  20602 045646       ISZ T1
0990  20603 045652       ISZ T2
0991  20604 066576       JMP EID3
0992*


0993* EXECUTE REPLACEMENT
0994*
0995  20605 021642 EMRP  LDA B1+2        CHECK FOR COMPATIBLE DIMENSIONS
0996  20606 063564       JSM COMP1
0997  20607 062613       JSM CMR4        DO THE TRANSFERS
0998  20610 164172       JMP XEC4A,I
1000*


1001* COMMON ROUTINE TO TRANSFER B3 + AR2
1002*
1003  20611 022641 CMR1  LDA ERPX
1004  20612 066617       JMP COMM
1005*


1006* COMMON ROUTINE TO TRANSFER B3 + B1
1007*
1008  20613 021641 CMR4  LDA B1+1        CHECK FOR REDUNDANT TRANSFER
1009  20614 011644       CPA B3+1
1010  20615 170402       RET
1011  20616 022640       LDA ETRX
1012*


1013* COMMON ROUTINE MAT EXECUTION (EXCEPT MULTIPLY, INVERSE, & PRINT)
1014*
1015*     LDA PAGE ADDRESS OF SOME ROUTINE
1016*     JSM COMM
1017*
1018  20617 031655 COMM  STA T5
1019  20620 020012       LDA .1          INITIALIZE OUTER LOOP
1020  20621 031646       STA T1
1021  20622 020012 COM1  LDA .1          INITIALIZE INNER LOOP
1022  20623 031652       STA T2
1023  20624 021655 COM2  LDA T5
1024  20625 070016       EXA             JSM TO DESIGNATED ROUTINE
1025  20626 021652       LDA T2
1026  20627 011654       CPA T4
1027  20630 066633       JMP *+3
```

```
        #01  MATRIX BLOCK

1028  20631 045652        ISZ T2
1029  20632 066624        JMP COM2
1030  20633 021646        LDA T1
1031  20634 011653        CPA T3
1032  20635 170402        RET
1033  20636 045646        ISZ T1
1034  20637 066622        JMP COM1
1035*
1036*
1037  20640 062642 ETRX   JSM *+2
1038  20641 062643 ERPX   JSM *+2
1039  20642 063600        JSM GETI1
1040  20643 066464        JMP STASH
1042*


1043* EXECUTE SCALAR MULTIPLY
1044*
1045  20644 061217 ESMY   JSM FETCH       EVALUATE EXPRESSION
1046  20645 045472        ISZ TEMPS
1047  20646 121472        LDA TEMPS,I
1048  20647 063541        JSM INFO1       ESTABLISH INFORMATION VECTOR
1049  20650 020214        LDA AR2A
1050  20651 027734        LDB TF1A
1051  20652 170004        XFR
1052  20653 021642        LDA B1+2        CHECK FOR COMPATIBLE DIMENSIONS
1053  20654 063564        JSM COMP1
1054  20655 022660        LDA ESMX        DO THE SCALAR MULTIPLY
1055  20656 062617        JSM COMM
1056  20657 164172        JMP XEC4A,I
1057*
1058  20660 062661 ESMX   JSM *+1
1059  20661 063600        JSM GETI1
1060  20662 020214        LDA AR2A
1061  20663 027735        LDB TF2A
1062  20664 170004        XFR
1063  20665 023734        LDA TF1A
1064  20666 027735        LDB TF2A
1065  20667 160220        JSM .FMPA,I
1066  20670 066464        JMP STASH
1067*


1068* EXECUTE ADD AND SUBTRACT
1069*
1070  20671 020216 EADD   LDA .FADA
1071  20672 066674        JMP *+2
1072  20673 020217 ESUB   LDA .FSBA
1073  20674 031656        STA T6
1074  20675 021642        LDA B1+2        CHECK FOR COMPATIBLE DIMENSIONS
1075  20676 011651        CPA B2+2
1076  20677 066701        JMP *+2
1077  20700 067547        JMP MERR8
1078  20701 063564        JSM COMP1
1079  20702 022705        LDA EASX        DO THE ADD OR SUBTRACT
1080  20703 062617        JSM COMM
1081  20704 164172        JMP XEC4A,I
1082*
1083  20705 062706 EASX   JSM *+1
1084  20706 063600        JSM GETI1
1085  20707 020214        LDA AR2A
1086  20710 027734        LDB TF1A
1087  20711 170004        XFR
1088  20712 063602        JSM GETI2
1089  20713 020214        LDA AR2A
1090  20714 027735        LDB TF2A
1091  20715 170004        XFR
1092  20716 023734        LDA TF1A
1093  20717 027735        LDB TF2A
1094  20720 161656        JSM T6,I
1095  20721 066464        JMP STASH
1097*


1098* EXECUTE TRANSPOSE
1099*
1100*     DOES A(I,J) ← A(J,I)
1101*
1102  20722 121472 ETRN   LDA TEMPS,I
1103  20723 063541        JSM INFO1       ESTABLISH INFORMATION VECTOR
1104  20724 063563        JSM COMP2       CHECK FOR COMPATIBLE DIMENSIONS
1105  20725 031656        STA T6
1106  20726 035655        STB T5
1107  20727 021642        LDA B1+2
1108  20730 025642        LDB B1+2
1109  20731 063565        JSM COMPR
1110  20732 011655        CPA T5
1111  20733 066735        JMP *+2
```

```
        #91  MATRIX BLOCK

112  20734 067567        JMP MERR8
113  20735 015656        CPB T6
114  20736 066740        JMP *+2
115  20737 067567        JMP MERR8
116  20740 022743        LDA ETNX        DO THE TRANSPOSE
117  20741 062617        JSM COMM
118  20742 164172        JMP XEC4A,I
119*
120  20743 062744 ETNX   JSM *+1
121  20744 063600        JSM GETI1
122  20745 062747        JSM REVRS
123  20746 062464        JSM STASH
124*
125  20747 021646 REVRS  LDA T1          SWAP INDEX POINTERS
126  20750 025652        LDB T2
127  20751 031652        STA T2
128  20752 035646        STB T1
129  20753 021653        LDA T3
130  20754 025654        LDB T4
131  20755 031654        STA T4
132  20756 035653        STB T3
133  20757 170402        RET
135*
136* EXECUTE MULTIPLY
137*
138* B3(I,K) = SUM B1(I,P)*B2(P,K) FOR P = 1 TO N
139*
140  20760 021642 EMUL   LDA B1+2        CHECK FOR COMPATIBLE DIMENSIONS
141  20761 025642        LDB B1+2
142  20762 063565        JSM COMPR       B1 IS (M,N)
143  20763 031656        STA T6          T6=N
144  20764 035661        STB T9          T9=M
145  20765 021651        LDA B2+2
146  20766 025651        LDB B2+2
147  20767 063565        JSM COMPR       B2 IS (N,L)
148  20770 015656        CPB T6
149  20771 066773        JMP *+2
150  20772 067567        JMP MERR8       N'S NOT SAME
151  20773 031656        STA T6          T6=L
152  20774 035660        STB T8          SAVE N
153  20775 063563        JSM COMP2       B3 IS (M,L)
154  20776 015661        CPB T9
155  20777 067001        JMP *+2
156  21000 067567        JMP MERR8
157  21001 011656        CPA T6
158  21002 067004        JMP *+2
159  21003 067567        JMP MERR8
160  21004 061256        JSM CLAR2       ZERO B3
161  21005 062611        JSM CMRI
162  21006 021660        LDA T8
163  21007 031653        STA T3          T3=N
164  21010 020012        LDA .1          DO THE MULTIPLY
165  21011 031656        STA T6          T6=I
166  21012 020012 EML3   LDA .1
167  21013 031652        STA T2          T2=K
168  21014 020012 EML4   LDA .1
169  21015 031646        STA T1          T1=P
170  21016 063602 EML5   JSM GETI2       B2(P,K)
171  21017 020214        LDA AR2A
172  21020 027735        LDB TF2A
173  21021 170004        XFR
174  21022 021653        LDA T3          SWAP N,L
175  21023 025654        LDB T4
176  21024 031654        STA T4          T4=N
177  21025 035653        STB T3          T3=L
178  21026 021656        LDA T6          PERMUTE P,K,I
179  21027 025646        LDB T1
180  21030 031646        STA T1          T1=I
181  21031 021652        LDA T2
182  21032 035652        STB T2          T2=P
183  21033 031656        STA T6          T6=K
184  21034 063600        JSM GETI1       B1(I,P)
185  21035 020214        LDA AR2A
186  21036 027734        LDB TF1A
187  21037 170004        XFR
188  21040 023734        LDA TF1A
189  21041 027735        LDB TF2A
190  21042 160220        JSM .FMPA,I     CALL TO MULTIPLY
191  21043 020214        LDA AR2A
192  21044 027735        LDB TF2A
193  21045 170004        XFR
194  21046 021653        LDA T3          SWAP L,N
195  21047 025654        LDB T4
196  21050 031654        STA T4          T4=L
197  21051 035653        STB T3          T3=N
198  21052 021652        LDA T2          SWAP P,K
199  21053 025656        LDB T6
200  21054 031656        STA T6          T6=P
201  21055 035652        STB T2          T2=K
202  21056 063604        JSM GETI3       B3(I,K)
```

```
        #01  MATRIX BLOCK

1203  21057 020214        LDA AR2A
1204  21060 027734        LDB TF1A
1205  21061 170004        XFR
1206  21062 027734        LDA TF1A
1207  21063 027735        LDB TF2A
1208  21064 160216        JSM .FADA,I      CALL TO ADD
1209  21065 062464        JSM STASH
1210  21066 021646        LDA T1           SWAP I,P
1211  21067 025656        LDB T6
1212  21070 031656        STA T6           T6=I
1213  21071 035646        STB T1           T1=P
1214  21072 115653        CPB T3           CHECK P
1215  21073 067076        JMP *+3
1216  21074 045646        ISZ T1
1217  21075 067016        JMP EML5
1218  21076 021652        LDA T2
1219  21077 011654        CPA T4           CHECK K
1220  21100 067103        JMP *+3
1221  21101 045652        ISZ T2
1222  21102 067014        JMP EML4
1223  21103 021656        LDA T6
1224  21104 011661        CPA T9           CHECK I
1225  21105 164172        JMP XEC4A,I      DONE
1226  21106 045656        ISZ T6
1227  21107 067212        JMP EML3
1229*
1230* EXECUTE INVERSE
1231*
1232*     USES GAUSS-JORDAN WITH ROW INTERCHANGE
1233*
1234  21110 021412 EINV   LDA LSTPT        SET UP FOR INVERSE EXECUTION
1235  21111 031656        STA T6
1236  21112 121472        LDA TEMPS,I
1237  21113 063541        JSM INFO1        ESTABLISH INFORMATION VECTOR
1238  21114 021642        LDA B1+2         CHECK FOR COMPATIBLE DIMENSIONS
1239  21115 063564        JSM COMP1
1240  21116 011653        CPA T3
1241  21117 067122        JMP *+3
1242  21120 160205 MERR5  JSM AERRA,I      MATRIX NOT SQUARE
1243  21121 177676        DEC -66
1244*
1245  21122 063127        JSM XIN2         CALL DET/INV EXECUTION
1246  21123 067736        JMP MER10        SINGULAR MATRIX
1247  21124 021656        LDA T6
1248  21125 031412        STA LSTPT        RESTORE LSTPT
1249  21126 164172        JMP XEC4A,I
1250*
1251*
1252  21127 021412 XIN2   LDA LSTPT        RESERVE INTERCHANGE VECTOR
1253  21130 031676        STA AUL
1254  21131 001654        ADA T4
1255  21132 031412        STA LSTPT        SAVE NEW LSTPT
1256  21133 060554        JSM BHSTP+1      CHECK FOR OVERLAP
1257  21134 062613        JSM CMR4         COPY B1 INTO B3
1258  21135 020020        LDA ONE          SET DEL = 1.0
1259  21136 027731        LDB DELA
1260  21137 170034        XFR
1261  21140 020012        LDA .1           DO THE INVERSE
1262  21141 031653 EIN1   STA T3           T3=K
1263  21142 070137        LDB A
1264  21143 005676        ADB AUL
1265  21144 074557        STA B,I          L(K)=K
1266  21145 031646        STA T1
1267  21146 031652        STA T2
1268  21147 063634        JSM GETI3
1269  21150 020214        LDA AR2A
1270  21151 027732        LDB BIGA         BIG=A(K,K)
1271  21152 170004        XFR
1273  21153 023732 EIN3   LDA BIGA         SEARCH FOR LARGEST ELEMENT
1274  21154 027734        LDB TF1A
1275  21155 170004        XFR
1276  21156 021630        LDA TF1
1277  21157 070131        SLA *+2,C        FORM ABS(BIG)
1278  21160 031630        STA TF1
1279  21161 063604        JSM GETI3
1280  21162 020214        LDA AR2A
1281  21163 027735        LDB TF2A
1282  21164 170004        XFR
1283  21165 020214        LDA AR2A
1284  21166 027733        LDB HLDA
1285  21167 170004        XFR
1286  21170 021634        LDA TF2
1287  21171 070131        SLA *+2,C        FORM ABS(A(I,K))
1288  21172 031634        STA TF2
1289  21173 023734        LDA TF1A
1290  21174 027735        LDB TF2A
1291  21175 160217        JSM .FSBA,I      ABS(BIG)-ABS(A(I,K))
1292  21176 021754        LDA AR2
1293  21177 070411        SLA *+8
1294  21200 023733        LDA HLDA
```

#01 MATRIX BLOCK

```
1295  21201 #27732        LDB BIGA
1296  21202 170004        XFR             BIG=A(I,K)
1297  21203 #21646        LDA T1
1298  21204 #25653        LDB T3
1299  21205 #25676        ADB ADL
1300  21206 #74557        STA B,I         L(K)=I
1301  21207 #21646        LDA T1          CHECK I
1302  21210 #11654        CPA T4
1303  21211 #67214        JMP EEIR
1304  21212 #45646        ISZ T1
1305  21213 #67153        JMP EIN3
1307  21214 #25653 EEIR   LDB T3          INTERCHANGE ROWS
1308  21215 #25676        ADB ADL
1309  21216 #74517        LDA B,I
1310  21217 #11653        CPA T3
1311  21220 #67261        JMP DCPV
1312  21221 #31655        STA T5          T5=J
1313  21222 #20012        LDA .1
1314  21223 #31652        STA T2          T2=I
1315  21224 #21653 EIN4   LDA T3
1316  21225 #31646        STA T1
1317  21226 #63604        JSM GETI3
1318  21227 #21755        LDA AR2+1       FORM U-
1319  21230 #70250        SZA *+5
1320  21231 #21754        LDA AR2
1321  21232 #72111        SLA *+2,S
1322  21233 #70071        SLA *+1,C
1323  21234 #31754        STA AR2
1324  21235 #20214        LDA AR2A
1325  21236 #27733        LDB HLDA        HLD=-A(K,I)
1326  21237 170004        XFR
1327  21240 #21655        LDA T5
1328  21241 #31646        STA T1
1329  21242 #63604        JSM GETI3
1330  21243 #21653        LDA T3
1331  21244 #31646        STA T1
1332  21245 #62464        JSM STASH       A(K,I)=A(J,I)
1333  21246 #23733        LDA HLDA
1334  21247 #24214        LDB AR2A
1335  21250 170004        XFR
1336  21251 #21655        LDA T5
1337  21252 #31646        STA T1
1338  21253 #62464        JSM STASH       A(J,I)=HLD
1339  21254 #21652        LDA T2          CHECK I
1340  21255 #11654        CPA T4
1341  21256 #67261        JMP DCPV
1342  21257 #45652        ISZ T2
1343  21260 #67224        JMP EIN4
1345  21261 #23732 DCPV   LDA BIGA        DIVIDE COLUMN BY MINUS PIVOT VALUE
1346  21262 #27734        LDB TF1A
1347  21263 170004        XFR
1348  21264 #21630        LDA TF1
1349  21265 #70131        SLA *+2,C       FORM ABS(BIG)
1350  21266 #31630        STA TF1
1351  21267 #23735        LDA TF2A
1352  21270 170000        CLR
1353  21271 #27740        LDB EPSF
1354  21272 #35634        STB TF2
1355  21273 #24004        LDB B10K
1356  21274 #35635        STB TF2+1       TOLERANCE = 1E-48
1357  21275 #23734        LDA TF1A
1358  21276 #27735        LDB TF2A
1359  21277 160217        JSM .FSBA,I
1360  21300 #21754        LDA AR2
1361  21301 #70211        SLA *+4
1362  21302 #23731        LDA DELA
1363  21303 170000        CLR
1364  21304 170402        RET             SINGULAR MATRIX DETECTED
1365  21305 #21653        LDA T3
1366  21306 #31652        STA T2          T2=K
1367  21307 #20012        LDA .1
1368  21310 #31646        STA T1          T1=I
1369  21311 #21653 EIN6   LDA T3
1370  21312 #11646        CPA T1
1371  21313 #67332        JMP EIN7
1372  21314 #63604        JSM GETI3
1373  21315 #20214        LDA AR2A
1374  21316 #27734        LDB TF1A
1375  21317 170004        XFR
1376  21320 #23734        LDA TF1A
1377  21321 #27732        LDB BIGA
1378  21322 160221        JSM .FDVA,I
1379  21323 #21755        LDA AR2+1       FORM U-
1380  21324 #70250        SZA *+5
1381  21325 #21754        LDA AR2
1382  21326 #72111        SLA *+2,S
1383  21327 #70071        SLA *+1,C
1384  21330 #31754        STA AR2
1385  21331 #62464        JSM STASH       A(I,K)=-A(I,K)/BIG
1386  21332 #21646 EIN7   LDA T1          CHECK I
```

#01 MATRIX BLOCK

```
1387  21333 011654        CPA T4
1388  21334 067337        JMP REDU
1389  21335 045646        ISZ T1
1390  21336 067311        JMP EIN6
1392  21337 020012 REDU    LDA .1          REDUCE MATRIX
1393  21340 031646         STA T1          T1=I
1394  21341 031655         STA T5          T5=I
1395  21342 021653 EIN8    LDA T3
1396  21343 011646         CPA T1
1397  21344 067411         JMP EIN11
1398  21345 031652         STA T2          T2=K
1399  21346 063604         JSM GETI3
1400  21347 020214         LDA AR2A
1401  21350 027733         LDB HLDA        HLD=A(I,K)
1402  21351 170004         XFR
1403  21352 020012         LDA .1
1404  21353 031652         STA T2          T2=J
1405  21354 021653 EIN9    LDA T3
1406  21355 011652         CPA T2
1407  21356 067404         JMP EIN10
1408  21357 031646         STA T1          T1=K
1409  21360 063604         JSM GETI3
1410  21361 020214         LDA AR2A
1411  21362 027734         LDB TF1A
1412  21363 170004         XFR
1413  21364 023733         LDA HLDA
1414  21365 027734         LDB TF1A
1415  21366 160220         JSM .FMPA,I
1416  21367 020214         LDA AR2A
1417  21370 027734         LDB TF1A        HLD*A(K,J)
1418  21371 170004         XFR
1419  21372 021655         LDA T5
1420  21373 031646         STA T1          T1=I
1421  21374 063604         JSM GETI3
1422  21375 020214         LDA AR2A
1423  21376 027735         LDB TF2A
1424  21377 170004         XFR
1425  21400 023734         LDA TF1A
1426  21401 027735         LDB TF2A
1427  21402 160216         JSM .FADA,I
1428  21403 062464         JSM STASH       A(I,J)=HLD*A(K,J)+A(I,J)
1429  21404 021652 EIN10   LDA T2          CHECK J
1430  21405 011654         CPA T4
1431  21406 067411         JMP *+3
1432  21407 045652         ISZ T2
1433  21410 067354         JMP EIN9
1434  21411 021646 EIN11   LDA T1          CHECK I
1435  21412 011654         CPA T4
1436  21413 067417         JMP *+4
1437  21414 045646         ISZ T1
1438  21415 045655         ISZ T5
1439  21416 067342         JMP EIN8
1441  21417 021653         LDA T3          DIVIDE ROW BY PIVOT
1442  21420 031646         STA T1          T1=K
1443  21421 020012         LDA .1
1444  21422 031652         STA T2          T2=J
1445  21423 021653 EIN12   LDA T3
1446  21424 011652         CPA T2
1447  21425 067436         JMP EIN13
1448  21426 063604         JSM GETI3
1449  21427 020214         LDA AR2A
1450  21430 027734         LDB TF1A
1451  21431 170004         XFR
1452  21432 023734         LDA TF1A
1453  21433 027732         LDB BIGA
1454  21434 160221         JSM .FDVA,I
1455  21435 062464         JSM STASH       A(K,J)=A(K,J)/BIG
1456  21436 021652 EIN13   LDA T2          CHECK J
1457  21437 011654         CPA T4
1458  21440 067443         JMP *+3
1459  21441 045652         ISZ T2
1460  21442 067423         JMP EIN12
1461  21443 023731         LDA DELA        PRODUCT OF PIVOTS
1462  21444 027732         LDB BIGA
1463  21445 160220         JSM .FMPA,I
1464  21446 020214         LDA AR2A
1465  21447 027731         LDB DELA
1466  21450 170004         XFR
1467  21451 023734         LDA TF1A
1468  21452 170030         CLR
1469  21453 024004         LDB B10K
1470  21454 035631         STB TF1+1
1471  21455 023734         LDA TF1A
1472  21456 027732         LDB BIGA
1473  21457 160221         JSM .FDVA,I
1474  21460 021653         LDA T3
1475  21461 031646         STA T1
1476  21462 031652         STA T2
1477  21463 062464         JSM STASH       A(K,K)=1./BIG
1478  21464 021653         LDA T3          CHECK K
1479  21465 011654         CPA T4
```

```
          #01  MATRIX BLOCK

1480  21466 067471        JMP *+3
1481  21467 070070        STA *+1
1482  21470 067141        JMP EIN1
1484  21471 055653 EIN14  DSZ T3          FINAL ROW INTERCHANGE
1485  21472 067474        JMP **2
1486  21473 064357        JMP RETN2       DONE
1487  21474 025653        LDB T3
1488  21475 005676        ADB ADL
1489  21476 074517        LDA B,I
1490  21477 011653        CPA T3
1491  21500 067471        JMP EIN14
1492  21501 031655        STA T5          T5=L(K)
1493  21502 020012        LDA .1
1494  21503 031646        STA T1          T1=J
1495  21504 021653 EIN15  LDA T3
1496  21505 031652        STA T2          T2=K
1497  21506 063604        JSM GETI3
1498  21507 020214        LDA AR2A
1499  21510 027733        LDB HLDA        HLD=A(J,K)
1500  21511 170004        XFR
1501  21512 021655        LDA T5
1502  21513 031652        STA T2          T2=I
1503  21514 063604        JSM GETI3
1504  21515 021755        LDA AR2,1       FORM U-
1505  21516 070250        SZA **5
1506  21517 021754        LDA AR2
1507  21520 072111        SLA **2,S
1508  21521 070071        SLA **1,C
1509  21522 031754        STA AR2
1510  21523 021653        LDA T3
1511  21524 031652        STA T2          T2=K
1512  21525 062464        JSM STASH       A(J,K)=-A(J,I)
1513  21526 021655        LDA T5
1514  21527 031652        STA T2          T2=I
1515  21530 027733        LDA HLDA
1516  21531 024214        LDB AR2A
1517  21532 170004        XFR
1518  21533 062464        JSM STASH       A(J,I)=HLD
1519  21534 021646        LDA T1          CHECK J
1520  21535 011654        CPA T4
1521  21536 067471        JMP EIN14
1522  21537 045646        ISZ T1
1523  21540 067504        JMP EIN15
1525*
1526* SUBROUTINE TO EXTRACT ARRAY INFORMATION
1527*
1528*     LDA NAME INFORMATION
1529*     LDB INFORMATION VECTOR ADDRESS
1530*     JSM INFOV
1531*
1532  21541 027726 INFO1  LDB B1A         ALTERNATE ENTRY FOR B1
1533  21542 035657 INFOV  STB T7          TEMPORARY SAVE
1534  21543 050115        AND B1777
1535  21544 160177        JSM SSYMA,I     SEARCH SYMBOL TABLE
1536  21545 164340        JMP E40+1,I     ERROR, NOT FOUND (UNDEFINED OPERAND)
1537  21546 074517        LDA B,I         GET TYPE INFORMATION
1538  21547 050040        AND .31         ISOLATE TYPE-FIELD
1539  21550 070042        SAR 2           LOOK AT SIZE
1540  21551 131657        STA T7,I         AND SAVE IT
1541  21552 045657        ISZ T7
1542  21553 004132        ADB M1
1543  21554 074517        LDA B,I         GET VALUE TABLE ADDRESS
1544  21555 131657        STA T7,I         AND SAVE AS BASE ADDRESS
1545  21556 045657        ISZ T7
1546  21557 000132        ADA M1
1547  21560 070517        LDA A,I         GET ACTUAL DIMENSIONS
1548  21561 131657        STA T7,I         AND SAVE THEM
1549  21562 170402        RET
1550*
1551* SUBROUTINE TO EXTRACT AND COMPARE DIMENSIONS OF TWO MATRICES
1552*
1553*     LDA DIMENSION INFORMATION #1
1554*     LDB DIMENSION INFORMATION #2
1555*     JSM COMPR
1556*
1557  21563 021645 COMP2  LDA B3+2
1558  21564 025645 COMP1  LDB B3+2
1559  21565 074057 COMPR  CPA B           EQUAL?
1560  21566 067571        JMP **3         YES
1561  21567 164205 MERRA  JSM AERRA,I     INCOMPATIBLE DIMENSIONS
1562  21570 177673        DEC -59
1563  21571 074342        SBR B
1564  21572 074070        SIB **1
1565  21573 035653        STB T3          SAVE # OF ROWS
1566  21574 050105        AND B377
1567  21575 070070        STA **1
1568  21576 031654        STA T4          SAVE # OF COLUMNS
1569  21577 170402        RET
```

#01 MATRIX BLOCK

```
1571*
1572* SUBROUTINE TO OBTAIN AN ARRAY ELEMENT
1573*
1574* ELEMENT PLACED IN AR2
1575*
1576* ROW & COLUMN VALUES ARE IN T1 AND T2
1577*
1578*     JSM GETI(N)
1579*
1580  21600 027726 GETI1 LDB B1A
1581  21601 067605       JMP *+4
1582  21602 027727 GETI2 LDB B2A
1583  21603 067605       JMP *+2
1584  21604 027730 GETI3 LDB B3A
1585  21605 063705       JSM WHERE
1586  21606 074517       LDA B,I          CHECK IF UNDEFINED
1587  21607 010062       CPA FLGBT
1588  21610 164337       JMP E40,I        ERROR, NOT DEFINED (NO VALUE ASSIGNED)
1589  21611 121657       LDA T7,I         FIND TYPE OF ARRAY
1590  21612 072110       RZA *+2
1591  21613 067620       JMP GTRL         REAL
1592  21614 010012       CPA .1
1593  21615 164231       JMP SPLFA,I      SPLIT
1594  21616 074537       LDB B,I
1595  21617 164227       JMP FXFLA,I      INTEGER
1596  21620 074117 GTRL  LDA B
1597  21621 064445       JMP XFAR2+1
1599*
1600* SUBROUTINE TO REDIMENSION AN ARRAY
1601*
1602  21622 021472 REDIM LDA TEMPS
1603  21623 011421       CPA PRADD
1604  21624 170402       RET
1605  21625 063673 REDIR JSM MCKS         GET ROW DIMENSION
1606  21626 035646       STB T1           SAVE T1
1607  21627 024012       LDB .1           LOAD DEFAULT COLUMN COUNT
1608  21630 121472       LDA TEMPS,I      HOW MANY DIMENSIONS?
1609  21631 050065       AND OPMSK
1610  21632 010121       CPA B4000
1611  21633 063673       JSM MCKS
1612  21634 045472       ISZ TEMPS
1613  21635 035652       STB T2           SAVE COLUMN DIMENSION
1614  21636 021646       LDA T1           RECALL ROW DIMENSION
1615  21637 061201       JSM IMPY
1616  21640 031657       STA T7           SAVE NEW PRODUCT
1617  21641 025644       LDB B3+1         RECALL FORMAL DIMENSIONS
1618  21642 004133       ADB M2
1619  21643 074517       LDA B,I
1620  21644 070137       LDB A
1621  21645 050105       AND B377
1622  21646 074342       SBR B
1623  21647 070070       STA *+1
1624  21650 074070       STB *+1
1625  21651 061201       JSM IMPY         RETURNS WITH FORMAL PRODUCT
1626  21652 070056       CMA
1627  21653 001657       ADA T7           COMPARE WITH NEW PRODUCT
1628  21654 070112       SAM *+2          NEW>OLD?
1629  21655 067703       JMP MERR7        YES, ERROR
1630  21656 021646       LDA T1           NO, ESTABLISH NEW ACTUAL DIMENSIONS
1631  21657 025652       LDB T2
1632  21660 031653       STA T3           FIRST INTO T3 & T4
1633  21661 035654       STB T4
1634  21662 000132       ADA M1
1635  21663 004132       ADB M1
1636  21664 070404       SAL B            THEN PACK THE DIMENSIONS
1637  21665 074217       IOR B
1638  21666 025644       LDB B3+1         GET VALUE TABLE ADDRESS
1639  21667 004132       ADB M1
1640  21670 074557       STA B,I          STASH PACKED DIMENSIONS
1641  21671 031645       STA B3+2
1642  21672 170402       RET
1643*
1644*
1645  21673 061217 MCKS  JSM FETCH        EVALUATE SUBSCRIPT EXPRESSION
1646  21674 160225       JSM FLTRA,I      CONVERT TO ROUNDED INTEGER
1647  21675 067703       JMP MERR7        INTEGER OVERFLOW
1648  21676 074117       LDA B
1649  21677 000132       ADA M1           <1?
1650  21700 070152       SAM *+3          YES
1651  21701 000157       ADA M256         >256?
1652  21702 071412       SAM MCKS-1
1653  21703 160205 MERR7 JSM AERRA,I      NEW DIMENSIONS IMPOSSIBLE
1654  21704 177675       DEC -67
1656*
1657* SUBROUTINE TO COMPUTE ADDRESS OF A MATRIX ELEMENT
1658*
1659* ROW & COLUMN VALUES ARE IN T1 AND T2 RESPECTIVELY
1660*
1661*     LDB INFORMATION VECTOR ADDRESS
1662*     JSM WHERE
1663*
```

#01 MATRIX BLOCK

```
1664* RETURNS WITH ADDRESS IN (A) & (B)
1665*
1666* ADDRESS = BASE+SIZE*(T4*(T1-1)+(T2-1))
1667*
1668  21705 035657 WHERE STB T7          TEMPORARY SAVE
1669  21706 021646       LDA T1
1670  21707 000132       ADA M1
1671  21710 025654       LDB T4
1672  21711 061201       JSM IMPY        RETURNS WITH T4*(T1-1)
1673  21712 001652       ADA T2
1674  21713 000132       ADA M1
1675  21714 125657       LDB T7,I
1676  21715 076110       MZB **2
1677  21716 070704       SAL 4
1678  21717 014012       CPB .1
1679  21720 070744       SAL 1
1680  21721 025657       LDB T7
1681  21722 074070       STB **1
1682  21723 074417       ADA B*I
1683  21724 070137       LDB A
1684  21725 170402       RET
1686*
1687* MATRIX CONSTANTS
1688*
1689  21726 001640 B1A   DEF B1
1690  21727 001647 B2A   DEF B2
1691  21730 001643 B3A   DEF B3
1692  21731 001662 DELA  DEF DEL
1693  21732 001666 BIGA  DEF BIG
1694  21733 001672 HLDA  DEF HLD
1695  21734 001630 TF1A  DEF TF1
1696  21735 001634 TF2A  DEF TF2
1697*
1698  21736 160205 MER1M JSM AERRA,I     SINGULAR MATRIX
1699  21737 177674       DEC -68
1700*
1701  21740 150000 EPSF  OCT 150000      EXPONENT FOR SINGULARITY TOLERANCE
1702  00025      LEFT  EQU 21766B-*
1703*
1704*
1705  01640              ORG 1640B
1706*
1707  01640 000000 B1    BSS 3           B1 & B3 MUST COME FIRST
1708  01643 000000 B3    BSS 3
1709  01646 000000 T1    NOP
1710  01647 000000 B2    BSS 3
1711  01652 000000 T2    NOP
1712  01653 000000 T3    NOP
1713  01654 000000 T4    NOP
1714  01655 000000 T5    NOP
1715  01656 000000 T6    NOP
1716  01657 000000 T7    NOP
1717  01660 000000 T8    NOP
1718  01661 000000 T9    NOP
1719  01662 000000 DEL   BSS 4
1720  01666 000000 BIG   BSS 4
1721  01672 000000 HLD   BSS 4
1722  01676 000000 ADL   NOP
1723*
1724  01630              ORG 1630B
1725*
1726  01630 000000 TF1   BSS 4
1727  01634 000000 TF2   BSS 4
1728*
1729  01640 070053       SAP **+1        CONVENIENT "NOP" FOR REFERENCE
1731  21766              ORG 21766B
1732*
1733  21766 176455       ABS EMATR-*     MAT READ EXECUTION
1734  21767 176347       ABS EMATP-*     MAT PRINT EXECUTION
1735  21770 176500       ABS EREDM-*     REDIM EXECUTION
1736  21771 176510       ABS EMATX-*     MAT ARITHMETIC EXECUTION
1737  21772 000000       NOP             ** CHECKSUM WORD **
1738*
1739  21773 176030       ABS NFOPT-*     NON-FORMULA OPERATOR TABLE
1740  21774 176041       ABS FUNCT-*     FUNCTION-NAME TABLE
1741  21775 176042       ABS BN777-*     COMMAND-NAME TABLE (NULL TABLE)
1742  21776 176006       ABS SYSNT-*     STATEMENT-NAME TABLE
1743  21777 006000 OPCOD OCT 6000        MAT OPCODE=3
1744                     END
*  NO ERRORS*
```

CROSS-REFERENCE TABLE

```
.1      0020    0770    0817    0819    0982    1019    1021    1164    1166    1168    1261
        1313    1367    1392    1403    1443    1493    1592    1607    1678
.10     0071    0072
.100    0070
.1000   0069
```

CROSS-REFERENCE TABLE

.11 0035 0043
.12 0036
.13 0039
.15 0048 0083
.16 0049 0084
.1E4 0068
.2 0027
.21 0050
.22 0051 0087
.23 0052 0088
.28 0053
.3 0028 0045 0668 0709 0802 0803
.31 0054 0090 0836 0925 0944 1538
.32 0055
.33 0056
.34 0057 0091
.37 0058
.4 0029
.40 0059
.41 0060
.43 0061
.45 0062
.46 0063
.48 0064
.5 0030
.58 0065 0094
.6 0031 0044
.63 0066
.7 0032 0750
.8 0033
.9 0034
.99 0074
.BUFA 0180
.FADA 0185 1070 1208 1427
.FDVA 0188 1378 1454 1473
.FMPA 0187 1065 1190 1415 1463
.FSBA 0186 1072 1291 1359
.FSR 0190
.LNUM 0314
.NPWR 0492
.SLRF 0181
A 0000 1263 1547 1620 1683
ACTSA 0174 0595 0631 0682
ADL 1722 1253 1264 1299 1308 1488
ADRSS 0327
AERRA 0176 1242 1561 1653 1698
AR1 0411 0182
AR1A 0182
AR2 0413 0183 0966 0981 1292 1318 1320 1323 1360 1379 1381
1384 1504 1506 1509
AR2A 0183 1049 1060 1085 1089 1171 1185 1191 1203 1269 1280
1283 1324 1334 1373 1400 1410 1416 1422 1449 1464 1498
1516
ARR1 0708 0614 0666
ARR1D 0706 0576 0585 0599 0639 0660 0677 0744
ARYAD 0401 0613 0683 0689
ASSOP 0508 0616
B 0000 0775 1265 1300 1309 1489 1537 1543 1559 1586 1594
1596 1619 1637 1640 1648 1682
B1 1707 0758 0766 0808 0809 0995 1008 1052 1074 1107 1108
1140 1141 1238 1689
B140 0095
B1000 0102
B10K 0114 0115 0571 0961 0980 1355 1469
B12K 0116 0117 0206
B133 0096
B140 0097
B1400 0103 0204
B17 0083
B170K 0121
B1740 0104 0203
B1760 0105 0158
B177 0098
B1774 0106
B1775 0107
B1777 0108 0856 0921 1534
B1A 1689 0749 0782 1532 1580
B2 1710 1075 1145 1146 1690
B20 0084
B22 0085
B23 0086
B2400 0110
B26 0087
B26K 0553 0673
B27 0088
B2A 1690 0934 1582
B3 1708 0759 0767 0774 0899 1009 1557 1558 1617 1638 1641
1691
B3000 0111
B36 0089
B37 0090 0626

CROSS-REFERENCE TABLE

B377	0099	1566	1621					
B3A	1691	0874	0897	0907	0917	1594		
B400	0100							
B4000	0112	0162	1610					
B42	0091							
B44	0092							
B52	0093							
B6K	0113	0167						
B72	0094							
B777	0101							
BADDR	0291	0592	0827					
BADRP	0284							
BEND	0198	0199						
BHSTP	0442	1256						
BIG	1720	1693						
BIGA	1693	1270	1273	1295	1345	1377	1453	1462	1472
BLANK	0386							
BN777	0550	1741						
C5	0015							
C6	0011	0114						
C7	0022	0100						
CHRSA	0219							
CLAR2	0489	0965	0974	1160				
CLERA	0160	0821						
CLINE	0316							
CLP1A	0217							
CMP1	1003	0968	0979	1161				
CMP4	1008	0997	1257					
COM1	1021	1034						
COM2	1023	1029						
COMCE	0434	0580	0603					
COMCK	0433	0723						
COMM	1018	0882	1004	1055	1080	1117		
COMMA	0507	0588						
COMP1	1558	0760	0996	1053	1078	1239		
COMP2	1557	0880	0967	0975	1104	1153		
COMPR	1559	0810	1109	1142	1147			
CONST	0482							
CPLAC	0331							
CRCKA	0228							
CRLF	0495							
CPLF1	0496	0843	0844	0851	0860			
CRTN	0037	0038	0039					
CTYPE	0330							
DCOMA	0210							
DCPV	1345	1311	1341					
DEL	1719	1692						
DELA	1692	0785	1259	1362	1461	1465		
DELSA	0175							
DETT	0736	0513						
DETX	0749	0737						
DFLAG	0388							
DIGCK	0461							
DPERR	0043							
DROST	0317							
DSTRT	0303							
E	0080							
E1	0235							
E10	0244	0589						
E15	0249	0602						
E20	0254							
E25	0259	0611	0707					
E30	0264	0617	0649	0659	0671	0686	0693	
E35	0269							
E40	0274	1536	1588					
E45	0279							
E5	0239							
E50	0284							
EADD	1070	0954						
EASX	1083	1079						
ECALA	0215							
ECON	0960	0951						
EEIR	1307	1303						
EID3	0985	0991						
EIDN	0973	0952						
EIN1	1262	1482						
EIN10	1429	1407						
EIN11	1434	1397						
EIN12	1445	1460						
EIN13	1456	1447						
EIN14	1484	1491	1521					
EIN15	1495	1523						
EIN3	1273	1305						
EIN4	1315	1343						
EIN6	1369	1390						
EIN7	1386	1371						
EIN8	1395	1439						
EIN9	1405	1433						
EINV	1234	0949						
ELINA	0169	0834						

CROSS-REFERENCE TABLE

EMATP	0793	1734									
EMATR	0872	0886	1733								
EMATX	0915	1736									
EML3	1166	1227									
EML4	1168	1222									
EML5	1170	1217									
EMRP	0995	0932									
EMRX	0888	0881									
EMUL	1140	0956									
ENDS	0478										
ENTEA	0218										
ENTFA	0189										
EOL	0838	0678									
EOLCK	0481	0579	0586	0667	0731						
EOST	0479	0642	0661	0687							
EPSE	1701	1353									
EREDM	0905	0913	1735								
ERPX	1038	1003									
ERRA	0177										
ESMX	1058	1054									
ESMY	1045	0927									
ESUB	1072	0955									
ETNX	1120	1116									
ETRN	1102	0948									
ETRX	1037	1011									
EXP	0399	0400									
EXPOP	0118										
EZER	0963	0950									
F	00A1										
FDAAA	0222	0889									
FETCH	0486	1045	1645								
FILLA	0201	0833									
FIRST	0365										
FIX	0499										
FLGBT	0073	1587									
FLOAT	0384	0824									
FLTFA	0193										
FLTRA	0192	1646									
FLTSA	0195										
FN	0148										
FND	0460										
FNDIA	0224										
FNDAA	0223										
FNDPS	0474										
FOPBS	0206										
FORMA	0172										
FORME	0336										
FORMS	0335										
FORMT	0334										
FPAGE	0449										
FROMA	0122										
FSCA	0173	0655	0722	0725							
FSCE2	0439										
FSCE4	0444										
FTMP1	0337										
FTMP2	0338										
FUNCT	0547	1740									
FWAM	0157										
FWUP	0296										
FXFLA	0194	1595									
GETAD	0459										
GETCH	0469										
GETI1	1580	0822	1039	1059	1084	1121	1184				
GETI2	1582	1088	1170								
GETI3	1584	1202	1268	1279	1317	1329	1372	1399	1409	1421	1448
	1497	1503									
GETSK	0473	0590	0630								
GFLAG	0402										
GFRST	0472										
GNEXT	0471	0565	0618	0637	0742						
GPARM	0422										
GSIGN	0484										
GTCON	0498										
GTMP	0470										
GTOPP	0464										
GTRL	1596	1591									
GTSTA	0213										
HFLG1	0321										
HLD	1721	1694									
HLDA	1694	1284	1294	1325	1333	1401	1413	1499	1515		
HSTPT	0302										
I	0067										
ICNT	0420										
ID	0513	0554	0554	0696	0696						
IMPY	0485	0765	1615	1625	1672						
INF	0077										
INFO1	1532	0757	0797	0929	1048	1103	1237				
INFOV	1533	0875	0908	0918	0935						
INITS	0454										
INREA	0226										

CROSS-REFERENCE TABLE

INTCK	0457										
INTEA	0178										
INTFL	0075										
INTGR	0400										
IOBUF	0357										
IOHEA	0229										
IOSTA	0225										
KEYFG	0324										
L2T1	0345										
L2T2	0346										
LASTN	0396										
LBOP	0119	0720	0878								
LDFF	0487										
LEFT	1702										
LFTCK	0462	0647									
LHSC	0683	0643									
LIMIT	0463										
LISTA	0167										
LN72	0867	0814									
LNFD	0072										
LNINC	0315										
LNUM	0455										
LOCL1	0343										
LOCL2	0344										
LPCK	0435	0718									
LPCKE	0436	0638	0654	0743							
LPOP	0120										
LPOMA	0123										
LSTAK	0299										
LSTPT	0301	0752	0772	0780	1234	1248	1252	1255			
LTR	0477	0610	0645	0706							
LWAM	0295										
M1	0124	0691	0812	1542	1546	1634	1635	1639	1649	1670	1674
M10	0131										
M100	0144										
M1024	0146										
M11	0132										
M12	0133										
M13	0134										
M14	0135										
M15	0136										
M2	0125	1618									
M23	0137										
M25	0138										
M256	0145	1651									
M3	0126	0628	0638								
M32	0139										
M36	0140										
M4	0127										
M48	0141										
M5	0128										
M6	0129										
M64	0142										
M7	0552	0779									
M8	0143										
M9	0130										
MASK1	0153										
MASK5	0154										
MATJ	0945	0939									
MATS	0559	0516									
MATS0	0585	0572	0593								
MATS1	0576	0581									
MATS2	0609	0569									
MATS3	0637	0629									
MATS4	0654	0646									
MATS5	0665	0648									
MATS6	0676	0694									
MATS7	0689	0674									
MATS8	0645	0623									
MATX0	0718	0577	0600	0632							
MATX1	0941	0926									
MAX	0929	0922									
MAXIN	0294										
MAXSN	0398										
MBIN1	0147										
MBIN2	0404										
MBOX1	0405										
MCKS	0349										
MEMA	1645	1605	1611	1652							
MEM0	0208										
MEM8	1698	1246									
MER8	0443										
MERS	1242	0763	0578								
MER7	1653	1629	1647								
MERR	1561	1077	1112	1115	1150	1156	1159				
MINIT	0453										
MONE	0305	0736	0793	0825	0864						
MOVST	0788	0751									
MSELG	0387										
MTP1	0819	0850									
MTP2	0822	0842	0861								

CROSS-REFERENCE TABLE

MTP3	0795	0858								
MTPX	0863	0837	0854	0857						
MVTOH	0475									
NDTSP	0380									
NDWRT	0494									
NEXTI	0448									
NEXTA	0465									
NEXTL	0447									
NFOPA	0554	0620								
NFOPT	0536	0554	0554	1739						
NMOTA	0200	0826								
NOFOF	0480	0633	0680							
NPRIN	0381									
NPWRT	0493									
NUM	0394	0395	0559							
NUMCA	0211									
NUMOA	0212									
NXTDT	0304									
NXTST	0403									
ORLNK	0466									
ODGIT	0490									
ONE	0045	1258								
ONEXT	0467									
OPCHA	0171									
OPCOD	1743	0739								
OPDMK	0079									
OPMSK	0076	0877	1609							
OPTYP	0395									
OTPTR	0293									
OUTCR	0468									
OUTIA	0179									
OVCHA	0216									
OVFW	0860	0829								
PARMA	0306									
PARMN	0307									
PBPTD	0313									
PBPTR	0312									
PBUFF	0311									
PBUFO	0312									
PEND	0353	0815	0828							
PEXMA	0165									
PFLAG	0389	0676	0681							
PGINM	0476									
PGINT	0456									
PI72	0044									
PLACE	0342	0613								
PLUS	0346	0670								
PMODE	0329									
PREF	0364									
PSAVD	0348	0845	0853	0884	0911	0931	1603			
PSFLG	0332	0841								
PTCHA	0230									
PTOP	0129	0161								
PUPTR	0536	0587	0588	0589	0510					
PWOP	117	0728								
PWORK	0358	0867	0867							
RCOUT	0331									
RDYDA	0166									
RECTP	1642	0960	0963	0973						
RECTU	1645	0879	0909							
RECS	0596	0815	0604							
RECU	1592	1388								
REED	0161									
RES	0415									
RESRI	0451									
RESRP	0452									
RET2	0419									
RETN2	0418	0719	1486							
RETN3	0417	0746								
REVRS	1125	1122								
RPCK	0437									
RPCKE	0438	0640	0656	0726	0745					
RPOP	0115									
RSAVE	0292									
RSLTE	0371									
RSPTR	0416									
RSTAK	0414									
RSTKA	0158									
RSULT	0424									
RUNA	2163									
S	0082									
SALTA	0207									
SAVRP	0450									
SAVEA	0325									
SAVER	0326									
SHADR	0392									
SRPTR	0385	0564	0570	0594	0609	0612	0619	0624	0665	0721	0727
	0729	0730	0740	0741						
SHOUD	0429									
SHUFA	0199									

CROSS-REFERENCE TABLE

SEC	0320										
SFLAG	0391										
SGNS	0491										
SIGN	0397	0398									
SLCOD	0328										
SLEND	0202										
SLWST	0445										
SNFLA	0197										
SPLFA	0196	1593									
SSWPT	0393										
SSYMA	0170	1535									
ST	0375										
START	0162	0007									
STASH	0847	0985	1040	1060	1095	1123	1209	1332	1338	1385	1428
	1455	1477	1512	1518							
STEXP	0441	0622									
STOR1	0426										
STORE	0425	0900									
STPFL	0323	0835									
STROP	0428	0710									
STSRH	0440	0568									
SUBTA	0696	0566									
SUBTB	0697	0696	0696								
SUMPA	0227										
SYF12	0483										
SYF25	0458										
SYMC1	0431	0615	0657								
SYMC2	0430	0587									
SYMCK	0432	0669									
SYMTP	0298										
SYSNT	0519	1742									
SYTSA	0209										
T1	1709	0818	0846	0849	0983	0986	0989	1020	1030	1033	1125
	1128	1169	1179	1180	1210	1213	1216	1266	1297	1301	1304
	1316	1328	1331	1337	1368	1370	1386	1389	1393	1396	1408
	1420	1434	1437	1442	1475	1494	1519	1522	1606	1614	1630
	1669										
T2	1711	0820	0838	0841	0984	0990	1022	1025	1028	1126	1127
	1167	1181	1182	1198	1201	1218	1221	1267	1314	1339	1342
	1366	1398	1404	1406	1429	1432	1444	1446	1456	1459	1476
	1496	1502	1511	1514	1613	1631	1673				
T3	1712	0761	0847	0976	0987	1031	1129	1132	1163	1174	1177
	1194	1197	1214	1240	1262	1298	1307	1310	1315	1330	1365
	1369	1395	1405	1441	1445	1474	1478	1484	1487	1490	1495
	1510	1565	1632								
T4	1713	0839	1026	1130	1131	1175	1176	1195	1196	1219	1254
	1302	1340	1387	1430	1435	1457	1479	1520	1568	1633	1671
T5	1714	0807	0832	1018	1023	1106	1110	1312	1327	1336	1394
	1419	1438	1492	1501	1513						
T6	1715	0753	0778	0794	0863	1073	1094	1105	1113	1143	1148
	1151	1157	1165	1178	1183	1199	1200	1211	1212	1223	1226
	1235	1247									
T7	1716	1533	1540	1541	1544	1545	1548	1589	1616	1627	1668
	1675	1680									
T8	1717	0562	0567	0621	1152	1162					
T9	1718	0627	0641	1144	1154	1224					
TEMP	0348	0349	0685	0692	0712						
TEMP1	0350	0708									
TEMP2	0351	0713									
TEMP3	0352										
TEMP4	0353										
TEMPS	0347	3754	0755	0756	0795	0796	0798	0804	0852	0855	0872
	0873	0876	0883	0905	0906	0910	0915	0916	0919	0920	0923
	0930	0933	0936	0941	0942	1046	1047	1102	1236	1602	1608
	1612										
TERM	0362										
TF1	1726	1276	1278	1348	1350	1470	1695				
TF1A	1695	1050	1063	1086	1092	1186	1188	1204	1206	1274	1289
	1346	1357	1374	1376	1411	1414	1417	1425	1450	1452	1467
	1471										
TF2	1727	1286	1288	1354	1356	1696					
TF2A	1696	1061	1064	1090	1093	1172	1189	1192	1207	1281	1290
	1351	1358	1423	1426							
TFLAG	0318										
TFLG1	0322										
TIMES	0510	0658									
TMPA	0203										
TMPOP	0184										
TREGE	0370										
TSPHA	0168										
TSTPT	0300										
TTY	0497										
TYPFL	0078										
UFLAG	0390										
UKPTR	0309										
UNITS	0319										
UNSTK	0446										
UPARA	0191										
VALUF	0247										
VAROA	0214										

CROSS-REFERENCE TABLE

```
WBUFF   0421
WHERE   1668    0898    1585
WIDTH   0383
WRITE   0159
XC      0410    0831
XEC4A   0164    0865    0885    0912    0969    0988    0998    1056    1081    1118    1225
        1249
XEC6A   0221
XFAR1   0423
XFAR2   0427    0786    1597
XIN2    1252    0776    1245
XSCRA   0220
XTBL1   0947    0945
YC      0412
ZAP     0152
ZONK    0488
```

#01 STRING OPTION BLOCK

```
0501                    LST
0502*
0503  22000          ORG 22000B
0504*
0505*     LINKS FOR THE MAIN SYSTEM AND
0506*     OTHER OPTION BLOCKS
0507*
0508*
0509  22000 066006 ID1   JMP MSL        MAIN SYSTEM LINK
0510  22001 066034       JMP SVARO
0511  22002 067357       JMP GADD
0512  22003 067421       JMP PSTR
0513  22004 066610       JMP TRSTR
0514  22005 067334       JMP .FDA1
0515*
0516*
0517*
0518*     EVALUATE A STRING VARIABLE
0519*     FOR THE PRINT ROUTINE
0520*
0521*     UPON EXIT:
0522*
0523*       B=STRING STARTING ADDRESS
0524*       A=STRING LENGTH
0525*       RETURN TO P+2
0526*
0527  22006 063357 MSL   JSM GADD       GET STRING ADDRESS
0528  22007 024133       LDB M2         EVALUATE SOURCE
0529  22010 063421       JSM PSTR         STRING
0530  22011 035471       STB L2T2
0531  22012 074117       LDA B          PUT COUNT IN A
0532  22013 025742       LDB TMP2       CHAR POINTER IN B
0533  22014 064357       JMP RETN2      RETURN TO P+2
0534*
0535*
0536*     LET STATEMENT SYNTAX
0537*     LOOK FOR SOURCE STRING =
0538*
0539  22015 062340 LETS  JSM SAVEP
0540  22016 062034       JSM SVARO      VARIABLE?
0541  22017 066064       JMP ABORT      NO
0542  22020 066064       JMP ABORT      NUMERIC
0543  22021 060463       JSM SYMC1      STRING
0544  22022 012020       OCT 12020      =?
0545  22023 164325       JMP E30,I      NO
0546*
0547*
0548*
0549*     LOOK FOR A DESTINATION STRING
0550*
0551*
0552  22024 062034 LTSYM JSM SVARO      VARIABLE?
0553  22025 066030       JMP QUOCK      NO, LOOK FOR QUOTE FEILD
0554  22026 164312       JMP E20-1,I    NUMERIC
0555  22027 065151       JMP EOST       YES, DEMAND END
0556  22030 045706 QUOCK ISZ SBPTR
0557  22031 160245       JSM GTSTA,I    GET A STRING CONSTANT
0558  22032 065151       JMP EOST
0559*
0560*
0562*
0563*
0564*     SVARO SUBROUTINE
0565*
0566*      EXIT CONDITIONS:
0567*
0568*       RETURN TO:
0569*         P+1  NO VARIABLE FOUND
0570*         P+2  NUMERIC VARIABLE
0571*         P+3  STRING VARIABLE
```

#01 STRING OPTION BLOCK

```
#572*
#573  22033 #62340        JSM SAVEP      SAVE PAGE
#574  22034 #61140 SVARO  JSM LTR        LETTER?
#575  22035 170402        RET            NO
#576  22036 #10077        CPA 044        $?
#577  22037 #66041        JMP *+2        YES
#578  22040 #64357        JMP RETN2      NO
#579  22041 #45644        ISZ ST+4       SET STRING OCCURRED FLAG
#580  22042 163610        JSM STRI2,I    STORE OPERAND
#581  22043 #60513        JSM LPCK       (?
#582  22044 #64356        JMP RETN3      NO
#583  22045 #20125        LDA LBOP
#584  22046 131706        STA SBPTR,I    RECORD RIGHT BRACKET
#585  22047 #20026        LDA .9
#586  22050 #31710        STA MSFLG      ENABLE LOGICL =
#587  22051 160202        JSM FSCA,I     GET FIRST INDEX
#588  22052 #60463        JSM SYMC1      LOOK FOR A COMMA
#589  22053 #12001        OCT 12001
#590  22054 #66056        JMP *+2        NOT FOUND
#591  22055 160202        JSM FSCA,I     FOUND
#592  22056 #60534        JSM RPCHK      DEMAND RIGHT PAREN
#593  22057 #24123        LDB RBOP
#594  22060 #55706        DSZ SBPTR
#595  22061 135706        STB SBPTR,I    RECORD A RIGHT BRACKET
#596  22062 #45706        ISZ SBPTR
#597  22063 #64356        JMP RETN3      YES
#599*
#600*    THIS ROUTINE RETURNS SYNTAX CONTROL TO OTHER
#601*    OPTION BLOCKS AND THE MAIN SYSTEM
#602*
#603*
#604  22064 #61064 ABORT JSM GFRST+2
#605  22065 #61055 AB3   JSM GNEXT      GET NEXT SYNTAX CHAR
#606  22066 #25416       LDB MODE
#607  22067 #74151       SLB *+3        CALC MODE?
#608  22070 #24233       LDB SBUFA
#609  22071 #66077       JMP AB1        YES
#610  22072 #60750       JSM DIGCK      CHAR IS DIGIT?
#611  22073 #66075       JMP *+2        NO
#612  22074 #66065       JMP ABORT+1    YES, LOOK FOR MORE
#613  22075 #24233       LDB SBUFA
#614  22076 #74070       SIB *+1
#615  22077 #35706 AB1   STB SBPTR      RESET SYNTAX BUFFER POINTER
#616  22100 #24132       LDB M1
#617  22101 #35473       STB TEMP       SEARCH FOR STATEMENTS
#618  22102 #25640       LDB ST
#619  22103 #35717       STB NUM        RESTORE PAGE
#620  22104 #74742       SBR 16
#621  22105 #35714       STB SFLAG
#622  22106 #35711       STB DFLAG
#623  22107 #15643       CPB ST+3       IMPLIED LET?
#624  22110 #70742       SAR 16
#625  22111 163602       JSM SALT,I     SEARCH ALL PAGES
#626  22112 #66112       JMP *          CHECKSUM WORD
#627  22113 167601       JMP STMTA,I
#628*
#629*
#630*
#631*    IF STATEMENT SYNTAX
#632*
#633  22114 #62033 IFSYN JSM SVARO-1    VARIABLE?
#634  22115 #66064       JMP ABORT      NO
#635  22116 #66064       JMP ABORT      NUMERIC
#636  22117 #26146       LDB GTEA
#637  22120 #05640       ADB ST
#638  22121 #05640       ADB ST
#639  22122 #60546       JSM STEXP      RELOP?
#640  22123 #66143       JMP IFERR      NO
#641  22124 #45706       ISZ SBPTR      YES
#642  22125 #70742       SAR 16
#643  22126 131706       STA SBPTR,I
#644  22127 #62034 IF1   JSM SVARO      VARIABLE?
#645  22130 #66133       JMP IF3        NO
#646  22131 #66143       JMP IFERR      NUMERIC
#647  22132 #66134       JMP IF2        YES
#648  22133 160245 IF3   JSM GTSTA,I    STRING CONSTANT?
#649  22134 #26145 IF2   LDB THENA      YES, DEMAND THEN
#650  22135 #05640       ADB ST
#651  22136 #05640       ADB ST
#652  22137 #60546       JSM STEXP        AND STORE AS EXPANDED OPERATOR
#653  22140 164317       JMP E20+4,I    NOT FOUND
#654  22141 #60644       JSM LNUM       DEMAND A SEQUENCE NUMBER
#655  22142 #65151       JMP EOST
#656  22143 160205 IFERR JSM AERRA,I    STRING IF ERROR
#657  22144 177672       DEC -70
#658  22145 174722 THENA ABS THEN+THEN-ID-ID
#659  22146 174727 GTEA  ABS GTE+GTE+1-ID-ID
#660*
#661*
#662*
```

#01 STRING OPTION BLOCK

```
#663*    IMPLIED DISP, LET SYNTAX
#664*
#665  22147 #62340 IMPLT JSM SAVEP
#666  22150 #31643       STA ST+3      SET IMPLIED LET FLAG
#667  22151 #55400       DSZ BADDR
#668  22152 #61055       JSM GNEXT
#669  22153 #55400       DSZ BADDR
#670  22154 #60752       JSM LETCK     LETTER?
#671  22155 #66167       JMP ILET1+1   NO, TRY IMPLIED DISP
#672  22156 #60626       JSM SAVBP     SAVE POINTERS
#673  22157 #62034       JSM SVAR0     VARIABLE?
#674  22160 #66166       JMP ILET1     NO
#675  22161 #66166       JMP ILET1     NUMERIC
#676  22162 #60463       JSM SYMC1     STRING
#677  22163 #12020       OCT 12020     =?
#678  22164 #66166       JMP ILET1     NO
#679  22165 #66024       JMP LTSYM     YES
#680  22166 #60640 ILET1 JSM RESBP     RESTORE POINTERS
#681  22167 #24123       LDB PRTOP     STORE
#682  22170 #55706       DSZ SBPTR       CODE FOR
#683  22171 135706       STB SBPTR,I       IMPLIED LET
#684  22172 #45706       ISZ SBPTR
#685  22173 #25416       LDB MODE
#686  22174 #74111       SLB NSTMT     SKIP IF PROG MODE
#687  22175 #62227       JSM PRNTS     PRINT STATEMENT?
#688  22176 #66200 NSTMT JMP ILET2     NO
#689  22177 #66213       JMP PRIN1     YES
#690  22200 #25644 ILET2 LDB ST+4      STRING
#691  22201 #74110       SZB *+2         OCCURRED?
#692  22202 164274       JMP E5,I      YES
#693  22203 #66064       JMP ABORT
#694  #0123        PRTOP EQU B12K
#695*
#696*
#698*
#699*
#700*
#701*    WRITE STATEMENT SYNTAX
#702*
#703  22204 160265 WRTS  JSM IOHEA,I   GET DEVICE AND FORMAT
#704  22205 #60534       JSM RPCKE     GET RIGHT PAREN
#705  22206 #55400       DSZ BADDR
#706  22207 #55706       DSZ SBPTR
#707*
#708*
#709  22210 #62340 PRINS JSM SAVEP
#710  22211 #62227       JSM PRNTS     PRINT SYNTAX?
#711  22212 #66216       JMP PRIN2
#712  22213 #25644 PRIN1 LDB ST+4      YES, STRING OCCURRED?
#713  22214 #74250       SZB ABADD     NO
#714  22215 #65151       JMP EOST      YES
#715  22216 #25644 PRIN2 LDB ST+4
#716  22217 #74110       SZB *+2       STRING OCCURRED?
#717  22220 164277       JMP E5+3,I    YES
#718  22221 #66064 ABADD JMP ABORT     NO
#719*
#720*
#721*
#722*    PRINT STATEMENT SYNTAX
#723*
#724*
#725*
#726*
#727  22222 #10043 PRSYN CPA B42       QUOTE FIELD?
#728  22223 #66237       JMP PR2+4     YES
#729  22224 #60461       JSM SYMC2     NO
#730  22225 #12001       OCT 12001     COMMA OR SEMICOLON?
#731  22226 #66233       JMP PR2       NO
#732  22227 #24132 PRNTS LDB M1
#733  22230 #35712       STB PFLAG     ENABLE FORMULA AND TAB
#734  22231 #61066       JSM GETSK
#735  22232 #66300       JMP PR1       END OF LINE
#736  22233 #10043 PR2   CPA B42       QUOTE?
#737  22234 #66236       JMP *+2       YES
#738  22235 #66247       JMP PR3       NO
#739  22236 #45706       ISZ SBPTR
#740  22237 160245       JSM GTSTA,I   RECORD STRING CONSTANT
#741  22240 #10031       CPA EOL       EOL?
#742  22241 #64357       JMP RETN2     YES
#743  22242 #74742       SBR 16
#744  22243 135706       STB SBPTR,I   CLEAR OUT NEXT WORD
#745  22244 #24132       LDB M1
#746  22245 #35712       STB PFLAG
#747  22246 #66222       JMP PRSYN
#748  22247 #45712 PR3   ISZ PFLAG     TAB OR FORMULA PERMITTED?
#749  22250 170402       RET
#750  22251 #27603       LDB TABA
#751  22252 160175       JSM TSRHA,I   TAB?
#752  22253 #66263       JMP PR4       NO
#753  22254 #70544       SAL 5         YES
```

```
        #01  STRING OPTION BLOCK

#754  22255 #40067        IOR TYPFL      SET PRE-DEFINED FUNCTION FLAGS
#755  22256 #60454        JSM SBPUD-2    STORE
#756  22257 #24123        LDB RBOP
#757  22260 135706        STB SBPTR,I
#758  22261 160202        JSM FSCA,I     GET TAB FORMULA
#759  22262 #66275        JMP PR5
#760  22263 #55400 PR4    DSZ BADDR      BACK UP POINTER
#761  22264 #21400        LDA BADDR
#762  22265 #31645        STA ST+5       SAVE BADDRR
#763  22266 #62034        JSM SVARO       VARIABLE?
#764  22267 #66272        JMP PR6        NO
#765  22270 #66272        JMP *+2        NUMERIC
#766  22271 #66275        JMP PR5        STRING
#767  22272 #21645 PR6    LDA ST+5
#768  22273 #31400        STA BADDR      RESTORE BADDR
#769  22274 160202        JSM FSCA,I     FETCH FORMULA
#770  22275 #10031 PR5    CPA EOL        EOL?
#771  22276 #64357        JMP RETN2
#772  22277 #66222        JMP PRSYN      NO
#773  22300 #45706 PR1    ISZ SBPTR
#774  22301 #64357        JMP RETN2
#775*
#776*
#777*    READ AND INPUT STATEMENT SYNTAX
#778*
#779  22302 #62340 READS  JSM SAVEP
#780  22303 #62034        JSM SVARO      VARIABLE?
#781  22304 #66315        JMP READ3
#782  22305 #66311        JMP READ1      NUMERIC
#783  22306 #60504 READ2  JSM COMCK      COMMA?
#784  22307 #66213        JMP PRIN1      NO
#785  22310 #66303        JMP READS+1    YES
#786  22311 163600 READ1  JSM VAR1,I
#787  22312 #66312        JMP *          CONSTANT STORAGE
#788  22313 #66306        JMP READ2
#789  22314 #66306        JMP READ2
#790  22315 #25644 READ3  LDB ST+4
#791  22316 #74110        SZB *+2        STRING OCCURRED?
#792  22317 164321        JMP E25+1,I    YES
#793  22320 #66064        JMP ABORT      NO
#794*
#795*
#796*    DATA STATEMENT SYNTAX
#797*
#798  22321 #62340 DATAS  JSM SAVEP
#799  22322 #45706        ISZ SBPTR
#800  22323 #61156        JSM CONST      LOOK FOR A NUMBER
#801  22324 #66331        JMP DATA1      NOT FOUND
#802  22325 160244        JSM NUMOA,I    FOUND , STORE
#803  22326 #60504 DATA2  JSM COMCK      COMMA?
#804  22327 #66213        JMP PRIN1      NO
#805  22330 #66322        JMP DATAS+1
#806  22331 #45644 DATA1  ISZ ST+4
#807  22332 #10043        CPA B42        QUOTE?
#808  22333 #66335        JMP *+2        YES
#809  22334 164316        JMP E20+3,I    NO
#810  22335 160245        JSM GTSTA,I    STRING CONSTANT?
#811  22336 #66326        JMP DATA2      YES
#812  22337 #66064        JMP ABORT      NO, ERROR
#813*
#814*
#815  22340 #21717 SAVEP  LDA NUM
#816  22341 #31640        STA ST
#817  22342 #31643        STA ST+3       RESET IMPLIED LET FLAG
#818  22343 #70742        SAR 16
#819  22344 #31644        STA ST+4       RESET STRING OCCURRED FLAG
#820  22345 170402        RET
#821*
#822*    NON-FORMULA OPERATORS
#823*
#824  22346 #46105 LEN    ASC 1,LE
#825  22347 #47215        OCT 47215      N, OPCODE 15
#826  22350 #52110 THEN   ASC 2,THEN
      22351 #42516
#827  22352 150074 GTE    OCT 150074     20,<
#828  22353 #36602        OCT 36602      =,2
#829  22354 #37075        ASC 1,>=
#830  22355 101443        OCT 101443     3,#
#831  22356 102074        OCT 102074     4,<
#832  22357 #37204        OCT 37204      >,4
#833  22360 #36605        OCT 36605      =,5
#834  22361 #36206        OCT 36206      <,6
#835  22362 #37307        OCT 37307      >,7
#836*
#837*
#838*    EXECUTE LEN STATEMENT
#839*
#840*
#841  22363 #62450 ELEN   JSM CKMOD
#842  22364 160227 ELEN1  JSM FXFLA,I     PUT NUMBER IN AR2
```

```
        #01  STRING OPTION BLOCK


0843  22365 045472 ELEN2 ISZ TEMPS
0844  22366 170402       RET
0845*
0846*     CHECK LEN FUNCTION SYNTAX
0847*
0848*
0849  22367 062536 LENS  JSM GVAR      GET FIRST PARAMETER
0850  22370 066446       JMP LENER
0851  22371 066446       JMP LENER     NO
0852  22372 025646 LENS1 LDB PTR2
0853  22373 135472       STB TEMPS,I    FLAG FOR FSC
0854  22374 060534       JSM RPCKE     )?
0855  22375 025651       LDB TEMP6      RESTORE STRING
0856  22376 035644       STB ST+4       OCCURRED FLG
0857  22377 064356       JMP RETN3     YES
0858  22400 045706 LENS2 ISZ SBPTR
0859  22401 160245       JSM GTSTA,I   SEEK STRING CONSTANT
0860  22402 066372       JMP LENS1
0861*
0862*
0863  22403 021416 EINDX LDA MODE
0864  22404 070113       SAP *+2       SYNTAX MODE?
0865  22405 066462       JMP INDXS     YES
0866  22406 062550       JSM GPTRS
0867  22407 035640       STB CP0
0868  22410 031646       STA PTR2
0869  22411 024133       LDB M2
0870  22412 063421       JSM PSTR
0871  22413 076110       RZB *+2       NULL STRING?
0872  22414 066434       JMP EIND3     YES
0873  22415 035647       STB TMPRE
0874  22416 031644       STA INTMP
0875  22417 025640       LDB CP0
0876  22420 035654       STB TNULL
0877  22421 027645 EIND2 LDB PTR1A
0878  22422 061041       JSM GETCH
0879  22423 031641       STA CP1
0880  22424 027731       LDB TMP2A
0881  22425 061041       JSM GETCH
0882  22426 011641       CPA CP1
0883  22427 066436       JMP EIND1
0884  22430 021646       LDA PTR2
0885  22431 031645       STA PTR1
0886  22432 055647       DSZ TMPRE
0887  22433 066417       JMP EIND2-2
0888  22434 074742 EIND3 SBR 16
0889  22435 066364       JMP ELEN1
0890  22436 055654 EIND1 DSZ TNULL
0891  22437 066421       JMP EIND2
0892  22440 025644       LDB INTMP
0893  22441 005640       ADB CP0
0894  22442 074076       TCB
0895  22443 005742       ADB TMP2
0896  22444 004012       ADB .1
0897  22445 066364       JMP ELEN1
0898*
0899*
0900  22446 160205 LENER JSM AERRA,I
0901  22447 177671       DEC -71
0902*
0903*
0904  22450 021416 CKMOD LDA MODE
0905  22451 070153       SAP *+3       SYNTAX MODE?
0906  22452 055777       DSZ RSPTR     YES
0907  22453 066367       JMP LENS
0908  22454 045472 CKM1  ISZ TEMPS
0909  22455 125472       LDB TEMPS,I
0910  22456 074442       SBR 10
0911  22457 014012       CPB .1        QUOTE?
0912  22460 066572       JMP GP1       YES
0913  22461 066563       JMP GP2       NO
0914*
0915  22462 062536 INDXS JSM GVAR      VARIABLE?
0916  22463 066446       JMP LENER
0917  22464 066446       JMP LENER     NUMERIC
0918  22465 060504 INDS2 JSM COMCK     STRING, COMMA?
0919  22466 164305       JMP E15-1,I   NO
0920  22467 062034       JSM SVAR0     YES, VARIABLE?
0921  22470 066400       JMP LENS2     NO, LOOK FOR QUOTE FEILD
0922  22471 164312       JMP E20-1,I    NUMERIC
0923  22472 066372       JMP LENS1     STRING
0924*
0925  22473 066465       JMP INDS2
0926*
0927*
0928  22474 062450 EVAL  JSM CKMOD
0929  22475 025400       LDB BADDR
0930  22476 060563       JSM SLWST     SAVE BADDR
0931  22477 031645       STA PTR1
0932  22500 045654       ISZ TNULL     NULL STRING?
```

```
          #01  STRING OPTION BLOCK

 0933  22501 066503          JMP *+2        NO
 0934  22502 066534          JMP VALER      YES
 0935  22503 021412          LDA LSTPT      SET BUFFER
 0936  22504 070070          STA *+1          START TO
 0937  22505 070744          SAL 1              LOW STACK
 0938  22506 031400          STA BADDR            POINTER PLUS 1
 0939  22507 031646          STA PTR2       SAVE BUFFER START
 0940  22510 021413          LDA HSTPT      SET BUFFER
 0941  22511 070744          SAL 1            END TO HIGH
 0942  22512 031623          STA PEND           STACK POINTER PLUS 1
 0943  22513 027645 EVAL1 LDB PTR1A
 0944  22514 061041          JSM GETCH      GET A SOURCE CHAR
 0945  22515 061016          JSM ONEXT      PUT IN BUFFER
 0946  22516 021400          LDA BADDR
 0947  22517 011623          CPA PEND       OVERFLOW?
 0948  22520 064561          JMP MERR       YES, ERROR
 0949  22521 045654          ISZ TNULL      NO,DONE?
 0950  22522 066513          JMP EVAL1      NO
 0951  22523 020055          LDA I          TERMINATE THE BUFFER
 0952  22524 061016          JSM ONEXT        WITH A NON-DIGIT CHARACTER
 0953  22525 021646          LDA PTR2
 0954  22526 031400          STA BADDR      INITIALIZE BADDR
 0955  22527 061156          JSM CONST
 0956  22530 066534          JMP VALER
 0957  22531 060567          JSM UNSTK
 0958  22532 031400          STA BADDR      RESTORE BADDR
 0959  22533 066365          JMP ELEN2
 0960  22534 160205 VALER JSM AERRA,I
 0961  22535 177664          DEC -76
 0962*
 0963*
 0964*
 0965*
 0966  22536 121472 GVAR  LDA TEMPS,I    GET LEFT PAREN COUNT
 0967  22537 031646          STA PTR2       AND SAVE
 0968  22540 021644          LDA ST+4       SAVE STRING
 0969  22541 031651          STA TEMP6        OCCURRED FLAG
 0970  22542 061055          JSM GNEXT      GET NEXT CHAR
 0971  22543 055400          DSZ BADDR
 0972  22544 045706          ISZ SHPTR
 0973  22545 060521          JSM LPCKE      DEMAND LEFT PAREN
 0974  22546 045400          ISZ BADDR
 0975  22547 066034          JMP SVAR0
 0976*
 0979*    STATEMENT EXECUTION
 0980*
 0981*
 0982*    LET EXECUTION
 0983*
 0984*
 0985*
 0986  22550 063357 GPTRS JSM GADD
 0987  22551 121472          LDA TEMPS,I
 0988  22552 045472          ISZ TEMPS
 0989  22553 031644          STA INTMP      SAVE RELATIONAL OPERATOR
 0990  22554 125472          LDB TEMPS,I
 0991  22555 074442          SBR 10
 0992  22556 014012          CPB .1         QUOTE?
 0993  22557 066572          JMP GP1        YES
 0994  22560 050065          AND OPMSK      NO
 0995  22561 010065          CPA OPMSK      NON-FORMULA OPERATOR?
 0996  22562 045472          ISZ TEMPS      YES
 0997  22563 055472 GP2   DSZ TEMPS
 0998  22564 063357          JSM GADD       GET SOURCE STRING ADDRESS
 0999  22565 024133          LDB M2
 1000  22566 063421          JSM PSTR       EVALUATE SOURCE STRING
 1001  22567 035647 XFER STB TMPRE      SAVE LENGTH
 1002  22570 031645          STA PTR1
 1003  22571 170402          RET
 1004  22572 063551 GP1   JSM GQUOT       GET QUOTE FIELD
 1005  22573 066567          JMP XFER
 1006*
 1007  22574 062550 LTEX  JSM GPTRS       GET DEST STRING POINTERS
 1008  22575 024132 LT4   LDB M1          NO
 1009  22576 063421          JSM PSTR       EVALUATE DESTINATION STRING
 1010  22577 074410          SZB LT2+1      NULL STRING?
 1011  22600 021416          LDA MODE
 1012  22601 070250          SZA LT2        CALC MODE?
 1013  22602 160170          JSM CLERA,I    YES
 1014  22603 021647          LDA TMPRE
 1015  22604 025645          LDB PTR1
 1016  22605 061330          JSM .NPWR      DISPLAY THE STRING
 1017  22606 062610 LT2   JSM TRSTR
 1018  22607 164172          JMP XEC4A,I
 1019*
 1020*
 1021*    STRING TRANSFER SUBROUTINE
 1022*
 1023*
 1024  22610 021645 TRSTR LDA PTR1
```

#01 STRING OPTION BLOCK

```
1025  22611 #70076        TCA
1026  22612 #01742        ADA TMP2
1027  22613 #70413        SAP TR1         FORWARD TRANSFER?
1028  22614 #45654 TR2    ISZ TNULL       YES
1029  22615 #66617        JMP *+2         YES
1030  22616 170402        RET
1031  22617 #63255        JSM FSCH        GET NEXT SOURCE CHAR
1032  22620 #27731 LT5    LDB TMP2A
1033  22621 #61017        JSM OUTCR       STORE IT
1034  22622 #66614        JMP TR2         LOOK FOR MORE
1035  22623 #21654 TR1    LDA TNULL
1036  22624 #70070        STA *+1
1037  22625 #70056        CMA
1038  22626 #70137        LDB A
1039  22627 #05645        ADB PTR1
1040  22630 #35645        STB PTR1        POINT TO END OF SOURCE
1041  22631 #01742        ADA TMP2
1042  22632 #31742        STA TMP2        POINT TO END OF DEST
1043  22633 #21654        LDA TNULL
1044  22634 #70056        CMA
1045  22635 #25647        LDB TMPRE
1046  22636 #74076        TCB
1047  22637 #70037        ADB A
1048  22640 #74113        SBP *+2         TOO MUCH SOURCE?
1049  22641 #74742        SBR 16          YES. NO BLANKS NEEDED
1050  22642 #35647        STB TMPRE       STORE BLANK COUNT
1051  22643 #45654 TR4    ISZ TNULL
1052  22644 #66646        JMP *+2
1053  22645 170402        RET
1054  22646 #25647        LDB TMPRE
1055  22647 #74250        SZB TR3         MORE BLANKS?
1056  22650 #20041        LDA .32         YES
1057  22651 #55647        DSZ TMPRE
1058  22652 #66653        JMP *+1
1059  22653 #66657        JMP TR5
1060  22654 #27645 TR3    LDB PTR1A
1061  22655 #61041        JSM GETCH
1062  22656 #55645        DSZ PTR1
1063  22657 #27731 TR5    LDB TMP2A
1064  22660 #61017        JSM OUTCR
1065  22661 #55742        DSZ TMP2
1066  22662 #55742        DSZ TMP2
1067  22663 #55645        DSZ PTR1
1068  22664 #66643        JMP TR4         NO
1069*
1070  22665 #45654 FINCH  ISZ TNULL       MORE TRANSFER STRING
1071  22666 #66670        JMP *+2         YES
1072  22667 170402        RET             NO
1073  22670 #61040        JSM GETCH-1
1074  22671 #10031        CPA EOL         END OF LINE?
1075  22672 170402        RET             YES
1076  22673 #10043        CPA B42         YES. QUOTE?
1077  22674 170402        RET             YES
1078  22675 #27731        LDB TMP2A       NO.
1079  22676 #61017        JSM OUTCR          STORE IT
1080  22677 #66665        JMP FINCH
1081*
1083*
1084*
1085*    INPUT STATEMENT EXECUTION
1086*
1087*
1088  22700 #63013 EINPT  JSM ESAVE       SAVE CURRENT PAGE ADDRESS
1089  22701 #20112        LDA RSTKA
1090  22702 #31777        STA RSPTR       RESET STACK
1091  22703 #21622        LDA TERM
1092  22704 #72110        RZA *+2
1093  22705 160170        JSM CLERA,I
1094  22706 #21472        LDA TEMPS
1095  22707 #00133        ADA M2
1096  22710 #31641        STA SPRDD       SAVE START OF INPUT STATEMENT
1097  22711 #21641 EIN2   LDA SPRDD
1098  22712 #31421        STA PRADD         INPUT STATEMENT
1099  22713 #21472        LDA TEMPS
1100  22714 #31640        STA TEPPP
1101  22715 121472        LDA TEMPS,I
1102  22716 #50040        AND B37
1103  22717 #10076        CPA B36         STRING VARIABLE?
1104  22720 #67017        JMP INSTR       YES
1105  22721 #23010        LDA INLNK
1106  22722 #01643        ADA SAVE
1107  22723 #31432        STA DRQST       SET UP RETURN LINK
1108  22724 164173 INRET  JMP PEXMA,I     CALL FOR INPUT
1109  22725 #55400        DSZ BADDR
1110  22726 #25641        LDB SPRDD
1111  22727 #61007        JSM NEXTA       SET PRADD BACK TO END
1112  22730 #35421        STB PRADD         OF INPUT STATEMENT
1113  22731 #21472        LDA TEMPS
1114  22732 #31640        STA TEPPP       SAVE START OF VARIABLE
1115  22733 160201 .EIN1  JSM FORMA,I     GET THE ADDRESS OF THE VARIABLE I N HSTPT
```

#01 STRING OPTION BLOCK

```
1116  22734 024017        LDB .2
1117  22735 035706        STB SBPTR      AIM SBPTR AT ROM
1118  22736 027007        LDB INLN1
1119  22737 005643        ADB SAVE
1120  22740 035432        STB DRQST
1121  22741 061156        JSM CONST      BUILD A NUMBER IN AR2
1122  22742 066765        JMP INCH5      P+1 - NO NUMBER FOUND, SO INC HSTPTP
1123  22743 070742        SAR 16
1124  22744 031432        STA DRQST      CLEAR FLAG
1125  22745 060412        JSM RSULT      SAVE ARE AT HSTPT,I
1126  22746 025472        LDB TEMPS
1127  22747 015421        CPB PRADD      END OF STATEMENT
1128  22750 067011        JMP CXEC4      YES
1129  22751 035640 EIN3   STB TEPPP      RESET THE TEMPORARY POINTER
1130  22752 055400        DSZ BADDR      NO - DINC BADDR TO LOOK AT THE LAST CHAR IN BUFF.
1131  22753 061066        JSM GETSK      GET LAST CHAR.
1132  22754 067005        JMP INCH1
1133  22755 013642        CPA B54        COMMA?
1134  22756 066760        JMP *+2        YES
1135  22757 066777        JMP INCH2      NO
1136  22760 121472        LDA TEMPS,I
1137  22761 050040        AND B37
1138  22762 010076        CPA B36        STRING VARIABLE?
1139  22763 067024        JMP EIN4       YES
1140  22764 066733        JMP .EIN1      NO
1141*
1142  22765 160170 INCH5  JSM CLERA,I
1143  22766 023643        LDA .17
1144  22767 027712        LDB ER3A
1145  22770 063137        JSM ER4+2
1146  22771 063135        JSM ER4
1147  22772 021640 INCHT  LDA TEPPP
1148  22773 031472        STA TEMPS      RESTORE VARIABLE POINTER
1149  22774 045413        ISZ HSTPT
1150  22775 045413        ISZ HSTPT
1151  22776 066711        JMP EIN2
1152*
1153  22777 160170 INCH2  JSM CLERA,I
1154  23000 020034        LDA .21
1155  23001 027725        LDB ER1A
1156  23002 063137        JSM ER4+2
1157  23003 063135        JSM ER4
1158  23004 066711        JMP EIN2
1159*
1160  23005 160170 INCH1  JSM CLERA,I
1161  23006 066711        JMP EIN2
1162  23007 000772 INLN1  ABS INCHT-ID1
1163  23010 000724 INLNK  ABS INRET-ID1
1164*
1165  23011 160170 CXEC4  JSM CLERA,I
1166  23012 164172        JMP XEC4A,I
1167*
1168*     SAVE PAGE ADDRESS DURING EXECUTION
1169*
1170  23013 074117 ESAVE  LDA B
1171  23014 050160        AND B174K
1172  23015 031643        STA SAVE
1173  23016 170402        RET
1174*
1175*
1176  23017 070742 INSTR  SAR 16
1177  23020 031701        STA NDISP
1178  23021 031702        STA NPRIN
1179  23022 031707        STA BLANK      DON'T IGNORE BLANKS
1180  23023 063142        JSM INCAL
1181  23024 063357 EIN4   JSM GADD       GET STRING ADDRESS
1182  23025 025412        LDB LSTPT      EXTRACT
1183  23026 004133        ADB M2           PHYSICAL
1184  23027 074517        LDA B,I            LENGTH OF
1185  23030 070002        SAR 1
1186  23031 000132        ADA M1                 DESTINATION
1187  23032 070517        LDA A,I                  STRING
1188  23033 070342        SAR 8
1189  23034 070076        TCA
1190  23035 004012        ADB .1         SUBTRACT FIRST
1191  23036 074417        ADA B,I          SUBSCRIPT
1192  23037 000132        ADA M1
1193  23040 070076        TCA            STORE IT AS END OF
1194  23041 031647        STA TMPRE        UNSPECIFIED DEST STRING
1195  23042 024132        LDB M1         PREPARE
1196  23043 063421        JSM PSTR         DESTINATION STRING
1197  23044 021742        LDA TMP2
1198  23045 031645        STA PTR1
1199  23046 025654        LDB TNULL      SAVE LENGTH
1200  23047 035644        STB INTMP        ALLOWANCE
1201  23050 061055        JSM GNEXT
1202  23051 067055        JMP EIN5
1203  23052 063142 EIN14  JSM INCAL      GET INPUT RECORD
1204  23053 061062        JSM GFRST
1205  23054 067054        JMP *          CHECKSUM WORD
```

```
1206  23055 025645 EIN5  LDB PTR1
1207  23056 035742       STB TMP2      RESTORE STRING POINTER
1208  23057 010043       CPA B42       QUOTE?
1209  23060 067062       JMP *+2       YES
1210  23061 055400       DSZ BADDR     NO
1211  23062 062665       JSM FINCH     TRANSFER THE STRING
1212  23063 074742       SBR 16        ALL REQUESTED
1213  23064 015654       CPB TNULL       CHARACTERS TRANSFERRED?
1214  23065 067112       JMP EIN21     YES
1215  23066 015656       CPB PS1       NO, TRANSFER LENGTH SPECIFIED
1216  23067 067106       JMP EIN20     NO
1217  23070 031644       STA INTMP     SAVE EOR CHAR
1218  23071 070742       SAR 16
1219  23072 031647       STA TMPRE
1220  23073 001654       ADA TNULL       TRANSFER
1221  23074 000132       ADA M1
1222  23075 031654       STA TNULL
1223  23076 062614       JSM TR2
1224  23077 010031 EIN18 CPA EOL       TRANSFER ENDED BY EOL?
1225  23100 067102       JMP EINP6     YES
1226  23101 061055 EIN19 JSM GNEXT     NO, WAS A "
1227  23102 025641 EINP6 LDB SPRDD
1228  23103 061007       JSM NEXTA     SET PRADD BACK TO
1229  23104 035421       STB PRADD       END OF INPUT STATEMENT
1230  23105 066746       JMP EIN3-1    LOOK FOR END
1231  23106 125651 EIN20 LDB TEMP6,I   SET LOGICAL
1232  23107 005654       ADB TNULL       TO ACTUAL
1233  23110 135651       STB TEMP6,I       STRING LENGTH
1234  23111 067077       JMP EIN18
1235  23112 015656 EIN21 CPB PS1       LENGTH OF TRANSFER SPECIFIED?
1236  23113 067121       JMP EIN15     NO
1237  23114 061066 EIN16 JSM GETSK     YES, END OF LINE?
1238  23115 067102       JMP EINP6     YES
1239  23116 010043       CPA B42
1240  23117 067101       JMP EIN19     YES
1241  23120 067114       JMP EIN16     NO, LOOK FOR " OR EOL
1242  23121 061066 EIN15 JSM GETSK     EOL?
1243  23122 067102       JMP EINP6     YES
1244  23123 010043       CPA B42       NO, CLOSING QUOTE?
1245  23124 067101       JMP EIN19     YES
1246  23125 021644       LDA INTMP     NO, DESTINATION STRING EXCEEDED
1247  23126 031654       STA TNULL     RESTORE DESTINATION
1248  23127 160170       JSM CLERA,I
1249  23130 020033       LDA .16
1250  23131 027726       LDB ER2A
1251  23132 063137       JSM ER4+2
1252  23133 063135       JSM ER4
1253  23134 067052       JMP EIN14
1254*
1255*
1256*
1257  23135 020025 ER4   LDA .8
1258  23136 027724       LDB ER4A
1259  23137 005643       ADB SAVE
1260  23140 005643       ADB SAVE
1261  23141 065330       JMP .NPWR
1262*
1263  23142 020012 INCAL LDA .1
1264  23143 031432       STA DRQST
1265  23144 027727       LDB QMRKA
1266  23145 061330       JSM .NPWR     OUTPUT A?
1267  23146 161462       JSM REEDL,I   GET INPUT RECORD
1268  23147 067164       JMP INC1      NULL RECORD
1269  23150 061062       JSM GFRST
1270  23151 067164       JMP INC1      YES
1271  23152 021432       LDA DRQST
1272  23153 072210       RZA *+4       EXECUTION EXIT OCCURRED?
1273  23154 020041       LDA .32       YES
1274  23155 031707       STA BLANK
1275  23156 167613       JMP REEDY,I
1276  23157 074742       SBR 16
1277  23160 035432       STB DRQST
1278  23161 035416       STB MODE
1279  23162 055400       DSZ BADDR
1280  23163 170432       RET
1281  23164 021432 INC1  LDA DRQST
1282  23165 070150       SZA INC2      EXECUTION EXIT OCCURRED?
1283  23166 160170       JSM CLERA,I   NO
1284  23167 067142       JMP INCAL
1285  23170 167612 INC2  JMP REEDX,I
1286*
1287*
1288*
1289*
1290*    IF STATEMENT EXECUTION
1291*
1292*
1293  23171 062550 IFEX  JSM GPTRS     GET STRING PARAMETERS
1294  23172 074056       CMB
1295  23173 035640       STB CP0
```

#01 STRING OPTION BLOCK

```
1296  23174 020133        LDA M2          LENGTH
1297  23175 063421        JSM PSTR        PREPARE TEST STRING
1298  23176 035641        STB CP1         SAVE ACTUAL LENGTH
1299  23177 045640 COMP2  ISZ CP0         MORE SPECIFIED STRING?
1300  23200 067203        JMP COMP3       YES
1301  23201 074742        SBR 16          NO. LOAD A
1302  23202 067206        JMP COMP4         NULL CHARACTER
1303  23203 027645 COMP3  LDB PTR1A
1304  23204 061041        JSM GETCH
1305  23205 070137        LDB A
1306  23206 045654 COMP4  ISZ TNULL       MORE SPECIFIED TEST STRING?
1307  23207 067216        JMP COMP6       YES
1308  23210 070742        SAR 16          NO
1309  23211 074076 COMP5  ICB             COMPARE
1310  23212 074017        ADA B             CHARACTERS
1311  23213 072110        RZA *+2         EXIT ON NOT EQUAL
1312  23214 077150        RZB COMP2       OR BOTH NULL CHARACTERS
1313  23215 067231        JMP BRANC       EXIT TO BRANCH
1314  23216 021641 COMP6  LDA CP1         MORE ACTUAL
1315  23217 072150        RZA COMP7         TEST STRING?
1316  23220 020041        LDA .32         NO
1317  23221 067211        JMP COMP5
1318  23222 000132 COMP7  ADA M1
1319  23223 031641        STA CP1
1320  23224 035650        STB ST+8
1321  23225 027731        LDB TMP2A       EXTRACT NEXT
1322  23226 061041        JSM GETCH         TEST
1323  23227 025650        LDB ST+8
1324  23230 067211        JMP COMP5           CHARACTER
1325*
1326  23231 070137 BRANC  LDB A
1327  23232 021644        LDA INTMP
1328  23233 050070        AND UPDMK
1329  23234 070202        SAR 5
1330  23235 074153        SBP BR1         LESS THAN?
1331  23236 070211        SLA GOTOE       YES. EVEN OP CODE?
1332  23237 164172        JMP XEC4A,I     NO
1333  23240 076310 BR1    RZB BR2         EQUAL?
1334  23241 010022        CPA .5          YES. - OP?
1335  23242 167577 GOTOE  JMP EGOS1,I     YES
1336  23243 000135        ADA M4
1337  23244 071712        SAM *-2
1338  23245 164172        JMP XEC4A,I
1339  23246 010023 BR2    CPA .6          <?
1340  23247 164172        JMP XEC4A,I
1341  23250 010022        CPA .5          =?
1342  23251 164172        JMP XEC4A,I
1343  23252 010017        CPA .2
1344  23253 164172        JMP XEC4A,I
1345  23254 167577        JMP EGOS1,I
1346*
1347*
1348  23255 025647 FSCH   LDB TMPRE
1349  23256 076150        RZB *+3         OUT OF STRING?
1350  23257 020041        LDA .32         YES. GET A
1351  23260 170402        RET               BLANK
1352  23261 004132        ADB M1
1353  23262 035647        STB TMPRE
1354  23263 027645        LDB PTR1A
1355  23264 065041        JMP GETCH
1356                      LST
1357*
1358  23265 121472 EREAD  LDA TEMPS,I
1359  23266 050040        AND B37
1360  23267 010076        CPA B36         STRING VARIABLE?
1361  23270 067310        JMP RDSTR       YES
1362  23271 160201        JSM FORMA,I     NO. GET VARIABLE ADDRESS
1363  23272 163611 ER1    JSM SWITH,I     EXCHANGE TEMPS AND NXTDT
1364  23273 015414        CPB DSTRT       NEED NEW DATA STATEMENT?
1365  23274 063334        JSM .FDA1       YES
1366  23275 163611        JSM SWITH,I     NO
1367  23276 125415        LDB NXTDT,I
1368  23277 074152        SBM *+3         NUMERIC DATA IS NEXT?
1369  23300 160205        JSM AERRA,I     NO
1370  23301 177665        DEC -75
1371  23302 160256        JSM FDAAA,I     YES
1372  23303 060412        JSM RSULT
1373  23304 021421 ER2    LDA PRADD
1374  23305 011472        CPA TEMPS       MORE VARIABLES?
1375  23306 164172        JMP XEC4A,I
1376  23307 067265        JMP EREAD
1377  23310 063357 RDSTR  JSM GADD        GET STRING VARIABLE PARAMETERS
1378  23311 163611        JSM SWITH,I
1379  23312 025472 ER3    LDB TEMPS
1380  23313 015414        CPB DSTRT       NEED NEW DATA STMT?
1381  23314 063334        JSM .FDA1       YES
1382  23315 045472        ISZ TEMPS
1383  23316 125472        LDB TEMPS,I
1384  23317 074442        SBR 10
1385  23320 014012        CPB .1          QUOTE?
```

#01 STRING OPTION BLOCK

```
1386  23321 #67324        JMP *+3        YES
1387  23322 160205        JSM AERRA,I    YES
1388  23323 177665        DEC -75
1389  23324 #63551        JSM GQUOT      NO
1390  23325 #35647        STB TMPRE
1391  23326 #31645        STA PTR1
1392  23327 #24132        LDB M1         PREPARE
1393  23330 #63421        JSM PSTR         DESTINATION STRING
1394  23331 #62610        JSM TRSTR      TRANSFER THE STRING
1395  23332 163611        JSM SWITH,I
1396  23333 #67304        JMP ER2
1397*
1398*
1399  23334 #21421 .FDA1 LDA PRADD
1400  23335 #31644        STA INTMP
1401  23336 #35421        STB PRADD
1402  23337 #25421 FD2    LDB PRADD
1403  23340 #15423        CPB PBPTR      END OF PROGRAM?
1404  23341 164350        JMP E50-1,I    OUT OF DATA
1405  23342 #35472        STB TEMPS
1406  23343 #60577        JSM NEXTL+3
1407  23344 #01466        ADA LOCL1      ENCODE PAGE AND OP CODE
1408  23345 #13732        CPA MSDAT      MAIN SYSTEM DATA STMT?
1409  23346 #67352        JMP FD3        YES
1410  23347 #13733        CPA STDAT      NO, STRING DATA STMT?
1411  23350 #67352        JMP FD3        YES
1412  23351 #67337        JMP FD2        NO, KEEP LOOKING
1413  23352 #25421 FD3    LDB PRADD
1414  23353 #35414        STB DSTRT
1415  23354 #25644        LDB INTMP
1416  23355 #35421        STB PRADD
1417  23356 170402        RET
1418*
1422*
1423*    GADD SUBROUTINE
1424*
1425*    FETCH STRING ADDRESS FROM SYMBOL TABLE
1426*
1427*    ENTRY CONDITIONS:
1428*
1429*       A=OPERAND
1430*       TEMPS POINTS TO STRING NAME
1431*
1432*    EXIT CONDITIONS:
1433*
1434*       STRING ADDRESS AND SUBSCRIPTS ARE
1435*       ON LOW STACK
1436*
1437  23357 #60771 GADD   JSM GTOPP      GET OPERAND
1438  23360 #63567        JSM GSYMB      GET SYMBOL ADDRESS
1439  23361 #04132        ADB M1
1440  23362 #74537        LDB B,I
1441  23363 #74744        SBL 1
1442  23364 #60563        JSM SLWST      PUT ON STACK
1443  23365 #45472        ISZ TEMPS
1444  23366 #21472        LDA TEMPS
1445  23367 #11421        CPA PRADD      OUT OF CODE?
1446  23370 #67375        JMP GA2
1447  23371 121472        LDA TEMPS,I
1448  23372 #50065        AND OPMSK      GET OP CODE
1449  23373 #10125        CPA L8OP       SUBSCRIPTED?
1450  23374 #67401        JMP GA1        YES
1451  23375 #74742 GA2    SBR 16         NO
1452  23376 #60563        JSM SLWST
1453  23377 #24132        LDB M1
1454  23400 #64563        JMP SLWST
1455  23401 #63412 GA1    JSM SUBEV      EVALUATE SUBSCRIPTS
1456  23402 #55472        DSZ TEMPS
1457  23403 121472        LDA TEMPS,I
1458  23404 #70442        SAR 10
1459  23405 #50040        AND B37
1460  23406 #10017        CPA ,C         COMMA?
1461  23407 #67412        JMP SUBEV      YES
1462  23410 #45472        ISZ TEMPS
1463  23411 #67377        JMP GA1-2      NO
1464*
1465*    SUBEV SUBROUTINE
1466*
1467*    EVALUATE AN EXPRESSION, CONVERT TO
1468*    AN INTEGER, STORE ON LOW STACK
1469*
1470  23412 #61217 SUBEV  JSM FETCH      EVALUATE EXPRESSION
1471  23413 160225        JSM FLTRA,I    CONVERT TO INTEGER
1472  23414 164341        JMP E40+2,I
1473  23415 #04132        ADB M1
1474  23416 #75712        SBM *-2        0 OR NEGATIVE?
1475  23417 #45472        ISZ TEMPS
1476  23420 #64563        JMP SLWST      NO
1477*
```

#01 STRING OPTION BLOCK

```
1479*
1480*    PSTR SUBROUTINE
1481*
1482*    EVALUATE A STRING OPERAND
1483*
1484*    ENTRY CONDITIONS:
1485*
1486*      B=-2 FOR SOURCE STRINGS
1487*       B=-1 FOR DESTINATION STRING
1488*       LOW STACK HAS SECOND SUBSCRIPT.
1489*       FIRST SUBSCRIPT, STRING ADDRESS,
1490*       RESPECTIVELY.
1491*       TMPRE=SOURCE LENGTH
1492*
1493*    EXIT CONDITIONS:
1494*
1495*       TMP2=STRING POINTER
1496*       B=LENGTH
1497*       TEMP6=STRING LENGTH WORD POINTER
1498*       A=TMP2 IF SOURCE STRING
1499*      TNULL=1'S COMPLEMENT OF REQUESTED LENGTH
1500*
1501*
1502  23421 035655 PSTR  STB PS0        SAVE MODE FLAG
1503  23422 025412       LDB LSTPT
1504  23423 004134       ADB M3
1505  23424 035412       STB LSTPT      BUMP LOW STACK POINTER
1506  23425 004012       ADB .1
1507  23426 035656       STB PS1        SET FLAG POSITIVE
1508  23427 074517       LDA B,I        GET STRING ADDRESS
1509  23430 031742       STA TMP2       SAVE IT
1510  23431 070002       SAR 1
1511  23432 000132       ADA M1         GET LENGTH WORD POINTER
1512  23433 031651       STA TEMP6      SAVE IT
1513  23434 004012       ADB .1
1514  23435 074517       LDA B,I        GET FIRST SUBSCRIPT
1515  23436 031652       STA MPT        SAVE IT
1516  23437 001742       ADA TMP2       RECORD STRING
1517  23440 031742       STA TMP2         START POINTER
1518  23441 004012 PS2   ADB .1         GET SECOND SUBSCRIPT
1519  23442 074517       LDA B,I        SPECIFIED?
1520  23443 070513       SAP PSTR2      NO, SOURCE MODE?
1521  23444 011655       CPA PS0        NO
1522  23445 067451       JMP PSTR1      YES
1523  23446 121651       LDA TEMP6,I    GET ACTUAL STRING LENGTH
1524  23447 050105       AND B377
1525  23450 067454       JMP PSTR2-1
1526  23451 031656 PSTR1 STA PS1        SET FLAG TO -1
1527  23452 021647       LDA TMPRE      COMPUTE END OF
1528  23453 001652       ADA MPT        STRING DESIGNATOR
1529  23454 000132       ADA M1
1530  23455 031653 PSTR2 STA NUT        SAVE IT
1531  23456 070056       CMA
1532  23457 001652       ADA MPT
1533  23460 000132       ADA M1
1534  23461 070152       SAM PSTR8      NEGATIVE LENGTH?
1535  23462 160205       JSM AERRA,I    YES
1536  23463 177670       DEC -72
1537  23464 031654 PSTR8 STA TNULL      SAVE 1'S COMPLEMENT OF LENGTH
1538  23465 121651       LDA TEMP6,I    DOES
1539  23466 050105       AND B377         START OF
1540  23467 070056       CMA            CHARACTER
1541  23470 045655       ISZ PS0              RELATE TO
1542  23471 070070       STA **1                PREVIOUS STRING
1543  23472 001652       ADA MPT                  VALUE
1544  23473 070112       SAM **2
1545  23474 067515       JMP PSTR3
1546  23475 121651       LDA TEMP6,I    YES
1547  23476 045655       ISZ PS0        DESTINATION MODE?
1548  23477 070346       RAR 8          YES, USE PHYSICAL LENGTH
1549  23500 050105       AND B377       NO USE ACTUAL LENGTH
1550  23501 000012       ADA .1
1551  23502 070076       TCA
1552  23503 001653       ADA NUT
1553  23504 024012       LDB .1
1554  23505 015655       CPB PS0        SOURCE MODE?
1555  23506 067525       JMP PSTR5      NO
1556  23507 025654       LDB TNULL      SPECIFIED SOURCE
1557  23510 000012       ADA .1           STRING CONTAINED
1558  23511 070452       SAM PSTR4          WITHIN DEFINED SOURCE STRING?
1559  23512 004012       ADB .1         NO,
1560  23513 070037       ADB A            CORRECT LENGTH
1561  23514 067522       JMP PSTR4
1562  23515 021655 PSTR3 LDA PS0        SOURCE MODE?
1563  23516 072150       RZA PSTR4-1    YES
1564  23517 160205       JSM AERRA,I    NO
1565  23520 177667       DEC -73
1566  23521 024132       LDB M1
1567  23522 021742 PSTR4 LDA TMP2       NO, GET STRING POINTER
1568  23523 074056       CMB
```

#01 STRING OPTION BLOCK

```
1569  23524 170402        RET
1570  23525 070152 PSTR5  SAM *+3         OVERFLOW?
1571  23526 160205        JSM AERRA,I     YES
1572  23527 177666        DEC -74
1573  23530 045656        ISZ PS1         NO, STRING END SPECIFIED?
1574  23531 067543        JMP PSTR7       YES
1575  23532 045653        ISZ NQT
1576  23533 067534        JMP *+1
1577  23534 121651 PSTR6  LDA TEMP6,I     NO
1578  23535 050157        AND M256          RESET
1579  23536 041653        IOR NQT             STRING
1580  23537 131651        STA TEMP6,I           ACTUAL LENGTH
1581  23540 050105        AND B377
1582  23541 070137        LDB A
1583  23542 170402        RET
1584  23543 121651 PSTR7  LDA TEMP6,I     IS NEW
1585  23544 050105        AND B377          DESTINATION
1586  23545 070076        TCA                 STRINT
1587  23546 001653        ADA NQT               LONGER
1588  23547 071153        SAP PSTR6-2             THAN OLD?
1589  23550 170402        RET               NO
1590*
1591*
1592  23551 121472 GQUOT  LDA TEMPS,I
1593  23552 050105        AND B377        GET STRING LENGTH
1594  23553 070137        LDB A
1595  23554 021472        LDA TEMPS
1596  23555 000012        ADA .1
1597  23556 031651        STA TEMP6
1598  23557 074117        LDA B
1599  23560 070070        SIA *+1
1600  23561 070002        SAR 1
1601  23562 001651        ADA TEMP6
1602  23563 031472        STA TEMPS       UPDATE TEMPS
1603  23564 021651        LDA TEMP6
1604  23565 070744        SAL 1
1605  23566 170402        RET
1606*
1607*
1608*
1609*    FIND VARIABLE ADDRESS
1610*
1611  23567 160177 GSYMB  JSM SSYMA,I     SYMBOL IN TABLE?
1612  23570 160340        JSM E40+1,I      NO
1613  23571 170402        RET
1614*
1615*
1616*
1617  23572 164172 EDATA  JMP XEC4A,I
1618  23573 066574 EILET  JMP LTEX
1619  23574 167604 WREX   JMP RITE,I
1620*
1621  23575 167605 PREX   JMP PRINT,I
1622*
1623  23576 167606 DSPEX  JMP DISP,I
1624*
1625*
1626*
1627*
1628*
1629*    LINKS THAT MOVE WITH MONIT
1630*
1631  23577 010202 EGOS1  OCT 10202
1632  23600 012233 VAR1   OCT 12233
1633  23601 004171 STMTA  OCT 4171
1634  23602 016524 SALT   OCT 16524
1635  23603 023773 TABA   OCT 23773
1636  23604 007072 RITE   OCT 7072
1637  23605 006745 PRINT  OCT 6745
1638  23606 006746 DISP   OCT 6746
1639  23607 012436 STRID  OCT 12436
1640  23610 012455 STRI2  OCT 12455
1641  23611 010437 SWITH  OCT 10437
1642  23612 004117 REEDX  OCT 4117
1643  23613 004120 REEDY  OCT +120
1644*
1645*
1646*
1647  23614 176660        ABS EVAL-*
1648  23615 176566        ABS EINDX-*
1649  23616 176545        ABS ELEN-*
1650  23617 046511 ERIL   ASC 11,MISSING COMMA, RETYPE
      23620 051523
      23621 044516
      23622 043440
      23623 041517
      23624 046515
      23625 040454
      23626 020122
      23627 042524
```

```
          #01  STRING OPTION BLOCK


       23630 #54520
       23631 #42440
 1651  23632 #41101 ER2L   ASC 8,BAD STRING INPUT
       23633 #42040
       23634 #51524
       23635 #51111
       23636 #47107
       23637 #20111
       23640 #47120
       23641 #52524
 1652  23642 #00054 B54    OCT 54
 1653  23643 #00021 .17    DEC 17
 1654  23644 #00377 NTAB   OCT 377
 1655  23645 #01645 PTR1A  DEF PTR1
 1656  23646 #46105 FTAB   ASC 1,LE
 1657  23647 #47230        OCT 47230      N, OPCODE 30
 1658  23650 #50117        ASC 1,PO
 1659  23651 #51631        OCT 51631      S, OPCODE 31
 1660  23652 #53101        ASC 1,VA
 1661  23653 #46332        OCT 46332      L, OPCODE 32
 1662*
 1663*
 1664*
 1665*
 1666*    SYNTAX JUMP TABLE
 1667*
 1668*
 1669  23654 176141        ABS LETS-*
 1670  23655 176425        ABS READS-*
 1671  23656 176424        ABS READS-*    INPUT
 1672  23657 176442        ABS DATAS-*
 1673  23660 176330        ABS PRINS-*    DISP
 1674  23661 176233        ABS IFSYN-*
 1675  23662 176322        ABS WRTS-*
 1676  23663 176325        ABS PRINS-*
 1677  23664 176263        ABS IMPLT-*
 1678*
 1679*    SYNTAX TABLE
 1680*
 1681  23665 #51105 SYN    ASC 2,READ
       23666 #40504
 1682  23667 104111        OCT 104111     OPCODE 8,I
 1683  23670 #47120        ASC 2,NPUT
       23671 #52524
 1684  23672 103504        OCT 103504     OPCODE 7,D
 1685  23673 #40524        ASC 1,AT
 1686  23674 #40606        OCT 40606      A, OPCODE 6
 1687  23675 #00241        OCT 241        IMPLIED LET
 1688  23676 #42111        ASC 2,DISP
       23677 #51520
 1689  23700 122511        OCT 122511      OPCODE 5,I
 1690  23701 #43204        OCT 43204      F, OPCODE 4
 1691  23702 #53522        ASC 2,WRIT
       23703 #44524
 1692  23704 #42643        OCT 42643      E, OPCODE 3
 1693  23705 #50122        ASC 2,PRIN
       23706 #44516
 1694  23707 #52242        OCT 52242      PRINT, OPCODE 2
 1695  23710 #46105        ASC 1,LE       LE
 1696  23711 #52351        OCT 52351      T, OPCODE 11
 1697*
 1698*
 1699*
 1700  23712 #03626 ER3A ABS ***+2-ID1-ID1
 1701  23713 #41101        ASC 9,BAD NUMERIC INPUT,
       23714 #42040
       23715 #47125
       23716 #46505
       23717 #51111
       23720 #41440
       23721 #44516
       23722 #50125
       23723 #52054
 1702  23724 #03453 ER4A   ABS ER1L+ER1L+13-ID1-ID1
 1703  23725 #03436 ER1A   ABS ER1L+ER1L-ID1-ID1
 1704  23726 #03464 ER2A   ABS ER2L+ER2L-ID1-ID1
 1705  #1640         CP0    EQU ST
 1706  #1641         CP1    EQU ST+1
 1707  #00160        B174K  EQU M1024
 1708  23727 #00131 QMRKA  DEF .63+.63+1
 1709  #1644         INTMP  EQU ST+4
 1710  23730 174716 LENA   ABS LEN+LEN-ID-ID
 1711  #1645         PTR1   EQU ST+5
 1712  #1646         PTR2   EQU ST+6
 1713  #1647         TMPRE  EQU ST+7
 1714  #1742         TMP2   EQU AC+2
 1715  #1640         TEPPP  EQU ST
 1716  #1641         SPRDD  EQU ST+1
 1717  #1643         SAVE   EQU ST+3
 1718  #1651         TEMP6  EQU ST+9
```

#01 STRING OPTION BLOCK

```
1719  01652          MPT    EQU ST+10
1720  01653          NQT    EQU ST+11
1721  01654          TNULL  EQU ST+12
1722  01655          PS0    EQU ST+13
1723  01656          PS1    EQU ST+14
1724  23731 001742   TMP2A  DEF TMP2
1725  23732 011761   MSDAT  OCT 11761
1726  23733 003772   STDAT  OCT 3772
1727  01462          REEOL  EQU 1462B
1728*
1729*
1730  23762                 ORG 23762B
1731*
1732*
1733*     STATEMENT EXECUTION TABLE
1734*
1735  23762 176612          ABS LIEX-*
1736  23763 177302          ABS EREAD-*
1737  23764 176714          ABS EINPT-*
1738  23765 177605          ABS EDATA-*
1739  23766 177610          ABS DSPEX-*
1740  23767 177202          ABS IFEX-*
1741  23770 177604          ABS WREX-*
1742  23771 177604          ABS PREX-*
1743  23772 177601          ABS EILET-*
1744*
1745*
1746  23773 176353          ABS LEN-*
1747  23774 177652          ABS FTAB-*
1748  23775 177647          ABS NTAB-*
1749  23776 177667          ABS SYN-*
1750  23777 004000 ID       OCT 4000
1751                        END
*  NO ERRORS*
```

CROSS-REFERENCE TABLE

```
.1      0020    0896    0911    0992    1190    1263    1385    1506    1513    151A    1550
        1553    1557    1559    1596
.10     0071    0072
.100    0070
.1000   0069
.11     0035    0043
.12     0036
.13     0039
.15     0048    0083
.16     0049    0084    1249
.17     1653    1143
.1F4    0068
.2      0027    1116    1343    1460
.21     0050    1154
.22     0051    0087
.23     0052    0088
.28     0053
.3      0028    0045
.31     0054    0090
.32     0055    1056    1273    1316    1350
.33     0056
.34     0057    0091
.37     0058
.4      0029
.40     0059
.41     0060
.43     0061
.45     0062
.46     0063
.48     0064
.5      0030    1334    1341
.58     0065    0094
.6      0031    0044    1339
.63     0066    1708    1708
.7      0032
.8      0033    1257
.9      0034    0585
.99     0074
.BUFA   0180
.EIN1   1115    1140
.FADA   0185
.FDA1   1399    0514    1365    1381
.FDVA   0188
.FMPA   0187
.FSBA   0186
.FSR    0190
.LNUM   0314
.NPWR   0492    1016    1261    1266
.SLBF   0181
```

CROSS-REFERENCE TABLE

A	0000	1038	1047	1187	1305	1326	1560	1582	1594	
AB1	0615	0609								
AB3	0605									
ABADD	0718	0713								
ABORT	0604	0541	0542	0612	0634	0635	0693	0718	0793	0812
ACTSA	0174									
ADRSS	0327									
AERRA	0176	0656	0900	0960	1369	1387	1535	1564	1571	
AR1	0411	0182								
AR1A	0182									
AR2	0413	0183								
AR2A	0183									
ARYAD	0401									
B	0000	0531	1170	1184	1191	1310	1440	1508	1514	1519	1598
B100	0095									
B1000	0102									
B10K	0114	0115								
B12K	0116	0117	0206	0694						
B133	0096									
B140	0097									
B1400	0103	0204								
B17	0083									
B170K	0121									
B1740	0104	0203								
B174K	1707	1171								
B1760	0105	0158								
B177	0098									
B1774	0106									
B1775	0107									
B1777	0108									
B20	0084									
B22	0085									
B23	0086									
B2400	0110									
B26	0087									
B27	0088									
B3000	0111									
B36	0089	1103	1138	1360						
B37	0090	1102	1137	1359	1459					
B377	0099	1524	1539	1549	1581	1585	1593			
B400	0100									
B4000	0112	0162								
B42	0091	0727	0736	0807	1076	1208	1239	1244		
B44	0092	0576								
B52	0093									
B54	1652	1133								
B6K	0113	0167								
B72	0094									
B777	0101									
BADDR	0291	0667	0669	0705	0760	0761	0768	0929	0938	0946	0954
	0958	0971	0974	1109	1130	1210	1279				
BADRP	0204									
BEND	0198	0199								
BHSTP	0442									
BLANK	0386	1179	1274							
BR1	1333	1330								
BR2	1339	1333								
BRANC	1326	1313								
C5	0015									
C6	0011	0114								
C7	0022	0100								
CHRSA	0219									
CKM1	0908									
CKMOD	0904	0841	0928							
CLAR2	0489									
CLERA	0160	1013	1093	1142	1153	1160	1165	1248	1283	
CLINE	0316									
CLP1A	0217									
COMCE	0434									
COMCK	0433	0783	0803	0918						
COMP2	1299	1312								
COMP3	1303	1300								
COMP4	1306	1302								
COMP5	1309	1317	1324							
COMP6	1314	1307								
COMP7	1318	1315								
CONST	0482	0800	0955	1121						
CP0	1705	0867	0875	0893	1295	1299				
CP1	1706	0879	0882	1298	1314	1319				
CPLAC	0331									
CRCKA	0228									
CRLF	0495									
CRLF1	0496									
CRTN	0037	0038	0039							
CTYPE	0330									
CXEC4	1165	1128								
DATA1	0806	0801								
DATA2	0803	0811								
DATAS	0798	0805	1672							
DCOMA	0210									

CROSS-REFERENCE TABLE

DELSA	0175						
DFLAG	0388	0622					
DIGCK	0461	0610					
DISP	1638	1623					
DPERR	0043						
DRQST	0317	1107	1120	1124	1264	1271	1277	1281
DSPEX	1623	1739					
DSTRT	0303	1364	1380	1414			
E	0080						
E1	0235						
E10	0244						
E15	0249	0919					
E20	0254	0554	0653	0809	0922		
E25	0259	0792					
E30	0264	0545					
E35	0269						
E40	0274	1472	1612				
E45	0279						
E5	0239	0692	0717				
E50	0284	1404					
ECALA	0215						
EDATA	1617	1738					
EGOS1	1631	1335	1345				
EILET	1618	1743					
EIN14	1203	1253					
EIN15	1242	1236					
EIN16	1237	1241					
EIN18	1224	1234					
EIN19	1226	1240	1245				
EIN2	1097	1151	1158	1161			
EIN20	1231	1216					
EIN21	1235	1214					
EIN3	1129	1230					
EIN4	1181	1139					
EIN5	1206	1202					
EIND1	0890	0883					
EIND2	0877	0887	0891				
EIND3	0888	0872					
EINDX	0863	1648					
EINP6	1227	1225	1238	1243			
EINPT	1088	1737					
ELEN	0841	1649					
ELEN1	0842	0889	0897				
ELEN2	0843	0959					
ELINA	0169						
ENDS	0478						
ENTEA	0218						
ENTFA	0189						
EOL	0038	0741	0770	1074	1224		
EOLCK	0481						
EOST	0479	0555	0558	0655	0714		
ER1	1363						
ER1A	1703	1155					
ER1L	1650	1702	1702	1703	1703		
ER2	1373	1396					
ER2A	1704	1250					
ER2L	1651	1704	1704				
ER3	1379						
ER3A	1700	1144					
ER4	1257	1145	1146	1156	1157	1251	1252
ER4A	1702	1258					
EREAD	1358	1376	1736				
ERRA	0177						
ESAVE	1170	1088					
EVAL	0928	1647					
EVAL1	0943	0950					
EXP	0399	0400					
EXPOP	0118						
F	0081						
FD2	1402	1412					
FD3	1413	1409	1411				
FDAAA	0222	1371					
FETCH	0486	1470					
FILLA	0201						
FINCH	1070	1080	1211				
FIRST	0365						
FIX	0499						
FLGBT	0873						
FLOAT	0384						
FLTFA	0193						
FLTRA	0192	1471					
FLTSA	0195						
FN	0148						
FND	0460						
FND1A	0224						
FNDAA	0223						
FNDPS	0474						
FOPBS	0206						
FORMA	0172	1115	1362				
FORME	0336						
FORMS	0335						

CROSS-REFERENCE TABLE

FORMT	0334									
FPAGE	0449									
FROMA	0122									
FSCA	0173	0587	0591	0758	0769					
FSCE2	0439									
FSCE4	0444									
FSCH	1348	1031								
FTAB	1656	1747								
FTMP1	0337									
FTMP2	0338									
FWAM	0157									
FWUP	0296									
FXFLA	0194	0842								
GA1	1455	1450	1463							
GA2	1451	1446								
GADD	1437	0511	0527	0986	0998	1181	1377			
GETAD	0459									
GETCH	0469	0878	0881	0944	1061	1073	1304	1322	1355	
GETSK	0473	0734	1131	1237	1242					
GFLAG	0402									
GFRST	0472	0604	1204	1269						
GNEXT	0471	0605	0668	0970	1201	1226				
GOTOE	1335	1331								
GP1	1004	0912	0993							
GP2	0997	0913								
GPARM	0422									
GPTRS	0986	0866	1007	1293						
GQUOT	1592	1004	1389							
GSIGN	0484									
GSYMB	1611	1438								
GTCON	0498									
GTE	0827	0659	0659							
GTEA	0659	0636								
GTMP	0478									
GTOPP	0464	1437								
GTSTA	0213	0557	0648	0740	0810	0859				
GVAR	0966	0849	0915							
HFLG1	0321									
HSTPT	0302	0940	1149	1150						
I	0067	0951								
ICNT	0420									
ID	1750	0658	0658	0659	0659	1710	1710			
ID1	0509	1162	1163	1700	1700	1702	1702	1703	1703	1704	1704
IF1	0644									
IF2	0649	0647								
IF3	0648	0645								
IFERR	0656	0640	0646							
IFEX	1293	1740								
IFSYN	0633	1674								
ILET1	0680	0671	0674	0675	0678					
ILET2	0690	0688								
IMPLT	0665	1677								
IMPY	0485									
INC1	1281	1268	1270							
INC2	1285	1282								
INCAL	1263	1180	1203	1284						
INCH1	1160	1132								
INCH2	1153	1135								
INCH5	1142	1122								
INCHT	1147	1162								
INDS2	0918	0925								
INDXS	0915	0865								
INF	0077									
INITS	0454									
INLN1	1162	1118								
INLNK	1163	1105								
INREA	0226									
INRET	1108	1163								
INSTR	1176	1104								
INTCK	0457									
INTEA	0178									
INTFL	0075									
INTGR	0400									
INTMP	1709	0874	0892	0989	1200	1217	1246	1327	1400	1415
IOBUF	0357									
IOHEA	0229	0703								
IOSTA	0225									
KEYFG	0324									
L2T1	0345									
L2T2	0346	0530								
LASTN	0396									
LBOP	0119	0583	1449							
LDEF	0487									
LEN	0824	1710	1710	1746						
LENA	1710									
LENER	0900	0850	0851	0916	0917					
LENS	0849	0907								
LENS1	0852	0860	0923							
LENS2	0858	0921								
LETCK	0462	0670								

CROSS-REFERENCE TABLE

LETS	0539	1669								
LIMIT	0463									
LISTA	0167									
LNFD	0072									
LNINC	0315									
LNUM	0455	0654								
LOCL1	0343	1407								
LOCL2	0344									
LPCK	0435	0581								
LPCKE	0436	0973								
LPOP	0120									
LROMA	0123									
LSTAK	0299									
LSTPT	0301	0935	1182	1503	1505					
LT2	1017	1010	1012							
LT4	1008									
LT5	1032									
LTEX	1007	1618	1735							
LTR	0477	0574								
LTSYM	0552	0679								
LWAM	0295									
M1	0124	0616	0732	0745	1008	1186	1192	1195	1221	1318	1352
	1392	1439	1453	1473	1511	1529	1533	1566			
M10	0131									
M100	0144									
M1024	0146	1707								
M11	0132									
M12	0133									
M13	0134									
M14	0135									
M15	0136									
M2	0125	0528	0869	0999	1095	1183	1296			
M23	0137									
M25	0138									
M256	0145	1578								
M3	0126	1504								
M32	0139									
M36	0140									
M4	0127	1336								
M48	0141									
M5	0128									
M6	0129									
M64	0142									
M80	0143									
M9	0130									
MASK1	0153									
MASK5	0154									
MAW	0294									
MAXIN	0398									
MAXSN	0147									
MBIN1	0404									
MBIN2	0405									
MBOX1	0349									
MDIMA	0208									
MER8	0443	0948								
MINIT	0453									
MODE	0305	0606	0685	0863	0904	1011	1278			
MPT	1719	1515	1528	1532	1543					
MSDAT	1725	1408								
MSFLG	0387	0586								
MSL	0527	0509								
MVTOH	0475									
NDISP	0380	1177								
NDWRT	0494									
NEXT1	0448									
NEXTA	0465	1111	1228							
NEXTL	0447	1406								
NMOTA	0200									
NOEOF	0480									
NPRIN	0381	1178								
NPWRT	0493									
NOT	1720	1530	1552	1575	1579	1587				
NSTMT	0688	0686								
NTAB	1654	1748								
NUM	0394	0395	0619	0815						
NUMCA	0211									
NUMOA	0212	0802								
NXTDT	0304	1367								
NXTST	0403									
OBLNK	0466									
ODGIT	0490									
ONE	0045									
ONEXT	0467	0945	0952							
OPCHA	0171									
OPDMK	0079	1328								
OPMSK	0076	0994	0995	1448						
OPTYP	0395									
OTPTR	0293									
OUTCR	0468	1033	1064	1079						
OUTIA	0179									

CROSS-REFERENCE TABLE

OVCHA	0216										
PARMA	0306										
PARMN	0307										
PBPTO	0313										
PBPTR	0310	1403									
PBUFF	0311										
PBUFO	0312										
PEND	0363	0942	0947								
PEXMA	0165	1108									
PFLAG	0389	0733	0746	0748							
PGIN0	0476										
PGINT	0456										
PI/2	0044										
PLACE	0382										
PMODE	0329										
PPFF	0364										
PR1	0773	0735									
PR2	0736	0728	0731								
PR3	074A	0738									
PR4	0760	0752									
PR5	0770	0759	0766								
PR6	0767	0764									
PRADD	0308	1098	1112	1127	1229	1373	1399	1401	1402	1413	1416
	1445										
PREX	1621	1742									
PRFLG	0332										
PRIN1	0712	0689	0784	0804							
PRIN2	0715	0711									
PRINS	0709	1673	1676								
PRINT	1637	1621									
PRNTS	0732	0687	0710								
PRSYN	0727	0747	0772								
PRTOP	0694	0681									
PS0	1722	1502	1521	1541	1547	1554	1562				
PS1	1723	1215	1235	1507	1526	1573					
PS2	151A										
PSTR	1502	0512	0529	0870	1000	1009	1196	1297	1393		
PSTR1	1526	1522									
PSTR2	1530	1520	1525								
PSTR3	1562	1545									
PSTR4	1567	1558	1561	1563							
PSTR5	1570	1555									
PSTR6	1577	1588									
PSTR7	1584	1574									
PSTR8	1537	1534									
PTR1	1711	0885	0931	1002	1015	1024	1039	1040	1062	1067	1198
	1206	1391	1655								
PTR1A	1655	0877	0943	1060	1303	1354					
PTR2	1712	0852	0868	0884	0939	0953	0967				
QMRKA	1708	1265									
QTCHA	0230										
QTOP	0109	0161									
QUOCK	0556	0553									
RBOP	0117	0593	0756								
RBUFF	0358										
RCOUT	0333										
RDSTR	1377	1361									
RDYDA	0166										
READ1	0786	0782									
READ2	0783	0788	0789								
READ3	0790	0781									
READS	0779	0785	1670	1671							
REED	0161										
REEDL	1727	1267									
REEDX	1642	1285									
REEDY	1643	1275									
RES	0415										
RESB1	0451										
RESBP	0452	0680									
RET2	0419										
RETN2	0418	0533	0578	0742	0771	0774					
RETN3	0417	0582	0597	0857							
RITE	1636	1619									
RPCK	0437										
RPCKE	0438	0592	0704	0854							
RPOP	0115										
RSAVE	0292										
RSLTE	0371										
RSPTR	0416	0906	1090								
RSTAK	0414										
RSTKA	0158	1089									
RSULT	0424	1125	1372								
RUNA	0163										
S	0082										
SALT	1634	0625									
SALTA	0207										
SAVBP	0450	0672									
SAVE	1717	1106	1119	1172	1259	1260					
SAVEA	0325										

CROSS-REFERENCE TABLE

SAVEB	0326										
SAVEP	0815	0539	0573	0665	0709	0779	0798				
SBADR	0392										
SBPTR	0385	0556	0584	0594	0595	0596	0615	0641	0643	0682	0683
	0684	0706	0739	0744	0757	0773	0799	0858	0972	1117	
SBPUD	0429	0755									
SBUFA	0199	0608	0613								
SEC	0320										
SFLAG	0391	0621									
SGNS	0491										
SIGN	0397	0398									
SLCOD	0328										
SLEND	0202										
SLWST	0445	0930	1442	1452	1454	1476					
SNFLA	0197										
SPLFA	0196										
SPRDD	1716	1096	1097	1110	1227						
SSAPT	0393										
SSYMA	0170	1611									
ST	0375	0579	0618	0623	0637	0638	0650	0651	0666	0690	0712
	0715	0762	0767	0790	0806	0816	0817	0819	0856	0968	1320
	1???	1705	1706	1709	1711	1712	1713	1715	1716	1717	1718
	1719	1720	1721	1722	1723						
START	0162	0007									
STDAT	1726	1410									
STEXP	0441	0639	0652								
STMTA	1633	0627									
STOR1	0426										
STORE	0425										
STPFL	0323										
STR12	1640	0580									
STRID	1639										
STROP	0428										
STSRH	0440										
SUBEV	1470	1455	1461								
SUMPA	0227										
SVARO	0574	0510	0540	0552	0633	0644	0673	0763	0788	0920	0975
SWITH	1641	1363	1366	1378	1395						
SYE12	0483										
SYE25	0458										
SYMC1	0431	0543	0588	0676							
SYMC2	0430	0729									
SYMCK	0432										
SYMTP	0298										
SYN	1681	1749									
SYTSA	0209										
TABA	1635	0750									
TEMP	0348	0349	0617								
TEMP1	0350										
TEMP2	0351										
TEMP3	0352										
TEMP4	0353										
TEMP6	1718	0855	0969	1231	1233	1512	1523	1538	1546	1577	1580
	1584	1597	1601	1603							
TEMPS	0347	0843	0853	0908	0909	0966	0987	0988	0990	0996	0997
	1094	1099	1101	1113	1126	1136	1148	1358	1374	1379	1382
	1383	1405	1443	1444	1447	1456	1457	1462	1475	1592	1595
	1602										
TEPPP	1715	1100	1114	1129	1147						
TERM	0362	1091									
TFLAG	0318										
TFLG1	0?22										
THEN	0826	0658	0658								
THENA	0658	0649									
TMP2	1714	0532	0895	1026	1041	1042	1065	1066	1197	1207	1509
	1516	1517	1567	1724							
TMP2A	1724	0880	1032	1063	1078	1321					
TMPA	0203										
TMPOP	0184										
TMPRE	1713	0873	0886	1001	1014	1045	1050	1054	1057	1194	1219
	1348	1353	1390	1527							
TNULL	1721	0876	0890	0932	0949	1028	1035	1043	1051	1070	1199
	1213	1220	1222	1232	1247	1306	1537	1556			
TR1	1035	1027									
TR2	1028	1034	1223								
TR3	1060	1055									
TR4	1051	1068									
TR5	1063	1059									
TREGE	0370										
TRSTR	1024	0513	1017	1394							
TSRHA	0168	0751									
TSTPT	0300										
TTY	0497										
TYPFL	0078	0754									
UFLAG	0390										
UKPTR	0309										
UNITS	0319										
UNSTK	0446	0957									
UPARA	0191										
VALER	0960	0934	0956								

CROSS-REFERENCE TABLE

```
VALUE   0297
VAR1    1632     0786
VAROA   0214
WBUFF   0421
WIDTH   0383
WREX    1619     1741
WRITE   0159
WRTS    0703     1675
XC      0410     1714
XEC4A   0164     1018     1166     1332     1338     1340     1342     1344     1375     1617
XEC6A   0221
XFAR1   0423
XFAR2   0427
XFER    1001     1005
XSCRA   0220
YC      0412
ZAP     0152
ZONK    048A
```

#01 9830--BASE PAGE READ-WRITE MEMORY

```
0507                    LST
0508*
0509*  R/W TEMPORARIES
0510*
0511  01640             ORG 1640B
0512*
0513  01640 000000 YH   BSS 1
0514  01641 000000 XH   BSS 1
0515  01642 000000 XW   BSS 1
0516  01643 000000 YW   BSS 1
0517  01644 000000 CP3  BSS 1
0518  01645 000000 CP4  BSS 1
0519  01646 000000 T3   BSS 1
0520  01647 000000 T1   BSS 1
0521*
0522  01650 000000 XN   BSS 4
0523  01654 000000 XX   BSS 4
0524  01660 000000 YN   BSS 4
0525  01664 000000 YX   BSS 4
0526*
0527  01670 000000 Z1   BSS 4
0528  01674 000000 Z2   BSS 4
0529*
0530*  THESE FOUR WORDS ARE NOT SAFE DURING FUNCTIONS
0531*
0532  01734             ORG 17348
0533  01734 000000 DX   BSS 1
0534  01735 000000 DY   BSS 1
0535  01736 000000 CPX  BSS 1
0536  01737 000000 CPY  BSS 1
0537*
0538  01473       CCNT  EQU TEMPS+1
0539  01477       TEMPS EQU TEMPS+5
0540  01734       CP2   EQU DX
0541  01735       CPT   EQU DY
0542  01736       T4    EQU CPX
0543  01737       T2    EQU CPY
```

#01 9830 PLOTTER BLOCK

```
0545  22000             ORG 22000R
0546*
0547*
0548*    THE PLOT BLOCK HAS ABSOLUTE ADDRESSES TO THE
0549*    FOLLOWING BASIC SYSTEM SUBROUTINES
0550*
0551*    COS
0552*    SIN
0553*    RITE  WRITE STATEMENT SYNTAX
0554*    PRINN  PRINT STATEMENT EXCUTION
0555*    SCER1  FORMAT EXCUTION
0556*    MOVCK  MOVE THE SYMBOL TO LOWER MEMORY
0557*    TYPEC
0558*    QTCHK
0559*    FSPR
0560*    CKEOL
0561*    PFORM
0562*    KEYIN
0563*    VALUP
0564*    TBSR1
0565*
0566*  SPECIAL SYMBOL CODE'S USED IN THIS BLOCK ARE AS FOLLOWS:
0567*
0568*  01740 SCALING IMFORMATION
0569*  01760 CHARACTER DEFINITION
0570*   1660 CARRIAGE RETURN IMFORMATION
0571*
```

```
          #01  9830 PLOTTER BLOCK

#572*     CODE STARTED 10-26-71
#573*
#574*     RELATIVE ADDRESSES TO SCALEING IMFORMATION IS AS FOLLOWS
#575*
#576*     0 = X SCALE FACTOR
#577*     4 = Y SCALE FACTOR
#578*     8 = X ADJUST
#579*    12 = Y ADJUST
#580*    16 = X PRESENT POSITION
#581*    20 = Y PRESENT POSITION
#582*    24 = X OFFSET
#583*    28 = Y OFFSET
#584*
#585*       CONSTANTS AND EQUATES
#586*
#587*
#588  22000 001400 N9/6  OCT 1400       1666.66  ** DON'T MOVE - IT'S RELATIVE ADDRESS = 0
#589  22001 013146       OCT 13146
#590  22002 063146       OCT 63146
#591  22003 063146       OCT 63146
#592*
#593  22004 001400       OCT 1400       9999  ** DON'T MOVE, IT'S RELATIVE ADDRESS =4
#594  22005 114631       OCT 114631
#595  22006 000000       OCT 0
#596  22007 000000       OCT 0
#597*
#599*
#600*       GET THE PEN CONTROL PARAMETER
#601*
#602  22010 025654 PNPAR LDB XX         RECALL THE PEN CONTROL
#603  22011 074113       SBP *+2        IF PSITIVE DO CONTROL NOW
#604  22012 170402       RET            IT'S NEGTIVE SOPEN CONTROL AFTER
#605  22013 075750 .P1   SZB *-1        IF ZERO, NO PEN CONTROL
#606  22014 074251       SLB PENDN
#607*
#608  22015 062032 PENUP JSM STATU
#609  22016 075613       SBP .P1-1
#610  22017 020017       LDA .2
#611  22020 066024       JMP PENDN+3
#612  22021 062032 PENDN JSM STATU
#613  22022 075412       SBM .P1-1
#614  22023 020020       LDA .3
#615  22024 070246       RAR 6
#616*
#617  22025 062032 BUSY  JSM STATU
#618  22026 074644       SBL 3
#619  22027 075713       SBP BUSY
#620  22030 043737       IOR SC16
#621  22031 065356       JMP TTY+3
#622  22032 027737 STATU LDB SC16
#623  22033 172741       STF 1
#624  22034 176141       OTB 1
#625  22035 172741       STF 1
#626  22036 177241       LIB 1,C
#627  22037 074546       RBR 12
#628  22040 170402       RET
#629*
#630  22041 070742 AMOVE SAR 16
#631  22042 031477       STA TEMP5      ALTERNATE ENTRY FOR ABINCRMENTAL MOVE
#632*
#633  22043 023500 MOVE  LDA XMIN
#634  22044 062070       JSM .M
#635  22045 040107       IOR B1000
#636  22046 062025       JSM BUSY
#637  22047 021754       LDA AR2
#638  22050 050105       AND B377
#639  22051 040013       IOR B400
#640  22052 062025       JSM BUSY
#641  22053 023502       LDA YMIN
#642  22054 062070       JSM .M
#643  22055 040013       IOR B400
#644  22056 062025       JSM BUSY
#645  22057 021754       LDA AR2
#646  22060 050105       AND B377
#647  22061 040013       IOR B400
#648  22062 041477       IOR TEMP5
#649  22063 062025       JSM BUSY
#650*
#651  22064 025442       LDB STPFL
#652  22065 014040       CPB .31        STOP KEY HIT
#653  22066 066554       JMP EPEN       YES
#654  22067 170402       RET            NO
#655*
#656  22070 060445 .M    JSM XFAR2+1
#657  22071 160225       JSM FLTRA,I
#658  22072 001654 XMAX  DEF XX
#659  22073 074117       LDA B
#660  22074 074212       SBM *+4
#661  22075 004161       ADB MAXSN
#662  22076 074112       SBM *+2
```

```
        #01  9830 PLOTTER BLOCK


#663  22077 #23514        LDA .9999
#664  22100 #31754 STOR2  STA AR2
#665  22101 #70342        SAR 8
#666  22102 170402        RET
#667*
#668*      FIND THE ADDRESS OF THE R/W WORDS .
#669*      ON RETURN THE ADDRESS OF THE 1ST WORD IS IN B AND T1.
#670*
#671  22103 #62015         JSM PENUP
#672  22104 #20111 GETRW LDA STEAL      LOAD THE CODE WORD
#673  22105 160177        JSM SSYMA,I   LOOK FOR IT IN SYMBOL TABLE
#674  22106 #66113        JMP *+5       NOT FOUND
#675  22107 #04132 GE1    ADB M1
#676  22110 #74537        LDB B,I
#677  22111 #35647        STB T1
#678  22112 170402        RET
#679*
#680  22113 160205        JSM AERRA,I
#681  22114 177660        DEC -80       NO SCALE COMMAND EXECUTED BEFORE PLOT, FCT
#682*
#683*
#684*      TRANSFER ROUTINES
#685  22115 #21646 WHERE LDA T3
#686  22116 #70352        SAM TXM
#687*
#688  22117 #27502 TYM    LDB YMIN
#689  22120 #64406        JMP XFAR1,1
#690*
#691  22121 #27504 TYX    LDB YMAX
#692  22122 #64406        JMP XFAR1,1
#693*
#694  22123 #21646 OWHER LDA T3
#695  22124 #71552        SAM TYM       SKIP TO TYM IF IN XAXIS MODE
#696  22125 #27500 TXM    LDB XMIN
#697  22126 #64406        JMP XFAR1,1
#698*
#699  22127 #26072 TXX    LDB XMAX
#700  22130 #64406        JMP XFAR1,1
#701*
#702  22131 #27772 TEMXX LDB TEMX
#703  22132 #64406        JMP XFAR1,1
#704*
#705  22133 #21737        LDA T2
#706  22134 160216        JSM .FADA,I
#707  22135 #25737 TRANS LDB T2
#708  22136 #64406        JMP XFAR1,1
#709*
#711*
#712  22137 #00051        ABS SPEN-*    LETTER SYNTAX
#713  22140 #00105        ABS OFFS-*    CPLOT SYNTAX
#714  22141 #00102        ABS SCALE-*   SCALE SYNTAX
#715  22142 #00047        ABS SAXIS-*
#716  22143 #00046        ABS SAXIS-*   XAXIS SYNTAX
#717  22144 #00103        ABS PLOT-*    PLOT SYNTAX
#718  22145 #00100        ABS OFFS-*
#719  22146 #00042        ABS SPEN-*
#720  22147 #00051        ABS LABEL-*
#721  22150 #00077        ABS PLOT-*
#722  22151 #00000        BSS 1         *** CHECKSUM WORD ***
#723*
#724  22152 #44520 SNAME ASC 1,IP       IPLOT
#725  22153 #46117        ASC 1,LO
#726  22154 #52242        OCT 52242     T
#727  22155 #46101        ASC 1,LA      LABEL
#728  22156 #41105        ASC 1,BE
#729  22157 #46243        OCT 46243     L
#730  22160 #50105        ASC 1,PE      PEN
#731  22161 #47244        OCT 47244     N
#732  22162 #47506        ASC 3,OFFSET
      22163 #43123
      22164 #42524
#733  22165 122520        OCT 122520    P
#734  22166 #46117        ASC 1,LO      PLOT
#735  22167 #52246        OCT 52246     T; OP CODE 6
#736  22170 #54101        ASC 1,XA      XAXIS
#737  22171 #54111        ASC 1,XI
#738  22172 #51647        OCT 51647     S
#739  22173 #54501        ASC 1,YA      YAXIS
#740  22174 #54111        ASC 1,XI
#741  22175 #51650        OCT 51650     S
#742  22176 #51503        ASC 2,SCAL    SCALE ; OP CODE 9
      22177 #40514
#743  22200 #42651        OCT 42651     E; OP CODE 9
#744  22201 #41520        ASC 2,CPLO
      22202 #46117
#745  22203 #52252        OCT 52252    T; OP CODE 10
#746  22204 #46105        ASC 3,LETTER  FREE MODE
      22205 #52124
      22206 #42522
#747  22207 165400        OCT 165400    OP CODE 11
#748*
#749*      END OF SNYTAX NAME TABLE
#750*
```

```
          #01  SYNTAX CODE FOR 9830 PLOTTER


0752*     PEN STATEMENT SYNTAX
0753*
0754  22210 065147 SPEN  JMP ENDS
0755*
0756*      AXIS SYNTAX
0757*
0758  22211 160202 SAXIS JSM FSCA,I     DEMAND AN OPPSITIE ORGIN
0759  22212 060504       JSM COMCK      ,?
0760  22213 065151       JMP EOST       NO DEMAND EOL
0761  22214 160202       JSM FSCA,I     YES, DEMAND A TIC PARAMETER
0762  22215 060504       JSM COMCK
0763  22216 065151       JMP EOST
0764  22217 066245       JMP OFFS
0765*
0766*      LABEL STATEMENT SYNTAX
0767*
0768  22220 060456 LABEL JSM SBPUD      UPDATE POINTER
0769  22221 060520       JSM LPCKE-1    DEMAND LEFT PAREN.
0770  22222 061055       JSM GNEXT
0771  22223 060456       JSM SBPUD
0772  22224 060463       JSM SYMC1      CHECK FOR "*"
0773  22225 012012       DEF 12012B
0774  22226 066240       JMP LAB2       NOT FOUND , MUST BE NUMBER
0775  22227 060456       JSM SBPUD
0776  22230 061055       JSM GNEXT
0777  22231 060504 LAB3  JSM COMCK     FOUND, NOW CHECK FOR OPTIONAL PARAMETERS
0778  22232 066237       JMP LAB1       P+1 , NO OPTIONS, GO DEMAND A RIGHT PAREN.
0779  22233 062254       JSM FORM3      P+2, OPTIONS ARE THERE
0780  22234 060504       JSM COMCK      CHECK IF PAPER DEMENSIONS ARE PRESENT
0781  22235 066237       JMP LAB1       NO
0782  22236 160202       JSM FSCA,I     YES, GET AFORMULA
0783  22237 167562 LAB1  JMP RITE,I
0784  22240 055706 LAB2  DSZ SBPTR
0785  22241 060663       JSM LNUM-1
0786  22242 066231       JMP LAB3
0787*
0788*      SCALE STATEMENT SYNTAX
0789*
0790  22243 062256 SCALE JSM FORM2
0791  22244 060510       JSM COMCE
0792  22245 062256 OFFS  JSM FORM2
0793  22246 065151       JMP EOST
0794*
0795*      PLOT AND IPLOT STATEMENT SYNTAX
0796*
0797  22247 062256 PLOT  JSM FORM2      DEMAND X AND Y COOR.
0798  22250 060504       JSM COMCK      CHECK FOR COMMA
0799  22251 065151       JMP EOST       "," NOT FOUND, DEMAND E.O.S.
0800  22252 160202       JSM FSCA,I     PROCESS OPTIONAL PEN COMMAND
0801  22253 065151       JMP EOST       DEMAND E.O.S.
0802*
0803  22254 160202 FORM3 JSM FSCA,I
0804  22255 060510       JSM COMCE      DEMAND ,
0805  22256 160202 FORM2 JSM FSCA,I
0806  22257 060510       JSM COMCE      DEMAND ,
0807  22260 164202       JMP FSCA,I

          #01  EXECUTION PHASE OF 9830 PLOTTER BLOCK

0809*
0810  22261 160201 FORM  JSM FORMA,I
0811  22262 160201       JSM FORMA,I
0812  22263 160201       JSM FORMA,I
0813  22264 164201       JMP FORMA,I
0814*
0815*      SCALE EXECUTION
0816*
0817  00111        STEAL EQU B1740
0818*
0819  22265 062652 ESCAL JSM CURPG      GET CURRENT PAGE
0820  22266 062261       JSM FORM      STACK 4 FORMULA
0821*
0822  22267 020111       LDA STEAL     LOAD SYMBOL FOR R/W WORDS
0823  22270 160177       JSM SSYMA,I   CHECH IF PLOTTER WORDS ARE SAVED ALREADY
0824  22271 066273       JMP *+2       NO, ENTER SAME
0825  22272 066306       JMP ES1       YES, SKIP ENTRY
0826  22273 025406       LDB VALUE     ENTER PLOTTER R/W WORDS INTO STACKS
0827  22274 023520       LDA M34       MOVE DOWN 34 PLACES, 32 FOR VALUES
0828*
0829*  MOVE H-STACK DOWN AND CHEKC FOR OVERFLOW
0830*
0831*    MOVE DOWN -(A) WORDS
0832*    (B) = ADDRESS+1 OF THE LAST WORD TO BE MOVED
0833*
0834  22275 163666       JSM MOVCK,I
0835*
0836  22276 003520       ADA M34
0837  22277 070137       LDB A         POINTER TO VALUE TABLE
0838  22300 000017       ADA .2        POINTER TO 1ST WORD OF R/W
0839  22301 074557       STA B,I       STORE ADDRESS POINTER TO PLOTTER R/W
0840  22302 074070       SIB *+1
```

```
PAGE 0019 #01  EXECUTION PHASE OF 9830 PLOTTER BLOCK


0841  22303 020111        LDA STEAL     LOAD PLOTTER R/W CODE
0842  22304 074557        STA B,I
0843  22305 035406        STB VALUE
0844*
0845  22306 062107 ES1    JSM GE1       LOAD THE ADDRESS POINTER TO 1ST WORD OF PLOTTER R/W
0846  22307 035737        STB T2        SAVE B TEMPORARLY FOR ZERING
0847  22310 070742        SAR 16        CLEAR A
0848  22311 027554        LDB M31
0849  22312 131737        STA T2,I      ZERO 33 WORDS OF R/W
0850  22313 045737        ISZ T2
0851  22314 077730        RIB *-2       LOOP BACK IF NOT ZERO
0852*
0853*
0854*  COMPUTE THE SCALE FACTOR = 9999/(YMAX-YMIN)
0855*  AND YADJUST = -(S.F.*YMIN)
0856*
0857  22315 020021        LDA .4        * SET-UP Y SCALE FACTOR
0858  22316 062322        JSM SCAL      *
0859  22317 070004        SAL 16        *
0860  22320 062322        JSM SCAL      *
0861  22321 164172        JMP XEC4A,I   RETURN TO MONITOR
0862*
0863*  SCAL ROUTINE IS USED FOR X AND Y SCALE FACTORS
0864*   AND ADJUST VALUES
0865*
0866  22322 031735 SCAL   STA CPT
0867  22323 061220        JSM FETCH+1   RECALL THE MAX
0868  22324 062127        JSM TXX
0869  22325 061220        JSM FETCH+1   RECALL THE MIN
0870  22326 062117        JSM TYM
0871  22327 024033        LDB .16
0872  22330 062344        JSM TR/W-1
0873  22331 022072        LDA XMAX
0874  22332 160217        JSM .FSBA,I   (MAX-MIN)
0875  22333 062125        JSM TXM
0876  22334 020021        LDA .4        RELATIVE ADDRESS TO 9999
0877  22335 041645        IOR CP4       INCLUDE CURRENT PAGE
0878  22336 160221        JSM .FDVA,I   9999/(MAX-MIN)
0879  22337 025735        LDB CPT       CPT = POINTER TO FIRST WORD IN R/W FOR X OR Y
0880  22340 062345        JSM TR/W      PLACE THE S.F. IN THE STACK
0881  22341 023502        LDA YMIN
0882  22342 160220        JSM .FMPA,I   S.F. * MIN
0883  22343 024025        LDB .8        SET(B) = ADJUST LOCATION POINTER
0884  22344 005735        ADB CPT
0885  22345 005647 TR/W   ADB T1
0886  22346 064406        JMP XFAR1+1
0888*
0889*       OFFSET COMMAND EXECUTION
0890*
0891  22347 062263 EOFF   JSM FORM+2
0892  22350 062104        JSM GETRW
0893  22351 023532        LDA B24
0894  22352 062356        JSM EOFF1
0895  22353 020033        LDA .16
0896  22354 062356        JSM EOFF1
0897  22355 164172        JMP XEC4A,I
0898*
0899  22356 031735 EOFF1  STA CPT
0900  22357 061220        JSM FETCH+1
0901  22360 062343        JSM TR/W-2
0902  22361 074117        LDA B
0903  22362 002610        ADA M8
0904  22363 160217        JSM .FSBA,I
0905  22364 025735        LDB CPT
0906  22365 066345        JMP TR/W
0907*
0909*
0910  22366 021646 PWHER  LDA T3        PWHER IS USED BY AXIS EXCUTION
0911  22367 070352        SAM PUNX+1
0912  22370 066372        JMP PUNY+1
0913*
0914*  ROUTINE TO GET A PARAMETER AND CONVERT IT TO PLOTTER UNITS
0915*
0916  22371 061220 PUNY   JSM FETCH+1
0917  22372 020021        LDA .4
0918  22373 027502        LDB YMIN
0919  22374 066400        JMP PUNIT
0920*
0921*PUNX FETCHES A PARAMETER AND CONVERTS IT TO PLOTTER UNITS
0922*
0923  22375 061220 PUNX   JSM FETCH+1
0924  22376 070742        SAR 16        SET A = TO AN OFFSET SO THAT A+T1 = 1ST WORD
0925  22377 027500        LDB XMIN
0926  22400 001647 PUNIT  ADA T1
0927  22401 031734        STA CP2       STORE POINTER TO 1ST X OR Y VALUE
0928  22402 000033        ADA .16
0929  22403 031736        STA T4         SAVE POINTER TO PRESENT POSITION
0930  22404 035737        STB T2        STORE POINTER TO XMIN OR YMIN
0931  22405 060406        JSM XFAR1+1     TRANSFER AR2 TO XMIN OR YMIN
0932  22406 121736        LDA T4,I       LOAD THE EXP WORD OF THE PRESENT POSITION
```

```
PAGE 0021 #01  EXECUTION PHASE OF 9830 PLOTTER BLOCK

 0933  22407 070006        RAR 1
 0934  22410 070231        SLA .PUN,C      IS BIT 1 = 1 ?
 0935  22411 070706        RAR 15          YES, CLEAR IT, AND GIVE PENUP
 0936  22412 131736        STA T4,I
 0937  22413 062015        JSM PENUP
 0938  22414 021644 .PUN   LDA CP3
 0939  22415 070153        SAP PUN1        SKIP IF IN ABS MODE
 0940  22416 025736        LDB T4
 0941  22417 062133        JSM TRANS-2     X/YMIN = LAST POSITION + DX/DY IF IN INCR.
 0942*
 0943  22420 025736 PUN1   LDB T4
 0944  22421 060406        JSM XFAR1+1     REPLACE PRESENY POSITION WITH NEW VALUE
 0945  22422 004025        ADB .8
 0946  22423 062133        JSM TRANS-2     NEW POSITION + OFFSET
 0947  22424 021734        LDA CP2
 0948  22425 160220        JSM .FMPA,I     S.F.*NEW POSITION
 0949  22426 062135        JSM TRANS
 0950  22427 074117        LDA B
 0951  22430 025734        LDB CP2
 0952  22431 004025        ADB .8
 0953  22432 160217        JSM .FSBA,I     SUBTRACT THE ADJUST VALUE
 0954  22433 062135        JSM TRANS       SAVE THIS ANSWER (PLOTTER UNITS) IN XMIN OR YMIN
 0955  22434 074742        SBR 16          CLEAR (B) FOR A FALG IF OUT OF RANGE
 0956*
 0957  22435 021754 LIMT   LDA AR2
 0958  22436 070451        SLA XLT2        WAS THE RESULT NEGATIVE
 0959  22437 021737        LDA T2          YES - CLEAR X OR Y MIN
 0960  22440 170000        CLR
 0961  22441 024033 LFLAG  LDB .16         FLAG PRESENT PEN POSITION
 0962  22442 005734        ADB CP2
 0963  22443 074517        LDA B,I
 0964  22444 040017        IOR .2
 0965  22445 074557        STA B,I
 0966  22446 170402        RET
 0967*
 0968  22447 003512 XLT2   ADA EXPM4       THE RESULT WAS NOT NEGATIVE , NOW CHECK IF
 0969  22450 071712        SAM *-2         <9999?
 0970  22451 020021        LDA .4          OTHERWISE SET XMIN OR YMIN (RELATIVE ADD. TO 9999)
 0971  22452 041645        IOR CP4
 0972  22453 025737        LDB T2          TO 9999
 0973  22454 170004        XFR
 0974  22455 066441        JMP LFLAG       AND FLAG THE LAST POSITION
 0976*
 0977  22456 075410 .XX    SZB XLT2-1
 0978  22457 024133        LDB M2
 0979  22460 035654        STB XX
 0980  22461 066015        JMP PENUP
 0981*    PLOT AND IPLOT EXECUTION
 0982*
 0983  22462 076052 EIPLT  SBM *+1,S
 0984  22463 035644 EPLOT  STB CP3
 0985  22464 074072        SBM *+1,C
 0986  22465 062652        JSM CURPG
 0987  22466 062263        JSM FORM+2      STACK X, AND Y
 0988  22467 021472        LDA TEMPS
 0989  22470 011421        CPA PRADD       PEN CONTROL?
 0990  22471 066474        JMP *+3
 0991  22472 061217        JSM FETCH
 0992  22473 160225        JSM FLTRA,I
 0993  22474 024133        LDB M2
 0994  22475 035654        STB XX
 0995*
 0996  22476 062104        JSM GETRW
 0997*
 0998  22477 062371        JSM PUNY        FETCH Y
 0999  22500 062456        JSM .XX         CHECK IF PEN IS OFF-SCALE
 1000  22501 062375        JSM PUNX        FETCH X
 1001  22502 062456        JSM .XX         CHECK IF PEN IS OFF-SCALE
 1002  22503 062010        JSM PNPAR       CHECK FOR ANY PEN CONTROL BEFORE MOVEMENT
 1003  22504 062041        JSM AMOVE       SET ABS FLAG AND MOVE PEN
 1004  22505 025654        LDB XX          LOAD THE PEN CONTROL WORD
 1005  22506 074113        SBP *+2
 1006  22507 062013        JSM .P1         AND CHECK FOR ANY PEN MOVEMENT AFTERWARDS
 1007  22510 164172        JMP XEC4A,I
 1009*
 1010*    X AND Y AXIS EXECUTION
 1011*
 1012  22511 076053 EXAXE  SBP *+1,S
 1013  22512 035646 EYAXE  STB T3
 1014  22513 074072        SBM *+1,C
 1015  22514 062652        JSM CURPG
 1016  22515 035644        STB CP3         SET PUNIT FOR ABS MODE
 1017  22516 160201        JSM FORMA,I     STACK ORIGIN
 1018  22517 021472        LDA TEMPS
 1019  22520 011421        CPA PRADD       ORIGIN ONLY?
 1020  22521 066546        JMP AX7
 1021  22522 160201        JSM FORMA,I     STACK TICK
 1022  22523 021472        LDA TEMPS
```

```
        #01  EXECUTION PHASE OF 9830 PLOTTER BLOCK


1023  22524 011421        CPA PRADD      ORIGIN, TICK ONLY?
1024  22525 066556        JMP AX3        YES
1025  22526 062263        JSM FORM+2     STACK POINT 1 AND 2
1026  22527 062103        JSM GETRW-1    DO PENUP AND GET THE R/W ADDRESS
1027  22530 062604        JSM CLIMT-2    RECALL POINT 2
1028  22531 035665        STB YX+1
1029  22532 061220        JSM FETCH+1
1030  22533 062127        JSM TXX        SAVE THE STARTING POINT
1031  22534 062605        JSM CLIMT-1
1032  22535 061220 .E1    JSM FETCH+1    FETCH THE TICK PARAMETER
1033  22536 062131        JSM TEMXX
1034*
1035*   YX = MIN VALUE IN PLOTTER UNITS
1036*   YX+1 = MAX VALUE IN PLOTTER UNITS
1037*   XMAX HAS PRESENT PEN POSITION
1038*   TEMX HAS THE TICK INCREMENT
1039*
1040*
1041  22537 062573 SLOOP JSM AX4
1042  22540 062633       JSM .TICK      PUT  PEN DOWN AND MAKE A TICK AT THE MIN POINT
1043  22541 022072 AX5   LDA XMAX
1044  22542 027772       LDB TEMX
1045  22543 160216       JSM .FADA,I    COMPUTE THE NEXT POINT
1046  22544 062614       JSM TICMK      GO TO THE TICMK ROUTINE TO CONVERT TO P.U.
1047  22545 066541       JMP AX5        MOVE THE PEN AND MAKE A TIC THEN LOOP BACK
1048*
1049  22546 062560 AX7   JSM AX2        NO TICKS OR NIN AND MAX
1050  22547 062573       JSM AX4
1051*
1052  22550 025665 AX6   LDB YX+1       TICMK WILL COME HERE IF IT GOES BEYOND MAX VALUE
1053  22551 160227       JSM FXFLA,I
1054  22552 062115       JSM WHERE
1055  22553 062602       JSM .AX4
1056  22554 062015 EPEN  JSM PENUP      DO PENUP, PEN EXCUTION ALSO
1057  22555 164172       JMP XEC4A,I    AND RETURN
1058*
1059*
1060*       SUBROUTINES FOR AXIS EXECUTION
1061*
1062  22556 062560 AX3   JSM AX2
1063  22557 066535       JMP .E1
1064*
1065  22560 062103 AX2   JSM GETRW-1    DO PENUP AND GET THE R/W ADDRESS
1066  22561 023514       LDA .9999
1067  22562 031665       STA YX+1       (B) = POINTER TO SCALE FACTOR
1068  22563 021646       LDA T3
1069  22564 070112       SAM *+2
1070  22565 004021       ADB .4
1071  22566 074117       LDA B
1072  22567 000025       ADA .8
1073  22570 160221       JSM .FOVA,I    ADJUST / SCALE FACTOR
1074  22571 062127       JSM TXX
1075  22572 066366       JMP PWHER
1076*
1077  22573 061220 AX4   JSM FETCH+1    RECALL THE OPPSITIE ORIGIN
1078  22574 021646       LDA T3
1079  22575 031664       STA YX         INITIALIZE YX TO A VALUE ABOVE 9999
1080  22576 070076       TCA
1081  22577 062367       JSM PWHER+1
1082  22600 062606       JSM CLIMT
1083  22601 035666       STB YX+2
1084  22602 062041 .AX4  JSM AMOVE
1085  22603 066021       JMP PENDN
1086*
1087  22604 061220       JSM FETCH+1
1088  22605 062366       JSM PWHER
1089  22606 076210 CLIMT SZB .CL2       IF (B) # 0 , THEN OUT OF RANGE
1090  22607 160225 .CL1  JSM FLTRA,I    PLOTTER UNITS DID NOT OVERFLOW,SO
1091  22610 177770 MA    DEC -8
1092  22611 170402       RET            RETURN AFTER CONVERTING TO INTEGER
1093  22612 160205 .CL2  JSM AERRA,I
1094  22613 177656       DEC -82        ERROR IF XO, YO, OR THE MIN , MAX VALUES OFF-SCALE
1095*
1096*     TICMK MOVES PEN AND THEN MAKES A TICK MARK
1097*      AR2 HAS THE NEXT TICK POSITION, TEMY,I HAS THE CONSTANT ORIGIN
1098*
1099  22614 062127 TICMK JSM TXX
1100  22615 062366       JSM PWHER
1101  22616 062607       JSM .CL1
1102  22617 015664       CPB YX         DID THE VALUE CHANGE?
1103  22620 066612       JMP .CL2       NO, MUST BE A ZERO TICK OR XMAX + TICK = XMAX
1104  22621 035664       STB YX         STORE A GOOD VALUE FOR THE 1ST INCREMENT
1105  22622 021670       LDA Z1         LOAD THE SIGN OF THE TICK
1106  22623 074076       TCB
1107  22624 005665       ADB YX+1
1108  22625 074150       SZB *+3        DO A TICK IF ZERO
1109  22626 070111       SLA *+2
1110  22627 076212       SBM .TICK,S    - DIRECTION
1111  22630 074153       SBP .TICK      + DIRECTION
1112  22631 055777 TBL   DSZ RSPTR
```

```
        #01  EXECUTION PHASE OF 9830 PLOTTER BLOCK


1113  22632 066550        JMP AX6       FINISH THE AXIS AND RETURN
1114  22633 062041 .TICK  JSM AMOVE
1115  22634 025666        LDB YX+2      RECALL THE OPSITIE AXIS
1116  22635 004052        ADB .48
1117  22636 074117        LDA B
1118  22637 000161        ADA MAXSN     (-10000)
1119  22640 070113        SAP *+2
1120  22641 062647        JSM .TIC      OUTPUT PLUS TICK
1121  22642 024153        LDB M48
1122  22643 005666        ADB YX+2
1123  22644 074112        SBM *+2
1124  22645 062647        JSM .TIC
1125  22646 025666        LDB YX+2
1126  22647 160227 .TIC   JSM FXFLA,I
1127  22650 062123        JSM OWHER
1128  22651 066041        JMP AMOVE
1129*
1131*
1132  22652 074442 CURPG  SBR 10
1133  22653 074304        SBL 10
1134  22654 035645        STB CP4
1135  22655 170402        RET
1136*
1137  22656 026072 ROT2   LDB XMAX
1138  22657 160221        JSM .FDVA,I    / A.R
1139  22660 062131 ROT4   JSM TEMXX
1140  22661 023504 ROT1   LDA YMAX
1141  22662 160220        JSM .FMPA,I      * HEIGHT
1142  22663 160225        JSM FLTRA,I      CONVERT TO INTGER
1143  22664 066672        JMP PRERR     OVERFLOW RETURN
1144  22665 074117        LDA B
1145  22666 070112        SAM *+2
1146  22667 070076        TCA
1147  22670 000122        ADA B6K
1148  22671 071213        SAP CURPG+3   RETURN
1149  22672 160205 PRERR  JSM AERRA,I
1150  22673 177657        DEC -81       CHARACTER SIZE TO BIG, OR BINARY FORMAT NOT ALLOWED
1151*
1152*    SET DEFAULT HEIGHT APPROX = 1.5%, A.R. =2
1153*
1154  22674 062652 CDEF   JSM CURPG
1155  22675 062015        JSM PENUP
1156  22676 020112        LDA B1760
1157  22677 160177        JSM SSYMA,I   SERACH THE SYMBOL TABLE FOR CHAR DEFINITION
1158  22700 066720        JMP .CDEF
1159  22701 062107        JSM GE1        FOUND
1160  22702 035640        STB YH
1161  22703 074070        SIB *+1
1162  22704 035641        STB XH
1163  22705 074070        SIB *+1
1164  22706 035642        STB XW
1165  22707 074070        SIB *+1
1166  22710 035643        STB YW
1167  22711 023502 GETLC  LDA PPPOS
1168  22712 160177        JSM SSYMA,I   SET UP A LOCATION FOR CRLF'S
1169  22713 066715        JMP *+2
1170  22714 066107        JMP GE1
1171  22715 160252        JSM ENTEA,I
1172  22716 062107        JSM GE1
1173  22717 067376        JMP .PC
1174*
1175  22720 160252 .CDEF  JSM ENTEA,I
1176  22721 062701        JSM CDEF+5
1177  22722 021640        LDA YH
1178  22723 170000        CLR
1179  22724 020105        LDA B377
1180  22725 131640        STA YH,I
1181  22726 020104        LDA B177
1182  22727 131642        STA XW,I
1183  22730 170402        RET
1184*
1185  22731 020020 ELAB3  LDA ONE
1186  22732 060445        JSM XFAR2+1
1187  22733 066757        JMP .ELAB
1188*
1189*    LETTING SUBROUTINE TO SET-UP CHARACTER DEFINITION
1190*
1191  22734 062652 ELAB   JSM CURPG     SAVE CURRENT PAGE
1192  22735 045472        ISZ TEMPS
1193  22736 023516        LDA FMTOP
1194  22737 031640        STA 1640B     SET OP CODE FOR FORMAT SREACH
1195  22740 074742        SBR 16       SET REPEAT COUNT TO ZERO
1196  22741 163572        JSM PLOTE,I   GO CHECK FOR FORMAT STATEMENT
1197  22742 074742        SBR 16        P+1, NOT FOUND
1198  22743 035646        STB T3        P+2, FOUND
1199  22744 062675        JSM CDEF+1
1200  22745 121472        LDA TEMPS,I
1201  22746 050065        AND OPMSK     CHECK IF THERE ARE MORE PARAMETERS
1202  22747 010004        CPA RPOP      WAS IT A RIGHT PAREN
1203  22750 067023        JMP DEFLT
1204*
```

```
          #01  EXECUTION PHASE OF 9830 PLOTTER BLOCK


1205  22751 062262        JSM FORM+1     STACK 3 FORMULA'S
1206  22752 121472        LDA TEMPS,I
1207  22753 050065        AND OPMSK      GET NEXT OPERAND
1208  22754 010004        CPA RPOP       WAS IT A PAREN
1209  22755 066731        JMP ELAB3      DEFAULT THE PAPER RATIO
1210  22756 061217        JSM FETCH      FETCH THE PAPER RATIO
1211  22757 062125 .ELAB  JSM TXM
1212  22760 061220        JSM FETCH+1    ROTATION
1213  22761 062117        JSM TYM
1214  22762 061220        JSM FETCH+1    ASPECT RATIO
1215  22763 062127        JSM TXX
1216  22764 061220        JSM FETCH+1    HEIGHT
1217  22765 063352        JSM .MPY3      REDUCE HEIGHT BY 1/10
1218  22766 062121        JSM TYX
1219*
1220*       PROCESS CHARACTER DIMENSION
1221*
1222  22767 021645        LDA CP4
1223  22770 160220        JSM .FMPA,I
1224  22771 062121        JSM TYX
1225  22772 023502        LDA YMIN
1226  22773 163566        JSM SIN,I
1227  22774 027522        LDB TEMY
1228  22775 060406        JSM XFAR1+1
1229  22776 023502        LDA YMIN
1230  22777 163570        JSM COS,I
1231  23000 062117        JSM TYM        YMIN = COS
1232*
1233  23001 062661        JSM ROT1       YH = 9999/6 * H * COS TH
1234  23002 135640        STB YH,I
1235*
1236  23003 023522        LDA TEMY       SIN TH
1237  23004 062656        JSM ROT2       XH = 9999/6 * H/A.R. * SIN TH
1238  23005 135641        STB XH,I
1239*
1240  23006 023502        LDA YMIN       COS TH
1241  23007 027500        LDB XMIN
1242  23010 160220        JSM .FMPA,I
1243  23011 062131        JSM TEMXX
1244  23012 023772        LDA TEMX
1245  23013 062656        JSM ROT2       XW = 9999/6 * H/A.R. * COS TH * PH/PW
1246  23014 135642        STB XW,I
1247*
1248  23015 027500        LDB XMIN
1249  23016 074637        ISZ B,I        SET PAPER RATIO NEGATIVE FOR -(RESULT)
1250  23017 023522        LDA TEMY
1251  23020 160220        JSM .FMPA,I    YW = 9999/6 * H * SIN TH * PH/PW
1252  23021 062660        JSM ROT4
1253  23022 135643        STB YW,I
1254*
1255  23023 021646 DEFLT  LDA T3
1256  23024 070110        SZA ELAB6
1257  23025 067054        JMP WREXC-3
1258*
1259*     PRINT SYNTAX EXECUTION
1260*
1261  23026 025416 ELAB6  LDB MODE
1262  23027 076150        RZB *+3
1263  23030 021622        LDA TERM
1264  23031 072150        RZA *+3
1265  23032 160170        JSM CLERA,I
1266  23033 035622        STB TERM
1267  23034 020146        LDA M15
1268  23035 031453        STA PRFLG
1269  23036 163510        JSM CKEOL,I
1270  23037 067177        JMP ELAB5
1271  23040 067177        JMP ELAB5
1272  23041 163574 ELAB1  JSM PRINN,I    SET THE BUFFER
1273  23042 067177        JMP ELAB5      END OF LINE
1274  23043 021625 ELAB4  LDA FIRST
1275  23044 070076        TCA
1276  23045 001400        ADA BADDR
1277  23046 071550        SZA ELAB1
1278  23047 025625        LDB FIRST
1279  23050 063203        JSM OUTCH      OUTPUT THE BUFFER
1280  23051 021400        LDA BADDR
1281  23052 031625        STA FIRST
1282  23053 067041        JMP ELAB1
1283*
1284*       FORMAT TYPE EXCUTION
1285*
1286  23054 163510        JSM CKEOL,I
1287  23055 000153 B153   OCT 153
1288  23056 035472        STB TEMPS
1289  23057 163652 WREXC  JSM TYPEC,I    GET NEXT FORMAT SPEC
1290  23060 067072        JMP WRITN      P+1, END OF FORMAT
1291  23061 067102        JMP WRITQ      P+2 QUOTE
1292  23062 074742        SBR 16         P+3 FIXED FORMAT
1293  23063 060563        JSM SLWST      P+4 STACK FLAG 0-F; +=E, -=BINARY
```

```
        #01  EXECUTION PHASE OF 9830 PLOTTER BLOCK


1294  23064 067104        JMP WRITF      P+5 (CODE 5 NOT USED)
1295  23065 066672        JMP PRERR      BINARY MODE NOT ALLOWED
1296  23066 067077        JMP WRITX      X!
1297  23067 163540        JSM FSPR,I
1298  23070 063370        JSM PCRLF+1    /! CRLF
1299  23071 067057        JMP WREXC
1300*
1301  23072 075753 WRITN  SBP *-1
1302  23073 074742        SBR 16         CLEAR FORMAT NESTING FLAG
1303  23074 035455        STB FORMT
1304  23075 067026        JMP ELAB6
1305*
1306  23076 000103 BLNKA  DEF .32+.32+1
1307  23077 163540 WRITX  JSM FSPR,I
1308  23100 020012        LDA .1
1309  23101 027076        LDB BLNKA
1310  23102 063203 WRITQ  JSM OUTCH
1311  23103 067057        JMP WREXC
1312*
1313*
1314  23104 025466 WRITF  LDB LOCL1      STACK POINTER
1315  23105 060563        JSM SLWST
1316  23106 163510        JSM CKEOL,I
1317  23107 074742        SBR 16
1318  23110 067173        JMP WREND-2
1319  23111 163564        JSM QTCHK,I    QUOTE NEXT IN WRITE?
1320  23112 067115        JMP WRITS      P+1 NO
1321  23113 063203        JSM OUTCH      P+2, YES (A) = LENTGH, (B) = POINTER
1322  23114 067106        JMP WRITF+2
1323*
1324  23115 163526 WRITS  JSM PFORM,I
1325  23116 061220        JSM FETCH+1
1326  23117 060567        JSM UNSTK
1327  23120 031466        STA LOCL1      RECALL FORMAT STATEMENT POINTER
1328  23121 060567        JSM UNSTK
1329  23122 031705        STA FLOAT      RECALL E, F OR B FLAG
1330  23123 045466        ISZ LOCL1
1331  23124 121466        LDA LOCL1,I
1332  23125 045466        ISZ LOCL1
1333  23126 045466        ISZ LOCL1
1334  23127 025456 WRITB  LDB FORMS
1335  23130 074073        SBP *+1,C
1336  23131 035456        STB FORMS      SET NUMERIC SPEC FOUND FLAG
1337  23132 125466        LDB LOCL1,I    EGT DECIMAL SPEC
1338  23133 163540        JSM FSPR,I     UPDATE FORMAT POINTERS
1339  23134 031704        STA WIDTH
1340*
1341  23135 035703 WRIT7  STB PLACE
1342  23136 021412        LDA LSTPT      START TEMP BUFFER
1343  23137 070070        STA *+1
1344  23140 070744        SAL 1
1345  23141 031400        STA BADDR
1346  23142 031706        STA SBPTR
1347  23143 021413        LDA HSTPT
1348  23144 070744        SAL 1
1349  23145 031623        STA PEND
1350  23146 021416        LDA MODE       SET END OF BUFFER AT HIGH STACK
1351  23147 031737        STA T2
1352  23150 070742        SAR 16
1353  23151 031416        STA MODE       SET FOR PROGRAM FORMAT
1354  23152 024215        LDB TMPOP      *
1355  23153 060406        JSM XFAR1+1    * OUTPUT NUMBER TO BUFFER
1356  23154 160234        JSM NMOTA,I    *
1357  23155 021737        LDA T2
1358  23156 031416        STA MODE       RESTORE THE MODE FLAG
1359  23157 021400        LDA BADDR
1360  23160 011623        CPA PEND       OVERFLOW?
1361  23161 064561        JMP MER8
1362  23162 025706        LDB SBPTR
1363  23163 060374        JSM WBUFF
1364  23164 063203        JSM OUTCH
1365  23165 025625        LDB FIRST
1366  23166 035400        STB BADDR      RESET BADDR
1367*
1368  23167 163510 WRIT6  JSM CKEOL,I    IS THIS THE END OF STATEMENT?
1369  23170 074742        SBR 16         P+1, YES CRLF NOT SURPPRESSED
1370  23171 067175        JMP WREND      #P+2, YES
1371  23172 067057        JMP WREXC      P+3, NO RETURN FOR MORE
1372*
1373  23173 055412        DSZ LSTPT
1374  23174 055412        DSZ LSTPT
1375  23175 070742 WREND  SAR 16
1376  23176 031455        STA FORMT
1377*
1378  23177 074110 ELAB5  SZB *+2
1379  23200 066554        JMP EPEN
1380  23201 063367        JSM PCRLF
1381  23202 164172        JMP XEC4A,I
1382*
```

```
        #01  EXECUTION PHASE OF 9830 PLOTTER BLOCK


1383  23203 #31741 OUTCH STA XC+1
1384  23204 #35740       STB XC
1385  23205 #62711       JSM GETLC     SET-UP T1(POINTER TO CRLF) IT MAY BE KILLED
1386  23206 #61053 .OUTC JSM GTMP
1387  23207 #63213       JSM PLET-1
1388  23210 #55741       DSZ XC+1
1389  23211 #67206       JMP .OUTC
1390  23212 170402       RET
1391*
1392*      CHARACTER ROUTINE
1393*
1394  23213 #61363       JSM GTCON+1   CONVERT FROM LOWER CASE
1395  23214 #03620 PLET  ADA M137B     CHECK IF > 137B
1396  23215 #70153       SAP *+3       SKIP IF THE CODE WAS >172B
1397  23216 #00054       ADA .63       (877) WITHIN RANGE ADJUST SO THAT 40B =0
1398  23217 #70113       SAP *+2
1399  23220 170402       RET           RETURN IF NOT A VALID CHARACTER
1400  23221 155647       DSZ T1,I      BUMP THE CHARACTER COUNT FOR CRLF
1401  23222 #74742       SBR 16
1402  23223 #35736       STB CPX
1403  23224 #35737       STB CPY
1404  23225 #70744       SAL 1         MULTIPY BY 2
1405  23226 #03476       ADA VTABE     ADD IN THE VECTOR TABLE POINTER
1406  23227 #01645       ADA CP4       ADD IN CURRENT PAGE
1407  23230 #24213       LDB AR1A
1408  23231 170004       XFR           TRANSFER THE TWO WORD VECTOR TO AR1+1 AR1+2
1409  23232 #24132       LDB M1
1410  23233 #35747       STB AR1+3     SET THE 9TH BLOCK TO GIVE END OF CHAR
1411  23234 175400 PL1   DLS
1412  23235 #13530       CPA B16       WAS THIS THE CCODE FOR SPEACIAL TABLE
1413  23236 #67250       JMP STABL     YES
1414  23237 #70046       RAR 2
1415  23240 #70137       LDB A
1416  23241 #74642       SBR 14
1417  23242 #50020       AND .3
1418  23243 #10020       CPA .3
1419  23244 #67264       JMP .PL2       END OF CHARACTER DEFINITION
1420  23245 #63302       JSM MPEN
1421  23246 #62021       JSM PENDN
1422  23247 #67234       JMP PL1
1423*
1424*      SPECIAL TABLE ROUTINE
1425*
1426  23250 #21745 STABL LDA AR1+1
1427  23251 #70142       SAR 4
1428  23252 #50115       AND B1777
1429  23253 #41645       IOR CP4
1430  23254 #70744       SAL 1
1431  23255 #31750       STA YC
1432  23256 #24057 STAB2 LDB .1000
1433  23257 #61041       JSM GETCH
1434  23260 #31466       STA LOCL1
1435  23261 #70306       RAR 7
1436  23262 #70353       SAP *+7
1437  23263 #70251       SLA *+5
1438*
1439  23264 #62015 .PL2  JSM PENUP
1440  23265 #70004       SAL 16
1441  23266 #24061       LDB .10
1442  23267 #67310       JMP MPEN2
1443*
1444  23270 #62021       JSM PENDN
1445  23271 #70111       SLA *+2
1446  23272 #62015       JSM PENUP
1447  23273 #21466       LDA LOCL1
1448  23274 #70106       RAR 3
1449  23275 #70137       LDB A
1450  23276 #74602       SBR 13
1451  23277 #50024       AND .7
1452  23300 #63310       JSM MPEN2
1453  23301 #67256       JMP STAB2
1454*
1455  23302 #31466 MPEN  STA LOCL1
1456  23303 #70744       SAL 1
1457  23304 #01466       ADA LOCL1     DELTA Y * 3 IN STANDARD MODE
1458  23305 #35466       STB LOCL1
1459  23306 #74744       SBL 1
1460  23307 #05466       ADB LOCL1      DELTA X * 3 IN STANDARD MODE
1461*
1462  23310 #31466 MPEN2 STA LOCL1
1463  23311 #21736       LDA CPX
1464  23312 #35736       STB CPX
1465  23313 #70076       TCA
1466  23314 #74017       ADA 8
1467  23315 #31734       STA DX        DELTA X
1468  23316 #21466       LDA LOCL1
1469  23317 #25737       LDB CPY
1470  23320 #31737       STA CPY
1471  23321 #74076       TCB
1472  23322 #70037       ADB A
```

#01 EXECUTION PHASE OF 9830 PLOTTER BLOCK

```
1473  23323 035735 MPEN3 STB DY        DELTA Y
1474  23324 121640       LDA YH,I
1475  23325 125641       LDB XH,I
1476  23326 063336       JSM MPY1      D.Y. = YH * D.Y. + XH * D.X.
1477  23327 062117       JSM TYM
1478  23330 121643       LDA YW,I
1479  23331 125642       LDB XW,I
1480  23332 063336       JSM MPY1      D.X. = YW * D.Y. + XW * D.X.
1481  23333 062125       JSM TXM
1482  23334 020107       LDA B1000
1483  23335 066042       JMP MOVE-1
1484*
1485  23336 031470 MPY1  STA L2T1
1486  23337 021734       LDA DX
1487  23340 063355       JSM MPY2
1488  23341 035471       STB L2T2
1489  23342 025470       LDB L2T1
1490  23343 021735       LDA DY
1491  23344 063355       JSM MPY2
1492  23345 005471       ADB L2T2
1493  23346 160227       JSM FXFLA,I
1494  23347 125647       LDB T1,I
1495  23350 014017       CPB .2        WAS THIS VECTOR FLAGED AS A CRLF
1496  23351 170402       RET           YES!
1497  23352 020157 .MPY3 LDA M256
1498  23353 001754       ADA AR2
1499  23354 066100       JMP STOR2
1500*
1501  23355 070153 MPY2  SAP *+3
1502  23356 070076       TCA
1503  23357 074076       TCB
1504  23360 035466       STB LOCL1
1505  23361 074742       SBR 16
1506  23362 070076       TCA
1507  23363 070170       SIA *+3
1508  23364 005466       ADB LOCL1
1509  23365 073770       RIA *-1
1510  23366 170402       RET
1512*    LETTER AND LABEL ROUTINES
1513*
1514  23367 160170 PCRLF JSM CLERA,I
1515  23370 121647       LDA T1,I
1516  23371 031734       STA DX
1517  23372 020017       LDA .2        SET THE FLAG SO MPEN3 DOESN'T REDUCE BY 1/10TH
1518  23373 131647       STA T1,I
1519  23374 024132       LDB M1
1520  23375 063323       JSM MPEN3
1521  23376 074742 .PC   SBR 16
1522  23377 135647       STB T1,I
1523  23400 170402       RET
1524*
1525*    LETTER EXECUTION
1526*
1527  23401 062674 EKEY  JSM CDEF      GET CURRENT PAGE, CHARACTER LINKS AND CRLF
1528  23402 074742       SBR 16
1529  23403 035473       STB CCNT
1530  23404 020054       LDA .63
1531  23405 061016       JSM ONEXT
1532  23406 163673 .EKEY JSM KEYIN,I
1533  23407 024141       LDB M10
1534  23410 070346       HAR 8
1535  23411 070133       SAP *+2,C
1536  23412 024132       LDB M1
1537  23413 070346       HAR 8
1538  23414 010034       CPA .21       UP ARROW (25B)
1539  23415 067446       JMP .EK2
1540  23416 013532       CPA B24       DOWN ARROW (24B)
1541  23417 067447       JMP .EK1
1542  23420 010036       CPA B27       RIGHT ARROW ( 27B)
1543  23421 067453       JMP .EK4
1544  23422 010035       CPA B26       LEFT ARROW (26B)
1545  23423 067454       JMP .EK3
1546  23424 010104       CPA B177      STOP KEY?
1547  23425 067442       JMP .STOP     YES
1548  23426 010027       CPA .11
1549  23427 067457       JMP .RCRL
1550  23430 013534       CPA B157
1551  23431 020071       LDA E
1552  23432 074311       SLB *+6
1553  23433 013055       CPA B153
1554  23434 020101       LDA B100
1555  23435 127506       LDB VALUP,I
1556  23436 163524       JSM TBSRH,I
1557  23437 067440       JMP *+1
1558  23440 063214       JSM PLET
1559  23441 067406       JMP .EKEY
1560*
1561  23442 074742 .STOP SBR 16
1562  23443 035442       STB STPFL
```

#01 EXECUTION PHASE OF 9830 PLOTTER BLOCK

```
1563  23444 160170        JSM CLERA,I
1564  23445 164172        JMP XEC4A,I
1565*
1566  23446 074076 .EK2   TCB             UP ARROW
1567  23447 070742 .EK1   SAR 16          DN ARROW
1568  23450 031734        STA DX
1569  23451 063323        JSM MPEN3
1570  23452 067406        JMP .EKEY
1571  23453 074076 .EK4   TCB             RIGHT ARROW
1572  23454 035734 .EK3   STB DX          LEFT ARROW
1573  23455 074742        SBR 16
1574  23456 067451        JMP .EK1+2
1575  23457 063367 .RCRL  JSM PCRLF
1576  23460 067402        JMP EKEY+1
1577*
1578*       CHARACTER PLOT
1579*
1580  23461 062674 CPLOT  JSM COEF
1581  23462 062263        JSM FORM+2
1582  23463 063471        JSM .CPLT
1583  23464 035735        STB DY
1584  23465 063471        JSM .CPLT
1585  23466 035734        STB DX
1586  23467 063324        JSM MPEN3+1
1587  23470 164172        JMP XEC4A,I
1588*
1589  23471 061220 .CPLT  JSM FETCH+1
1590  23472 020013        LDA C7            MULT RESULT BY 10
1591  23473 063353        JSM .MPY3+1
1592  23474 066663        JMP ROT1+2
```

#01 LETTER AND NUMBER VECTOR TABLE

```
1594*
1595* LETTER VECTOR TABLE
1596*
1597*
1598  23475 167715        JMP BLKN,I     SPACE 40B
1599  23476 001474 VTABE  DEF *-2-N9/6
1600  23477 167713        JMP EXCM,I     !
1601  23500 001650 XMIN   DEF XN
1602  23501 167746        JMP QUOTE,I     " 42B
1603  23502 001660 YMIN   DEF YN
1604  23503 167716        JMP NUMBR,I    #
1605  23504 001664 YMAX   DEF YX
1606  23505 167734        JMP DOLLR,I    $
1607  23506 002412 VALUP  DEF 2412B      ABS ADDRESS
1608  23507 167701        JMP PERCT,I
1609  23510 007227 CKEOL  DEF 7227B      ABS ADDRESS
1610  23511 167723        JMP AMP,I      &
1611  23512 176000 EXPM4  OCT 176000
1612  23513 112700        OCT 112700     '
1613  23514 023417 .9999  DEC 9999
1614  23515 167740        JMP LPARN,I    S.T. POINTER FOR (
1615  23516 012025 FMTOP  OCT 12025
1616  23517 167743        JMP RPARN,I    S.T. POINTER FOR )
1617  23520 177736 M34    DEC -34
1618  23521 167674        JMP MULT,I     *
1619  23522 001674 TEMY   DEF Z2
1620  23523 167676        JMP PLUS,I
1621  23524 001072 TBSRH  DEF J0728
1622  23525 002700        DEC 1472        ,
1623  23526 006773 PFORM  DEF 6773B      ABS ADDRESS
1624  23527 043300        DEC 18112       -
1625  23530 000016 B16    OCT 16
1626  23531 016000        DEC 7168        .
1627  23532 000024 B24    OCT 24
1628  23533 005300        DEC 2752        /
1629  23534 000157 B157   OCT 157
1630  23535 004242        OCT 4242,5300   0
      23536 005300
1631  23537 014700        DEC 6592        1
1632  23540 007356 FSPR   DEF 7356B      ABS ADDRESS
1633  23541 105145        OCT 105145,1300 2
      23542 001300
1634  23543 105145        DEC -30107,25100 3
      23544 061014
1635  23545 102152        DEC -31638,11264 4
      23546 026000
1636  23547 124105        OCT 124105,20300 5
      23550 020300
1637  23551 124106        DEC -22458,8268  6
      23552 020114
1638  23553 105034        OCT 105034      7
1639  23554 177741 M31    DEC -31
1640  23555 044246        DEC 18598,16422  8
      23556 040046
1641  23557 001250        DEC 680,18112    9
      23560 043300
```

```
PAGE 0037 #01  LETTER AND NUMBER VECTOR TABLE


1642  23561 167710        JMP COLON,I    :
1643  23562 004303 RITE   DEF 4303B      RITE+1 ABS ADDRESS
1644  23563 167707        JMP SCOLN,I    ;
1645  23564 007240 QTCHK  DEF 7240B      ABS ADDRESS
1646  23565 112034        DEC -27620     <
1647  23566 012761 SIN    DEF 12761B     ABS ADDRESS
1648  23567 167720        JMP EQUAL,I    =
1649  23570 013007 COS    DEF 13007B     ABS ADDRESS
1650  23571 113034        DEC -27108     >
1651  23572 007402 PLOTE  DEF 7402B     ABS ADDRESS
1652  23573 167727        JMP QUES,I     ?
1653  23574 006656 PRINN  DEF 6656B
1654  23575 004242        OCT 4242,12554      @ 100B
      23576 012554
1655  23577 004242        DEC 2210,25792      A
      23600 062300
1656  23601 004245        DEC 2213,17952      B
      23602 043040
1657  23603 124501        DEC -22207,11264    C
      23604 026000
1658  23605 004226        DEC 2198,4288       D
      23606 010300
1659  23607 124105        DEC -22459,16428    E
      23610 040054
1660  23611 124105        DEC -22459,16576    F
      23612 040300
1661  23613 124002        DEC -22526,26048    G
      23614 062700
1662  23615 004106        DEC 2118,-23872     H
      23616 121300
1663  23617 014700        DEC 6592            I
1664  23620 177641 M137B  OCT -137
1665  23621 121004        DEC -24060,-16384   J
      23622 140000
1666  23623 004112        DEC 2122,17088      K
      23624 041300
1667  23625 100054        DEC -32724          L
1668  23626 000000        BSS 1          CHECK SUM WORD
1669  23627 004132        DEC 2138,11264      M
      23630 026000
1670  23631 004052        DEC 2090,-16384     N
      23632 140000
1671  23633 004242        DEC 2210,3072       O
      23634 006000
1672  23635 004246        DEC 2214,19456      P
      23636 046000
1673  23637 020212        DEC 8330,9664       Q
      23640 022700
1674  23641 004246        DEC 2214,17708      R
      23642 042454
1675  23643 001144        DEC 612,-30016      S
      23644 105300
1676  23645 105221        DEC -30063,-16384   T
      23646 140000
1677  23647 100052        DEC -32726,-16384   U
      23650 140000
1678  23651 100654        DEC -32340          V
1679  23652 007303 TYPEC  DEF 7303B      ABS ADDRESS
1680  23653 100122        DEC -32686,-21504   W
      23654 126000
1681  23655 005130        DEC 2648,11264      X
      23656 026000
1682  23657 102645        DEC -31323,7168     Y
      23660 016000
1683  23661 105002        DEC -30206,-16384   Z
      23662 140000
1684  23663 010211        OCT 10211,140000    133B
      23664 140000
1685  23665 101300        OCT 101300
1686  23666 005555 MOVCK  DEF 5555B
1687  23667 011251        OCT 11251,140000    135 B
      23670 140000
1688  23671 014511        OCT 14511,66000     136B
      23672 066000
1689*
1690  23502         PPPOS EQU YMIN
1691*
1692*  MORE ABS ADRESSESS
1693*
1694  23673 002753 KEYIN DEF 2753B
1695*
1696*       SPECIAL SYMBOL VECTOR TABLE
1697*
1698*       BIT 15 =1 PENUP BEFORE MOVEMENT
1699*       BIT 14 = 1 PEN DOWN BEFORE MOVEMENT
1700*       BITS 13,12,11 = Y COOR. OF 1ST LOCATION
1701*       BIT 10,9,8    = X COOR. OF 1ST LOCATION
1702*       BIT 7 = 1 PENUP
1703*       BIT 6 = 1 PEN DOWN
```

```
        #01  LETTER AND NUMBER VECTOR TABLE


1704*       BIT 5,4,0 = Y OF 2ND
1705*       BIT 2,1,0 = X OF 2ND
1706*
1707*     PEN CONDITIONS
1708*
1709*   PEN UP ON ENTRY
1710*   PEN UP ON EXIT
1711*   A 7 IN Y INDICATES END OF CHAR.
1712*
1713*   THIS TABLE WORKS ON A 7X7 MATRIX (6-0)
1714*
1715  23674 104555 MULT  OCT 104555,124515
      23675 124515
1716  23676 101563 PLUS  OCT 101563,114136,140000  +
      23677 114136
      23700 140000
1717  23701 073161 PERCT OCT 73161,24041,25061  %
      23702 024041
      23703 025061
1718  23704 112514       OCT 112514,2416,12700
      23705 002416
      23706 012700
1719  23707 000523 SCOLN OCT 523            ; 738
1720  23710 011523 COLON OCT 11523,125553,140000  : 728
      23711 125553
      23712 140000
1721  23713 031523 EXCM  OCT 31523                 !  418
1722  23714 101503       OCT 101503
1723  23715 140000 BLKN  OCT 140000
1724  23716 131102 NUMBR OCT 131102,102164     #
      23717 102164
1725  23720 110126 EQUAL OCT 110126                =
1726  23721 123140       OCT 123140
1727  23722 140000       OCT 140000
1728  23723 003152 AMP   OCT 3152        &
1729  23724 031454       OCT 31454
1730  23725 010403       OCT 10403
1731  23726 012700       OCT 12700
1732  23727 024562 QUES  OCT 24562       ?
1733  23730 031454       OCT 31454
1734  23731 022033       OCT 22033
1735  23732 011603       OCT 11603
1736  23733 041700       OCT 41700
1737  23734 004515 DOLLR OCT 4515              $ 448
1738  23735 024455       OCT 24455
1739  23736 131503       OCT 131503
1740  23737 160000 SC16  OCT 160000
1741  23740 001511 LPARN OCT 1511
1742  23741 014051       OCT 14051
1743  23742 031700       OCT 31700
1744  23743 001515 RPARN OCT 1515
1745  23744 017055       OCT 17055
1746  23745 031700       OCT 31700
1747  23746 021162 QUOTE OCT 21162       "
1748  23747 132144       OCT 132144
1749  23750 140000       OCT 140000
1750*
1751*
1752  23760              ORG 23760B
1753*
1754*       STATEMENT EXECUTION JUMP TABLE
1755*
1756  23760 177421       ABS EKEY-*
1757  23761 177500       ABS CPLOT-*
1758  23762 176303       ABS ESCAL-*
1759  23763 176527       ABS EYAXE-*
1760  23764 176525       ABS EXAXE-*
1761  23765 176476       ABS EPLOT-*
1762  23766 176361       ABS EOFF-*
1763  23767 176565       ABS EPEN-*
1764  23770 176744       ABS ELAB-*
1765  23771 176471       ABS EIPLT-*
1766  23772 001670 TEMX  DEF Z1          OP CODE 1 IS NOT USED, SO USE IT FOR A CONSTANT
1767*
1768  23773 176636       ABS TBL-*
1769  23774 176635       ABS TBL-*
1770  23775 176634       ABS TBL-*
1771  23776 176154       ABS SNAME-*
1772  23777 010000       OCT 10000       CODE TO IDENTIFY THE OPTION BLOCK

        CROSS-REFERENCE TABLE


.1      0020      1308
.10     0071      0072      1441
.100    0070
.1000   0069      1432
.11     0035      0043      1548
.12     0036
```

CROSS-REFERENCE TABLE

.13	0039										
.15	0048	0083									
.16	0049	0084	0871	0895	0928	0961					
.1E4	0068										
.2	0027	0610	0838	0964	1495	1517					
.21	0050	1538									
.22	0051	0087									
.23	0052	0088									
.28	0053										
.3	0028	0045	0614	1417	1418						
.31	0054	0090	0652								
.32	0055	1306	1306								
.33	0056										
.34	0057	0091									
.37	0058										
.4	0029	0857	0876	0917	0970	1070					
.40	0059										
.41	0060										
.43	0061										
.45	0062										
.46	0063										
.48	0064	1116									
.5	0030										
.58	0065	0094									
.6	0031	0044									
.63	0066	1397	1530								
.7	0032	1451									
.8	0033	0883	0945	0952	1072						
.9	0034										
.99	0074										
.9999	1613	0663	1066								
.AX4	1084	1055									
.BUFA	0180										
.CDEF	1175	1158									
.CL1	1090	1101									
.CL2	1093	1089	1103								
.CPLT	1589	1582	1584								
.E1	1032	1063									
.EK1	1567	1541	1574								
.EK2	1566	1539									
.EK3	1572	1545									
.EK4	1571	1543									
.EKEY	1532	1559	1570								
.ELAB	1211	1187									
.FADA	0185	0706	1045								
.FDVA	0188	0878	1073	1138							
.FMPA	0187	0882	0948	1141	1223	1242	1251				
.FSBA	0186	0874	0904	0953							
.FSR	0190										
.LNUM	0314										
.M	0656	0634	0642								
.MPY3	1497	1217	1591								
.NPWR	0498										
.OUTC	1386	1389									
.P1	0685	0609	0613	1006							
.PC	1521	1173									
.PL2	1439	1419									
.PUN	0938	0934									
.RCRL	1575	1549									
.SLBF	0181										
.STOP	1561	1547									
.TIC	1126	1120	1124								
.TICK	1114	1042	1110	1111							
.XX	0977	0999	1001								
A	0000	0837	1415	1449	1472						
ACTSA	0174										
ADRSS	0327										
AERRA	0176	0680	1093	1149							
AMOVE	0630	1003	1084	1114	1128						
AMP	1728	1610									
AR1	0413	0182	1410	1426							
AR1A	0182	1407									
AR2	0415	0183	0637	0645	0664	0957	1498				
AR2A	0183										
ARYAD	0403										
AX2	1065	1049	1062								
AX3	1062	1024									
AX4	1077	1041	1050								
AX5	1043	1047									
AX6	1052	1113									
AX7	1049	1020									
B	0000	0659	0676	0839	0842	0982	0950	0963	0965	1071	1117
	1144	1249	1466								
B100	0095	1554									
B1000	0102	0635	1482								
B10K	0114	0115									
B12K	0116	0117	0206								
B133	0096										
B140	0097										

CROSS-REFERENCE TABLE

```
B1400   0103   0204
B153    1287   1553
B157    1629   1550
B16     1625   1412
B17     0083
B170K   0121
B1740   0104   0203   0817
B1760   0105   0158   1156
B177    0098   1181   1546
B1774   0106
B1775   0107
B1777   0108   1428
B20     0084
B22     0085
B23     0086
B24     1627   0893   1540
B2400   0110
B26     0087   1544
B27     0088   1542
B3000   0111
B36     0089
B37     0090
B377    0099   0638   0646   1179
B400    0100   0639   0643   0647
B4000   0112   0162
B42     0091
B44     0092
B52     0093
B6K     0113   0167   1147
B72     0094
B777    0101
BADDR   0291   1276   1280   1345   1359   1366
BADRP   0284
BEND    0198   0199
BHSTP   0444
BLANK   0388
BLKN    1723   1598
BLNKA   1306   1309
BUSY    0617   0619   0636   0640   0644   0649
C5      0015
C6      0011   0114
C7      0022   0100   1590
CCNT    0538   1529
CDEF    1154   1176   1199   1527   1580
CHRSA   0219
CKEOL   1609   1269   1286   1316   1368
CLAR2   0492
CLERA   0160   1265   1514   1563
CLIMT   1089   1027   1031   1082
CLINE   0316
CLP1A   0217
COLON   1720   1642
COMCE   0436   0791   0804   0806
COMCK   0435   0759   0762   0777   0780   0798
CONST   0484
COS     1649   1230
CP2     0540   0927   0947   0951   0962
CP3     0517   0938   0984   1016
CP4     0518   0877   0971   1134   1222   1406   1429
CPLAC   0331
CPLOT   1588   1757
CPT     0541   0866   0879   0884   0899   0905
CPX     0535   0542   1402   1463   1464
CPY     0536   0543   1403   1469   1470
CRCKA   0228
CRLF    0501
CRLF1   0502
CRTN    0037   0038   0039
CTYPE   0330
CURPG   1132   0819   0986   1015   1148   1154   1191
DCOMA   0210
DEFLT   1255   1203
DELSA   0175
DFLAG   0390
DIGCK   0463
DOLLR   1737   1606
DPERR   0043
DRQST   0317
DSTRT   0303
DX      0533   0540   1467   1486   1516   1568   1572   1585
DY      0534   0541   1473   1490   1583
E       0080   1551
E1      0235
E10     0244
E15     0249
E20     0254
E25     0259
E30     0264
E35     0269
```

CROSS-REFERENCE TABLE

E40 0274
E45 0279
E5 0239
E50 0284
ECALA 0215
EIPLT 0983 1765
EKEY 1527 1576 1756
ELAB 1191 1764
ELAB1 1272 1277 1282
ELAB3 1185 1209
ELAB4 1274
ELAB5 1378 1270 1271 1273
ELAB6 1261 1256 1304
ELINA 0169
ENDS 0480 0754
ENTEA 0218 1171 1175
ENTFA 0189
EOFF 0891 1762
EOFF1 0899 0894 0896
EOL 0038
EOLCK 0483
EOST 0481 0760 0763 0793 0799 0801
EPEN 1056 0653 1379 1763
EPLOT 0984 1761
EQUAL 1725 1648
ERRA 0177
ES1 0845 0825
ESCAL 0819 1758
EXAXE 1012 1760
EXCM 1721 1600
EXP 0401 0402
EXPM4 1611 0968
EXPOP 0118
EYAXE 1013 1759
F 0081
FDAAA 0222
FETCH 0488 0867 0869 0900 0916 0923 0991 1029 1032 1077 1087
1210 1212 1214 1216 1325 1589
FILLA 0201
FIRST 0367 1274 1278 1281 1365
FIX 0505
FLGBT 0073
FLOAT 0386 1329
FLTFA 0193
FLTRA 0192 0657 0992 1090 1142
FLTSA 0195
FMTOP 1615 1193
FN 0148
FND 0462
FND1A 0224
FNDAA 0223
FNDPS 0476
FOPBS 0206
FORM 0810 0820 0891 0987 1025 1205 1501
FORM2 0805 0790 0792 0797
FORM3 0803 0779
FORMA 0172 0810 0811 0812 0813 1017 1021
FORME 0336
FORMS 0335 1334 1336
FORMT 0334 1303 1376
FPAGE 0451
FROMA 0122
FSCA 0173 0758 0761 0782 0800 0803 0805 0807
FSCE2 0441
FSCE4 0446
FSPR 1632 1297 1307 1338
FTMP1 0337
FTMP2 0338
FWAM 0157
FWUP 0296
FXFLA 0194 1053 1126 1493
GE1 0675 0845 1159 1170 1172
GETAD 0461
GETCH 0471 1433
GETLC 1167 1385
GETRW 0672 0892 0996 1026 1065
GETSK 0475
GFLAG 0404
GFRST 0474
GNEXT 0473 0770 0776
GPARM 0424
GSIGN 0486
GTCON 0504 1394
GTMP 0472 1386
GTOPP 0466
GTSTA 0213
HFLG1 0321
HSTPT 0302 1347
I 0067

CROSS-REFERENCE TABLE

ICNT	0422									
IMPY	0487									
INF	0077									
INITS	0456									
INREA	0226									
INTCK	0459									
INTEA	0178									
INTEL	0340									
INTFL	0075									
INTGR	0402									
IOBUF	0359									
IOHEA	0229									
IOSTA	0225									
KEYFG	0324									
KEYIN	1694	1532								
L2T1	0347	1485	1489							
L2T2	0348	1488	1492							
LAB1	0783	0778	0781							
LAB2	0784	0774								
LAB3	0777	0786								
LABEL	0768	0720								
LASTN	039A									
LBOP	0119									
LDEF	0489									
LETCK	0464									
LFLAG	0961	0974								
LIMIT	0465									
LIMT	0957									
LISTA	0167									
LNFD	0072									
LNINC	0315									
LNUM	0457	0785								
LOCL1	0345	1314	1327	1330	1331	1332	1333	1337	1434	1447	1455
	1457	1458	1460	1462	1468	1504	1508			
LOCL2	0346									
LPARN	1741	1614								
LPCK	0437									
LPCKE	0438	0769								
LPOP	0120									
LROMA	0123									
LSTAK	0299									
LSTPT	0301	1342	1373	1374						
LTR	0479									
LWAM	0295									
M1	0124	0675	1409	1519	1536					
M10	0131	1533								
M100	0144									
M1024	0146									
M11	0132									
M12	0133									
M13	0134									
M137B	1664	1395								
M14	0135									
M15	0136	1267								
M2	0125	0978	0993							
M23	0137									
M25	0138									
M256	0145	1497								
M3	0126									
M31	1639	0848								
M32	0139									
M34	1617	0827	0836							
M36	0140									
M4	0127									
M48	0141	1121								
M5	0128									
M6	0129									
M64	0142									
M8	1091	0903								
M80	0143									
M9	0130									
MASK1	0153									
MASK5	0154									
MAW	0294									
MAXIN	0400									
MAXSN	0147	0661	1118							
MBIN1	0406									
MBIN2	0407									
MBOX1	0351									
MDIMA	0208									
MER8	0445	1361								
MINIT	0455									
MODE	0305	1261	1350	1353	1358					
MOVCK	1686	0834								
MOVE	0633	1483								
MPEN	1455	1420								
MPEN2	1462	1442	1452							
MPEN3	1473	1520	1569	1586						
MPY1	1485	1476	1480							
MPY2	1501	1487	1491							

CROSS-REFERENCE TABLE

MSFLG	0389							
MULT	1715	1618						
MVTOH	0477							
N9/6	0588	1599						
NDISP	0382							
NOWRT	0500							
NEXT1	0450							
NEXTA	0467							
NEXTL	0449							
NMOTA	0200	1356						
NOEOF	0482							
NPRIN	0383							
NPWRT	0499							
NUM	0396	0397						
NUMBR	1724	1604						
NUMCA	0211							
NUMOA	0212							
NXTDT	0304							
NXTST	0405							
OBLNK	0468							
ODGIT	0496							
OF	0493							
OFFS	0792	0713	0718	0764				
ONE	0045	1185						
ONEXT	0469	1531						
OPCHA	0171							
OPDMK	0079							
OPMSK	0076	1201	1207					
OPTYP	0397							
OTPTR	0293							
OUTCH	1383	1279	1310	1321	1364			
OUTCR	0470							
OUTIA	0179							
OVCHA	0216							
OWHER	0694	1127						
PARMA	0306							
PARMN	0307							
PBPTO	0313							
PBPTR	0310							
PBUFF	0311							
PBUFO	0312							
PCRLF	1514	1298	1380	1575				
PEND	0365	1349	1360					
PENDN	0612	0606	0611	1085	1421	1444		
PENUP	0608	0671	0937	0980	1056	1155	1439	1446
PERCT	1717	1608						
PEXMA	0165							
PFLAG	0391							
PFORM	1623	1324						
PGIN0	0478							
PGINT	0458							
PI/2	0044							
PL1	1411	1422						
PLACE	0384	1341						
PLET	1395	1387	1558					
PLOT	0797	0717	0721					
PLOTE	1651	1196						
PLUS	1716	1620						
PMODE	0329							
PNPAR	0602	1002						
PPFF	0366							
PPPOS	1690	1167						
PRADD	0308	0989	1019	1023				
PRERR	1149	1143	1295					
PRFLG	0332	1268						
PRINN	1653	1272						
PUN1	0943	0939						
PUNIT	0926	0919						
PUNX	0923	0911	1000					
PUNY	0916	0912	0998					
PWHER	0910	1075	1081	1088	1100			
QTCHA	0230							
QTCMK	1645	1319						
QTOP	0109	0161						
QUES	1732	1652						
QUOTE	1747	1602						
RBOP	0117							
RBUFF	0360							
RCOUT	0333							
RDYDA	0166							
REED	0161							
REEDL	0339							
RES	0417							
RESB1	0453							
RESBP	0454							
RET2	0421							
RETN2	0420							
RETN3	0419							
RITE	1643	0783						
ROT1	1140	1233	1592					

CROSS-REFERENCE TABLE

ROT2 1137 1237 1245
ROT4 1139 1252
ROUND 0495
RPARN 1744 1616
RPCK 0439
RPCKE 0440
RPOP 0115 1202 1208
RSAVE 0292
RSLTE 0373
RSPTR 0418 1112
RSTAK 0416
RSTKA 0158
RSULT 0426
RUNA 0163
S 0082
SALTA 0207
SAVBP 0452
SAVEA 0325
SAVEB 0326
SAXIS 0758 0715 0716
SBADR 0394
SBPTR 0387 0784 1346 1362
SBPUD 0431 0768 0771 0775
SBUFA 0199
SC16 1740 0620 0622
SCAL 0866 0858 0860
SCALE 0790 0714
SCOLN 1719 1644
SEC 0320
SFLAG 0393
SGNS 0497
SIGN 0399 0400
SIN 1647 1226
SLC0D 0328
SLEND 0202
SLOOP 1041
SLWST 0447 1293 1315
SNAME 0724 1771
SNFLA 0197
SPEN 0754 0712 0719
SPLFA 0196
SSBPT 0395
SSYMA 0170 0673 0823 1157 1168
ST 0377
STAB2 1432 1453
STABL 1426 1413
STAR2 0490
START 0162 0007
STATU 0622 0608 0612 0617
STEAL 0817 0672 0822 0841
STEXP 0443
STMAX 0494
STOR1 042A
STOR2 0664 1499
STORE 0427
STPFL 0323 0651 1562
STROP 0430
STSRH 0442
SUMPA 0227
SYE12 0485
SYE25 0460
SYMC1 0433 0772
SYMC2 0432
SYMCK 0434
SYMTP 0298
SYTSA 0209
T1 0520 0677 0885 0926 1400 1494 1515 1518 1522
T2 0543 0705 0707 0846 0849 0850 0930 0959 0972 1351 1357
T3 0519 0685 0694 0910 1013 1068 1078 1198 1255
T4 0542 0929 0932 0936 0940 0943
TBL 1112 1768 1769 1770
TBSRH 1621 1556
TEMP 0350 0351
TEMP1 0352
TEMP2 0353
TEMP3 0354
TEMP4 0355
TEMP5 0539 0631 0648
TEMP6 0349 0538 0539 0988 1018 1022 1192 1200 1206 1288
TEMX 1766 0702 1044 1244
TEMXX 0702 1033 1139 1243
TEMY 1619 1227 1236 1250
TERM 0364 1263 1266
TFLAG 0318
TFLG1 0322
TICMK 1099 1046
TMPA 0203
TMPOP 0184 1354
TR/W 0885 0872 0880 0901 0906
TRANS 0707 0941 0946 0949 0954

CROSS-REFERENCE TABLE

```
TREGE   0372
TSRHA   0168
TSTPT   0300
TTY     0503    0621
TXM     0696    0686    0875    1211    1481
TXX     0699    0868    1030    1074    1099    1215
TYM     0688    0695    0870    1213    1231    1477
TYPEC   1679    1289
TYPFL   0078
TYX     0691    1218    1224
UFLAG   0392
UKPTR   0309
UNITS   0319
UNSTK   0448    1326    1328
UPARA   0191
VALUE   0297    0826    0843
VALUP   1607    1555
VARDA   0214
VTABE   1599    1405
WBUFF   0423    1363
WHERE   0685    1054
WIDTH   0385    1339
WREND   1375    1318    1370
WREXC   1289    1257    1299    1311    1371
WRITS   1324    1320
WRIT6   1368
WRIT7   1341
WRITB   1334
WRITE   0159
WRITF   1314    1294    1322
WRITN   1301    1290
WRITQ   1310    1291
WRITX   1307    1296
XC      0412    1383    1384    1388
XEC4A   0164    0861    0897    1007    1057    1381    1564    1587
XEC6A   0221
XFAR1   0425    0689    0692    0697    0700    0703    0708    0886    0931    0944    1228
        1355
XFAR2   0429    0656    1186
XH      0514    1162    1238    1475
XLT2    0968    0958    0977
XMAX    0658    0699    0873    1043    1137
XMIN    1601    0633    0696    0925    1241    1248
XN      0522    1601
XSCRA   0220
XW      0515    1164    1182    1246    1479
XX      0523    0602    0658    0979    0994    1004
YC      0414    1431
YH      0513    1160    1177    1180    1234    1474
YMAX    1605    0691    1140
YMIN    1603    0641    0688    0881    0918    1225    1229    1240    1690
YN      0524    1603
YW      0516    1166    1253    1478
YX      0525    1028    1052    1067    1079    1083    1102    1104    1107    1115    1122
        1125    1605
Z1      0527    1105    1766
Z2      0528    1619
ZAP     0152
ZONK    0491
```

#01 TERMINAL OPTION BLOCK

```
0503                    LST
0504  22000             ORG 22000B
0505*
0506* TERMINAL OPTION BLOCK
0507*
0508  22000 000052 TO    ABS XTRM-*      COMMAND JUMP TABLE
0509*
0510  22001 052105 COMTB  OCT 052105      TE
0511  22002 051115        OCT 051115      RM
0512  22003 170400        OCT 170400        (1)   END   ALLOWS TWO INTEGERS
0513*
0514*
0515  22004 001757        ABS ERRX-*      SYNTAX JUMP TABLE
0516  22005 000000        NOP             ** CHECKSUM WORD **
0517*
0518  22006 020302 SYSNT  OCT 020302        (2)   END
0519*
0520*
0521  22007 000777 AN777  OCT 777
0522*
0523*
0524  22010 005032 VALUP  DEF 2415B+2415B
0525  22011 002617 PREED  DEF 2617B
0526  22012 002753 KEYIN  DEF 2753B
0527  22013 003647 GETLN  DEF 3647B
0528  22014 004036 LISTS  DEF 4036B
0529  22015 006306 MCOUT  DEF 6306B
```

#01 TERMINAL OPTION BLOCK

```
#530*
#531*
#532* SECONDARY ENTRY FROM MAIN SYSTEM THRU REED LINK
#533*
#534  22016 300017 RE1A  DEF *+1-TO
#535*
#536  22017 023647        LDA AT1S
#537  22020 027646        LDB AT1
#538  22021 170004        XFR                 RESTORE CRITICAL PARAMETERS
#539  22022 170004        XFR
#540  22023 020662        LDA FLGRT
#541  22024 031416        STA MODE            RESTORE MODE FLAG
#542*
#543  22025 021433        LDA TFLAG
#544  22026 010172        CPA M1              CHECK FOR COMPILE
#545  22027 067104        JMP XCOM1
#546  22030 070210        SZA *+4             CHECK FOR PTAPE
#547  22031 162011        JSM PREED,I
#548  22032 066031        JMP *-1             (NULL RECORD)
#549  22033 066517        JMP EOL1
#550*
#551  22034 024212        LDB .SLBF
#552  22035 074056        CMB
#553  22036 005400        ADB HADDR
#554  22037 004132        ADB M1              HADDR-.SLBF-2
#555  22040 074113        SBP *+2             IS DISPLAY EMPTY?
#556  22041 066134        JMP RE1             YES
#557  22042 066147        JMP R17             NO, PRESERVE SYSTEM MESSAGE
#559*
#560* "WHAT?" MESSAGE
#561*
#562  22043 062542 WHAT  JSM MTIR            SAVE MYSTERIOUS LINE
#563  22044 160170        JSM CLERA,I         CLEAR THE DISPLAY
#564  22045 020022        LDA .5
#565  22046 027765        LDB WHAM
#566  22047 066145        JMP R10
#567*
#568*
#569  22050 000156 .110   DEC 110             CONSTANTS USED TO CHECK BAUD RATE
#570  22051 174702 M159R  DEC -159R
#571*
#572  22052 072110 XTRM  SZA *+2             TEST FOR DEFAULT SELECT CODE (4)
#573  22053 020021        LDA .4
#574  22054 070204        SAL 12              SAVE SELECT CODE
#575  22055 031651        STA T1
#576  22056 014056        CPB .1F4            TEST FOR DEFAULT BAUD RATE (110)
#577  22057 026050        LDB .110
#578  22060 074117        LDA B
#579  22061 000134        ADA M3
#580  22062 072153        SAP *+3
#581  22063 160225 TERR1  JSM AERRA,I         ERROR, INVALID BAUD RATE
#582  22064 177642        DEC -7R
#583  22065 000251        ADA M159R
#584  22066 072453        SAP *-3
#585  22067 160227        JSM FXFLA,I         CALCULATE TIMING CONSTANT
#586  22070 020214        LDA AR2A
#587  22071 027650        LDB TF1A
#588  22072 170004        XFR
#589  22073 021717        LDA NUM
#590  22074 070442        SAR 10
#591  22075 070244        SAL 11
#592  22076 031654        STA T4              SAVE PAGE INFO
#593  22077 070002        SAR 1
#594  22100 031655        STA T5              TEMPORARY
#595  22101 003651        ADA TCON
#596  22102 027650        LDB TF1A
#597  22103 160221        JSM .FDVA,I
#598  22104 160225        JSM FLTRA,I
#599  22105 066105        JMP *               OVERFLOW (SHOULD NEVER HAPPEN)
#600  22106 004025        ADB .8
#601  22107 035652        STB T2              SAVE TIMING CONSTANT
#602*
#603  22110 025655        LDB T5
#604  22111 006016        ADB RE1A            FORM NEW REED LINK
#605  22112 035662        STB REFOL
#606*
#607  22113 021651        LDA T1
#608  22114 070502        SAR 11              2 * SELECT CODE
#609  22115 003335        ADA INTAD             + ADDRESS OF INTERRUPT TABLE - 1
#610  22116 025655        LDB T5
#611  22117 007336        ADB INTSA
#612  22120 070577        STB A,I             PUT INTERRUPT LINK ADDRESS IN TABLE
#614  22121 070742        SAR 16
#615  22122 031653        STA T3              SET TO "EVEN" PARITY
#616*
#617  22123 023641        LDA B4001           TURN ON CARRIER
#618  22124 041651        IOR T1
#619  22125 172741        STF 1
#620  22126 172141        OTA 1
#621  22127 173601        STC 1,C
```

```
          #01  TERMINAL OPTION BLOCK

@622*
@623  22130 @70742 RF0    SAR 16
@624  22131 @31430        STA LNINC       GET OUT OF AUTO
@625  22132 @31431        STA CLINE       INDICATE TOP OF PROGRAM
@626  22133 @31655        STA T5          SET PRINT/STORE TO "PRINT"
@627*
@628  22134 16@170 RF1    JSM CLERA,I     CLEAR THE DISPLAY
@629  22135 @21430        LDA LNINC
@630  22136 @70250        SZA *+5         AUTO MODE?
@631  22137 @25427        LDB .LNUM       YES, OUTPUT LINE NUMBER
@632  22140 16@210        JSM OUTIA,I
@633  22141 @45400        ISZ BADDR         FOLLOWED BY ONE BLANK
@634  22142 @66147        JMP R17
@635*
@636  22143 @20421        LDA .4          NO, SUPPLY "TEXT" MESSAGE
@637  22144 @27223        LDB MSSG1
@638  22145 @@5654 R10    ADB T4
@639  22146 @61335        JSM NPWRT
@640*
@641  22147 @24236 R17    LDB SLEND       SET UP FOR KEYBOARD INPUT
@642  22150 @35623        STB PEND          FOR OUTCR
@643  22151 @70742        SAR 16
@644  22152 @31442        STA STPFL         CLEAR STOP FLAG
@645  22153 @31473        STA CCNT          MULTIPLE CHARACTERS VIA KEYIN
@646  22154 @31704        STA BKPTR         CURSOR POINTER
@647  22155 @31707        STA BLANK         DO NOT IGNORE BLANKS
@648  22156 @31657        STA T7          SAY OK TO CLEAR DISPLAY ON INTERRUPT
@649  22157 @31660        STA T8          CLEAR NO-LINE-NUMBERS FLAG
@650*
@651  22160 162@12        JSM KEYIN,I     CALL FOR FIRST KEY
@652  22161 @1@105        CPA B377        DUMP I/O BUFFER?
@653  22162 @66346        JMP DIOB        YES
@654  22163 @1@074        CPA B22         BACKSPACE?
@655  22164 @66420        JMP BKSP        YES
@656  22165 @31661        STA T9
@657*
@658  22166 16@170        JSM CLERA,I     CLEAR THE DISPLAY
@659  22167 @21430        LDA LNINC
@660  22170 @70210        SZA *+4         AUTO MODE?
@661  22171 @25427        LDB .LNUM       YES, OUTPUT LINE NUMBER
@662  22172 16@210        JSM OUTIA,I
@663  22173 @45400        ISZ BADDR         FOLLOWED BY ONE BLANK
@664  22174 @21661        LDA T9
@665  22175 @66204        JMP R36
@667  22176 @20212 RF6    LDA .SLBF
@668  22177 @11400        CPA BADDR       DISPLAY EMPTY?
@669  22200 @66134        JMP RE1
@670  22201 162@12 R46    JSM KEYIN,I     CALL FOR NEXT KEY
@671  22202 @1@105        CPA B377        DUMP I/O BUFFER?
@672  22203 @66346        JMP DIOB        YES
@673  22204 @24104 R36    LDB B177
@674  22205 @35663        STB CTFLG       CLEAR ASCII-CNTRL FLAG
@675  22206 @74742        SBR 16
@676  22207 @35662        STB SHFLG       CLEAR ASCII-SHIFT FLAG
@677  22210 @74742 R26    SBR 16
@678  22211 @70346        RAR 8
@679  22212 @70133        SAP *+2,C       TEST BIT 7 FOR SHIFT
@680  22213 @24132        LDB M1
@681  22214 @35472        STB TEMPS       SET SHIFT FLAG
@682  22215 @70346        RAR 8
@683  22216 @31661        STA T9          INITIAL CODE WITHOUT SHIFT BIT
@684*
@685  22217 @27656        LDB EDITP
@686  22220 @@5654        ADB T4
@687  22221 16@175        JSM TSRHA,I     CODE IN EDIT TABLE?
@688  22222 @66245        JMP R16         NO
@689  22223 @@2225        ADA *+2
@690  22224 @70016        EXA             YES, GO EXECUTE IMMEDIATELY
@691*
@692  22225 @66226        JMP *+1
@693  22226 @66564        JMP OTPT  @     EXECUTE
@694  22227 @66@43        JMP WHAT  1     STEP
@695  22230 @66402        JMP RECL  2     RECALL
@696  22231 @66420        JMP BKSP  3     BACKSPACE
@697  22232 @66435        JMP FORD  4     FORWARDSPACE
@698  22233 @66715        JMP DNRW  5     DOWNARROW
@699  22234 @66722        JMP UPRW  6     UPARROW
@700  22235 @66332        JMP LEFT  7     LEFTARROW
@701  22236 @66341        JMP RITE  8     RIGHTARROW
@702  22237 @66677        JMP CLVR  9     CLEAR VAR
@703  22240 @6645@        JMP INDL  10    INSERT/DELETE
@704  22241 @66170        JMP RE@   11    CLEAR
@705  22242 @66662        JMP KEST  12    ACC
@706  22243 @66622        JMP PALL  13    PRINT ALL
@707  22244 @66513        JMP EOLL  14    EOL
@709  22245 @25472 R16    LDB TEMPS
@710  22246 @@@141        ADA M1@         DEFINED KEY?
@711  22247 @70253        SAP RE?         NO
@712  22250 @74150        SZB *+3         SHIFT?
@713  22251 @@3644        ADA .74         YES
@714  22252 @31661        STA T9
@715  22253 @67122        JMP KEYPR
```

#01 TERMINAL OPTION BLOCK

```
0716*
0717  22254 003642 RE2   ADA M22          CODE < 40B?
0718  22255 070612       SAM R13          YES
0719  22256 000154       ADA M64          CODE < 140B?
0720  22257 070353       SAP RE3          NO
0721*
0722  22260 074250       SZB *+5          SHIFT?
0723  22261 200040       ADA .31          YES. LOWER CASE LETTER?
0724  22262 070212       SAM RE3          NO
0725  22263 003645       ADA .97          YES
0726  22264 031661       STA T9
0727  22265 066314       JMP RE5
0728*
0729  22266 021661 RE3   LDA T9
0730  22267 026010       LDB VALUP
0731  22270 160175       JSM TSRHA,I      CODE IN VALUE TABLE?
0732  22271 066300 R13   JMP RDD          NO
0733  22272 000151       ADA M32          CODE < 40B?
0734  22273 070113       SAP *+2          NO
0735  22274 000101       ADA B100         YES
0736  22275 000041       ADA .32
0737  22276 031661       STA T9
0738  22277 066314       JMP RE5
0739*
0740  22300 021661 RDD   LDA T9
0741  22301 027676       LDB MULTP
0742  22302 005654       ADB T4
0743  22303 160175       JSM TSRHA,I      CODE IN MULTI-CHARACTER TABLE?
0744  22304 066043       JMP WHAT         NO. QUESTION IT
0745  22305 031661       STA T9
0746  22306 160170       JSM CLERA,I      CLEAR THE DISPLAY
0747  22307 021661       LDA T9
0748  22310 027714       LDB MULTN
0749  22311 005654       ADB T4
0750  22312 162015       JSM MCOUT,I      OUTPUT THE MNEMONIC
0751  22313 066176       JMP RE6
0753  22314 021661 RE5   LDA T9           ALL CONVERSION COMPLETE
0754  22315 051662       AND SHFLG        EXCEPT FOR ASCII-SHIFT & ASCII-CNTRL
0755  22316 070056       CMA
0756  22317 070137       LDB A
0757  22320 021661       LDA T9
0758  22321 041662       IOR SHFLG
0759  22322 074257       AND B
0760  22323 051663       AND CTFLG
0761  22324 025704       LDB BKPTR
0762  22325 074210       SZB RE9          CURSOR?
0763  22326 027637       LDB BKPTA        YES
0764  22327 061017       JSM OUTCR        REPLACE CHARACTER
0765  22330 066440       JMP FW2          SCROLL IF NECESSARY
0766*
0767  22331 061016 RE9   JSM UNEXT        PUT CHARACTER AT END OF LINE
0768*                                        AND SCROLL IF NECESSARY
0769* LEFTARROW
0770*
0771  22332 021400 LEFT  LDA HADDR        IS LAST CHARACTER AT RIGHT END OF DISPLAY?
0772  22333 000151       ADA M32
0773  22334 070076       TCA
0774  22335 001402       ADA OTPTR
0775  22336 070113       SAP *+2
0776  22337 045402       ISZ OTPTR        NO. SCROLL
0777  22340 066176       JMP RE6
0778*
0779* RIGHTARROW
0780*
0781  22341 021402 RITE  LDA OTPTR        IS FIRST CHARACTER AT LEFT END OF DISPLAY?
0782  22342 010212       CPA .SLRF
0783  22343 066176       JMP RE6
0784  22344 055402       DSZ OTPTR        NO. SCROLL
0785  22345 066176       JMP RE6
0787*
0788* DUMP I/O BUFFER FOR MODEM MESSAGE
0789*
0790  22346 021655 DIOB  LDA T5
0791  22347 070110       SZA *+2
0792  22350 164203       JMP ACTSA,I
0793*
0794  22351 024211       LDB .BUFA
0795  22352 035740       STB TMP
0796*
0797  22353 160261       JSM IOSTA,I      IS PRINTER BUSY?
0798  22354 066147       JMP R17          (STOP KEY RETURN)
0799*
0800  22355 021740 DIO2  LDA TMP          GET A CHARACTER
0801  22356 025740       LDB TMP
0802  22357 070002       SAR 1
0803  22360 070517       LDA A,I
0804  22361 074111       SLB *+2
0805  22362 070346       HAR 8
0806  22363 070342       SAR 8
```

```
        #01  TERMINAL OPTION BLOCK


0807  22364 010031        CPA EOL         IS IT EOL?
0808  22365 066374        JMP DIO3        YES
0809  22366 041447        IOR SLCOD       NO, OUTPUT IT
0810  22367 172741        STF 1
0811  22370 172141        OTA 1
0812  22371 173021        STC 1,C
0813  22372 045740        ISZ TMP
0814  22373 066355        JMP DIO2
0815*
0816  22374 020041 DIO3   LDA LNFD
0817  22375 041447        IOR SLCOD
0818  22376 172741        STF 1
0819  22377 172141        OTA 1
0820  22400 173021        STC 1,C
0821  22401 066147        JMP R17
0825*
0826* RECALL
0827*
0828  22402 160170 RECL   JSM CLERA,I     CLEAR THE DISPLAY
0829  22403 024211        LDB .BUFA
0830  22404 035740        STB TMP
0831  22405 061053 REC1   JSM GTMP        GET A CHARACTER
0832  22406 010031        CPA EOL
0833  22407 066176        JMP RE6         DONE
0834  22410 061016        JSM ONEXT       OUTPUT TO DISPLAY
0835  22411 021400        LDA BADDR
0836  22412 000151        ADA M32
0837  22413 070076        TCA
0838  22414 001402        ADA OTPTR
0839  22415 070113        SAP *+2
0840  22416 045402        ISZ OTPTR
0841  22417 066405        JMP REC1
0842*
0843* BACKSPACE CURSOR
0844*
0845  22420 021704 BKSP   LDA BKPTR
0846  22421 072310        SZA BK2         FIRST BACKSPACE?
0847  22422 021400        LDA BADDR       YES
0848  22423 010212        CPA .SLBF       NULL RECORD?
0849  22424 066176 BK3    JMP RE6         YES
0850  22425 031704        STA BKPTR
0851  22426 066433        JMP BK1         PUT POINTER AT LAST CHARACTER
0852*
0853  22427 010212 BK2    CPA .SLBF       ALREADY AT FIRST CHARACTER?
0854  22430 066176        JMP RE6         YES
0855  22431 011402        CPA OTPTR       LEFTMOST ON SCREEN?
0856  22432 055402        DSZ OTPTR       YES, SCROLL
0857  22433 055704 BK1    DSZ BKPTR       BACKSPACE CURSOR
0858  22434 066176        JMP RE6
0859*
0860* FORWARDSPACE CURSOR
0861*
0862  22435 021704 FORD   LDA BKPTR
0863  22436 071310        SZA BK3         IGNORE IF NO BACKSPACE HAS OCCURRED
0864  22437 045704        ISZ BKPTR
0865  22440 021402 FW2    LDA OTPTR
0866  22441 000041        ADA .32
0867  22442 011704        CPA BKPTR       ON RIGHT OF SCREEN?
0868  22443 011400        CPA BADDR       YES, LINE FULL?
0869  22444 066176        JMP RE6
0870  22445 045402        ISZ OTPTR       NO, SCROLL
0871  22446 066176        JMP RE6
0873*
0874* INSERT/DELETE
0875*
0876  22447 066176        JMP RE6
0877  22450 025704 INDL   LDB BKPTR       CHECK CURSOR POSITION
0878  22451 075710        SZB *-2         ANY BACKSPACE YET?
0879  22452 021472        LDA TEMPS
0880  22453 070650        SZA INSR        SHIFT FLAG SET?
0881*
0882* DELETE
0883*
0884  22454 055400        DSZ BADDR       YES, MOVE END-OF-LINE POINTER
0885  22455 035740        STB TMP
0886  22456 015400 DLC1   CPB BADDR       LAST CHARACTER?
0887  22457 066507        JMP INS2+1      YES, OUTPUT A BLANK
0888  22460 045740        ISZ TMP
0889  22461 061053        JSM GTMP        GET CHARACTER
0890  22462 055740        DSZ TMP
0891  22463 055740        DSZ TMP
0892  22464 024111        LDB TMPA        STORE SHIFTED LEFT ONE
0893  22465 061017        JSM OUTCR
0894  22466 025740        LDB TMP
0895  22467 066456        JMP DLC1
0896*
0897* INSERT
0898*
```

```
          #01  TERMINAL OPTION BLOCK


#899  22470 025400 INSR  LDA BADDR
#900  22471 014236       CPA SLEND
#901  22472 066176       JMP RE6          LINE FULL
#902  22473 004132       ADA M1
#903  22474 035740 INS1  STA TMP
#904  22475 061053       JSM GTMP         GET CHARACTER
#905  22476 025740       LDB TMP
#906  22477 015744       CPB BKPTR        AT CURSOR POSITION?
#907  22500 066506       JMP INS2         YES
#908  22501 024111       LDA TMPA         NO, MOVE RIGHT ONE
#909  22502 061017       JSM OUTCR
#910  22503 025740       LDA TMP
#911  22504 034134       ADA M3           BACK UP TO NEXT CHARACTER TO LEFT
#912  22505 066474       JMP INS1
#913*
#914  22506 045400 INS2  ISZ BADDR        UPDATE END-OF-LINE POINTER
#915  22507 020041       LDA .32
#916  22510 024111       LDB TMPA
#917  22511 061017       JSM OUTCR        INSERT A BLANK
#918  22512 066176       JMP RE6
#920*
#921* PROCESS STORE KEY
#922*
#923  22513 062542 EOLL  JSM MTIB         MOVE RECORD TO I/O BUFFER
#924  22514 060373       JSM ICNT         GET CHARACTER COUNT
#925  22515 010012       CPA .1
#926  22516 066134       JMP RE1          IGNORE NULL RECORD
#927*
#928  22517 062552 EOL1  JSM CRIT         SAVE CRITICAL PARAMETERS
#929  22520 061062       JSM GFRST        GET FIRST CHARACTER
#930  22521 066134       JMP RE1          (NULL RECORD RETURN)
#931  22522 060750       JSM DIGCK        DIGIT?
#932  22523 066620       JMP MAINS        NO, GIVE TO SYSTEM
#933*
#934  22524 060675       JSM INTCK-1      BUILD LINE #
#935  22525 035427       STB .LNUM        STORE LINE #
#936  22526 010031       CPA EOL          END OF LINE?
#937  22527 164244       JMP DELSA,I      YES, DELETE LINE
#938  22530 023640       LDA BSK          OPCODE ? W/SPECIAL BIT
#939  22531 131706       STA SBPTR,I
#940  22532 045706       ISZ SBPTR
#941  22533 023777       LDA OPCOD
#942  22534 131706       STA SBPTR,I
#943  22535 055400       DSZ BADDR
#944  22536 020031       LDA EOL
#945  22537 024012       LDB .1
#946  22540 160253       JSM CHRSA,I
#947  22541 164203       JMP ACTSA,I
#948*
#949*
#950* SUBROUTINE TO MOVE RECORD TO I/O BUFFER
#951*
#952  22542 045623 MTIB  ISZ PEND         GUARANTEE ROOM FOR EOL
#953  22543 020031       LDA EOL
#954  22544 061016       JSM ONEXT        STORE EOL
#955  22545 024233       LDB BEND
#956  22546 035476       STB TEMP3
#957  22547 035477       STB TEMP4
#958  22550 004046       ADA .41
#959  22551 065122       JMP MVTOH+1      MOVE THE RECORD AND RETURN
#960*
#961* SUBROUTINE TO SAVE CRITICAL PARAMETERS (T1 THRU T7)
#962*
#963  22552 025450 CRIT  LDB PMODE
#964  22553 076210       RZB *+4          PRINT ALL?
#965  22554 024211       LDB .BUFA        YES
#966  22555 061337       JSM NOWRT        (A-REG HAS COUNT)
#967  22556 061344       JSM CRLF1
#968*
#969  22557 023646       LDA AT1
#970  22560 027647       LDB AT1S
#971  22561 170004       XFR
#972  22562 170004       XFR
#973  22563 170402       RET
#975*
#976* PROCESS EXECUTE KEY
#977*
#978  22564 062542 OTPT  JSM MTIB         MOVE RECORD TO I/O BUFFER
#979  22565 060373       JSM ICNT         GET CHARACTER COUNT
#980  22566 010012       CPA .1
#981  22567 066134       JMP RE1          IGNORE NULL RECORD
#982*
#983  22570 062552       JSM CRIT         SAVE CRITICAL PARAMETERS
#984  22571 061062       JSM GFRST        GET FIRST CHARACTER
#985  22572 000000       NOP              (NULL RECORD RETURN)
#986  22573 027714       LDB MULTN
#987  22574 005654       ADB T4
#988  22575 160175       JSM TSRHA,I      SEARCH MULTI-CHARACTER TABLE
#989  22576 066620       JMP MAINS        NOT FOUND, GIVE TO MAIN SYSTEM
#990  22577 002601       ADA *+2
#991  22600 070016       EXA
```

#01 TERMINAL OPTION BLOCK

```
0992*
0993  22601 000002        JMP *+1
0994  22602 007013        JMP XLIST 0  LIST
0995  22603 006620        JMP MAINS 1  CONT
0996  22604 006620        JMP MAINS 2  TRACE
0997  22605 006620        JMP MAINS 3  RUN
0998  22606 006710        JMP XFECH 4  FETCH
0999  22607 006620        JMP MAINS 5  LOAD
1000  22610 006620        JMP MAINS 6  STORE
1001  22611 006620        JMP MAINS 7  NORMAL
1002  22612 006620        JMP MAINS 8  FIX
1003  22613 006620        JMP MAINS 9  FLT
1004  22614 006620        JMP MAINS 10 SCRATCH
1005  22615 006620        JMP MAINS 11 AUTO
1006  22616 006731        JMP XEXIT 12 EXIT ("STOP" KEY)
1007*
1008  22617 007007        JMP XCOMP 13 COMP
1009*
1010  22620 045416 MAINS  ISZ MODE          INDICATE CALC MODE
1011  22621 004357        JMP RETN2         RETURN TO MAIN SYSTEM
1013*
1014* PRINT ALL
1015*
1016  22622 160170 PALL   JSM CLERA,I       CLEAR THE DISPLAY
1017  22623 021450        LDA PMODE         REVERSE THE FLAG
1018  22624 070130        SIA *+2
1019  22625 070742        SAR 16
1020  22626 031450        STA PMODE
1021  22627 026634        LDB MON
1022  22630 070110        SZA *+2
1023  22631 026637        LDB MOF
1024  22632 020020        LDA .3
1025  22633 006145        JMP R10
1026*
1027  22634 001472 MON    DEF *+*+2-ID-ID
1028  22635 047516        ASC 2,ON
      22636 020040
1029  22637 001500 MOF    DEF *+*+2-ID-ID
1030  22640 047506        ASC 2,OFF
      22641 043040
1031*
1032* CHANGE PARITY
1033*
1034  22642 160170 EVOD   JSM CLERA,I       CLEAR THE DISPLAY
1035  22643 021653        LDA T3            REVERSE THE FLAG
1036  22644 072113        SAP *+2,S
1037  22645 070073        SAP *+1,C
1038  22646 031653        STA T3
1039  22647 026654        LDB MEV
1040  22650 070113        SAP *+2
1041  22651 026657        LDB MOD
1042  22652 020021        LDA .4
1043  22653 006145        JMP R10
1044*
1045  22654 001532 MEV    DEF *+*+2-ID-ID
1046  22655 042526        ASC 2,EVEN
      22656 042516
1047  22657 001540 MOD    DEF *+*+2-ID-ID
1048  22660 047504        ASC 2,ODD
      22661 042040
1050*
1051* ACC
1052*
1053  22662 021472 REST   LDA TEMPS         CHECK SHIFT FLAG
1054  22663 070210        SZA *+4
1055  22664 020101        LDA B100          USE "@"
1056  22665 031661        STA T9
1057  22666 006314        JMP RES
1058*
1059  22667 020020        LDA .3            USE "RES"
1060  22670 026674        LDB RESM
1061  22671 025654        ADB T4
1062  22672 061335        JSM NPWRT
1063  22673 006176        JMP RE6
1064*
1065  22674 001572 RESM   DEF *+*+2-ID-ID
1066  22675 051105        ASC 2,RES
      22676 051440
1067*
1068* CLEAR VAR
1069*
1070  22677 020056 CLVR   LDA .1E4
1071  22700 164171        JMP RUNA,I
1072*
1073* EXIT FROM TERMINAL MODE
1074*
1075  22701 021651 XEXIT  LDA T1            TURN OFF CARRIER
1076  22702 172741        STF 1
1077  22703 172141        OTA 1
```

```
        #01  TERMINAL OPTION BLOCK


1078  22704 173641        STC 1,C
1079  22705 024116        LDB QTOP        (=2000B)
1080  22706 035462        STB REEDL       RESTORE REED LINK
1081  22707 164174        JMP RDYDA,I
1083*
1084* FETCH
1085*
1086  22710 162013 XFECH  JSM GETLN,I     GET PARAMETER
1087  22711 020012        LDA .1
1088  22712 066713        JMP *+1         NOP
1089  22713 000132        ADA M1
1090  22714 031431        STA CLINF
1091*
1092* DOWNARROW
1093*
1094  22715 025424 DNRW   LDB PBUFF       CHECK FOR NULL PROGRAM
1095  22716 015423        CPB PBPTR
1096  22717 066134        JMP RE1
1097  22720 062731        JSM DNRWS
1098  22721 066726        JMP UPR1
1099*
1100* UPARROW
1101*
1102  22722 025424 UPRW   LDB PBUFF       CHECK FOR NULL PROGRAM
1103  22723 015423        CPB PBPTR
1104  22724 066134        JMP RE1
1105  22725 062737        JSM UPRWS
1106  22726 070742 UPR1   SAR 16
1107  22727 031430        STA LNINC       GET OUT OF AUTO
1108  22730 066147        JMP R17
1109*
1110* DOWNARROW SUBROUTINE
1111*
1112  22731 021431 DNRWS  LDA CLINF
1113  22732 070070        SIA *+1
1114  22733 061075        JSM FNDPS       FIND LINE
1115  22734 066747        JMP U3          PAST END OF PROGRAM
1116  22735 066753        JMP U5          BETWEEN LINES
1117  22736 066753        JMP U5
1118*
1119* UPARROW SUBROUTINE
1120*
1121  22737 021431 UPRWS  LDA CLINF
1122  22740 072110        RZA *+2         CLINE=0?
1123  22741 020056        LDA .1E4        YES, FIND LAST LINE OF PROGRAM
1124  22742 061075        JSM FNDPS       FIND LINE
1125  22743 066747        JMP U3          PAST END OF PROGRAM
1126  22744 066745        JMP *+1         BETWEEN LINES
1127  22745 015424        CPB PBUFF       FIRST LINE?
1128  22746 066753        JMP U5          YES
1129*
1130  22747 021720 U3     LDA LASTN       GET PREVIOUS LINE
1131  22750 060742        JSM FND
1132  22751 070742        SAR 16
1133  22752 031656        STA T6
1134*
1135  22753 035661 U5     STB T9          SAVE ADDRESS
1136  22754 035472        STB TEMPS
1138*
1139* SUBROUTINE TO OUTPUT A LINE TO DISPLAY BUFFER
1140*
1141* LOCATION IS IN T3
1142*
1143  22755 160170 LNOUT  JSM CLERA,I     CLEAR THE DISPLAY
1144  22756 125661        LDB T9,I        GET LINE #
1145  22757 045661        ISZ T9
1146  22760 035431        STB CLINE       UPDATE CLINE
1147*
1148  22761 021661        LDA T9
1149  22762 070070        SIA *+1
1150  22763 070517        LDA A,I
1151  22764 050065        AND OPMSK       (076000)
1152  22765 013777        CPA OPCOD       TEXT?
1153  22766 066771        JMP *+3         YES
1154  22767 025472        LDB TEMPS
1155  22770 166014        JMP LISTS,I
1156*
1157  22771 021660        LDA T8          GET NO-LINE-NUMBERS FLAG
1158  22772 072110        RZA *+2
1159  22773 160210        JSM OUTIA,I     OUTPUT LINE #
1160  22774 121661        LDA T9,I
1161  22775 045661        ISZ T9
1162  22776 025661        LDB T9
1163  22777 050104        AND B177        CALCULATE CHARACTER COUNT
1164  23000 000134        ADA M3
1165  23001 070037        ADR A
1166  23002 070744        SAL 1
1167  23003 074537        LDR B,I
1168  23004 074404        SBL 8
```

```
        #01  TERMINAL OPTION BLOCK


1169  23005 074110          SZA **2
1170  23006 070070          STA **1
1171  23007 025661          LDB T9          CALCULATE CHARACTER POINTER
1172  23010 074744          SBL 1
1173  23011 074070          STB **1
1174  23012 065335          JMP NPWRT       OUTPUT TO BUFFER
1176*
1177* LIST ROUTINE
1178*
1179  23013 061055 XLIST    JSM GNEXT       GET NEXT CHARACTER
1180  23014 013062          CPA B130        "X"?
1181  23015 067051          JMP XLIS3       YES
1182  23016 013063          CPA B43         "#"?
1183  23017 067053          JMP XLIS4       YES
1184  23020 055400          DSZ BADDR       NO, RESET POINTER
1185  23021 162013 XLIS1    JSM GETLN,I     GET OPTIONAL LINE NUMBERS
1186  23022 020012          LDA .1
1187  23023 024056          LDB .1E4
1188  23024 000132          ADA M1
1189  23025 031431          STA CLINE
1190  23026 035656          STB T6          SAVE ENDING LINE NUMBER
1191  23027 025424          LDB PBUFF       CHECK FOR NULL PROGRAM
1192  23030 015423          CPB PBPTR
1193  23031 066130          JMP RE0
1194*
1195  23032 062731 XLIS2    JSM DNRWS       OUTPUT LINE TO DISPLAY BUFFER
1196  23033 025431          LDB CLINE
1197  23034 074076          TCB
1198  23035 035656          ADB T6          T6 - CLINE
1199  23036 074152          SRM **3
1200  23037 025442          LDB STPFL       CHECK STOP FLAG
1201  23040 014040          CPB .31
1202  23041 067064          JMP LOUT
1203  23042 021651          LDA T1
1204  23043 011447          CPA SLCOD       CHECK SELECT CODES
1205  23044 067057          JMP XLIS5
1206*
1207  23045 060373          JSM ICNT        GET CHARACTER COUNT
1208  23046 061337          JSM NOWRT       PRINT THE LINE
1209  23047 061344          JSM CRLF1
1210  23050 067032          JMP XLIS2
1211*
1212  23051 045660 XLIS3    ISZ T8          SET NO-LINE-NUMBERS FLAG
1213  23052 067013          JMP XLIST
1214  23053 060664 XLIS4    JSM LNUM        FETCH SELECT CODE
1215  23054 074204          SBL 12
1216  23055 035447          STB SLCOD
1217  23056 067021          JMP XLIS1
1218*
1219  23057 062542 XLIS5    JSM MTIR        MOVE RECORD TO I/O BUFFER
1220  23060 063246          JSM MDRVR       CALL MODEM DRIVER
1221  23061 067032          JMP XLIS2
1222*
1223  23062 000130 B130     OCT 130
1224  23063 000043 B43      OCT 43
1225*
1226  23064 024127 LOUT     LDB B170K
1227  23065 035447          STB SLCOD
1228  23066 066130          JMP RE0
1230*
1231* COMPILE ROUTINE
1232*
1233  23067 020132 XCOMP    LDA M1
1234  23070 031433          STA TFLAG
1235  23071 020041          LDA .32
1236  23072 031707          STA BLANK
1237  23073 162013          JSM GETLN,I     GET OPTIONAL LINE NUMBERS
1238  23074 020012          LDA .1
1239  23075 024056          LDB .1E4
1240  23076 000132          ADA M1
1241  23077 031431          STA CLINE
1242  23100 035656          STB T6          SAVE ENDING LINE NUMBER
1243  23101 025424          LDB PBUFF       CHECK FOR NULL PROGRAM
1244  23102 015423          CPB PBPTR
1245  23103 067117          JMP XCOM2       DONE
1246*
1247  23104 062731 XCOM1    JSM DNRWS       OUTPUT LINE TO DISPLAY BUFFER
1248  23105 025431          LDB CLINE
1249  23106 074076          TCB
1250  23107 035656          ADB T6          T6 - CLINE
1251  23110 074352          SBM XCOM2
1252*
1253  23111 062542          JSM MTIR        MOVE LINE TO I/O BUFFER
1254  23112 027646          LDA ATI
1255  23113 027647          LDB ATIS
1256  23114 170004          XFR
1257  23115 170004          XFR
1258  23116 064357          JMP RETN2       GO COMPILE
1259*
1260  23117 070742 XCOM2    SAR 16
1261  23120 031433          STA TFLAG
1262  23121 066130          JMP RE0
```

#01 TERMINAL OPTION BLOCK

```
1264*
1265* PROCESS DEFINED KEY
1266*
1267  23122 @21661 KEYPR LDA T9          CHECK FOR SPECIAL FUNCTION
1268  23123 @10022       CPA .5
1269  23124 @67212       JMP SETAS       SET ASCII-SHIFT
1270  23125 @10023       CPA .6
1271  23126 @67215       JMP SETAC       SET ASCII-CNTRL
1272  23127 @10024       CPA .7
1273  23130 @67226       JMP PRST        SET PRINT/STORE
1274  23131 @10025       CPA .8
1275  23132 @66642       JMP EVOD        CHANGE PARITY
1276  23133 @10026       CPA .9
1277  23134 @67237       JMP XMIT        TRANSMIT
1278*
1279  23135 @24166       LDB FWAM        SEARCH FOR KEY, STARTING AT FWAM
1280  23136 @15405 KEY1  CPB FWUP
1281  23137 @66043       JMP WHAT        UNDEFINED
1282  23140 @74517       LDA B,I
1283  23141 @74070       SIB *+1
1284  23142 @11661       CPA T9
1285  23143 @67146       JMP *+3         FOUND IT
1286  23144 @74437       ADB B,I
1287  23145 @67136       JMP KEY1        KEEP SEARCHING
1288*
1289  23146 @74517       LDA B,I
1290  23147 @74070       SIB *+1
1291  23150 @10012       CPA .1          NULL DEFINITION?
1292  23151 @66043       JMP WHAT        YES
1293  23152 @31466       STA LOCL1       NO, SAVE LENGTH WORD
1294  23153 @74517       LDA B,I
1295  23154 @74070       SIB *+1
1296  23155 @70110       SZA *+2         IS THIS A TYPING AID?
1297  23156 @66043       JMP WHAT        NO
1298  23157 @74744       SBL 1           YES
1299  23160 @35740       STB TMP         SAVE CHARACTER POINTER
1300  23161 @21466       LDA LOCL1
1301  23162 @70744       SAL 1
1302  23163 @00136       ADA M5          ASSUME ODD COUNT
1303  23164 @31473       STA CCNT        SAVE CHARACTER COUNT
1304  23165 @70037       ADB A
1305  23166 @74517       LDA B,I         LOOK AT LAST WORD
1306  23167 @50105       AND B377
1307  23170 @70450       SZA *+9
1308  23171 @45473       ISZ CCNT        CORRECT COUNT
1309  23172 @10102       CPA ASTR1       CHECK FOR *
1310  23173 @67175       JMP *+2
1311  23174 @66201       JMP R46
1312  23175 @74517       LDA B,I
1313  23176 @50157       AND M256
1314  23177 @40027       IOR AIEX1       IEX CODE
1315  23200 @67206       JMP *+6
1316  23201 @74517       LDA B,I
1317  23202 @13210       CPA ASTR2       CHECK FOR *
1318  23203 @67205       JMP *+2
1319  23204 @66201       JMP R46
1320  23205 @23211       LDA AIEX2       IEX CODE
1321  23206 @74557       STA B,I
1322  23207 @66201       JMP R46         EXECUTE IMMEDIATELY
1323*
1324  00100       ASTR1 EQU B52
1325  23210 @25000 ASTR2 OCT 025000
1326  00027       AIEX1 EQU .11
1327  23211 @05400 AIEX2 OCT 005400
1328*
1329*
1330  23212 @24033 SETAS LDB B20
1331  23213 @35662       STB SHFLG
1332  23214 @67217       JMP *+3
1333  23215 @24054 SETAC LDB .63
1334  23216 @35663       STB CTFLG
1335  23217 162012       JSM KEYIN,I
1336  23220 @10105       CPA B377        DUMP I/O BUFFER?
1337  23221 @66346       JMP DIOB        YES
1338  23222 @66210       JMP R26
1339*
1340  23223 @02450 MSSG1 DEF *+*+2-ID-ID
1341  23224 @52105       ASC 2,TEXT
      23225 *54124
1342*
1343  23226 162170 PRST  JSM CLERA,I     CLEAR THE DISPLAY
1344  23227 @20062       LDA FLGRT
1345  23230 @31655       STA T5
1346  23231 @27234       LDB STMS
1347  23232 @20021       LDA .4
1348  23233 @66145       JMP R10
1349*
1350  23234 @02472 STMS  DEF *+*+2-ID-ID
1351  23235 @51501       ASC 2,SAVE
      23236 @53105
```

#01 TERMINAL OPTION BLOCK

```
1352*
1353*
1354  23237 062542 XMIT  JSM MTIB        MOVE RECORD TO I/O BUFFER
1355  23240 060373       JSM ICNT        GET CHARACTER COUNT
1356  23241 010012       CPA .1
1357  23242 066134       JMP RE1         IGNORE NULL RECORD
1358*
1359  23243 062552       JSM CRIT
1360  23244 063246       JSM MDRVR       CALL MODEM
1361  23245 066134       JMP RE1
1363*
1364* MODEM OUTPUT DRIVER
1365*
1366  23246 021651 MDRVR LDA T1          GET SELECT CODE
1367  23247 043641       IOR B4001
1368  23250 063625       JSM IOST        WAIT FOR MODEM TO BE READY
1369  23251 170402       RET             STOP-FLAG RETURN
1370*
1371  23252 172741       STF 1           TURN OFF INTERRUPT
1372  23253 020211       LDA .BUFA       SET TO START OF I/O BUFFER
1373  23254 031400       STA BADDR
1374*
1375  23255 021400 MDR1  LDA BADDR
1376  23256 025400       LDB BADDR
1377  23257 070002       SAR 1
1378  23260 070517       LDA A,I         FETCH TWO CHARACTERS
1379  23261 074111       SLB *+2
1380  23262 070346       HAR 8
1381  23263 070342       SAR 8
1382  23264 010031       CPA EOL         IS IT EOL?
1383  23265 067271       JMP MDR3        YES
1384  23266 063277       JSM MDRS        NO
1385  23267 045400       ISZ BADDR
1386  23270 067255       JMP MDR1
1387*
1388  23271 020031 MDR3  LDA CRTN        OUTPUT CRTN
1389  23272 063277       JSM MDRS
1390  23273 020194       LDA B177        OUTPUT RUBOUT
1391  23274 063277       JSM MDRS
1392  23275 173741       CLF 1
1393  23276 170402       RET
1395*
1396* SERIALIZING SUBROUTINE
1397*
1398  23277 031661 MDRS  STA T9
1399  23300 025653       LDB T3          INCLUDE PARITY
1400  23301 074111       SLA *+2
1401  23302 004062       ADB FLGBT
1402  23303 070002       SAR 1
1403  23304 073650       RZA *-3
1404  23305 021661       LDA T9
1405  23306 074342       SBR 8
1406  23307 074017       ADA B
1407  23310 040110       IOR B1400       INCLUDE STOP BITS
1408  23311 070744       SAL 1
1409  23312 031661       STA T9
1410  23313 025652       LDB T2          SET UP BIT COUNTER
1411  23314 004106       ADB B777        SWITCH TO SINGLE STOP BIT @ 130 BAUD
1412  23315 074700       AHR 15
1413  23316 004141       ADB M10
1414  23317 035470       STB L2T1
1415  23320 041651 MDR2  IOR T1          "OR" IN SELECT CODE
1416  23321 040121       IOR B4000       "OR" IN SO3
1417  23322 172141       OTA 1           LOAD I/O REGISTER
1418  23323 172601       STC 1           OUTPUT THE DATA
1419  23324 172741       STF 1
1420  23325 021661       LDA T9          SHIFT DATA RIGHT
1421  23326 070002       SAR 1
1422  23327 031661       STA T9          SAVE SHIFTED DATA
1423  23330 025652       LDB T2          LOAD TIMING CONSTANT
1424  23331 076030       RIB *           LOOP FOR TIMING
1425  23332 045470       ISZ L2T1        INCREMENT BIT COUNTER
1426  23333 067320       JMP MDR2
1427  23334 170402       RET
1429*
1430* SERVICE MODEM INTERRUPTS
1431*
1432  23335 040377 INTAD DEF 40377B      ADDRESS OF INTERRUPT TABLE - 1
1433  23336 001337 INTSA DEF *+1-ID
1434*
1435  23337 074742       SBR 16          INITIALIZE DATA WORD
1436  23340 023643       LDA M8          SET UP BIT COUNTER
1437  23341 031446       STA ADRSS
1438  23342 021652       LDA T2
1439  23343 070000       AAR 1           LOAD HALF-WAIT
1440  23344 000042       ADA .33         ALLOW FOR DELAY IN INTERRUPT ROUTINE
1441  23345 072030       RIA *           LOOP FOR TIMING
1442*
```

#01 TERMINAL OPTION BLOCK

```
543  23346 021651 MDU1  LDA T1          GET SELECT CODE
544  23347 043641       IOR B4001
545  23350 172141       OTA 1
546  23351 172601       STC 1           SAMPLE
547  23352 172741       STF 1
548  23353 172241       LIA 1
549  23354 074002       SBR 1           POSITION OLD DATA WORD
550  23355 073111       SLA *+2
551  23356 040062       ADB FLGBT
552  23357 021652       LDA T2          LOAD TIMING CONSTANT
553  23360 072030       HIA *           LOOP FOR TIMING
554  23361 045446       ISZ ADRSS
555  23362 067346       JMP MDU1
556  23363 074422       SBR 9
557  23364 035446       STB ADRSS       SAVE CODE
558*
559  23365 021655       LDA T5
560  23366 072110       SZA *+2
561  23367 067507       JMP MDSS
562*
563  23370 014031       CPB CRTN
564  23371 067463       JMP SCRN
565  23372 034151       ADB M32         CODE < 40B?
566  23373 074152       SBM *+3         YES
567  23374 034154       ADB M64         CODE < 140B?
568  23375 074112       SBM *+2         YES
569  23376 067450       JMP INT?
570*
571  23377 021657       LDA T7          OK TO CLEAR DISPLAY?
572  23400 072210       SZA *+4
573  23401 162170       JSM CLEA,I      CLEAR THE DISPLAY
574  23402 020211       LDA .BUFA
575  23403 031657       STA T7
576*
577  23404 021400       LDA BADDR       STASH IN DISPLAY BUFFER
578  23405 014236       CPA SLEND
579  23406 067450       JMP INT?        AT END OF BUFFER
580  23407 025400       LDB BADDR
581  23410 045400       ISZ BADDR
582  23411 074006       RBR 1
583  23412 070002       SAR 1
584  23413 070517       LDA A,I         FETCH WORD IN BUFFER
585  23414 074112       SBM *+2
586  23415 070346       RAR 8
587  23416 050157       AND M256
588  23417 041446       IOR ADRSS
589  23420 074132       SBM *+2,C
590  23421 070346       RAR 8
591  23422 074557       STA B,I
592*
593  23423 021400       LDA BADDR       SCROLL IF NECESSARY
594  23424 000151       ADA M32
595  23425 070076       TCA
596  23426 001402       ADA OTPTR
597  23427 074113       SAP *+2
598  23430 045402       ISZ OTPTR
599*
600  23431 021657 MDU2  LDA T7          STASH IN I/O BUFFER
601  23432 013506       CPA IOEND
602  23433 067450       JMP INT?        AT END OF BUFFER
603  23434 025657       LDB T7
604  23435 045657       ISZ T7
605  23436 074006       RBR 1
606  23437 070002       SAR 1
607  23440 070517       LDA A,I         FETCH WORD IN BUFFER
608  23441 074112       SBM *+2
609  23442 070346       RAR 8
610  23443 050157       AND M256
611  23444 041446       IOR ADRSS
612  23445 074132       SBM *+2,C
613  23446 070346       RAR 8
614  23447 074557       STA B,I
615*
616  23450 021651 INT?  LDA T1          GET SELECT CODE
617  23451 043641       IOR B4001
618  23452 172141       OTA 1
619  23453 172601       STC 1
620  23454 172741       STF 1
621  23455 172241       LIA 1
622  23456 071511       SLA INT?        WAIT FOR STOP BIT
623  23457 021444       LDA SAVEA       RESTORE A-REG
624  23460 025445       LDB SAVEB       RESTORE B-REG
625  23461 173741       CLF 1           RESTORE INTERRUPT
626  23462 170402       RET
627*
628  23463 020105 SCRN  LDA B377
629  23464 031443       STA KEYFG       SAY TO DUMP I/O BUFFER
630*
631  23465 021657       LDA T7          PUT EOL IN I/O BUFFER
632  23466 072110       SZA *+2
```

#01 TERMINAL OPTION BLOCK

```
1533  23467 020211        LDA .BUFA
1534  23470 070137        LDB A
1535  23471 074006        RBR 1
1536  23472 070002        SAR 1
1537  23473 070517        LDA A,I
1538  23474 074112        SBM *+2
1539  23475 070346        RAR 8
1540  23476 050157        AND M256
1541  23477 040031        IOR EOL
1542  23500 074132        SBM *+2,C
1543  23501 070346        RAR 8
1544  23502 074557        STA B,I
1545  23503 070742        SAR 16
1546  23504 031657        STA T7
1547  23505 067450        JMP INT2
1548*
1549  23506 003320 IOEND DEF IOBUF+IOBUF+80
1550*
1551*
1552* STORE FROM INTERRUPT
1553*
1554  23507 014031 MDSS   CPB CRTN
1555  23510 067606        JMP MDS5
1556  23511 004151        ADB M32         CODE < 40B?
1557  23512 074152        SBM *+3         YES
1558  23513 004154        ADB M64         CODE < 140B?
1559  23514 074112        SBM *+2         YES
1560  23515 067450        JMP INT2
1561*
1562  23516 021657        LDA T7
1563  23517 072610        RZA MDS2
1564  23520 021446        LDA ADRSS       PROCESS FIRST CODE
1565  23521 060750        JSM DIGCK       DIGIT?
1566  23522 067530        JMP MDS1        NO
1567  23523 035722        STB INTGR       YES, SAVE IT
1568  23524 024233        LDB SBUFA
1569  23525 035706        STB SBPTR
1570  23526 021133        LDA M2
1571  23527 067531        JMP *+2
1572  23530 021132 MDS1   LDA M1
1573  23531 031657        STA T7
1574  23532 067450        JMP INT2
1575*
1576  23533 010132 MDS2   CPA M1          IGNORE?
1577  23534 067450        JMP INT2        YES
1578  23535 010133        CPA M2          IN LINE NUMBER?
1579  23536 067557        JMP MDS3
1580  23537 021657 MDST   LDA T7          NO, STORE CHARACTER
1581  23540 010236        CPA SLEND
1582  23541 067450        JMP INT2        AT END OF BUFFER
1583  23542 025657        LDB T7
1584  23543 045657        ISZ T7
1585  23544 074006        RBR 1
1586  23545 070002        SAR 1
1587  23546 070517        LDA A,I         FETCH WORD IN BUFFER
1588  23547 074112        SBM *+2
1589  23550 070346        RAR 8
1590  23551 050157        AND M256
1591  23552 041446        IOR ADRSS
1592  23553 074132        SBM *+2,C
1593  23554 070346        RAR 8
1594  23555 074557        STA B,I
1595  23556 067450        JMP INT2
1596*
1597  23557 021446 MDS3   LDA ADRSS       PROCESS LINE NUMBER
1598  23560 060750        JSM DIGCK       DIGIT?
1599  23561 067571        JMP MDS4        NO
1600  23562 021722        LDA INTGR
1601  23563 070704        SAL 2
1602  23564 001722        ADA INTGR
1603  23565 070744        SAL 1
1604  23566 074017        ADA B
1605  23567 031722        STA INTGR
1606  23570 067450        JMP INT2
1607*
1608  23571 021722 MDS4   LDA INTGR
1609  23572 031427        STA .LNUM
1610  23573 131706        STA SBPTR,I     STORE LINE NUMBER
1611  23574 045706        ISZ SBPTR
1612  23575 023640        LDA 95K
1613  23576 131706        STA SBPTR,I     STORE SPECIAL OPCODE
1614  23577 045706        ISZ SBPTR
1615  23600 023777        LDA OPCOD
1616  23601 131706        STA SBPTR,I     STORE PAGE CODE
1617  23602 024212        LDB .SLBF
1618  23603 004022        ADB .5
1619  23604 035657        STB T7
1620  23605 067537        JMP MDST
1621*
1622  23606 021657 MDS5   LDA T7          PROCESS CRTN
```

```
          #01  TERMINAL OPTION BLOCK


1623  23607 070550        SZA MDS6
1624  23610 010132        CPA M1
1625  23611 067622        JMP MDS6
1626  23612 010133        CPA M2
1627  23613 067622        JMP MDS6
1628  23614 021657        LDA T7
1629  23615 073002        SAR 1
1630  23616 070070        STA **+1
1631  23617 031706        STA SAPTR
1632  23620 020195        LDA B377
1633  23621 031443        STA KEYFG
1634  23622 073742 MDS6   SAR 16
1635  23623 031657        STA T7
1636  23624 067458        JMP INT2
1637*
1638* I/O STATUS ROUTINE
1639*
1640* LDA SELECT CODE (SAVED)
1641* JSM IOST
1642*
1643* RETURN TO P+1 IF STOP FLAG
1644*           P+2 OTHERWISE
1645*
1646  23625 025442 IOST   LDB STPFL       CHECK STOP FLAG
1647  23626 014040        CPB .31
1648  23627 170432        RET
1649*
1650  23630 172741        STF 1           TURN OFF INTERRUPT SYSTEM
1651  23631 172141        OTA 1           OUTPUT SELECT CODE
1652  23632 172741        STF 1           LOAD DEVICE STATUS
1653  23633 177241        LIB 1,C          AND TURN-ON INTERRUPT
1654  23634 074406        RBR 9
1655  23635 075413        SBP IOST        LOOP IF BUSY (TESTS ST0)
1656  23636 064357        JMP RETN2
1658  23637 001704 AKPTA  DEF AKPTR
1659  23640 005000 A5K    OCT 5000
1660  23641 004001 A4001  OCT 4001
1661  23642 177752 M22    DEC -22
1662  23643 177770 M8     DEC -8
1663  23644 000112 .74    DEC 74
1664  23645 000141 .97    DEC 97
1665*
1666  23646 001651 AT1    DEF T1
1667  23647 040424 AT1S   DEF 40424B
1668  23650 001652 TF1A   DEF TF1
1669  23651 001652 TCON   DEF **1-10      CONSTANT FOR TIMING LOOP
1670  23652 002001        OCT 002001  10*4
1671  23653 062531        OCT 062531  6559
1672  23654 021451        OCT 021451  2329
1673  23655 104511        OCT 104511  8949
1674*
1675* EDIT KEY TABLE (IMMEDIATE EXECUTION)
1676*
1677  23656 003536 EDITP  DEF ***+2-ID-ID
1678  23657 005600        OCT 05600 EXECUTE          13  0
1679  23660 006601        OCT 06601 STEP             15  1
1680  23661 010202        OCT 10202 RECALL           20  2
1681  23662 011203        OCT 11203 BACKSPACE        22  3
1682  23663 011604        OCT 11604 FORWARDSPACE     23  4
1683  23664 012205        OCT 12205 DOWNARROW        24  5
1684  23665 012606        OCT 12606 UPARROW          25  6
1685  23666 013207        OCT 13207 LEFTARROW        26  7
1686  23667 013610        OCT 13610 RIGHTARROW       27  8
1687  23670 015211        OCT 15211 CLEAR VAR        32  9
1688  23671 017612        OCT 17612 INSERT/DELETE    37 10
1689  23672 065213        OCT 65213 CLEAR           152 11
1690  23673 065614        OCT 65614 ACC             153 12
1691  23674 070215        OCT 70215 PRINT ALL       160 13
1692  23675 071316        OCT 71316 EOL             162 14   END
1694*
1695* MULTI-CHARACTER KEY TABLE
1696*
1697  23676 003576 MULTP  DEF ***+2-ID-ID
1698  23677 005200        OCT 05200 LIST     12 0
1699  23700 006201        OCT 06201 CONT     14 1
1700  23701 007202        OCT 07202 TRACE    16 2
1701  23702 007603        OCT 07603 RUN      17 3
1702  23703 010604        OCT 10604 FETCH    21 4
1703  23704 014205        OCT 14205 LOAD     30 5
1704  23705 014606        OCT 14606 STORE    31 6
1705  23706 017207        OCT 17207 NORMAL   36 7
1706  23707 072210        OCT 72210 FIX     164 8
1707  23710 072611        OCT 72611 FLT     165 9
1708  23711 073212        OCT 73212 SCRATCH 166 10
1709  23712 073613        OCT 73613 AUTO    167 11
1710  23713 077714        OCT 77714 EXIT    177 12   END
1711*
1712* MULTI-CHARACTER MNEMONIC TABLE
1713*
1714  23714 003632 MULTN  DEF ***+2-ID-ID
```

```
          #01  TERMINAL OPTION BLOCK

1715  23715 046111        OCT 046111  L   I
1716  23716 051524        OCT 051524  S   T
1717  23717 100103        OCT 100103  (0) C
1718  23720 047516        OCT 047516  O   N
1719  23721 052201        OCT 052201  T   (1)
1720  23722 052122        OCT 052122  T   R
1721  23723 040503        OCT 040503  A   C
1722  23724 042602        OCT 042602  E   (2)
1723  23725 051125        OCT 051125  R   U
1724  23726 047203        OCT 047203  N   (3)
1725  23727 043105        OCT 043105  F   E
1726  23730 052103        OCT 052103  T   C
1727  23731 044204        OCT 044204  H   (4)
1728  23732 046117        OCT 046117  L   O
1729  23733 040504        OCT 040504  A   D
1730  23734 102523        OCT 102523  (5) S
1731  23735 052117        OCT 052117  T   O
1732  23736 051105        OCT 051105  R   E
1733  23737 103116        OCT 103116  (6) N
1734  23740 047522        OCT 047522  O   R
1735  23741 046501        OCT 046501  M   A
1736  23742 046207        OCT 046207  L   (7)
1737  23743 043111        OCT 043111  F   I
1738  23744 054210        OCT 054210  X   (8)
1739  23745 043114        OCT 043114  F   L
1740  23746 052211        OCT 052211  T   (9)
1741  23747 051503        OCT 051503  S   C
1742  23750 051101        OCT 051101  R   A
1743  23751 052103        OCT 052103  T   C
1744  23752 044212        OCT 044212  H   (10)
1745  23753 040525        OCT 040525  A   U
1746  23754 052117        OCT 052117  T   O
1747  23755 105505        OCT 105505  (11)E
1748  23756 054111        OCT 054111  X   I
1749  23757 052214        OCT 052214  T   (12)
1750  23760 041517        OCT 041517  C   O
1751  23761 046520        OCT 046520  M   P
1752  23762 146400        OCT 146400  (13)       END
1754  23763 160205 ERRX   JSM AERRA,I       MODE ERROR
1755  23764 177661        DEC -79
1756*
1757  23765 003754 WHAM   DEF ***+2-ID-ID
1758  23766 053510        ASC 3,WHAT?
      23767 040524
      23770 037440
1759  00000         LEFF  EQU 23771B-*
1760*
1761  01651               ORG 1651B
1762*
1763  01740         TMP   EQU XC
1764  01473         CONT  EQU TEMPS+1
1765  01704         BKPTR EQU WIDTH
1766*
1767  01651 000040 T1     NOP             SELECT CODE
1768  01652 000000 T2     NOP             TIMING CONSTANT
1769  01653 000000 T3     NOP             PARITY INDICATOR
1770  01654 000000 T4     NOP             PAGE INFO
1771  01655 000030 T5     NOP             PRINT/STORE FLAG
1772  01656 000000 T6     NOP             FOR LISTING
1773  01657 020000 T7     NOP             OK/NOT OK TO CLEAR DISPLAY ON INTERRUPT
1774  01660 000000 T8     NOP             NO-LINE-NUMBERS FLAG
1775  01661 000000 T9     NOP
1776*
1777  01662 000000 TF1    BSS 4           FLOATING TEMPORARY
1778  01662         SHFLG EQU TF1
1779  01663         CTFLG EQU TF1+1
1780*
1781  23771               ORG 23771B
1782*
1783  23771 177772        ABS ERRX-*      EXECUTION JUMP TABLE
1784  23772 000000        NOP             ** CHECKSUM WORD **
1785*
1786  23773 176014        ABS BN777-*     NON-FORMULA OPERATOR TABLE (NULL TABLE)
1787  23774 176013        ABS BN777-*     FUNCTION-NAME TABLE (NULL TABLE)
1788  23775 176004        ABS COMTB-*     COMMAND-NAME TABLE
1789  23776 176010        ABS SYSNT-*     STATEMENT-NAME TABLE
1790  23777 016000 OPCOD  OCT 16000
1791                      END
*  NO ERRORS*

        CROSS-REFERENCE TABLE


.1      0020    0925    0945    0980    1087    1186    1238    1291    1356
.10     0071    0072
.100    0070
.1000   0069
.11     0035    0043    1326
.110    0569    0577
```

CROSS-REFERENCE TABLE

```
.12     0036
.13     0039
.15     0048    0083
.16     0049    0084
.1F4    006B    0576    1070    1123    1187    1239
.2      0027
.21     0050
.22     0051    0087
.23     0052    0088
.28     0053
.3      0028    0045    1024    1059
.31     0054    0090    0723    1201    1647
.32     0055    0776    0866    0915    1235
.33     0056    1440
.34     0057    0091
.37     0058
.4      0029    0573    0636    1042    1347
.40     0059
.41     0060    0958
.43     0061
.45     0062
.46     0063
.48     0064
.5      0032    0564    1268    1618
.58     0065    0094
.6      0031    0044    1270
.63     0066    1333
.7      0033    1272
.74     1663    0713
.8      0033    0600    1274
.9      0034    1276
.97     1664    0725
.99     0074
.BUFA   0183    0794    0829    0965    1372    1474    1533
.FADA   0185
.FDVA   0188    0597
.FMPA   0187
.FSBA   0186
.FSR    0190
.LNUM   0014    0631    0661    0935    1609
.NPWR   0494
.SLRF   0181    0551    0667    0782    0848    0853    1617
A       0000    0612    0756    0803    1150    1165    1304    1378    1484    1507    1534
        1537    1587
ACTSA   0174    0792    0947
ADRSS   0327    1437    1454    1457    1488    1511    1564    1591    1597
AERRA   0176    0581    1754
AIEX1   1326    1314
AIEX2   1327    1320
AR1     0413    0182
AR1A    0182
AR2     0415    0183
AR2A    0183    0586
ARYAD   0403
ASTR1   1324    1309
ASTR2   1325    1317
AT1     1666    0537    0969    1254
AT1S    1667    0536    0970    1255
B       0000    0578    0759    1167    1282    1286    1289    1294    1305    1312    1316
        1321    1406    1491    1514    1544    1594    1604
B100    0005    0735    1055
B1000   0102
B13K    0114    0115
B12K    0116    0117    0206
B130    1223    1140
B133    0096
B140    0097
B1400   0103    0204    1407
B17     0083
B170K   0121    1226
B1740   0104    0203
B1760   0145    0158
B177    0098    0673    1163    1390
B1774   0106
B1775   0107
B1777   0108
B20     0084    1330
B22     0085    0654
B23     0086
B2400   0110
B26     0087
B27     0088
B3900   0111
B36     0089
B37     0090
B377    0099    0652    0671    1306    1336    1528    1632
B400    0100
B4000   0112    0162    1416
B4001   1662    0617    1367    1444    1517
B42     0091
```

CROSS-REFERENCE TABLE

1224	1182								
0092									
0093	1324								
1659	0938	1612							
0113	0167								
0094									
0101	1411								
0291	0553	0633	0663	0668	0771	0835	0847	0868	0884	0886
0999	0914	0943	1184	1373	1375	1376	1385	1477	1480	1481
1493									
0204									
0198	0199	0955							
0444									
0857	0851								
0853	0846								
0849	0863								
1658	0763								
1765	0646	0761	0845	0850	0857	0862	0864	0867	0877	0906
1658									
0845	0655	0696							
0388	0647	1236							
0521	1786	1787							
0015									
0011	0114								
0022	0100								
1764	0645	1303	1308						
0219	0946								
0491									
0160	0563	0628	0658	0746	0828	1016	1034	1143	1343	1473
0316	0625	1090	1112	1121	1146	1189	1196	1241	1248	
0217									
1070	0702								
0436									
0435									
0510	1788								
0484									
0331									
0228									
0963	0928	0983	1359						
0497									
0498	0967	1209							
0037	0038	0039	1388	1463	1554				
1779	0674	0760	1334						
0330									
0210									
0175	0937								
0309									
0463	0931	1565	1598						
0800	0814								
0816	0808								
0790	0653	0672	1337						
0886	0895								
1094	0698								
1112	1097	1195	1247						
0043									
0317									
0303									
0080									
0235									
0244									
0249									
0254									
0259									
0264									
0269									
3274									
0279									
0239									
0284									
0215									
1677	0685								
0169									
0480									
0218									
0189									
0038	0807	0832	0936	0944	0953	1382	1541		
0928	0549								
2483									
0923	0707								
0481									
0177									
1754	0515	1783							
1234	1275								
0401	0402								
0118									
0081									
0222									
0488									
0201									
0367									

CROSS-REFERENCE TABLE

```
FIX     0E01
FLGRT   0073    0540    1344    1401    1451
FLOAT   0386
FLTFA   0193
FLTRA   0192    0598
FLTSA   0195
FN      0148
FND     0462    1131
FND1A   0324
FNDAA   0323
FNDPS   0476    1114    1124
FORRS   0206
FORD    0862    0697
FORMA   0172
FORMF   0336
FORMS   0335
FORMT   0334
FPAGE   0451
FROMA   0122
FSCA    0173
FSCE2   0441
FSCE4   0446
FTMP1   0337
FTMP2   0338
FW2     0865    0765
FWAM    0157    1279
FWUP    0296    1280
FXFLA   0194    0585
GETAD   0461
GETCH   0471
GETLN   0527    1086    1185    1237
GETSK   0475
GFLAG   0404
GFRST   0474    0929    0984
GNEXT   0473    1179
GPARM   0424
GSIGN   0486
GTCON   0500
GTMP    0472    0831    0889    0904
GTOPP   0466
GTSTA   0213
HFLG1   0321
HSTPT   0302
I       0067
ICNT    0422    0924    0979    1207    1355
ID      0508    0534    1027    1027    1029    1029    1045    1045    1047    1047    1065
        1065    1340    1340    1350    1350    1433    1669    1677    1677    1697    1697
        1714    1714    1757    1757
IMPY    0487
INDL    0877    0703
INF     0077
INITS   0456
INREA   0226
INS1    0903    0912
INS2    0914    0887    0907
INSR    0899    0880
INT2    1516    1469    1479    1502    1522    1547    1560    1574    1577    1582    1595
        1606    1636
INTAD   1432    0609
INTCK   0459    0934
INTFA   0178
INTFL   0340
INTFL   0075
INTGR   0402    1567    1600    1602    1605    1608
INTSA   1433    0611
IORUF   0359    1549    1549
IOFND   1549    1501
IOHFA   0229
IOST    1546    1368    1655
IOSTA   0225    0797
KEY1    1280    1287
KEYFG   0324    1529    1633
KEYIN   0526    0651    0670    1335
KEYPR   1267    0715
L2T1    0347    1414    1425
L2T2    0348
LASTN   0398    1130
LBOP    0119
LDEF    0489
LEFF    1759
LEFT    0771    0700
LFTCK   0464
LIMIT   0465
LISTA   0167
LISTS   0528    1155
LNFD    0872    0816
LNINC   0315    0624    0629    0659    1107
LNOUT   1143
LNUM    0457    1214
LOCL1   0345    1293    1300
```

CROSS-REFERENCE TABLE

LOCL2 0346
LOUT 1226 1202
LPCK 0437
LPCKE 043A
LPOP 0120
LROMA 0123
LSTAK 0299
LSTPT 0301
LTR 0479
LWAM 0295
M1 0124 0544 0554 0680 0902 1089 118A 1233 1240 1572 1576
1624
M10 0131 0710 1413
M100 0144
M1024 0146
M11 0132
M12 0133
M13 0134
M14 0135
M15 0136
M159A 0570 0583
M2 0125 1570 1578 1626
M22 1661 0717
M23 0137
M25 0138
M256 0145 1313 1487 1510 1540 1590
M3 0126 0579 0911 1164
M32 0139 0733 0772 0836 1465 1494 1556
M36 0140
M4 0127
M48 0141
M5 0128 1302
M6 0129
M64 0142 0719 1467 1558
M8 1662 1436
M80 0143
M9 0130
MAINS 1410 0932 0989 0995 0996 0997 0999 1000 1001 1002 1003
1004 1005
MASK1 0153
MASK5 0154
MAW 0294
MAXIN 0403
MAXSN 0147
MBIN1 0406
MBIN2 0407
MBOX1 0351
MCOUT 0529 0750
MDIMA 0208
MDR1 1375 1386
MDR2 1415 1426
MDR3 1388 1383
MDRS 1398 1384 1389 1391
MDRVR 1366 1220 1360
MDS1 1572 1566
MDS2 1576 1563
MDS3 1597 1579
MDS4 1608 1599
MDS5 1622 1555
MDS6 1634 1623 1625 1627
MDSS 1554 1461
MDST 1582 1628
MDU1 1443 1455
MDU2 1520
MERR 0445
MEV 1345 1039
MINIT 0455
MOD 1347 1041
MODE 0345 0541 1210
MOE 1329 1423
MOM 1327 1421
MSELG 0286
MSSG1 1342 0437
MTIR 0952 0562 0923 0978 1219 1253 1354
MULTN 1714 0748 0546
MULTP 1697 0741
MVTOH 0477 0959
NOTSP 0383
NOWRT 0496 0966 1238
NEXT1 0452
NEXTA 0447
NEXTI 0449
NMOTA 0214
NOEOF 0482
NPRTN 0381
NPWRT 0495 0639 1062 1174
NUM 0396 0397 589
NUMCA 0211
NUMOA 0212
NXTDT 0214

CROSS-REFERENCE TABLE

[illegible]	[illegible]									
[illegible]	[illegible]									
[illegible]	[illegible]									
[illegible]	[illegible]									
[illegible]	[illegible]	[illegible]767	[illegible]34	0954						
[illegible]	[illegible]									
[illegible]	[illegible]	[illegible]61	1152	1415						
[illegible]	[illegible]									
[illegible]	[illegible]	[illegible]6	1151							
[illegible]	[illegible]									
[illegible]	[illegible]	[illegible]33								
[illegible]	[illegible]	[illegible]76	[illegible]76	[illegible]741	0784	0838	0840	0855	0856	0865	0870
[illegible]	[illegible]	[illegible]1[illegible]9								
[illegible]	[illegible]	[illegible]74	[illegible]93	[illegible]5[illegible]4	0917					
[illegible]	[illegible]	[illegible]43	[illegible]63	1154						
[illegible]	[illegible]	[illegible]40								
[illegible]	[illegible]	[illegible]1[illegible]5	[illegible]6							
[illegible]	[illegible]	[illegible]4[illegible]4								
[illegible]	[illegible]	[illegible]1[illegible]1								
[illegible]	[illegible]	[illegible]1[illegible]0								
[illegible]T	[illegible]	[illegible]1[illegible]0	[illegible]123	1192	1244					
[illegible]6[illegible]	[illegible]11	[illegible]1[illegible]0	1142	1127	1191	1243				
[illegible]4[illegible]1	[illegible]1									
[illegible]1[illegible]5	[illegible]1	[illegible]47	[illegible]52							
[illegible]1[illegible]4	[illegible]1									

PFLAG 3391
PGTN0 3479
PGTNT 3458
PIP2 3344
PLACE 3344
PMODE 3329 3963 1017 1020
PREF 3366
PRMDD 3338
PREED 3525 0547
PRFLG 3332
PRST 1343 1273
QTCHA 3233
QTOP 0109 0161 1079
R10 3638 0566 1025 1043 1348
R13 3732 3718
R15 3709 3688
R17 3641 0557 0834 0798 0821 1108
R26 3477 1478
R36 3673 3665
R46 3674 1311 1319 1322
RADP 0117
RBUFF 3360
RCOUT 3333
RDD 3749 0732
RDYDA 0166 1081
RE0 3623 0736 1193 1234 1262
RE1 3628 3556 3649 0926 0930 0981 1096 1104 1357 1361
RE1A 3534 3644
RE2 3717 3711
RE3 0729 3724 3724
RE5 3753 3727 3738 1457
RE6 3667 3751 777 0743 0785 0833 0849 0854 0858 0869 0871
3876 3941 518 1463
RE9 3767 3743
REC1 3931 3841
RECL 3928 3695
REED 3151
REEDL 3339 3045 1180
RES 0417
RESB1 3453
RESBP 3454
RESM 1365 1344
REST 1353 0705
RET2 3421
RETN2 3420 1011 1258 1656
RETN3 3419
RITE 0791 0741
RPCK 3439
RPCKE 3443
RPOP 3115
RSAVE 0292
RSLTE 3373
RSPTR 3418
RSTAK 3416
RSTKA 3158
RSULT 3426
RUNA 0163 1071
S 3383
SALTA 0207
SAVBP 3453
SAVEA 3325 1523
SAVER 3326 1524
SBADR 3394
SBPTR 0387 0939 0940 0942 1569 1610 1611 1613 1614 1616 1631
SBPUD 0431

CROSS-REFERENCE TABLE

SBUFA	0199	1568									
SCRN	1529	1464									
SEC	0332										
SETAC	1333	1271									
SETAS	1332	1269									
SFLAG	0393										
SGNS	0403										
SHFLG	1778	0676	0754	0758	1331						
SIGN	0399	0400									
SLCOD	0328	0809	0817	1204	1216	1227					
SLEND	0202	0641	0680	1478	1581						
SLWST	0447										
SNFLA	0197										
SPLFA	0196										
SSBPT	0395										
SSYMA	0178										
ST	0377										
START	0163	0487									
STEXP	0443										
STMS	1351	1346									
STOR1	0429										
STORE	0427										
STPFL	0323	0644	1240	1646							
STPOP	0433										
STSPH	0443										
SUMPA	0227										
SYF12	0435										
SYF25	0460										
SYMC1	0433										
SYMC2	0432										
SYMCK	0434										
SYMTP	0208										
SYSNT	0519	1789									
SYTSA	0209										
T1	1767	0575	0607	0618	1075	1203	1366	1415	1443	1516	1666
T2	1769	0631	1410	1423	1439	1452					
T3	1769	0615	1035	1038	1399						
T4	1770	0592	0638	0686	0742	0749	0987	1061			
T5	1771	0594	0603	0610	0626	0790	1345	1459			
T6	1772	1133	1190	1198	1242	1250					
T7	1773	0648	1471	1475	1500	1503	1504	1531	1546	1562	1573
	1582	1583	1584	1619	1622	1628	1635				
T8	1774	0649	1157	1212							
T9	1775	0656	0664	0683	0714	0726	0729	0737	0740	0745	0747
	0753	0757	1056	1135	1144	1145	1148	1160	1161	1162	1171
	1267	1284	1398	1404	1409	1420	1422				
TCON	1669	0585									
TEMP	0352	0351									
TEMP1	0352										
TEMP2	0353										
TEMP3	0354	0956									
TEMP4	0355	0957									
TEMPS	0349	0681	0709	0879	1053	1136	1154	1764			
TERM	0364										
TERR1	0581										
TF1	1777	1668	1778	1779							
TF1A	1668	0597	0596								
TFLAG	0318	0543	1234	1261							
TFLG1	0322										
TMP	1763	0795	0800	0801	0813	0830	0885	0888	0890	0891	0894
	0933	0905	0910	1299							
TMPA	0203	0892	0908	0916							
TMPOP	0184										
TREGF	0373										
TSPHA	0168	0697	0731	0743	0988						
TSTPT	0305										
TTY	0499										
TYPFL	0378										
U3	1133	1115	1125								
U5	1135	1116	1117	1128							
UFLAG	0392										
UKPTR	0200										
UNITS	0319										
UNSTK	0448										
UPARA	0191										
UPR1	1136	1098									
UPRW	1102	0690									
UPRWS	1121	1105									
VALUE	0207										
VALUP	0524	0730									
VAPOA	0214										
WBUFF	0423										
WHAM	1757	0565									
WHAT	0562	0694	0744	1281	1292	1297					
WIDTH	0385	1765									
WRTTF	0159										
XC	0412	1763									
XCOM1	1247	0545									
XCOM2	1268	1245	1251								
XCOMP	1237	1408									

```
        CROSS-REFERENCE TABLE

C4A   1164
C6A   1221
EXIT  1175      1446
AR1   0435
AR2   0429
ECH   1246      1098
TS1   1195      1217
TS2   1195      1214      1221
TS3   1212      1181
TS4   1214      1183
TS5   1219      1225
TST   1179      1094      1213
ATT   1254      1277
SCRA  0220
FRM   0572      0598
      0414
SP    0152
ONK   0408

        #01  EXTENDED I/O BLOCK'S CONSTANTS, SYNTAX, AND LINKAGES


508                     LST
509*    BASE PAGE EQUATES AND TEMPORARIES
510*
511   01641               ORG 16418
512*
513   01641 000000 TS     BSS 1
514   01642 000000 TA     BSS 1
515   01643 000000 TAEND  BSS 1
516   01644 000000 T1     BSS 1
517   01645 000000        BSS 1
518   01645        PTR1   EQU 16458
519   01646 000000 RLFLG  BSS 1
520   01647 000000 TW     BSS 1
521   01647        TMPRE  EQU 16478
522   01650 000000 TX     BSS 1
523   01651 000000 TD     BSS 1
524   01652 000000 MF     BSS 1
525   01653 000000 SD     BSS 1
526   01654 000000 DP     BSS 1
527   01654        TNULL  EQU 16548
528   01655 000000 EX     BSS 1
529   01656 000000 T2     BSS 1
530   01657 000000 T3     BSS 1
531   01660 000000 T6     BSS 1
532   01661 000000 T7     BSS 1
533   01662 000000 T9     BSS 1
534   01663 000000 .FLP   BSS 1
535*
536   01665               ORG 16658
537   01665 000000 FLOOP  BSS 1
538*
539*     LOCAL TEMPORARIES FOR IMLIED "FOR" LOOP IS AS FOLLOWS:
540*
541*     WORD 1 ADDRESS OF VARIABLE
542*     WORD 2 NAME OF VARIABLE
543*     WORD 3 FINAL VALUE (INTEGER FORM)
544*     WORD 4 STEP SIZE      "      "
545*     WORD 5 ADDRESS OF 1ST WORD AFTER FOR STMT.
546*
547   01666 000000 .FLOP  BSS 5         FIRST LOOP BLOCK
548   01673 000000        BSS 5          SECOND LOOP BLOCK
549*
550   01700 000000 .BADR  BSS 1
551   00051        ASC.   EQU .46        .
552   00047        ASCP   EQU .43
553   00050        ASCM   EQU .45
554   01712        .SIGN  EQU 17128
555   00074        .18    EQU 422
557   22000               ORG 22000B
558*
559* FORMATTED I/O BLOCK
560*
561   12001        QUOTE  EQU 12001B
562   12007        ASSOP  EQU QUOTE+6
563*
564*
565   22000 000213 IO     ABS SOUT-*     OUTPUT SYNTAX
566   22001 000117        ABS SGET-*
567   22002 000270        ABS CMDS-*
568   22003 000000        BSS 1
569*
570   22004 041515 SYSNT  ASC 1,CMD
571   22005 042242        OCT 42242      0: OPCODE 2
572   22006 042516        ASC 2,ENTER
      22007 052105
573   22010 051243        OCT 51243      R: OP CODE 3
574   22011 047525        ASC 3,OUTPUT
      22012 052120
```

#01 EXTENDED I/O BLOCK'S CONSTANTS, SYNTAX, AND LINKAGES

```
      22013 052524
#575  22014 162377 NULL  OCT 162777     OP CODE 4; NULL
#576*
#577*
#578  22015 043117 NFOPT OCT 043117       FO      NON-FORMULA OPERATOR TABLE
#579  22016 051220       OCT 051220       R (16)
#580  22017 052117       OCT 052117       TO
#581  22020 110523       OCT 110523         (17) S
#582  22021 052105       OCT 052105       TE
#583  22022 050322       OCT 50322     P (18)
#584*
#585*
#586  22023 000331       ABS EIAND-*
#587  22024 000324       ABS EIOR-*
#588  22025 000313       ABS EROT-*
#589  22026 000441       ABS STAT-*
#590  22027 000452       ABS CHAR-*
#591  22030 000500       ABS BIN-*
#592  22031 000457       ABS LIN-*
#593  22032 000472       ABS SPA-*
#594  22033 000410       ABS EOCT-*
#595*
#596  22034 001661 FLOP  DEF .FLOP-5
#597  22035 150000 .SC13 OCT 150000
#598  22036 177770 M8    DEC -8
#599  22037 154000 SC13  OCT 154000    SELECT CODE 13 FOR ASCII BUSS
#600  22040 001652 AMF   DEF MF
#601  22041 000137 AROW  OCT 137        LEFT ARROW
#602  22042 000176 ALTM  OCT 176        ALT MODE
#603  22043 000033 ESCM  OCT 33         ESC MODE
#604  22044 000105 ASCE  OCT 105        E
#605  22045 031166 B3116 OCT 31166
#606  22046 000120 .80   DEC 80
#607  22047 002001 B2001 OCT 2001
#608  22050 100700 STEC  OCT -77100
#609  22051 012025 FMTOP OCT 12025
#610  22052 000021 .17   DEC 17
#611  22053 007777 BIT12 OCT 7777
#612  22054 000032 NFOPA DEF NFOPT+NFOPT-ID-ID
#613*
#614*
#615  22055 047503 FUNNT ASC 1,OCT
#616  22056 052222       OCT 52222      T;22
#617  22057 051520       ASC 1,SPA
#618  22060 040623       OCT 40623      A;23
#619  22061 046111       ASC 1,LIN
#620  22062 047224       OCT 47224      N;24
#621  22063 041111       ASC 1,BIN
#622  22064 047225       OCT 47225      N;25
#623  22065 041510       ASC 2,CHAR
      22066 040522
#624  22067 113123       OCT 113123     26;STAT
#625  22070 052101       ASC 1,TAT
#626  22071 052227       OCT 52227      T;27
#627  22072 051117       ASC 1,ROT
#628  22073 052230       OCT 52230      T;30
#629  22074 044516       ASC 2,INOR
      22075 047522
#630  22076 114502       OCT 114502     31;BIAND
#631  22077 044501       ASC 2,IAND
      22100 047104
#632  22101 155000       OCT 155000     32; NULL
#633*
#634*
#635*  MORE ABSOLUTE LINKS TO THE BASIC SYSTEM
#636*
#637  22102 004304 RITE  DEF 4304B
#638  22103 007474 MOUTA DEF 7474B
#639  22104 006656 PRINN DEF 6656B
#640  22105 007366 SFMT  DEF 7366B
#641  22106 007303 TYPEC DEF 7303B
#642  22107 007240 QTCHK DEF 7240B
#643  22110 006773 PFORM DEF 6773B
#644  22111 007007 PRCHK DEF 7007B
#645  22112 007227 CKEOL DEF 7227B
#646  22113 011444 OPCH1 DEF 11444B
#647  22114 007356 FSPR  DEF 7356B
#648  22115 012233 VAR01 DEF 12233B    ADDRESS OF VAROP+2
#649  22116 016504 IOHED DEF 16504B
#650  22117 007402 PLOTE  DEF 7402B
#652*
#653* GET STATEMENT SYNTAX
#654*
#655  22120 070742 SGET  SAR 16
#656  22121 031656       STA T2           CLEAR NEST COUNTER
#657  22122 021717       LDA NUM
#658  22123 050065       AND OPMSK
#659  22124 070744       SAL 1
#660  22125 031642       STA T8           SAVE PAGE INFO
#661  22126 062247       JSM STBLK
```

#01 EXTENDED I/O BLOCK'S CONSTANTS, SYNTAX, AND LINKAGES

```
#662*
#663  22127 #62434        JSM IOHD      GET (S.C.,FMT#
#664*
#665  22130 #60513 SGP1   JSM LPCK        LOOK FOR LEFT BRACKET
#666  22131 #66171        JMP SGP4        NOT FOUND
#667  22132 #61055        JSM GNEXT
#668  22133 #45706        ISZ SBPTR
#669  22134 #62422        JSM NFOP        LOOK FOR "FOR"
#670  22135 164307        JMP E15+1,I     NOT FOUND
#671  22136 #10033        CPA .16         FOR?
#672  22137 #66141        JMP *+2         YES
#673  22140 164307        JMP E15+1,I     NO, ERROR
#674  22141 #45656        ISZ T2          INCREMENT NEST COUNTER
#675  22142 160246        JSM VAR0A,I     GO LOOK FOR A VARIABLE
#676  22143 164321        JMP E25+1,I     NONE FOUND, ERROR
#677  22144 164321        JMP E25+1,I     SUBSCRIPT, ERROR
#678  22145 #60463        JSM SYMC1       FOUND ONE, LOOK FOR "="
#679  22146 #12006        DEF ASSOP-1
#680  22147 164324        JMP E25+4,I
#681  22150 160202        JSM FSCA,I      GO LOOK FOR A FORMULA
#682  22151 #62422        JSM NFOP        LOOK FOR "TO"
#683  22152 164322        JMP E25+2,I     NOT FOUND, ERROR
#684  22153 #12052        CPA .17         TO?
#685  22154 #66156        JMP *+2         YES
#686  22155 164322        JMP E25+2,I     NO, ERROR
#687  22156 160202        JSM FSCA,I      GO LOOK FOR A FORMULA
#688  22157 #62422        JSM NFOP        LOOK FOR "STEP"
#689  22160 #66165        JMP SGP2        NOT FOUND
#690  22161 #10074        CPA .18         STEP?
#691  22162 #66164        JMP *+2         YES
#692  22163 164323        JMP E25+3,I     NO, ERROR
#693  22164 160202        JSM FSCA,I      GO LOOK FOR A FORMULA
#694  22165 #60510 SGP2   JSM COMCF       DEMAND A COMMA
#695  22166 #61055 SGP3   JSM GNEXT
#696  22167 #45706        ISZ SBPTR
#697  22170 #66130        JMP SGP1
#698*
#699  22171 #55400 SGP4   DSZ BADDR
#700  22172 #55706        DSZ SBPTR
#701  22173 #62233        JSM SVARA     LOOK FOR A VARIABLE (STRING ALLOWED)
#702  22174 #10031 BACK   CPA EOL         END-OF-STATEMENT?
#703  22175 #66201        JMP NESC        YES
#704  22176 #60504        JSM COMCK       LOOK FOR A COMMA
#705  22177 #66205        JMP SGP5        NOT FOUND
#706  22200 #66166        JMP SGP3
#707*
#708  22201 #21656 NESC   LDA T2          CHECK NEST COUNTER
#709  22202 #72110        SZA *+2
#710  22203 164203        JMP ACTSA,I     STATEMENT IS OK
#711  22204 164301        JMP E10,I
#712*
#713  22205 #60534 SGP5   JSM RPCKE       DEMAND A RIGHT BRACKET
#714  22206 #55656        DSZ T2          DECREMENT NEST COUNTER
#715  22207 #66210        JMP *+1         ALLOW FOR SKIP
#716  22210 #25656        LDB T2
#717  22211 #75153        SBP BACK
#718  22212 164302        JMP E10+1,I
#719*
#720*    OUTPUT STMT SYNTAX
#721*
#722  22213 #62247 SOUT   JSM STBLK
#723  22214 #72150        SZA *+3
#724  22215 #62434        JSM IOHD      STRING BLOCK NOT IN DO STRANDARD SYNTAX
#725  22216 166102        JMP RITE,I
#726  22217 #45706        ISZ SBPTR
#727  22220 #60520        JSM LPCKF-1   STRING BLOCK IN, TRY INTERNAL CONVERSION
#728  22221 161641        JSM T5,I      LOOK FOR STRING VARIABLE
#729  22222 #66230        JMP SOUT2     P+1
#730  22223 #66227        JMP SOUT1     P+2
#731  22224 162116 SOUT3  JSM IOHED,I   P+3, STRING ARRAY NAME FOUND
#732  22225 #62435        JSM IOHD+1
#733  22226 166102        JMP RITE,I
#734*
#735  22227 154110 SOUT1  DSZ BADRP,I
#736  22230 154110 SOUT2  DSZ BADRP,I
#737  22231 160202        JSM FSCA,I    CHECK SELECT CODE
#738  22232 #66224        JMP SOUT3
#739*
#740*    CHECK SYNTAX ON NUMERIC OR STRING VARIABLES
#741*
#742  22233 #25641 SVARA  LDB T5
#743  22234 #76250        RZA RECO2     IS STRING BLOCK IN?
#744  22235 160246        JSM VAR0A,I   NO, ERROR IF ANY STRINGS
#745  22236 164307        JMP E15+1,I   P+1, NO LETTER FOUND
#746  22237 170402        RET           P+2, ARRAY VARIABLE FOUND AND STORED
#747  22240 170402        RET           P+3, SIMPLE    "       "   "    "
#748*
#749  22241 161641 RECO2  JSM T5,I      YES, TRY STRING BLOCK FIRST
#750  22242 164307        JMP E15+1,I   P+1, NO LETTER FOUND
#751  22243 162115        JSM VAR1,I    P+2, MUST BE NUMERIC OR ARRAY VARIABLE
```

#01 EXTENDED I/O BLOCK'S CONSTANTS, SYNTAX, AND LINKAGES

```
#752  22244 170402        RET           P+3, STRING VARIABLE FOUND AND STORED
#753  22245 170402        RET
#754  22246 170402        RET
#755*
#756*    LOOK FOR THE STRING BLOCK ANS SAVE IT'S ADDRESS
#757*
#758  22247 020121 STBLK  LDA B4000
#759  22250 024130        LDB FROMA
#760  22251 074457        CPA B,I
#761  22252 066264        JMP FPAG2
#762  22253 014131        CPB LROMA
#763  22254 066257        JMP FPAG1
#764  22255 004116        ADB QTOP
#765  22256 066251        JMP *-5
#766  22257 025403 FPAG1  LDB MAW
#767  22260 111403        CPA MAW,I
#768  22261 066264        JMP FPAG2
#769  22262 074742        SBR 16
#770  22263 066267        JMP *+4
#771  22264 074442 FPAG2  SBR 10
#772  22265 074304        SBL 10
#773  22266 074070        SIB *+1
#774  22267 035641        STB T5
#775  22270 074117        LDA B
#776  22271 170402        RET
#834  22345 070006        RAR 1
#835  22346 077770        RIB *-1       LOOP , ROTATING (A) ONE BIT UNTIL DONE
#836  22347 066357        JMP ESTO      EXIT LOOP
#837*
#838  22350 062361 EIOR   JSM SCHK
#839  22351 021644        LDA T1        T1 IOR T2
#840  22352 041656        IOR T2
#841  22353 066357        JMP ESTO
#842*
#843  22354 062361 EIAND  JSM SCHK
#844  22355 021644 EAND   LDA T1          T1 AND T2
#845  22356 051656        AND T2
#846  22357 070137 ESTO   LDB A
#847  22360 164227        JMP FXFLA,I
#848*
#849  22361 021416 SCHK   LDA MODE
#850  22362 070552        SAM .SCHK     SKIP IF SYNTAX PHASE
#851*
#852  22363 061217        JSM FETCH
#853  22364 160225        JSM FLTRA,I
#854  22365 024062        LDB FLGRT     IF OVERFLOW USE 100000 OCTAL
#855  22366 035644        STB T1
#856  22367 061217        JSM FETCH
#857  22370 160225        JSM FLTRA,I
#858  22371 164273        JMP E5-1,I
#859  22372 035656        STB T2
#860  22373 045472        ISZ TEMPS
#861  22374 170402        RET
#862*
#863*    CHECK SYNTAX OF FUNCTIONS
#864*
#865  22375 055777 .SCHK  DSZ RSPTR
#866  22376 121472        LDA TEMPS,I
#867  22377 031641        STA ST+1      SAVE LEFT BRACKET COUNT
#868  22400 061055        JSM GNEXT
#869  22401 045706        ISZ SBPTR
#870  22402 060521        JSM LPCKE
#871  22403 160202        JSM FSCA,I
#872  22404 060510        JSM COMCE
#873  22405 160202        JSM FSCA,I
#874  22406 025641        LDB ST+1
#875  22407 135472        STB TEMPS,I
#876  22410 060534        JSM RPCKE
#877  22411 064356        JMP RETN3
#878*
#880*
#881* MISCELLANEOUS ROUTINES & ETC.
#882*
#883* SUBROUTINE TO RECORD CODE-CONVERSION TABLE
#884*
#885  22412 060510 RTBL   JSM COMCE      DEMAND COMMA
#886  22413 061140        JSM LTR       LETTER?
#887  22414 164307        JMP E15+1,I   NO, ERROR
#888  22415 021474        LDA TEMP1     RECORD ARRAY IDENTIFIER
#889  22416 024020        LDB .3        OF UNKNOWN DIMENSIONS
#890  22417 060451        JSM STROP+1
#891  22420 021475        LDA TEMP2     RETRIEVE FOLLOWING CHARACTER
#892  22421 170402        RET
#893*
#894* SUBROUTINE TO FIND A NON-FORMULA OPERATOR
#895*
#896  22422 026054 NFOP   LDB NFOPA     GET NON-FORMULA OPERATOR TABLE ADDRESS
#897  22423 005642        ADB T8
#898  22424 060546        JSM STEXP     DO TABLE SEARCH
#899  22425 170402        RET           NOT FOUND
```

```
          #01  EXTENDED I/O BLOCK'S CONSTANTS, SYNTAX, AND LINKAGES


#900  22426 #60456        JSM SHPUD
#901  22427 #74742        SBR 16
#902  22430 135706        STB SBPTR,I
#903  22431 #70202        SAR 5
#904  22432 #50040        AND .31
#905  22433 #64357        JMP RETN?
#906*
#907* RECORD I/O HEADING
#908*
#909  22434 160265 IOHD   JSM IOHEA,I    GET (S.C.,*
#910  22435 #60524        JSM RPCK       RIGHT PAREN?
#911  22436 #66440        JMP *+2        NO
#912  22437 #65153        JMP EOLCK      YES, IF EOL RETURN TO BASIC SYSTEM
#913  22440 #62412        JSM RTBL       CHECK FOR CONVERSION TABLE
#914  22441 #60534        JSM RPCKE      DEMMAND RIGHT BRACKET
#915  22442 #65153        JMP EOLCK      CHECK FOR EOL AND RETURN IF NOT


          #01  FORMATTED INPUT BLOCK EXECUTION


#917*
#918* EVALUATE OCT EXPRESSION
#919*
#920  22443 #60445 EOCT   JSM XFAR2+1
#921  22444 #74742        SBR 16
#922  22445 #21754        LDA AR2          GET EXPONENT
#923  22446 #70652        SAM *+13          AND TEST IT
#924  22447 #00157 EOC1   ADA M256
#925  22450 #31466        STA LOCL1
#926  22451 #74117        LDA B            CHECK FOR POSSIBLE OVERFLOW
#927  22452 #70602        SAR 13
#928  22453 #72550        RZA **11
#929  22454 171400        MLS              GET NEXT DIGIT
#930  22455 #74606        RBR 13
#931  22456 #70037        ADB A            ADD IN NEW DIGIT
#932  22457 #50025        AND .8           TEST FOR LEGAL OCTAL DIGIT
#933  22460 #72310        RZA **6
#934  22461 #21446        LDA LOCL1
#935  22462 #71253        SAP EOC1
#936*
#937  22463 #70111        SLA *+2          SKIP IF +
#938  22464 #74076        TCB
#939  22465 164227        JMP FXFLA,I      DONE
#940*
#941  22466 164273        JMP ES-1,I       ERROR
#942*
#943  22467 #60445 STAT   JSM XFAR2+1
#944  22470 #63461        JSM UNNM+1
#945  22471 #40121        IOR B4000
#946  22472 172741        STF 1
#947  22473 172141        OTA 1
#948  22474 172741        STF 1
#949  22475 177241        LIB 1,C
#950  22476 #74604        SBL 4
#951  22477 #74542        SBR 12
#952  22500 164227        JMP FXFLA,I
#953*
#954  22501 #60445 CHAR   JSM XFAR2+1
#955  22502 #63461        JSM UNNM+1
#956  22503 #40121        IOR B4000
#957  22504 #63447        JSM .CHAR
#958  22505 #74073        SBP **1,C
#959  22506 #74302        SBR /
#960  22507 164227        JMP FXFLA,I
#961*
#962  22510 #62536 LIN    JSM .LIN         OUTPUT X LINE FEEDS AND CARRIAGE RETURNS
#963  22511 #76150        RZB **3
#964  22512 #62546        JSM .LIN?
#965  22513 #66533        JMP .LIN1         IF ARG =#, GIVE CARRIAGE RETURN
#966  22514 #70112        SAM **2
#967  22515 #62546        JSM .LIN?         IF ARG IS NEGATIVE SKIP CARRIAGE RETURN
#968  22516 #20061        LDA LNFD
#969  22517 162111        JSM PRCHK,I
#970  22520 #61353        JSM TTY
#971  22521 #55644        DSZ T1
#972  22522 #66516        JMP *-4
#973  22523 #66533        JMP .LIN1
#974*
#975  22524 #62536 SPA    JSM .LIN         OUTPUT X SPACES
#976  22525 #70312        SAM .LIN1
#977  22526 162103        JSM MOUTA,I
#978  22527 #66533        JMP .LIN1
#979*
#980  22530 #62536 BIN    JSM .LIN         OUTPUT THE BINARY CODE FOR X
#981  22531 #52053        AND BIT12
#982  22532 #61353        JSM TTY
#983  22533 #20132 .LIN1  LDA M1
#984  22534 #31755        STA AR2+1
#985  22535 170402        RET
```

#01 FORMATTED INPUT BLOCK EXECUTION

```
#986*
#987  22536 #60445 .LIN  JSM XFAR2+1
#988  22537 160226       JSM FLTFA,I
#989  22540 164273       JMP E5-1,I
#990  22541 #74117       LDA B
#991  22542 #74113       SBP *+2
#992  22543 #74076       TCB
#993  22544 #35644       STB T1
#994  22545 170402       RET
#995*
#996  22546 #20031 .LIN2 LDA CRTN
#997  22547 162111       JSM PRCHK,I
#998  22550 #61353       JSM TTY
#999  22551 #70742       SAR 16
1000  22552 #31622       STA TERM
1001  22553 170402       RET
1003*
1004* EXECUTE GET STATEMENT
1005*
1006  22554 #62247 EGET  JSM STBLK     CHECK IF STRING BLOCK IS IN.
1007  22555 #45472       ISZ TEMPS
1008  22556 #63460       JSM UNNM
1009  22557 #40121       IOR B4000
1010  22560 #31644       STA T1
1011  22561 #22051       LDA FMTOP
1012  22562 #31640       STA ST
1013  22563 #74742       SBR 16
1014  22564 162117       JSM PLOTF,I
1015  22565 #74742       SHR 16         P+1, NOT FOUND- FREE FIELD INPUT
1016  22566 #35662       STB T9         P+2, FOUND FORMATTED INPUT
1017*
1018  22567 #63471       JSM .CTBL      CHECK FOR CONVERSION TABLE
1019*
1020  22570 #22034       LDA FLOP
1021  22571 #31665       STA FLOOP      INITIALIZE FOR LOOP POINTER
1022*
1023  22572 #63073       JSM RREC         READ AN INPUT RECORD
1024  22573 #25662       LDB T9
1025  22574 #76510       RZB EGE1
1026*
1027*    FREE FIELD INPUT
1028*
1029  22575 #62723       JSM NADD       GET THE VARIABLE
1030  22576 #61156 .EGE1 JSM CONST      FETCH A NUMBER
1031  22577 #66603       JMP *+4        NO NUMBER FOUND, ZERO IT, AND THE REMAINING NOS.
1032  22600 #60412       JSM RSULT      NO - STORE IT
1033  22601 #66575       JMP .EGE1-1    LOOP
1034  22602 #62723       JSM NADD
1035  22603 #61256       JSM CLAR2      SET THE REMAINING VARIABLES TO ZERO
1036  22604 #60412       JSM RSULT
1037  22605 #66602       JMP *-3        LOOP
1038*
1039*    FORMATTED INPUT
1040*
1041  22606 #21456 EGE1  LDA FORMS
1042  22607 #70073       SAP *+1,C
1043  22610 #31456       STA FORMS
1044  22611 #62723 EGE2  JSM NADD
1045  22612 162106 EGE3  JSM TYPEC,I    GET A FORMAT SPEC.
1046  22613 #66631       JMP ESLH+1     P+1, END OF FORMAT, READ NEXT RECORD AND JUMP TO EGE
1047  22614 #66623       JMP E71        P+2, QUOTE FIELD NOT ALLOWED
1048  22615 #66642       JMP EFMT         3 F
1049  22616 #66642       JMP EFMT         4 E
1050  22617 #66623       JMP E71          5 %
1051  22620 #66633       JMP BFMT         6 B
1052  22621 #66625       JMP ESKP         7 X
1053  22622 #66630       JMP ESLH         8 /
1054  22623 160205 E71   JSM AERRA,I      INVALID FORMAT SPECIFICATION
1055  22624 177654       DEC -84
1056*
1058  22625 162114 ESKP  JSM FSPR,I
1059  22626 #63055       JSM NBYT
1060  22627 #66612       JMP EGE3
1061*
1062  22630 162114 ESLH  JSM FSPR,I
1063  22631 #63073       JSM RREC       READ A NEW RECORD
1064  22632 #66612       JMP EGE3
1065*
1066  22633 162114 BFMT  JSM FSPR,I
1067  22634 #63055       JSM NBYT         GET NEXT BYTE
1068  22635 #100A1       CPA LNFD       IF LINE FEED USE ZERO
1069  22636 #70742       SAR 16
1070  22637 #70137       LDB A
1071  22640 160227       JSM FXFLA,I      CONVERT TO F.P.
1072  22641 #66670       JMP EFM3+1
1073*
1074  22642 #45466 EFMT  ISZ LOCL1
1075  22643 121466       LDA LOCL1,I    GET W SPEC
1076  22644 #45466       ISZ LOCL1
1077  22645 #45466       ISZ LOCL1
```

```
        #01  FORMATTED INPUT BLOCK EXECUTION


1078  22646 070076        TCA
1079  22647 031647        STA TW
1080  22650 125466        LDB LOCL1,I   GET D SPEC
1081  22651 074017        ADA B
1082  22652 070056        CMA
1083  22653 070404        SAL 8
1084  22654 031651        STA TD
1085  22655 162114        JSM FSPR,I
1086  22656 021647        LDA TW          SET FIELD WIDTH
1087  22657 031650        STA TX
1088  22660 070742        SAR 16          CLEAR IGNORE-D FLAG
1089  22661 031662        STA T9
1090  22662 063153        JSM NMBR        CALL FOR REAL NUMBER
1091  22663 021662        LDA T9          ANY . OR E?
1092  22664 072150        RZA EFM3        YES
1093  22665 021651        LDA TD          NO, ADJUST EXPONENT
1094  22666 031754        STA AR2
1095  22667 061254 EFM3   JSM ZONK
1096  22670 060412        JSM RSULT
1097  22671 066611        JMP EGE2
1098*
1100*
1101  22672 050040 .NADD  AND B37       FIND NUMERIC OR STRING VARIABLE
1102  22673 010076        CPA B36       STRING NEXT?
1103  22674 066676        JMP *+2       YES
1104  22675 164201        JMP FORMA,I   NO, GET THE VARIABLE 'S ADDRESS
1105  22676 021641        LDA T5
1106  22677 072110        RZA *+2
1107  22700 164311        JMP E15+3,I   STRINGS NOT AVAILABLE
1108  22701 000012        ADA .1        JSM GADD ON THE STRING BLOCK
1109  22702 070717        JSM A,I
1110  22703 021400        LDA BADDR
1111  22704 031645        STA PTR1      PTR1 = CURRENT WORD OF BUFFER
1112  22705 070076        TCA
1113  22706 001623        ADA PEND
1114  22707 031647        STA TMPRE     TMPRE = REMAINING LENGTH OF THE BUFFER
1115  22710 024132        LDB M1
1116  22711 020017        LDA .2        JSM PSTR (S,B,), PREPARE DESTINATION
1117  22712 001641        ADA T5
1118  22713 070717        JSM A,I
1119  22714 021654        LDA TNULL
1120  22715 070056        CMA
1121  22716 001400        ADA BADDR
1122  22717 031400        STA BADDR
1123  22720 020020        LDA .3        JSM TRSTR(S,B,), TRANSFER DATA
1124  22721 001641        ADA T5
1125  22722 070717        JSM A,I
1126*
1127*     TRANSFER COMPLETE GO GET NEXT VARIABLE
1128*
1129*
1130* SUBROUTINE TO GET AN ADDRESS
1131*
1132  22723 021472 NADD   LDA TEMPS
1133  22724 011421        CPA PRADD       DONE?
1134  22725 067443        JMP EXIT
1135  22726 121472 NAD1   LDA TEMPS,I
1136  22727 010121        CPA B4000       "," ?
1137  22730 066742        JMP PFOR        GO SET UP FOR LOOP
1138  22731 010004        CPA B10K        RIGHT PAREN.?
1139  22732 066734        JMP *+2
1140  22733 066672        JMP .NADD
1141*
1142  22734 021472        LDA TEMPS       IS IT
1143  22735 070070        SIA *+1         FOLLOWED BY
1144  22736 070517        LDA A,I
1145  22737 210126        CPA LPOP        ( LEFT PAREN.?
1146  22740 066742        JMP PFOR        YES , GO SET FOR LOOP
1147  22741 067003        JMP TFOR        NO, MUST BE DURING EXCUTION OF FOR LOOP
1148*
1149  22742 021665 PFOR   LDA FLOOP
1150  22743 000022        ADA .5          BUMP FOR LOOP POINTER
1151  22744 031665        STA FLOOP
1152  22745 021472        LDA TEMPS
1153  22746 000020        ADA .3
1154  22747 031472        STA TEMPS       MOVE TEMPS PASS "FOR"
1155  22750 160201        JSM FORMA,I     EVALUATE STARTING VALUE
1156  22751 021413        LDA HSTPT
1157  22752 000133        ADA M2
1158  22753 025665        LDB FLOOP
1159  22754 170004        XFR             MOVE THE VARIABLE NAME AND ADDRESS TO
1160  22755 004132        ADB M1          THE LOOP BLOCK
1161  22756 035663        STB .FLP        SET .FLP TO FLOOP+2
1162  22757 045472        ISZ TEMPS       "TO"
1163  22760 061217        JSM FETCH       EVALUATE FINAL VALUE
1164  22761 160225        JSM FLTRA,I
1165  22762 164273        JMP E5-1,I
1166  22763 074076        TCB
1167  22764 135663        STB .FLP,I      SAVE "FINAL"
1168  22765 045663        ISZ .FLP
```

#01 FORMATTED INPUT BLOCK EXECUTION

```
1169  22766 #24012        LDB .1
1170  22767 121472        LDA TEMPS,I
1171  22770 #02050        ADA STEC
1172  22771 #72250        RZA **5         ANY STEP ?
1173  22772 #45472        ISZ TEMPS       YES, MOVE PASS "STEP"
1174  22773 #61217        JSM FETCH
1175  22774 160225        JSM FLTRA,I
1176  22775 164273        JMP E5-1,I
1177  22776 135663        STB .FLP,I      SAVE STEP
1178  22777 #45663        ISZ .FLP
1179  23000 #25472        LDB TEMPS
1180  23001 135663        STB .FLP,I      SAVE THE ADDRESS AFTER FOR LOOP IMFORMATION
1181  23002 #66726        JMP NAD1
1182*
1183*
1184  23003 #25665 TFOR   LDB FLOOP
1185  23004 #74070        SIB **1
1186  23005 #61372        JSM FIX+1
1187  23006 #31660        STA T6
1188  23007 #10025        CPA .8          WAS IT AN INTEGER?
1189  23010 #67015        JMP **5         YES
1190  23011 162113        JSM OPCH1,I
1191  23012 #74117        LDA B
1192  23013 #60445        JSM XFAR2+1
1193  23014 160225        JSM FLTRA,I
1194  23015 #74537        LDB B,I         FOR VARIABLE IN (B), NO OVERFLOW
1195  23016 #21665        LDA FLOOP
1196  23017 #00020        ADA .3
1197  23020 #70437        ADB A,I         ADD THE STEP SIZE TO B
1198  23021 #21660        LDA T6
1199  23022 #02036        ADA M8
1200  23023 #72210        RZA **4         IF NOT AN INTEGER SKIP 4
1201  23024 121665        LDA FLOOP,I     YES IT WAS AN INTEGER
1202  23025 #70577        STB A,I
1203  23026 #67036        JMP .TFOR
1204  23027 #35660        STB T6
1205  23030 160227        JSM FXFLA,I
1206  23031 #25665        LDB FLOOP
1207  23032 #74070        SIB **1
1208  23033 #61372        JSM FIX+1
1209  23034 #60416        JSM STORE
1210  23035 #25660        LDB T6
1211  23036 #21665 .TFOR LDA FLOOP
1212  23037 #00017        ADA .2
1213  23040 #70437        ADB A,I         CHECK IF AT FINAL VALUE
1214  23041 #74110        SZB **2
1215  23042 #74313        SBP **6
1216  23043 #21665        LDA FLOOP       NO
1217  23044 #00021        ADA .4
1218  23045 #70517        LDA A,I
1219  23046 #31472        STA TEMPS
1220  23047 #66726        JMP NAD1
1221*
1222  23050 #45472        ISZ TEMPS
1223  23051 #21665        LDA FLOOP
1224  23052 #00136        ADA M5
1225  23053 #31665        STA FLOOP       DONE WITH LOOP BUMP TEMPS, AND LOOP POINTER
1226  23054 #66723        JMP NADD
1227*
1229*
1230* SUBROUTINE #1 TO GET A BYTE
1231*
1232  23055 #61040 NBYT   JSM GETCH-1     GET A CHARACTER
1233  23056 #10061        CPA LNFD
1234  23057 155466        OSZ LOCL1,I
1235  23060 170402        RET
1236*
1237* SUBROUTINE #2 TO GET A BYTE
1238*
1239  23061 #21650 NBYB   LDA TX          CHECK FIELD WIDTH COUNTER
1240  23062 #72170        RIA **3
1241  23063 #20061        LDA LNFD        RETURN LINEFEED IF AT END
1242  23064 170402        RET
1243*
1244  23065 #31650        STA TX
1245  23066 #63055        JSM NBYT        GET CHARACTER
1246  23067 #10041        CPA .32
1247  23070 #67061        JMP NBYB        IGNORE BLANKS
1248  23071 170402        RET
1250  23072 164271 MEOV   JMP E1+1,I      MEMORY OVERFLOW
1251*
1252* SUBROUTINE TO READ ONE ASCII RECORD INTO BUFFER
1253*
1254  23073 #21412 RREC   LDA LSTPT
1255  23074 #00061        ADA .10         ALLOW 10 WORDS IN LOW-STACK FOR FORMT
1256  23075 #70744        SAL 1
1257  23076 #31400        STA BADDR
1258  23077 #31706        STA SBPTR
1259  23100 #02046        ADA .80
1260  23101 #31623        STA PEND
```

```
        #01  FORMATTED INPUT BLOCK EXECUTION


1261  23102 070002        SAR 1
1262  23103 070076        ICA
1263  23104 001413        ADA HSTPT
1264  23105 071252        SAM MEOV
1265*
1266  23106 020141        LDA M10         SET FOR 10 BLANKS
1267  23107 031466        STA LOCLI        THE CARD READER ONLY GIVES 10
1268  23110 063321        JSM INPT
1269  23111 072310        RZA RRE1+1      SKIP IF NOT A BLANK
1270  23112 045466        ISZ LOCLI
1271  23113 067110        JMP *-3
1272*
1273  23114 160205        JSM AERRA,I     ERROR 83 - END OF TAPE
1274  23115 177655        DEC -83
1275*
1276*
1277  23116 063321 RRE1   JSM INPT          CALL FOR INPUT
1278  23117 010061        CPA LNFD          IS IT LINEFEED?
1279  23120 067131        JMP BEND          YES STORE IN BUFFER AND RETURN
1280  23121 012041        CPA ARDW          IS IT LEFT-ARROW?
1281  23122 067142        JMP BKUP          YES, BACK UP
1282  23123 012042        CPA ALTM          IS IT ALT MODE?
1283  23124 067147        JMP SKND          YES, RE-READ RECORD
1284  23125 012043        CPA ESCM          IS IT ESC MODE?
1285  23126 067147        JMP SKND          YES, RE-READ RECORD
1286  23127 061016        JSM ONEXT        NO, STORE IN BUFFER
1287  23130 067116        JMP RRE1
1288*
1289  23131 045623 BEND   ISZ PEND          MOVE END POINTER
1290  23132 020031        LDA EOL
1291  23133 061016        JSM ONEXT
1292  23134 021400        LDA BADDR
1293  23135 000132        ADA M1          SET PEND TO THE EOL CHAR.
1294  23136 031623        STA PEND
1295  23137 021706        LDA SBPTR
1296  23140 031400        STA BADDR
1297  23141 170402        RET
1298*
1299  23142 021400 BKUP   LDA BADDR
1300  23143 011706        CPA SBPTR
1301  23144 067116        JMP RRE1          DON'T GO BEYOND START OF BUFFER
1302  23145 055400        DSZ BADDR        BACK UP
1303  23146 067116        JMP RRE1
1304*
1305  23147 063321 SKND   JSM INPT          SKIP TO END OF RECORD
1306  23150 010061        CPA LNFD
1307  23151 067073        JMP RREC          RE-READ
1308  23152 067147        JMP *-3
1309*
1311*
1312* SUBROUTINE TO BUILD A NUMBER IN AR2
1313*
1314  23153 061256 NMBR   JSM CLAR2         SET AR2 = 0.
1315  23154 022340        LDA AMF
1316  23155 170000        CLR               CLEAR WORK AREA
1317  23156 020132        LDA M1
1318  23157 031660        STA T6            M=-1
1319  23160 020030        LDA .12
1320  23161 031661        STA T7            N=12
1321*
1322  23162 063061        JSM NBYB          GET FIRST CHARACTER
1323  23163 024132        LDB M1
1324  23164 010050        CPA ASCM          IS IT -?
1325  23165 074742        SBR 16            YES
1326  23166 035712        STB .SIGN
1327  23167 010050        CPA ASCM          BYPASS - OR +
1328  23170 067172        JMP *+2
1329  23171 010047        CPA ASCP
1330*
1331  23172 063061 NMB1   JSM NBYR          GET NEXT CHARACTER
1332  23173 060750        JSM DIGCK         IS IT A DIGIT?
1333  23174 067222        JMP NMB3          NO
1334  23175 045652        ISZ MF            YES
1335  23176 076350        RZB NMB2          IS IT ZERO?
1336  23177 021653        LDA SD            YES, ANY SIGNIFICANT DIGITS YET?
1337  23200 072310        RZA NMB2+1        YES
1338  23201 021654        LDA DP            NO, ANY DECIMAL POINT YET?
1339  23202 071410        SZA NMB1          NO, IGNORE THIS ZERO
1340  23203 055660        DSZ T6            YES, M=M-1
1341  23204 067172        JMP NMB1
1342*
1343  23205 045653 NMB2   ISZ SD            SIGNIFICANT DIGIT (1-9)
1344  23206 021654        LDA DP
1345  23207 072150        RZA *+3           IS DP=0?
1346  23210 045660        ISZ T6            YES, M=M+1
1347  23211 067212        JMP *+1           (NOP)
1348  23212 021661        LDA T7            IS N=0?
1349  23213 070310        SZA *+6           YES, IGNORE REMAINING DIGITS
1350  23214 055661        DSZ T7            NO, N=N-1
1351  23215 067216        JMP *+1           (NOP)
```

#01 FORMATTED INPUT BLOCK EXECUTION

```
1352  23216 171400        MLS
1353  23217 035757        ADB AR2+1       INCLUDE DIGIT IN AR2
1354  23220 035757        STB AR2+1
1355  23221 067172        JMP NMB1
1356*
1357  23222 010051 NMB3   CPA ASC.        DECIMAL POINT?
1358  23223 067251        JMP NMB5        YES
1359  23224 012044        CPA ASCE        EEX?
1360  23225 067256        JMP NMB6        YES
1361  23226 021652        LDA MF          NO. IS MF=0?
1362  23227 072150        RZA *+3       IF MF IS ZERO,AND THERE WAS A (-) SIGN - ERROR
1363  23230 025712        LDB .SIGN
1364  23231 074710        SZB E73
1365  23232 021660        LDA T6          NO. TEST FOR EXPONENT OVERFLOW
1366  23233 070137 NMB4   LDB A
1367  23234 074113        SBP *+2
1368  23235 074076        TCB
1369  23236 004156        ADB M100
1370  23237 074413        SBP E73         OVERFLOW. ERROR
1371  23240 171450        NRM
1372  23241 070254        SES *+5         SKIP IF ZERO MANTISSA
1373  23242 070404        SAL 8
1374  23243 045712        ISZ .SIGN       TEST MANTISSA SIGN (-1 = +)
1375  23244 070070        SIA *+1
1376  23245 031754        STA AR2
1377  23246 170402        RET
1378*
1379  23247 160205 E73    JSM AERRA,I     NUMBER SYNTAX ERROR
1380  23250 177652        DEC -86
1381  23251 031662 NMB5   STA T9          SET IGNORE-D FLAG
1382  23252 021654        LDA DP          TEST DP FLAG
1383  23253 073610        RZA E73         SET ALREADY. ERROR
1384  23254 045654        ISZ DP
1385  23255 067172        JMP NMB1
1386*
1387  23256 031662 NMB6   STA T9          SET IGNORE-D FLAG
1388  23257 021652        LDA MF
1389  23260 072250        RZA *+5         MANTISSA DIGITS ENTERED
1390  23261 021654        LDA DP          ANY DECIMAL POINT?
1391  23262 073250        RZA E73         YES. BUT NO MANTISSA DIGITS. ERROR
1392  23263 020020        LDA ONE         NO. SET AR2 = 1.
1393  23264 060445        JSM XFAR2+1
1394  23265 063061        JSM NBYB        GET NEXT CHARACTER
1395  23266 024132        LDB M1
1396  23267 010050        CPA ASCM        IS IT -?
1397  23270 074742        SBR 16          YES
1398  23271 035654        STB DP          USE DP FOR EXPONENT SIGN
1399  23272 010050        CPA ASCM        BYPASS - OR +
1400  23273 067275        JMP *+2
1401  23274 010047        CPA ASCP
1402  23275 063061        JSM NBYB        GET NEXT CHARACTER
1403  23276 060750        JSM DIGCK       DIGIT?
1404  23277 067247        JMP E73         NO. ERROR
1405  23300 035655 NMB7   STB EX          YES
1406  23301 063061        JSM NBYB        GET NEXT CHARACTER
1407  23302 060750        JSM DIGCK       DIGIT?
1408  23303 067313        JMP NMBB        NO
1409  23304 021655        LDA EX          YES
1410  23305 070744        SAL 1
1411  23306 031466        STA LOCL1
1412  23307 070704        SAL 2
1413  23310 001466        ADA LOCL1
1414  23311 070037        ADB A
1415  23312 067300        JMP NMB7
1416*
1417  23313 021655 NMBB   LDA EX
1418  23314 025654        LDB DP
1419  23315 076110        RZB *+2         ADJUST EXPONENT SIGN
1420  23316 070076        TCA
1421  23317 001660        ADA T6
1422  23320 067233        JMP NMB4
1424*
1425* INPUT DRIVER
1426*
1427* T1 HAS SELECT CODE
1428*
1429  23321 021644 INPT   LDA T1
1430  23322 063447        JSM .CHAR     GET A CHAR IN B
1431  23323 074117        LDA B
1432  23324 070073        SAP *+1,C
1433  23325 170302        SAR 7           MASK DATA
1434  23326 025642        LDB TB          CONVERSION NEEDED?
1435  23327 070510        SZA RET1      IF(A) = 0. DON'T CONVERT
1436  23330 074310        SZB *+6       IF(B) IS 0. SKIP CONVERSION
1437  23331 000132        ADA M1
1438  23332 070037        ADB A           YES. TREAT INPUT CODE AS ADDRESS
1439  23333 074517        LDA B,I
1440  23334 070113        SAP *+2
1441  23335 067505        JMP E79
1442  23336 050104        AND B177
```

#01 FORMATTED INPUT BLOCK EXECUTION

```
1443  23337 #10031        CPA CRTN
1444  23340 #67321        JMP INPT        IGNORE CARRIAGE RETURNS
1445  23341 170402 RET1   RET             RETURN DATA IN A-REGISTER
1446*
1447* OUTPUT DRIVER
1448*
1449* T2 HAS SELECT CODE
1450*
1451* LDA DATA
1452* JSM OTPT
1453*
1454  23342 #31466 OTPT   STA LOCL1
1455  23343 #21656        LDA T2
1456  23344 #63431        JSM IOST        CHECK I/O STATUS
1457  23345 #41466        IOR LOCL1     GET DATA
1458  23346 #65356        JMP TTY+3      STF 1, OTA 1, STC 1, AND RETURN
1459*
1460  23347 #00143 LINEF  DEF .10+.10+1
1461  23350 #00063 CARRF  DEF CRTN+CRTN+1
1462*
1463  23351 #27350 .CRLF  LDB CARRF       CARRIAGE RETURN
1464  23352 #63354        JSM **2
1465  23353 #27347        LDB LINEF       LINE FEED
1466  23354 #20012        LDA .1
1467  23355 #67374        JMP .OUTP
1468*
1469  23356 #25642 .OUT.  LDB T8
1470  23357 #74510        SZB OUT1
1471  23360 #74457        CPA B+1
1472  23361 #67365        JMP **4
1473  23362 #15643        CPB IBEND
1474  23363 170402        RET             IF NO MATCH FOUND - ASSUME A NULL CHARACTER
1475  23364 #77630        MIB *-4
1476  23365 #21642        LDA T8
1477  23366 #70076        TCA
1478  23367 #74417        ADA B
1479  23370 #70070        STA **1
1480  23371 #65353 OUT1   JMP TTY
1481*
1482  23372 #25706        LDB SHPTR
1483  23373 #60374        JSM MBUFF
1484  23374 #31741 .OUTP  STA XC+1
1485  23375 #35740        STB XC
1486  23376 #25641        LDB T5
1487  23377 #76350        MZB XFTO%       ANY INTERNAL CONVERSION
1488  23400 #70250        SZA **5
1489  23401 #61053        JSM GTMP
1490  23402 #63356 .OUTT  JSM .OUT.       NO - OUTPUT THE DATA TO THE DEVICE
1491  23403 #55741        DSZ XC+1
1492  23404 #67401        JMP *-3
1493  23405 170402        RET
1494*
1495  23406 #31647 XFTO%  STA TMPRF
1496  23407 #25740        LDB XC
1497  23410 #35645        STB PTR1        PLACE THE COUNT AND ADDRESS IN TMPRE, PTR1
1498  23411 #22034        LDA FLOP
1499  23412 #25412        LDB LSTPT       TRANSFER THE SUBSCRIPTS TO THE LSTPT
1500  23413 #74070        STB **1
1501  23414 170004        XFR
1502  23415 #04132        ADB M1
1503  23416 #35412        STB LSTPT       BUMP THE LOW STACK POINTER
1504  23417 #21647        LDA TMPRF
1505  23420 #01662        ADA T9
1506  23421 #31662        STA T9
1507  23422 #21641        LDA T5
1508  23423 #00017        ADA .2           JSM PSTR
1509  23424 #24132        LDB M1
1510  23425 #70717        JSM A+I         PREPARE THE DESTINATION
1511  23426 #21641        LDA T5
1512  23427 #00020        ADA .3           JSM TRSTR, TRANSFER DATA
1513  23430 #70737        JMP A+I
1515*
1516* CHECK I/O STATUS
1517*
1518* LDA SELECT CODE (SAVED)
1519* JSM IOST
1520*
1521  23431 #25442 IOST   LDB STPFL         CHECK STOP FLAG
1522  23432 #14040        CPB .31
1523  23433 #67443        JMP EXIT
1524  23434 172741        STF 1             TURN OFF INTERRUPT SYSTEM
1525  23435 172141        OTA 1             OUTPUT SELECT CODE
1526  23436 172741        STF 1             LOAD DEVICE STATUS
1527  23437 177241        LIB 1+C            AND TURN ON INTERRUPT
1528  23440 #74406        MBR 9
1529  23441 #75413        SMP IOST          LOOP IF BUSY
1530  23442 170402        RET
1531*
1532* EXIT ROUTINE
1533*
```

```
          #01  FORMATTED INPUT BLOCK EXECUTION


1534  23443 070742 EXIT  SAR 16
1535  23444 031455       STA FORMT
1536  23445 160170       JSM CLERA,I
1537  23446 164172       JMP XEC4A,I
1538*
1539  23447 012037 .CHAR CPA SC13       SELECT CODE 13 ?
1540  23450 067452       JMP *+2        YES, SKIP
1541  23451 063456       JSM CEO-1      NO, SET CONTROL FIRST
1542  23452 063431       JSM IOST       GET A BYTE
1543  23453 012037       CPA SC13       SELECT CODE 13 ?
1544  23454 065356       JMP TTY+3      YES, SET CONTROL AFTER
1545  23455 170402       RET
1546*
1547  23456 063431       JSM IOST
1548  23457 065356 CFO   JMP TTY+3      SET CONTROL
1550*
1551* SAVE STACK POINTERS
1552* EVALUATE AND CHECK UNIT #
1553*
1554  23460 061217 UNNM  JSM FETCH
1555  23461 160225       JSM FLTRA,I
1556  23462 164342       JMP E40+3,I
1557  23463 074117       LDA B
1558  23464 075710       SZB *-2          ERROR, ZERO SELECT CODE
1559  23465 074142       SBR 4
1560  23466 077610       RZB *-4          ERROR, SELECT CODE >15 OR <0
1561  23467 070146       RAR 4
1562  23470 170402       RET
1563*
1564* GET OPTIONAL CONVERSION TABLE
1565*
1566  23471 121472 .CTBL LDA TEMPS,I
1567  23472 050065       AND UPMSK
1568  23473 010004       CPA RPOP       ANY CONVERSION TABLE
1569  23474 067514       JMP CTB1       NO, STORE ZERO
1570  23475 121472       LDA TEMPS,I      YES, GET ARRAY NAME
1571  23476 050115       AND B1777
1572  23477 160177       JSM SSYMA,I      SEARCH TABLE
1573  23500 067505       JMP E79          ERROR, NOT FOUND
1574  23501 074517       LDA B,I          GET TYPE INFORMATION
1575  23502 050037       AND .28
1576  23503 010025       CPA .5           INTEGER?
1577  23504 067507       JMP *+3
1578*
1579  23505 160205 E79   JSM AERRA,I      NO SUCH TABLE OR UNSUITABLE TABLE
1580  23506 177651       DEC -87
1581*
1582  23507 004132       ADB M1
1583  23510 074517       LDA B,I          GET BASE ADDRESS OF TABLE
1584  23511 063515       JSM CTB1+1     GO COMPUTE LAST ADDRESS
1585  23512 045472       ISZ TEMPS
1586  23513 170402       RET
1587*
1588  23514 070742 CTB1  SAR 16
1589  23515 031642       STA T8
1590  23516 071650       SZA CTB1-1     RETURN IF NOT CONVERSION
1591  23517 000132       ADA M1
1592  23520 070517       LDA A,I
1593  23521 070137       LDB A          GET ROW AND COLUMN IMFORMATION
1594  23522 074342       SBR 8
1595  23523 074070       SIB *+1        (B) = # OF ROW
1596  23524 050105       AND B377
1597  23525 070070       SIA *+1        (A) = # OF COLUMNS
1598  23526 061201       JSM IMPY
1599  23527 001642       ADA T8
1600  23530 031643       STA TBEND      SAVE THE END OF TABLE ADDRESS
1601  23531 170402       RET
1603*
1604*    PREPARE FOR INTERNAL CONVERSION
1605*
1606  23532 072110 INTCV SZA *+2
1607  23533 164311       JMP E15+3,I    STRING BLOCK NOT AVAILABE
1608  23534 000012       ADA .1
1609  23535 070717       JSM A,I          JSM GADD  * PROCESS POINTERS TO DESTINATIONS
1610  23536 021412       LDA LSTPT      GADD SETS UP STRING POINTERS ON LOW STACK.
1611  23537 000133       ADA M2         THEY MUST BE SAVED.
1612  23540 026034       LDB FLOP
1613  23541 170004       XFR
1614  23542 000135       ADA M4
1615  23543 031412       STA LSTPT
1616  23544 022051       LDA FMTOP
1617  23545 031640       STA ST
1618  23546 074742       SBR 16
1619  23547 162117       JSM PLOTE,I    LOOK FOR A FORMAT STATEMENT
1620  23550 067722       JMP EWRT1-1    P+1, NOT FOUND, USE PRINT EXECUTION
1621  23551 067567       JMP .EOUT+1    P+2, FOUND
1622*
1623*      OUTPUT STATEMENT EXECUTION
1624*
1625  23552 062247 EOUT  JSM STALK      LOOK FOR THE STRING BLOCK
```

```
            #21  FORMATTED INPUT BLOCK EXECUTION


1626  23553 045472          ISZ TEMPS
1627  23554 125472          LDB TEMPS,I
1628  23555 074244          SBL 11
1629  23556 074542          SHR 12
1630  23557 014032          CPB .15         STRING ARRAY ROOM?
1631  23560 067532          JMP INTCV       YES
1632  23561 055472          DSZ TEMPS       NO
1633  23562 074742          SBR 16
1634  23563 035641          STB I5
1635  23564 022051          LDA FMTOP
1636  23565 162105          JSM SFMT,I      SET UP DEVICE AND FORMAT CODES
1637  23566 067722 .FOUT    JMP EWRI1-1     NO FORMAT SPECIFIED
1638  23567 063471          JSM .CTBL       CHECK CONVERSION TABLE
1639  23570 162112          JSM CKEOL,I     NULL LIST ?
1640  23571 000163 BLNKA    DEF .32+.32+1   IMPOSSIBLE STATE - WORD USED AS A CONSTANT
1641  23572 035472          STB TEMPS       YES, SET TEMPS=PRADD
1642  23573 162106 WRIT2    JSM TYPEC,I     GET NEXT FORMAT SPEC
1643  23574 067606          JMP WRITN       END OF FORMAT
1644  23575 067615          JMP WRITQ       QUOTE FIELD
1645  23576 074742          SBR 16          SET FLAG FOR FIXED
1646  23577 060563          JSM SLWST       STACK FLAG: 0=F, +=E, -=B
1647  23600 067617          JMP WRITF       (CODE 5 IS NOT USED)
1648  23601 077713          SBP *-2,S       SET B FLAG
1649  23602 067612          JMP WRITX       X:
1650*  END OF FORMAT STATEMENT
1651  23603 162114          JSM FSPR,I      /: UPDATE FORMAT POINTER
1652  23604 063351 WRITS    JSM .CRLF
1653  23605 067573          JMP WRIT2
1654  23606 075713 WRITN    SBP WRITS       ANY NUMERIC SPEC FOUND?
1655  23607 074742          SBR 16
1656  23610 035455          STB FORMT       CLEAR FORMAT NESTING FLAG
1657  23611 067723          JMP EWRI1       NO, GOTO STANDARD FORMAT
1658*  X FIELD
1659  23612 162114 WRITX    JSM FSPR,I      UPDATE FORMAT POINTER
1660  23613 020012          LDA .1
1661  23614 027571          LDB BLNKA
1662*  QUOTE FIELD
1663  23615 063374 WRITQ    JSM .OUTP       OUTPUT TO DEVICE
1664  23616 067573          JMP WRIT2
1665*
1666*  OUTPUT E,E, OR B FIELD
1667  23617 025466 WRITF    LDB LOCL1
1668  23620 060563          JSM SLWST       STACK FORMAT STMT POINTER
1669  23621 162112 WRIT4    JSM CKEOL,I     END OF WRITE STMT?
1670  23622 074742          SBR 16          YES, CRLF NOT SUPRESSED
1671  23623 067715          JMP WRITB-2     YES
1672  23624 162107          JSM QTCHK,I     QUOTE NEXT IN WRITE
1673  23625 067630          JMP WRITS       NO
1674  23626 063374          JSM .OUTP       YES, OUTPUT IT
1675  23627 067621          JMP WRIT4
1676*
1677  23630 162110 WRITS    JSM PFORM,I     SAVE FLAGS AND
1678  23631 061222          JSM FETCH+1       EVALUATE FORMULA
1679  23632 060567          JSM UNSTK       RECALL FORMAT STMT POINTER
1680  23633 031466          STA LOCL1
1681  23634 060567          JSM UNSTK       RECALL E,F, OR B FLAG
1682  23635 031705          STA FLOAT
1683  23636 070252          SAM WRITB       SKIP IF B
1684  23637 045466          ISZ LOCL1
1685  23640 121466          LDA LOCL1,I     GET FIELD WIDTH SPEC
1686  23641 045466          ISZ LOCL1
1687  23642 045466          ISZ LOCL1
1688  23643 025456 WRITB    LDB FORMS
1689  23644 074073          SBP *+1,C
1690  23645 035456          STB FORMS       SET NUMERIC SPEC FOUND FLAG
1691  23646 125466          LDB LOCL1,I     GET DECIMAL SPEC
1692  23647 162114          JSM FSPR,I      UPDATE FORMAT POINTERS
1693  23650 031724          STA WIDTH
1694  23651 070353          SAP WRIT7       SKIP IF E OR F
1695*  OUTPUT B FIELD CHARACTER
1696  23652 160225          JSM FLTRA,I     CONVERT TO ROUNDED INTEGER
1697  23653 074742          SBR 16          OVERFLOW: USE 0
1698  23654 074117          LDA B
1699  23655 052053          AND BIT12       MASKTHE LOWER 12 BITS
1700  23656 061353          JSM TTY         OUTPUT SINGLE CHAR
1701  23657 067711          JMP WRIT6
1702*  OUTPUT E OR F NUMBER
1703  23660 035703 WRIT7    STB PLACE
1704  23661 021412          LDA LSTPT       START TEMP BUFFER
1705  23662 000061          ADA .10
1706  23663 070070          STA *+1
1707  23664 070744          SAL 1
1708  23665 031400          STA BADDR         ABOVE LOW STACK
1709  23666 031706          STA SBPTR
1710  23667 021413          LDA HSTPT
1711  23670 070744          SAL 1
1712  23671 031623          STA PEND        SET END OF BUFFER AT H-STACK
1713  23672 021416          LDA MODF
1714  23673 031641          STA SI+1
```

```
        #01  FORMATTED INPUT BLOCK EXECUTION


1715  23674 070742        SAR 16
1716  23675 031416        STA MODE       SET FOR PROGRAM FORMAT
1717  23676 024215        LDB TMPOP
1718  23677 060406        JSM XFAR1+1
1719  23700 140234        JSM NMOTA,I    OUTPUT A NUMBER TO THE BUFFER
1720  23701 021641        LDA ST+1
1721  23702 031416        STA MODE
1722  23703 021400        LDA BADDR
1723  23704 011623        CPA PEND       OVERFLOW?
1724  23705 064561        JMP MERR
1725  23706 063372        JSM .OUTP-2    COMPUTE LENGTH AND OUTPUT  THE BUFFER
1726  23707 025625        LDB FIRST
1727  23710 035400        STB BADDR      RESET BADDR
1728  23711 162112 WRIT6  JSM CKEOL,I     END OF STMT?
1729  23712 074742        SBR 16         YES, CRLF NOT SUPRESSED
1730  23713 067717        JMP WRIT8      YES
1731  23714 067573        JMP WRIT2      NO
1732*   END OF WRITE STMT
1733  23715 055412        DSZ LSTPT
1734  23716 055412        DSZ LSTPT      UNSTACK FLAGS
1735  23717 070742 WRIT8  SAR 16
1736  23720 031455        STA FORMT      SET FORMAT COMPLETE FLAG
1737  23721 067747        JMP PR2+1
1738*
1739  23722 063471        JSM .CTRL
1740  23723 025416 EWRI1  LDB MODE
1741  23724 076150        RZB **3
1742  23725 021622        LDA TERM
1743  23726 072150        RZA **3
1744  23727 160170        JSM CLERA,I
1745  23730 162112        JSM CKEOL,I
1746  23731 067745        JMP PR2-1
1747  23732 067745        JMP PR2-1
1748  23733 162104 PR3    JSM PRINN,I
1749  23734 067746        JMP PR2
1750  23735 021625        LDA FIRST
1751  23736 070076        TCA
1752  23737 001400        ADA BADDR
1753  23740 025625        LDB FIRST
1754  23741 063374        JSM .OUTP
1755  23742 021400        LDA BADDR
1756  23743 031625        STA FIRST
1757  23744 067733        JMP PR3
1758*
1759  23745 074742        SBR 16
1760  23746 035622 PR2    STB TERM
1761  23747 076150        RZB PR17
1762  23750 162111        JSM PRCHK,I
1763  23751 063351        JSM .CRLF
1764  23752 164172 PR17   JMP XEC4A,I
1765*
1766*
1767*
1768  23767               ORG 23767B
1769  23767 177563        ABS EOUT-*
1770  23770 176564        ABS EGET-*       GET EXECUTION
1771  23771 176314        ABS CMD-*      ASCII BUSS EXECUTION
1772  23772 000000        BSS 1            ** CHECKSUM WORD  ******
1773*
1774  23773 176022        ABS NFOPT-*      NON-FORMULA OPERATOR TABLE
1775  23774 176061        ABS FUNNT-*      FUNCTION-NAME TABLE
1776  23775 176017        ABS NULL-*     COMMAND - NAME TABLE (NULL)
1777  23776 176006        ABS SYSNT-*      STATEMENT-NAME TABLE
1778  23777 026000 PAGE   OCT 26000
1779*
1780                      END
*   NO ERRORS*


              CROSS-REFERENCE TABLE


.1      0020    1108    1169    1466    1608    1660
.10     0071    0072    1255    1460    1460    1705
.100    0070
.1000   0069
.11     0035    0043
.12     0036    1319
.13     0039
.15     0048    0083    1630
.16     0049    0084    0671
.17     0610    0684
.18     0555    0690
.1E4    0068
.2      0027    1116    1212    1508
.21     0050
.22     0051    0087
.23     0052    0088
.28     0053    1575
.3      0028    0045    0889    1123    1153    1196    1512
.31     0054    0090    0904    1522
.32     0055    1246    1640    1640
```

CROSS-REFERENCE TABLE

```
.33     0056
.34     0057    0091
.37     0058
.4      0029    1217
.40     0059
.41     0060
.43     0061    0552
.45     0062    0553
.46     0063    0551
.48     0064
.5      0030    1150
.58     0065    0094
.6      0031    0044
.63     0066
.7      0032
.8      0033    0932    1188    1576
.80     0606    1259
.9      0034
.99     0074
.BADR   0550
.BUFA   0180
.CHAR   1539    0957    1430
.CRLF   1463    1652    1763
.CTRL   1566    1018    1638    1739
.EGET   1030    1033
.EOUT   1637    1621
.FADA   0185
.FDVA   0188
.FLOP   0547    0596
.FLP    0534    1161    1167    1168    1177    1178    1180
.FMPA   0187
.FSBA   0186
.FSR    0190
.LIN    0987    0962    0975    0980
.LIN1   0983    0965    0973    0976    097A
.LIN2   0996    0964    0967
.LNUM   0314
.NADD   1101    1140
.NPWR   049A
.OUT.   1469    1490
.OUTP   1484    0803    0811    1467    1663    1674    1725    1754
.OUTT   1490
.SC13   0597    0797
.SCHK   0A65    0850
.SIGN   0554    1326    1363    1374
.SLBF   0181
.TFOR   1211    1203
A       0000    0A46    0931    107A    1109    1118    1125    1144    1197    1202    1213
        1218    1366    1414    1438    1510    1513    1592    1593    1609
ACTSA   0174    0710
ADRSS   0327
AERRA   0176    1054    1273    1379    1579
ALTM    0602    1282
AMF     0600    1315
AR1     0413    0182
AR1A    0182
AR2     0415    0183    0922    0984    1094    1353    1354    1376
AR2A    0183
AROW    0601    1280
ARYAD   0403
ASC.    0551    1357
ASCE    0604    1359
ASCM    0553    1324    1327    1396    1399
ASCP    0552    1329    1401
ASSOP   0562    0679
B       0000    0760    0775    0821    0926    0990    1081    1191    1194    1431    1439
        1471    1478    1557    1574    1583    1698
B100    0095
B1300   0102    0804
B10K    0114    0115    1138
B12K    0116    0117    0206
B133    0096
B140    0097
B1400   0103    0204
B17     00A3
B170K   0121
B1740   0104    0203
B1760   0105    0158
B177    009A    1442
B1774   0106
B1775   0107
B1777   010A    1571
B20     00A4
B2001   0607
B22     00A5    0555
B23     0086
B2400   0110
B26     0087
B27     00AA
B3000   0111
B3116   0605
```

CROSS-REFERENCE TABLE

B36	0089	1102									
B37	0090	1101									
B377	0099	1596									
B400	0100	0799									
B4000	0112	0162	0758	0945	0956	1009	1136				
B42	0091										
B44	0092										
B52	0093										
B6K	0113	0167									
B72	0094										
B777	0101										
BACK	0702	0717									
BADDR	0291	0699	1110	1121	1122	1257	1292	1296	1299	1302	1708
	1722	1727	1752	1755							
BADRP	0204	0735	0736								
BEND	0198	0199									
BFMT	1066	1051									
BFND	1289	1279									
BHSTP	0444										
BIN	0980	0591									
BIT12	0611	0981	1699								
BKUP	1299	1281									
BLANK	0388										
BLNKA	1640	1661									
C5	0015										
C6	0011	0114									
C7	0022	0100									
CARRF	1461	1463									
CEO	1548	1541									
CHAR	0954	0590									
CHRSA	0219										
CKFOL	0645	1639	1669	1728	1745						
CLAR2	0492	1035	1314								
CLERA	0160	1536	1744								
CLINE	0316										
CLRIA	0217										
CMD	0796	0815	1771								
CMDS	0781	0567	0791								
COMCE	0436	0694	0872	0885							
COMCK	0435	0704	0789								
CONST	0484	1030									
CPLAC	0331										
CRCKA	0228										
CRLF	0501										
CRLF1	0502										
CRTN	0037	0038	0039	0996	1443	1461	1461				
CTRL	1588	1569	1584	1590							
CTYPE	0330										
DCOMA	0210										
DELSA	0175										
DFLAG	0390										
DIGCK	0463	1332	1403	1407							
DP	0526	1338	1344	1382	1384	1390	1398	1418			
DPERR	0043										
DROST	0317										
DSTRT	0303										
E	0080										
E1	0235	1250									
E10	0244	0711	0718								
E15	0249	0670	0673	0745	0750	0887	1107	1607			
E20	0254										
E25	0259	0676	0677	0680	0683	0686	0692				
E30	0264										
E35	0269										
E40	0274	1556									
E45	0279										
E5	0239	0819	0958	0941	0989	1165	1176				
E50	0284										
E71	1054	1047	1050								
E73	1379	1364	1370	1383	1391	1404					
E79	1579	1441	1573								
EAND	0844										
ECALA	0215										
EFM3	1095	1072	1092								
EFMT	1074	1048	1049								
EGE1	1041	1025									
EGE2	1044	1097									
EGE3	1045	1060	1064								
EGET	1006	1770									
EIAND	0843	0586									
EIOR	0839	0587									
ELINA	0169										
ENDS	0480										
ENTEA	0218										
ENTFA	0189										
EOC1	0924	0935									
EOCT	0920	0594									
EOL	0038	0702	1290								
EOLCK	0483	0912	0915								
EOST	0481	0790									

CROSS-REFERENCE TABLE

```
EOUT    1625    1769
EROT    0829    0589
ERRA    0177
ESCM    0603    1284
ESKP    1058    1052
ESLH    1062    1046    1053
ESTO    0846    0836    0841
EWRI1   1740    1620    1637    1657
EX      052A    1405    1409    1417
EXP     0401    0402
EXPOP   0118
F       0081
FDAAA   0222
FETCH   0488    0852    0856    1163    1174    1554    1678
FILLA   0201
FIRST   0367    1726    1750    1753    1756
FIX     0505    1186    1208
FLGRT   0073    0854
FLOAT   0386    1682
FLOOP   0537    1021    1149    1151    1158    1184    1195    1201    1206    1211    1216
        1223    1225
FLOP    0596    1020    1498    1612
FLTEA   0193    0988
FLTRA   0192    0853    0857    1164    1175    1193    1555    1696
FLTSA   0195
FMTOP   0609    1011    1616    1635
FN      0148
FND     0462
FNDIA   0224
FNDAA   0223
FNDPS   0476
FOPRS   0206
FORMA   0172    1104    1155
FORME   0336
FORMS   0335    1041    1043    1688    1690
FORMT   0334    1535    1656    1736
FPAG1   0766    0763
FPAG2   0771    0761    0768
FPAGE   0451
FROMA   0122    0759
FSCA    0173    0681    0687    0693    0737    0871    0873
FSCE2   0441
FSCE4   0446
FSPR    0647    1058    1062    1066    1085    1651    1659    1692
FTMP1   0337
FTMP2   0338
FUNNT   0615    1775
FWAM    0157
FWUP    0296
FXFLA   0194    0847    0939    0952    0960    1071    1205
FXIT    1534    1134    1523
GETAD   0461
GETCH   0471    1232
GETSK   0475
GFLAG   0404
GFRST   0474
GNEXT   0473    0667    0695    0784    0868
GPARM   0424
GSIGN   0486
GTCON   0504
GTMP    0472    1489
GTOPP   0466
GTSTA   0213    0788
HFLG1   0321
HSTPT   0302    1156    1263    1710
I       0067
ICNT    0422
ID      0565    0612    0612
IMPY    0487    1598
INF     0077
INITS   0456
INPT    1429    1268    1277    1305    1444
INREA   0226
INTCK   0459
INTCV   1606    1631
INTEA   0178
INTEL   0340
INTFL   0075
INTGR   0402
IOBUF   0359
IOHD    0909    0663    0724    0732
IOHEA   0229    0909
IOHED   0649    0731
IOST    1521    1456    1529    1542    1547
IOSTA   0225
KEYFG   0324
L2T1    0347
L2T2    0348
LASTN   0398
LBOP    0119
```

CROSS-REFERENCE TABLE

LDEF	0489										
LFTCK	0464										
LIMIT	0465										
LIN	0962	0592									
LINEF	1460	1465									
LISTA	0167										
LNFD	0072	0968	1068	1233	1241	1278	1306				
LNINC	0315										
LNUM	0457										
LOCL1	0345	0925	0934	1074	1075	1076	1077	1080	1234	1267	1270
	1411	1413	1454	1457	1667	1680	1684	1685	1686	1687	1691
LOCL2	0346										
LPCK	0437	0665									
LPCKE	0439	0727	0870								
LPOP	0128	1145									
LROMA	0123	0762									
LSTAK	0299										
LSTPT	0301	1254	1499	1503	1610	1615	1704	1733	1734		
LTR	0479	0886									
LWAM	0295										
M1	0124	0983	1115	1160	1293	1317	1323	1395	1437	1502	1509
	1582	1591									
M10	0131	1266									
M100	0144	1369									
M1024	0146										
M11	0132										
M12	0133										
M13	0134										
M14	0135										
M15	0136										
M2	0125	1157	1611								
M23	0137										
M25	0138										
M256	0145	0924									
M3	0126										
M32	0139										
M36	0140										
M4	0127	1614									
M48	0141										
M5	0128	1224									
M6	0129										
M64	0142										
M8	0598	1199									
M80	0143										
M9	0130										
MASK1	0153										
MASK5	0154										
MAW	0294	0766	0767								
MAXIN	0400										
MAXSN	0147										
MBIN1	0406										
MBIN2	0407										
MBOX1	0351										
MDIMA	0208										
MEOV	1250	1264									
MER8	0445	1724									
MF	0524	0600	1334	1361	1388						
MINIT	0455										
MODE	0305	0849	1713	1716	1721	1740					
MOUTA	0638	0977									
MSFLG	0389										
MVTOH	0477										
NAD1	1135	1181	1220								
NADD	1132	1029	1034	1044	1226						
NBYB	1239	1247	1322	1331	1394	1402	1406				
NBYT	1232	1059	1067	1245							
NDISP	0392										
NDWRT	0500										
NESC	0708	0703									
NEXT1	0450										
NEXTA	0467										
NEXTL	0449										
NFOP	0896	0669	0682	0688							
NFOPA	0612	0896									
NFOPT	0578	0612	0612	1774							
NMR1	1331	1339	1341	1355	1385						
NMR2	1343	1335	1337								
NMR3	1357	1333									
NMR4	1366	1422									
NMR5	1381	1358									
NMR6	1387	1360									
NMR7	1405	1415									
NMR8	1417	1408									
NMRR	1314	1090									
NMOTA	0200	1719									
NOEOF	0482										
NPRIN	0383										
NPWRT	0499										
NULL	0575	1776									
NUM	0396	0397	0657								

CROSS-REFERENCE TABLE

NUMCA	0211									
NUMOA	0212									
NXTDT	0304									
NXTST	0405									
OBLNK	0468									
ODDIT	0496									
OF	0493									
ONE	0045	1392								
ONEXT	0469	1286	1291							
OPCH1	0646	1190								
OPCHA	0171									
OPDMK	0079									
OPMSK	0076	0658	1567							
OPTYP	0397									
OTPT	1454	0800	0805							
OTPTR	0293									
OUT1	1480	1470								
OUTCR	0470									
OUTIA	0179									
OVCHA	0216									
PAGE	177A									
PARMA	0306									
PARMN	0307									
PBPTO	0313									
PBPTR	0310									
PBUFF	0311									
PBUFO	0312									
PEND	0365	1113	1260	1289	1294	1712	1723			
PEXMA	0165									
PFLAG	0391									
PFOR	1149	1137	1146							
PFORM	0643	1677								
PGIN0	047A									
PGINT	045A									
PI/2	0044									
PLACE	0384	1703								
PLOTE	0650	1014	1619							
PMODE	0329									
PPFF	0366									
PR17	1764	1761								
PR2	1760	1737	1746	1747	1749					
PR3	174A	1757								
PRADD	030A	0807	0813	1133						
PRCHK	0644	0969	0997	1762						
PRFLG	0332									
PRINN	0639	1748								
PTR1	051A	1111	1497							
QTCHA	0230									
QTCHK	0642	0801	0809	1672						
QTOP	0109	0161	0764							
QUOTE	0561	0562								
RBOP	0117									
RBUFF	0360									
RCOUT	0333									
RDYDA	0166									
RECO2	0749	0743								
REFD	0161									
REFDL	0339									
RES	0417									
RESB1	0453									
RESBP	0454									
RET1	1445	1435								
RET2	0421									
RETN2	0420	0905								
RETN3	0419	0877								
RITE	0637	0725	0733							
RLFLG	0519									
ROUND	0495									
RPCK	0439	0910								
RPCKE	0440	0713	0876	0914						
RPOP	0115	156A								
RRF1	1277	1269	1287	1301	1303					
RREC	1254	1023	1063	1307						
RSAVE	0292									
RSLTE	0373									
RSPTR	041A	0865								
RSTAK	0416									
RSTKA	015A									
RSULT	0426	1032	1036	1096						
RTRL	2885	0913								
RUNA	0163									
S	0082									
SALTA	0207									
SAVBP	0452									
SAVEA	0325									
SAVER	0326									
SBADR	0394									
SBPTR	0387	0668	0696	0700	0726	0787	0869	0902	1258	1295	1300
	14A2	1709								
SBPUD	0431	0900								

CROSS-REFERENCE TABLE

SBUFA	0199									
SC13	0599	1539	1543							
SCHK	0849	0829	0838	0843						
SD	0525	1336	1343							
SEC	0320									
SFLAG	0393									
SFMT	0640	1636								
SGET	0655	0566								
SGNS	0497									
SGP1	0665	0697								
SGP2	0694	0689								
SGP3	0695	0706								
SGP4	0699	0666								
SGP5	0713	0705								
SIGN	0399	0400								
SKND	1305	1283	1285							
SLCOD	0328									
SLEND	0202									
SLWST	0447	1646	1668							
SNFLA	0197									
SOUT	0722	0565								
SOUT1	0735	0730								
SOUT2	0736	0729								
SOUT3	0731	0738								
SPA	0975	0593								
SPLFA	0196									
SSBPT	0395									
SSYMA	0170	1572								
ST	0377	0867	0874	1012	1617	1714	1720			
START2	0490									
START	0162	0007								
STAT	0943	0589								
STBLK	0758	0661	0722	0781	0796	1006	1625			
STEC	0608	1171								
STEXP	0443	0498								
STMAX	0494									
STOR1	0428									
STORE	0427	1209								
STPFL	0323	1521								
STROP	0430	0890								
STSRH	0442									
SUMPA	0227									
SVAR	0817	0802	0810							
SVARA	0742	0701								
SYF12	0485									
SYF25	0460									
SYMC1	0433	0678								
SYMC2	0432									
SYMCK	0434									
SYMTP	0298									
SYSNT	0570	1777								
SYTSA	0209									
T1	0516	0831	0839	0844	0855	0971	0993	1010	1429	
T2	0529	0656	0674	0708	0714	0716	0798	0840	0845	0859
T3	0530									
T5	0513	0728	0742	0749	0774	0782	0786	0817	1105	1117
	1486	1507	1511	1634						
T6	0531	1187	1198	1204	1210	1318	1340	1346	1365	1421
T7	0532	1320	1348	1350						
T8	0514	0660	0897	1434	1469	1476	1589	1599		
T9	0533	1016	1024	1089	1091	1381	1387	1505	1506	
TBEND	0515	1473	1600							
TD	0523	1084	1093							
TEMP	0350	0351								
TEMP1	0352	0888								
TEMP2	0353	0891								
TEMP3	0354									
TEMP4	0355									
TEMPS	0349	0806	0812	0860	0866	0975	1007	1132	1135	1142
	1154	1162	1170	1173	1179	1219	1222	1566	1570	1585
	1627	1632	1641							
TERM	0364	1000	1742	1760						
TFLAG	0318									
TFLG1	0322									
TFOR	1184	1147								
TMPA	0203									
TMPOP	0184	1717								
TMPRE	0521	1114	1495	1504						
TNULL	0527	1119								
TREGE	0372									
TSRHA	0168									
TSTPT	0300									
TTY	0503	0970	0982	0998	1458	1480	1544	1548	1700	
TW	0520	1079	1086							
TX	0522	1087	1239	1244						
TYPEC	0641	1045	1642							
TYPFL	0078									
UFLAG	0392									

CROSS-REFERENCE TABLE

UKPTR	0309						
UNITS	0319						
UNNM	1554	0944	0955	1008			
UNSTK	044A	1679	1681				
UPARA	0191						
VALUE	0297						
VAR01	064A	0751					
VAR0A	0214	0675	0744				
WBUFF	0423	1483					
WIDTH	0385	1693					
WRIT2	1642	1653	1664	1731			
WRIT4	1669	1675					
WRIT5	1677	1673					
WRIT6	172A	1701					
WRIT7	1703	1694					
WRITA	1735	1671	1730				
WRITB	168A	1683					
WRITE	0159						
WRITF	1667	1647					
WRITN	1654	1643					
WRITQ	1663	1644					
WRITS	1652	1654					
WRITX	1659	1649					
XC	0412	1484	1485	1491	1496		
XEC4A	0164	0808	0814	1537	1764		
XEC6A	0221						
XFAR1	0425	1718					
XFAR2	0429	0920	0943	0954	0987	1192	1393
XFTOS	1495	1497					
XSCRA	0220						
YC	0414						
ZAP	0152						
ZONK	0491	1095					

MICROPROCESSOR

All of the above-listed routines and subroutines of basic instructions are implemented by the basic computing system shown in FIGS. 3A–B. Central control of this system is achieved by microprocessor **120.** As shown in the block diagram of FIG. **27** and in the detailed scehmatic diagram of FIGS. **28** A–D, the microprocessor comprises a bipolar ROM **300** including seven ROM chips organized into 256 words of 28 bits. Eight J-K flip-flops contain the ROM address; (i.e. a 4-bit primary address and a 4-bit secondary address). A single chip 16-bit data selector permits any one of 16 different qualifier lines to be tested with a 4-bit qualifier code. This 4-bit qualifier code ROM chip serves a dual function in that it provides a complementing code to the 4 primary address flip-flops as well as selecting the proper qualifier to be tested. If branching in any ROM state is desired, the microinstruction BRC must also be given, BRC occurring with a QN (qualifier not met) signal from the data selector will cause the least significant bit of the address code to be inhibited to the secondary address flip-flop, thus causing the address to branch according to the state of the qualifier.

An additional feature of this ROM organization is the IQN microinstruction (inhibit if qualifier not met). When the IQN is given and the qualifier selected by the qualifier code is not met, the signal CCO (clock code zero) goes low. This inhibits all shift clock pulses from the clock decoder which in effect prevents execution of microinstructions in that ROM state.

To minimize the ROM word length, two 3-to-8 line decoders are used to expand 3 R-code outputs and 3 X-code outputs into a total of 14 microinstructions. Also the SCO and SCI outputs from ROM No. **5** are decoded in the Memory. The ALU code outputs **AC0, AC1,** and **AC2** are treated as address inputs to the ALU ROM and therefore need no decoding.

The microprocessor is responsible for the following:

1. Issuing a four-bit clock code to the clock decoder during each ROM state.
2. Issuing microinstructions to the memory, including the read and write microinstructions.
3. Issuing microinstructions to the shift registers for gating serial data into or out of the proper registers.
4. Issuing a four-bit ALU code to the Arith Logic Unit to select the proper binary or BCD arithmetic function.
5. Performing logical decisions (branching) based on the states of 16 qualifier inputs to the microprocessor.
6. Issuing next address information to the ROM address flip-flops in the microprocessor.
7. Transferrinng control to the input/output controller via the I/O strobe for execution of input or output instructions.

The full set of 28 ROM outputs with their associated microinstructions, the list of 16 qualifiers and assigned codes, and the microprocessor mnemonics are contained in the following tables:

MICRO-INSTRUCTION SET TABLE

CONTROL FIELD	← POSITIVE TRUE ROM OUTPUT →	← NEGATIVE TRUE OUTPUTS DECODED μ-INSTRUCTION OUTPUT →	FUNCTION
GENERAL	1. IQN		INHIBIT SHIFT CLOCK IF QUALIFIER NOT MET.
	2. BRC		BRANCH: INHIBITS S00 IF QUALIFIER NOT MET.
	3. $\overline{TTT}$		T BUS ⟶ T REG

-continued

CONTROL FIELD	← POSITIVE TRUE ROM OUTPUT	← NEGATIVE TRUE OUTPUTS DECODED μ-INSTRUCTION OUTPUT				FUNCTION
	4. $\overline{TTM}$					T BUS → M REG
	5. $\overline{XTR}$	DECODED IN MEMORY				A/B REG → R BUS
			SC1-	SC0		
S-CODE	6. SC1	ZTS	0	0		ZERO → S BUS
	7. SC0	MTS	0	1		M REG → S BUS
		TTS	1	0		T REG → S BUS
		UTS	1	1		ONE → S BUS
		DECODED IN MICROPROCESSOR	RC2	RC1	RC0	
R-CODE	8. RC2	UTR	0	0	0	ONE → R BUS
	9. RC1	PTR	0	0	1	P REG → R BUS
	10. RC0	TRE	0	1	0	T REG → E REG → R BUS
		WTM . 2TR	0	1	1	store contents T REG → MEMORY
		TQ6 . 2TR	1	0	0	T BUS → Q REG (BIT 6)
		QTR	1	0	1	Q REG → R BUS
		RDM . ZTR	1	1	0	READ MEMORY <M> → T REG
		ZTR	1	1	1	ZERO → R BUS
		DECODED IN MICROPROCESSOR	XC2	XC1	XC0	
X-CODE	11. XC2	TTQ	0	0	0	T BUS → Q REG
	12. XC1	QAB	0	0		Q REG (BIT n) → AB FLIP-FLOP
	13. XC0	BCD	0	1	0	BCD ARITHMETIC MODE OF ALU
		TBE	0	1	1	T BUS → E REG → R BUS
		CAB	1	0	0	COMPLEMENT THE AB FLIP-FLOP
		TPP	1	0	1	T BUS → P REG
		TTX	1	1	0	T BUS → A/B REG
		NOP	1	1	1	NONE OF THE ABOVE
		DECODED IN ALU	AC2	AC1	AC0	
ALU	14. AC2	XOR	0	0	0	EXCLUSIVE OR . . . . . R ⊕ S → T BUS
	15. AC1	AND	0	0	1	LOGICAL AND . . . . . R · S → T BUS
	16. AC0	IOR	0	1	0	INCLUSIVE OR . . . . . R + S → T BUS
		ZTT	0	1	1	ZERO → T BUS
		ZTT . CBC	1	0	0	ZERO → T BUS, CLEAR BINARY CARRY
		IOR . CBC	1	0	1	INCLUSIVE OR, CLEAR BINARY CARRY
		IOR . SBC	1	1	0	INCLUSIVE OR, SET BINARY CARRY
		ADD	1	1	1	BINARY ADD

CONTROL FIELD	ROM OUTPUT	FUNCTION
CLOCK	17. CC1 18. CC2 19. CC4 20. CC8	THIS 4-BIT CODE INITIALIZES A PRESETTABLE DOWN COUNTER TO GENERATE ANY NUMBER OF SHIFT CLOCKS FROM 1 THROUGH 16. SHIFT IS INHIBITED BY IQN IF QUALIFIER NOT MET.
QUALIFIER	21. QC3 22. QC2 23. QC1 24. QC0	THIS 4-BIT CODE PERFORMS TWO FUNCTIONS: (1.) ADDRESSING THE DATA SELECTOR TO SELECT ONE OF SIXTEEN QUALIFIER INPUTS, (2.) PROVIDES COMPLEMENT CODE TO PRIMARY FLIP-FLOPS
SECONDARY ADDRESS	25. S03 26. S02 27. S01 28. S00	10 THIS 4-BIT-CODE PROVIDES COMPLEMENT CODE TO THE SECONDARY FLIP-FLOPS. IF BRC IS GIVEN AND QUALIFIER IS NOT MET, THE S00 BIT IS INHIBITED

-continued

SPECIAL MICRO-INSTRUCTIONS:

TQR = UTR . XTR . . . . . TRANSFERS Q-REGISTER TO PRIMARY ADDRESS AS SHOWN

- Q14 . Q11 ⟶ PO4
- Q12 ⟶ PO5
- Q13 ⟶ PO6
- Q14s,6 ⟶ PO7

IOS = PTR · XTR . . . . . INITIATES:
a) TRANSFER OF CONTROL TO I/O if Q10 = 1
b) SETS "SINGLE SERVICE" FF IN I/O VIA $\overline{SRA}$ IF Q10 = 0

SPECIAL OPERATIONS:

BCD SUM ⟶ $A_{<0-3>}$ = BCD . $\overline{UTR}$. ROM CLOCK

CLEAR DECIMAL CARRY = QAB . ROM CLOCK

SET DECIMAL CARRY = UTR . BCD . ROM CLOCK

DECIMAL ADD = BCD . ZTT . . . . . $T_{<0-3>} + A_{<0-3>} \longrightarrow Q_{<0-3>}$

10'S COMPLEMENT W/DECIMAL ADD = $\overline{0}$ BCD . ADD . . . . .

. . . . . $\overline{T_{<0-3>}} + A_{<0-3>} \longrightarrow Q_{<0-3>}$

QUALIFIER SET TABLE

QUALIFIER CODE QC3	QC2	QC1	QC0	MNEMONIC	FUNCTION
0	0	0	0	Q00	SHIFT/SKIP ONE BIT
0	0	0	1	Q01	SHIFT/SKIP TWO BITS
0	0	1	0	Q02	SHIFT/SKIP FOUR BITS
0	0	1	1	Q03	SHIFT/SKIP EIGHT BITS
0	1	0	0	Q04	FAST SQUARE ROOT QUALIFIER
0	1	0	1	Q05	SET BIT IN G/S GROUP; FDV QUALIFIER
0	1	1	0	Q06	T-BUS QUALIFIER VIA TQ6
0	1	1	1	QBC	BINARY CARRY FROM ALU
1	0	0	0	QP0	P REGISTER, BIT 0, FOR BCD COUNTING
1	0	0	1	Q15	INDIRECT ADDRESS, CLEAR BIT IN G/S GROUP
1	0	1	0	QMR	MEMORY REFERENCE QUALIFIER
1	0	1	1	Q10	CURRENT PAGE QUALIFIER, FXA QUALIFIER
1	1	0	0	QNR	NON-SERVICE REQUEST QUALIFIER
1	1	0	1	Q08	FMP QUALIFIER
1	1	1	0	QDC	DECIMAL CARRY FROM ALU
1	1	1	1	*QRD	ROM DISABLE (NORMALLY ZERO)

POP WILL PRESET ROM ADDRESS FLIP-FLOPS AT TURN-ON

*QRD MAY BE USED WITH ION TO INSURE ZERO SHIFT EXCEPT WHEN IN I/O LOOP

MICROPROCESSOR MNEMONICS

Clock Signals

$\overline{MCK}$	Memory Clock
SCK	Shift Clock
XTC	External Clock
RCF	ROM Clock for Flip Flops
RCA	ROM Clock for Address Flip Flops
$\overline{IIC}$	Inhibit Internal Clock
$\overline{INH}$	Inhibit Clock
$\overline{IPS}$	Inhibit ROM Clock (Also primary and secondary Flip Flop)
CC8, CC4, CC2, CC1	Clock Code: Binary Code that programs the number of shift clocks
$\overline{CC0}$	Inhibits Shift Clocks

Address Mnemonics

$\overline{POP}$	Power on Preset
ION	Inhibit if Qualifier not met
BRC	Branch

Q-Register

$\overline{TQR}$	Transfer Q11, Q12, Q13, Q14, to primary addr. Flip Flops
$\overline{TTQ}$	T-Bus to Q-Register

-continued

MICROPROCESSOR MNEMONICS

$\overline{QTR}$	Q-Register to R-Bus
Q10, Q9, Q8, Q7, Q6, Q5, Q4, Q3, Q2, Q1, Q0	Bits 10 – 0 of Q-Register

Data Qualifiers

$\overline{QP0}$	Bit 0 of P-Register
QRD	Qualifier ROM Disable (I/O interupt)
QNR	Qualifier No Request (Keyboard Interupt)
QDC	Decimal Carry
QBC	Binary Carry

Memory

$\overline{SC0}$, SC1	S-Bus Code
$\overline{TTT}$	T-Bus to T-Register
$\overline{TTM}$	T-Bus to M-Register
$\overline{RDM}$	Read Memory

-continued

MICROPROCESSOR MNEMONICS	
$\overline{WTM}$	Write Memory
A, B, P, E-Registers	
$\overline{QAB}$	Q-Register to B Flip Flop
AB = 0	A-Register Operation
$\overline{AB = 0}$	B-Register Operation
$\overline{TTX}$ (ROM)	T-Bus to A or B Register (Originates at ROM Decoder)
$\overline{XTR}$	A or B Register to $\bar{R}$-Bus
$\overline{TTP}$	T-Bus to P-Register
$\overline{PTR}$	P-Registers to $\bar{R}$-Bus
$\overline{TBE}$	T-Bus to E-Register toR-Bus
$\overline{TRE}$	T-Register to E-Register

-continued

MICROPROCESSOR MNEMONICS	
$\overline{UTR}$	Units to $\bar{R}$-Bus
AC2, AC1, AC0	Arithmetic Codes for Arithmetic Logic Unit
$\overline{BCD}$	Decimal Arithmetic
SDR	Disables ROMs for Single Step Tester Operation

Each of the ROM chips of FIGS. **27** and **28A–D** is organized into 256 words of 4 bits each constructed in accordance with the following table, where each L represents a low (or 0) state and each H represents a high or (1) state:

ROM CHIP 3

0016	000-007	LLLL	LLHL	HHHH	LHHL	HHHH	LHHH	HHLH	HHLH
0017	008-015	LHLL	LHLL	HLHH	LHHL	HLHL	LHLH	LLLL	LLLL
0018	016-023	LHHH	LHHH	LLLL	LHHH	LLLL	LLLL	LHLL	HHLL
0019	024-031	LLHL	LLHL	LHLH	HLHL	LHLL	HLHL	LLLL	LLLL
0020	032-039	LHLL	LLLL	HHLL	LLLL	HLLL	HLLL	LLHL	HLLH
0021	046-047	LLHH	LLHH	LLLL	LLLL	LHLH	HLLL	HHLH	LHHH
0022	048-055	HLLL	HHLL	LLLH	LLLH	LHHL	LLLH	HLHH	HLHH
0023	056-063	LLLH	LLLH	HHLL	LHLH	LLLH	HHLH	HLHH	LHLL
0024	064-071	LHHL	LHHH	HLHH	HHHL	LLLL	HLLL	LHHL	HLHH
0025	072-079	LLLH	LLLH	HLHH	LHLL	HLLL	LLHL	LLLL	HLLH
0026	080-087	LHHH	LHLH	HLLL	LHHL	HLHH	HHLL	LLLL	LLLL
0027	088-095	LLHH	LLLL	HHHH	LHHH	LLLL	LLHH	LLLL	LHLL
0028	096-103	HHHL	HLLL	LHHL	LHHH	LHHH	LHHH	LHHH	HHHH
0029	104-111	LLHH	LLHL	LHLL	LLLL	LLLL	LHLH	HHLL	HHLL
0030	112-119	LLLH	LLLH	LLLL	LHLH	LHLH	LHLH	HLLH	LHHH
0031	120-127	LLLL	LHLH	HHHH	LHHL	LLLH	LHLL	LLLL	LHLH
0032	128-135	HLHL	LHLL	HHLL	LHLH	LHHH	HLHH	LLLL	LLLL
0033	136-143	LHLH	LLHL	LLHH	HHLH	LLHL	HLHL	HLLH	LHLL
0034	144-151	LLLL	LLHL	HLLH	HLLL	HLLH	LLLH	HLLH	HLLH
0035	152-159	HHHL	LHHH	HLHH	HLLH	LLLH	LLLL	LHHH	HHLH
0036	160-167	HLLL	HHHL	LLLH	HHHL	LHHH	LLHL	LLHL	HHHH
0037	168-175	LHHL	HLLL	LHLL	HLHL	LLLL	LLLH	LHLL	HLLL
0038	176-183	LLLL	LHHL	LLHL	HLLH	LHHH	LLHH	HLHH	HLHH
0039	184-191	LHHH	HLLL	LLHL	LLHL	LHLH	LLLL	LLHL	LLLL
0040	192-199	LLLH	LLLL	LLLH	LLLH	HHLH	HHLH	LLHH	HLHH
0041	200-207	HHHH	LHLH	HLLH	LHLL	HHLH	LHLH	LLLL	LLLL
0042	208-215	HLHH	HLLL	LHHL	LLLH	HHHL	LLHL	HLHL	HLHL
0043	216-223	HHHH	HHLH	LLLL	LLLL	LLLL	LHLL	LLLH	HLLH
0044	224-231	HLLH	HLLH	LLLL	LLLL	HHLH	HLLL	LHHL	HHHL
0045	232-239	LLLL	HHHL	HLLL	HLLH	HHLL	HLHL	LLLL	LHHH
0046	240-247	LLHL	LLHL	LLHH	LLLL	LLLH	HHLH	LLHL	LHLL
0047	248-255	LHLL	LHLH	HLLL	LHHH	LLLL	LLLL	LHLH	LLLL
0048									

ROM CHIP 4

0053	000-007	HLHL	HLLH	LLLH	HLLL	LHHL	HLLL	HLHH	LLHH
0054	008-015	LHHH	HHLH	HLLL	HLLL	HLLH	LHHL	HLLL	LHHH
0055	016-023	HLLL	LLHH	LLHL	LLLL	HLHH	HLLH	LLLL	LLLH

ROM CHIP 4 -continued

0056	024-031	LLLL	LLLL	HLHL	LLHL	LLLL	LLLH	LHLL	LHHH
0057	032-039	LHLH	HLLH	HLLL	HLHH	LHLH	HHLH	HHLL	HLLL
0058	040-047	LLLL	LLLL	HHLH	HHLL	HHLH	HHLL	HLLH	HLHL
0059	048-053	HHHH	LLHH	LLLL	HLLL	LLLL	HLHH	HHLL	LHLL
0060	056-063	LLLH	LLLH	HHHL	HHHL	LLLL	HLLL	HLHL	LLLL
0061	064-071	LLLH	HHLH	HHLH	HLLL	HLHL	LLLL	HHHH	HLHH
0062	072-079	LLLL	LLLL	HLHL	LLHL	LLHH	HLLL	LHLH	LLHH
0063	080-087	HHHL	HLHH	HLHL	HLLL	LLLL	HHLH	LHLL	LLLH
0064	088-095	HLLL	LHLH	HHHL	HHLL	LHHH	LHLL	HLLL	LLLL
0065	096-103	HHHL	HLHH	HLLL	HLHL	HHHH	HHLL	HHHL	LLHL
0066	104-111	LLLL	LLLL	LHLL	LLHL	HLLL	LLLL	HLHL	LLHH
0067	112-119	LHHH	LHLH	HLLL	HLLL	LLHH	HLHH	HHHH	LHLL
0068	120-127	LLHL	HHHH	HLLL	HLHH	LHLL	HLLH	HLLH	HHHL
0069	128-135	LLLH	HLHH	LHHL	HLHH	HLHH	HLLH	HLHH	LLHH
0070	136-143	HLLH	LHLL	HLHL	LLHH	LLLL	HHHH	LLLL	HLLH
0071	144-151	HLHL	HLHL	HLLL	HLLH	HLHH	LHLL	HHLL	HHLL
0072	152-159	HLLL	HLLH	HLHL	HHHH	LLLL	HLLL	HLHL	LHLL
0073	160-167	LLLH	HHHL	HHHL	LLHH	HHHL	HLHH	HHHH	HHHL
0074	168-175	HLLL	HHLH	LHLL	HLLL	HLLL	LLLL	LLHL	LLLH
0075	176-183	HHLL	HLLH	HLLH	HLLL	HHHH	HLLL	HHLL	LHLL
0076	184-191	HLLL	LLHH	LHLL	HHLL	LLHH	HLLL	LHLH	LHHH
0077	192-199	LHHL	HLLH	LLLL	HLLL	HLHH	HLLL	HHHL	HHHL
0078	200-207	LLHL	HHHH	HHHL	HHHL	LLHH	LLLH	HLLL	LHHH
0079	208-215	LHHH	HLLL	HLLL	LLLH	HLHL	LLLH	LHLL	LHHL
0080	216-223	LLHL	HLHL	HLHL	LLHL	HLLL	LLLL	LLLL	HLHL
0081	224-231	HLLL	HLLH	LHHL	HHLL	HLLL	LHLL	LHLL	LHHH
0082	232-239	HHLL	LLHH	LLHL	LHLH	LHHH	LLLL	HLLL	LLHH
0083	240-247	HHLL	LHLL	LLHH	HLLL	HLLL	HLHH	LHLL	LHLL
0084	248-255	LLHH	HHLH	LLLH	LLLL	HHLL	HHLH	LHLL	HLLL
0085									

ROM CHIP 5

0090	000-007	HHHH	LLLL	LLLL	HHHH	LLLL	LLLL	LLLH	LLLH
0091	008-015	HHHH	LLLL	HLLH	LLLL	HHHH	HHHH	LLLL	LLHH
0092	016-023	LLLL	LLLL	LLLL	LLHH	LLLL	HHHH	HLHH	LLLL
0093	024-031	LLHH	LLHH	LLLL	LHLH	LLHH	LHLH	HLHH	HLHH
0094	032-039	HHHH	LLLL	LLLL	LLLL	HLHH	HLHH	HHHL	LLLL
0095	040-047	LHHH	LHHH	LLLL	LLLL	LLHH	HLHH	LLLL	LLLH
0096	048-055	LHLH	HLHH	LLLL	HHHH	LLLL	LLLL	LLLL	LLLL
0097	056-063	LLLH	LLLH	LLHH	LLHH	LLLL	HLHH	HHHH	HHHH
0098	064-071	HHHH	HLHH	LLHH	LLLL	LLLH	LLLL	LLLL	LHLH
0099	072-079	HHHH	LLLH	HHHH	LLLL	HHHH	HHHH	LLLH	HHHL
0100	080-087	LLLL	LLLL	HLHH	LLLL	HHHH	HHHH	HHHH	HHHH
0101	088-095	LHHH	LLLL	LLLH	LLLL	LHLL	LLHH	LLHL	HHLL
0102	096-103	HHHH	HLHH	HLHH	HLHH	LLHH	HLHH	LLLL	LLLH
0103	104-111	LLLL	LLHH	HHHH	HHHH	HLHH	HHHH	HHHH	LLLL
0104	112-119	HHHH	HLHH	HHHL	LLLL	LLHL	LLLL	LLHH	LLLL
0105	120-127	HLHH	LLLL	LLHH	HLHH	LLHH	HHLH	LLLL	HHHH
0106	128-135	HHHH	LHLH	HHHH	LLLL	LLLH	LLHH	LLLL	LLLL
0107	136-143	HHHH	HHHH	LHLH	HLHH	HLHH	HLHH	LLLL	LLLH
0108	144-151	LLLL	LLHH	HLHH	LLHH	LLLL	HHHL	LLLL	HHHL
0109	152-159	LLLL	LLLL	LLLL	LHLL	HHHH	HLHH	LLHH	LHLL
0110	160-167	HHHH	HHHH	LHLH	LLHL	HHHH	LLLL	HHHH	LLLH

ROM CHIP 5 -continued

0111	168-175	HHHH	HHHH	HHHH	LLLL	HHHH	HHHH	HHHH	LHLL
0112	176-183	HHHH	LHLH	HHHH	HHHH	LLLL	LLHL	HHHL	LLLL
0113	184-191	HHHH	HLHH	LHLH	LHLH	HHHH	HLHH	LLHH	LLHH
0114	192-199	HHHH	HLHH	HHHH	HHHH	LHLH	HLHH	LHLH	LHLH
0115	200-207	LLLL	HHHL	HHHH	LLLH	HHHH	LLLL	HLHH	HHHH
0116	208-215	HHHH	LLHH	HLHH	HHHH	LLHH	HHHH	LLLL	HHHH
0117	216-223	LLLL	HLHH	LLLL	LLLL	HLLH	LHLH	LHLH	LLLL
0118	224-231	LLLL	LLLL	LLLL	HHHL	HLHH	HLHH	LLHL	LLLL
0119	232-239	HHHH	LLLL	HLHH	HHHH	LLLL	LLLL	HHHL	HHHL
0120	240-247	LLHH	LLLL	LHLL	LLLH	HHHH	HHHH	HLLH	LHLH
0121	248-255	HHHH	LLLL	HLHH	LLHH	HHHH	LLLL	HHHH	HHHH
0122									

ROM CHIP 6

0127	000-007	LHHL	LHHH	HHHL	LLHH	HHHL	HLHH	HHHL	HHHL
0128	008-015	LLHL	LLHH	LLHH	LHLH	LHLH	LHLL	LHHH	LHLH
0129	016-023	LHHH	LHHH	LHHH	LHHL	LLLH	LHHL	LHHH	LHHH
0130	024-031	HHHH	HLHH	LLHH	LLLH	LHLH	LLLH	LHHH	LHHH
0131	032-039	LLLL	LHHH	LHHH	LHHH	LHLH	LHHH	LHHL	HHHH
0132	040-047	HHHH	HLHH	HHHH	HLHH	HLHH	LHLH	LHHH	HHHL
0133	048-055	LLLH	LHLH	LHHH	LHLL	LHHH	LLLH	LLHH	LLHH
0134	056-063	HHHH	HLHH	LHHH	LHHH	LHHH	LLHH	LHLH	LLLH
0135	064-071	LLHH	LHHL	LHHH	LHHH	LHHL	LHLH	HLHH	LLLH
0136	072-079	LHHH	HLHL	LHHL	LHHH	LHLH	LHHL	LLHL	LHHH
0137	080-087	LHHH	LHHH	LLHH	HHHL	LHHH	LHHL	LHLH	LHHH
0138	088-095	HLHL	HLHL	LLHH	LHHH	LHHH	LHHL	LLHH	LHHH
0139	096-103	LLHL	LHHH	LHHH	LHHL	LHLH	LHHL	LHHH	HHHL
0140	104-111	LLHH	HLHL	LHHH	LLHL	LLHH	LHHL	LHLH	LHHH
0141	112-119	LHLL	LHHH	LHHL	LHLH	LLLH	LHHH	HHLH	LHHH
0142	120-127	LHLH	HLHH	LHLL	LHHH	LHHH	LHHH	LHLH	LHHH
0143	128-135	LLHL	LHHH	LHHL	LHHH	LHLH	LHHH	LHLH	LHLH
0144	136-143	LHHL	LLHH	LLLH	LHLH	LHLH	LHHH	LHHH	LLLH
0145	144-151	LHHH	LHHH	LLLH	LHLL	LHHH	LHHH	LLHH	HHHL
0146	152-159	HLHL	HHHL	LHHH	LHHH	LHHH	LLHH	LHHH	LLHH
0147	160-167	LHLL	LLLL	LLLH	LLHH	LHHH	LLHH	LHLH	LLHH
0148	168-175	LLLL	LLHL	LHHH	LHHH	LHHH	LHLH	LHHH	LHHH
0149	176-183	LHHH	LHHH	LHLH	LHLH	LHLH	LLHH	LHHL	LHHH
0150	184-191	LLHL	LLHH	LHHH	LHHH	LHLH	LHHH	LHHH	LLHH
0151	192-199	LHLH	LLHH	LHLH	LHHH	LHLH	LHHH	LLLH	LLLH
0152	200-207	HLLH	HHHL	LHHL	LHHH	LHHL	LHHH	LHLH	LHHH
0153	208-215	LHLH	LHLL	LHLH	LHLH	HHHH	LHLH	LHHH	LHHH
0154	216-223	HLHH	LHHH	LHHH	LHHH	LLHH	LHHH	LHLH	LLHH
0155	224-231	HLHH	HHHL	LHHH	LHHH	LLHH	LHLH	LHHH	HLH
0156	232-239	HLLL	LHHH	LLLH	LLHH	LLLH	LHHH	LLHH	LHH
0157	240-247	LHLL	LHHH	LHLH	LLLH	LHLL	LHLH	LLHH	LLL
0158	248-255	LLHL	LHHH	LHHH	LHLL	LHLH	LHLH	LHHH	LHH
0159									

ROM CHIP 7

0164	000-007	LHHH	HHHH	LLLL	LLLH	LLLL	HHHH	LHHH	LHHH
0165	008-015	LLLH	HLLH	HHHH	HHHH	LLHL	LHHH	LHHH	LHLH
0166	016-023	LHLL	LHLL	LHHH	LHHH	LHHH	LHHH	LLHH	LHHH
0167	024-031	LHHH	LLLL	LHHH	LHHH	LLHL	LHHH	LHHH	LLHH
0168	032-039	LHHH	LHHH	LHLL	HHHH	LLLL	LHHH	LHHH	HHHH
0169	040-047	LHHH	LLLL	LHHH	LLLL	HHHH	HHHL	HHLH	LHLL
0170	048-055	LHHH	LHHL	LHHH	LHHH	LHLH	LHHH	LHHH	LLLL
0171	056-063	LHHH	LLLL	LHHH	LLLL	LHHH	HHHL	HLHL	LLLL
0172	064-071	LHHH	HHHH	LHHH	HHHH	LHHH	LHLL	HLLL	HHHH
0173	072-079	LLLH	LHHH	LHHH	LHHH	LHLL	LHHH	LHHH	LHHH
0174	080-087	LHLL	HHLL	HHHL	HHHH	LHHH	LHLL	LLLL	LLLL
0175	088-095	LHHH	LHHH	LHHH	LHHH	LHLL	LHHH	LHHH	LHHH
0176	096-103	LHHH	HHHL	HLHH	HHHH	HHHH	HHHH	HHLL	LHHH
0177	104-111	LLLH	LHHH	LLLL	LHHH	LLHH	LHHH	HLLL	LHHH
0178	112-119	LHHH	LHHL	LHHH	LHHH	LHHH	LHHH	LHHH	LHHH
0179	120-127	LHHH	HLLL	LHHH	LLHH	LLLL	LHHH	LHLL	LHHH
0180	128-135	LHHH	HLLL	LHHH	HHHH	HHHH	HHHH	LLLL	LHHH
0181	136-143	HHHH	LHHH	LHHH	LLHH	LHHL	LHHL	LHHH	HLLL
0182	144-151	LLLL	HHHH	HLHH	HHHH	HHHH	LHHH	LLLL	HHHH
0183	152-159	LLLH	LHLL	HHHH	LHLH	LHHH	LLHH	HHHH	HLLL
0184	160-167	LHHH	LHHH	LHHH	LLLL	LLLL	HLLH	LHLL	HHHH
0185	168-175	LLLH	LHHH	LHLL	HHHH	LLLL	LHLL	LHLL	LHLL
0186	176-183	LLLL	HHHH	LLLL	LLLL	LLLH	HLLH	LHHH	LHHH
0187	184-191	LLLH	LHHL	LHHH	LHHH	LHLH	LHHL	LHLH	LLLL
0188	192-199	LLLL	HHHL	LLLL	LHHH	HHHH	HHHH	HHHH	HHHH
0189	200-207	LLLH	HHHH	LHLH	HLLL	LHHH	LHHH	LLHH	LLLL
0190	205-215	LHHH	HHHH	HLHH	LHLL	HHHH	LLHL	LHHH	LHHH
0191	216-223	LLLH	HLHH	LLLH	LLLH	LLLL	LHHH	LHHH	HHHH
0192	224-231	HLLL	HHHH	LHHH	LHLL	HHHL	LLHH	LLLL	LHHH
0193	232-239	HHHH	LHHH	LLHH	LLLL	LHHH	LHLH	LHHH	LHHH
0194	240-247	LHHH	LHHH	LLLL	LLLL	HHHH	LLHL	LLLH	LHHH
0195	248-255	LHHH	HHHH	LLHH	LHHH	HHHH	HLLH	LHHH	LHHH
0196									

ROM CHIP 8

0201	000-007	LHHH	HHHL	HHHL	HHHL	HHHL	HLHL	LLHL	LLHH
0202	000-015	HHHL	HHHL	HHHL	HHHH	HHHL	LHHL	LHHL	HHHH
0203	016-023	HHHL	HHHL	HHHL	LHHH	HHHH	LHHH	LHLL	LHLL
0204	024-031	LHHL	LHHL	HHHH	HHHH	HHHL	HHHH	HHHL	HHHL
0205	032-039	HHHL	LHHL	HHLH	HHHL	HHHH	HHHL	HHHL	LHHL
0206	040-047	LHHL	LHHL	LHHL	LHHL	HLHL	HHHH	HHHL	HHHL
0207	048-055	HHHH	HHHH	HHHL	LHHL	HHHL	HHHH	HHHH	HHHH
0208	056-063	LHHL	LHHL	HLHH	HHHH	HHHL	LHLH	HLHL	HHHL
0209	064-071	LHHL	HHHL	HLHH	HHHL	HHHH	HHHL	LHHL	HHHH
0210	072-079	HHHL	HHHL	LHHH	HHHL	LLLH	LHHH	HHHH	HHHH
0211	080-087	HHHL	HHHL	HHHH	HHHL	HHHH	HHHH	LHHL	LHLH
0212	088-095	HHHL	HHHL	HHHL	HHHL	HHHL	LHHH	HHHH	HHHH
0213	096-103	HHHL	LHLL	HHHL	HHHL	LHHL	HHHL	HHHL	LLHL
0214	104-111	HHHL	HHHL	HHHH	HHHL	LHLH	LHHH	LLLL	LHLL
0215	112-119	HHHH	HHHL	LHHH	HHHH	HLHH	HHHL	HLHL	HHHL
0216	120-127	LHHL	LHHL	HLHL	LHLL	HHHH	HHHL	HHHL	HHHH
0217	128-135	LHHL	HHHH	LHHH	HHHL	HHHL	HLHL	HHHH	HHHH
0218	136-143	HHHH	LHHL	HHHH	LHLH	HHHH	HHHL	HHHH	LLHL

ROM CHIP 8-continued

0219	144-151	HHHH	HLHH	HHHH	HHHL	HLLL	HHHH	LHHL	HHHL
0220	152-159	HHHL	HHHL	HHHL	HHHL	HHHH	HHHH	HLHL	HHHL
0221	160-167	LHHL	HHLL	HHHH	HHHH	HHHH	HHHH	LHLL	HHHL
0222	168-175	HHHL	HHHL	HHLH	HLLL	HHHH	HHHH	HHLH	HHHL
0223	176-183	HHHH	LHLL	HHLL	HHLL	HHHH	HHHL	HHHL	HHHL
0224	184-191	HHHL	HHHH	HHHL	HHHL	HHLL	HHHL	HHHL	HLLL
0225	192-199	LLLL	HHHH	HHHL	HHHH	HLHL	HLLL	HLHH	HLHH
0226	200-207	HLHL	HHHL	HHHH	HHHH	HHHH	HHHL	HHHH	HHHH
0227	208-215	HHLH	HHHL	HHHH	HHHL	HLHH	HLHL	LHLL	LHLH
0228	216-223	HLHL	HHHL	HHHH	HHHH	HHHL	HHHL	HHHL	HHHH
0229	224-231	HHHL	LHHH	LLLL	HHLH	HHHH	LHLH	HHHH	HHHH
0230	232-239	LHHL	HHHL	HHHH	LLLL	HHHH	HHHL	LLLH	HHHH
0231	240-247	HLHL	HHHL	LHLH	LLHL	LLLH	HLHL	HHHL	HHHH
0232	248-255	HHHL	HHHL	LHLL	HLHL	HHHL	HHHL	LHHL	LHHL
0233									

ROM CHIP 9

0238	000-007	HHLH	HHLL	HHLL	LHLH	HHLL	HHHH	HHLH	HHLH
0239	008-015	HHLL	HHHH	LHLH	HHLL	HLHH	HHLL	HHLL	LHLL
0240	016-023	HHLL	HHLL	HHLL	HHHH	HHLL	HHLL	HHLL	HHLL
0241	024-031	HHLL	HHHH	HHHL	LHLL	HLHH	LHLL	HHHL	HHLL
0242	032-039	HLLH	HHLL	LHHH	HHLL	LHHH	LHLL	HHLL	HHLL
0243	040-047	HHLL	HHHH	HHLL	HHHH	HHHH	HHLL	HHLL	HHLL
0244	048-055	LHLL	HHLL	HHLL	HHLH	HHLL	HHLL	LHLL	LHHH
0245	056-063	HHLL	HHHH	HHLL	HHLH	HHLL	HHLL	HHHH	HLLH
0246	064-071	HHLH	HHLH	HHLL	HHLL	HHLL	LHLL	HHHH	LHLL
0247	072-079	HLLL	HLHH	HHLL	HHLL	HHLH	LHLL	HHLL	LHHL
0248	080-087	HHLL	HHLL	HHLL	HHLL	LHHL	HHHH	HHLH	HHLH
0249	088-095	HLHH	HLHH	HHLL	HHLL	HHLL	HHHH	HHHL	LHHL
0250	096-103	HLLL	HHLL	HHLL	HHLL	HHLL	HHLL	HHLL	HHLH
0251	104-111	HHHH	HLHH	HHLH	LHLL	HHLL	LHLL	HHLH	HHLL
0252	112-119	HLLL	HHLL	HHLL	HHLL	HHLL	HHLL	HHLL	HHLL
0253	120-127	HHLH	HHHH	HHHH	HHLL	HHLH	HHHL	LHLL	HLLH
0254	128-135	HHLH	LHHH	HHLL	HHLL	LHLL	HHLL	LHHH	LHLL
0255	136-143	HHHH	HHHL	LHLL	HHLL	HHLL	HHLL	LHHL	HHLL
0256	144-151	HLLH	HHHH	HHLL	HLHH	HHLL	HLLH	HHHH	HHLL
0257	152-159	HHLL	HHLL	HHLL	HHLL	HLLH	HHLL	HHLL	LHHH
0258	160-167	HHLH	HHHL	LHLL	LHHH	LHHL	HHHH	LLLL	HHLL
0259	168-175	HHHH	LHLL	HHHH	HHLL	LHLH	HLLL	LHHL	HHLL
0260	176-183	HLLH	LHLL	HHLH	HHLH	LHHH	LHHH	HHLL	HHLL
0261	184-191	HHLL	HHLL	LHLL	HHHL	LHHH	HHLL	HHLL	LHHH
0262	192-199	HHHL	HHLL	LHLH	LHLL	LHLL	HHHL	LHLL	LHLL
0263	200-207	HHHH	HHLL	HLHL	LHHH	LHLL	LHLL	HHLL	HLLH
0264	208-215	HHHL	HLHH	HHLL	HLHH	HLLL	HHHH	HHHL	LHHL
0265	216-223	HHLL	HHLL	HHLL	HHHH	LHHH	LHLL	LHLL	HHLL
0266	224-231	LHHH	HHHH	LHLL	LHLL	HHLL	HHLL	LHHL	HHLL
0267	232-239	HHHH	HHLL	HHLL	LLHH	HHLL	HHLL	HHLL	LHLL
0268	240-247	HHHH	HHLL	LHHH	HHLL	LHHH	HHHH	LHHH	LHLL
0269	248-255	HLLL	HHLL	HHLL	HHHH	HLLL	HHLL	HHLL	HHLL
0270									

Each of the 71 basic instructions employed by the calculator is implemented by one or more of the above-described microinstructions and associated control signals issued by the microprocessor. The manner in which this is accomplished is shown and described in detail in the flow charts of FIGS. **29A–H**. Each rectangular box of these flow charts represents a state of ROM **300** of the microprocessor and includes the mnemonic of the microinstructions and control signals stored in that ROM state. The number at the upper right-hand corner of each of these rectangular boxes represents the number of shift clock pulses required by the microinstructions of that ROM state. A simplified overview of these detailed flow charts is shown in FIGS. **6A–B**.

PROGRAMMABLE CLOCK

Given a computing system organized to process binary data serially and under control of microinstructions stored in ROM **300** as shown in FIGS. **3A–B** and **27**, the implementation of a general purpose instruction set requires that some number of bits be shifted into or out of the storage registers. Depending on the operation being performed, the number of bits may vary from zero to *n*, where *n* is the number of bits in a single machine word.

If each clock period of the ROM clock corresponds to a one bit shift, a count loop must be employed to provide the desired number of shifts. A rather large number of such count loops would exist in order to implement an entire instruction set. An alternative method is to provide additional hardware which permits assignment of the desired number of shifts in a single state of ROM **300**. Such an arrangement requires a variable cycle time for each state of ROM **300** but results in a very substantive saving in total number of ROM states.

To implement a variable number of shift clocks in a single state of the microprocessor, two separate clocks are required. The shift clock is applied to the data storage registers in the memory, the shift register block, the arithmetic logic unit and the input/output block. The ROM clock is applied to the ROM address flip-flops in the microprocessor, and occurs once for each state in the microprogram. The number of shift clock pulses that occurs in any given ROM state is determined by a 4-bit clock code set to the clock decoder from the microprocessor.

If no shift clocks are desired, a separate signal $\overline{CC}$ from the microprocessor inhibits the shift clock output, independent of the clock code issued in that state. In this way, any number of shifts between and including zero and 16 may be implemented with a 4-bit clock code and an inhibit signal.

This inhibit signal offers an additional powerful feature when gated by the qualifier test logic in the microprocessor as shown in FIG. **3A**. The qualifier test logic includes a 4-bit qualifier code from ROM **3** that selects one of 16 qualifier inputs to the data selector. The data selector output QN (qualifier not met) will be high if the selected qualifier input was low. By using the QN signal to gate the inhibit microinstruction, IQN, the shift clock will be inhibited only when the qualifier is not met. Thus, all microinstructions requiring shift clocks that are issued in a given ROM state may be either executed or inhibited, depending on the logical state of the qualifier under test.

The ROM clock is applied to the eight J-K flip-flops which address the 256 word microprocessor ROM. During any given state, the complementing (J-K) inputs to the 4 primary address flip-flops are set up by the qualifier code or q-register code. The 4 secondary address flip-flop inputs are determined by the ROM **4** outputs, the BRC microinstruction, and the data selector output QN. Where ROM clock goes low, the negative edge-triggered flip-flops will cause transition of the ROM address to the next ROM state.

As shown in the block diagram of FIG. **30** and the detailed schematic diagram of FIGS. **31A–C**, a crystal controlled system clock output is inverted to generate memory clock, MCK. This signal is again inverted to clock a D flip-flop having an output (control clock), which will go low if the end-of-count signal (borrow) from the down counter has occurred at the D input. The ROM clock will also go low at this time, initiating a new ROM state in the microprocessor. Control clock will normally remain low for one system clock period, and in turn generates a load signal which is delayed a half period from control clock by means of a second D flip-flop. The 4-bit clock code from the microprocessor is preset into the counter while the load signal is low.

As the load signal goes high, ROM clock also goes high, completing the fixed interval portion of ROM clock and shift clock as shown in FIG. **32**. A series of clock pulses are now gated onto shift clock, SCK, until the preset counter has counted down to zero, causing control clock to again go low, completing the ROM cycle.

The inhibit signal, INH, from memory may lengthen the normal fixed interval of ROM clock by clearing the D flip-flop and holding control clock low. This may occur during memory refresh or external test operations. In this situation, the counter remains preset and the correct number of shifts will be generated when the inhibit goes away.

SHIFT REGISTER UNIT

As shown in the detailed schematic diagrams of FIGS. **28A–D** and **33A–D**, A-register **122**, B-register **124**, P-register **126**, Q-register **128**, and E-register **130** of FIGS. **3A–B** comprise bipolar status registers, the contents of which are recirculated when data is output to the R-bus or the S-bus. Full control of these registers in use and type of operations performed is maintained by the microinstructions from the microprocessor. The number of bits to be shifted in any one ROM state of the microprocessor is determined by the number of shift clocks from the clock decoder. This shift clock appears at the shift clock input of each shift register that is enabled by the microprocessor during that ROM cycle.

ARITHMETIC LOGIC UNIT

The development of complex read-only memory arrays on a single chip have made possible a hardware implementation of central processing units (CPUs) and arithmetic logic units (ALUs) with far fewer components than were previously possible. In this application, two bipolar read-only memory chips are combined with carry flip-flops and adapted to perform one-bit binary logic and arithmetic operations as well as four-bit binary-coded-decimal (BCD) arithmetic operations. The two bipolar read-only memory chips may comprise, for example, Hewlett-Packard 16-pin dual-in-line packaged bipolar ROMs organized into 256 words by 4-bits and of the same type as shown and described in U.S. Pat. No. 3,721,964.

The binary/BCD Arithmetic Logic Unit consists of five integrated circuits connected as shown in the block diagram of FIG. 34 and the detailed schematic diagram of FIGS. 33A–D. Specifically, the packages consist of two 1024-bit ROMs, a dual D-type flip-flop and two quad two-input NAND gates.

Internally the desired binary logical function, binary arithmetic operation or BCD operation is selected by the ALU code as shown below.

	ALU CODE BCD	AC2	AC1	AC0	ALU FUNCTION	DESCRIPTION
Binary Functions	0	0	0	0	XOR	Exclusive OR . . . . R ⊕ S ⟶ T
	0	0	0	1	AND	Logical AND . . . . R . S ⟶ T
	0	0	1	0	IOR	Inclusive OR . . . . R + S ⟶ T
	0	0	1	1	ZTT	Zero ⟶ T-BUS
	0	1	0	0	ZZT.CBC	Zero ⟶ T-BUS, Clear Binary Carry
	0	1	0	1	IOR.CBC	Inclusive OR, Clear Binary Carry
	0	1	1	0	IOR.SBC	Inclusive OR, Set Binary Carry
BCD Functions	0	1	1	1	ADD	Binary ADD. . . . B + S + BC ⟶ T,C
	1	0	1	1	BCD ADD	BCD ADD. . . . $T_{0-3} + A_{0-3} \longrightarrow T\,\Sigma_{0-3}$
	1	1	1	1	BCD COMP/ADD	10's Complement and BCD ADD

ALU FUNCTION CODE ASSIGNMENTS

The function code input BCD selects between the binary mode and BCD mode of operation.

In the binary mode, the function code inputs AC0 AC1, and AC2 select the desired logical function or arithmetic operation. The binary input data enters ROM #1 on the carry, S-bus and R-bus input lines, and the binary result appears on the T-bus and binary carry output lines. ROM #2 is not used in the binary mode.

In the BCD mode of operation, the two function code lines AC0 and AC1 are disabled from the Microprocessor and these two lines carry the T02 and T03 bits of BCD data from the T-Register. The ALU function code line AC2 is used to select the desired BCD operation. If AC2 is low, the four-bit output Σ0,Σ1,Σ2,Σ3 will be the BCD sum of the two BCD data inputs. If AC2 is high and decimal carry has been set, the four-bit output Σ0,Σ1,Σ2,Σ3 will be the BCD Tens Complement of the BCD data from the T-Register. In the BCD mode, the binary carry output will be disabled and the decimal carry output will be enabled to ROM #1.

Although only one-fourth of the available registers in ROM #1 are required for the eight binary operations, the concept of adding a second 1024-bit ROM to perform the BCD operations grew from several basic concepts:

1. The least significant BCD sum bit, Σ0, is always identical to the binary sum bit; therefore, only three additional outputs Σ1,Σ2, and Σ3 need be generated. For BCD complement operations, the decimal carry flip-flop defines whether or not the least significant bit should be complemented.
2. In forming the nine's complement of the T-Register BCD data in ROM #1, it can be seen that for 8421 code the second least significant bit T01 is the same before and after forming the complement. Thus only two bits, T02 and T03 need be complemented prior to input into ROM No. 2. The ten's complement with add is then found by presetting decimal carry and performing a BCD sum of the three most significant digits in ROM No. 2.
3. With only eight ROM inputs available, some sharing of inputs is required for ROM No. 1. During binary operations, all four function codes and only one bit of T-Register data is required. During BCD operations, all four bits of T-Register data and only two function codes are required. Use of two NAND gates in wire-OR connection with the open collector function codes AC0 and AC1 permits sharing of the two inputs.

This arrangement left one input still available to ROM #2. By programming this input to always make output DCI true, the micro-instruction $\overline{UTR}$ can serve two purposes—placing units on the R-bus and also set decimal carry if BCD is true. When BCD is false, clock is inhibited to decimal carry. This feature permits saving decimal carry information during all binary operations. Similarly, binary carry is saved during the four binary operations AND, IOR, XOR, and ZTT by connecting AC2 such that when AC2 is false the shift clock is inhibited to the binary carry flip-flop.

In summary, the mode select input BCD performs the following functions:

1. Addresses the proper 128 word set of word lines in ROM #1.
2. Enables the T02 and T03 data lines to ROM #1 only in BCD mode.
3. Enables clock to decimal carry flip-flop only in BCD mode.
4. Selects binary carry or decimal carry into ROM #1 as appropriate.
5. Transfers outputs Σ0,Σ1,Σ2,Σ3, to A-Register only in BCD mode.

The remaining three ALU function codes select the proper set of word lines in ROM No. 1 to perform the eight binary functions. In addition, the AC2 input performs the following functions.

1. Enables clock to binary carry flip-flop only during the four carry-related binary functions and the BCD comp/add function.
2. In the BCD mode, AC2 causes BCD data bit T00, T02 and T03 to convert to nine's complement form.

The ALU has a total of 15 inputs which include 8

data inputs, 2 clock inputs and 5 microinstructions. Four data output lines are required, and two additional output lines from carry flip-flops are available as qualifier inputs to the microprocessor. The ALU and shift register mnemonics are listed in the following table:

SHIFT REGISTERS & ALU BOARD MNEMONICS	
$\overline{TRE}$	T-Register to E-Register to $\overline{R}$-Bus
TO0	Bit 0 of T-Register
$\overline{TBE}$	T-Bus to E-Register to $\overline{R}$-Bus
$\overline{TTX}$ - TEST	T-Bus to A/B-Register from Tester
$\overline{TTX}$ - I/O	T-Bus to A/B-Register from I/O (Board # 12)
$\overline{TTX}$ - ROM	T-Bus to A/B-Register from ⊢ Processo (Board #13)
$\overline{TTX}$	Logical "OR" of Three $\overline{TTX}$ Signals
AB	Status of AB-Flip-Flop AB = 0 A-Reg. Operation AB - 1 B-Reg. Operation
$\overline{XTR}$	A/B Register to $\overline{R}$-Bus
$\overline{UTR}$	Logical "1" to $\overline{R}$-Bus
$\overline{TQR}$	Q-Register to Primary Address Flip-Flop
$\overline{AB}$	Complement of AB
$\overline{TTP}$	T-Bus to P-Register

-continued

SHIFT REGISTERS & ALU BOARD MNEMONICS	
SCK	Shift Clock
QP0	Qualifier, Bit 0 of P-Register
$\overline{PTR}$	P-Register to $\overline{R}$-Bus
QO0	Q-Register Bit 0
$\overline{QTR}$	Q-Register to $\overline{R}$-Bus
RCK	ROM Clock
QAB	Q-Register to AB-Flip-Flop, also clears decimal carry.
SCB	Set Binary Carry
$\overline{BCD}$	Decimal Arithmetic
AC2	ALU Operation Code
QBC	Qualifier, Binary Carry
$\overline{S}$ -BUS	Data Bus
AC1	ALU Operation Code
AC0	ALU Operation Code
T02	Bit 2 of T-Register
T03	Bit 3 of T-Register
SDR	Signal to Disable ROMs
T01	Bit 1 of T-Register
T-BUS	Data Bus
ALU	Arithmetic Logic Unit
(−) Indicates Negative True Signal	

The following table gives an example of how the two ALU ROM chips shown in FIGS. 33A–D and 34 can be constructed to implement the above described ALU functions (in this table each 1 represents a low state and each 0 represents a high state):

ROM #1

/	1000;	1/	0000;	2/	0000;	3/	1000
4/	1000;	5/	0000;	6/	0000;	7/	1000
8/	0000;	9/	0000;	10/	0000;	11/	1000
12/	0000;	13/	0000;	14/	0000;	15/	1000
16/	0000;	17/	1000;	18/	1000;	19/	1000
20/	0000;	21/	1000;	22/	1000;	23/	1000
24/	1000;	25/	1000;	26/	1000;	27/	1000
28/	1000;	29/	1000;	30/	1000;	31/	1000
32/	1000;	33/	0000;	34/	0000;	35/	1000
36/	1000;	37/	0000;	38/	0000;	39/	1000
40/	0000;	41/	0000;	42/	0000;	43/	1000
44/	0000;	45/	0000;	46/	0000;	47/	1000
48/	0000;	49/	1000;	50/	1000;	51/	1000
52/	0000;	53/	1000;	54/	1000;	55/	1000
56/	1000;	57/	1000;	58/	1000;	59/	1000
60/	1000;	61/	1000;	62/	1000;	63/	1000
64/	1100;	65/	1100;	66/	1100;	67/	1100
68/	1100;	69/	1100;	70/	1100;	71/	1100
72/	0000;	73/	0000;	74/	0000;	75/	1000
76/	0000;	77/	0000;	78/	0000;	79/	1000
80/	0100;	81/	0100;	82/	0100;	83/	1100
84/	0100;	85/	0100;	86/	0100;	87/	1100
88/	0000;	89/	1000;	90/	1000;	91/	0100
92/	1000;	93/	0100;	94/	0100;	95/	1100
96/	1100;	97/	1100;	98/	1100;	99/	1100
100/	1100;	101/	1100;	102/	1100;	103/	1100
104/	0000;	105/	0000;	106/	0000;	107/	1000
108/	0000;	109/	0000;	110/	0000;	111/	1000
112/	0100;	113/	0100;	114/	0100;	115/	1100
116/	0100;	117/	0100;	118/	0100;	119/	1100
120/	0000;	121/	1000;	122/	1000;	123/	0100
124/	1000;	125/	0100;	126/	0100;	127/	1100
128/	0000;	129/	0000;	130/	0000;	131/	0000
132/	0000;	133/	0000;	134/	0000;	135/	0000
136/	0010;	137/	1010;	138/	1010;	139/	0110
140/	1010;	141/	0110;	142/	0110;	143/	1110
144/	0001;	145/	1001;	146/	1001;	147/	0101

ROM #1 -continued

```
148/  1001;   149/  0101;   150/  0101;   151/  1101
152/  0011;   153/  1011;   153/  1011;   155/  0111
156/  1011;   157/  0111;   158/  0111;   159/  1111
160/  0000;   161/  0000;   162/  0000;   163/  0000
164/  0000;   165/  0000;   166/  0000;   167/  0000
168/  0010;   169/  1010;   170/  1010;   171/  0110
172/  1010;   173/  0110;   174/  0110;   175/  1110
176/  0000;   177/  0000;   178/  0000;   179/  0000
180/  0000;   181/  0000;   182/  0000;   183/  0000
184/  0011;   185/  1011;   186/  1011;   187/  0111
186/  1011;   189/  0111;   190/  0111;   191/  1111
192/  0000;   193/  0000;   194/  0000;   195/  0000
196/  0000;   197/  0000;   198/  0000;   199/  0000
200/  1010;   201/  0010;   202/  0110;   203/  1010
204/  0110;   205/  1010;   206/  1110;   207/  0110
208/  1011;   209/  0010;   210/  0111;   211/  1011
212/  0111;   213/  1011;   214/  1111;   215/  0111
216/  1001;   217/  0001;   218/  0101;   219/  1001
220/  0101;   221/  1001;   222/  1101;   223/  0101
224/  0000;   225/  0000;   226/  0000;   227/  0000
228/  0000;   229/  0000;   230/  0000;   231/  0000
232/  1011;   233/  0011;   234/  0111;   235/  1011
236/  0111;   237/  1011;   238/  1111;   239/  0111
240/  0000;   241/  0000;   242/  0000;   243/  0000
244/  0000;   245/  0000;   246/  0000;   247/  0000
248/  1010;   249/  0010;   250/  0110;   251/  1011
252/  0110;   253/  1010;   254/  1110;   255/  0110
```

ROM #2

```
  /  0000;     1/  0000;     2/  0000;     3/  0000
  4/  0000;    5/  0000;     6/  0000;     7/  0000
  8/  0000;    9/  0000;    10/  0000;    11/  0000
 12/  0000;   13/  0000;    14/  0000;    15/  0000
 16/  0000;   17/  0000;    18/  0000;    19/  0000
 20/  0000;   21/  0000;    22/  0000;    23/  0000
 24/  0000;   25/  0000;    26/  0000;    27/  0000
 28/  0000;   29/  0000;    30/  0000;    31/  0000
 32/  0000;   33/  0000;    34/  0000;    35/  0000
 36/  0000;   37/  0000;    38/  0000;    39/  0000
 40/  0000;   41/  0000;    42/  0000;    43/  0000
 44/  0000;   45/  0000;    46/  0000;    47/  0000
 48/  0000;   49/  0000;    50/  0000;    51/  0000
 52/  0000;   53/  0000;    54/  0000;    55/  0000
 56/  0000;   57/  0000;    58/  0000;    59/  0000
 60/  0000;   61/  0000;    62/  0000;    63/  0000
 64/  0000;   65/  0000;    66/  0000;    67/  0000
 68/  0000;   69/  0000;    70/  0000;    71/  0000
 72/  0000;   73/  0000;    74/  0000;    75/  0000
 76/  0000;   77/  0000;    78/  0000;    79/  0000
 80/  0000;   81/  0000;    82/  0000;    83/  0000
 84/  0000;   85/  0000;    86/  0000;    87/  0000
 88/  0000;   89/  0000;    90/  0000;    91/  0000
 92/  0000;   93/  0000;    94/  0000;    95/  0000
 96/  0000;   97/  0000;    98/  0000;    99/  0000
100/  0000;  101/  0000;   102/  0000;   103/  0000
104/  0000;  105/  0000;   106/  0000;   107/  0000
108/  0000;  109/  0000;   110/  0000;   111/  0000
112/  0000;  113/  0000;   114/  0000;   115/  0000
116/  0000;  117/  0000;   118/  0000;   119/  0000
120/  0000;  121/  0000;   122/  0000;   123/  0000
124/  0000;  125/  0000;   126/  0000;   127/  0000
128/  1111;  129/  1101;   130/  1011;   131/  1001
```

ROM #2 -continued

132/	0111;	133/	0000;	134/	0000;	135/	0000
136/	1101;	137/	1011;	138/	1001;	139/	0111
140/	1110;	141/	0000;	142/	0000;	143/	0000
144/	1011;	145/	1001;	146/	0111;	147/	1110
148/	1100;	149/	0000;	150/	0000;	151/	0000
152/	1001;	153/	0111;	154/	1110;	155/	1100
156/	1010;	157/	0000;	158/	0000;	159/	0000
160/	0111;	161/	1110;	162/	1100;	163/	1010
164/	1000;	165/	0000;	166/	0000;	167/	0000
168/	0000;	169/	0000;	170/	0000;	171/	0000
172/	0000;	173/	0000;	174/	0000;	175/	0000
176/	0000;	177/	0000;	178/	0000;	179/	0000
180/	0000;	181/	0000;	182/	0000;	183/	0000
184/	0000;	185/	0000;	186/	0000;	187/	0000
188/	0000;	189/	0000;	190/	0000;	191/	0000
192/	1101;	193/	1011;	194/	1001;	195/	0111
196/	1110;	197/	0000;	198/	0000;	199/	0000
200/	1011;	201/	1001;	202/	0111;	203/	1110
204/	1100;	205/	0000;	206/	0000;	207/	0000
208/	1001;	209/	0111;	210/	1110;	211/	1100
212/	1010;	213/	0000;	214/	0000;	215/	0000
216/	0111;	217/	1110;	218/	1100;	219/	1010
220/	1000;	221/	0000;	222/	0000;	223/	0000
224/	1110;	225/	1100;	226/	1010;	227/	1000
228/	0110;	229/	0000;	230/	0000;	231/	0000
232/	0000;	233/	0000;	234/	0000;	235/	0000
236/	0000;	237/	0000;	238/	0000;	239/	0000
240/	0000;	241/	0000;	242/	0000;	243/	0000
244/	0000;	245/	0000;	246/	0000;	247/	0110
248/	0000;	249/	0000;	250/	0000;	251/	0000
252/	0000;	253/	0000;	254/	0000;	255/	0000

MEMORY UNIT

The calculator uses an all semiconductor memory system. Peripheral circuitry is bipolar and the memory consists of n-channel MOS read-only memory (ROM) and p-channel MOS read/write memory (RWM).

Addressing and physical layout of the memory module is done so that the number of words can be increased from 5K in the basic machine to 9K in the largest machine. The smallest increment of memory that can be added is 512 words.

The basic machine contains 9.5K words of memory, organized into 7.5K × 16 ROM, and 2K × 16 RWM. The 16-bit RWM words are divided into user registers and processor words. The largest machine contains 15.5K words of ROM and 4K words of RWM.

Read/Write Memory

As shown in FIGS. **35–37B** memory is made up of 1024 × 1, dynamics, read/write memory chips (Intel 1103). These devices are P-channel, MOS using silicon gate technology. To maintain the contents of memory, the device must be refreshed every 2 ms. This is accomplished by performing a read cycle at a given address. On each chip are 32 refresh amplifiers so that each read cycle, 32 cells get refreshed. The entire chip is then refreshed by cycling through the lower 5 address bits and reading each distinct address. The refresh period is 20 μs at least every 2 ms.

Logic levels on all input lines to the RWM chips are 0 to + 16v. This includes the 3 clock lines (chip select, Y-enable or write, and precharge), 10 address lines, and input data. The output data, however, is a current of 600 μa or more into 1K ohms or less. This low level output is wire-or-able with other chips to build larger systems.

Read Only Memory

As shown in Figs, **35** and **38–40** ROM chips are 4096 bit, *n*-channel MOS arranged 512 × 8. The devices are static and consume no power when not enabled. Data is retrieved from the ROMs by pulling the chip enable line from 0 to + 12v (turning the chip on), addressing the desired cells (0 or 4v levels) and selecting which output devices are to be enabled (4v or 0v). The output levels are sufficient to drive one TTL gate directly, and can be wire-or/ed for large systems.

As further shown in FIGS. **41** and **42A–D**, each ROM chip comprises six input buffers. These input buffers generate both the input and its complement. On the basis of the 64 possible combinations of the 6-inputs I_0-I_5, one of the 64 lines in the decoder is selected. The selected line enables one of the vertical lines in the 64 × **64** bit storage array. For example, let $I_0 - I_5 = 0$ and $_6 - I_8$ be don't-cares. This means line **X00** (octal) is selected.

The two 8 out of 32 select decoders must choose 16 lines from the 64 horizontal lines selected by the vertical line **X00.** (The 8 out of 32 select decoder is actually a 2 out of 8 decoder repeated 4 times in each of the sections A – B). The output from four MOS Fet's *a*, *b*, *c*, and *d* are wire-or/ed. MOS devices *a'*, *b'*, *c'*, and *d'* are also connected similarly. If I_6 and $I_7 = 0$, horizontal lines **1XX 2XX, 3XX, 5XX, 6XX, 7XX** are grounded in each of the four sections A–B. This insures that MOS FET's *b*, *c*, *d*, *b'*, *c'*, and *d'* are non-conductive. This allows signals on lines **0XX** and **4XX** to pass into the output sections through transistors *a* and *a'*.

The output section contains the output buffer, 1 of 2 decoder, and the output drivers *s*. The output buffer provides a stage of gain and wire-or-'s 4 lines from the storage array. The 1 of 2 decoder clamps the gates of 2 of the 4 output drivers in each section A-B by enabling either line I_8 or its complement ($\overline{I_8}$). This disables 1 of 2 signals coming from the output buffer. The output drivers then can be tied together with line (*e*) for a 512 × 8 organization.

Each of the above-listed constants and routines and subroutines of basic instructions employed by the calculator is stored in these ROM chips. The sixteen bits of each constant and basic instruction are stored in the $512_{10} \times 8_{10}$ ROM chips by organiziang the ROM chips into 64 ×64 bit matrices and computing the row and column numbers of each bit of each matrix by operating on each address and the particular bit (**15** through **8**, or 7 through **0**). The column number is computed by subtracting the last two digits of the address from 100_8. For example, the column number of address **000** = $100_8 - 00_8 = 100 = 64_{10}$ and the column number of address **777** = $100_8 - 77_8 = 1$. The computation of the row number (referred to as IR in the flow-chart of FIG. **44**) can best be described by referring to the flowchart of FIG. **44** and the associated table of FIG. **45**. Once the row and column numbers are found it is a simple matter of storing in that location of the matrix that particular bit (i.e., a 1 or a 0). A 0 is stored at a designated location by forming a metal gate to complete a MOS FET device at that location, and a 1 is stored at a designated location by leaving off the metal gate so that a a MOS FET device is not formed at that location.

M-Register

As shown in FIGS. **35** and **46A–B** included on the M-Register board is the 16-bit Address or M-Register, all chip enable decoding and buffering, and address buffers for both ROM and RWM. The register uses four, four bit, serial in and out, parallel in and out shift Registers. Upon receipt of a TTM instruction from the microprocessor, serial data from the T-Bus is accepted into the M-register. Nothing is done with this data until either a read or write instruction is received, then one of two decoders are enabled. These chip Enable decoders uniquely decode which block of 512 words, either ROM or RWM, is being addressed. If ROM is being addressed, the signal is inverted and amplified to +12v. For RWM the Chip Enable enables a gate, which allows a 16 Volt clock signal to reach the enabled RWM chips. The clock wave-form is generated on the control card.

The dynamic characteristic of the RWM chips, requires that all chips be enabled simultaneously during a refresh cycle, to refresh the entire read/write memory. The buffer circuits in the output of the Chip enable decoders allow the chip select clock to reach all of the RWM chips during refresh but only those being accessed, during a read or write cycle.

Totem Pole outputs or gates with resistor pull-ups are used a buffers for the ROM address lines. Using the totem pole output gates, the effects of crosstalk can be minimized while the resister pullup lifts the address lines above the required 4v level. The nand gates are enabled during a memory cycle so that the ROM address lines are inhibited at a 5v level. The RWM address lines must pull from 0v to + 16v. High voltage, open collector, inverters with discrete transistor pull-ups are used as buffers for all the address bits.

Control

A memory cycle consists of a read or write instruction from the processor accompanied by 12 clock pulses from the shift clock. As shown in FIGS. **35**, **46A–B** and **47A–B**, control uses these pluses and instructions to generate the clocks required by the RWM chips. A synchronous system of flip-flops and gates are used. The outputs from the flip-flops are then buffered to become the required clock signals (Pre-charge, Y-enable, chip select).

Refreshing the read/write memory is also taken care of by the control. An astable multivibrator with a repetition rate of 500 HZ minimum generates a signal which allows a refresh cycle to occur. A flip-flop generates the actual signal (REF), but only if the astable multivibrator signal is high, there is no read or write cycle in progress and the processor signal, $\overline{CCT}$, is high. $\overline{CCT}$ goes high between processor instructions, thus it is known that nothing is going to be interrupted when REF is generated. REF is then buffered by an open collector inverter and given to the processor $\overline{INH}$. $\overline{INH}$ halts the machine and the refresh cycle begins.

The same system used for a memory cycle is used during refresh to again generate the necessary clocks (Precharge and chip select). When the system returns to state 0 and REF is present, a counter is advanced one count. This counter provides the refresh addresses which go to the RWM only if REF is present. When this counter returns to state 0, it causes REF and $\overline{INH}$ to return to present conditions and the machine continues normal operation.

Another function of the control is to provide for extended memory capability. The control handles any external memory as if it were an extension of the internal memory. From the user's point of view, he does not need to know if an extended memory is connected other than the fact that avaiable memory has increased.

In addition, the control has the provision for extracting information from or loading information into the calculator T-Register through the D-Bus (data bus).

Other signals generated on the control are employed to direct the flow of data in the T-register.

T-Register

Data to and form the memory is temporarily stored in the T-register. As shown in FIGS. **35** and **48A–B** four 4 -bit, serial in and out, parallel in and out shift registers make up the actual T-register. The registers have a mode control (TMC) which when low, allows serial data flow and when high, allows parallel data flow.

Serial data enters the T-register in the presence of the $\overline{TTT}$ instruction, and in the presence of a $\overline{TTS}$ recirculated in the T-register to prevent loss of data.

Parallel data is accepted from eigher ROM or RWM during a read cycle. The ROM data is buffered by NANd gates and the RWM by sense amplifiers followed by the same NAND gates. All 16 bits are read from the ROM simultaneously. Eight bits are read from RWM twice during a read cycle. The eight bits to be written into RWM have their own discrete buffer stage that translates T^2L logic levels into 16v logic levels used by the RWM.

MEMORY SYSTEM MNEMONIC TABLE

SIGNALS GENERATED OUTSIDE MEMORY I/O CONNECTOR

$\overline{CCT}$	— Control clock-not, the inverted envelop of SCK
SCK	— Shift clock.

-continued

MCK — Memory clock, a continuous pulse train, used by the memory control for timing of the memory and refresh cycles.

IOD — I/O Data. Goes to control board to be gated to S-BUS.

ITS — I/O to S-BUS, the signal which gates IOD to S-BUS.

SCO & SCI — Coded signals which generate UTS – Units to S-BUS
ZTS – Zero to S-BUS
MTS – M-Reg to S-BUS
TTS – T-Reg to S-BUS

TTT — T–BUS to T–Reg, OV = True.

T-BUS — Data on this bus acts as inputs to M & T registers.

$\overline{RDM}$ — Read memory, negative true. Lasts for 12 clock pulses.

$\overline{WTM}$ — Write memory, negative true. Lasts for 12 clock pulses.

$\overline{INH}$ — Inhibit, negative true. The processor is stopped whenever $\overline{INH}$ is at zero volts. The memory control generates this signal while a R/W memory refresh cycle is present. I/O also generates it.

OTHER SIGNALS AT I/O CONNECTOR

Name	Source
$\overline{TOO}$	T-Register
TO1	T-Register
TO2	T-Register
TO3	T-Register

D-BUS — Data Bus - external data (extended memory data) enters machine via this bus.

$\overline{EDT}$ — External Data Transfer gates D-Bus data into machine O^{r} = True.

$\overline{EMB}$ — Extended memory busy. Signal provided by extended memory that tells memory control
a. Extended memory cycle is complete
b. Extended memory is present.

SIGNALS GENERATED ON READ/WRITE MEMORY CARDS

RWD(XX) — Read/Write data. Output from the 1103 memory.
600μa into 150$^{\Omega} = \Omega_1$''
0 current = 0

OTHER SIGNALS USED BY (RWM)

Name	Source
$\overline{A00–AO4}$	$\overline{CONTROL}$
AO5–AO9	M-REG
CEN	M-REG
RWI(XX)	T-REG
R/W	CONTROL
PCG	CONTROL

SIGNALS GENERATED ON T-REGISTER CARD

T00–T15 — T-register data bits. Used as data into memory T00-T03 are also outputs to the CPU. (4 bit processing)

RWI(XX) — Read/write inputs. T-register data gates to Read/write memory. +16V⟶GND.

OTHER SIGNALS USED BY T-REGISTER

Name	Source
$\overline{ROD(XX)}$	$\overline{ROM}$
TRI	CONTROL
TSC	CONTROL
$\overline{TMC}$	CONTROL
TPC	CONTROL
RWD(XX)	R/W MEM
RWE	M-REG

SIGNALS GENERATED ON M-REGISTER CARD

M00–M15 — M-register data bits. Used to generate address and chip select information. (M00 also is gated out on S-BUS by MTS)

I00–I07 — ROM address bits. Decodes down to two bits available at ROM output buffers.

I00–$\overline{I08}$ — Selects which RM output buffers are enabled.

CS(XX) — Chip enable, basic machine selects which ROM chips are turned on (+12V – ON)

AEN — Address enable. AEN = RDM + WTM

A05–A09 — Address bits for R/W memory. (+16V & GND)

CEN — Chip select, basic machine a negative true clock which selects which R/W chips are turned on. (+16V & GND)

RWE — Read/Write enable. A +5V signal any time a R/W chip is addressed for a machine memory cycle.

OTHER SIGNALS USED BY THE M-REGISTER

Name	Source
$\overline{T\text{-}BUS}$	$\overline{PROCESSOR}$
SCK	''
$\overline{RDM}$	''

-continued

$\overline{WTM}$	"
$\overline{TTM}$	"
VLD	CONTROL
MTS	"
$\overline{REF}$	"

SIGNALS GENERATED BY CONTROL

VOR — A signal generated half way through the memory cycle to disable the active pull up devices on the ROM outputs.

TRI — T-Register input TRI = (T-BUS) . (TTT) + (TOO) . (TTS)

MTS — M-Register to $\bar{S}$-BUS. Generated from SCO, $\overline{SCI}$

AO0–AO4 — Address bits for R/W memory also used during memory refresh.

$\overline{TSC}$ — T-Register series clock. $\overline{TSC} = \overline{SCK . (TTS + TTT + EDT)}$

TPC — T-Register parallel clock. (Strobes in data from memory) only during internal memory read cycle.

R/W — Read/Write. A clock which left at +16V for a read and clocked to GND during a write. (R/W memory only)

PCG — Precharge. The 3rd 16V clock required by the 1103 R/W memory chips.

$\overline{REF}$ — Refresh. OV when the memory is in a refresh cycle.

$\overline{CEM}$ — Call extended memory. Prevents ROM clock from changing μprocessor states. Given for all read and write commands. Signal is removed if the memory cycle is not extended memory cycle. If extended memory cycle, $\overline{CEM}$ is removedafter extended memory has completed cycle. O^{ε} = True.

$\bar{S}$-BUS — Gates I/O register, data, T00, M00 or ones onto $\bar{S}$-BUS and sent to processor. 0 = True.

$\overline{INH}$ — Inhibit negative true. The processor is stopped whenever INH is at zero volts. The memory control generates this signal while a R/W memory refresh cycle is present. I/O also generates it.

$\overline{TMC}$ — T-reg mod control. $\overline{TMC} = 0$; T-reg is set up to accept information from memory. $\overline{TMC} = 1$; T-reg. is set up to shift serially.

$\overline{EMC}$ — Extended memory cycle. +5" signal used to signal extended memory to being its cycle. 0^{ε} = True.

VOR — A signal generated half way thru memory cycle to allow data to flow out of ROM.

OTHER SIGNALS USED BY THE CONTROL

Name	Source
$\overline{CCT}$	PROCESSOR
T00	T-REG
IOD	PROCESSOR
ITS	"
SCO	"
SCI	"
$\overline{TTT}$	"
T-BUS	"
SCK	"
$\overline{RDM}$	"
$\overline{WTM}$	"
MCK	"
M00–M04	M-REG
AEN	M-REG
$\overline{EDT}$	EXTENDED MEMORY
D-BUS	I-O CONNECTOR
$\overline{EMB}$	EXTENDED MEMORY

SIGNALS GENERATED ON ROM BOARD

ROD(XX) — Read Only Data

OTHER SIGNALS USED BY ROM

Name	Source
$\overline{CS(XX)}$	$\overline{M\text{-}REG}$
I00—I07	M-REG
IO8 $\overline{IO8}$	M-REG
VOR	CONTROL

INPUT-OUTPUT CONTROL UNIT

The input-output control unit allows the calculator to communicate with the internal input, input-output, and output units and with external peripheral devices. As shown in FIGS. **31** A–C and **49** A–D, the input-output control unit is contained on two printed circuit boards, the "control and system clock" board and the "I/O register and gate interface" board. A third board, shown in FIG. **50**, is an I/O motherboard providing room for connecting four external interface cards to the calculator.

The internal input, input-output, and output units are distinguished from peripheral devices by the fact that the I/O language set addresses them directly. Hence, each I/O instruction contains an internal peripheral address as part of its makeup. The four internal directly-addressable input, input-output, and output units are the I/O register, the magnetic card reading and recording unit, the output printer unit and display unit.

The external peripheral devices are indirectly addressable and are connected via cable to an interface card which is plugged into the I/O motherhood at the rear of the calculator. The term indirectly addressable is defined here to mean the external peripheral devices are addressed by lines leading from the four most significant bits in the I/O register, thereby requiring an address word to be loaded into the directly addressable I/O register.

I/O CONTROL AND SYSTEM CLOCK SECTION

The function of the I/O control and system clock section is to provide control to the I/O register and gate interface section. This is accomplished by use of an I/O instruction set stored in the main memory of the calculator.

The microprocessor causes instructions from the memory unit to be loaded into the T-Register and then to be transferred to the Q-Register. The microprocessor determines the type of instruction and causes the proper execution of the instruction. If the instruction is an I/O type, control is transferred by the microprocessor to the I/O control and system clock section.

The microprocessor remains in a two-state waiting loop while the I/O control section is active. Time in the wait loop is between 0.72 microseconds and 6.5 microseconds.

Bits **5** through **10** from the Q-Register are connected to the I/O control section and remain constant during an I/O instruction execution time. Bits **5** through **8** representing the I/O instruction code are gated to the I/O address flip flops and entered on each clock time while the I/O is inactive. The four outputs of the address flip flops are connected to the address input of a 1 of 16 decoder and represent the starting state address of the I/O instruction to be executed. When the I/O control section is enabled, the input gates passing bits **5** through **8** to the I/O address flip flops are closed and the 1 of 16 decoder enabled. This allows the starting state I/O micro instructions to come from the 1 of 16 decoder. The next state address coming from the closed input gates will be the exit state ($1111=17_8$) unless modified by reopening the gates to let the original starting state code through or by modifying the output of one or more of the input gates using a wire or connection coming from the 1 of 16 decoder output. This address is sent to the I/O address flip flops inputs and clocked in on the leading edge of the first half clock cycle. The first half clock cycle turns off the 1 of 16 decoder and the address changes. The second half clock cycle enables the 1 of 16 decoder, allowing the next state micro instruction to appear. (See FIG. **51** for the timing described above). This process continues until the exit state is encountered. On the exit state, the I/O Control is disabled and control is returned to the microprocessor.

The I/O instructions involving the transfer of data between the I/O and the CPU (OT, LI, MI), require 16 passes through the same state (1 pass for each of 16 bits). This is achieved by checking the output of a 16-bit down counter and then decrementing after each pass through the state. If the counter indicates 0 has not been reached, it causes the starting state address to be reloaded into the address flip flops by opening the input gates. When 16 passes have been indicated by the counter, the input gates are not allowed to open; however, the next state (1111) is modified by the output of the 1 of 16 decoder through a wire-or connection on the 2nd bit to give state 1101. This address is input to the I/O address flip flops as in the preceeding paragraph.

The above-described operation of the I/O control section is also illustrated and further described in the flow chart of FIG. **52**.

Bit **9** is called a hold/clear bit. It allows a clear flag (CLF) to take place or not to take place after execution of the other I/O instructions (STF excepted).

Bit **10** is used in conjunction with the micro instructions $\overline{\text{PTR}}$ and $\overline{\text{XTR}}$ to give control to the I/O.

The I/O control and programmable clock mnemonics are given in the following table:

I/O CONTROL BOARD MNEMONICS

$\overline{\text{CC0}}$	Clock Code Zero
CC1	Clock Code One
CC2	Clock Code Two
CC4	Clock Code Four
CC8	Clock Code Eight
CCT	Control Clock to Tester
CEM	Call Extended Memory
$\overline{\text{CLC}}$	Clear Control
CLF	Clear Flag
DRC	Data Register Clock
EBT	Eight Bit Transfer
EOW	End of Word
$\overline{\text{IIO}}$	Inhibit Internal OSC
$\overline{\text{INH}}$	Inhibit Clock
$\overline{\text{IPS}}$	Inhibit Primary/Secondary
ITS	Input to S-Bus
MCK	Memory Clock
$\overline{\text{POP}}$	Power on Pulse
$\overline{\text{PTR}}$	P-Reg to R-Bus
QFG	Qualifier Flag
Q5	Qualifier Five
Q6	Qualifier Six
Q7	Qualifier Seven
Q8	Qualifier Eight
Q9	Qualifier Nine
Q10	Qualifier Ten
QRD	Qualifier ROM Disable
RCA	ROM Clock Address
RCF	ROM Clock Flip Flop
SCB	Set Carry Bit
SCK	Shift Clock
SCT	Shift Clock to Tester
$\overline{\text{SRA}}$	Service Request Acknowledge
STC	Set Control
STF	Set Flag
TCK	Tester Clock
TTO	T-Bus to Output
$\overline{\text{TTX}}$	T-Bus to A/B Reg.
XTO	External OSC
$\overline{\text{XTR}}$	A/B Reg. to R-Bus

Note:

($\overline{\quad}$) indicates negative true signal

I/O REGISTER AND GATE INTERFACE SECTION

As shown in FIGS. **49** A-D, the directly addressable I/O register (address **01**) is a 16 bit universal parallel in/out, serial in/out) register that is connected to the calculator processor by the serial-in S-Bus and the serial-out T-Bus. Information is passed non-inverted from the A or B registers bit serial to the I/O register with the I/O instruction OTX **01.** Sixteen lines connected to the parallel outputs of the I/O register provide data out to the internal input, input-output, and output units and to the external output interfaces. (NOTE: each I/O unit or interface may place only 1 TTL load on the output lines.)

Parallel entry to the I/O register is through 12 party lines connected to the 12 least significant parallel inputs. The input lines are negative true with all input interfaces tying to the lines through open collectors. Care must be taken to insure there is no distrubance to the lines while an interface is inactive. Input information is passed inverted to the A or B register bit serially with the I/O instructions LIX **01** or MIX **01.** (The inversion puts positive true information into the A or B register).

Input information is entered into the I/O register in three ways:

a. Service Request

Entry by the services request method is controlled by a service inhibit flip flop. When the service inhibit flip flop has been cleared with the I/O instruction CLF **01,** a service request may be initiated by returning the $\overline{SSI}$ (Sevice Strobe Input) party line to ground through an open collector on the interface. This signal causes the parallel inputs to be strobed into the I/O register and sends a request for service (QNR) to the microprocessor. The microprocessor prior to receiving a request for service would have been cycling through various instruction paths and checking for a service instruction paths and checking for a service request after execution of each instruction. Upon receipt of a request for service, the processor interrupts the sequence of instructions it was doing and loads an address into the M-Register which contains the starting address of the service routine. At the same time a signal, $\overline{SRA}$ (Service Request Acknowledge), turns off the service inhibit flip flop and also sets the single service flip flop which permits only one service interrupt to the processor per service strobe input. The single service flip flop is reset when the service strobe is removed. All lines from an interface using the service request method for entering information are inhibited when the service inhibit flip flop is set.

b. Return of Channel Flag After Command is Given to an External Peripheral Device.

This method implies the calculator must control the peripheral. That is to say the calculator transmits the indirect address and control enable ($\overline{CEO}$) from the I/O Register and gate interface section to the interface with the expectation of information being returned by the peripheral through the interface to the I/O register. Because of this expectation, only limitd instructions may be performed by the calculator while waiting. The service request method must be inhibited during this wait so that input information is not destroyed by another peripheral using service request.

When a controlled peripheral, its flag and data are processed at the interface. The signal $\overline{CFI}$ (Channel Flag In) causes the loading of parallel data from the interface into the I/O register and clears the control enable flip flop so that the $\overline{CEO}$ signal is removed from the interface. The calculator can interrogate the control enable flip flop with the instructions SFS **01** or SFC **01** to determine when data has been loaded in.

c. Giving the I/O Instruction STF **01.**

The instruction STF **01** as described in (a) sets the service inhibit flip flop inhibiting the service request mode of entry. The STF **01** instruction also causes a parallel load of the input lines into the I/O register.

The output display (address **08**) receives information from the I/O register. A 16 bit word is transferred to the I/O register with the instruction OTX **08.** The address **08** allows the display enable flip flop to be set with the micro-instruction EOW after the 16th bit has been transferred. The display enable flip flop sends a signal $\overline{DEN}$ to the display indicating information is ready in the I/O Register. The display enable flip flop is cleared with the I/O instruction CLF **08.**

The keyboard operates as described below. 7 bit ASCII assigned keycodes are entered into the calculator by an interrupt process. When a key on the keyboard is depressed the keyboard interface card requests service. Input data is stored along with the request for service on the keyboard interface card. The stored signal for service is gated with the Prevent Interrupt signal through an open collector NAND gate onto the Service Request party line (SSI = Low for service). The giving of Service Request causes the I/O register to be loaded. However, input data from the keyboard interface card is not enabled yet. Thus all status and data inputs are high. This indicates to the CPU that a keyboard is interrupting. An OT × 16 instruction is given by the firmware. The select code of 16 enables the gate of the data input lines by a STF 1 instruction and data is loaded into the I/O register. LIA **0** allows data to be taken from the I/O register.

All external peripheral interfaces are indirectly addressed from the four most significant bits in the I/O register. Thus to communicate with an external peripheral, an address (**0000** excluded) must be loaded into the I/O register. Data and status will be loaded at the same time if the peripheral is to act as a receiver. If the peripheral is to act as a transmitter, only the address and status need be loaded. Next, the I/O instruction STC **01** sets the Control Enable Out flip flop. This flip flop sends a signal $\overline{CEO}$ to all external interface slots. The $\overline{CEO}$ signal and the decoded (from the 4 bit address) address allow the interface to command the peripheral. After the peripheral has responded, information given back to the interface by the peripheral is processed to the I/O register in the manner described above under (b) Return of Channel Flag After Command is Given to an External Peripheral Device.

The I/O register and gating control circuit mnemonics are given in the following table:

I/O REGISTER AND GATE BOARD

$\overline{CEC}$	Control Enable Out
$\overline{CFI}$	Channel Flag In
CLF	Clear Flag
CO0, 1,2,3	Code Out
$\overline{DEN}$	Display Enable
DI0, 1,2,3,4,5,6,7	Data In
DO0, 1,2,3,4,5,6,7	Data Out
DRC	Data Register Clock
EBT	Eight Bit Transfer

I/O REGISTER AND GATE BOARD-continued

EOW	End of Word
IOD	I/O Data
KLS	Key Lights Strobe
MCR	Mag Card Reset
$\overline{MFL}$	Mag Flag
MLS	Mag Latch Strobe
PEN	Printer Enable
$\overline{POP}$	Power On Pulse
$\overline{PTF}$	Printer Flag
Q0	Qualifier Bit 0
Q1	Qualifier Bit 1
Q2	Qualifier Bit 2
Q3	Qualifier Bit 3
Q4	Qualifier Bit 4
QFG	Qualifier Flag
QNR	Qualifier Not Request
$\overline{SIH}$	Service Inhibit
SI0, 1,2,3	Status In
SO0, 1,2,3	Status Out
$\overline{SRA}$	Service Request Acknowledge
$\overline{SSI}$	Service Strobe In
STC	Set Control
STP	Stop
STF	Set Flag
T-Bus	T-Bus
TTO	T Bus to Output

NOTE:

($\overline{\quad}$) indicates negative true signal

As shown in FIG. **53**, when addressing a peripheral device, bits loaded into the 4 most significant locations in the I/O register from the CPU constitute the peripheral address code. As part of the output party line system the address code is routed to all I/O interface slots. Each I/O interface card decodes the 4 line address code to a unique single line for use on that particular I/O card. The binary codes **10** through **15** have been reserved for dedicated peripheral addresses which are used by dedicated keys (from the keyboard) and dedicated I/O drivers. Binary codes **1** through **9** are for general use. Code 0 is a non-addressing code and is used in operations that do not involve addressing a specific peripheral. The following table summarizes the address code assignments:

ADDRESS CODE ASSIGNMENTS

ADDRESS	4-BIT CODE	ASSIGNED PERIPHERAL
15	HHHH	TYPEWRITER
14	HHHL	PLOTTER
13	HHLH	
12	HHLL	KEYBOARD & KEYBOARD-LIKE PERIPHERALS
11	HLHH	
10	HLHL	
9	HLLH	GENERAL USE; ONE OF NINE SELECTABLE
8	HLLL	"
7	LHHH	"
6	LHHL	"
5	LHLH	"
4	LHLL	"
3	LLHH	"
2	LLHL	"
1	LLLH	"
0	LLLL	USED ON INTERRUPT I/O INTERFACE CARDS WHEN THE INTERRUPT BECOMES ENABLED

The general usage codes (**1–9**) are decoded outputs from a 4 line to 1 of 10 decoder (SN **7442** for example). It is intended that the codes **1** through **9** be jumper selectable. This would allow the user to select a code for his system peripherals or allow him to use more than one of the same peripheral by selecting different address codes.

Since the I/O register is used to communicate with the internal input, input-output, and output units as well as peripheral devices, a given peripheral's address code will appear randomly in the I/O register address field with there being no intention of expecting the peripheral to respond. Therefore, a second piece of information is necessary for the I/O interface card to form a unique signal which will indicate to the peripheral to respond. This second piece of information is control information and is described hereinafter.

The I/O interface cards contain TTL compatible logic for manipulating control and data from the calculator and/or the peripheral. All I/O interface cards which are intended to be used with the calculator must provide storage either on the I/O interface card or in the peripheral. Thus data being transferred from the calculator to the I/O card must be stored at the instant the peripheral is requested to respond. Likewise data coming from a peripheral must be stored until the calculator accepts it. This requirement is important and must be considered on all compatible interface cards.

The calculator can supply up to 100 ma. maximum at +5 volts to each I/O interface card. Power exceeding this absolute maximum must be supplied by the peripheral.

The following table lists the pin assignments for all I/O lines at the plug-in slots on the calculator back plane, as viewed from the rear of the calculator, left to right.

EXTERNAL I/O INTEFACE PIN ASSIGNMENTS

Bottom		Top	
1		A	
2	+5	B	+5
3	USED	C	USED
4	USED	D	10/20
5	USED	E	USED
6	DI 0	F	DO 0
7	DO 1	H	DO 2
8	DI 3	J	DO 3
9	DI 2	K	DI 1
10	DO 4	L	DI 4
11	DO 5	M	DI 5
12	DO 6	N	DI 6
13	DO 7	P	DI 7
14	SO 0	R	SI 0
15	SO 1	S	SI 1

-continued

16	SO 2	T	SI 2
17	SO 3	U	SI 3
18	CO 0	V	CO 1
19	CO 2	W	CO 3
20	$\overline{SSI}$	X	$\overline{SIH}$
21	$\overline{CEO}$	Y	$\overline{CFI}$
22	⊥	Z	$\overline{STP}$

The chart below lists all I/O lines with brief definitions and specifications and FIG. **50** shows the source and relative relationship of the I/O lines. The output address data lines (Co **0–3**) transmit the address code along the party lines to all interface slots. These lines will go high and low according to information being shifted in or out of the I/O Register. At anytime a peripheral is addressed the lines will become steady 1 instruction time (8 μs) before control information is passed to the I/O interface card or before data or status is taken from the I/O interface card and will remain constant until the control information is removed. After the control information is removed, the state of the I/O lines become unpredicatable until the next addressing takes place. Address data coming to the I/O interface card is positive true and each interface may place 1 TTL load on each address line.

tional information to a peripheral. The logic state is positive true. One standard TTL load may be placed on each output data status line. (Special drivers, fast data transfer, and interrupt do not make use of SO **3**).

The input data lines (DI **0–7**) transmit input data in 8 bit bytes to the 8 least significant bit positions of the I/O register (Locations **0** thru **7**) from the I/O interface card. Each Data In line has a **1K** pull up resistor to +5 volts and under th party line system must be driven low for a logical 1 from open collector gates on each addressed I/O interface card. The logic state is negative true.

The input data status lines (SI **0–3**) receive information from the I/O interface cards and transmit it to location **8** through **11** in the I/O register. Each line has a **1K** pull us resistor to +5 volts. These lines are used to provide additional information to the calculator about the state of a peripheral. The logic state is negative true.

The negative true Device Ready output line ($\overline{CEO}$) transmits a control signal, which when combined with an address code will initiate a peripheral response on the addressed I/O interface card. Device Ready is controlled by the I/O interface driver and therefore may look different depending upon the driver. For example, when the calculator wishes to transmit data to the I/O interface card or to initiate a peripheral response prior to receiving data from the peripheral, the calculator

I/O Line Specification Chart

	Name of Line	Line Definition	Direction	Load/Loading	Voltage High	Voltage Low	No. of Lines
1	Address Data (CO 0-3)	Transmits a 4 bit address from the I/O Register to be recognized by an interface card. (Data = High)	Out	1 TTL (1.6ma) allowed per interface.	2.4v	.4v	4
2	Device Ready $\overline{CLO}$	Indicates calculator is ready for information interchange with an addressed peripheral. (Active State = Low)	Out	1 TTL (1.6ma) allowed per interface.	2.4v	.4v	1
3	Device Request ($\overline{CFI}$)	Acknowledges receipt of data by a peripheral from the calculator or indicates data is to be input to the calculator. (Active Stage = Low)	In	Loading of 6.6 ma to the interface card.	1k resistor to + 5v use open collector	Must be driven below .4v.	1
4	Halt Status ($\overline{STP}$)	Indicates stop key has been depressed. (Active State = Low)	Out	1 TTL (1.6ma) Allowed/interface.	2.4v	.4	1
5	Input Data (DI 0-7, SI 0-3)	Receives input data to I/O register. (Data = Low)	In	Loading of 6.1ma	1k Res. To + 5v	Driven .4v	12
6	Output Data (DO 0-7, SO 0-3)	Transmits Data from the I/O register. (Data = High)	Out	1 TTL (1.6 ma) Allowed/interface.	2.4v	.4v	12
7	Prevent Interrupt ($\overline{BIH}$)	Indicates data cannot be entered under service request. (Interrupt) (Active State = Low)	Out	1 TTL (1.6 ma) Allowed/interface	2.4v	.4	1
8	Service Request (Lo) ($\overline{SSI}$)	Indicates a CPU interrupt is to take place to allow data to enter. (Active State = Low)	In	Loading of 6.6ma	1k Res. to + 5v	Driven .4	1

The output data lines (DO **0–7**) output data from the A or B accumulator in 8 bit bytes from the 8 least significant locations in the I/O register to all interface card slots. The logic state is positive true (Data = 1 = High). Each interface card may place **1** standard TTL load on each data line.

The output data status lines (SO **0–3**) output status data from the A or B accumulator and are driven from the next four locations above the data out positions in the I/O register. (DO positions = **0** thru **7**; SO positions = **8** thru **11**). These lines are used for sending addi-

causes the Device Ready output line to go low and stay low until the peripheral response is over and the calculator receives the signal Device Request ($\overline{CFI}$) from the I/O interface card. The Device Ready flip-flop always receives a clear signal whenever the I/O register completes a parallel load.

The Device Request party line $\overline{CFI}$ when driven low from an open collector gate on the I/O interface card will cause the loading to parallel input information into the 12 least significant locations of the I/O register. The active state of the line is low (negative true).

The peripheral flag, indicating to the I/O interface card the peripheral has received data/control or is ready to input data, is gated through an open collector nand gate onto the Device Request ($\overline{CFI}$) party line. The open collector gate is enabled by the I/O interface card's address and Device Ready ($\overline{CEO}$). The Device Request line is pulled up inside the calculator by a 1K resistor to +5 volts.

The Device Request ($\overline{CFI}$) signal must stay low until Device Ready ($\overline{CEO}$) has been cleared (goes high). At this time data transfer has terminated and peripheral's flag and control must be cleared in preparation for the next pass. Since a parallel load in the I/O register causes the Device Ready flip-flop to receive a clear signal, when a Device request ($\overline{CFI}$) is entered, a parallel load takes place and afterward Device Ready ($\overline{CEO}$) is cleared. The calculator uses Device Request in its general mode of data transfer.

The Halt Status output line ($\overline{STP}$) is a line that goes low when the STOP key on the calculator is depressed. It will stay low for the duration of the key depression. One standard TTL load may be placed on this line by each I/O interface card.

The Prevent Interrupt output line ($\overline{SIH}$), when low indicates to the I/O interface card that a request for service must not be given to the calculator. One standard TTL load may be placed on this line by each I/O interface card.

The Service Request (Lo) line ($\overline{SSI}$), when driven low causes the loading of parallel input information into the 12 least significant locations of the I/O register and causes a CPU interrupt for service. The peripheral's request for service is gated with the Prevent Interrupt ($\overline{SIH}$) line onto the Service Request party line through an open collector nand gate. A 1K pull-up resistor to +5 volts is connected to the line inside the calculator.

The general format for all data transfer consists of 8 bit parallel bytes. Other data formats are handled by specially developed drivers, such as the ROM plug-in module employed for driving the typewriter.

The state of a peripheral is generally checked before attempting an output. This is done by first inhibiting the interrupt system. The address of the I/O interface card is shifted into the I/O register. The decoded address code enables the open collector gates on the I/O interface card. The status of the peripheral is passed to the Status In lines and loaded into the I/O register with a I/O instruction issued by the calculator. The I/O register information is transferred to the A or B accumulator and processed. If the peripheral is ready, the output data word consisting of the address code, output status (if necessary) and the eight bit data byte is formed in the A or B accumulator. The output data word is transferred to the I/O register after which the Device Ready ($\overline{CEO}$) flip-flop is set. The I/O interface card receives the data, address code and Device Ready and a peripheral response is initiated. The calculator interrogates the state of the Device Ready flip-flop to determine when the I/O interface card has received the information and the peripheral response is done. The peripheral I/O interface card signals the calculator it is done by transmitting the Device Request ($\overline{CIF}$) signal to the calculator. The output waveforms are shown in FIG. **54.**

Before inputing data from the I/O interface card it is necessary to determine if the peripheral has responded and is ready to input data. After a peripheral response has been initiated, as described previously, the calculator waits for the Device Request ($\overline{CFI}$) which loads the data into the I/O register and clears the Device Ready ($\overline{CEO}$). The calculator checks the state of Device Ready and when it goes false ($\overline{CEO}$ = HIGH), the calculator knows data is present in the I/O register and proceeds to shift it into the A or B accumulators for processing. The input waveforms are shown in FIG. **61.**

When blocks of data are to be transferred between a peripheral and the calculator, the interrupt is turned off, and transfer rates as high as 100,000 bits/sec may be possible. Before either input or output of a block of data can start, it is necessary for the calculator to check the status of the peripheral to see if it is turned on and ready. The address locations of the I/O register will remain unchanged during the block transfer. A single I/O instruction shifts the 8 bit byte of data from the 8 least signficant locations in A or B to the 8 data locations in the I/O register; gives: Device Ready ($\overline{CEO}$ goes low) 120 nanoseconds after the shift is completed; and shifts the 8 most significant bits in A or B to the 8 least significant locations in A or B in preparation for the next transfer. (Note the address and status field in the I/O register are not disturbed in the shifting). Device Ready stays true (low) until the peripheral has received the data and is ready for more. The I/O interface card then returns Device Request ($\overline{CFI}$) to the calculator. The receiving the Device Request ($\overline{CFI}$) to the calculator causes loading of the parallel input party lines into the input status and input data locations of the I/O register, and clears the Device Ready signal ($\overline{CEO}$ goes high). The logic sense of Device Ready is observed by the calculator and when it goes false ($\overline{CEO}$ = HIGH) the CPU proceeds to output the next 8-bit byte of data.

If the output I/O interface card is not returning information on the input lines all input lines will be high when the loading, described in the preceeding paragraph, takes place. Therefore, if at the beginning the code in the output status field is being used by the I/O interface card and must remain something other than all high it will be necessary for the I/O interface card to receive the output status from the calculator and return it back to the status inputs so that when Device Request occurs the status field does not get changed in the I/O register.

Input: After determining if the peripheral is ready to start transferring a block of data the calculator turns off the interrupt and shifts the address code into the I/O Register. (The address code remains unchanged during the block transfer). The Device Ready is given ($\overline{CEO}$ = Low) to the calculator when the 8-bit data byte is ready for input. The Device Request signal causes the input data and status to be loaded into the I/O register and causes Device Ready to go false ($\overline{CEO}$ = High). The calculator by checking when Device Ready goes false knows the data has been loaded. A single I/O instruction shifts the 8-bit data byte from the I/O register into the 8 most significant locations in the A or B accumulators (Shifting the previous information in A or B 8 places to the right) and causes Device Ready to go true ($\overline{CEO}$ = Low) 120 ns after the last bit has been shifte into A or B. As before if output status is to be retained on the I/O interface card it must be returned to the I/O register upon each input data transfer. Wave forms illustrating high speed operations are shown in FIGS. **56** and **57.**

The calculator software makes use of the interrupt system in two different manners. The first is for remote keyboard like peripherals.

These are those peripherals which logically resemble the calculator keyboard. Only 7-bit ASCII assigned keycodes are recognized by the calculator. The interrupt takes place by the peripheral indicating to the I/O interface card that a request for service exists. Input data must be stored along with the request for service on the I/O interface card or in the peripheral itself. The stored signal for service is gated with the Prevent Interrupt signal through an open collector NAND gate onto the Service Request party line ($\overline{SSI}$ = Low for service). The giving of Service Request causes the I/O register to be loaded. However, input data from the I/O interface card is not enabled yet. Thus all status and data inputs are high. This indicates to the CPU that a keyboard-like peripheral is interrupting and address code 12 is shifted into the I/O register. The decoded address 12 on the I/O interface card enables the gates to the data in lines and data is now loaded into the I/O register. After the data has been taken from the I/O register address 12 is again put into the I/O register and Device Ready is given as a 360 nanosecond pulse to clear all stored keyboard-like requests for service. This implies all keyboard-like peripherals must be user controlled such that only one interrupt at a time is taking place. The second is nonkeyboard-like peripherals.

These peripherals will output or enter standard ASCII codes for data by using a special ROM (other ROMs may be developed to handle different codes). When a request for service is given to the I/O interface card by a peripheral the request and all data must be stored until serviced by the calculator. The interface card may have any of 9 addresses (1 thru 9). The stored request for service is gated with Prevent Interrupt through an open collector NAND gate onto the Service Request party line. At the time Service Request is recognized address 0 is gated with the stored request for service through an open collector onto an input data or status line which corresponds with the address of the I/O interface card. For example, Data In 0 which is the 1st position in the I/O register represents card address 1, and 2nd position is card address 2, etc. When the I/O register is loaded as a result of the Service Request the interrupting I/O car's address is loaded into the I/O register and Prevent Interrupt enabled ($\overline{SIH}$ = Low). The contents of the I/O register are processed by the CPU which then shifts the interrupting card's address into the I/O register. The address enables the gates to the data-in lines and data is loaded into the I/O register. After the data is processed by the CPU the interrupting card's address is shifted from the CPU into the I/O register and a 360 nanosecond Device Ready pulse ($\overline{CEO}$ = Low) given to clear the stored request for service on the I/O interface card, after which the Prevent Interrupt is disabled and the next interrupt allowed to take place. Under this system, multiple interrupts may take place without consequence. Each will be serviced in turn from low to high address position. An interrupting peripheral may also interrupt to request output data from the I/O register. The interrupting process is the same as above except the calculator transmits data rather than receives data. FIG. 8 shows waveforms illustrating the interrupt.

The following table lists the general I/O instruction set and the associated codes.

I/O INSTRUCTION SET

NAME	INSTRUCTION EXECUTION TIME	INSTRUCTION CODE 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STF	9 μs	H	H	H	H	—	H	L	H	H	H	H			SELECT CD		
CLF	9 μs	H	H	H	H	—	H	H	H	H	H	H			"		
SFC	9 μs	H	H	H	H	—	H	H/C	H	H	H	L			"		
SFS	9 μs	H	H	H	H	—	H	H/C	H	L	H	L			"		
CLC	9 μs	H	H	H	H	—	H	H/C	H	L	H	H			"		
STC	9 μs	H	H	H	H	—	H	H/C	H	H	L	L			"		
OT*	15 μs	H	H	H	H	A/B	H	H/C	L	L	H	H			"		
LI*	15 μs	H	H	H	H	A/B	H	H/C	L	H	L	H			"		
MI*	15 μs	H	H	H	H	A/B	H	H/C	L	L	L	H			"		

The following describes the function of each I/O instruction with the 5 allowable select codes.

STF <SC> — Set the flag. STF is a 240 nanosecond positive true pulse which accomplishes the following with the various select codes.

STF 00 — Not used by the calculator.

STF 01 — a. Sets the "Service Inhibit" flip-flop to the true state ($\overline{SIH}$ = Low; interrupt not allowed).
b. Causes parallel input data and status to be loaded into the I/O register.

STF 02 — Generates a 240 nanosecond positive true MCR pulse.

STF 04,08,16 — Not used by the calculator.

CLF <SC> — Clear the flag. CLF is a 240 ns positive true pulse which accomplishes the following with the various select codes.

CLF 00 — Not used by the calculator

CLF 01 — a. Clears the "Service Inhibit" flip-flop to the false state. ($\overline{SIH}$ = High; interrupt allowed.)
b. Loads address locations in I/O register gister with 0's. (0 = Low)
c. Clears "Device Ready" flip-flop ($\overline{CEO}$ = High).

CLF 02 — Clears MCR flag flip-flop.

CLF 04 — Clears PEN flip-flop (PEN = Low).

-continued

CLF 08	Clears DEN flip-flop ($\overline{DEN}$ = High).
CLF 16	Generates a 240 nanosecond positive true KLS pulse
SFC <SC> H/C	Skip if flag clear. SFC is a 240 ns positive true pulse which accomplishes the following with the various select codes. If C is given a 240 nanosecond CLF pulse is given after SFC.
SFC 00	Causes the next instruction to be skipped if the STOP key has not been depressed.
SFC 01	Causes the next instruction to be skipped if Device Ready is true ($\overline{CEO}$ = Low).
SFC 02	Causes the next instruction to be skipped if the MCR flag flip-flop is clear.
SFC 04	Causes the next instruction to be skipped if the PEN flip-flop is clear. ($\overline{PEN}$ = Low).
SFS <SC> H/C	Skip is flag set. SFS is a 240 nanosecond positive true pulse which accomplishes the following with the various codes. If C is given then a 240 nanosecond CLF Pulse is issued after SFS.
SFS 00	Causes the next instruction to be skipped if the STOP key is depressed.
SFS 01	Causes the next instruction to be skipped if "Device Ready" is false ($\overline{CEO}$ = High).
SFS 02	Causes the next instruction to be skipped if the MCR flag flip-flop is set.
SFS 04	Causes the next instruction to be skipped if the PEN flip-flop is set (PEN = High).
$\overline{CLC}$ <SC> H/C	Clear Control. $\overline{CLC}$ is a 240 nanosecond negative true pulse and is not used by the calculator. If C is given then a 240 nanosecond positive true CLF pulse is given after $\overline{CLC}$.
STC <SC> H/C	Set the Control. STC is a 240 nanosecond positive true pulse which accomplishes the following with the various select codes. If C is given a 240 nanosecond CLF pulse is issued after STC.
STC 00	Not used by the calculator.
STC 01	Sets the "Device Ready" flip-flop ($\overline{CEO}$ = Low).
STC 02	Generates a 240 nanosecond positive true MLS pulse for the magnetic card reader.
STC 04, 08, 16	Not used by the calculator.
OTX <SC> H/C	Output A or B causes data bits from A or B to be shifted to the I/O register and accomplishes the following with the various select codes. If C is given, a 240 nanosecond CLF pulse is given after OTX is executed.
OTX 00	The 8 least significant bits in the A or B register are shifted non-inverted to the 8 least significant locations in the I/O register, and 120 nanosecond after the 8th shift the "Device Ready" flip-flop is set ($\overline{CEO}$ = Low). The 8 most significant bits are shifted right 8 places and the least 8 significant bits are recirculated to the 8 most significant locations in the A or B registers. The 8 most significant bits in the I/O register are untouched.
OTX 01	Sixteen bits from the A or B register are shifted non-inverted to the I/O register. The data in A or B recirculates.
OTX 02	Not used by the calculator
OTX 04	Same as OTX 01 and in addition, 120 ns after the 16th bit has been shifted nanoseconds printer enable flip-flop is set
OTX 03	Same as OTX 01 and in addition, 120 nanoseconds after the 16th bit has been shifted the display enable flip-flop is set.
OTX 16	Same as OTX 01 and in addition, 120 nanoseconds after the 16th bit has been shifted the 240 nanosecond KLS signal is generated
LIX <SC> H/C	Load into A or B. Loads data bits from the I/O register into the A or B register and accomplishes the following with the various select codes. If C is given, a 240 nanosecond CLF pulse is given after LIX is executed.
LIX 00	The eight least significant bits in the I/O register are shifted inverted to the eight most significant locations of A or B, and 120 nanoseconds after the 8th shift the "Device Ready" flip-flop is set ($\overline{CEO}$ = Low). A or B is shifted right eight places as

-continued

the I/O register data comes in. The 8 most significant bits in the I/O register are untouched.

LIX 01 The 16 bits of the I/O register are transferred inverted to the A or B register. Data in the I/O register is lost.

LIX 02, 04, 08, 16 Not used by the calculator.

MIX <SC> H/C Merge into A or B. Merges data from the I/O register into A or B registers and accomplishes the following with various select codes. If C is given, a 240 nanosecond CLF pulse is given after MIX is executed.

MIX 00 The eight least significant bits in the I/O register are merged with the eight least significant bits of the A or B register and shited to the 8 most significant locations of A or B; 120 nanosecond the merge takes place the Device Ready flip-flop is set (CEO = Low). A or B shifts right 8 places as the data is merged and shifted to the most significant locations. The 8 most significant bits of the I/O register are untouched.

MIX 01 The 16 bits of the I/O register are merged with the 16 bits of the A or B register and contained in the A or B register.

MIX 02, 04, 08, 16 Not used by the calculator.

Examples of various drivers which transfer data are given below.

Example 1: Typical Subroutine to Get Status of I/O Device.

Calling Sequence:

```
        LDB Select Code
        JSM Stat
Stat    STF 1       Turn off the interrupt system.
        OTB 1       Load I/O register with select code.
        STF 1       Load I/O register with status of I/O
                    device.
        LIA 1       Load A-Register with status information.
        CLF 1       Turn on interrup
        RET         Return.
```

Example 2: Typical Subroutine to Output an 8 bit character.

Calling Sequence:

```
        OTA 1       Output 16 bits to the I/O register.
        STC 1
        SFS 1       Loop until I/O flag is set by the
        JMP *-1       output device.
        CLF 1
```

Example 3: High Speed Output Where the Calculator is Faster than Output Device.

Calling Sequence:

```
        ST* I       -(Number of 16 bit words to be output) + 1
        ST* J       Address of first word in the array.
        LDB SC      Select Code
        JSM OUT2
OUT2    JSM STAT    Get status of output device
        RAR 9         and position it.
```

Example 4A: Typical Subroutine to Input an 8-bit Character.

Calling sequence is:

```
        LDB Select code
        JSM In
        . . . Return is made with the data in the A Register.
In      STF 1       Turn off interrupt system
        OTB 1       Load I/O register with the select code
        STC 1, C    Pulse the flag & turn interrupt system on
        JSM STAT    Get status off the input device
        RAR 9         and position it.
        SAP *-2, C  If device is busy then continue to loop
        SAR 7         else position data bits
        RET         Return.
        SAP OUT2    If device is busy, continue to loop
        STF 1       Turn off interrupt system.
        OTB 1       Output select code
        LDB 1
                    B <-- Counter for number of words to be output
        LDA J, I    Load next data word
        SEC *+1, C
                    E <-- 0
        OTA 0       Output 8 bits from A
        SFS 1       Loop until device sets
        JMP *-1       flag.
        SEC *-3, S
                    If E=0 and E <-- J then loop to output last
                      8 bits
        ISZ J       Increment array address pointer
        RIB *-7     Increment count and loop if not finished.
        CLF 1       Turn on interrupt system
        RET         Return
```

Example 3B: If the Output Device is Faster than the Calculator then n Fewer Instructions can be Used.

```
        OTA 0       Output first 8 bits
        OTA 0       Output second 8 bits.
```

-continued

```
Example 5A:   High speed input where the calculator is faster than
              the input device.
              Calling sequence:
                   ST* I      -(Number of 16 bit words to be input) + 1
                   ST* J      Address
                   LDB SC     Select code
                   JSM In2
In 2               JSM STAT   Get status of input device
                   RAR 9        and position it.
                   SAP In2    If device is busy, continue to loop
```

All output I/O interface cards which are to be fully interchangeable with both the present and other calculators must have storage either on the I/O interface card or in the peripheral to which information is being transmitted.

Blocks (A) and (B) are the storage latches which store information coming from the I/O register. When the output of gate (C) goes high, data is latched; when low, the outputs of the latch track the inputs. Gates (D) decode the address code (14 = 1110) and pass it positive true to gate (E). Device Ready ($\overline{CEO}$) is also passed positive true to gate (E). Gates (H) are open collector and pass status and Device Request ($\overline{CFI}$) onto the input party lines.

An example of a calculator output would be: output the address 14 which enables status gates (H) and see if the power is on. If on, output address, status, and data to gates (A), (B), and (D). The output of (C) is low allowing data and status to pass. Next give Device Ready ($\overline{CEO}$ = Low); this enables flip-flop (G), clocks flip-flop (F) which causes (A) and (B) to latch, and sends control to the peripheral. The peripheral acknowledges receipt of control by returning FLAG (FLAG = High) in a busy state this continues to keep (A) and (B) latched and clears control flip-flop (F). When the peripheral is done acting, the FLAG is returned to the not busy state (FLAG = Low) which clocks flip-flop (G) and causes output at (C) to go low enabling (A) and (B). The output of (G) drives the $\overline{CFI}$ gate which has been enabled from (E) and $\overline{CFI}$ goes low. $\overline{CFI}$ is received by the calculator which responds by returning $\overline{CEO}$ high. This causes the output of (E) to go low, clearing flip-flop (G) and returning $\overline{CFI}$ high. This completes 1 output cycle.

All input I/0 interface cards which are to be fully interchangeable with both the present and other calculators must have storage either on the I/O interface card or in the peripheral from which information is being received.

Block (A) is used to store information coming from the peripheral. (B) stores status coming from the I/O register which may be needed by the peripheral. The output tracks the input whenever the enable on the latch is low. Block (C) decodes the address code into one of 10 addresses which are jumper selectable. An example of a calculator input would be as follows: the address code would be decoded by (C); the calculator would load status through the open collector input status gates (D). If the peripheral is on and ready, the address code and output status (if necessary) would be sent to (B) and (C). The decoded address is passed, positive true, to gate (E). The enable at (B) is low so that status is passed to the peripheral. The Device Ready is given ($\overline{CEO}$ = Low) and comes to (E) positive true. The output of (E) clocks flip-flop (F) through gate (H). The output of (F) gives control to the peripheral and also enables (A) to receive data. The peripheral responds in a busy state (FLAG = High). When data is ready to be input the FLAG is driven low. Data is latched when the FLAG goes low in (A). Also when FLAG goes low, (G), having been enabled by the output of (H), is clocked driving (J) from its Q output. (I) is enabled by the output of (H) and so $\overline{CFI}$ is driven low. Data is loaded into the I/O register from open collector gates (I) and $\overline{CEO}$ driven high as a result of the calculator receiving $\overline{CFI}$. This clears flip-flop (G) and disables the input gates (I) completing an input cycle.

FIG. **59** illustrates the logic required on an I/O interface module to interface the calculator with an external X-Y plotter.

FIG. **60** illustrates the logic required on an I/O interface module to interface the calculator with an external line printing unit.

FIG. **61** illustrates the logic required on an I/O interface module to interface the calculator with an external modem for transmitting and receiving information via telephone lines.

FIG. **62** illustrates the logic required on an I/O interface module to input or output any eight-bit code.

A power preset circuit is employed on interface modules using the interrupt system to prevent an interrupt when the peripheral power is turned off or on. This can usually be done by sensing the peripherals' +5 volts and presetting when the voltage drops below 3 to 4 volts.

An example of a calculator interrupt would be as follows: (B) may be clocked at any time storing the data is (E) and (F). The calculator enables the interrupt to take place by making Prevent Interrupt false ($\overline{SIH}$ = High) and outputting address 0 to decoder (L). (G) is enabled when $\overline{SIH}$ goes high through gate (M) causing $\overline{SSI}$ to be driven low. The calculator responds by loading the I/O register. Gates (H) are inhibited by gates (N) and (J) and gate (K) is enabled because of address 0, thus DI0 is the only true signal loaded into the I/O register. The calculator interprets this to mean the I/O interface card at address 1 has caused the interrupt. The calculator outputs address 1 to the decoder which enables gates (H) with (N) and (J) and then loads the data. After the data is stored the calculator outputs address 1 and sends Device Ready ($\overline{CEO}$ = Low) as a 360 nanosecond pulse which is used to clear (B) through gates (O) and (D). This completes an input cycle.

KEYBOARD INPUT UNIT

The keyboard input unit is shown in FIG. **63** and is designed around a 128-position matrix. Each position is scanned sequentially. When a key is depressed, the counter, which drives the scanner, is stopped. The address of the 7-bit counter corresponds to the ASCII code of the key depressed. The entire scan period is 4

milliseconds. Two key rollover is incorporated in this design.

When a key is depressed, $\overline{SSI}$ is generated unless inhibited by either $\overline{SIH}$ or KLS. As soon as the service request is acknowledged, the CPU will give a select code of 12 which will gate the ASCII keycode onto the data lines.

Another feature of this keyboard is automation repeat. If a key is depressed for more than 1.5 seconds, the keycode will be entered repeatedly at a rate of 15 entries per second until the key is released.

OUTPUT DISPLAY UNIT

Referring to FIGS. 72A–B and 73A–B, there is shown the hardware associated with the calculator display. The display comprises a single register **400** of 32 alphanumeric characters, each character position of which is a seven row by five column matrix of light emitting diodes (LED). In addition to the display hardware illustrated, the complete calculator display system comprises a firmware display routine which is detailed on page 35 of the basic system firmware listing and an I/O register shown in FIGS. 49A–D. The firmware display routine delivers a 16-bit word to the display circuitry. Four of these bits **402** and **404** are decoded into one of 16 character positions. The remaining 12 bits **422** are two six-bit ASCII coded characters which are alternatively decoded in ROM **430** into their row and column information. A seven-bit data latch **432** is provided to allow the ROM **430** to decode both the *n*th and the (n = 16)th characters which are displayed simultaneously.

An assymmetrical clock circuit **426** drives input gating circuitry **436** and counter-decoder circuitry **434** to decode the row information of the *n*th and the (n +16)th characters. Protection circuitry **428** is provided to blank the display in case of failure of either logic signal $\overline{DEN}$ **424** or CA0 **438.**

Three-bit column data **410** and three of the four-bit character position bits **402** are applied to a one-of-four decoder circuit **420.** The output of circuit **420** is fed to the column drivers **418.** Four columns eight emitting diodes **400** are driven by each of the 40 column driver. Thus, all 160 columns of the display can be scanned. The remaining character position bit **404** enables alternate sections of row drivers **416** to complete the one-of-16 character position decoding.

MAGNETIC TAPE CASSETTE READING AND RECORDING UNIT

The magnetic tape cassette reading and recording unit is shown in the block diagram of FIG. **64** and in the detailed schematic diagram of FIGS. **65–69.** Operation of the magnetic tape cassette reading and recording unit is largely automatic. It is only necessary to specify the type of operation to be performed and the limits desired. The commands for doing this may either be entered directly from the keyboard input unit or as part of a program. The calculator then determines the necessary commands required to cause the magnetic tape cassette reading and recording unit to perform the desired operation.

Several modes of operation are possible. Secure and unsecure programs, data, and sets of user definable key definitions may be recorded on the magnetic tape and subsequently loaded back into the calculator.

Referring now to FIGS. **64** and **65,** the operation of the interface portion of the magnetic tape cassette reading and recording unit will be described.

I. Address Decoder

IC **6** and IC **7** are used to detect a select code of **10** when the cassette is accessed. When a select code of **10** is detected, and the control line $\overline{CEO}$ is true the cassette is enabled.

II. Enabling Input Data Bits

The open collector nands in IC **2** and IC **3** allow the input data bits (ID **0** through ID **7**) to be gated onto the calculator input bus when the card is selected by the address and control line as described in Section I.

III. Enabling Input Status Bits

The open collector nands in IC **1** allow the input status bits (IS **0** through IS **3**) to be gated onto the calculator input bus when a select code of **10** is given and the control line is true.

IV. Flag

D type flip-flop **10** is the flag flip-flop. It is held cleared except when the I/O card is selected as described in Section I. When the I/O card is selected the flip-flop is set by the trailing edge of the flag pulse from the cassette. In addition, the flag flip-flop is held preset (at pin **5**) when the cassette door is open, clear leader is detected, or an interrupting character is read. A simultaneous preset and clear results in a true output from the flip-flop. The flag flip-flop signal is gated to the calculator ($\overline{CIF}$, pin **2-***m*) by the open collector nand gate only when the I/O card is selected.

V. Interrupt

D-type flip-flop **13** is the interrupt flip-flop. It is held cleared whenever the I/O card is selected as described in Section I. The interrupt is allowed only if the cassette is in control mode (pin **2-3**). An interrupt signal comes whenever the cassette goes onto clear leader, the cassette door is opened, or an interrupting character is read. An interrupting character is a character with data bits **2, 3, 4, 5** all ones, read when the cassette is in control mode. When the signal Service Interrupt Inhibit is false ($\overline{SIH}$ = **1,** pin **2-1**) the interrupt signal is gated through IC **4** pin **6** and **8** to the calculator. The data bit SI **1,** (Pin **1-4**) is pulled low to tell the calculator the address of the peripheral which interrupted.

Referring to FIGS. **64, 66A–14** B, and **66'** the operation of the control logic portion of the magnetic tape cassette reading and recording unit will be described.

I.1 Power on Preset

Transistor Q1 and diodes CR1 and CR2 are part of the power on circuit which makes sure that the cassette powers up in the proper modes.

I.2 Control Signal

The control signal (YCNT on pin 11) comes from the Interface card as a positive true signal telling the Logic card to process the command on the output lines. The positive going edge of the control signal fires the one-shot (IC 7) giving a 300 nsec. control Pulse used to strobe storage elements.

I.3 Decode and Storage of Commands

The 1-of-10 open collector decoder (IC **19**) is strobed on its D input by the control pulse. (See Section I.2) On the A, B, and C inputs of the decoder are output status bits OS0 through OS2. Those output status bits contain commands for the transport as follows:

OS2 (Write)	OS1 (Reverse)	OS0 (Fast)	Command
0	0	0	Read-Forward-Slow
0	0	1	Read-Forward-Fast
0	1	0	Read-Reverse-Slow
0	1	1	Read-Reverse-Fast
1	0	0	Write-Forward-Slow
1	0	1	STOP
1	1	0	Continue Present Command
1	1	1	Continue Present Command

Output status bit 3 (OS3) indicates if the cassette should be in control mode or data mode. Control mode is explained in Section I.12.

If when the decoder is strobed, the command on the output status lines is other than STOP or CONTINUE, the open collector wired-or decoded outputs **0** through **4** are low for the width of the control pulse. This wired-or output pulse strobes the command on the output status bits into the Quad Latch, **IC20**, and preset the run flip-flop **15-2**. The command on the run lines are gated off the logic card to the motor control card, and cassette motion begins.

If the command is STOP, the decoder output **5** (pin **6**) goes low, which clears both the run flip-flop (**15-2**) and the Quad Latch (**IC20**). If the command is CONTINUE, both the Quad Latch and the run flip-flop remain unchanged.

The run flip-flop and the Quad Latch are also cleared by power on preset.

I.4 Run Flip-Flop

The run flip-flop (**1502**) is preset by the issuing of a command as described in Section 6.3. It is cleared by an interrupt signal from the I/O card, a stop command, power on preset, or a cassette not being in place. The run flip-flop is also cleared by the clear leader signal on its clock input, whenever the cassette runs into clear leader. Since the clock input of the flip-flop is edge sensitive, the transport is stopped only when it first goes onto clear leader, and tape motion is allowed by the issuing of a new command even when on clear leader.

The Schmidt-triggered gates **1-1** and **1-2** are used to filter noise on the interrupt, clear leader, and cassette in place lines.

I.5 Rewind Mode

D type flip-flop **15-1** is the rewind flip-flop. It is held preset as long as the rewind button is held in. It is cleared by clear leader detection, cassette not in place, power on preset, stop command, or on interrupt from the I/O card. The rewind flip-flop is also cleared through its clock input, whenever a new command is issued. A simultaneous preset and clear results in the Q output of the flip-flop being true. When the transport is stopped (flip-flop **15-2** cleared) and the rewind flip-flop is set the output of gate **16-3** will be true, which will force the NOR gates **17-1**, **17-2**, and **17-3** to 0 outputs. These outputs will cause the tape to be rewound in high speed reverse to clear reader, unless the rewind flip-flop is cleared as described above.

I.6 Transport Status

Input status bits; YIS3 (pin F), YIS2 (pin 6), YIS1 (pin E), and YIS0 (pin 5), reflect the status of the transport as follows:

YIS3 — Negative true, cassette in place

YIS2 — Positive true, clear leader detected

YIS1 — Negative true, writing on cassette permitted

YIS0 — Positive true, control mode

I.7 Internal Clock

The internal clock used for writing data is generated by the two one-shorts **IC6** and **IC13**. The clock is not symmetrical, but **IC6** is true for 133 μsec and **IC13** is true for 220 μsec. Padding resistor **R10** is then used to adjust the total time of 333 μsec for a 3KC clock rate. The internal clock is enabled only when the cassette command is write as detected at pin **5** of **IC13**.

I.8 Nine Bit Shift Register

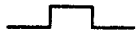
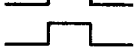
A nine bit shift register is made from **IC3**, **IC4**, and flip-flop **5-2**. In write operations this register is loaded parallel by a pulse from gate **16-1**, and then shifts the data serially to **IC3** pin **10** for writing on the tape. In read mode the data is clocked from the tape, serially, to **IC5** pin **12** where it is serially loaded into the register. From the shift register it is read as parallel data, at the flag, by the calculator.

I.9 Data, Clock Mark Sequence

During write operation a write clock, a write data, and a write mark signal are sent from the logic card. During read operations a read clock, a read data, and a read mark signal are received by the logic card. The data lines are NRZ; clocked by the appropriate clock. When clock pulses and marker pulses are properly sequenced they constitute a character. The sequence is 9 data bits per character, separated by marks.

Mark - 9 data bits - mark - 9 data bits - mark.

BIT-MARK-SEQUENCE CODE

Three types of information	Track	
"1"	A	
	B	
"0"	A	
	B	
Mark	A	
	B	

10 bit character

M D_7 D_6 D_5 D_4 C D_3 D_2 D_1 D_0 M

8 data bits

1 marker

1 control bit

I.10 Divide by 10 Counter During Write

The divide by 10 counter (**IC14**) is used to sequence the clock and mark pulses during write (See section 6.9 for a sequence description). When the counter is at counts **0** through **8** the internal clock pulses generated by **IC6** and **IC13** are gated through gate **18-4** from where it goes off the card as the write clock, and through gate **18-2** to shift the shift register. When the divide by 10 counter is at count **9**, the internal clock pulse is inhibited at gate **18-4**, but is enabled at gate **10-2** from where it passes off the board as a write mark pulse. Since the counter wraps around from count **9** to count **0**, we get the repeating sequence, marker - 9 bits - marker.

Each time the counter goes to **9**, a character is completed and the write mark pulse, in addition to going off the board, is gated through gate **12-3** to make up a flag, telling the calculator that the character has been written. The calculator will then send another parallel character to the shift register and the sequence will be repeated.

Special conditions prevail for the writing of the first of a string of characters. When the transport goes from read to write mode, flip-flops **8-1** and **8-2** are set in their edge sensitive clock inputs. Flip-flop **8-2** presets the counter to count **9**, making sure the sequence begins with a mark pulse. The same flip-flop **8-2** is cleared by this first mark pulse, after which the counter continues its wrap-around sequence. Flip-flop **8-2** being set, inhibits this first leading write mark pulse from passing through gate **12-3** and becoming a flag. Flip-flop **8-2** is cleared by the first write pulse, after which the write sequence continues as normal.

I.11 Divide by 10 Counter During Read

Only a sequence of 9 data bits followed by a mark is to be recognized as a character during read. The divide by 10 counter is used to detect this sequence.

Whenever the transport is stopped, flip-flop **5-1** is held preset. The output of flip-flop **5-1**, presets the counter to a count of 0. Flip-flop **5-1** is cleared with the leading edge of the read data clock and the counter counts on the trailing edge. The read data pulses are counted by the counter and also pass through gate **18-2** to shift the shift register. When the counter has counted 9 read pulses, its count of 9 puts a true signal on pin **13** gate **19-1**. If a read mark pulse comes next, pin **1** of gate **19-1** if true for the width of the read mark pulse. These two signals make up part of the flag signal which tells the calculator a valid character is in the shift register. The trailing edge of the read marker pulse, sets flip-flop **5-1**, which presets the counter to **0** and the sequence is ready to be repeated.

If a marker is read when the counter is at a count other than **9**; no flag is given, flip-flop **5-1** is set, and the counter is preset to **0**. If 10 read pulses come in a row, the counter wraps around to **0** and no flag is given. Thus nothing but 9 read clock pulses, followed by a read mark pulse, generates a flag and is recognized as a character.

I.12 Control Mode and Flags

Each character contains 8 data bits. The middle bit of the 9 bit character is a control bit. If the center bit is a 1, the character is a control character. If the center bit is a 0, the character is a data character. If the transport is in control mode during write, the center bit is loaded as a 1 at IC4 pin **5**, and the character is written as a control character. In data mode the center bit of each character written is 0.

If the transport is reading in control mode, flags are sent to the calculator only when a control character is in the shift register. If the transport is in control mode, and a data character is in the shift register, the read flag is inhibited at IC9-1 pin **2**. This IC9-1 pin **2** is always true, except when the cassette is in control mode and the center bit of the shift register is 0. In data mode, flags are sent to the calculator for both data and control characters.

The flag circuitry on the interface card is trailing edge sensitive, making sure the character is completely written or completely read before the calculator is flagged.

Referring now to FIGS. **64** and **68A–B**, the operation of the read/write portion of the magnetic tape cassette reading and recording unit will be described.

I. General Description

The read/write board has two main functions. In the WRITE mode it encodes bit serial data into two-channel Bit-Mark-Sequence (BMS) data to be fed to the head driver and written on tape.

In the READ mode it decodes the two-channel analog BMS data from the head preamplifier into clock, mark, and bit-serial data pulses from the tape signals.

II. WRITE Mode

When Write Permit is true (YWPT = 1) and Write Command is true (YWTC = 1) then Write Enable is true (YWEN = 1; Q7 on) and writing on the tape is allowed. The 3-input NAND gates of IC2 are used to encode the bit-serial data (YWDT) into BMS. Logic 1's (YWDT = 1) are written on Channel A. Logic 0's (YWDT = 0) are written on Channel B. A mark is written when Write Mark is true (YWMK = 1). The data and marks are clocked in my means of the Write Clock (YWCL).

III. READ Mode

The two amplifiers of IC6 and their associated circuitry comprise two threshold detectors that convert the analog signals from the head (ARA, ARB) to digital signals. The amplifiers switch between their positive and negative saturated states. The thresholds are adjusted so that the amplifiers switch states when the analog signal is approximately 30% of its peak value measured near BOT, moving forward.

The head signal amplitude is proportional to tape speed. Therefore, the thresholds are adjusted high for fast tape motion and low for slow tape motion by switching Q1 and Q2 with the Fast Command (NFTC) line. The positive and negative references for the thresholds are derived from the 12 volt power supplies.

The two channels of digital data are gated to the decode circuitry through IC1. The decoder consists of IC3, IC5, IC8, and IC9.

IC8 is a 0.5 μs one-shot that is fired once for each data bit or mark that is gated in from the threshold detectors. This one-shot pulse comprises a read clock signal that is gated to NRCL by IC5 when the bit associated with the pulse is not a mark.

IC9-1 is a flip-flop that serves three functions. First, it provides a Read Mark pulse (YRMK) whenever a mark is read from the tape. Second, it controls IC5 to allow the one-shot pulse to appear as Read Clock (NRCL). Third, it controls IC3 and IC5 to allow data to be gated through to the data output flip-flop, IC9-2.

Referring now to FIGS. **64** and **68A–B**, the operation of the motor control portion of the magnetic tape cassette reading and recording unit will be described.

I'. Speed Reference

Resistors R1, R2, and R3 are used to generate a reference voltage proportional to the desired motor speed. The voltage seen at the positive input of U3 is about 3.0 volts for low speed and 8.0 volts for high speed.

II'. Comparator-Amplifier

Op Amp U3 compares the speed reference with a signal representing the actual motor speed (see Item IV' below). The difference signal is amplified and fed to the driver circuitry which increases or decreases the drive voltage available to the motor.

Resistor R8 and capacitor C1 create negative feed back around U1, resulting in a DC gain of 26 db and a single pole at about 100 Hz. This tailors the servo loop frequency response to provide stable operation with rapid error correction.

III'. Motor Driver

Diode D1 shifts the DC drive level by about 7 volts. Transistors Q1, Q2, and the Motor Pass Transistor (located on the Regulator Board) furnish current gain to drive the motor. Resistor R7 ensures turnoff for the

Motor Pass Transistor. Transistor Q3 is used to dynamically brake the motor if its speed is greater than desired.

Diode D3 reduces the maximum voltage at the motor when on clear leader. This voltage limit effectively reduces the motor torque, preventing damage to the motor, friction drive, or tape cassette should the tape be pulled against the hub. Diode D2 prevents damage to Q1 during this reduced torque condition.

IV'. Back-EMF Amplifier

The motor terminal voltage is composed of two parts — the IR voltage drop in the motor armature resistance, and the motor generated voltage, or Back-EMF. Back-EMF is directly proportional to motor speed, and is used as the motor speed feedback signal.

Sense resistor R9 produces a voltage proportional to the motor armature current, and therefore proportional to the IR voltage drop. Op amp U4 and resistors R10 thru R14 subtract the IR voltage drop from the motor terminal voltage, resulting in a voltage proportional to the motor speed. This signal is about 3.0 volts for low speed and 8.0 volts for high speed; it is fed back and compared with the reference voltage, as described in Item II' above.

The process of sensing the IR voltage drop, as described above, is accurate only for a single winding temperature. Because lower armature resistances (caused by lower temperatures) could cause instability in the servo, this perfect-compensation point is placed at the bottom of the operating range — 0° C in this case. Increasing temperature causes reduced load regulation (greater motor current causes reduced speed), but servo operation remains stable.

V'. Motor and Solenoid Selector

The selector circuitry activates the proper motor and solenoid. When in Run Fwd mode, Q4, Q5, and Q8 are saturated on; in Run Rev mode Q6, Q7, and Q9 are saturated. When in Stop mode, all devices are off.

Referring now to FIGS. 64 and 69, the operation of the interconnect portion of the magnetic tape cassette reading and recording unit will be described.

I. Connection From Transport to Mother Board

The main function of the Interconnect Board is to electrically connect the transport mechanism to the Mother Board. There are four groups of wires coming from the transport: the Motor wires, the Solenoid and Switch wires, the Head Board wires, and the Photosensor wires. Each group of wires is terminated on a Pin Board which plugs into the Interconnect Board; this partitioning allows for ease of assembly and service.

II. Clear Leader Signal

Most of the circuitry on the Interconnect Board is used to generate the Clear Leader signal. The photosensor assembly on the transport contains an incandescent lamp and a photoconductive cell. When magnetic tape is over the photosensor, no optical coupling occurs, resulting in a high photoconductor impedance. When clear leader is over the photosensor, light is reflected off the light colored plastic of the cassette and onto the photoconductor, resulting in a low photoconductor impedance.

Integrated circuit U1 and resistors R4, 5, and 6 form a comparitor which switches when the photoconductor impedance is approximately 25kΩ. Capacitor C5 filters the output of U1 to eliminate false signals of less than 5ms. duration; transistor Q1 amplifies this signal.

Under certain conditions when the tape stops or changes direction a tape loop may form over the photosensor; a reflection can result which causes the same photoconductor impedance as a clear leader. IC's U2, 3, and 4 are used to prevent a tape loop from giving a false clear leader indication. Flip-flop U4 stores the clear leader signal; this flip-flop may be turned "on" when the cassette has been running on clear leader for the duration of one-shot U2 (85 ms). This delay period ensures that any loop has been pulled out of the tape, preventing reflection problems.

A mag tape (Clear Leader Not) indication is always accurate; therefore it directly resets flip-flop U4. The gates tied to the preset of U4 force the flip-flop on during power turn on and when the cassette loader is open; if the tape is, in fact, on mag tape, this signal persists after preset is removed and resets U4.

III. Solenoid Turnoff

Diodes D1 and D2 and resistor R1 limit the peak solenoid flyback voltage during turnoff to about 24 volts. This also provides rapid solenoid dropout by applying up to −12 volts across the solenoid during turnoff.

IV. Motor Padding Resistors

Resistors R2 and R3 are selected to trim a given motor to between 10.30 and 10.60 ohms effective armature resistance. Capacitors C1 and C2 are used to suppress motor brush noise.

V. Cassette Sense Switches

Resistors R11 and R12 and pullup resistors for the mechanical cassette sense switches. Cassette In (positive true) and Write Permit (positive true) are the signals generated.

POWER SUPPLY

The power supply system employed in the calculator is constructed as shown in FIGS. 74 and 75A–B. As shown in FIGS. 75A–B, a centertapped transformer secondary is connected to supply the unregulated DC voltages indicated. Referring to FIG. 75A, the AC voltage from the transformer is rectified by diodes 454 and 456 and filtered by capacitor 478. The output of this rectifier/filter circuit is nominally 19 volts DC at 2.7 amps with a 2 volt peak-to-peak ripple. Transistors 440 and 442 serve as a switch to connect the 5-volt output bus to the 19 volt unregulated supply through inductors 450 and 452. Diode 444 serves to clamp the input of inductor 450 to ground when transistors 440 and 442 are switched off. Current flow in inductors 450 and 452 is substantially constant and equal to the load current.

Loss in high current transistor 440 is minimized because it can be completely saturated. Loss in driver transistor 442 is minimized because it can also be saturated. Resistor 446 limits the maximum drive current to transistor 440. Losses in resistor 446 can be minimized by proper positioning of the tap on inductor 450 consistent with transistor parameters and circuit requirements.

Integrated circuit 448 is a linear differential amplifier to drive Q1 and Q2. Any differential amplifier with sufficient voltage capability and bandwidth will work. Since the amplifier employed is linear, R7 and R4 have been included in the circuit to provide sufficient hysteresis for reliable switching. This hysteresis stabilizes the switching frequency and thus stabilizes the switching losses.

Because hysteresis has been added to the circuit, a significant ripple signal (at switching frequency) must be present on the feedback signal to the amplifier. This need for a ripple signal limits the amount of capacitance that can appear between the output of inductor

450 and ground. Inductor 452 serves to isolate this point from the rest of the system. The amount of capacity that can appear between the output of inductor 452 and ground is essentially unlimited and significantly reduces power supply ripple, and greatly improves response to load transients.

The second winding of inductor 452 is a path for the feedback from the remote sensing. The required ripple signal is added to the feedback signal by transformer action in inductor 452.

The power supply also includes an overvoltage crowbar circuit comprising transistor 458, diode 460, and resistor 462 and a short circuit shut-down circuit (using transistor 464). In the event that the +5 volt bus is grounded, or the crowbar is triggered, transistor 464 saturates and locks integrated circuit 448 off.

The resistor 466 makes a current generator of integrated circuit 448. Resistors 468 and 470 discharge the bases of transistors 450 and 452, respectively. Integrated circuit 472 and its associated components generate a power-on-pulse, $\overline{POP}$, to initialize the instrument. Integrated Circuit 448 is referenced and powered from an external +12 volt supply. Powering the IC from +12 rather than the unregulated +19 reduces power dissipation in IC 448.

The +12 volt supply of FIGS. 75A–B references the −12, +5, and +16 supplies directly. The +12 amplifier 448 may be biased either from the unregulated supply for the +12 volt supply or from the operating +16 volt supply. Diodes 474 and 476 determine the appropriate source. This provides a greater power supply margin for the +12 volt supply.

All supplies except the +20 volt supply are current limited. All supplies except the +20 volt supply are crowbar protected against over-voltage.

TYPEWRITER INTERFACE

This interface couples the Facit-Odhner model 3841 output typewriter to the calculator.

The unit mounts directly on the back of the typewriter. Communications with the calculator are made through about five feet of cable which is terminated by the I/O plug containing a board for buffering and some logic.

Referring to FIGS. 76A–82, characters from the calculator appear on the data lines as ASCII codes. These codes are recorded by a ROM into the six bit Facit typewriter code for the 46 type bars, and one bit for upper case shift. Functions such as space, tab, line feed, etc. are recoded for easy recognition in the interface since each function must be driven by a separate line. A data latch after the ROM holds codes for processing. If new data arrives during this processing, the two codes are compared to determine if they both drive the same type bar and if they are both numbers. Non-repeating numbers can be typed at 14.5 characters per second, otherwise typing speed is 12 characters per second (reduce these speeds 17% for 50HZ operation). Codes in the latch are gated to the program solenoids or the function solenoids by the control logic.

To understand the coding, notice that two blocks of codes on the Facit typewriter code map are empty. If all function codes are put in these blocks, they can be identified by control logic by testing for ($\overline{6.4}$). Each function code puts a 1 on one of five lines and this line opens the correct solenoid gate. Bit 8 is used to discriminate between two sets of function gates. In the case of a program solenoid code, bit 8 identifies numerals.

The control clock is provided by a synchronizing pulse which is generated in the typewriter by a vaned wheel attached to the end of the main drive shaft. The vanes interrupt a light beam. When a type cycle is initiated, a modulo eight counter counts synchronizing pulses and the count is decoded by a 1-of-8 decoder. At each of the eight states, comtinational logic can enable solenoid gates, set or clear flag flip-flops or change the counter to state zero, or state 6, or inhibit the counter.

The tables below contain a guide for interpretation of bit pattern data as well as the actual bit patterns for ROM No. 10 and ROM No. 11 as shown in FIG. 81.

INTERPRETATION OF BIT PATTERN DATA

(Bipolar ROM of FIG. 81)

1. FORMAT

The bit pattern information is in the following format:

$X_1X_2X_3$-$X_5X_6X_7$ B B $X_{10}X_{11}X_{12}X_{13}$ B $X_{15}X_{16}X_{17}X_{18}$ B...$X_{45}X_{46}X_{47}X_{48}$

A. $X_1X_2X_3$ — Three digits indicating the address (decimal) of the first word of that line. *1

B. $X_5X_6X_7$ — Three digits indicating the address of the last word in that line. *1

C. $X_9X_{10}X_{11}X_{12}$ — Four characters indicating the output states of the first word of that line (corresponding to address $X_1X_2X_3$). *2

D. $X_{15}X_{16}X_{17}X_{18}$ through $X_{40}X_{41}X_{42}X_{43}$ indicate successive output states. *2

E. $X_{45}X_{46}X_{47}X_{48}$ — Four characters indicating the output states of the last word of that line (corresponding to address $X_5X_6X_7$). *2

F. B = Blank or space between group of characters.

*1. Addresses are the digital equivalent of the binary address $A_7A_6A_5A_4A_3A_2A_1A$. Where a low input address equals a binary 0 and a high input equals a binary 1.

*2. Groups of output states are listed $O_4O_3O_2O_1$ respectively.

2. TRUTH TABLE

Logic level definition

L — Output Low (Logic 0)

H — Output High or Open Collector (Logic 1)

X — Don's Care — Output may be High or Low

BIPOLAR ROM 10 OF FIGURE 81

000-007	LLLL	LLLH	LLLL	LLLL	LLLL	LLLL	HLLL	LHLL
008-015	HLLL	LLLL	LHLL	LLHL	LLLL	LLHL	LLLL	LLLL
016-023	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
024-031	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL

BIPOLAR ROM 10 OF FIGURE 81 -continued

032-039	LLLH	LLLH	LLHL	LLHH	LHLL	LHLH	LHHL	LHHH
040-047	HLLL	HLLH	HLHL	HLHH	HHLL	HHLH	HHHL	HHHH
048-055	LLLL	LLLH	LLHL	LLHH	LHLL	LHLH	LHHL	LHHH
056-063	HLLL	HLLH	HLHL	HLHH	HHLL	HHLH	HHHL	HHHH
064-071	LLLL	LLLH	LLHL	LLHH	LHLL	LHLH	LHHL	LHHH
072-079	HLLL	HLLH	HLHL	HLHH	HHLL	HHLH	HHHL	HHHH
080-087	LLLL	LLLH	LLHL	LLHH	LHLL	LHLH	LHHL	LHHH
088-095	HLLL	HLLH	HLHL	HLHH	LLLL	HHLH	HHHL	LLLL
096-103	LLLL	LLLH	LLHL	LLHH	LHLL	LHLH	LHHL	LHHH
104-111	HLLL	HLLH	HLHL	HLHH	HHLL	HHLH	HHHL	HHHH
112-119	LLLL	LLLH	LLHL	LLHH	LHLL	LHLH	LHHL	LHHH
120-127	HLLL	HLLH	HLHL	HLHH	LLLL	HHLH	HHHL	LLLL
128-135	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
136-143	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
144-151	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
152-159	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
160-167	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
168-175	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
176-183	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
184-191	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
192-199	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
200-207	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
208-215	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
216-223	LLLL	LLLL	LLLL	HHLH	HLHH	LLLL	HHHL	HHHL
224-231	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
232-239	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
240-247	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
248-255	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL

BIPOLAR ROM 11 OF FIGURE 81

000-007	LLLL	HLLL	LLLL	LLLL	LLLL	LLLL	HLLL	HLLL
008-015	LLLL	LHLL	LLLL	HLLL	HHLL	LLLL	LLLL	LLLL
016-023	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
024-031	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
032-039	LLLL	LHLH	LHLH	LHLH	LHLH	LHLH	LHLH	LHLH
040-047	LHLH	LHLH	LHLH	LHLH	LLLH	HLLH	LLLH	LLLH
048-055	HLLH	HLLH	HLLH	HLLH	HLLH	HLLH	HLLH	HLLH
056-063	HLLH	HLLH	HLLH	LLLH	LHLH	LHLH	LHLH	LHLH
064-071	LLHL	LHHL	LHHL	LHHL	LHHL	LHHL	LHHL	LHHL
072-079	LHHL	LHHL	LHHL	LHHL	LHHL	LHHL	LHHL	LHHL
080-087	LHHH	LHHH	LHHH	LHHH	LHHH	LHHH	LHHH	LHHH
088-095	LHHH	LHHH	LHHH	LLHH	LLLL	LLHH	LLHH	LHLH
096-103	LHHL	LLHL	LLHL	LLHL	LLHL	LLHL	LLHL	LLHL
104-111	LLHL	LLHL	LLHL	LLHL	LLHL	LLHL	LLHL	LLHL
112-119	LLHH	LLHH	LLHH	LLHH	LLHH	LLHH	LLHH	LLHH
120-127	LLHH	LLHH	LLHH	LHHH	LLLL	LHHH	LHHH	LLLL
128-135	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
136-143	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
144-151	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL

BIPOLAR ROM 11 OF FIGURE 81-continued

152-159	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
160-167	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
168-175	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
176-183	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
184-191	LLLL	LLLL	LLLL	LLLL	LHHL	LLLL	LLLL	LLLL
192-199	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
200-207	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
208-215	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
216-223	LLLL	LLLL	LLLL	LHHH	LHHH	LLLL	LLHH	LHHH
224-231	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
232-239	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
240-247	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL
248-255	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL	LLLL

We claim:

1. Electronic data processing apparatus comprising:

keyboard input means for entering lines of one or more alphameric characters each into said electronic data processing apparatus;

first memory means for storing lines of one or more alphameric characters each;

second memory means, coupled to said keyboard input means and first memory means, for temporarily storing each line of one or more alphameric characters being entered into said electronic data processing apparatus from said keyboard input means or recalled from said first memory means;

said keyboard input means including store control means for causing a line of one or more alphameric characters then stored in said second memory means to be stored in said first memory means and including recall control means for causing a designated line of one or more alphameric characters stored in said first memory means to be recalled to said second memory means;

first logic means, coupled to said keyboard input means, first memory means, and second memory means, for transferring a line of one or more alphameric characters then stored in said second memory means to said first memory means in response to actuation of said store control means and for transferring a designated line of one or more alphameric characters from said first memory means to said second memory means in response to actuation of said recall control means;

display means, coupled to said second memory means, for visually displaying a line of one or more alphameric characters then stored in said second memory means;

said display means including means for visually displaying a cursor for designating any character position of the displayed line of one or more alphameric characters;

said keyboard input means including cursor control means for controlling the position of said cursor; and

second logic means, coupled to said keyboard input means, second memory means, and display means, for positioning said cursor to designate any character position of the displayed line of one or more alphameric characters in response to actuation of said cursor control means.

2. Electronic data processing apparatus in claim 1 wherein said cursor control means comprises a single control key for moving said cursor to the left and a single control key for moving said cursor to the right.

3. Electronic data processing apparatus as in claim 1 wherein:

said keyboard input means includes a space bar; and

said second logic means is responsive to actuation of said space bar for introducing a blank character at one or more character positions in a line of one or more alphameric characters then stored or being stored in said second memory means.

4. Electronic data processing apparatus as in claim 1 wherein:

said keyboard input means includes insertion control means for controlling the insertion of one or more characters into a line of one or more alphameric characters; and

said second logic means is responsive to actuation of said insertion control means for inserting a blank character into a line of one or more alphameric characters then stored in said second memory means and displayed by said display means at the character position then designated by said cursor so that an alphameric character designated by actuation of an alphameric key of said keyboard input means may be inserted at the character position occupied by that blank character.

5. Electronic data processing apparatus as in claim 4 wherein said insertion control means comprises a single control key.

6. Electronic data processing apparatus as in claim 1 wherein:

said keyboard input means includes deletion control means for controlling the deletion of one or more characters from a line of one or more alphameric characters; and

said second logic means is responsive to actuation of said deletion control means for deleting a character from a line of one or more alphameric characters then stored in said second memory means and displayed by said display means at the character position then designated by said cursor.

7. Electronic data processing apparatus as in claim 6 wherein:

said deletion control means comprises a shift control key and one other control key; and

said second logic means is responsive to simultaneous actuation of those two keys for deleting a character from a line of one or more alphameric characters then stored in said second memory means and displayed by said display means at the character position then designated by said cursor.

8. Electronic data processing apparatus as in claim **1** wherein:

said keyboard input means includes display position control means for controlling the position of the displayed line of one or more alphameric characters; and

said second logic means is responsive to actuation of said display position control means for moving the displayed line of one or more alphameric characters either to the left or the right independently of said cursor.

9. Electronic data processing apparatus as in claim **8** wherein said display position control means comprises a single control key for moving the displayed line of one or more alphameric characters to the left, and a single control key for moving the displayed line of one or more alphameric characters to the right.

10. Electronic data processing apparatus as in claim **8** wherein said first logic means is responsive to actuation of said store control means for transferring a line of one or more alphameric characters then stored in said second memory means and displayed by said display means from said second memory means to said first memory means irrespective of the position of said cursor or of the position of the displayed line of one or more alphameric characters.

11. Electronic data processing apparatus as in claim **10** wherein:

said store control means comprises a single control key; and

said recall control means comprises a single control key actuated in sequence with one or more line number designating keys of the keyboard input means.

12. Electronic data processing apparatus as in claim **1** wherein said first logic means is responsive to actuation of said store control means for transferring a line of one or more alphameric characters then stored in said second memory means and displayed by said display means from said second memory means to said first memory means irrespective of the position of said cursor in the displayed line of one or more alphameric characters.

13. Electronic data processing apparatus as in claim **1** wherein:

said store control means comprises a single control key; and

said recall control means comprises a single control key actuated in sequence with one or more line number designating keys of the keyboard input means.

14. Electronic data processing apparatus comprising:

keyboard input means, including a plurality of numeric keys, for entering lines of one or more alphameric characters each into said electronic data processing apparatus;

first memory means for storing lines of one or more alphameric characters each;

second memory means, coupled to said keyboard input means and first memory means, for temporarily storing each line of one or more alphameric characters entered into said electronic data processing apparatus from said keyboard input means or recalled from said first memory means;

said keyboard input means including a store control key for causing the line of one or more alphameric characters then stored in said second memory means to be stored in said first memory means; and

logic means, coupled to said keyboard input means, first memory means, and second memory means, for transferring the line of one or more alphameric characters then stored in said second memory means to said first memory means in response to actuation of said store control key;

display means, coupled to said second memory means, for visually displaying the contents thereof;

said keyboard input means including a recall control key for recalling a designated line of one or more alphameric characters from said first memory means to said second memory means;

each line of one or more alphameric characters stored in said first memory means being associated with a separate line number;

said logic means is also operable for transferring a line of one or more alphameric characters from said first memory means to said second memory means in response to sequential actuation of said recall control key and one or more of said numeric keys designating the line number associated with the line of one or more alphameric characters to be transferred.

15. Electronic data processing apparatus as in claim **14** wherein:

said keyboard input means includes first and second memory control keys;

said logic means includes a line number counter for designating the line number associated with the line of one or more alphameric characters then stored in said second memory means; and

said logic means is responsive to actuation of said first memory control key or said second memory control key for incrementing or decrementing, respectively, said line number counter and for transferring the line of one or more alphameric characters associated with the line number then designated by said line number counter from said first memory to said second memory means.

16. Electronic data processing apparatus as in claim **15** wherein said logic means is responsive to successive actuation of said first memory control key or said second memory control key for successively incrementing or decrementing, respectively, said line number counter and for successively transferring the lines of one or more alphameric characters associated with the line numbers successively designated by said line number counter from said memory means to said second memory means.

17. Electronic data processing apparatus as in claim **14** wherein said first means stores each separate line number as part of the line of one or alphameric characters associated therewith.

18. Electronic data processing apparatus as in claim **17** wherein:

said logic means includes comparison means for comparing the line number designating the line of one or more alphameric characters to be transferred from said first memory means to said second memory means with the line numbers stored in said first memory means;

said logic means being responsive to a condition of

equality indicated by said comparison means for transferring the designated line of one or more alphameric characters from said first memory means to said second memory means; and

said logic means being responsive to said comparison means in the event no condition of equality is detected for transferring, from said first memory means to said second memory means, the next higher numbered line above the designated line number.

19. Electronic data processing apparatus as in claim **14** wherein:

said keyboard input means includes first and second memory control keys;

said logic means includes a line number counter for storing the line number associated with a line of one or more alphameric characters stored in said first memory means that has been recalled to said second memory means; and

said logic means is responsive to actuation of said first memory control key or said second memory control key for incrementing or decrementing, respectively, said line number counter and for transferring the next higher numbered line or the next lower numbered line, respectively from said first memory means to said second memory means.

20. Electronic data processing apparatus comprising:

keyboard input means for entering lines of one or more alphameric characters each into said electronic data processing apparatus;

first memory means, coupled to said keyboard input means, for temporarily storing each line of one or more alphameric characters being entered into said electronic data processing apparatus from said keyboard input means;

display means, coupled to said first memory means, for visually displaying the contents thereof;

said keyboard input means including one or more control keys for terminating entry of each line of one or more alphameric characters into said electronic data processing apparatus and including a recall key for recalling the line of one or more alphameric characters most recently terminated by one of said control keys to said first memory means;

second memory means, coupled to said keyboard input means and first memory means, for temporarily storing a line of one or more alphameric characters; and

logic means, coupled to said keyboard input means, first memory means, and second memory means, for transferring the most recently terminated line of one or more alphameric characters then stored in said first memory means to said second memory means in response to termination of that line of one or more alphameric characters and for thereafter transferring the most recently terminated line of one or more alphameric characters then stored in said second memory means to said first memory means in response to actuation of said recall key.

21. An electronic calculator comprising:

keyboard input means for entering lines of one or more alphameric characters each into said electronic calculator;

first memory means for storing a program of one or more lines of one or more alphameric characters each;

second memory means, coupled to said keyboard input means and first memory means, for temporarily storing each line of one or more alphameric characters entered into said calculator from said keyboard input means or recalled from said first memory means;

said keyboard input means including store control means for causing a line of one or more alphameric characters then stored in said second memory means to be stored in said first memory means and including recall control means for causing a designated line of one or more alphameric characters stored in said first memory means to be recalled to said second memory means;

processing means, coupled to said keyboard input means, first memory means, and second memory means, for transferring a line of one or more alphameric characters then stored in said second memory means to said first memory means in response to actuation of said store control means, for transferring a designated line of one or more alphameric characters from said first memory means to said second memory means in response to actuation of said recall control means, and for executing said program to perform a selected calculation and storing the result of the selected calculation in said second memory means;

output display means, coupled to said second memory means, for visually displaying the contents thereof;

said keyboard input means including program halt control means for halting the execution of said program by said processing means, and program continue control means for resuming the execution of said program by said processing means; and

logic means, coupled to said keyboard input means, first memory means, second memory means, and processing means, for halting execution of said program in response to actuation of said program halt control means, for thereafter transferring a designated line of one or more alphameric characters of said program from said first memory means to said second memory means in response to actuation of said recall control means, and for thereafter resuming execution of said program, at the point where its execution was halted, in response to actuation of said program continue control means.

22. An electronic calculator as in claim **21** wherein:

said recall control means comprises a recall control key operable in sequence with one or more numeric keys for recalling a designated line of one or more alphameric characters of said program from said first memory means to said second memory means;

said program halt control means comprises a single control key; and

said program continue control means comprises a single control key.

23. Electronic data processing apparatus comprising:

keyboard input means for entering lines of one or more alphameric characters each into said electronic data processing apparatus;

memory means, coupled to said keyboard input means, for storing lines of one or more alphameric characters each entered into said electronic data processing apparatus, every one of said lines of one or more alphameric characters each stored in said memory means being associated with a separate line number;

output printing means, coupled to said keyboard input means and memory means, for printing lines of one or more alphameric characters each;

said keyboard input means including memory listing means for designating, by line number, any number of lines of one or more alphameric characters each then stored in said memory means to be printed and for initiating the printing of those designated lines of one or more alphameric characters each; and

logic means, coupled to said keyboard input means, memory means, and output printing means, for selectively causing said output printing means to print out the designated lines of one or more alphameric characters each in response to actuation of said memory listing means.

24. An electronic calculator as in claim **21** wherein:

said keyboard input means includes an execute control key for initiating execution of a line of one or more alphameric statements entered into the calculator from said keyboard input means; and

said logic means is responsive to entry of a line of one or more alphameric statements into the calculator and to actuation of said execute control key, during a period of time after which execution of a program has been halted by actuation of said program halt control means, for causing said processing means to execute that entered line of one or more alphameric statements, said logic means thereafter being responsive to actuation of said program continue control means for resuming execution of the program at the point where its execution was previously halted.

25. Electronic data processing apparatus as in claim **24** wherein:

said memory listing means comprises a list control key; and

said logic means is responsive to sequential actuation of said list control key and selected other keys of said keyboard input means designating a beginning line number and an ending line number for causing said output printing means to print out the lines of one or more alphameric characters each stored in said memory means having line numbers within the range defined by said beginning and ending line numbers, inclusive.

26. Electronic data processing apparatus as in claim **24** wherein:

said memory listing means comprises a list control key; and

said logic means is responsive to sequential actuation of said list control key and selected other keys of said keyboard input means designating a beginning line number for causing said output printing means to print out the lines of one or more alphameric characters each stored in said memory means commencing with the line associated with said beginning line number and including all lines associated with line numbers greater than said beginning line number.

27. Electronic data processing apparatus comprising:

keyboard input means, including a plurality of alphameric keys, for entering lines of one or more alphameric characters each into said electronic data processing apparatus;

memory means for storing lines of one or more alphameric characters each, every stored line of one or more alphameric characters being associated with a separate line number;

buffer storage means, coupled to said keyboard input means and memory means, for storing a line number and for storing an associated line of one or more alphameric characters entered into said electronic data processing apparatus;

display means, coupled to said buffer storage means, for visually displaying the contents thereof;

said keyboard input means including a control key for designating an automatic line numbering mode;

a line number counter for storing a current line number;

temporary storage means for storing a line number increment; and

logic means coupled to said keyboard input means and buffer storage means, said logic means being responsive to actuation of said control key, followed by actuation of one or more alphameric keys designating both a starting line number and a line number increment, for storing the starting line number in both said buffer storage means and said line number counter and for storing the line number increment in said temporary storage means, said logic means being further responsive to completion of each entry of a line of one or more alphameric characters into the electronic data processing apparatus for combining the contents of said line number counter and said temporary storage means and for storing the result in both said line number counter and said buffer storage means.

28. Electronic data processing apparatus as in claim **27** wherein:

said keyboard input means includes a delete key; and

said logic means is responsive to actuation of said delete key during entry of a line of one or more alphameric characters into the electronic data processing apparatus for deleting from said buffer storage means all of the characters previously entered as part of that line, while retaining the associated line number stored in said buffer storage means.

29. Electronic data processing apparatus as in claim **27** wherein said logic means is responsive to actuation of said control key, without further actuation of alphameric keys designating a starting line number and a line number increment, for storing a predetermined starting line number in both said buffer storage means and said line number counter and for storing a predetermined line number increment in said temporary storage means.

30. Electronic data processing apparatus as in claim **27** wherein said logic means is responsive to actuation of said control key, followed by actuation of one or more alphameric keys designating a starting line number, for storing that starting line number in both said buffer storage means and said line number counter and for storing a predetermined line number increment in said temporary storage means.

31. Electronic data processing apparatus comprising:

keyboard input means having a plurality of alphameric keys for entering lines of one or more alphameric characters each into said electronic data processing apparatus and having a plurality of user definable keys for entering an associated plurality of user-defined functions into said electronic data processing apparatus;

memory means, coupled to said keyboard input means, for storing a mainline program of one or more lines of one or more alphameric characters each entered into said electronic data processing

apparatus and for storing said plurality of user-defined functions associated with said plurality of user-definable keys;

said keyboard input means including first, second, and third memory erase means for selectively initiating the erasure of said mainline program and/or said user-defined functions stored in said memory means; and

logic means, coupled to said keyboard input means and memory means, for erasing said mainline program in response to said first memory erase means, for erasing a selected one of said plurality of user-defined functions in response to said second memory erase means, and for erasing all of said plurality of user-defined functions in response to said third memory erase means.

32. Electronic data processing apparatus as in claim **31** wherein:

said keyboard input means includes a memory erase control key and an execute control key; and

said first memory erase means comprises sequential actuation of said memory erase control key and said execute control key.

33. Electronic data processing apparatus as in claim **31** wherein:

said keyboard input means includes a memory erase control key and an execute control key; and

said second memory erase means comprises sequential actuation of said memory erase control key, a selected one of said plurality of user definable keys, and said execute control key.

34. Electronic data processing apparatus as in claim **31** wherein:

said keyboard input means includes a memory erase control key and an execute control key; and

said third memory erase means comprises sequential actuation of said memory erase control key, one of said alphameric keys, and said execute control key.

35. An electronic calculator comprising:

keyboard input means having a plurality of alphameric keys for entering lines of one or more alphameric characters each into said calculator;

memory means, coupled to said keyboard input means, for storing a program of lines of one or more alphameric characters each entered into said calculator, every stored line being associated with a separate line number;

processing means, coupled to said keyboard input means and memory means, for executing the program stored in said memory means;

said keyboard input means including a halt execution key for conditioning the processing means to halt execution of the program immediately prior to the line thereof associated with a designated line number specified by one or more of said alphameric keys;

a program counter, coupled to said processing means, for storing the line number associated with the line currently being executed;

temporary storage means, coupled to said keyboard input means and memory means, for storing the designated line number; and

logic means, coupled to said keyboard input means, processing means, program counter, and temporary storage means, for storing the designated line number in said temporary storage means in response to actuation of said halt execution key, followed by actuation of said one or more alphameric keys, prior to commencing execution of the program stored in said memory means and for subsequently halting the execution of the program in response to the occurrence of a condition of equality between the contents of said program counter and said temporary storage means.

36. An electronic calculator as in claim **35** wherein:

said keyboard input means includes an execute control key for initiating execution of a line of one or more alphameric statements entered into the calculator from said keyboard input means;

said keyboard input means further including program continue control means for resuming execution of the program whose execution has been halted; and

said logic means is responsive to entry of a line of one or more alphameric statements into the calculator and to actuation of said execute control key, during a period of time after which execution of the program has been halted, for causing said processing means to execute that entered line of one or more alphameric statements, said logic means thereafter being responsive to actuation of said program continue control means for resuming execution of the program at the point where its execution was previously halted.

37. An electronic calculator comprising:

keyboard input means, including a plurality of alphameric keys, for entering lines of one or more alphameric characters each into said calculator, said keyboard input means further including a first program control key;

memory means, coupled to said keyboard input means, for storing a program of one or more lines of one or more alphameric characters each entered into said calculator from said keyboard input means;

line numbering means for associating every stored line with a separate line number;

processing means, coupled to said keyboard input means and memory means, for executing each line of the program stored in said memory means; and

logic means coupled to said keyboard input means and processing means, said logic means being responsive to actuation of said first program control key, in sequence with one or more of said alphameric keys designating the line number of any selected line of the program stored in said memory means, for initiating execution of the program at that selected line.

38. An electronic calculator as in claim **37** wherein:

said keyboard input means still further includes a second program control key;

said memory means stores variables forming part of the program stored in said memory means;

said logic means is further responsive to actuation of said first program control key, in sequence with said one or more alphameric keys, for setting all of the variables of the program stored in said memory means to an initial state prior to initiating execution of the program; and

said logic means is further responsive to actuation of said second program control key, followed by actuation of one or more of said alphameric keys designating the line number of any selected line of the program stored in said memory means, for initiating execution of the program at that line number without altering the variables stored in said memory means.

39. An electronic calculator comprising:

keyboard input means for entering lines of one or

more alphameric characters each into the calculator, said keyboard input means including a line termination key for terminating entry of each line of one or more alphameric characters into the calculator and a print control key for designating a print-all mode of calculator operation;

memory means, coupled to said keyboard input means, for storing lines of one or more alphameric characters each entered into the calculator;

processing means, coupled to said keyboard input means and memory means, for executing lines of one or more alphameric characters each stored in said memory means;

output display means, coupled to said keyboard input means, memory means and processing means, for visually displaying lines of one or more alphameric characters each;

logic means, coupled to said keyboard input means, memory means, processing means, and display means, for initiating the display of lines of one or more alphameric characters entered into the calculator, alphameric error messages generated during execution of lines of one or more alphameric characters by said processing means, and executed display commands;

printing means for printing lines of one or more alphameric characters;

said logic means also being coupled to said printing means and being responsive to actuation of said print control key for initiating the printing of lines of one or more alphameric characters entered into the calculator and terminated by said line termination key, error messages generated during execution of lines of one or more alphameric characters by said processing means, and executed display commands.

40. An electronic calculator as in claim **39** wherein said logic means is responsive to two successive actuations of said print control key, after the print-all mode of calculator operation has been designated, for cancelling the previously designated print-all mode of calculator operation.

41. Electronic data processing apparatus comprising:

keyboard input means including a plurality of keys for entering operands into the electronic data processing apparatus;

first memory means, coupled to said keyboard input means, for temporarily storing operands being entered into the electronic data processing apparatus from said keyboard input means;

display means, coupled to said first memory means, for visually displaying the contents thereof;

said keyboard input means including one or more control keys for terminating entry of operands into the electronic data processing apparatus and including a recall key for recalling the operand most recently terminated by one of said control keys to said first memory means;

second memory means, coupled to said keyboard input means and first memory means, for temporarily storing an operand; and

logic means, coupled to said keyboard input means, first memory means, and second memory means, for transferring the most recently terminated operand then stored in said first memory means to said second memory means in response to termination of that operand and for thereafter transferring the most recently terminated operand then stored in said second memory means to said first memory means in response to actuation of said recall key.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 4,012,725

DATED : March 15, 1977

INVENTOR(S) : Richard M. Spangler et al

It is certified that error appears in the above—identified patent and that said Letters Patent is hereby corrected as shown below:

Column 16, line 51, "Fig. 1-B" should be --Figs. 1-3B--;

Column 26, line 41, "illustrted" should be --illustrated--;

Column 700, line 55, Claim 16, after "said" (first occurrence) insert --first--;

Column 703, line 35, Claim 25, "24" should be --23--;

Column 703, line 49, Claim 26, "24" should be --23--.

Signed and Sealed this

Twenty-ninth Day of *September 1981*

[SEAL]

Attest:

GERALD J. MOSSINGHOFF

Attesting Officer

Commissioner of Patents and Trademarks