United States Patent [19]

Miller et al.

[11] **4,089,059**

[45] **May 9, 1978**

[54] **PROGRAMMABLE CALCULATOR EMPLOYING A READ-WRITE MEMORY HAVING A MOVABLE BOUNDARY BETWEEN PROGRAM AND DATA STORAGE SECTIONS THEREOF**

[75] Inventors: **Bradley W. Miller; Franklin T. Hickenlooper; David C. Uhlrich; Marl D. Godfrey; Douglas M. Clifford; Rex L. James; Robert E. Watson; John C. Keith; Alan C. Mortensen,** all of Loveland, Colo.

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

[21] Appl. No.: **597,957**

[22] Filed: **Jul. 21, 1975**

[51] **Int. Cl.**[2] **G06F 9/06**

[52] **U.S. Cl.** **364/706;** 364/200

[58] **Field of Search** 235/152, 156; 340/172.5

[56] **References Cited**

U.S. PATENT DOCUMENTS

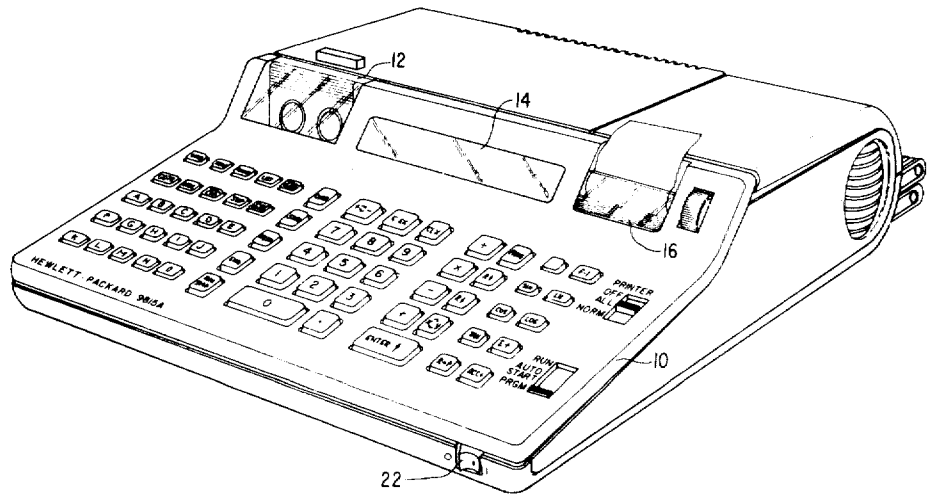
3,461,434	8/1969	Barton et al.	340/172.5
3,548,384	12/1970	Barton et al.	340/172.5
3,588,841	6/1971	Ragen	340/172.5
3,596,257	7/1971	Patel	340/172.5
3,878,513	4/1975	Werner	340/172.5
3,904,862	9/1975	Cochrun et al.	235/156

Primary Examiner—David H. Malzahn
Attorney, Agent, or Firm—William E. Hein

[57] **ABSTRACT**

An adaptable programmable calculator employs modular read-write and read-only memories separately expandable to provide additional program and data storage functions within the calculator oriented toward the environment of the user, an LSI NMOS central processing unit, and an LSI NMOS peripheral interface adaptor capable of bidirectionally transferring information between the read-write memory and central processing unit and a number of input/output units. The modular read-write memory includes a movable boundary between a program storage section thereof and a data storage section thereof to permit the user to adjust the size of those sections of the read-write memory in accordance with his present problem solving requirements. The input/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, a seven-segment gas discharge display for displaying data entered into the calculator, the results of computations, and selected alphanumeric messages, and a 16-column alphanumeric thermal printer for printing results of computations, program listings, messages generated by the user and the calculator itself, and error conditions encountered during use of the calculator. All of these input/output units are included within the calculator itself. Many other external input/output units may be employed with the calculator. The calculator may be operated manually by the user from the keyboard input unit or automatically through a program stored within the read-write memory to perform calculations and to provide an output indication of the results thereof. The calculator employs reverse polish notation (RPN) language including an operational stack of registers for efficiently evaluating algebraic expressions. The language is arranged on a modified key per function basis, incorporating some of the features of higher level languages such as loops. The language also includes sophisticated editing features that enhance the usefulness of the calculator.

6 Claims, 226 Drawing Figures



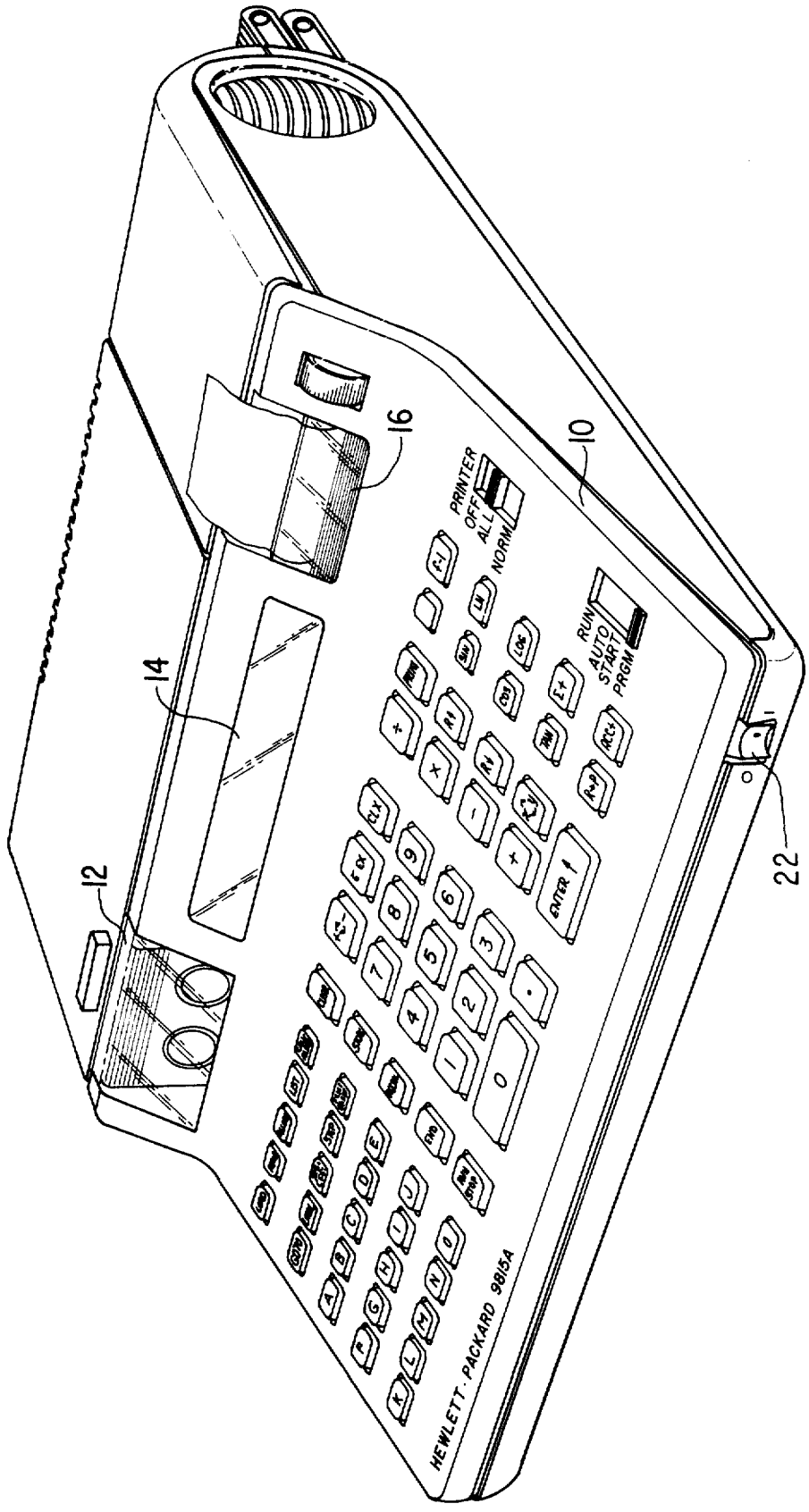

16
10
14
12
22
PRINTER
OFF
ALL
NORM
RUN
AUTO
START
PRGM
HEWLETT-PACKARD 9815A

FIG 1

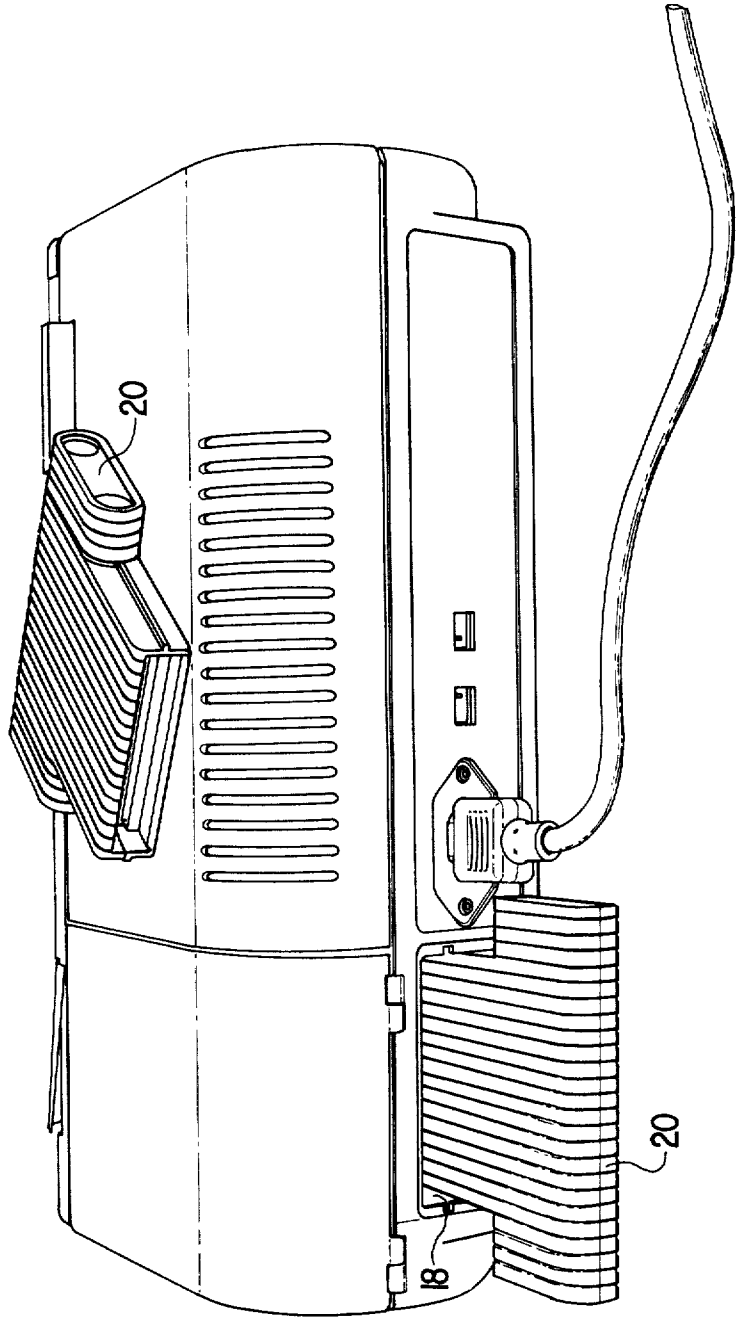


FIG 2

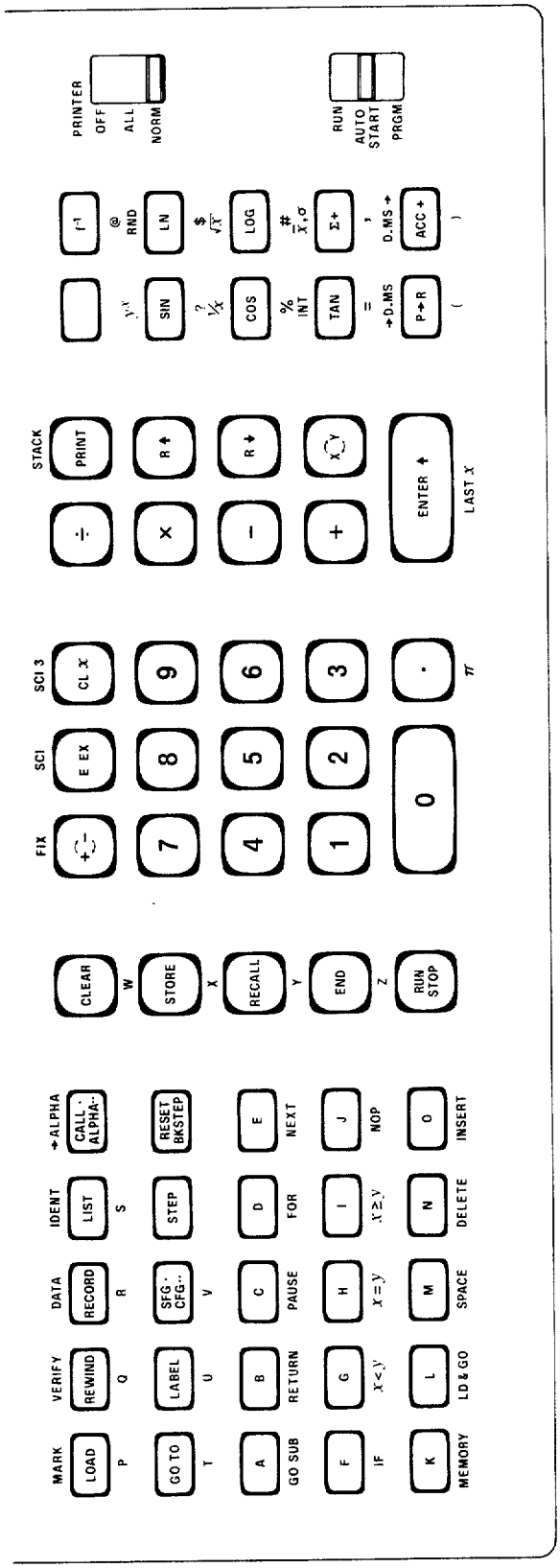

PRINTER
OFF
ALL
NORM
RUN
AUTO START
PRGM
STACK
PRINT
R↓
R↑
X⇄Y
LAST X
ENTER ↑
SIN
COS
TAN
P→R
LN
LOG
Σ+
ACC+
CLEAR
STORE
RECALL
END
RUN STOP
CL X
E EX
FIX
SCI
SCI 3
CALL ALPHA
RESET BKSTEP
LIST
STEP
RECORD
SFG CFG
LABEL
REWIND
LOAD
GO TO
MARK
VERIFY
DATA
IDENT
→ALPHA
NEXT
NOP
INSERT
DELETE
SPACE
LD & GO
MEMORY
GO SUB
RETURN
PAUSE
FOR
IF

FIG. 3

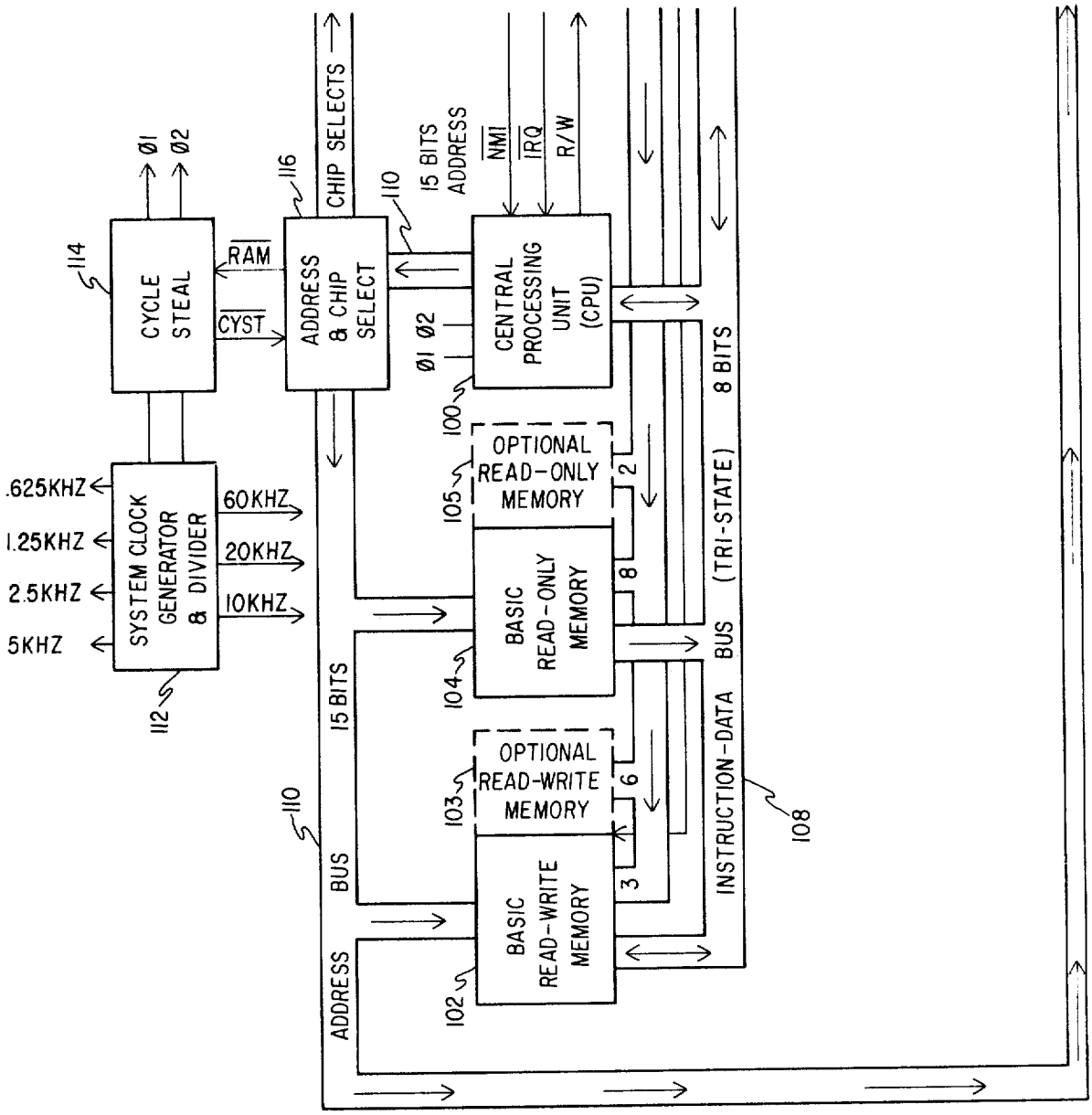

Ø1
Ø2
CYCLE STEAL
114
RAM
CYST
ADDRESS & CHIP SELECT
116
CHIP SELECTS
110
15 BITS ADDRESS
NMI
IRQ
R/W
CENTRAL PROCESSING UNIT (CPU)
Ø1 Ø2
100
OPTIONAL READ-ONLY MEMORY
105
BASIC READ-ONLY MEMORY
104
OPTIONAL READ-WRITE MEMORY
103
BASIC READ-WRITE MEMORY
102
2
8
6
3
8 BITS
(TRI-STATE)
BUS
INSTRUCTION-DATA
108
15 BITS
BUS
ADDRESS
110
.625KHZ
1.25KHZ
2.5KHZ
5KHZ
SYSTEM CLOCK GENERATOR & DIVIDER
112
60KHZ
20KHZ
10KHZ

FIG 4A

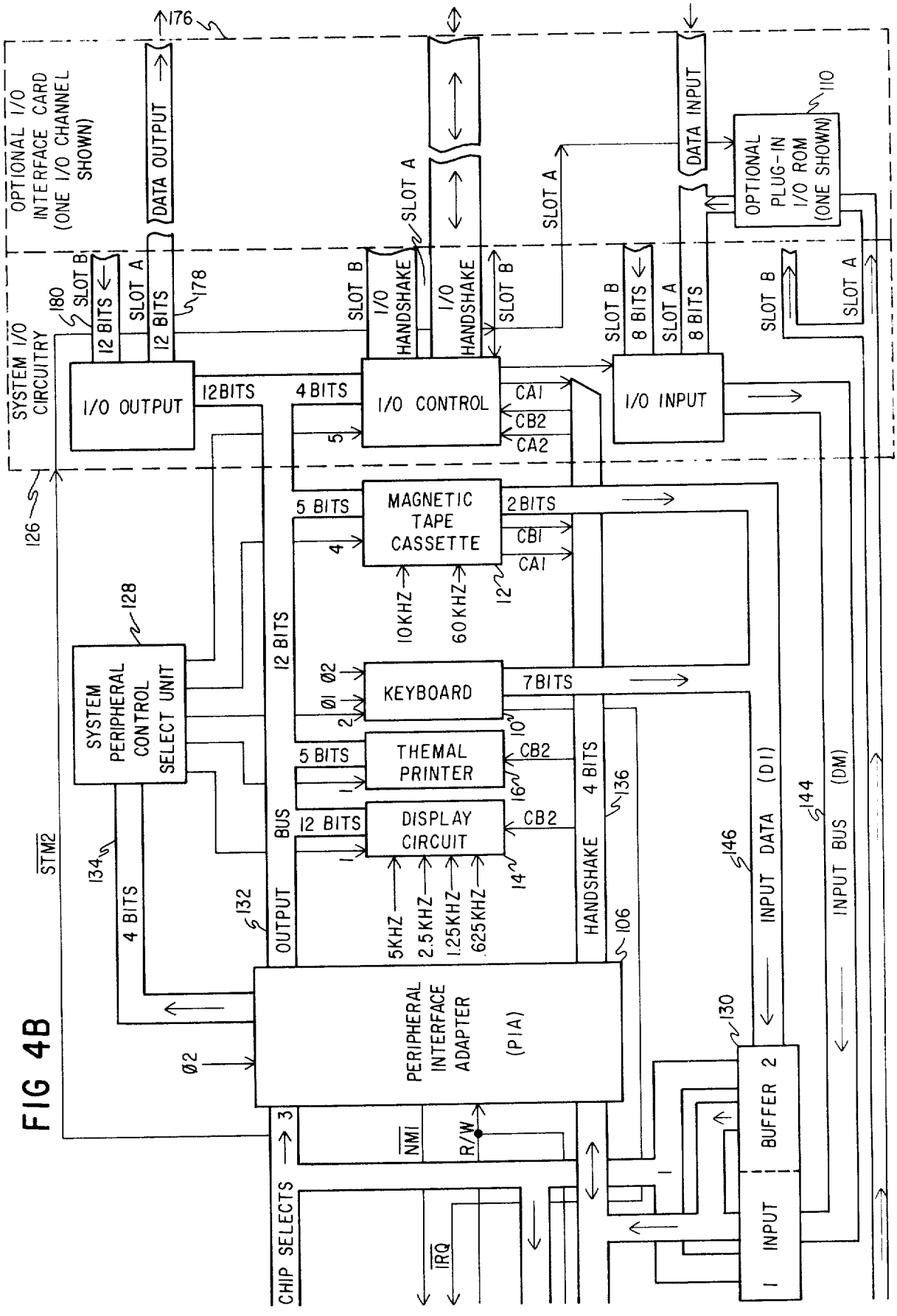

OPTIONAL I/O INTERFACE CARD (ONE I/O CHANNEL SHOWN)
SYSTEM I/O CIRCUITRY
176
DATA OUTPUT
DATA INPUT
OPTIONAL PLUG-IN I/O ROM (ONE SHOWN)
110
SLOT A
SLOT B
12 BITS
8 BITS
178
180
I/O HANDSHAKE
I/O OUTPUT
I/O CONTROL
I/O INPUT
12BITS
4 BITS
CA1
CB2
CA2
126
MAGNETIC TAPE CASSETTE
5 BITS
2 BITS
CB1
10KHZ
60KHZ
12
128
SYSTEM PERIPHERAL CONTROL SELECT UNIT
KEYBOARD
7BITS
Ø1 Ø2
THEMAL PRINTER
DISPLAY CIRCUIT
CB2
5KHZ
2.5KHZ
1.25KHZ
.625KHZ
14
16
STM2
134
132
OUTPUT BUS
HANDSHAKE
136
106
INPUT DATA (DI)
INPUT BUS (DM)
144
146
PERIPHERAL INTERFACE ADAPTER (PIA)
Ø2
R/W
NMI
IRQ
CHIP SELECTS
BUFFER 2
INPUT
130

FIG 4B

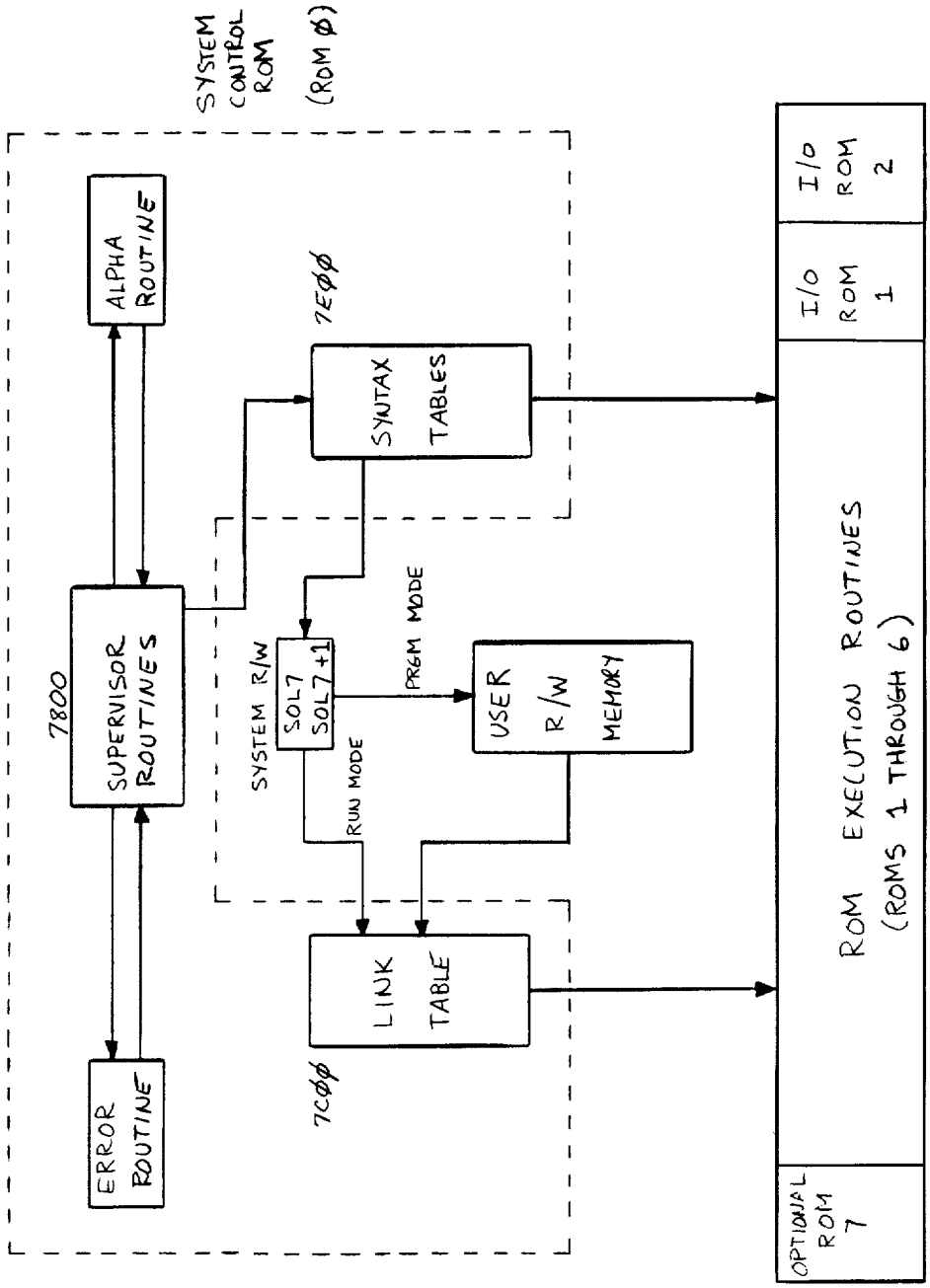

SYSTEM CONTROL ROM (ROM Ø)
ALPHA ROUTINE
7EØØ
SYNTAX TABLES
7800
SUPERVISOR ROUTINES
SYSTEM R/W
SOL7
SOL7+1
PRGM MODE
USER R/W MEMORY
RUN MODE
LINK TABLE
7CØØ
ERROR ROUTINE
I/O ROM 1
I/O ROM 2
ROM EXECUTION ROUTINES (ROMS 1 THROUGH 6)
OPTIONAL ROM 7

FIG. 5

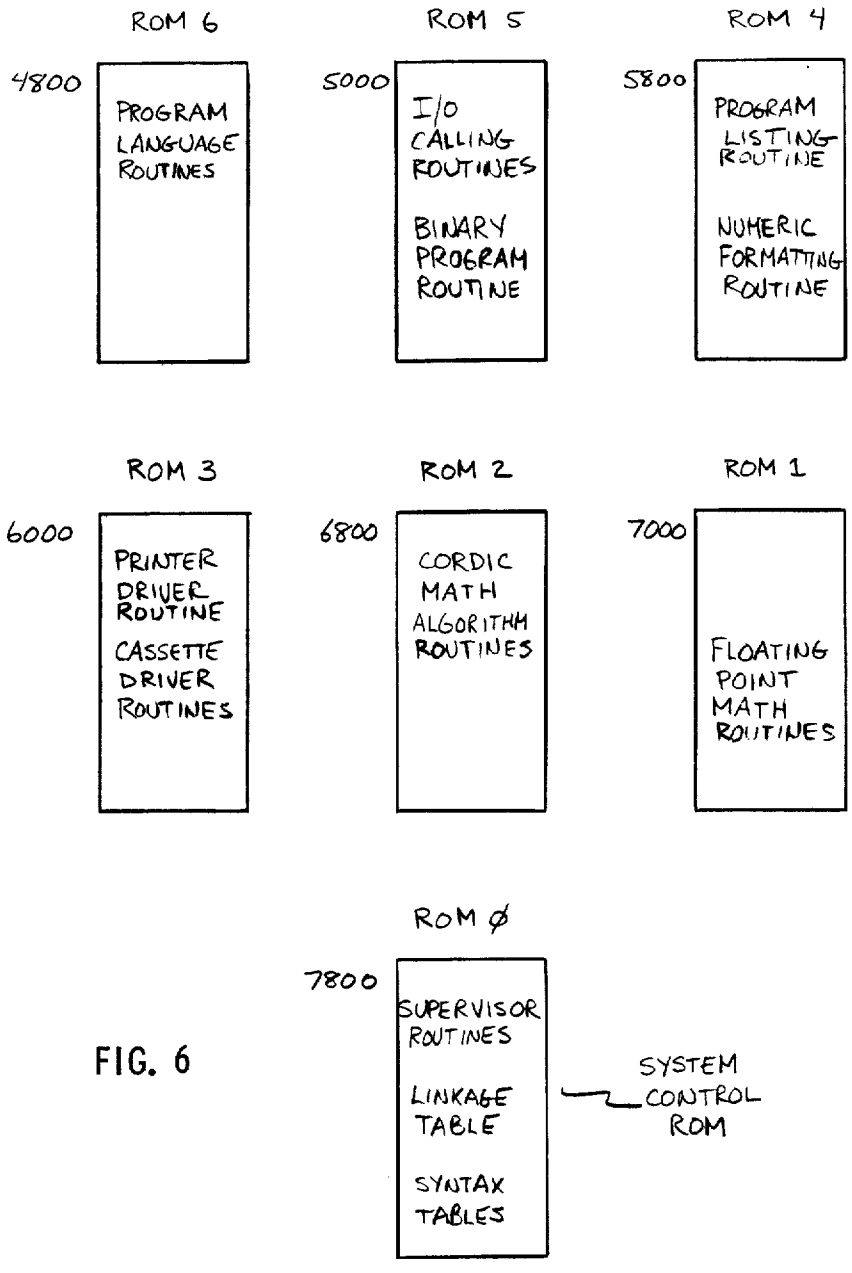
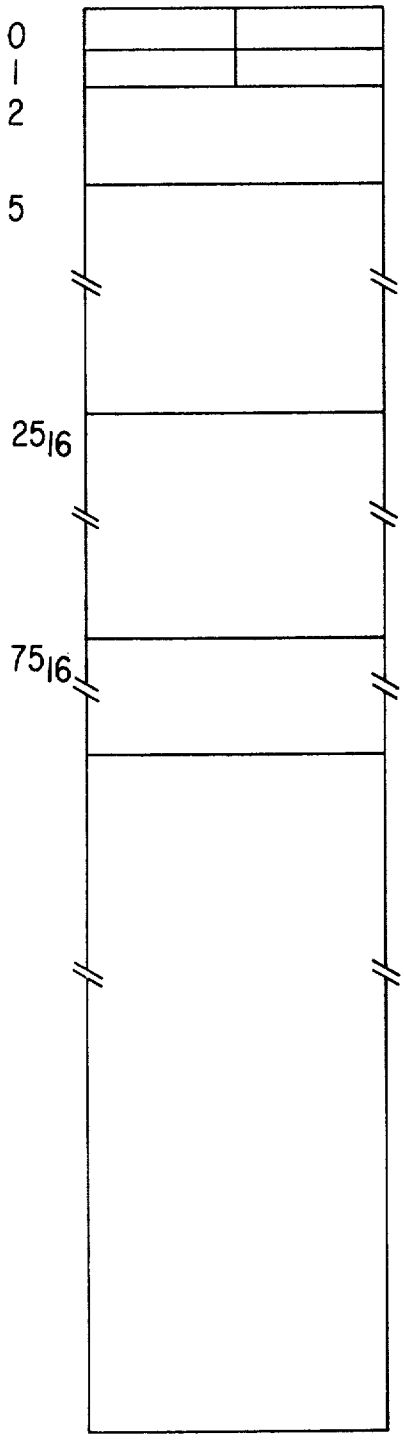

BASIC ROM OVERVIEW
ROM 6
4800
PROGRAM LANGUAGE ROUTINES
ROM 5
5000
I/O CALLING ROUTINES
BINARY PROGRAM ROUTINE
ROM 4
5800
PROGRAM LISTING ROUTINE
NUMERIC FORMATTING ROUTINE
ROM 3
6000
PRINTER DRIVER ROUTINE
CASSETTE DRIVER ROUTINES
ROM 2
6800
CORDIC MATH ALGORITHM ROUTINES
ROM 1
7000
FLOATING POINT MATH ROUTINES
ROM Ø
7800
SUPERVISOR ROUTINES
LINKAGE TABLE
SYNTAX TABLES
SYSTEM CONTROL ROM

FIG. 6

I/O ROM FORMAT



PRIMARY ROM SELECT CODE } IN UPPER
SECONDARY ROM SELECT CODE } FOUR BITS.

JUMP INSTRUCTION FOR INITIALIZATION DURING POWER ON SEQUENCE.

SIXTEEN 16-BIT ADDRESS TO COMMAND EXECUTION ROUTINES FOR ENTRY INTO ROM. FIRST ADDRESS IS TO ALPHA ROUTINE, NEXT TO A, B, C, ETC.

SIXTEEN 5-CHARACTER BLOCKS OF ASCII USED IN USING THE PROGRAM. THE ORDER IS THE SAME AS THE JUMP ADDRESSES. THE FIRST CHARACTER IN EACH BLOCK MUST HAVE ITS M, S, B, SET.

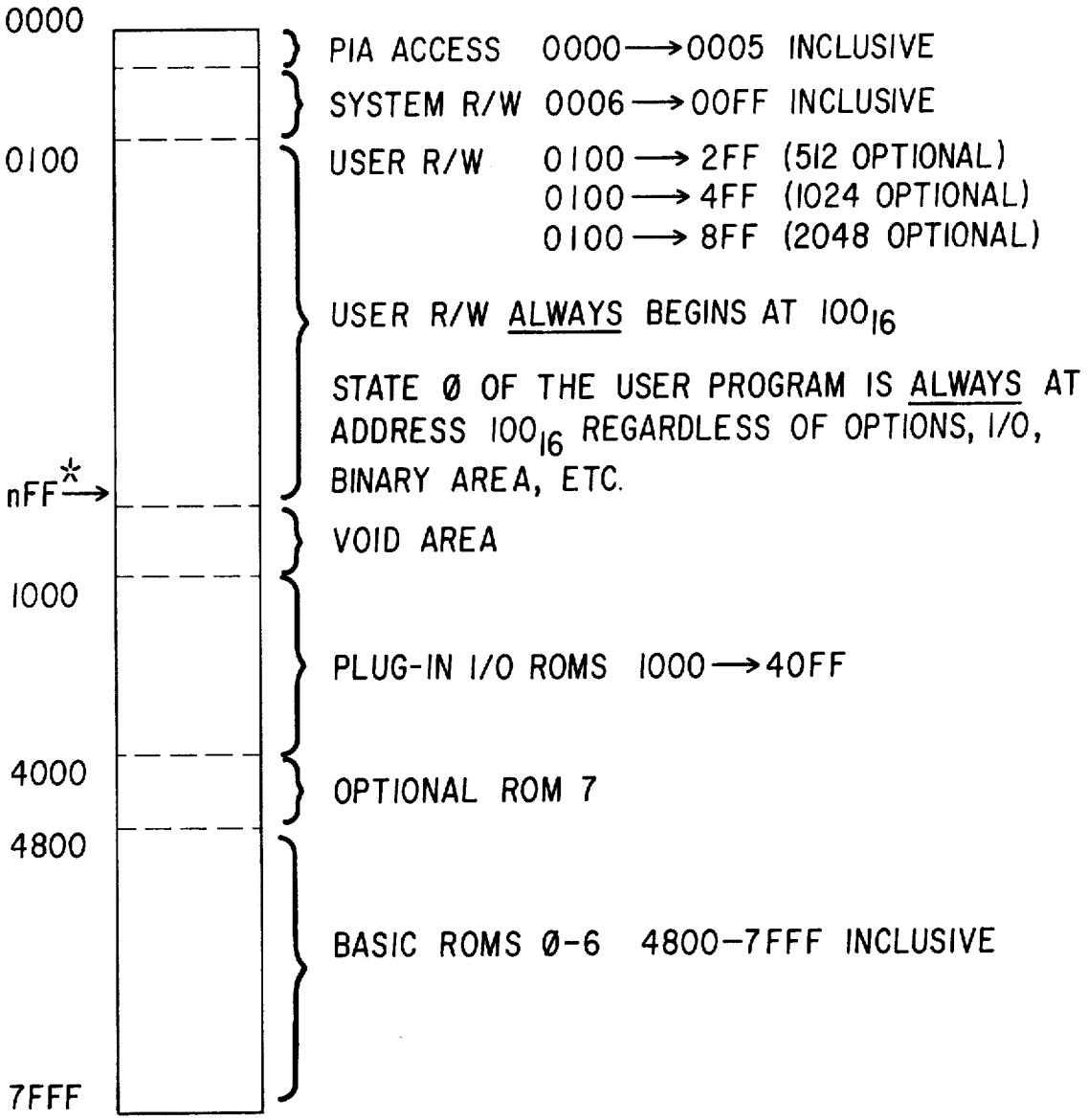
I/O ROM ERROR ASCII. THE MSB OF THE FIRST CHARACTER OF EACH MESSAGE MUST BE SET TO SEPERATE MESSAGES.

REST OF ROM FREE FOR CODE.

FIG 7

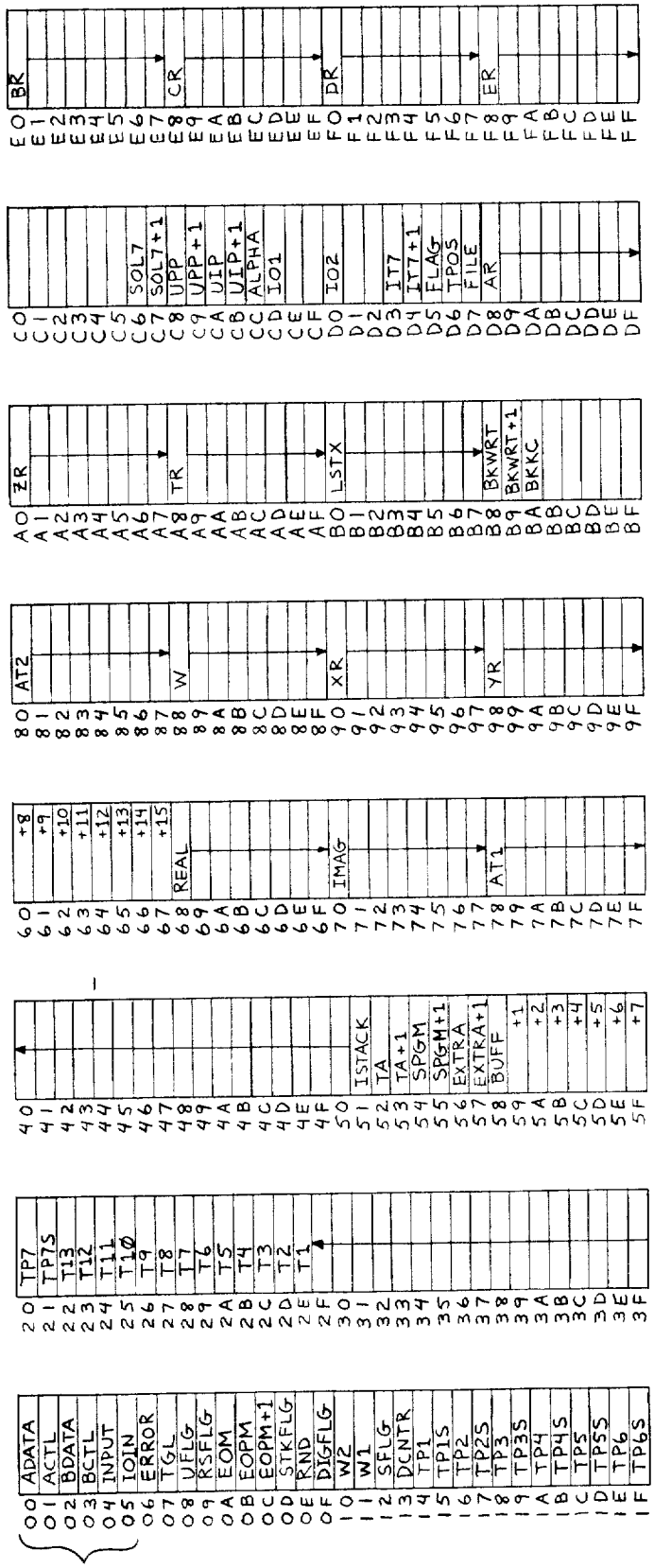
MEMORY ALLOCATION MAP

CALCULATOR MEMORY IS DIVIDED INTO BOTH R/W AND ROM. THE R/W IS SPLIT BETWEEN BASE PAGE (SYSTEM) R/W AND EXPANDABLE USER R/W. ROM IS DIVIDED BETWEEN SYSTEM ROM AND I/O ROM. BELOW IS A GENERAL MAP SHOWING ALL THE MEMORY SPACE. ALL ADDRESSES ARE HEXADECIMAL.



*n=2 FOR 512 OPTION; n=4 FOR 1024 OPTION; n=8 FOR 2048 OPTION

FIG 8



INTERNAL STACK (35 DEEP) ALLOWS FOR :

7 USER SUBROUTINES

7 SYSTEM SUBROUTINES

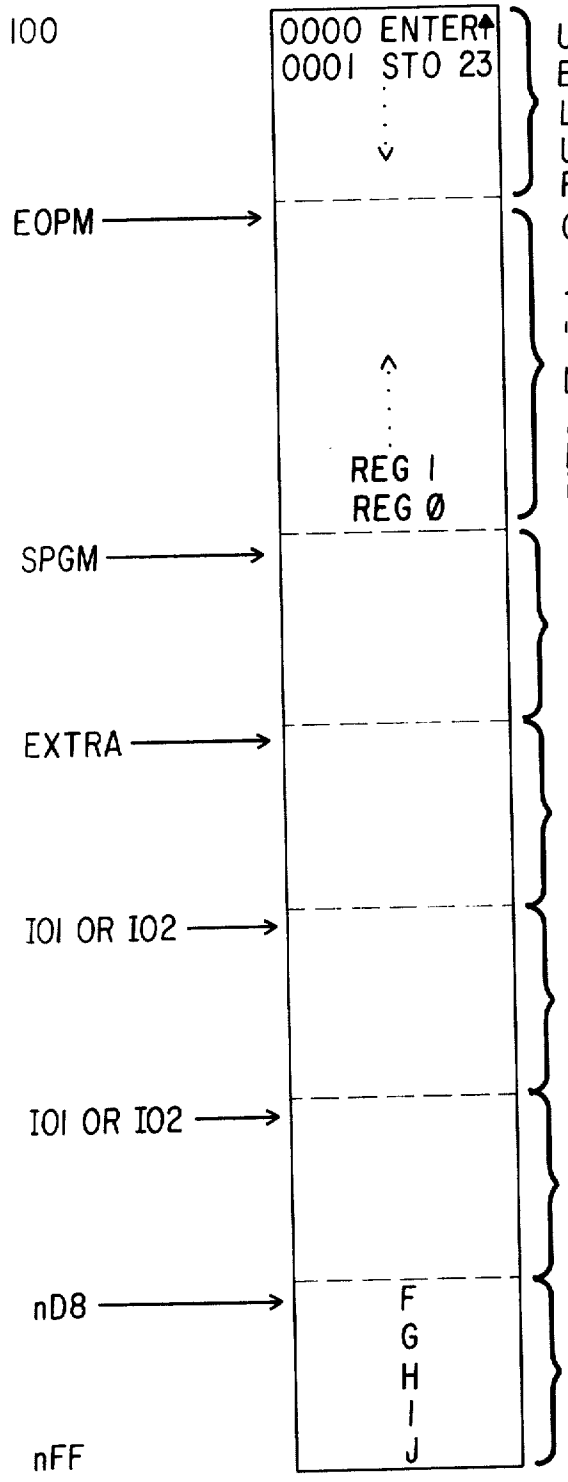
1 SYSTEM INTERRUPT

KEY BUFFER = 12 DEEP

FIG. 9

USER READ-WRITE MEMORY ALLOCATION

USER MEMORY (100_{16} → 2FF OR 4FF OR 8FF DEPENDING UPON THE OPT) IS SHOWN BELOW AS TO ITS USE AND ALLOCATION. ADDRESSES DEPEND UPON OPTION SO "n" IS USED TO INDICATE THEM. (FOR 512, n=2; 1024, n=4; 2048, n=8) USER MEMORY ALWAYS BEGINS AT 100_{16}. ALL ADDRESSES ARE HEXADECIMAL.



USER PROGRAM MEMORY. WILL ALWAYS BE A MULTIPLE OF 8, CAN NEVER BE LESS THAN 8. MAX SIZE DEPENDS UPON HIS I/O SETUP, HIS STORAGE REGISTES, MEMORY OPTION, ETC.

OPTIONAL USER NUMERIC STORAGE REGISTERS. REGISTERS BEGIN WITH Ø AT THE LARGEST ADDRESS AND WORK "BACK". EACH REGISTER IS 8 BYTES LONG, MUST BEGIN AT A MULTIPLE OF 8, AND THE EXPONENT IS ALWAYS THE LOWEST ADDRESS OF THE 8 BYTES. (SEE DESCRIPTIONS OF DATA FORMATS FOR MORE INFORMATION)

OPTIONAL BINARY PROGRAM AREA. ITS START IS INDICATED BY "SPGM" ON BASE PAGE. SIZE MUST BE A MULTIPLE OF 8.

OPTIONAL R/W ALLOCATION FOR THE KATAKANA ROM SLOT (WHICH WILL USE POINTER "EXTRA" EXPLAINED ON BASE PAGE). MUST BE A MULTIPLE OF 8.

OPTIONAL R/W ALLOCATION FOR I/O. THE I/O ROM WITH THE LARGEST STARTING ADDRESS WILL USE THIS AREA. SIZE OF ALLOCATION WILL BE A MULTIPLE OF 8.

OPTIONAL R/W ALLOCATION FOR I/O. THE I/O ROM WITH THE SMALLEST STARTING ADDRESS WILL USE THIS AREA. SIZE OF ALLOCATION WILL BE A MULTIPLE OF 8.

USE REGISTERS F-J. F BEGINS AT nD8 AND THE REST FOLLOW AT 8 BYTES EACH. G=nEØ H=nE8 I=nFØ J=nF8

FIG 10

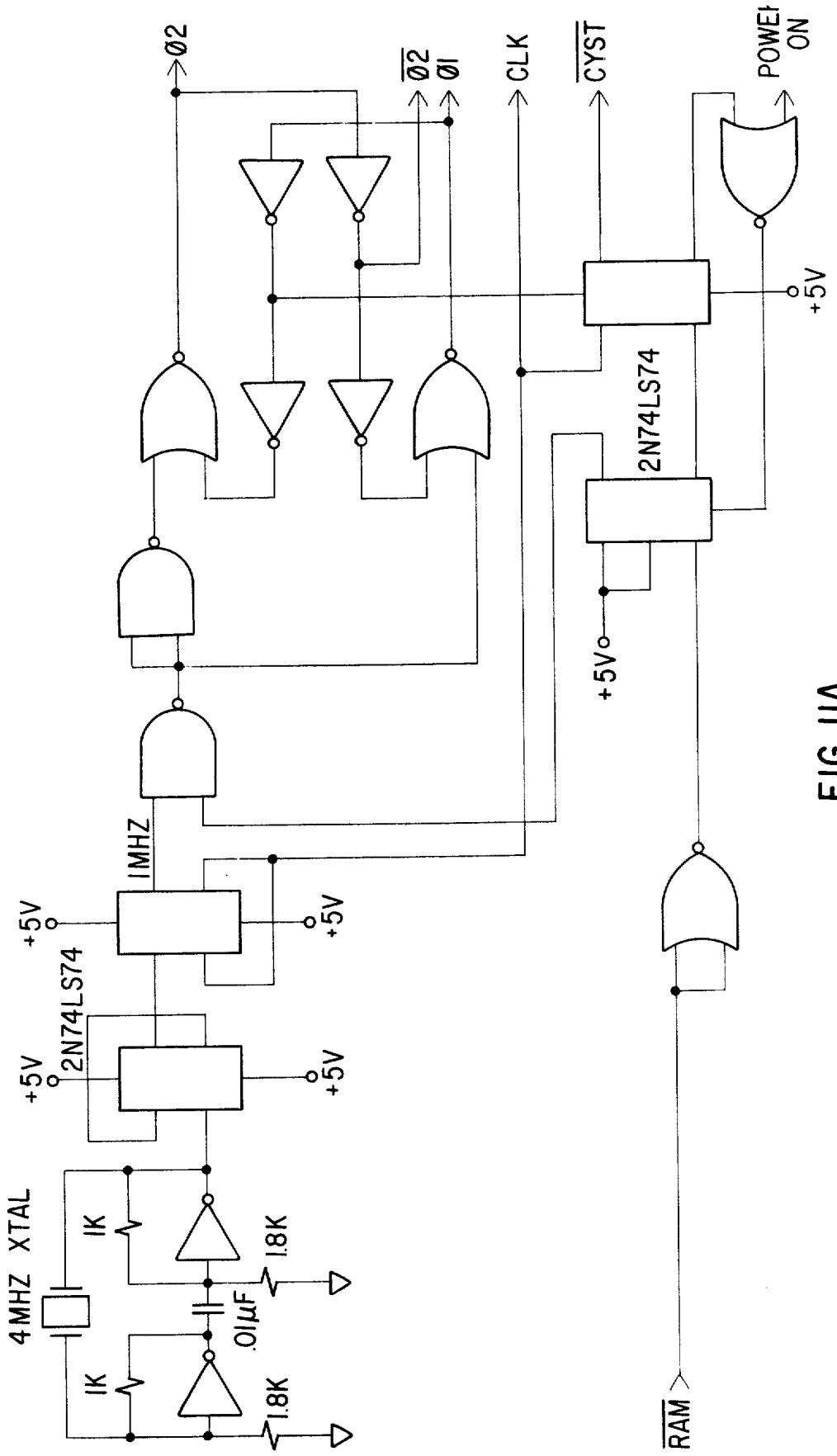

Ø2
Ø2
Ø1
CLK
CYST
POWER ON
+5V
2N74LS74
+5V
+5V
1MHZ
+5V
+5V
2N74LS74
+5V
+5V
4MHZ XTAL
1K
1.8K
.01μF
1K
1.8K
RAM

FIG IIA

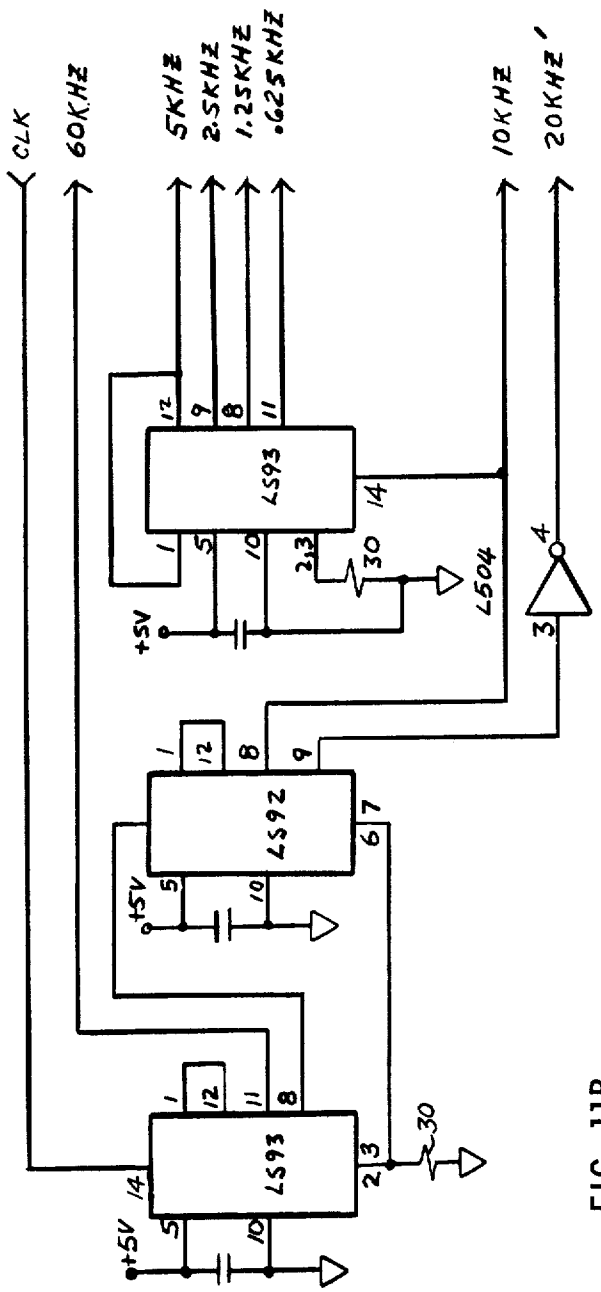

CLK
60KHZ
5KHZ
2.5KHZ
1.25KHZ
.625KHZ
10KHZ
20KHZ'
LS93
LS92
LS04
+5V
30

FIG. 11B

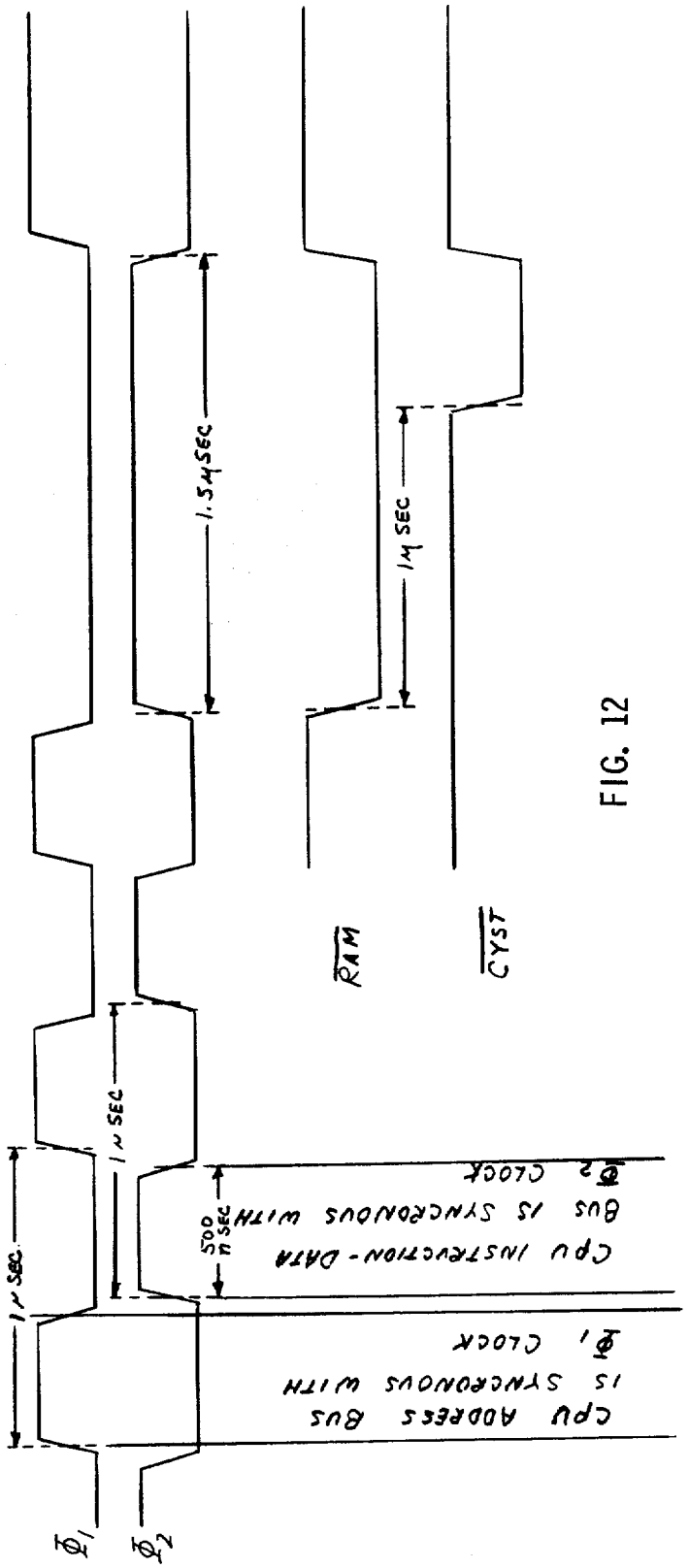

1.54 SEC
1M SEC
RAM
CYST
1M SEC
1M SEC
500 n SEC
CPU INSTRUCTION-DATA BUS IS SYNCRONOUS WITH Φ2 CLOCK
CPU ADDRESS BUS IS SYNCRONOUS WITH Φ1 CLOCK
Φ1
Φ2

FIG. 12

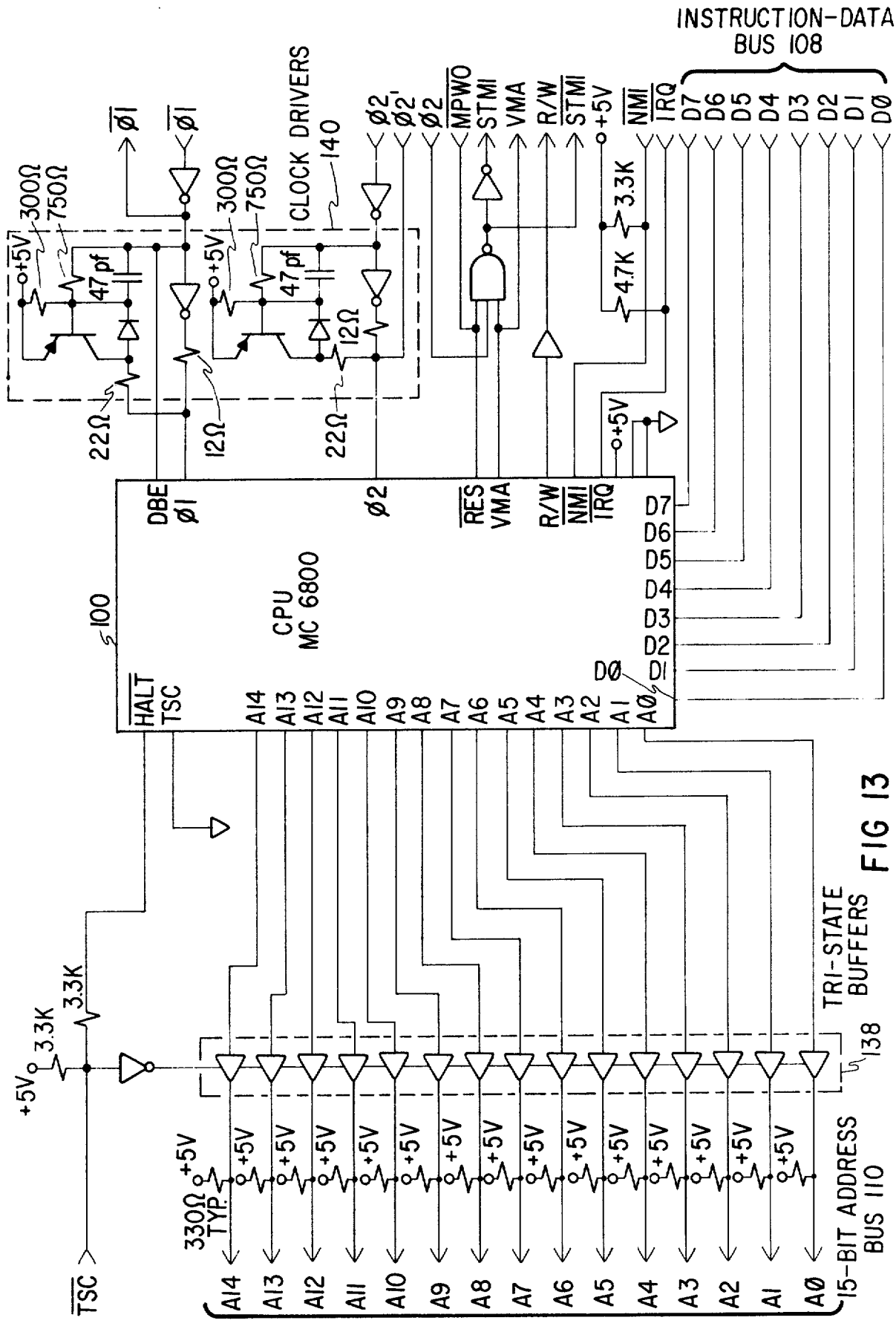

INSTRUCTION-DATA BUS 108
CLOCK DRIVERS
140
CPU MC 6800
100
TRI-STATE BUFFERS
138
15-BIT ADDRESS BUS 110
+5V
3.3K
4.7K
300Ω
750Ω
47 pf
22Ω
12Ω
330Ω TYP.
DBE
HALT
TSC
RES
MPWO
STMI
VMA
R/W
NMI
IRQ
D7
D6
D5
D4
D3
D2
D1
D0
A14
A13
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0

FIG 13

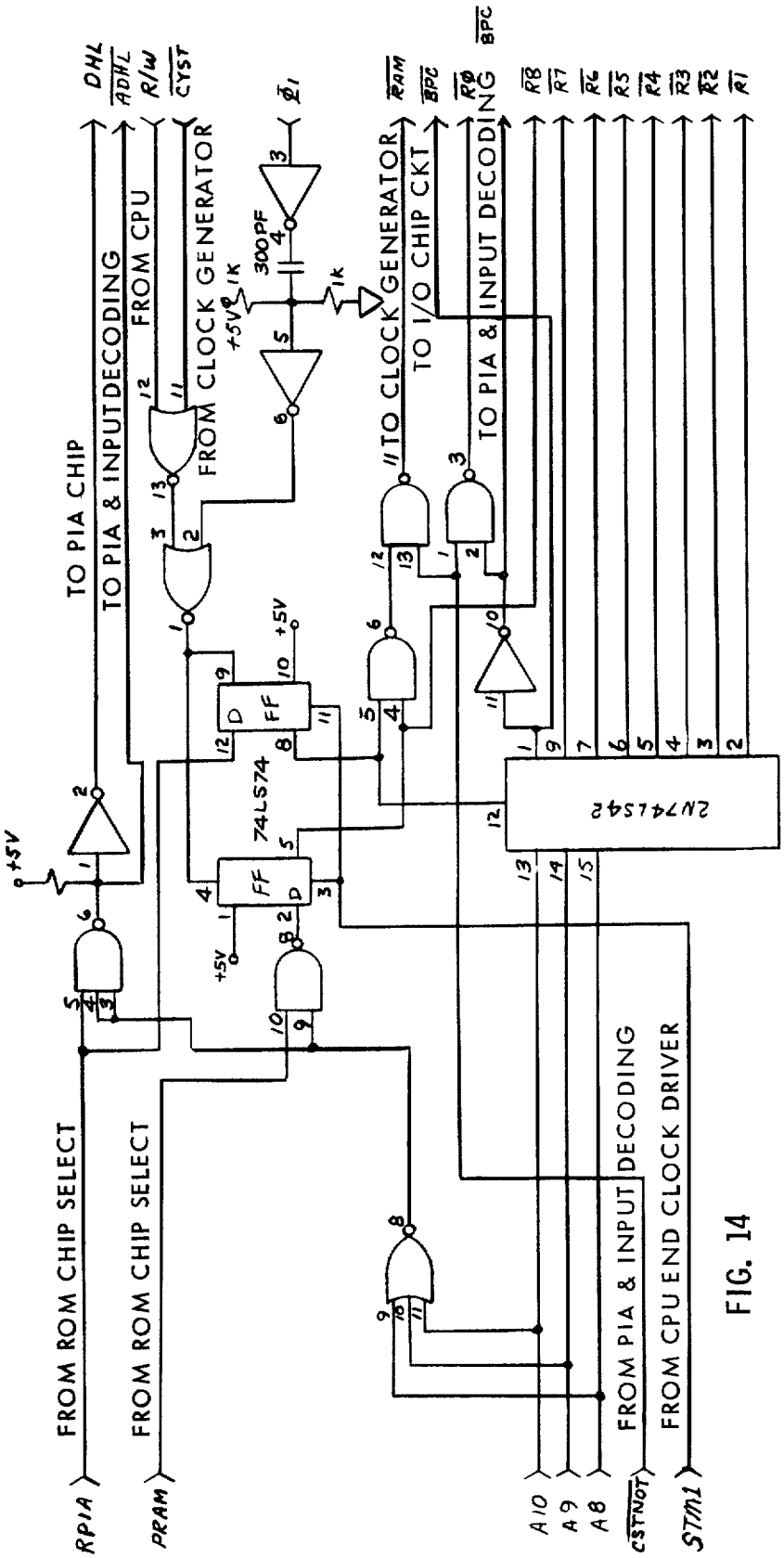

TO PIA CHIP
TO PIA & INPUT DECODING
FROM CPU
FROM CLOCK GENERATOR
TO CLOCK GENERATOR
TO I/O CHIP CKT
TO PIA & INPUT DECODING
DHL
ADHL
R/W
CYST
Ø1
RAM
BPC
R0
R1
R2
R3
R4
R5
R6
R7
R8
300PF
1K
+5V
74LS74
FF
D
2N74LS42
FROM ROM CHIP SELECT
FROM ROM CHIP SELECT
FROM PIA & INPUT DECODING
FROM CPU END CLOCK DRIVER
RPIA
PRAM
A10
A9
A8
CSTNOT
STM1

FIG. 14

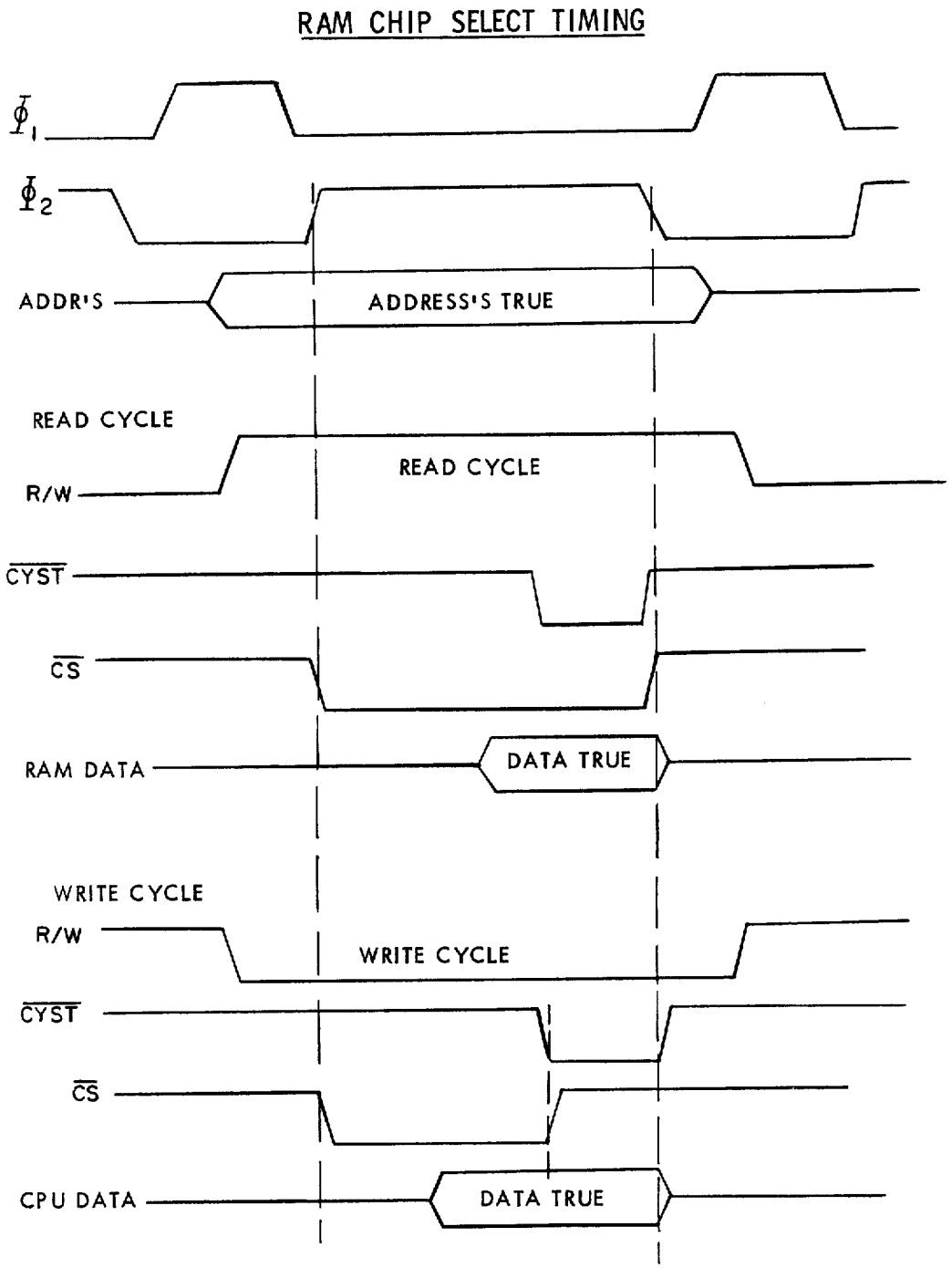

RAM CHIP SELECT TIMING
ϕ_1
ϕ_2
ADDR'S
ADDRESS'S TRUE
READ CYCLE
R/W
READ CYCLE
$\overline{CYST}$
$\overline{CS}$
RAM DATA
DATA TRUE
WRITE CYCLE
R/W
WRITE CYCLE
$\overline{CYST}$
$\overline{CS}$
CPU DATA
DATA TRUE

FIG. 15

FIG 16

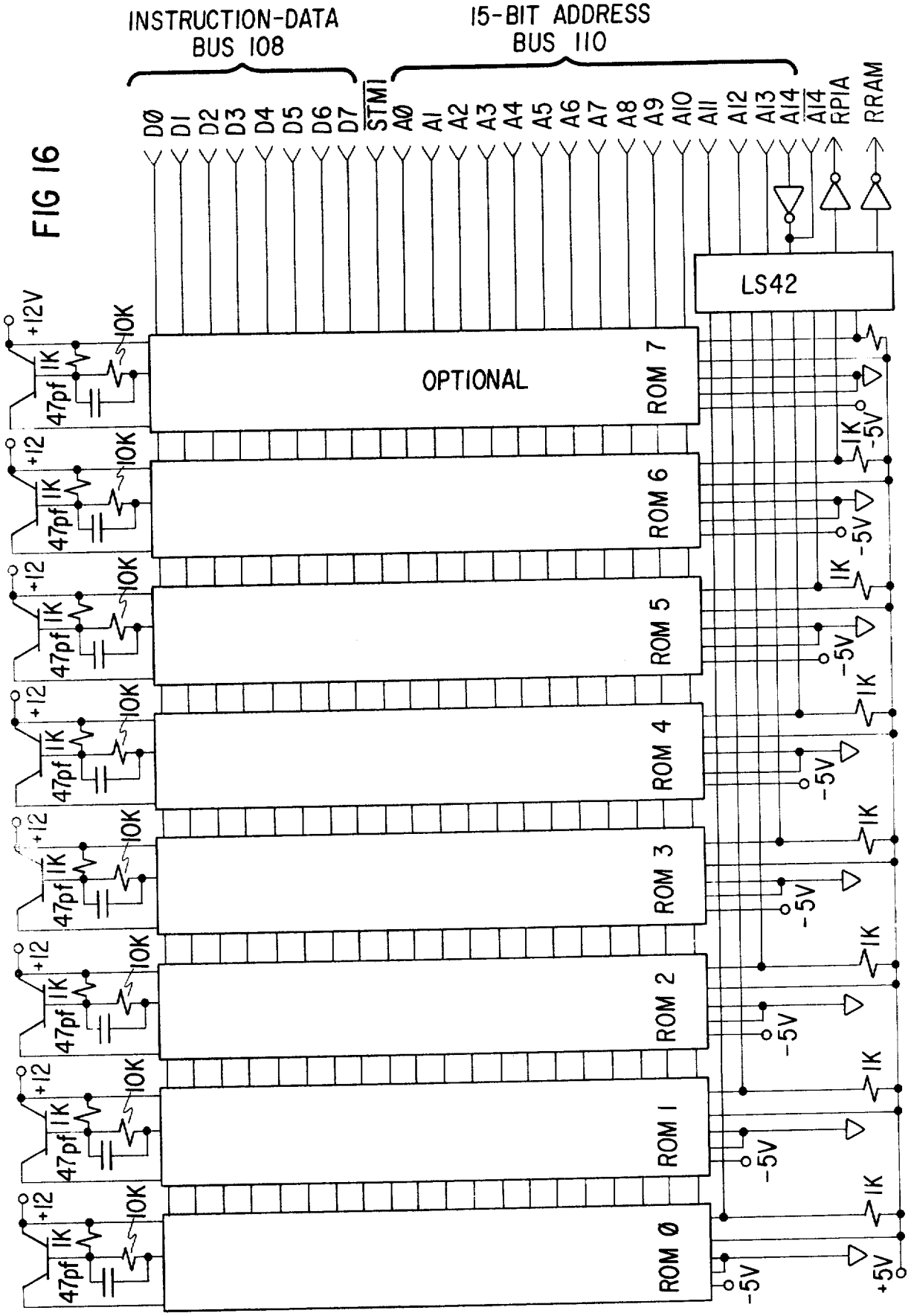
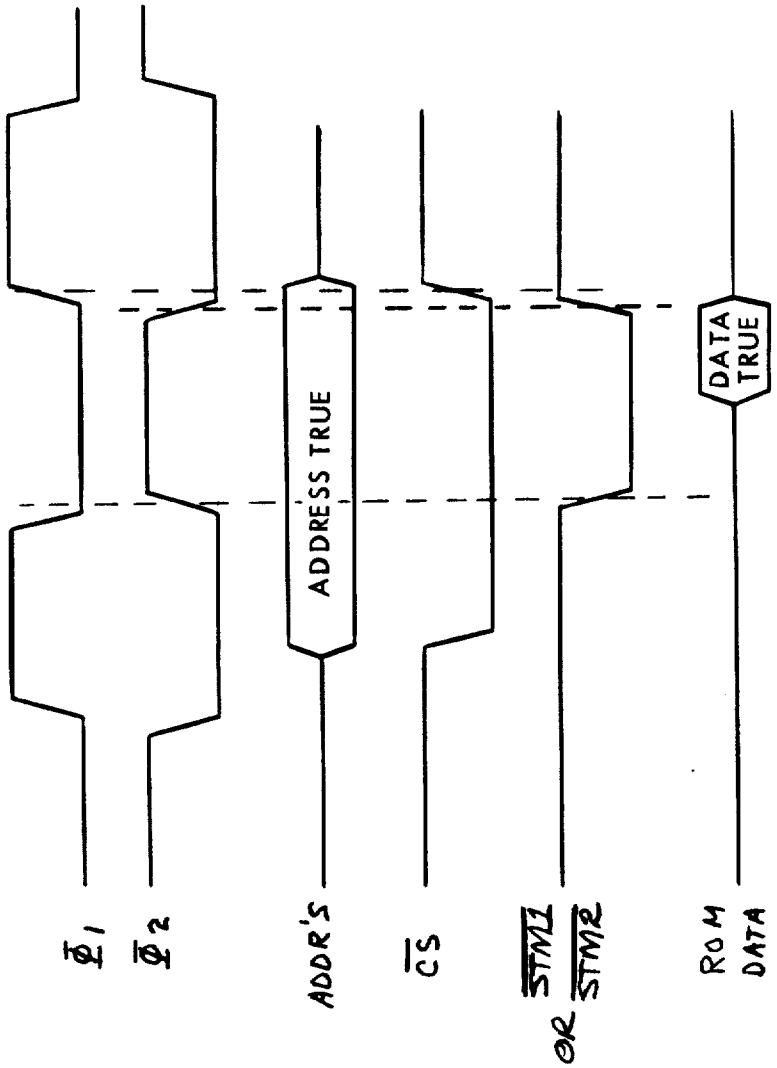

INSTRUCTION-DATA BUS 108
15-BIT ADDRESS BUS 110
D0 D1 D2 D3 D4 D5 D6 D7
STM1
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A14 RPIA RRAM
LS42
OPTIONAL
ROM 7
ROM 6
ROM 5
ROM 4
ROM 3
ROM 2
ROM 1
ROM 0
+12V
+12
47pf
1K
10K
-5V
+5V

FIG. 17


ADDRESS TRUE
DATA TRUE
$\overline{\Phi 1}$
$\overline{\Phi 2}$
ADDR'S
$\overline{CS}$
$\overline{STM1}$ OR $\overline{STM2}$
ROM DATA

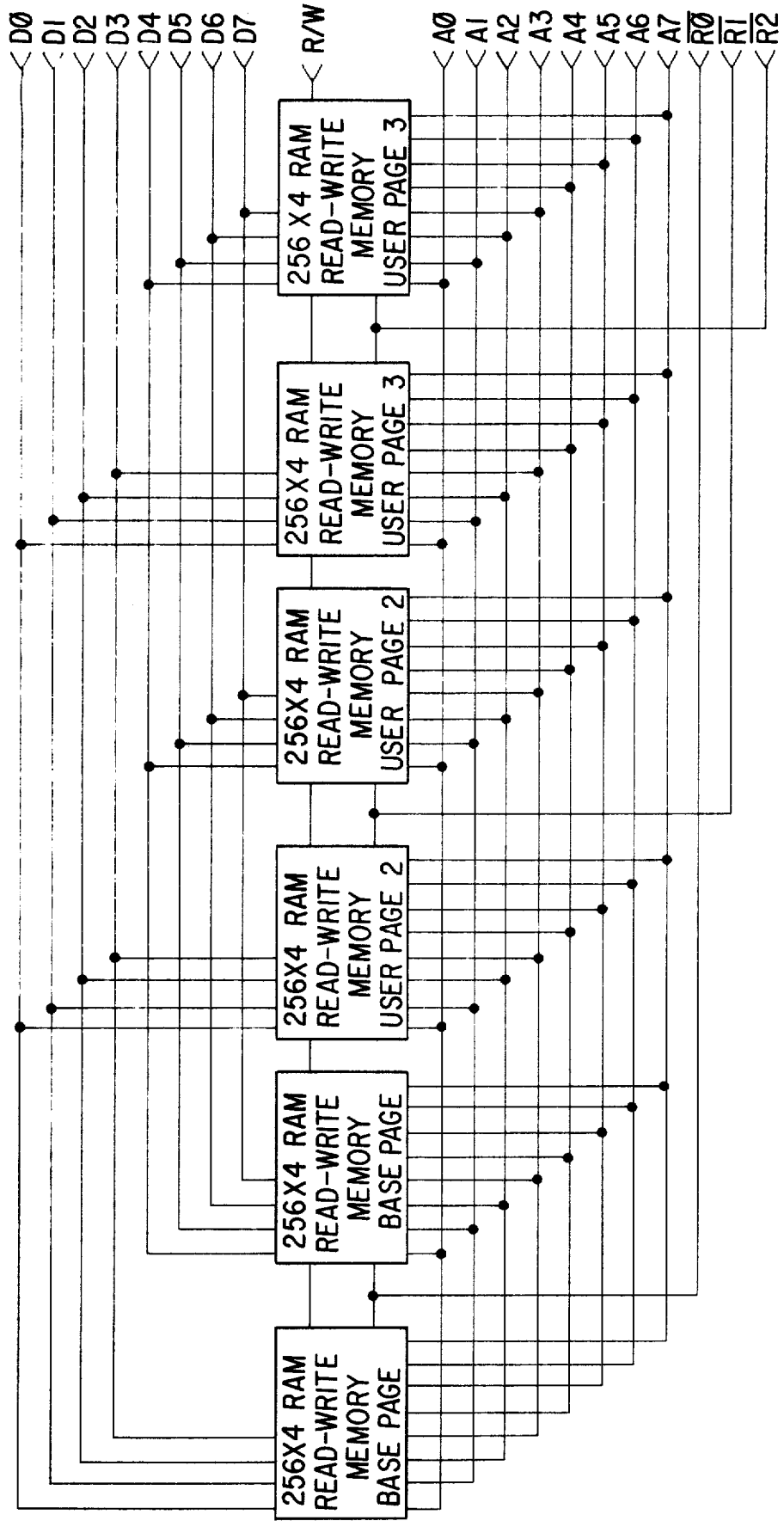

D0
D1
D2
D3
D4
D5
D6
D7
R/W
A0
A1
A2
A3
A4
A5
A6
A7
R0
R1
R2
256 X 4 RAM READ-WRITE MEMORY USER PAGE 3
256X4 RAM READ-WRITE MEMORY USER PAGE 3
256X4 RAM READ-WRITE MEMORY USER PAGE 2
256X4 RAM READ-WRITE MEMORY USER PAGE 2
256X4 RAM READ-WRITE MEMORY BASE PAGE
256X4 RAM READ-WRITE MEMORY BASE PAGE

FIG 18

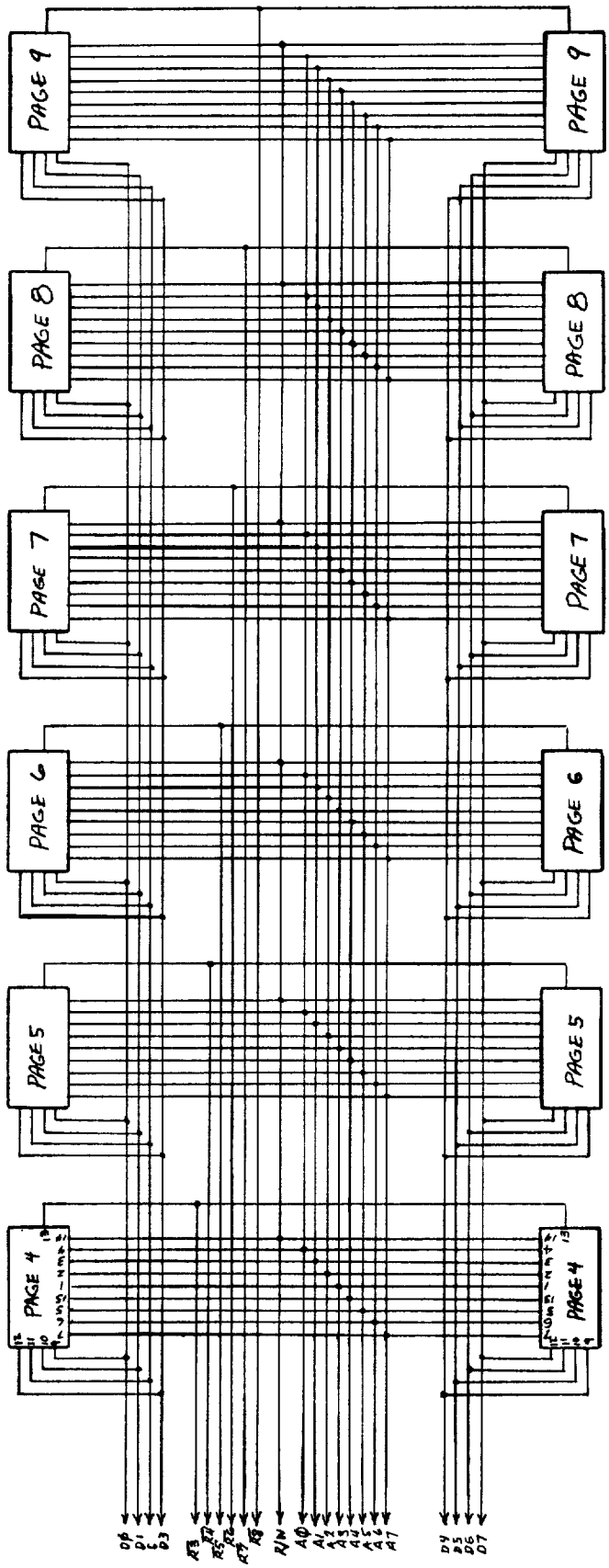

PAGE 9
PAGE 9
PAGE 8
PAGE 8
PAGE 7
PAGE 7
PAGE 6
PAGE 6
PAGE 5
PAGE 5
PAGE 4
PAGE 4

FIG. 19

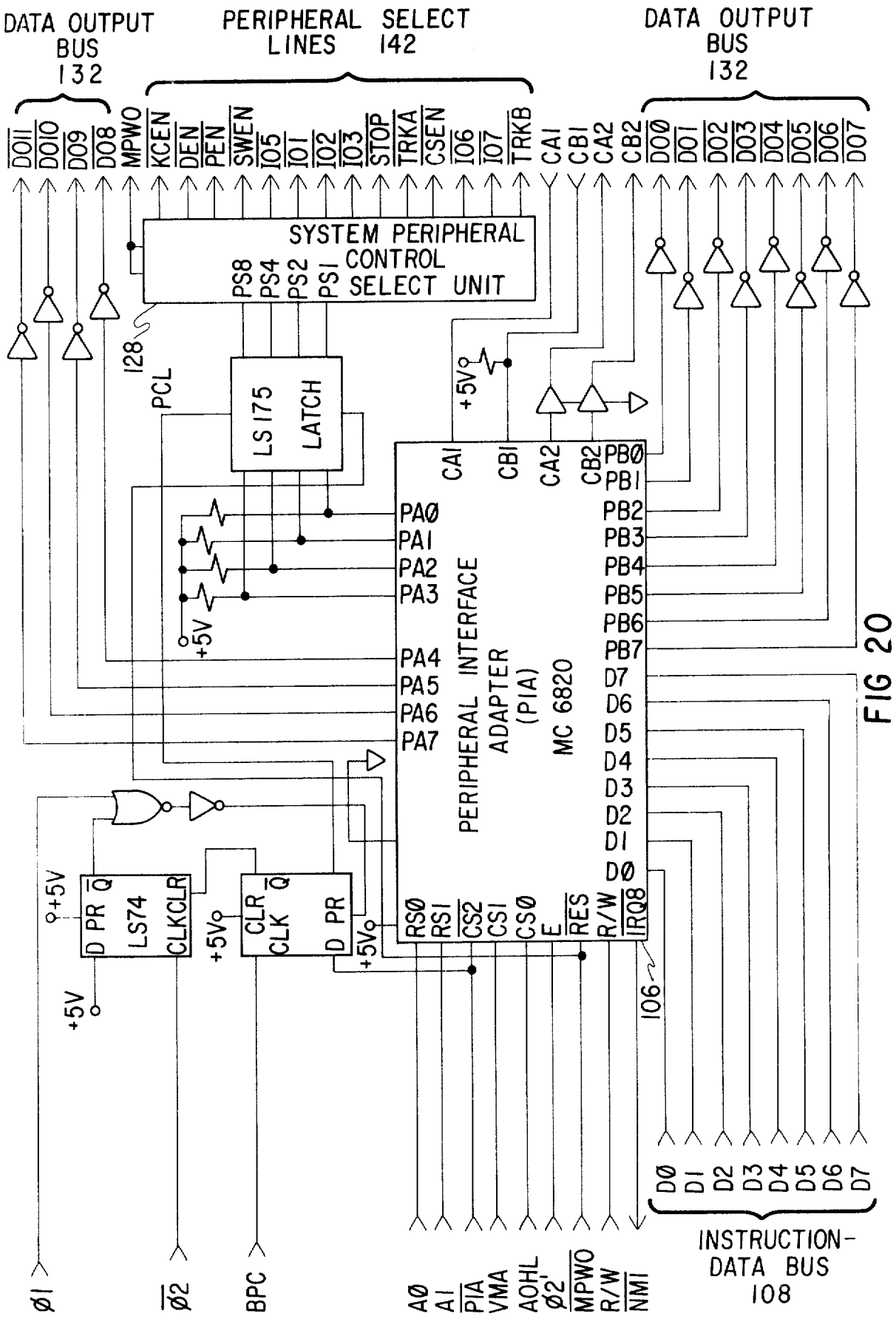

DATA OUTPUT BUS 132
PERIPHERAL SELECT LINES 142
DATA OUTPUT BUS 132
DO11 DO10 DO9 DO8 MPWO KCEN DEN PEN SWEN IO5 IO1 IO2 IO3 STOP TRKA CSEN IO6 IO7 TRKB CA1 CB1 CA2 CB2 DO0 DO1 DO2 DO3 DO4 DO5 DO6 DO7
SYSTEM PERIPHERAL CONTROL SELECT UNIT
PS8 PS4 PS2 PS1
128
PCL
LS 175 LATCH
+5V
CA1 CB1 CA2 CB2 PB0 PB1 PB2 PB3 PB4 PB5 PB6 PB7
PA0 PA1 PA2 PA3 PA4 PA5 PA6 PA7
PERIPHERAL INTERFACE ADAPTER (PIA) MC 6820
D7 D6 D5 D4 D3 D2 D1 D0
RS0 RS1 CS2 CS1 CS0 E RES R/W IRQ8
106
D PR Q LS74 CLK CLR
CLR Q CLK D PR
Ø1 Ø2 BPC A0 A1 PIA VMA AOHL Ø2' MPWO R/W NMI
D0 D1 D2 D3 D4 D5 D6 D7
INSTRUCTION-DATA BUS 108

FIG 20

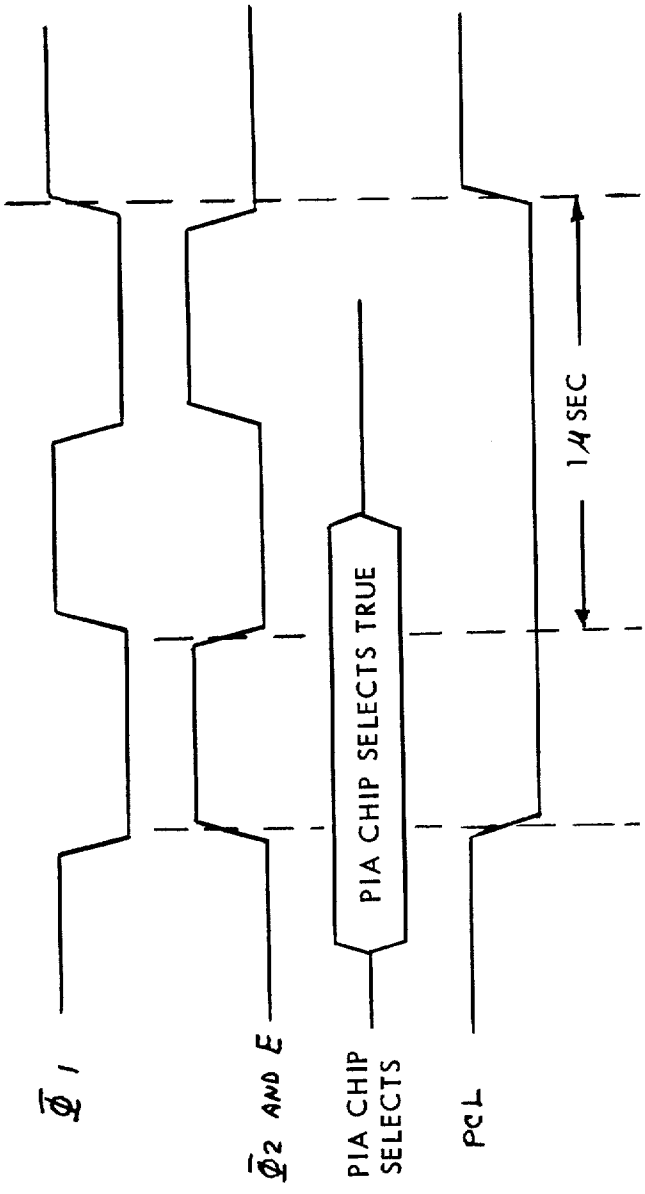

1.4 SEC
PIA CHIP SELECTS TRUE
$\bar{\Phi}_1$
$\bar{\Phi}_2$ AND E
PIA CHIP SELECTS
PCL

FIG. 21

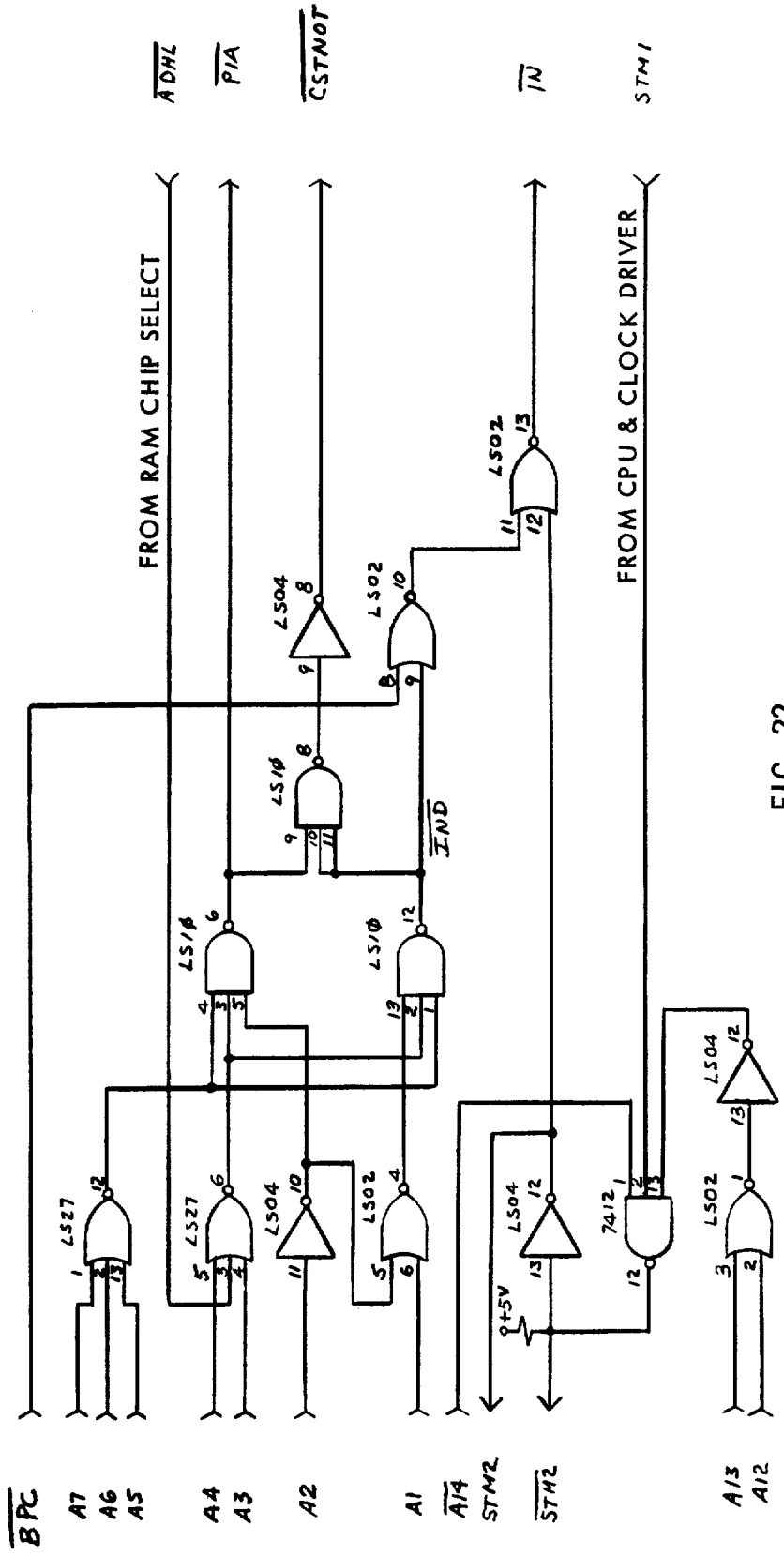

FROM RAM CHIP SELECT
FROM CPU & CLOCK DRIVER
ADHL
PIA
CSTNOT
IN
STM1
END
+5V
LS27
LS04
LS02
LS10
LS08
7412
BPC
A7
A6
A5
A4
A3
A2
A1
A14
STM2
STM2
A13
A12

FIG. 22

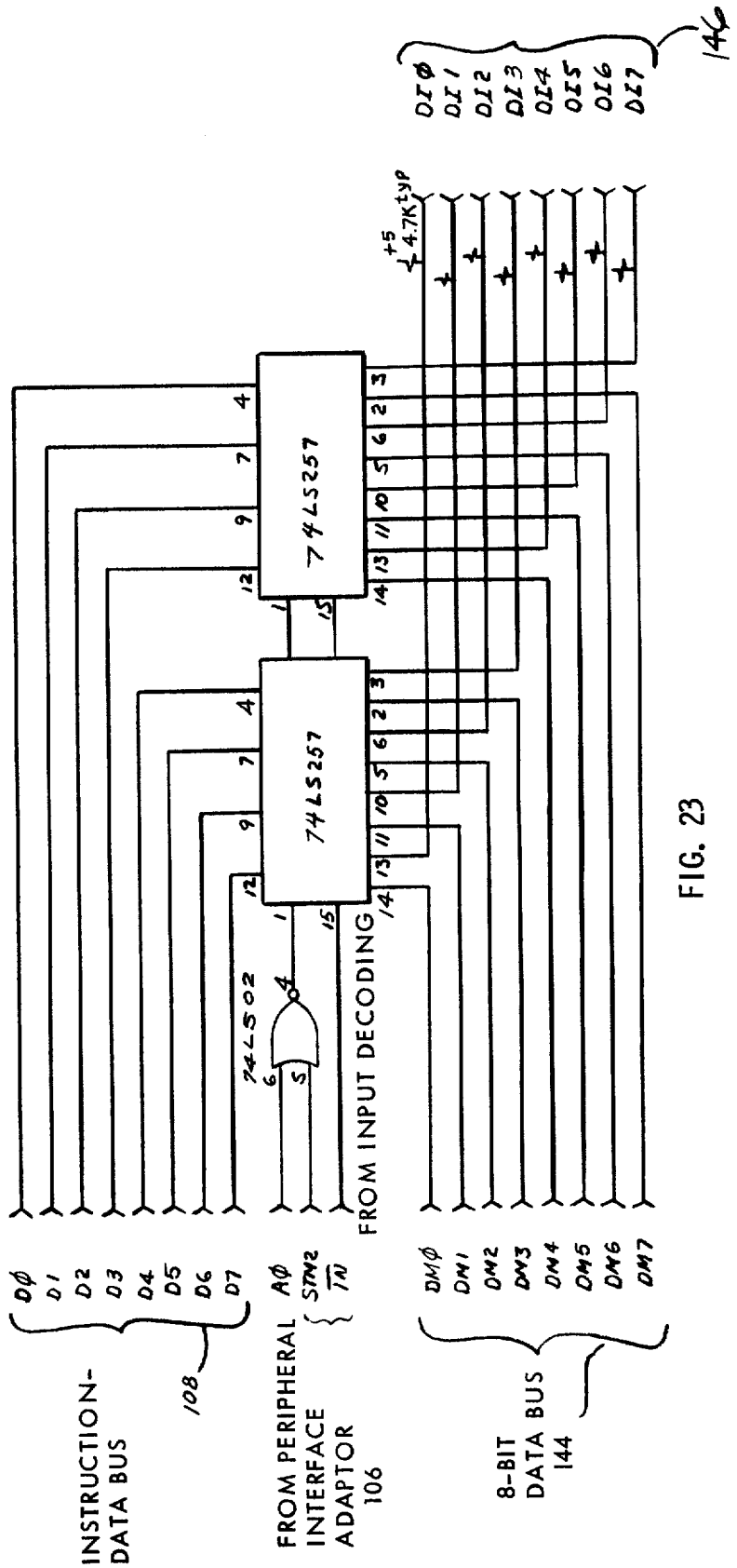

DI0
DI1
DI2
DI3
DI4
DI5
DI6
DI7
146
+5
4.7K typ
74LS257
74LS257
74LS02
FROM INPUT DECODING
D0
D1
D2
D3
D4
D5
D6
D7
108
INSTRUCTION-DATA BUS
A0
STN2
IN
FROM PERIPHERAL INTERFACE ADAPTOR 106
DM0
DM1
DM2
DM3
DM4
DM5
DM6
DM7
8-BIT DATA BUS 144

FIG. 23

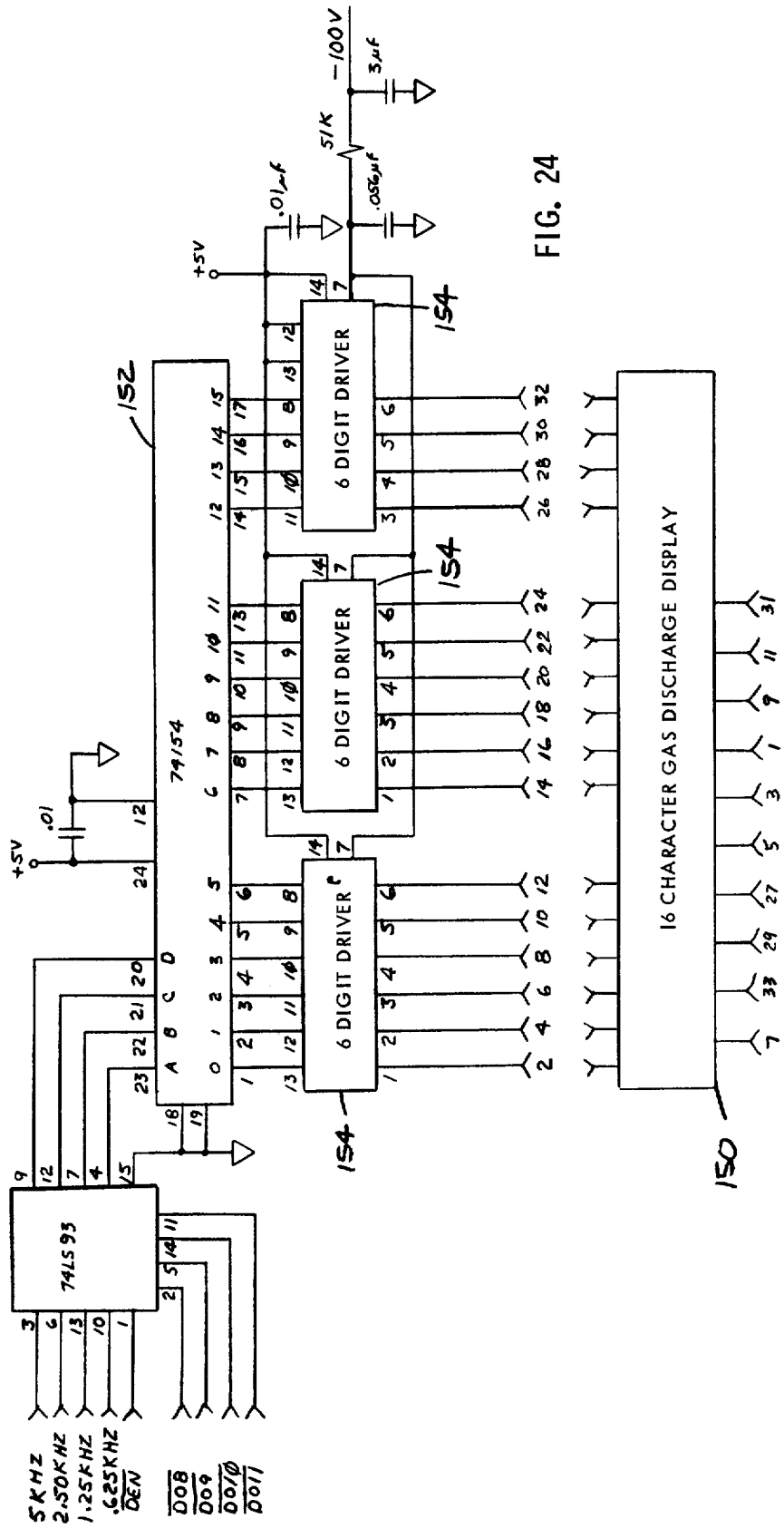

5KHZ
2.50KHZ
1.25KHZ
.625KHZ
DEN
D08
D09
D010
D011
74LS93
74154
+5V
.01
6 DIGIT DRIVER
16 CHARACTER GAS DISCHARGE DISPLAY
150
152
154
-100V
3μf
51K
.056μf
.01μf

FIG. 24

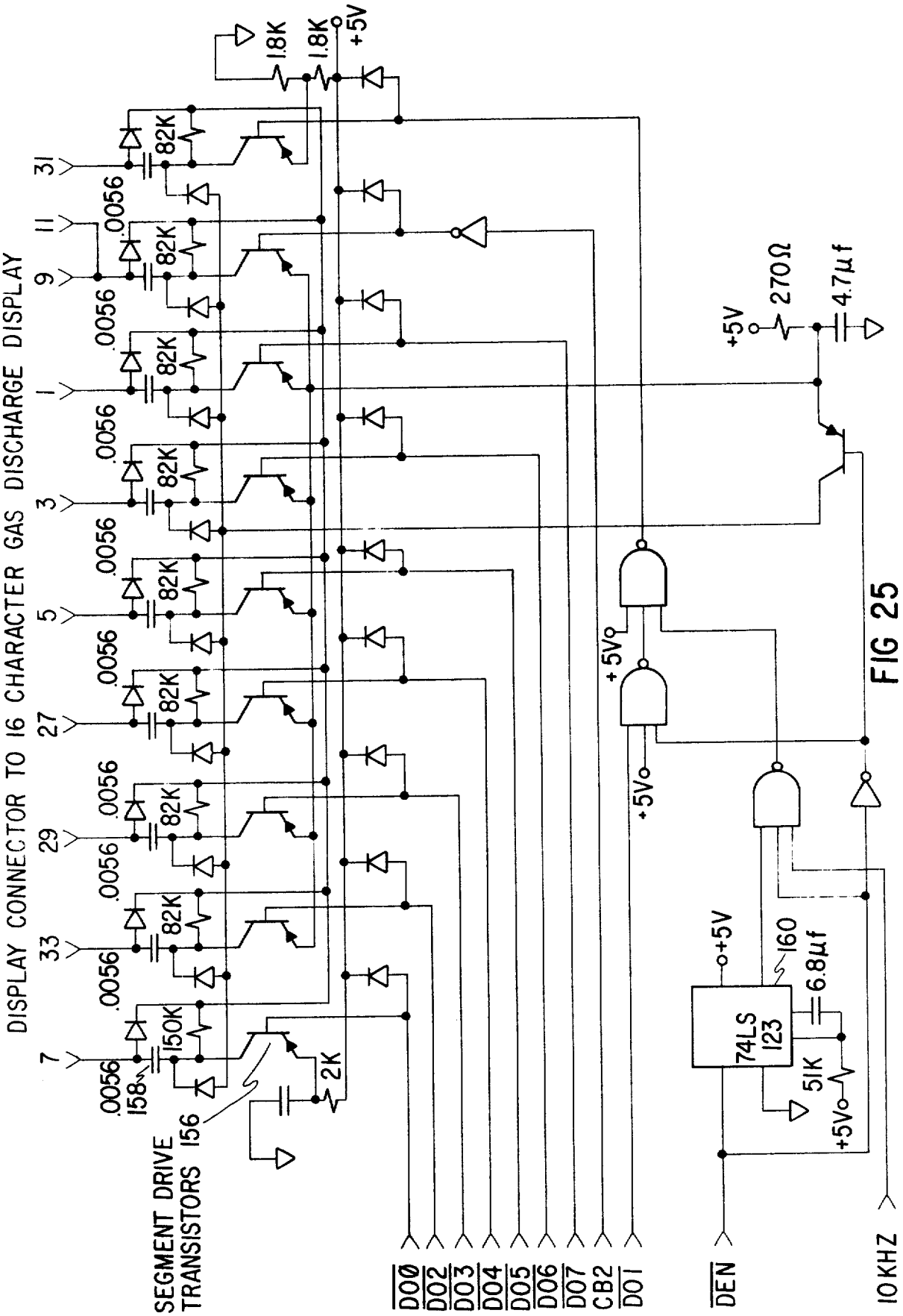
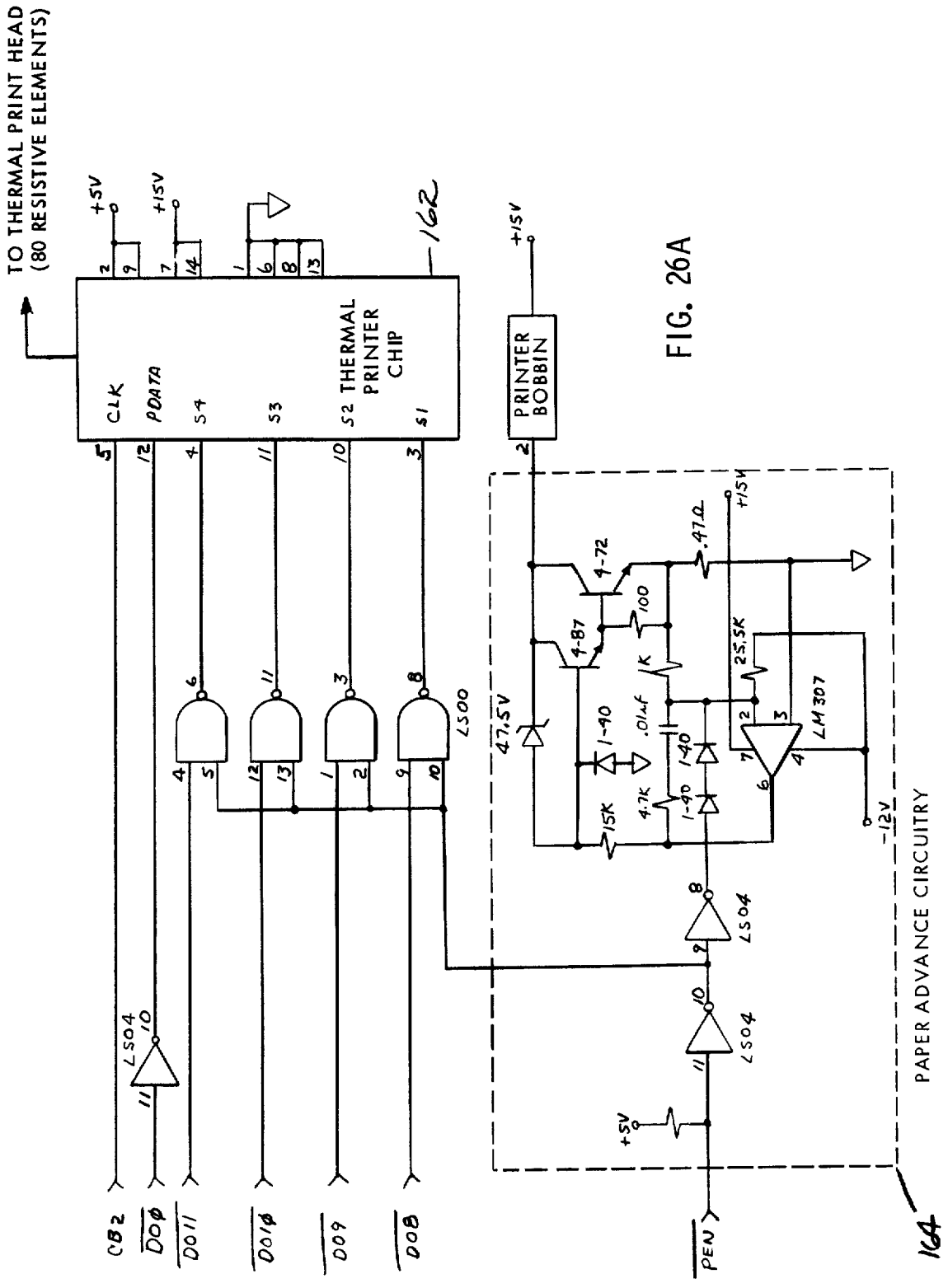

DISPLAY CONNECTOR TO 16 CHARACTER GAS DISCHARGE DISPLAY
SEGMENT DRIVE TRANSISTORS 156
158
150K
2K
82K
.0056
.0056
1.8K
1.8K
+5V
270Ω
4.7μf
+5V
+5V
+5V
74LS 123
160
6.8μf
51K
+5V
DEN
10 KHZ
D00
D02
D03
D04
D05
D06
D07
CB2
D01
7
33
29
27
5
3
1
9
11
31

FIG 25


TO THERMAL PRINT HEAD
(80 RESISTIVE ELEMENTS)
+5V
+15V
162
THERMAL PRINTER CHIP
CLK
PDATA
S4
S3
S2
S1
PRINTER BOBBIN
+15V
FIG. 26A
LS00
LS04
LS04
LS04
4-72
4-87
100
.47Ω
K
25.5K
47.5V
1-90
.01uf
1-90
1-90
4.7K
15K
LM307
+15V
-12V
+5V
PAPER ADVANCE CIRCUITRY
164
CB2
DOØ
DO11
DO10
DO9
DO8
PEN

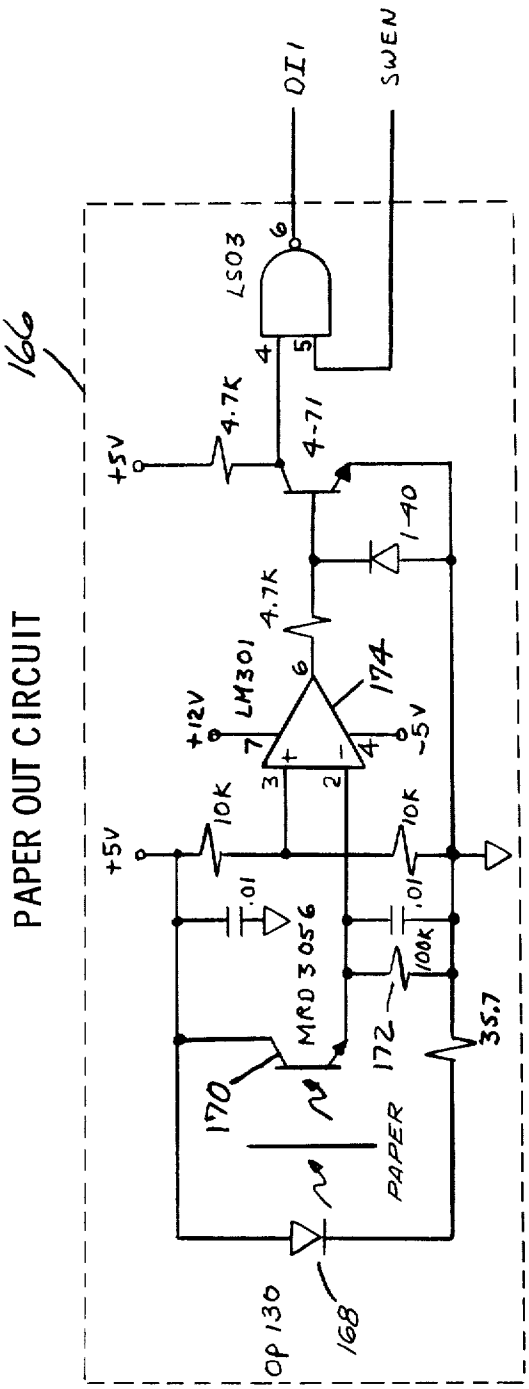

PAPER OUT CIRCUIT
166
OI1
SWEN
LS03
6
4
5
4.7K
4.7K
+5V
1-40
4.7K
LM301
174
+12V
-5V
7
3
2
10K
10K
+5V
.01
.01
100K
MRD 3056
172
35.7
170
PAPER
OP 130
168

FIG. 26B

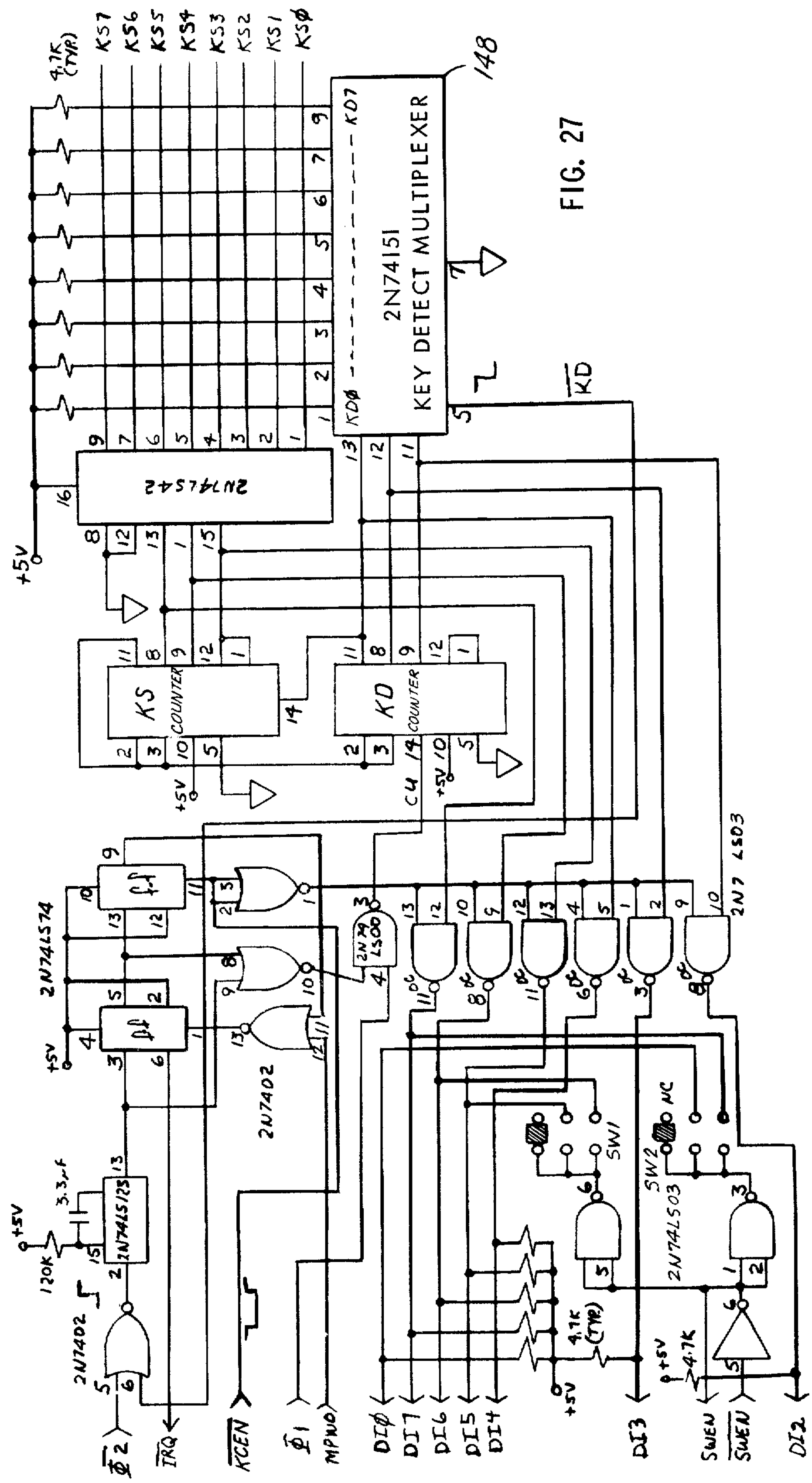

2N74151
KEY DETECT MULTIPLEXER
148
2N74LS42
KS COUNTER
KD COUNTER
2N74LS74
2N7402
2N74LS123
2N74LS03
SW1
SW2
IRQ
KCEN
MPWO
DIØ
DI7
DI6
DI5
DI4
DI3
SWEN
DI2

FIG. 27

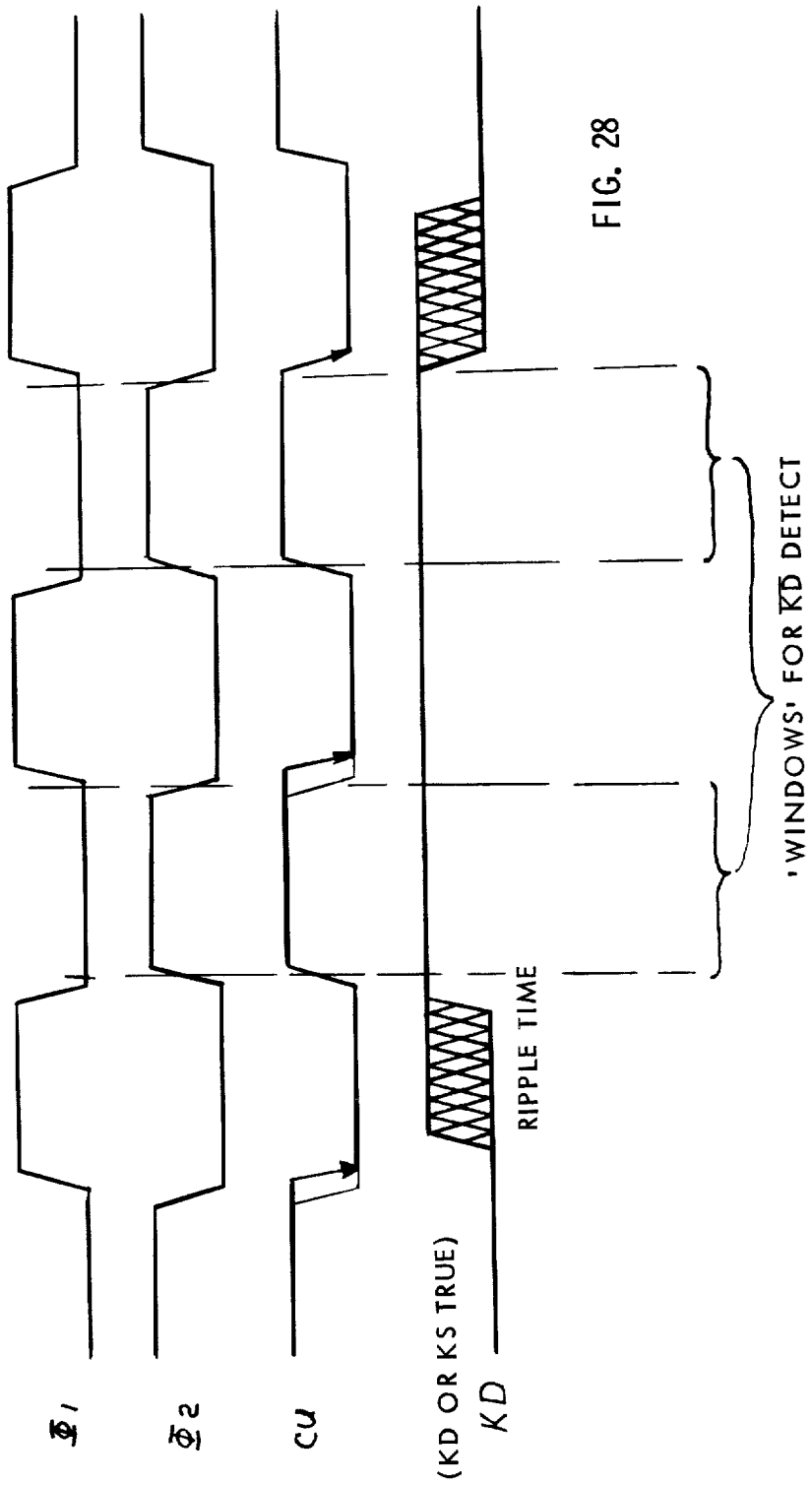


FIG. 28

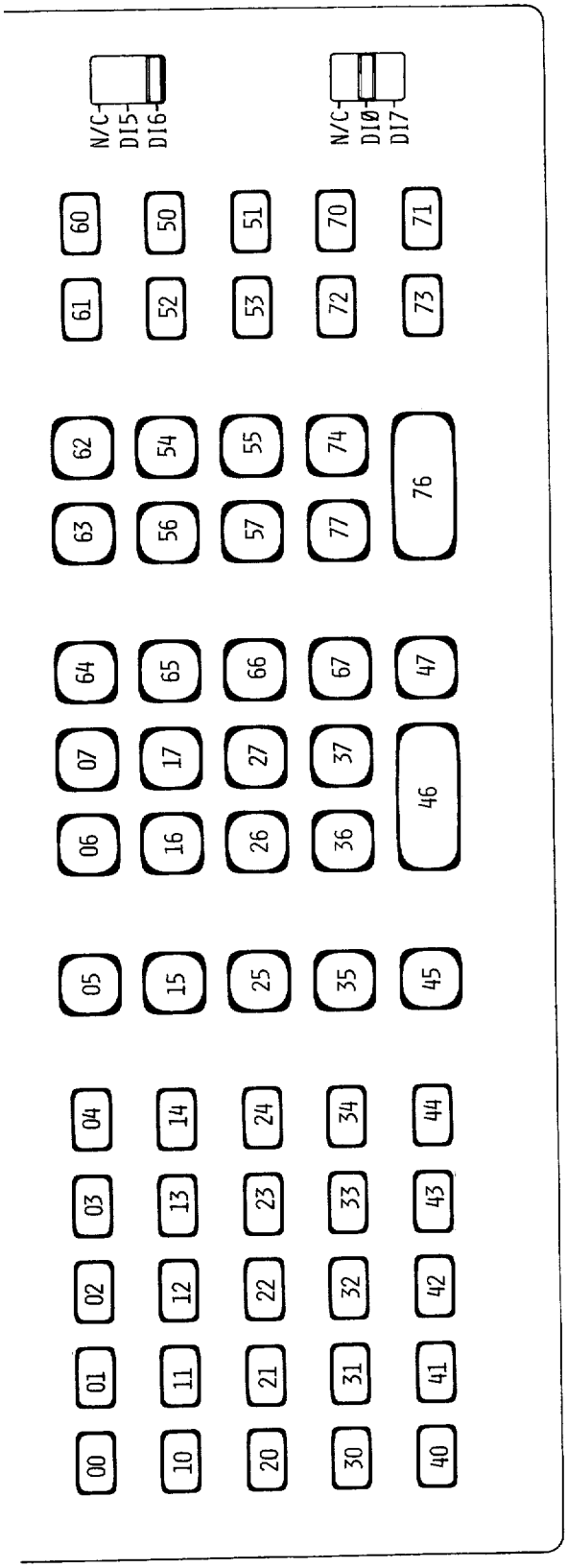

N/C
D15
D16
N/C
D10
D17
60
50
51
70
71
61
52
53
72
73
62
54
55
74
76
63
56
57
77
64
65
66
67
47
07
17
27
37
46
06
16
26
36
05
15
25
35
45
04
14
24
34
44
03
13
23
33
43
02
12
22
32
42
01
11
21
31
41
00
10
20
30
40

FIG. 29

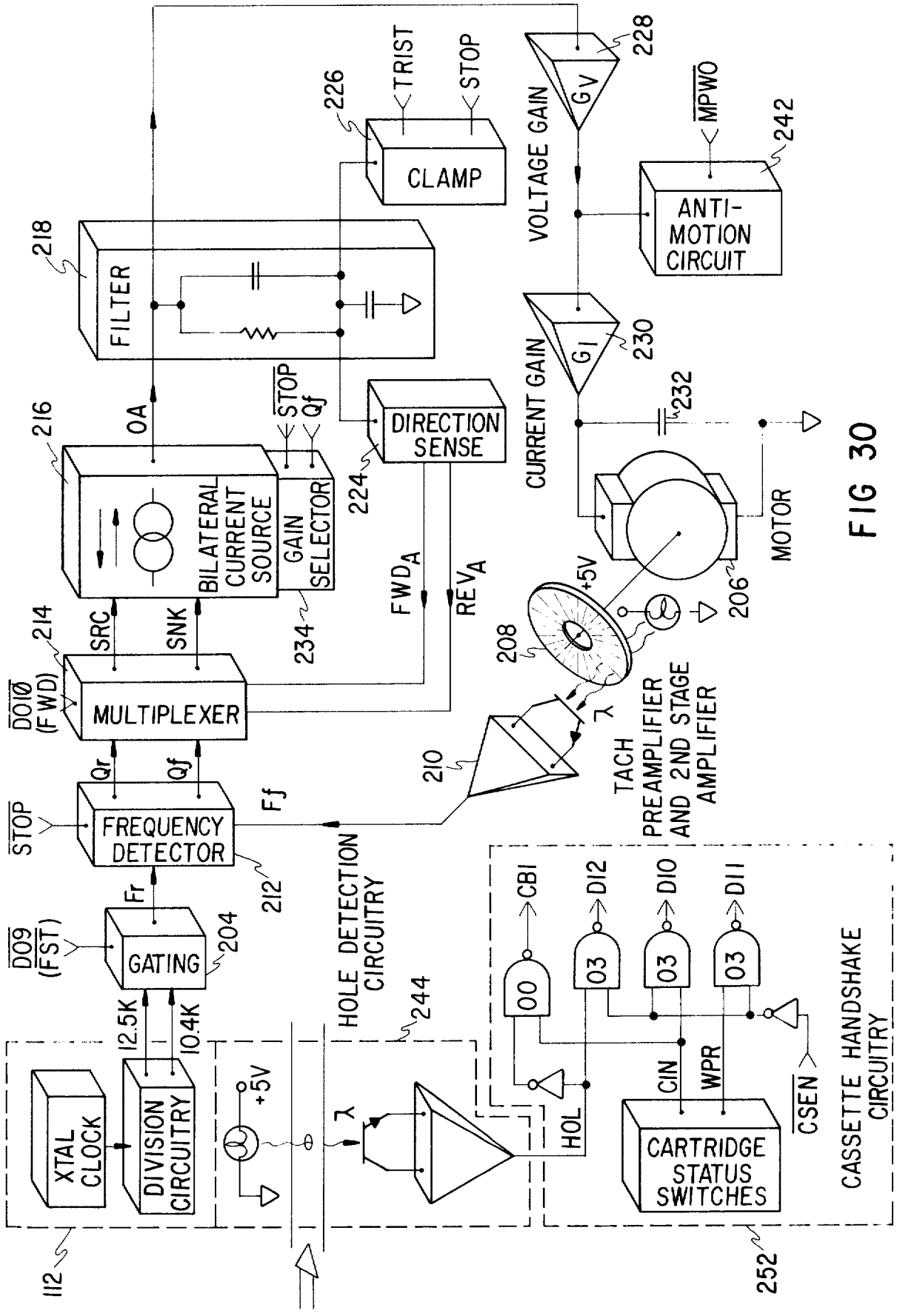

XTAL CLOCK
DIVISION CIRCUITRY
12.5K
10.4K
+5V
112
GATING
204
DO9 (FST)
Fr
FREQUENCY DETECTOR
212
STOP
Qr
Qf
Ff
MULTIPLEXER
214
DO10 (FWD)
SRC
SNK
BILATERAL CURRENT SOURCE
GAIN SELECTOR
216
234
0A
FILTER
218
STOP
Qf
DIRECTION SENSE
224
FWDA
REVA
CLAMP
226
TRIST
STOP
VOLTAGE GAIN
GV
228
CURRENT GAIN
GI
230
232
ANTI-MOTION CIRCUIT
242
MPWO
MOTOR
206
+5V
208
TACH PREAMPLIFIER AND 2ND STAGE AMPLIFIER
210
HOLE DETECTION CIRCUITRY
244
CBI
DI2
DIO
DI1
00
03
03
03
HOL
CIN
WPR
CSEN
CARTRIDGE STATUS SWITCHES
CASSETTE HANDSHAKE CIRCUITRY
252

FIG 30

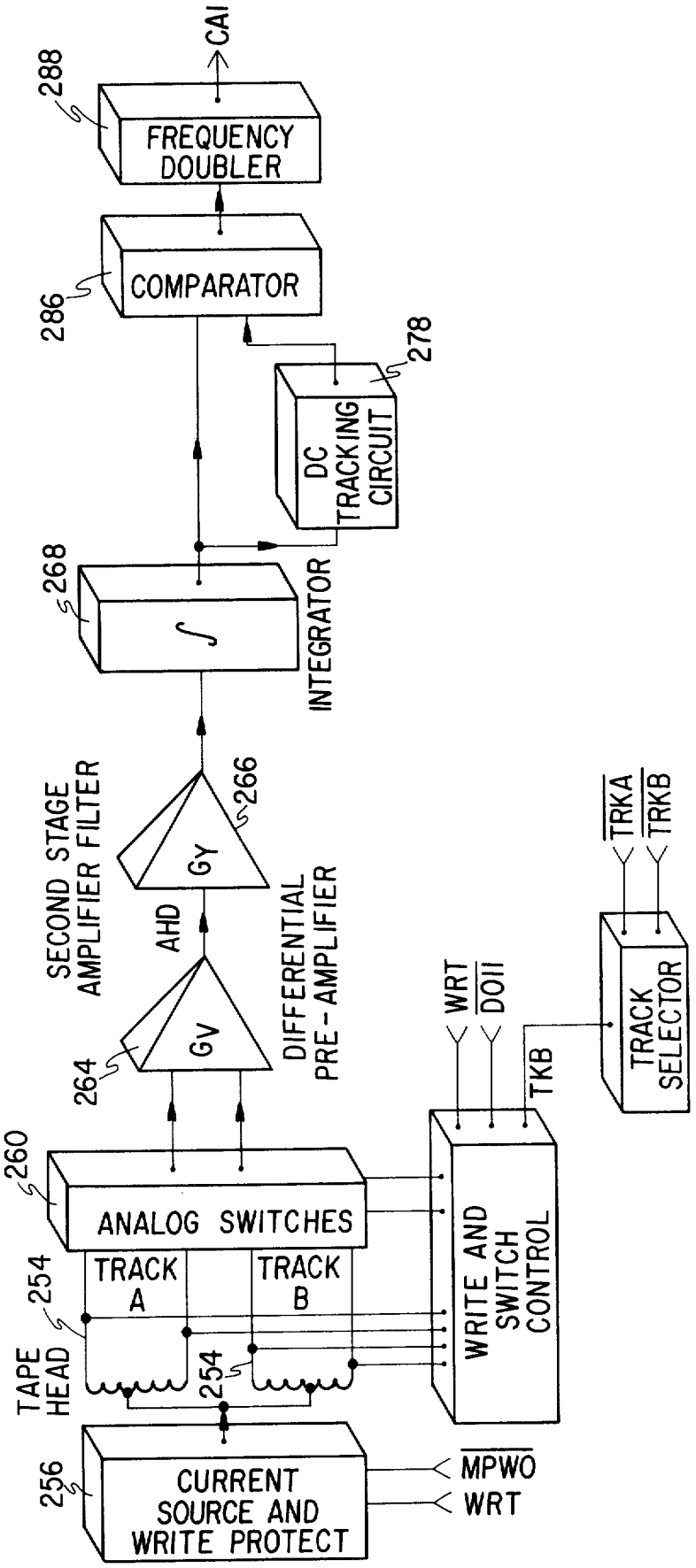

CAI
288
FREQUENCY DOUBLER
286
COMPARATOR
278
DC TRACKING CIRCUIT
268
∫
INTEGRATOR
SECOND STAGE AMPLIFIER FILTER
266
GY
AHD
GV
264
DIFFERENTIAL PRE-AMPLIFIER
WRT
DOII
TKB
TRKA
TRKB
TRACK SELECTOR
260
ANALOG SWITCHES
TRACK A
TRACK B
254
254
TAPE HEAD
WRITE AND SWITCH CONTROL
256
CURRENT SOURCE AND WRITE PROTECT
MPWO
WRT

FIG 31

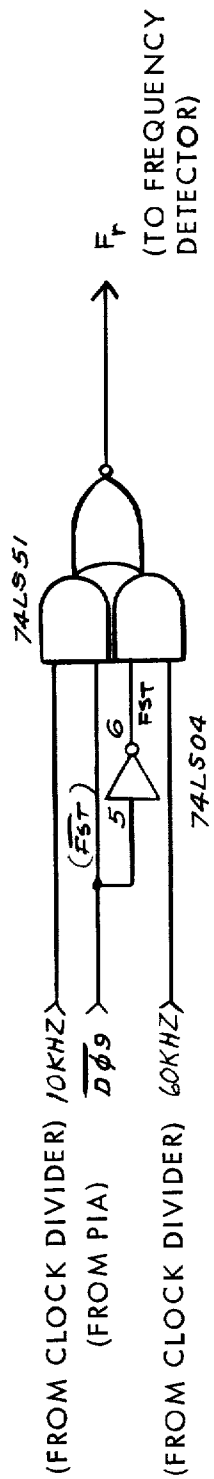

F_r
(TO FREQUENCY DETECTOR)
74LS51
6
FST
5
($\overline{FST}$)
74LS04
(FROM CLOCK DIVIDER) 10KHZ
(FROM PIA) $\overline{D\phi 9}$
(FROM CLOCK DIVIDER) 60KHZ

FIG. 32

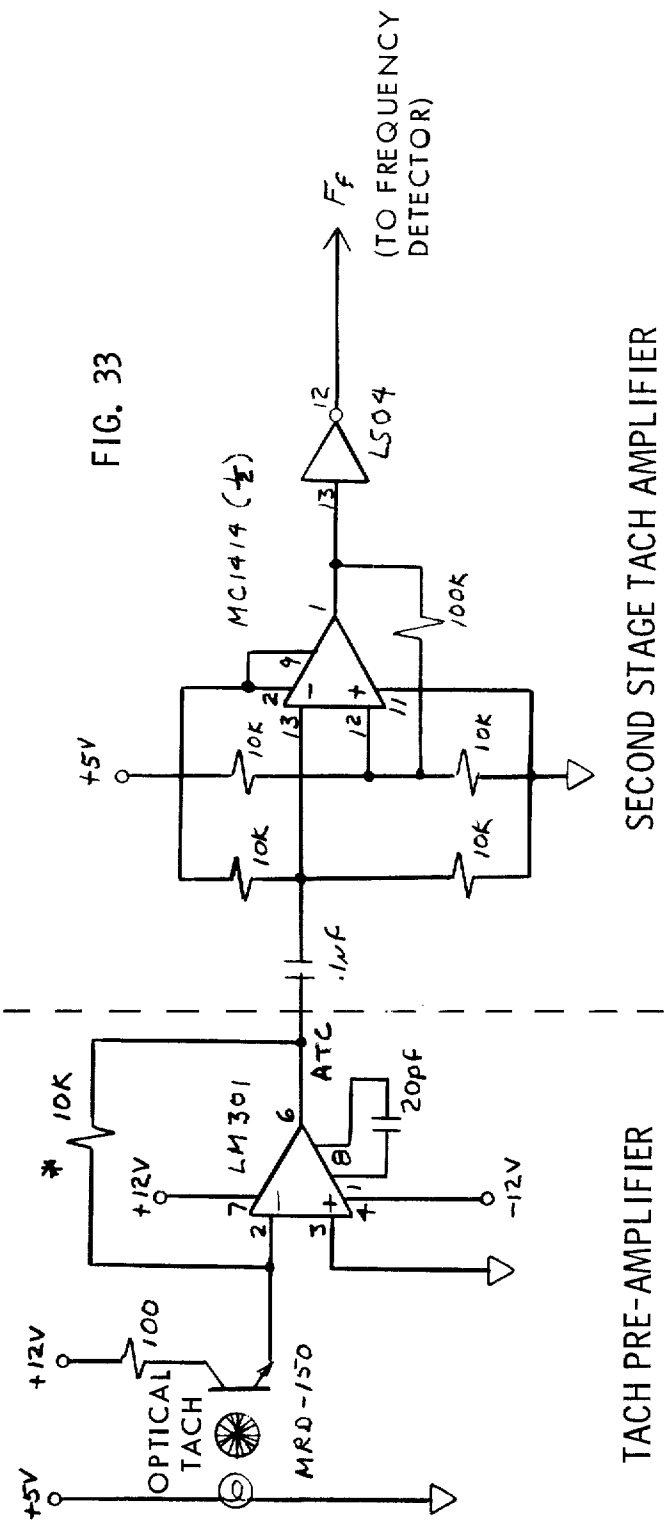


FIG. 33
TACH PRE-AMPLIFIER
SECOND STAGE TACH AMPLIFIER
OPTICAL TACH
MRD-150
100
+12V
+5V
LM301
10K
20pf
ATC
.1μf
MC1414 (4)
100K
10K
LS04
(TO FREQUENCY DETECTOR)
-12V

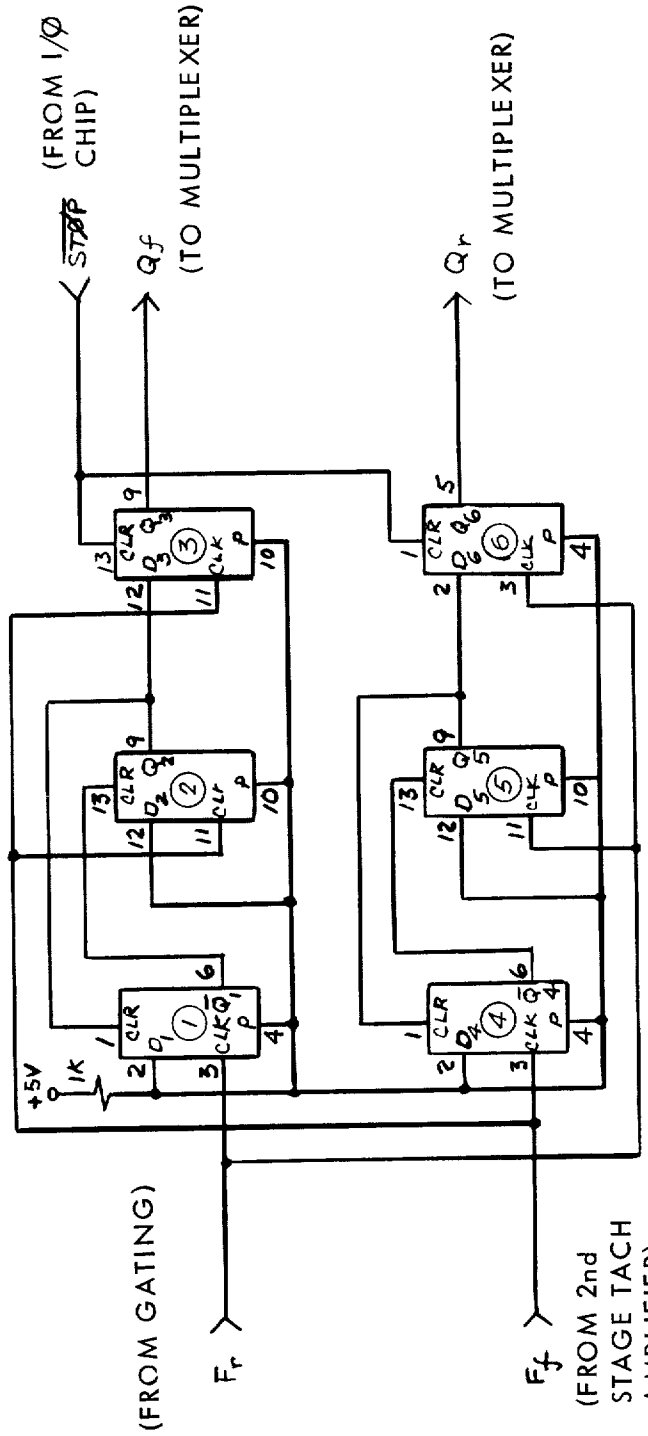

(FROM I/Ø CHIP)
STOP
Qf
(TO MULTIPLEXER)
Qr
(TO MULTIPLEXER)
+5V
1K
(FROM GATING)
Fr
Ff
(FROM 2nd STAGE TACH AMPLIFIER)

FIG. 34

7 LS7

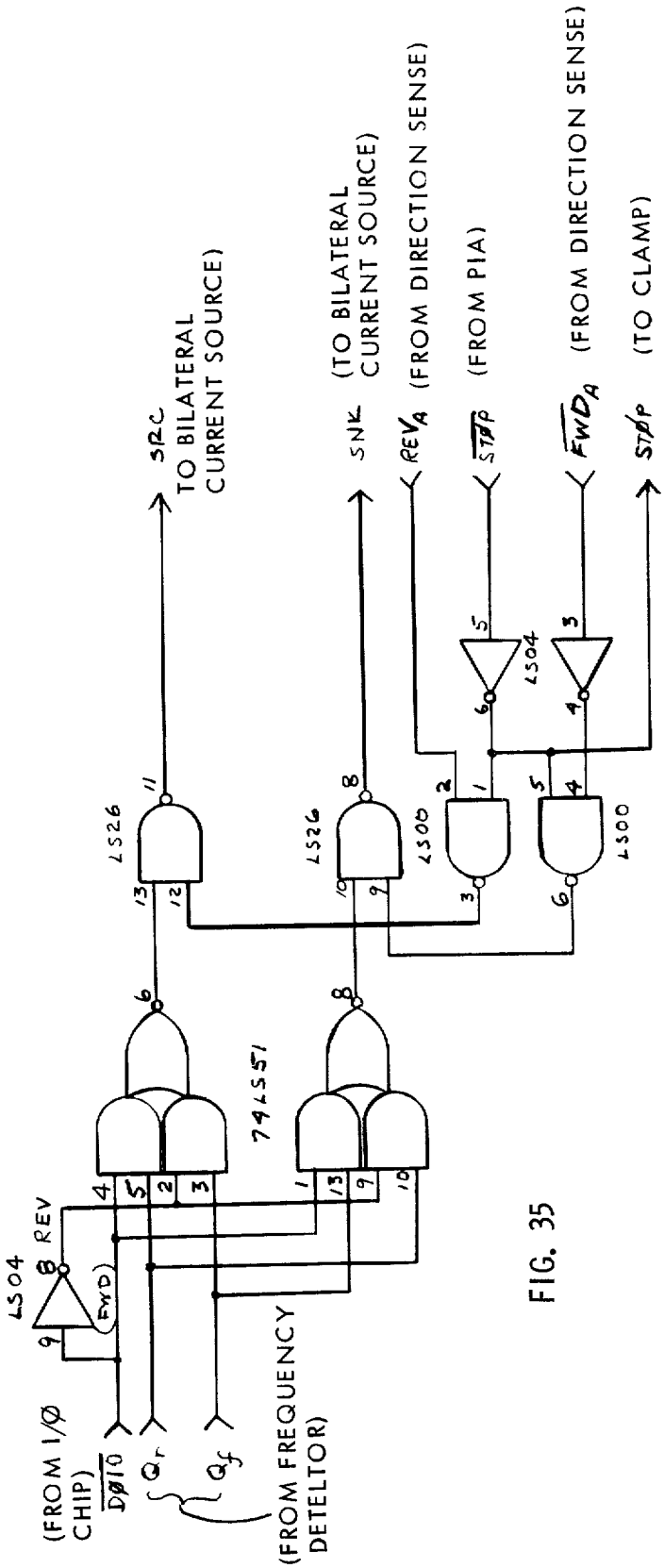

SRC
TO BILATERAL
CURRENT SOURCE)
SNK
(TO BILATERAL
CURRENT SOURCE)
REV_A
(FROM DIRECTION SENSE)
STOP
(FROM PIA)
FWD_A
(FROM DIRECTION SENSE)
STOP
(TO CLAMP)
LS26
LS26
LS00
LS00
LS04
LS04
74LS51
8 REV
FWD)
(FROM I/O
CHIP)
D010
Q_r
Q_f
(FROM FREQUENCY
DETELTOR)

FIG. 35

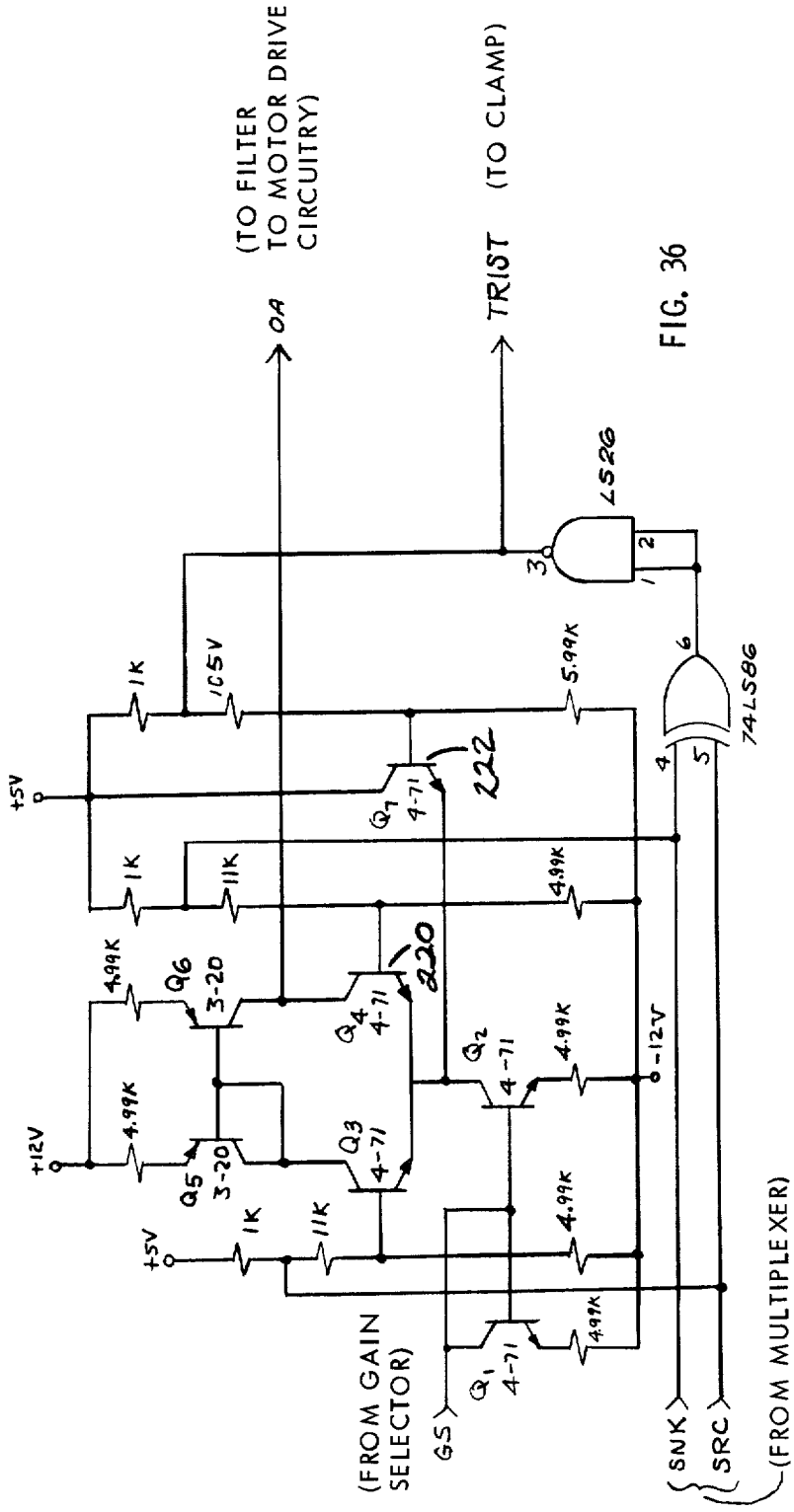

(TO FILTER TO MOTOR DRIVE CIRCUITRY)
OA
TRIST
(TO CLAMP)
LS26
74LS86
+5V
+12V
-12V
1K
105V
5.99K
222
11K
4.99K
Q6
3-20
Q4
4-71
220
Q2
Q5
Q3
Q1
Q7
(FROM GAIN SELECTOR)
GS
SNK
SRC
(FROM MULTIPLEXER)

FIG. 36

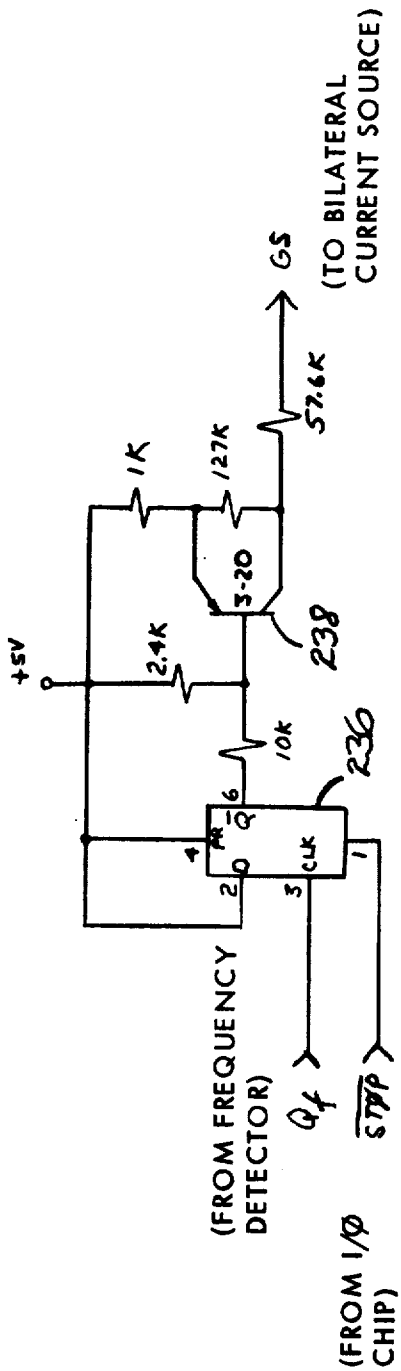

GS
(TO BILATERAL CURRENT SOURCE)
57.6K
127K
1K
3-20
238
+5V
2.4K
10K
236
4
PR
Q
6
D
2
CLK
3
1
5574
(FROM FREQUENCY DETECTOR)
Qf
STOP
(FROM I/O CHIP)

FIG. 37

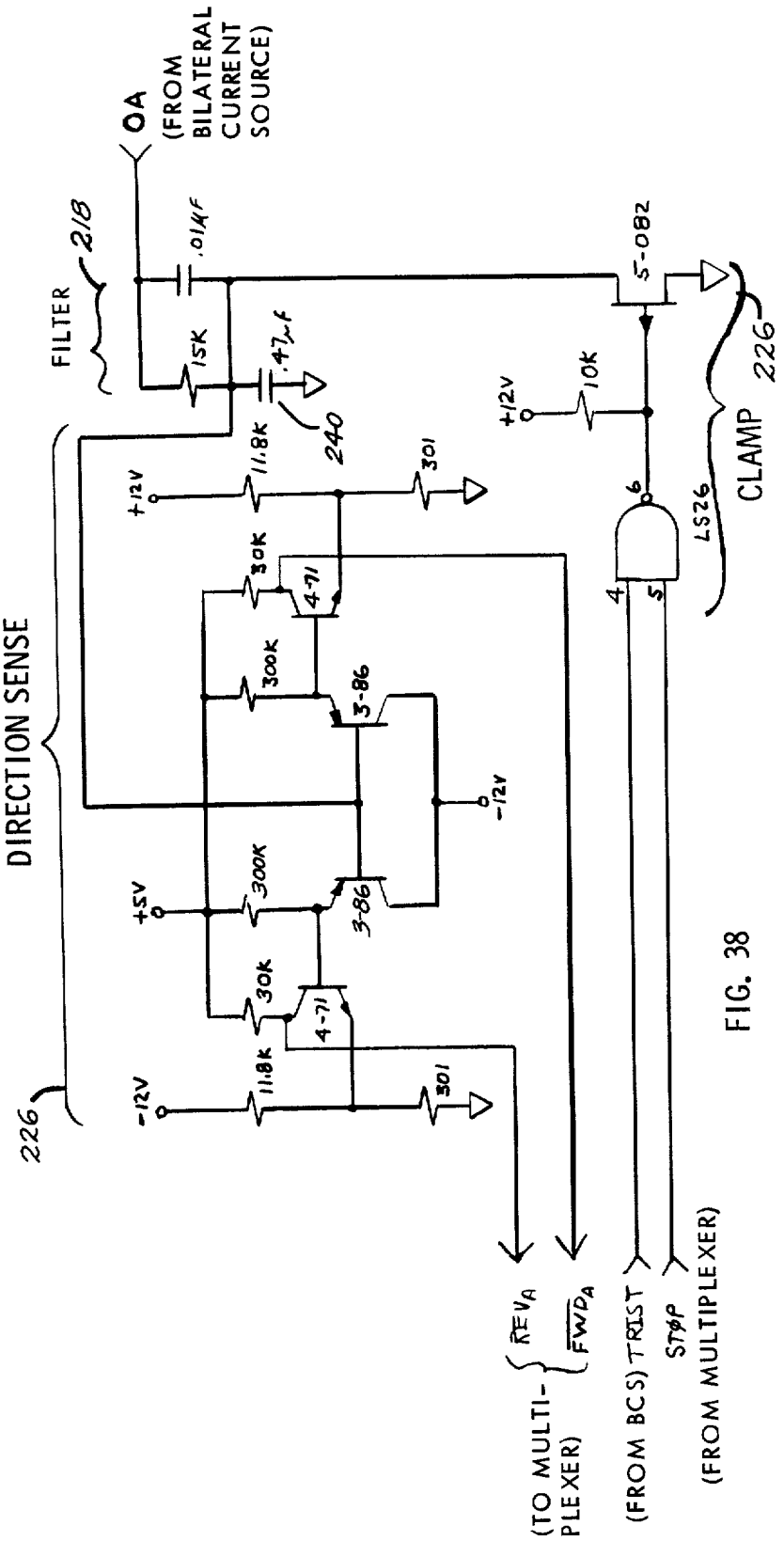

OA
(FROM BILATERAL CURRENT SOURCE)
FILTER
218
.01μF
15K
.47μF
240
+12V
11.8K
301
30K
4-71
300K
3-86
-12V
+5V
300K
3-86
30K
4-71
11.8K
-12V
301
226
DIRECTION SENSE
+12V
10K
5-082
CLAMP
226
LS26
6
4
5
(TO MULTI-PLEXER)
REV_A
FWD_A
(FROM BCS) TRIST
STOP
(FROM MULTIPLEXER)

FIG. 38

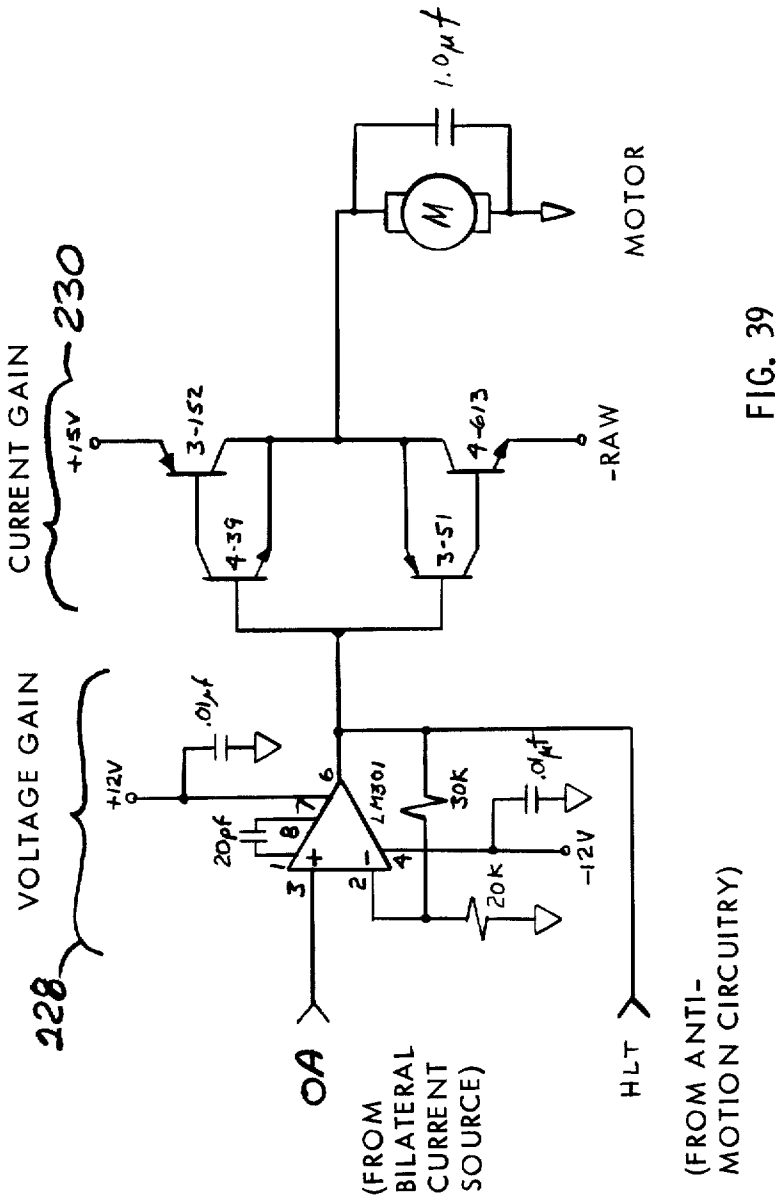

1.0μf
M
MOTOR
CURRENT GAIN
230
+15V
3-152
4-613
-RAW
4-39
3-51
VOLTAGE GAIN
228
+12V
.01μf
20pf
LM301
30K
.01μf
-12V
20K
OA
(FROM BILATERAL CURRENT SOURCE)
HLT
(FROM ANTI-MOTION CIRCUITRY)

FIG. 39

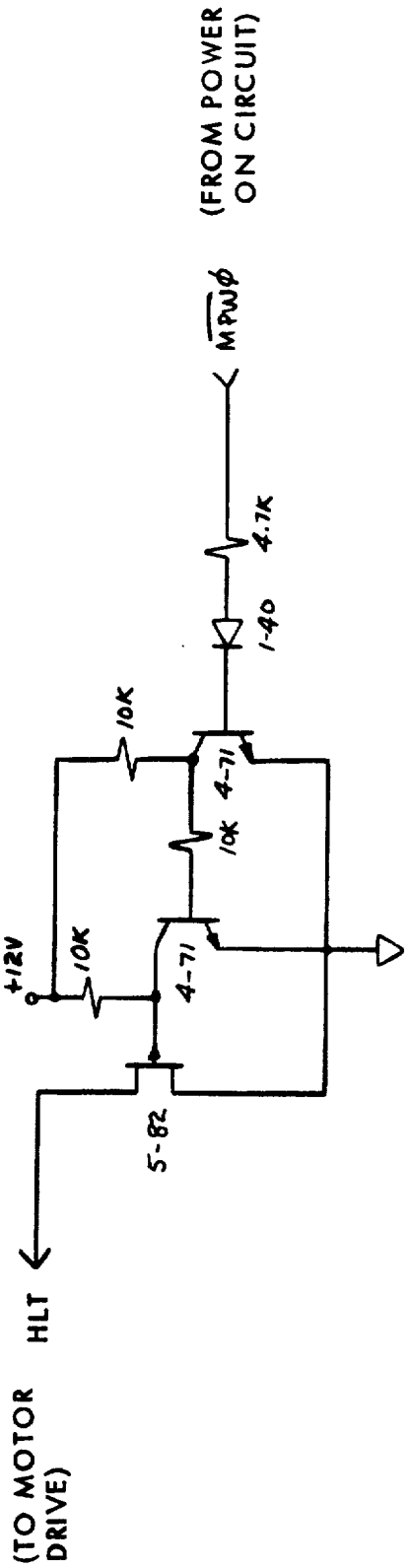

(FROM POWER ON CIRCUIT)
MPWϕ
4.7K
1-40
10K
4-71
10K
+12V
10K
4-71
5-82
HLT
(TO MOTOR DRIVE)

FIG. 40

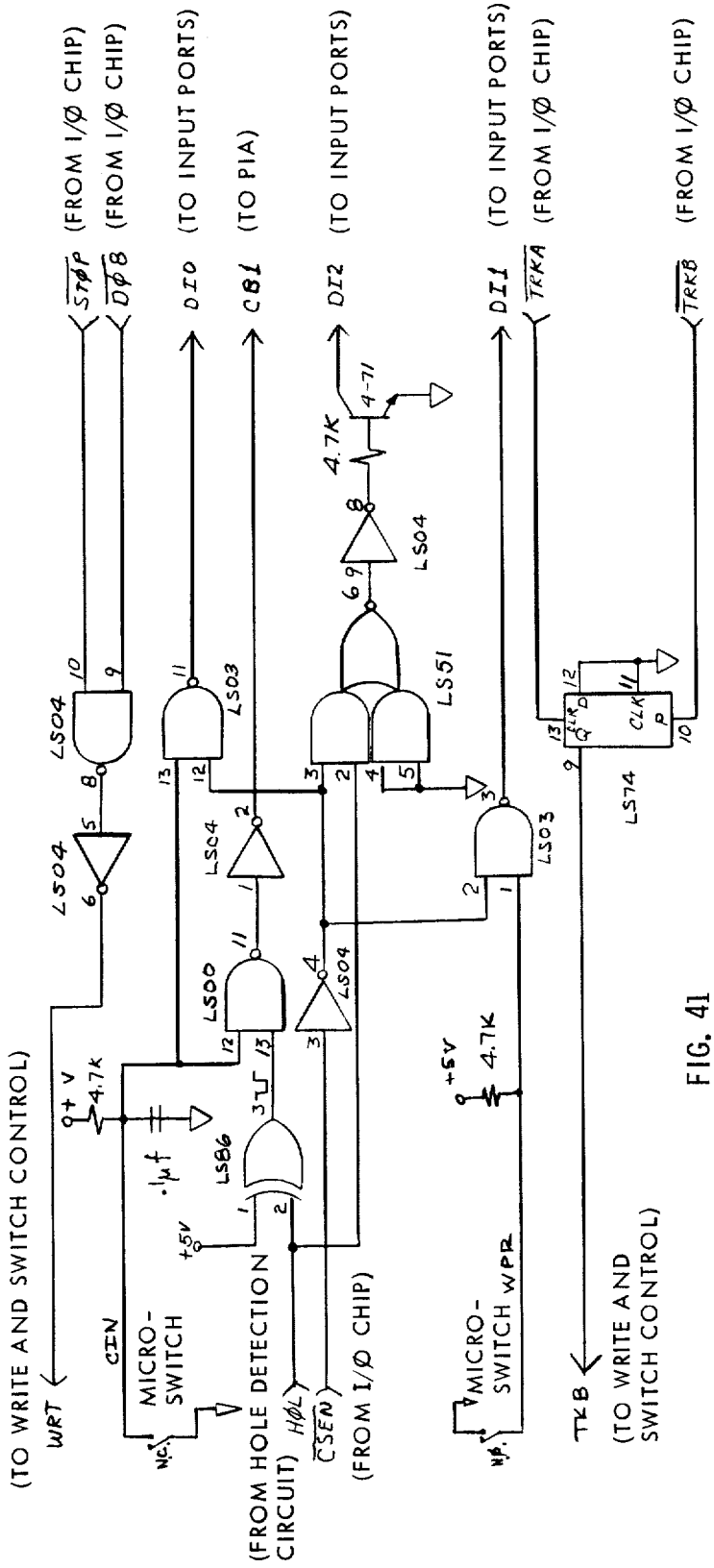

(TO WRITE AND SWITCH CONTROL)
WRT
CEN
MICRO-SWITCH
N.C.
+5V
4.7K
.1μf
LS86
(FROM HOLE DETECTION CIRCUIT)
HØL
CSEN
(FROM I/Ø CHIP)
LS00
LS04
LS04
LS04
LS04
LS03
LS04
LS51
LS03
LS74
4.7K
4.7K
+5V
MICRO-SWITCH WPR
WP.
STØP
DØ8
(FROM I/Ø CHIP)
(FROM I/Ø CHIP)
DI0
(TO INPUT PORTS)
CB1
(TO PIA)
DI2
(TO INPUT PORTS)
DI1
(TO INPUT PORTS)
TRKA
(FROM I/Ø CHIP)
TRKB
(FROM I/Ø CHIP)
TKB
(TO WRITE AND SWITCH CONTROL)

FIG. 41

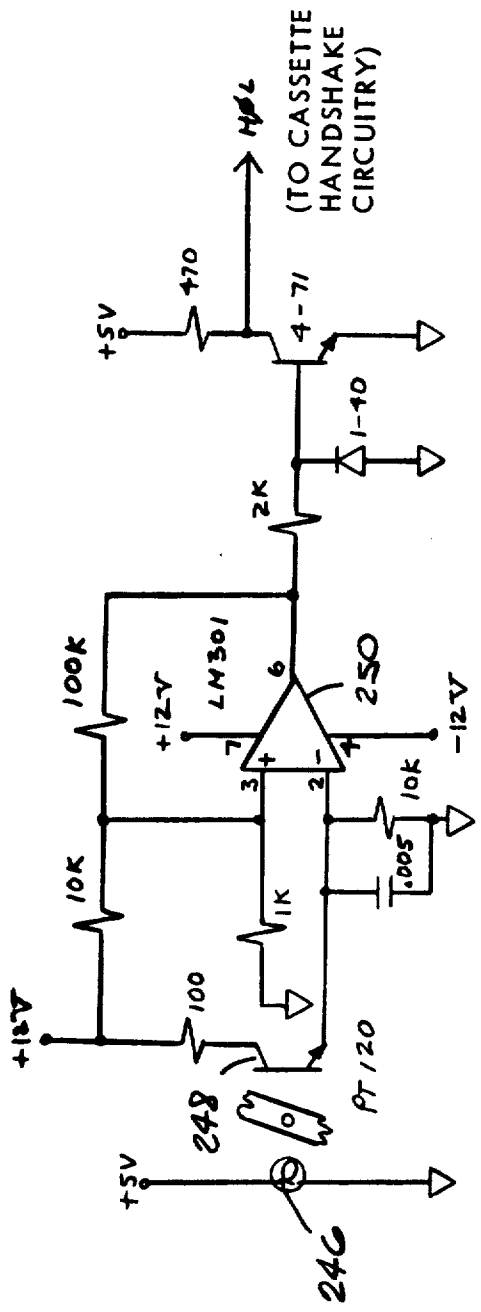

(TO CASSETTE HANDSHAKE CIRCUITRY)
+5V
470
4-71
1-10
2K
100K
LM301
+12V
-12V
250
10K
10K
1K
.005
100
+12V
248
PT 120
+5V
246

FIG. 42

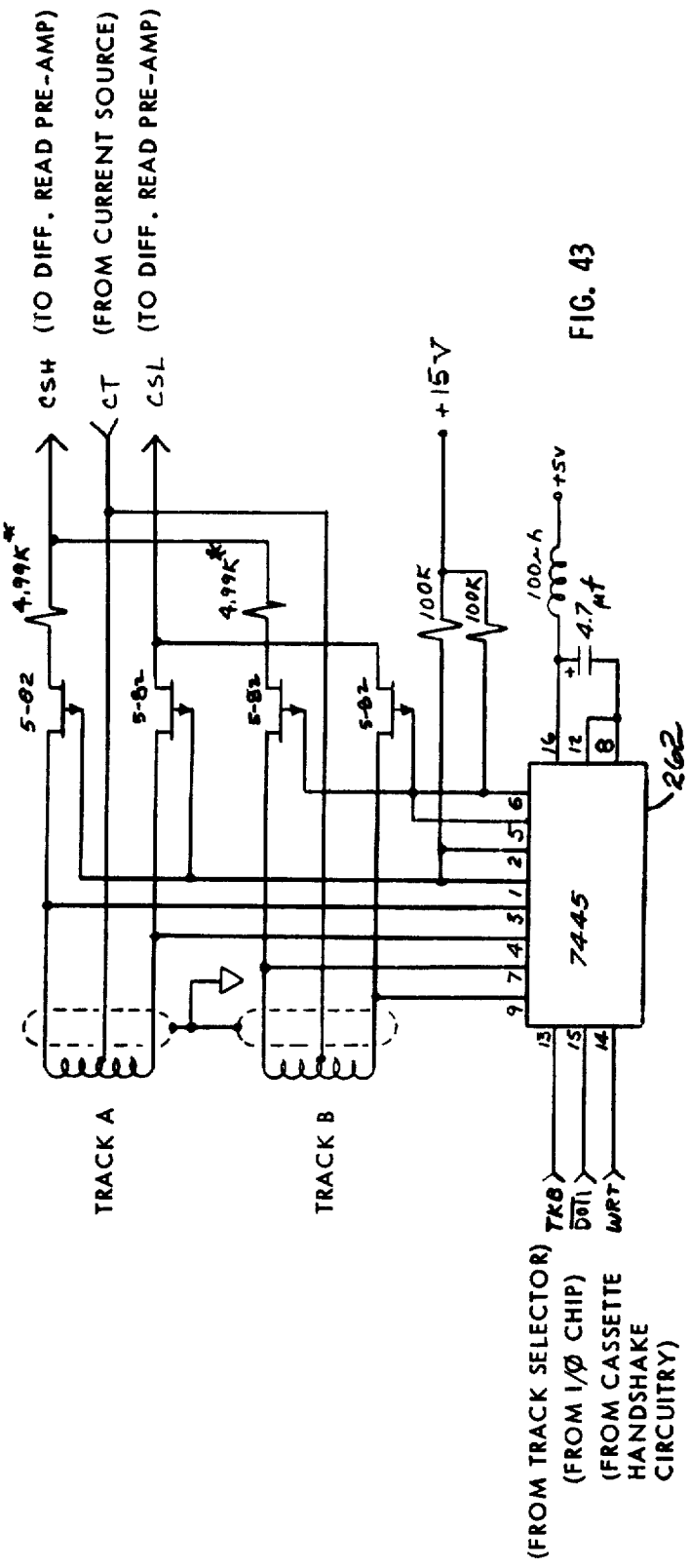

CSH (TO DIFF. READ PRE-AMP)
CT (FROM CURRENT SOURCE)
CSL (TO DIFF. READ PRE-AMP)
+15V
+5V
4.99K*
4.99K*
100K
100K
100Ω
4.7μf
5-82
5-82
5-82
5-82
16
12
8
6
5
2
1
3
4
7
9
13
15
14
7445
262
TRACK A
TRACK B
(FROM TRACK SELECTOR)
(FROM I/Ø CHIP)
(FROM CASSETTE HANDSHAKE CIRCUITRY)
TKB
DOTI
WRT

FIG. 43

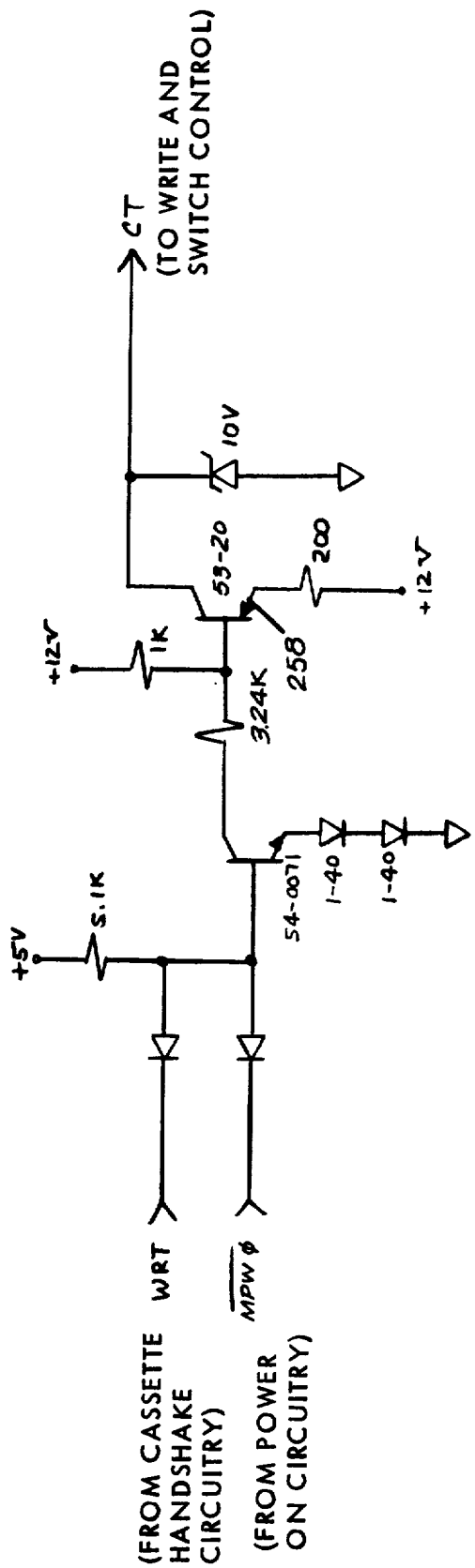

CT
(TO WRITE AND SWITCH CONTROL)
10V
53-20
200
+12V
+12V
1K
324K
258
54-0071
1-40
1-40
+5V
5.1K
WRT
MPW ϕ
(FROM CASSETTE HANDSHAKE CIRCUITRY)
(FROM POWER ON CIRCUITRY)

FIG. 44

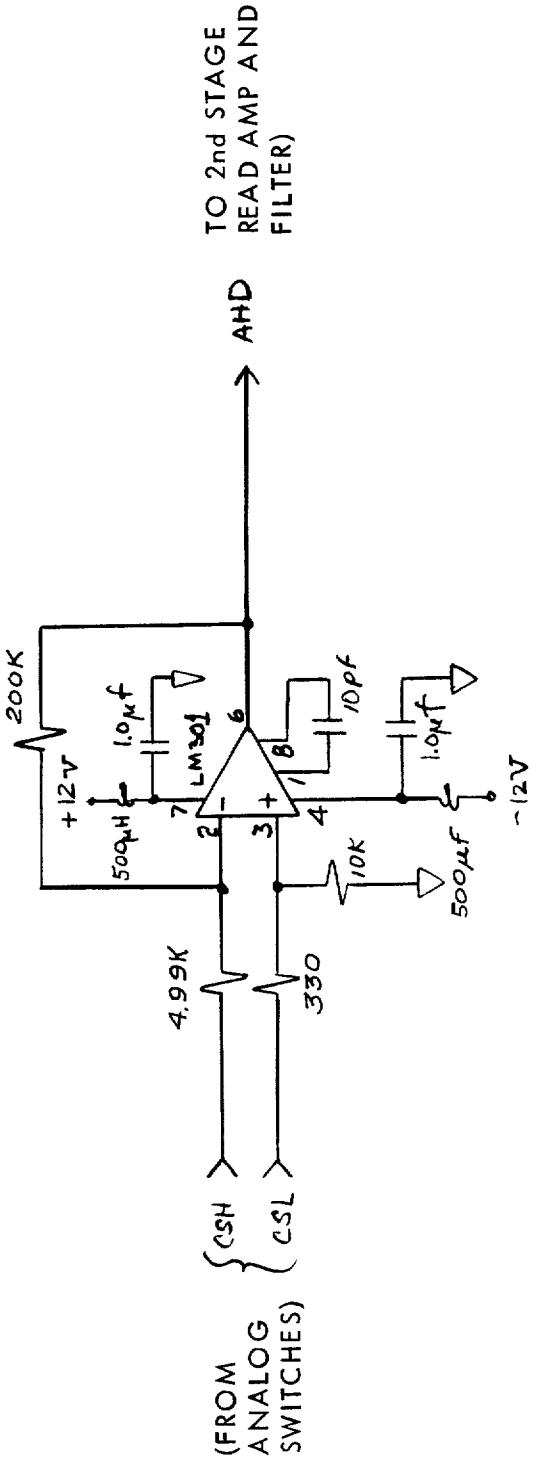

TO 2nd STAGE READ AMP AND FILTER)
AHD
200K
+12V
500μH
1.0μf
LM301
10pf
1.0μf
-12V
500μf
10K
4.99K
330
(FROM ANALOG SWITCHES)
CSH
CSL

FIG. 45

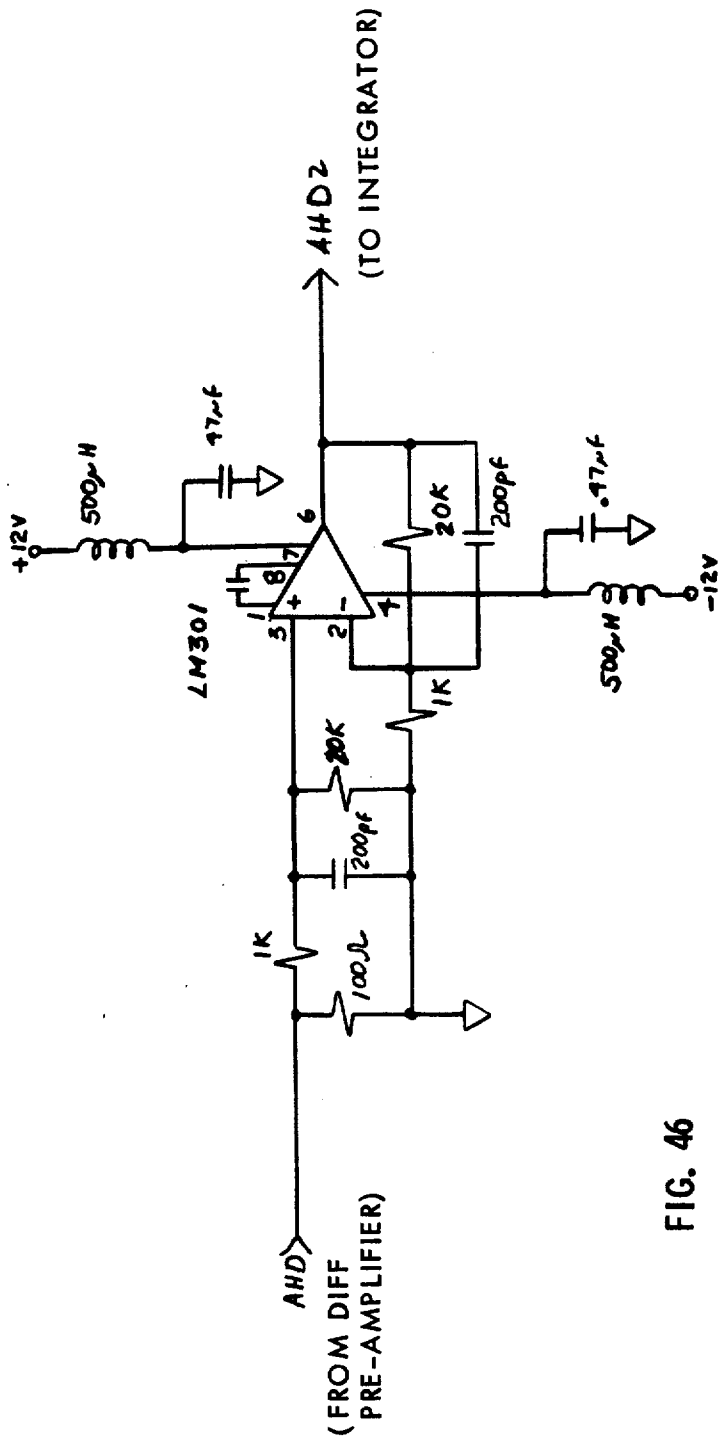

AHD2
(TO INTEGRATOR)
+12V
500μH
.47μF
6
8
7
20K
200pf
.47μF
-12V
500μH
LM301
1
3
2
4
1K
20K
200pf
1K
100Ω
AHD
(FROM DIFF
PRE-AMPLIFIER)

FIG. 46

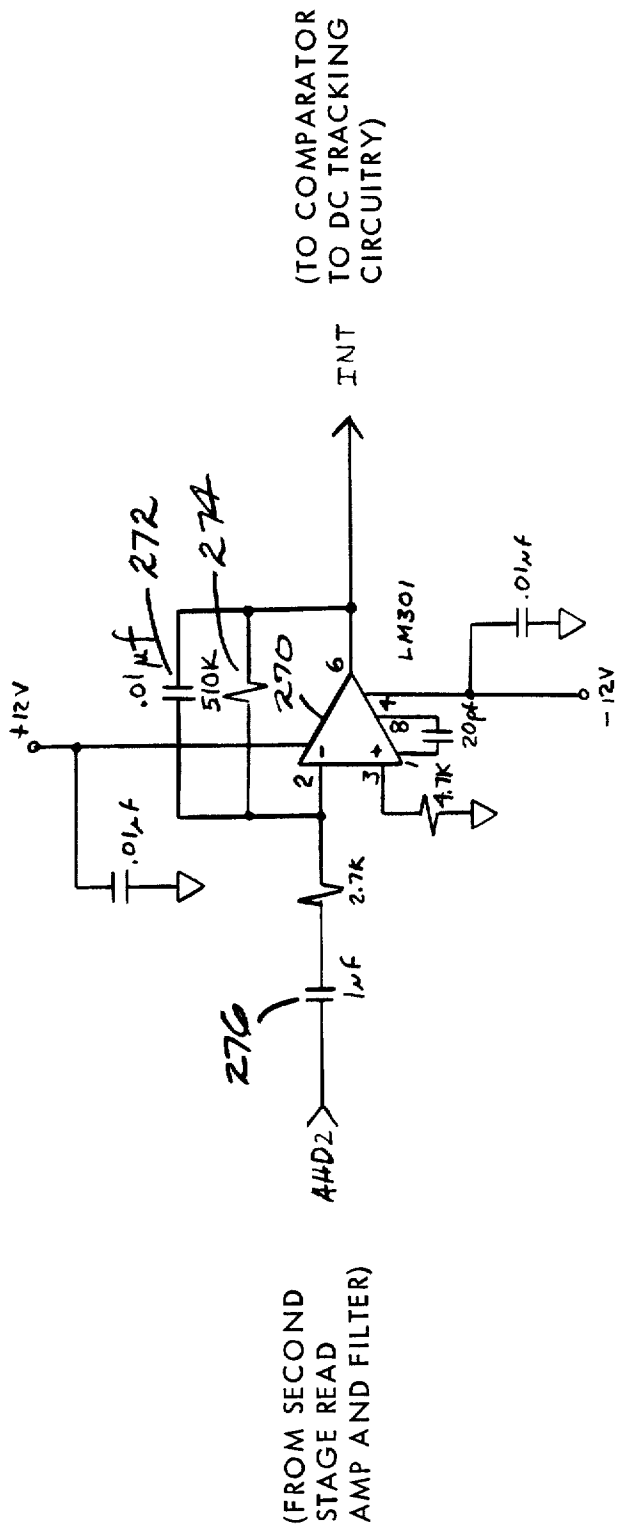

(TO COMPARATOR TO DC TRACKING CIRCUITRY)
INT
LM301
+12V
-12V
.01μf
510K
270
272
274
276
2.7K
1μf
4.7K
20pf
44D2
(FROM SECOND STAGE READ AMP AND FILTER)

FIG. 47

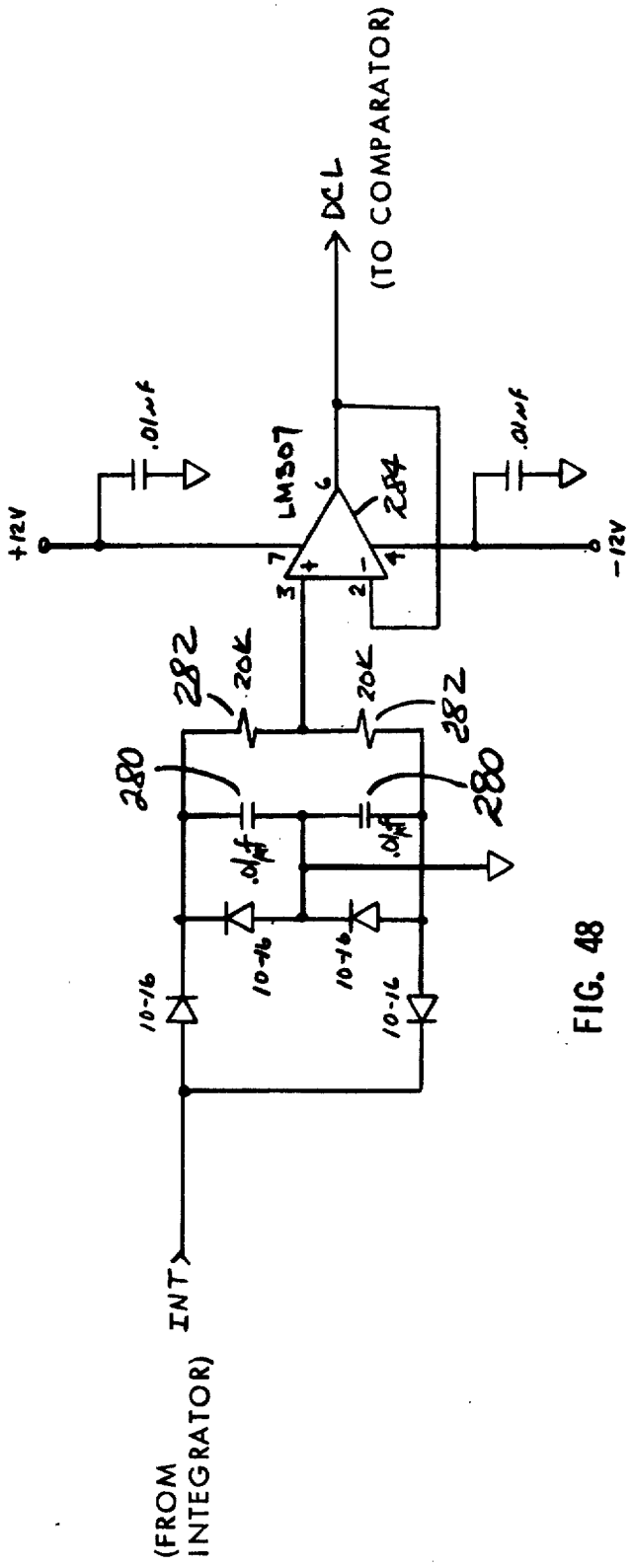

DCL
(TO COMPARATOR)
.01μf
LM307
284
.01μf
+12V
-12V
3
7
6
2
4
282
20K
20K
282
280
280
.01μf
.01μf
10-16
10-16
10-16
10-16
INT
(FROM INTEGRATOR)

FIG. 48

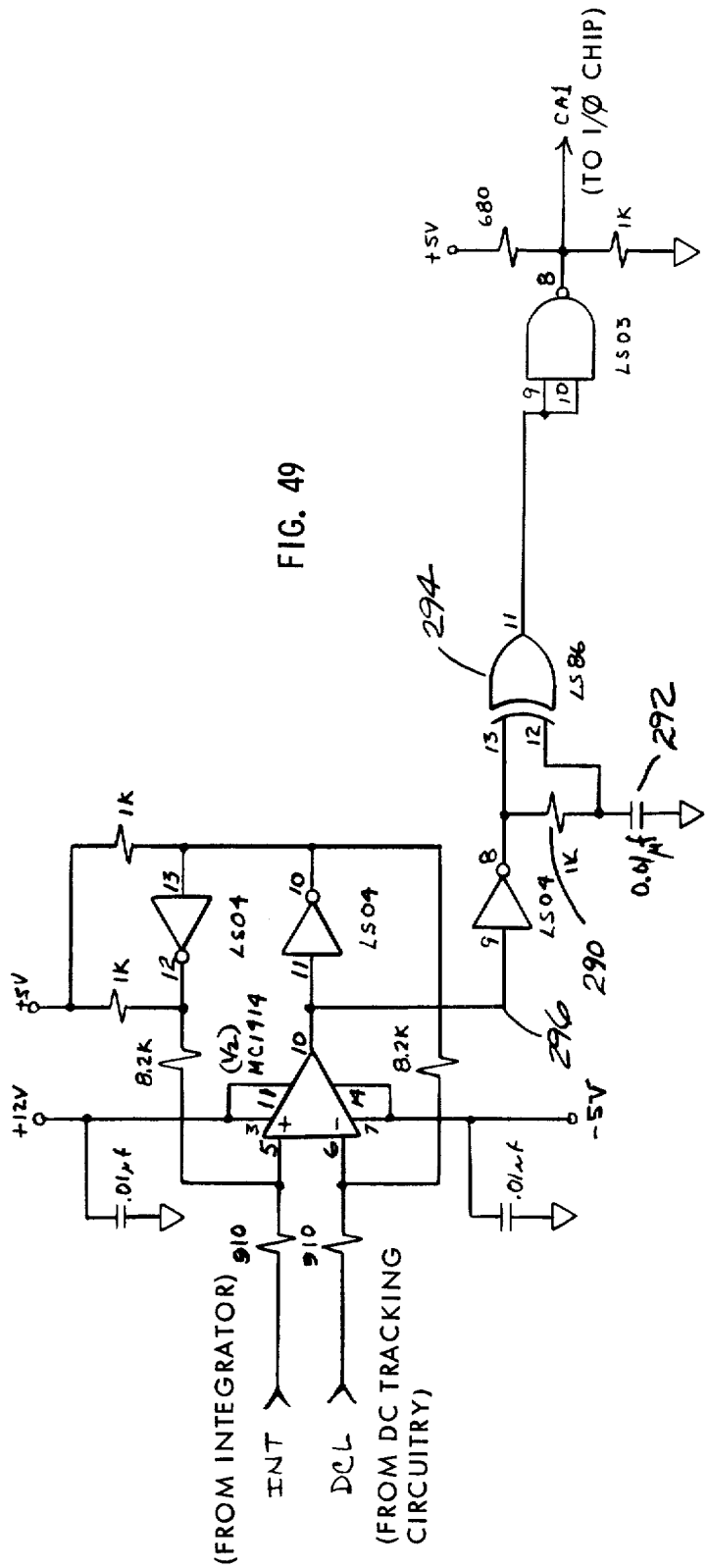

+5V
680
CAL
(TO I/Ø CHIP)
1K
LS03
294
LS86
292
0.01μf
LS04
290
296
+12V
-5V
8.2K
(V2) MC1414
.01μf
910
(FROM INTEGRATOR)
INT
DCL
(FROM DC TRACKING CIRCUITRY)

FIG. 49

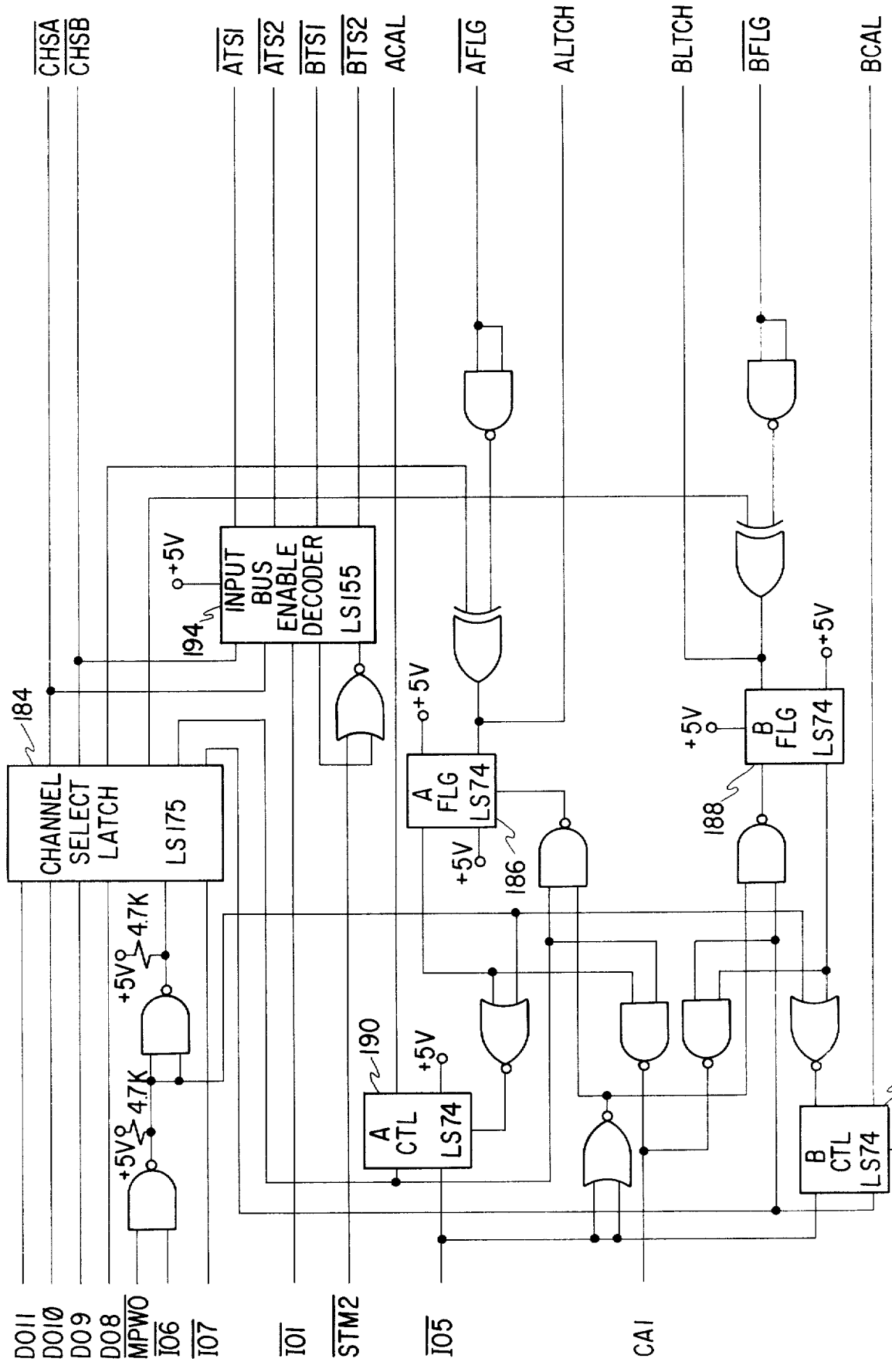

CHSA
CHSB
ATSI
ATS2
BTSI
BTS2
ACAL
AFLG
ALTCH
BLTCH
BFLG
BCAL
INPUT BUS ENABLE DECODER LS155
194
+5V
184
CHANNEL SELECT LATCH LS175
4.7K
A FLG LS74
186
B FLG LS74
188
A CTL LS74
190
B CTL LS74
192
DO11
DO10
DO9
DO8
MPWO
IO6
IO7
IO1
STM2
IO5
CA1

FIG 50

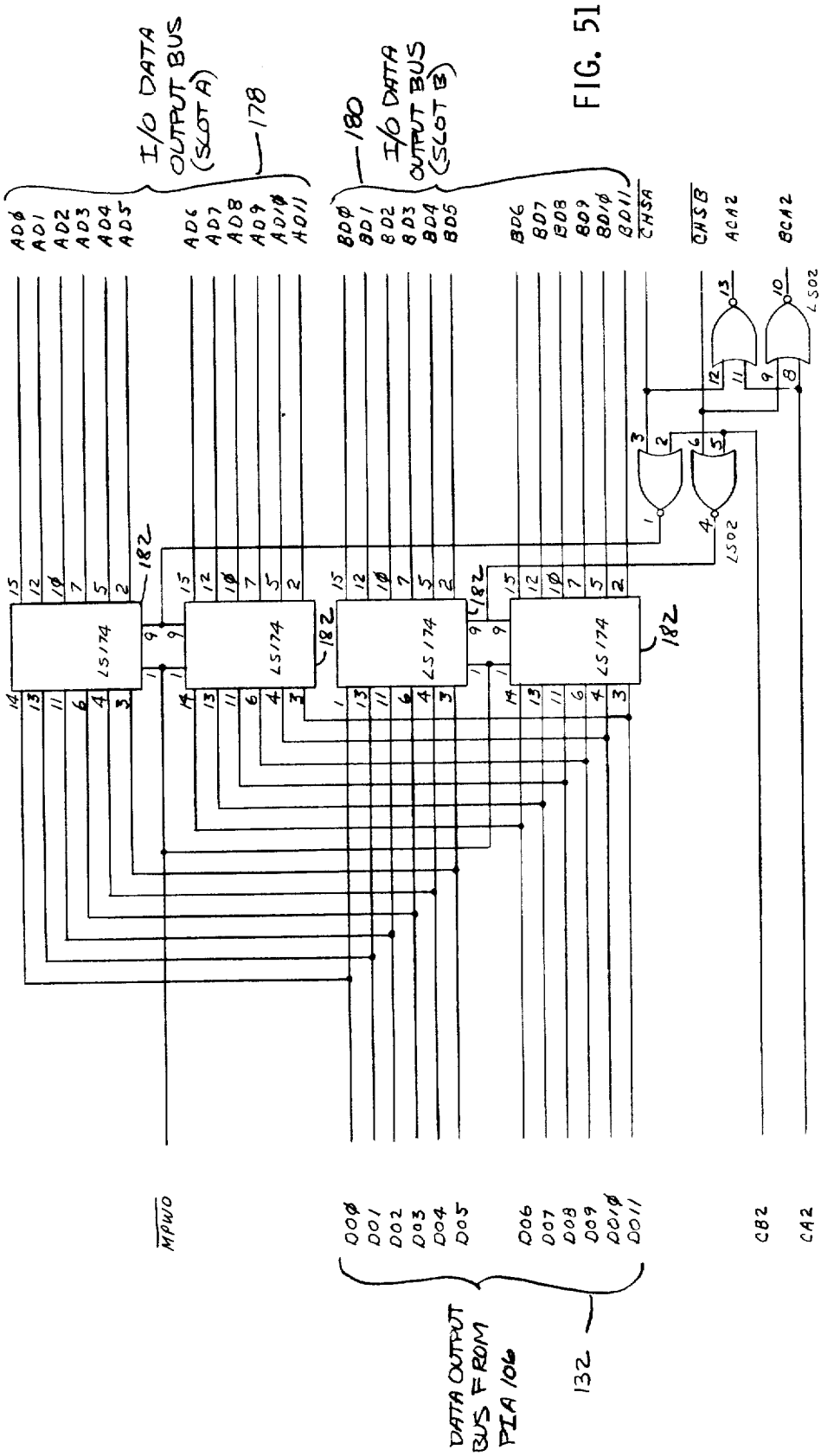

I/O DATA OUTPUT BUS (SLOT A)
178
I/O DATA OUTPUT BUS (SLOT B)
180
AD0 AD1 AD2 AD3 AD4 AD5
AD6 AD7 AD8 AD9 AD10 AD11
BD0 BD1 BD2 BD3 BD4 BD5
BD6 BD7 BD8 BD9 BD10 BD11 CHSA
CHSB CA2
BCA2
LS174
182
LS02
MPWIO
D00 D01 D02 D03 D04 D05
D06 D07 D08 D09 D010 D011
CB2 CA2
DATA OUTPUT BUS FROM PIA 106
132

FIG. 51

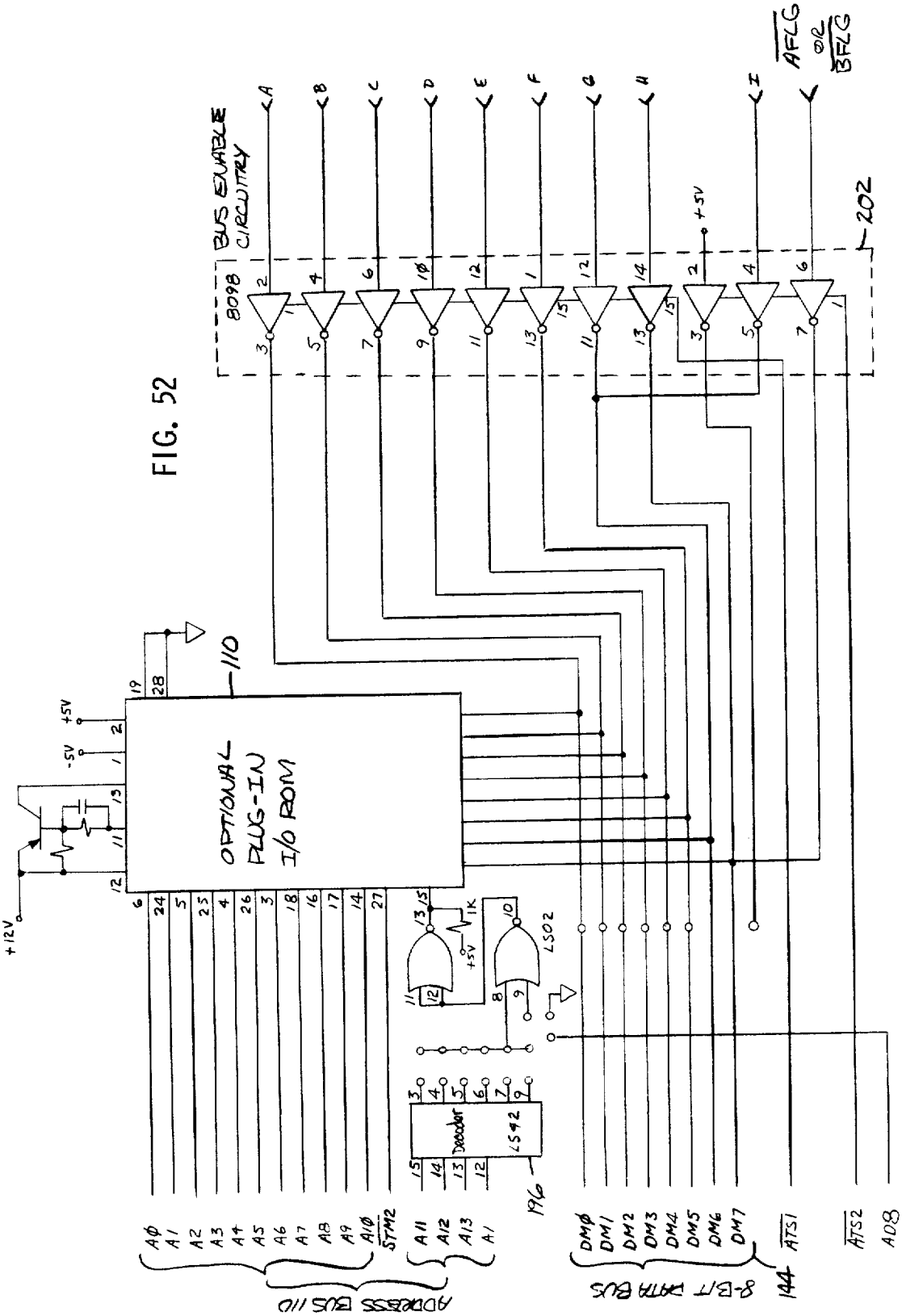

8098 BUS ENABLE CIRCUITRY
202
OPTIONAL PLUG-IN I/O ROM
110
+12V
+5V
-5V
1K
LS02
LS42
Decoder
196
ADDRESS BUS I/O
8-BIT DATA BUS
144
AFLG
BFLG
ATS1
ATS2
A08
STM2

FIG. 52

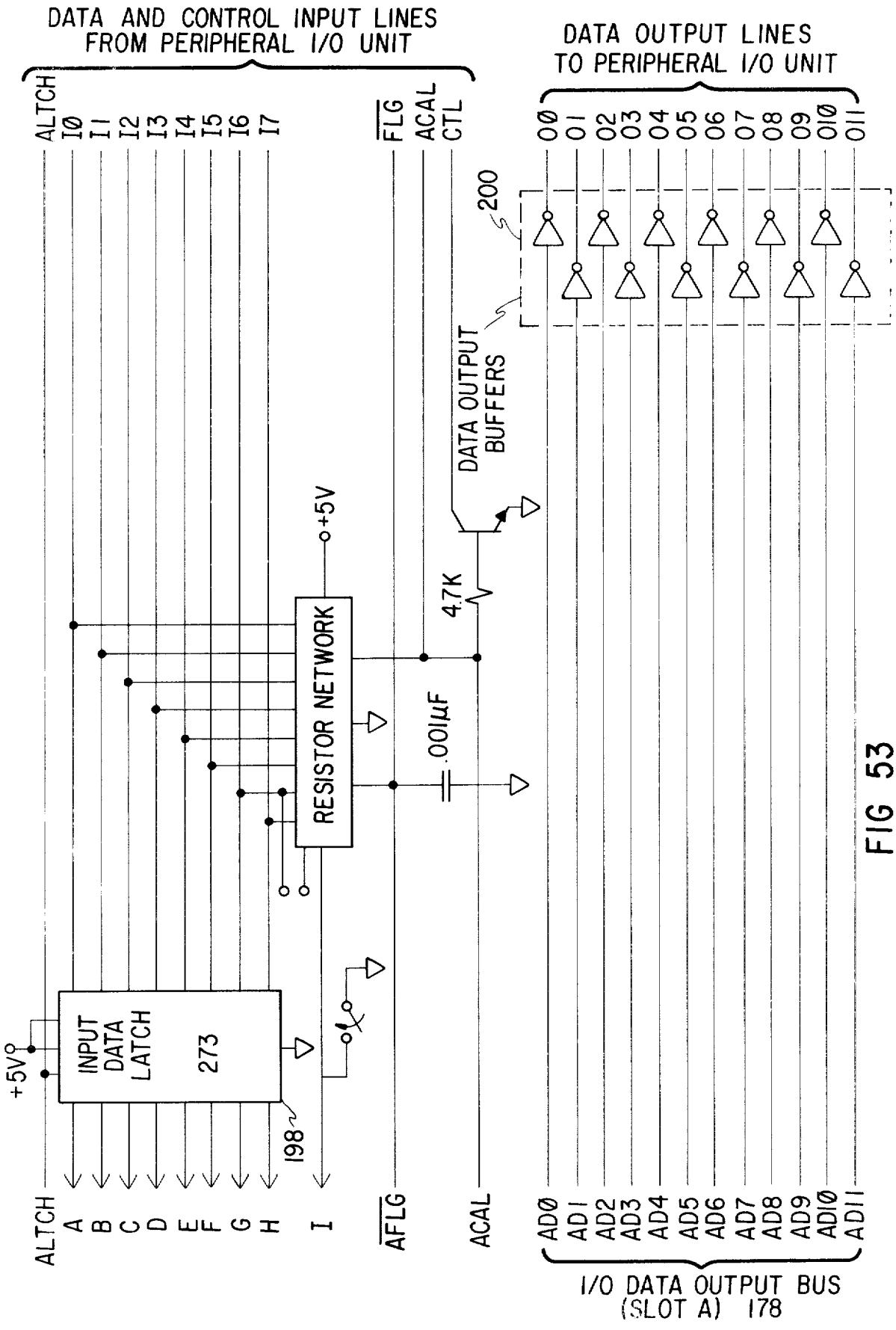

DATA AND CONTROL INPUT LINES FROM PERIPHERAL I/O UNIT
ALTCH
I0
I1
I2
I3
I4
I5
I6
I7
FLG
ACAL CTL
DATA OUTPUT LINES TO PERIPHERAL I/O UNIT
00
01
02
03
04
05
06
07
08
09
010
011
200
DATA OUTPUT BUFFERS
+5V
4.7K
RESISTOR NETWORK
.001μF
INPUT DATA LATCH
273
198
+5V
ALTCH
A
B
C
D
E
F
G
H
I
AFLG
ACAL
AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
AD8
AD9
AD10
AD11
I/O DATA OUTPUT BUS (SLOT A) 178

FIG 53

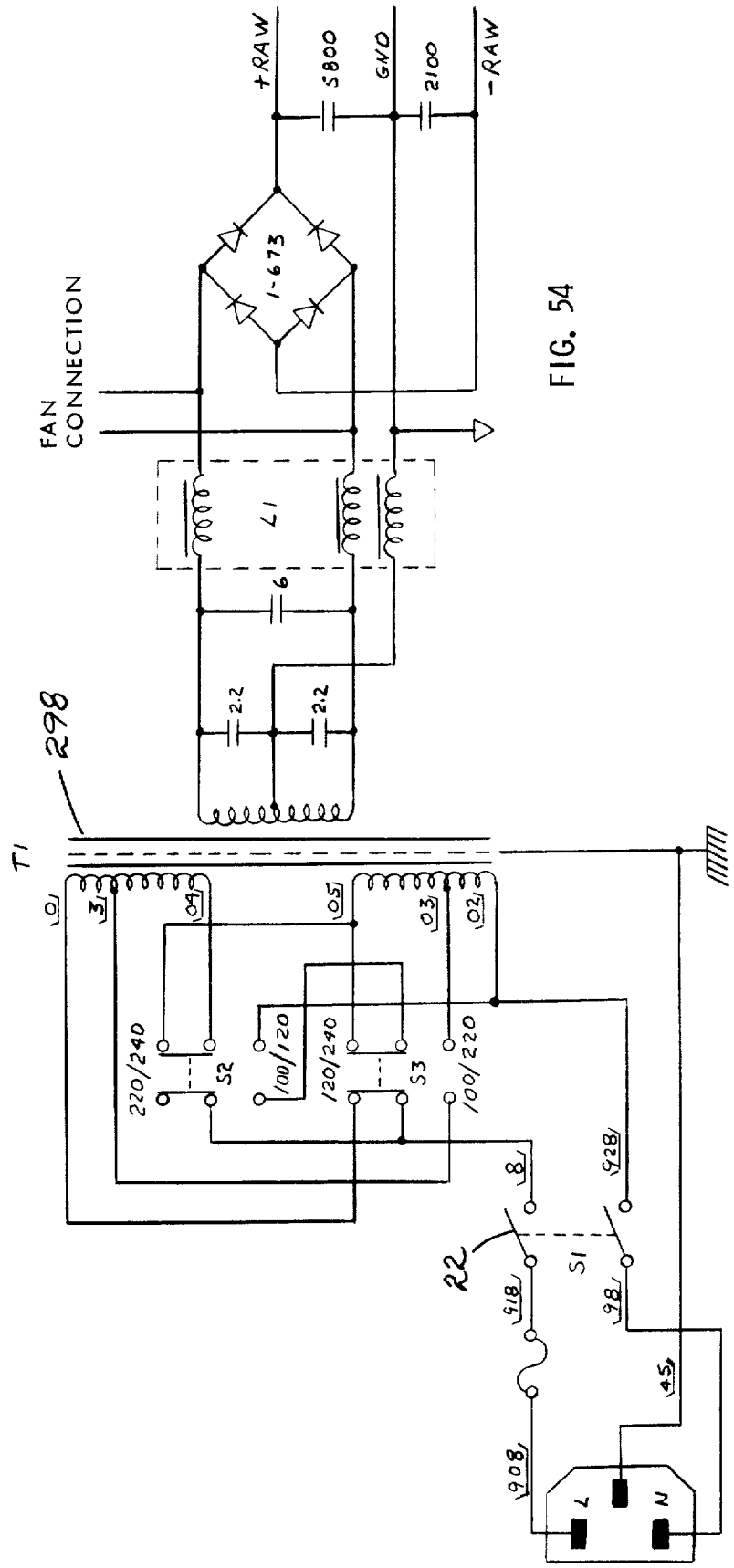

+RAW
5800
GND
2100
-RAW
1-673
FAN CONNECTION
L1
6
2.2
2.2
298
T1
0
31
04
05
03
02
220/240
100/120
120/240
100/220
S2
S3
8
928
22
S1
918
98
45
908
L
N

FIG. 54

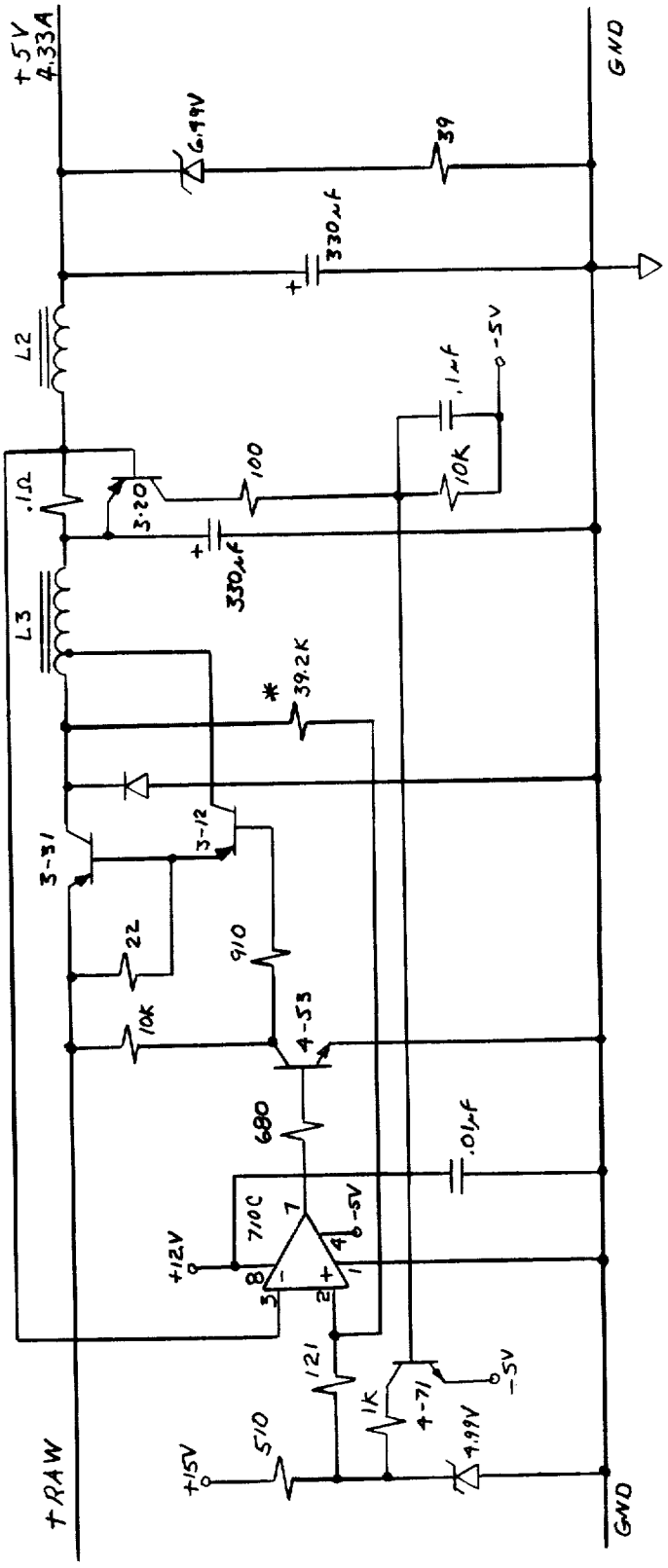

+5V 4.33A
GND
6.49V
39
330uf
L2
.1Ω
3-20
100
.1uf
-5V
10K
330uf
L3
*39.2K
3-31
3-12
22
910
10K
4-53
680
.01uf
710C
+12V
-5V
121
1K
4-71
4.99V
-5V
+15V
510
+RAW
GND

FIG. 55

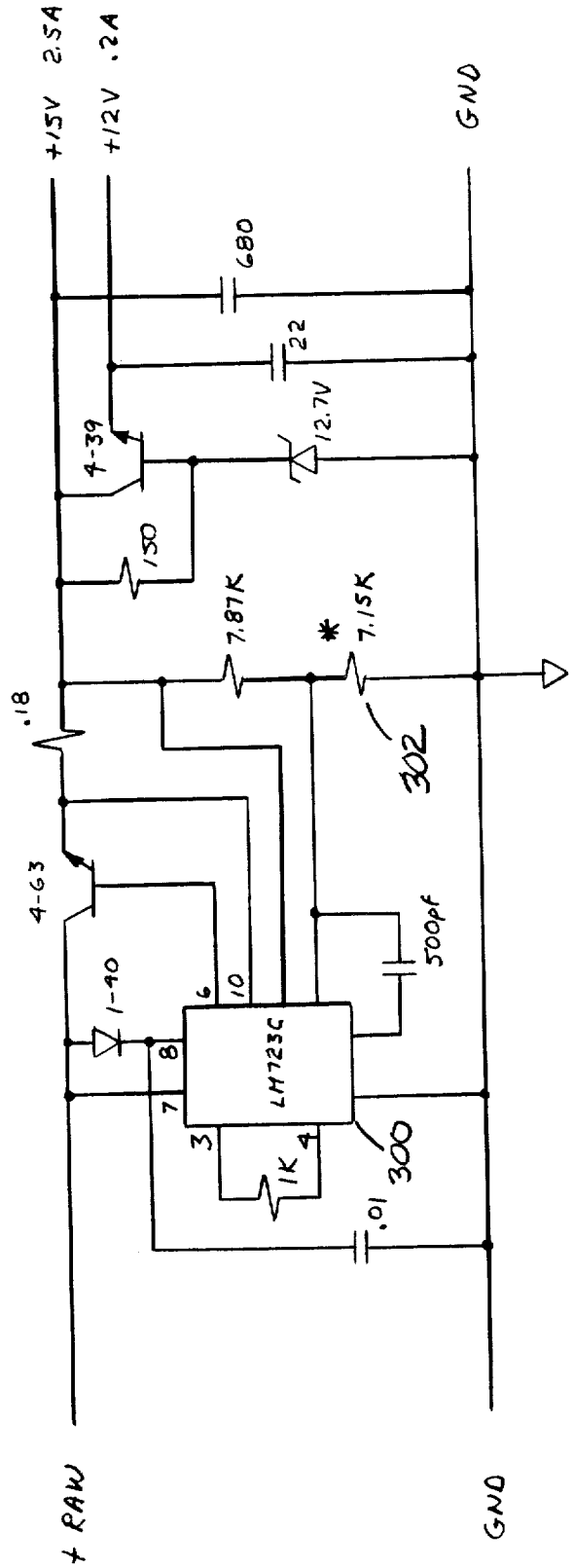

+15V 2.5A
+12V .2A
GND
680
22
12.7V
4-39
150
7.87K
7.15K
.18
302
4-63
1-90
500pf
LM723C
1K
.01
300
+ RAW
GND

FIG. 56

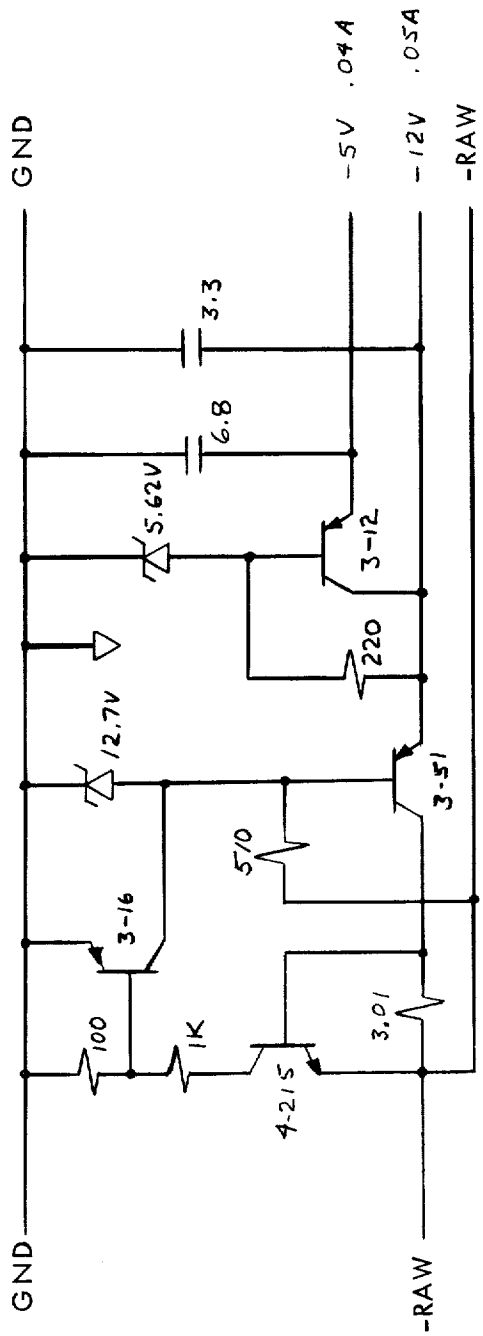

GND
-5V .04A
-12V .05A
-RAW
3.3
6.8
5.62V
3-12
220
12.7V
3-51
570
3-16
100
1K
3.01
4-215
GND
-RAW

FIG. 57

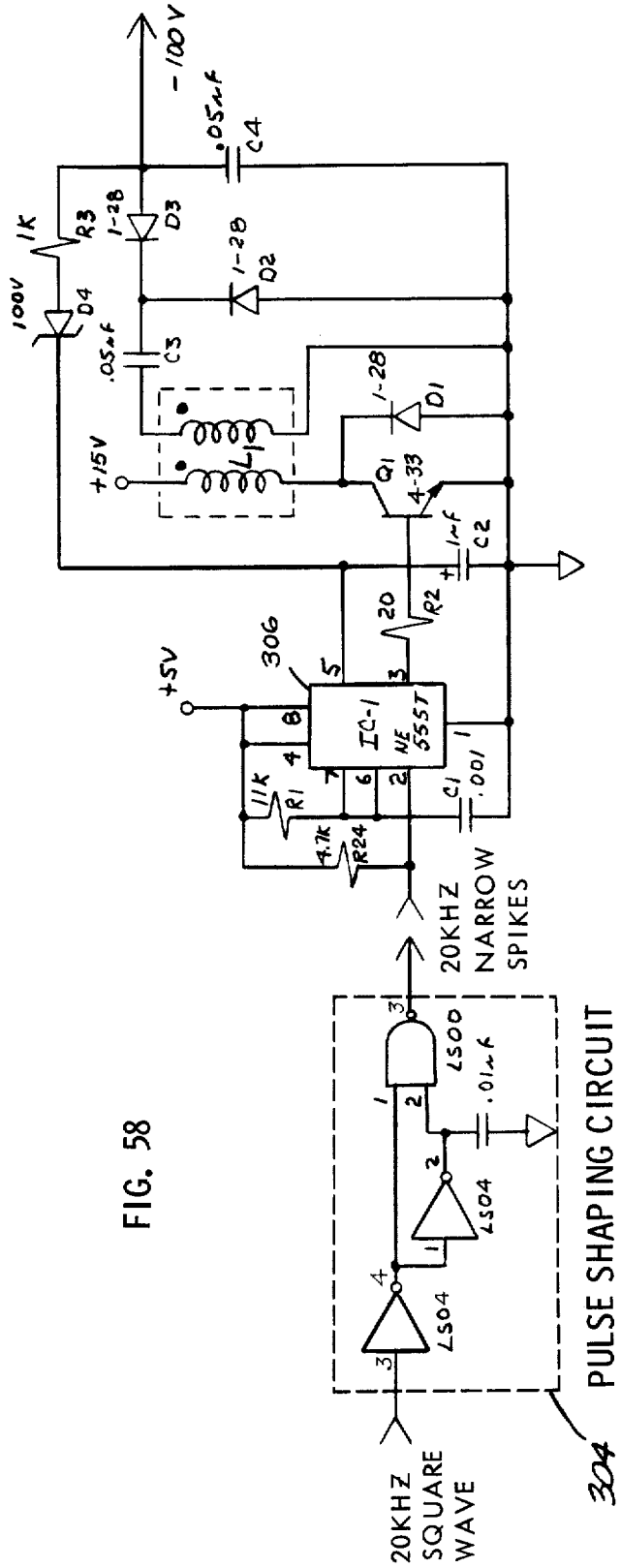

-100 V
.05uf
C4
1K
R3
1-28
D3
1-28
D2
100V
D4
.05uf
C3
1-28
D1
+15V
Q1
4-33
1uf
C2
20
R2
306
+5V
IC-1
NE 555T
11K
R1
4.7K
R24
.001
C1
20KHZ NARROW SPIKES
LS00
.01uf
LS04
LS04
20KHZ SQUARE WAVE
PULSE SHAPING CIRCUIT
304

FIG. 58

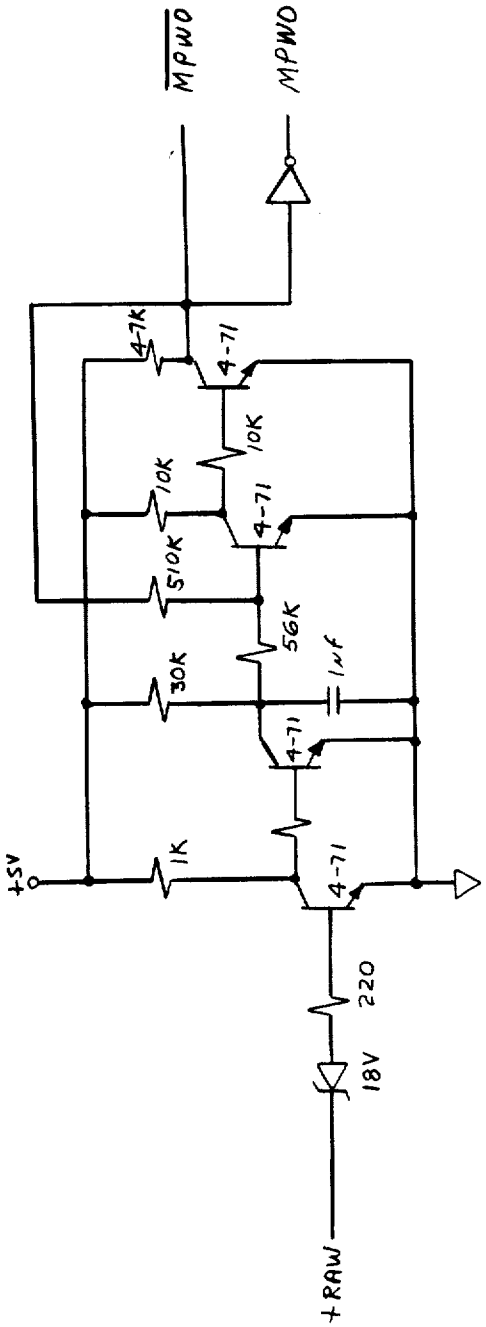

MPWD
MPWD
4-7K
4-71
10K
10K
4-71
510K
56K
30K
1uf
4-71
+5V
1K
4-71
220
18V
+RAW

FIG. 59

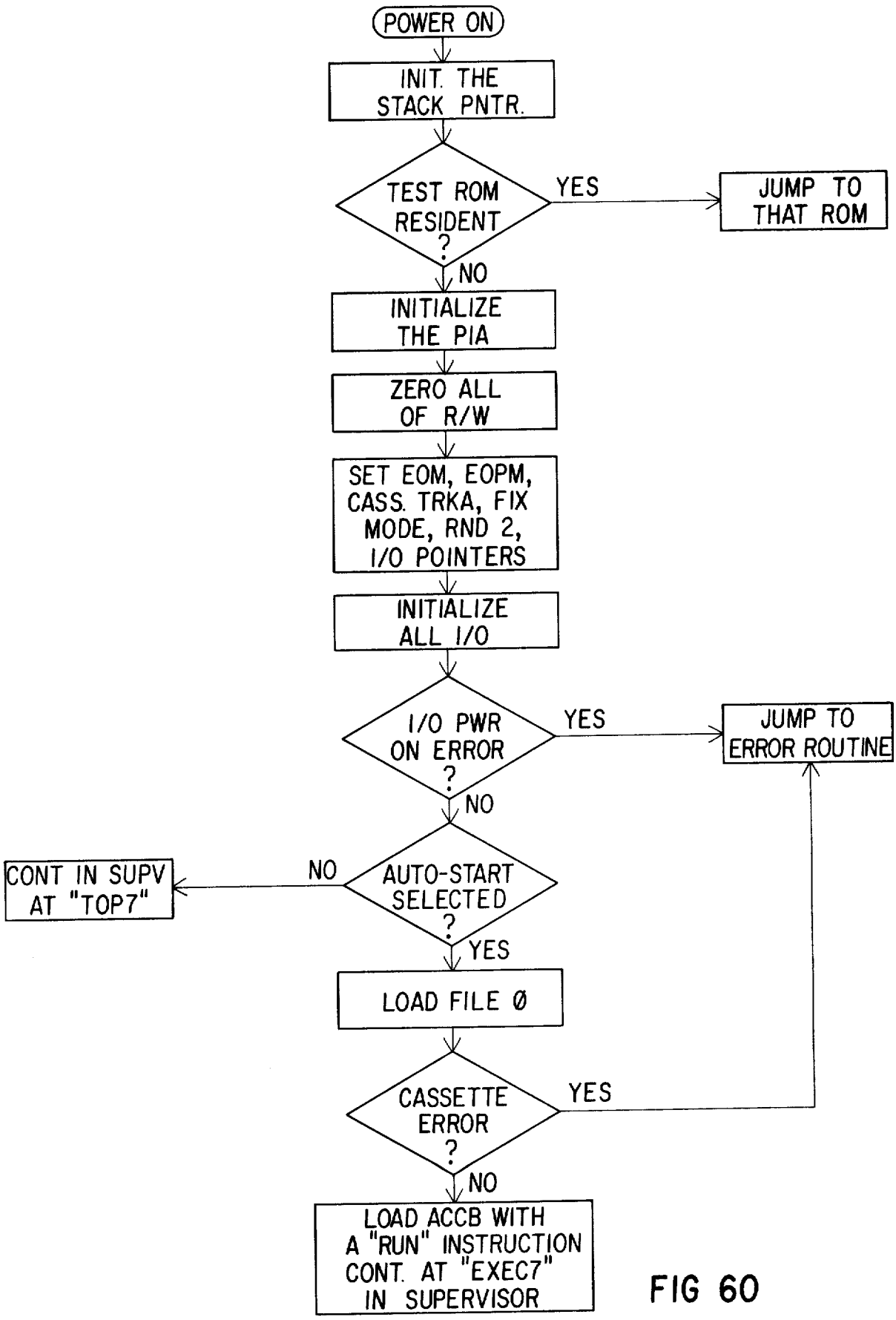

POWER ON
INIT. THE STACK PNTR.
TEST ROM RESIDENT ?
YES
JUMP TO THAT ROM
NO
INITIALIZE THE PIA
ZERO ALL OF R/W
SET EOM, EOPM, CASS. TRKA, FIX MODE, RND 2, I/O POINTERS
INITIALIZE ALL I/O
I/O PWR ON ERROR ?
YES
JUMP TO ERROR ROUTINE
NO
AUTO-START SELECTED ?
NO
CONT IN SUPV AT "TOP7"
YES
LOAD FILE Ø
CASSETTE ERROR ?
YES
NO
LOAD ACCB WITH A "RUN" INSTRUCTION CONT. AT "EXEC7" IN SUPERVISOR

FIG 60

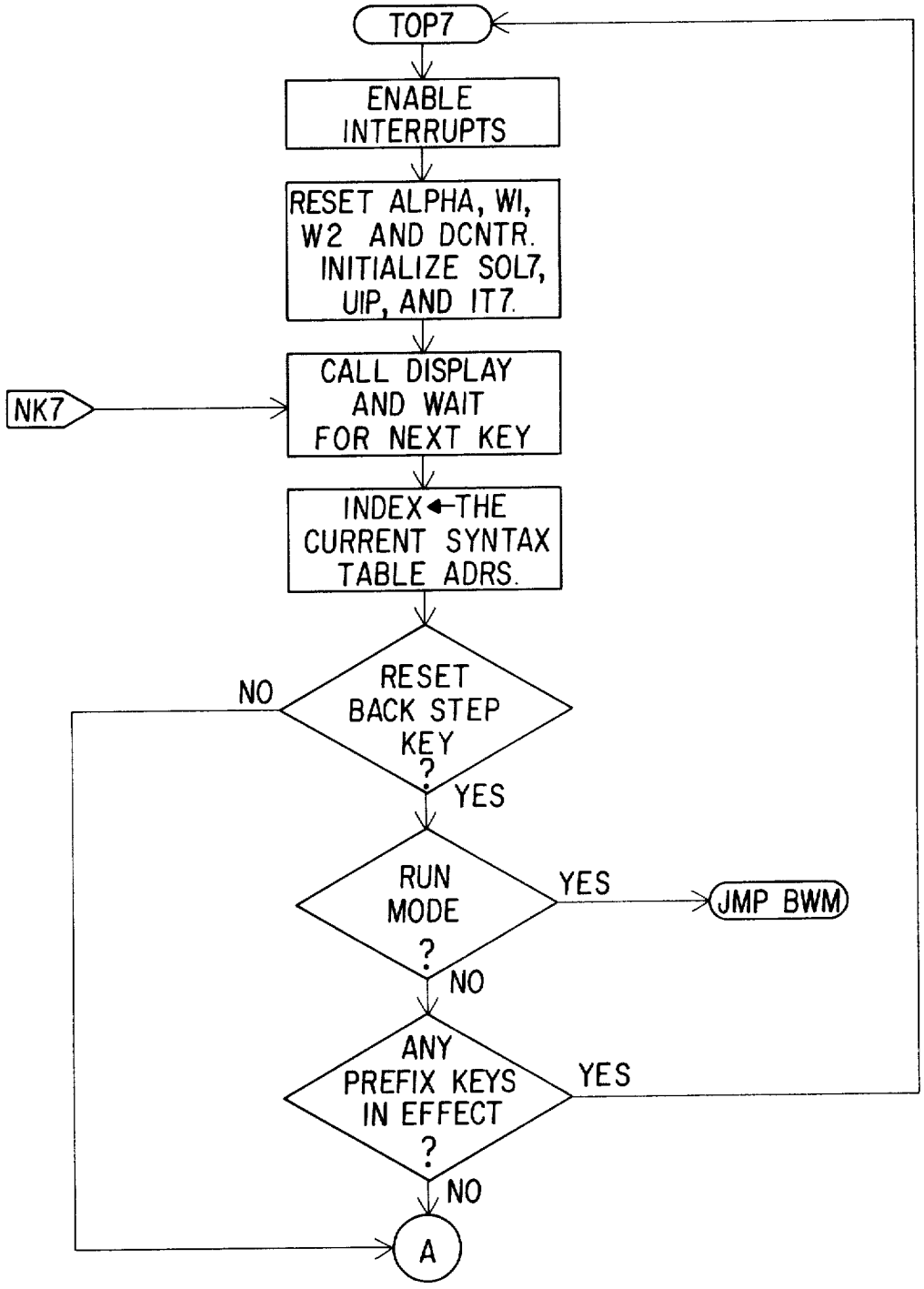

TOP7
ENABLE INTERRUPTS
RESET ALPHA, WI, W2 AND DCNTR. INITIALIZE SOL7, UIP, AND IT7.
NK7
CALL DISPLAY AND WAIT FOR NEXT KEY
INDEX←THE CURRENT SYNTAX TABLE ADRS.
RESET BACK STEP KEY ?
NO
YES
RUN MODE ?
YES
JMP BWM
NO
ANY PREFIX KEYS IN EFFECT ?
YES
NO
A

FIG 61A

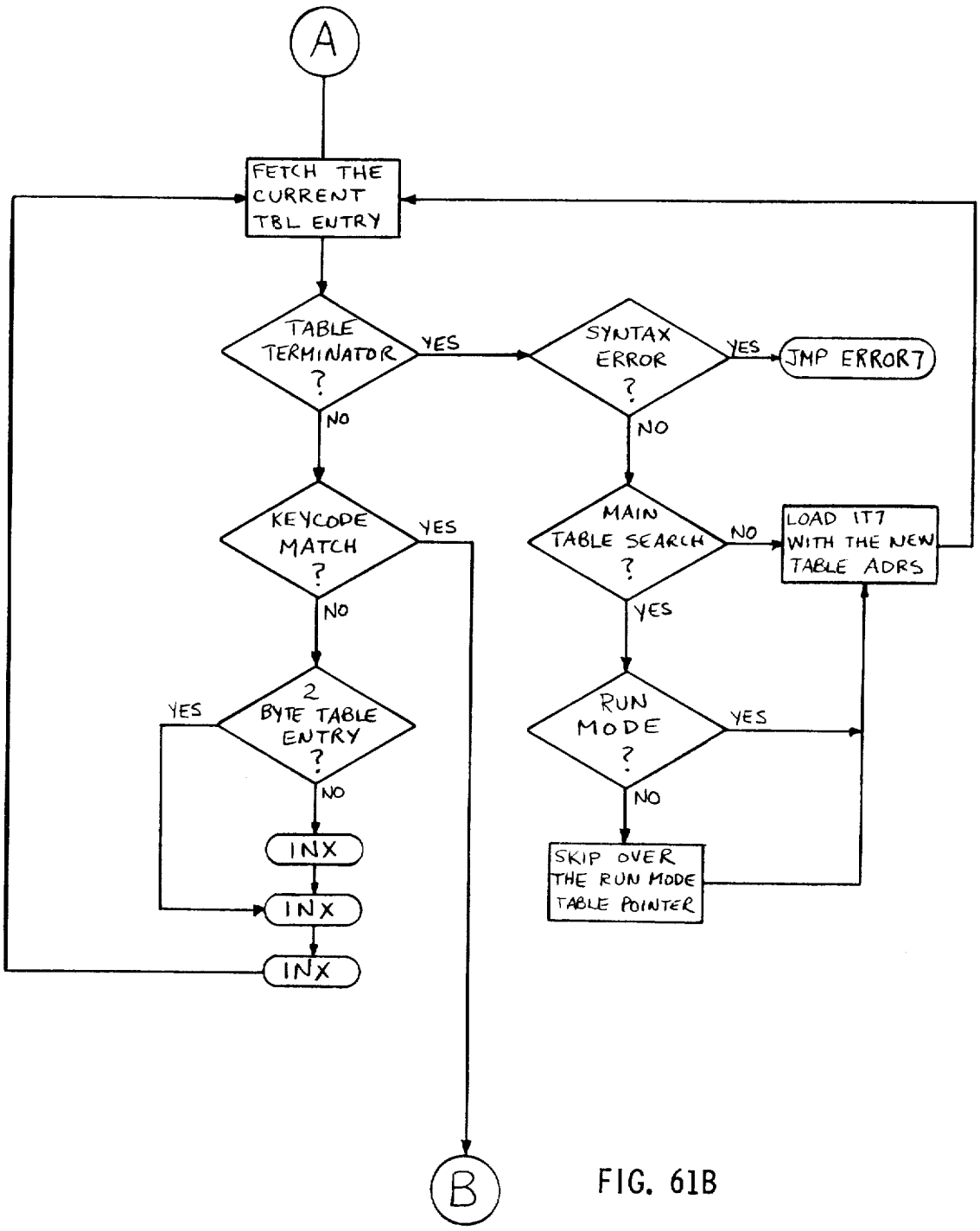

A
FETCH THE CURRENT TBL ENTRY
TABLE TERMINATOR ?
YES
SYNTAX ERROR ?
YES
JMP ERROR7
NO
NO
KEYCODE MATCH ?
YES
MAIN TABLE SEARCH ?
NO
LOAD IT7 WITH THE NEW TABLE ADRS
NO
YES
YES
2 BYTE TABLE ENTRY ?
RUN MODE ?
YES
NO
NO
INX
SKIP OVER THE RUN MODE TABLE POINTER
INX
INX
B

FIG. 61B

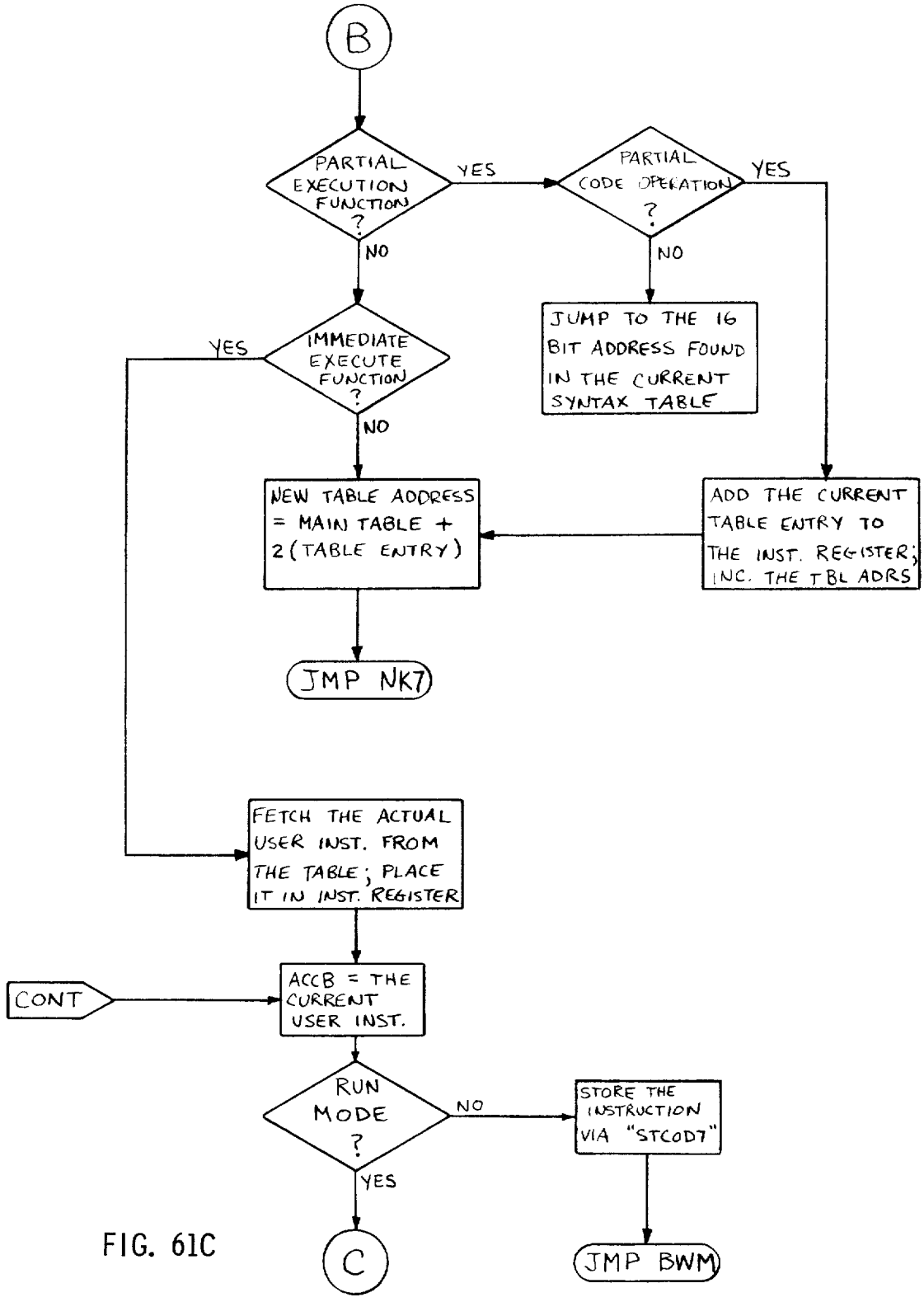

B
PARTIAL EXECUTION FUNCTION ?
YES
PARTIAL CODE OPERATION ?
YES
NO
NO
JUMP TO THE 16 BIT ADDRESS FOUND IN THE CURRENT SYNTAX TABLE
YES
IMMEDIATE EXECUTE FUNCTION ?
NO
NEW TABLE ADDRESS = MAIN TABLE + 2(TABLE ENTRY)
ADD THE CURRENT TABLE ENTRY TO THE INST. REGISTER; INC. THE TBL ADRS
JMP NK7
FETCH THE ACTUAL USER INST. FROM THE TABLE; PLACE IT IN INST. REGISTER
CONT
ACCB = THE CURRENT USER INST.
RUN MODE ?
NO
STORE THE INSTRUCTION VIA "STCOD7"
YES
C
JMP BWM

FIG. 61C

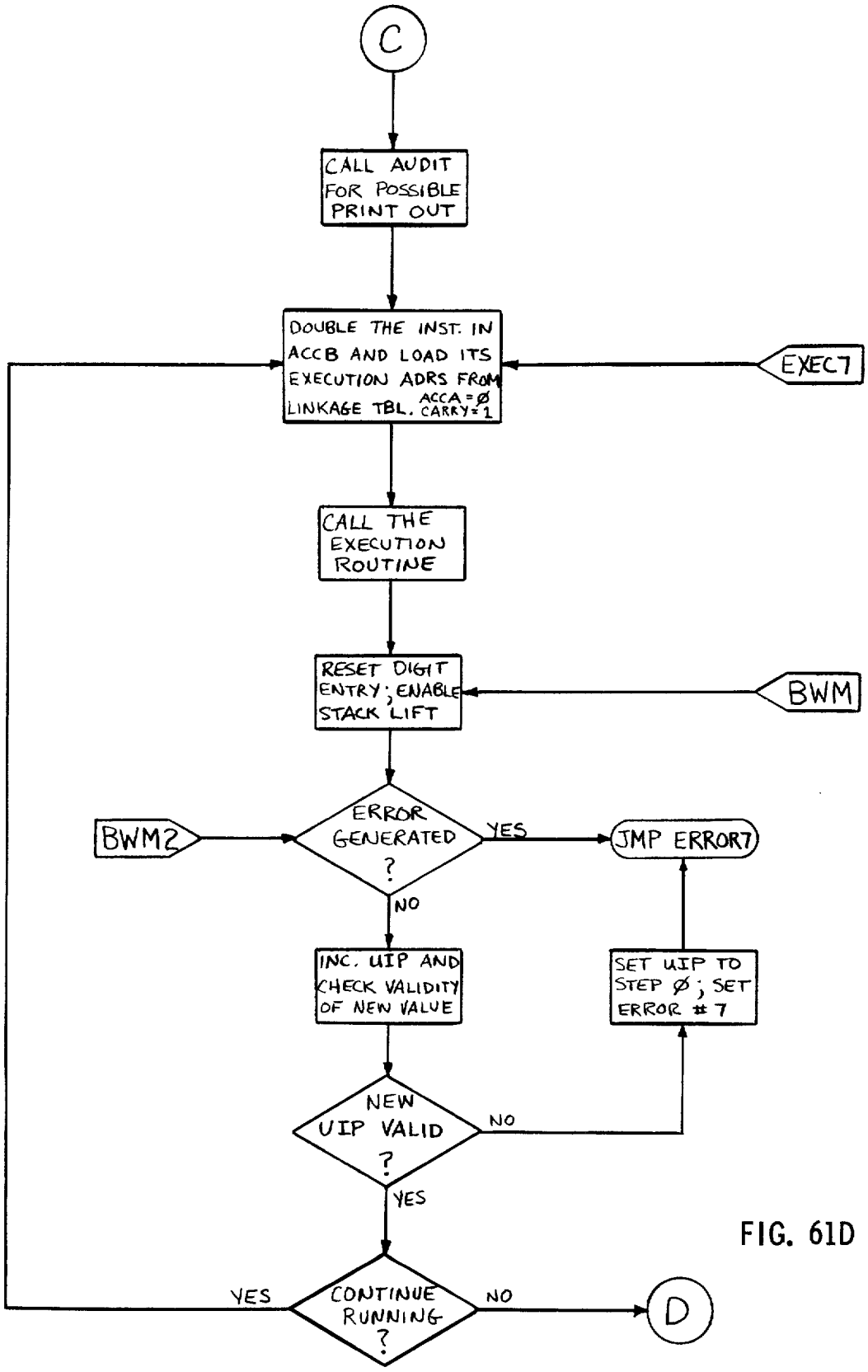

C
CALL AUDIT FOR POSSIBLE PRINT OUT
DOUBLE THE INST. IN ACCB AND LOAD ITS EXECUTION ADRS FROM LINKAGE TBL. ACCA=Ø CARRY=1
EXEC7
CALL THE EXECUTION ROUTINE
RESET DIGIT ENTRY; ENABLE STACK LIFT
BWM
BWM2
ERROR GENERATED ?
YES
JMP ERROR7
NO
INC. UIP AND CHECK VALIDITY OF NEW VALUE
SET UIP TO STEP Ø; SET ERROR # 7
NEW UIP VALID ?
NO
YES
CONTINUE RUNNING ?
YES
NO
D

FIG. 61D

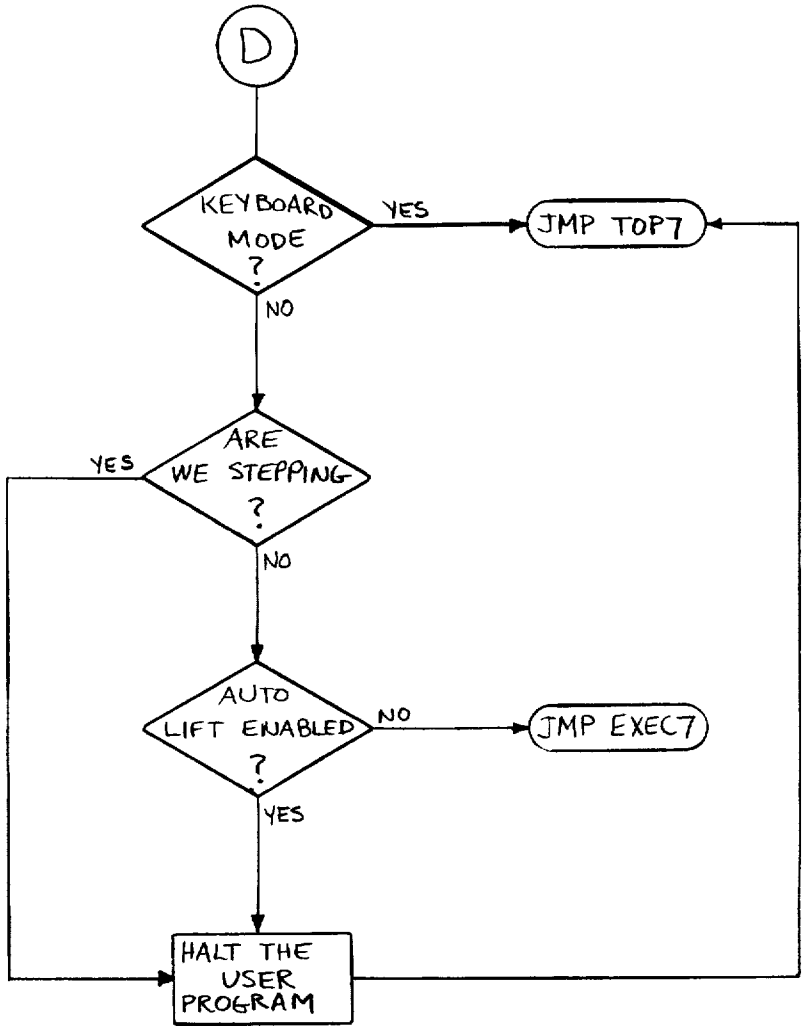

D
KEYBOARD MODE ?
YES
JMP TOP7
NO
ARE WE STEPPING ?
YES
NO
AUTO LIFT ENABLED ?
NO
JMP EXEC7
YES
HALT THE USER PROGRAM

FIG. 61E

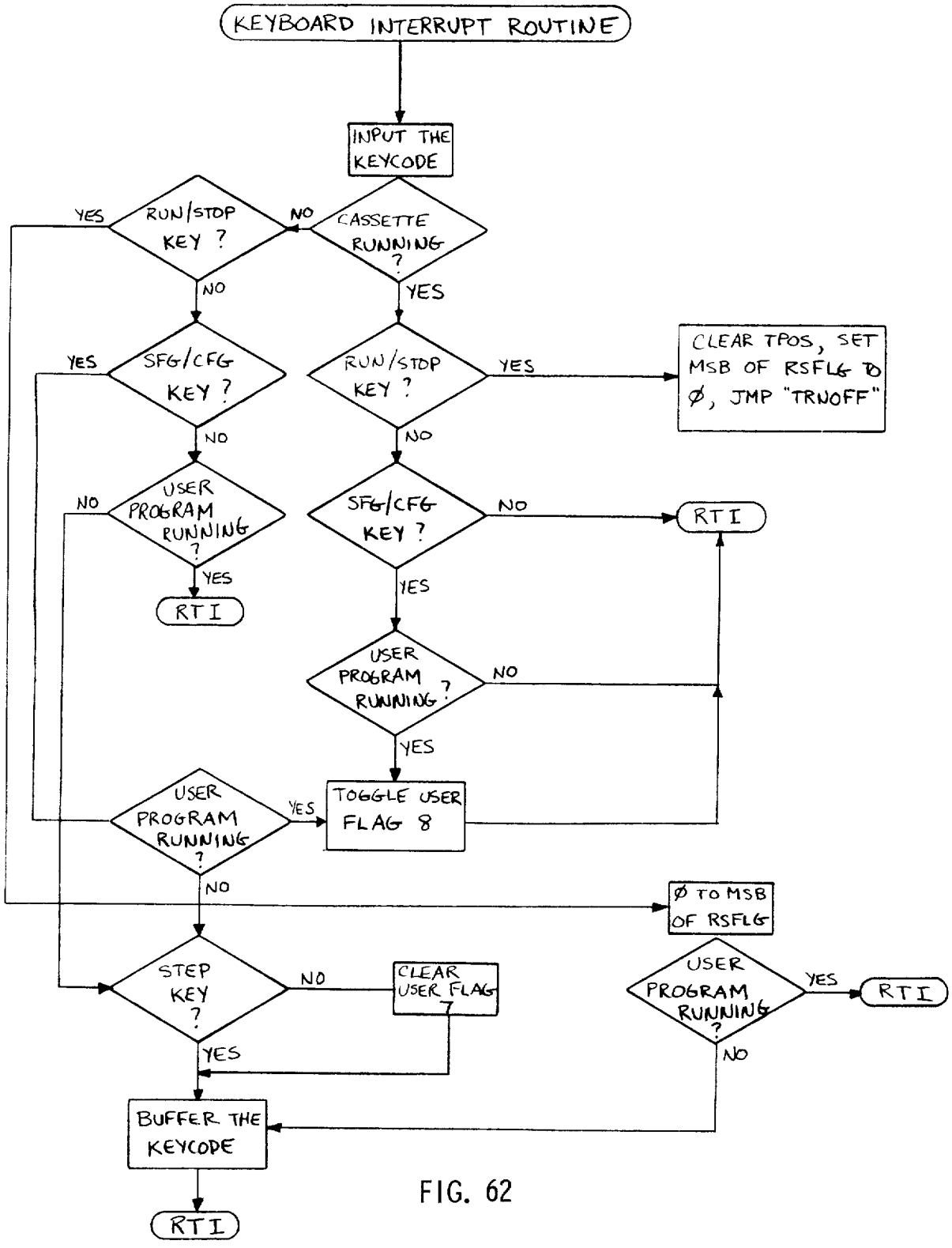

KEYBOARD INTERRUPT ROUTINE
INPUT THE KEYCODE
RUN/STOP KEY ?
YES
NO
CASSETTE RUNNING ?
NO
YES
SFG/CFG KEY ?
YES
NO
RUN/STOP KEY ?
YES
CLEAR TPOS, SET MSB OF RSFLG TO Ø, JMP "TRNOFF"
NO
USER PROGRAM RUNNING ?
NO
YES
RTI
SFG/CFG KEY ?
NO
RTI
YES
USER PROGRAM RUNNING ?
NO
YES
USER PROGRAM RUNNING ?
YES
TOGGLE USER FLAG 8
NO
Ø TO MSB OF RSFLG
STEP KEY ?
NO
CLEAR USER FLAG
YES
USER PROGRAM RUNNING ?
YES
RTI
NO
BUFFER THE KEYCODE
RTI

FIG. 62

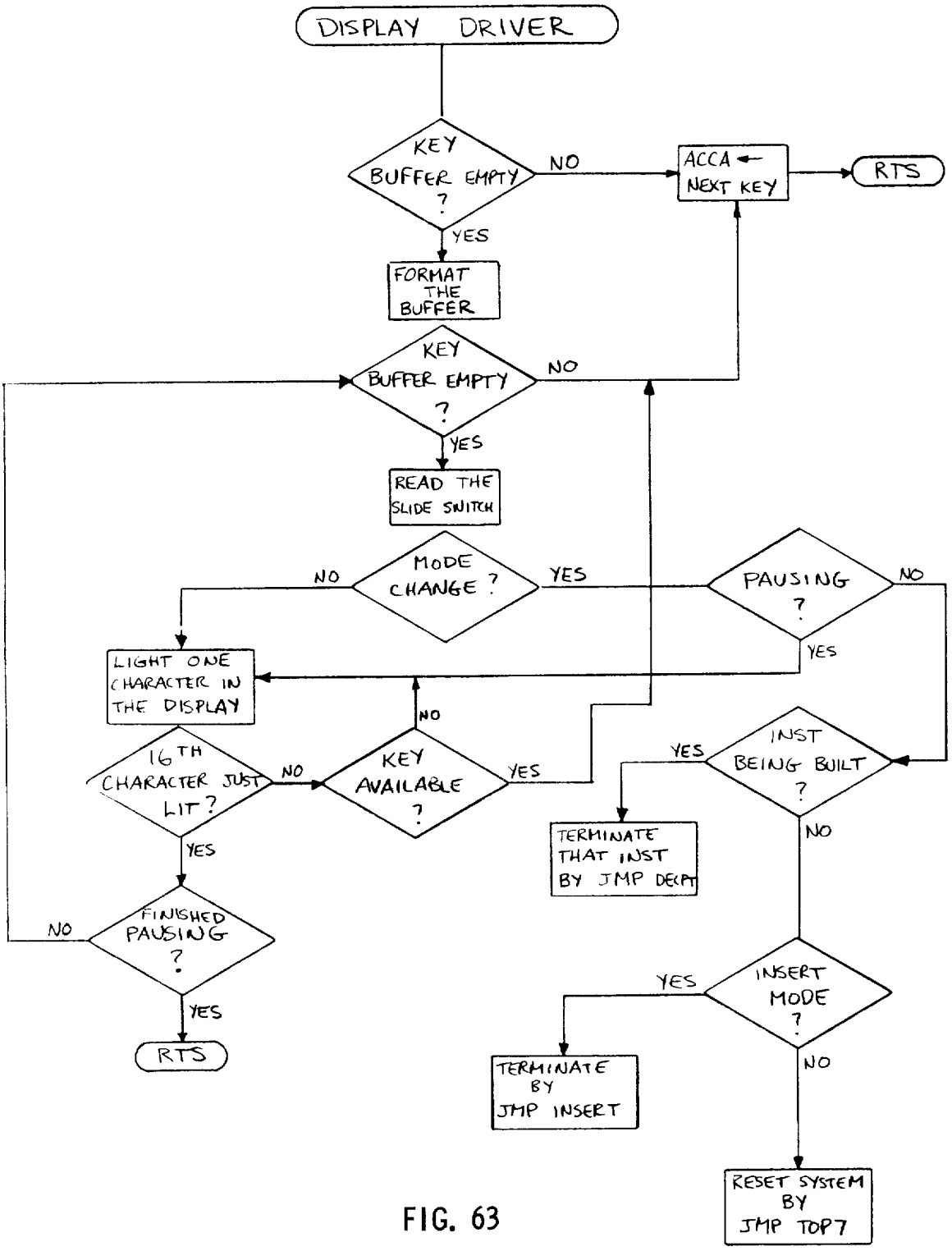

DISPLAY DRIVER
KEY BUFFER EMPTY ?
NO
ACCA ← NEXT KEY
RTS
YES
FORMAT THE BUFFER
KEY BUFFER EMPTY ?
NO
YES
READ THE SLIDE SWITCH
MODE CHANGE ?
NO
YES
PAUSING ?
NO
YES
LIGHT ONE CHARACTER IN THE DISPLAY
16TH CHARACTER JUST LIT ?
NO
KEY AVAILABLE ?
NO
YES
YES
INST BEING BUILT ?
YES
TERMINATE THAT INST BY JMP DECPT
NO
FINISHED PAUSING ?
NO
YES
RTS
INSERT MODE ?
YES
NO
TERMINATE BY JMP INSERT
RESET SYSTEM BY JMP TOP7

FIG. 63

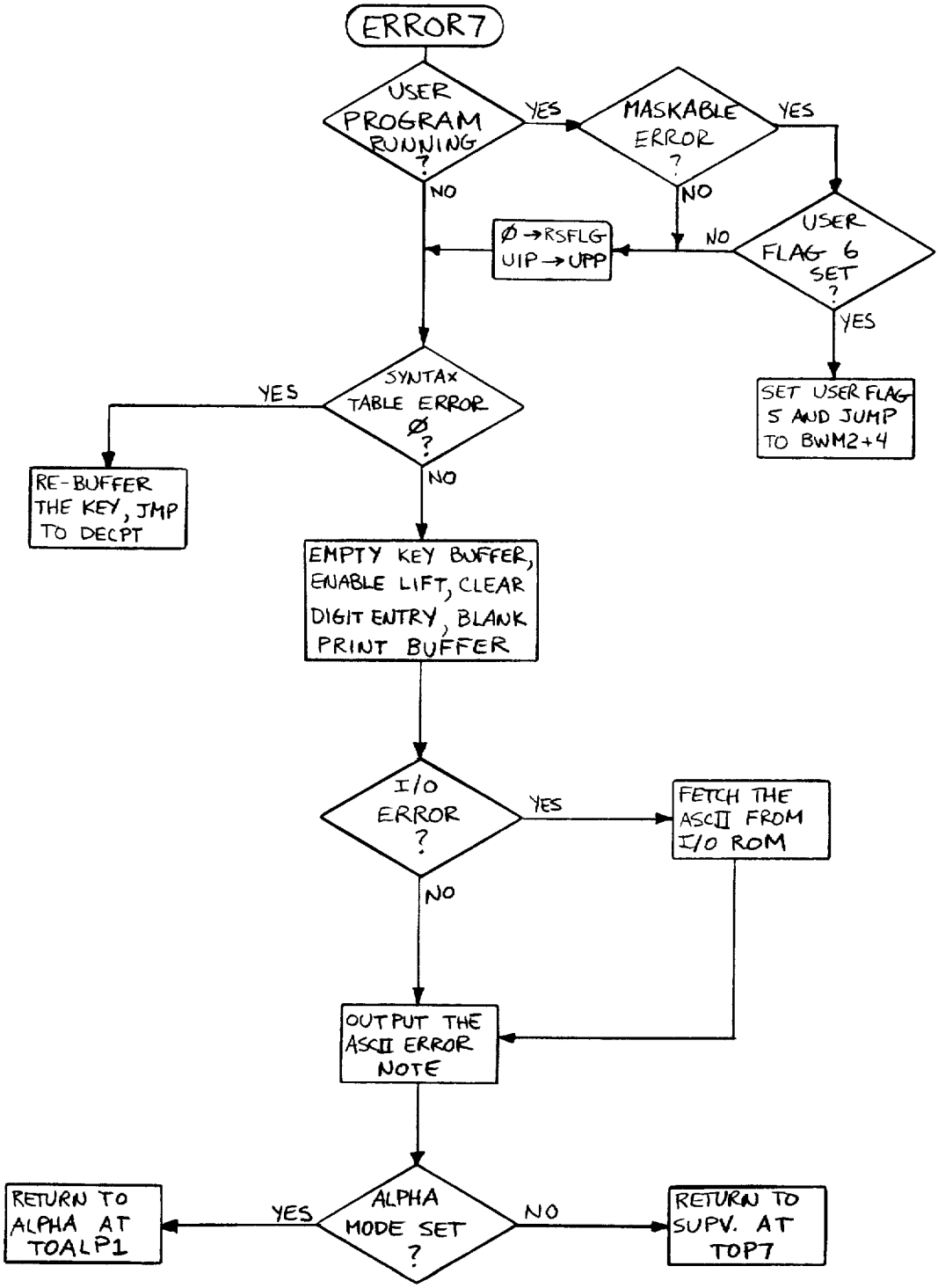

ERROR7
USER PROGRAM RUNNING ?
YES
MASKABLE ERROR ?
YES
NO
NO
USER FLAG 6 SET ?
NO
Ø→RSFLG
UIP→UPP
YES
SET USER FLAG 5 AND JUMP TO BWM2+4
SYNTAX TABLE ERROR Ø ?
YES
RE-BUFFER THE KEY, JMP TO DECPT
NO
EMPTY KEY BUFFER, ENABLE LIFT, CLEAR DIGIT ENTRY, BLANK PRINT BUFFER
I/O ERROR ?
YES
FETCH THE ASCII FROM I/O ROM
NO
OUTPUT THE ASCII ERROR NOTE
ALPHA MODE SET ?
YES
RETURN TO ALPHA AT TOALP1
NO
RETURN TO SUPV. AT TOP7

FIG. 64

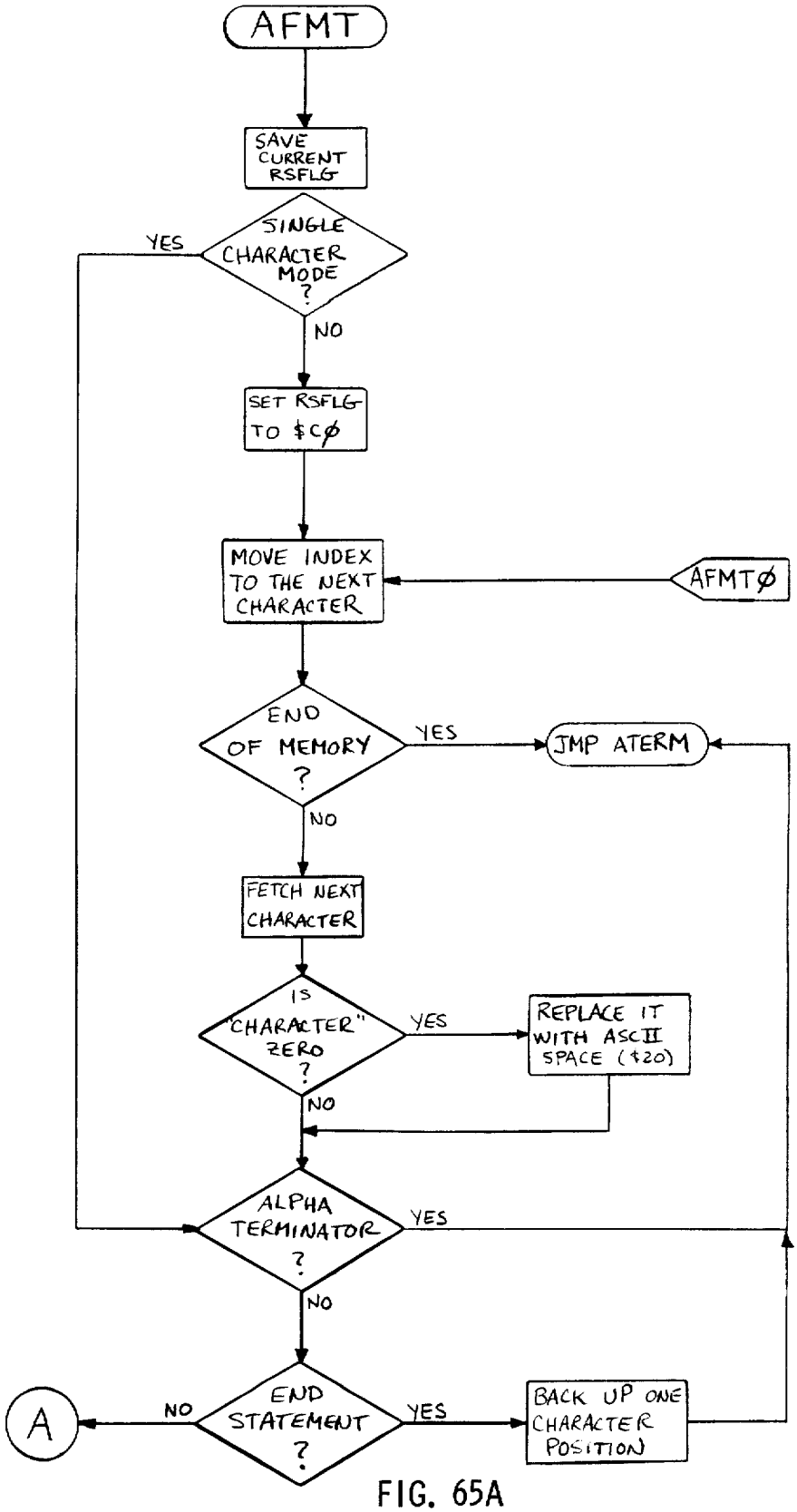

AFMT
SAVE CURRENT RSFLG
SINGLE CHARACTER MODE ?
YES
NO
SET RSFLG TO $CØ
MOVE INDEX TO THE NEXT CHARACTER
AFMTØ
END OF MEMORY ?
YES
JMP ATERM
NO
FETCH NEXT CHARACTER
IS "CHARACTER" ZERO ?
YES
REPLACE IT WITH ASCII SPACE (†20)
NO
ALPHA TERMINATOR ?
YES
NO
END STATEMENT ?
NO
A
YES
BACK UP ONE CHARACTER POSITION

FIG. 65A

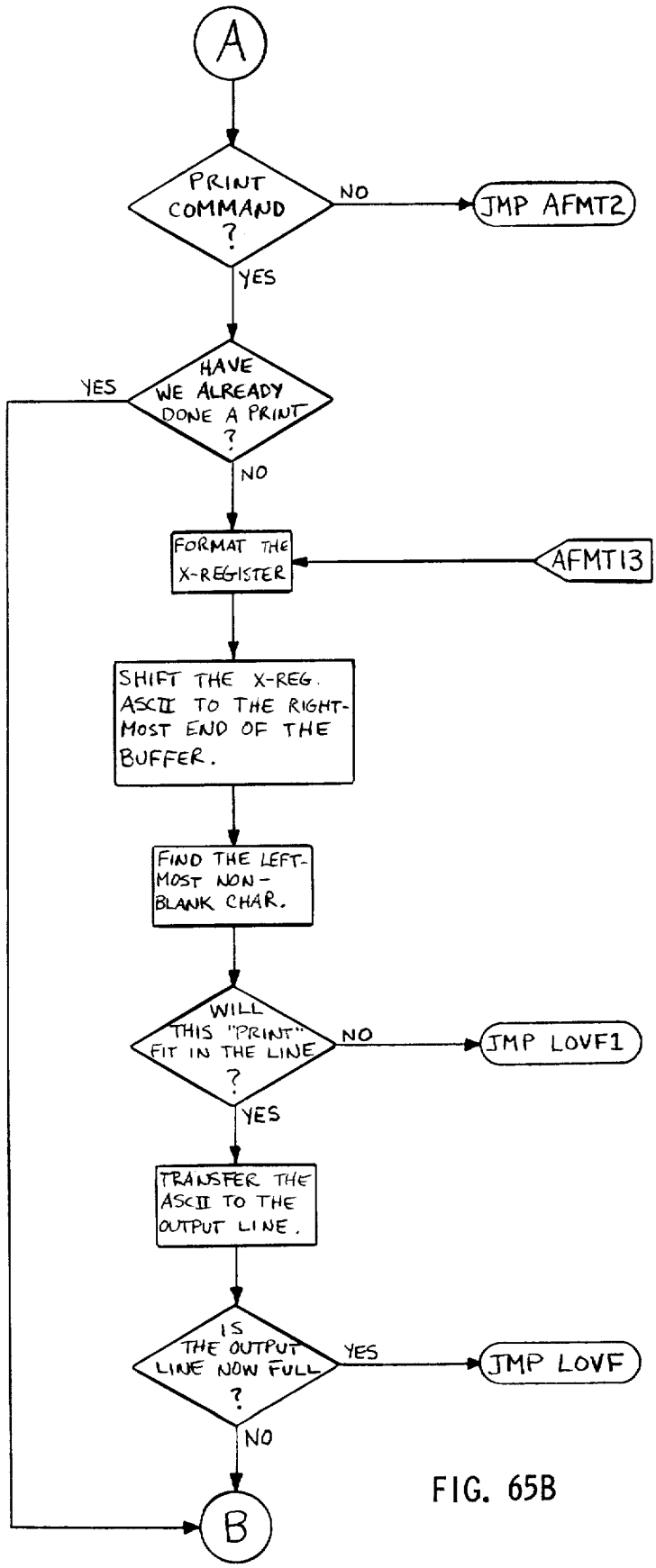

A
PRINT COMMAND ?
NO
JMP AFMT2
YES
HAVE WE ALREADY DONE A PRINT ?
YES
NO
FORMAT THE X-REGISTER
AFMT13
SHIFT THE X-REG. ASCII TO THE RIGHT-MOST END OF THE BUFFER.
FIND THE LEFT-MOST NON-BLANK CHAR.
WILL THIS "PRINT" FIT IN THE LINE ?
NO
JMP LOVF1
YES
TRANSFER THE ASCII TO THE OUTPUT LINE.
IS THE OUTPUT LINE NOW FULL ?
YES
JMP LOVF
NO
B

FIG. 65B

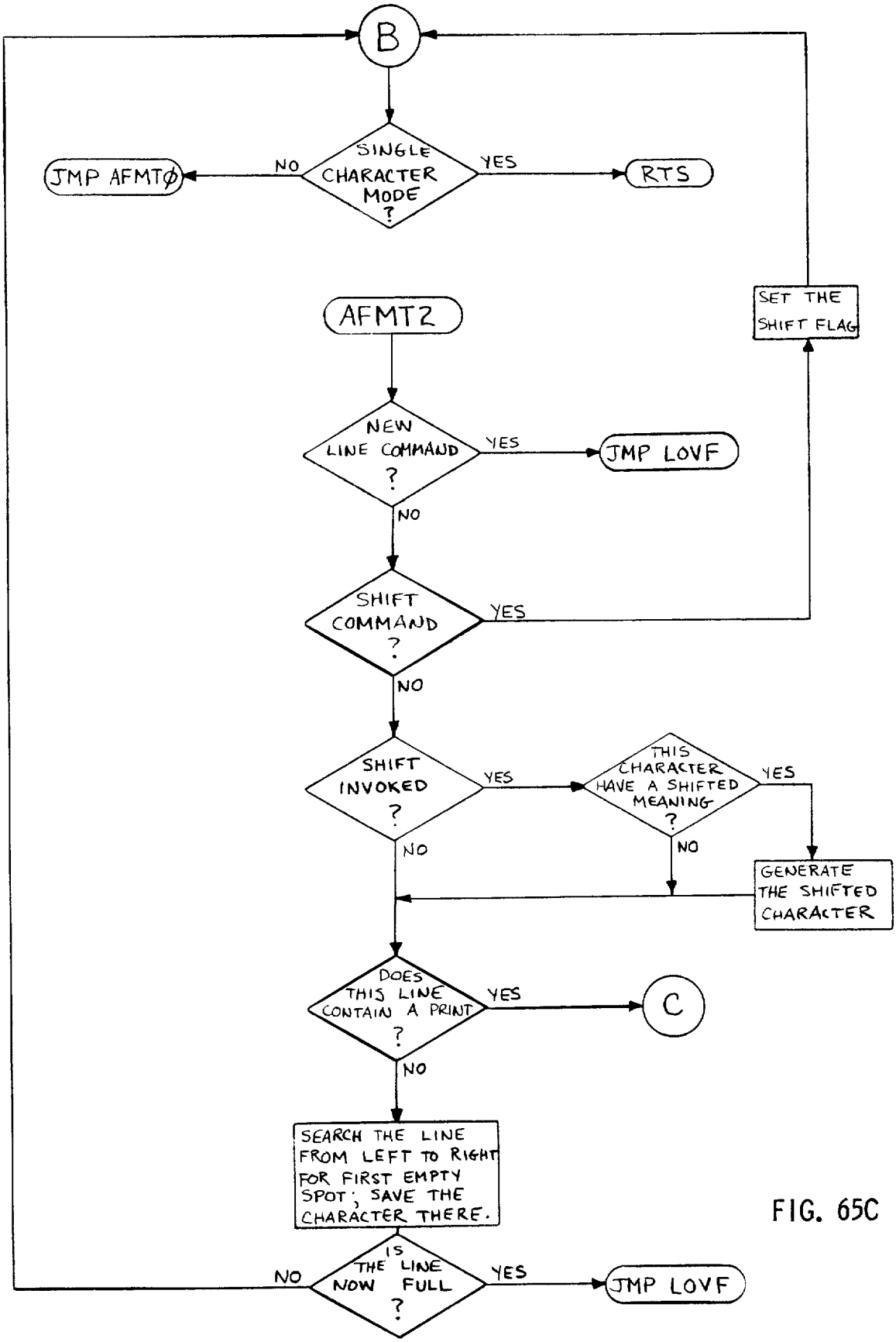

B
SINGLE CHARACTER MODE ?
NO
YES
JMP AFMTØ
RTS
AFMT2
SET THE SHIFT FLAG
NEW LINE COMMAND ?
YES
JMP LOVF
NO
SHIFT COMMAND ?
YES
NO
SHIFT INVOKED ?
YES
THIS CHARACTER HAVE A SHIFTED MEANING ?
YES
NO
NO
GENERATE THE SHIFTED CHARACTER
DOES THIS LINE CONTAIN A PRINT ?
YES
C
NO
SEARCH THE LINE FROM LEFT TO RIGHT FOR FIRST EMPTY SPOT; SAVE THE CHARACTER THERE.
IS THE LINE NOW FULL ?
NO
YES
JMP LOVF

FIG. 65C

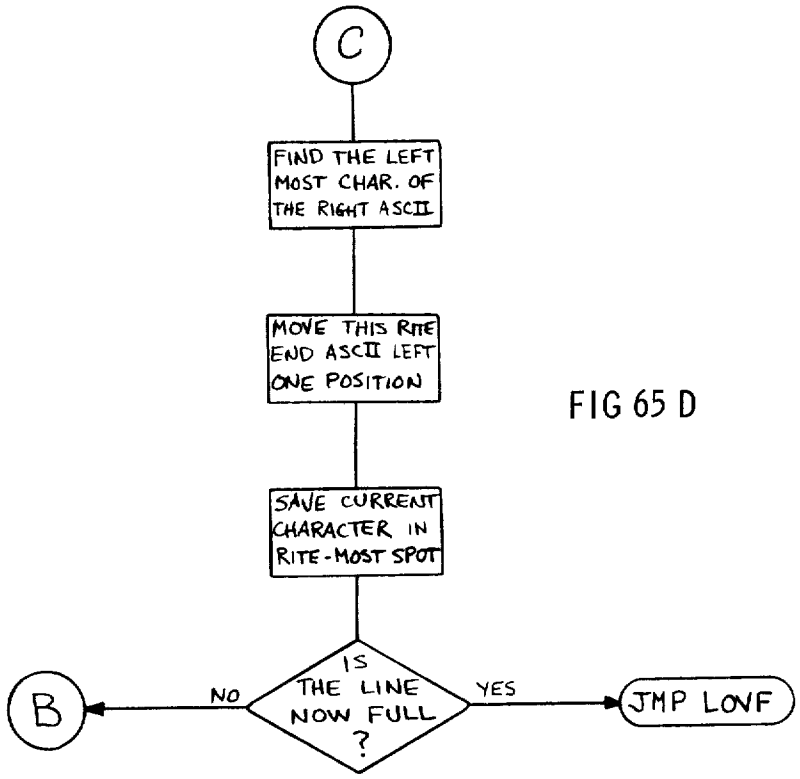


FIG 65 D

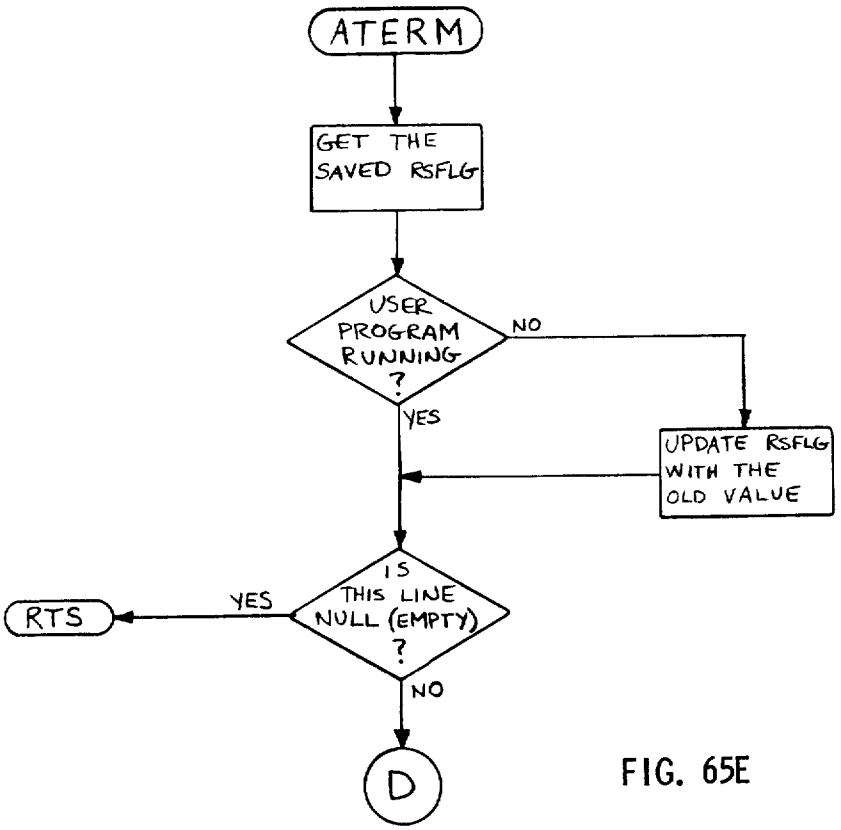


FIG. 65E

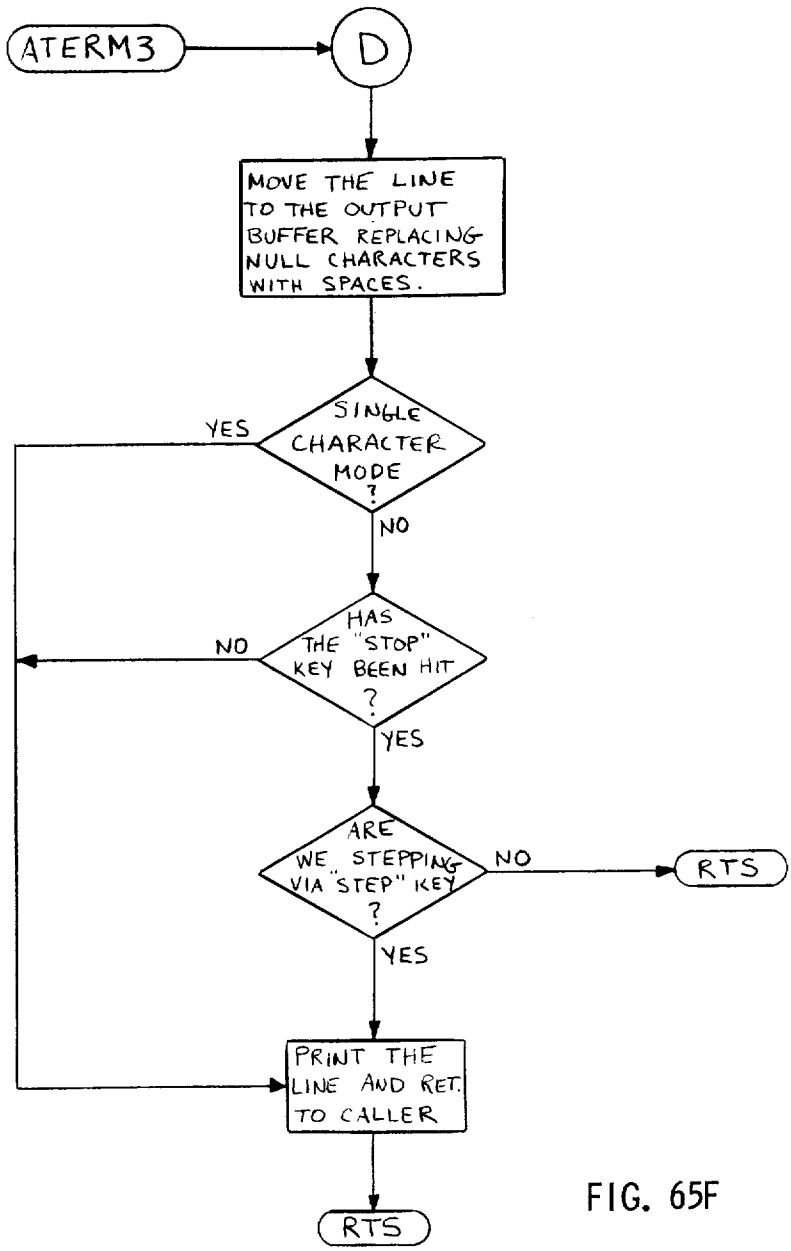

ATERM3
D
MOVE THE LINE TO THE OUTPUT BUFFER REPLACING NULL CHARACTERS WITH SPACES.
SINGLE CHARACTER MODE ?
YES
NO
HAS THE "STOP" KEY BEEN HIT ?
NO
YES
ARE WE STEPPING VIA "STEP" KEY ?
NO
RTS
YES
PRINT THE LINE AND RET. TO CALLER
RTS

FIG. 65F

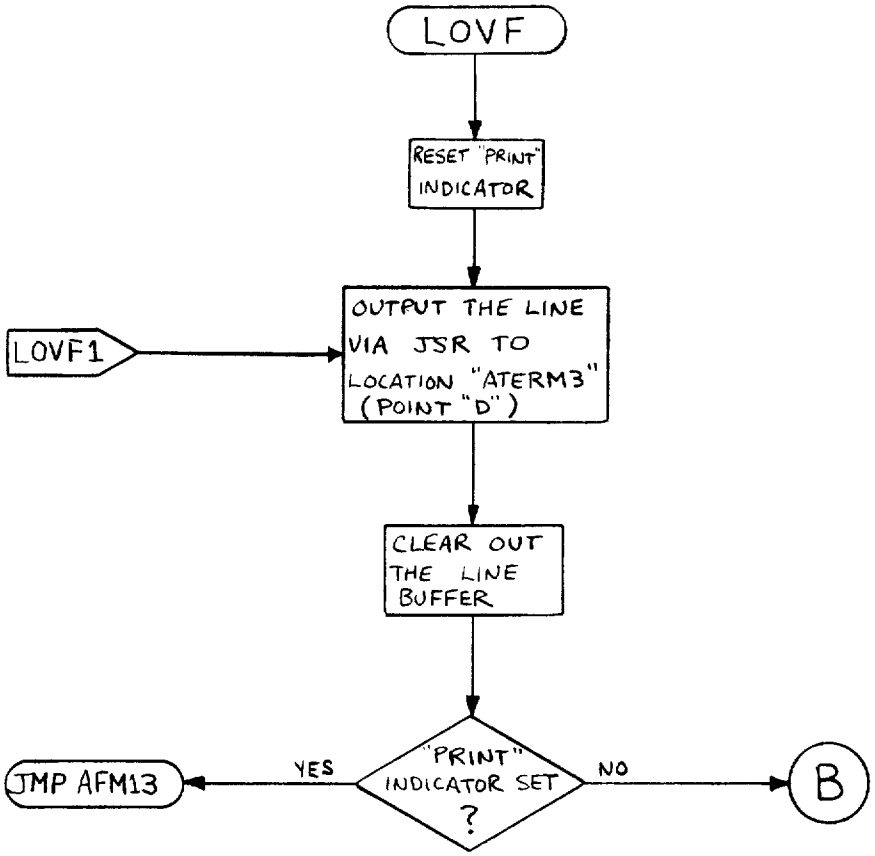

LOVF
RESET "PRINT" INDICATOR
LOVF1
OUTPUT THE LINE VIA JSR TO LOCATION "ATERM3" (POINT "D")
CLEAR OUT THE LINE BUFFER
"PRINT" INDICATOR SET ?
YES
NO
JMP AFM13
B

FIG. 65G

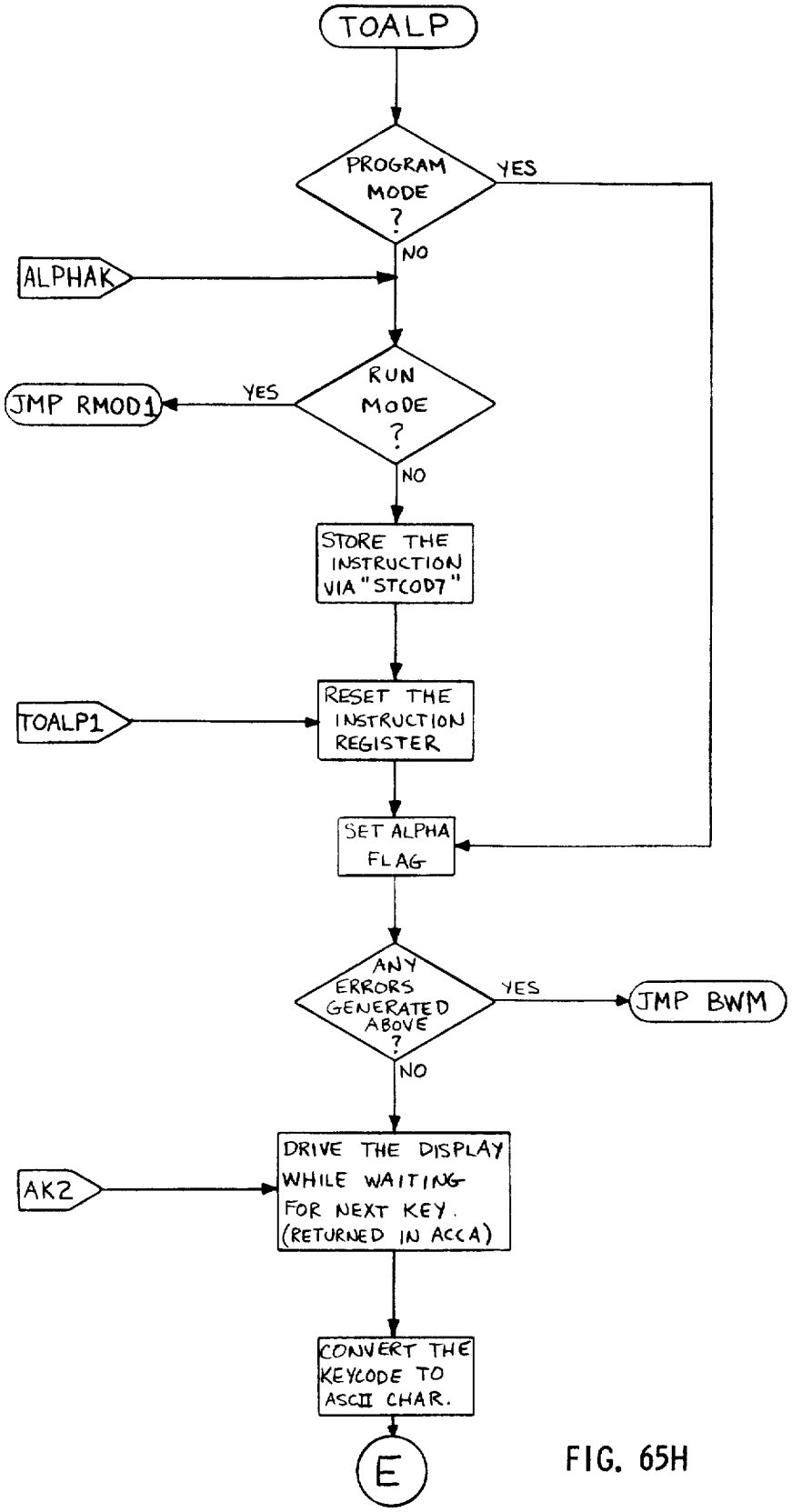

TOALP
PROGRAM MODE ?
YES
NO
ALPHAK
RUN MODE ?
YES
JMP RMOD1
NO
STORE THE INSTRUCTION VIA "STCOD7"
TOALP1
RESET THE INSTRUCTION REGISTER
SET ALPHA FLAG
ANY ERRORS GENERATED ABOVE ?
YES
JMP BWM
NO
AK2
DRIVE THE DISPLAY WHILE WAITING FOR NEXT KEY. (RETURNED IN ACCA)
CONVERT THE KEYCODE TO ASCII CHAR.
E

FIG. 65H

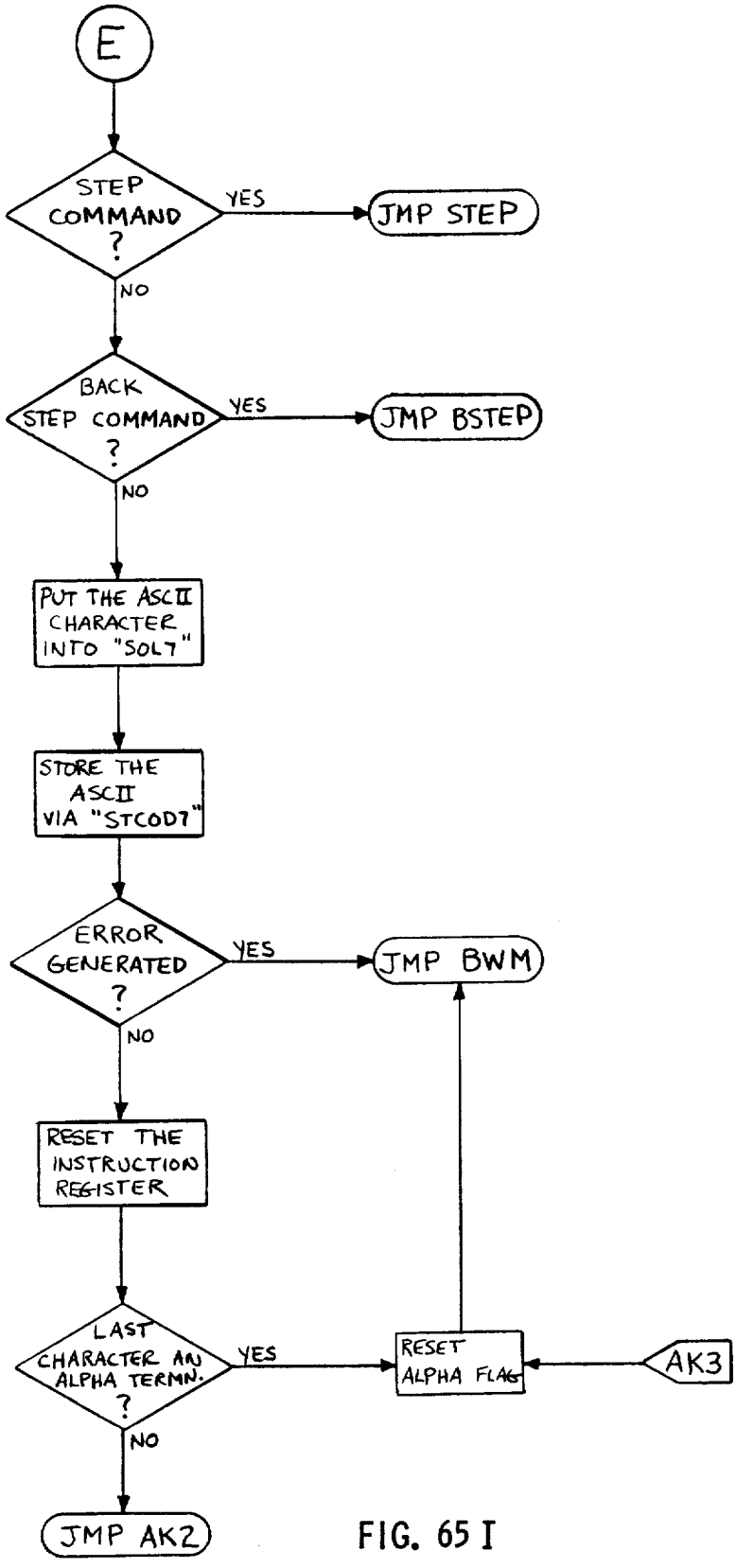

E
STEP COMMAND ?
YES
JMP STEP
NO
BACK STEP COMMAND ?
YES
JMP BSTEP
NO
PUT THE ASCII CHARACTER INTO "SOL7"
STORE THE ASCII VIA "STCOD7"
ERROR GENERATED ?
YES
JMP BWM
NO
RESET THE INSTRUCTION REGISTER
LAST CHARACTER AN ALPHA TERMN. ?
YES
RESET ALPHA FLAG
AK3
NO
JMP AK2

FIG. 65 I

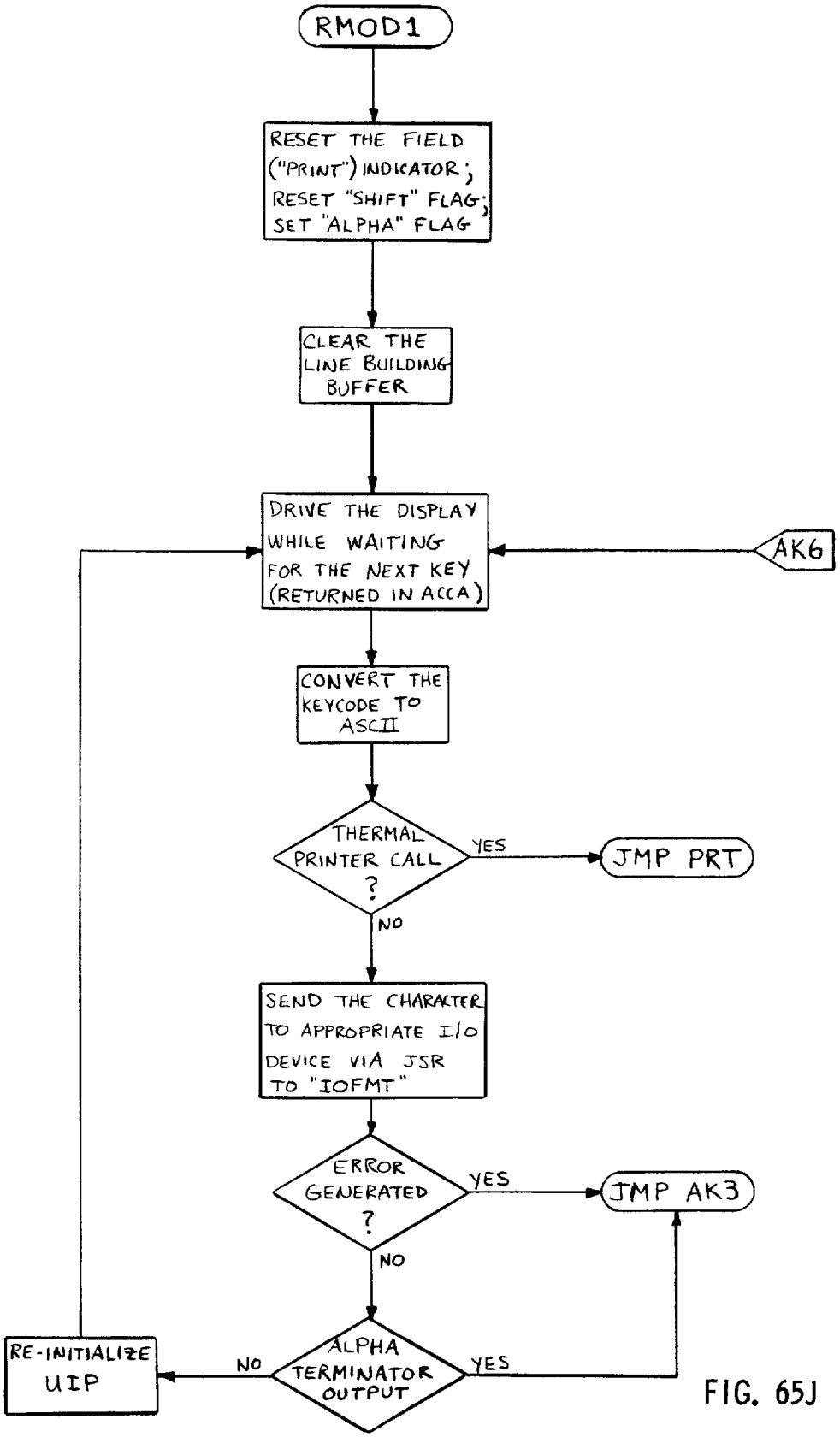

RMOD1
RESET THE FIELD ("PRINT") INDICATOR; RESET "SHIFT" FLAG; SET "ALPHA" FLAG
CLEAR THE LINE BUILDING BUFFER
DRIVE THE DISPLAY WHILE WAITING FOR THE NEXT KEY (RETURNED IN ACCA)
AK6
CONVERT THE KEYCODE TO ASCII
THERMAL PRINTER CALL ?
YES
JMP PRT
NO
SEND THE CHARACTER TO APPROPRIATE I/O DEVICE VIA JSR TO "IOFMT"
ERROR GENERATED ?
YES
JMP AK3
NO
ALPHA TERMINATOR OUTPUT
YES
NO
RE-INITIALIZE UIP

FIG. 65J

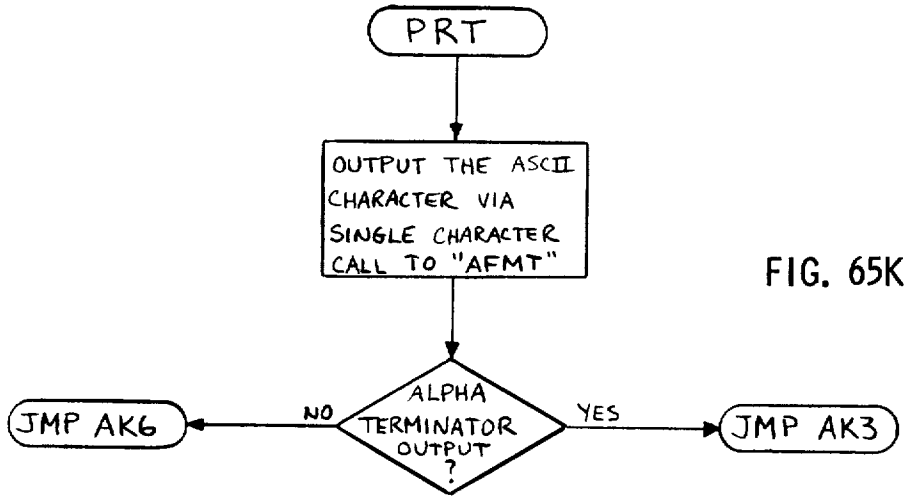


FIG. 65K

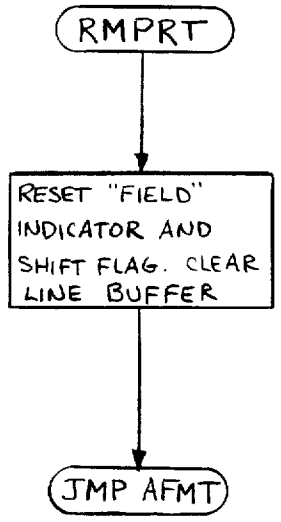
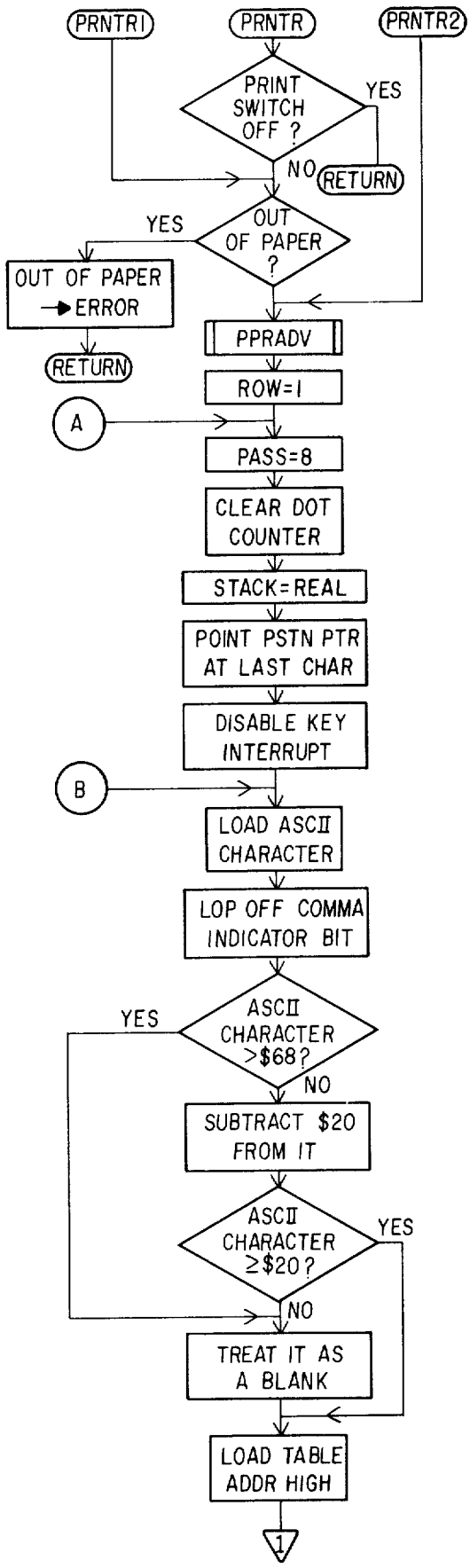
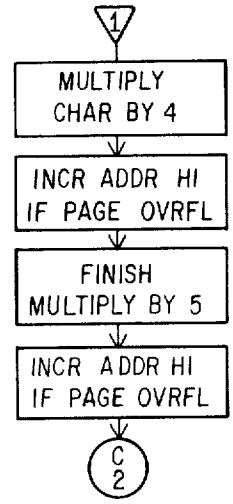


FIG. 65L





PRINT SELECTS

PASS		S4	S3	S2	S1

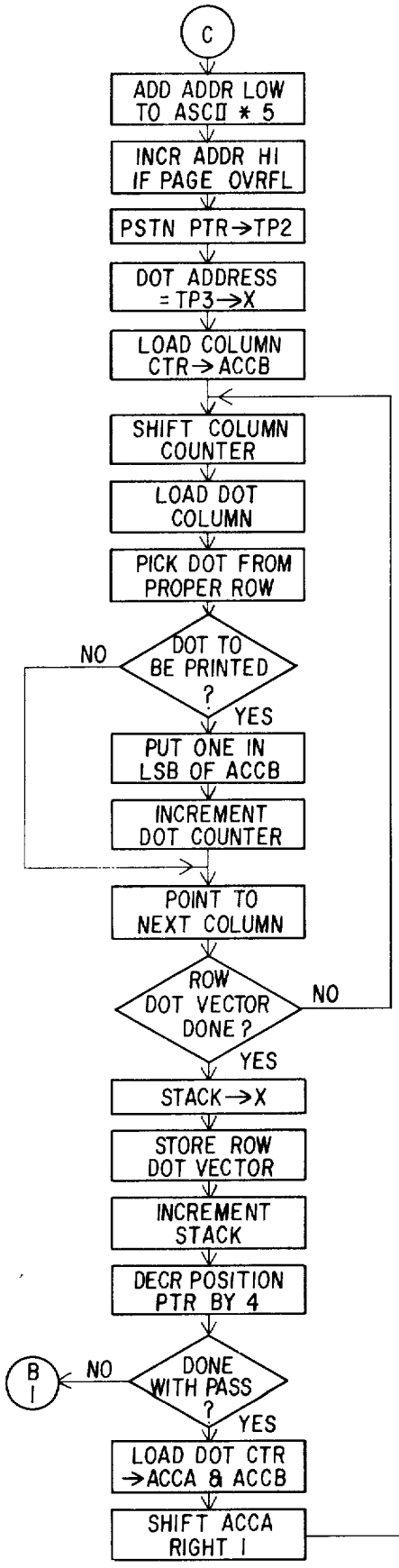
DOT COUNTER COUNTS THE NUMBER OF DOTS TO BE BURNED / PASS.

DOT MATRICES ARE STORED BY COLUMN:

	ROW 7 6 5 4 3 2 1
$\overline{\text{}}$$5EC0	COLUMN 5
	4
SPACE	3
	2
	1
—	5
	4
⇌	3
	2
—	1
	° ° ° °

ADDRESS OF DOT COLUMNS FOR PARTICULAR ASCII CHARACTER = $5EC0 + [(ASCII − $20) * 5].
IF DURING THE CALCULATION OF $C0 + (ASCII − $20) * 5, THE RESULT CROSSES A PAGE BOUNDARY, THE TABLE ADDRESS HIGH MUST BE INCREMENTED.

FIG 66A

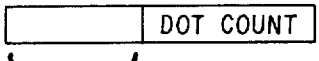


DOT ROW TO BE BUILT: $ØE = ØIIIØ

I	I	I	I	I	Ø	Ø	Ø
I	I	I	I	Ø	Ø	Ø	Ø
I	I	I	Ø	Ø	Ø	Ø	I
I	I	Ø	Ø	Ø	Ø	I	I
I	Ø	Ø	Ø	Ø	I	I	I
Ø	Ø	Ø	Ø	I	I	I	Ø

INDICATES DOT ROW IS COMPLETED

DOT COUNTER CONTAINS NOT ONLY DOT COUNT BUT, ALSO THE DOT OVERFLOW INFORMATION IF THERE IS MORE THAN IO DOTS IN A PASS



DOT COUNT OVERFLOW BITS

BIT 5 PASS #2 DOT OVERFLOW
BIT 6 PASS #3 DOT OVERFLOW
BIT 7 PASS #4 DOT OVERFLOW

BIT 4 WILL BE PASS #1 OVERFLOW BIT WHEN LAST DOT COUNT HAS BEEN EVALUATED.

FIG 66B

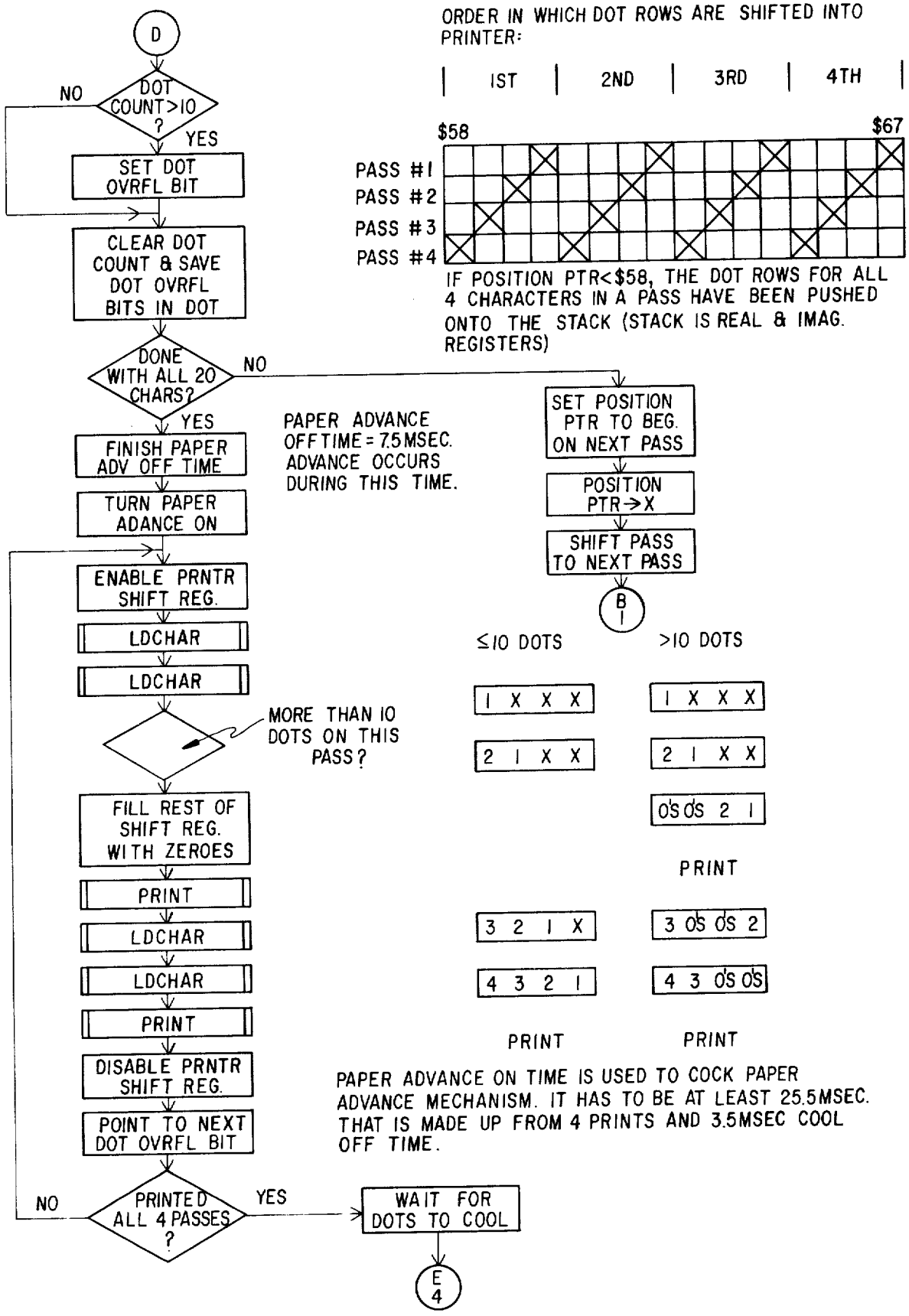

D
DOT COUNT>10 ?
NO
YES
SET DOT OVRFL BIT
CLEAR DOT COUNT & SAVE DOT OVRFL BITS IN DOT
DONE WITH ALL 20 CHARS?
NO
YES
FINISH PAPER ADV OFF TIME
TURN PAPER ADANCE ON
ENABLE PRNTR SHIFT REG.
LDCHAR
LDCHAR
MORE THAN 10 DOTS ON THIS PASS?
FILL REST OF SHIFT REG. WITH ZEROES
PRINT
LDCHAR
LDCHAR
PRINT
DISABLE PRNTR SHIFT REG.
POINT TO NEXT DOT OVRFL BIT
PRINTED ALL 4 PASSES ?
NO
YES
WAIT FOR DOTS TO COOL
E 4
PAPER ADVANCE OFF TIME = 7.5 MSEC. ADVANCE OCCURS DURING THIS TIME.
SET POSITION PTR TO BEG. ON NEXT PASS
POSITION PTR→X
SHIFT PASS TO NEXT PASS
B 1
ORDER IN WHICH DOT ROWS ARE SHIFTED INTO PRINTER:
1ST
2ND
3RD
4TH
$58
$67
PASS #1
PASS #2
PASS #3
PASS #4
IF POSITION PTR<$58, THE DOT ROWS FOR ALL 4 CHARACTERS IN A PASS HAVE BEEN PUSHED ONTO THE STACK (STACK IS REAL & IMAG. REGISTERS)
≤10 DOTS
>10 DOTS
1 X X X
2 1 X X
1 X X X
2 1 X X
0'S 0'S 2 1
PRINT
3 2 1 X
4 3 2 1
3 0'S 0'S 2
4 3 0'S 0'S
PRINT
PRINT
PAPER ADVANCE ON TIME IS USED TO COCK PAPER ADVANCE MECHANISM. IT HAS TO BE AT LEAST 25.5 MSEC. THAT IS MADE UP FROM 4 PRINTS AND 3.5 MSEC COOL OFF TIME.

FIG 66C

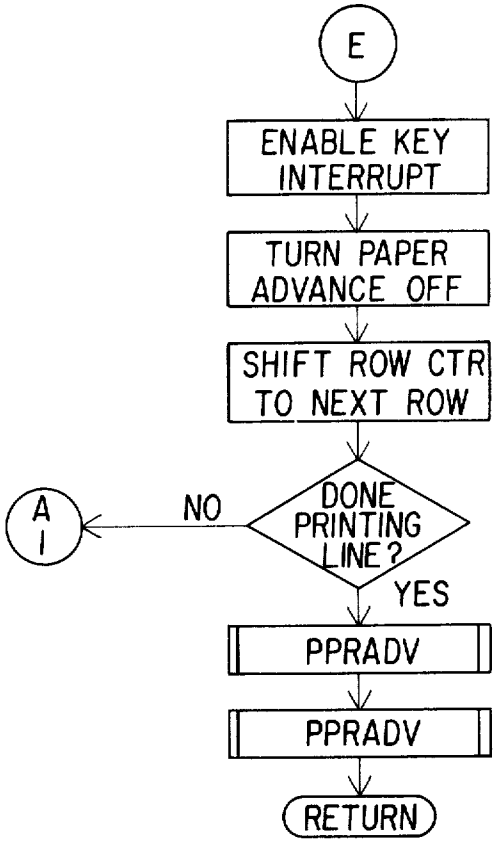


FIG 66D

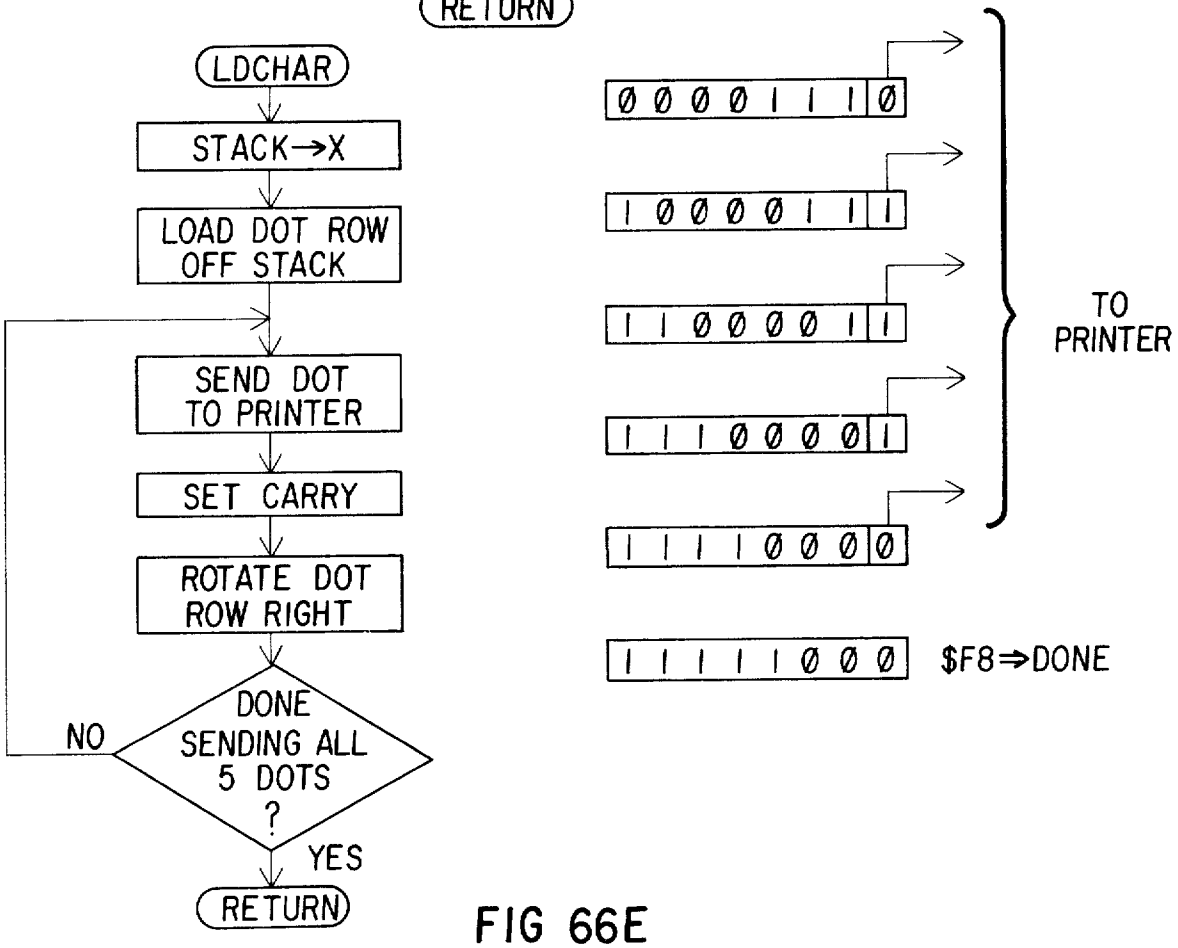


FIG 66E

PAPER ADVANCE MECHANISM IS OFF ON ENTRY TO THIS SUBROUTINE.

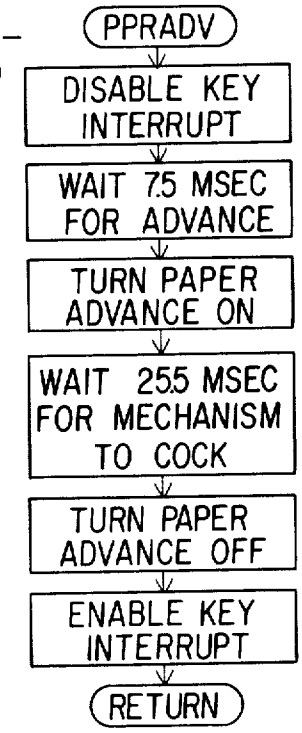
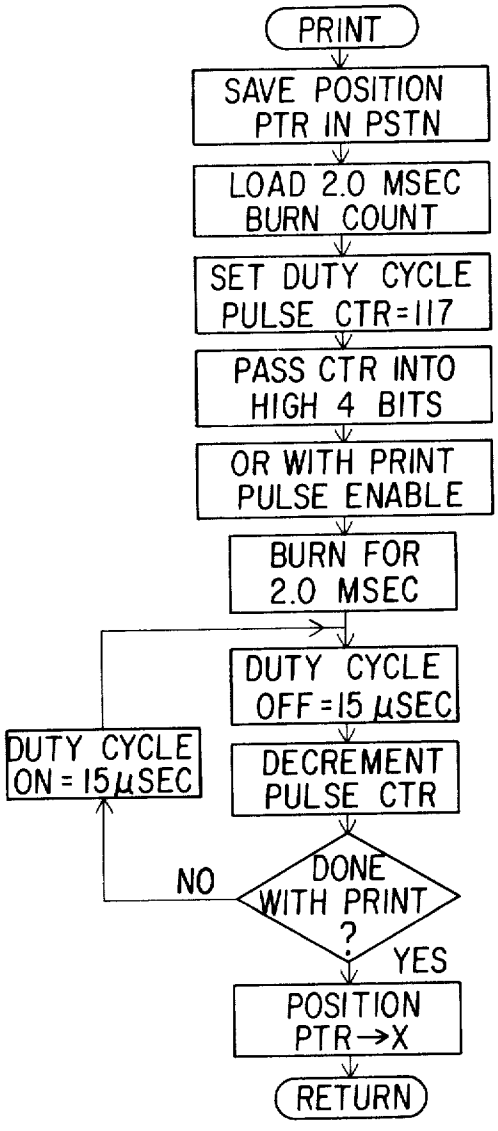
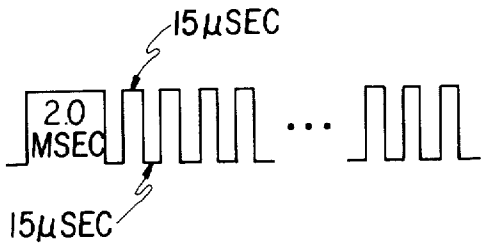


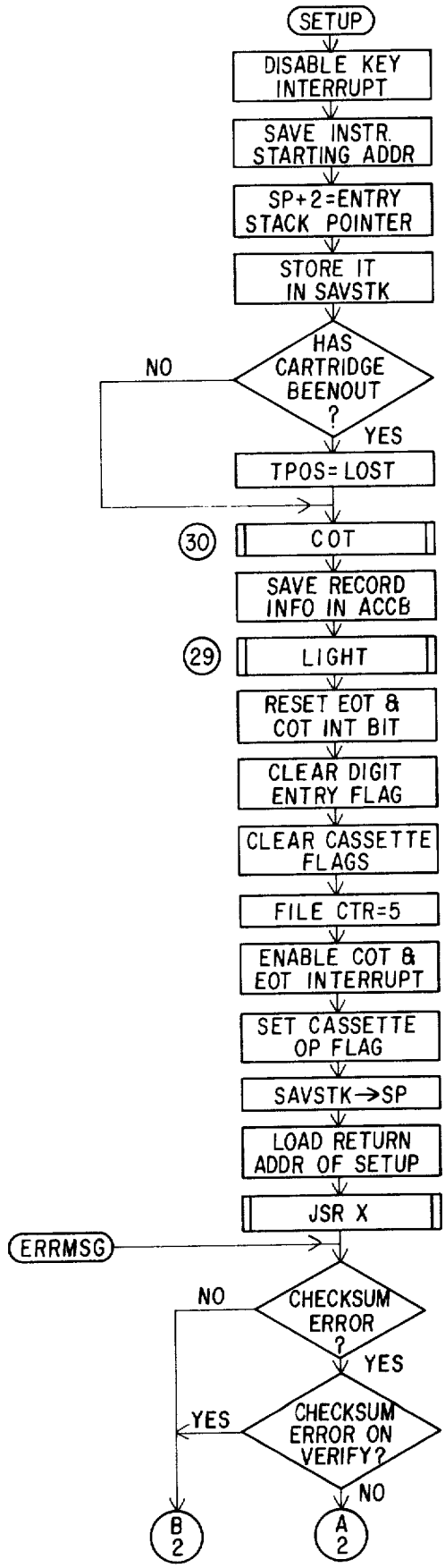
FIG 66F





TOTAL PRINT PULSE = 5.5μSEC
117 PULSES ARE NEEDED TO GET 3.5 MSEC
[5.5-2.0=3.5MSEC]

FIG 66G



SP+2=ENTRY STACK POINTER BECAUSE JSR SETUP ADDED ONE SUBROUTINE RETURN VECTOR ONTO STACK.

CHECK IF CARTRIDGE IS STILL OUT.

DETERMINE IF TAPE IS SPOOLED OFF THE HUBS.

EOT = END OF TAPE
COT = CARTRIDGE OUT

THIS COUNTER COUNTS THE NO. OF TIMES A FILE IS SEARCHED FOR ON THE SAME INSTRUCTION. IF THE FILE IS NOT FOUND IN 5 TRIES, THE FILE CANNOT BE FOUND AND AN ERROR IS RETURNED.

JUMP SUBROUTINE INDEXED BACK TO MAIN ROUTINE SO THAT ERRORS CAN JUMP TO ERRMSG USING A ONE-BYTE RTS INSTEAD OF A 3-BYTE JMP.

A CHECKSUM ERROR ON A VERIFY IS REALLY A VERIFY FAILED ON THE CHECKSUM WORD.

FIG 67A

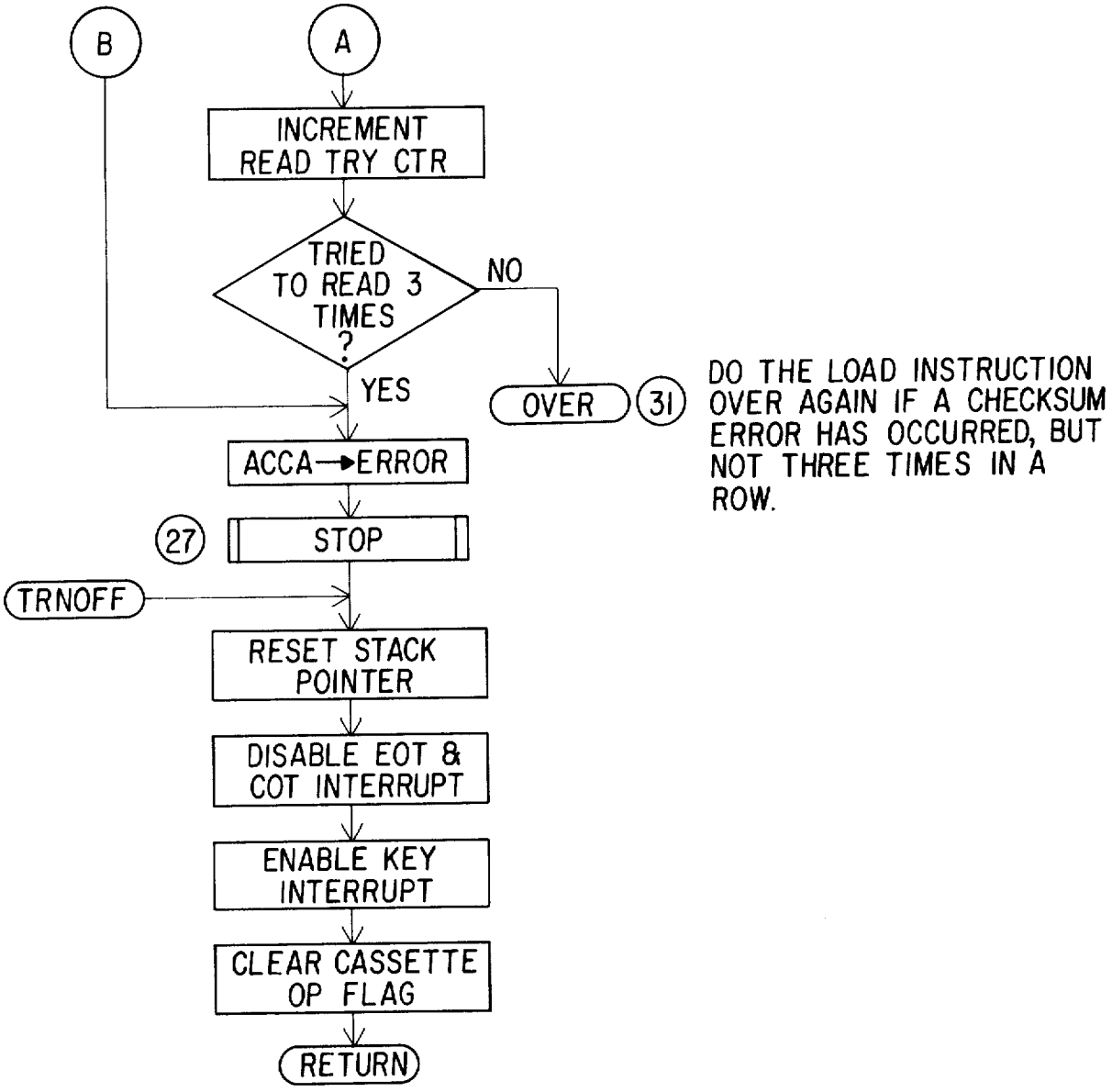

B
A
INCREMENT READ TRY CTR
TRIED TO READ 3 TIMES ?
NO
YES
OVER
31
DO THE LOAD INSTRUCTION OVER AGAIN IF A CHECKSUM ERROR HAS OCCURRED, BUT NOT THREE TIMES IN A ROW.
ACCA→ERROR
27
STOP
TRNOFF
RESET STACK POINTER
DISABLE EOT & COT INTERRUPT
ENABLE KEY INTERRUPT
CLEAR CASSETTE OP FLAG
RETURN

FIG 67B

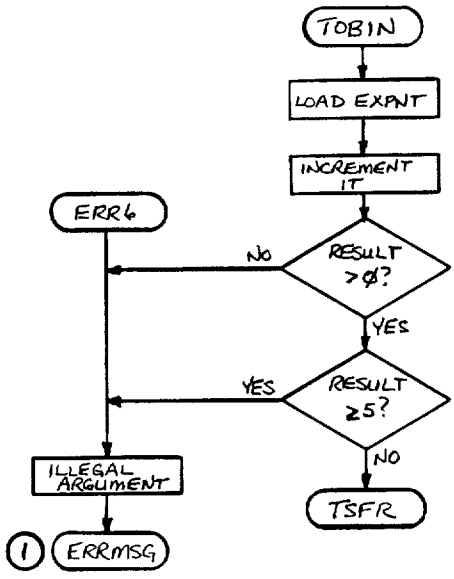

TOBIN
LOAD EXPNT
INCREMENT IT
ERR6
NO
RESULT >0?
YES
YES
RESULT ≥5?
NO
ILLEGAL ARGUMENT
TSFR
1
ERRMSG
CONVERTS NO, IN REGISTER POINTED TO BY INDEX TO BINARY

FIG. 67C

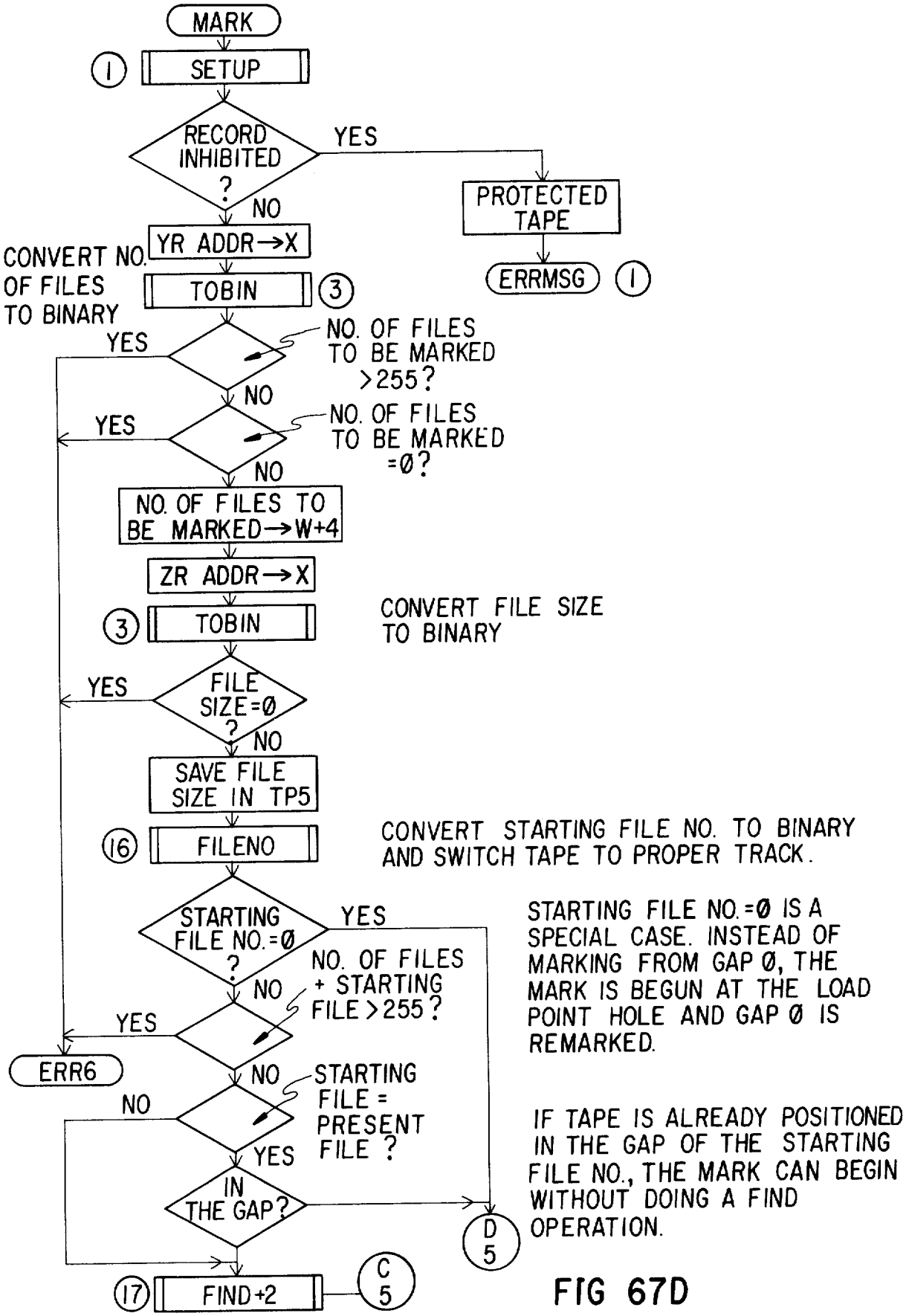

MARK
1
SETUP
RECORD INHIBITED ?
YES
PROTECTED TAPE
ERRMSG
1
NO
YR ADDR→X
CONVERT NO. OF FILES TO BINARY
TOBIN
3
NO. OF FILES TO BE MARKED >255?
YES
NO
NO. OF FILES TO BE MARKED =Ø?
YES
NO
NO. OF FILES TO BE MARKED→W+4
ZR ADDR→X
3
TOBIN
CONVERT FILE SIZE TO BINARY
FILE SIZE=Ø ?
YES
NO
SAVE FILE SIZE IN TP5
16
FILENO
CONVERT STARTING FILE NO. TO BINARY AND SWITCH TAPE TO PROPER TRACK.
STARTING FILE NO.=Ø ?
YES
NO
NO. OF FILES + STARTING FILE >255?
YES
ERR6
NO
STARTING FILE = PRESENT FILE ?
NO
YES
IN THE GAP?
STARTING FILE NO.=Ø IS A SPECIAL CASE. INSTEAD OF MARKING FROM GAP Ø, THE MARK IS BEGUN AT THE LOAD POINT HOLE AND GAP Ø IS REMARKED.
IF TAPE IS ALREADY POSITIONED IN THE GAP OF THE STARTING FILE NO., THE MARK CAN BEGIN WITHOUT DOING A FIND OPERATION.
D 5
17
FIND+2
C 5

FIG 67D

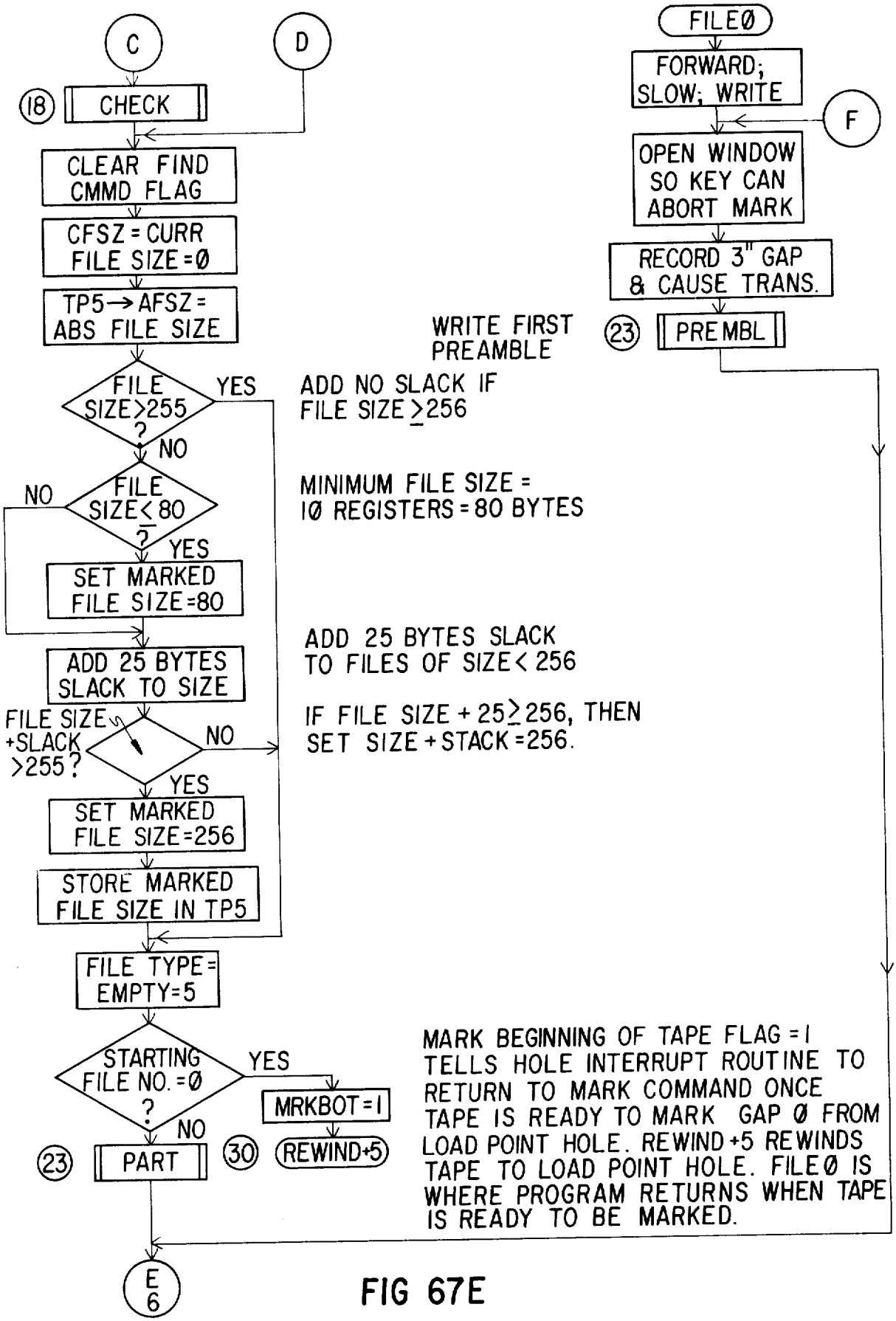

C
D
18
CHECK
CLEAR FIND CMMD FLAG
CFSZ = CURR FILE SIZE = Ø
TP5 → AFSZ = ABS FILE SIZE
FILE SIZE > 255 ?
YES
NO
ADD NO SLACK IF FILE SIZE ≥ 256
FILE SIZE < 80 ?
NO
YES
MINIMUM FILE SIZE = 1Ø REGISTERS = 80 BYTES
SET MARKED FILE SIZE = 80
ADD 25 BYTES SLACK TO SIZE
ADD 25 BYTES SLACK TO FILES OF SIZE < 256
FILE SIZE + SLACK > 255 ?
NO
YES
IF FILE SIZE + 25 ≥ 256, THEN SET SIZE + STACK = 256.
SET MARKED FILE SIZE = 256
STORE MARKED FILE SIZE IN TP5
FILE TYPE = EMPTY = 5
STARTING FILE NO. = Ø ?
YES
NO
MRKBOT = 1
23
PART
30
REWIND+5
E 6
FILEØ
FORWARD; SLOW; WRITE
F
OPEN WINDOW SO KEY CAN ABORT MARK
RECORD 3" GAP & CAUSE TRANS.
WRITE FIRST PREAMBLE
23
PREMBL
MARK BEGINNING OF TAPE FLAG = 1 TELLS HOLE INTERRUPT ROUTINE TO RETURN TO MARK COMMAND ONCE TAPE IS READY TO MARK GAP Ø FROM LOAD POINT HOLE. REWIND+5 REWINDS TAPE TO LOAD POINT HOLE. FILEØ IS WHERE PROGRAM RETURNS WHEN TAPE IS READY TO BE MARKED.

FIG 67E

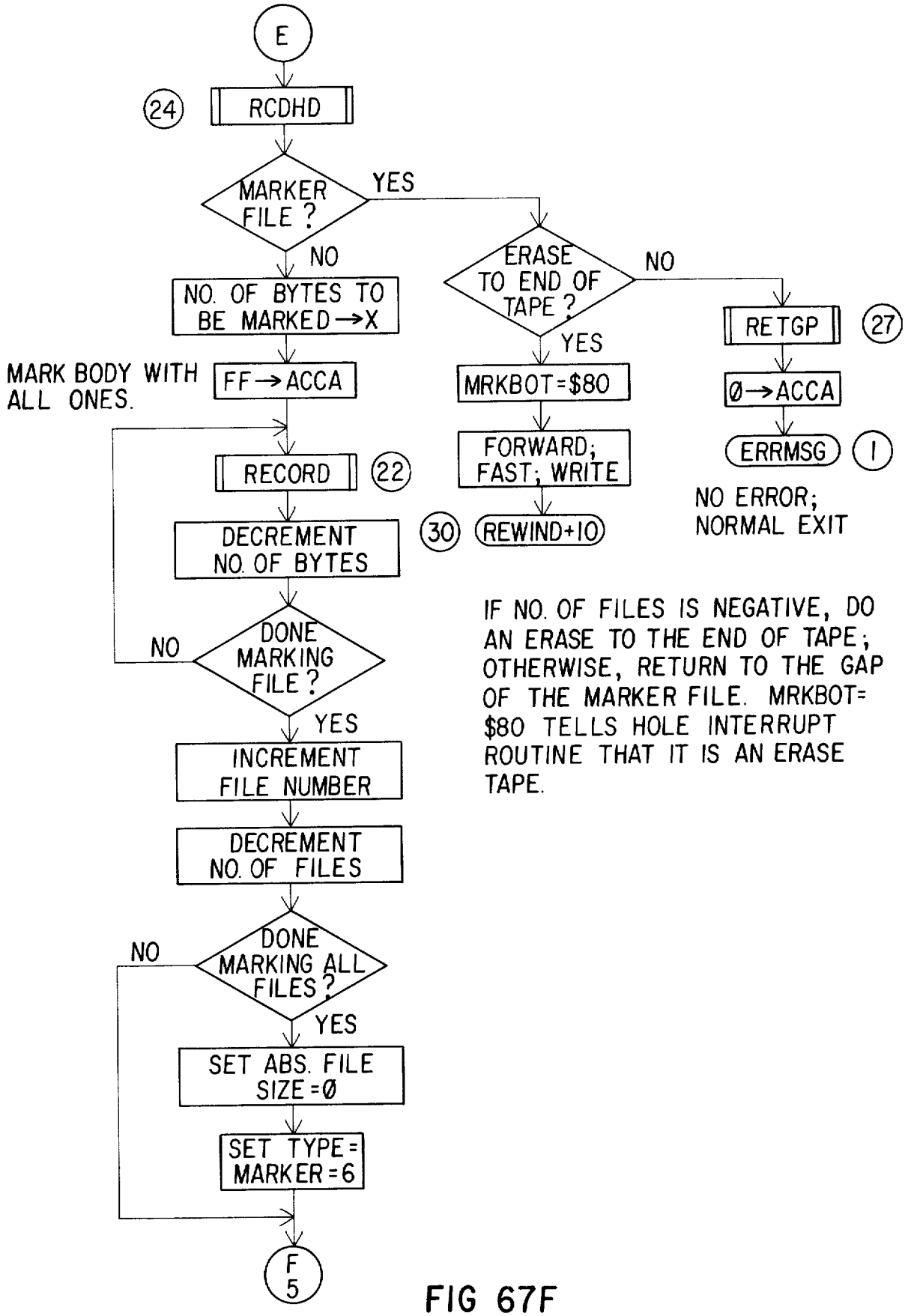

E
24
RCDHD
MARKER FILE ?
YES
NO
NO. OF BYTES TO BE MARKED→X
MARK BODY WITH ALL ONES.
FF→ACCA
RECORD
22
DECREMENT NO. OF BYTES
NO
DONE MARKING FILE ?
YES
INCREMENT FILE NUMBER
DECREMENT NO. OF FILES
NO
DONE MARKING ALL FILES ?
YES
SET ABS. FILE SIZE=Ø
SET TYPE= MARKER=6
F 5
ERASE TO END OF TAPE ?
NO
YES
MRKBOT=$80
FORWARD; FAST; WRITE
30
REWIND+IO
RETGP
27
Ø→ACCA
ERRMSG
1
NO ERROR; NORMAL EXIT
IF NO. OF FILES IS NEGATIVE, DO AN ERASE TO THE END OF TAPE; OTHERWISE, RETURN TO THE GAP OF THE MARKER FILE. MRKBOT= $80 TELLS HOLE INTERRUPT ROUTINE THAT IT IS AN ERASE TAPE.

FIG 67F

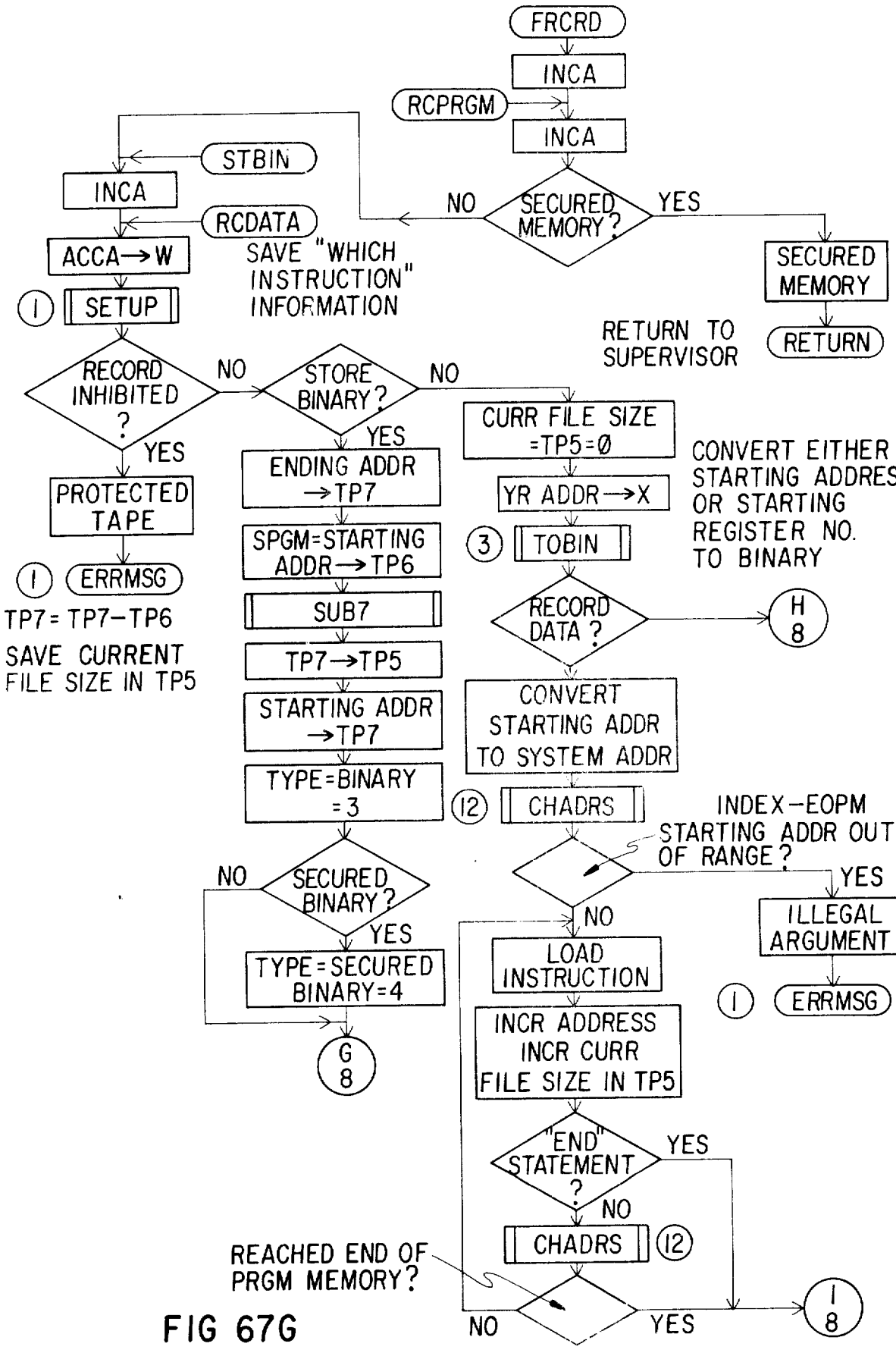

FRCRD
INCA
RCPRGM
INCA
SECURED MEMORY?
NO
YES
SECURED MEMORY
RETURN
RETURN TO SUPERVISOR
STBIN
INCA
RCDATA
SAVE "WHICH INSTRUCTION" INFORMATION
ACCA→W
1
SETUP
RECORD INHIBITED ?
NO
YES
PROTECTED TAPE
1
ERRMSG
TP7 = TP7−TP6
SAVE CURRENT FILE SIZE IN TP5
STORE BINARY?
NO
YES
ENDING ADDR →TP7
SPGM=STARTING ADDR→TP6
SUB7
TP7→TP5
STARTING ADDR →TP7
TYPE=BINARY =3
SECURED BINARY?
NO
YES
TYPE=SECURED BINARY=4
G 8
CURR FILE SIZE =TP5=Ø
YR ADDR→X
3
TOBIN
CONVERT EITHER STARTING ADDRESS OR STARTING REGISTER NO. TO BINARY
RECORD DATA ?
H 8
CONVERT STARTING ADDR TO SYSTEM ADDR
12
CHADRS
INDEX−EOPM STARTING ADDR OUT OF RANGE?
YES
NO
ILLEGAL ARGUMENT
1
ERRMSG
LOAD INSTRUCTION
INCR ADDRESS INCR CURR FILE SIZE IN TP5
"END" STATEMENT ?
YES
NO
CHADRS
12
REACHED END OF PRGM MEMORY?
NO
YES
1
8

FIG 67G

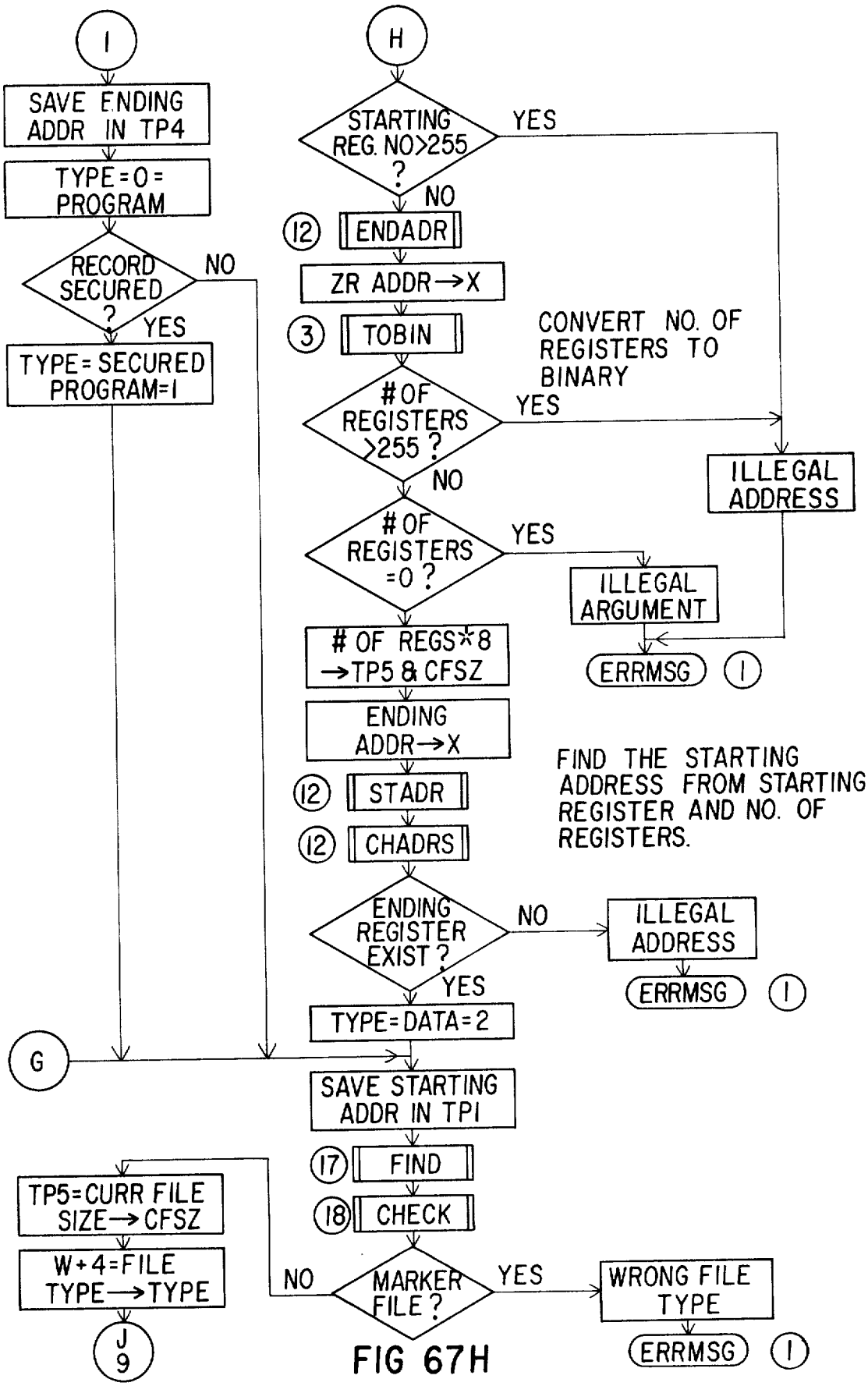

I
SAVE ENDING ADDR IN TP4
TYPE=0= PROGRAM
RECORD SECURED ?
NO
YES
TYPE= SECURED PROGRAM=1
G
H
STARTING REG. NO>255 ?
YES
NO
12 ENDADR
ZR ADDR→X
3 TOBIN
CONVERT NO. OF REGISTERS TO BINARY
OF REGISTERS >255 ?
YES
NO
ILLEGAL ADDRESS
OF REGISTERS =0 ?
YES
ILLEGAL ARGUMENT
ERRMSG I
OF REGS*8 →TP5 & CFSZ
ENDING ADDR→X
FIND THE STARTING ADDRESS FROM STARTING REGISTER AND NO. OF REGISTERS.
12 STADR
12 CHADRS
ENDING REGISTER EXIST ?
NO
ILLEGAL ADDRESS
ERRMSG I
YES
TYPE=DATA=2
SAVE STARTING ADDR IN TPI
17 FIND
18 CHECK
MARKER FILE ?
NO
YES
WRONG FILE TYPE
ERRMSG I
TP5=CURR FILE SIZE→CFSZ
W+4=FILE TYPE→TYPE
J 9

FIG 67H

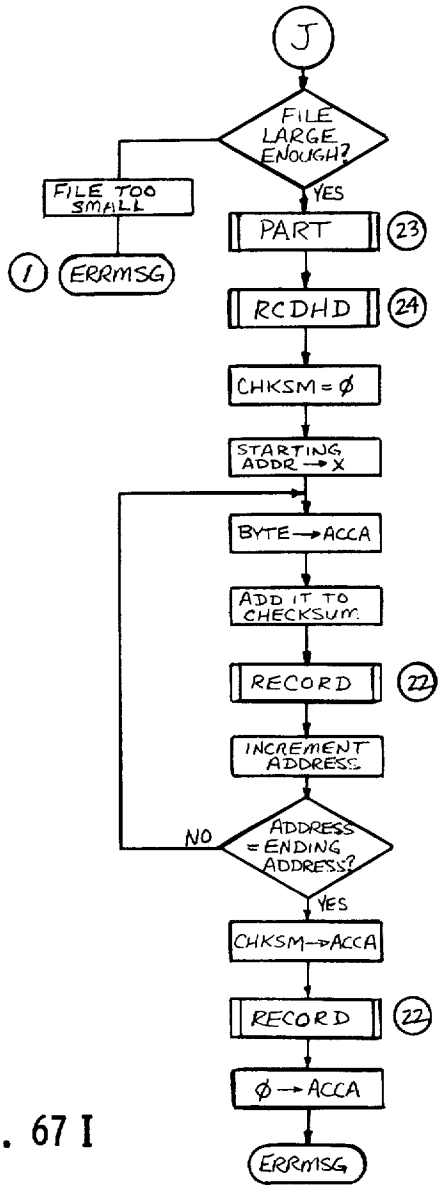

J
FILE LARGE ENOUGH?
YES
FILE TOO SMALL
1
ERRMSG
PART
23
RECORD FIRST PREAMBLE
RCDHD
24
RECORD HEADING AND 2ND PREAMBLE
CHKSM = ∅
STARTING ADDR → X
BYTE → ACCA
ADD IT TO CHECKSUM
RECORD
22
RECORD BODY ONE BYTE AT A TIME
INCREMENT ADDRESS
NO
ADDRESS = ENDING ADDRESS?
YES
CHKSM → ACCA
RECORD
22
RECORD CHECKSUM
∅ → ACCA
ERRMSG
NO ERROR; NORMAL EXIT

FIG. 67 I

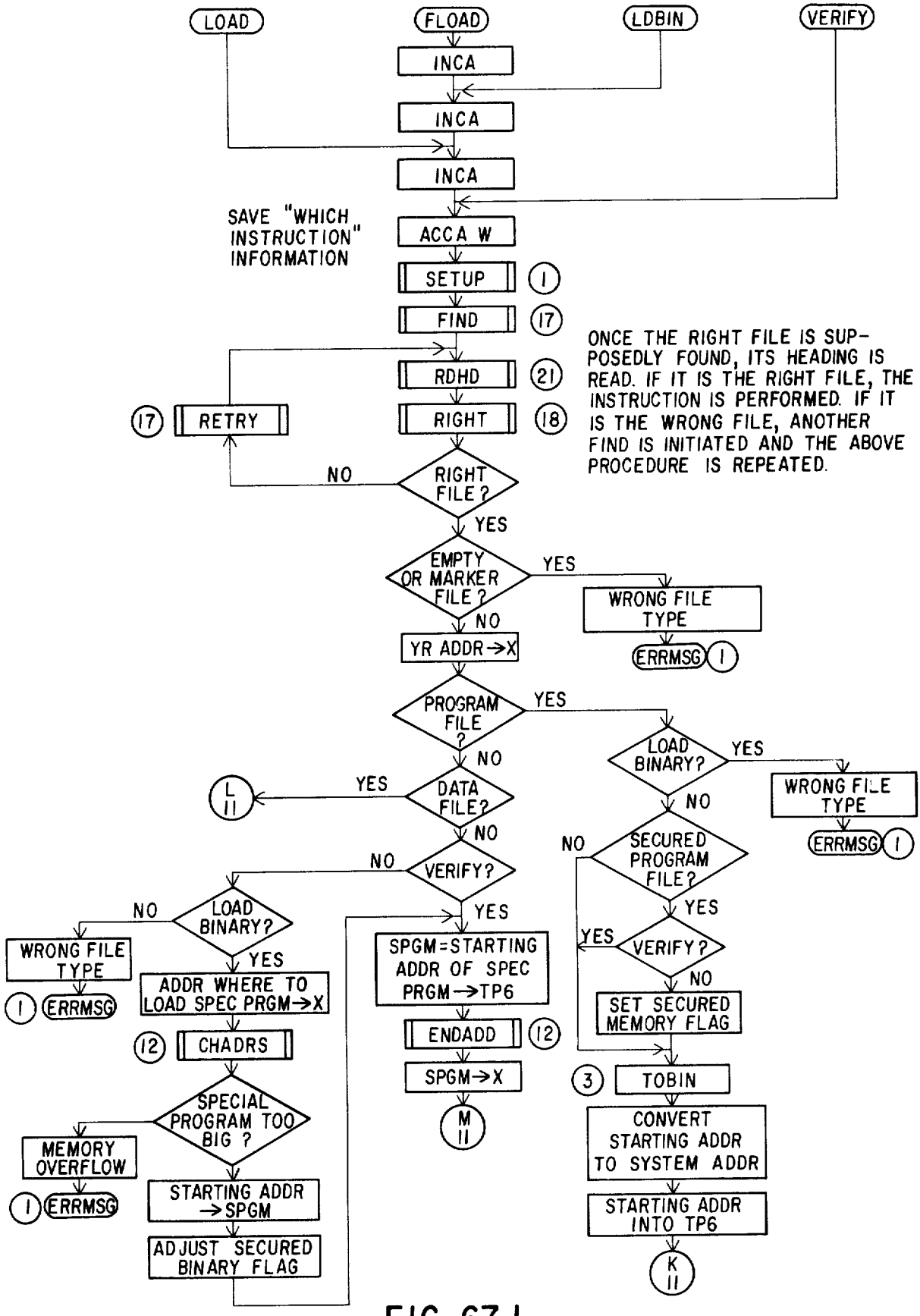

LOAD
FLOAD
LDBIN
VERIFY
INCA
INCA
INCA
SAVE "WHICH INSTRUCTION" INFORMATION
ACCA W
SETUP
1
FIND
17
RDHD
21
RIGHT
18
17
RETRY
NO
RIGHT FILE?
YES
ONCE THE RIGHT FILE IS SUPPOSEDLY FOUND, ITS HEADING IS READ. IF IT IS THE RIGHT FILE, THE INSTRUCTION IS PERFORMED. IF IT IS THE WRONG FILE, ANOTHER FIND IS INITIATED AND THE ABOVE PROCEDURE IS REPEATED.
EMPTY OR MARKER FILE?
YES
WRONG FILE TYPE
ERRMSG
1
NO
YR ADDR→X
PROGRAM FILE?
YES
NO
L
11
YES
DATA FILE?
NO
NO
VERIFY?
LOAD BINARY?
YES
NO
WRONG FILE TYPE
1
ERRMSG
ADDR WHERE TO LOAD SPEC PRGM→X
12
CHADRS
SPECIAL PROGRAM TOO BIG ?
MEMORY OVERFLOW
1
ERRMSG
STARTING ADDR →SPGM
ADJUST SECURED BINARY FLAG
YES
SPGM=STARTING ADDR OF SPEC PRGM→TP6
ENDADD
12
SPGM→X
M
11
LOAD BINARY?
YES
WRONG FILE TYPE
ERRMSG
1
NO
NO
SECURED PROGRAM FILE?
YES
YES
VERIFY?
NO
SET SECURED MEMORY FLAG
3
TOBIN
CONVERT STARTING ADDR TO SYSTEM ADDR
STARTING ADDR INTO TP6
K
11

FIG 67J

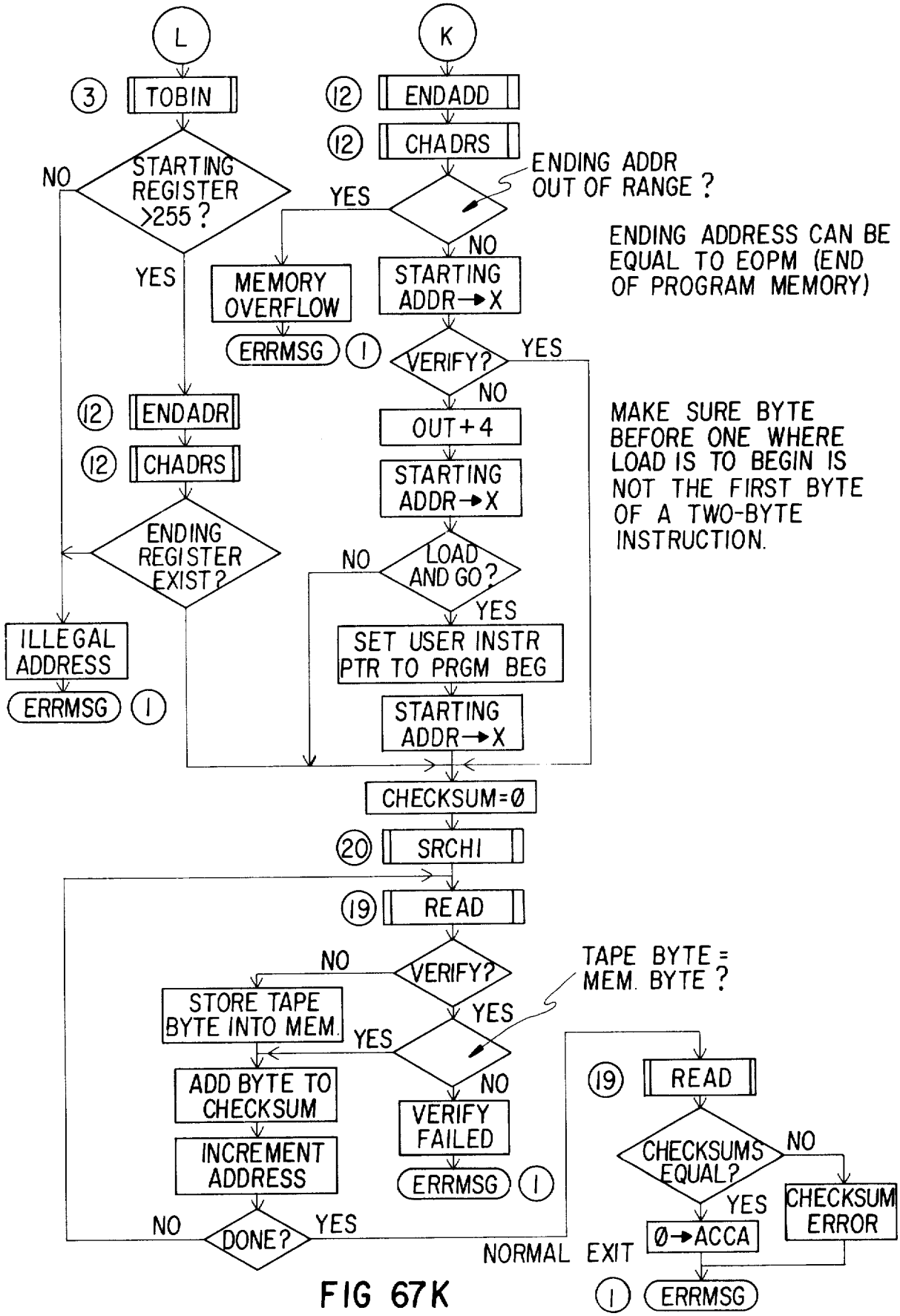

L
3
TOBIN
STARTING REGISTER >255 ?
NO
YES
MEMORY OVERFLOW
ERRMSG
1
12
ENDADR
12
CHADRS
ENDING REGISTER EXIST ?
ILLEGAL ADDRESS
ERRMSG
1
K
12
ENDADD
12
CHADRS
ENDING ADDR OUT OF RANGE ?
YES
NO
STARTING ADDR→X
VERIFY?
YES
NO
OUT+4
STARTING ADDR→X
LOAD AND GO ?
NO
YES
SET USER INSTR PTR TO PRGM BEG
STARTING ADDR→X
ENDING ADDRESS CAN BE EQUAL TO EOPM (END OF PROGRAM MEMORY)
MAKE SURE BYTE BEFORE ONE WHERE LOAD IS TO BEGIN IS NOT THE FIRST BYTE OF A TWO-BYTE INSTRUCTION.
CHECKSUM=Ø
20
SRCHI
19
READ
VERIFY?
NO
YES
TAPE BYTE = MEM. BYTE ?
STORE TAPE BYTE INTO MEM.
YES
NO
ADD BYTE TO CHECKSUM
INCREMENT ADDRESS
VERIFY FAILED
ERRMSG
1
NO
DONE?
YES
19
READ
CHECKSUMS EQUAL?
NO
YES
CHECKSUM ERROR
Ø→ACCA
NORMAL EXIT
1
ERRMSG

FIG 67K

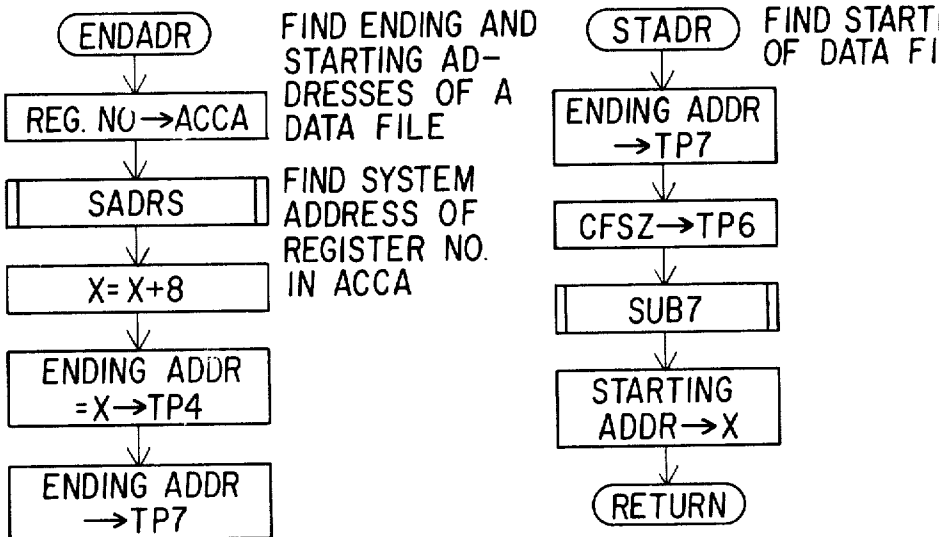


FIG 67M

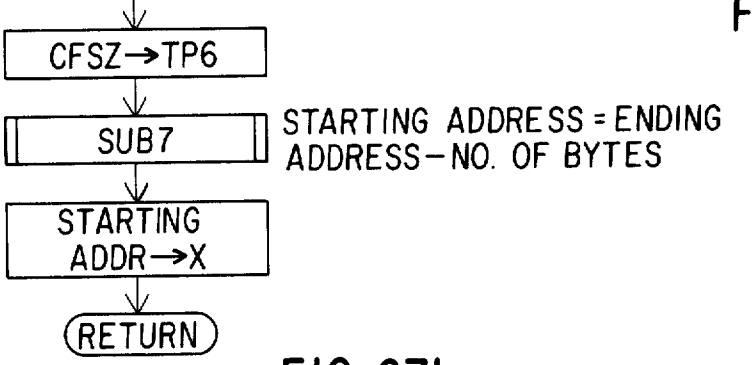


FIG 67L

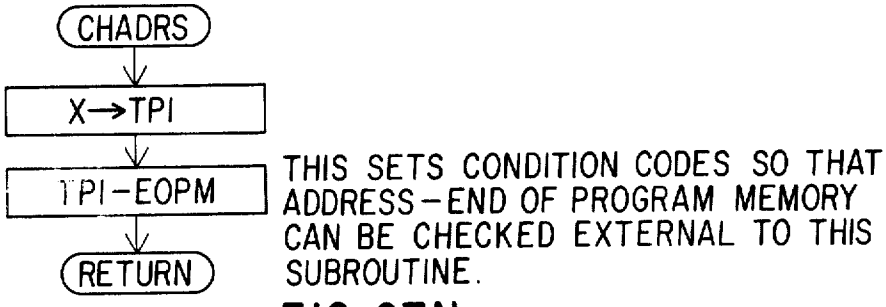


FIG 67N

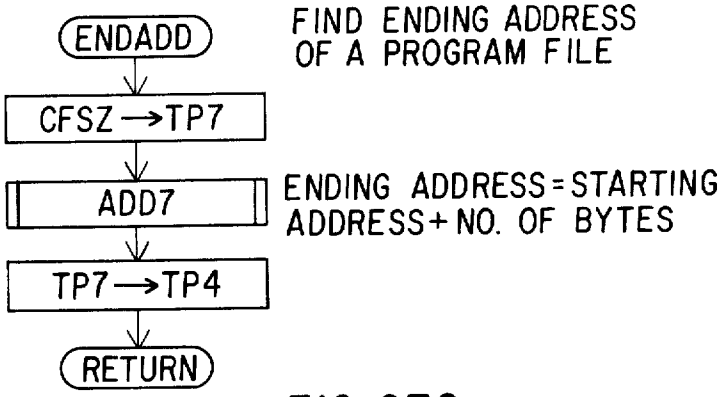


FIG 67O

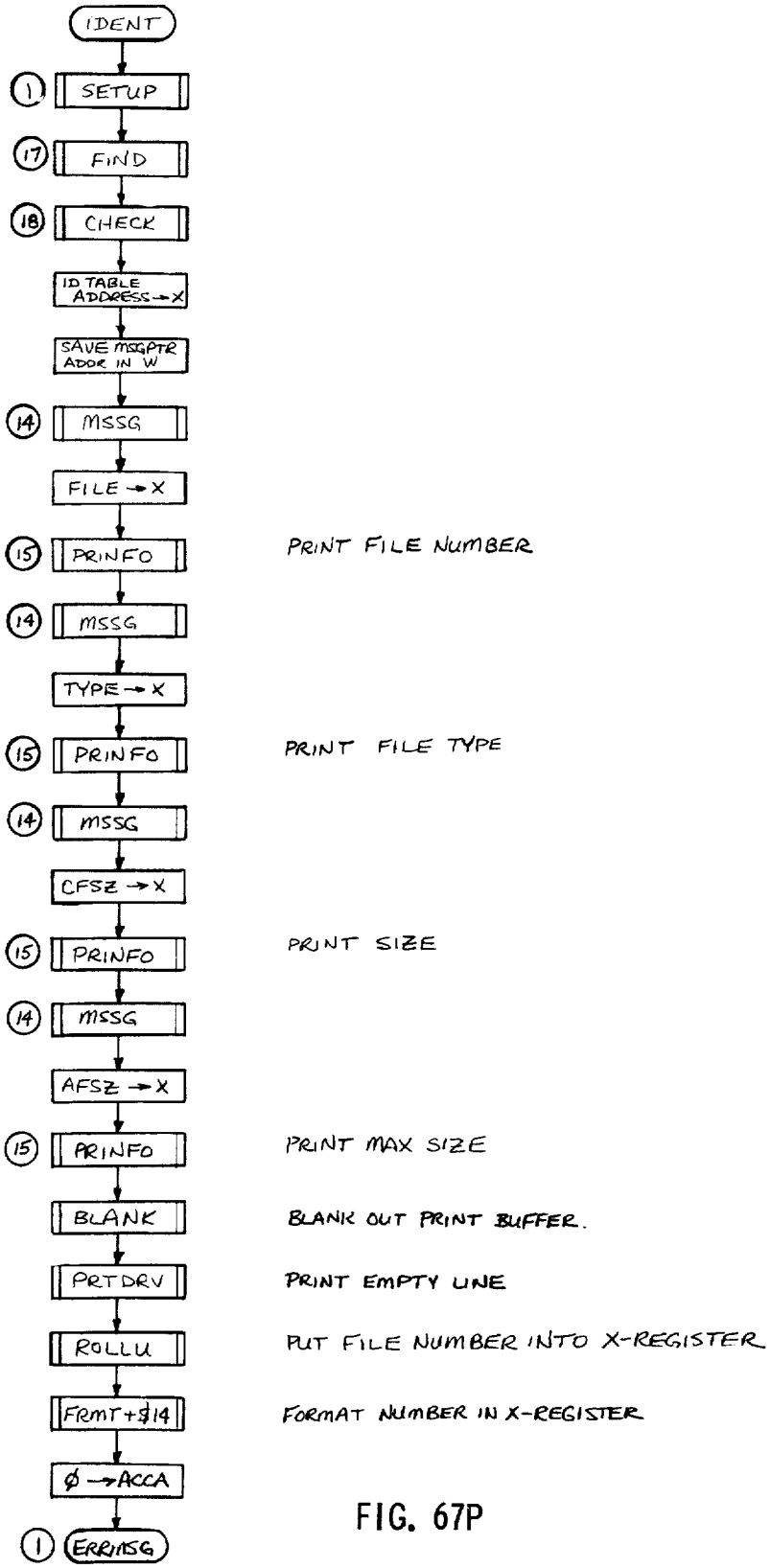

IDENT
1 SETUP
17 FIND
18 CHECK
ID TABLE ADDRESS→X
SAVE MSGPTR ADDR IN W
14 MSSG
FILE→X
15 PRINFO
PRINT FILE NUMBER
14 MSSG
TYPE→X
15 PRINFO
PRINT FILE TYPE
14 MSSG
CFSZ→X
15 PRINFO
PRINT SIZE
14 MSSG
AFSZ→X
15 PRINFO
PRINT MAX SIZE
BLANK
BLANK OUT PRINT BUFFER.
PRTDRV
PRINT EMPTY LINE
ROLLU
PUT FILE NUMBER INTO X-REGISTER
FRMT+$14
FORMAT NUMBER IN X-REGISTER
Ø→ACCA
1 ERRMSG

FIG. 67P

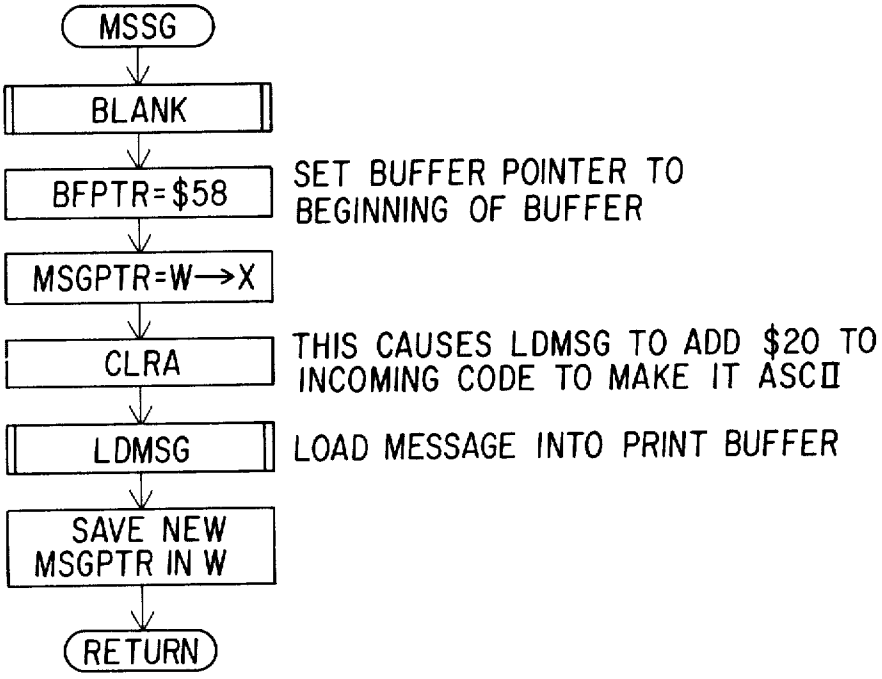

MSSG
BLANK
BFPTR=$58
SET BUFFER POINTER TO BEGINNING OF BUFFER
MSGPTR=W→X
CLRA
THIS CAUSES LDMSG TO ADD $20 TO INCOMING CODE TO MAKE IT ASCII
LDMSG
LOAD MESSAGE INTO PRINT BUFFER
SAVE NEW MSGPTR IN W
RETURN

FIG 67Q

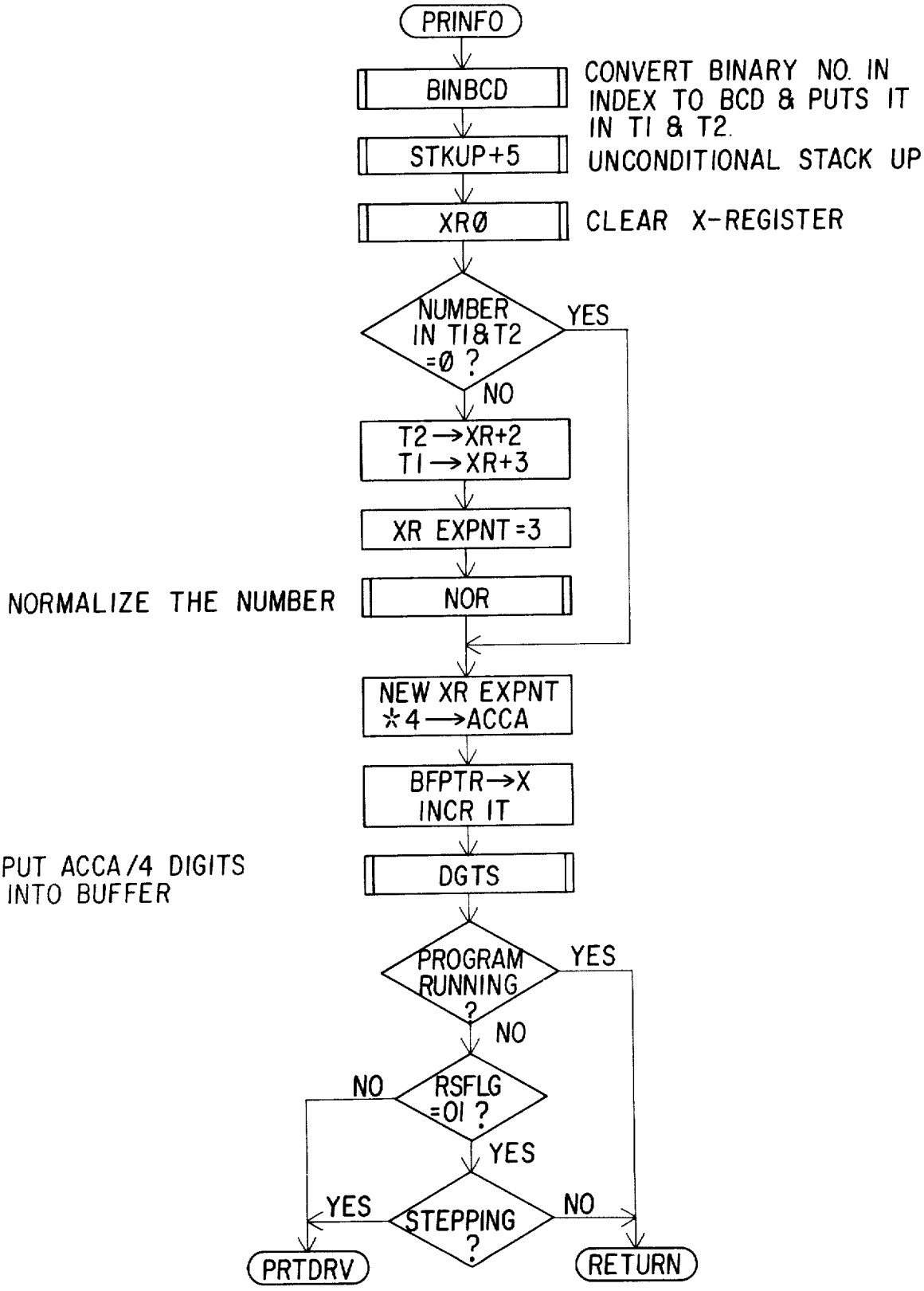

PRINFO
BINBCD
CONVERT BINARY NO. IN INDEX TO BCD & PUTS IT IN T1 & T2.
STKUP+5
UNCONDITIONAL STACK UP
XRØ
CLEAR X-REGISTER
NUMBER IN T1&T2 =Ø ?
YES
NO
T2→XR+2
T1→XR+3
XR EXPNT=3
NORMALIZE THE NUMBER
NOR
NEW XR EXPNT *4→ACCA
BFPTR→X INCR IT
PUT ACCA/4 DIGITS INTO BUFFER
DGTS
PROGRAM RUNNING ?
YES
NO
NO
RSFLG =01 ?
YES
YES
STEPPING ?
NO
PRTDRV
RETURN

FIG 67R

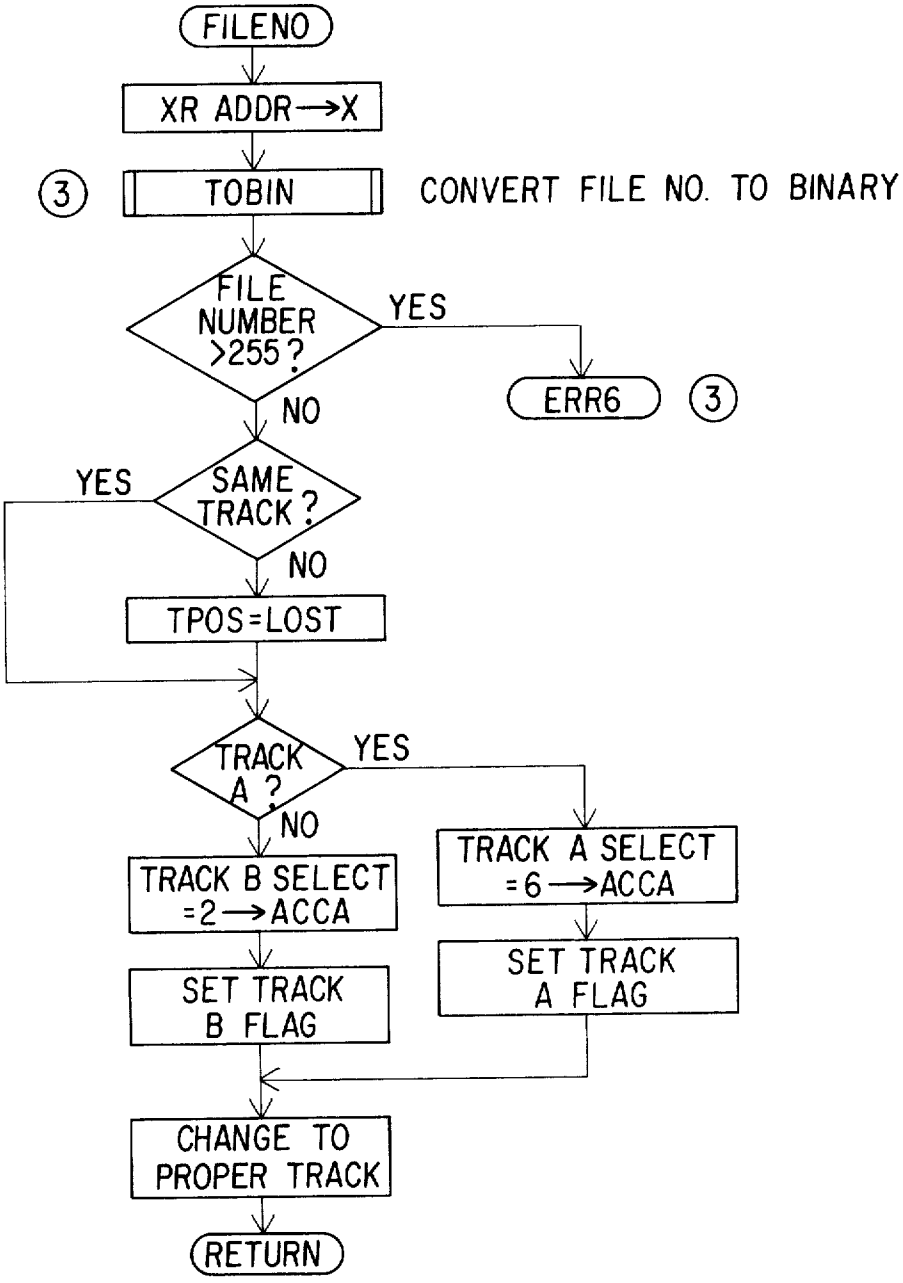
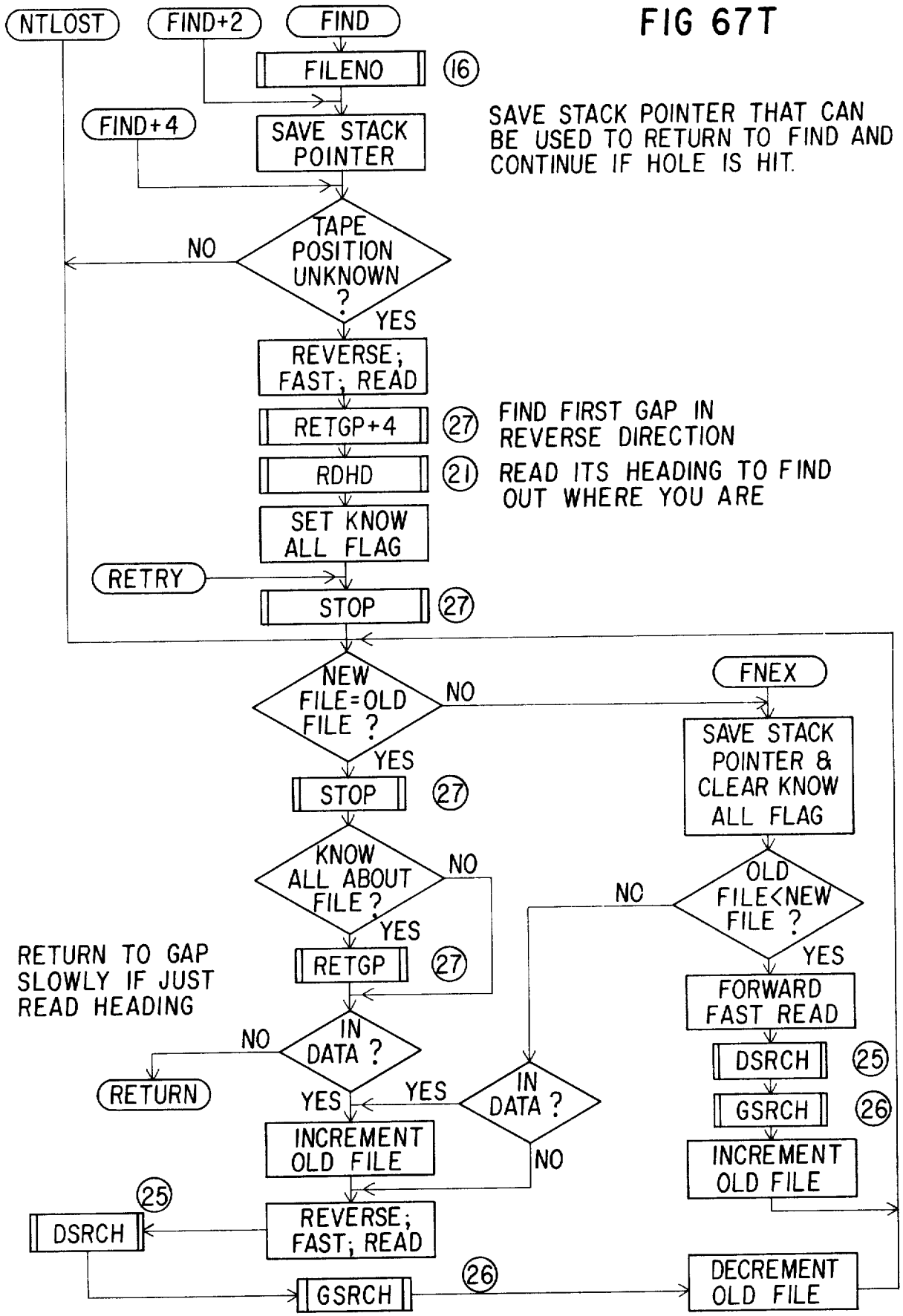

FILENO
XR ADDR→X
3
TOBIN
CONVERT FILE NO. TO BINARY
FILE NUMBER >255?
YES
ERR6
3
NO
YES
SAME TRACK?
NO
TPOS=LOST
TRACK A?
YES
NO
TRACK B SELECT =2→ACCA
SET TRACK B FLAG
TRACK A SELECT =6→ACCA
SET TRACK A FLAG
CHANGE TO PROPER TRACK
RETURN

FIG 67S

FIG 67T


NTLOST
FIND+2
FIND
FILENO
16
FIND+4
SAVE STACK POINTER
SAVE STACK POINTER THAT CAN BE USED TO RETURN TO FIND AND CONTINUE IF HOLE IS HIT.
TAPE POSITION UNKNOWN ?
NO
YES
REVERSE; FAST; READ
RETGP+4
27
FIND FIRST GAP IN REVERSE DIRECTION
RDHD
21
READ ITS HEADING TO FIND OUT WHERE YOU ARE
SET KNOW ALL FLAG
RETRY
STOP
27
NEW FILE=OLD FILE ?
NO
YES
FNEX
SAVE STACK POINTER & CLEAR KNOW ALL FLAG
STOP
27
KNOW ALL ABOUT FILE ?
NO
YES
OLD FILE<NEW FILE ?
NO
YES
RETURN TO GAP SLOWLY IF JUST READ HEADING
RETGP
27
FORWARD FAST READ
IN DATA ?
NO
YES
DSRCH
25
GSRCH
26
RETURN
YES
YES
IN DATA ?
INCREMENT OLD FILE
NO
INCREMENT OLD FILE
25
DSRCH
REVERSE; FAST; READ
26
GSRCH
DECREMENT OLD FILE

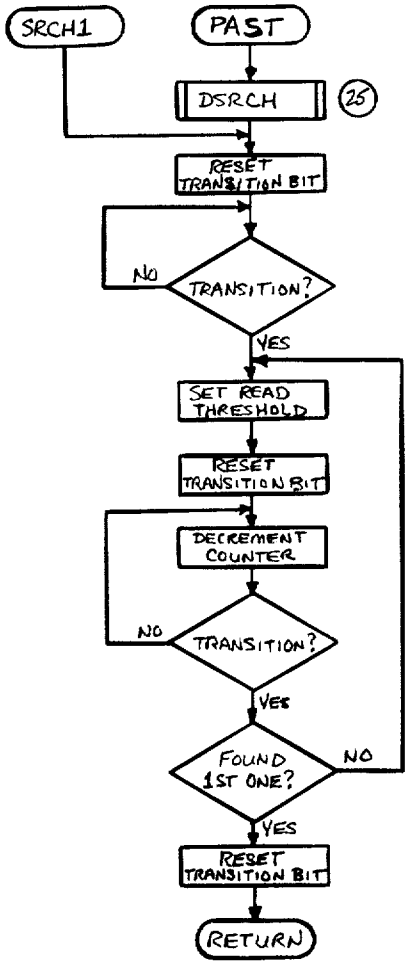

SRCH1
PAST
DSRCH
25
RESET TRANSITION BIT
NO
TRANSITION?
YES
SET READ THRESHOLD
RESET TRANSITION BIT
DECREMENT COUNTER
NO
TRANSITION?
YES
FOUND 1ST ONE?
NO
YES
RESET TRANSITION BIT
RETURN

FIG. 67X

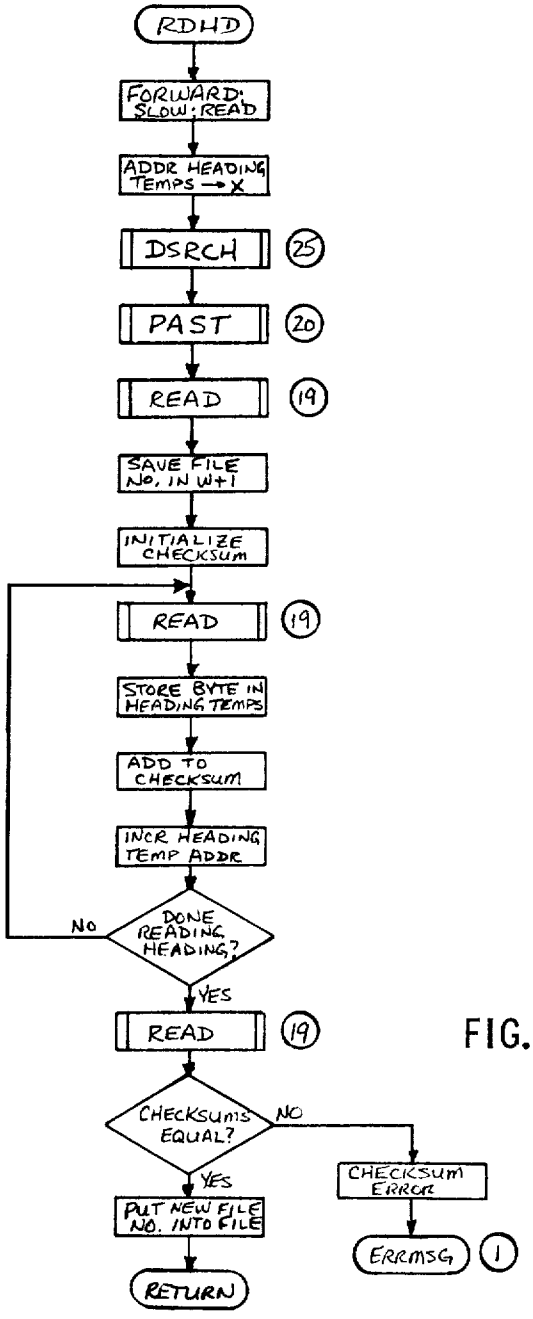

RDHD
FORWARD; SLOW; READ
ADDR HEADING TEMPS → X
DSRCH
25
PAST
20
READ
19
SAVE FILE NO. IN W+1
INITIALIZE CHECKSUM
READ
19
STORE BYTE IN HEADING TEMPS
ADD TO CHECKSUM
INCR HEADING TEMP ADDR
NO
DONE READING HEADING?
YES
READ
19
CHECKSUMS EQUAL?
NO
YES
CHECKSUM ERROR
ERRMSG
1
PUT NEW FILE NO. INTO FILE
RETURN

FIG. 67Y

THIS ENTRY POINT IS USED BY PREMBL SO THAT THE 9 TH BIT WHICH IS LONG ENOUGH TO BE MISTAKEN FOR A ONE, IS NOT MARKED IN THE MIDDLE OF THE 15 ZEROES.

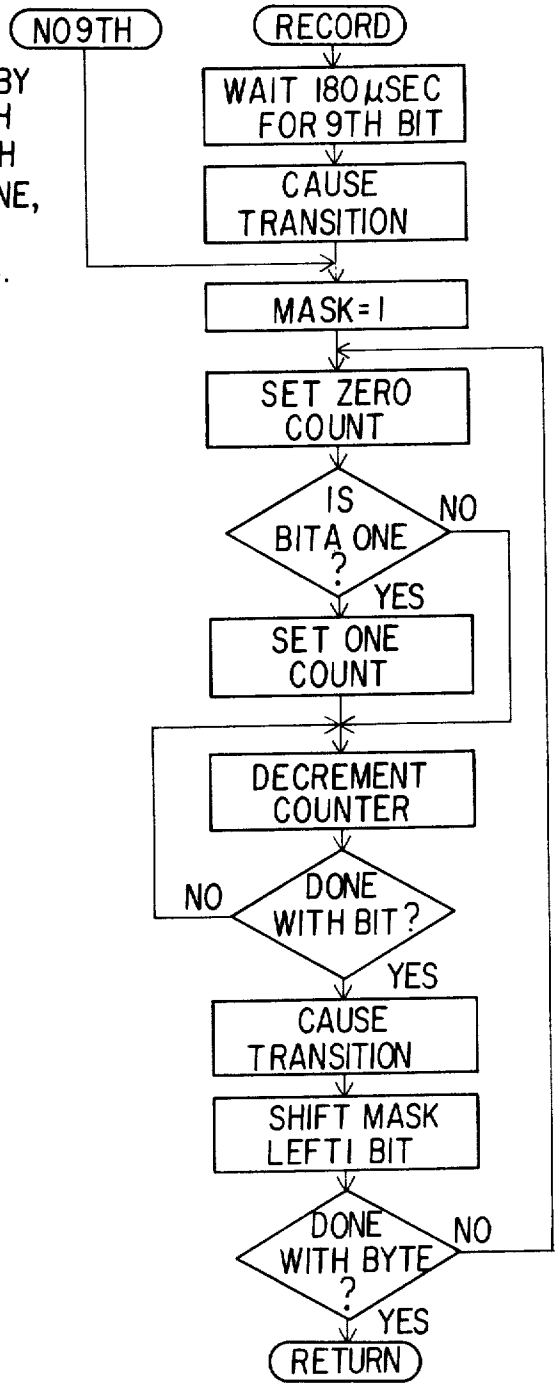


FIG 67Z

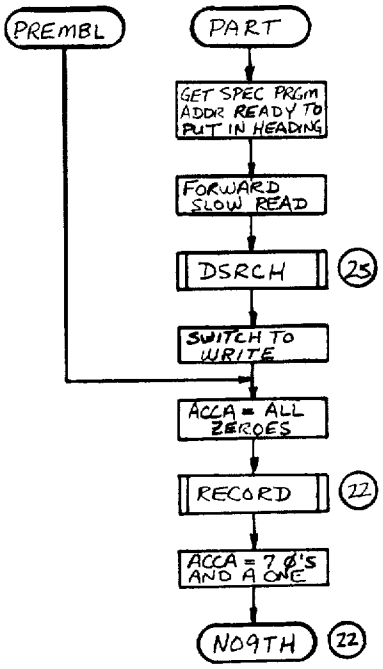

PREMBL
PART
GET SPEC PRGm ADDR READY TO PUT IN HEADING
FORWARD SLOW READ
DSRCH
25
SWITCH TO WRITE
ACCA = ALL ZEROES
RECORD
22
ACCA = 7 Ø'S AND A ONE
NO9TH
22

FIG. 68A

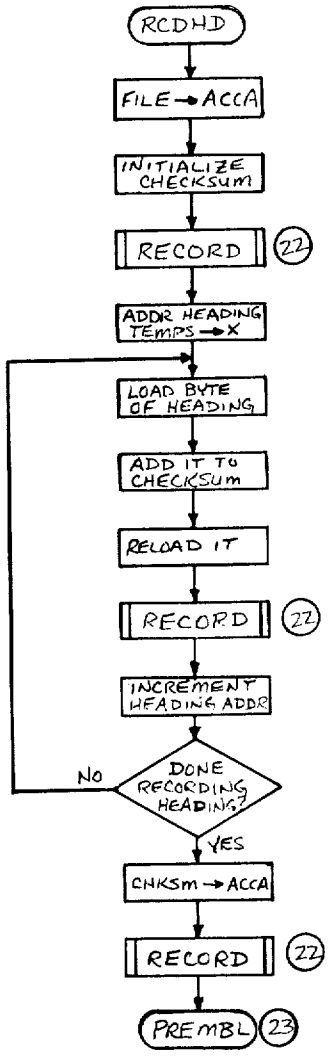

RCDHD
FILE→ACCA
INITIALIZE CHECKSUM
RECORD
22
ADDR HEADING TEMPS→X
LOAD BYTE OF HEADING
ADD IT TO CHECKSUM
RELOAD IT
RECORD
22
INCREMENT HEADING ADDR
DONE RECORDING HEADING?
NO
YES
CNKSm→ACCA
RECORD
22
PREMBL
23

FIG. 68B

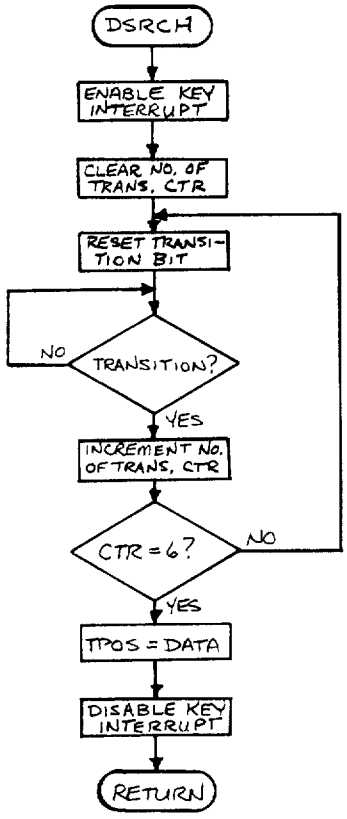

DSRCH
ENABLE KEY INTERRUPT
CLEAR NO. OF TRANS. CTR
RESET TRANSITION BIT
NO
TRANSITION?
YES
INCREMENT NO. OF TRANS. CTR
CTR = 6?
NO
YES
TPOS = DATA
DISABLE KEY INTERRUPT
RETURN

FIG. 68C

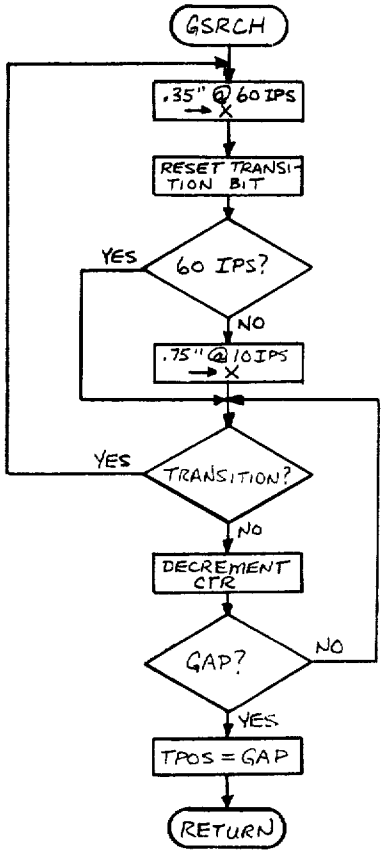

GSRCH
.35" @ 60 IPS
RESET TRANSITION BIT
60 IPS?
YES
NO
.75" @ 10 IPS
TRANSITION?
YES
NO
DECREMENT CTR
GAP?
NO
YES
TPOS = GAP
RETURN

FIG. 68D

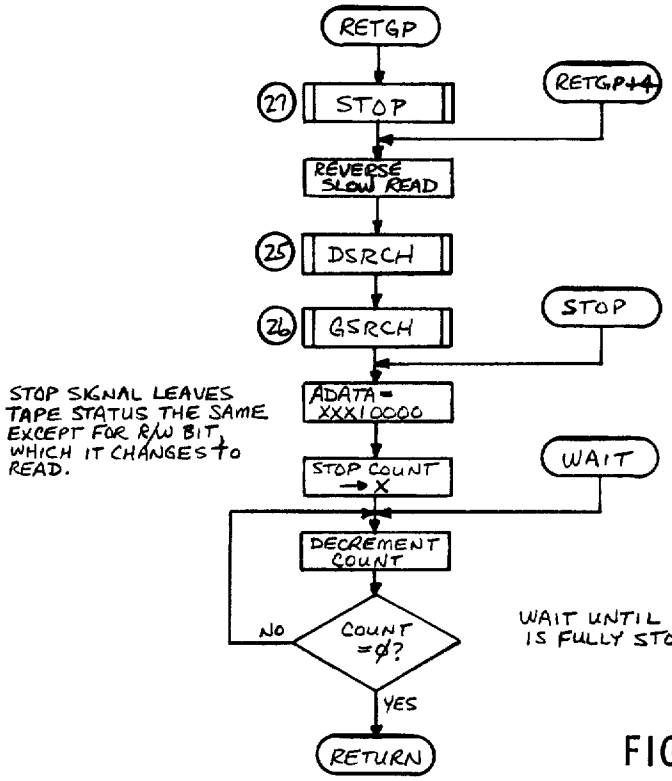

RETGP
27
STOP
RETGP+4
REVERSE SLOW READ
25
DSRCH
26
GSRCH
STOP
STOP SIGNAL LEAVES TAPE STATUS THE SAME EXCEPT FOR R/W BIT, WHICH IT CHANGES TO READ.
ADATA = XXX10000
STOP COUNT → X
WAIT
DECREMENT COUNT
NO
COUNT = ∅?
YES
WAIT UNTIL TAPE IS FULLY STOPPED
RETURN

FIG. 68E

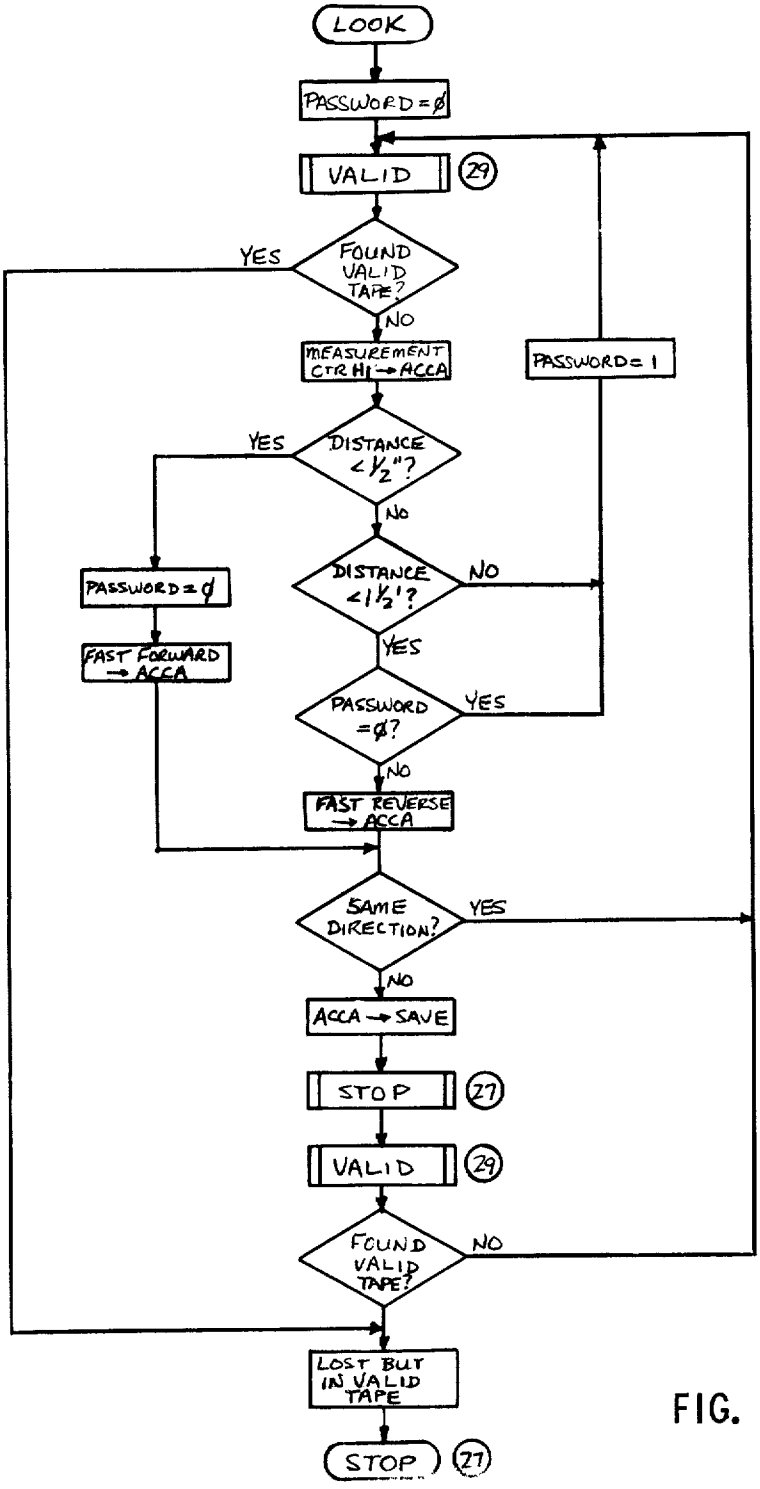

LOOK
PASSWORD=ø
VALID
29
FOUND VALID TAPE?
YES
NO
MEASUREMENT CTR HI → ACCA
PASSWORD=1
DISTANCE <½"?
YES
NO
PASSWORD=ø
DISTANCE <1½'?
NO
YES
FAST FORWARD → ACCA
PASSWORD =ø?
YES
NO
FAST REVERSE → ACCA
SAME DIRECTION?
YES
NO
ACCA → SAVE
STOP
27
VALID
29
FOUND VALID TAPE?
NO
LOST BUT IN VALID TAPE
STOP
27

FIG. 68F

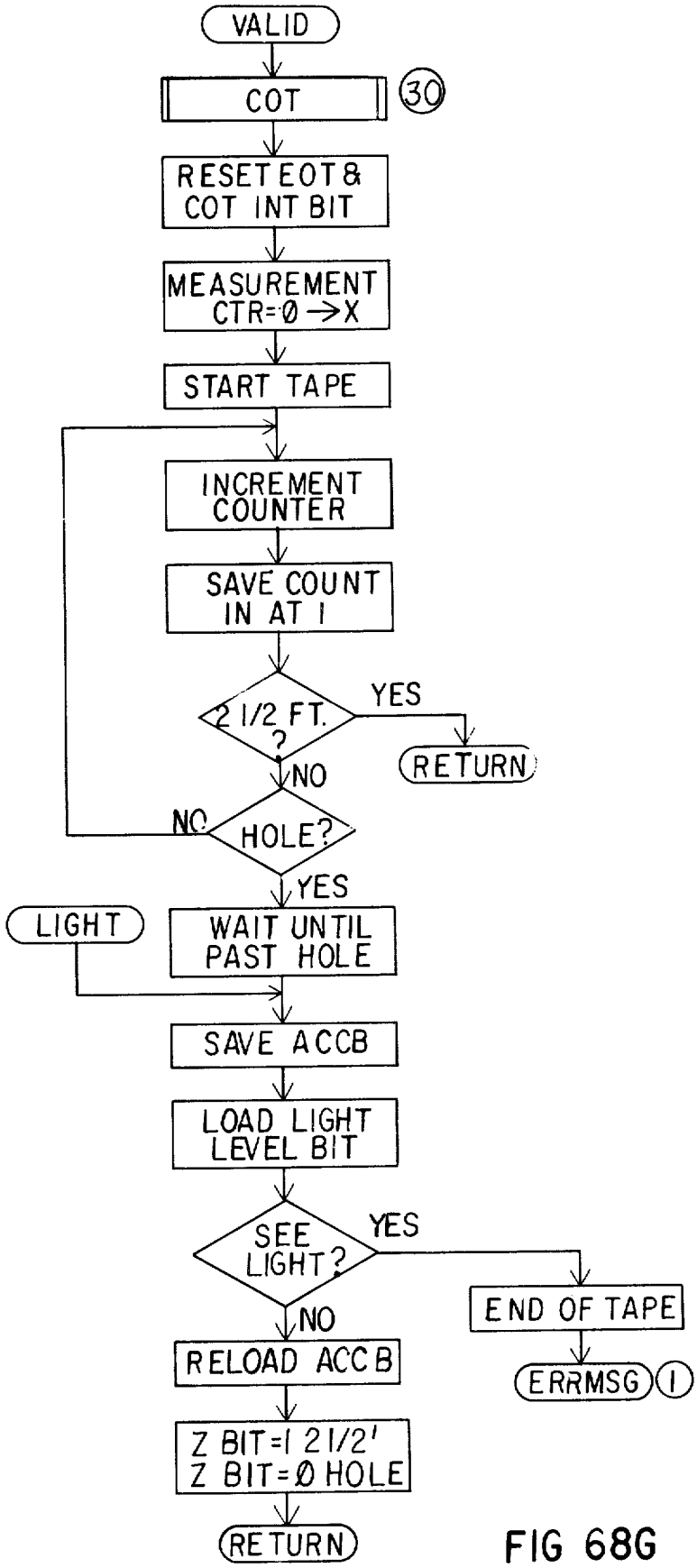

VALID
COT
30
RESET EOT & COT INT BIT
MEASUREMENT CTR=0 → X
START TAPE
INCREMENT COUNTER
SAVE COUNT IN AT I
2 1/2 FT. ?
YES
RETURN
NO
NO
HOLE?
YES
LIGHT
WAIT UNTIL PAST HOLE
SAVE ACCB
LOAD LIGHT LEVEL BIT
SEE LIGHT?
YES
END OF TAPE
NO
RELOAD ACCB
ERRMSG
I
Z BIT=1 2 1/2'
Z BIT=0 HOLE
RETURN

FIG 68G

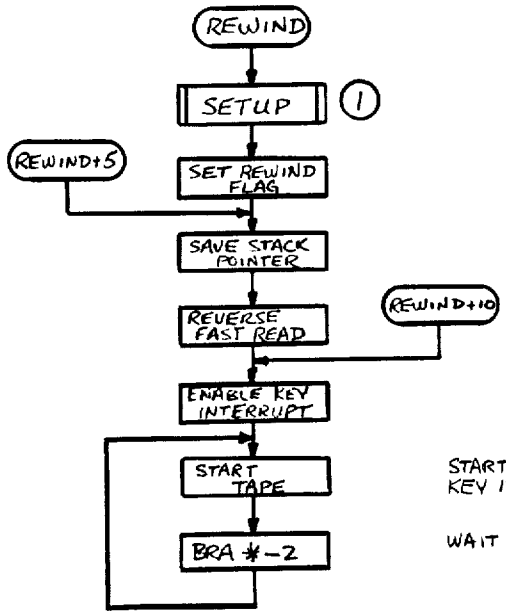


FIG. 68H

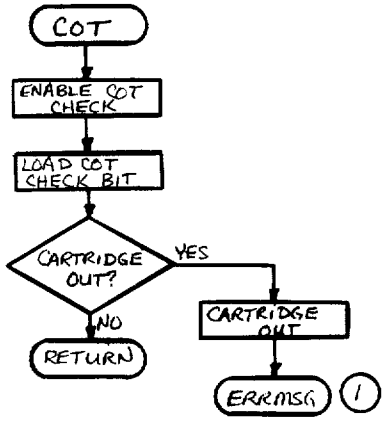


FIG. 68 I

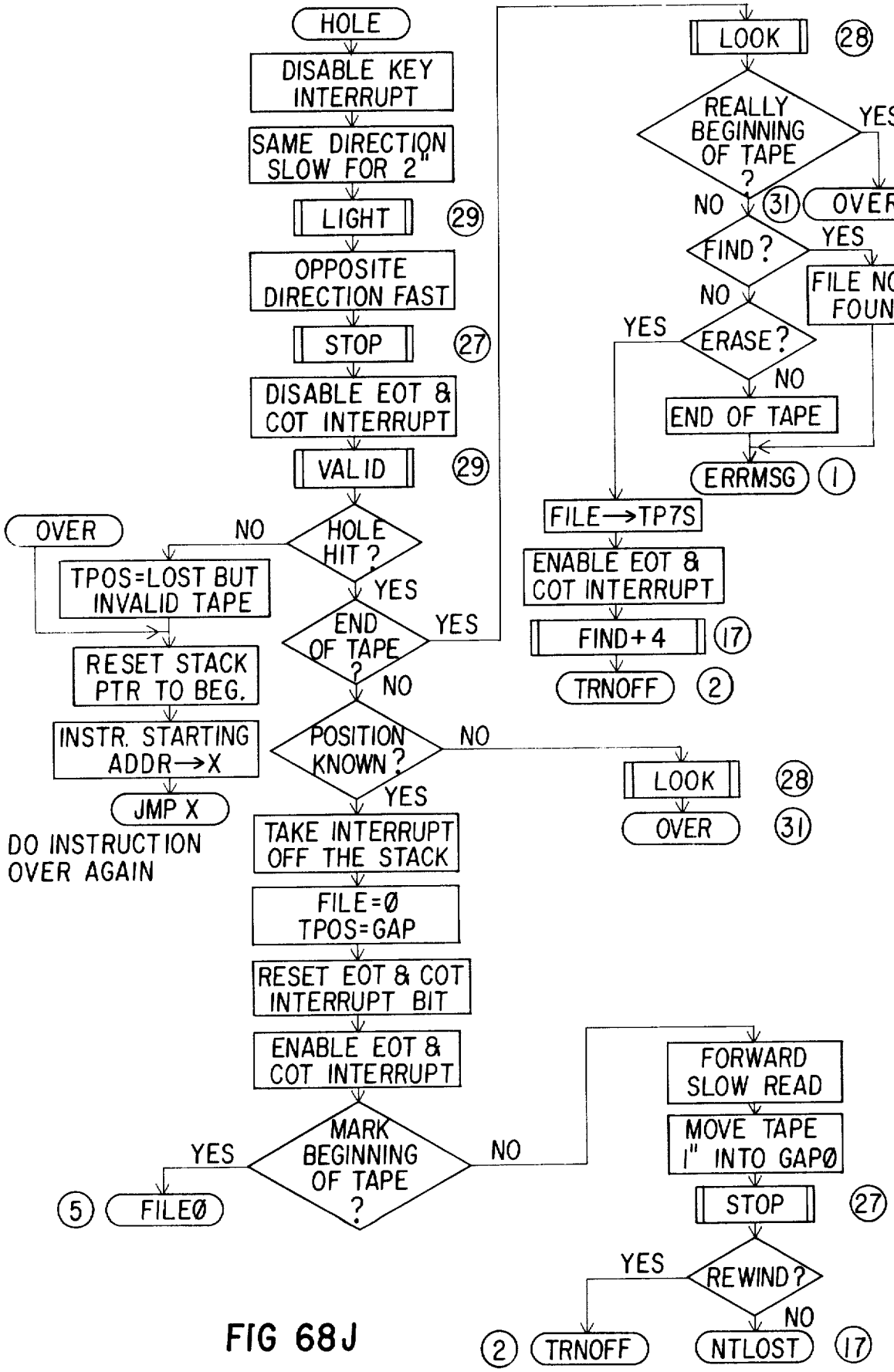

HOLE
DISABLE KEY INTERRUPT
SAME DIRECTION SLOW FOR 2"
LIGHT
29
OPPOSITE DIRECTION FAST
STOP
27
DISABLE EOT & COT INTERRUPT
VALID
29
HOLE HIT ?
NO
YES
OVER
TPOS=LOST BUT INVALID TAPE
RESET STACK PTR TO BEG.
INSTR. STARTING ADDR→X
JMP X
DO INSTRUCTION OVER AGAIN
END OF TAPE ?
YES
NO
POSITION KNOWN ?
NO
YES
TAKE INTERRUPT OFF THE STACK
FILE=Ø TPOS=GAP
RESET EOT & COT INTERRUPT BIT
ENABLE EOT & COT INTERRUPT
MARK BEGINNING OF TAPE ?
YES
NO
5
FILEØ
LOOK
28
REALLY BEGINNING OF TAPE ?
YES
NO
31
OVER
FIND ?
YES
NO
FILE NOT FOUND
ERASE ?
YES
NO
END OF TAPE
ERRMSG
1
FILE→TP7S
ENABLE EOT & COT INTERRUPT
FIND+4
17
TRNOFF
2
LOOK
28
OVER
31
FORWARD SLOW READ
MOVE TAPE 1" INTO GAPØ
STOP
27
REWIND ?
YES
NO
2
TRNOFF
NTLOST
17

FIG 68J

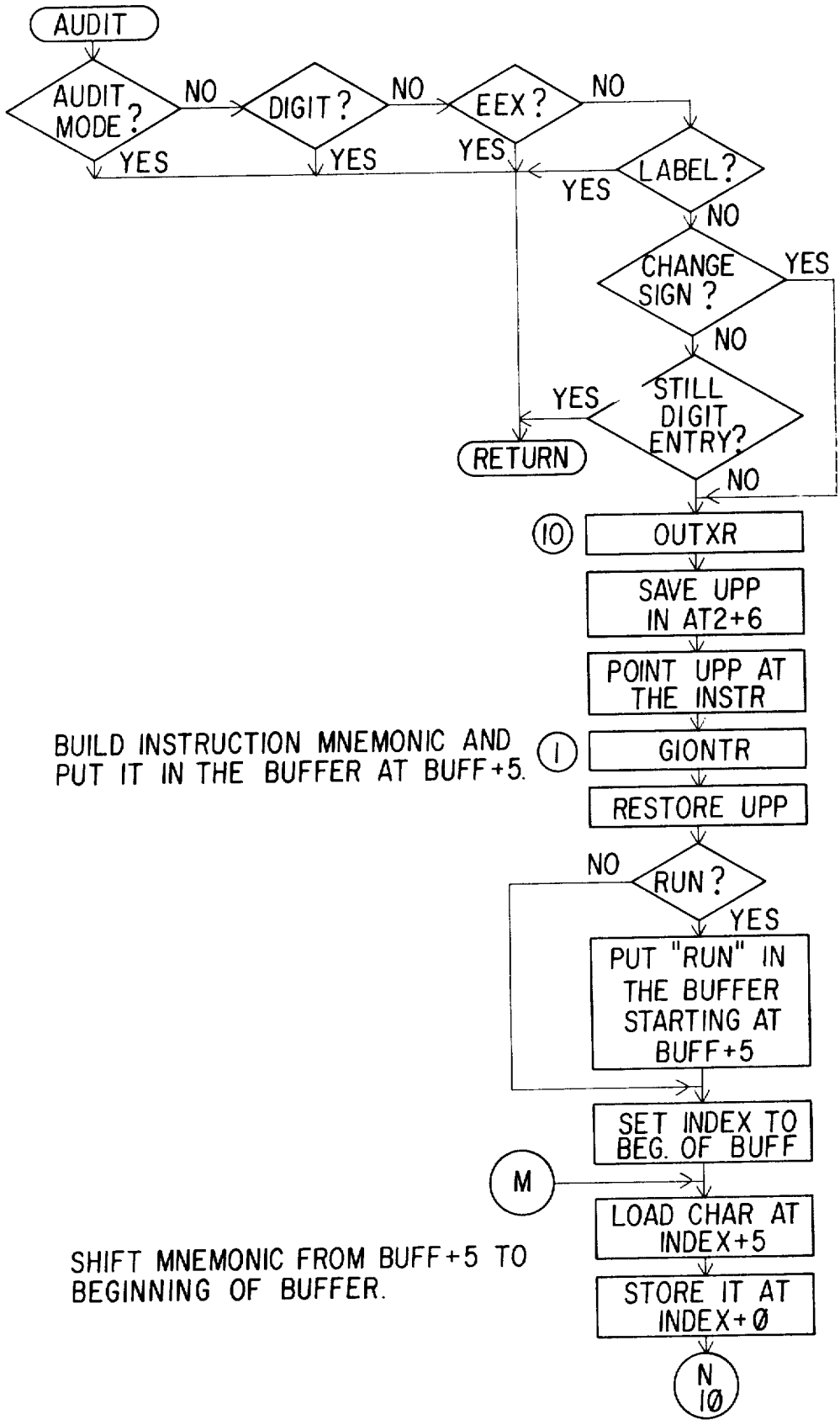

AUDIT
AUDIT MODE?
NO
YES
DIGIT?
NO
YES
EEX?
NO
YES
LABEL?
YES
NO
CHANGE SIGN?
YES
NO
STILL DIGIT ENTRY?
YES
NO
RETURN
10
OUTXR
SAVE UPP IN AT2+6
POINT UPP AT THE INSTR
BUILD INSTRUCTION MNEMONIC AND PUT IT IN THE BUFFER AT BUFF+5.
1
GIONTR
RESTORE UPP
NO
RUN?
YES
PUT "RUN" IN THE BUFFER STARTING AT BUFF+5
SET INDEX TO BEG. OF BUFF
M
LOAD CHAR AT INDEX+5
SHIFT MNEMONIC FROM BUFF+5 TO BEGINNING OF BUFFER.
STORE IT AT INDEX+Ø
N
1Ø

FIG 69K

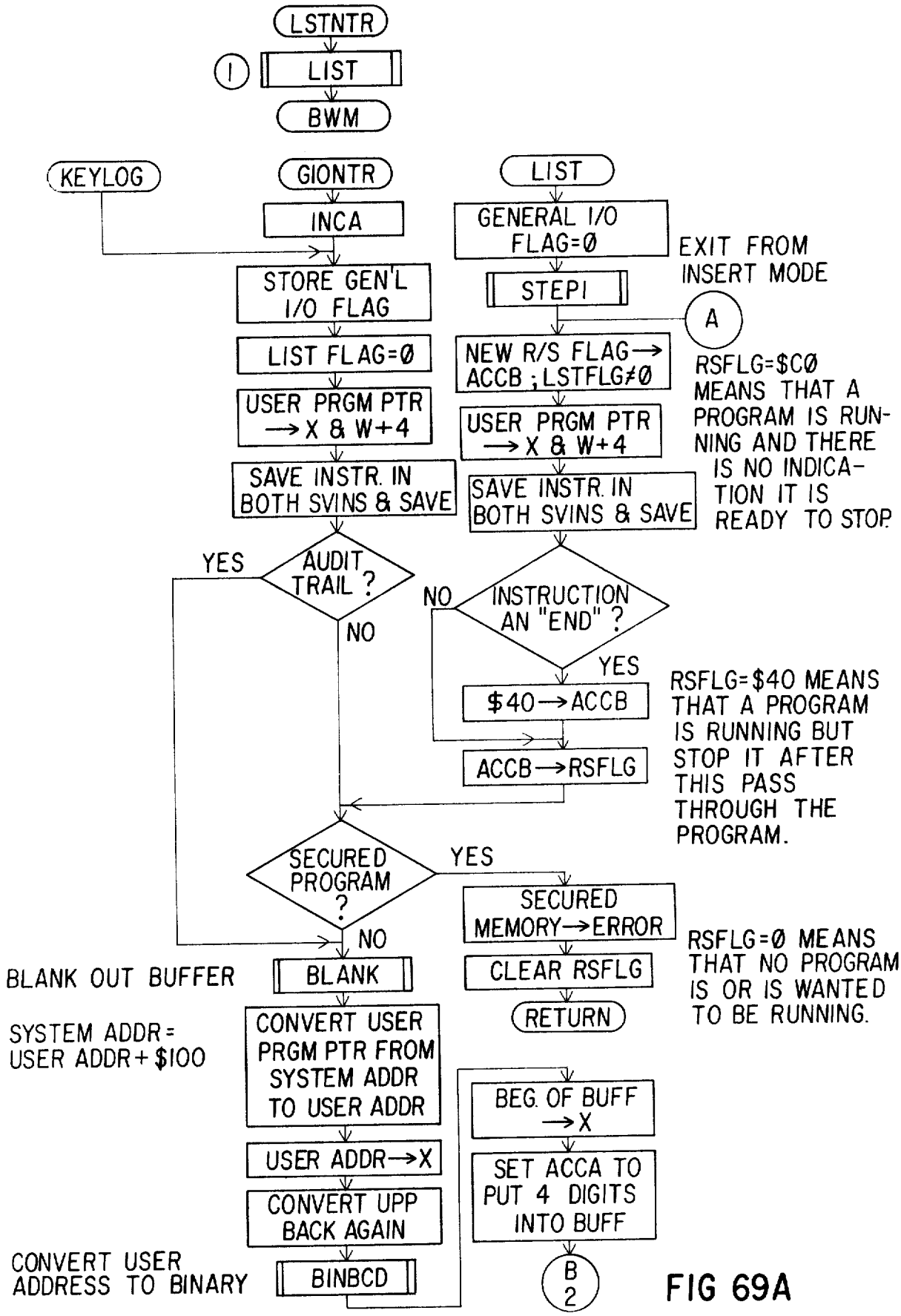

LSTNTR
1
LIST
BWM
KEYLOG
GIONTR
INCA
STORE GEN'L I/O FLAG
LIST FLAG=Ø
USER PRGM PTR → X & W+4
SAVE INSTR. IN BOTH SVINS & SAVE
YES
AUDIT TRAIL ?
NO
SECURED PROGRAM ?
YES
NO
BLANK OUT BUFFER
BLANK
SYSTEM ADDR = USER ADDR + $100
CONVERT USER PRGM PTR FROM SYSTEM ADDR TO USER ADDR
USER ADDR → X
CONVERT UPP BACK AGAIN
CONVERT USER ADDRESS TO BINARY
BINBCD
LIST
GENERAL I/O FLAG=Ø
STEPI
NEW R/S FLAG → ACCB ; LSTFLG≠Ø
USER PRGM PTR → X & W+4
SAVE INSTR. IN BOTH SVINS & SAVE
NO
INSTRUCTION AN "END" ?
YES
$40 → ACCB
ACCB → RSFLG
SECURED MEMORY → ERROR
CLEAR RSFLG
RETURN
BEG. OF BUFF → X
SET ACCA TO PUT 4 DIGITS INTO BUFF
B 2
A
EXIT FROM INSERT MODE
RSFLG=$CØ MEANS THAT A PROGRAM IS RUNNING AND THERE IS NO INDICATION IT IS READY TO STOP
RSFLG=$40 MEANS THAT A PROGRAM IS RUNNING BUT STOP IT AFTER THIS PASS THROUGH THE PROGRAM.
RSFLG=Ø MEANS THAT NO PROGRAM IS OR IS WANTED TO BE RUNNING.

FIG 69A

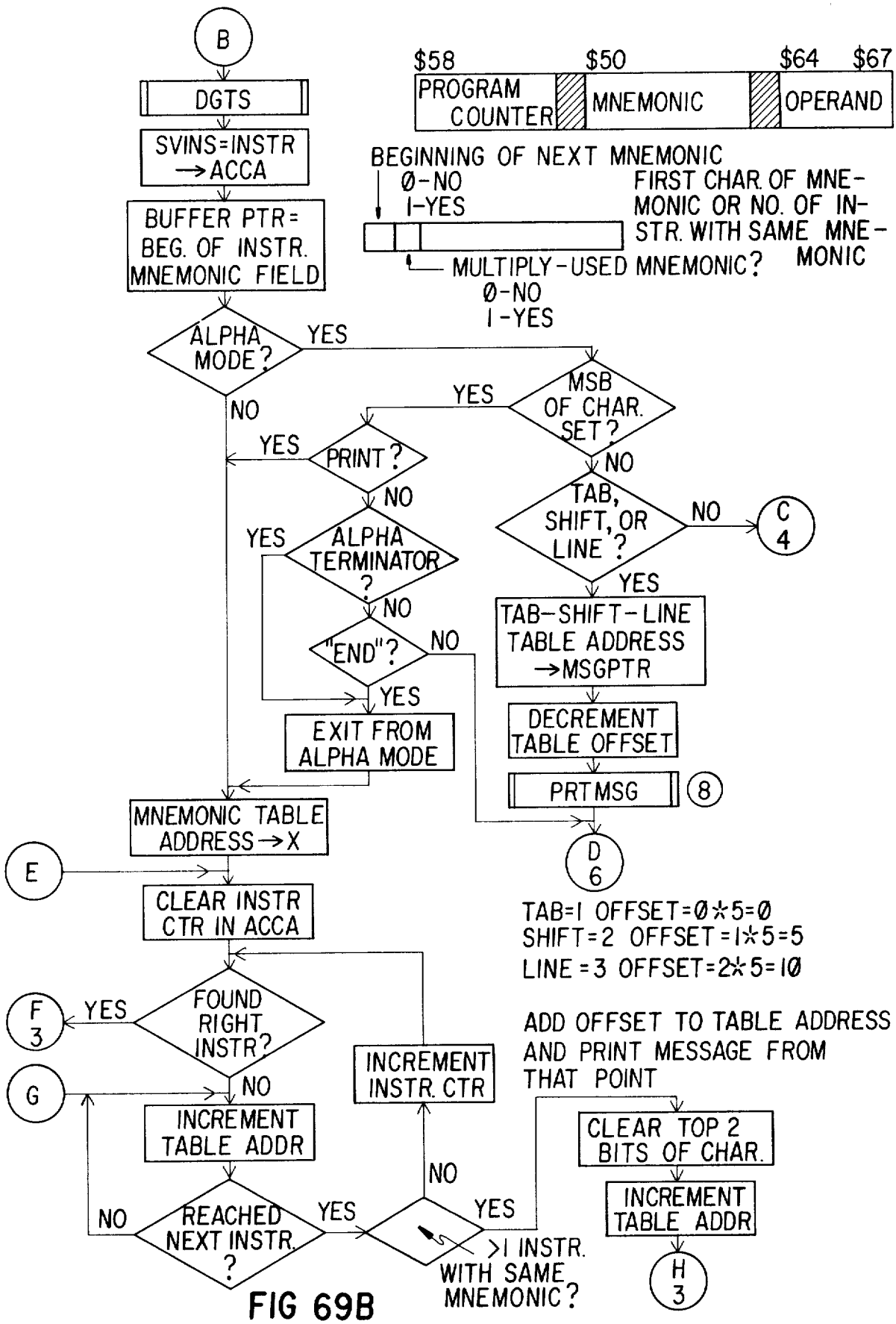

B
DGTS
SVINS=INSTR →ACCA
BUFFER PTR= BEG. OF INSTR. MNEMONIC FIELD
ALPHA MODE?
YES
NO
MSB OF CHAR. SET?
YES
NO
PRINT?
YES
NO
TAB, SHIFT, OR LINE?
NO
C 4
YES
ALPHA TERMINATOR?
YES
NO
"END"?
NO
YES
EXIT FROM ALPHA MODE
TAB-SHIFT-LINE TABLE ADDRESS →MSGPTR
DECREMENT TABLE OFFSET
PRTMSG
8
D 6
MNEMONIC TABLE ADDRESS→X
E
CLEAR INSTR CTR IN ACCA
F 3
YES
FOUND RIGHT INSTR?
G
NO
INCREMENT TABLE ADDR
NO
REACHED NEXT INSTR.?
YES
INCREMENT INSTR. CTR
NO
YES
>1 INSTR. WITH SAME MNEMONIC?
CLEAR TOP 2 BITS OF CHAR.
INCREMENT TABLE ADDR
H 3
$58
$50
$64
$67
PROGRAM COUNTER
MNEMONIC
OPERAND
BEGINNING OF NEXT MNEMONIC
Ø-NO
I-YES
FIRST CHAR. OF MNEMONIC OR NO. OF INSTR. WITH SAME MNEMONIC
MULTIPLY-USED MNEMONIC?
Ø-NO
I-YES
TAB=1 OFFSET=Ø*5=Ø
SHIFT=2 OFFSET=1*5=5
LINE=3 OFFSET=2*5=1Ø
ADD OFFSET TO TABLE ADDRESS AND PRINT MESSAGE FROM THAT POINT

FIG 69B

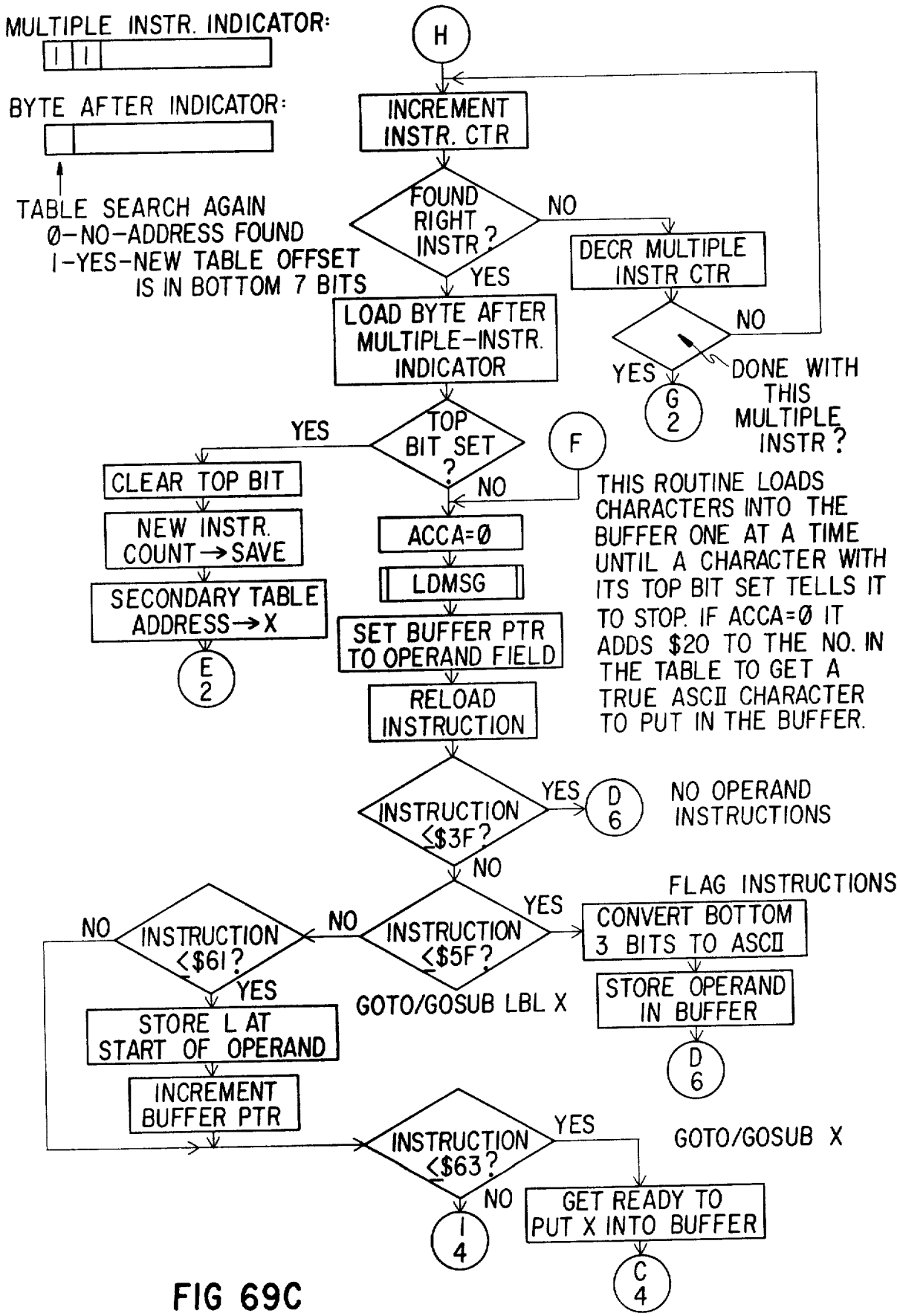
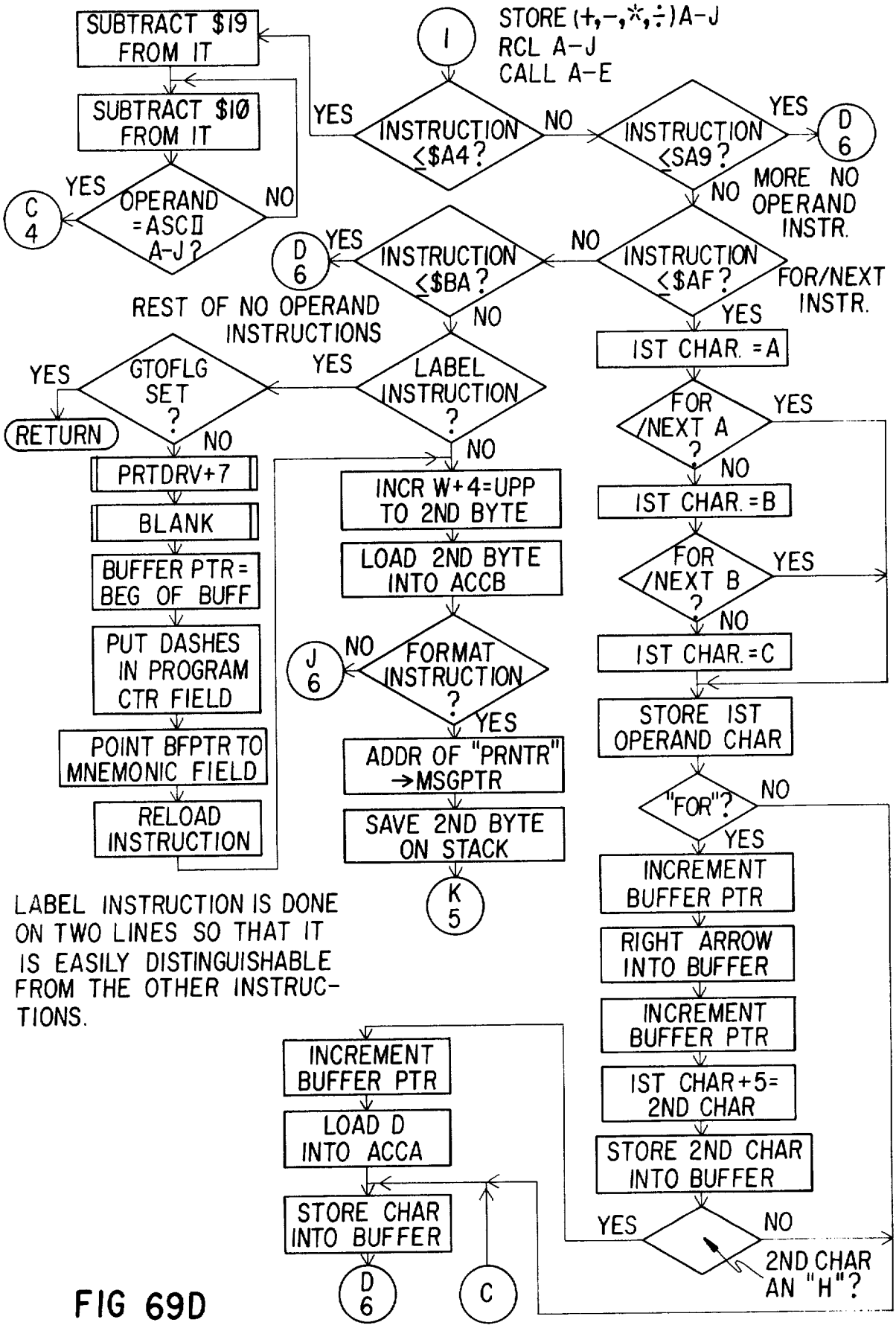

MULTIPLE INSTR. INDICATOR:
BYTE AFTER INDICATOR:
TABLE SEARCH AGAIN
Ø-NO-ADDRESS FOUND
1-YES-NEW TABLE OFFSET IS IN BOTTOM 7 BITS
H
INCREMENT INSTR. CTR
FOUND RIGHT INSTR?
NO
YES
DECR MULTIPLE INSTR CTR
NO
YES
DONE WITH THIS MULTIPLE INSTR?
G 2
LOAD BYTE AFTER MULTIPLE-INSTR. INDICATOR
TOP BIT SET?
YES
NO
F
CLEAR TOP BIT
NEW INSTR. COUNT→SAVE
SECONDARY TABLE ADDRESS→X
E 2
ACCA=Ø
LDMSG
SET BUFFER PTR TO OPERAND FIELD
RELOAD INSTRUCTION
THIS ROUTINE LOADS CHARACTERS INTO THE BUFFER ONE AT A TIME UNTIL A CHARACTER WITH ITS TOP BIT SET TELLS IT TO STOP. IF ACCA=Ø IT ADDS $20 TO THE NO. IN THE TABLE TO GET A TRUE ASCII CHARACTER TO PUT IN THE BUFFER.
INSTRUCTION ≤$3F?
YES
D 6
NO OPERAND INSTRUCTIONS
NO
FLAG INSTRUCTIONS
INSTRUCTION ≤$5F?
NO
YES
CONVERT BOTTOM 3 BITS TO ASCII
STORE OPERAND IN BUFFER
D 6
INSTRUCTION ≤$61?
NO
YES
GOTO/GOSUB LBL X
STORE L AT START OF OPERAND
INCREMENT BUFFER PTR
INSTRUCTION ≤$63?
YES
GOTO/GOSUB X
NO
GET READY TO PUT X INTO BUFFER
I 4
C 4

FIG 69C



LABEL INSTRUCTION IS DONE ON TWO LINES SO THAT IT IS EASILY DISTINGUISHABLE FROM THE OTHER INSTRUCTIONS.

FIG 69D

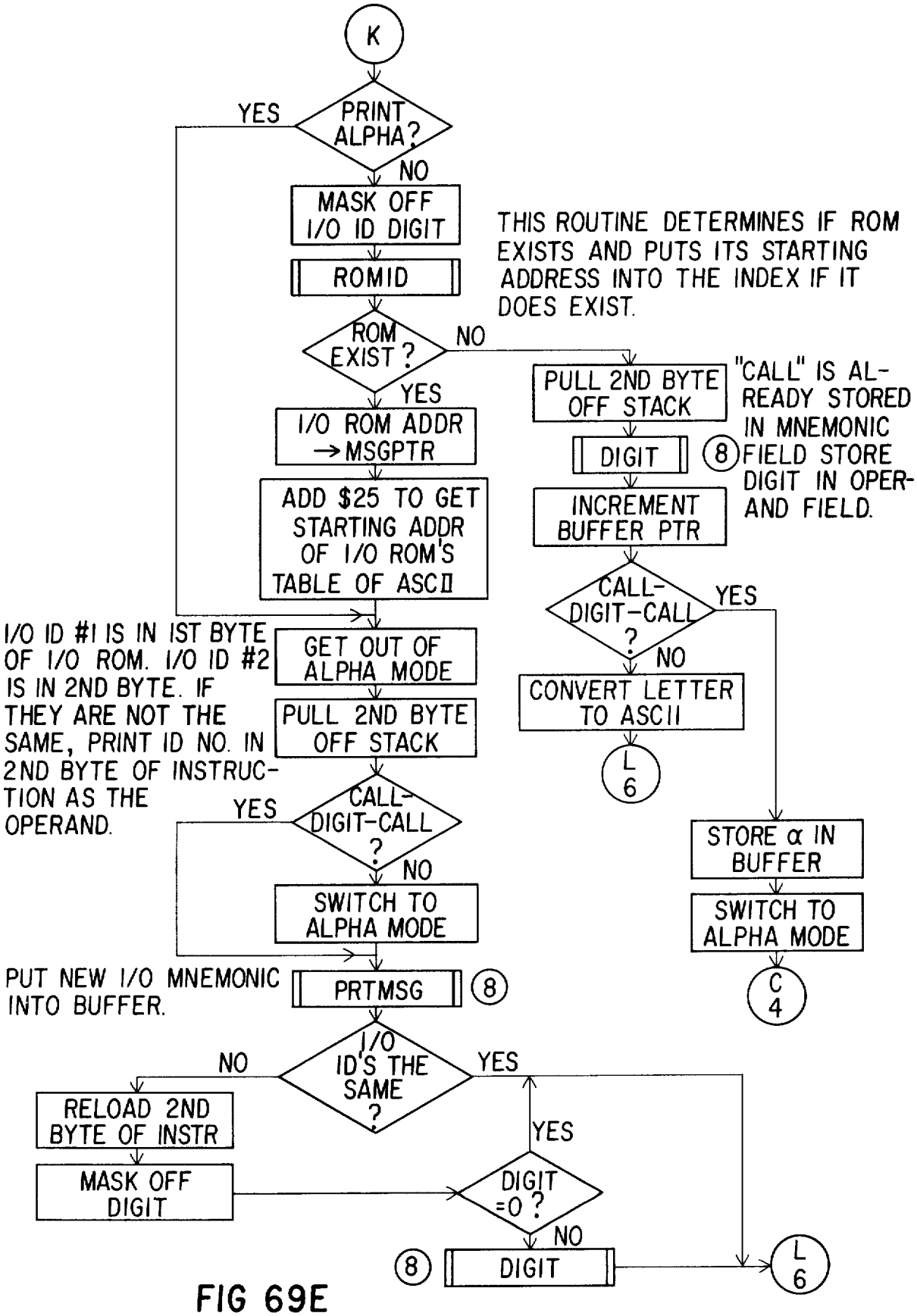

K
PRINT ALPHA?
YES
NO
MASK OFF I/O ID DIGIT
ROMID
THIS ROUTINE DETERMINES IF ROM EXISTS AND PUTS ITS STARTING ADDRESS INTO THE INDEX IF IT DOES EXIST.
ROM EXIST ?
NO
YES
I/O ROM ADDR →MSGPTR
ADD $25 TO GET STARTING ADDR OF I/O ROM'S TABLE OF ASCII
PULL 2ND BYTE OFF STACK
DIGIT
8
"CALL" IS ALREADY STORED IN MNEMONIC FIELD STORE DIGIT IN OPERAND FIELD.
INCREMENT BUFFER PTR
CALL-DIGIT-CALL ?
YES
NO
CONVERT LETTER TO ASCII
L 6
I/O ID #1 IS IN IST BYTE OF I/O ROM. I/O ID #2 IS IN 2ND BYTE. IF THEY ARE NOT THE SAME, PRINT ID NO. IN 2ND BYTE OF INSTRUCTION AS THE OPERAND.
GET OUT OF ALPHA MODE
PULL 2ND BYTE OFF STACK
CALL-DIGIT-CALL ?
YES
NO
SWITCH TO ALPHA MODE
STORE α IN BUFFER
SWITCH TO ALPHA MODE
C 4
PUT NEW I/O MNEMONIC INTO BUFFER.
PRTMSG
8
I/O ID'S THE SAME ?
NO
YES
RELOAD 2ND BYTE OF INSTR
MASK OFF DIGIT
DIGIT =0 ?
YES
NO
8
DIGIT
L 6

FIG 69E

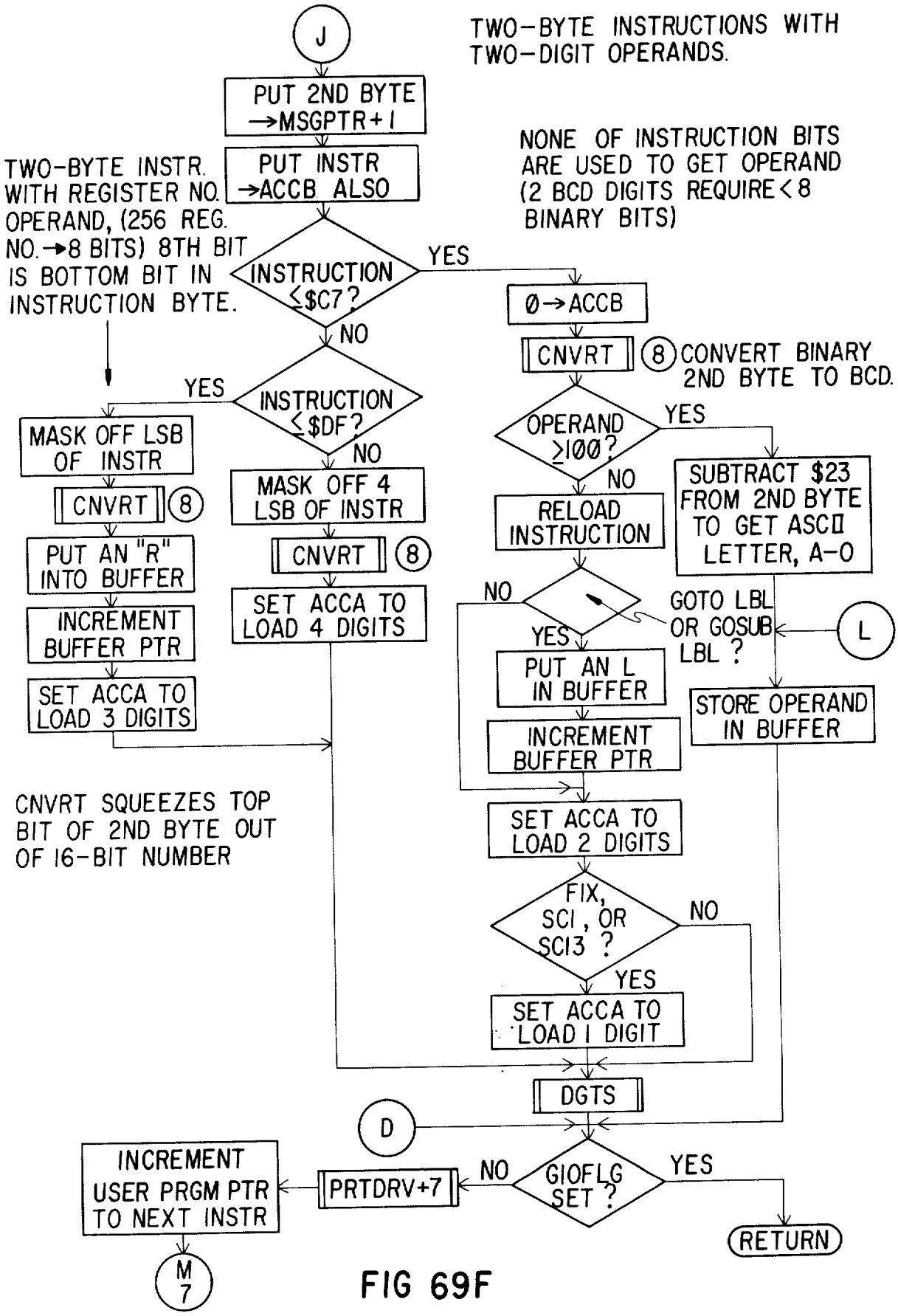

J
TWO-BYTE INSTRUCTIONS WITH TWO-DIGIT OPERANDS.
PUT 2ND BYTE →MSGPTR+1
PUT INSTR →ACCB ALSO
TWO-BYTE INSTR. WITH REGISTER NO. OPERAND, (256 REG. NO.→8 BITS) 8TH BIT IS BOTTOM BIT IN INSTRUCTION BYTE.
NONE OF INSTRUCTION BITS ARE USED TO GET OPERAND (2 BCD DIGITS REQUIRE < 8 BINARY BITS)
INSTRUCTION ≤$C7?
YES
NO
Ø→ACCB
CNVRT (8) CONVERT BINARY 2ND BYTE TO BCD.
INSTRUCTION ≤$DF?
YES
NO
OPERAND ≥1ØØ?
YES
NO
MASK OFF LSB OF INSTR
CNVRT (8)
PUT AN "R" INTO BUFFER
INCREMENT BUFFER PTR
SET ACCA TO LOAD 3 DIGITS
MASK OFF 4 LSB OF INSTR
CNVRT (8)
SET ACCA TO LOAD 4 DIGITS
RELOAD INSTRUCTION
SUBTRACT $23 FROM 2ND BYTE TO GET ASCII LETTER, A-O
NO
GOTO LBL OR GOSUB LBL ?
YES
L
PUT AN L IN BUFFER
INCREMENT BUFFER PTR
STORE OPERAND IN BUFFER
CNVRT SQUEEZES TOP BIT OF 2ND BYTE OUT OF 16-BIT NUMBER
SET ACCA TO LOAD 2 DIGITS
FIX, SCI, OR SCI3 ?
NO
YES
SET ACCA TO LOAD 1 DIGIT
DGTS
D
INCREMENT USER PRGM PTR TO NEXT INSTR
PRTDRV+7
NO
GIOFLG SET ?
YES
RETURN
M 7

FIG 69F

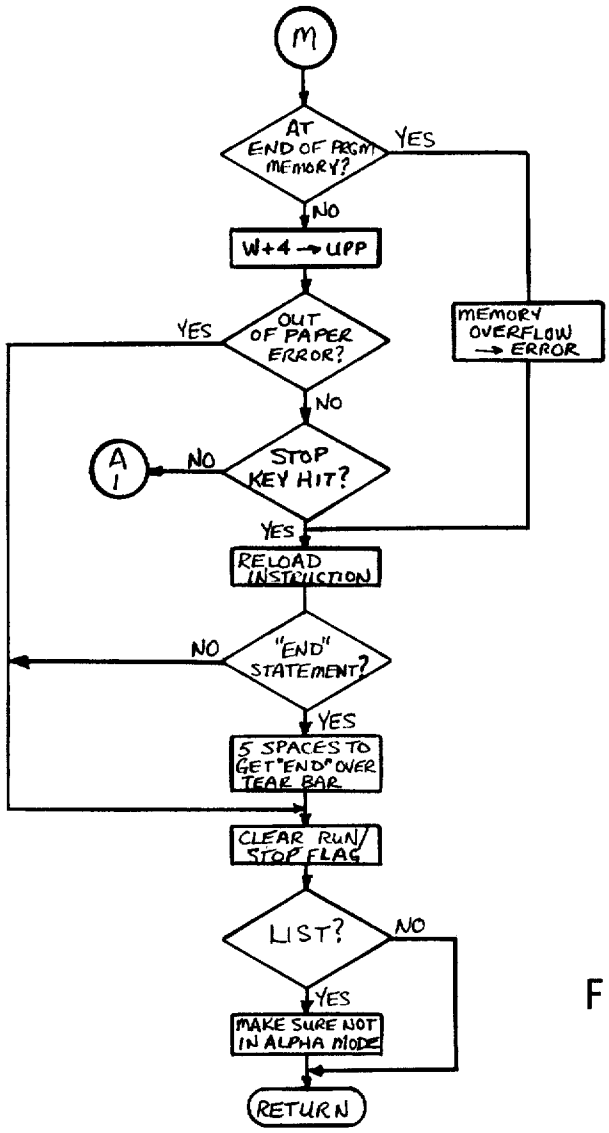

M
AT END OF PRGM MEMORY?
YES
NO
W+4 → UPP
OUT OF PAPER ERROR?
YES
NO
MEMORY OVERFLOW → ERROR
A 1
NO
STOP KEY HIT?
YES
RELOAD INSTRUCTION
"END" STATEMENT?
NO
YES
5 SPACES TO GET "END" OVER TEAR BAR
CLEAR RUN/STOP FLAG
LIST?
NO
YES
MAKE SURE NOT IN ALPHA MODE
RETURN

FIG. 69G

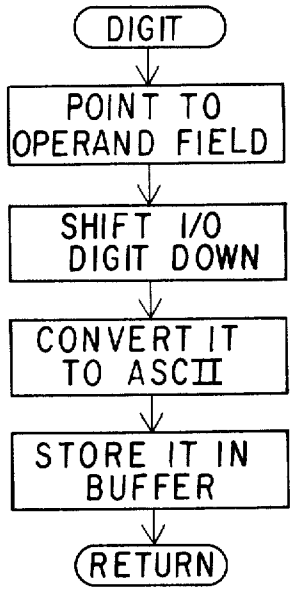


FIG 69H

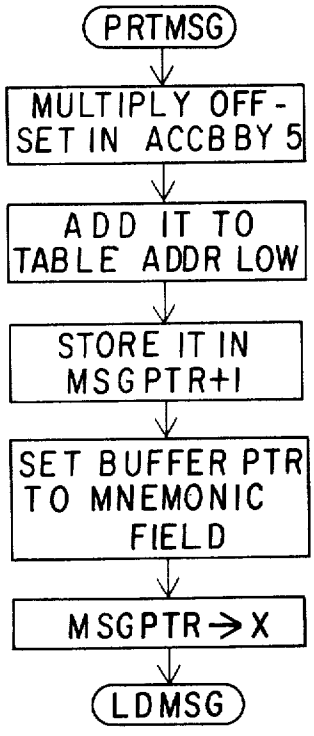


FIG 69I

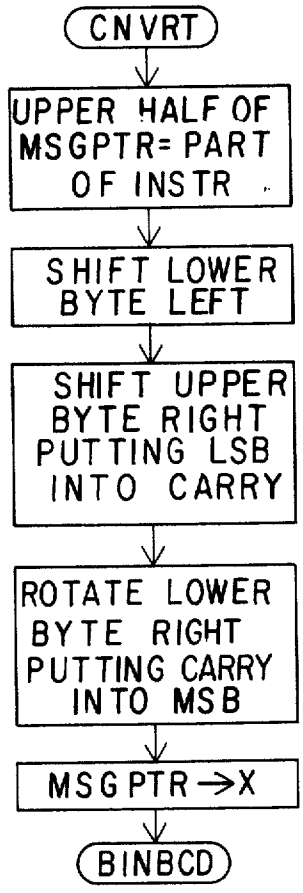


FIG 69J

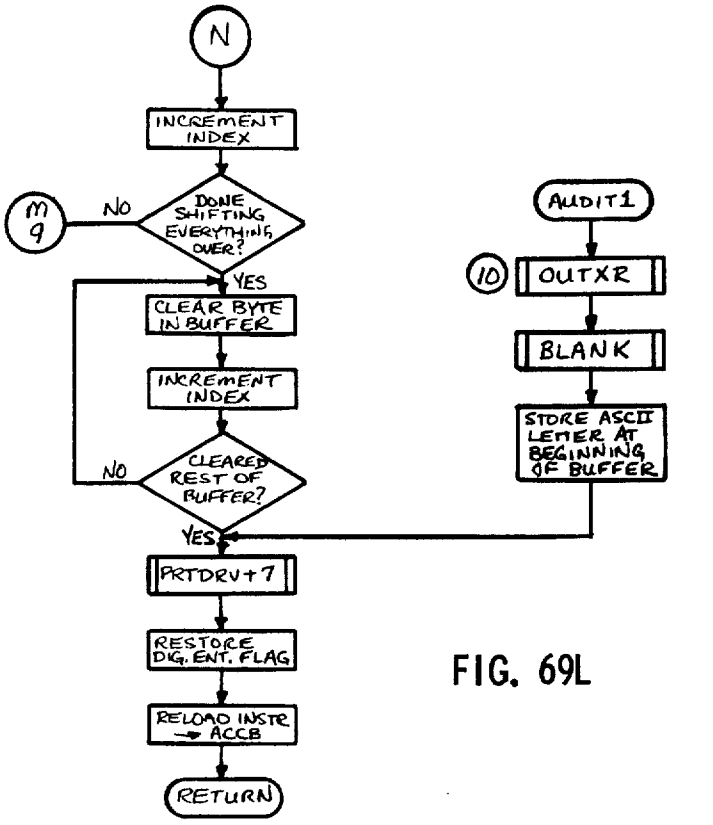


FIG. 69L

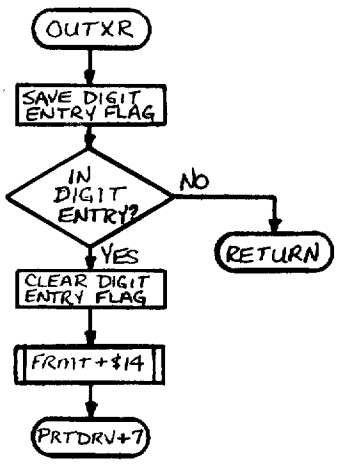


FIG. 69M

THIS ROUTINE TAKES TWO'S COMPLEMENT EXPONENT IN ACCA, CONVERTS IT TO BCD, AND STORES THE RESULT IN XPNT.

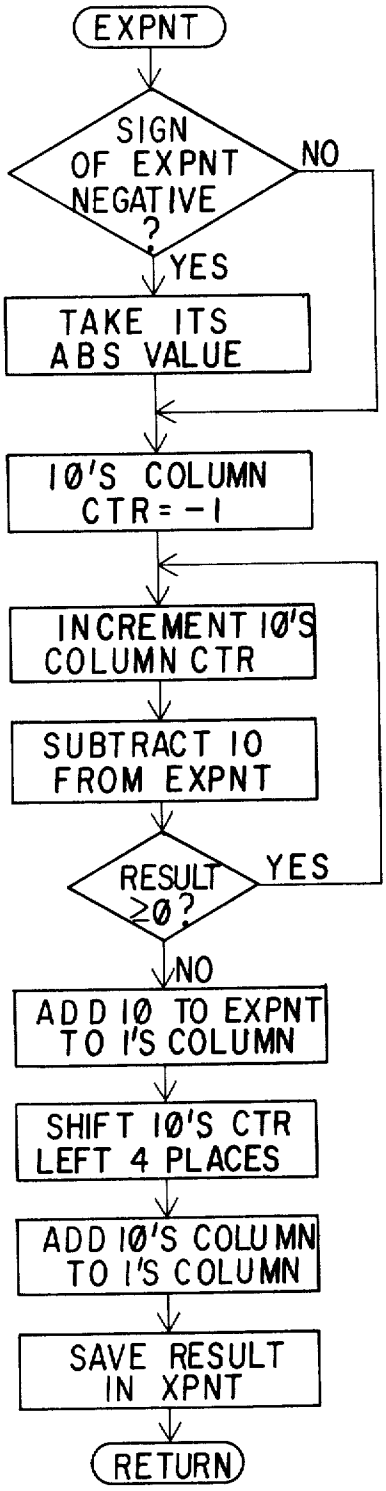


FIG 70A

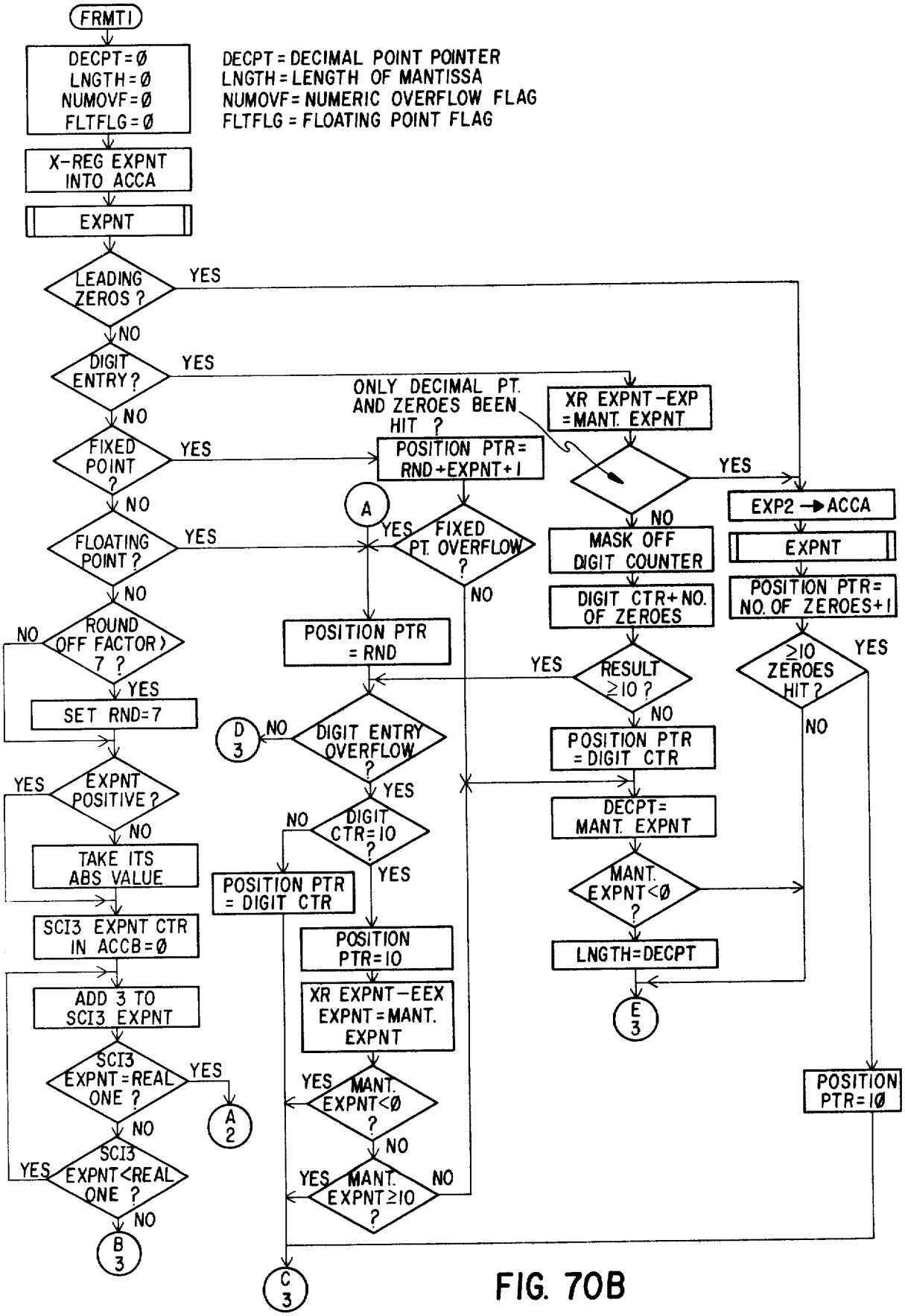

FRMTI
DECPT=Ø
LNGTH=Ø
NUMOVF=Ø
FLTFLG=Ø
DECPT = DECIMAL POINT POINTER
LNGTH = LENGTH OF MANTISSA
NUMOVF = NUMERIC OVERFLOW FLAG
FLTFLG = FLOATING POINT FLAG
X-REG EXPNT INTO ACCA
EXPNT
LEADING ZEROS ?
YES
NO
DIGIT ENTRY ?
YES
NO
FIXED POINT ?
YES
NO
FLOATING POINT ?
YES
NO
ROUND OFF FACTOR > 7 ?
NO
YES
SET RND=7
EXPNT POSITIVE ?
YES
NO
TAKE ITS ABS VALUE
SCI3 EXPNT CTR IN ACCB=Ø
ADD 3 TO SCI3 EXPNT
SCI3 EXPNT=REAL ONE ?
YES
NO
SCI3 EXPNT<REAL ONE ?
YES
NO
A 2
B 3
ONLY DECIMAL PT. AND ZEROES BEEN HIT ?
POSITION PTR= RND+EXPNT+1
A
FIXED PT. OVERFLOW ?
YES
NO
POSITION PTR = RND
D 3
DIGIT ENTRY OVERFLOW ?
NO
YES
DIGIT CTR=10 ?
NO
YES
POSITION PTR = DIGIT CTR
POSITION PTR=10
XR EXPNT-EEX EXPNT=MANT. EXPNT
MANT. EXPNT<Ø ?
YES
NO
MANT. EXPNT≥10 ?
YES
NO
C 3
XR EXPNT-EXP =MANT. EXPNT
YES
NO
MASK OFF DIGIT COUNTER
DIGIT CTR+NO. OF ZEROES
RESULT ≥10 ?
YES
NO
POSITION PTR = DIGIT CTR
DECPT= MANT. EXPNT
MANT. EXPNT<Ø ?
LNGTH=DECPT
E 3
EXP2 → ACCA
EXPNT
POSITION PTR= NO. OF ZEROES+1
≥10 ZEROES HIT ?
YES
NO
POSITION PTR=10

FIG. 70B

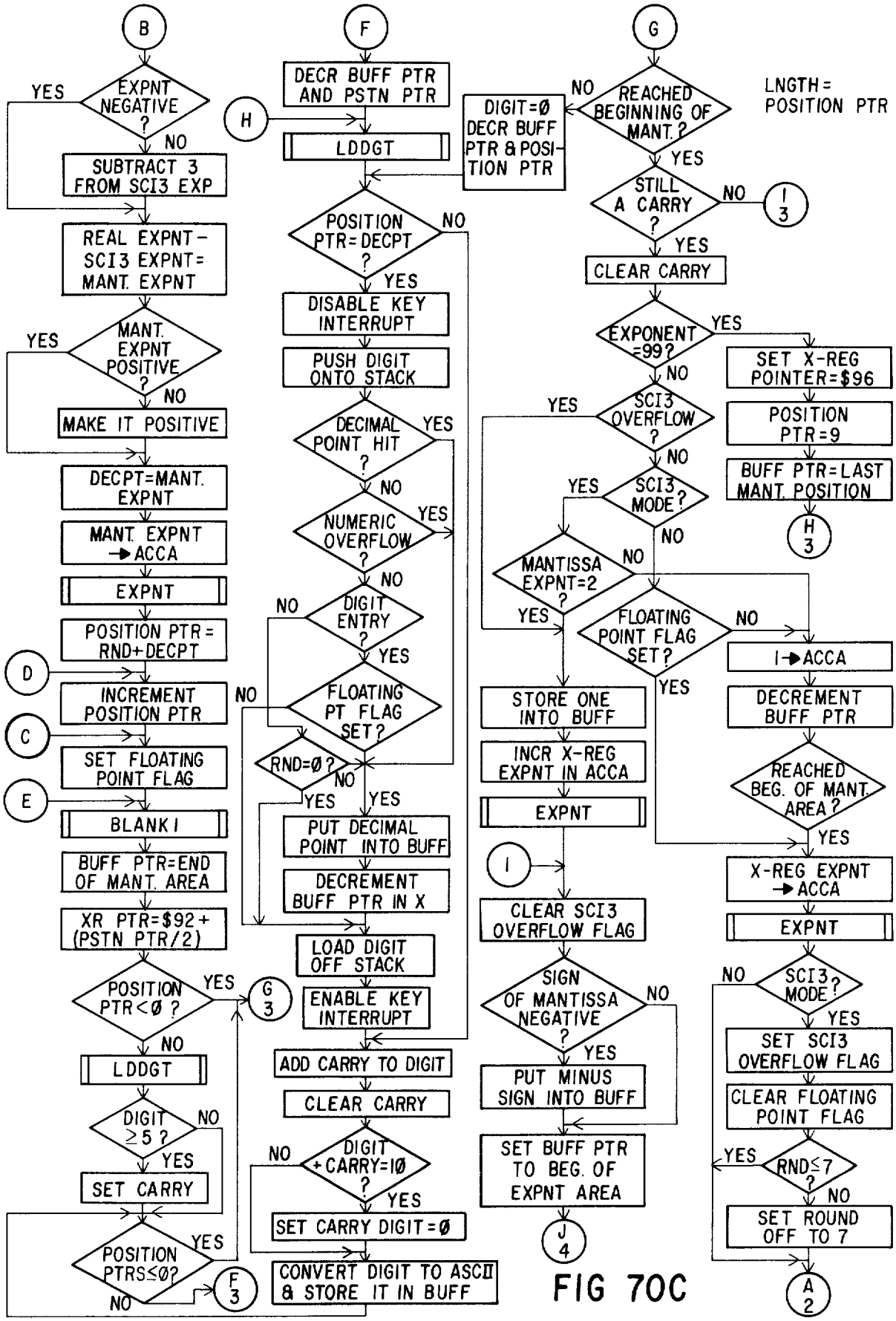

B
EXPNT NEGATIVE ?
YES
NO
SUBTRACT 3 FROM SCI3 EXP
REAL EXPNT - SCI3 EXPNT = MANT. EXPNT
MANT. EXPNT POSITIVE ?
YES
NO
MAKE IT POSITIVE
DECPT=MANT. EXPNT
MANT. EXPNT →ACCA
EXPNT
POSITION PTR = RND+DECPT
D
INCREMENT POSITION PTR
C
SET FLOATING POINT FLAG
E
BLANK I
BUFF PTR=END OF MANT. AREA
XR PTR=$92+ (PSTN PTR/2)
POSITION PTR<Ø ?
YES
G 3
NO
LDDGT
DIGIT ≥5 ?
NO
YES
SET CARRY
POSITION PTRS≤Ø?
YES
NO
F 3
F
DECR BUFF PTR AND PSTN PTR
H
LDDGT
POSITION PTR=DECPT ?
NO
YES
DISABLE KEY INTERRUPT
PUSH DIGIT ONTO STACK
DECIMAL POINT HIT ?
YES
NO
NUMERIC OVERFLOW ?
YES
NO
DIGIT ENTRY ?
NO
YES
FLOATING PT FLAG SET?
NO
YES
RND=Ø ?
NO
YES
PUT DECIMAL POINT INTO BUFF
DECREMENT BUFF PTR IN X
LOAD DIGIT OFF STACK
ENABLE KEY INTERRUPT
ADD CARRY TO DIGIT
CLEAR CARRY
DIGIT +CARRY=1Ø ?
NO
YES
SET CARRY DIGIT=Ø
CONVERT DIGIT TO ASCII & STORE IT IN BUFF
G
REACHED BEGINNING OF MANT.?
NO
YES
DIGIT=Ø DECR BUFF PTR & POSITION PTR
LNGTH = POSITION PTR
STILL A CARRY ?
NO
I 3
YES
CLEAR CARRY
EXPONENT =99?
YES
NO
SET X-REG POINTER=$96
POSITION PTR=9
BUFF PTR=LAST MANT. POSITION
H 3
SCI3 OVERFLOW ?
YES
NO
SCI3 MODE?
YES
NO
MANTISSA EXPNT=2 ?
NO
YES
FLOATING POINT FLAG SET?
NO
YES
STORE ONE INTO BUFF
INCR X-REG EXPNT IN ACCA
EXPNT
I
CLEAR SCI3 OVERFLOW FLAG
SIGN OF MANTISSA NEGATIVE ?
NO
YES
PUT MINUS SIGN INTO BUFF
SET BUFF PTR TO BEG. OF EXPNT AREA
J 4
I →ACCA
DECREMENT BUFF PTR
REACHED BEG. OF MANT. AREA ?
YES
X-REG EXPNT →ACCA
EXPNT
NO
SCI3 MODE?
YES
SET SCI3 OVERFLOW FLAG
CLEAR FLOATING POINT FLAG
YES
RND≤7 ?
NO
SET ROUND OFF TO 7
A 2

FIG 70C

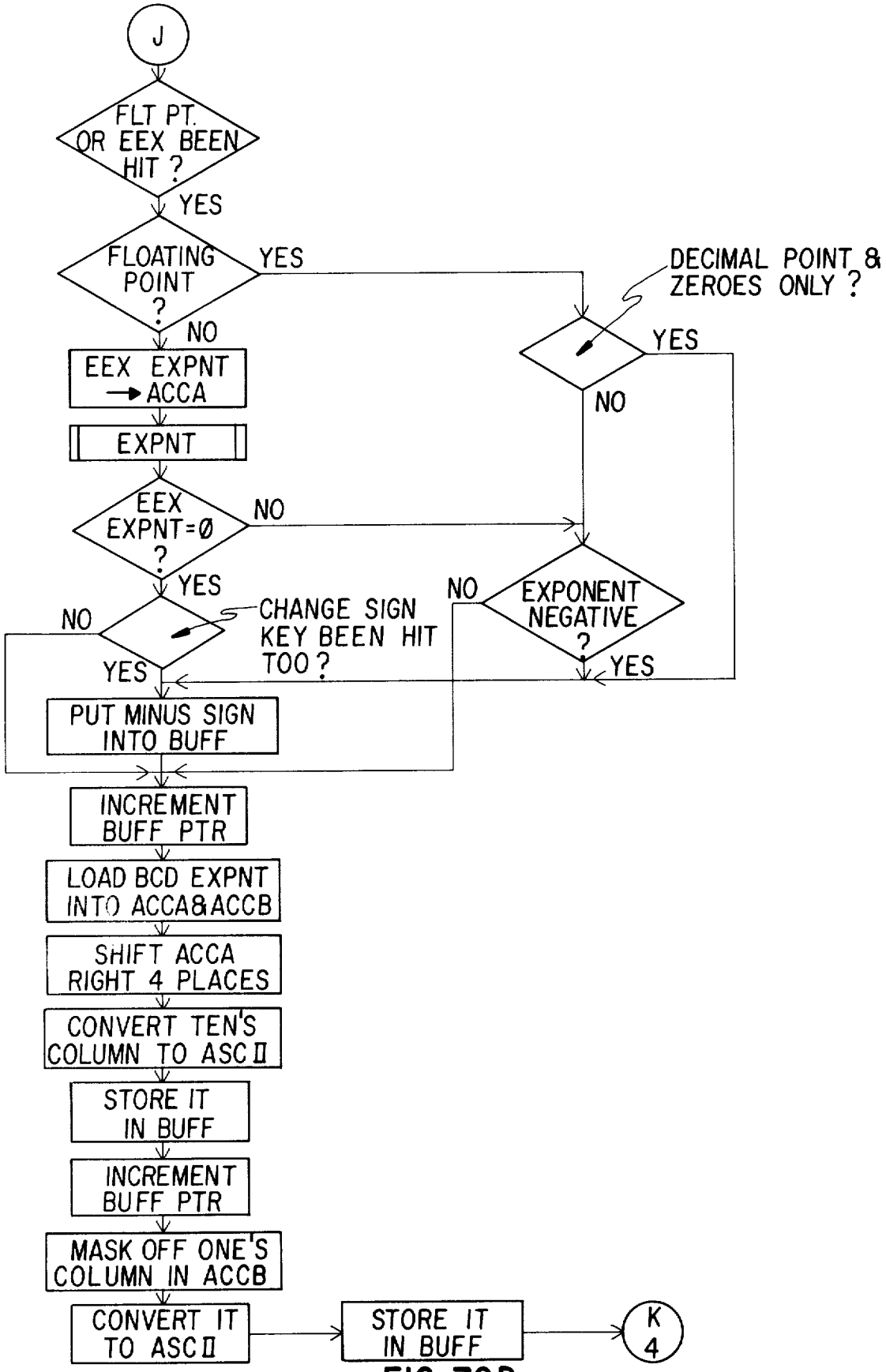

J
FLT PT. OR EEX BEEN HIT ?
YES
FLOATING POINT ?
YES
NO
DECIMAL POINT & ZEROES ONLY ?
YES
NO
EEX EXPNT → ACCA
EXPNT
EEX EXPNT=Ø ?
NO
YES
NO
CHANGE SIGN KEY BEEN HIT TOO ?
NO
EXPONENT NEGATIVE ?
YES
YES
PUT MINUS SIGN INTO BUFF
INCREMENT BUFF PTR
LOAD BCD EXPNT INTO ACCA&ACCB
SHIFT ACCA RIGHT 4 PLACES
CONVERT TEN'S COLUMN TO ASCII
STORE IT IN BUFF
INCREMENT BUFF PTR
MASK OFF ONE'S COLUMN IN ACCB
CONVERT IT TO ASCII
STORE IT IN BUFF
K 4

FIG 70D

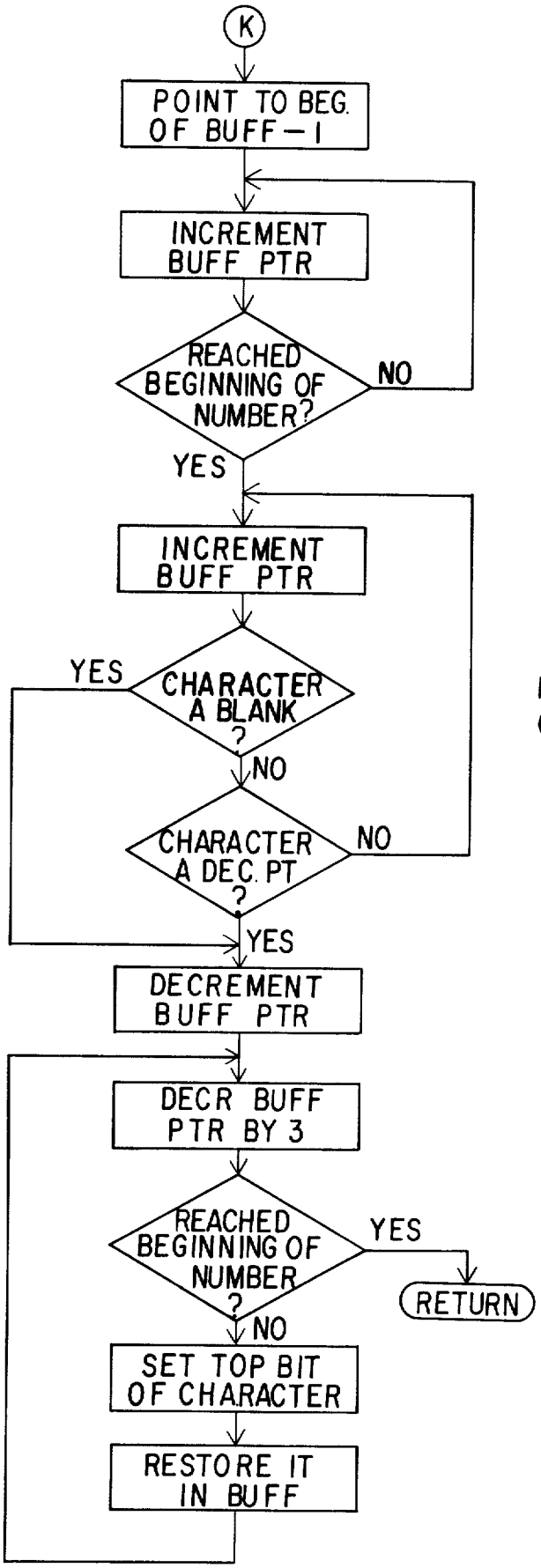

K
POINT TO BEG. OF BUFF-1
INCREMENT BUFF PTR
REACHED BEGINNING OF NUMBER?
NO
YES
INCREMENT BUFF PTR
YES
CHARACTER A BLANK ?
NO
CHARACTER A DEC. PT ?
NO
YES
DECREMENT BUFF PTR
DECR BUFF PTR BY 3
REACHED BEGINNING OF NUMBER ?
YES
RETURN
NO
SET TOP BIT OF CHARACTER
RESTORE IT IN BUFF
REACHED END OF NUMBER?

FIG 70E

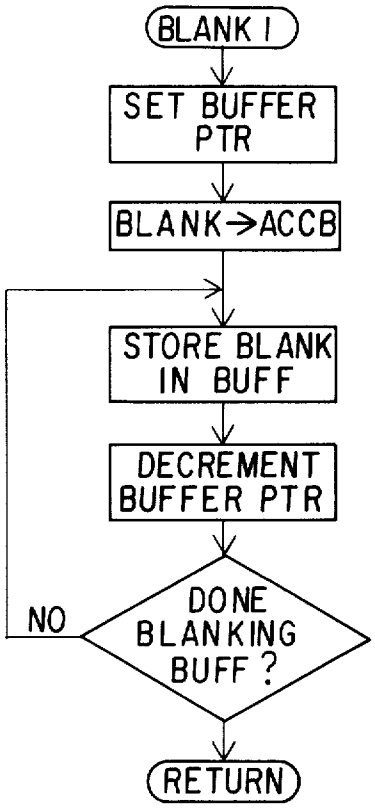


FIG 70F

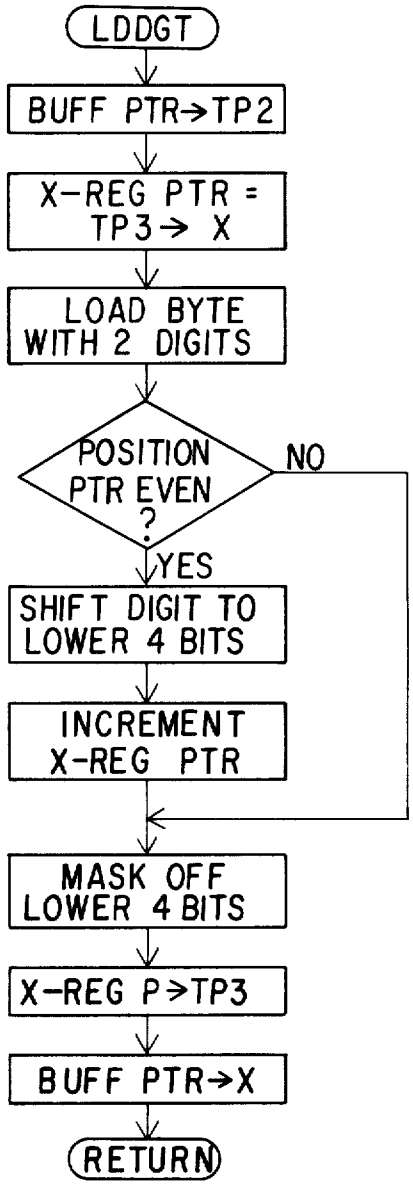


FIG 70G

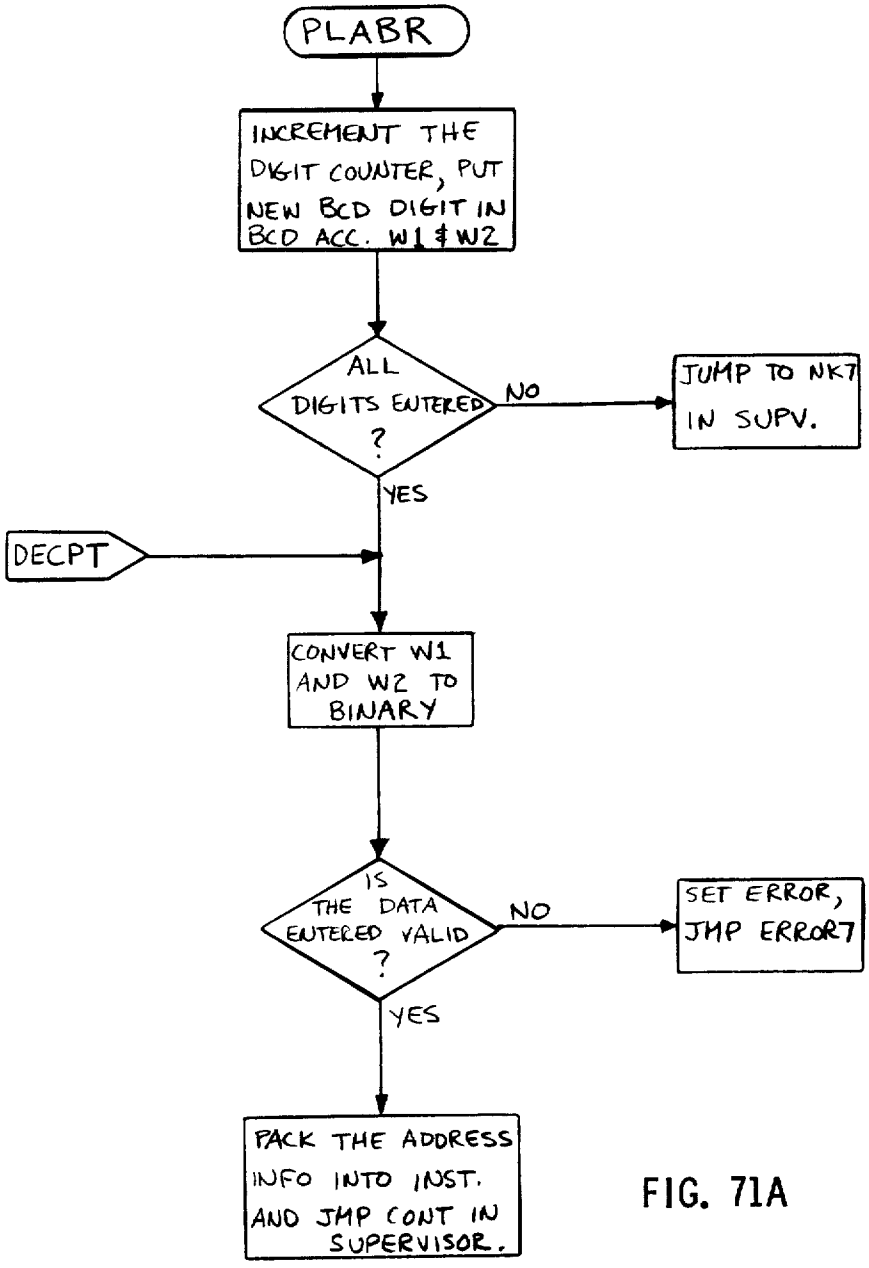

PLABR
INCREMENT THE DIGIT COUNTER, PUT NEW BCD DIGIT IN BCD ACC. W1 & W2
ALL DIGITS ENTERED ?
NO
JUMP TO NK7 IN SUPV.
YES
DECPT
CONVERT W1 AND W2 TO BINARY
IS THE DATA ENTERED VALID ?
NO
SET ERROR, JMP ERROR7
YES
PACK THE ADDRESS INFO INTO INST. AND JMP CONT IN SUPERVISOR.

FIG. 71A

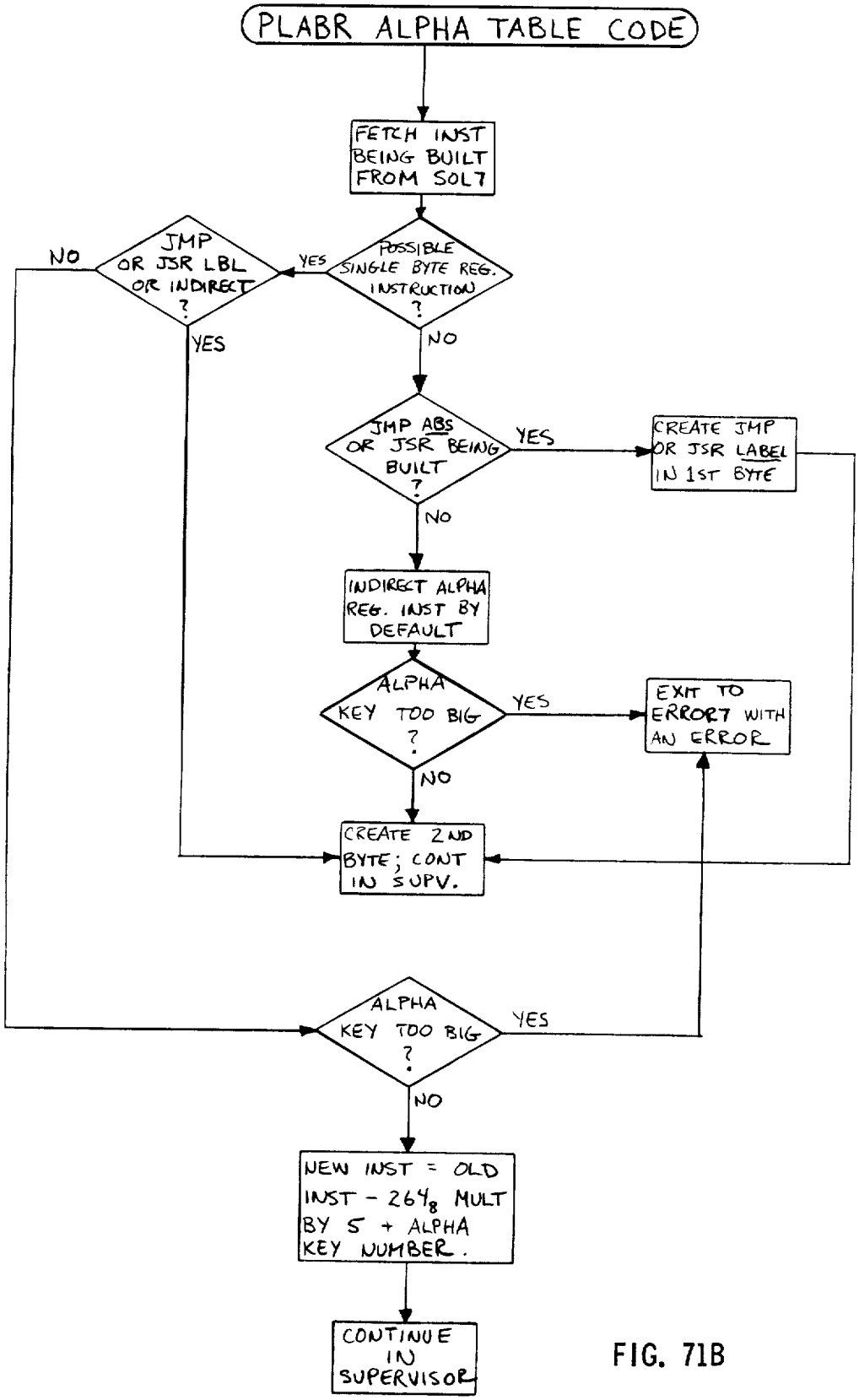

PLABR ALPHA TABLE CODE
FETCH INST BEING BUILT FROM SOLT
POSSIBLE SINGLE BYTE REG. INSTRUCTION ?
YES
JMP OR JSR LBL OR INDIRECT ?
NO
YES
NO
JMP ABS OR JSR BEING BUILT ?
YES
CREATE JMP OR JSR LABEL IN 1ST BYTE
NO
INDIRECT ALPHA REG. INST BY DEFAULT
ALPHA KEY TOO BIG ?
YES
EXIT TO ERRORT WITH AN ERROR
NO
CREATE 2ND BYTE; CONT IN SUPV.
ALPHA KEY TOO BIG ?
YES
NO
NEW INST = OLD INST - 264$_8$ MULT BY 5 + ALPHA KEY NUMBER.
CONTINUE IN SUPERVISOR

FIG. 71B

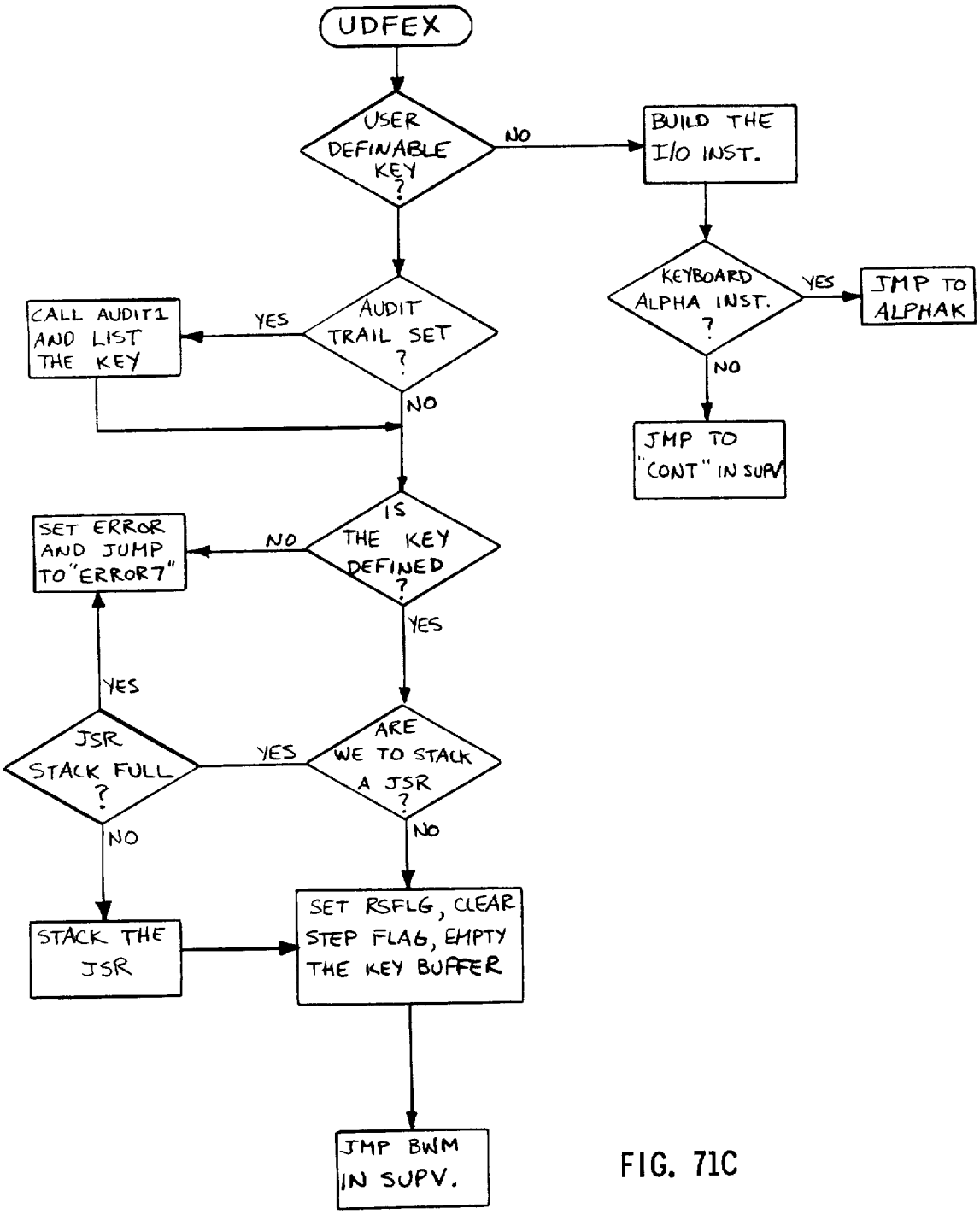

UDFEX
USER DEFINABLE KEY ?
NO
BUILD THE I/O INST.
KEYBOARD ALPHA INST. ?
YES
JMP TO ALPHAK
NO
JMP TO "CONT" IN SUPV
AUDIT TRAIL SET ?
YES
CALL AUDIT1 AND LIST THE KEY
NO
IS THE KEY DEFINED ?
NO
SET ERROR AND JUMP TO "ERROR7"
YES
ARE WE TO STACK A JSR ?
YES
JSR STACK FULL ?
YES
NO
NO
STACK THE JSR
SET RSFLG, CLEAR STEP FLAG, EMPTY THE KEY BUFFER
JMP BWM IN SUPV.

FIG. 71C

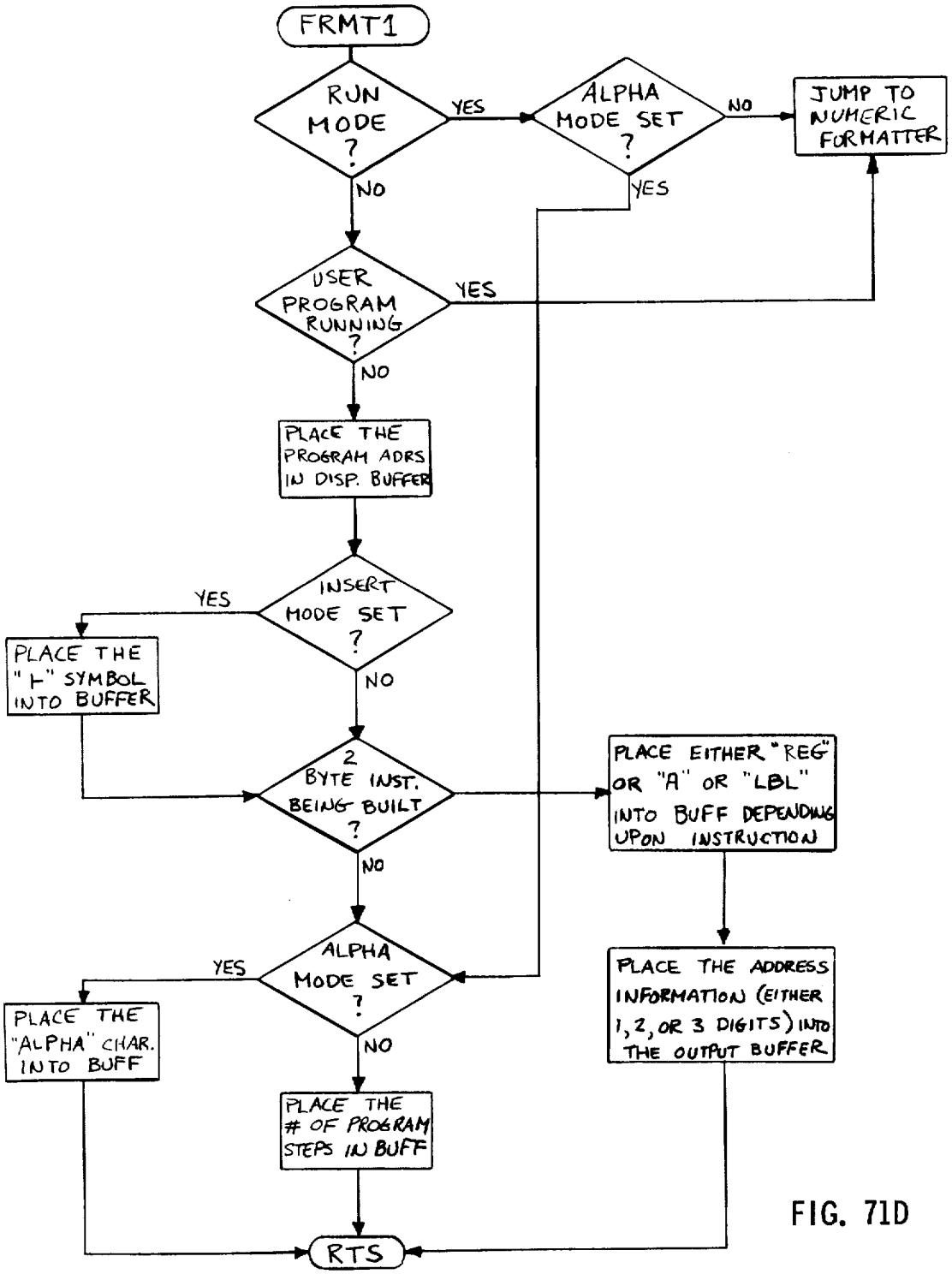

FRMT1
RUN MODE ?
YES
ALPHA MODE SET ?
NO
JUMP TO NUMERIC FORMATTER
NO
YES
USER PROGRAM RUNNING ?
YES
NO
PLACE THE PROGRAM ADRS IN DISP. BUFFER
INSERT MODE SET ?
YES
PLACE THE "⊢" SYMBOL INTO BUFFER
NO
2 BYTE INST. BEING BUILT ?
PLACE EITHER "REG" OR "A" OR "LBL" INTO BUFF DEPENDING UPON INSTRUCTION
NO
ALPHA MODE SET ?
YES
PLACE THE "ALPHA" CHAR. INTO BUFF
PLACE THE ADDRESS INFORMATION (EITHER 1, 2, OR 3 DIGITS) INTO THE OUTPUT BUFFER
NO
PLACE THE # OF PROGRAM STEPS IN BUFF
RTS

FIG. 71D

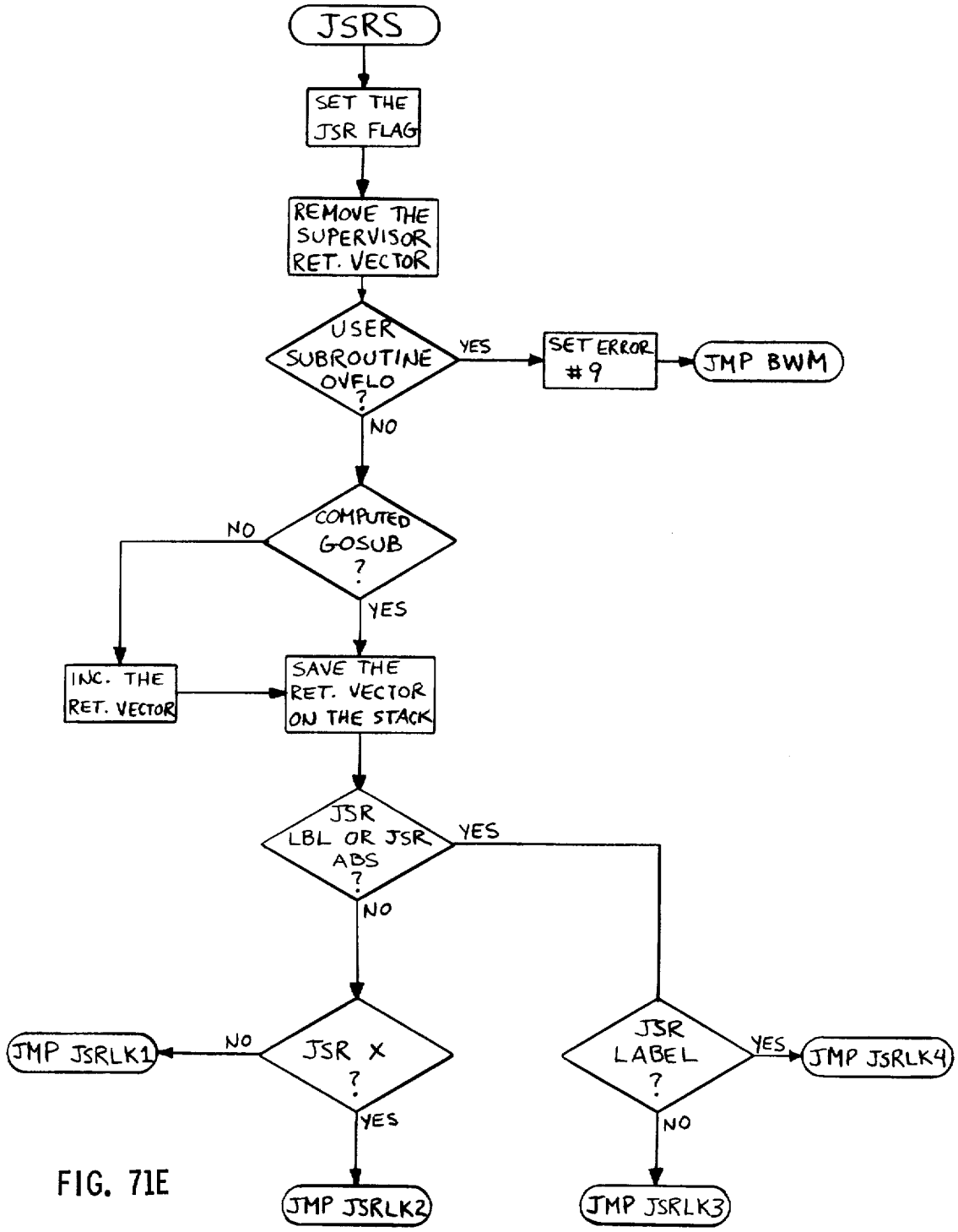

JSRS
SET THE JSR FLAG
REMOVE THE SUPERVISOR RET. VECTOR
USER SUBROUTINE OVFLO ?
YES
SET ERROR #9
JMP BWM
NO
COMPUTED GOSUB ?
NO
YES
INC. THE RET. VECTOR
SAVE THE RET. VECTOR ON THE STACK
JSR LBL OR JSR ABS ?
YES
NO
JMP JSRLK1
NO
JSR X ?
YES
JSR LABEL ?
YES
JMP JSRLK4
NO
JMP JSRLK2
JMP JSRLK3

FIG. 71E

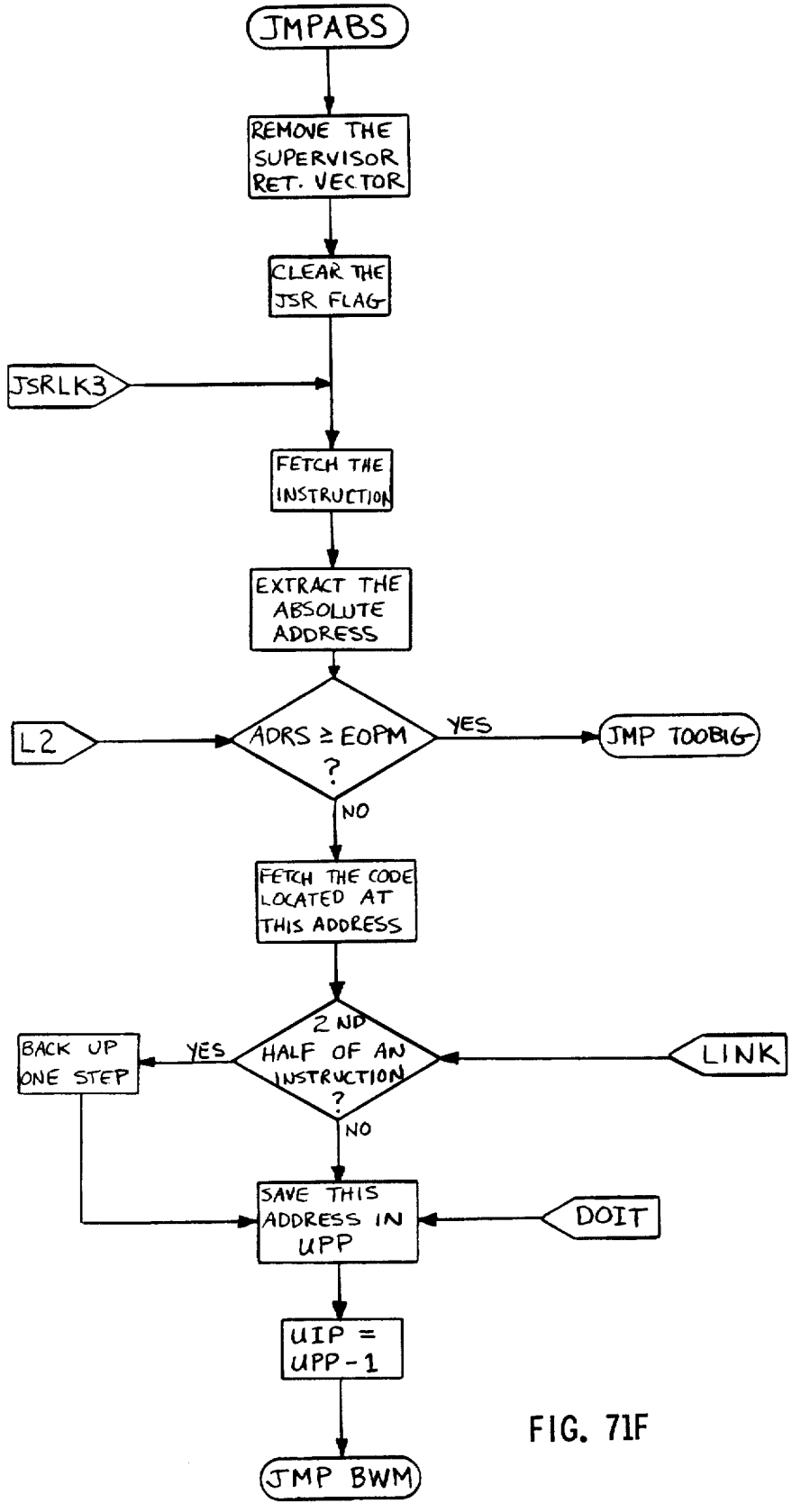

JMPABS
REMOVE THE SUPERVISOR RET. VECTOR
CLEAR THE JSR FLAG
JSRLK3
FETCH THE INSTRUCTION
EXTRACT THE ABSOLUTE ADDRESS
L2
ADRS ≥ EOPM ?
YES
JMP TOOBIG
NO
FETCH THE CODE LOCATED AT THIS ADDRESS
2 ND HALF OF AN INSTRUCTION ?
YES
BACK UP ONE STEP
LINK
NO
SAVE THIS ADDRESS IN UPP
DOIT
UIP = UPP-1
JMP BWM

FIG. 71F

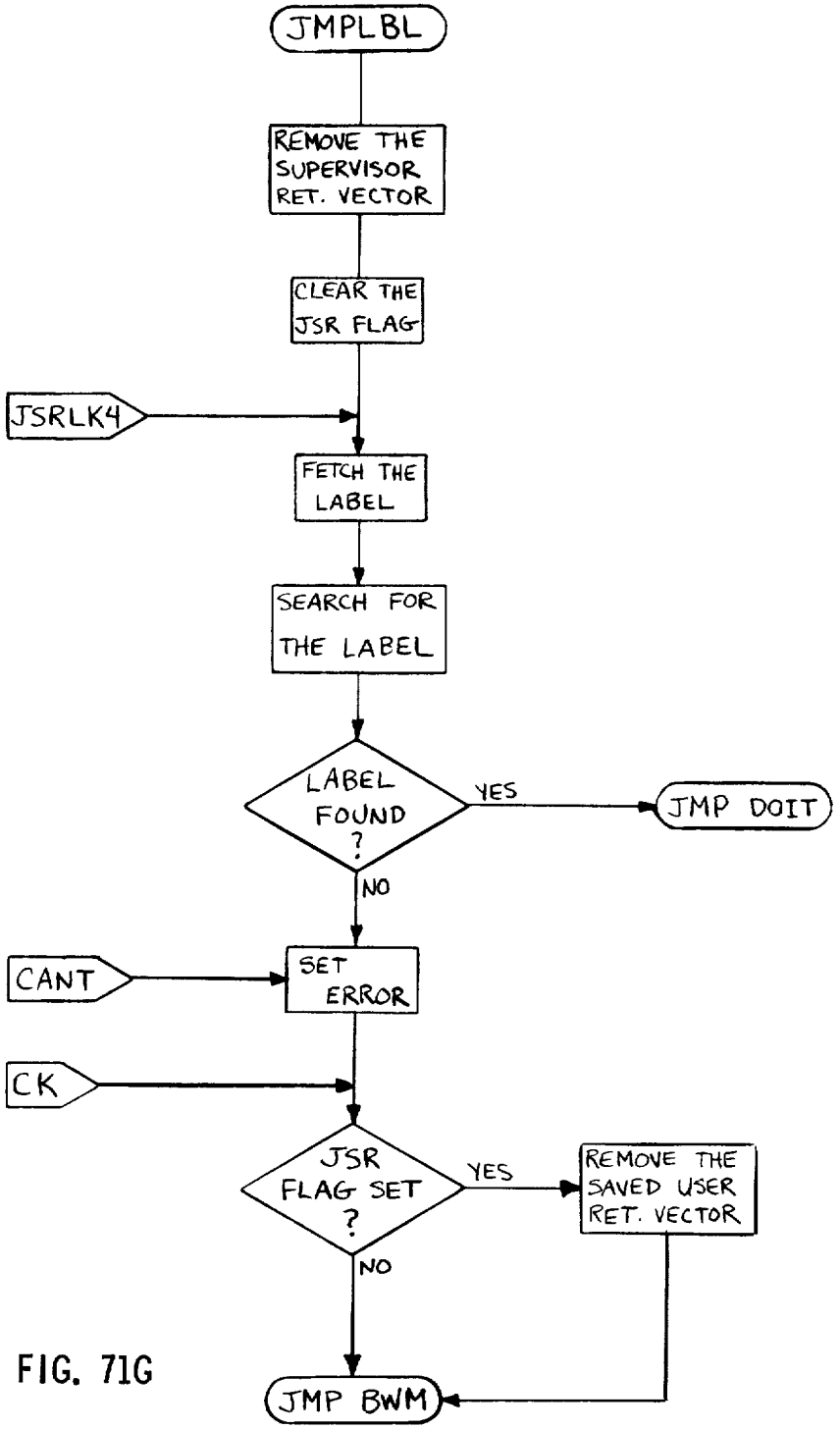

JMPLBL
REMOVE THE SUPERVISOR RET. VECTOR
CLEAR THE JSR FLAG
JSRLK4
FETCH THE LABEL
SEARCH FOR THE LABEL
LABEL FOUND ?
YES
JMP DOIT
NO
CANT
SET ERROR
CK
JSR FLAG SET ?
YES
REMOVE THE SAVED USER RET. VECTOR
NO
JMP BWM

FIG. 71G

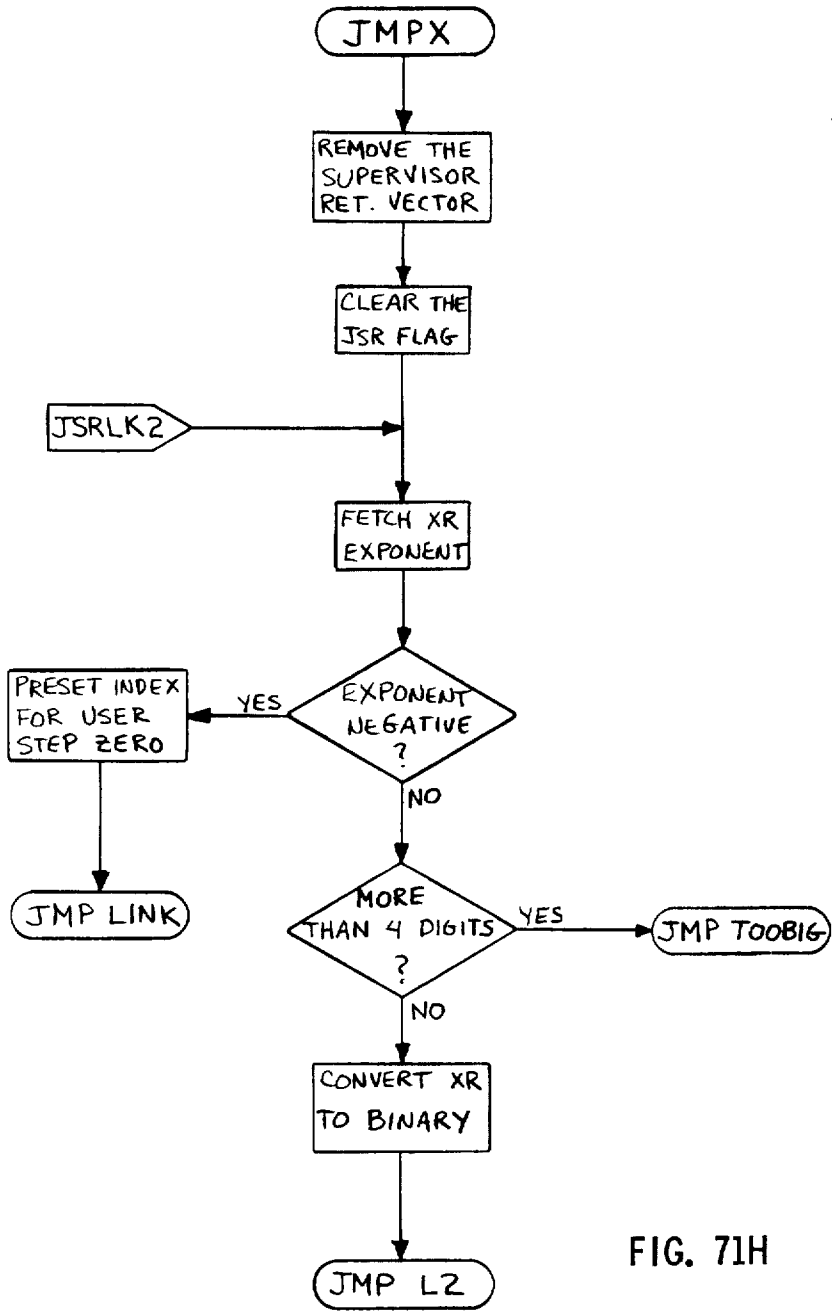

JMPX
REMOVE THE SUPERVISOR RET. VECTOR
CLEAR THE JSR FLAG
JSRLK2
FETCH XR EXPONENT
EXPONENT NEGATIVE ?
YES
PRESET INDEX FOR USER STEP ZERO
JMP LINK
NO
MORE THAN 4 DIGITS ?
YES
JMP TOOBIG
NO
CONVERT XR TO BINARY
JMP L2

FIG. 71H

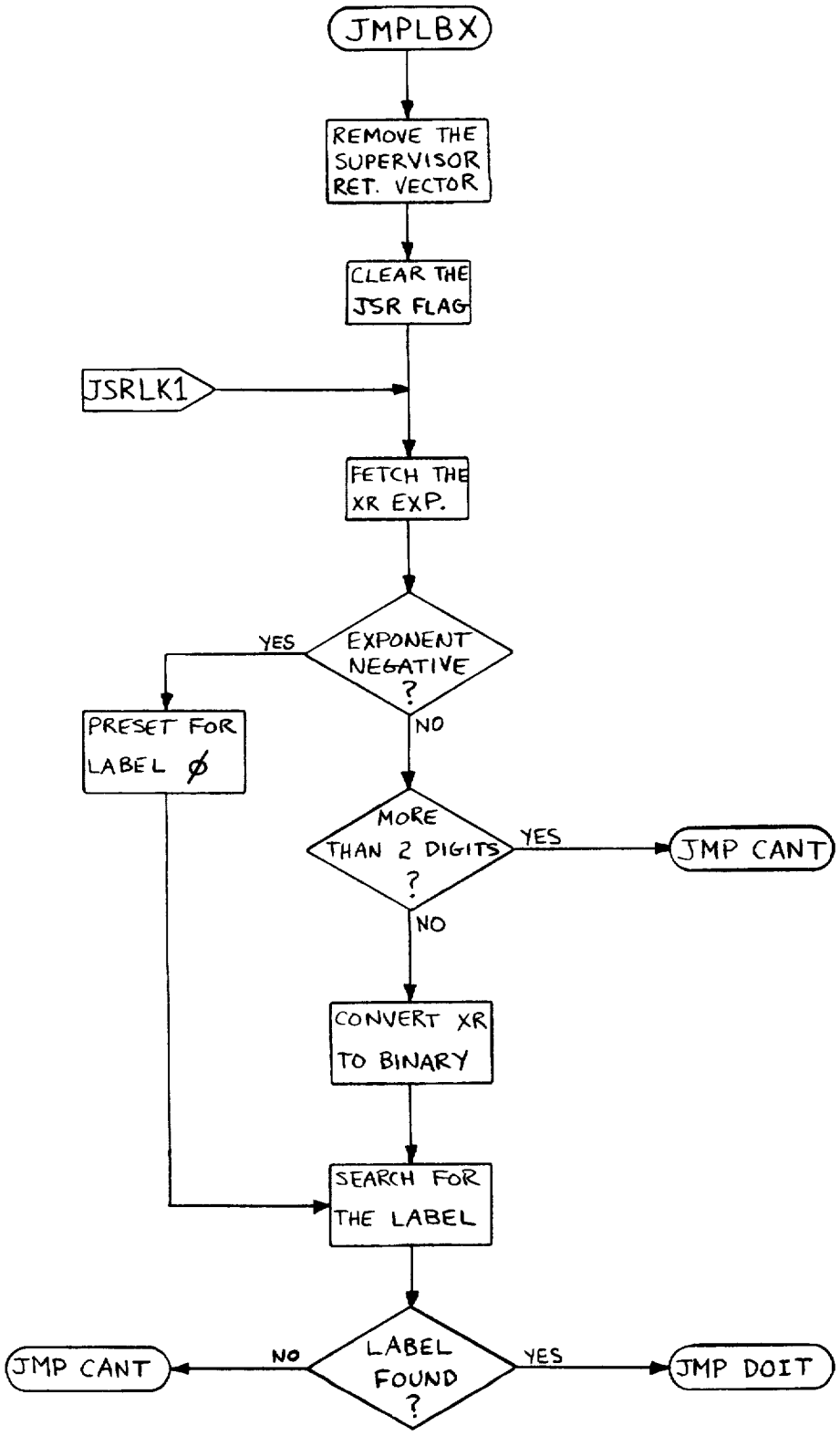

JMPLBX
REMOVE THE SUPERVISOR RET. VECTOR
CLEAR THE JSR FLAG
JSRLK1
FETCH THE XR EXP.
EXPONENT NEGATIVE ?
YES
NO
PRESET FOR LABEL ∅
MORE THAN 2 DIGITS ?
YES
JMP CANT
NO
CONVERT XR TO BINARY
SEARCH FOR THE LABEL
LABEL FOUND ?
NO
JMP CANT
YES
JMP DOIT

FIG. 71 I

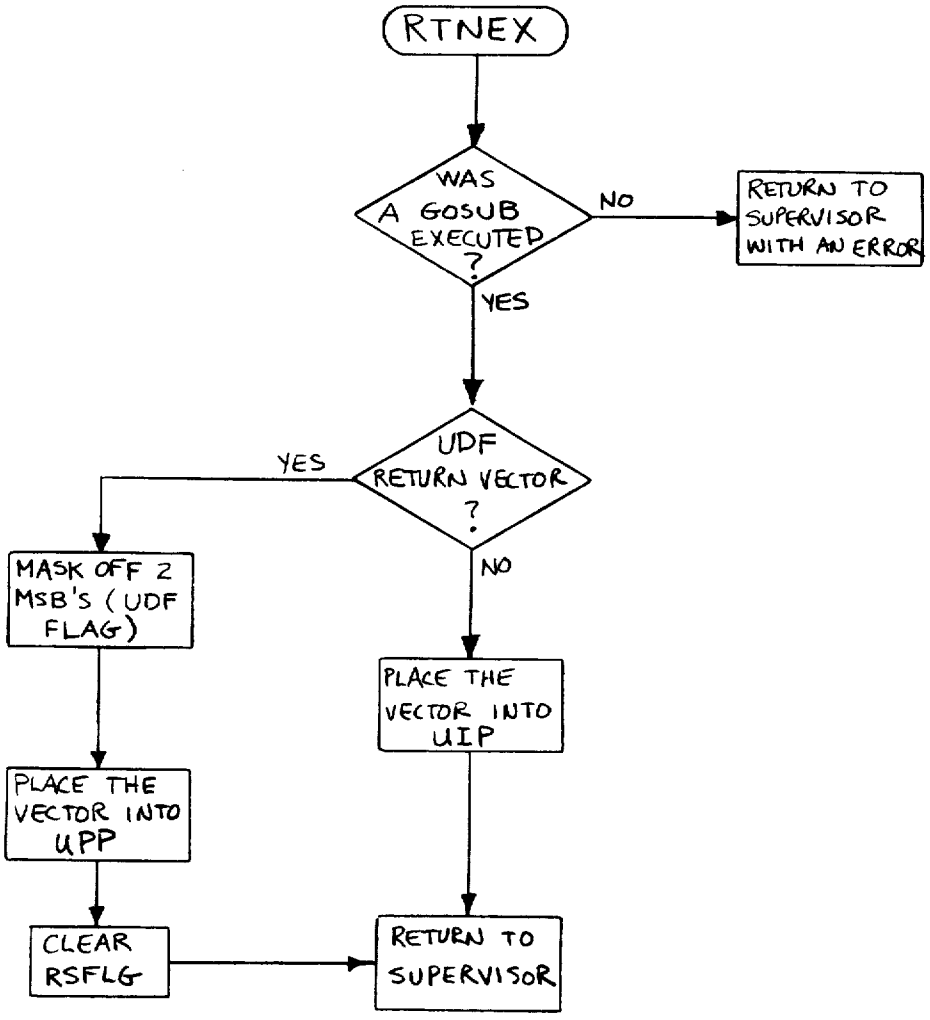

RTNEX
WAS A GOSUB EXECUTED ?
NO
RETURN TO SUPERVISOR WITH AN ERROR
YES
UDF RETURN VECTOR ?
YES
NO
MASK OFF 2 MSB'S (UDF FLAG)
PLACE THE VECTOR INTO UPP
CLEAR RSFLG
PLACE THE VECTOR INTO UIP
RETURN TO SUPERVISOR

FIG. 71J

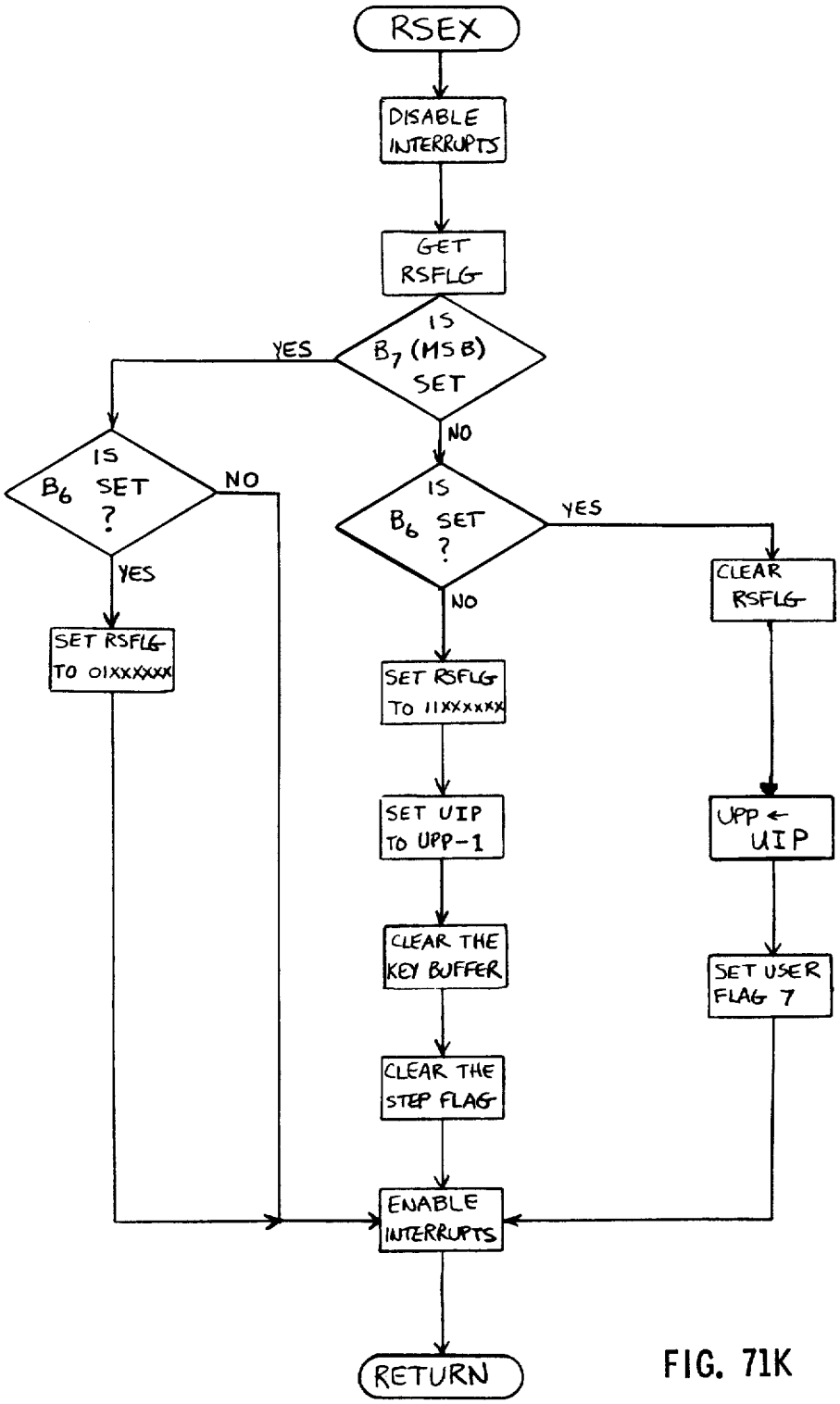

RSEX
DISABLE INTERRUPTS
GET RSFLG
IS B_7 (MSB) SET
YES
NO
IS B_6 SET ?
NO
YES
SET RSFLG TO 01XXXXXX
IS B_6 SET ?
YES
NO
CLEAR RSFLG
SET RSFLG TO 11XXXXXX
SET UIP TO UPP-1
UPP ← UIP
CLEAR THE KEY BUFFER
SET USER FLAG 7
CLEAR THE STEP FLAG
ENABLE INTERRUPTS
RETURN

FIG. 71K

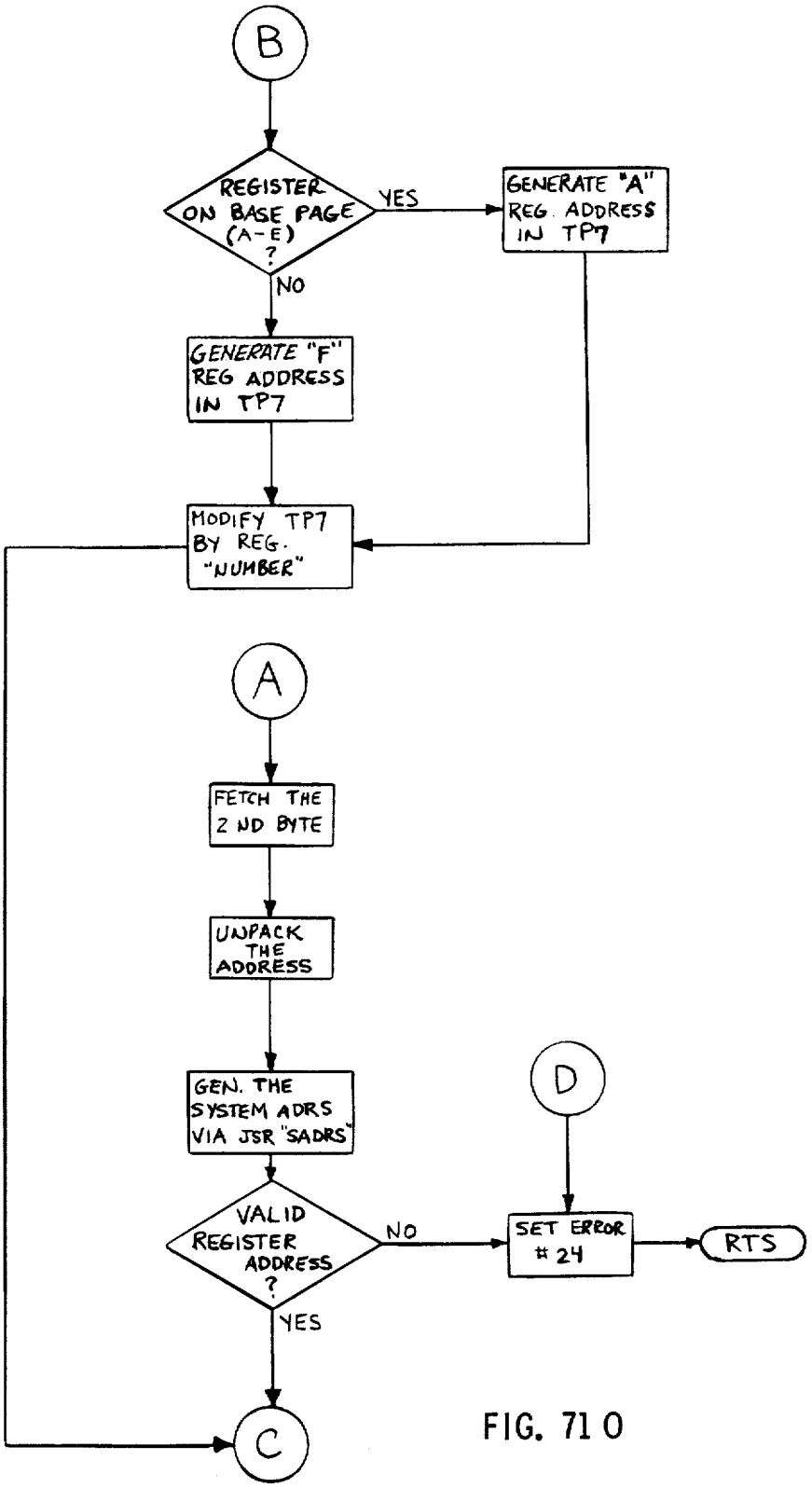

B
REGISTER ON BASE PAGE (A-E) ?
YES
GENERATE "A" REG. ADDRESS IN TP7
NO
GENERATE "F" REG ADDRESS IN TP7
MODIFY TP7 BY REG. "NUMBER"
A
FETCH THE 2 ND BYTE
UNPACK THE ADDRESS
GEN. THE SYSTEM ADRS VIA JSR "SADRS"
D
VALID REGISTER ADDRESS ?
NO
SET ERROR # 24
RTS
YES
C

FIG. 71 0

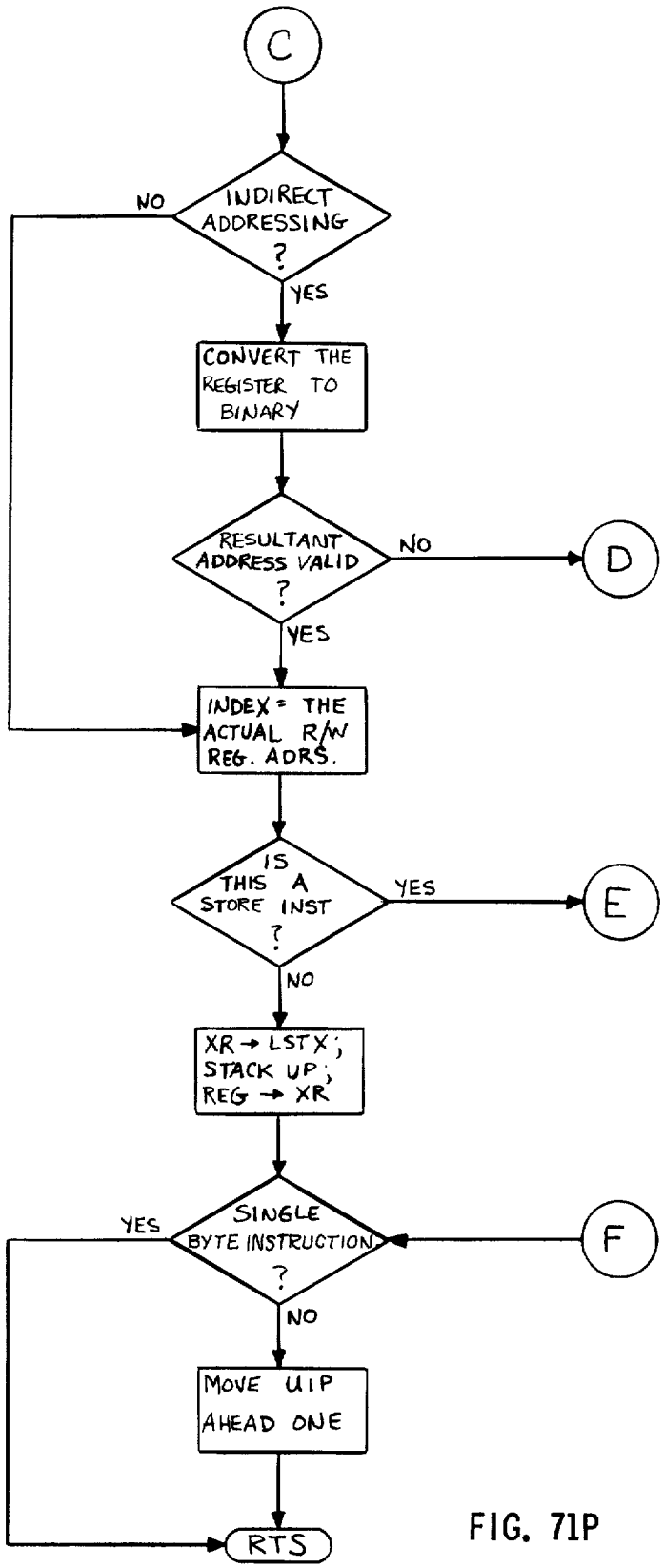

C
INDIRECT ADDRESSING ?
NO
YES
CONVERT THE REGISTER TO BINARY
RESULTANT ADDRESS VALID ?
NO
D
YES
INDEX = THE ACTUAL R/W REG. ADRS.
IS THIS A STORE INST ?
YES
E
NO
XR → LSTX; STACK UP; REG → XR
SINGLE BYTE INSTRUCTION ?
YES
F
NO
MOVE UIP AHEAD ONE
RTS

FIG. 71P

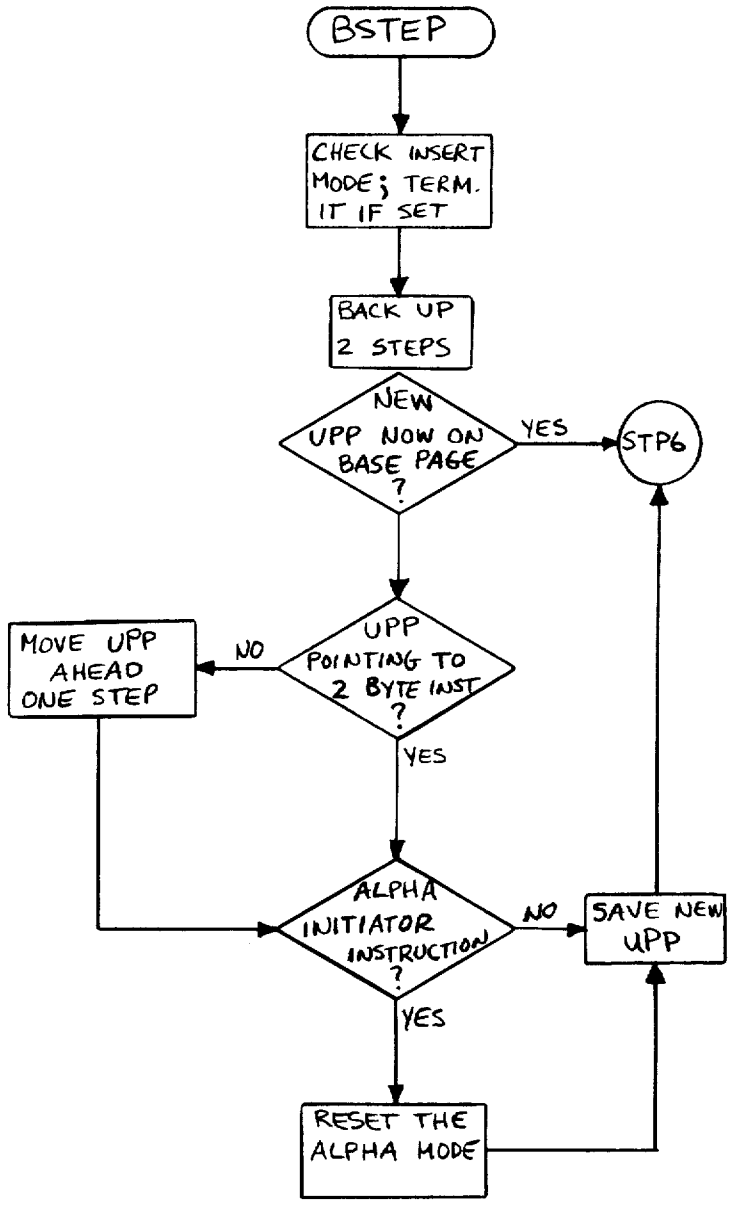

BSTEP
CHECK INSERT MODE; TERM. IT IF SET
BACK UP 2 STEPS
NEW UPP NOW ON BASE PAGE ?
YES
STP6
UPP POINTING TO 2 BYTE INST ?
NO
MOVE UPP AHEAD ONE STEP
YES
ALPHA INITIATOR INSTRUCTION ?
NO
SAVE NEW UPP
YES
RESET THE ALPHA MODE

FIG. 71X

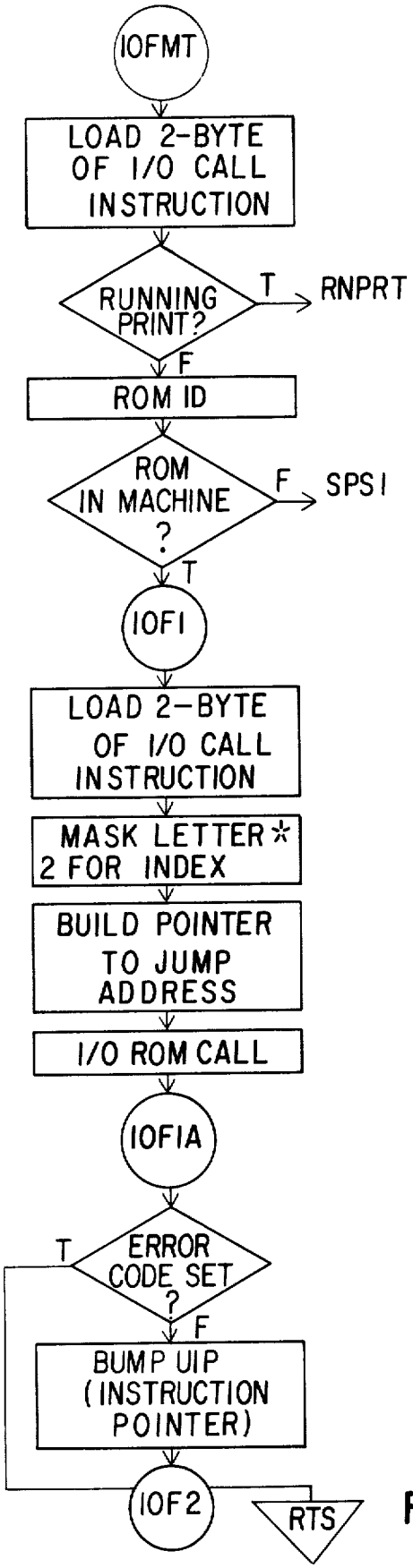


FIG 72A

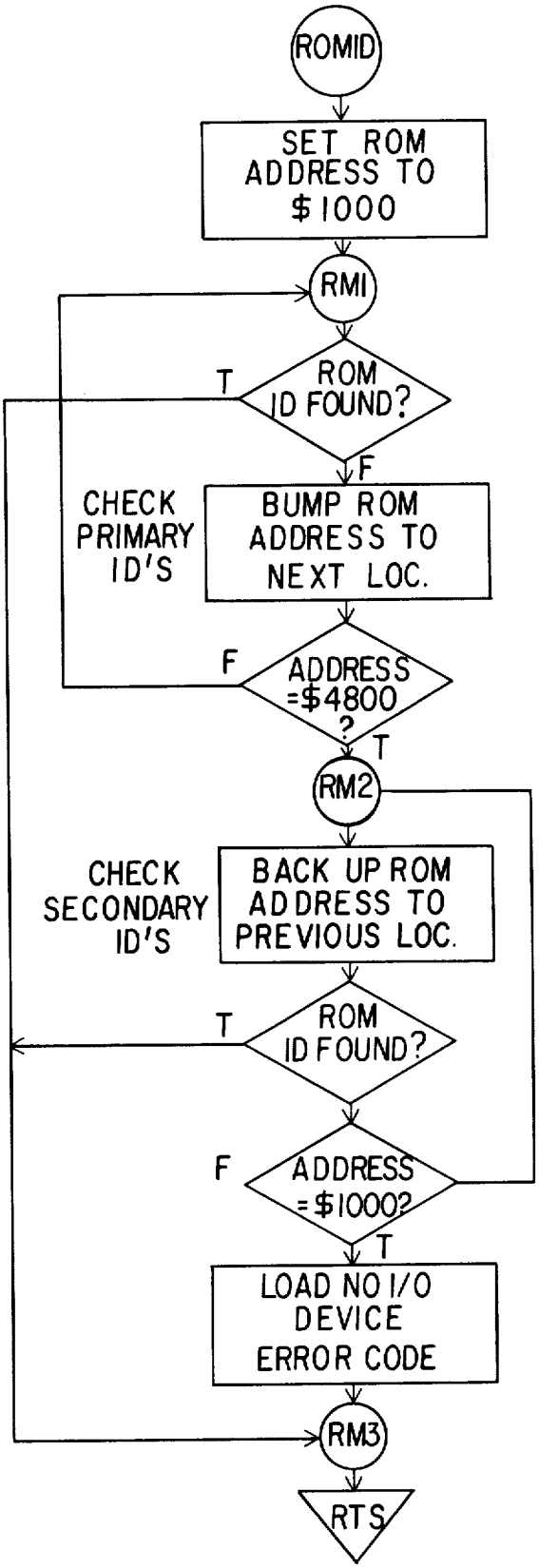


FIG 72B

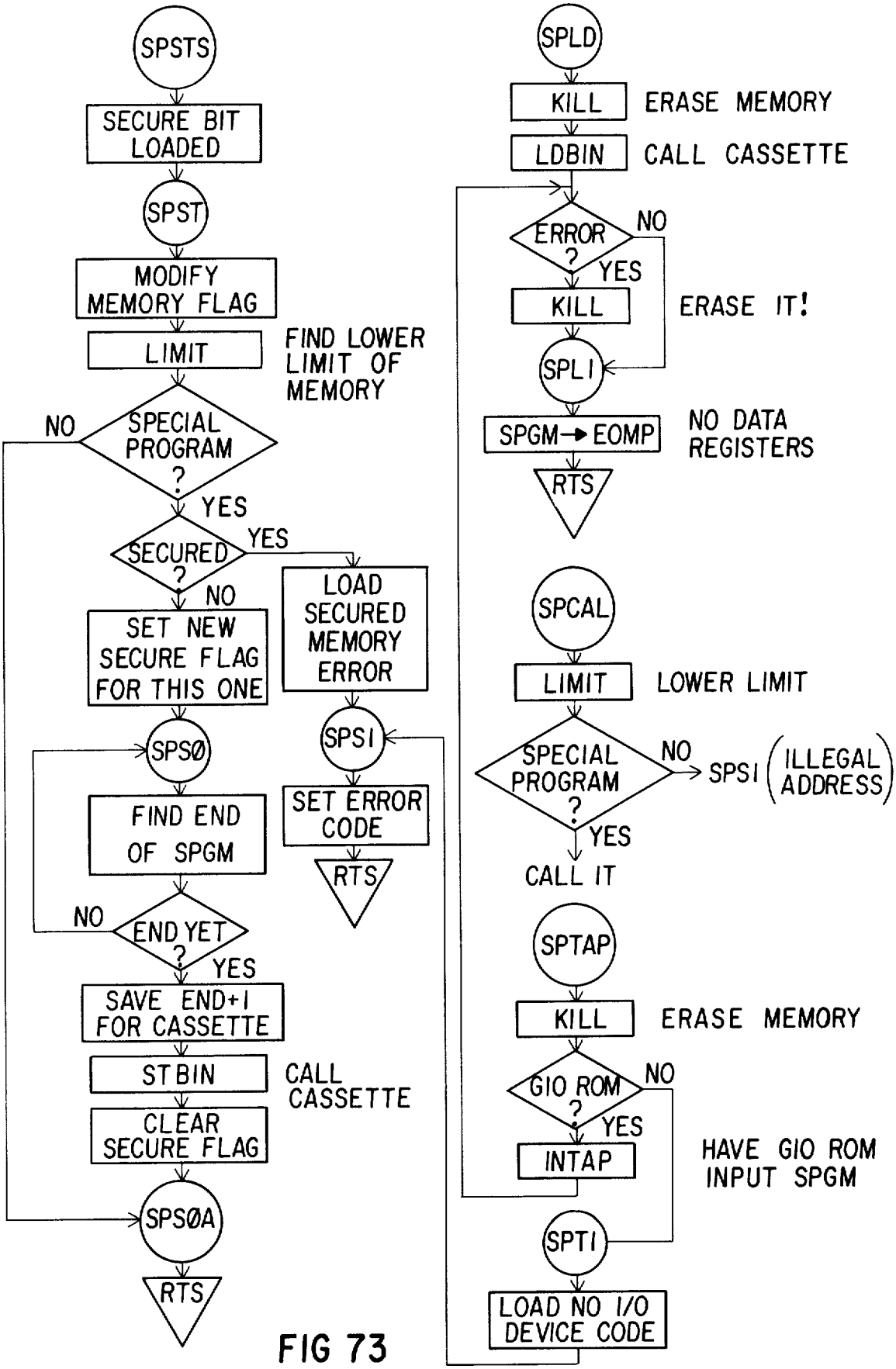

SPSTS
SECURE BIT LOADED
SPST
MODIFY MEMORY FLAG
LIMIT
FIND LOWER LIMIT OF MEMORY
SPECIAL PROGRAM ?
NO
YES
SECURED ?
YES
NO
SET NEW SECURE FLAG FOR THIS ONE
LOAD SECURED MEMORY ERROR
SPSØ
SPSI
FIND END OF SPGM
SET ERROR CODE
RTS
END YET ?
NO
YES
SAVE END+1 FOR CASSETTE
STBIN
CALL CASSETTE
CLEAR SECURE FLAG
SPSØA
RTS
SPLD
KILL
ERASE MEMORY
LDBIN
CALL CASSETTE
ERROR ?
NO
YES
KILL
ERASE IT!
SPLI
SPGM→EOMP
NO DATA REGISTERS
RTS
SPCAL
LIMIT
LOWER LIMIT
SPECIAL PROGRAM ?
NO
SPSI (ILLEGAL ADDRESS)
YES
CALL IT
SPTAP
KILL
ERASE MEMORY
GIO ROM ?
NO
YES
INTAP
HAVE GIO ROM INPUT SPGM
SPTI
LOAD NO I/O DEVICE CODE

FIG 73

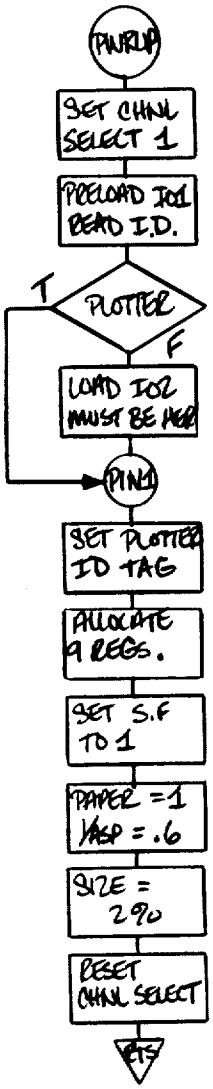

PWRUP
SET CHNL SELECT 1
PRELOAD IO1 READ I.D.
PLOTTER
T
F
LOAD IO2 MUST BE HERE
PIN1
SET PLOTTER ID TAG
ALLOCATE 9 REGS.
SET S.F TO 1
PAPER = 1
ASP = .6
SIZE = 29
RESET CHNL SELECT
ETS

FIG. 74A

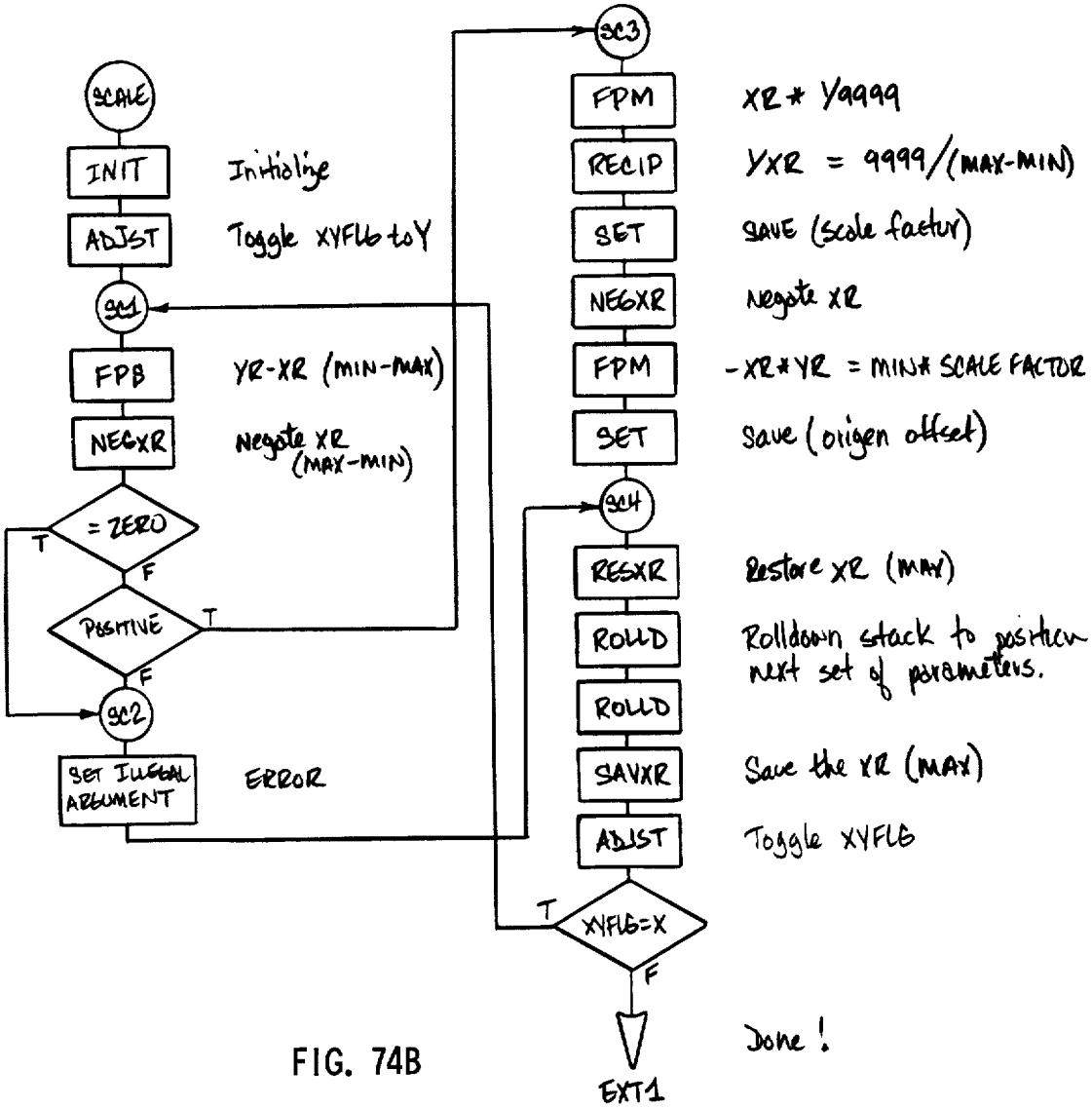

SCALE
INIT
Initialize
ADJST
Toggle XYFLG to Y
SC1
FPB
YR-XR (MIN-MAX)
NEGXR
Negate XR (MAX-MIN)
= ZERO
T
F
POSITIVE
T
F
SC2
SET ILLEGAL ARGUMENT
ERROR
SC3
FPM
XR * Y9999
RECIP
YXR = 9999/(MAX-MIN)
SET
SAVE (scale factor)
NEGXR
Negate XR
FPM
-XR*YR = MIN* SCALE FACTOR
SET
Save (origen offset)
SC4
RESXR
Restore XR (MAX)
ROLLD
Rolldown stack to position next set of parameters.
ROLLD
SAVXR
Save the XR (MAX)
ADJST
Toggle XYFLG
T
XYFLG=X
F
Done !
EXT1

FIG. 74B

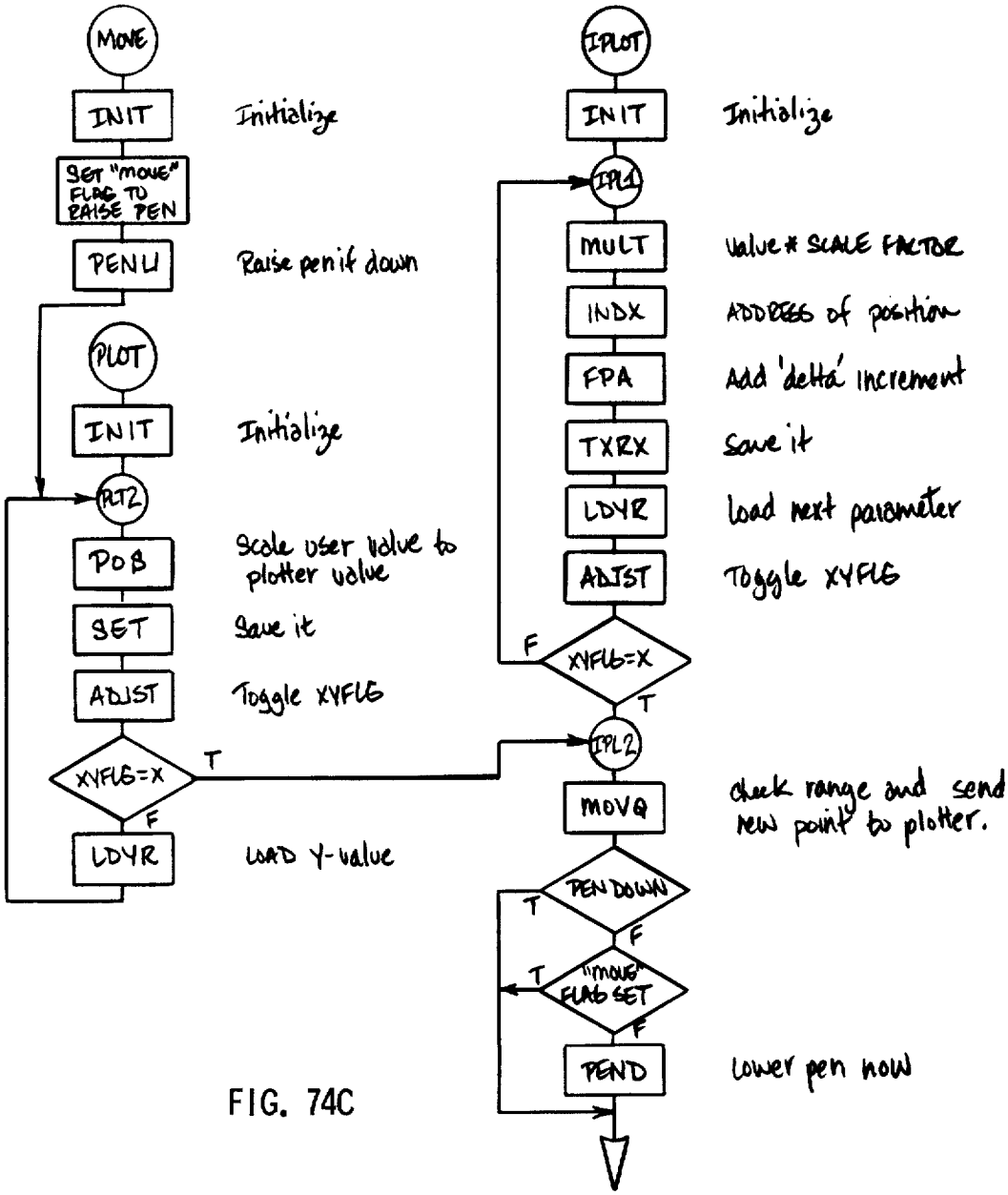

MOVE
INIT
Initialize
SET "move" FLAG TO RAISE PEN
PENU
Raise pen if down
PLOT
INIT
Initialize
PLT2
POB
Scale user value to plotter value
SET
Save it
ADJST
Toggle XYFLG
XYFLG=X
T
F
LDYR
Load Y-value
IPLOT
INIT
Initialize
IPL1
MULT
Value * SCALE FACTOR
INDX
ADDRESS of position
FPA
Add 'delta' increment
TXRX
Save it
LDYR
Load next parameter
ADJST
Toggle XYFLG
F
XYFLG=X
T
IPL2
MOVQ
check range and send new point to plotter.
PEN DOWN
T
F
"move" FLAG SET
T
F
PEND
Lower pen now

FIG. 74C

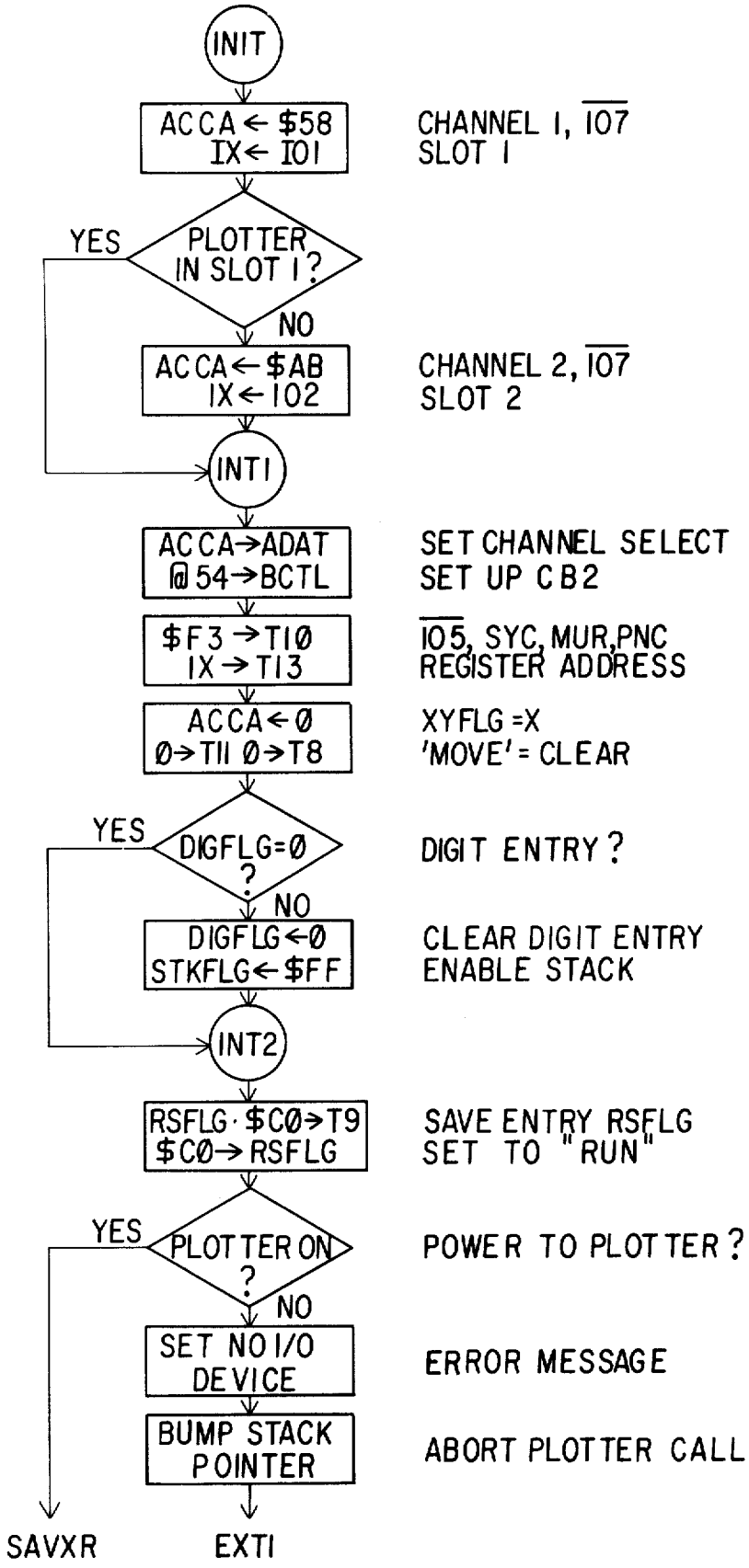
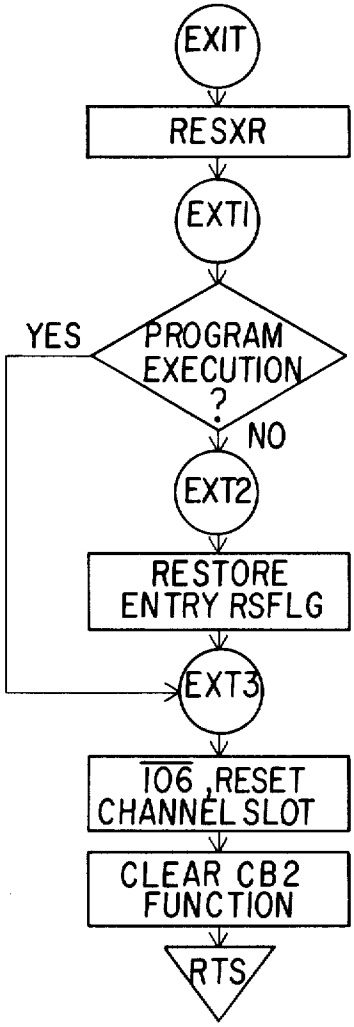

INIT
ACCA ← $58
IX ← I01
CHANNEL 1, $\overline{I07}$
SLOT 1
YES
PLOTTER IN SLOT 1?
NO
ACCA ← $AB
IX ← I02
CHANNEL 2, $\overline{I07}$
SLOT 2
INT1
ACCA → ADAT
@54 → BCTL
SET CHANNEL SELECT
SET UP CB2
$F3 → TI0
IX → TI3
$\overline{I05}$, SYC, MUR, PNC
REGISTER ADDRESS
ACCA ← 0
0 → TI1 0 → T8
XYFLG = X
'MOVE' = CLEAR
YES
DIGFLG=0 ?
DIGIT ENTRY?
NO
DIGFLG ← 0
STKFLG ← $FF
CLEAR DIGIT ENTRY
ENABLE STACK
INT2
RSFLG · $C0 → T9
$C0 → RSFLG
SAVE ENTRY RSFLG
SET TO "RUN"
YES
PLOTTER ON ?
POWER TO PLOTTER?
NO
SET NO I/O DEVICE
ERROR MESSAGE
BUMP STACK POINTER
ABORT PLOTTER CALL
SAVXR
EXTI

FIG 74D



RESTORE ENTRY XR VALUE

FIG 74E

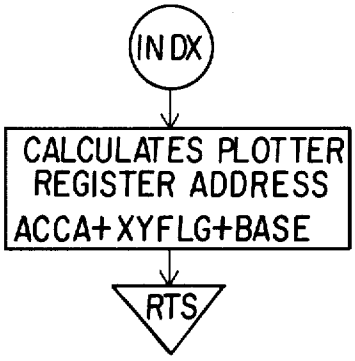


FIG 74F

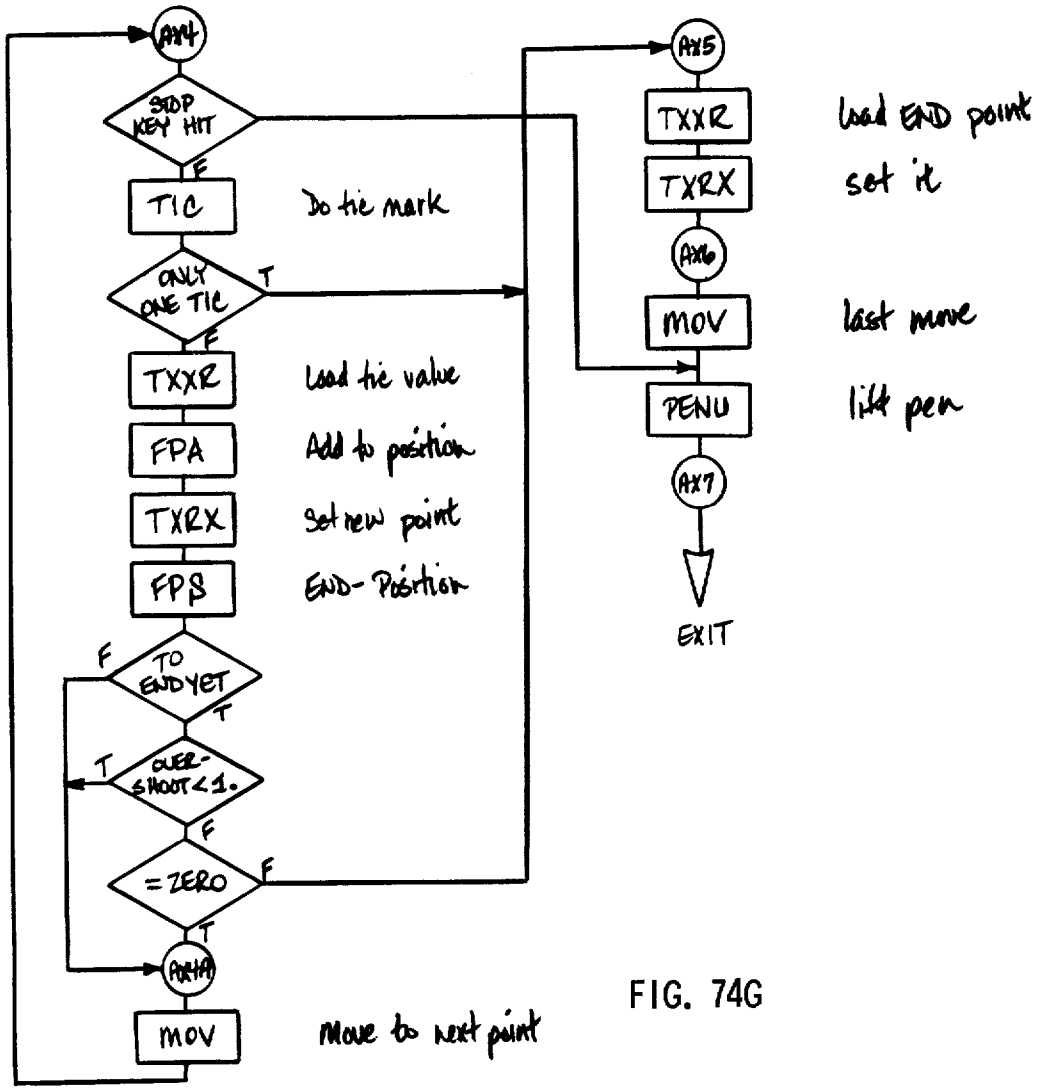

AX4
STOP KEY HIT
F
TIC
Do tic mark
ONLY ONE TIC
T
F
TXXR
Load tic value
FPA
Add to position
TXRX
Set new point
FPS
END-Position
F
TO END YET
T
T
OVER-SHOOT < 1.
F
= ZERO
F
T
AX4A
MOV
Move to next point
AX5
TXXR
TXRX
AX6
MOV
PENU
AX7
EXIT
Load END point
set it
last move
lift pen

FIG. 74G

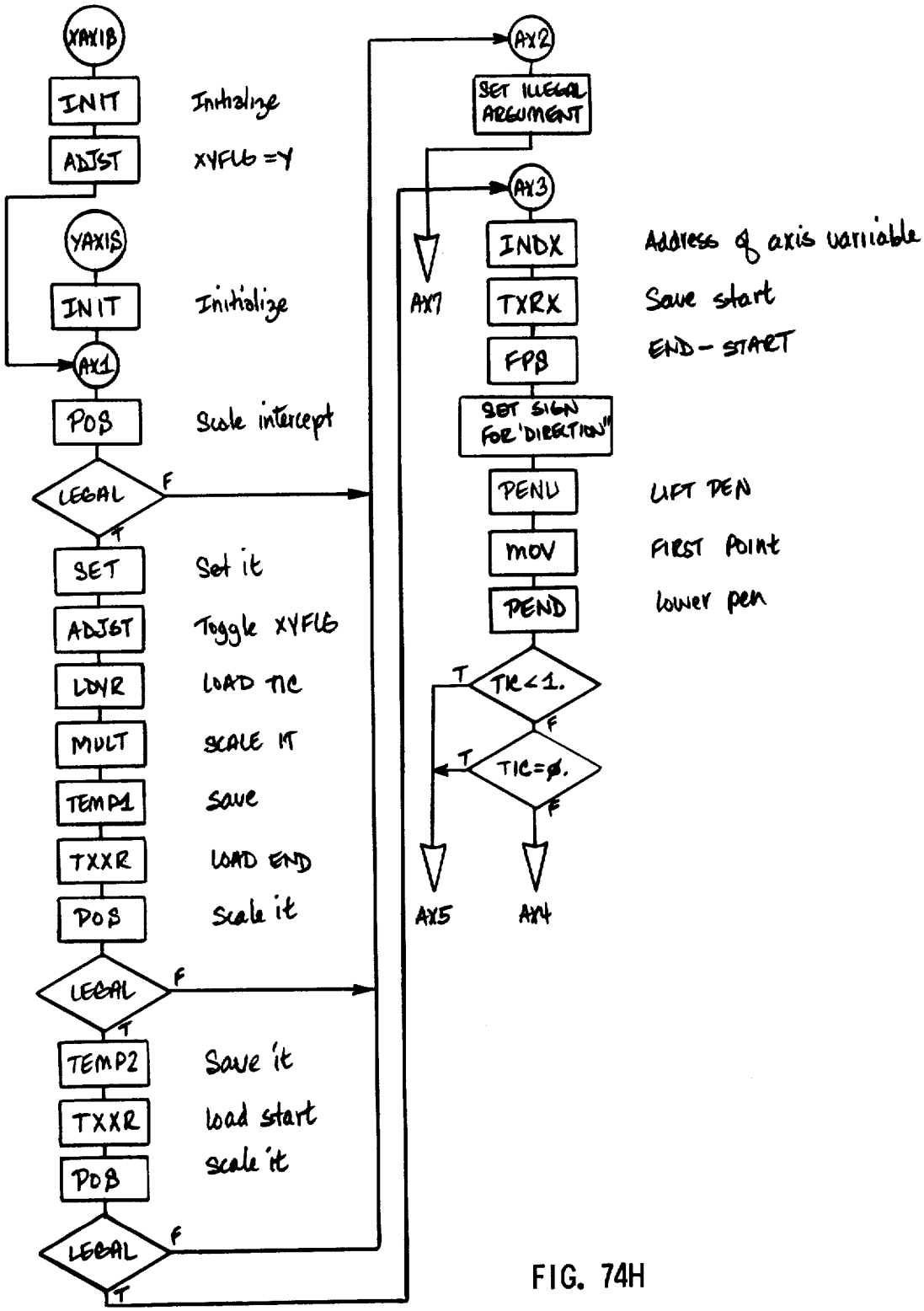

XAXIB
INIT
Initialize
ADJST
XYFLG =Y
YAXIS
INIT
Initialize
AX1
POS
Scale intercept
LEGAL
F
T
SET
Set it
ADJST
Toggle XYFLG
LDVR
LOAD TIC
MULT
SCALE IT
TEMP1
Save
TXXR
LOAD END
POS
Scale it
LEGAL
F
T
TEMP2
Save it
TXXR
load start
POS
scale it
LEGAL
F
T
AX2
SET ILLEGAL ARGUMENT
AX3
INDX
Address of axis variable
TXRX
Save start
FPS
END - START
SET SIGN FOR "DIRECTION"
PENU
LIFT PEN
MOV
FIRST POINT
PEND
lower pen
TIC<1.
T
F
TIC=0.
T
F
AX7
AX5
AX4

FIG. 74H

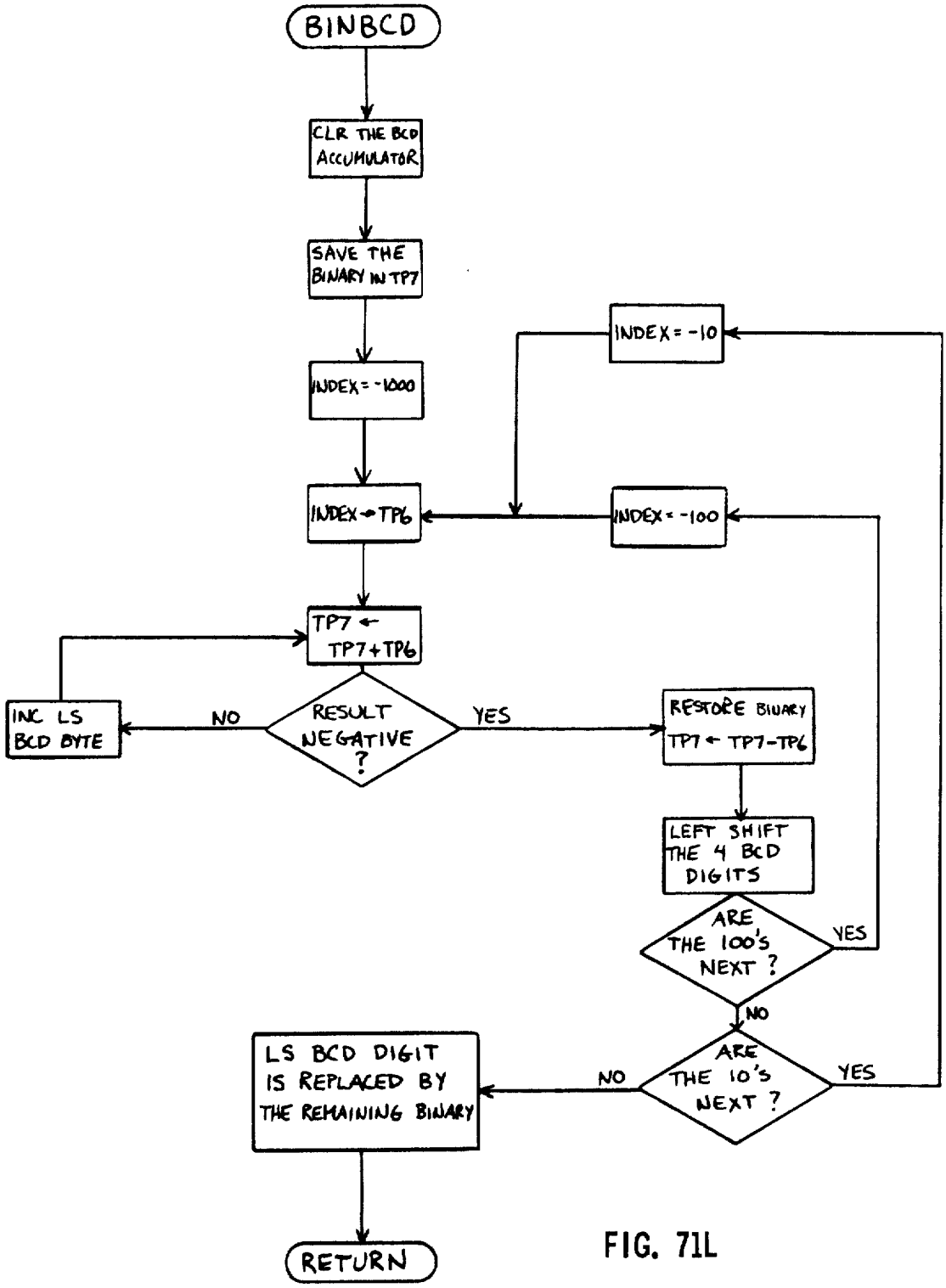

BINBCD
CLR THE BCD ACCUMULATOR
SAVE THE BINARY IN TP7
INDEX = -1000
INDEX → TP6
TP7 ← TP7+TP6
RESULT NEGATIVE ?
NO
YES
INC LS BCD BYTE
RESTORE BINARY TP7 ← TP7-TP6
LEFT SHIFT THE 4 BCD DIGITS
ARE THE 100's NEXT ?
YES
NO
INDEX = -100
ARE THE 10's NEXT ?
NO
YES
INDEX = -10
LS BCD DIGIT IS REPLACED BY THE REMAINING BINARY
RETURN

FIG. 71L

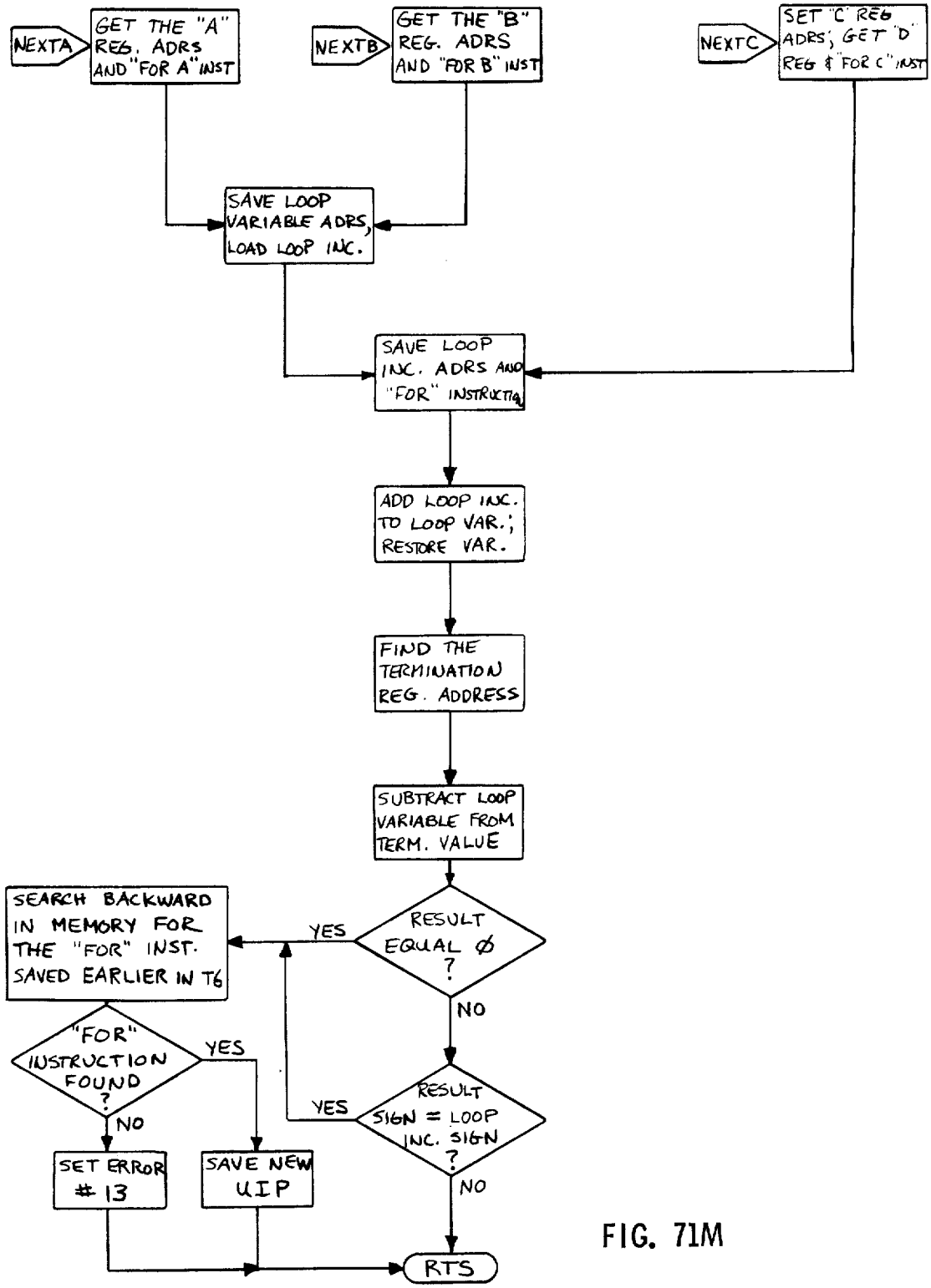

NEXTA
GET THE "A" REG. ADRS AND "FOR A" INST
NEXTB
GET THE "B" REG. ADRS AND "FOR B" INST
NEXTC
SET "C" REG ADRS; GET "D" REG & "FOR C" INST
SAVE LOOP VARIABLE ADRS, LOAD LOOP INC.
SAVE LOOP INC. ADRS AND "FOR" INSTRUCTN
ADD LOOP INC. TO LOOP VAR.; RESTORE VAR.
FIND THE TERMINATION REG. ADDRESS
SUBTRACT LOOP VARIABLE FROM TERM. VALUE
RESULT EQUAL Ø ?
YES
NO
SEARCH BACKWARD IN MEMORY FOR THE "FOR" INST. SAVED EARLIER IN T6
"FOR" INSTRUCTION FOUND ?
YES
NO
SET ERROR # 13
SAVE NEW UIP
RESULT SIGN = LOOP INC. SIGN ?
YES
NO
RTS

FIG. 71M

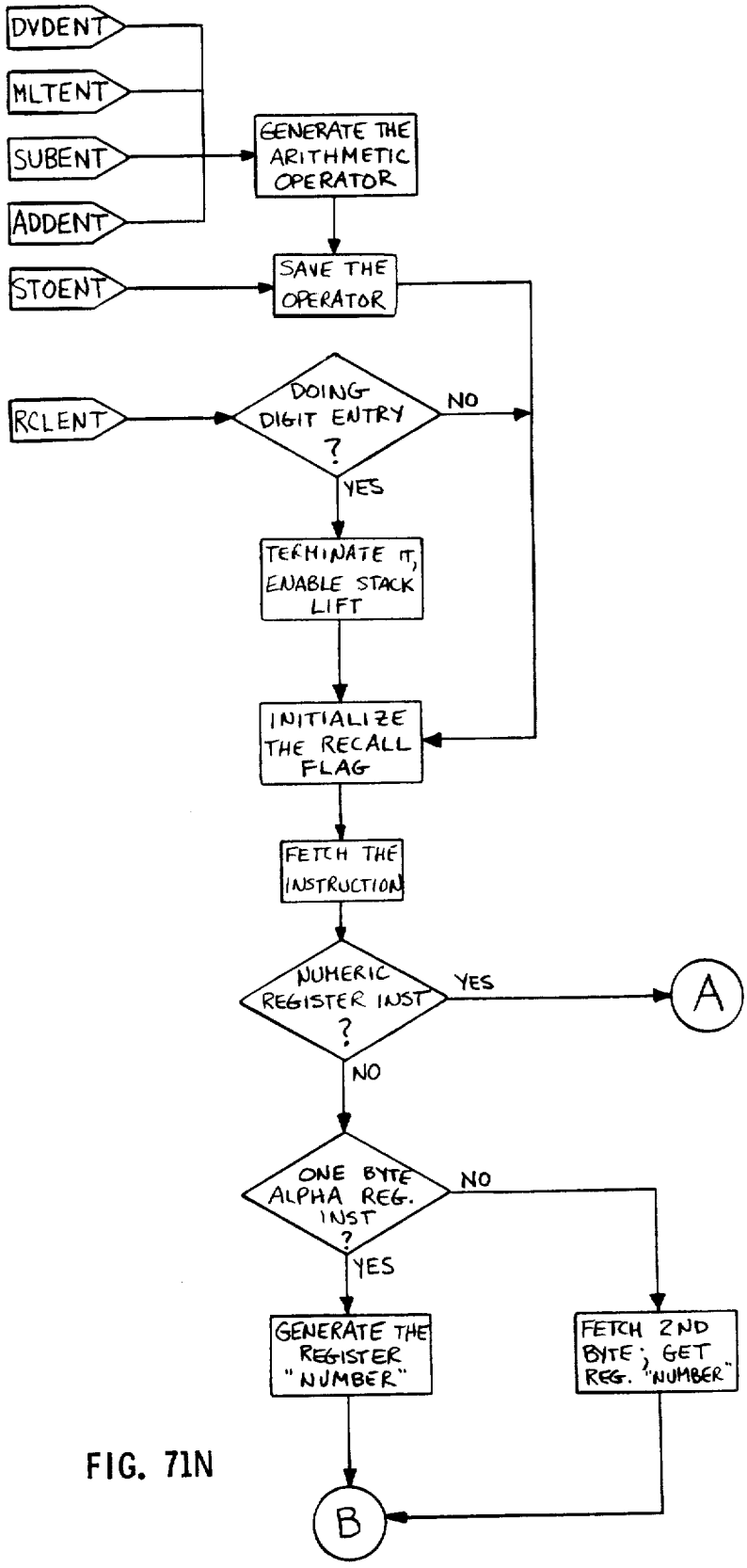

DVDENT
MLTENT
SUBENT
ADDENT
GENERATE THE ARITHMETIC OPERATOR
STOENT
SAVE THE OPERATOR
RCLENT
DOING DIGIT ENTRY ?
NO
YES
TERMINATE IT, ENABLE STACK LIFT
INITIALIZE THE RECALL FLAG
FETCH THE INSTRUCTION
NUMERIC REGISTER INST ?
YES
A
NO
ONE BYTE ALPHA REG. INST ?
NO
YES
GENERATE THE REGISTER "NUMBER"
FETCH 2ND BYTE; GET REG. "NUMBER"
B

FIG. 71N

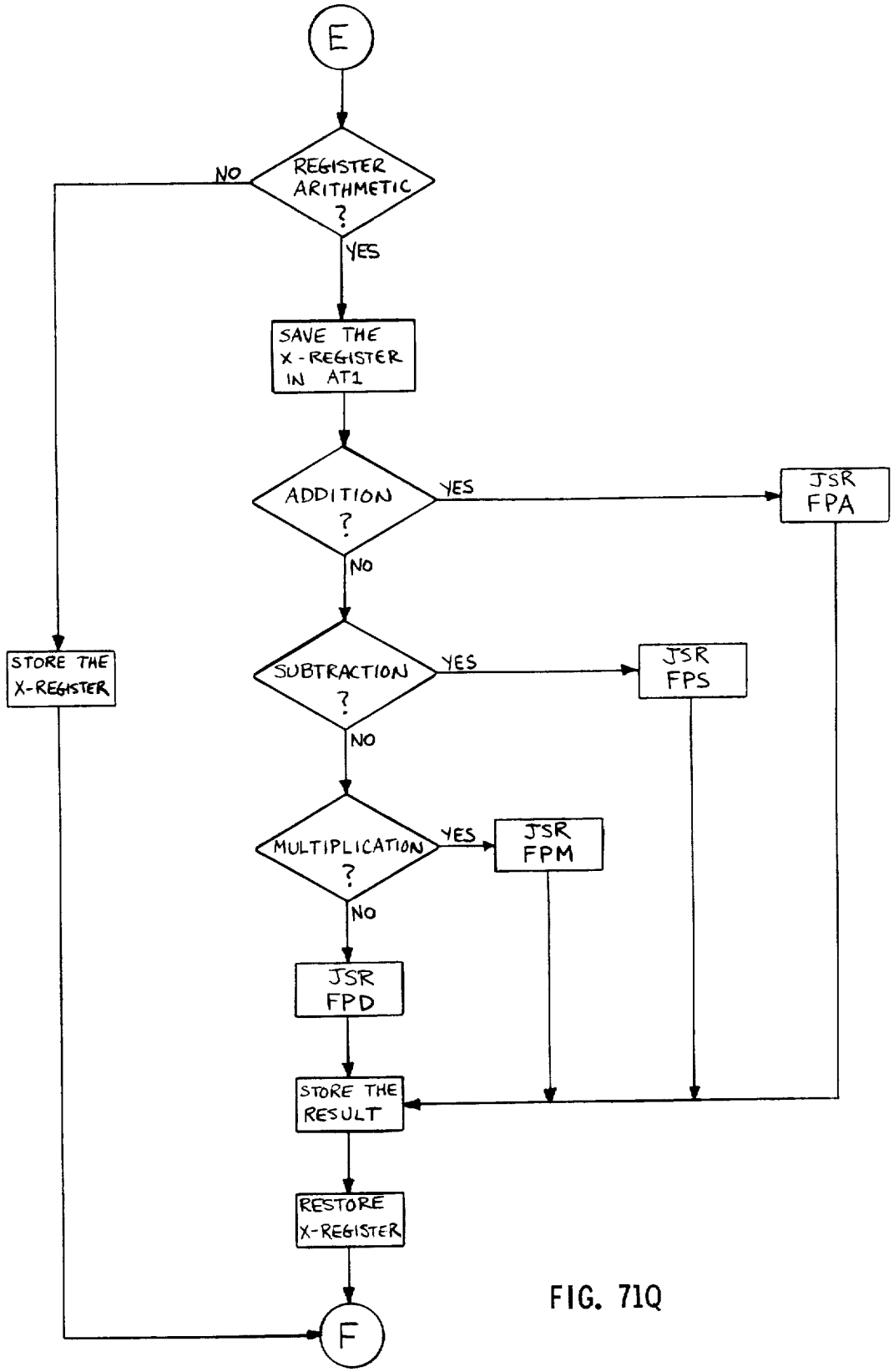

E
REGISTER ARITHMETIC ?
NO
YES
SAVE THE X-REGISTER IN AT1
ADDITION ?
YES
JSR FPA
NO
STORE THE X-REGISTER
SUBTRACTION ?
YES
JSR FPS
NO
MULTIPLICATION ?
YES
JSR FPM
NO
JSR FPD
STORE THE RESULT
RESTORE X-REGISTER
F

FIG. 71Q

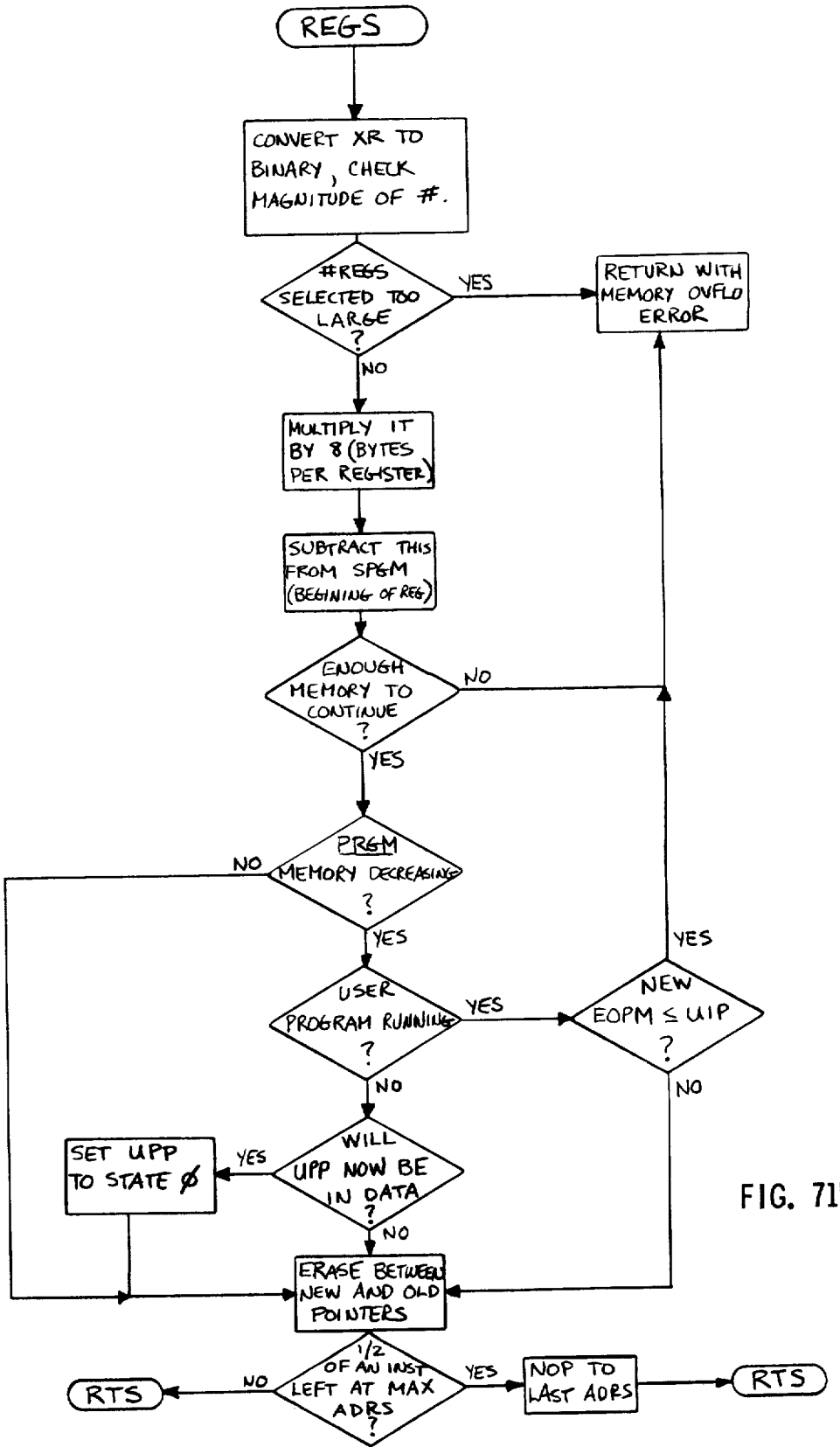

REGS
CONVERT XR TO BINARY, CHECK MAGNITUDE OF #.
#REGS SELECTED TOO LARGE ?
YES
RETURN WITH MEMORY OVFLO ERROR
NO
MULTIPLY IT BY 8 (BYTES PER REGISTER)
SUBTRACT THIS FROM SP&M (BEGINING OF REG)
ENOUGH MEMORY TO CONTINUE ?
NO
YES
PRGM MEMORY DECREASING ?
NO
YES
USER PROGRAM RUNNING ?
YES
NEW EOPM ≤ UIP ?
YES
NO
NO
WILL UPP NOW BE IN DATA ?
YES
SET UPP TO STATE ∅
NO
ERASE BETWEEN NEW AND OLD POINTERS
1/2 OF AN INST LEFT AT MAX ADRS ?
NO
RTS
YES
NOP TO LAST ADRS
RTS

FIG. 71R

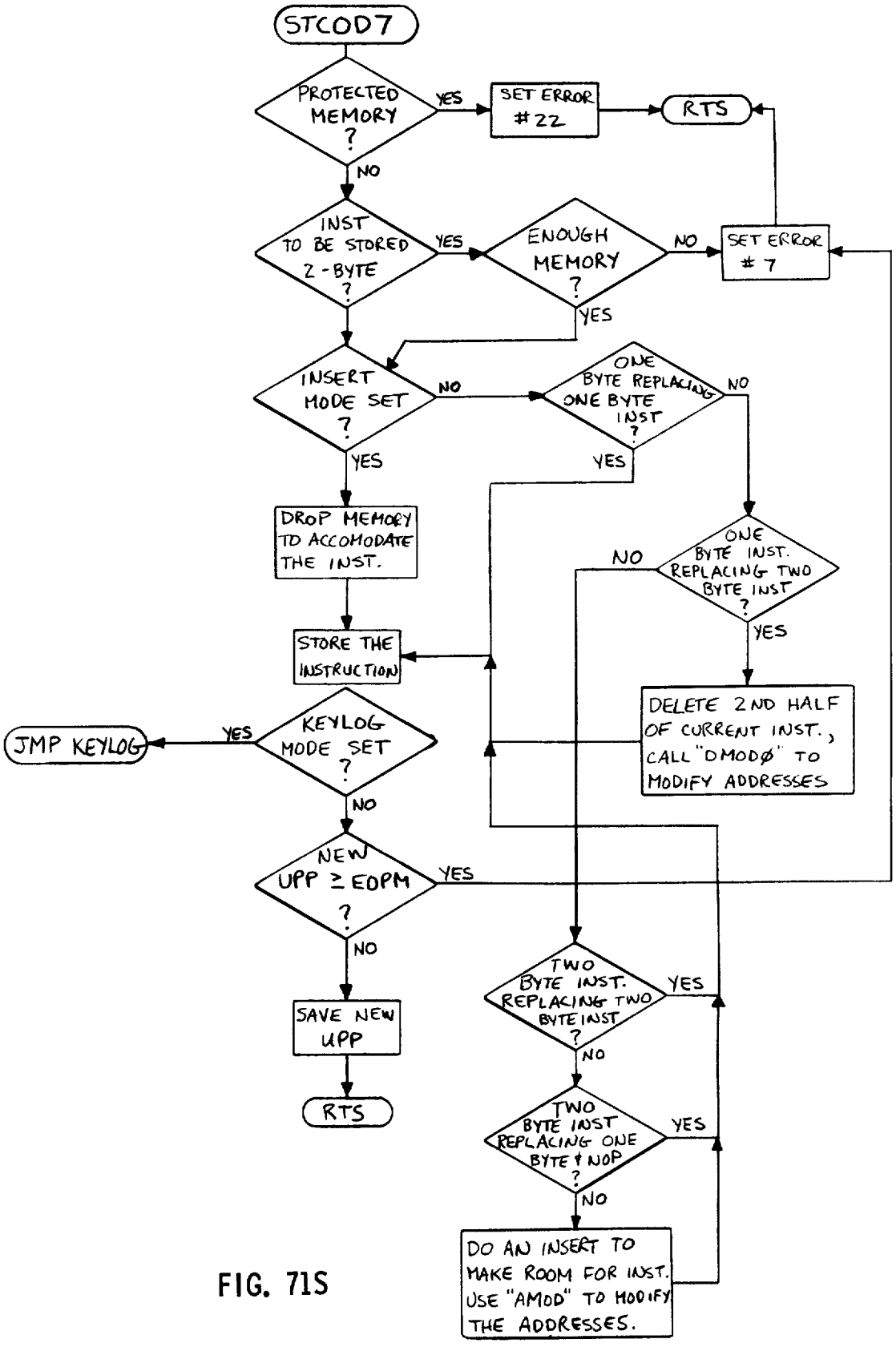

STCOD7
PROTECTED MEMORY ?
YES
SET ERROR #22
RTS
NO
INST TO BE STORED 2-BYTE ?
YES
ENOUGH MEMORY ?
NO
SET ERROR # 7
YES
INSERT MODE SET ?
NO
ONE BYTE REPLACING ONE BYTE INST ?
NO
YES
YES
DROP MEMORY TO ACCOMODATE THE INST.
ONE BYTE INST. REPLACING TWO BYTE INST ?
NO
YES
STORE THE INSTRUCTION
DELETE 2ND HALF OF CURRENT INST., CALL "DMODØ" TO MODIFY ADDRESSES
JMP KEYLOG
YES
KEYLOG MODE SET ?
NO
NEW UPP ≥ EOPM ?
YES
NO
TWO BYTE INST. REPLACING TWO BYTE INST ?
YES
NO
SAVE NEW UPP
RTS
TWO BYTE INST REPLACING ONE BYTE ≠ NOP ?
YES
NO
DO AN INSERT TO MAKE ROOM FOR INST. USE "AMOD" TO MODIFY THE ADDRESSES.

FIG. 71S

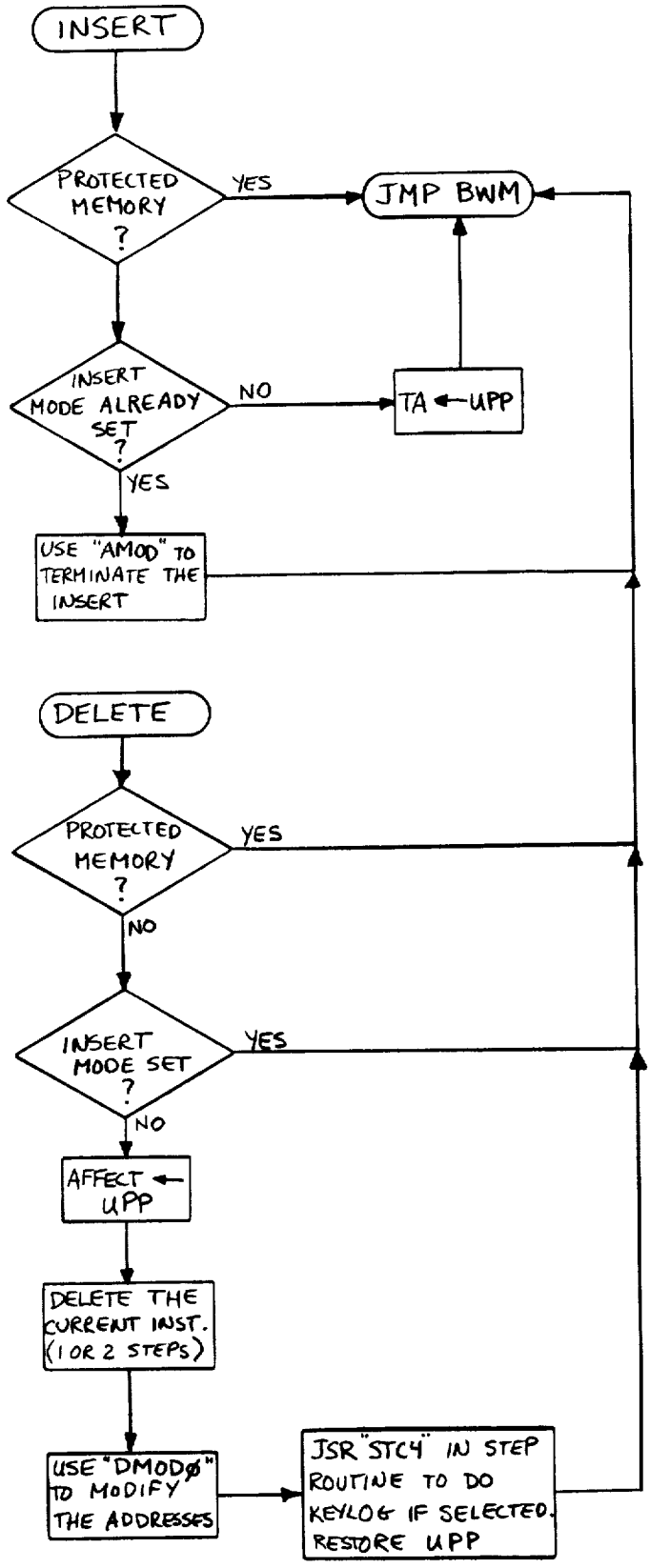

INSERT
PROTECTED MEMORY ?
YES
JMP BWM
INSERT MODE ALREADY SET ?
NO
TA ← UPP
YES
USE "AMOD" TO TERMINATE THE INSERT
DELETE
PROTECTED MEMORY ?
YES
NO
INSERT MODE SET ?
YES
NO
AFFECT ← UPP
DELETE THE CURRENT INST. (1 OR 2 STEPS)
USE "DMOD0" TO MODIFY THE ADDRESSES
JSR "STC4" IN STEP ROUTINE TO DO KEYLOG IF SELECTED. RESTORE UPP

FIG. 71T

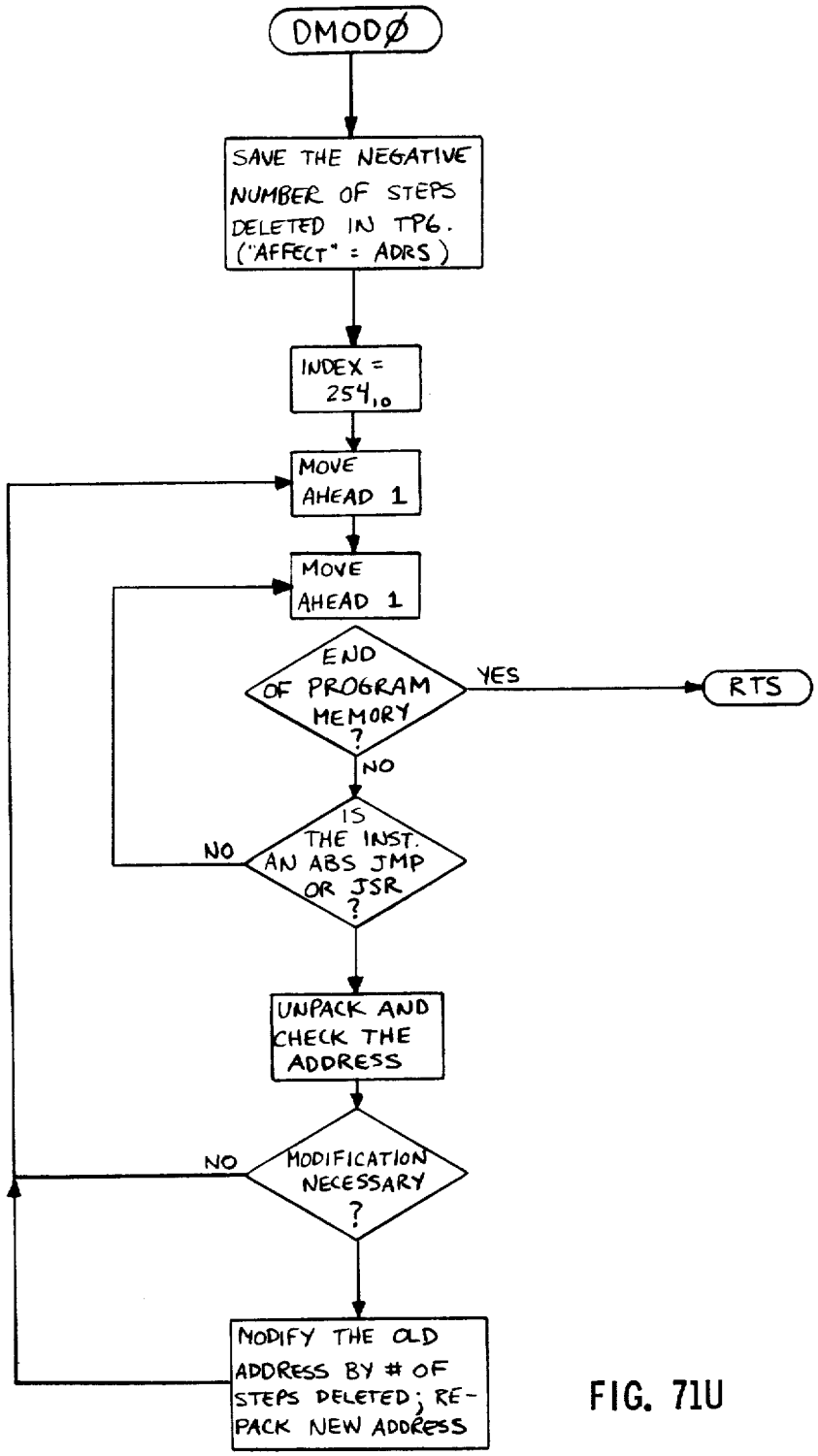

DMOD∅
SAVE THE NEGATIVE NUMBER OF STEPS DELETED IN TP6. ("AFFECT" = ADRS)
INDEX = 254₁₀
MOVE AHEAD 1
MOVE AHEAD 1
END OF PROGRAM MEMORY ?
YES
RTS
NO
IS THE INST. AN ABS JMP OR JSR ?
NO
UNPACK AND CHECK THE ADDRESS
MODIFICATION NECESSARY ?
NO
MODIFY THE OLD ADDRESS BY # OF STEPS DELETED; RE-PACK NEW ADDRESS

FIG. 71U

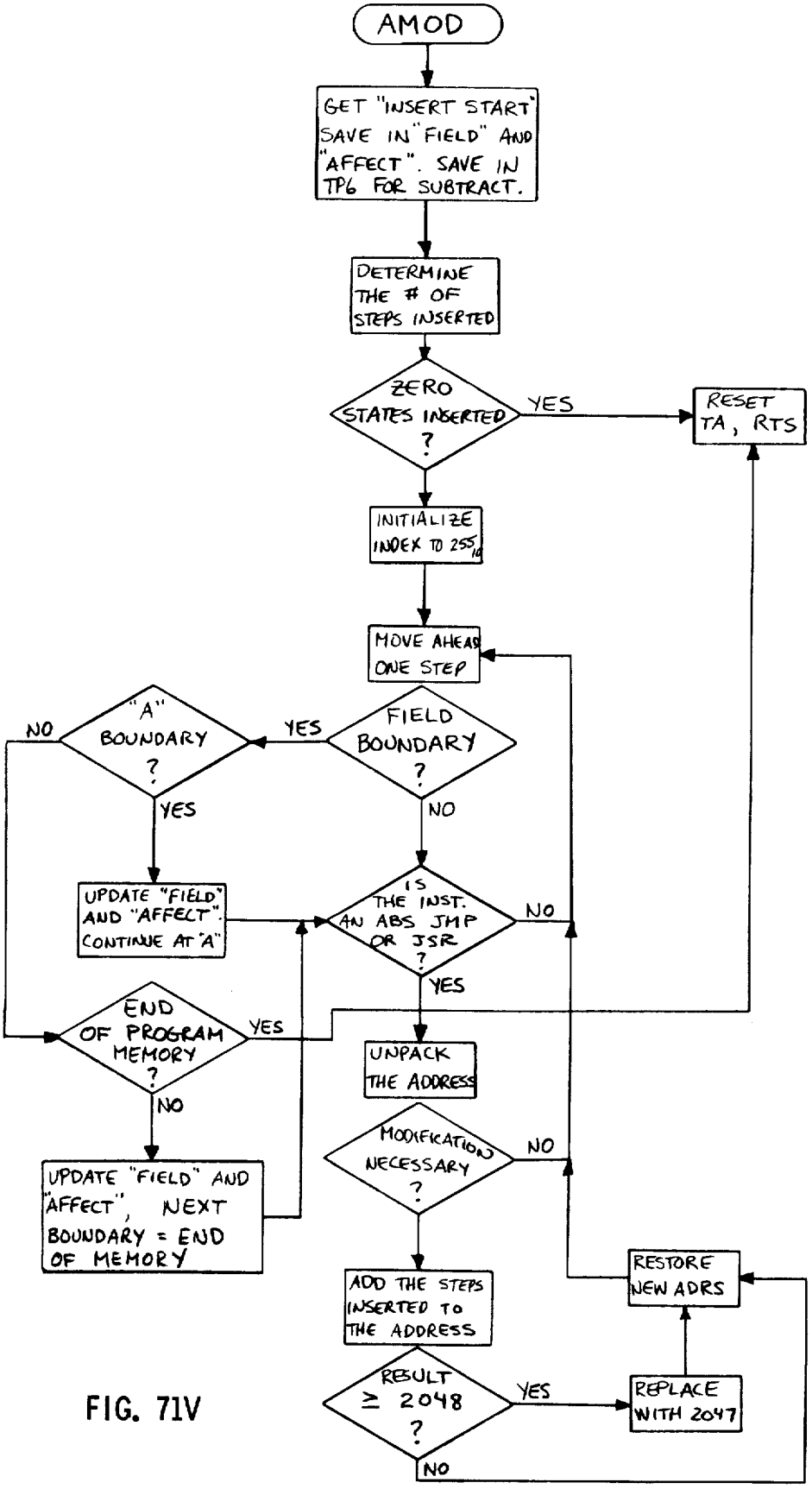

AMOD
GET "INSERT START" SAVE IN "FIELD" AND "AFFECT". SAVE IN TP6 FOR SUBTRACT.
DETERMINE THE # OF STEPS INSERTED
ZERO STATES INSERTED ?
YES
RESET TA, RTS
INITIALIZE INDEX TO 255₁₀
MOVE AHEAD ONE STEP
"A" BOUNDARY ?
NO
YES
FIELD BOUNDARY ?
YES
NO
UPDATE "FIELD" AND "AFFECT". CONTINUE AT "A"
IS THE INST. AN ABS JMP OR JSR ?
NO
YES
END OF PROGRAM MEMORY ?
YES
NO
UNPACK THE ADDRESS
MODIFICATION NECESSARY ?
NO
UPDATE "FIELD" AND "AFFECT", NEXT BOUNDARY = END OF MEMORY
ADD THE STEPS INSERTED TO THE ADDRESS
RESTORE NEW ADRS
RESULT ≥ 2048 ?
YES
REPLACE WITH 2047
NO

FIG. 71V

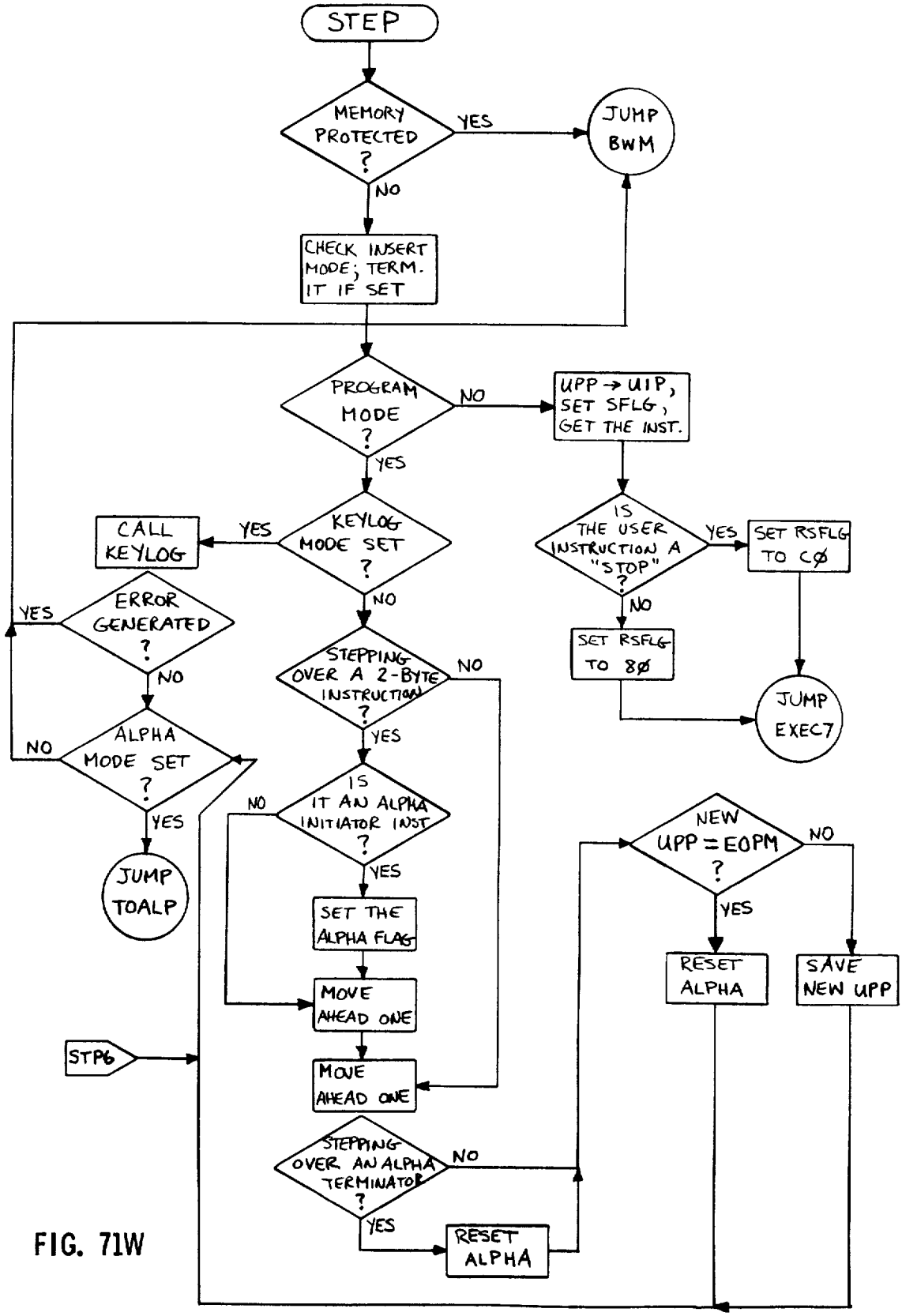

STEP
MEMORY PROTECTED ?
YES
JUMP BWM
NO
CHECK INSERT MODE; TERM. IT IF SET
PROGRAM MODE ?
NO
UPP → UIP, SET SFLG, GET THE INST.
YES
KEYLOG MODE SET ?
YES
CALL KEYLOG
NO
IS THE USER INSTRUCTION A "STOP" ?
YES
SET RSFLG TO CØ
NO
SET RSFLG TO 8Ø
ERROR GENERATED ?
YES
NO
STEPPING OVER A 2-BYTE INSTRUCTION ?
NO
YES
ALPHA MODE SET ?
NO
YES
JUMP TOALP
JUMP EXEC7
IS IT AN ALPHA INITIATOR INST ?
NO
YES
SET THE ALPHA FLAG
MOVE AHEAD ONE
STP6
MOVE AHEAD ONE
NEW UPP = EOPM ?
NO
YES
RESET ALPHA
SAVE NEW UPP
STEPPING OVER AN ALPHA TERMINATOR ?
NO
YES
RESET ALPHA

FIG. 71W

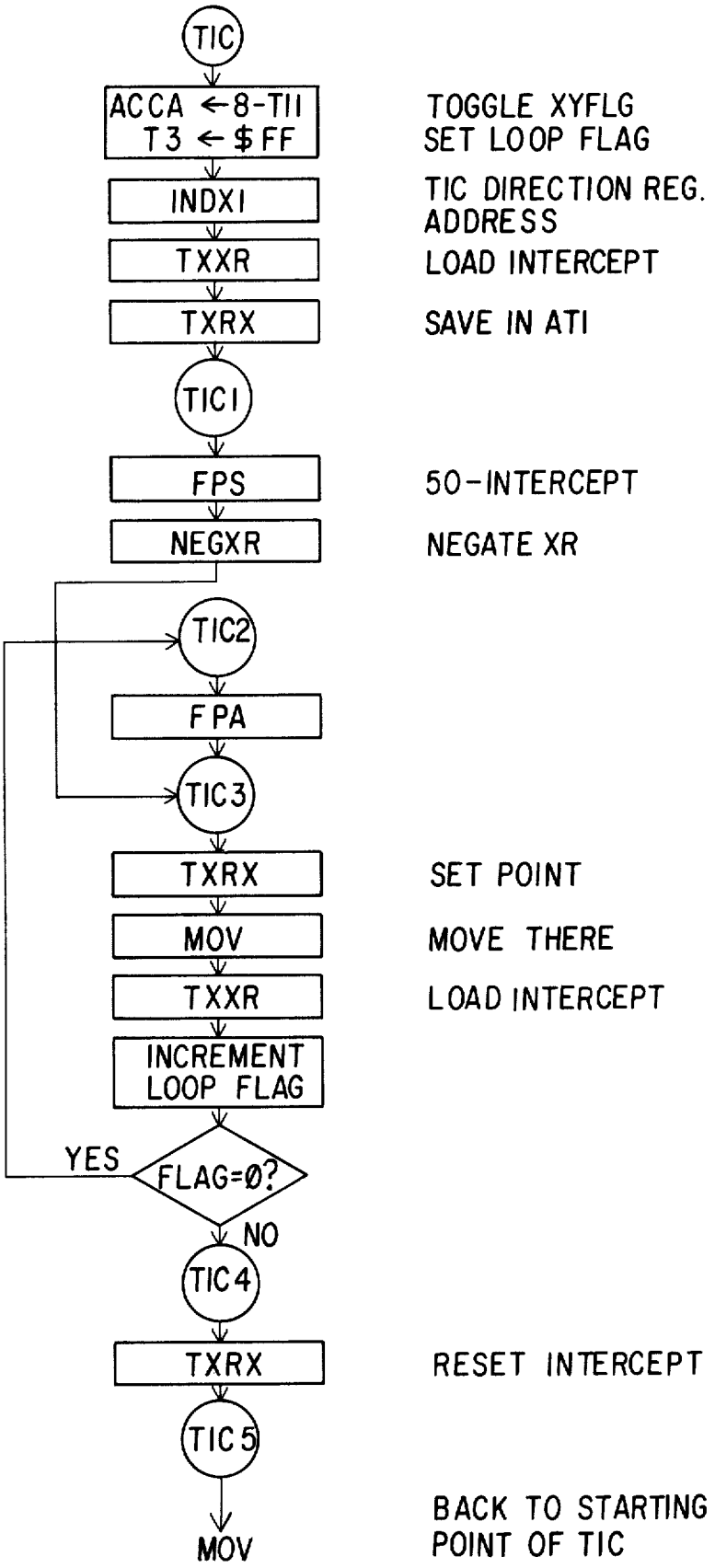

TIC
ACCA ←8-TII
T3 ←$FF
TOGGLE XYFLG
SET LOOP FLAG
INDXI
TIC DIRECTION REG. ADDRESS
TXXR
LOAD INTERCEPT
TXRX
SAVE IN ATI
TIC1
FPS
50-INTERCEPT
NEGXR
NEGATE XR
TIC2
FPA
TIC3
TXRX
SET POINT
MOV
MOVE THERE
TXXR
LOAD INTERCEPT
INCREMENT LOOP FLAG
YES
FLAG=Ø?
NO
TIC4
TXRX
RESET INTERCEPT
TIC5
MOV
BACK TO STARTING POINT OF TIC

FIG 741

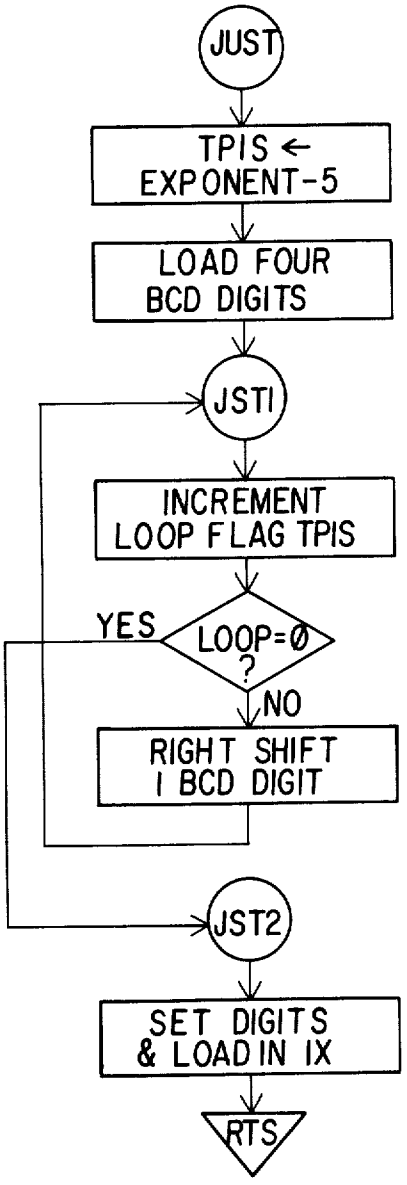

JUST
TPIS ← EXPONENT-5
SET # OF DIGITS COUNTER
LOAD FOUR BCD DIGITS
JST1
INCREMENT LOOP FLAG TPIS
BUMP COUNTER
YES
LOOP=Ø ?
DONE WITH JUSTIFICATION
NO
RIGHT SHIFT 1 BCD DIGIT
JST2
SET DIGITS & LOAD IN IX
RIGHT JUSTIFIED
RTS

FIG 74J

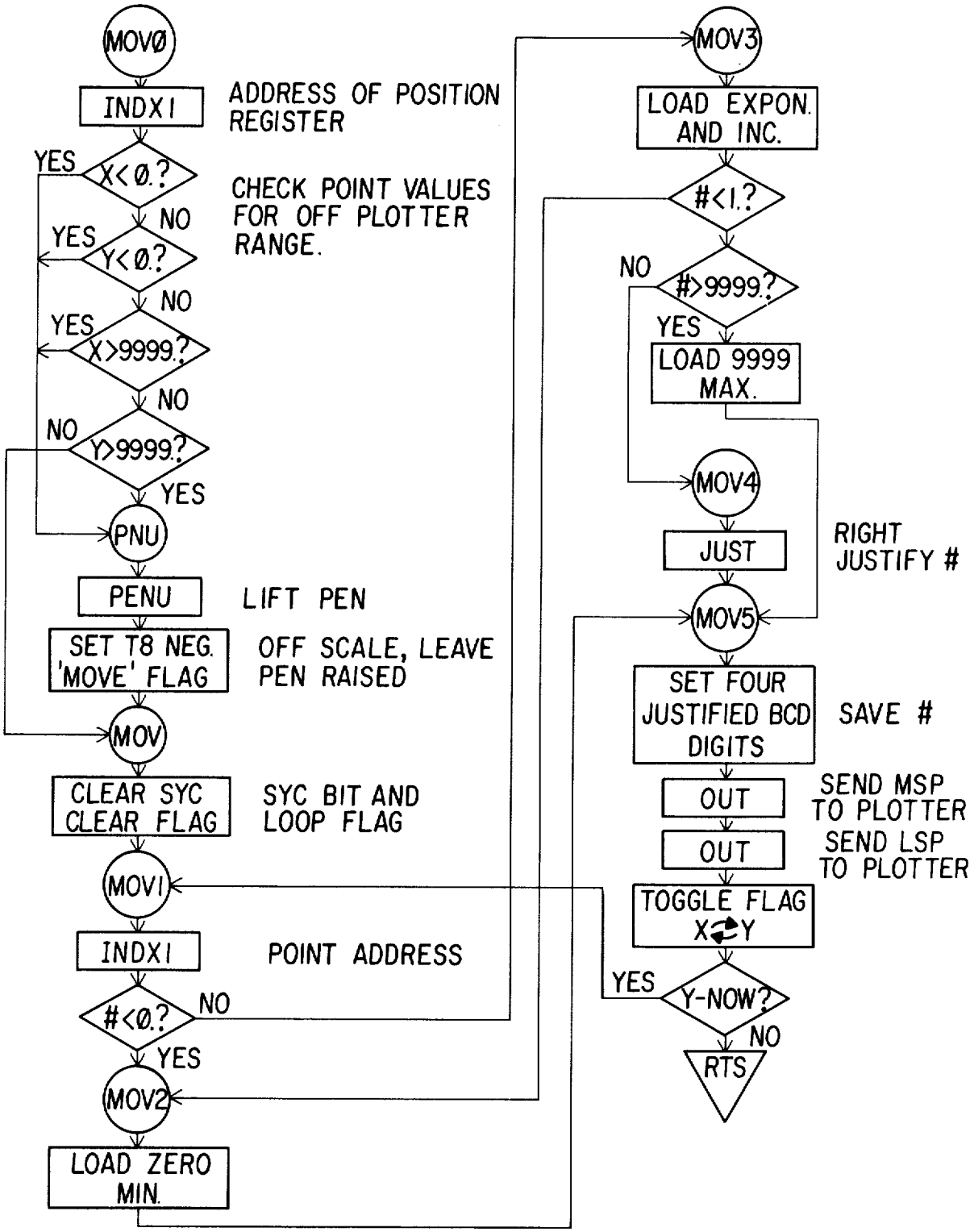

MOVØ
INDX1
ADDRESS OF POSITION REGISTER
YES
X<Ø.?
CHECK POINT VALUES FOR OFF PLOTTER RANGE.
NO
YES
Y<Ø.?
NO
YES
X>9999.?
NO
NO
Y>9999.?
YES
PNU
PENU
LIFT PEN
SET T8 NEG. 'MOVE' FLAG
OFF SCALE, LEAVE PEN RAISED
MOV
CLEAR SYC CLEAR FLAG
SYC BIT AND LOOP FLAG
MOV1
INDX1
POINT ADDRESS
#<Ø.?
NO
YES
MOV2
LOAD ZERO MIN.
MOV3
LOAD EXPON. AND INC.
#<1.?
NO
#>9999.?
YES
LOAD 9999 MAX.
MOV4
JUST
RIGHT JUSTIFY #
MOV5
SET FOUR JUSTIFIED BCD DIGITS
SAVE #
OUT
SEND MSP TO PLOTTER
OUT
SEND LSP TO PLOTTER
TOGGLE FLAG X⇄Y
YES
Y-NOW?
NO
RTS

FIG 74K

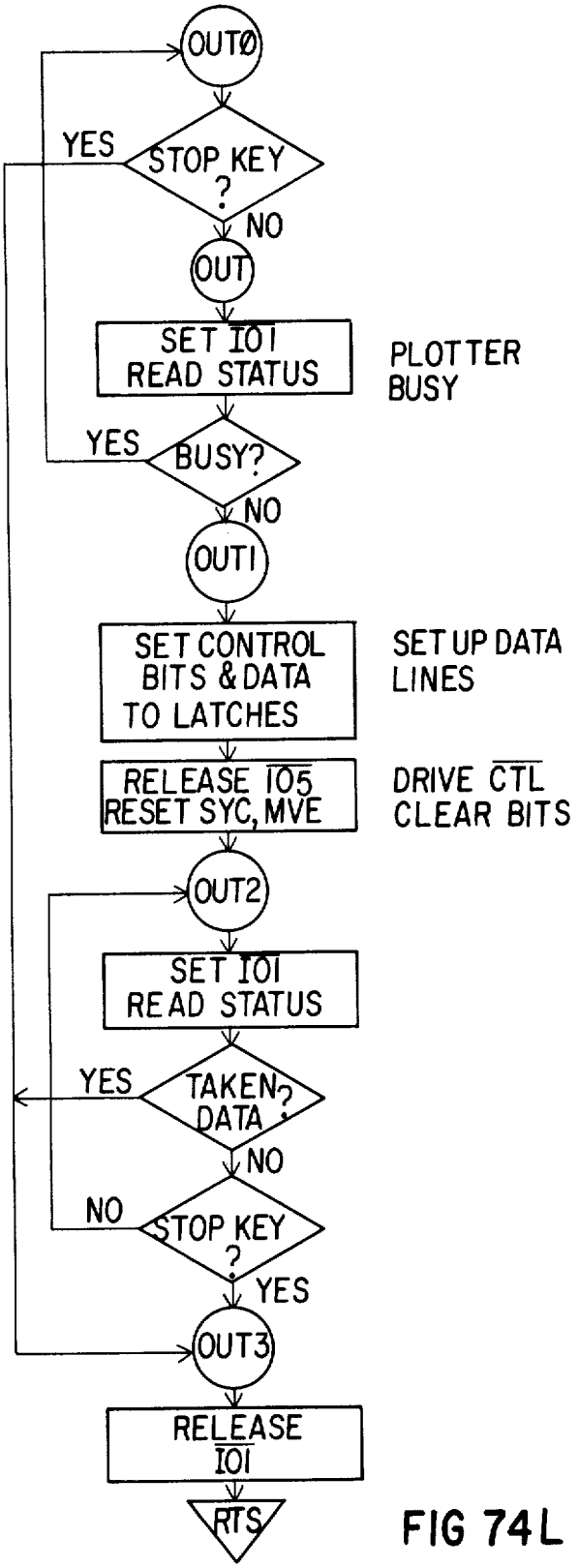


FIG 74L

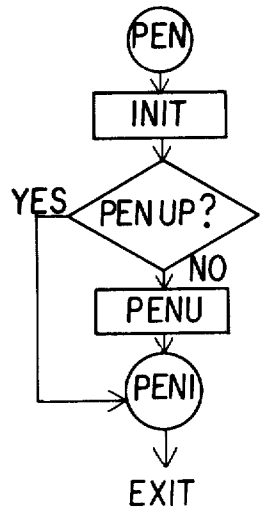


FIG 74M

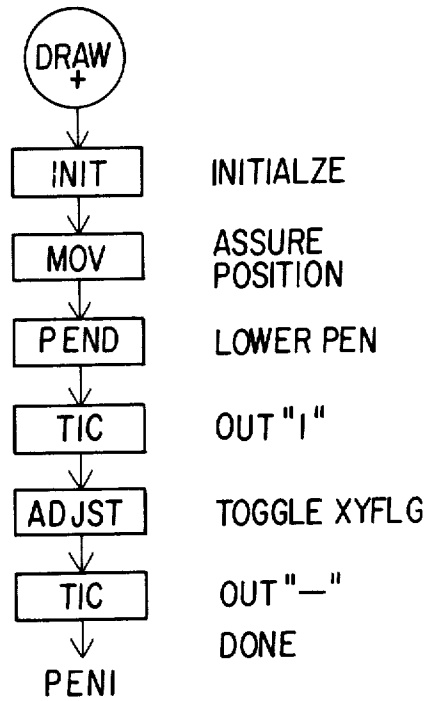


FIG 74N

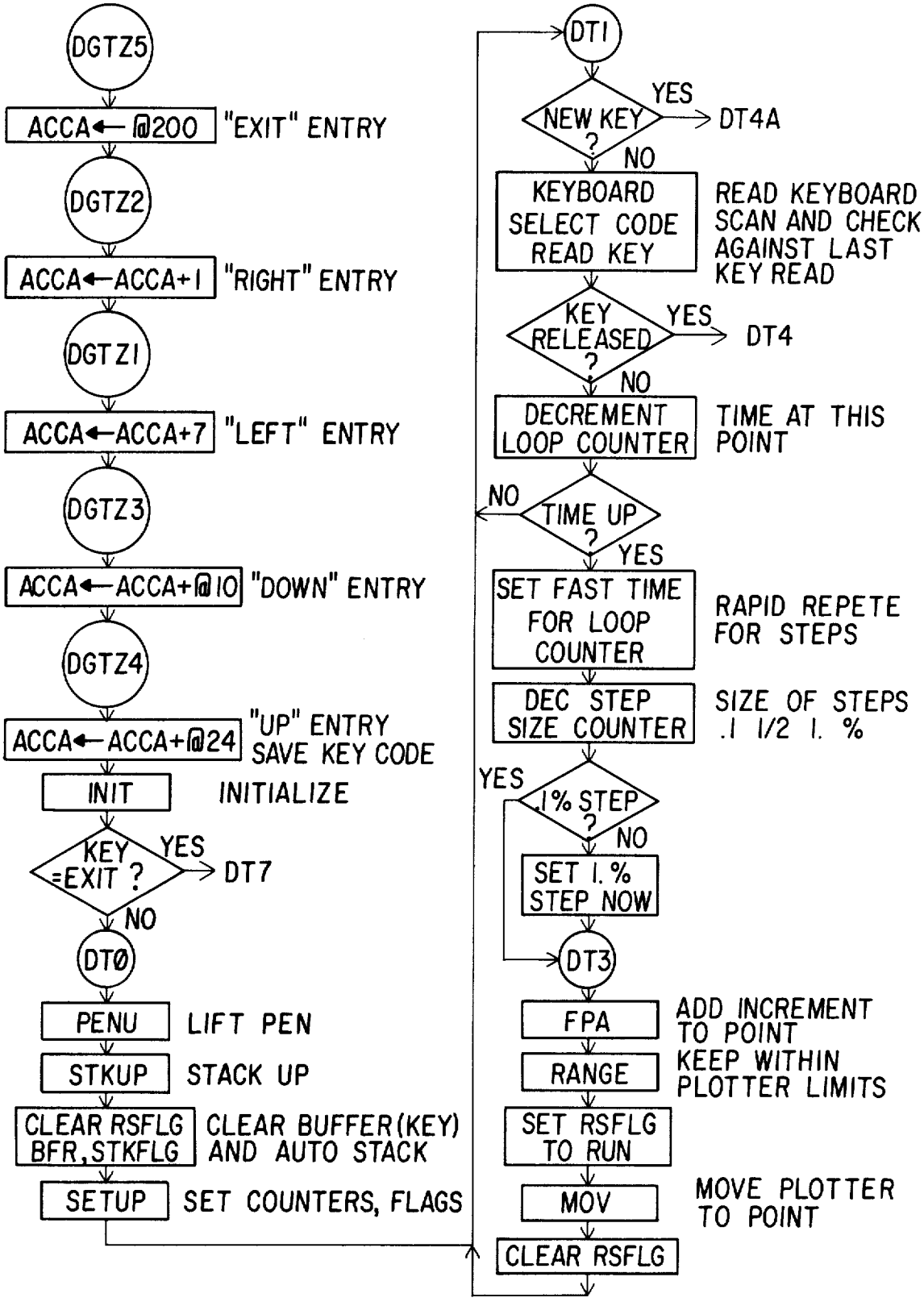

DGTZ5
ACCA←@200
"EXIT" ENTRY
DGTZ2
ACCA←ACCA+1
"RIGHT" ENTRY
DGTZ1
ACCA←ACCA+7
"LEFT" ENTRY
DGTZ3
ACCA←ACCA+@10
"DOWN" ENTRY
DGTZ4
ACCA←ACCA+@24
"UP" ENTRY
SAVE KEY CODE
INIT
INITIALIZE
KEY =EXIT ?
YES
DT7
NO
DT0
PENU
LIFT PEN
STKUP
STACK UP
CLEAR RSFLG BFR,STKFLG
CLEAR BUFFER(KEY) AND AUTO STACK
SETUP
SET COUNTERS, FLAGS
DT1
NEW KEY ?
YES
DT4A
NO
KEYBOARD SELECT CODE READ KEY
READ KEYBOARD SCAN AND CHECK AGAINST LAST KEY READ
KEY RELEASED ?
YES
DT4
NO
DECREMENT LOOP COUNTER
TIME AT THIS POINT
NO
TIME UP ?
YES
SET FAST TIME FOR LOOP COUNTER
RAPID REPETE FOR STEPS
DEC STEP SIZE COUNTER
SIZE OF STEPS .1 1/2 1. %
YES
.1% STEP ?
NO
SET 1.% STEP NOW
DT3
FPA
ADD INCREMENT TO POINT
RANGE
KEEP WITHIN PLOTTER LIMITS
SET RSFLG TO RUN
MOV
MOVE PLOTTER TO POINT
CLEAR RSFLG

FIG 740

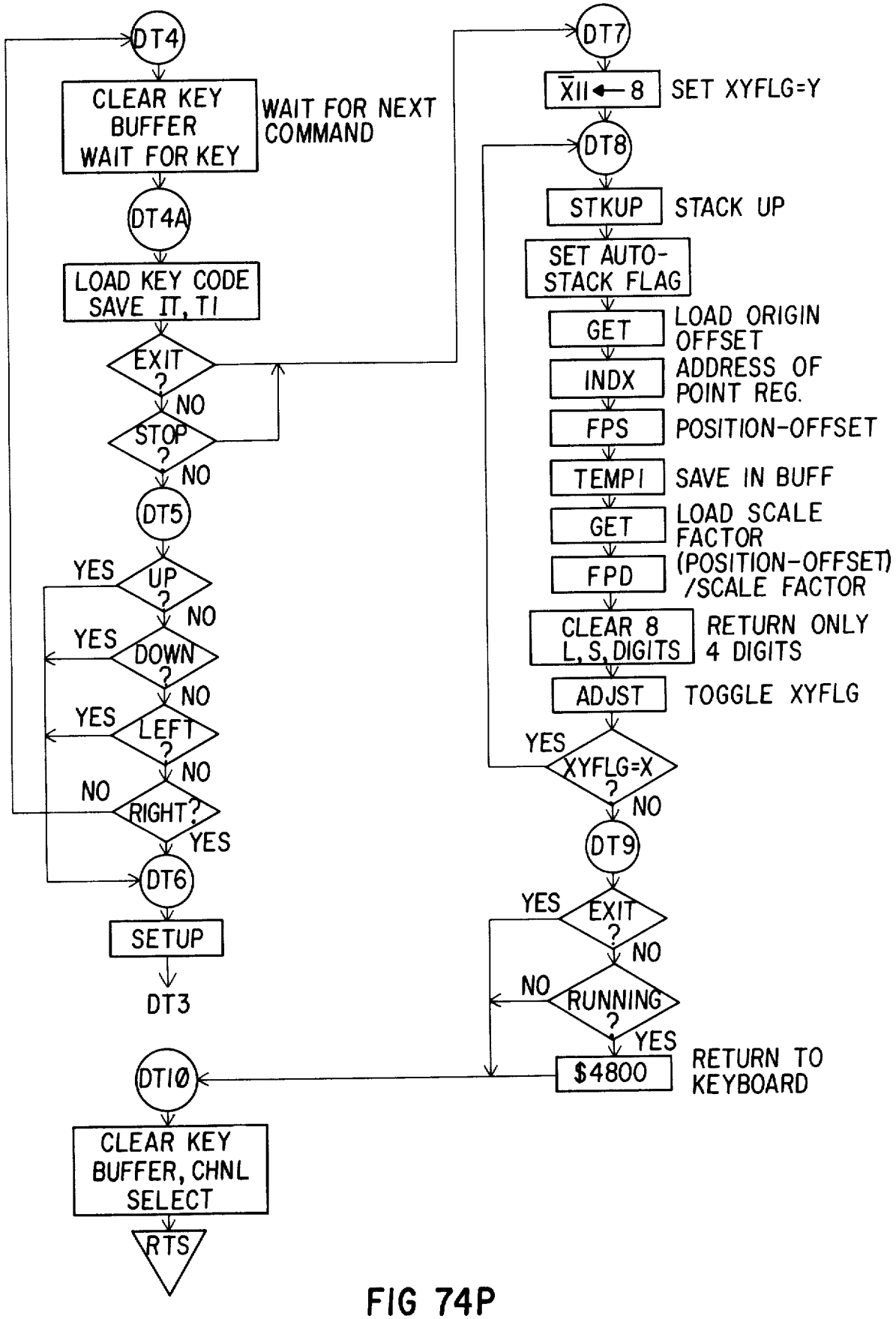

DT4
CLEAR KEY BUFFER WAIT FOR KEY
WAIT FOR NEXT COMMAND
DT4A
LOAD KEY CODE SAVE IT, TI
EXIT ?
NO
STOP ?
NO
DT5
YES
UP ?
NO
YES
DOWN ?
NO
YES
LEFT ?
NO
NO
RIGHT ?
YES
DT6
SETUP
DT3
DT10
CLEAR KEY BUFFER, CHNL SELECT
RTS
DT7
X11 ← 8
SET XYFLG=Y
DT8
STKUP
STACK UP
SET AUTO-STACK FLAG
GET
LOAD ORIGIN OFFSET
INDX
ADDRESS OF POINT REG.
FPS
POSITION-OFFSET
TEMP1
SAVE IN BUFF
GET
LOAD SCALE FACTOR
FPD
(POSITION-OFFSET) /SCALE FACTOR
CLEAR 8 L, S, DIGITS
RETURN ONLY 4 DIGITS
ADJST
TOGGLE XYFLG
YES
XYFLG=X ?
NO
DT9
YES
EXIT ?
NO
NO
RUNNING ?
YES
$4800
RETURN TO KEYBOARD

FIG 74P

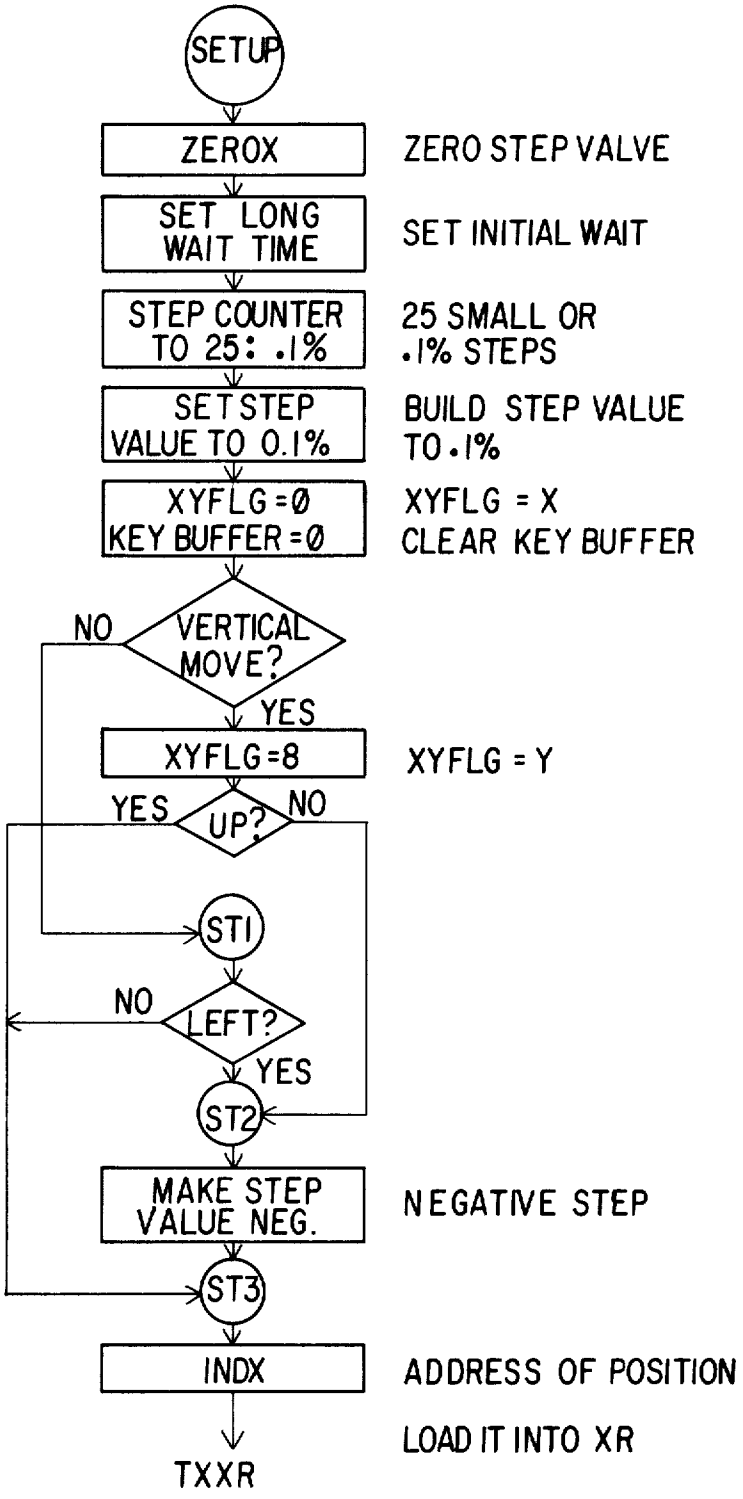

SETUP
ZEROX
ZERO STEP VALVE
SET LONG WAIT TIME
SET INITIAL WAIT
STEP COUNTER TO 25: .1%
25 SMALL OR .1% STEPS
SET STEP VALUE TO 0.1%
BUILD STEP VALUE TO .1%
XYFLG=Ø KEY BUFFER=Ø
XYFLG = X
CLEAR KEY BUFFER
NO
VERTICAL MOVE?
YES
XYFLG=8
XYFLG = Y
YES
UP?
NO
STI
NO
LEFT?
YES
ST2
MAKE STEP VALUE NEG.
NEGATIVE STEP
ST3
INDX
ADDRESS OF POSITION
LOAD IT INTO XR
TXXR

FIG 74Q

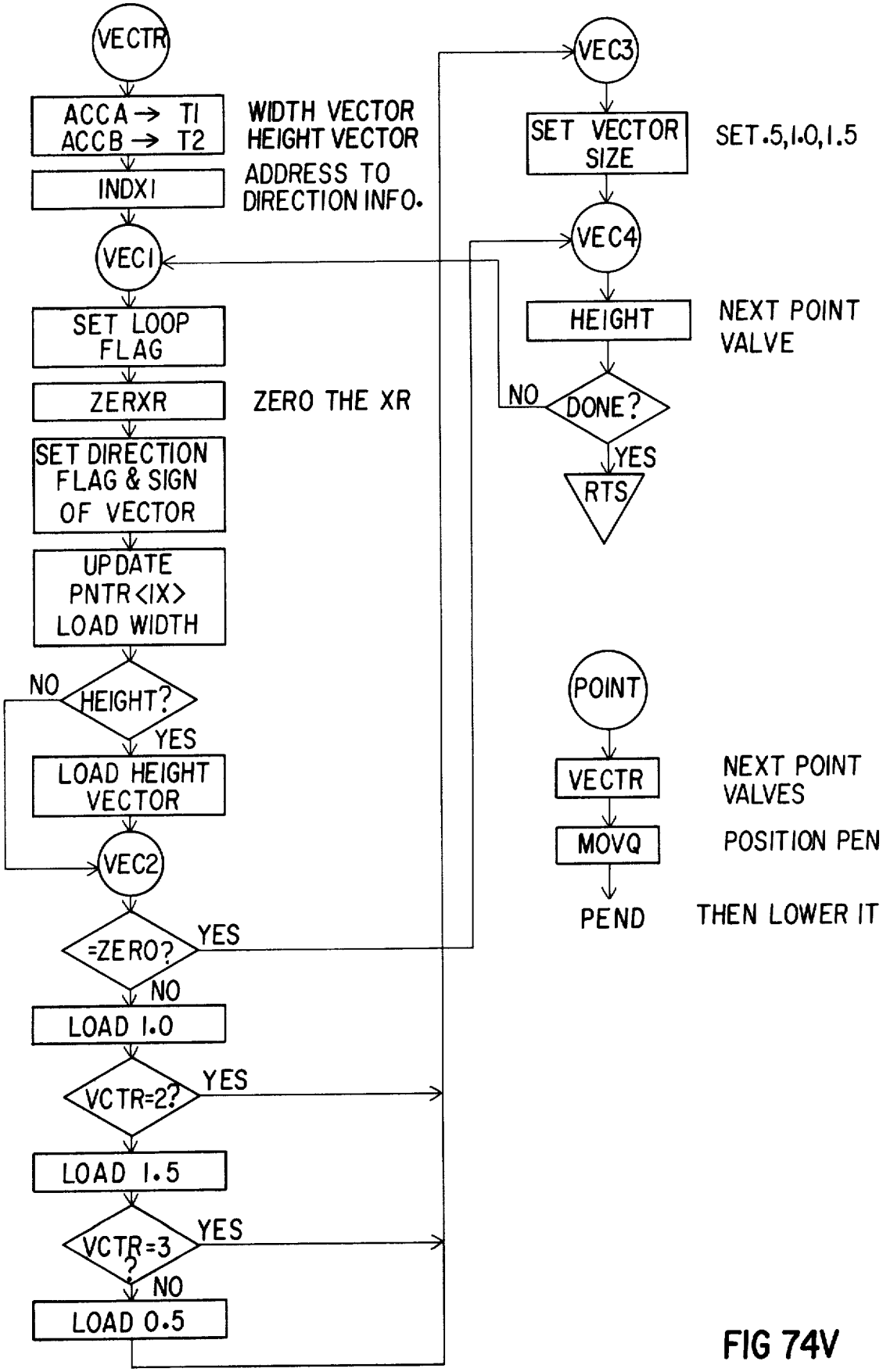

VECTR
ACCA → T1
ACCB → T2
WIDTH VECTOR
HEIGHT VECTOR
INDX1
ADDRESS TO
DIRECTION INFO.
VEC1
SET LOOP
FLAG
ZERXR
ZERO THE XR
SET DIRECTION
FLAG & SIGN
OF VECTOR
UPDATE
PNTR<IX>
LOAD WIDTH
NO
HEIGHT?
YES
LOAD HEIGHT
VECTOR
VEC2
=ZERO?
YES
NO
LOAD 1.0
VCTR=2?
YES
LOAD 1.5
VCTR=3
?
YES
NO
LOAD 0.5
VEC3
SET VECTOR
SIZE
SET.5,1.0,1.5
VEC4
HEIGHT
NEXT POINT
VALVE
NO
DONE?
YES
RTS
POINT
VECTR
NEXT POINT
VALVES
MOVQ
POSITION PEN
PEND
THEN LOWER IT

FIG 74V

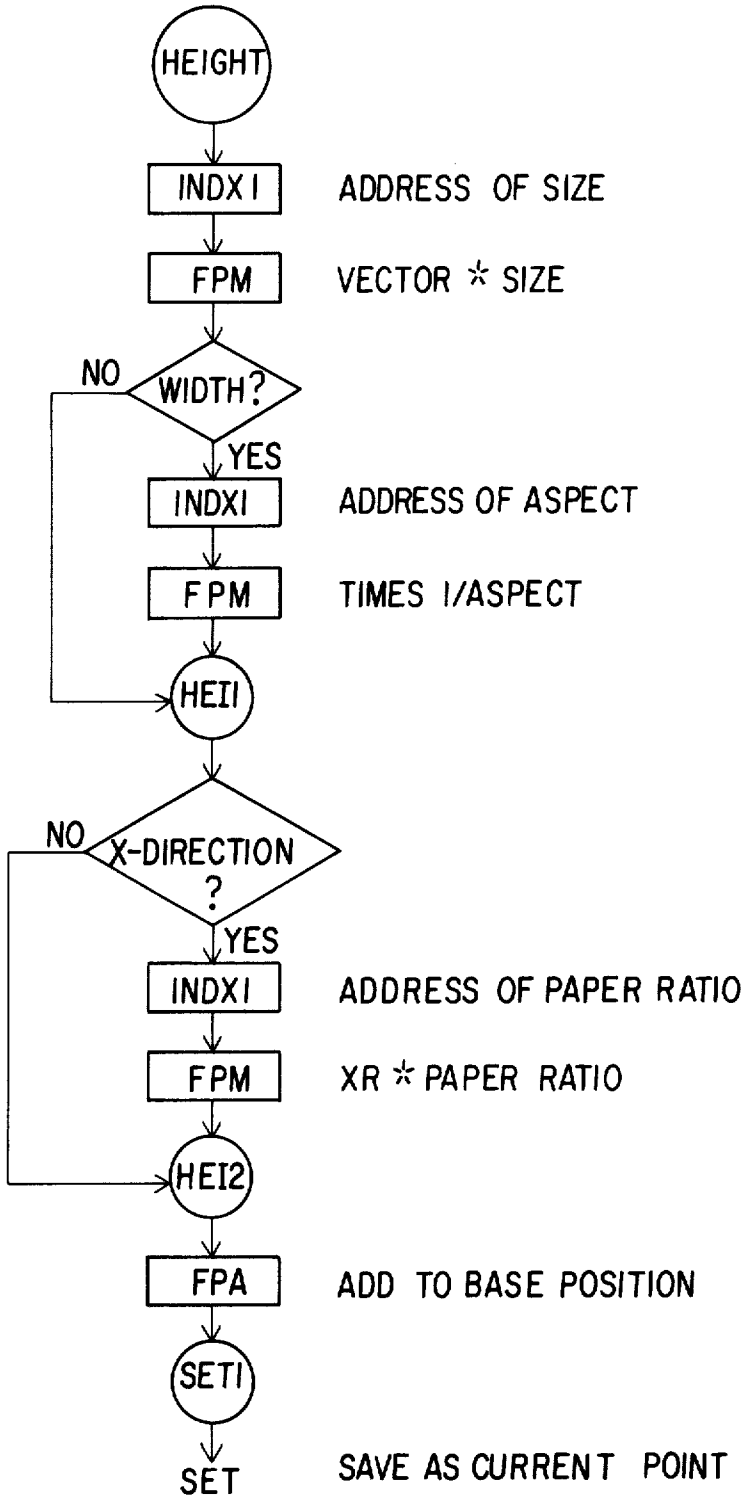

HEIGHT
INDX I
ADDRESS OF SIZE
FPM
VECTOR * SIZE
NO
WIDTH?
YES
INDXI
ADDRESS OF ASPECT
FPM
TIMES I/ASPECT
HEII
NO
X-DIRECTION ?
YES
INDXI
ADDRESS OF PAPER RATIO
FPM
XR * PAPER RATIO
HEI2
FPA
ADD TO BASE POSITION
SETI
SET
SAVE AS CURRENT POINT

FIG 74W

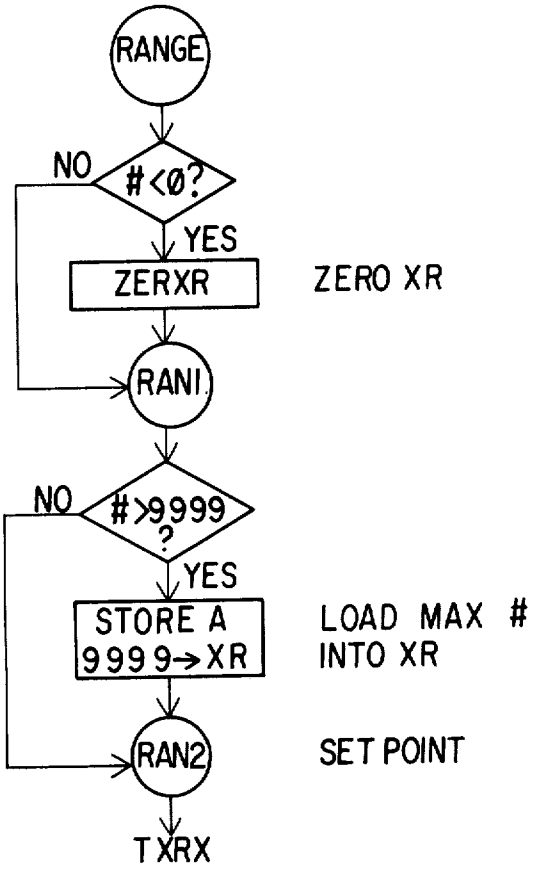

RANGE
NO
#<Ø?
YES
ZERXR
ZERO XR
RAN1
NO
#>9999
?
YES
STORE A
9999→XR
LOAD MAX #
INTO XR
RAN2
SET POINT
TXRX

FIG 74R

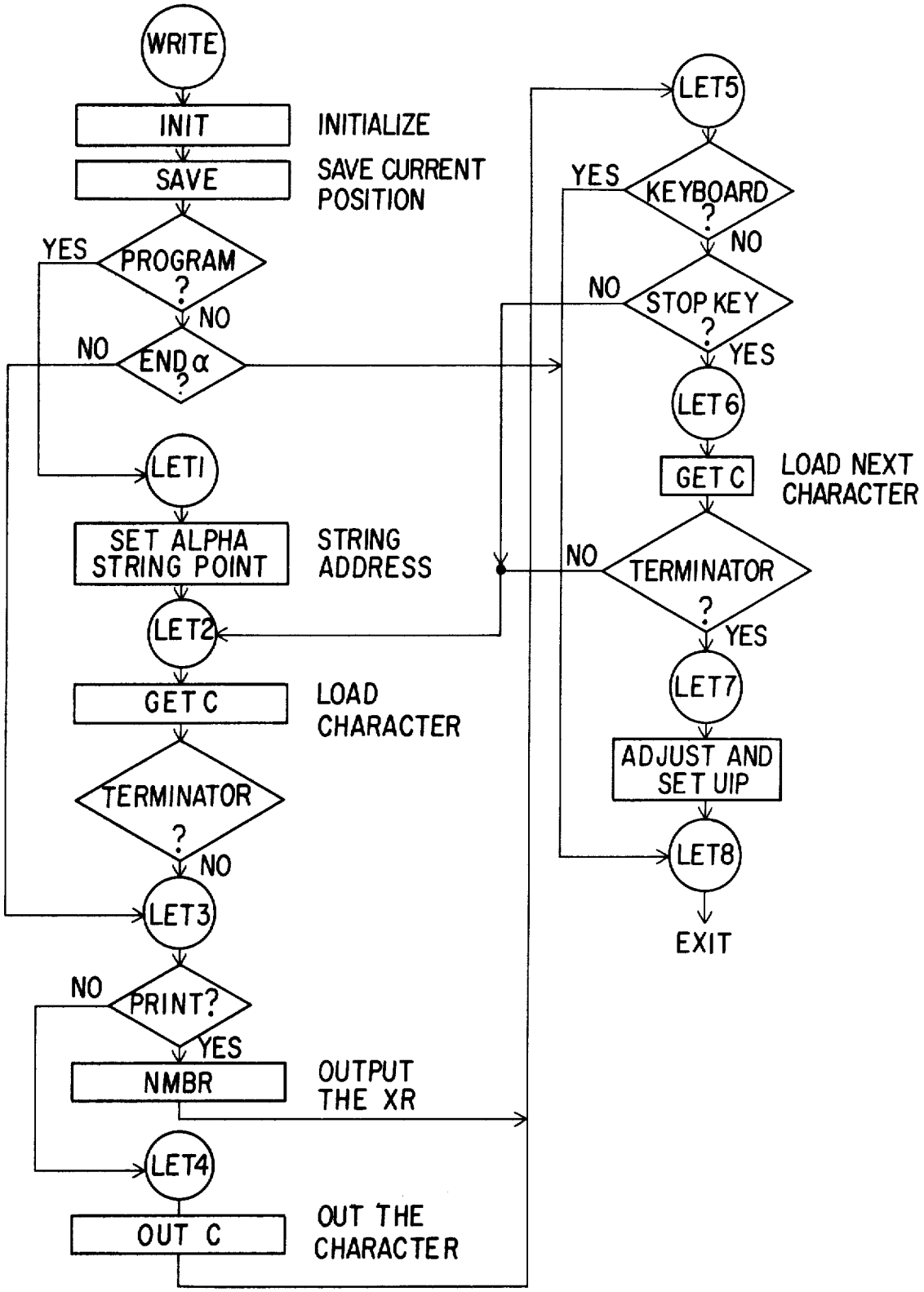

WRITE
INIT
INITIALIZE
SAVE
SAVE CURRENT POSITION
PROGRAM ?
YES
NO
END α ?
NO
LET1
SET ALPHA STRING POINT
STRING ADDRESS
LET2
GET C
LOAD CHARACTER
TERMINATOR ?
NO
LET3
PRINT?
NO
YES
NMBR
OUTPUT THE XR
LET4
OUT C
OUT THE CHARACTER
LET5
KEYBOARD ?
YES
NO
STOP KEY ?
NO
YES
LET6
GET C
LOAD NEXT CHARACTER
TERMINATOR ?
NO
YES
LET7
ADJUST AND SET UIP
LET8
EXIT

FIG 74S

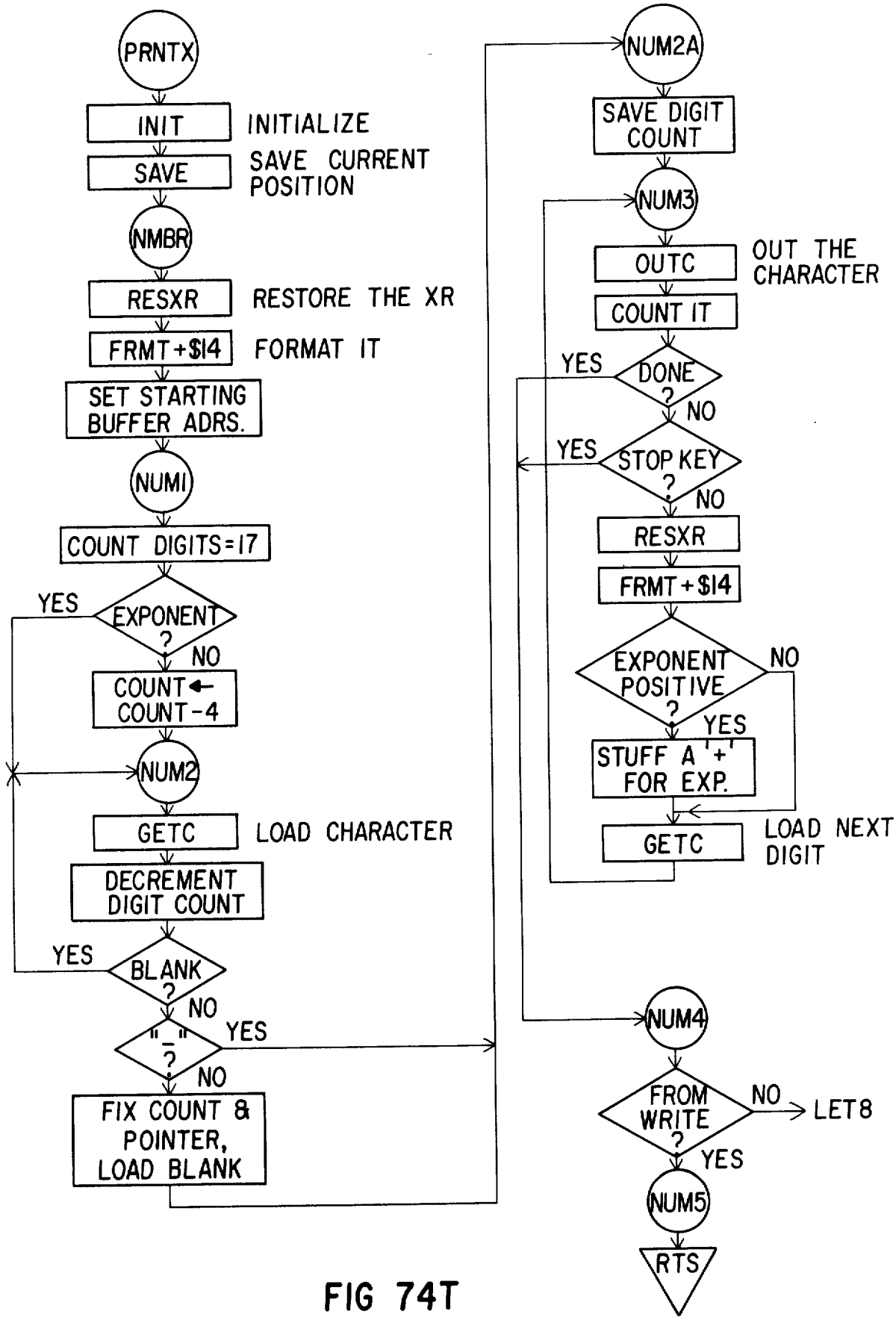

PRNTX
INIT
INITIALIZE
SAVE
SAVE CURRENT POSITION
NMBR
RESXR
RESTORE THE XR
FRMT+$14
FORMAT IT
SET STARTING BUFFER ADRS.
NUMI
COUNT DIGITS=17
YES
EXPONENT ?
NO
COUNT ← COUNT-4
NUM2
GETC
LOAD CHARACTER
DECREMENT DIGIT COUNT
YES
BLANK ?
NO
"-" ?
YES
NO
FIX COUNT & POINTER, LOAD BLANK
NUM2A
SAVE DIGIT COUNT
NUM3
OUTC
OUT THE CHARACTER
COUNT IT
YES
DONE ?
NO
YES
STOP KEY ?
NO
RESXR
FRMT+$14
EXPONENT POSITIVE ?
NO
YES
STUFF A '+' FOR EXP.
GETC
LOAD NEXT DIGIT
NUM4
FROM WRITE ?
NO
LET8
YES
NUM5
RTS

FIG 74T

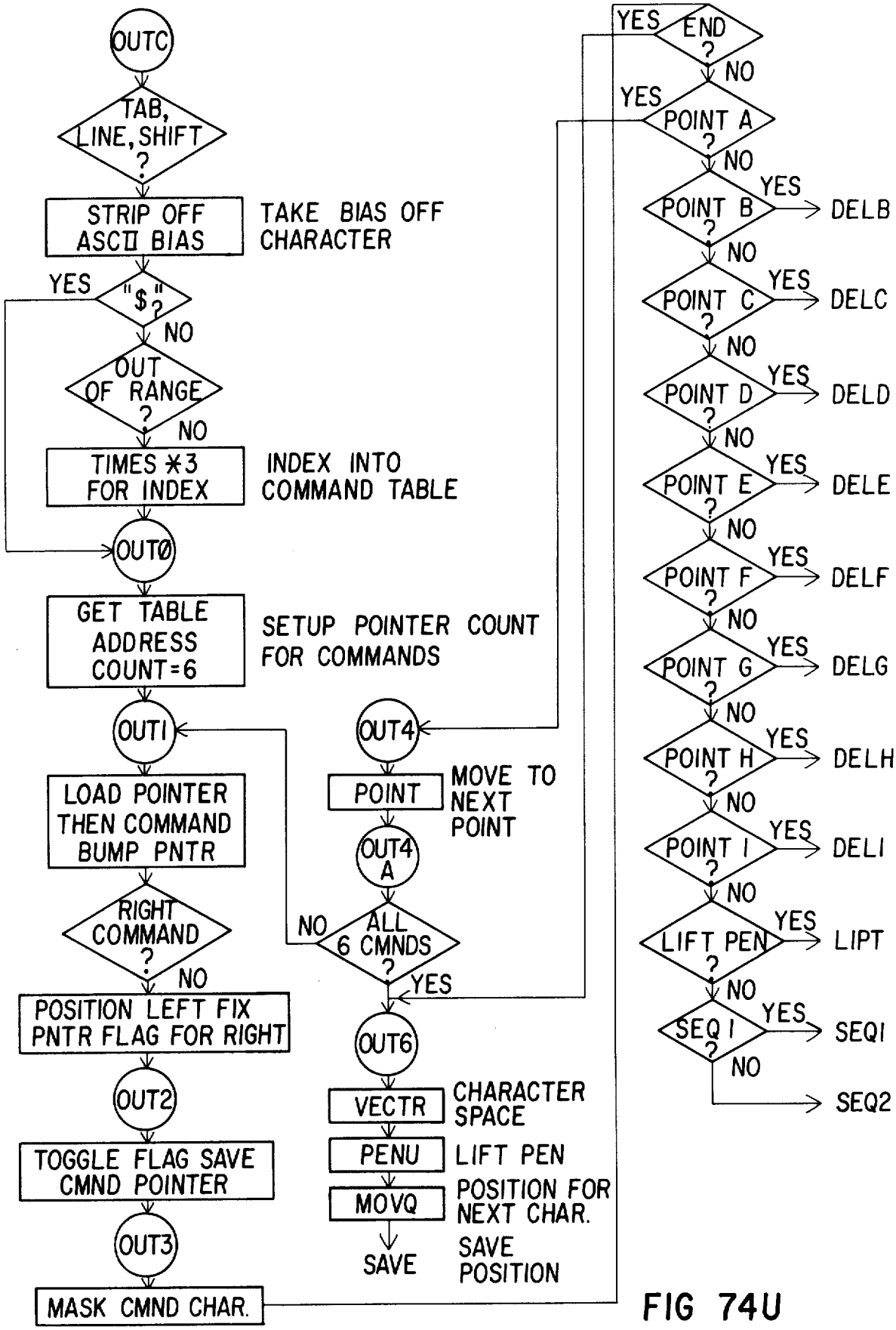

OUTC
TAB, LINE, SHIFT ?
STRIP OFF ASCII BIAS
TAKE BIAS OFF CHARACTER
YES
"$" ?
NO
OUT OF RANGE ?
NO
TIMES *3 FOR INDEX
INDEX INTO COMMAND TABLE
OUTØ
GET TABLE ADDRESS COUNT=6
SETUP POINTER COUNT FOR COMMANDS
OUTI
LOAD POINTER THEN COMMAND BUMP PNTR
RIGHT COMMAND ?
NO
POSITION LEFT FIX PNTR FLAG FOR RIGHT
OUT2
TOGGLE FLAG SAVE CMND POINTER
OUT3
MASK CMND CHAR.
OUT4
POINT
MOVE TO NEXT POINT
OUT4 A
NO
ALL 6 CMNDS ?
YES
OUT6
VECTR
CHARACTER SPACE
PENU
LIFT PEN
MOVQ
POSITION FOR NEXT CHAR.
SAVE
SAVE POSITION
YES
END ?
NO
YES
POINT A ?
NO
POINT B ?
YES
DELB
NO
POINT C ?
YES
DELC
NO
POINT D ?
YES
DELD
NO
POINT E ?
YES
DELE
NO
POINT F ?
YES
DELF
NO
POINT G ?
YES
DELG
NO
POINT H ?
YES
DELH
NO
POINT I ?
YES
DELI
NO
LIFT PEN ?
YES
LIPT
NO
SEQ I ?
YES
SEQI
NO
SEQ2

FIG 74U

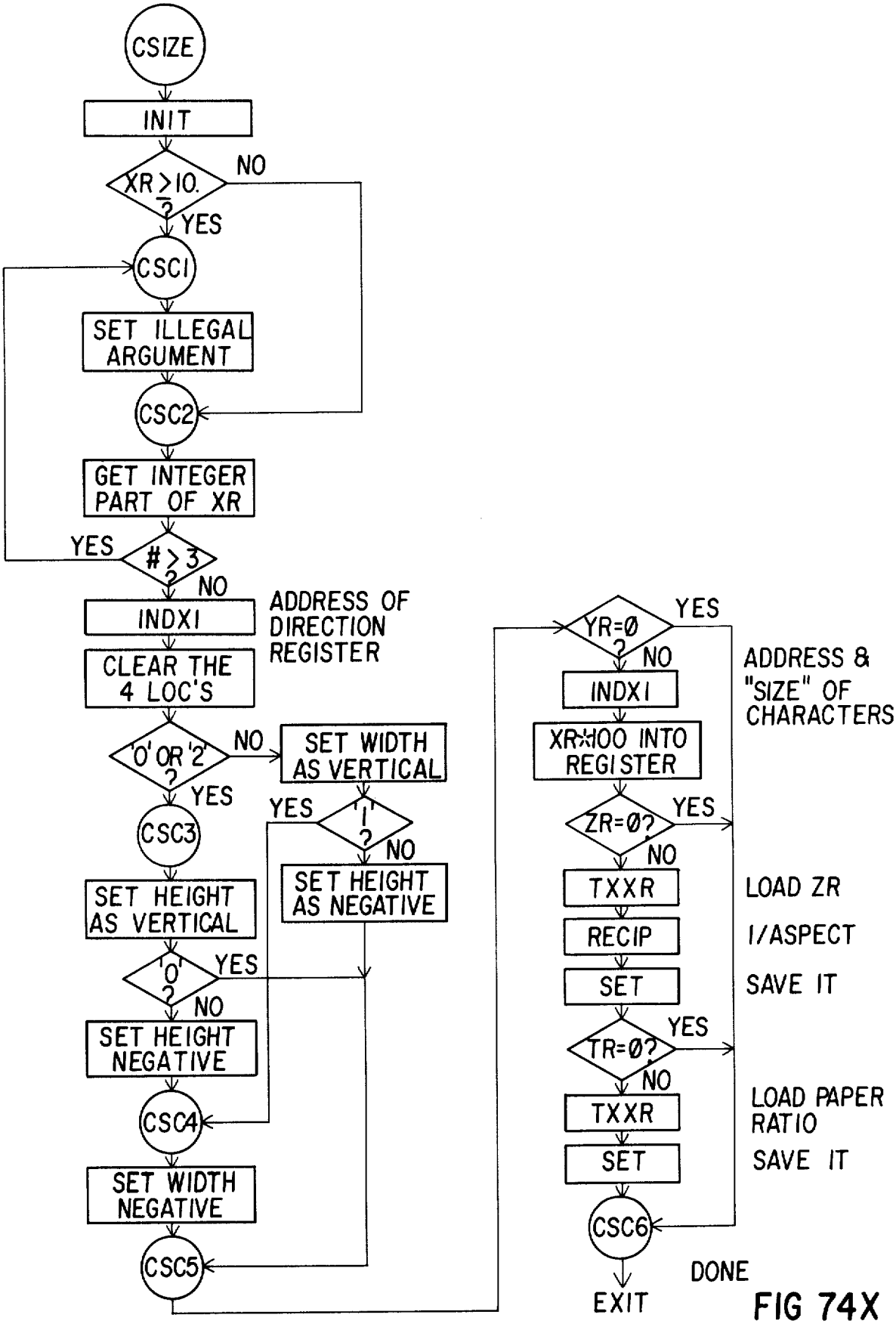

CSIZE
INIT
XR ≥ 10. ?
NO
YES
CSC1
SET ILLEGAL ARGUMENT
CSC2
GET INTEGER PART OF XR
YES
> 3 ?
NO
INDX1
ADDRESS OF DIRECTION REGISTER
CLEAR THE 4 LOC'S
'0' OR '2' ?
NO
SET WIDTH AS VERTICAL
YES
CSC3
YES
'1' ?
NO
SET HEIGHT AS NEGATIVE
SET HEIGHT AS VERTICAL
'0' ?
YES
NO
SET HEIGHT NEGATIVE
CSC4
SET WIDTH NEGATIVE
CSC5
YR=0 ?
YES
NO
INDX1
ADDRESS & "SIZE" OF CHARACTERS
XR*100 INTO REGISTER
ZR=0?
YES
NO
TXXR
LOAD ZR
RECIP
1/ASPECT
SET
SAVE IT
TR=0?
YES
NO
TXXR
LOAD PAPER RATIO
SET
SAVE IT
CSC6
DONE
EXIT

FIG 74X

FIG. 75

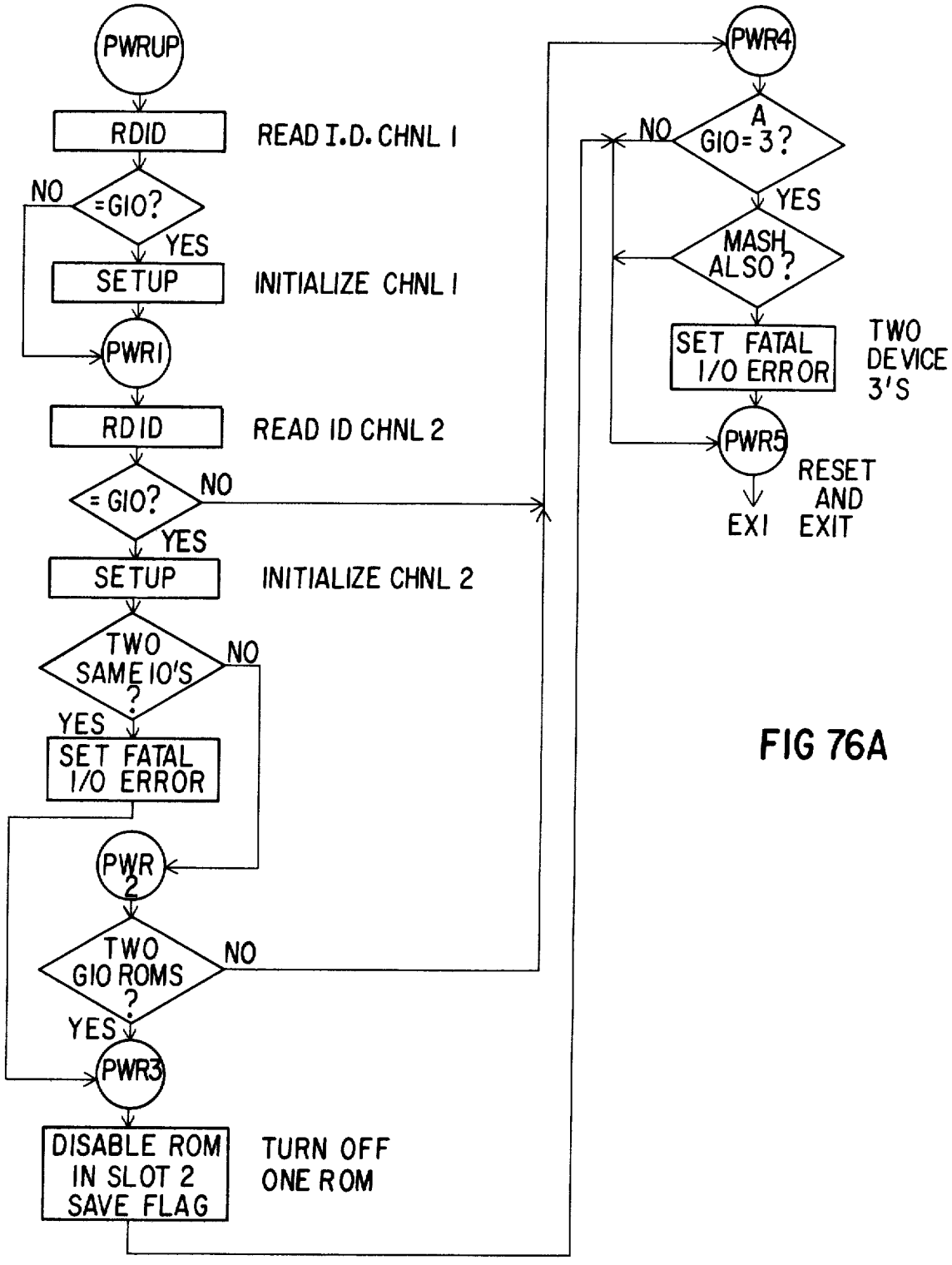

PWRUP
RDID
READ I.D. CHNL 1
NO
=GIO?
YES
SETUP
INITIALIZE CHNL 1
PWR1
RDID
READ ID CHNL 2
= GIO?
NO
YES
SETUP
INITIALIZE CHNL 2
TWO SAME IO'S ?
NO
YES
SET FATAL I/O ERROR
PWR 2
TWO GIO ROMS ?
NO
YES
PWR3
DISABLE ROM IN SLOT 2 SAVE FLAG
TURN OFF ONE ROM
PWR4
NO
A GIO=3?
YES
MASH ALSO?
SET FATAL I/O ERROR
TWO DEVICE 3'S
PWR5
RESET AND EXIT
EXI

FIG 76A

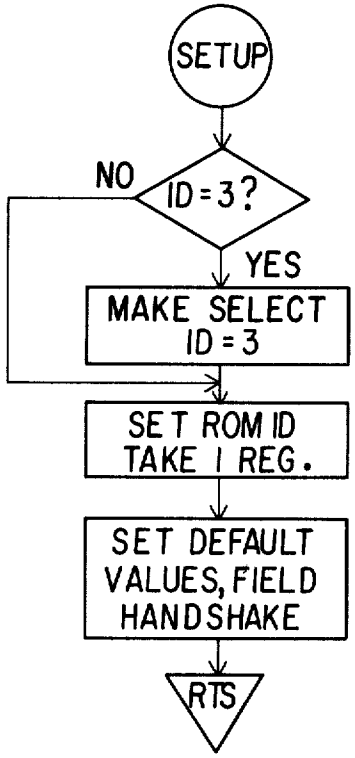


FIG 76B

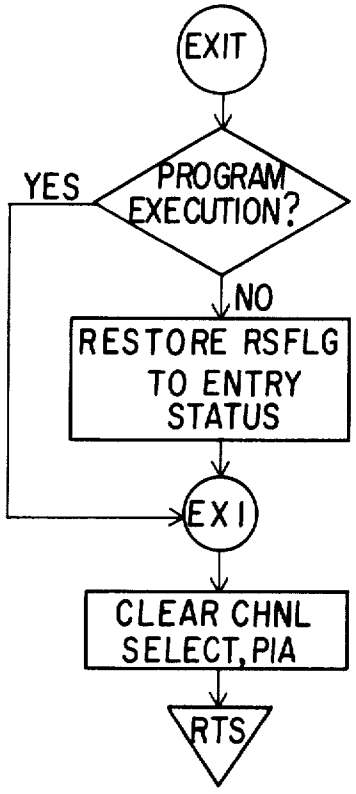


FIG 76C

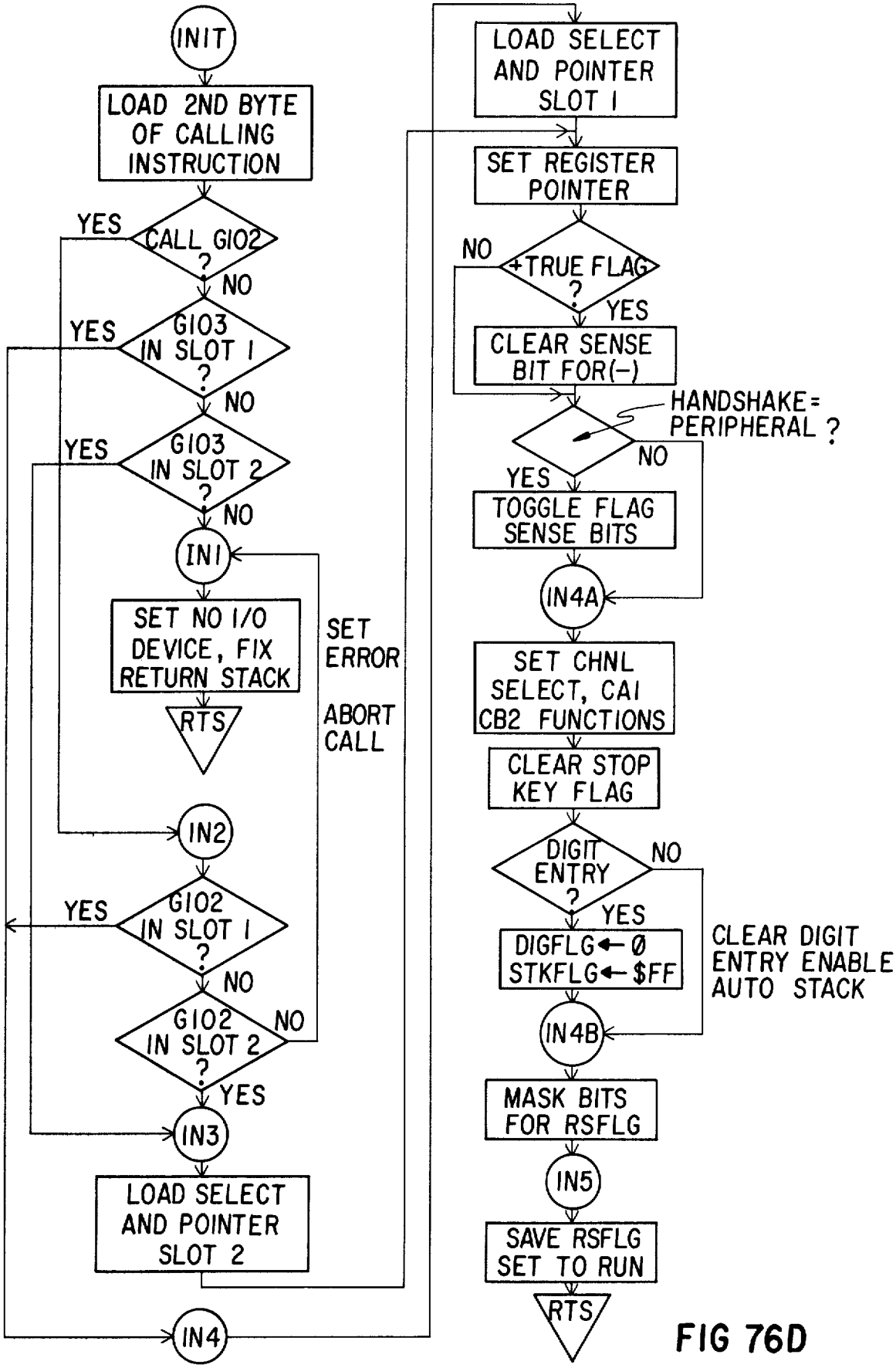

INIT
LOAD 2ND BYTE OF CALLING INSTRUCTION
YES
CALL GIO2 ?
NO
YES
GIO3 IN SLOT 1 ?
NO
YES
GIO3 IN SLOT 2 ?
NO
IN1
SET NO I/O DEVICE, FIX RETURN STACK
SET ERROR
ABORT CALL
RTS
IN2
YES
GIO2 IN SLOT 1 ?
NO
GIO2 IN SLOT 2 ?
NO
YES
IN3
LOAD SELECT AND POINTER SLOT 2
IN4
LOAD SELECT AND POINTER SLOT 1
SET REGISTER POINTER
NO
+TRUE FLAG ?
YES
CLEAR SENSE BIT FOR(-)
HANDSHAKE= PERIPHERAL ?
NO
YES
TOGGLE FLAG SENSE BITS
IN4A
SET CHNL SELECT, CAI CB2 FUNCTIONS
CLEAR STOP KEY FLAG
DIGIT ENTRY ?
NO
YES
DIGFLG ← Ø
STKFLG ← $FF
CLEAR DIGIT ENTRY ENABLE AUTO STACK
IN4B
MASK BITS FOR RSFLG
IN5
SAVE RSFLG SET TO RUN
RTS

FIG 76D

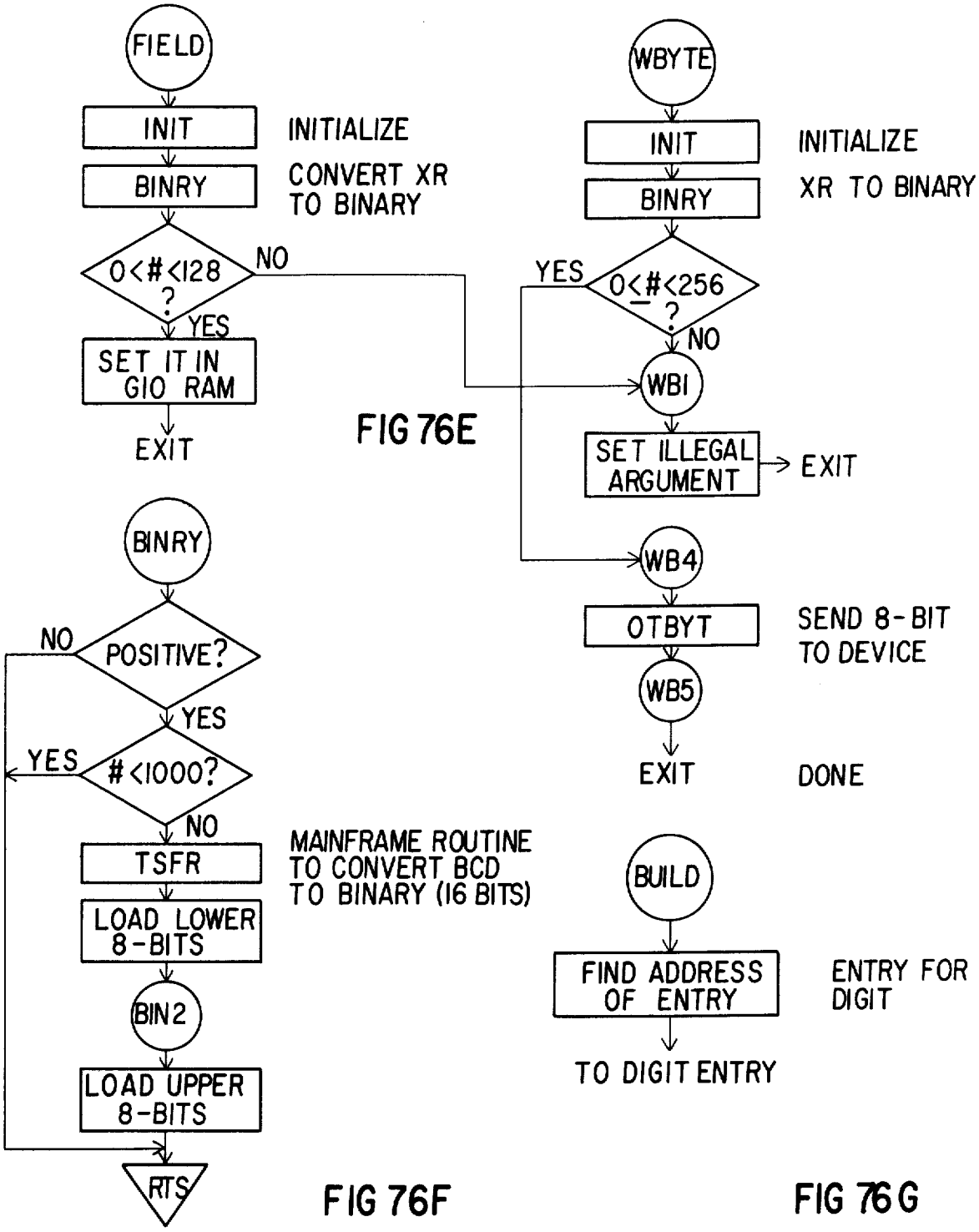


FIG 76E

FIG 76F

FIG 76G

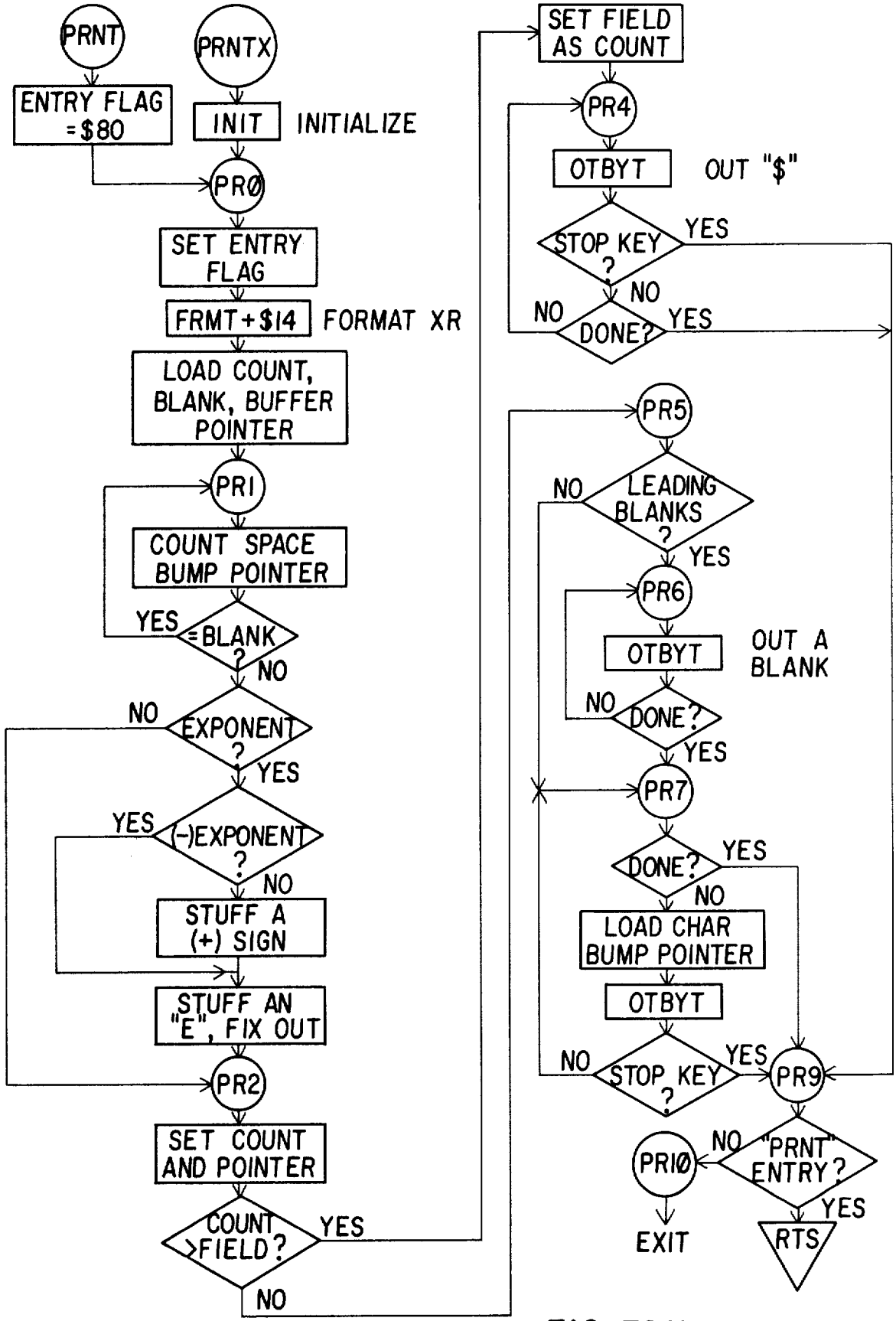

PRNT
PRNTX
ENTRY FLAG =$80
INIT
INITIALIZE
PRØ
SET ENTRY FLAG
FRMT+$14
FORMAT XR
LOAD COUNT, BLANK, BUFFER POINTER
PR1
COUNT SPACE BUMP POINTER
YES
=BLANK ?
NO
NO
EXPONENT ?
YES
YES
(-)EXPONENT ?
NO
STUFF A (+) SIGN
STUFF AN "E", FIX OUT
PR2
SET COUNT AND POINTER
COUNT >FIELD ?
YES
NO
SET FIELD AS COUNT
PR4
OTBYT
OUT "$"
STOP KEY ?
YES
NO
NO
DONE?
YES
PR5
NO
LEADING BLANKS ?
YES
PR6
OTBYT
OUT A BLANK
NO
DONE?
YES
PR7
DONE?
YES
NO
LOAD CHAR BUMP POINTER
OTBYT
NO
STOP KEY ?
YES
PR9
PRIØ
NO
"PRNT" ENTRY?
YES
EXIT
RTS

FIG 76H

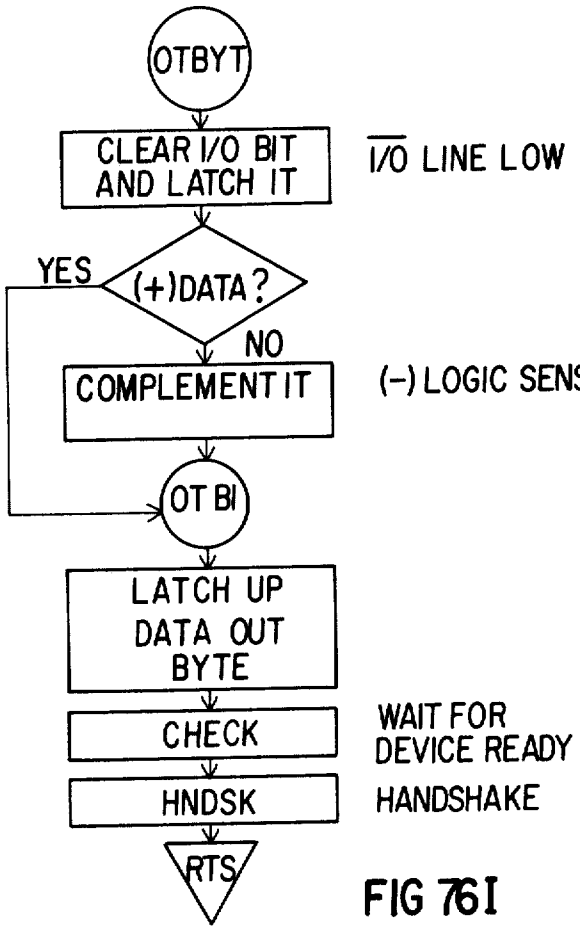


FIG 76I

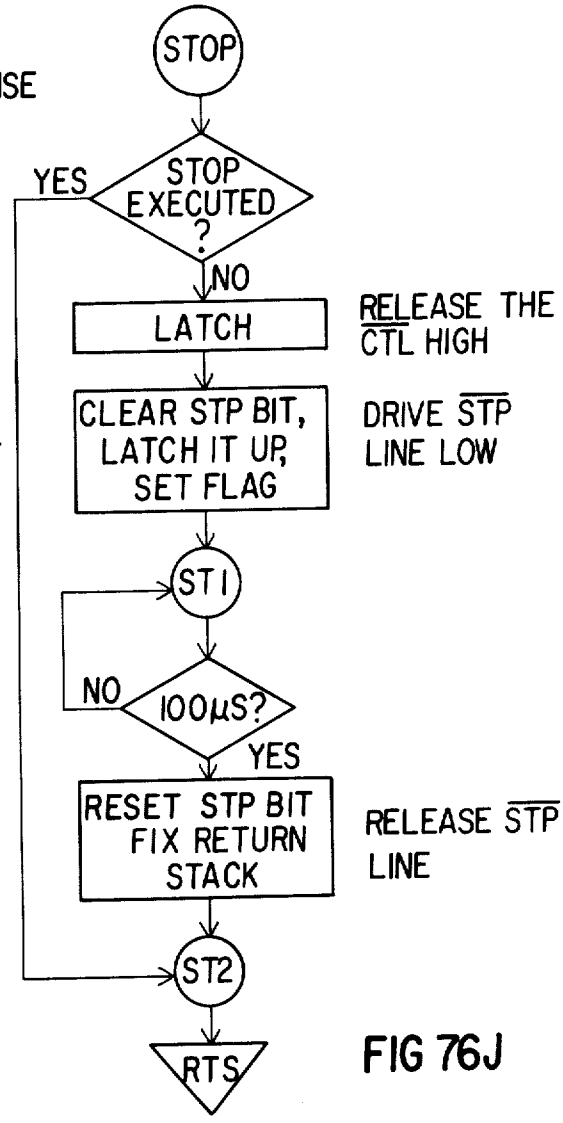


FIG 76J

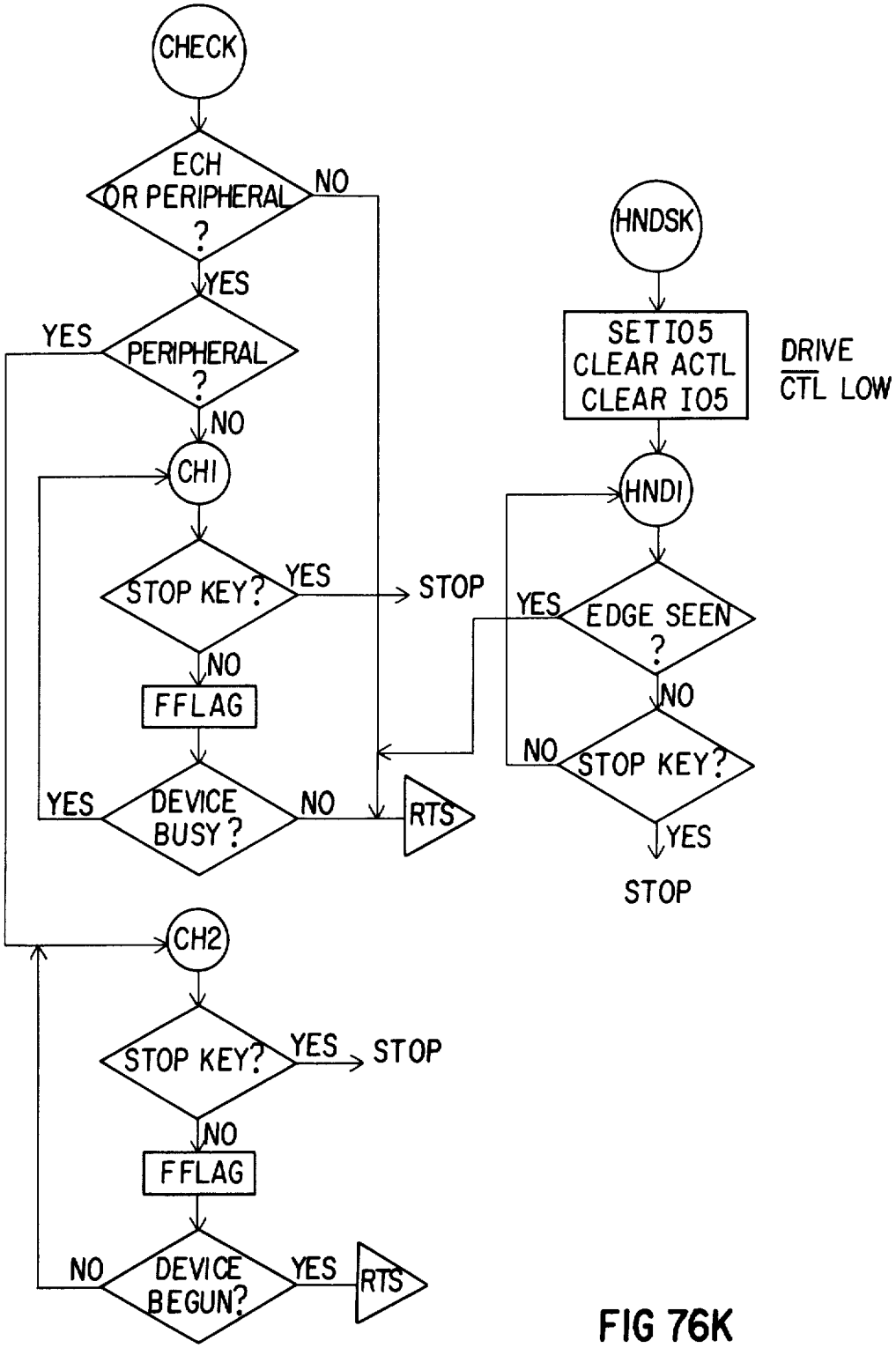

CHECK
ECH OR PERIPHERAL ?
NO
YES
PERIPHERAL ?
YES
NO
CHI
STOP KEY?
YES
STOP
NO
FFLAG
DEVICE BUSY ?
YES
NO
RTS
CH2
STOP KEY?
YES
STOP
NO
FFLAG
DEVICE BEGUN?
NO
YES
RTS
HNDSK
SETIO5
CLEAR ACTL
CLEAR IO5
DRIVE CTL LOW
HNDI
EDGE SEEN ?
YES
NO
STOP KEY?
NO
YES
STOP

FIG 76K

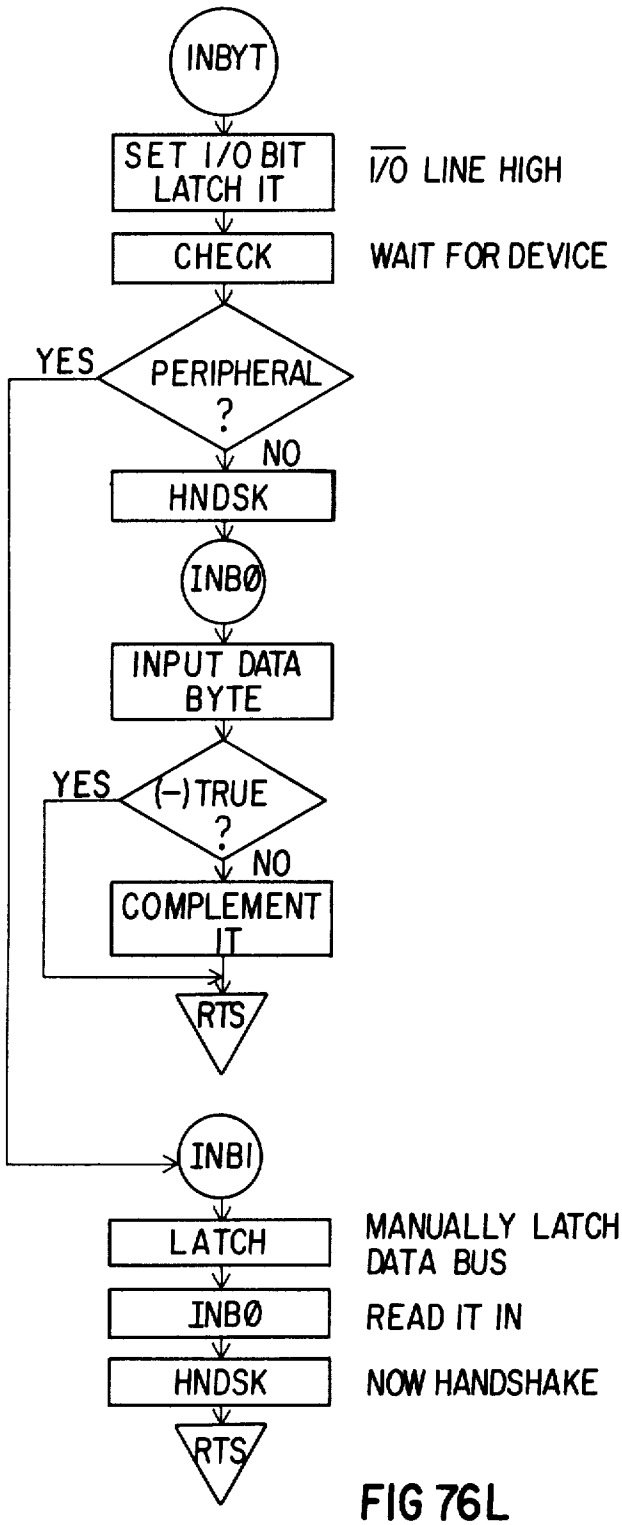


FIG 76L

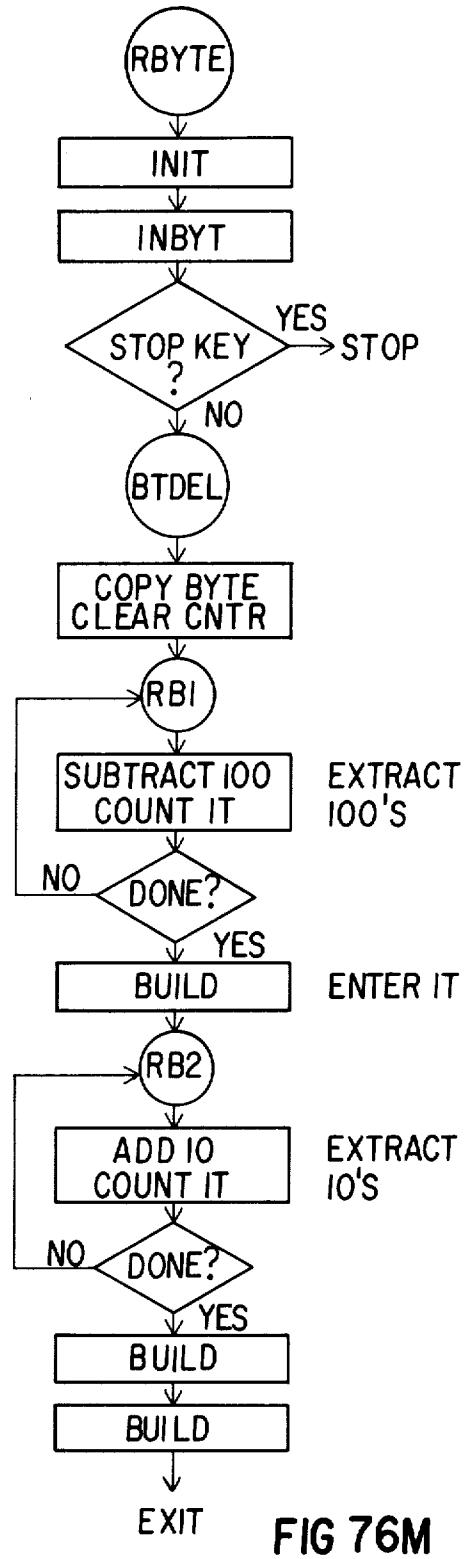


FIG 76M

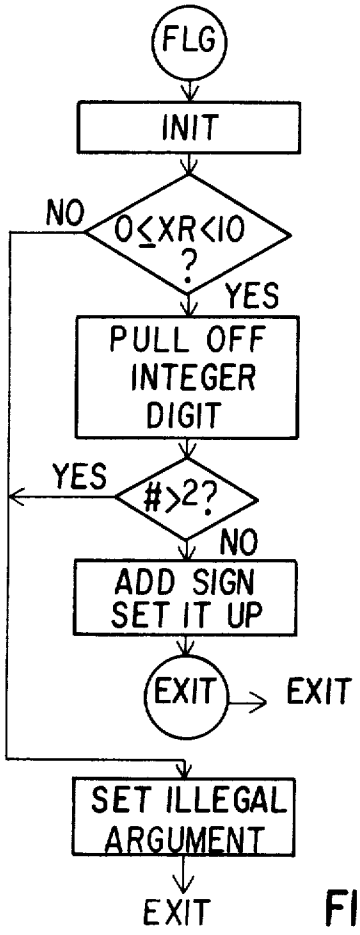


FIG 76N

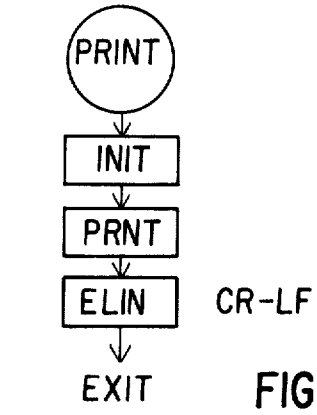


FIG 76O

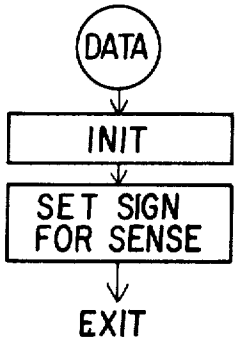


FIG 76P

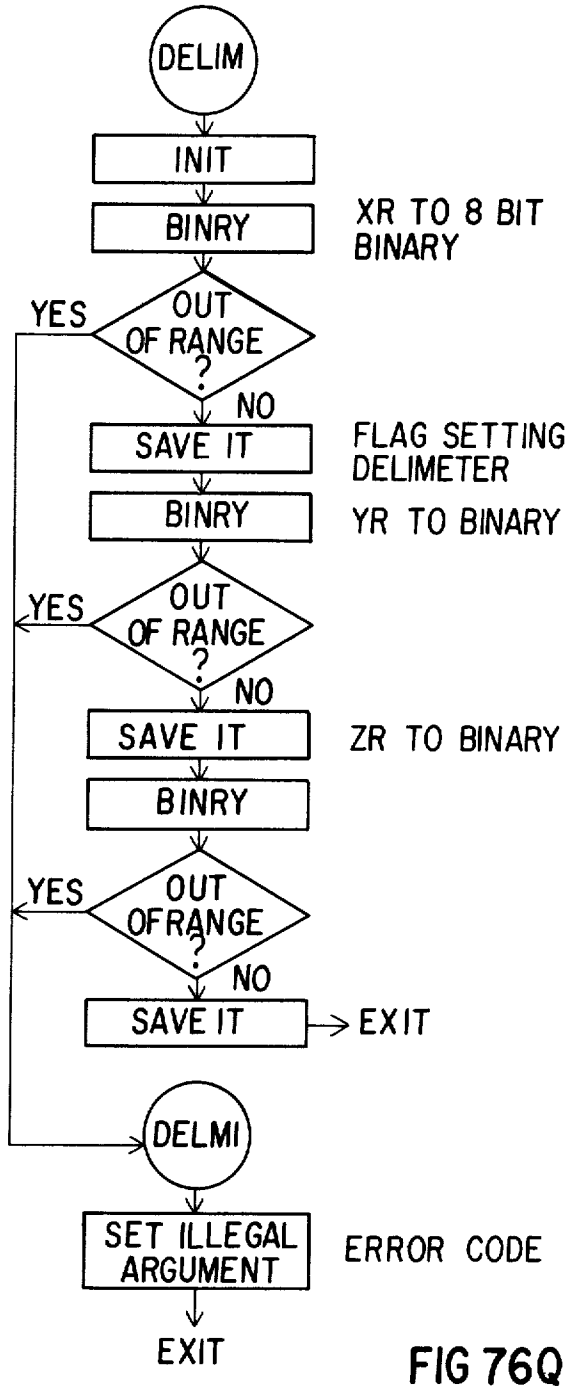


FIG 76Q

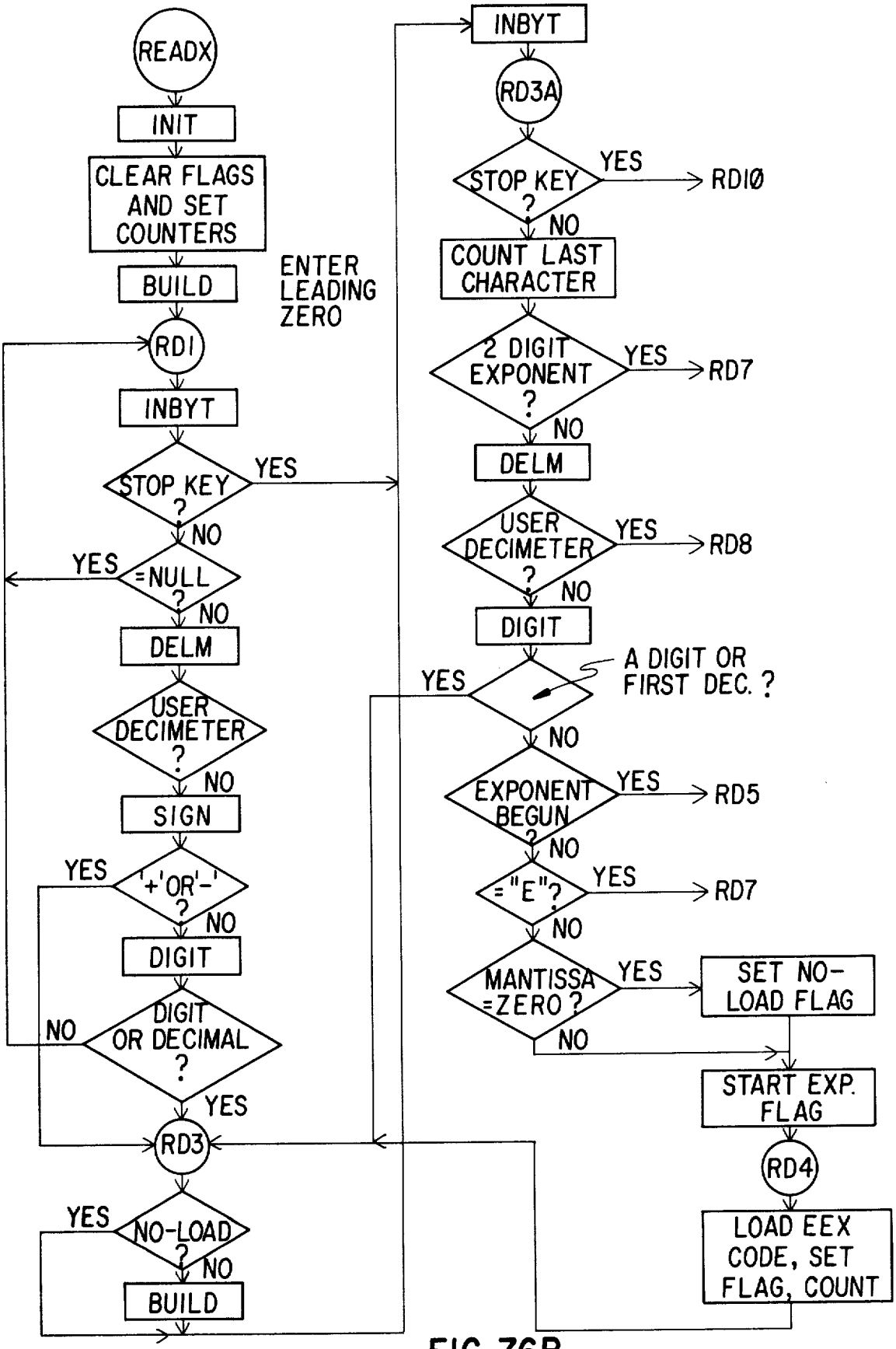

READX
INIT
CLEAR FLAGS AND SET COUNTERS
BUILD
ENTER LEADING ZERO
RD1
INBYT
STOP KEY ?
YES
NO
=NULL ?
YES
NO
DELM
USER DECIMETER ?
NO
SIGN
'+'OR'-' ?
YES
NO
DIGIT
DIGIT OR DECIMAL ?
NO
YES
RD3
NO-LOAD ?
YES
NO
BUILD
INBYT
RD3A
STOP KEY ?
YES
RD10
NO
COUNT LAST CHARACTER
2 DIGIT EXPONENT ?
YES
RD7
NO
DELM
USER DECIMETER ?
YES
RD8
NO
DIGIT
A DIGIT OR FIRST DEC. ?
YES
NO
EXPONENT BEGUN ?
YES
RD5
NO
="E"?
YES
RD7
NO
MANTISSA =ZERO ?
YES
SET NO-LOAD FLAG
NO
START EXP. FLAG
RD4
LOAD EEX CODE, SET FLAG, COUNT

FIG 76R

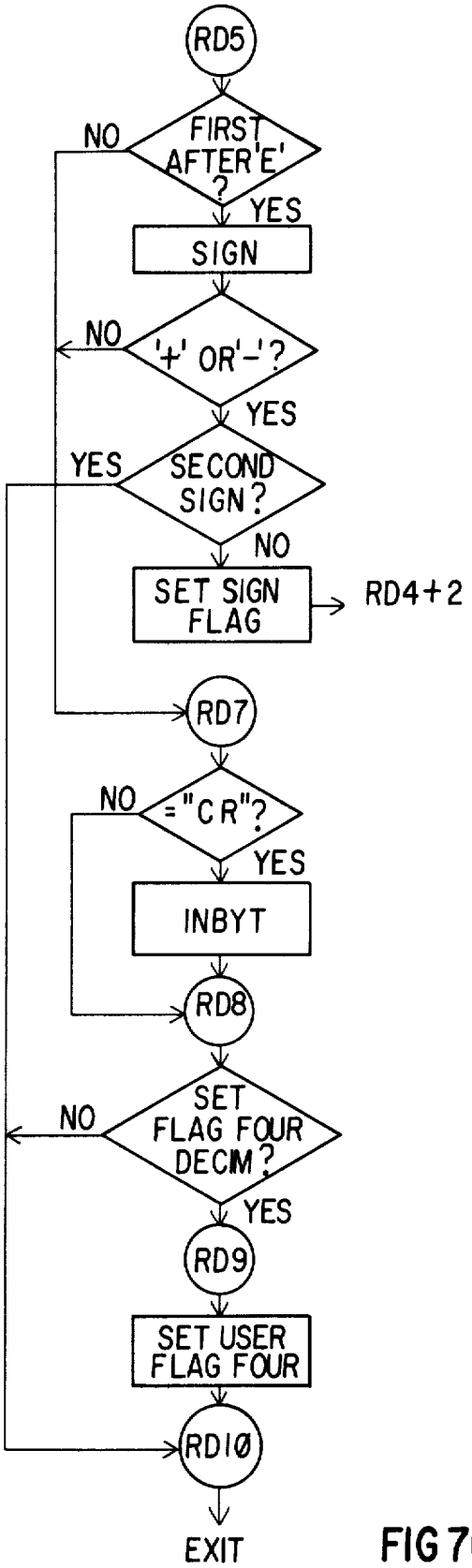


FIG 76S

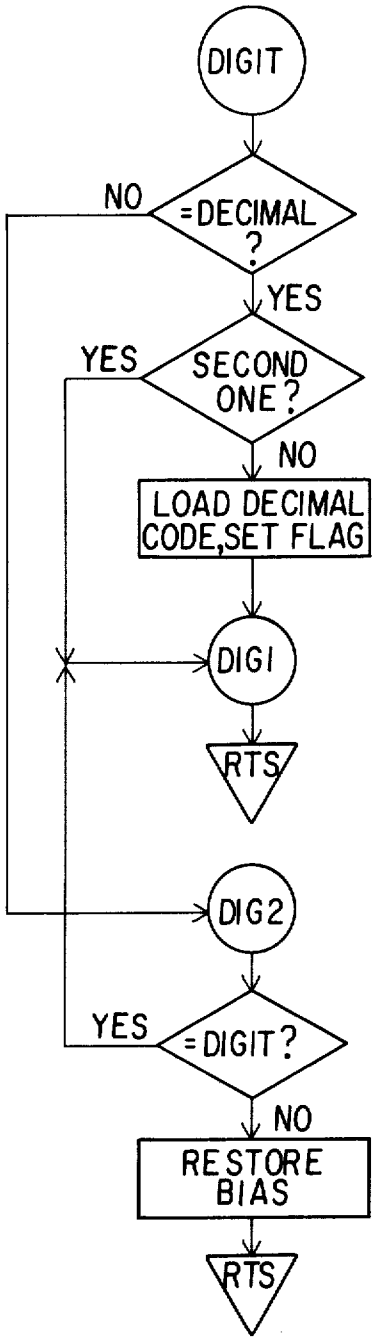


FIG 76T

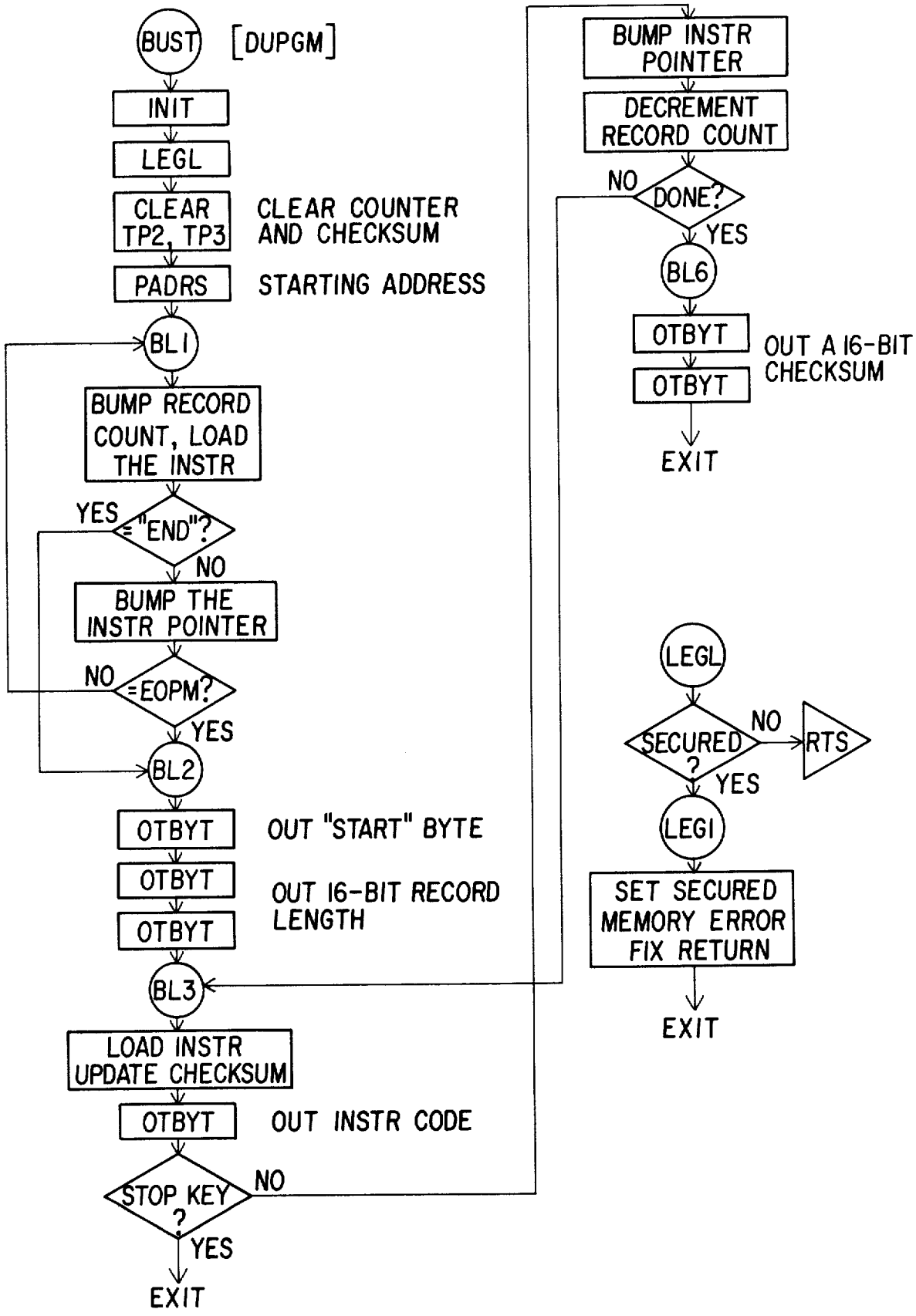

BUST
[DUPGM]
INIT
LEGL
CLEAR TP2, TP3
CLEAR COUNTER AND CHECKSUM
PADRS
STARTING ADDRESS
BL1
BUMP RECORD COUNT, LOAD THE INSTR
YES
="END"?
NO
BUMP THE INSTR POINTER
NO
=EOPM?
YES
BL2
OTBYT
OUT "START" BYTE
OTBYT
OTBYT
OUT 16-BIT RECORD LENGTH
BL3
LOAD INSTR UPDATE CHECKSUM
OTBYT
OUT INSTR CODE
STOP KEY ?
NO
YES
EXIT
BUMP INSTR POINTER
DECREMENT RECORD COUNT
NO
DONE?
YES
BL6
OTBYT
OTBYT
OUT A 16-BIT CHECKSUM
EXIT
LEGL
SECURED ?
NO
RTS
YES
LEGI
SET SECURED MEMORY ERROR FIX RETURN
EXIT

FIG 76U

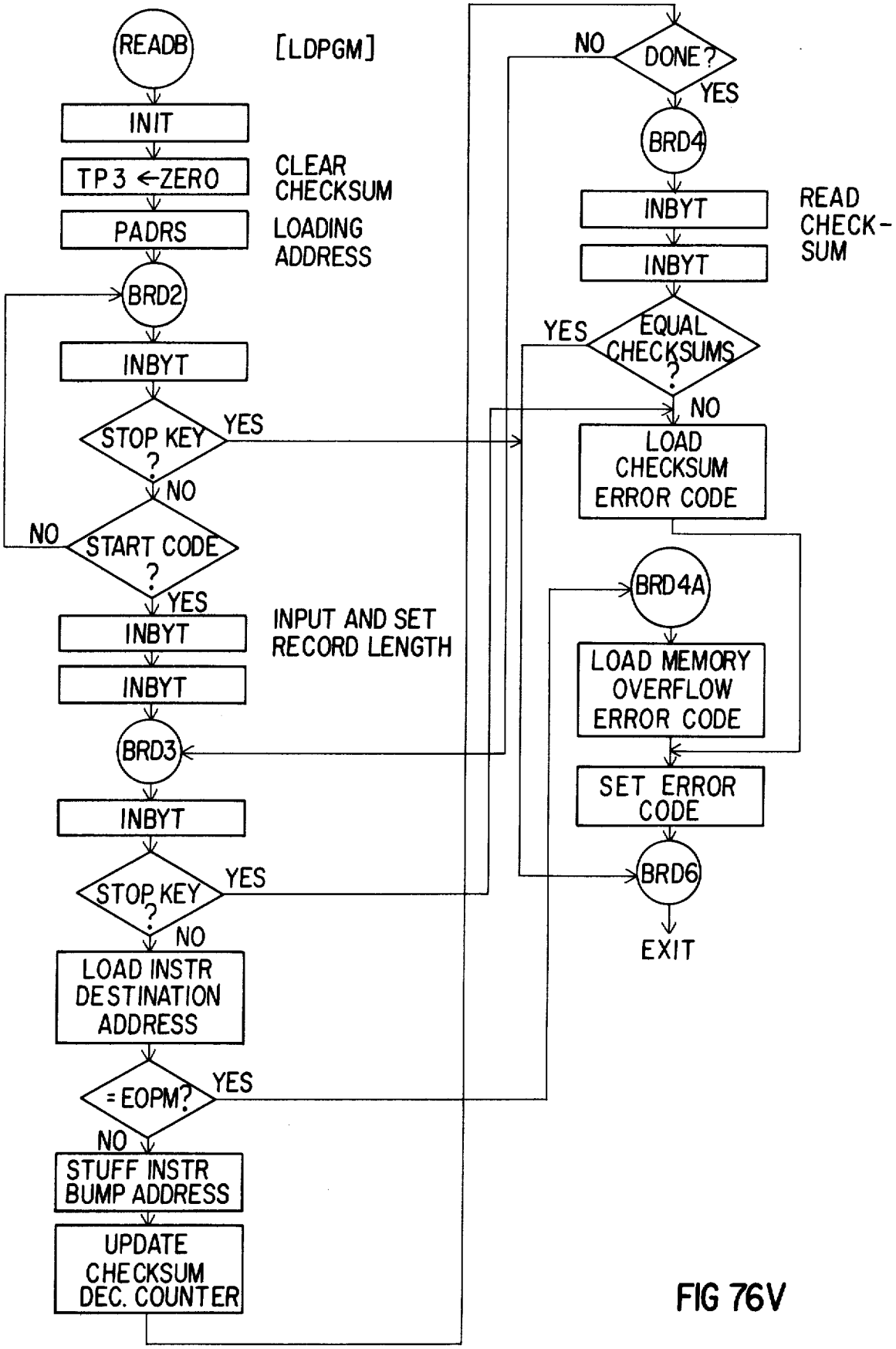

READB
[LDPGM]
INIT
TP3 ←ZERO
CLEAR CHECKSUM
PADRS
LOADING ADDRESS
BRD2
INBYT
STOP KEY ?
YES
NO
START CODE ?
NO
YES
INBYT
INBYT
INPUT AND SET RECORD LENGTH
BRD3
INBYT
STOP KEY ?
YES
NO
LOAD INSTR DESTINATION ADDRESS
= EOPM?
YES
NO
STUFF INSTR BUMP ADDRESS
UPDATE CHECKSUM DEC. COUNTER
DONE?
NO
YES
BRD4
INBYT
INBYT
READ CHECK-SUM
EQUAL CHECKSUMS ?
YES
NO
LOAD CHECKSUM ERROR CODE
BRD4A
LOAD MEMORY OVERFLOW ERROR CODE
SET ERROR CODE
BRD6
EXIT

FIG 76V

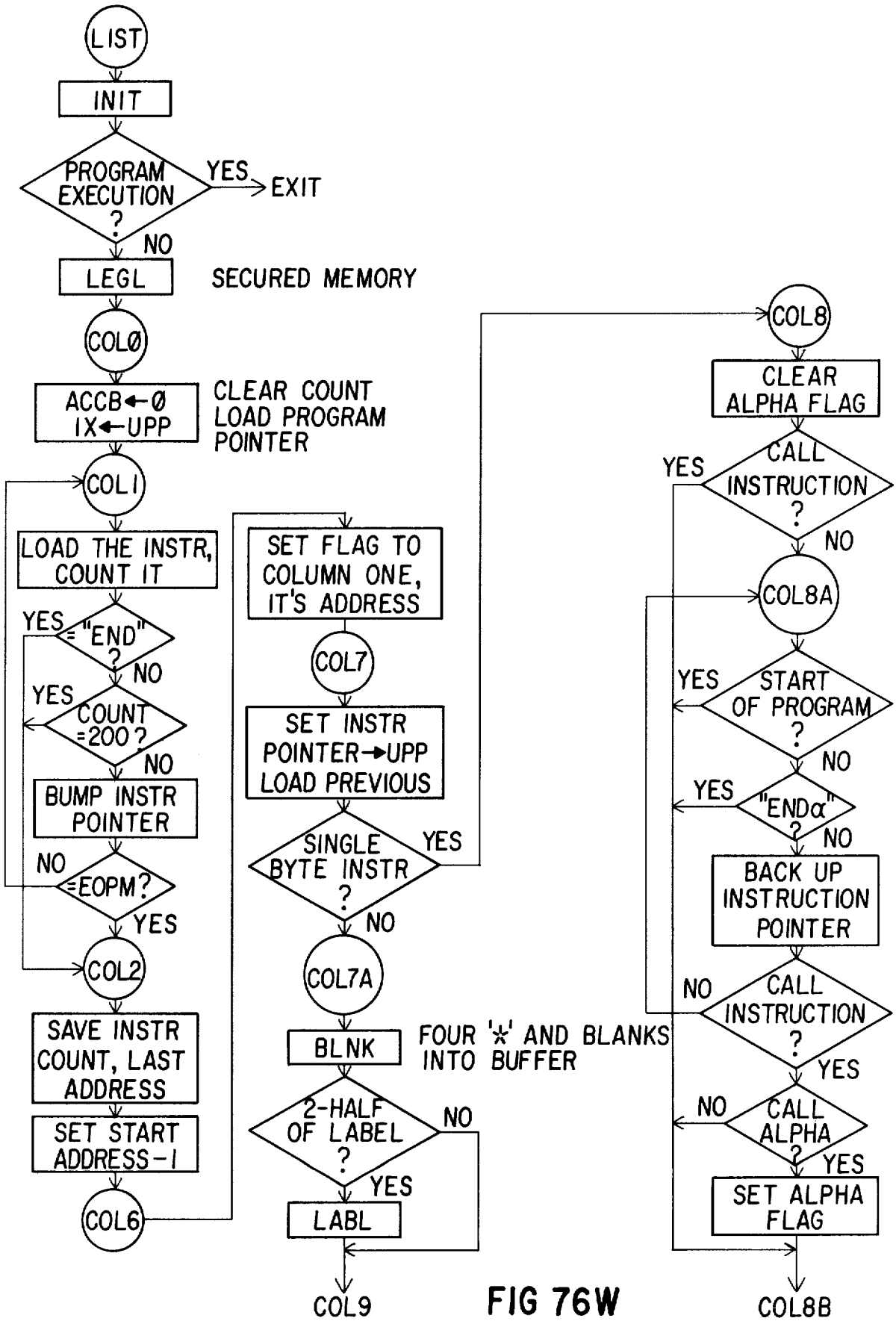
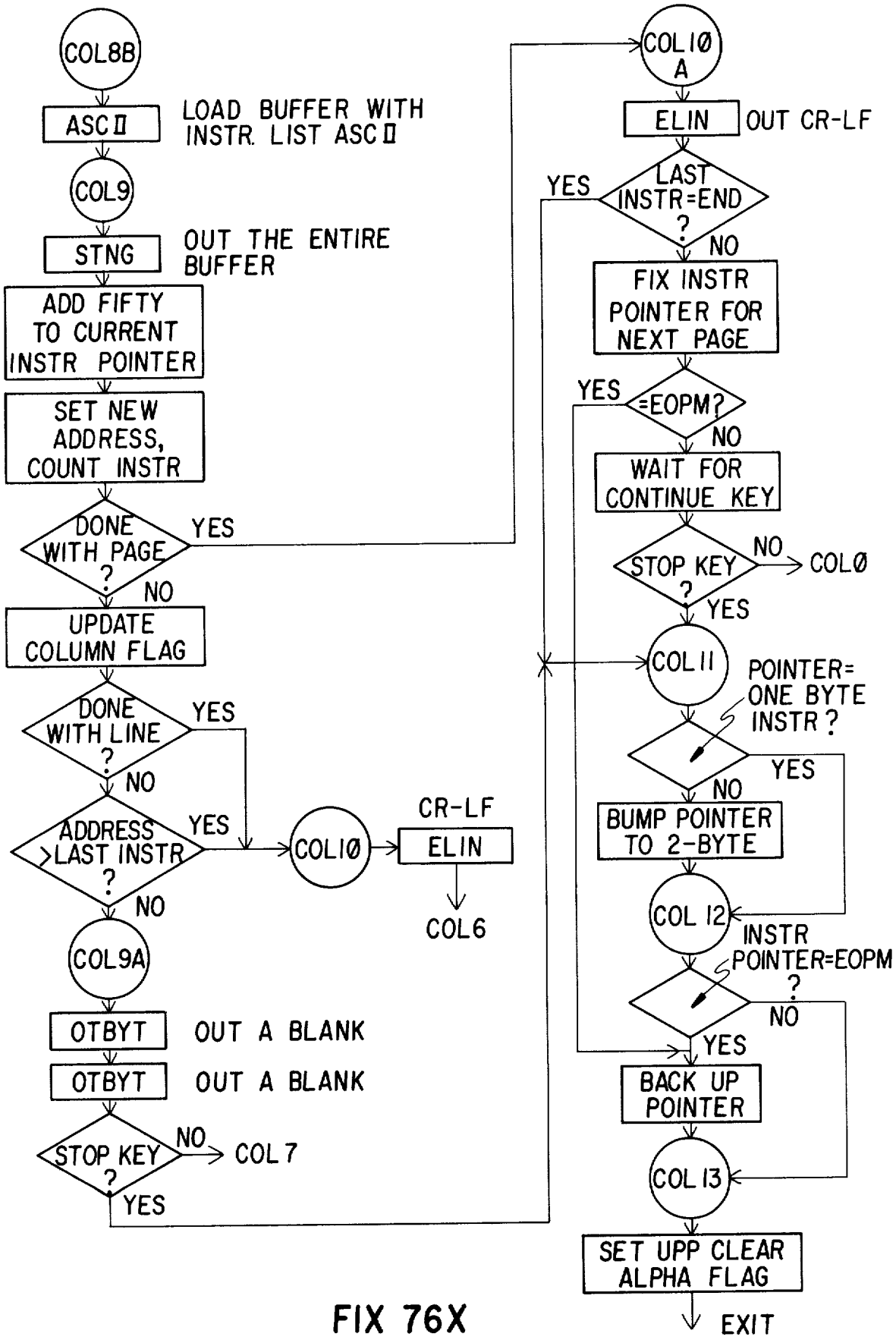
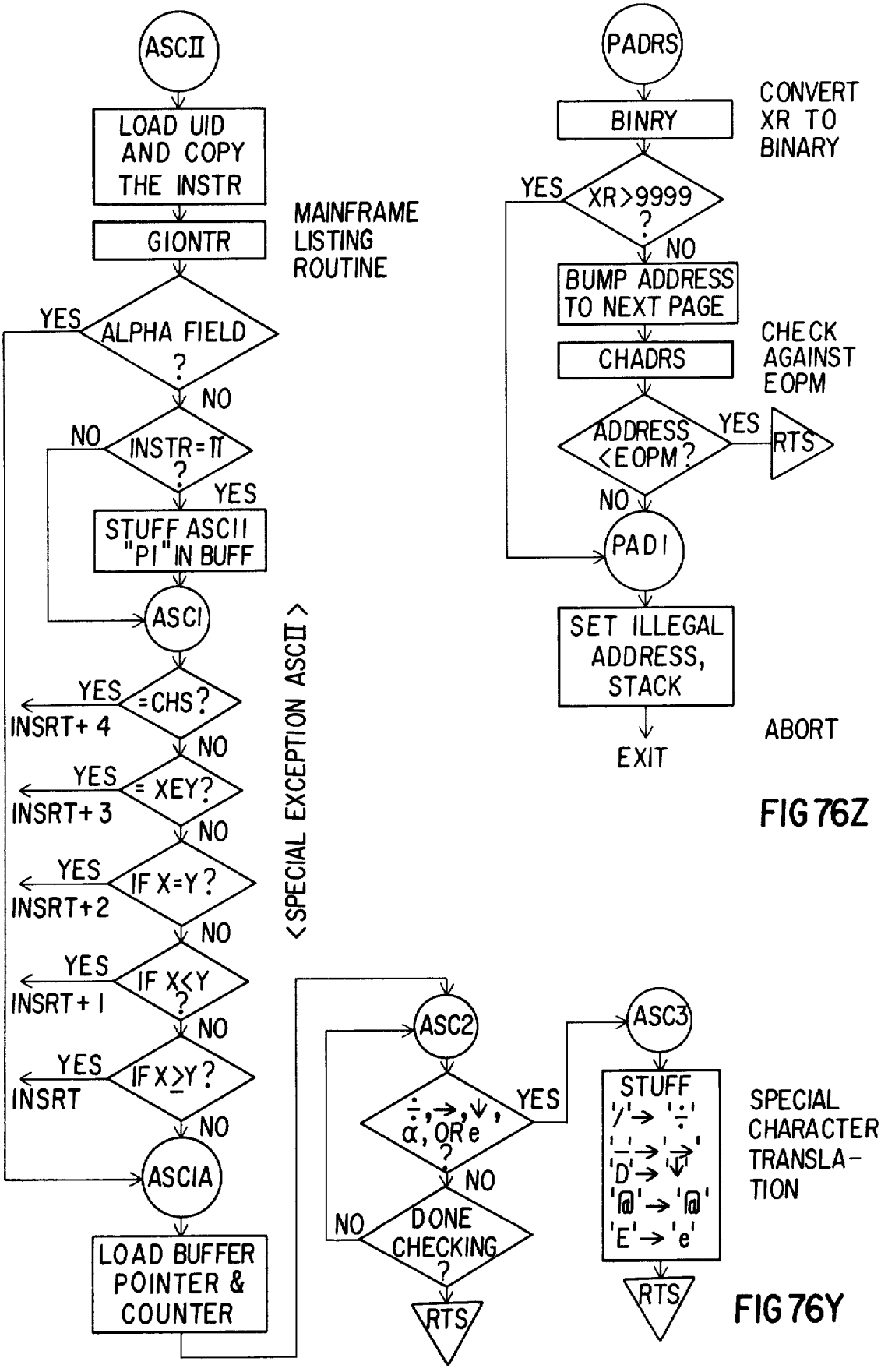

LIST
INIT
PROGRAM EXECUTION ?
YES
EXIT
NO
LEGL
SECURED MEMORY
COLØ
ACCB←Ø
IX←UPP
CLEAR COUNT
LOAD PROGRAM POINTER
COL1
LOAD THE INSTR, COUNT IT
YES
="END" ?
NO
YES
COUNT =200 ?
NO
BUMP INSTR POINTER
NO
=EOPM?
YES
COL2
SAVE INSTR COUNT, LAST ADDRESS
SET START ADDRESS-1
COL6
SET FLAG TO COLUMN ONE, IT'S ADDRESS
COL7
SET INSTR POINTER→UPP
LOAD PREVIOUS
SINGLE BYTE INSTR ?
YES
NO
COL7A
BLNK
FOUR '*' AND BLANKS INTO BUFFER
2-HALF OF LABEL ?
NO
YES
LABL
COL9
COL8
CLEAR ALPHA FLAG
YES
CALL INSTRUCTION ?
NO
COL8A
YES
START OF PROGRAM ?
NO
YES
"ENDα" ?
NO
BACK UP INSTRUCTION POINTER
NO
CALL INSTRUCTION ?
YES
NO
CALL ALPHA ?
YES
SET ALPHA FLAG
COL8B

FIG 76W


COL8B
ASCII
LOAD BUFFER WITH INSTR. LIST ASCII
COL9
STNG
OUT THE ENTIRE BUFFER
ADD FIFTY TO CURRENT INSTR POINTER
SET NEW ADDRESS, COUNT INSTR
DONE WITH PAGE ?
YES
NO
UPDATE COLUMN FLAG
DONE WITH LINE ?
YES
NO
ADDRESS >LAST INSTR ?
YES
NO
COL10
CR-LF
ELIN
COL6
COL9A
OTBYT
OUT A BLANK
OTBYT
OUT A BLANK
STOP KEY ?
NO
COL7
YES
COL10 A
ELIN
OUT CR-LF
LAST INSTR=END ?
YES
NO
FIX INSTR POINTER FOR NEXT PAGE
=EOPM?
YES
NO
WAIT FOR CONTINUE KEY
STOP KEY ?
NO
COL0
YES
COL11
POINTER= ONE BYTE INSTR ?
YES
NO
BUMP POINTER TO 2-BYTE
COL12
INSTR POINTER=EOPM ?
NO
YES
BACK UP POINTER
COL13
SET UPP CLEAR ALPHA FLAG
EXIT

FIX 76X


ASCII
LOAD UID AND COPY THE INSTR
GIONTR
MAINFRAME LISTING ROUTINE
YES
ALPHA FIELD ?
NO
NO
INSTR=II ?
YES
STUFF ASCII "PI" IN BUFF
ASCI
YES
=CHS?
INSRT+4
NO
YES
=XEY?
INSRT+3
NO
YES
IF X=Y?
INSRT+2
NO
YES
IF X<Y ?
INSRT+1
NO
YES
IF X≥Y?
INSRT
NO
ASCIA
LOAD BUFFER POINTER & COUNTER
<SPECIAL EXCEPTION ASCII>
ASC2
YES
÷, →, ↓, α, OR e ?
NO
NO
DONE CHECKING ?
RTS
ASC3
STUFF
'/' → '÷'
'_' → '→'
'D' → '↓'
'@' → '@'
'E' → 'e'
RTS
SPECIAL CHARACTER TRANSLATION
FIG 76Y
PADRS
BINRY
CONVERT XR TO BINARY
YES
XR>9999 ?
NO
BUMP ADDRESS TO NEXT PAGE
CHADRS
CHECK AGAINST EOPM
ADDRESS <EOPM?
YES
RTS
NO
PADI
SET ILLEGAL ADDRESS, STACK
ABORT
EXIT
FIG 76Z

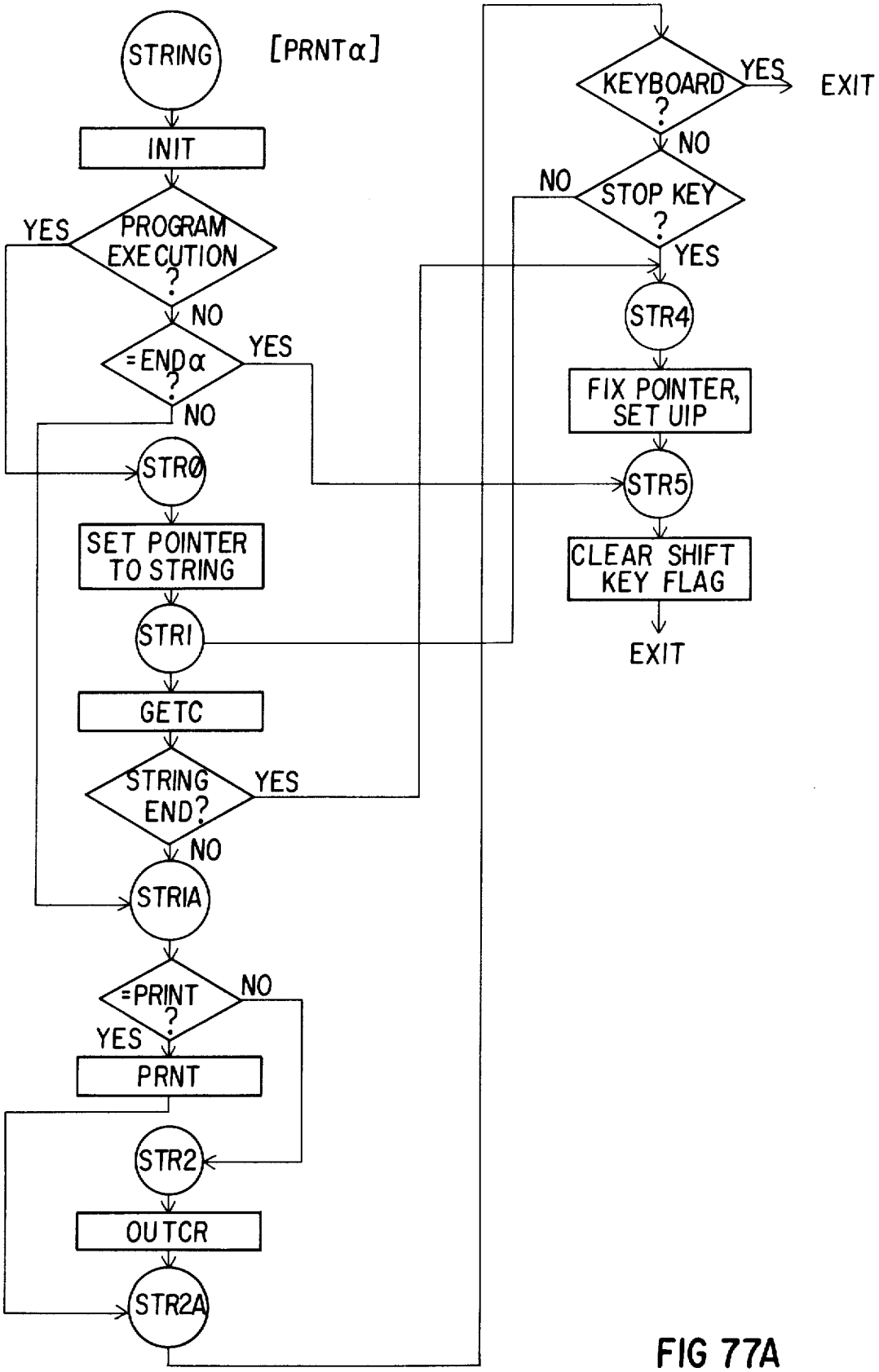

STRING
[PRNTα]
INIT
PROGRAM EXECUTION ?
YES
NO
=ENDα ?
YES
NO
STRØ
SET POINTER TO STRING
STRI
GETC
STRING END?
YES
NO
STRIA
=PRINT ?
NO
YES
PRNT
STR2
OUTCR
STR2A
KEYBOARD ?
YES
EXIT
NO
STOP KEY ?
NO
YES
STR4
FIX POINTER, SET UIP
STR5
CLEAR SHIFT KEY FLAG
EXIT

FIG 77A

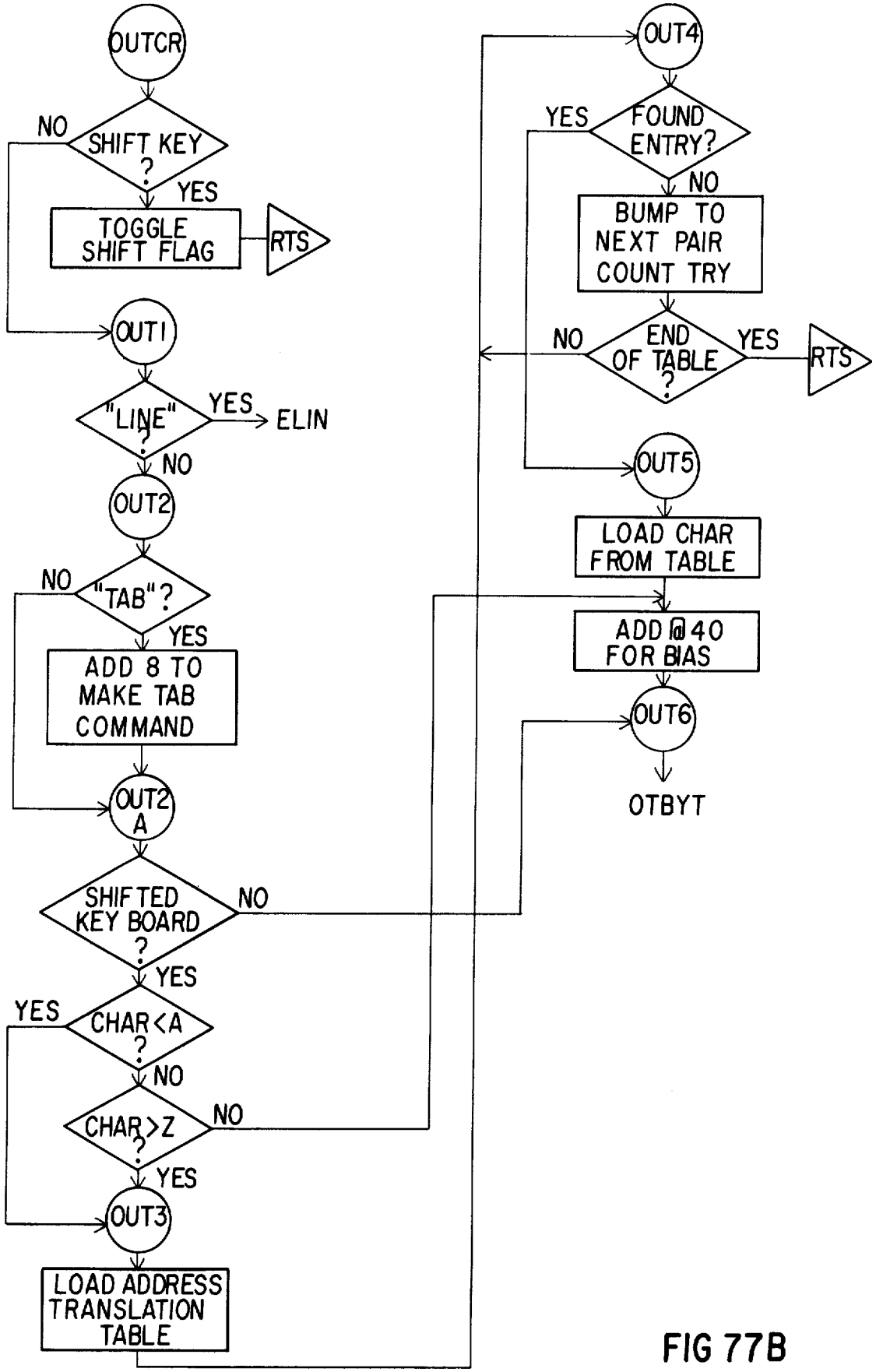

OUTCR
SHIFT KEY ?
NO
YES
TOGGLE SHIFT FLAG
RTS
OUT1
"LINE" ?
YES
ELIN
NO
OUT2
"TAB" ?
NO
YES
ADD 8 TO MAKE TAB COMMAND
OUT2A
SHIFTED KEY BOARD ?
NO
YES
CHAR<A ?
YES
NO
CHAR>Z ?
NO
YES
OUT3
LOAD ADDRESS TRANSLATION TABLE
OUT4
FOUND ENTRY?
YES
NO
BUMP TO NEXT PAIR COUNT TRY
END OF TABLE ?
NO
YES
RTS
OUT5
LOAD CHAR FROM TABLE
ADD @40 FOR BIAS
OUT6
OTBYT

FIG 77B

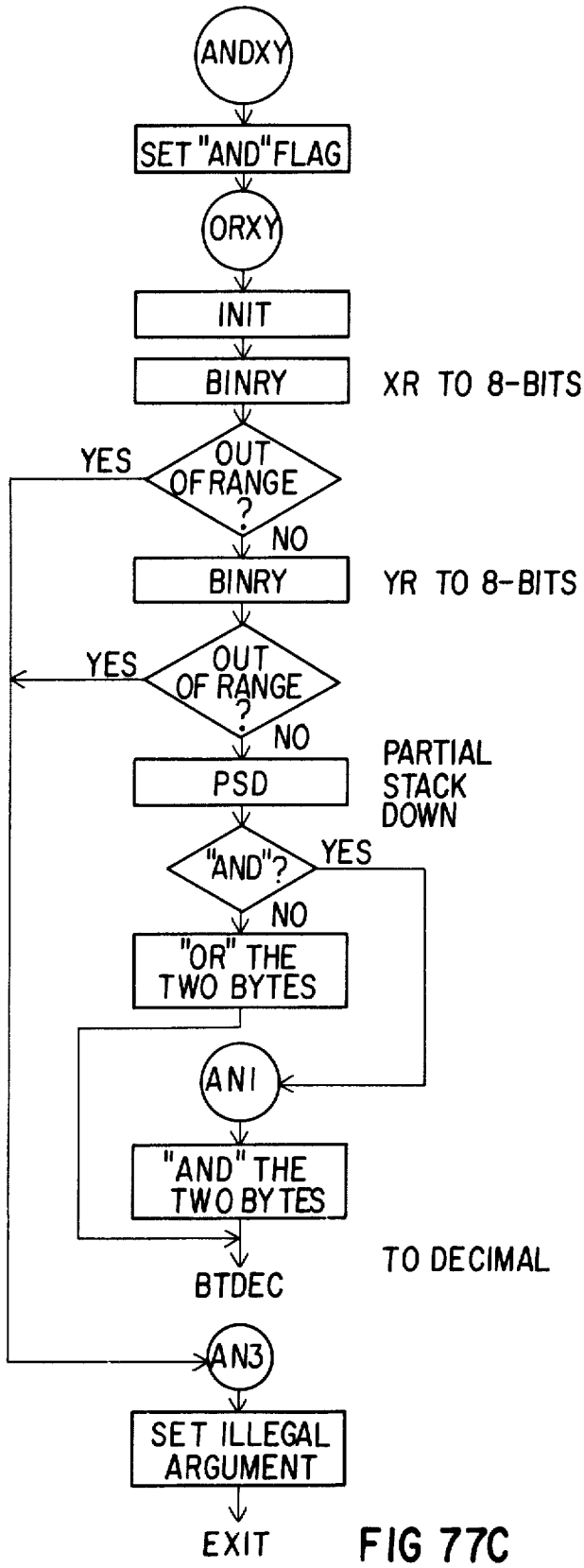


FIG 77C

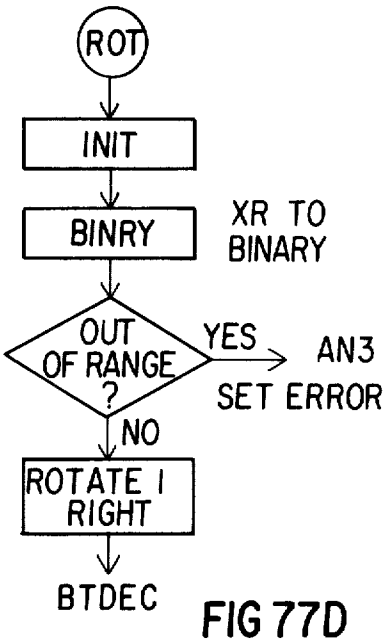


FIG 77D

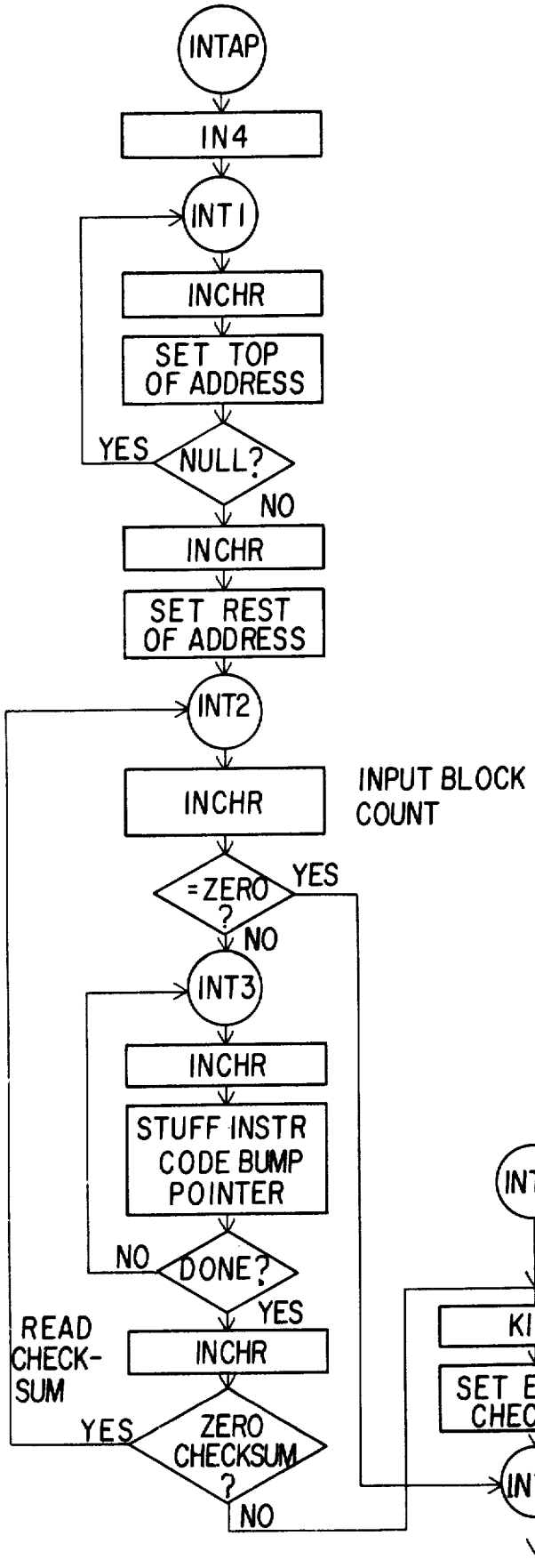


FIG 77E

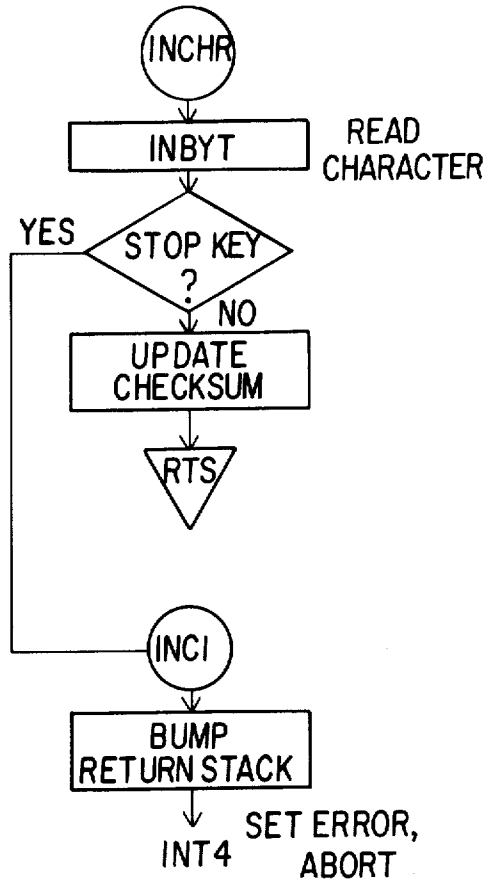


FIG 77F

PROGRAMMABLE CALCULATOR EMPLOYING A READ-WRITE MEMORY HAVING A MOVABLE BOUNDARY BETWEEN PROGRAM AND DATA STORAGE SECTIONS THEREOF

TABLE OF CONTENTS

Title of Section

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 magnetic 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.

Many programmable calculators constructed according to the prior art have employed step oriented memories and have handled memory transfer of conditional or unconditional transfer statements through the use of absolute step references. This technique leaves the user with sole responsibility for statement address modification in the event a transfer statement is edited, thus increasing the user's workload, as well as the chances for introduction of errors, during program editing operations. In addition, these prior art calculators rarely include language features useful in performing iterative looping functions encountered in programming complex problems.

These earlier step oriented calculators produced printed program listings that were very difficult to read because information syntactically representing a single statement was generated by several separate key actuations and then listed in a similar fashion with the information associated with each key being listed on a separate line.

Conventional programmable calculators are limited as to the complexity of the problems they are able to solve because of memory capacity limitations. Magnetic tape storage has been employed in some calculators to store program segments and data for use during execution of a program, thereby effectively increasing the size of the calculater read-write memory. These magnetic tape storage systems have been of limited usefulness, however, because of the relatively long access times involved.

Conventional programmable calculators in the low cost range have presented a communication problem for the user in that they typically have not employed output printers with fully formatted alphanumeric printing capabilities. It would be advantageous in calculators of this type to provide a low cost thermal printer, for example, that may be called upon by the user to print a variety of characters and numeric data according to a format designated by the user.

Conventional programmable calculators have been arranged to respond to power turn on by entering a standby mode, after which the user may enter a program from the keyboard or from a magnetic tape cassette, for example, for execution by the calculator. This arrangement is disadvantageous in that it requires of the user a considerable degree of knowledge regarding operation of the calculator. It would be advantageous to provide a programmable calculator that automatically responds to application of operating power by loading a program from an external magnetic record member into the calculator memory and by subsequently automatically initiating execution of that program.

SUMMARY OF THE INVENTION

The principal object of this invention is to provide an improved programmable calculator that has more capability and flexibility than conventional programmable calculators, that is smaller, less expensive, and more efficient in evaluating elementary mathematical functions than are conventional computer systems, and that is much easier for the untrained user to operate than either conventional programmable calculators or computer systems.

Another object of this invention is to provide a programmable calculator that employs a magnetic tape cassette unit for storing a program and in which the user

may select an auto start mode of operation for automatically initializing the calculator, loading into calculator memory a program from the magnetic tape cassette unit, and executing that program, all in response to application of operating power by the user.

Another object of this invention is to provide a programmable calculator that automatically adjusts addresses designated in absolute branch statements in accordance with any program editing performed by the user.

Another object of this invention is to provide a programmable calculator that may be coupled to an X-Y plotter and in which the user may employ keys on the calculator to move the plotter pen to a desired point for obtaining a readout from the calculator of the coordinates of that point.

Another object of this invention is to provide a programmable calculator in which the user may employ a single general input/output read-only memory to couple a variety of peripheral input/output units to the calculator.

Another object of this invention is to provide a programmable calculator in which syntax and execution errors are directly communicated to the user, thereby eliminating the need for an error look up table.

Another object of this invention is to provide a programmable calculator employing a user read-write memory having a movable boundary between a program storage section thereof and a data storage section thereof and in which the location of that boundary may be defined by the user.

Another object of this invention is to provide a programmable calculator employing a user read-write memory including a program storage section and a separate data storage section and in which the user is prevented from writing program information into the data storage section and vice versa.

Another object of this invention is to provide a programmable calculator in which the user may assign one of two meanings to every key of an entire block of keys of a keyboard input unit by actuating a single switch.

Another object of this invention is to provide a programmable calculator including an output printer and in which the user may obtain formatted output from the printer without the use of a format statement.

Another object of this invention is to provide a programmable calculator employing reverse polish notation language in which certain combinations of key actuations are associated with a single internal instruction.

Another object of this invention is to provide a programmable calculator in which the user may designate an absolute step location in memory or a label number to be used by the calculator as a memory destination location in association with a transfer statement.

Another object of this invention is to provide a programmable calculator in which the user may select a normal print mode of operation to enable printing during program entry but to suppress printing during manual execution.

Another object of this invention is to provide a programmable calculator in which the user may call for a bit-by-bit comparison between information transferred between the calculator memory and a magnetic record member.

Another object of this invention is to provide a programmable calculator employing a magnetic tape cassette unit and in which old files on a magnetic tape are automatically erased when new files are being marked.

Another object of this invention is to provide a programmable calculator employing a magnetic tape cassette unit in which the current tape position is stored in memory to enable high speed accessing of tape files.

Another object of this invention is to provide a programmable calculator in which programs stored in a memory unit may be listed on an output printer unit in more than one column to facilitate more efficient use of printer paper.

Another object of this invention is to provide a programmable calculator in which the user may write a program involving plug-in read-only memory commands without a plug-in read-only memory present, may later plug a read-only memory into the calculator, and may then obtain a listing of that program including the previously chosen commands associated with that plug-in read-only memory.

Another object of this invention is to provide a programmable calculator employing a dual track magnetic tape cassette unit and in which the specification of all files on a magnetic tape includes a track designation.

Another object of this invention is to provide a programmable calculator employing a thermal dot matrix output printer and in which dots are selectively printed to reduce the power requirements of the printer.

Another object of this invention is to provide a programmable calculator in which the user is given a continuous indication of the amount of available program storage during a program entering mode of operation.

These objects are accomplished in accordance with the preferred embodiment of this invention by employing a keyboard input unit, a magnetic tape cassette reading and recording unit, a gas discharge output display unit, a 16-character thermal printer unit, a peripheral interface adaptor (PIA), a memory unit, and a central processing unit (CPU) to provide an adaptable programmable calculator having manual operating, automatic operating, program entering, magnetic tape reading, magnetic tape recording, and numeric display and alphanumeric print modes.

The keyboard input unit includes a group of numeric data keys for entering data into the calculator, a group of data manipulation keys, a group of function keys for selecting various mathematical functions and operators, a group of memory control keys for controlling the program and data storage areas of the calculator memory, another group of control keys for controlling the operation of the magnetic tape cassette reading and recording unit, and a group of user-definable keys. Many of these groups of keys are useful in both the manual and programmable operating modes. In addition, each of the keys of the user-definable group assumes a secondary meaning during program entry to automatically provide functions that are unnecessary when executing commands manually from the keyboard.

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 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 read-write memory module. Additional read-write memory made available to the user is automatically accommodated by the calculator, and the user is automatically informed of the number of available program storage locations and 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. The routines and subroutines stored in the read-only memory may be expanded to provide routines required to interface various peripheral input/output units to the calculator and to provide some additional functions oriented toward the specific needs of the user. This is accomplished by simply plugging additional read-only memory modules (ROMs) into either or both of two receptacles provided in the rear 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 select codes.

Plug-in ROMs include, for example, a plotter ROM, a typewriter control ROM, a general input/output ROM, a binary-coded-decimal input/output ROM, and an ASCII bus interface ROM. Additional read-only memory modules may be added to a printed circuit board inside the calculator to allow printing characters of foreign languages on both the 16-character thermal printer unit and on an output typewriter that has the desired foreign language character set.

The gas discharge output display unit features 16-character seven segment numeric output with a minus sign, a decimal point, and the capability of displaying commas in selected locations within displayed data.

The 16-character thermal printer unit can print out messages to the user such as error conditions, listings of the user's program and any other message selected by the user that may be formed from the character set available in the calculator. Some alphanumeric data formatting can also be accomplished in the printed output of a single line of information.

The peripheral interface adaptor (PIA) may comprise, for example, a Motorola MC6820 PIA. The PIA operates in conjunction with the central processing unit of the calculator and is capable of dual 8-bit parallel input/output with associated flag, control, handshake, and interrupt hardware that enables the calculator central processing unit to communicate with the above-mentioned internal input/output units that include the keyboard, printer, display, and magnetic tape cassette units. The PIA also has the capability of enabling the calculator to communicate with a plurality of external or peripheral input/output units such as paper tape readers and punches, X-Y plotters, typewriters, and various types of measurement and data gathering instrumentation. This external input/output capability is available to the user through either or both of two input/output connectors located on the rear panel of the calculator that connect the external input/output unit to the PIA through some input/output interface circuitry.

The central processing unit (CPU) may comprise, for example, a Motorola MC6800 8-bit parallel processor with a 1-megahertz clock rate and 65K addressability. This processor includes two 8-bit accumulators, a 16-bit index register, a 16-bit stack pointer, and a 6-bit condition code register.

In the run mode of operation, the calculator is controlled by keycodes received sequentially from the keyboard input unit resulting from key actuations by the user. These keycodes are examined within the calculator immediately upon receipt from the keyboard input unit and are checked for proper syntactical meaning as required by the calculator language. An internal instruction code is generated by the calculator from these keycodes to represent the keyboard instruction desired by the user. This instruction code is then used as a pointer to the address of the routine stored in the read-only memory that is responsible for the execution of the selected instruction.

In the program mode of operation the internal instruction codes generated by the calculator during program entry are stored in the program storage area of the user read-write memory at an address specified by the current value of a user program pointer. These stored instructions constitute a program that may be automatically executed upon request by the user. During program entry, the output printer may be commanded, by means of a keyboad switch, to provide a printed listing of the keyboard commands selected by the user together with the corresponding program address at which the associated internal instruction code is stored. Since several key actuations may result in generation by the calculator of a single internal instruction code and since the calculator executes only these internal instruction codes, a complex stored program can be executed by the calculator very efficiently and in a short period of time.

An autostart mode of operation may be switchably selected by the user to automatically enter into the calculator and execute a program stored on a magnetic tape. This feature allows the use of the calculator by persons unfamiliar with the details of its operation and provides a means for restoring the calculator to working condition in the event a power failure occurs at a time when the calculator is unattended by the user or is attended by an unskilled user.

DESCRIPTION OF THE DRAWINGS

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

FIG. 2 i a rear perspective view of the programmable calculator of FIG. 1.

FIG. 3 is a plan view of the keyboard input unit employed in the programmable calculator of FIG. 1.

FIG. 4 is a simplified block diagram of the hardware associated with the calculator of FIG. 1.

FIG. 5 is a simplified block diagram of the firmware associated with the calculator of FIG. 1.

FIG. 6 is a simplified block diagram showing the ROMs 1–6 and the system control ROM of FIG. 5.

FIG. 7 is a simplified block diagram showing the typical format of I/O ROMs 1 and 2 of FIG. 5.

FIG. 8 is an overall memory map showing system and user read-write (R/W) memory, basic and optional ROM, and plug-in U/O ROM of FIGS. 4 and 5.

FIG. 9 is a detailed memory map of the system read-write memory of FIGS. 4, 5, and 8.

FIG. 10 is a detailed memory map of the user read-write memory of FIGS. 4, 5, and 8.

FIGS. 11A–B are a detailed schematic diagram of the system clock generator and divider and cycle steal blocks of FIG. 4.

FIG. 12 is a timing diagram illustrating waveforms associated with the system clock generator and divider circuitry of FIGS. 4 and 11A-B.

FIG. 13 is a detailed schematic diagram of the central processing unit (CPU) of FIG. 4.

FIG. 14 is a detailed schematic diagram of a portion of the address and chip select block of FIG. 4.

FIG. 15 is a timing diagram illustrating waveforms associated with address and chip select circuitry of FIGS. 4 and 14.

FIG. 16 is a detailed schematic diagram of the basic read-only memory and optional read-only memory of FIG. 4.

FIG. 17 is a timing diagram illustrating waveforms associated with the basic and optional read-only memories of FIGS. 4 and 16.

FIG. 18 is a detailed schematic diagram of the basic read-write memory of FIG. 4.

FIG. 19 is a detailed schematic diagram of the optional read-write memory of FIG. 4.

FIG. 20 is a detailed schematic diagram of the peripheral interface adaptor (PIA) and system peripheral control select unit of FIG. 4 together with some associated buffer and timing circuitry. FIG. 21 is a timing diagram illustrating selected waveforms associated with the system peripheral control select unit of FIGS. 4 and 20.

FIG. 21 is a timing diagram illustrating selected waveforms associated with the system peripheral control select unit of FIGS. 4 and 20.

FIG. 22 is a detailed schematic diagram of a portion of the address and chip select block of FIG. 4 relating to the peripheral interface adaptor and input buffer of FIG. 4.

FIG. 23 is a detailed schematic diagram of the input buffer of FIG. 4.

FIG. 24 is a detailed schematic diagram of a portion of the display circuit of FIG. 4.

FIG. 25 is a detailed schematic diagram of another portion of the display circuit of FIG. 4.

FIGS. 26A–B are detailed schematic diagrams of driver circuitry and paper sense circuitry, respectively, employed in the thermal printer of FIG. 4.

FIG. 27 is a detailed schematic diagram of the keyboard circuitry of FIG. 4.

FIG. 28 is a timing diagram illustrating selected waveforms associated with the keyboard circuitry of FIGS. 4 and 27.

FIG. 29 is a diagram showing the unique keycode associated with each one of the keys of the keyboard of FIG. 3.

FIG. 30 is a block diagram of a portion of the circuitry associated with the magnetic tape cassette unit of FIG. 4.

FIG. 31 is a block diagram of another portion of the circuitry associated with the magnetic tape cassette unit of FIG. 4.

FIG. 32 is a detailed schematic diagram of the gating circuitry of FIG. 30.

FIG. 33 is a detailed schematic diagram of the tach preamplifier and second stage tach amplifier of FIG. 30.

FIG. 34 is a detailed diagram of the frequency detector of FIG. 30.

FIG. 35 is a detailed schematic diagram of the multiplexer of FIG. 30.

FIG. 36 is a detailed schematic diagram of the bilateral current source of FIG. 30.

FIG. 37 is a detailed schematic diagram of the gain selector of FIG. 30.

FIG. 38 is a detailed schematic diagram of the filter, direction sense, and clamp circuits of FIG. 30.

FIG. 39 is a detailed schematic diagram of the voltage gain and current gain circuits of FIG. 30.

FIG. 40 is a detailed schematic diagram of the antimation circuit of FIG. 30.

FIG. 41 is a detailed schematic diagram of the magnetic tape cassette handshake circuitry of FIG. 30 and the track selector circuitry of FIG. 31.

FIG. 42 is a detailed schematic diagram of the hole detector of FIG. 30.

FIG. 43 is a detailed schematic diagram of the write and switch control circuitry and the analog switches of FIG. 31.

FIG. 44 is a detailed schematic diagram of the current source and write protect circuitry of FIG. 31.

FIG. 45 is a detailed schematic diagram of the differential preamplifier of FIG. 31.

FIG. 46 is a detailed schematic diagram of the second stage amplifier/filter of FIG. 31.

FIG. 47 is a detailed schematic diagram of the integrator of FIG. 31.

FIG. 48 is a detailed schematic diagram of the DC tracking circuit of FIG. 31.

FIG. 49 is a detailed schematic diagram of the comparator and frequency doubler of FIG. 31.

FIG. 50 is a detailed schematic diagram of some I/O control and handshake circuitry forming part of the I/O control block of FIG. 4.

FIG. 51 is a detailed schematic diagram of some I/O data output latches forming part of the I/O output block of FIG. 4.

FIG. 52 is a detailed schematic diagram of the optional plug-in I/O ROM of FIG. 4 together with some input buffers associated with the I/O input block of FIG. 4.

FIG. 53 is a detailed schematic diagram of an I/O data input latch and some output buffers forming part of the I/O input and I/O output blocks of FIG. 4.

FIG. 54 is a detailed schematic diagram of the raw power supply employed in the calculator of FIG. 1.

FIG. 55 is a detailed schematic diagram of the +5 volt switching regulator power supply employed in the calculator of FIG. 1.

FIG. 56 is a detailed schematic diagram of the +12 and +15 volt power supplies employed in the calculator of FIG. 1.

FIG. 57 is a detailed schematic diagram of the −5 and −12 volt power supplies employed in the calculator of FIG. 1.

FIG. 58 is a detailed schematic of the −100 volt power supply employed in the calculator of FIG. 1.

FIG. 59 is a detailed schematic diagram of a power on and power off detection circuit employed in the calculator of FIG. 1.

FIG. 60 is a flow chart of a power on routine comprising one of the supervisor routines of FIG. 5.

FIGS. 61A-E are a flow chart of a supervisor control routine comprising one of the supervisor routines of FIG. 5.

FIG. 62 is a flow chart of a keyboard interrupt routine comprising one of the supervisor routines of FIG. 5.

FIG. 63 is a flow chart of a display driver routine comprising one of the supervisor routines of FIG. 5.

FIG. 64 is a flow chart of the error routine of FIG. 5.

FIGS. 65A–L are a flow chart of the alpha routine of FIG. 5.

FIGS. 66A–G are a flow chart of the printer driver routine stored in ROM 3 of FIG. 6.

FIGS. 67A–Z are a flow chart of a portion of the cassette driver routines stored in ROM 3 of FIG. 6.

FIGS. 68A–J are a flow chart of another portion of the cassette driver routines stored in ROM 3 of FIG. 6.

FIGS. 69A–M are a flow chart of the program list routine stored in ROM 4 of FIG. 6.

FIGS. 70A–G are a flow chart of the numeric formatting routine stored in ROM 4 of FIG. 6.

FIGS. 71A–X are a flow chart of the program list routine stored in ROM 4 of FIG. 6.

FIGS. 72A–B are a flow chart of the I/O calling routines stored in ROM 5 of FIG. 6.

FIG. 73 is a flow chart of the binary program routines stored in ROM 5 of FIG. 6.

FIGS. 74A–X are a flow chart of X-Y plotter routines that may be stored in one of the I/O ROMs of FIG. 5.

FIG. 75 is a diagram showing the character set that may be generated when an X-Y plotter is employed with the calculator of FIG. 1.

FIGS. 76A–Z are a flow chart of a portion of some general I/O routines that may be stored in one of the I/O ROMs of FIG. 5.

FIGS. 77A–F are a flow chart of another portion of some general I/O routines that may be stored in one of the I/O ROMs of FIG. 5.

DESCRIPTION OF THE PREFERRED EMBODIMENT

GENERAL DESCRIPTION

Referring to FIG. 1, there is shown a programmable calculator including both a keyboard 10 for entering information into the calculator and for controlling the operation of the calculator and a magnetic tape cassette reading and recording unit 12 for recording information stored within the calculator onto one or more external tape cartridges and for loading information stored on these magnetic tape cartridges back into the calculator. The calculator also includes a seven-segment gas discharge display for displaying data entered into the calculator, the results of computations, and selected alphanumeric messages. The calculator further includes a 16-column alphanumeric thermal printer 16 for printing computation results, program listings, messages generated by the calculator system and the user, and error conditions encountered during use of the calculator. All of these input/output (I/O) units are included within the calculator itself.

As shown in FIG. 2, the calculator includes two input/output (I/O) receptacles 18 for accepting I/O interface connectors 20 each of which includes a read-only memory (ROM) module. These interface connectors serve to couple the calculator to various selected peripheral I/O units such as X-Y plotters, typewriters, photoreaders, paper tape punches, digitizers, BCD-compatible data gathering instruments such as digital voltmeters, frequency synthesizers, and network analyzers, and a universal interface bus for interfacing to most bus-compatible instrumentation.

The overall operation of the calculator hardware may be understood with reference to the block diagram of FIG. 4. The hardware includes a central processing unit (CPU) 100, basic read-write memory 102, optional read-write memory 103, basic read-only memory 104, optional read-only memory 105, and optional plug-in I/O ROM 110. Support hardware for CPU 100 and the above-listed memories includes a clock generator and divider 112, cycle steal circuitry 114, and address and chip select circuitry 116. Also included are a display circuit 118, a thermal printer 120, a keyboard 122, a magnetic tape cassette unit 124, system I/O circuitry 126, a peripheral interface adaptor (PIA) 106, a system peripheral control select unit 128, and input buffer circuitry 130.

CPU 100 may comprise, for example, a Motorola MC6800 microprocessor. The CPU interfaces with basic read-write memory 102, optional read-write memory 103, basic read-only memory 104, optional read-only memory 105, and PIA 106 via an 8-bit bidirectional tri-state instruction-data bus 108. CPU 100 is capable of directly addressing 64K of memory via a 16-bit address bus. However, since the calculator employs only 32K of addressable memory, a 15-bit address bus 110 is provided. A first interrupt port $\overline{IRQ}$ on CPU 100 is used by the keyboard 122, and a second interrupt port $\overline{NMI}$ is employed by the magnetic tape cassette unit 124 via the PIA 106. Two clock phases and instruction-data synchronization on bus 108 are required by CPU 100 for dynamic operation.

The basic ROM 104 and optional ROM 105 comprise the firmware necessary for providing data and instructions to CPU 100. These ROMs are 16,384 bits deep, organized 2048 × 8. The coincidence of two signals is necessary to initiate a ROM access. First, the address bus 110 is decoded to provide a ROM chip select signal, and then a start memory signal synchronized with a phased clock signal Φ_2 is provided to synchronize a group of tri-state buffers inside the ROMs to allow accessed information to be gated onto the instruction-data bus 108. One or two optional plug-in I/O ROMs 110 may be plugged into the calculator to provide additional firmware for driving peripheral I/O units. These plug-in I/O ROMs are accessed by the calculator through a buffered input port that also multiplexes data from peripheral I/O units onto the instruction-data bus 108.

The basic read-write memory 102 and optional read-write memory 103 comprise static NMOS random access memories (RAMs) organized 256 ×4. The basic calculator read-write memory 102 includes a 256 × 8 base page portion employed by the calculator system and a 512 × 8 user portion available for program and data storage. The base page portion or system read-write memory is employed by the calculator firmware as a scratchpad memory. Optional read-write memory 103 may be added to the calculator to increase the size of the user portion of the basic read-write memory 102 by 1536 program steps.

Data is transferred between the CPU 100 and the various I/O units during CPU read and write cycles at designated memory locations. In order to take advantage of the fastest instruction addressing mode of CPU 100, four locations within the base page portion of basic read-write memory 102 are used to transfer data to and from PIA 106. Two other locations on the base page are employed to input data via input buffer 130 from the various internal and peripheral I/O units to the CPU instruction-data bus 108. PIA 106 outputs twelve bits of

data on a bus **132** and four control bits on a bus **134**. The PIA also provides four handshake lines on a bus **136** over which a system handshake between the CPU **100** and the various I/O units is accomplished.

Various signals referenced in the following detailed descriptions of the individual hardware blocks of FIG. **4** may be understood by examination of the corresponding Boolean logic definitions set forth in Table **1** below.

Table 1

LINE		BOOLEAN EQUIVALENT EXPRESSION	
RPIA	=	$\overline{A14} \cdot \overline{A13} \cdot \overline{A12} \cdot \overline{A11}$	
RRAM	=	$\overline{A14} \cdot \overline{A13} \cdot \overline{A12} \cdot A11$	
ADHL	=	$RPIA \cdot \overline{A10} \cdot \overline{A9} \cdot \overline{A8}$	
PIA	=	$ADHL \cdot \overline{A7} \cdot \overline{A6} \cdot \overline{A5} \cdot \overline{A4} \cdot \overline{A3} \cdot \overline{A2}$	
IND	=	$ADHL \cdot \overline{A7} \cdot \overline{A6} \cdot \overline{A5} \cdot \overline{A4} \cdot \overline{A3} \cdot A2 \cdot \overline{A1}$	
CSTNOT	=	$\overline{PIA} + \overline{IND}$	
STM1	=	$\overline{\overline{VMA} \cdot \Phi_2 \cdot \overline{MPWO}}$	
STM2	=	$\overline{\overline{STM1} \cdot \overline{A14} \cdot (A13+A12)}$	
R8	=	$\overline{STM1} \cdot RRAM \cdot \overline{A10} \cdot \overline{A9} \cdot \overline{A8}$	CLEARED BY CYST WRITE + Φ_1
R7	=	$\overline{STM1} \cdot RPIA \cdot A10 \cdot A9 \cdot A8$	"
R6	=	$\overline{STM1} \cdot RPIA \cdot A10 \cdot A9 \cdot \overline{A8}$	"
R5	=	$\overline{STM1} \cdot RPIA \cdot A10 \cdot \overline{A9} \cdot A8$	"
R4	=	$\overline{STM1} \cdot RPIA \cdot A10 \cdot \overline{A9} \cdot \overline{A8}$	"
R3	=	$\overline{STM1} \cdot RPIA \cdot \overline{A10} \cdot A9 \cdot A8$	"
R2	=	$\overline{STM1}\ RPIA\ \overline{A10}\ A9\ \overline{A8}$	"
R1	=	$\overline{STM1} \cdot RPIA \cdot \overline{A10} \cdot \overline{A9} \cdot A8$	"
BPC	=	$\overline{STM1} \cdot RPIA \cdot \overline{A10} \cdot \overline{A9} \cdot \overline{A8}$	"
R0	=	$BPC \cdot CSTNOT$	"
RAM	=	$(\overline{R8} + \overline{STM1} \cdot RPIA) \cdot CSTNOT$	"
IN	=	$IND \cdot \overline{BPC} + \overline{STM2}$	"

SYSTEM CLOCK

Operation of the system clock generator and divider **112** and cycle steal circuitry **114** of FIG. **4** may be understood with reference to the detailed schematic diagram of these circuits shown in FIGS. **11A–B**. The basic clock oscillator shown in FIG. **11A** employs positive feedback and is constructed using linearly biased TTL circuitry. A 4-megahertz crystal filters all but the fundamental frequency to generate a 4-megahertz clock signal that is divided by four to produce the 1-megahertz system clock signal. This 1-megahertz system clock signal is then separated into two non-overlapping phases of equal period. These phased clock signals are designated Φ_1 and Φ_2, and their relative timing is illustrated in the waveform diagram of FIG. **12**. Also illustrated in FIG. **12** and accomplished by way of the circuitry of FIG. **11A** is a cycle steal feature employed during read-write memory access because the memory access time is greater than the 500-nanosecond period of clock signal Φ_2. The 1-megahertz system clock signal is divided as shown in FIG. **11B** to provide signals required for clocking and synchronizing the various internal I/O units.

CENTRAL PROCESSING UNIT

Operation of CPU **100** and its associated circuitry shown in FIG. **4** may be understood with reference to the detailed schematic diagram of FIG. **13**. The 8-bit instruction-data bus **108** is unbuffered and is connected, as shown in FIG. **4**, to read-write memories **102** and **103**, read-only memories **104** and **105**, PIA **106**, and a tri-state input buffer **130**. Fifteen of the sixteen available address lines provided by CPU **100** are buffered by a group of tri-state, non-inverting buffers **138** to form the address bus **110** connected as shown in FIG. **4**.

The two phased clock signals Φ_1 and Φ_2 are received by a pair of clock drivers **140** that in turn provide clock signals having voltage levels and rise and fall times as required by CPU **100**. A start memory signal, $\overline{STM1}$, is generated as shown in FIG. **13** using one of the phased clock signals Φ_2, a signal VMA from CPU **100**, and a system reset/restart (master power on) line $\overline{MPWO}$. For test purposes, a $\overline{HALT}$ line provided by CPU **100** along with signals associated with the tri-state buffers **138** are made available as a line $\overline{TSC}$. The interrupt lines $\overline{NMI}$ and $\overline{IRQ}$ available at CPU **100** are provided with external pull up resistors for improved noise immunity.

A read-write line R/W also available at CPU **100** is buffered and connected to the PIA **106** and basic and optional read-write memories **102** and **103**.

READ-ONLY MEMORY

Operation of the basic read-only memory **104** and optional read-only memory **105** of FIG. **4** may be understood with reference to the detailed schematic diagram of FIG. **16**. ROMs **0–6** comprise basic read-only memory **104** and ROM **7** comprises optional read-only memory **105**. ROMs **0–7** are accessed by decoding the address bus **110** to generate a ROM chip select signal. The chip select signal is buffered by the particular ROM accessed and is used to turn on a power pulse transistor that applies +12 volts to that ROM. When one of the ROMs **0-7** has been chip selected and a start memory signal $\overline{STM1}$ occurs, the information stored in the addressed cell is gated onto the instruction-data bus **108**. ROMs **0–7**, as opposed to other portions of calculator memory, may be selected only when bit **A14** of address bus **110** is high.

The timing relationship of selected signals associated with the various read-only memories employed in the calculator is illustrated in the waveform diagram of FIG. **17**.

READ-WRITE MEMORY

Operation of the basic read-write memory **102** and optional read-write memory **103** of FIG. **4** may be understood with reference to the detailed schematic diagrams of FIGS. **18** and **19**. All read-write memory in the calculator comprises static NMOS 256 × 4 RAM chips. Six of these chips are connected as shown in FIG. **18** to form the basic read-write memory **102**, and twelve of the chips are connected as shown in FIG. **19** to form the optional read-write memory **103**. Basic read-write memory **102** is divided into a 256 × 8 base page or system portion employed as a scratch pad memory and

a 512 × 8 user portion. The twelve 256 × 4 RAM chips connected according to FIG. **19** to form optional read-write memory **103** brings the total user read-write memory to 2048 × 8 words. The calculator employs part of the user read-write memory as system storage registers and as an I/O temporary scratch pad.

The read-write cycle steal timing is shown in FIG. **12**, and the RAM chip select circuitry is shown in FIG. **14**. A base page chip select line $\overline{R0}$ is pulled low during access of read-write memory addresses **6–255**, inclusive. Line $\overline{R0}$ and a cycle steal initiator line $\overline{RAM}$ are inhibited during a PIA access or an input buffer port access by a line $\overline{CSTNOT}$. A line $\overline{R8}$ is decoded separately from the remaining RAM chip select lines $\overline{R1}$–$\overline{R7}$ because address bit A11 is high during access of the corresponding RAM chip but is low during access of all other RAM chips. As seen in FIG. **14**, the status of line $\overline{R8}$ is dependent on a signal RRAM from the ROM chip select circuitry. All RAM chip selects are synchronous with phased clock signal Φ_2 through start memory signal STM1 and all initiate a cycle steal, as shown in FIG. **12**, bgy pulling the line $\overline{RAM}$ low.

The timing relationship of selected signals associated with RAM chip select cycles is illustrated in the waveform diagram of FIG. **15**. During a write cycle, a chip select signal $\overline{CS}$ is removed 500 nanoseconds before the falling edge of phased clock signal Φ_2 to insure data hold time for the RAM chips.

Referring again to FIG. **14**, a line ADHL, synchronous with the address bus **110**, is employed as a chip select line to the PIA **106**. Like line R8, line ADHL is also dependent on line A11 of address bus **110**. Decoding of line ADHL and a line BPC differs only in that BPC is synchronous with phased clock signal Φ_2 while ADHL is dependent only on the state of the address bus **110**, as shown in Table **1** above.

PERIPHERAL INTERFACE ADAPTOR

Operation of peripheral interface adaptor (PIA) **106** of FIG. **4** may be understood with reference to the detailed schematic diagram of FIG. **20**. PIA **106** may comprise, for example, a Motorola MC**6820** peripheral interface adaptor and is employed to output I/O control information and data and to handshake with the various internal I/O units as well as any peripheral I/O units that may be connected to the calculator. Although the two 8-bit peripheral data buses internal to PIA **106** are bidirectional, the only input to CPU **100** during a PIA read cycle is handshake information stored in the control registers of PIA **106**. When the calculator is turned on, PIA **106** is reset by the master power on line $\overline{MPWO}$. The calculator firmware programs the peripheral data buses PA0–PA7 and PB0–PB7 as outputs, and all subsequent PIA read or write cycles to addresses **0-3** of the base page portion of basic read-write memory **102** are made to the A data, A control, B data, and B control registers, respectively, of PIA **106**.

All eight bits of the B data register and the four most significant bits of the A data register form a 12-bit peripheral data output bus **132** comprising lines DO0–DO11. The four least significant bits of the A data register are decoded into fourteen peripheral select lines **142** by the system peripheral control select unit **128**. Because of propagation delays and bit skewing through PIA **106**, these four bits are latched 1 microsecond after each PIA access to prevent false peripheral select line transitions.

The timing relationship of selected signals associated with the hardware of FIG. **20** is shown in the waveform diagram of FIG. **21**. All CPU data transfers to the PIA **106** are referenced to the trailing edge of the phased clock signal Φ_2 that also serves as an enable line for PIA **106**. The chip select lines for the PIA **106** are decoded synchronously with address bus **110** and the VMA line from the CPU **100** to provide chip select set-up time for PIA **106**, as shown in FIGS. **14** and **22**. The handshake functions of PIA **106** are accomplished through the A and B control registers and the four handshake lines CA1, CA2, CB1, and CB2 associated with the PIA. Lines CA1 and CB1 are input handshake lines used by the peripheral I/O and magnetic tape cassette units. Line CB1 activates the output line $\overline{IRQB}$ of the PIA **106** that is connected to the $\overline{NMI}$ interrupt request port of CPU **100**. This arrangement allows the CPU to quickly respond to an end-of-tape handshake associated with magnetic tape cassette unit **124**. Lines CA2 and CB2 are programmed through the calculator firmwave to be output lines. Line CA2 is employed exclusively as a control line in connection with peripheral I/O units, and line CB2 is employed as a system data strobe line. Line CB2 clocks data to printer **16**, controls a comma in display **14**, and clocks data to any peripheral I/O units that may be connected to the calculator.

A line $\overline{CSTNOT}$, encoded as shown in FIG. **22**, is the logical OR of a PIA chip select signal $\overline{PIA}$ and an input buffer chip select signal. Line $\overline{CSTNOT}$ inhibits line $\overline{RAM}$ from cycle stealing the clock signals during a PIA access or an input buffer across and is asynchronous with phased clock signal Φ_2, being derived directly from address bus **110**.

INPUT BUFFER

Operation of the input buffer **130** of FIG. **4** may be understood with reference to the detailed schematic diagram of FIG. **23**. Instructions and data from the optional plug-in I/O ROMs **110** and data from the I/O inputs of system I/O circuitry **126** are multiplexed onto a tri-state 8-bit data bus **144** comprising lines DM0–DM7. Data from the various internal I/0 units of the calculator is multiplexed onto an 8-bit open collector bus **146** comprising lines DI0–DI7. The DM bus **144** and the DI bus **146** are in turn multiplexed onto the cpu instruction-data bus **108**. The DM bus **144** is accessed by either an optional plug-in I/O ROM access or an I/O data read at base page address **5** of basic read-write memory **102**. The decoding for an optional plug-in I/O ROM access select signal $\overline{STM2}$ is illustrated in FIG. **22**. Line $\overline{STM2}$ generates the necessary tri-state control signals for the optional plug-in I/O ROMs **110** and the I/0 inputs within system I/0 circuitry **126**. The DI bus **146** is accessed as a peripheral data read cycle at read-write memory base page address **5**. A signal $\overline{IN}$, encoded as shown in FIG. **22**, enables either the DM bus **144** or the DI bus **146** to become active on the CPU instruction-data bus **108**.

KEYBOARD

Operation of the keyboard **10** shown in FIG. **4** may be understood with reference to the detailed schematic diagram of FIG. **27**. The master power on signal MPWO initializes the keyboard scan circuitry, and the phased clock signal Φ_1 counts up a key scan counter KS and a key detect counter KD. The outputs of the KS counter are decoded into eight lines labelled KS0–KS7 that are connected to a keyboard switch matrix. The

outputs of the KD counter are connected to a key detect multiplexer **148** whose eight input lines KD0–KD7 are received from the keyboard switch matrix. The keyboard circuitry continuously scans the keyboard switch matrix until a switch closure is detected on a $\overline{KD}$ line, as illustrated in the waveform diagram of FIG. **28**. The $\overline{KD}$ line gates the phased clock signal Φ_2 to a one-shot debouncer that in turn triggers a flip-flop to inhibit the CU line and requests an interrupt of the CPU **100** via line $\overline{IRQ}$. When the interrupt has been granted by the calculator firmware, a line $\overline{KCEN}$ enables the state of the KS and KD counters to be read to CPU **100** on lines DI2-DI7 of bus **146**. The state of the KS and KD counters generates an octal keycode in accordance with FIG. **29** to identify the key that has been actuated. Lines DI5, DI6, and DI7 form an octal word having DI5 as its least significant bit and DI7 as its most significant bit. This octal word corresponds to the most significant digit of the octal keycode of the key that has been actuated. Similarly, lines DI2, DI3, and DI4 form an octal word having DI2 as its least significant bit and DI4 as its most significant bit. This octal word corresponds to the least significant digit of the octal keycode of the key that has been actuated. Calculator firmware acknowledges receipt of a key code by removing signal $\overline{KCEN}$. The keyboard scan is restored only if the calculator firmware has accepted the key code and if the one-shot debouncer has indicated that the key switch is open. The calculator firmware periodically updates the status of the two toggle switches located on the far right-hand side of keyboard **10**. A line $\overline{SWEN}$ enables the state of these toggle switches to be read to CPU **100** on lines DI7, DI6, DI5, and DI0 of bus **146**.

DISPLAY

Operation of the display circuit **14** of FIG. **4** may be understood with reference to the detailed schematic diagrams of FIGS. **24** and **25**. A display readout **150** comprises a 16-digit high voltage gas discharge display unit. Each of the characters is formed by selectively energizing seven bar segments, a decimal point, and a comma. By enabling each of the sixteen character positions and simultaneously energizing the appropriate bar segments, a desired character is displayed. A strobing technique is employed to enable only one character position at a time. However, because of the high scan speed involved, all energized character positions appear to glow at the same time.

When the display circuitry is enabled, a line $\overline{DEN}$ allows character position information carried on lines DO8–DO11 to be applied to a decoder **152**. The calculator firmware permutes these inputs in a binary fashion, thereby enabling one of three digit drivers **154** at a time. The output of digit drivers **154**, normally at −45 volts, is pulled to ground when enabled.

Lines DO0–DO7 and CB2, all shown in FIG. **25**, supply segment information. Initially, a bank of segment drive transistors **156** is turned on, thus allowing a bank of segment capacitors **158** to charge to −55 volts with respect to the off character positions. This voltage is insufficient to cause ionization within readout **150**, and so no visible glow appears. When one of the segment drive transistors **156** is turned off, the corresponding segment capacitor **158** immediately applies −200 volts to the associated segment. Since cathode segments for all character positions are connected together, this negative voltage is present on the corresponding segment of each character position. Ionization and resultant glow discharge will only occur between segments at −200 volts and anodes at ground. Although all like cathode segments are at −200 volts, no discharge occurs at those anoes held to −45 volts.

A calculator busy signal comprising minus signs at each character position of display **14** occurs when the calculator is performing extensive calculations or program operations. During this time, line $\overline{DEN}$ is at logical one, and the character scan is applied to decoder **152** by square wave signals of 5, 2.5, 1.25, and 0.625 kilohertz, as shown in FIG. **24**. All character segments except the minus sign are disabled by holding the segment capacitors **158** to −100 volts, and a 10-kilohertz square wave signal simultaneously drives the minus sign segment. Thus, minus signs appear across the entire display. A multivibrator **160** shown in FIG. **25** inhibits the busy signal if the calculator is busy for less than 140 milliseconds.

PRINTER

Operation of the thermal printer **16** of FIG. **4** may be understood with reference to the detailed schematic diagram of FIGS. **26A–B**. Printer **16** comprises a printer chip **162** that includes eighty thermal print elements, a paper advance circuit **164**, and a paper out circuit **166**. Printed characters are formed within a 5 × 7 dot matrix. The eighty thermal print elements on printer chip **162** are arranged in a horizontal line. A line of printed characters is built up by printing all the dots on each of the seven matrix rows in sequence by incrementally advancing the paper past the horizontal line of thermal print elements. The thermal print elements are arranged in four groups of twenty elements each, each of the groups being controlled by one of the select lines S1–S4 shown in FIG. **26A**. A 20-bit shift register within printer chip **162** is loaded via a PDATA line and a CLK line. Each bit of the shift register then controls one of the print elements.

The paper advance circuit **164** comprises a Darlington switch controlled by a $\overline{PEN}$ line. This switch draws current through a printer bobbin that in turn cocks and fires the advance mechanism of the printer.

The paper out circuit **166** shown in FIG. **26B** comprises a light emitting diode **168**, a photo transistor **170**, and some detection circuitry. When paper is present in the printer, light from diode **168** is reflected to photo transistor **170** that produces current flow in resistor **172**. This current is detected by an operational amplifier **174**. Information regarding the presence of a paper supply is available to CPU **100** on a line D11 when a switch enable line SWEN is high.

PERIPHERAL INPUT/OUTPUT

Operation of the system I/O circuitry **126** and an optional I/O interface card **176** of FIG. **4** may be understood with reference to the detailed schematic diagrams of FIGS. **50–53**. System I/O circuitry **126** includes channel select latch circuitry, handshake circuitry, and input bus enable circuitry, all of which circuitry is shown in detail in FIG. **50**.

I/O receptacles **18** shown in FIG. **2** allow connection of two peripheral I/O units to the calculator. These two receptacles are variously referred to in the following detailed description as slot A or channel A and slot B or channel B. As shown in FIG. **51**, I/O channels A and B output data on an I/O data output bus **178** and an I/O data output bus **180**, respectively. These buses each comprise twelve bits of latched data, represented as

lines AD0–AD11and BD0–BD11, respectively. Data is latched by a line CB2 applied through some logic circuitry to a group of data latches **182**. When power to the calculator is turned on, these latches are cleared, and a channel select latch **184** is reset by the master power on line $\overline{\text{MPWO}}$, as shown in FIG. **50**. Referring again to FIG. **50**, the channel select latch **184**, a channel A flag sense flip-flop **186**, and a channel B flag flip-flop **188** are set by a line $\overline{\text{IO7}}$ through the calculator firmware. After selection of the proper channel, either a channel A control flip-flop **190** or a channel B control flip-flop **192** is set by a line $\overline{\text{IO5}}$. The selected peripheral I/O unit responds on either an $\overline{\text{AFLG}}$ or a $\overline{\text{BFLG}}$ line. This response sets the appropriate one of flag sense flip-flops **186** and **188**, clears the previously set one of control flip-flops **190** and **192**, and drives a line CA1 that is interrogated by the calculator firmware.

Referring now to FIG. **52**, there is illustrated a portion of the I/O interface card **176** of FIG. **4**. This circuitry is shown for one of the two peripheral I/O channels and is merely duplicated for the other channel. The 8-bit data bus **144** is controlled by a pair of lines $\overline{\text{ATS1}}$ and $\overline{\text{ATS2}}$ that are generated by an input bus enable decoder **194** of FIG. **50**. Channel select latch **184** of FIG. **50** may be cleared with a line IO6 or by setting a null select code in the latch through line IO7. FIG. **52** also illustrates the plug-in I/O ROM **110** that stores routines and subroutines of instructions necessary for interfacing the calculator to the associated peripheral I/O unit. The plug-in I/O ROM associated with the selected I/O channel is enabled through a decoder **196** when the proper address is placed on selected lines of the 15-bit address bus **110**.

Referring now to FIG. **53**, there is shown another portion of the I/O interface card **176** of FIG. **4**. FIG. **53** includes an input data latch **198** that receives data directly from the attached peripheral I/O unit. Also included is a bank of data output buffers **200** for buffering data received on bus **178** before it is transmitted to the attached peripheral I/O unit. The circuitry of FIG. **53** is shown in connection with I/O channel A. This circuitry is merely duplicated for I/O channel B. Input data from latch **198** and a flag line carrying status information regarding the attached peripheral I/O unit are enabled onto bus **144** through a bus enable circuit **202** that is controlled by lines $\overline{\text{ATS1}}$ and $\overline{\text{ATS2}}$. This is done to prevent multiple data sources on bus **144** at the same time. Data output line $\overline{\text{AD8}}$ of bus **178** performs a special function in the event two peripheral I/O units employing identical plug-in I/O ROMs are connected to the calculator at the same time. Jumpered as shown in FIG. **53**, this bit serves to disable one of the ROMs to prevent simultaneous access to both ROMs.

MAGNETIC TAPE CASSETTE UNIT

Operation of the magnetic tape cassette unit **12** of FIG. **4** may be understood with reference to the detailed block diagrams of FIGS. **30** and **31** and the detailed schematic diagrams of FIGS. **32–49**.

Referring to FIG. **30**, there is shown a detailed block diagram of a motor speed control system employed in the magnetic tape cassette unit **12**. This system is configured as a frequency locked electronic servo loop whose output signal is locked to a reference input signal. The motor speed control system employs calculator system clock generator and divider **112**, described hereinabove, to generate, through a gating circuit **204**, shown in detail in FIG. **32**, two reference frequency signals F_r and F_f. F_r is associated with a signal FST, and F_f is associated with a signal $\overline{\text{FST}}$. F_r is a 62.5-kilohertz signal that provides a magnetic tape search speed of approximately 60 inches/second. Data transfer is accomplished at 10 inches/second using F_f, a 10.4-kilohertz signal. The appropriate reference frequency is gated into the servo loop as F_r under control of CPU **100** via line $\overline{\text{DO9}}$ of the data output bus **132**.

A servo motor **206** is provided for driving a tape capstan. Capstan motion is translated into frequency feedback information F_f by means of a **1000**-line optical tachometer **208** coupled to the motor shaft. The circuitry associated with optical tachometer **208** includes a tach preamplifier and second stage amplifier **210**, shown in detail in FIG. **33**. The tach preamplifier comprises a photo transistor driving a current-to-voltage converter. An amplified analog signal ATC is AC coupled into a voltage comparator to provide a TTL signal F_f. Positive feedback is employed to insure that F_f is a clean waveform.

The reference signal F_r and the feedback signal F_f are applied to a frequency detector **212**, shown in detail in FIG. **34**. Frequency detector **212** dynamically compares F_r and F_f to produce two TTL error correction bits Q_r and Q_f. Frequency coincidence or mismatch is determined on the basis of the rising edges of F_r and F_f. If two rising edges of F_r are detected without an intervening rising edge of F_f, then $F_r < F_f$ and an appropriate error condition is set. Similarly, if multiple rising edges of F_f occur without an intervening rising edge of F_r, then $F_r > F_f$ and another error condition is set. Thus, frequency coincidence is determined for alternating rising edges of F_r and F_f. Frequency detector outputs are created and sustained solely on the basis of frequency data, independent of phase information. A summary of the possible combinations of logic states of Q_r and Q_f together with interpretive information is shown in Table **2** below. In this table logic levels are positive true, a logical zero being $\leqq$ 0.4 volts and a logical one being $\geqq$ 2.4 volts.

Table 2

Q_r	Q_f	INTERPRETATION
0	0	$F_r = F_f$ More information is required to determine frequency mismatch.
1	0	$F_r > F_f$
0	1	$F_r < F_f$
1	1	Don't care condition.

Bidirectional tape motion is employed in magnetic tape cassette unit **12**. Tape direction is specified by a line $\overline{\text{DO10}}$ of data output bus **132**. A signal $\overline{\text{DO10}}$ indicates forward tape motion and its complement indicates reverse tape motion. Line $\overline{\text{DO10}}$ mulitplexes Q_r and Q_f onto selected ones of a number of control lines associated with a multiplexer **214**, shown in detail in FIG. **35**. A line $\overline{\text{FWD}}$ couples Q_r to a source control input line SRC and Q_f to a sink control input line SNK. A line REV gates Q_r and Q_f to the SNK and SRC lines, respectively. Lines SRC and SNK are control inputs to a bilateral current source **216**, shown in detail in FIG. **36**. Bilateral current source **216** responds to the condition of line SRC being a logical one and line SNK being a logical zero by sourcing current on a line OA into a filter **218**. This condition forces a transistor **220** and a transistor **222** of FIG. **36** to cutoff. For the condition wherein lines SRC and SNK are both at logical zero, no corrective action is indicated because frequency coincidence exists. For this condition, line OA forces the

output of bilateral current source **216** into a tri-state mode, and a line TRIST is set to a logical one. The tri-state mode also applies for the condition wherein lines SRC and SNK are both at logical one.

A basic function of filter **218**, shown in the detailed schematic diagram of FIG. **38** along with a direction sense circuit **224** and a clamp circuit **226**, is to remove noise and high frequency components from the error current signals on line OA. It is also important in determining the stability and dynamic performance of the servo loop. The bilateral current source **216** pumps charge on and off the capacitors within filter **218**, thereby creating a dynamic voltage signal that is applied to direction sense circuit **224**. This signal completes a digital-to-analog conversion from frequency detector **212**.

The analog control signal on line OA is amplified and buffered by an operational amplifier comprising a voltage gain circuit **228**, shown in detail in FIG. **39**. Voltage gain circuit **228** drives a class B current gain circuit **230** is drive servo motor **206**. Servo motor **206** may be characterized as a fractional horsepower DC permanent magnet motor. A 1-microfarad capacitor **232** is mounted across the motor terminals to restrict high frequency brush noise to the grounded motor housing.

Operation of the motor speed control system may be divided into an acceleration mode, a servo lock or steady state mode, and a deceleration mode.

During the acceleration mode, the servo loop is closed but is not locked to the reference frequency signal. In order to avoid excessive stress on the tape, servo motor, power supplies, and other components of the magnetic tape cassette unit, the gain of the servo loop is reduced. Loop gain is directly proportional to the value of the current on line OA from bilateral current source **216**. The magnitude of this current is determined by a gain selector **234**, shown in detail in the schematic diagram of FIG. **37**. The state of a D flip-flop **236** switches a transistor **238** from cutoff to saturation. If transistor **238** is saturated, a high gain condition exists, and the current from bilateral current source **216** is at a maximum. On the other hand, if transistor **238** is at cutoff, a low gain condition exists, and the current from bilateral current source **216** is reduced. Before the acceleration mode is entered, a signal from PIA **106** on a $\overline{\text{STOP}}$ line selects the low gain condition.

When the servo loop has locked to the reference frequency signal, the acceleration mode has been completed. At this point it is desirable to increase the bandwidth of the servo loop by increasing its gain. The high gain condition is restored by pulling a line Q_f high, signifying that $F_f > F_r$. The high gain condition remains until the deceleration mode is initiated.

During the deceleration mode the low gain condition is again selected by pulling the $\overline{\text{STOP}}$ line low. For controlled deceleration, a capacitor **240** in FIG. **38** is sensed to determine whether it is charged positively or negatively and is then linearly discharged or charged via bilateral current source **216** toward ground. Direction sense circuit **226** provides a 2-bit low power TTL-compatible output $\overline{\text{FWD}}_A$ and REV_A. If the voltage on capacitor **240** is greater than +0.3 volts, $\overline{\text{FWD}}_A = 0$ and $\text{REV}_A = 0$. If the capacitor voltage is less than −0.3 volts, $\overline{\text{FWD}}_A = 1$ and $\text{REV}_A = 1$. If the capacitor voltage lies within these limits, then $\overline{\text{FWD}}_A = 1$ and $\text{REV}_A = 0$. Pulling the line $\overline{\text{STOP}}$ to logical zero forces both Q_r and Q_f to logical zero, as shown in FIG. **34**. This condition also gates the $\overline{\text{FWD}}_A$ and REV_A lines into mulitplexer **214** to control the SRC and SNK lines, as shown in FIG. **35**. Thus, the bilateral current source **216** is enabled to either charge or discharge capacitor **240** of FIG. **38** toward ground. When the capacitor voltage is reduced to lie within the range of +0.3 volts to −0.3 volts so that $\overline{\text{FWD}}_A = 1$ and $\text{REV}_A = 0$ the TRIST line is pulled high, and capacitor **240** is clamped to ground until the STOP line is pulled low to enter the acceleration mode. Regardless of the load presented to motor **206** by a particular tape cartridge at any time during the steady state mode, the stopping distance remains nearly constant. This results from the fact that a heavy load requires a higher voltage at the motor for servo lock. In addition, a heavier load means that the motor will stall at a higher voltage level. Hence, tape movement will halt in an approximately constant distance independent of load and voltage levels.

An antimotion circuit **242** prevents servo motor **206** from moving during the power turn-on and turn-off cycles of the calculator. This circuit is shown in the detailed schematic diagram of FIG. **40**. Motion is inhibited as long as line $\overline{\text{MPWO}}$ is held to logical zero.

General tape position information exists as a punched hole configuration in the magnetic tape. These holes are detected by means of a hole detection circuit **244**, shown in detail in the schematic diagram of FIG. **42**, basically comprising an incandescent light source **246** and a photo transistor **248**. Photo transistor **248** drives a passive low pass filter, the voltage at which is related to a fixed threshold at the differential inputs of an operational amplifier **250**. The operational amplifier **250** is configured as a comparator with positive feedback. Some logic circuitry following operational amplifier **250** generates a TTL-compatible logic signal HOL. The line HOL, a cartridge status line CIN, and a write prevent line WPR are inverted and presented on lines D12, D10, and D11, respectively, of data input bus **146**. These signals are issued in response to a signal $\overline{\text{CSEN}}$, by cassette handshake circuitry **252**, shown in detail in the schematic diagram of FIG. **41**. If a tape cartridge is ejected from magnetic tape cassette unit **12**, line CIN is pulled low. If a hole is detected in the magnetic tape, line HOL is pulled high. In the event either of these conditions exists, a signal is issued on an interrupt line CB1.

Referring now to FIG. **31**, there is shown a detailed block diagram of read-write circuitry associated with the magnetic tape cassette unit **12**. A dual track, dual center tapped magnetic head **254** is employed for information transfer. A current source and write protect circuit **256**, shown in detail in the schematic diagram of FIG. **44**, drives magnetic head **254**. A transistor **258** serves as the current source. Writing occurs when one of the head lines is switched from an open condition to a low condition. Current, as set by current source **256**, is then allowed to flow from the center tap to the selected head line, thus setting up a flux field in the head gap. At some later point in time, the second head line associated with the selected track is switched from an open condition to a low condition. At the same time, the previously switched head line returns to an open condition. Current now flows from the center tap to the head line that is being held low. The flux field at the head gap is reversed, and the magnetic tape is saturated in the opposite polarity. A flux reversal is said to have been written on the tape. Information is written on the tape by alternately producing low and open conditions on the head lines associated with a selected track.

A group of analog switches **260** performs the function of switching the magnetic head lines. These switches and associated logic circuitry are shown in detail in the schematic diagram of FIG. **43**. A binary-to-decimal decoder **262** with high voltage open collector outputs is arranged to decode an incoming data stream on line DO11 of data output bus **132**. Decoder **262** also decodes a track select line TKB and a write line WRT. Current source **256** is turned off when power turn-on or turn-off occurs in the calculator or when line WRT is pulled low, as shown in FIG. **44**.

A read operation uses the full track width of the magnetic head **254** for maximum signal strength. The center tap of the head is not used. Analog switches **260** gate the magnetic head signals of the selected track to the inverting and non-inverting inputs of a differential preamplifier **264**, shown in detail in the schematic diagram of FIG. **45**. As in write operations, the binary-to-decimal decoder **262** of FIG. **43** controls the analog switching. Preamplifier **264** is configured differentially to maximize common mode rejection. The gain of this preamplifier is adjusted to compensate for differences in individual head characteristics. Flux reversals previously written on the moving tape produce current reversals in the magnetic head. These current reversals appear as positive and negative voltage pulses on an output line AHD of differential preamplifier **264**. The nominal signal level on line AHD is 136 millivolts peak to peak.

A second stage amplifier/filter **266** applies an additional voltage gain factor of twenty to read signal. Circuit details of second stage amplifier/filter **266** are shown in the schematic diagram of FIG. **46**. A low impedance input is provided for better noise immunity. Amplifier/filter **266** is configured to provide equal gain for the signal on line AHD and for signals appearing at an inverting input to improve the common mode rejection. A single pole filter at approximately 40 kilohertz provides high frequency attenuation. The signal on an output line AHD2 of amplifier/filter **266** is nominally 2.6 volts peak to peak.

The output of second stage amplifier/filter **266** is connected to an integrator **268**, shown in detail in the schematic diagram of FIG. **47**. The inverting input of an operational amplifier **270** is at virtual ground. Thus, the voltage on an integrating capacitor **272** relative to ground is dynamically adjusted to be proportional to the area of the signal on input line AHD2. A resistor **274** provides a feedback path for DC biasing purposes. A capacitor **276** blocks the DC offset of previous amplifier stages and allows only unity DC gain for integrator **268**. This arrangement serves to attenuate low frequency noise. Integraror **268** also attenuates high frequency noise because of the fact that integrators inherently respond to signal area. As the magnetic tape accelerates or decelerates, the level of the signal at the magnetic head, as well as its frequency, increases or decreases. Hence, the area of the voltage pulses remains relatively constant, and the integrator can track speed variations with small peak to peak variations from the nominal level of the output signal on a line INT.

Because the integrator is sensitive to input signal area, changes in area produce dynamic variations in the DC component of the signal on line INT. This condition is compounded by the loss introduced by biasing resistor **274**. To alleviate this problem, the signal on line INT is sampled both above and below ground level and a CD tracking circuit **278**, shown in detail in the schematic diagram of FIG. **48**. Germanium diodes are employed because of their low voltage turn-on characteristics and so that the sampling signal is in phase with the signal on input line INT. A pair of capacitors **280** retains the sampled voltage levels. Two resistors **282** are employed as summing inputs for an operational amplifier **284** configured as a voltage follower. These resistors are also required for charging and discharging capacitors **280** to enable sampling of subsequent voltage peaks.

A comparator **286**, shown in detail in the schematic diagram of FIG. **49**, receives line INT from integrator **268** and a line DCL from DC tracking circuit **278**. Since the signal on line DCL should track the DC component of the signal on line INT, comparator **286** functions basically as a relative zero crossing switch with a TTL-compatible output. To create some effective hysteresis for noise immunity, positive feedback is provided through inverters to each of the inputs of comparator **286**. Voltage division is employed to determine the amount of voltage hysteresis.

A frequency doubler **288** in FIG. **31** is also included in the circuitry of FIG. **49**. A resistor **290** and a capacitor **292** provide a slight time dealy at one of the inputs to an exclusive OR gate **294**. The other input is not delayed. Thus, each rising or falling edge of the signal on a comparator output line results in a pulse at the output of exclusive OR gate **294**. The rising edge of each of these pulses is coincident with an edge of the output signal of the comparator. The rising edge becomes a falling edge at line CA1 that is fed to PIA **106**.

POWER SUPPLIES

Operation of the power supplies that power the calculator hardware may be understood with reference to the detailed schematic diagrams of FIGS. **54–59**. When a power switch **22** of FIGS. **1** and **54** is placed in the "on" position, AC line voltage is supplied to the primary of a transformer **298** through a pair of switches on the primary side of transformer **298**. These switches are arranged to accept any one of four AC line voltages. These may be 100, 120, 220 or 240 volts. Secondary filtering is employed to reduce interference on the AC line. A full wave bridge rectifier is employed to provide both positive and negative raw voltages of approximately 25 volts on a pair of lines +RAW and −RAW.

Referring now to FIG. **55**, there is shown a detailed schematic diagram of a switching regulator for deriving +5 volts from line −RAW.

Referring now to FIG. **56**, there is shown a detailed schematic diagram of circuitry for supplying regulated voltages of +12 and +15 volts from line +RAW. The +15 volt supply employs a series pass regulator **300** that includes a current limit circuit. A resistor **302** may be adjusted to set the output voltage between 14.7 volts and 15.9 volts.

Referring now to FIG. **57**, there is shown a detailed schematic diagram for supplying −5 and −12 volts from line −RAW. This circuitry employs series pass regulators.

Referring now to FIG. **58**, there is shown a detailed schematic diagram of a −100 volt power supply and an associated pulse shaping circuit **304**. Pulse shaping circuit **304** receives a 20-kilohertz square wave from system clock generator and divider **112** and produces a 20-kilohertz train of narrow pulses for use by the −100 volt power supply. The −100 volt supply is controlled by a timer **306**. The negative pulses from pulse shaping circuit **304** are applied at pin 2 of timer **306**. These

pulses trigger the timer, resulting in charging a capacitor C1 through a resistor R1. At the same time, pin 3 is pulled high and turns on a transistor Q1. Timer **306** remains on until the voltage at pins 7 and 6 reaches the internal level or the feedback voltage on pin 5. At that point in time, the output at pin 3 is turned off, and pins 6 and 7 are clamped to ground, thus discharging the capacitor C1. These conditions remain until the next negative pulse appears at pin 2.

When transistor Q1 is turned on, 15 volts is applied across coil L1. Because of a 4:1 turns ratio on coil L1, the voltage applied to a capacitor C3 is 60 volts.

Timer **306** switches before the core of coil L1 saturates, thus generating a high flyback voltage across coil L1. As the voltage at the collector of transistor Q1 increases, the voltage at capacitor C3 decreases until a diode D3 becomes forward biased. This clamps the ringing voltage and dumps the energy into a capacitor C4. A diode D1 is employed to clamp the output voltage of transistor Q1 so that the negative ringing does not destroy the transistor. When the output voltage appearing across a capacitor C4 reaches −100 volts, a diode D4 begins to conduct. This begins to pull pin 5 of timer **306** lower than the internal reference. As the voltage on pin 5 decreases, the on time of timer **306** also decreases. This reduces the energy stored in the coil L1 and results in stabilizing the output voltage near −100 volts. A resistor R3 is employed to limit the charging current to a feedback capacitor C2.

Referring now to FIG. **59**, there is shown a power on detection circuit employed to sense whether operating power has been applied to the calculator by interrogating the line +RAW. A pulse is generated on the line MPWO by an RC time constant after the +5 volt power supply reaches its operating voltage. When operating power to the calculator is turned off, the line +RAW is the first of the power supply lines to die. This condition is detected, and another pulse on line $\overline{\text{MPWO}}$ is generated. The line $\overline{\text{MPWO}}$ is used by various portions of the calculators hardware for initialization purposes.

CALCULATOR FIRMWARE

Operation of the calculator firmware may be understood with reference to FIGS. **5–10**, the calculator firmware listing of routines and subroutines stored within the calculator read-only memory, and the flow charts of these routines and subroutines illustrated in FIGS. **60–77F**.

Referring to FIG. **5**, there is shown a simplified block diagram of the calculator firmware. Included are ROMs **0-6** comprising basic read-only memory **104** of FIG. **4**, ROM **7** comprising the optional read-only memory **105** of FIG. **4**, and the two I/O ROMs comprising the optional plug-in I/O ROMs **110** of FIG. **4**.

ROM **0**, also referred to as the system control ROM, contains a group of supervisor routines, a linkage table, and syntax tables, as shown in FIG. **6**. ROMs **1–6** contain various ROM execution routines also shown in FIG. **6**. ROM **7** is available for storing routines and subroutines of additional instructions to expand the capability of the calculator. Optional plug-in I/O ROMs **1** and**2** of FIG. **5** contain routines and subroutines of instructions for interfacing various peripheral I/O units to the calculator.

A detailed listing of the routines and subroutines of instructions stored in ROMs **0** and **3–6**, together with a listing of the routines and subroutines that may be stored in two typical plug-in I/O ROMs, is provided hereinafter. In addition, detailed flow charts of these routines and subroutines are variously shown in FIGS. **60–77F**. No listing of the floating point math routines stored in ROM **1** or the cordic math alogorithm routines stored in ROM **2** is provided since these routines are well known and may be readily implemented by those persons skilled in the art of computer logic.

Referring now to FIG. **7**, there is shown a memory allocation diagram of the optional plug-in I/O ROMs illustrating their format by hexadecimal addresses.

Referring now to FIG. **8**, there is shown a map illustrating the allocation, by hexadecimal addresses of the entire calculator memory.

Referring now to FIG. **9**, there is shown a detailed memory map of the base page or system read-write portion of the basic read-write memory **102** of FIG. **4**. This base page is employed for storing several words of information used by the calculator firmware. It includes a status storage area used by the calculator firmware, a subroutine vector stack for storing return addresses associated with user subroutines, a temporary read-write or scratch pad memory, a buffer register used by the calculator and printer, a user operational stack including X, Y, Z, and T registers, a keycode buffer register, five user data storage registers A–E, pointers associated with the plug-in I/O ROMs, and various other pointers used by the calculator firmware.

Referring now to FIG. **10**, there is shown a detailed memory map of the user portion of basic read-write memory **104** of FIG. **4**. This map illustrates a pointer EOPM separating a program storage portion of user read-write memory from a data storage portion. This boundary pointer EOPM may be moved within the user read-write memory at the discretion of the user by execution of an instruction from the keyboard or under program control, as described in detail hereinafter. This arrangement results in more efficient use of the calculator read-write memory by allowing the user to quickly and easily adjust the respective sizes of the program and data storage portions thereof to suit his present needs.

DETAILED LISTING OF ROUTINES AND SUBROUTINES OF INSTRUCTIONS

A complete assembly language listing of all of the routines and subroutines of instructions employed by the calculator is given below. The listing includes all routines and subroutines stored in ROMs **0** and **3–6** of the basic read-only memory **104** of FIG. **4** as well as all the routines and subroutines stored in a general purpose plug-in I/O ROM and a plotter plug-in I/O ROM. Each page within the listing is numbered at the upper left-hand corner, and its page 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 numbering arrangement facilitates reference to different portions of the listing. Descriptive headings are also provided throughout the listing to identify routines, subroutines, groups of constants, linkage tables, optional plug-in I/O ROM routines and subroutines, etc. Each instructiom of each routine or subroutine and each constant stored in the ROMs of the basic read-only memory or the optional plug-in I/O ROMs is represented in hexadecimal form by two, four or six characters in the third and fourth columns from the left-hand side of the page. Each of these instructions may be understood in detail by referring to published literature

associated with the Motorola MC6800 microprocessor. The hexadecimal address of the ROM location in which each such instruction or constant is stored is given in the second column from the left-hand side of the page. By comparing the hexadecimal address given in the listing for a particular instruction to the addresses associated with the various ROMs shown in FIG. 6, it can be seen in which of the ROMs 0–6 that instruction resides.

Mnemonic labels serving as symbolic addresses or names are given in the fifth column from the left-hand side of the page. A mnemonic code for each of the instructions is given in the sixth column from the left-hand side of the page. In the case of those instructions involving a reference to one of the two accumulators within CPU 100, either the letter A or the letter B appears in the seventh column from the left-hand side of the page to designate the appropriate accumulator. Operands that may be either labels or literals associated with each of the instructions are located in the eighth column from the left-hand side of the page. Explanatory comments are given in the remaining portion of each page.

In addition, symbol tables are included following various sections of the listing to relate various mnemonic labels to their hexadecimal values.

15

```
00215                          OPT     LIST,MEM
00218                          NAM     POSUP
00219                          OPT     LIST,MEM,NG
00220 7FFE                     ORG     $7FFE
00221                *
00222                *
00223                *****POWER-ON AND SUPERVISOR*****
00224                *
00225                *
00226                *THIS ROUTINE IS THE POWER-ON SEQUENCE AND
00227                *THE SUPERVISOR FOR CJ.
00228                *
00229 7FFE 7800                FDB     $7800     POWER FAIL/RESTART VECTOR
00230 7800                     ORG     $7800     POWER ON BEGINS HERE
00231 7800 8E 0051             LDS     #ISTACK    INITIALIZE STACK POINTER
00232 7803 B6 1002             LDA A   $1002     GET POSSIBLE TEST ROM
00233 7806 81 39               CMP A   #$39      IS IT THERE ?
00234 7809 26 03               BNE     NOTEST    BRANCH IF NOT
00235 780A 7E 1025             JMP     $1025     ELSE LET IT HAVE CONTROL
00236 780D 86 FF    NOTEST     LDA A   #$FF      ACCA = ALL ONES
00237 780F 97 00               STA A   ADATA     AIO IS NOW ALL OUTPUTS
00238 7811 97 02               STA A   BDATA     BIO IS NOW ALL OUTPUTS
00239 7813 86 3C               LDA A   #$3C      PREPARE TO SELECT DATA REGISTERS
00240 7815 97 01               STA A   ACTL      SELECT ADATA
00241 7817 97 03               STA A   BCTL      SELECT BDATA
00242 7819 CE 0000             LDX     #0        CLEAR THE INDEX
00243 781C DF 20    MORE       STX     TP7       SAVE ON BASE PAGE
00244 781E 7C 0020             INC     TP7       INC THE PAGE ADRS
00245 7821 DE 20               LDX     TP7       RESTORE NEW INDEX
00246 7823 86 49               LDA A   #@111     GET MEMORY CHECK WORD
00247 7825 A7 00               STA A   X         STORE IN "SUPPOSED" MEMORY
00248 7827 A1 00               CMP A   X         WAS MEMORY OUT THERE ?
00249 7829 27 F1               BEQ     MORE      BRANCH IF YES
00250 782B D6 20               LDA B   TP7       ELSE ACCB=PAGE ADRS OF EMPTY PAGE
00251 782D 6F 00    ERASE      CLR     X         ERASE ALL OF R/W
00252 782F 09                  DEX
00253 7830 8C 0005             CPX     #5        FINISHED YET ?
00254 7833 26 F8               BNE     ERASE
00255 7835 D7 0A               STA B   EOM       SAVE END OF MEMORY ADRS
00256 7837 5A                  DEC B             BACK UP ONE PAGE
00257 7838 D7 0B               STA B   EOPM      AND END OF PRGM MEMORY
00258 783A C6 D8               LDA B   #$D8      GET ADRS ON PAGE
00259 783C D7 0C               STA B   EOPM+1    INITIALIZE ALPHA REG.
00260 783E DE 0B               LDX     EOPM      GET CURRENT END POINTER
00261 7840 DF C0               STX     IO1       INITIALIZE I/O 1
00262 7842 DF D0               STX     IO2       INITIALIZE I/O 2
00263 7844 7C 00C8             INC     UPP       GENERATE PAGE ADRS
00264 7847 86 06               LDA A   #6        LOAD TRACK A SEL. CODE
00265 7849 BD 7A26             JSR     RDTGL1+3    SET CASSETTE TRACK A
00266 784C C6 10               LDA B   #$10      GET "FIXED MODE" BIT
00267 784E D7 07               STA B   TGL       SET TOGGLE WORD
00268 7850 C6 02               LDA B   #2        LOAD DISPLAY POUND NUMBER
00269 7852 D7 0E               STA B   RND       SAVE IN ROUND WORD
00270 7854 96 20    IOPOLL     LDA A   TP7       GET MSB'S OF POLLING ADDRESS
00271 7856 8B 08               ADD A   #8        BUMP BY ONE ROM
```

```
00272 785A 97 20          STA A  TP7      RESTORE IT
00273 785A A1 48          CMP A  #$48     FINISHED YET ?
00274 785C 27 0C          BEQ    NOI02    BRANCH IF YES
00275 785E DF 20          LDX    TP7      ELSE LOAD CURRENT ROM ADRS
00276 7860 E6 02          LDA B  2,X      GET THIRD ROM WORD
00277 7862 C1 7E          CMP B  #$7E     IS IT A JMP STATEMENT ?
00278 7864 26 EE          BNE    IOPOLL   CONTINUE POLLING IF NOT
00279 7866 AD 02          JSR    2,X      ELSE OFFER INITIALIZATION
00280 7868 20 EA          BRA    IOPOLL   CONTINUE POLLING
00281 786A DE 0B    NOI02 LDX    EOPM     GET MODIFIED EOPM
00282 786C DF 54          STX    SPGM     SAVE IN SPECIAL PRGM PNTR
00283 786E D6 0C          LDA B  EOPM+1   GET END PNTR
00284 7870 96 0B          LDA A  EOPM
00285 7872 C0 50          SUB B  #80      INITIALIZE 10 REGISTERS
00286 7874 D7 0C          STA B  EOPM+1   RESTORE PNTR
00287 7876 82 00          SBC A  #0
00288 7878 97 0B          STA A  EOPM
00289 787A 96 06          LDA A  ERROR    POWER-ON I/O ERROR ?
00290 787C 26 38          BNE    IOERR    BRANCH IF YES
00291 787E 86 0C          LDA A  #$C      LOAD TOGGLE S.C.
00292 7880 97 00          STA A  ADATA    SEND TOGGLE SELECT CODE
00293 7882 96 04          LDA A  INPUT    GET TOGGLE CONDITIONS
00294 7884 7F 0000        CLR    ADATA    DISABLE THE I/O TRANSFER
00295 7887 44             LSR A           SHIFT AUTO-START BIT TO CARRY
00296 7888 25 3F          BCS    TOP7     BRANCH IF NO AUTO START
00297 788A BD 5D75        JSR    BLANK    BLANK PRINT BUFFER
00298 788D CE 005A        LDX    #BUFF    ELSE GET BUFFER ADRS
00299 7890 DF 16          STX    TP2      SAVE FOR "LOAD MESSAGE"
00300 7892 CE 788E        LDX    #MSG1    GET "AUTO START" MESSAGE ADRS
00301 7895 BD 57B0        JSR    LDMSG    DO MESSAGE TO BUFFER
00302 7898 BD 6046        JSR    PRTDRV+25   PRINT AUTO START
00303 789B FE 7D4C        LDX    $7D4C    GET THE "LOAD & GO" ADDRESS
00304 789E 4F             CLR A           INITIALIZE ACCA
00305 789F AD 00          JSR    X        CALL THE CASSETTE
00306 78A1 96 06          LDA A  ERROR    ERROR GENERATED ?
00307 78A3 27 14          BEQ    AUTOOK   BRANCH IF NOT
00308 78A5 BD 5D75        JSR    BLANK    BLANK THE BUFFER
00309 78A8 CE 005A        LDX    #BUFF    GET BUFFER ADRS
00310 78AB DF 16          STX    TP2      SAVE FOR "LOAD MESSAGE"
00311 78AD CE 54D8        LDX    #EMSG+$D8   GET "FAILED" MESSAGE ADRS
00312 78B0 BD 57B0        JSR    LDMSG    DO MESSAGE TO BUFFER
00313 78B3 BD 6046        JSR    PRTDRV+25   PRINT FAILED
00314 78B6 7E 7840  IOERR JMP    ERROR7   OUTPUT THE SPECIFIC CASSETTE ERROR
00315 78B9 C6 80    AUTOOK LDA B  #$80     ACCB="RUN" INSTRUCTION
00316 78BB 7E 796B        JMP    EXEC7    RUN !!!!!
00317 78BE 41       MSG1  FCC    /AUTO START/
00318 78C8 80             FCB    $80
00320                     *
00321                     *
00322                     ****THE SUPERVISOR FOLLOWS. IT IS THE
00323                     ****CONTROL SECTION FOR THE ENTIRE MACHINE
00324                     *
00325                     *
00326 78C9 0E       TOP7  CLI              ENABLE INTERRUPTS
00327 78CA 4F             CLR A
00328 78CB 97 CC          STA A  ALPHA     RESET ALPHA MODE
00329 78CD 97 C6          STA A  SOL7      CLEAR SINGLE OP LOCATION
00330 78CF 97 11          STA A  W1        CLEAR BCD ACC.
00331 78D1 97 10          STA A  W2        CLEAR BCD ACC.
00332 78D3 97 13          STA A  DCNTR     CLEAR PLABR DIGIT COUNTER
00333 78D5 86 80          LDA A  #@200     BIT 7=1
00334 78D7 97 C7          STA A  SOL7+1    INIT SECOND BYTE
00335 78D9 CE 00C6        LDX    #SOL7     GET SINGLE OP LOC ADDRESS
00336 78DC DF CA          STX    UIP       PLACE IT IN THE INST CNTR
00337 78DE CE 7E00        LDX    #MT '     GET MAIN TABLE ADDRESS
00338 78E1 DF D3          STX    IT7       SAVE IN INDEX TEMP
00339 78E3 BD 7A64  NK7   JSR    SDISP1    GO TO DISPLAY DRIVER
00340                     *
00341                     * IF A KEY WAS AVAILABLE, ITS CODE IS RETURNED
00342                     *IN THE A ACCUMULATOR
00343                     *
00344 78E6 DE D3          LDX    IT7       GET CURRENT SEARCH TABLE ADRS
00345 78E8 81 0C          CMP A  #@14      BACK STEP KEY ?
00346 78EA 26 0C          BNE    TSRCH7    BRANCH IF NOT AND SEARCH
00347 78EC D6 07          LDA B  TGL       ELSE CHECK MODE
```

```
00348 78FE 2B 03          BMI     TSRCH6   BRANCH IF PRGM MODE
00349 7800 7E 7978        JMP     BWM      ELSE DO "OOPS"
00350 7803 8C 7E00 TSRCH6 CPX     #MT      ANY PREFIX KEYS HIT ?
00351 7806 26 D1          BNE     TOP7     BRANCH IF YES AND FORGET THEM
00352 7808 E6 00   TSRCH7 LDA B   X        PUT TBL ENTRY INTO ACCB
00353 780A C1 3D          CMP B   #TERMN7   TBL TERMINATOR?
00354 780C 27 0E          BEQ     TEND7    GO TO "TERMINATOR FOUND"
00355 780E C4 3F          AND B   #@77     MASK OFF 2 MSB'S
00356 7900 11             CBA              MATCH FOUND?
00357 7901 27 30          BEQ     MEND7    YES: GO TO "MATCH FOUND"
00358 7903 6D 00          TST     X        NO: CONTINUE
00359 7905 2A 01          BPL     TWOB7    1 BYTE FOLLOWS
00360 7907 08             INX              INC INDEX
00361 7908 08       TWOB7 INX
00362 7909 08             INX
00363 790A 20 EC          BRA     TSRCH7   CONTINUE SEARCHING
00364                     *
00365                     * "TBL TERMINATOR FOUND" IS HANDLED HERE
00366                     *
00367 790C 08       TEND7 INX
00368 790D E6 00          LDA B   X        ERROR OR "OR" CONDITION ?
00369 790F 2B 18          BMI     ERR7     ERROR! GO TO ERROR HANDLER
00370 7911 DF 14          STX     TP1      SAVE TABLE ADDRESS
00371 7913 DE D3          LDX     IT7      GET TABLE STARTING ADRS
00372 7915 8C 7E00        CPX     #MT      MAIN TABLE?
00373 7918 26 0A          BNE     FIX      NO, CONTINUE
00374 791A D6 07          LDA B   TGL      ELSE CHECK MODE
00375 791C 2A 06          BPL     FIX      CONTINUE IF RUN MODE
00376 791E DE 14          LDX     TP1      ELSE RE-LOAD INDEX
00377 7920 08             INX              INCREMENT OVER THE RUN MODE
00378 7921 08             INX              "OR" CONDITION
00379 7922 20 02          BRA     FIX+2    GET THE NEW TBL ADRS
00380 7924 DE 14    FIX   LDX     TP1      RESTORE TBL ADRS
00381 7926 EE 00          LDX     X        GET THE NEW TBL ADRS
00382 7928 DF D3          STX     IT7      SAVE IN INDEX TEMP
00383 792A 20 CC          BRA     TSRCH7   CONTINUE SEARCHING
00384 792C C4 7F    ERR7  AND B   #$7F     MASK OFF THE "N" BIT
00385 792E D7 06          STA B   ERROR    SAVE IN ERROR WORD
00386 7930 7E 7840 EJSR7  JMP     ERROR7   GO TO ERROR OUTPUT ROUTINE
00387                     *
00388                     *
00389                     *
00390 7933 A6 00    MEND7 LDA A   X
00391 7935 08             INX
00392 7936 4D             TST A            TEST TBL ENTRY
00393 7937 2B 1E          BMI     PART17   BIT 7=1
00394 7939 48             ASL A            GET BIT 6 (7=0)
00395 793A 2B 0C          BMI     IMEX7    GO TO IMMEDIATE EXECUTE
00396 793C 86 3F    L7    LDA A   #@77     GET ADRS HI
00397 793E E6 00          LDA B   X        GET TBL ENTRY
00398 7940 58             ASL B            DOUBLE IT
00399 7941 49             ROL A            SHIFT C INTO MSW
00400 7942 97 D3          STA A   IT7      RESTORE MSW
00401 7944 D7 D4          STA B   IT7+1    RESTORE LSW
00402 7946 20 98          BRA     NK7      GO GET NEW KEY
00403 7948 E6 00    IMEX7 LDA B   X        GET THE INST CODE
00404 794A D7 C6          STA B   SOL7     STORE IN SINGLE OP LOC
00405 794C D6 C6    CONT  LDA B   SOL7     ACCB=THE INSTRUCTION
00406 794E 96 07          LDA A   TGL      PROGRAM MODE?
00407 7950 2A 16          BPL     EXEC6    NO: GO EXECUTE
00408 7952 BD 4ECB        JSR     STCOD7   ELSE STORE INSTRUCTION
00409 7955 20 24          BRA     BWM      WRAP IT UP !
00410 7957 48       PART17 ASL A           BIT 6=?
00411 7958 2B 05          BMI     PARCD7   YES: PARTIAL CODE OPERATION
00412 795A EE 00          LDX     X        FOR PARTIAL EXECUTION
00413 795C 4F             CLR A
00414 795D 6E 00          JMP     X        DO "PARTIAL EXECUTION"
00415 795F D6 C6    PARCD7 LDA B  SOL7     GET SINGLE OP WORD
00416 7961 EB 00          ADD B   X        ADD THE PARTIAL CODE
00417 7963 D7 C6          STA B   SOL7     RESTORE IT
00418 7965 08             INX
00419 7966 20 D4          BRA     L7       GO FORM NEW TBL ADRS
00420 7968 BD 5C31 EXEC6  JSR     AUDIT    AUDIT TRAIL ?
00421 796B 86 BE    EXEC7 LDA A   #@276     GET PAGE OFFSET AND C
00422 796D 58             ASL B            DOUBLE ACCB
```

```
00423 796E D7 15          STA B  TP1S     STORE IN LS WORD
00424 7970 49             ROL A           SHIFT IN C
00425 7971 97 14          STA A  TP1      RESTORE MS WORD
00426 7973 DE 14          LDX    TP1      GET TBL ADRS
00427 7975 EE 00          LDX    X        GET ROUTINE ADRS
00428 7977 86 00          LDA A  #0       CLEAR ACCA (BUT NOT C)
00429 7979 AD 00          JSR    X        GO EXECUTE IT!
00430 797B 7F 000F BWM    CLR    DIGFLG   CLEAR DIGIT ENTRY FLG
00431 797E C6 80          LDA B  #@200    MSB=1
00432 7980 D7 0D          STA B  STKFLG   ENABLE STACK LIFT
00433 7982 D6 06    BWM2  LDA B  ERROR    GET ERROR WORD
00434 7984 26 AA          BNE    EJSR7    YES; PRINT ERROR
00435 7986 DE CA          LDX    UIP      GET USER INST. POINTER
00436 7988 08             INX             POINT TO NEXT INSTRUCTION
00437 7989 DF CA          STX    UIP      SAVE THE UPDATE
00438 798B E6 00          LDA B  X        GET THE NEXT USER INSTRUCTION
00439 798D 96 CB          LDA A  UIP+1    CHECK VALIDITY OF NEW POINTER
00440 798F 90 0C          SUB A  EOPM+1   BY DOING (UIP)-(EOPM)
00441 7991 96 CA          LDA A  UIP
00442 7993 92 0B          SBC A  EOPM
00443 7995 2A 15          BPL    MAW      ERROR IF INVALID ADRS
00444 7997 96 09          LDA A  RSFLG    SHOULD WE STOP ?
00445 7999 2B D0          BMI    EXEC7    BRANCH IF NOT
00446 799B 48             ASL A           GET "ARE DOING" BIT
00447 799C 2A 08          BPL    TT7      RETURN TO TOP IF STOPPED
00448 799E 96 12          LDA A  SFLG     ELSE GET STEP FLAG
00449 79A0 26 04          BNE    SS       BRANCH IF STEPPING
00450 79A2 96 0D          LDA A  STKFLG   ELSE CHECK STACK FLAG
00451 79A4 27 C5          BEQ    EXEC7    CONTINUE IF LIFT DISABLED
00452 79A6 BD 4800  SS    JSR    RSEX     ELSE STOP THE PROGRAM
00453 79A9 7E 78C9  TT7   JMP    TOP7     GO BACK TO THE KEYBOARD
00454 79AC C6 07    MAW   LDA B  #7       LOAD ERROR
00455 79AE D7 06          STA B  ERROR    SAVE IN ERROR LOC.
00456 79B0 CE 0100        LDX    #256     GET USER STATE 0
00457 79B3 DF CA          STX    UIP      RESET INST PNTR
00458 79B5 20 CD          BRA    BWM2+2   PRINT THE ERROR
00459                *
00460                *
00461                *
00462       4800     RSEX   EQU    $4800
00463       4EC8     STCOD7 EQU    $4EC8
00464       5C31     AUDIT  EQU    $5C31
00467                       NAM    RDKEY2
00468                       OPT    LIST,MEM
00469                *
00470                *
00471                *     RDKEY1 IS THE INTERRUPT SERVICE ROUTINE
00472                *     TO PROCESS A KEYBOARD INTERRUPT.
00473                *     THE KEYCODE FOR THE KEY WHICH
00474                *     IS DOWN WILL BE STORED IN THE BUFFER
00475                *     AREA AT LOCATION (BKWRT+1)+BKKC. IF THE
00476                *     BUFFER IS NOT FULL.
00477                *     IF THIS ROUTINE IS ENTERED AT
00478                *     RDTGL1 THEN THE CURRENT CONDITION OF
00479                *     THE TOGGLE SWITCHES IS READ.
00480                *     IF THERE IS NO CHANGE DETECTED BETWEEN
00481                *     (TGL) AND THE CURRENT SETTINGS THE B-REG
00482                *     IS CLEARED. IF THERE IS A DIFFERENCE THEN THE
00483                *     B-REG REFLECTS WHAT BITS CHANGED AND THE
00484                *     NEW SETTINGS ARE STORED IN TGL.
00485                *
00486                *     BRAD MILLER    3/4/74
00487                *
00488                *
00489 79B7 96 00    RDKEY1 LDA A  ADATA     GET CURRENT I/O SELECT
00490 79B9 8A 1F           ORA A  #$1F      SELECT THE KEYBOARD (SAVE CASS. BIT)
00491 79BB 97 00           STA A  ADATA     OUTPUT IT
00492 79BD 96 04           LDA A  INPUT     INPUT THE KEYCODE
00493 79BF 43              COM A           INVERT THE BITS
00494 79C0 44              LSR A           MOVE THEM TO LSB'S
00495 79C1 44              LSR A
00496 79C2 D6 00           LDA B  ADATA     GET I/O SELECT
00497 79C4 C4 F0           AND B  #$F0      MASK OFF THE KEYBOARD
00498 79C6 D7 00           STA B  ADATA     DISABLE THE KEYBOARD
00499 79C8 D6 D5           LDA B  FLAG      CASSETTE RUNNING ?
00500 79CA C5 02           BIT B  #2        TEST BIT 1
```

```
00501 79CC 27 19          BEQ     NOPE2     BRANCH IF NOT
00502 79CE 81 25          CMP A   #@45      R/S KEY ?
00503 79D0 26 0C          BNE     YES3      BRANCH IF NOT
00504 79D2 7F 00D6        CLR     TPOS      ELSE RESET TAPE POSITION POINTER
00505 79D5 78 0009        ASL     RSFLG     ZERO TO MSB OF
00506 79D8 74 0009        LSR     RSFLG     RUN/STOP FLAG
00507 79DB 7E 61BE        JMP     TRNOFF    TURN OFF CASSETTE
00508 79DE 81 0A   YES3   CMP A   #@12      SET FLAG KEY ?
00509 79E0 26 12          BNE     YES1-1    RETURN IF NOT
00510 79E2 D6 09          LDA B   RSFLG     PROGRAM RUNNING ?
00511 79E4 2B 1E          BMI     YES2+3    BRANCH IF YES
00512 79E6 3B             RTI               ELSE RETURN
00513 79E7 D6 09   NOPE2  LDA B   RSFLG     GET THE RUN/STOP FLAG
00514 79E9 81 25          CMP A   #@45      RUN/STOP KEY ?
00515 79EB 27 08          BEQ     YES1      YES; GO PROCESS
00516 79ED 81 0A          CMP A   #@12      SET FLAG KEY ?
00517 79EF 27 10          BEQ     YES2      YES; GO PROCESS
00518 79F1 58             ASL B             PROGRAM RUNNING ?
00519 79F2 2A 17          BPL     NOPE      BRANCH IF NOT
00520 79F4 3B             RTI               ELSE IGNORE THE KEY
00521 79F5 58      YES1   ASL B             ZERO TO MSB
00522 79F6 54             LSR B             OF RUN/STOP FLAG
00523 79F7 D7 09          STA B   RSFLG     RESTORE NEW FLAG
00524 79F9 C5 C0          BIT B   #$C0      SHOULD WE BUFFER THE KEY ?
00525 79FB 27 18          BEQ     NOPE1     BRANCH IF YES AND BUFFER
00526 79FD 7F 0012        CLR     SFLG      RESET STEP FLAG
00527 7A00 3B             RTI               ELSE RETURN
00528 7A01 5D      YES2   TST B             CHECK RUN/STOP FLAG
00529 7A02 2A 07          BPL     NOPE      CONTINUE IF B7=0
00530 7A04 86 80          LDA A   #$80      PRESET ACCA
00531 7A06 98 08          EOR A   UFLG      TOGGLE FLAG B
00532 7A08 97 08          STA A   UFLG      RESTORE THE FLAGS
00533 7A0A 3B             RTI               RETURN FROM THE INTERRUPT
00534 7A0B 81 08   NOPE   CMP A   #@13      STEP KEY ?
00535 7A0D 27 06          BEQ     NOPE1     BRANCH IF YES
00536 7A0F C6 BF          LDA B   #$BF      PRESET ACCB
00537 7A11 D4 08          AND B   UFLG      CLEAR ENTRY FLAG
00538 7A13 D7 08          STA B   UFLG      RESTORE FLAGS
00539 7A15 D6 89   NOPE1  LDA B   BKWRT+1    LOAD WRITE POINTER
00540 7A17 C1 0C          CMP B   #12       IS BUFFER FULL ?
00541 7A19 27 07          BEQ     RDKEY3    YES, IGNORE KEY AND RETURN
00542 7A1B DE 88          LDX     BKWRT     NO,LOAD WRITE POINTER
00543 7A1D A7 8A          STA A   BKKC,X    STORE KEYCODE IN NEXT LOCATION
00544 7A1F 5C             INC B             INCREMENT WRITE POINTER FOR NEXT KEY
00545 7A20 D7 89          STA B   BKWRT+1    STORE UPDATED WRITE POINTER
00546 7A22 3B      RDKEY3 RTI               RETURN FROM INTERRUPT
00547                     *
00548                     * TOGGLE SWITCHES ARE READ HERE
00549                     *
00550 7A23 0F      RDTGL1 SEI               DISABLE INTERRUPTS
00551 7A24 86 0C          LDA A   #$C       LOAD TOGGLE SELECT CODE
00552 7A26 97 00          STA A   ADATA     OUTPUT THE SELECT CODE
00553 7A28 96 04          LDA A   INPUT     INPUT THE TOGGLE CONDITIONS
00554 7A2A 43             COM A             INVERT THE BITS
00555 7A2B 7F 0000        CLR     ADATA     DISABLE THE I/O TRANSFER
00556 7A2E 0E             CLI               RE-ENABLE INTERRUPTS
00557 7A2F D6 07          LDA B   TGL       LOAD OLD CONDITIONS
00558 7A31 C4 1F          AND B   #$1F      MASK OFF TRIG AND FORMAT
00559 7A33 84 E0          AND A   #$E0      MASK OFF AUTO-START AND FORMAT
00560 7A35 1B      RDT    ABA               COMBINE NEW WORD
00561 7A36 D6 07          LDA B   TGL       GET OLD CONDITIONS
00562 7A38 97 07          STA A   TGL       STORE THE NEW CONDITIONS
00563 7A3A 17             TBA               SAVE OLD CONDITIONS
00564 7A3B D8 07          EOR B   TGL       ACCB=CHANGED BITS
00565 7A3D 39             RTS               RETURN
00566      61BE    TRNOFF EQU     $61BE
00567                     *
00568                     *ROUTINE TO SET DEGREES, RADIANS, GRADS
00569                     *
00570 7A3E 49      RADS   ROL A
00571 7A3F 49      GRADS  ROL A
00572 7A40 D6 07   DEGS   LDA B   TGL       GET OLD SETTING
00573 7A42 C4 FC          AND B   #$FC      MASK OFF OLD MODE
00574 7A44 20 EF          BRA     RDT       INVOKE NEW SETTING
00575                     *
00576 7A46 49      FIXED  ROL A
```

```
00577 7A47 49       SCI3   ROL A
00578 7A48 49       SCI    ROL A
00579 7A49 49              ROL A
00580 7A4A 49              ROL A
00581 7A4B D6 07           LDA B  TGL       GET OLD DISPLAY MODE
00582 7A4D C4 E3           AND B  #$E3      MASK OFF OLD SETTING
00583 7A4F 1B              ABA               INVOKE NEW SETTING
00584 7A50 97 07           STA A  TGL       SAVE THE UPDATE
00585 7A52 DE CA           LDX    UIP       GET THE INSTRUCTION POINTER
00586 7A54 08              INX               MOVE AHEAD ONE
00587 7A55 DF CA           STX    UIP       SAVE THE UPDATE
00588 7A57 A6 00           LDA A  X         GET THE ROUND SETTING
00589 7A59 81 09           CMP A  #9        LARGER THAN 9 ?
00590 7A5B 22 02           BHI    RD7       IGNORE IT; RAM/INSTRUCTION FAILURE
00591 7A5D 97 0E           STA A  RND       SAVE IT
00592 7A5F 39       RD7    RTS              RETURN
00593 7C5C                 ORG    $7C5C
00594 7C5C 7A40            FDB    DEGS
00595 7C7C                 ORG    $7C7C
00596 7C7C 7A3F            FDB    GRADS
00597 7C7E 7A3E            FDB    RADS
00598 7D76                 ORG    $7D76
00599 7D76 7A46            FDB    FIXED
00600 7D78 7A48            FDB    SCI
00601 7D7A 7A47            FDB    SCI3
00602 7FF8                 ORG    $7FF8
00603 7FF8 7987            FDB    RDKEY1
00606                      NAM    DISPLAY
00607                      OPT    LIST,MEM
00608 7A60                 ORG    $7A60
00609                *
00610                *THIS IS THE ROUTINE THAT WILL DRIVE THE
00611                *DISPLAY WHILE WAITING FOR THE NEXT KEY.
00612                *IT WILL CHECK FOR A KEY AFTER EACH CHARACTER
00613                *DISPLAYED AND WILL READ THE TOGGLE SWITCHES
00614                *AFTER EACH COMPLETE PASS (16 CHAR.) THRU THE
00615                *DISPLAY. IF THE SWITCHES HAVE CHANGED, A MODE
00616                *CHANGE WILL RESULT IN TERMINATION OF THE
00617                *PREVIOUS CODE BEING BUILT (IF POSSIBLE) AND
00618                *A FORMAT CHANGE WILL RESULT IN A RE-FORMATTING
00619                *OF THE DISPLAY BUFFER.
00620                *
00621                *BRAD MILLER  MAY 7, 1974
00622                *
00623                *
00624 7A60 86 40    PAUSE  LDA A  #$40     GET PAUSE FACTOR
00625 7A62 20 02           BRA    SDISP1+2   GO SAVE IT
00626 7A64 86 01    SDISP1 LDA A  #1       ODD # DISABLES PAUSE
00627 7A66 97 1A           STA A  TP4       SAVE PAUSE FACTOR
00628 7A68 D6 B9           LDA B  BKWRT+1    IS A KEY AVAILABLE ?
00629 7A6A 26 07           BNE    SDISP3    BRANCH IF YES
00630 7A6C BD 4E09 SDISPC  JSR    FRMT1     INITIALIZE DISPLAY BUFFER
00631 7A6F D6 B9   SDISP0  LDA B  BKWRT+1    KEY BUFFER EMPTY ?
00632 7A71 27 12           BEQ    SDISP2    BRANCH IF YES
00633 7A73 0F      SDISP3  SEI               DISABLE INTERRUPTS
00634 7A74 7A 00B9         DEC    BKWRT+1    DECREMENT KEY POINTER
00635 7A77 96 BA           LDA A  BKKC      LOAD THE CURRENT KEYCODE
00636 7A79 CE FFF5         LDX    #-11       PRESET LOOP COUNTER
00637 7A7C E6 C6   SDISP4  LDA B  BKKC+12,X   LOAD THE NEXT KEYCODE
00638 7A7E E7 C5           STA B  BKKC+11,X   RESTORE IT IN NEW LOCATION
00639 7A80 08              INX               IS LOOP COMPLETED?
00640 7A81 26 F9           BNE    SDISP4    NO: KEEP DROPPING BUFFER
00641 7A83 0E              CLI               YES: RE-ENABLE INTERRUPTS
00642 7A84 39      PAUSE1 RTS               RETURN
00643 7A85 BD 7A23 SDISP2  JSR    RDTGL1    READ THE TOGGLE SWITCHES
00644 7A88 2A 1E           BPL    SDISP7    BRANCH IF NO MODE CHANGE
00645 7A8A D6 1A           LDA B  TP4       PAUSING OR DISPLAYING ?
00646 7A8C 54              LSR B
00647 7A8D 24 19           BCC    SDISP7    IGNORE THE CHANGE IF PAUSING
00648 7A8F D6 C6           LDA B  SOL7      GET INST BEING BUILT
00649 7A91 C1 BE           CMP B  #$276     2 BYTE INST ?
00650 7A93 22 0C           BHI    SDISP6    BRANCH IF YES
00651 7A95 31              INS               WIPE OUT THE
00652 7A96 31              INS               OLD RETURN VECTOR
00653 7A97 96 52           LDA A  TA        INSERT MODE SET ?
```

```
00654 7A99 27 03          BEQ     SDISP6-3   BRANCH IF NOT
00655 7A9B 7E 4F48        JMP     INSERT     ELSE TERMN. INSERT
00656 7A9E 7E 78C9        JMP     TOP7       RESET THE SYSTEM
00657 7AA1 97 07   SDISP6 STA A   TGL        RESTORE OLD CONDITIONS
00658 7AA3 31             INS                WIPE OUT THE RETURN
00659 7AA4 31             INS                VECTOR AND TERMINATE
00660 7AA5 7E 4A22        JMP     DECPT      THE PREVIOUS OPERATION
00661 7AA8 86 0E   SDISP7 LDA A   #$E         LOAD DISPLAY S.C.
00662 7AAA CE 0058        LDX     #BUFF      INDEX=BUFFER ADDRESS
00663 7AAD DF 16          STX     TP2        SAVE BUFFER ADDRESS
00664 7AAF DE 16   SDISP8 LDX     TP2        GET LATEST BUFFER ADDRESS
00665 7AB1 E6 00          LDA B   X          FETCH THE ASCII CHARACTER
00666 7AB3 D7 18          STA B   TP4S       SAVE IT FOR COMMA CHECK
00667 7AB5 C4 7F          AND B   #$7F       MASK OFF COMMA INDICATOR(MSB)
00668 7AB7 C0 2D          SUB B   #$2D       REMOVE ASCII OFFSET
00669 7AB9 24 02          BCC     SDISP9     CONTINUE IF "LEGAL" CODE
00670 7ABB C6 02          LDA B   #2         ELSE LOAD "BLANK" BY DEFAULT
00671 7ABD 08      SDISP9 INX                INC BUFFER POINTER
00672 7ABE DF 16          STX     TP2        SAVE UPDATED VALUE
00673 7AC0 CE 7B13        LDX     #DSPTBL    SET INDEX TO CHAR DECODE TBL
00674 7AC3 DF 18          STX     TP3        SAVE IN TEMP
00675 7AC5 DB 19          ADD B   TP3S       ADD CHAR OFFSET TO TBL ADRS
00676 7AC7 D7 19          STA B   TP3S       RESTORE NEW ADDRESS
00677 7AC9 DE 18          LDX     TP3        RESTORE THE INDEX
00678 7ACB C6 FF          LDA B   #$FF       ACCB="BLANK"
00679 7ACD D7 02          STA B   BDATA      TURN OFF THE SEGMENTS
00680 7ACF C6 3C          LDA B   #$3C       PRESET ACCB
00681 7AD1 D7 03          STA B   BCTL       TURN OFF THE COMMA
00682 7AD3 97 00          STA A   ADATA      OUTPUT S.C. AND CHAR PNTR
00683 7AD5 C6 04          LDA B   #4         LOAD "SEGMENT ON" WAIT COUNTER
00684 7AD7 5A      SDISPF DEC B               DECREMENT IT
00685 7AD8 26 FD          BNE     SDISPF     CONTINUE THE LOOP
00686 7ADA E6 00          LDA B   X          GET THE NEW SEGMENT INFO
00687 7ADC D7 02          STA B   BDATA      OUTPUT SEGMENT INFORMATION
00688 7ADE D6 18          LDA B   TP4S       GET THE ORIGINAL CHARACTER
00689 7AE0 2A 04          BPL     SDISPD     BRANCH IF NO COMMA
00690 7AE2 C6 34          LDA B   #$34       ELSE LOAD COMMA CODE (CB2)
00691 7AE4 D7 03          STA B   BCTL       LIGHT THE COMMA
00692 7AE6 8B 10   SDISPD ADD A   #$10       INC. THE CHARACTER POINTER
00693 7AE8 24 16          BCC     SDISPA     CHECK FOR A NEW KEY
00694 7AEA C6 16          LDA B   #22        LOAD LAST DIGIT WAIT
00695 7AEC BD 4BA0        JSR     WAIT       GO WAIT
00696 7AEF 53             COM B              ACCB="BLANK"
00697 7AF0 D7 02          STA B   BDATA      TURN OFF CHARACTER 15
00698 7AF2 5F             CLR B
00699 7AF3 D7 00          STA B   ADATA      DISABLE DISPLAY
00700 7AF5 7A 001A        DEC     TP4        PAUSE=PAUSE-2
00701 7AF8 7A 001A        DEC     TP4
00702 7AFB 27 87          BEQ     PAUSE1     RETURN IF ZERO
00703 7AFD 7E 7A6F        JMP     SDISP0     ELSE CONTINUE DRIVING
00704 7B00 C6 09   SDISPA LDA B   #9         LOAD DIGIT "ON" TIME
00705 7B02 BD 4BA0        JSR     WAIT       GO WAIT
00706 7B05 D6 B9          LDA B   BKWRT+1    KEY BUFFER EMPTY ?
00707 7B07 27 A6          BEQ     SDISP8     BRANCH IF YES
00708 7B09 7F 0000        CLR     ADATA      ELSE DISABLE DISPLAY
00709 7B0C C6 3C          LDA B   #$3C       PRESET ACCB
00710 7B0E D7 03          STA B   BCTL       TURN OFF THE COMMA
00711 7B10 7E 7A73        JMP     SDISP3     GET NEW KEY AND RETURN
00712                     *
00713                     **********CHARACTER DECODE TABLE**********
00714                     *
00715                     *ALL ENTRIES MUST BE ON THE SAME PAGE !
00716                     *
00717                     *A "0" ENABLES A GIVEN SEGMENT
00718                     *
00719                     *CODING IS A,B,C,D,E,F,G,DEC. PT. IN BITS 7 THRU 0
00720                     *
00721 7B13 FD      DSPTBL FCB     @375       -
00722 7B14 FE             FCB     @376       .
00723 7B15 FF             FCB     @377       BLANK
00724 7B16 03             FCB     @3         0
00725 7B17 9F             FCB     @237       1
00726 7B18 25             FCB     @45        2
00727 7B19 0D             FCB     @15        3
00728 7B1A 99             FCB     @231       4
00729 7B1B 49             FCB     @111       5
00730 7B1C 41             FCB     @101       6
```

```
00731 781D 1F              FCB    @37      7
00732 781E 01              FCB    @1       8
00733 781F 09              FCB    @11      9
00734 7820 31              FCB    @61      P
00735 7821 63              FCB    @143     C
00736 7822 FD              FCB    @375     -
00737 7823 11              FCB    @21      A
00738 7824 FD              FCB    @375     -
00739 7825 E3              FCB    @343     L
00740 7826 01              FCB    @1       B
00741 7827 E3              FCB    @343     L
00742 7828 FD              FCB    @375     -
00743 7829 73              FCB    @163     R
00744 782A 21              FCB    @41      E
00745 782B 43              FCB    @103     G
00746 782C FD              FCB    @375     -
00747 782D 11              FCB    @21      A
00748 782E E3              FCB    @343     L
00749 782F 31              FCB    @61      P
00750 7830 91              FCB    @221     H
00751 7831 11              FCB    @21      A
00752 7832 F1              FCB    @361     INSERT CHARACTER
00753 7833 00              FCB    0        ALL SEGMENTS AND DP
00754 7834 61              FCB    @141     E
00755 7835 71              FCB    @161     F
00756 7836 87              FCB    @207     J
00757 7837 83              FCB    @203     U
00758 7838 C1              FCB    @301     B
00759 7839 85              FCB    @205     D
00760 48A0                 ORG    $4BA0
00761                 *
00762                 *WAIT LOOP IN ANOTHER ROM FOR POWER CONSIDERATIONS
00763                 *
00764 48A0 5A         WAIT   DEC B
00765 48A1 26 FD           BNE    WAIT
00766 48A3 39              RTS
00767                 *
00768                 *
00769       4F48      INSERT EQU    $4F48
00770       4E09      FRMT1  EQU    $4E09
00771       4A22      DECPT  EQU    $4A22
00772 7D52                 ORG    $7D52
00773 7D52 7A60            FDB    PAUSE
00774 7D68                 ORG    $7D68
00775 7D68 7A84            FDB    PAUSE1
00776 7C00                 ORG    $7C00
00777 7C00 7A84            FDB    PAUSE1
00778 7D74                 ORG    $7D74
00779 7D74 7A84            FDB    PAUSE1
00782                      NAM    ERROR
00783                      OPT    LIST,MEM,NG
00784 7B40                 ORG    $7B40
00785       5300      ROMID  EQU    $5300
00786       51A7      TOALP1 EQU    $51A7
00787                 *
00788                 *THIS IS THE ERROR OUTPUT ROUTINE. IT ACCEPTS
00789                 *AN ERROR NUMBER IN THE BASE PAGE LOCATION "ERROR"
00790                 *AND OUTPUTS THE ASSOCIATED ALPHA MESSAGE TO
00791                 *THE THERMAL PRINTER (REGARDLESS OF "PAPER OUT" BIT)
00792                 *
00793                 *ALL MAINFRAME ERROR NUMBERS ARE POSITIVE AND
00794                 *SIMPLY SPECIFY THE NUMBER IN THE LIST OF MESSAGES.
00795                 *FOR EXAMPLE: 1 MEANS THE FIRST ENTRY, 2 MEANS THE SECOND AN
00796                 *SO ON. THE MSB OF THE FIRST CHARACTER OF EACH ENTRY
00797                 *IS SET TO INDICATE ITS START.
00798                 *
00799                 *ALL I/O DEVICE ERROR NUMBERS ARE NEGATIVE AND THE
00800                 *ROM ID NUMBER IS CONTAINED IN BITS B6 THRU B4.
00801                 *BITS B3 THRU B0 CONTAIN THE TABLE OFFSET AS DESCRIBED
00802                 *ABOVE FOR MAINFRAME ERRORS. NOTICE THAT
00803                 *I/O ROMS ARE RESTRICTED TO NO MORE THAN
00804                 *15 ERROR MESSAGES.
00805                 *
00806                 *MAINFRAME ERRORS 1 THRU 5 ARE MASKABLE IN A
00807                 *RUNNING PROGRAM. IF USER FLAG 6 IS SET WHEN
00808                 *THEY OCCUR. FLAG 5 WILL BE SET AND THE PROGRAM
```

```
00809                *WILL CONTINUE RUNNING.
00810                *
00811                *ERROR ZERO IS A SPECIAL ERROR GENERATED BY THE
00812                *SYNTAX TABLES WHICH IS USED TO TERMINATE NUMERIC
00813                *ADDRESSES UPON RECEIPT OF A NON-NUMERIC KEY.
00814                *
00815                *WRITTEN BY BRAD MILLER
00816                *
00817 7B40 D6 09    ERROR7 LDA B  RSFLG     GET RUN/STOP FLAG
00818 7B42 58              ASL B            RUNNING?
00819 7B43 2A 26           BPL    CNT       BRANCH IF NOT
00820 7B45 D6 06           LDA B  ERROR     GET ERROR NUMBER
00821 7B47 C1 05           CMP B  #5        MASKABLE ERROR ?
00822 7B49 23 08           BLS    MSKABL    BRANCH IF YES
00823 7B4B C1 0E           CMP B  #14       MASKABLE ERROR ?
00824 7B4D 27 04           BEQ    MSKABL    BRANCH IF YES
00825 7B4F C1 10           CMP B  #16       MASKABLE ERROR ?
00826 7B51 26 11           BNE    CNTT      BRANCH IF NOT
00827 7B53 C6 20    MSKABL LDA B  #$20      LOAD FLAG 6 BIT
00828 7B55 D5 08           BIT B  UFLG      FLAG 6 SET ?
00829 7B57 27 0B           BEQ    CNTT      BRANCH IF NOT
00830 7B59 54              LSR B            ELSE PRESET ACCB
00831 7B5A DA 08           ORA B  UFLG      SET USER FLAG 5
00832 7B5C D7 08           STA B  UFLG      RESTORE FLAGS
00833 7B5E 7F 0006         CLR    ERROR     CLEAR OUT THE ERROR
00834 7B61 7E 7986         JMP    RWM2+4    CONTINUE RUNNING PRGM
00835 7B64 7F 0009 CNTT    CLR    RSFLG     STOP RUNNING
00836 7B67 DE CA           LDX    UIP       GET CURRENT PRGM ADRS
00837 7B69 DF C8           STX    UPP       SAVE IN PRGM PNTR
00838 7B6B D6 06    CNT    LDA B  ERROR     GET ERROR NUMBER
00839 7B6D 26 1A           BNE    EOUT      BRANCH IF NOT ERROR 0
00840 7B6F 0F              SEI              DISABLE KEY INTERRUPT
00841 7B70 CE 000B         LDX    #11       PRESET LOOP COUNTER
00842 7B73 E6 B9    KB1    LDA B  BKKC-1,X    GET "BOTTOM" KEY
00843 7B75 E7 BA           STA B  BKKC,X    MOVE IT DOWN
00844 7B77 09              DEX              DECREMENT POINTER/COUNTER
00845 7B78 26 F9           BNE    KB1       CONT IF NOT FINISHED
00846 7B7A 97 BA           STA A  BKKC      PUT THE OLD KEY BACK ON TOP
00847 7B7C D6 B9           LDA B  BKWRT+1     GET THE WRITE INDICATOR
00848 7B7E C1 0C           CMP B  #12       BUFFER FULL ?
00849 7B80 2C 03           BGE    CB        BRANCH IF YES
00850 7B82 5C              INC B            ELSE REGISTER THIS ENTRY
00851 7B83 D7 B9           STA B  BKWRT+1     SAVE THE NEW INDICATOR
00852 7B85 0E       CB     CLI              RE-ENABLE THE INTERRUPT SYSTEM
00853 7B86 7E 4A22         JMP    DECPT     TERMINATE THE PREVIOUS INSTRUCTION
00854 7B89 4F       EOUT   CLR A            PRESET ACCA
00855 7B8A 97 B9           STA A  BKWRT+1     EMPTY KEY BUFFER
00856 7B8C 97 0F           STA A  DIGFLG    TERM. DIGIT ENTRY
00857 7B8E 43              COM A
00858 7B8F 97 0D           STA A  STKFLG    ENABLE STACK LIFT
00859 7B91 BD 5D75         JSR    BLANK     BLANK THE BUFFER
00860 7B94 CE 0058         LDX    #BUFF     PRESET POINTER FOR
00861 7B97 DF 16           STX    TP2       THE LOAD MESSAGE ROUTINE
00862 7B99 CE 53FF         LDX    #EMSG-1     GET ERROR MESSAGE TABLE POINTER
00863 7B9C 96 06           LDA A  ERROR     GET THE ERROR
00864 7B9E 2A 0F           BPL    MAINF     BRANCH IF MAINFRAME ERROR
00865 7BA0 84 70           AND A  #$70      ELSE MASK OFF I/O ID NUMBER
00866 7BA2 BD 5300         JSR    ROMID     GENERATE ROM ADDRESS
00867 7BA5 C6 74           LDA B  #$74      GET ERROR TABLE OFFSET
00868 7BA7 D7 25           STA B  T10       SAVE IN ROM ID TEMP (LSB'S)
00869 7BA9 DE 24           LDX    T11       INDEX=ERROR TABLE ADRS-1
00870 7BAB 96 06           LDA A  ERROR     ACCA=ERROR
00871 7BAD 84 0F           AND A  #$F       ACCA=TABLE OFFSET NUMBER
00872 7BAF 08       MAINF  INX              BUMP THE TABLE SEARCH ADDRESS
00873 7BB0 E6 00           LDA B  X         GET CURRENT ENTRY
00874 7BB2 2A FB           BPL    MAINF     CONTINUE LOOKING IF POSITIVE
00875 7BB4 4A              DEC A            ELSE DEC. OFFSET NUMBER
00876 7BB5 26 F8           BNE    MAINF     CONTINUE IF ITS NOT THE ONE
00877 7BB7 97 06           STA A  ERROR     CLEAR ERROR WORD
00878 7BB9 4A              DEC A            MAKE ACCA NON-ZERO FOR LDMSG
00879 7BBA BD 57BD         JSR    LDMSG     ERROR NOTE TO BUFFER
00880 7BBD BD 6046         JSR    PRTDRV+25   PRINT THE ERROR
00881 7BC0 96 CC           LDA A  ALPHA     ALPHA MODE ?
00882 7BC2 26 03           BNE    TA1       BRANCH IF YES
00883 7BC4 7E 78C9         JMP    TOP7      ELSE RETURN TO TOP
00884 7BC7 7E 51A7 TA1     JMP    TOALP1    DO "TO ALPHA"
```

```
00885 5400                ORG    $5400
00886 5400 CF       EMSG  FCB    $CF
00887 5401 56             FCC    /VERFLOW/
00888 5408 D3             FCB    $D3
00889 5409 51             FCC    /QRT OF NEG #/
00890 5415 C4             FCB    $C4
00891 5416 49             FCC    /IVISION BY ZERO/
00892 5425 CC             FCB    $CC
00893 5426 4F             FCC    /OG OF # <=0/
00894 5431 CE             FCB    $CE
00895 5432 4F             FCC    .O I/O DEVICE.
00896 543E C9             FCB    $C9
00897 543F 4C             FCC    /LLEGAL ARGUMENT/
00898 544E CD             FCB    $CD
00899 544F 45             FCC    /EMORY OVERFLOW/
00900 545D CC             FCB    $CC
00901 545E 41             FCC    /ABEL NOT FOUND/
00902 546C C7             FCB    $C7
00903 546D 4F             FCC    /OSUB OVERFLOW/
00904 547A CD             FCB    $CD
00905 547B 49             FCC    /ISSING GOSUB/
00906 5487 CB             FCB    $CB
00907 5488 45             FCC    /EY NOT DEFINED/
00908 5496 C9             FCB    $C9
00909 5497 4D             FCC    /MPROPER SYNTAX/
00910 54A5 CD             FCB    $CD
00911 54A6 49             FCC    /ISSING FOR STMT/
00912 54B5 C3             FCB    $C3
00913 54B6 48             FCC    /HECKSUM ERROR/
00914 54C3 C6             FCB    $C6
00915 54C4 49             FCC    /ILE TOO SMALL/
00916 54D1 D6             FCB    $D6
00917 54D2 45             FCC    /ERIFY FAILED/
00918 54DE D7             FCB    $D7
00919 54DF 52             FCC    /RONG FILE TYPE/
00920 54ED C6             FCB    $C6
00921 54EE 49             FCC    /ILE NOT FOUND/
00922 54FB C5             FCB    $C5
00923 54FC 4E             FCC    /ND OF TAPE/
00924 5506 C3             FCB    $C3
00925 5507 41             FCC    /ARTRIDGE OUT/
00926 5513 D0             FCB    $D0
00927 5514 52             FCC    /ROTECTED TAPE/
00928 5521 D3             FCB    $D3
00929 5522 45             FCC    /ECURED MEMORY/
00930 552F CF             FCB    $CF
00931 5530 55             FCC    /UT OF PAPER/
00932 553B C9             FCB    $C9
00933 553C 4C             FCC    /LLEGAL ADDRESS/
00934 554A 80             FCB    $80
00937                     END
```

SYMBOL TABLE

ADATA	0000	ACTL	0001	BDATA	0002	BCTL	0003	INPUT	0004	IOIN	0005
ERROR	0006	TGL	0007	UFLG	0008	RSFLG	0009	EOM	000A	EOPM	000B
STKFLG	000D	RND	000E	DIGFLG	000F	W2	0010	W1	0011	SFLG	0012
DCNTR	0013	TP1	0014	TP1S	0015	TP2	0016	TP2S	0017	TP3	0018
TP3S	0019	TP4	001A	TP4S	001B	TP5	001C	TP5S	001D	TP6	001E
TP6S	001F	TP7	0020	TP7S	0021	T13	0022	T12	0023	T11	0024
T10	0025	T9	0026	T8	0027	T7	0028	T6	0029	T5	002A
T4	002B	T3	002C	T2	002D	T1	002E	ISTK	002F	ISTACK	0051
TA	0052	SPGM	0054	EXTRA	0056	BUFF	0058	PEAL	0068	IMAG	0070
AT1	0078	AT2	0080	W	0088	XR	0090	YR	0098	ZR	00A0
TR	00A8	LSTX	00B0	BKWRT	00B8	BKKC	00BA	SOL7	00C6	UPP	00C8
UIP	00CA	ALPHA	00CC	IO1	00CD	IO2	0000	IT7	00D3	FLAG	00D5
TPOS	00D6	FILE	00D7	AR	00D8	BR	00E0	CR	00E8	DR	00F0
ER	00F8	SDBB	00BA	MT	7E00	TERMN7	003D	IMED	0040	PARCD	00C0
PAREX	0080	NTBL	0000	DOTS	5EC0	PRTDRV	602D	FRMT	5CA8	BLANK	5D75
LOMSG	57BD	ROLLO	55B2	ROLLU	57F1	PSD	55DA	TXL	55E9	STKUP	55EF
MAD	749B	CMP	74AA	NOR	74D6	TXW	7424	TXXR	743B	EXXR	7452
ARSR	753B	OVUNF	75B6	OVERF	75DD	XR0	740A	XRNINE	75C8	UNDRF	75F1
IMULT	7689	ODG	7669	FPA	75FC	FPS	75F6	FPM	7735	FPD	7793
FPAEX	7630	FPMEX	7780	LSHIFT	7521	ZEROX	7489	XZERO0	7416	XZERO2	7417

RECIP	73E6	TXRX	73F3	CONST	6800	FPDBRC	6898	TAN	68A9	ATN	69C3
DSZERO	6A46	NTLN	6A58	EXPN	6AC9	SIN	6B94	COS	6B9A	ASIN	6BF2
ACOS	6BF7	PH1	6C5D	PH2	6C8D	PH3	6D34	PH4	6D00	LSFT8	6E47
SQRT	6E65	MAD8	6F2C	CMP8	53E4	IOUPX	6F52	LOG10	6FA7	YUPX	6FE9
RTOP	7328	PTOR	7386	NOTEST	780D	MORE	781C	ERASE	7820	IOPOLL	7854
NOIO2	786A	IOERR	7886	AUTOOK	7889	MSG1	788E	TOP7	78C9	NK7	78E3
TSRCH6	78F3	TSRCH7	78F8	TWOB7	7908	TFND7	790C	FIX	7924	ERR7	792C
EJSR7	7930	MFND7	7933	L7	793C	IMEX7	7948	CONT	794C	PARTI7	7957
PARCD7	795F	EXEC6	7968	EXEC7	796B	RWM	797B	RWM2	7982	SS	79A6
TT7	79A9	MAW	79AC	RSEX	4800	STCOD7	4EC8	AUDIT	5C31	RDKEY1	7987
YES3	79DE	NOPE2	79E7	YES1	79F5	YES2	7A01	NOPE	7A08	NOPE1	7A15
RDKEY3	7A22	RDTGL1	7A23	RDT	7A35	TRNOFF	618E	RADS	7A3E	GRADS	7A3F
DEGS	7A40	FIXED	7A46	SCI3	7A47	SCI	7A48	RD7	7A5F	PAUSE	7A60
SDISP1	7A64	SDISPC	7A6C	SDISP0	7A6F	SDISP3	7A73	SDISP4	7A7C	PAUSE1	7A84
SDISP2	7A85	SDISP6	7AA1	SDISP7	7AA8	SDISP8	7AAF	SDISP9	7AB0	SDISPF	7AD7
SDISPD	7AE6	SDISPA	7B00	DSPTBL	7B13	WAIT	4BA0	INSERT	4F48	FRMT1	4E09
DECPT	4A22	ROMID	5300	TOALP1	51A7	ERROR7	7B40	MSKABL	7B53	CNTT	7B64
CNT	7B68	KB1	7B73	CB	7B85	EOUT	7B99	MAINF	7BAF	TA1	78C7
EMSG	5400										

TOTAL ERRORS 4

```
****ERROR 201
 157  NAM CJTBLS
****ERROR 213
 177 EQU11 EQU ST11-MT
****ERROR 213
 184 EQU12 EQU ST12-MT
****ERROR 213
 187 EQU13 EQU ST13-MT
****ERROR 213
 191 EQU14 EQU ST14-MT
****ERROR 213
 203 EQU22 EQU ST22-MT
****ERROR 213
 260 EQU1 EQU ST1-MT
****ERROR 213
 264 EQU2 EQU ST2-MT
****ERROR 213
 265 EQU16 EQU TT2-MT
****ERROR 213
 268 EQU3 EQU ST3-MT
****ERROR 213
 271 EQU4 EQU ST4-MT
****ERROR 213
 287 EQU5 EQU ST5-MT
****ERROR 213
 288 EQU20 EQU ST20-MT
****ERROR 213
 309 EQU7 EQU ST7-MT
****ERROR 213
 310 EQU15 EQU TT1-MT
****ERROR 213
 372 EQU6 EQU ST6-MT
****ERROR 213
 438 EQU21 EQU ST21-MT
****ERROR 213
 544 EQU17 EQU TT3-MT
****ERROR 213
 553 EQU18 EQU TT4-MT
****ERROR 201
 669  NAM PLABR
****ERROR 201
 856  NAM UDF
****ERROR 201
 932  NAM DTBL
00154                    OPT    LIST,MEM
00157                    NAM    CJTBLS
00158                    OPT    LIST,DB16,MEM
00159 7BCA               ORG    $7BCA
00160              *
00161              *FOLLOWING ARE THE KEY SYNTAX TABLES FOR CJ.
00162              *THESE TABLES, ALONG WITH THE SUPERVISOR SEARCH
```

```
00163                *ROUTINE, DEFINE THE KEY SEQUENCES RECEIVED FROM
00164                *THE KEYBOARD. THERE ARE FOUR POSSIBLE ACTIONS THAT
00165                *CAN BE TAKEN UPON FINDING A MATCH WITH THE
00166                *CURRENT KEYCODE: 1) IMMEDIATE EXECUTION OF A FUNCTION
00167                *2) RESET TO A NEW TABLE FOR FURTHER SEARCHING
00168                *3) JUMP TO A GIVEN ADDRESS AND EXECUTE SOME
00169                *INTERMEDIATE CODE 4) ADD A "BUILDING" CODE TO THE
00170                *INSTRUCTION REGISTER AND RESET TO A NEW SEARCH TABLE.
00171                *
00172                *
00173                *WRITTEN BY BRAD MILLER
00174                *
00175 7BCA D0        PRGM1  FCB    @20+PARCD   GOSUB
00176 7BCB E0               FCB    @340
00177       0168     EQU11  EQU    ST11-MT
00178 7BCC B4               FCB    EQU11/2
00179 7BCD 51               FCB    @21+IMED    RETURN
00180 7BCE B3               FCB    @263
00181 7BCF 52               FCB    @22+IMED    PAUSE
00182 7BD0 A9               FCB    @251
00183 7BD1 13               FCB    @23+NTBL    FOR
00184       0170     EQU12  EQU    ST12-MT
00185 7BD2 B8               FCB    EQU12/2
00186 7BD3 14               FCB    @24+NTBL    NEXT
00187       0178     EQU13  EQU    ST13-MT
00188 7BD4 BC               FCB    EQU13/2
00189 7BD5 D8               FCB    @30+PARCD   IF
00190 7BD6 48               FCB    @110
00191       00AC     EQU14  EQU    ST14-MT
00192 7BD7 56               FCB    EQU14/2
00193 7BD8 5C               FCB    @34+IMED    NOP
00194 7BD9 00               FCB    @0
00195 7BDA 62               FCB    @42+IMED    LINE FEED
00196 7BDB 3D               FCB    @75
00197 7BDC 61               FCB    @41+IMED    LOAD AND RUN
00198 7BDD A6               FCB    @246
00199 7BDE A4               FCB    @44+PAREX   INSERT
00200 7BDF 4F48             FDB    $4F48
00201 7BE1 A3               FCB    @43+PAREX   DELETE
00202 7BE2 4F60             FDB    $4F60
00203       00A0     EQU22  EQU    ST22-MT
00204 7BE4 20               FCB    @40+NTBL    MEMORY DELETE
00205 7BE5 50               FCB    EQU22/2
00206 7BE6 3D               FCB    TERMN7
00207 7BE7 7EAC             FDB    ST14
00208                *
00209                *
00210                *THE MAIN TABLE BEGINS HERE
00211                *
00212                *
00213 7E00                  ORG    MT
00214 7E00 66               FCB    @46+IMED    0
00215 7E01 01               FCB    @1
00216 7E02 5E               FCB    @36+IMED    1
00217 7E03 02               FCB    @2
00218 7E04 5F               FCB    @37+IMED    2
00219 7E05 03               FCB    @3
00220 7E06 77               FCB    @67+IMED    3
00221 7E07 04               FCB    @4
00222 7E08 56               FCB    @26+IMED    4
00223 7E09 05               FCB    @5
00224 7E0A 57               FCB    @27+IMED    5
00225 7E0B 06               FCB    @6
00226 7E0C 76               FCB    @66+IMED    6
00227 7E0D 07               FCB    @7
00228 7E0E 4E               FCB    @16+IMED    7
00229 7E0F 08               FCB    @10
00230 7E10 4F               FCB    @17+IMED    8
00231 7E11 09               FCB    @11
00232 7E12 75               FCB    @65+IMED    9
00233 7E13 0A               FCB    @12
00234 7E14 67               FCB    @47+IMED    .
00235 7E15 0B               FCB    @13
00236 7E16 46               FCB    @6+IMED     CHS
00237 7E17 12               FCB    @22
00238 7E18 47               FCB    @7+IMED     EEX
```

```
00239 7E19 11              FCB    @21
00240 7E1A 74              FCB    @64+IMED    CLX
00241 7E1B 13              FCB    @23
00242 7E1C 7F              FCB    @77+IMED    +
00243 7E1D 0C              FCB    @14
00244 7E1E 6F              FCB    @57+IMED    -
00245 7E1F 0D              FCB    @15
00246 7E20 6E              FCB    @56+IMED    X
00247 7E21 0E              FCB    @16
00248 7E22 73              FCB    @63+IMED    /
00249 7E23 0F              FCB    @17
00250 7E24 72              FCB    @62+IMED    PRINT
00251 7E25 82              FCB    @262
00252 7E26 7C              FCB    @74+IMED    XEY
00253 7E27 15              FCB    @25
00254 7E28 6D              FCB    @55+IMED    ROLL DOWN
00255 7E29 16              FCB    @26
00256 7E2A 6C              FCB    @54+IMED    ROLL UP
00257 7E2B 24              FCB    @44
00258 7E2C D5              FCB    @25+PARCD   RCL
00259 7E2D D2              FCB    @322
00260      00BC    EQU1    EQU    ST1-MT
00261 7E2E 5E              FCB    EQU1/2
00262 7E2F CD              FCB    @15+PARCD   STO
00263 7E30 C8              FCB    @310
00264      00C4    EQU2    EQU    ST2-MT
00265      01B0    EQU16   EQU    TT2-MT
00266 7E31 D8              FCB    EQU16/2
00267 7E32 30              FCB    @60+NTBL    F-1
00268      00CA    EQU3    EQU    ST3-MT
00269 7E33 65              FCB    EQU3/2
00270 7E34 31              FCB    @61+NTBL    SHIFT
00271      00E2    EQU4    EQU    ST4-MT
00272 7E35 71              FCB    EQU4/2
00273 7E36 45              FCB    @5+IMED     CLEAR
00274 7E37 14              FCB    @24
00275 7E38 6A              FCB    @52+IMED    SIN
00276 7E39 19              FCB    @31
00277 7E3A 68              FCB    @53+IMED    COS
00278 7E3B 1A              FCB    @32
00279 7E3C 7A              FCB    @72+IMED    TAN
00280 7E3D 1B              FCB    @33
00281 7E3E 68              FCB    @50+IMED    LN
00282 7E3F 1F              FCB    @37
00283 7E40 69              FCB    @51+IMED    LOG
00284 7E41 20              FCB    @40
00285 7E42 79              FCB    @71+IMED    ACC +
00286 7E43 2B              FCB    @53
00287      011C    EQU5    EQU    ST5-MT
00288      00A6    EQU20   EQU    ST20-MT
00289 7E44 78              FCB    @70+IMED    SIGMA+
00290 7E45 27              FCB    @47
00291 7E46 7B              FCB    @73+IMED    P TO R
00292 7E47 2A              FCB    @52
00293 7E48 65              FCB    @45+IMED    R/S
00294 7E49 80              FCB    @260
00295 7E4A 7E              FCB    @76+IMED    ENTER
00296 7E4B 10              FCB    @20
00297 7E4C 41              FCB    @1+IMED     REWIND
00298 7E4D AB              FCB    @250
00299 7E4E 40              FCB    @0+IMED     LD PRGM
00300 7E4F B9              FCB    @271
00301 7E50 83              FCB    @3+PAREX    LIST
00302 7E51 59D4            FDB    $59D4
00303 7E53 42              FCB    @2+IMED     RECORD PROGRAM
00304 7E54 B5              FCB    @265
00305 7E55 5D              FCB    @35+IMED    END
00306 7E56 B1              FCB    @261
00307 7E57 C8              FCB    @10+PARCD   GOTO
00308 7E58 F0              FCB    @360
00309      0160    EQU7    EQU    ST7-MT
00310      01B0    EQU15   EQU    TT1-MT
00311 7E59 B0              FCB    EQU7/2
00312 7E5A C9              FCB    @11+PARCD   LABEL
00313 7E5B BF              FCB    @277
00314 7E5C C0              FCB    EQU15/2
```

```
00315 7E5D CA            FCB    @12+PARCD   SFG
00316 7E5E 40            FCB    @100
00317 7E5F 53            FCB    EQU20/2
00318 7E60 C4            FCB    @4+PARCD    FORMAT
00319 7E61 BE            FCB    @276
00320 7E62 BE            FCB    EQU5/2
00321 7E63 8B            FCB    @13+PAREX   STEP
00322 7E64 527A          FDB    $527A
00323 7E66 8C            FCB    @14+PAREX   BACK STEP
00324 7E67 52CE          FDB    $52CE
00325 7E69 3D            FCB    TERMN7
00326 7E6A 7E6E          FDB    RUN1      "OR" FOR RUN MODE
00327 7E6C 7BCA          FDB    PRGM1     "OR" FOR PRGM MODE
00328 7E6E 90      RUN1  FCB    @20+PAREX   A
00329 7E6F 4AEF          FDB    UDFA
00330 7E71 91            FCB    @21+PAREX   B
00331 7E72 4AEE          FDB    UDFB
00332 7E74 92            FCB    @22+PAREX   C
00333 7E75 4AED          FDB    UDFC
00334 7E77 93            FCB    @23+PAREX   D
00335 7E78 4AEC          FDB    UDFD
00336 7E7A 94            FCB    @24+PAREX   E
00337 7E7B 4AEB          FDB    UDFE
00338 7E7D 98            FCB    @30+PAREX   F
00339 7E7E 4AEA          FDB    UDFF
00340 7E80 99            FCB    @31+PAREX   G
00341 7E81 4AE9          FDB    UDFG
00342 7E83 9A            FCB    @32+PAREX   H
00343 7E84 4AE8          FDB    UDFH
00344 7E86 9B            FCB    @33+PAREX   I
00345 7E87 4AE7          FDB    UDFI
00346 7E89 9C            FCB    @34+PAREX   J
00347 7E8A 4AE6          FDB    UDFJ
00348 7E8C A0            FCB    @40+PAREX   K
00349 7E8D 4AE5          FDB    UDFK
00350 7E8F A1            FCB    @41+PAREX   L
00351 7E90 4AE4          FDB    UDFL
00352 7E92 A2            FCB    @42+PAREX   M
00353 7E93 4AF3          FDB    UDFM
00354 7E95 A3            FCB    @43+PAREX   N
00355 7E96 4AE2          FDB    UDFN
00356 7E98 A4            FCB    @44+PAREX   O
00357 7E99 4AE1          FDB    UDFO
00358 7E9B 84            FCB    @4+PAREX
00359 7E9C 4AF0          FDB    FMT
00360 7E9E 3D            FCB    TERMN7
00361 7E9F 8C            FCB    12+$80    SYNTAX ERROR
00362                    ********************************************************
00363                    *****SECONDARY TABLES*********************************
00364                    ********************************************************
00365 7EA0 A3      ST22  FCB    @43+PAREX   ERASE
00366 7EA1 4B7F          FDB    ERASE
00367 7EA3 3D            FCB    TERMN7
00368 7EA4 7E00          FDB    MT
00369                    ********************************************************
00370 7EA6 CA      ST20  FCB    @12+PARCD   CFG
00371 7EA7 10            FCB    @20
00372      0146  EQU6  EQU    ST6-MT
00373 7EA8 A3            FCB    EQU6/2
00374 7EA9 3D            FCB    TERMN7
00375 7EAA 7F46          FDB    ST6
00376                    ********************************************************
00377 7EAC 66      ST14  FCB    @46+IMED    IF 0
00378 7EAD 33            FCB    @63
00379 7EAE 7F            FCB    @77+IMED    IF+
00380 7EAF 31            FCB    @61
00381 7EB0 6F            FCB    @57+IMED    IF-
00382 7EB1 32            FCB    @62
00383 7EB2 59            FCB    @31+IMED    IFX<Y
00384 7EB3 35            FCB    @65
00385 7EB4 5A            FCB    @32+IMED    IFX=Y
00386 7EB5 34            FCB    @64
00387 7EB6 5B            FCB    @33+IMED    IFX>=Y
00388 7EB7 36            FCB    @66
00389 7EB8 0A            FCB    @12+NTBL    IF SFG
00390 7EB9 53            FCB    EQU20/2
```

```
00391 7E8A 3D          FCB   TERMN7
00392 7E8B 8C          FCB   12+$80   SYNTAX ERROR
00393                  ********************************************************
00394 7E8C D5      ST1 FCB   @25+PARCD  RCL RCL
00395 7E8D 02          FCB   @2
00396 7E8E C0          FCB   EQU15/2
00397 7E8F 79          FCB   @71+IMED  RCL ACC+
00398 7EC0 38          FCB   @73
00399 7EC1 3D          FCB   TERMN7
00400 7EC2 7F80        FDB   TT1       (ALPHA TABLE)
00401                  ********************************************************
00402 7EC4 D5      ST2 FCB   @25+PARCD  STO RCL
00403 7EC5 0E          FCB   @16
00404 7EC6 C0          FCB   EQU15/2
00405 7EC7 3D          FCB   TERMN7
00406 7EC9 7F80        FDB   TT1
00407                  ********************************************************
00408 7ECA 30      ST3 FCB   @60+NTBL   F-1
00409 7ECB 65          FCB   EQU3/2
00410 7ECC 31          FCB   @61+NTBL   SHIFT
00411 7ECD 71          FCB   EQU4/2
00412 7ECE 6A          FCB   @52+IMED   ARCSIN
00413 7ECF 1C          FCB   @34
00414 7ED0 6B          FCB   @53+IMED   ARCCOS
00415 7ED1 1D          FCB   @35
00416 7ED2 7A          FCB   @72+IMED   ARCTAN
00417 7ED3 1E          FCB   @36
00418 7ED4 68          FCB   @50+IMED   EAX
00419 7ED5 21          FCB   @41
00420 7ED6 69          FCB   @51+IMED   10AX
00421 7ED7 22          FCB   @42
00422 7ED8 78          FCB   @70+IMED   SIGMA -
00423 7ED9 28          FCB   @50
00424 7EDA 7B          FCB   @73+IMED   R TO P
00425 7EDB 29          FCB   @51
00426 7EDC 79          FCB   @71+IMED   ACC-
00427 7EDD 2C          FCB   @54
00428 7EDE 3D          FCB   TERMN7
00429 7EDF 7E00        FDB   MT
00430 7EE1 00          FCB   @0
00431                  ********************************************************
00432 7EE2 30      ST4 FCB   @60+NTBL   F-1
00433 7EE3 65          FCB   EQU3/2
00434 7EE4 31          FCB   @61+NTBL   SHIFT
00435 7EE5 71          FCB   EQU4/2
00436 7EE6 40          FCB   @0+IMED    MARK TAPE
00437 7EE7 B7          FCB   @267
00438      0140  EQU21 EQU   ST21-MT
00439 7EE8 5E          FCB   @36+IMED   DEGREES
00440 7EE9 2E          FCB   @56
00441 7EEA 5F          FCB   @37+IMED   RADIANS
00442 7EEB 3F          FCB   @77
00443 7EEC 77          FCB   @67+IMED   GRADS
00444 7EED 3E          FCB   @76
00445 7EEE C6          FCB   @6+PARCD   FIX MODE
00446 7EEF B8          FCB   @273
00447 7EF0 A0          FCB   EQU21/2
00448 7EF1 C7          FCB   @7+PARCD   SCIENTIFIC MODE
00449 7EF2 8C          FCB   @274
00450 7EF3 A0          FCB   EQU21/2
00451 7EF4 F4          FCB   @64+PARCD   SCIENTIFIC 3 MODE
00452 7EF5 BD          FCB   @275
00453 7EF6 A0          FCB   EQU21/2
00454 7EF7 84          FCB   @4+PAREX   TO ALPHA
00455 7EF8 519C        FDB   $519C
00456 7EFA 68          FCB   @50+IMED   ROUND
00457 7EFB 3A          FCB   @72
00458 7EFC 41          FCB   @1+IMED    VERIFY
00459 7EFD A5          FCB   @245
00460 7EFE 42          FCB   @2+IMED    RC DATA
00461 7EFF A7          FCB   @247
00462 7F00 43          FCB   @3+IMED    IDENTIFY
00463 7F01 B8          FCB   @270
00464 7F02 6A          FCB   @52+IMED   YAX
00465 7F03 23          FCB   @43
00466 7F04 69          FCB   @51+IMED   SQRT X
```

```
00467 7F05 25          FCB    @45
00468 7F06 68          FCB    @53+IMED    1/X
00469 7F07 26          FCB    @46
00470 7F08 7A          FCB    @72+IMED    INT
00471 7F09 20          FCB    @55
00472 7F0A 78          FCB    @70+IMED    MEAN, STD DEV
00473 7F0B 37          FCB    @67
00474 7F0C 78          FCB    @73+IMED    TO D.MS
00475 7F0D 2F          FCB    @57
00476 7F0E 79          FCB    @71+IMED    FROM D.MS
00477 7F0F 30          FCB    @60
00478 7F10 7E          FCB    @76+IMED    LST X
00479 7F11 18          FCB    @30
00480 7F12 67          FCB    @47+IMED    PI
00481 7F13 17          FCB    @27
00482 7F14 40          FCB    @15+IMED    # OF R'S
00483 7F15 38          FCB    @70
00484 7F16 72          FCB    @62+IMED    LIST STACK
00485 7F17 39          FCB    @71
00486 7F18 3D          FCB    TERMN7
00487 7F19 7E00        FDB    MT
00488 7F1B 00          FCB    @0
00489                  ********************************************************************
00490 7F1C 9E     ST5  FCB    @36+PAREX   1
00491 7F1D 4857        FDB    D1
00492 7F1F 9F          FCB    @37+PAREX   2
00493 7F20 4856        FDB    D2
00494 7F22 B7          FCB    @67+PAREX   3
00495 7F23 4855        FDB    D3
00496 7F25 96          FCB    @26+PAREX   4
00497 7F26 4854        FDB    D4
00498 7F28 97          FCB    @27+PAREX   5
00499 7F29 4853        FDB    D5
00500 7F2B B6          FCB    @66+PAREX   6
00501 7F2C 4852        FDB    D6
00502 7F2E 8E          FCB    @16+PAREX   7
00503 7F2F 4851        FDB    D7
00504 7F31 42          FCB    @2+IMED   RECORD SECURED
00505 7F32 B6          FCB    @266
00506 7F33 50          FCB    @20+IMED    LD BIN
00507 7F34 A0          FCB    @240
00508 7F35 51          FCB    @21+IMED    RUN BIN
00509 7F36 A1          FCB    @241
00510 7F37 52          FCB    @22+IMED    RC BIN
00511 7F38 A2          FCB    @242
00512 7F39 53          FCB    @23+IMED    RC BIN SEC
00513 7F3A A3          FCB    @243
00514 7F3B 54          FCB    @24+IMED    PTAPE
00515 7F3C A4          FCB    @244
00516 7F3D 3D          FCB    TERMN7
00517 7F3E 7E6E        FDB    RUN1
00518                  ********************************************************************
00519 7F40 A6     ST21 FCB    @46+PAREX   0
00520 7F41 486F        FDB    F0
00521 7F43 B5          FCB    @65+PAREX   9
00522 7F44 4866        FDB    F9
00523 7F46 9E     ST6  FCB    @36+PAREX   1
00524 7F47 486E        FDB    F1
00525 7F49 9F          FCB    @37+PAREX   2
00526 7F4A 486D        FDB    F2
00527 7F4C B7          FCB    @67+PAREX   3
00528 7F4D 486C        FDB    F3
00529 7F4F 96          FCB    @26+PAREX   4
00530 7F50 486B        FDB    F4
00531 7F52 97          FCB    @27+PAREX   5
00532 7F53 486A        FDB    F5
00533 7F55 B6          FCB    @66+PAREX   6
00534 7F56 4869        FDB    F6
00535 7F58 8E          FCB    @16+PAREX   7
00536 7F59 4868        FDB    F7
00537 7F5B 8F          FCB    @17+PAREX   8
00538 7F5C 4867        FDB    F8
00539 7F5E 3D          FCB    TERMN7
00540 7F5F 8C          FCB    12+$80    SYNTAX ERROR
```

```
00541                   ************************************************************
00542 7F60 C9     ST7   FCB   @11+PARCD   GOTO LABEL
00543 7F61 D1           FCB   @321
00544      01C2 .EQU17  EQU   TT3-MT
00545 7F62 E1           FCB   EQU17/2
00546 7F63 6E           FCB   @56+IMED   GOTO X
00547 7F64 62           FCB   @142
00548 7F65 3D           FCB   TERMN7
00549 7F66 7F80         FDB   TT1       (ALPHA TABLE)
00550                   ************************************************************
00551 7F68 C9     ST11  FCB   @11+PARCD   GOSUB LABEL
00552 7F69 F0           FCB   @340
00553      01C8 EQU18   EQU   TT4-MT
00554 7F6A E4           FCB   EQU18/2
00555 7F6B 6E           FCB   @56+IMED   GOSUB X
00556 7F6C 63           FCB   @143
00557 7F6D 3D           FCB   TERMN7
00558 7F6E 7F80         FDB   TT1       (ALPHA TABLE)
00559                   ************************************************************
00560 7F70 50     ST12  FCB   @20+IMED   FOR A
00561 7F71 AC           FCB   @254
00562 7F72 51           FCB   @21+IMED   FOR B
00563 7F73 AA           FCB   @252
00564 7F74 52           FCB   @22+IMED   FOR C
00565 7F75 AB           FCB   @253
00566 7F76 3D           FCB   TERMN7
00567 7F77 8C           FCB   12+$80    SYNTAX ERROR
00568                   ************************************************************
00569 7F78 50     ST13  FCB   @20+IMED   NEXT A
00570 7F79 AD           FCB   @255
00571 7F7A 51           FCB   @21+IMED   NEXT B
00572 7F7B AE           FCB   @256
00573 7F7C 52           FCB   @22+IMED   NEXT C
00574 7F7D AF           FCB   @257
00575 7F7E 3D           FCB   TERMN7
00576 7F7F 8C           FCB   12+$80    SYNTAX ERROR
00577                   ************************************************************
00578                   ***** THIRD LEVEL TABLES ************************************
00579                   ************************************************************
00580 7F80 90     TT1   FCB   @20+PAREX   A
00581 7F81 4AA7         FDB   UDFKA
00582 7F83 91           FCB   @21+PAREX   B
00583 7F84 4AA6         FDB   UDFKB
00584 7F86 92           FCB   @22+PAREX   C
00585 7F87 4AA5         FDB   UDFKC
00586 7F89 93           FCB   @23+PAREX   D
00587 7F8A 4AA4         FDB   UDFKD
00588 7F8C 94           FCB   @24+PAREX   E
00589 7F8D 4AA3         FDB   UDFKE
00590 7F8F 98           FCB   @30+PAREX   F
00591 7F90 4AA2         FDB   UDFKF
00592 7F92 99           FCB   @31+PAREX   G
00593 7F93 4AA1         FDB   UDFKG
00594 7F95 9A           FCB   @32+PAREX   H
00595 7F96 4AA0         FDB   UDFKH
00596 7F98 9B           FCB   @33+PAREX   I
00597 7F99 4A9F         FDB   UDFKI
00598 7F9B 9C           FCB   @34+PAREX   J
00599 7F9C 4A9E         FDB   UDFKJ
00600 7F9E A0           FCB   @40+PAREX   K
00601 7F9F 4A9D         FDB   UDFKK
00602 7FA1 A1           FCB   @41+PAREX   L
00603 7FA2 4A9C         FDB   UDFKL
00604 7FA4 A2           FCB   @42+PAREX   M
00605 7FA5 4A9B         FDB   UDFKM
00606 7FA7 A3           FCB   @43+PAREX   N
00607 7FA8 4A9A         FDB   UDFKN
00608 7FAA A4           FCB   @44+PAREX   O
00609 7FAB 4A99         FDB   UDFKO
00610 7FAD 3D           FCB   TERMN7
00611 7FAE 7FCE         FDB   TT5       (DIGIT TABLE)
00612                   ************************************************************
00613 7FB0 FF     TT2   FCB   @77+PARCD   STO [+] RCL
00614 7FB1 02           FCB   @2
00615 7FB2 62           FCB   EQU2/2
00616 7FB3 EF           FCB   @57+PARCD   STO [-] RCL
```

```
00617 7FB4 04            FCB    @4
00618 7FB5 62            FCB    EQU2/2
00619 7FB6 EE            FCB    @56+PARCD   STO [*] RCL
00620 7FB7 06            FCB    @6
00621 7FB8 62            FCB    EQU2/2
00622 7FB9 F3            FCB    @63+PARCD   STO [/] RCL
00623 7FBA 08            FCB    @10
00624 7FBB 62            FCB    EQU2/2
00625 7FBC 45            FCB    @5+IMED   STO CLEAR (CLEAR ALPHA REGS)
00626 7FBD 3C            FCB    @74
00627 7FBE 3D            FCB    TERMN7
00628 7FBF 7EC4          FDB    ST2
00629 7FC1 00            FCB    @0
00630                    ********************************************************
00631 7FC2 6E       TT3  FCB    @56+IMED   GOTO LBL X
00632 7FC3 60            FCB    @140
00633 7FC4 3D            FCB    TERMN7
00634 7FC5 7F80          FDB    TT1       (ALPHA TABLE)
00635 7FC7 00            FCB    @0
00636                    ********************************************************
00637 7FC8 6E       TT4  FCB    @56+IMED   GOSUB LBL X
00638 7FC9 61            FCB    @141
00639 7FCA 3D            FCB    TERMN7
00640 7FCB 7F80          FDB    TT1       (ALPHA TABLE)
00641 7FCD 00            FCB    @0
00642                    ********************************************************
00643 7FCE A6       TT5  FCB    @46+PAREX   0
00644 7FCF 49F0          FDB    ZERO
00645 7FD1 9E            FCB    @36+PAREX   1
00646 7FD2 49EF          FDB    ONE
00647 7FD4 9F            FCB    @37+PAREX   2
00648 7FD5 49EE          FDB    TWO
00649 7FD7 B7            FCB    @67+PAREX   3
00650 7FD8 49ED          FDB    THREE
00651 7FDA 96            FCB    @26+PAREX   4
00652 7FDB 49EC          FDB    FOUR
00653 7FDD 97            FCB    @27+PAREX   5
00654 7FDE 49E8          FDB    FIVE
00655 7FE0 B6            FCB    @66+PAREX   6
00656 7FE1 49EA          FDB    SIX
00657 7FE3 8E            FCB    @16+PAREX   7
00658 7FE4 49E9          FDB    SEVEN
00659 7FE6 8F            FCB    @17+PAREX   8
00660 7FE7 49E8          FDB    EIGHT
00661 7FE9 B5            FCB    @65+PAREX   9
00662 7FEA 49E7          FDB    NINE
00663 7FEC A7            FCB    @47+PAREX   .
00664 7FED 4A22          FDB    DECPT
00665 7FEF 3D            FCB    TERMN7
00666 7FF0 80            FCB    $80       ERROR ZERO (SPECIAL CASE)
00669                    NAM    PLABR
00670                    OPT    LIST,MEM
00671 49E7               ORG    $49E7
00672                    *
00673                    *PROGRAM LANGUAGE ADDRESS BUILDING ROUTINE
00674                    *
00675 49E7 4C       NINE INC A              AFTER ENTERING AND
00676 49E8 4C       EIGHT INC A             FALLING THRU THIS LIST
00677 49E9 4C       SEVEN INC A             OF INCREMENT INSTS,
00678 49EA 4C       SIX  INC A              ACCA WILL CONTAIN
00679 49EB 4C       FIVE INC A              THE PROPER BCD DIGIT
00680 49EC 4C       FOUR INC A              ENTERED BY THE USER
00681 49ED 4C       THREE INC A
00682 49EE 4C       TWO  INC A
00683 49EF 4C       ONE  INC A
00684 49F0 7C 0013  ZERO INC    DCNTR       ADD 1 TO DIGIT COUNTER
00685 49F3 C6 04         LDA B  #4          SET ACCB FOR ROTATE
00686 49F5 0C       LL   CLC               CLEAR CARRY
00687 49F6 79 0011       ROL    W1          ROLL LEFT W1 (LSW)
00688 49F9 79 0010       ROL    W2          ROLL W2 WITH CARRY
00689 49FC 5A            DEC B              DECREMENT COUNTER
00690 49FD 26 F6         BNE    LL          CONTINUE IF NOT 0
00691 49FF 9A 11         ORA A  W1          "OR" NEW BCD DIGIT
00692 4A01 97 11         STA A  W1          RESTORE W1
00693 4A03 96 13         LDA A  DCNTR       GET DIGIT COUNTER
00694 4A05 D6 C6         LDA B  SOL7        GET THE INSTRUCTION
```

```
00695 4A07 C1 DF          CMP B  #@337     FOUR DIGIT ADDRESS?
00696 4A09 22 08          BHI    F4D       BRANCH IF YES
00697 4A0B C1 C7          CMP B  #@307     THREE DIGIT ADDRESS?
00698 4A0D 22 0D          BHI    T3D       BRANCH IF YES
00699 4A0F 81 02          CMP A  #2        ELSE 2 DIGIT
00700 4A11 27 0F          BEQ    DECPT     BRANCH IF 2 ENTERED
00701 4A13 7E 78E3 BLK    JMP    NK7       ELSE GET NEXT KEY
00702 4A16 81 04   F4D    CMP A  #4        FOUR DIGITS ENTERED?
00703 4A18 27 08          BEQ    DECPT     BRANCH IF YES
00704 4A1A 20 F7          BRA    BLK       ELSE GET NEXT KEY
00705 4A1C 81 03   T3D    CMP A  #3        THREE DIGITS ENTERED?
00706 4A1E 27 02          BEQ    DECPT     BRANCH IF YES
00707 4A20 20 F1          BRA    BLK       ELSE GET NEXT KEY
00708                     *
00709                     *THIS HANDLES THE DEC PT TERMINATOR
00710                     *
00711 4A22 8D 02   DECPT  BSR    BCDBIN    BCD TO BINARY CONV.
00712 4A24 20 42          BRA    TYPE      CHECK INST TYPE
00713                     *
00714                     *THE FOLLOWING ROUTINE DOES A BCD TO BINARY
00715                     *CONVERSION ON W1 & W2.
00716                     *
00717                     *
00718                     *BCD TO BINARY CONVERSION ON W1 AND W2 WITH
00719                     *BINARY RESULT LEFT IN TP7 AND TP7S. RESULT IS
00720                     *EQUAL TO L+64H+32H+4H WHERE L=BINARY VALUE
00721                     *OF THE LOW 2 BCD DIGITS AND H=BINARY VALUE
00722                     *OF THE HIGH 2 BCD DIGITS.
00723                     *
00724 4A26 D6 10   BCDBIN LDA B  W2        GET MS BCD DIGITS
00725 4A28 8D 33          BSR    BCDB      CONVERT TO BINARY
00726 4A2A 5F             CLR B            PRESET ACCB FOR MULTIPLIES
00727 4A2B 48             ASL A            MULTIPLY RESULT BY 4
00728 4A2C 59             ROL B
00729 4A2D 48             ASL A
00730 4A2E 59             ROL B
00731 4A2F D7 20          STA B  TP7       SAVE 4*(HIGH BINARY)
00732 4A31 97 21          STA A  TP7S
00733 4A33 48             ASL A            MULTIPLY BY 32
00734 4A34 59             ROL B
00735 4A35 48             ASL A
00736 4A36 59             ROL B
00737 4A37 48             ASL A
00738 4A38 59             ROL B
00739 4A39 D7 1E          STA B  TP6       SAVE 32*(HIGH BINARY)
00740 4A3B 97 1F          STA A  TP6S
00741 4A3D 48             ASL A            MULTIPLY BY 64
00742 4A3E 59             ROL B
00743 4A3F 9B 1F          ADD A  TP6S      DO 64H +32H
00744 4A41 97 1F          STA A  TP6S
00745 4A43 D9 1E          ADC B  TP6
00746 4A45 D7 1E          STA B  TP6
00747 4A47 BD 484B        JSR    ADD7      DO 64H + 32H + 4H
00748 4A4A D6 11          LDA B  W1        GET LOW 2 BCD DIGITS
00749 4A4C 8D 0F          BSR    BCDB      CONVERT TO BINARY
00750 4A4E 9B 21          ADD A  TP7S      DO L + 64H + 32H + 4H
00751 4A50 97 21          STA A  TP7S
00752 4A52 24 03          BCC    BCD1      BRANCH IF NO CARRY
00753 4A54 7C 0020        INC    TP7       ELSE BUMP TP7 BY ONE
00754 4A57 CE 0000 BCD1   LDX    #0        CLEAR THE INDEX
00755 4A5A DF 10          STX    W2        CLEAR THE BCD LOCATIONS
00756 4A5C 39             RTS              RETURN WITH BINARY IN TP7 AND TP7+1
00757                     *
00758                     *THIS ROUTINE DOES A BCD TO BINARY CONVERSION
00759                     *ON ACCB WITH RETURNED BINARY IN ACCA.
00760                     *
00761 4A5D 17      BCDB   TBA              ACCA = THE BCD DIGITS
00762 4A5E 84 0F          AND A  #$F       ACCA = THE LS DIGIT
00763 4A60 C4 F0          AND B  #$F0      ACCB = THE MS DIGIT
00764 4A62 54             LSR B           BINARY = MS DIGIT * 10 + LS DIGIT
00765 4A63 1B             ABA
00766 4A64 54             LSR B
00767 4A65 54             LSR B
00768 4A66 1B             ABA
00769 4A67 39             RTS             ACCA = THE BINARY
00770                     *
```

```
00771                *THIS ROUTINE CHECKS INST TYPE, CHECKS
00772                *FOR OVERFLOW FOR THAT TYPE, AND INSERTS
00773                *ADDRESS OR LABEL INFO IF VALID
00774                *
00775 4A68 86 E0    TYPE   LDA A  #@340     >=340=JMP OR JSR
00776 4A6A 91 C6           CMP A  SOL7      COMPARE TO INST
00777 4A6C 23 0C           BLS    GRP1      BRANCH IF JMP OR JSR
00778 4A6E 86 C8           LDA A  #@310     DATA STORAGE INST?
00779 4A70 91 C6           CMP A  SOL7      COMPARE TO INST
00780 4A72 23 1A           BLS    GRP2      BRANCH IF YES
00781 4A74 96 21           LDA A  TP7+1     GET LS BINARY
00782 4A76 97 C7           STA A  SOL7+1    OK: SAVE DATA
00783 4A78 20 11           BRA    CCNT      CONTINUE IN SUPV
00784 4A7A 96 20    GRP1   LDA A  TP7       GET MS BINARY
00785 4A7C 85 F8           BIT A  #@370     >2047?
00786 4A7E 26 12           BNE    AERR      ERROR IF YES
00787 4A80 D6 21    FINE1  LDA B  TP7+1     GET LS BINARY
00788 4A82 58              ASL B            CARRY=B7
00789 4A83 49              ROL A            LSB OF MSW=B7
00790 4A84 54              LSR B            RESTORE LSW
00791 4A85 D7 C7           STA B  SOL7+1    SAVE IN 2ND BYTE
00792 4A87 9A C6           ORA A  SOL7      "OR" IN INST
00793 4A89 97 C6           STA A  SOL7      RESTORE WITH ADRS
00794 4A8B 7E 794C  CCNT   JMP    CONT      CONTINUE IN SUPV
00795 4A8E 96 20    GRP2   LDA A  TP7       GET MS BINARY
00796 4A90 27 EE           BEQ    FINE1     OK IF ZERO
00797 4A92 86 18    AERR   LDA A  #24       LOAD ERROR
00798 4A94 97 06           STA A  ERROR     SAVE IN ERROR WORD
00799 4A96 7E 7B40         JMP    ERROR7    JMP TO ERROR OUTPUT
00800                *
00801                *ALPHA TABLE PARTIAL EXECUTE CODE
00802                *
00803 4A99 4C       UDFKO  INC A
00804 4A9A 4C       UDFKN  INC A
00805 4A9B 4C       UDFKM  INC A
00806 4A9C 4C       UDFKL  INC A
00807 4A9D 4C       UDFKK  INC A
00808 4A9E 4C       UDFKJ  INC A
00809 4A9F 4C       UDFKI  INC A
00810 4AA0 4C       UDFKH  INC A             ACCA=7 IF
00811 4AA1 4C       UDFKG  INC A             "H" KEY AND
00812 4AA2 4C       UDFKF  INC A             ACCA=0 IF
00813 4AA3 4C       UDFKE  INC A             "A" KEY.
00814 4AA4 4C       UDFKD  INC A
00815 4AA5 4C       UDFKC  INC A
00816 4AA6 4C       UDFKB  INC A
00817 4AA7 D6 C6    UDFKA  LDA B  SOL7       GET THE INST
00818 4AA9 C1 D2           CMP B  #@322      <=322?
00819 4AAB 23 14           BLS    ONEBYT     YES: SINGLE BYTE
00820 4AAD C1 DF           CMP B  #@337      JMP OR JSR ?
00821 4AAF 22 24           BHI    HANK1      BRANCH IF YES
00822 4AB1 81 09           CMP A  #9         ALPHA KEY TOO BIG ?
00823 4AB3 2E DD           BGT    AERR       BRANCH IF YES
00824 4AB5 54              LSR B            INST=INST/2
00825 4AB6 CB 58           ADD B  #@130      INST=INST+130
00826 4AB8 D7 C6    HANK3  STA B  SOL7       RESTORE THE NEW INST.
00827 4ABA 8B 64    HANK   ADD A  #100       ADD KEY "NUMBER" TO LABEL OFFSET
00828 4ABC 97 C7           STA A  SOL7+1     SAVE IN 2ND BYTE
00829 4ABE 7E 794C  DOUG   JMP    CONT       CONTINUE IN SUPV
00830 4AC1 C1 C1    ONEBYT CMP B  #@301      TWO BYTE INST?
00831 4AC3 23 F5           BLS    HANK       BRANCH IF YES
00832 4AC5 81 09           CMP A  #9         ALPHA KEY TOO BIG ?
00833 4AC7 2E C9           BGT    AERR       BRANCH IF YES
00834 4AC9 C0 B4           SUB B  #@264      NEW INST =
00835 4ACB 37              PSH B            OLD INST - 264
00836 4ACC 58              ASL B            MULT BY 5
00837 4ACD 58              ASL B            + ACCA
00838 4ACE 1B              ABA
00839 4ACF 33              PUL B
00840 4AD0 1B              ABA              PLUS "KEY #"
00841 4AD1 97 C6           STA A  SOL7      RESTORE NEW INST
00842 4AD3 20 E9           BRA    DOUG       CONT. IN SUPV
00843 4AD5 C1 EF    HANK1  CMP B  #@357      JMP INST ?
00844 4AD7 22 04           BHI    HANK2      BRANCH IF YES
00845 4AD9 C6 C0           LDA B  #@300      LOAD JSR LBL
00846 4ADB 20 DB           BRA    HANK3      GO SAVE IT
```

```
00847 4ADD C6 C1    HANK2  LDA B  #@301     LOAD JMP LBL
00848 4ADF 20 D7           BRA    HANK3     CONTINUE
00849      484B    ADD7   EQU    $484B
00850      7B40    ERROR7 EQU    $7B40
00851      78E3    NK7    EQU    $78E3
00852      794C    CONT   EQU    $794C
00853      797B    BWM    EQU    $797B
00856             NAM    UDF
00857             OPT    LIST,MEM
00858             *
00859             *THIS ROUTINE HANDLES EXECUTION OF THE USER DEFINABLE FUNCTI
00860             *
00861 4AE1 4C      UDF0   INC A
00862 4AE2 4C      UDFN   INC A
00863 4AE3 4C      UDFM   INC A
00864 4AE4 4C      UDFL   INC A
00865 4AE5 4C      UDFK   INC A
00866 4AE6 4C      UDFJ   INC A
00867 4AE7 4C      UDFI   INC A
00868 4AE8 4C      UDFH   INC A
00869 4AE9 4C      UDFG   INC A
00870 4AEA 4C      UDFF   INC A
00871 4AEB 4C      UDFE   INC A
00872 4AEC 4C      UDFD   INC A
00873 4AED 4C      UDFC   INC A
00874 4AEE 4C      UDFB   INC A
00875 4AEF 4C      UDFA   INC A
00876 4AF0 D6 C6   FMT    LDA B  SOL7      GET INST
00877 4AF2 26 47          BNE    FMT0      BRANCH IF "CALL" INSTRUCTION
00878 4AF4 97 2E          STA A  T1        SAVE ALPHA KEY
00879 4AF6 96 07          LDA A  TGL       GET SWITCH SETTING
00880 4AF8 85 20          BIT A  #$20      AUDIT TRAIL ?
00881 4AFA 27 07          BEQ    UDF0      BRANCH IF NOT
00882 4AFC BD 5C8C        JSR    AUDIT1    ELSE PRINT
00883 4AFF D6 06          LDA B  ERROR     PAPER OUT ERROR ?
00884 4B01 26 0D          BNE    UDF3      BRANCH IF YES
00885 4B03 96 2E   UDF0   LDA A  T1        RESTORE ALPHA KEY
00886 4B05 8B 63          ADD A  #99       ADD LABEL OFFSET
00887 4B07 BD 497D        JSR    LBLSCH    FIND THE LABEL
00888 4B0A 2A 07          BPL    UDF1      BRANCH IF FOUND
00889 4B0C 86 0B          LDA A  #11       LOAD ERROR
00890 4B0E 97 06          STA A  ERROR     SAVE IT
00891 4B10 7E 7B40 UDF3   JMP    ERROR7    OUTPUT IT
00892 4B13 DF CA   UDF1   STX    UIP       SAVE ROUTINE ADDRESS
00893 4B15 E1 01          CMP B  1,X       CHECK TO SEE IF
00894 4B17 26 04          BNE    UDF4      TWO IDENTICAL LABELS
00895 4B19 A1 02          CMP A  2,X       ARE CONSECUTIVE AND
00896 4B1B 27 12          BEQ    UDF5      IF SO, DON'T STACK THE RETURN
00897 4B1D 30      UDF4   TSX               GET THE STACK POINTER
00898 4B1E 8C 0044        CPX    #@104     SUBROUTINE OVERFLOW ?
00899 4B21 26 04          BNE    UDF2      BRANCH IF NOT
00900 4B23 86 09          LDA A  #9        LOAD ERROR
00901 4B25 20 E7          BRA    UDF3-2    OUTPUT IT
00902 4B27 96 C9   UDF2   LDA A  UPP+1     GET PRGM PNTR LO
00903 4B29 36             PSH A            SAVE IT
00904 4B2A 96 C8          LDA A  UPP       GET PRGM PNTR HI
00905 4B2C 8A C0          ORA A  #$C0      ADD UDF FLAG
00906 4B2E 36             PSH A            SAVE IT
00907 4B2F 86 FF   UDF5   LDA A  #$FF      ACCA=ALL ONES
00908 4B31 97 09          STA A  RSFLG     SET R/S FLAG
00909 4B33 4F             CLR A
00910 4B34 97 12          STA A  SFLG      CLEAR THE STEP FLAG
00911 4B36 97 B9          STA A  BKWRT+1   EMPTY THE KEY BUFFER
00912 4B38 7E 797B        JMP    BWM       CONTINUE
00913             *
00914             *"FORMAT STATEMENT" BUILDING
00915             *
00916 4B3B 16      FMT0   TAB               SAVE "ALPHA" NUMBER
00917 4B3C DA C7          ORA B  SOL7+1     ADD ALPHA KEY TO INST
00918 4B3E C4 7F          AND B  #$7F       INSURE MSB=0
00919 4B40 D7 C7          STA B  SOL7+1     RESTORE INSTRUCTION
00920 4B42 40             TST A            ALPHA OUTPUT INSTRUCTION ?
00921 4B43 27 09          BEQ    ALP       BRANCH IF YES
00922 4B45 86 0C          LDA A  #12       LOAD SYNTAX ERROR
00923 4B47 C5 F0          BIT B  #$F0      MAINFRAME CALL INSTRUCTION ?
00924 4B49 27 C3          BEQ    UDF3-2    ERROR IF YES
```

```
00925 484B 7E 794C COV1   JMP    CONT      CONTINUE IN SUPV
00926 484E 7E 51A0 ALP    JMP    ALPHAK    GO TO KEYBOARD ALPHA
00927      51A0    ALPHAK EQU    $51A0
00928      497D    LBLSCH EQU    $497D
00929      5C8C    AUDIT1 EQU    $5C8C
00932             NAM    DTBL
00933             OPT    LIST,MEM
00934             *
00935             *DIGIT ACCEPTANCE TABLE FOR "FLAG" AND "CALL" INSTRUCTIONS.
00936             *
00937             *THE FOLLOWING TABLE ACCEPTS I/O DEVICE NUMBERS AND
00938             *PACKS THEM INTO THE INSTRUCTION BEING BUILT.
00939             *
00940 4851 4C      D7     INC A
00941 4852 4C      D6     INC A
00942 4853 4C      D5     INC A
00943 4854 4C      D4     INC A
00944 4855 4C      D3     INC A
00945 4856 4C      D2     INC A
00946 4857 4C      D1     INC A
00947 4858 48      D0     ASL A             MOVE THE DEVICE CALL NUMBER
00948 4859 48             ASL A             INTO THE MS FOUR BITS
00949 485A 48             ASL A             OF ACCA
00950 485B 48             ASL A
00951 485C 97 C7          STA A  SOL7+1     SAVE IT IN THE INSTRUCTION
00952 485E CE 7E6E        LDX    #RUN1      GET NEW SYNTAX TABLE ADRS
00953 4861 DF D3          STX    IT7        SAVE IT FOR NEXT SEARCH
00954 4863 7E 78E3        JMP    NK7        GO GET THE NEXT KEY
00955             *
00956             *THE FOLLOWING TABLE ACCEPTS FLAG NUMBER INFORMATION
00957             *AND PACKS IT INTO THE FLAG INSTRUCTION BEING BUILT.
00958             *
00959 4866 4C      F9     INC A
00960 4867 4C      F8     INC A
00961 4868 4C      F7     INC A
00962 4869 4C      F6     INC A
00963 486A 4C      F5     INC A
00964 486B 4C      F4     INC A
00965 486C 4C      F3     INC A
00966 486D 4C      F2     INC A
00967 486E 4C      F1     INC A
00968 486F D6 C6   F0     LDA B  SOL7       GET CURRENT INSTRUCTION
00969 4871 2B 08          BMI    RNDD       BRANCH IF NOT A FLAG INST
00970 4873 4A             DEC A             FLAG 1 = "FLAG 0"
00971 4874 9A C6          ORA A  SOL7       GENERATE THE NEW INST.
00972 4876 97 C6          STA A  SOL7       SAVE IT
00973 4878 7E 794C RNDD1  JMP    CONT       GO DO IT !
00974 487B 97 C7   RNDD   STA A  SOL7+1     SAVE THE ROUND SETTING
00975 487D 20 F9          BRA    RNDD1      GO DO IT !
00976             *
00977             *ERASE PROGRAM MEMORY ROUTINE.
00978             *
00979 487F CE 0100 ERASE  LDX    #256       INDEX=USER STATE ZERO
00980 4882 6F 00   ERA1   CLR    X          NOP TO CURRENT STATE
00981 4884 08             INX               BUMP POINTER
00982 4885 9C 08          CPX    EOPM       END OF PROGRAM MEMORY ?
00983 4887 26 F9          BNE    ERA1       BRANCH IF NOT
00984 4889 7E 482A        JMP    $482A      GO TO SPECIAL "END" STMT
00987             END
```

SYMBOL TABLE

ADATA	0000	ACTL	0001	BDATA	0002	BCTL	0003	INPUT	0004	IOIN	0005
EPROR	0006	TGL	0007	UFLG	0008	RSFLG	0009	EOM	000A	EOPM	0008
STKFLG	000D	RND	000E	DIGFLG	000F	W2	0010	W1	0011	SFLG	0012
DCNTR	0013	TP1	0014	TP1S	0015	TP2	0016	TP2S	0017	TP3	0018
TP3S	0019	TP4	001A	TP4S	001B	TP5	001C	TP5S	001D	TP6	001E
TP6S	001F	TP7	0020	TP7S	0021	T13	0022	T12	0023	T11	0024
T10	0025	T9	0026	T8	0027	T7	0028	T6	0029	T5	002A
T4	002B	T3	002C	T2	002D	T1	002E	ISTK	002F	ISTACK	0051
TA	0052	SPGM	0054	EXTRA	0056	BUFF	0058	REAL	0068	IMAG	0070
AT1	0078	AT2	0080	W	0088	XR	0090	YR	0098	ZR	00A0
TR	00A8	LSTX	00B0	BKWRT	00B8	BKKC	00BA	SOL7	00C6	UPP	00C8
UIP	00CA	ALPHA	00CC	IO1	00CD	IO2	00D0	IT7	00D3	FLAG	00D5

```
TPOS    00D6  FILE    00D7  AR      00D8  BR      00E0  CR      00E8  DR      00F0
ER      00F8  MT      7E00  TERMN7  003D  IMED    0040  PARCO   00C0  PAREX   0080
NTBL    0000  PRGM1   78CA  EQU11   0168  EQU12   0170  EQU13   0178  EQU14   00AC
EQU22   00A0  EQU1    00BC  EQU2    00C4  EQU16   0180  EQU3    00CA  EQU4    00E2
EQU5    011C  EQU20   00A6  EQU7    0160  EQU15   0180  RUN1    7E6E  ST22    7EA0
ST20    7EA6  EQU6    0146  ST14    7EAC  ST1     7EBC  ST2     7EC4  ST3     7ECA
ST4     7EE2  EQU21   0140  ST5     7F1C  ST21    7F40  ST6     7F46  ST7     7F50
EQU17   01C2  ST11    7F68  EQU18   01C8  ST12    7F70  ST13    7F78  TT1     7F80
TT2     7F80  TT3     7FC2  TT4     7FC8  TT5     7FCE  NINE    49E7  EIGHT   49E8
SEVEN   49E9  SIX     49EA  FIVE    49EB  FOUR    49EC  THREE   49ED  TWO     49EE
ONE     49EF  ZERO    49F0  LL      49F5  BLK     4A13  F4D     4A16  T3D     4A1C
DECPT   4A22  BCDBIN  4A26  BCD1    4A57  BCDB    4A5D  TYPE    4A68  GRP1    4A7A
FINE1   4A80  CCNT    4A88  GRP2    4A8E  AERR    4A92  UDFK0   4A99  UDFKN   4A9A
UDFKM   4A9B  UDFKL   4A9C  UDFKK   4A9D  UDFKJ   4A9E  UDFKI   4A9F  UDFKH   4AA0
UDFKG   4AA1  UDFKF   4AA2  UDFKE   4AA3  UDFKD   4AA4  UDFKC   4AA5  UDFKB   4AA6
UDFKA   4AA7  HANK3   4AB8  HANK    4ABA  DOUG    4ABE  ONEBYT  4AC1  HANK1   4AD5
HANK2   4ADD  ADD7    4B4B  ERROR7  7B40  NK7     78E3  CONT    794C  BWM     797B
UDF0    4AE1  UDFN    4AE2  UDFM    4AE3  UDFL    4AE4  UDFK    4AE5  UDFJ    4AE6
UDFI    4AE7  UDFH    4AE8  UDFG    4AE9  UDFF    4AEA  UDFE    4AEB  UDFD    4AEC
UDFC    4AED  UDFB    4AEE  UDFA    4AEF  FMT     4AF0  UDF0    4B03  UDF3    4B10
UDF1    4B13  UDF4    4B1D  UDF2    4B27  UDF5    4B2F  FMT0    4B38  COV1    4B48
ALP     4B4E  ALPHAK  51A0  LBLSCH  497D  AUDIT1  5C8C  D7      4B51  D6      4B52
D5      4B53  D4      4B54  D3      4B55  D2      4B56  D1      4B57  D0      4B58
F9      4B66  F8      4B67  F7      4B68  F6      4B69  F5      4B6A  F4      4B6B
F3      4B6C  F2      4B6D  F1      4B6E  F0      4B6F  RNDD1   4B78  RNDD    4B7B
ERASE   4B7F  ERA1    4B82

TOTAL ERRORS  22


****ERROR 201
 218  NAM FORMAT
****ERROR 201
 327  NAM JSRJMP
****ERROR 201
 528  NAM ENDEX
****ERROR 201
 550  NAM RTNEX

00215                        OPT    LIST,MEM
00218                        NAM    FORMAT
00219                        OPT    LIST,MEM
00220 4E09                   ORG    $4E09
00221                *
00222                *THIS IS THE DISPLAY FORMATTER FOR CJ.
00223                *IF IN RUN MODE, THE X REG IS DISPLAYED.
00224                *IF PRGM MODE IS SET, THE PRGM CNTR
00225                *AND DESTINATION ADDRESS ARE DISPLAYED.
00226                *
00227 4E09 96 07    FRMT1  LDA A  TGL      WHAT MODE ?
00228 4E0B 2B 11           BMI    PMODE    BRANCH IF PRGM MODE
00229 4E0D 96 CC           LDA A  ALPHA    ALPHA FLAG SET ?
00230 4E0F 27 05           BEQ    DONTT    BRANCH IF NOT
00231 4E11 BD 5D75         JSR    BLANK    ELSE BLANK BUFFER
00232 4E14 20 54           BRA    FC3      DISPLAY "ALPHA"
00233 4E16 7E 5CBC  DONTT  JMP    FRMT+$14   ELSE USE RUN FORMATTER
00234 4E19 CE 0100  FC4    LDX    #256     INDEX = USER STATE 0
00235 4E1C DF C8           STX    UPP      UPDATE INCORRECT PRGM PNTR
00236 4E1E 96 09    PMODE  LDA A  RSFLG    RUNNING ?
00237 4E20 2B F4           BMI    DONTT    IF YES, DON'T FORMAT HERE
00238 4E22 BD 5D75         JSR    BLANK    ELSE BLANK BUFFER
00239 4E25 7A 00C8         DEC    UPP      REMOVE SYSTEM ADRS OFFSET
00240 4E28 DE C8           LDX    UPP      GET PRGM PNTR
00241 4E2A 7C 00C8         INC    UPP      RESTORE SYSTEM OFFSET
00242 4E2D BD 488C         JSR    BINBCD   CONVERT TO BCD
00243 4F30 CE 0058         LDX    #BUFF    GET BUFFER ADRS
00244 4E33 86 3A           LDA A  #$3A     GET THE "P"
00245 4E35 C6 03           LDA B  #3       SET CHAR. COUNT
00246 4E37 8D 38           BSR    LOAD     LOAD CHARS.
00247 4E39 86 10           LDA A  #16      SET DIGITS CNTR
00248 4E3B 8D 67           BSR    DGTS     GO LOAD THE DIGITS
00249 4E3D 96 52           LDA A  TA       INSERT MODE ?
00250 4E3F 27 06           BEQ    FC2      BRANCH IF NOT
00251 4E41 08              INX             PISITION BUFFER PNTR
00252 4E42 08              INX
```

```
00253 4E43 86 4C          LDA A  #$4C      GET "INSERT" CHAR.
00254 4E45 A7 00          STA A  X         SAVE IT
00255 4E47 96 C6    FC2   LDA A  SOL7      GET CURRENT INST
00256 4E49 81 BE          CMP A  #@276     LIGHT DISPLAY ?
00257 4E4B 22 2C          BHI    FC1       BRANCH IF YES
00258 4E4D 96 CC          LDA A  ALPHA
00259 4E4F 26 19          BNE    FC3
00260 4E51 DE 08          LDX    EOPM
00261 4E53 DF 20          STX    TP7
00262 4E55 DE C8          LDX    UPP
00263 4E57 DF 1E          STX    TP6
00264 4E59 BD 4840        JSR    SUB7
00265 4E5C DE 20          LDX    TP7
00266 4E5E 2F B9          BLE    FC4       BRANCH IF UPP IS IN DATA REGS
00267 4E60 BD 48BC        JSR    BINBCD
00268 4E63 CE 0064        LDX    #BUFF+12
00269 4E66 86 10          LDA A  #16
00270 4E68 20 3A          BRA    DGTS      LOAD DIGITS AND RETURN
00271 4E6A CE 0063  FC3   LDX    #BUFF+11
00272 4E6D 86 47          LDA A  #$47      GET "ALPHA" MESSAGE
00273 4E6F C6 05          LDA B  #5        GET CHAR. COUNT
00274 4E71 A7 00    LOAD  STA A  X         SAVE THE CHAR.
00275 4E73 4C             INC A            GET NEXT CHARACTER
00276 4E74 08             INX              POSITION THE BUFFER POINTER
00277 4E75 5A             DEC B            DEC. THE CHARACTER COUNT
00278 4E76 26 F9          BNE    LOAD      CONTINUE IF NOT FINISHED YET
00279 4E78 39       FC    RTS              ELSE RETURN
00280 4E79 DE 10    FC1   LDX    W2        GET BCD ADDRESS
00281 4E7B DF 20          STX    T2        MOVE TO TEMPS FOR LOAD
00282 4E7D CE 0064        LDX    #BUFF+12    SET BUFFER POINTER
00283 4E80 86 10          LDA A  #16       SET DIGITS COUNTER
00284 4E82 8D 20          BSR    DGTS      GO LOAD THE DIGITS
00285 4E84 CE 0062        LDX    #BUFF+10    SET PNTR
00286 4E87 96 C6          LDA A  SOL7      DECODE INST FOR MESSAGE TYPE
00287 4E89 81 DF          CMP A  #@337
00288 4E8B 22 0A          BHI    FD
00289 4E8D 81 C1          CMP A  #@301
00290 4E8F 22 0C          BHI    TD
00291 4E91 86 3F          LDA A  #$3F      GET "LBL-" MESSAGE
00292 4E93 C6 04          LDA B  #4        GET CHAR. COUNT
00293 4E95 20 DA          BRA    LOAD      LOAD AND RETURN
00294 4E97 86 3D    FD    LDA A  #$3D      GET THE "A-" MESSAGE
00295 4E99 C6 02          LDA B  #2        GET CHAR. COUNT
00296 4E9B 20 D4          BRA    LOAD      LOAD AND RETURN
00297 4E9D 09       TD    DEX              BACK UP ONE
00298 4E9E 86 43          LDA A  #$43      GET "REG-" MESSAGE
00299 4EA0 C6 04          LDA B  #4        GET CHAR. COUNT
00300 4EA2 20 CD          BRA    LOAD      LOAD AND RETURN
00301                *
00302                *DGTS TRANSFERS BCD DIGITS TO THE BUFFER
00303                *
00304 4EA4 C6 10    DGTS  LDA B  #16       LOAD DIGIT/SHIFT CTR INTO ACCB
00305 4EA6 6F 00    PFMT2 CLR    X         CLEAR BYTE IN BUFFER
00306 4EA8 79 002E  PFMT1 ROL    T1        ROLL LSW LEFT
00307 4EAB 79 002D        ROL    T2        ROLL MSW LEFT
00308 4EAE 69 00          ROL    X         ROLL BIT INTO BUFFER WORD
00309 4EB0 5A             DEC B            DECR DIGIT/SHIFT CTR
00310 4EB1 C5 03          BIT B  #3        4 SHIFTS YET?
00311 4EB3 26 F3          BNE    PFMT1     NO, ROLL NEXT BIT
00312 4EB5 D7 2C          STA B  T3        YES, SAVE DIGIT/SHIFT CTR
00313 4EB7 E6 00          LDA B  X         LOAD ACCB WITH DIGIT JUST LOADED
00314 4EB9 CA 30          ORA B  #$30      CONVERT IT TO ASCII
00315 4EBB E7 00          STA B  X         RESTORE IT IN THE BUFFER
00316 4EBD D6 2C          LDA B  T3        RELOAD DIGIT/SHIFT CTR
00317 4EBF 26 01          BNE    INC       DONE?
00318 4EC1 39             RTS              YES, RETURN
00319 4EC2 11       INC   CBA              NO, LEADING ZREO LOADED ?
00320 4EC3 2F E3          BLE    PFMT1     YES, LOAD NEXT DIGIT OVER IT
00321 4EC5 08             INX              NO, INCR BUFFER PTR
00322 4EC6 20 DE          BRA    PFMT2     LOAD NEXT DIGIT INTO NEXT BUFFER LOC
00323      48BC    BINBCD EQU    $48BC
00324      4840    SUB7   EQU    $4840
00327                     NAM    JSRJMP
00328                     OPT    LIST,MEM
00329 4900                ORG    $4900
00330                *
```

```
00331                    *THIS ROUTINE IS THE EXECUTION CODE FOR ALL JSR
00332                    *STATEMENTS. IT CHECKS FOR LEVEL OVERFLOW (>5),
00333                    *SAVES THE RETURN ADRS ON THE STACK, AND
00334                    *TRANSFERS CONTROL TO THE PROPER JMP ROUTINE FOR
00335                    *THE TYPE OF TRANSFER DESIRED.
00336                    *
00337 4900 4C       JSRS    INC A            SET THE JSR FLAG
00338 4901 97 2E            STA A  T1        TO A ONE
00339 4903 31               INS              WIPE OUT OLD STACK
00340 4904 31               INS              RETURN VECTOR
00341 4905 30               TSX              STACK TO INDEX
00342 4906 8C 0044          CPX    #@104     SUBROUTINE OVERFLOW?
00343 4909 26 06            BNE    NOV       BRANCH IF NOT
00344 490B C6 09            LDA B  #9        LOAD ERROR
00345 490D D7 06            STA B  ERROR     STORE IN ERROR WORD
00346 490F 20 69            BRA    MWB       RET. TO SUPV.
00347 4911 DE CA    NOV     LDX    UIP       GET INST PNTR
00348 4913 E6 00            LDA B  X         GET THE INST
00349 4915 2A 01            BPL    SING      BRANCH IF SINGLE BYTE
00350 4917 08               INX              ELSE INC RTN PNTR
00351 4918 DF 20    SING    STX    TP7       SAVE IT IN TEMPORARY
00352 491A 96 21            LDA A  TP7+1     GET THE LS BYTE
00353 491C 36               PSH A            PUSH IT ONTO STACK
00354 491D 96 20            LDA A  TP7       GET THE MS BYTE
00355 491F 36               PSH A            PUSH IT ONTO STACK
00356 4920 5D               TST B            WHAT TYPE OF JSR ?
00357 4921 2B 07            BMI    NX        BRANCH IF ABS OR LBL
00358 4923 C1 63            CMP B  #@143     JSR X ?
00359 4925 27 71            BEQ    JSRLK2    BRANCH IF YES
00360 4927 7E 49B7          JMP    JSRLK1    ELSE DO JSR LBL X
00361 492A C1 C0    NX      CMP B  #@300     ABSOLUTE OR LABEL?
00362 492C 27 3A            BEQ    JSRLK4    GO DO JSR LBL
00363 492E 20 04            BRA    JSRLK3    GO DO JSR ABS
00364                    *
00365                    *THIS ROUTINE EXECUTES ABSOLUTE JUMPS
00366                    *AND ABSOLUTE JSR'S DEPENDING UPON THE
00367                    *ENTRY POINT.
00368                    *
00369 4930 31       JMPABS  INS              WIPE OUT OLD STACK
00370 4931 31               INS              RETURN VECTOR
00371 4932 97 2E            STA A  T1        CLEAR JSR FLAG
00372 4934 DE CA    JSRLK3  LDX    UIP       INDEX=INST ADRS
00373 4936 A6 00            LDA A  X         ACCA=FIRST BYTE
00374 4938 84 0F            AND A  #@17      ACCA=MSB'S OF ADRS
00375 493A E6 01            LDA B  1,X       ACCB=LSB'S OF ADRS
00376 493C 58               ASL B            B7=B6 OF THE ADRS
00377 493D 44               LSR A            CARRY=B7 OF ADRS
00378 493E 56               ROR B            B7 OF ACCB=B7 OF ADRS
00379 493F 4C               INC A            ADRS=ADRS+256
00380 4940 D7 21            STA B  TP7S      SAVE LS BITS
00381 4942 97 20    L2      STA A  TP7       SAVE MS BITS
00382 4944 D0 0C            SUB B  EOPM+1    DO ADRS-EOPM
00383 4946 92 0B            SBC A  EOPM
00384 4948 2A 60            BPL    TOOBIG    ERROR IF ADRS >= EOPM
00385 494A DE 20            LDX    TP7       GET THE ADDRESS
00386 494C E6 00            LDA B  X         ACCB=INST THERE
00387 494E 2B 0D            BMI    LINK      POSSIBLE 2ND BYTE?
00388 4950 8C 0100          CPX    #256      STATE 0?
00389 4953 27 08            BEQ    LINK      YES; CONTINUE
00390 4955 09               DEX              NO; CHECK FIRST BYTE
00391 4956 E6 00            LDA B  X         GET "FIRST" BYTE
00392 4958 C1 BA            CMP B  #$DBB     TWO BYTE INSTRUCTION ?
00393 495A 22 01            BHI    LINK      BRANCH IF NOT
00394 495C 08       DOIT    INX              INCREMENT ADRS
00395 495D DF C8    LINK    STX    UPP       UPDATE THE PRGM PNTR
00396 495F 09               DEX              BACK UP
00397 4960 DF CA            STX    UIP       UPDATE INST PNTR
00398 4962 20 16            BRA    MWB       RETURN TO SUPV.
00399                    *
00400                    *THIS ROUTINE WILL SET THE USER INST PNTR
00401                    *AND THE USER PRGM PNTR TO THE ADDRESS
00402                    *OF THE LABEL IN QUESTION IF IT FINDS THAT LBL.
00403                    *IF THE LABEL IS NON-EXISTANT,AN ERROR
00404                    *WILL RESULT.
00405                    *
```

```
00406 4964 31       JMPLBL INS              WIPE OUT OLD STACK
00407 4965 31              INS              RETURN VECTOR
00408 4966 97 2E           STA A  T1        CLEAR JSR FLAG
00409 4968 DE CA    JSRLK4 LDX    UIP       GET INST ADRS
00410 496A A6 01           LDA A  1,X       ACCA=THE LABEL
00411 496C 8D 0F           BSR    LBLSCH    GO FIND THE LBL
00412 496E 2A EC           BPL    DOIT      BRANCH IF LABEL FOUND
00413 4970 C6 08    CANT   LDA B  #8         LOAD ERROR
00414 4972 D7 06    CK     STA B  ERROR      SAVE IN ERROR WORD
00415 4974 D6 2E           LDA B  T1         TEST JSR FLAG
00416 4976 27 02           BEQ    MWB        BRANCH IF NOT JSR
00417 4978 31              INS               ELSE WIPE OUT THE
00418 4979 31              INS               STORED RETURN VECTOR
00419 497A 7E 7978  MWB    JMP    BWM        RETURN TO SUPV
00420                      *
00421                      *THIS ROUTINE ACCEPTS A LABEL IN ACCA AND RETURNS
00422                      *WITH THE R/W ADRS OF THAT LABEL IN THE
00423                      *INDEX REGISTER. SEARCHING BEGINS AT STATE 0
00424                      *OF THE USER PRGM. IF NO LBL WAS FOUND,
00425                      *THE N BIT WILL BE SET.
00426                      *
00427 497D CE 00FF  LBLSCH LDX    #255       INIT START ADRS
00428 4980 C6 BF           LDA B  #@277      ACCB = LABEL INSTRUCTION
00429 4982 08       SEARCH INX               INC ADRS
00430 4983 9C 0B           CPX    EOPM       END OF MEMORY?
00431 4985 27 0A           BEQ    SORRY      BRANCH IF YES
00432 4987 E1 00           CMP B  X          LABEL INST ?
00433 4989 26 F7           BNE    SEARCH     BRANCH IF NOT
00434 498B 08              INX               ELSE INC ADRS
00435 498C A1 00           CMP A  X          IS IT THE RIGHT LABEL ?
00436 498E 26 F2           BNE    SEARCH     IF NOT, SEARCH
00437 4990 39              RTS               YES! RETURN
00438 4991 86 FF    SORRY  LDA A  #$FF       SET N BIT (ERROR)
00439 4993 39              RTS               RETURN
00440                      *
00441                      *THIS ROUTINE SETS THE USER INST PNTR AND
00442                      *THE USER PRGM PNTR TO THE ABSOLUTE VALUE OF
00443                      *THE INTEGER PART OF THE X REGISTER. IF THE
00444                      *RESULTANT ADRS IS OUT OF RANGE, AN ERROR
00445                      *WILL RESULT AND NO CHANGE IN
00446                      *POINTERS WILL OCCUR.
00447                      *
00448 4994 31       JMPX   INS               WIPE OUT OLD STACK
00449 4995 31              INS               RETURN VECTOR
00450 4996 97 2E           STA A  T1         CLEAR JSR FLAG
00451 4998 D6 90    JSRLK2 LDA B  XR         ACCB=X REG EXP
00452 499A 2B 12           BMI    ADRS0      BRANCH IF NEG EXP
00453 499C 5C              INC B             INCREMENT EXPONENT
00454 499D C1 04           CMP B  #4         COMPARE TO +4
00455 499F 2E 09           BGT    TOOBIG     BRANCH IF BIGGER
00456 49A1 8D 28           BSR    TSFR1      CONVERT X-REG
00457 49A3 96 20           LDA A  TP7        GET MSB'S OF RESULT
00458 49A5 4C              INC A             INC PAGE ADRS
00459 49A6 D6 21           LDA B  TP7+1      GET LS BITS
00460 49A8 20 98           BRA    L2         CHECK FOR VALID ADDRESS
00461 49AA C6 18    TOOBIG LDA B  #24        LOAD ERROR
00462 49AC 20 C4           BRA    CK         GO CHECK FOR JSR
00463 49AE CE 0100  ADRS0  LDX    #256       GET ADRS 0
00464 49B1 20 AA           BRA    LINK       CONTINUE
00465                      *
00466                      *THIS ROUTINE SETS THE USER INST PNTR AND
00467                      *THE USER PRGM PNTR TO THE ADRS OF THE
00468                      *LABEL FOUND IN THE X REG. IF THAT
00469                      *LABEL IS NOT FOUND. NO CHANGE IN
00470                      *POINTERS WILL OCCUR.
00471                      *
00472 49B3 31       JMPLBX INS               WIPE OUT OLD STACK
00473 49B4 31              INS               RETURN VECTOR
00474 49B5 97 2E           STA A  T1         CLEAR JSR FLAG
00475 49B7 4F       JSRLK1 CLR A             PRESET ACCA FOR LABEL 0
00476 49B8 D6 90           LDA B  XR         ACCB=X-REG EXPONENT
00477 49BA 2B 09           BMI    ZOOP       IF NEG., FIND LABEL 0
00478 49BC 5C              INC B             ACCB=# OF INT DIGITS
00479 49BD C1 02           CMP B  #2         MORE THAN 2?
00480 49BF 2E AF           BGT    CANT       ERROR IF YES
```

```
00481 49C1 8D 08          BSR    TSFR1     CONVERT X-REG
00482 49C3 96 21          LDA A  TP7+1     GET BINARY RESULT
00483 49C5 8D 86    ZOOP  BSR    LALSCH    GO SEARCH FOR LABEL
00484 49C7 2A 93          BPL    DOIT      BRANCH IF FOUND
00485 49C9 20 A5          BRA    CANT      ELSE ERROR
00486                *
00487                *THIS ROUTINE TRANSFERS THE INTEGER
00488                *PART OF THE X REG TO W1 & W2 FOR A
00489                *BCD TO BINARY CONVERSION. ON
00490                *ENTRY. ACCB=THE # OF INTEGER DIGITS
00491                *AND INDEX = REGISTER ADDRESS
00492                *
00493                *SPECIAL ENTRY FOR X-REG ONLY
00494                *
00495 49C9 CE 0090 TSFR1  LDX    #XR       PRESET THE INDEX
00496                *
00497 49CE EE 02    TSFR  LDX    2,X       GET 4 MANTISSA DIGITS
00498 49D0 DF 10          STX    W2        SAVE IN BCD CONV WORDS
00499 49D2 C0 04          SUB B  #4        ACCB= -(DIGIT SHIFT COUNT)
00500 49D4 27 0E          BEQ    TSFRC     NO SHIFT; RETURN
00501 49D6 86 04    RB1   LDA A  #4        ELSE LOAD ROTATE COUNTER
00502 49D8 74 0010  RB    LSR    W2        SHIFT MOST SIGNIFICANT WORD
00503 49DB 76 0011        ROR    W1        ROTATE LSW
00504 49DE 4A             DEC A            DEC ROTATE COUNTER
00505 49DF 26 F7          BNE    RB        CONTINUE IF NOT 0
00506 49E1 5C             INC B            INC DIGIT SHIFT COUNTER
00507 49E2 26 F2          BNE    RB1       CONTINUE IF NOT 0
00508 49E4 7E 4A26 TSFRC  JMP    BCDBIN    CONVERT TO BINARY
00509      7978   BWM    EQU    $7978
00510      4A26   BCDBIN EQU    $4A26
00511 7CC0                ORG    $7CC0
00512 7CC0 4983           FDB    JMPLBX
00513 7CC2 4900           FDB    JSRS
00514 7CC4 4994           FDB    JMPX
00515 7CC6 4900           FDB    JSRS
00516 7D80                ORG    $7D80
00517 7D80 4900           FDB    JSRS
00518 7D82 4964           FDB    JMPLAL
00519 7DC0                ORG    $7DC0
00520 7DC0 4900           FDB    JSRS,JSRS,JSRS,JSRS,JSRS,JSRS,JSRS,JSRS
00521 7DD0 4900           FDB    JSRS,JSRS,JSRS,JSRS,JSRS,JSRS,JSRS,JSRS
00522 7DE0 4930           FDB    JMPABS,JMPABS,JMPABS,JMPABS
00523 7DE8 4930           FDB    JMPABS,JMPABS,JMPABS,JMPABS
00524 7DF0 4930           FDB    JMPABS,JMPABS,JMPABS,JMPABS
00525 7DF8 4930           FDB    JMPABS,JMPABS,JMPABS,JMPABS
00528                     NAM    ENDEX
00529                     OPT    LIST,MEM
00530 482A                ORG    $482A
00531                *
00532                *SPECIAL ENTRY FOR "ERASE" ROUTINE
00533                *
00534 482A D6 D5          LDA B  FLAG      GET MEMORY SECURE INFO
00535 482C C4 FE          AND B  #$FE      CLEAR SECURE BIT
00536 482E D7 D5          STA B  FLAG      RESTORE FLAG
00537 4830 97 52          STA A  TA        CLEAR INSERT MODE
00538                *
00539                *THIS ROUTINE EXECUTES THE END STATEMENT
00540                *
00541 4832 8E 0051 ENDEX  LDS    #ISTACK   RESET STACK POINTER
00542 4835 97 08          STA A  UFLG      CLEAR USER FLAGS
00543 4837 97 09          STA A  RSFLG     CLEAR R/S FLAG
00544 4839 CE 0100        LDX    #256      256=USER STATE 0
00545 483C 7E 495D        JMP    LINK      PRESET UPP AND RETURN
00546 7D62                ORG    $7D62
00547 7D62 4832           FDB    ENDEX
00550                     NAM    RTNEX
00551                     OPT    LIST,MEM
00552 4859                ORG    $4859
00553                *
00554                *THIS ROUTINE HANDLES THE RETURN FROM SUBROUTINE
00555                *INSTRUCTION. IT FIRST CHECKS TO SEE IF A JSR
00556                *HAS BEEN EXECUTED. IF NOT, AN ERROR WILL RESULT
00557                *
00558 4859 31       RTNEX INS              WIPE OUT OLD STACK
00559 485A 31             INS              RETURN VECTOR
00560 485B 30             TSX              STACK TO INDEX
```

```
00561 485C 8C 0052          CPX     #ISTACK+1   WAS A GOSUB EXECUTED ?
00562 485F 27 0C            BEQ     NOJSR       BRANCH IF NOT
00563 4861 32               PUL A               ELSE GET MSW OF ADRS
00564 4862 4D               TST A               UDF RETURN ?
00565 4863 2B 0E            BMI     RTNEX1      BRANCH IF YES
00566 4865 97 CA            STA A   UIP         SAVE IN INST PNTR
00567 4867 32               PUL A               GET LSW OF ADRS
00568 4868 97 CB            STA A   UIP+1       SAVE IN INST PNTR
00569 486A 7E 797B RETN     JMP     BWM         RETURN TO SUPV
00570 486D 86 0A   NOJSR    LDA A   #10         LOAD ERROR
00571 486F 97 06            STA A   ERROR       SAVE IN ERROR WORD
00572 4871 20 F7            BRA     RETN        RETURN TO SUPV
00573 4873 84 3F   RTNEX1   AND A   #$3F        REMOVE UDF FLAG
00574 4875 97 C8            STA A   UPP         RESTORE PRGM PNTR HI
00575 4877 32               PUL A               GET PRGM PNTR LO
00576 4878 97 C9            STA A   UPP+1       RESTORE IT
00577 487A 7F 0009          CLR     RSFLG       STOP RUNNING
00578 487D 20 EB            BRA     RETN        RETURN TO KEYBOARD
00579 7D66                  ORG     $7D66
00580 7D66 4859             FDB     RTNEX
00583                       END
```

SYMBOL TABLE

ADATA	0000	ACTL	0001	BDATA	0002	BCTL	0003	INPUT	0004	IOIN	0005
ERROR	0006	TGL	0007	UFLG	0008	RSFLG	0009	EOM	000A	EOPM	000B
STKFLG	000D	RND	000E	DIGFLG	000F	W2	0010	W1	0011	SFLG	0012
DCNTR	0013	TP1	0014	TP1S	0015	TP2	0016	TP2S	0017	TP3	0018
TP3S	0019	TP4	001A	TP4S	001B	TP5	001C	TP5S	001D	TP6	001E
TP6S	001F	TP7	0020	TP7S	0021	T13	0022	T12	0023	T11	0024
T10	0025	T9	0026	T8	0027	T7	0028	T6	0029	T5	002A
T4	002B	T3	002C	T2	002D	T1	002E	ISTK	002F	ISTACK	0051
TA	0052	SPGM	0054	EXTRA	0056	BUFF	0058	REAL	0068	IMAG	0070
AT1	0078	AT2	0080	W	0088	XR	0090	YR	0098	ZR	00A0
TR	00A8	LSTX	00B0	BKWRT	00B8	BKKC	00BA	SOL7	00C6	UPP	00C8
UIP	00CA	ALPHA	00CC	IO1	00CD	IO2	00D0	IT7	00D3	FLAG	00D5
TPOS	00D6	FILE	00D7	AR	00D8	BR	00E0	CR	00E8	DR	00F0
ER	00F8	SDBR	00BA	MT	7E00	TERMN7	003D	IMED	0040	PARCD	00C0
PAREX	0080	NTBL	0000	DOTS	5EC0	PRTDRV	602D	FRMT	5CA8	BLANK	5D75
LOMSG	5780	ROLLD	5582	ROLLU	57F1	PSD	550A	TXL	55E9	STKUP	55EF
MAD	749B	CMP	74AA	NOR	74D6	TXW	7424	TXXR	743B	EXXR	7452
ARSR	753B	OVUNF	75B6	OVERF	75DD	XR0	740A	XRNINE	75C8	UNDRF	75F1
IMULT	7689	QDG	7669	FPA	75FC	FPS	75F6	FPM	7735	FPD	7793
FPAEX	763D	FPMEX	7780	LSHIFT	7521	ZEROX	7489	XZERO0	7416	XZERO2	7417
RECIP	73E6	TXRX	73F3	CONST	6800	FPDBRC	6898	TAN	6849	ATN	69C3
DSZERO	6A45	NTLN	6A58	EXPN	6AC9	SIN	6894	COS	6894	ASIN	6BF2
ACOS	6BF7	PH1	6C5D	PH2	6C8D	PH3	6D34	PH4	6DD0	LSFT8	6E47
SQRT	6E65	MAD8	6F2C	CMP8	53E4	IOUPX	6F52	LOG10	6FA7	YUPX	6FE9
RTOP	7328	PTOR	7386	FRMT1	4E09	DONTT	4E16	FC4	4E19	PMODE	4E1E
FC2	4E47	FC3	4E6A	LOAD	4E71	FC	4E78	FC1	4E79	FD	4E97
TO	4E9D	DGTS	4EA4	PFMT2	4EA6	PFMT1	4EAB	INC	4EC2	BINBCD	488C
SUB7	4840	JSRS	4900	NOV	4911	SING	4918	NX	492A	JMPABS	4930
JSRLK3	4934	L2	4942	DOIT	495C	LINK	495D	JMPLBL	4964	JSRLK4	4968
CANT	4970	CK	4972	MWB	497A	LBLSCH	497D	SEARCH	4982	SORRY	4991
JMPX	4994	JSRLK2	4998	TOOBIG	49AA	ADRS0	49AE	JMPLBX	49B3	JSRLK1	49B7
ZOOP	49C5	TSFR1	49CB	TSFR	49CE	RB1	49D6	RB	49D8	TSFRC	49E4
BWM	797B	BCDBIN	4A26	ENDEX	4B32	RTNEX	4859	RETN	486A	NOJSR	486D
RTNEX1	4873										

TOTAL ERRORS 4

```
****ERROR 201
 218   NAM RSEXR
****ERROR 201
 254   NAM SUBADD
****ERROR 201
 282   NAM BINBCD
****ERROR 201
 325   NAM CJUFLG
```

```
00215                          OPT    LIST,MEM
00218                          NAM    RSEXR
00219                          OPT    LIST,MEM
00220 4800                     ORG    $4800
00221                  *
00222                  *THIS ROUTINE PROCESSES THE RUN-STOP KEY
00223                  *
00224 4800 0F          RSEX    SEI                DISABLE INTERRUPTS
00225 4801 96 09               LDA A  RSFLG      GET R/S FLAG
00226 4803 48                  ASL A             SHIFT B7 TO C
00227 4804 25 1F               BCS    B7ON       BRANCH IF B7 WAS A 1
00228 4806 2B 10               BMI    B6ON       BRANCH IF B6 IS 1
00229 4808 44                  LSR A             B7=B6=0
00230 4809 43                  COM A             B7=B6=1 (RUN!)
00231 480A DE C8               LDX    UPP        GET PRGM PNTR
00232 480C 09                  DEX               SUBTRACT 1
00233 480D DF CA               STX    UIP        SAVE IN INST PNTR
00234 480F 5F                  CLR B
00235 4810 D7 B9               STA B  BKWRT+1    CLEAR KEY BUFFER
00236 4812 D7 12               STA B  SFLG       CLEAR STEP FLAG
00237 4814 97 09       ZIP     STA A  RSFLG      RESTORE NEW FLAG
00238 4816 0E          BACK    CLI               ENABLE INTERRUPTS
00239 4817 39                  RTS               RETURN
00240 4818 4F          B6ON    CLR A             CLEAR R/S FLAG
00241 4819 DE CA               LDX    UIP        GET INST PNTR
00242 481B DF C8       ZIP1    STX    UPP        SAVE PRGM PNTR
00243 481D C6 40               LDA B  #$40       PRESET ACCB
00244 481F DA 08               ORA B  UFLG       SET THE ENTRY FLAG
00245 4821 D7 08               STA B  UFLG       RESTORE FLAGS
00246 4823 20 EF               BRA    ZIP        GO STORE FLAG
00247 4825 2A EF       B7ON    BPL    BACK       RETURN IF B6=0
00248 4827 44                  LSR A             0 TO B7
00249 4828 20 EA               BRA    ZIP        RETURN
00250 7D60                     ORG    $7D60
00251 7D60 4800                FDB    RSEX
00254                          NAM    SUBADD
00255                          OPT    LIST,MEM
00256 4840                     ORG    $4840
00257                  *
00258                  *     SUBADD - 16 BIT 2'S COMP ADD/SUBTRACT
00259                  *
00260                  *     BRAD MILLER  3/4/74
00261                  *
00262                  *THIS SUBROUTINE PERFORMS EITHER TP7-TP6 OR TP7+TP6
00263                  *DEPENDING UPON THE ENTRY POINT WHERE TP7 AND
00264                  *TP6 ARE "16 BIT WORDS" STORED IN MEMORY. THE CONTENTS
00265                  *OF THE INDEX ARE DESTROYED.
00266                  *THE RESULT IS STORED IN TP7 AND TP7+1
00267                  *
00268 4840 73 001E     SUB7    COM    TP6        1'S COMP THE MSB'S
00269 4843 73 001F             COM    TP6+1      1'S COMP LSB'S
00270 4846 DE 1E               LDX    TP6        INDEX=THE NUMBER
00271 4848 08                  INX               INC FOR 2'S COMP
00272 4849 DF 1E               STX    TP6        RESTORE THE NUMBER
00273 484B 96 21       ADD7    LDA A  TP7S       GET LS BITS
00274 484D 9B 1F               ADD A  TP6S       ADD
00275 484F 97 21               STA A  TP7S       RESTORE
00276 4851 96 20               LDA A  TP7        GET MS BITS
00277 4853 99 1E               ADC A  TP6        ADD
00278 4855 97 20               STA A  TP7        RESTORE
00279 4857 39                  RTS               RETURN
00282                          NAM    BINBCD
00283                          OPT    LIST,MEM
00284 488C                     ORG    $488C
00285                  *
00286                  *THIS ROUTINE DOES A BINARY TO BCD CONV
00287                  *ON THE INDEX REGISTER. THE MAX INPUT IS
00288                  *DECIMAL 9999. THE BCD DIGITS ARE STORED
00289                  *IN T2 AND T1 WITH MS DIGITS IN T2
00290                  *
00291 488C DF 20       BINBCD  STX    TP7        SAVE BINARY NUMBER
00292 488E 7F 002E             CLR    T1         CLEAR LS BCD ACC
00293 48C1 7F 002D             CLR    T2         CLEAR MS BCD ACC
00294 48C4 CE FC18             LDX    #-1000     PREPARE TO SUBT 1000
00295 48C7 DF 1E       START1  STX    TP6        SAVE THE SUBT VALUE
00296 48C9 BD 484B     MORE1   JSR    ADD7       DO TP7=TP7+TP6
```

```
00297 48CC 2B 05          BMI     RESTOR   RESTORE IF NEGATIVE
00298 48CE 7C 002E        INC     T1       ELSE INC LS BCD WORD
00299 48D1 20 F6          BRA     MORE1    CONTINUE SUBTRACTING
00300 48D3 BD 4840 RESTOR JSR     SUB7     RESTORE BINARY #
00301 48D6 20 11          BRA     STOBCD   STORE BCD DIGIT
00302 48D8 CE FF9C BIN3   LDX     #-100    PREPARE TO SUBT 100
00303 48DB 20 EA          BRA     START1   GO DO 100'S
00304 48DD CE FFF6 BIN2   LDX     #-10     PREPARE TO SUBT 10
00305 48E0 20 E5          BRA     START1   GO DO 10'S
00306 48E2 96 21   BIN1   LDA A   TP7+1    GET LS BCD DIGIT
00307 48E4 9A 2E          ORA A   T1       COMBINE WITH LS BCD WORD
00308 48E6 97 2E          STA A   T1       RESTORE THE RESULT
00309 48E8 39             RTS               CONVERSION COMPLETE
00310 48E9 96 1F   STOBCD LDA A   TP6+1    GET LSB'S OF SUBT WORD
00311 48EB 44             LSR A            SHIFT RIGHT FOR TEST
00312 48EC C6 04          LDA B   #4       ACCB=ROTATE COUNT
00313 48EE 0C      ROLLIT CLC              CLEAR C FOR ROTATE
00314 48EF 79 002E        ROL     T1       ROT LEFT LSW,C=B7
00315 48F2 79 002D        ROL     T2       ROT LEFT MSW,B0=C
00316 48F5 5A             DEC B            DEC ROTATE CNTR
00317 48F6 26 F6          BNE     ROLLIT   4 ROTATES YET?
00318 48F8 44             LSR A            SHIFT LSB TO C
00319 48F9 25 E7          BCS     BIN1     C=1 MEANS 10'S DONE
00320 48FB 44             LSR A            SHIFT AGAIN
00321 48FC 25 DF          BCS     BIN2     C=1 MEANS 100'S DONE
00322 48FE 20 D8          BRA     BIN3     1000'S DONE BY DEFAULT
00325                     NAM     CJUFLG
00326                     OPT     LIST,MEM
00327 4880                ORG     $4880
00328               *
00329               *    BRAD MILLER  3/4/74
00330               *
00331               *    USER FLAG SUBROUTINES
00332               *
00333               ***SET USER FLAG***
00334               *
00335               *THIS ROUTINE HAS 8 ENTRY POINTS, 1 FOR EACH
00336               *USER FLAG. ON ENTRY, ACCA MUST BE ZERO AND
00337               *THE CARRY BIT MUST BE SET.
00338               *
00339 4880 49       SF8    ROL A             THE C BIT IS SET
00340 4881 49       SF7    ROL A             AND ROTATED THRU ACCA
00341 4882 49       SF6    ROL A             STOPPING AT THE
00342 4883 49       SF5    ROL A             CORRECT POSITION FOR
00343 4884 49       SF4    ROL A             EACH FLAG. B7=FLG 8
00344 4885 49       SF3    ROL A             WHILE B0= FLG 1
00345 4886 49       SF2    ROL A
00346 4887 49       SF1    ROL A
00347 4888 9A 08           ORA A   UFLG      "OR" IN THE USER FLG WORD
00348 488A 97 08           STA A   UFLG      RESTORE NEW STATUS
00349 488C 39              RTS               RETURN
00350               *
00351               *    CLEAR USER FLAG SUBROUTINE
00352               *
00353               *THIS ROUTINE HAS 8 ENTRY POINTS, 1 FOR
00354               *EACH USER FLAG. ON ENTRY, ACCA MUST BE
00355               *ZERO AND THE CARRY BIT MUST BE SET.
00356               *
00357 488D 49       CF8    ROL A             THE C BIT IS
00358 488E 49       CF7    ROL A             SET AND ROTATED
00359 488F 49       CF6    ROL A             THRU ACCA STOPPING
00360 4890 49       CF5    ROL A             AT THE CORRECT
00361 4891 49       CF4    ROL A             LOCATION FOR EACH
00362 4892 49       CF3    ROL A             FLAG, B7= FLG 8
00363 4893 49       CF2    ROL A             WHILE B0= FLG 1
00364 4894 49       CF1    ROL A
00365 4895 43              COM A             1'S COMP ACCA
00366 4896 94 08           AND A   UFLG      CLEAR THE FLAG
00367 4898 97 08           STA A   UFLG      RESTORE THE FLAG WORD
00368 489A 39              RTS               RETURN
00369               *
00370               *    IF FLAG SET SUBROUTINE
00371               *
00372               *THIS ROUTINE HAS 8 ENTRY POINTS, 1 FOR
00373               *EACH USER FLAG. ON ENTRY, ACCA MUST
00374               *BE ZERO AND THE CARRY BIT MUST BE SET
```

```
00375                    *
00376 489B 49            IFS8   ROL A              THE C BIT IS
00377 489C 49            IFS7   ROL A              SET AND ROTATED
00378 489D 49            IFS6   ROL A              THRU ACCA STOPPING
00379 489E 49            IFS5   ROL A              AT THE CORRECT
00380 489F 49            IFS4   ROL A              LOCATION FOR EACH
00381 48A0 49            IFS3   ROL A              FLAG. B7= FLG 8
00382 48A1 49            IFS2   ROL A              WHILE B0= FLG 1
00383 48A2 49            IFS1   ROL A
00384 48A3 94 08                AND A  UFLG        ACCA=FLAG STATUS
00385 48A5 27 01                BEQ    NOTSET      ACCA=0 IMPLIES NOT SET
00386 48A7 39                   RTS                FLAG SET; RETURN
00387 48A8 7E 48AB NOTSET       JMP    NMET        GO MODIFY PRGM PNTR AND RETURN
00388                    *
00389                    *     IF FLAG CLEAR SUBROUTINE
00390                    *
00391                    *THIS ROUTINE HAS 8 ENTRY POINTS. 1 FOR
00392                    *EACH USER FLAG. ON ENTRY. ACCA MUST BE
00393                    *ZERO AND THE CARRY BIT MUST BE SET.
00394                    *
00395 48AB 49            IFC8   ROL A              THE C BIT IS
00396 48AC 49            IFC7   ROL A              SET AND ROTATED
00397 48AD 49            IFC6   ROL A              THRU ACCA STOPPING
00398 48AE 49            IFC5   ROL A              AT THE CORRECT
00399 48AF 49            IFC4   ROL A              LOCATION FOR EACH
00400 48B0 49            IFC3   ROL A              FLAG. B7= FLG 8
00401 48B1 49            IFC2   ROL A              WHILE B0= FLG 1
00402 48B2 49            IFC1   ROL A
00403 48B3 94 08                AND A  UFLG        ACCA = FLAG STATUS
00404 48B5 26 F1                BNE    NOTSET      ACCA#0 IMPLIES FLG SET
00405 48B7 39                   RTS                NOT SET; RETURN
00406       48A8        NMET    EQU    $48A8
00407 7C80                      ORG    $7C80
00408 7C80 4887                 FDB    SF1
00409 7C82 4886                 FDB    SF2
00410 7C84 4885                 FDB    SF3
00411 7C86 4884                 FDB    SF4
00412 7C88 4883                 FDB    SF5
00413 7C8A 4882                 FDB    SF6
00414 7C8C 4881                 FDB    SF7
00415 7C8E 4880                 FDB    SF8
00416 7C90 48A2                 FDB    IFS1
00417 7C92 48A1                 FDB    IFS2
00418 7C94 48A0                 FDB    IFS3
00419 7C96 489F                 FDB    IFS4
00420 7C98 489E                 FDB    IFS5
00421 7C9A 489D                 FDB    IFS6
00422 7C9C 489C                 FDB    IFS7
00423 7C9E 4898                 FDB    IFS8
00424 7CA0 4894                 FDB    CF1
00425 7CA2 4893                 FDB    CF2
00426 7CA4 4892                 FDB    CF3
00427 7CA6 4891                 FDB    CF4
00428 7CA8 4890                 FDB    CF5
00429 7CAA 488F                 FDB    CF6
00430 7CAC 488E                 FDB    CF7
00431 7CAE 488D                 FDB    CF8
00432 7CB0 4882                 FDB    IFC1
00433 7CB2 4881                 FDB    IFC2
00434 7CB4 4880                 FDB    IFC3
00435 7CB6 48AF                 FDB    IFC4
00436 7CB8 48AE                 FDB    IFC5
00437 7CBA 48AD                 FDB    IFC6
00438 7CBC 48AC                 FDB    IFC7
00439 7CBE 48AB                 FDB    IFC8
00442                           END
```

SYMBOL TABLE

```
ADATA   0000  ACTL    0001  BDATA   0002  BCTL    0003  INPUT   0004  IOIN    0005
ERROR   0006  TGL     0007  UFLG    0008  RSFLG   0009  EOM     000A  EOPM    000B
STKFLG  000D  RND     000E  DIGFLG  000F  W2      0010  W1      0011  SFLG    0012
DCNTR   0013  TP1     0014  TP1S    0015  TP2     0016  TP2S    0017  TP3     0018
TP3S    0019  TP4     001A  TP4S    001B  TP5     001C  TP5S    001D  TP6     001E
```

```
TP6S    001F  TP7     0020  TP7S    0021  T13     0022  T12     0023  T11     0024
T10     0025  T9      0026  T8      0027  T7      0028  T6      0029  T5      002A
T4      002B  T3      002C  T2      002D  T1      002E  ISTK    002F  ISTACK  0051
TA      0052  SPGM    0054  EXTRA   0056  BUFF    0058  REAL    0068  IMAG    0070
AT1     0078  AT2     0080  W       0088  XR      0090  YR      0098  ZR      00A0
TR      00A8  LSTX    00B0  BKWRT   00B8  BKKC    00BA  SOL7    00C6  UPP     00C8
UIP     00CA  ALPHA   00CC  IO1     00CD  IO2     00D0  IT7     00D3  FLAG    00D5
TPOS    00D6  FILE    00D7  AR      00D8  BR      00E0  CR      00E8  DR      00F0
ER      00F8  SDBB    00BA  MT      7E00  TERMN7  003D  IMED    0040  PARCD   00C0
PAREX   00B0  NTBL    0000  DOTS    5EC0  PRTDRV  602D  FRMT    5CAB  BLANK   5D75
LDMSG   57BD  ROLLD   55B2  ROLLU   57F1  PSD     550A  TXL     55E9  STKUP   55EF
MAD     7498  CMP     744A  NOR     7406  TXW     7424  TXXR    743B  EXXR    7452
APSR    7538  OVUNF   7586  OVERF   75DD  XR0     740A  XPNINE  75C8  UNDRF   75F1
IMULT   7649  UDG     7669  FPA     75FC  FPS     75F6  FPM     7735  FPD     7793
FPAEX   763D  FPMEX   7780  LSHIFT  7521  ZEROX   7489  XZERO0  7416  XZERO2  7417
RECIP   73E6  TXRX    73F3  CONST   6B00  FPDBRC  6B98  TAN     6BA9  ATN     69C3
DSZERO  6A46  NTLN    6A58  EXPN    6AC9  SIN     6B94  COS     6B9A  ASIN    6BF2
ACOS    6BF7  PH1     6C5D  PH2     6C8D  PH3     6D34  PH4     6DD0  LSFT8   6E47
SQRT    6E65  MAD8    6F2C  CMP8    53E4  IOUPX   6F52  LOG10   6FA7  YUPX    6FE9
RTOP    7328  PTOR    7386  RSEX    4800  ZIP     4814  BACK    4816  B60N    4818
ZIP1    481B  B70N    4825  SUB7    4840  ADD7    484B  BINBCD  48BC  START1  48C7
MORE1   48C9  RESTOR  48D3  BIN3    48D8  BIN2    48DD  BIN1    48E2  STOBCD  48E9
ROLLIT  48EF  SF8     4880  SF7     4881  SF6     4882  SF5     4883  SF4     4884
SF3     4885  SF2     4886  SF1     4887  CF8     4880  CF7     488E  CF6     488F
CF5     4890  CF4     4891  CF3     4892  CF2     4893  CF1     4894  IFS8    4898
IFS7    489C  IFS6    489D  IFS5    489E  IFS4    489F  IFS3    48A0  IFS2    48A1
IFS1    48A2  NOTSET  48A8  IFC8    48AB  IFC7    48AC  IFC6    48AD  IFC5    48AE
IFC4    48AF  IFC3    48B0  IFC2    48B1  IFC1    48B2  NMET    48A8

TOTAL ERRORS   4


****ERROR 201
 218  NAM IFEX
****ERROR 201
 360  NAM REGEX

00215                          OPT    LIST,MEM
00218                          NAM    IFEX
00219                          OPT    LIST,MEM
00220 48A4                     ORG    $48A4
00221                  *
00222                  *THIS ROUTINE DOES IF-, IF+, AND IF0
00223                  *
00224 48A4 91 92       IF0     CMP A  XR+2      XREG = 0 ?
00225 48A6 27 0C               BEQ    MET       BRANCH IF YES
00226 48A8 DE CA       NMET    LDX    UIP       ELSE GET INST PNTR
00227 48AA 08                  INX               SKIP ONE INST
00228 48AB A6 00               LDA A  X         GET CURRENT INST
00229 48AD 81 BA               CMP A  #SDBB     TWO BYTE INSTRUCTION ?
00230 48AF 23 01               BLS    MET-2     BRANCH IF NOT
00231 48B1 08                  INX              ELSE MOVE FORWARD
00232 48B2 DF CA               STX    UIP       RESTORE POINTER
00233 48B4 39          MET     RTS              RETURN
00234                  *
00235 48B5 96 91       IFPLS   LDA A  XR+1      GET MANTISSA SIGN
00236 48B7 2B EF               BMI    NMET      NOT MET IF NEG.
00237 48B9 39                  RTS
00238                  *
00239 48BA 96 91       IFMI    LDA A  XR+1      GET MANTISSA SIGN
00240 48BC 2A EA               BPL    NMET      NOT MET IF POS.
00241 48BE 39                  RTS
00242                  *
00243 48BF DE 90       IFXEY   LDX    XR        GET X
00244 48C1 9C 98               CPX    YR        = Y ?
00245 48C3 26 E3               BNE    NMET      BRANCH IF NOT
00246 48C5 DE 92               LDX    XR+2      ELSE CONTINUE TEST
00247 48C7 9C 9A               CPX    YR+2
00248 48C9 26 DD               BNE    NMET
00249 48CB DE 94               LDX    XR+4
00250 48CD 9C 9C               CPX    YR+4
00251 48CF 26 D7               BNE    NMET
00252 48D1 DE 96               LDX    XR+6
00253 48D3 9C 9E               CPX    YR+6
```

```
00254 4BD5 26 D1          BNE    NMET
00255 4BD7 39             RTS                  RETURN IF CONDITION MET
00256                     *
00257 4BD8 4C       IFXGEY INC A                SET "REVERSE SENSE" FLAG
00258 4BD9 97 2E    IFXLTY STA A  T1            0 TO LESS THAN FLAG
00259 4BDB CE 0078        LDX    #AT1           PRESET INDEX
00260 4BDE BD 73F3        JSR    TXRX           TRANSFER X REG
00261 4BE1 CE 0098        LDX    #YR            PRESET INDEX
00262 4BE4 BD 75F6        JSR    FPS            SUBTRACT
00263 4BE7 96 91          LDA A  XR+1           GET MANTISSA SIGN
00264 4BE9 D6 92          LDA B  XR+2           GET MANTISSA DIGITS
00265 4BEB CE 0078        LDX    #AT1           PRESET INDEX
00266 4BEE BD 743B        JSR    TXXR           RESTORE X REG
00267 4BF1 4D             TST A                 TEST SIGN OF RESULT
00268 4BF2 2B 08          BMI    NMET1          BRANCH IF NEG.
00269 4BF4 5D             TST B                 ELSE TEST FOR ZERO
00270 4BF5 27 05          BEQ    NMET1          BRANCH IF ZERO RESULT
00271 4BF7 96 2E    MET1  LDA A  T1             GET SKIP FLAG
00272 4BF9 26 AD          BNE    NMET           REVERSE SENSE IF SET
00273 4BFB 39             RTS                   ELSE RETAIN SENSE
00274 4BFC 96 2E    NMET1 LDA A  T1             NOT MET IF X>Y
00275 4BFE 27 A8          BEQ    NMET           RETAIN SKIP SENSE
00276 4C00 39             RTS                   ELSE REVERSE SENSE
00277                     *
00278                     *
00279                     *THIS ROUTINE HANDLES EXECUTION OF ALL "FOR" STATEMENTS
00280                     *
00281                     *
00282 4C01 CE 00E8  NEXTC LDX    #CR            GET C-REG ADRS
00283 4C04 DF 27          STX    T8             SAVE IT
00284 4C06 CE 00F0        LDX    #DR            GET LOOP INC. ADRS
00285 4C09 86 AB          LDA A  #$AB           GET "FOR C" INST
00286 4C0B 20 11          BRA    NEXTJ          SAVE THEM
00287 4C0D CE 00E0  NEXTB LDX    #BR            GET B-REG ADRS
00288 4C10 86 AA          LDA A  #$AA           GET "FOR B" INST
00289 4C12 20 05          BRA    NEXTI          SAVE THEM
00290 4C14 CE 00D8  NEXTA LDX    #AR            GET A-REG ADRS
00291 4C17 86 AC          LDA A  #$AC           GET "FOR A" INST
00292 4C19 DF 27    NEXTI STX    T8             SAVE LOOP VARIABLE ADRS
00293 4C1B CE 6838        LDX    #FP1           GET LOOP INCREMENT
00294 4C1E DF 2D    NEXTJ STX    T2             SAVE LOOP INC. ADRS
00295 4C20 97 29          STA A  T6             SAVE "FOR" INST
00296 4C22 CE 0078        LDX    #AT1           GET ARITH TEMP ADRS
00297 4C25 BD 73F3        JSR    TXRX           TRANSFER X-REG
00298 4C28 DE 2D          LDX    T2             GET LOOP INC. ADRS
00299 4C2A BD 743B        JSR    TXXR           MOVE IT TO X-REG
00300 4C2D DE 27          LDX    T8             GET LOOP VARIABLE ADRS
00301 4C2F BD 75FC        JSR    FPA            ADD THE "1"
00302 4C32 DE 27          LDX    T8             RESTORE LOOP VAR. ADRS
00303 4C34 BD 73F3        JSR    TXRX           REPLACE WITH UPDATED NUMBER
00304 4C37 D6 0A          LDA B  EOM            GET END OF MEMORY PNTR
00305 4C39 5A             DEC B                 DEC PAGE ADRS
00306 4C3A D7 1C          STA B  TP5            SAVE IT
00307 4C3C DE CA          LDX    UIP            GET INST PNTR
00308 4C3E E6 00          LDA B  X              GET THE "NEXT" INST
00309 4C40 C0 AD          SUB B  #$AD           REMOVE OFFSET
00310 4C42 58             ASL B                 MULTIPLY BY 2
00311 4C43 58             ASL B                 MULTIPLY BY FOUR
00312 4C44 58             ASL B
00313 4C45 CB D8          ADD B  #$D8           ADD ALPHA REG OFFSET
00314 4C47 D7 1D          STA B  TP5S           SAVE IN LS BITS OF PNTR
00315 4C49 DE 1C          LDX    TP5            GET ALPHA REG PNTR
00316 4C4B BD 75F6        JSR    FPS            DO THE "END OF LOOP" CHECK
00317 4C4E 96 91          LDA A  XR+1           GET SIGN OF RESULT
00318 4C50 D6 92          LDA B  XR+2           GET MANTISSA DIGITS
00319 4C52 CE 0078        LDX    #AT1           GET X-REG TEMP ADRS
00320 4C55 BD 743B        JSR    TXXR           RESTORE X-REG
00321 4C58 5D             TST B                 ZERO RESULT ?
00322 4C59 27 07          BEQ    MOREL          CONTINUE IF YES
00323 4C5B DE 2D          LDX    T2             GET LOOP INC. ADRS
00324 4C5D A8 01          EOR A  1,X            SIGNS DIFFER ?
00325 4C5F 2A 01          BPL    MOREL          CONTINUE IF NOT
00326 4C61 39             RTS                   ELSE RETURN
```

```
00327 4C62 DE CA    MOREL   LDX     UIP         GET CURRENT USER ADRS
00328 4C64 D6 29            LDA B   T6          GET "FOR" INST FOR SEARCH
00329 4C66 8C 0100  MOREL1  CPX     #256        USER STATE ZERO YET ?
00330 4C69 27 08            BEQ     FEROR       ERROR IF YES
00331 4C6B 09               DEX                 ELSE DEC SEARCH PNTR
00332 4C6C E1 00            CMP B   X           IS THIS THE "FOR" STMT ?
00333 4C6E 26 F6            BNE     MOREL1      BRANCH IF NOT
00334 4C70 DF CA            STX     UIP         ELSE SAVE NEW INST ADRS
00335 4C72 39               RTS                 RETURN
00336 4C73 96 0D    FEROR   LDA A   #13         LOAD ERROR
00337 4C75 97 06            STA A   ERROR       SAVE IT FOR OUTPUT
00338 4C77 39       FORS    RTS                 RETURN
00339               *
00340               *
00341       6838    FP1     EQU     $6838       FLOATING POINT "1"
00342               *
00343               *
00344 7C62                  ORG     $7C62
00345 7C62 4BB5             FDB     IFPLS
00346 7C64 4BBA             FDB     IFMI
00347 7C66 4BA4             FDB     IF0
00348 7C68 4BBF             FDB     IFXEY
00349 7C6A 4BD9             FDB     IFXLTY
00350 7C6C 4BD8             FDB     IFXGEY
00351 7D54                  ORG     $7D54
00352 7D54 4C77             FDB     FORS
00353 7D56 4C77             FDB     FORS
00354 7D58 4C77             FDB     FORS
00355 7D5A 4C14             FDB     NEXTA
00356 7D5C 4C0D             FDB     NEXTB
00357 7D5E 4C01             FDB     NEXTC
00360                       NAM     REGEX
00361                       OPT     LIST.MEM
00362 4C78                  ORG     $4C78
00363               *
00364               *THIS ROUTINE HANDLES ALL 78 OF THE STORAGE
00365               *REGISTER INSTRUCTIONS. THIS INCLUDES REG. ARITHMETIC.
00366               *INDIRECTS, ETC. FOR ALL RECALL INSTRUCTIONS.
00367               *THE X REGISTER IS COPIED TO LAST X. TP7.T1.T2.
00368               *AND T3 ARE USED BY THIS ROUTINE. THE FLOATING
00369               *POINT ROUTINES ARE CALLED IF NECESSARY.
00370               *
00371 4C78 4C       DVDENT  INC A               ENTRY PT. FOR DIVISION
00372 4C79 4C       MLTENT  INC A               ENTRY PT. FOR MULTIPLICATION
00373 4C7A 4C       SUBENT  INC A               ENTRY PT. FOR SUBTRACTION
00374 4C7B 4C       ADDENT  INC A               ENTRY PT FOR ADDITION
00375 4C7C 97 2E    STOENT  STA A   T1          SAVE ARITHMETIC OPERATOR
00376 4C7E 5F               CLR B
00377 4C7F 20 09            BRA     PENT        CLEAR RECALL FLAG
00378 4C81 91 0F    RCLENT  CMP A   DIGFLG      DOING DIGIT ENTRY ?
00379 4C83 27 05            BEQ     RENT        BRANCH IF NOT
00380 4C85 97 0F            STA A   DIGFLG      ELSE TERM. IT
00381 4C87 43               COM A               SET MSB
00382 4C88 97 0D            STA A   STKFLG      ENABLE LIFT
00383 4C8A D7 20    RENT    STA B   T2          SET UP RECALL FLAG
00384 4C8C DE CA            LDX     UIP         GET INST ADDRESS
00385 4C8E E6 00            LDA B   X           GET THE INSTRUCTION
00386 4C90 17               TBA                 SAVE IN ACCA
00387 4C91 97 2C            STA A   T3          AND IN T3
00388 4C93 C1 C7            CMP B   #@307       ALPHA REGISTER?
00389 4C95 22 32            BHI     NUMRIC      BRANCH IF NOT
00390 4C97 C1 C1            CMP B   #@301       ONE OR TWO BYTE ?
00391 4C99 22 09            BHI     TWOBT       BRANCH IF TWO BYTE
00392 4C9B 80 64            SUB A   #$64        REMOVE INST OFFSET
00393 4C9D BD 5CA8          JSR     FRMT        CONVERT TO BCD
00394 4CA0 84 0F            AND A   #$F         GET REG #
00395 4CA2 20 04            BRA     IADRS       CONVERT TO R/W ADDRESS
00396 4CA4 A6 01    TWOBT   LDA A   1,X         GET SECOND BYTE
00397 4CA6 80 64            SUB A   #100        REMOVE OFFSET
00398 4CA8 81 05    IADRS   CMP A   #5          REG IN USER MEMORY ?
00399 4CAA 2C 07            BGE     USMEM       BRANCH IF YES
00400 4CAC CE 00D8          LDX     #AR         ELSE GET BASE PAGE ADRS
00401 4CAF DF 20            STX     TP7         SAVE FOR MODIFICATION
00402 4CB1 20 0B            BRA     GEN         GO GENERATE ADDRESS
00403 4CB3 D6 0A    USMEM   LDA B   EOM         GET END OF MEM PAGE
00404 4CB5 5A               DEC B               BACK UP ONE
```

```
00405 4CB6 D7 20          STA B  TP7      SAVE IN MS BITS OF ADRS
00406 4CB8 C6 D8          LDA B  #$D8     GET ADRS ON PAGE
00407 4CBA D7 21          STA B  TP7+1    SAVE IN LS BITS OF ADRS
00408 4CBC 80 05          SUB A  #5       REMOVE OFFSET
00409 4CBE 48       GEN   ASL A           MULTIPLY REG # BY 8
00410 4CBF 48             ASL A
00411 4CC0 48             ASL A
00412 4CC1 9B 21          ADD A  TP7+1    ADD TO ADDRESS
00413 4CC3 97 21          STA A  TP7+1    RESTORE
00414 4CC5 DE 20          LDX    TP7      INDEX=ADDRESS
00415 4CC7 20 0A          BRA    VALID    CONTINUE
00416 4CC9 A6 01    NUMRIC LDA A  1,X      GET SECOND BYTE
00417 4CCB 48             ASL A           SHIFT ADRS TO B7
00418 4CCC 57             ASR B           MSB TO CARRY
00419 4CCD 46             ROR A           CARRY TO B7
00420 4CCE BD 4D58        JSR    SADRS    GO GENERATE SYSTEM ADDRESS
00421 4CD1 26 26          BNE    OVFLO    BRANCH IF ERROR
00422 4CD3 D6 2C    VALID LDA B  T3       GET THE INST
00423 4CD5 C1 D3          CMP B  #@323    POSSIBLE INDIRECT?
00424 4CD7 22 08          BHI    INDR     BRANCH IF YES
00425 4CD9 C1 C7          CMP B  #@307    POSSIBLE INDIRECT?
00426 4CDB 22 21          BHI    INSTYP   BRANCH IF NOT
00427 4CDD C1 C1          CMP B  #@301    POSSIBLE INDIRECT?
00428 4CDF 23 1D          BLS    INSTYP   BRANCH IF NOT
00429 4CE1 4F       INDR  CLR A           PRESET FOR REG 0
00430 4CE2 E6 00          LDA B  X        GET REG EXPONENT
00431 4CE4 2B 0E          BMI    REG0     REG 0 IF NEG
00432 4CE6 5C             INC B           ACCB=# OF DIGITS
00433 4CE7 C1 03          CMP B  #3       GREATER THAN 3?
00434 4CE9 2E 0E          BGT    OVFLO    BRANCH IF YES
00435 4CEB BD 49CE        JSR    TSFR     GO GENERATE NEW ADRS
00436 4CEE 96 20          LDA A  TP7      GET MS BINARY
00437 4CF0 26 07          BNE    OVFLO    ERROR IF NON-ZERO
00438 4CF2 96 21          LDA A  TP7+1    GET LS BINARY
00439 4CF4 BD 4D5A  REG0  JSR    SADRS    GENERATE SYSTEM ADRS
00440 4CF7 27 05          BEQ    INSTYP   BRANCH IF NO ERROR
00441 4CF9 86 18    OVFLO LDA A  #24       LOAD ERROR
00442 4CFB 97 06          STA A  ERROR    SAVE IN ERROR WORD
00443 4CFD 39             RTS             RETURN
00444 4CFE 96 2D    INSTYP LDA A  T2      GET RECALL FLAG
00445 4D00 27 19          BEQ    STOINS   0 IMPLIES STORE INST.
00446 4D02 DF 1E          STX    TP6      SAVE REG ADDRESS
00447 4D04 BD 55E9        JSR    TXL      X TO LAST X
00448 4D07 BD 55EF        JSR    STKUP    STACK UP ?
00449 4D0A DE 1E          LDX    TP6      RESTORE REG ADDRESS
00450 4D0C BD 743B        JSR    TXXR     DO THE RECALL
00451 4D0F D6 2C    PCHEK LDA B  T3       GET THE INSTRUCTION
00452 4D11 C1 BA          CMP B  #$DBB    TWO BYTE INSTRUCTION ?
00453 4D13 23 05          BLS    DDONE    BRANCH IF YES
00454 4D15 DE CA    LBLEX LDX    UIP      GET INST PNTR
00455 4D17 08             INX             INC TO SKIP 2ND BYTE
00456 4D18 DF CA          STX    UIP      RESTORE IT
00457 4D1A 39       DDONE RTS             RETURN
00458 4D1B 96 2E    STOINS LDA A  T1      GET ARITHMETIC FLAG
00459 4D1D 27 34          BEQ    NOARI    0 IMPLIES NO ARITHMETIC
00460 4D1F DF 20          STX    TP7      SAVE REG ADRS
00461 4D21 CE 0078        LDX    #AT1     GET ARITH. TEMP. ADRS
00462 4D24 BD 73F3        JSR    TXRX     TRANSFER X TO TEMP.
00463 4D27 DE 20          LDX    TP7      GET REG ADRS
00464 4D29 96 2E          LDA A  T1       GET OPERATION FLAG
00465 4D2B 4A             DEC A           =1?
00466 4D2C 27 15          BEQ    REGADD   YES; DO ADDITION
00467 4D2E 4A             DEC A           =2?
00468 4D2F 27 0D          BEQ    REGSUB   YES; DO SUBTRACTION
00469 4D31 4A             DEC A           =3?
00470 4D32 27 05          BEQ    REGMLT   YES; DO MULTIPLICATION
00471 4D34 BD 7793        JSR    FPD      DIVISION BY DEFAULT
00472 4D37 20 0D          BRA    MDONE    GO WRAP IT UP!
00473 4D39 BD 7735  REGMLT JSR   FPM      MULTIPLY
00474 4D3C 20 08          BRA    MDONE    WRAP IT UP!
00475 4D3E BD 75F6  REGSUB JSR   FPS      SUBTRACT
00476 4D41 20 03          BRA    MDONE    WRAP IT UP!
00477 4D43 BD 75FC  REGADD JSR   FPA      ADD
00478 4D46 DE 20    MDONE LDX    TP7      GET REG ADRS
00479 4D48 BD 73F3        JSR    TXRX     STORE THE F.P. NUMBER
00480 4D4B CE 0078        LDX    #AT1     GET TEMP. ADRS
```

```
00481 4D4E BD 7438          JSR    TXXR     RESTORE X REG.
00482 4D51 20 8C            BRA    PCHEK    GO CHECK PRGM CNTR
00483 4D53 BD 73F3 NOARI    JSR    TXRX     STORE THE NUMBER
00484 4D56 20 87            BRA    PCHEK    GO CHECK PRGM CNTR
00485                       *
00486                       *THIS SUBROUTINE CONVERTS A NUMERIC REGISTER ADDRESS
00487                       *TO THE ACTUAL R/W ADDRESS OF THAT REGISTER. IT
00488                       *THEN CHECKS FOR VALIDITY GENERATING ERROR 7 IF
00489                       *INVALID AND CLEARING ACCA IF VALID. THE ADDRESS
00490                       *IS RETURNED IN THE INDEX REGISTER.
00491                       *ON ENTRY, ACCA=THE REGISTER ADDRESS
00492                       *
00493 4D58 5F       SADRS   CLR B            ZERO TO ACCB
00494 4D59 48               ASL A            MULTIPLY THE REGISTER
00495 4D5A 59               ROL B            ADDRESS BY 8 BY
00496 4D5B 48               ASL A            SHIFTING LEFT
00497 4D5C 59               ROL B            3 TIMES.
00498 4D5D 48               ASL A
00499 4D5E 59               ROL B
00500 4D5F 8B 08            ADD A  #8        ADD 8 TO THE RESULT
00501 4D61 C9 00            ADC B  #0        ADD CARRY TO MSB'S
00502 4D63 D7 1E            STA B  TP6       SAVE RESULT FOR SUBTRACT
00503 4D65 97 1F            STA A  TP6S
00504 4D67 DE 54            LDX    SPGM      GET REG START ADRS
00505 4D69 DF 20            STX    TP7       SAVE FOR SUBTRACT
00506 4D6B BD 4840          JSR    SUB7      DO EOM-REG ADRS
00507 4D6E DE 20            LDX    TP7       GET RESULT
00508 4D70 DF 1C            STX    TP5       SAVE FOR LATER
00509 4D72 DE 08            LDX    EOPM      GET END OF PRGM MEM
00510 4D74 DF 1E            STX    TP6       PREPARE TO SUBTRACT
00511 4D76 BD 4840          JSR    SUB7      DO REG ADRS-EOPM
00512 4D79 2A 04            BPL    SAD1      BRANCH IF NO ERROR
00513 4D7B 86 18            LDA A  #24       LOAD ERROR
00514 4D7D 20 0F            BRA    ERR+2     SAVE THE ERROR AND RETURN
00515 4D7F DE 1C    SAD1    LDX    TP5       INDEX=VALID ADDRESS
00516 4D81 4F               CLR A            NO ERROR; CLR ACCA
00517 4D82 39               RTS              RETURN
00518                       *
00519 4D83 D6 90    REGS    LDA B  XR        GET X-REG EXPONENT
00520 4D85 2B 0A            BMI    REGZ      ZERO REGS IF NEG
00521 4D87 5C               INC B            ACCB=# OF DIGITS
00522 4D88 C1 03            CMP B  #3        TOO MANY ?
00523 4D8A 2F 0B            BLE    OK        BRANCH IF NOT
00524 4D8C 86 07    ERR     LDA A  #7        LOAD ERROR
00525 4D8E 97 06            STA A  ERROR     SAVE IT
00526 4D90 39               RTS              RETURN
00527 4D91 DE 54    REGZ    LDX    SPGM      GET REG START ADRS
00528 4D93 DF 1C            STX    TP5       SAVE IT
00529 4D95 20 2A            BRA    OLD       GO ERASE BETWEEN PNTRS
00530 4D97 CE 0090 OK       LDX    #XR       GET X-REG ADDRESS
00531 4D9A BD 49CE          JSR    TSFR      CONVERT X-REG
00532 4D9D 96 20            LDA A  TP7       GET MS BITS OF RESULT
00533 4D9F D6 21            LDA B  TP7+1     GET LS BITS
00534 4DA1 58               ASL B            MULTIPLY RESULT BY 8
00535 4DA2 49               ROL A
00536 4DA3 58               ASL B
00537 4DA4 49               ROL A
00538 4DA5 58               ASL B
00539 4DA6 49               ROL A
00540 4DA7 97 1E            STA A  TP6       SAVE MS BITS
00541 4DA9 D7 1F            STA B  TP6+1     SAVE LS BITS
00542 4DAB DE 54            LDX    SPGM      GET REG START ADRS
00543 4DAD DF 20            STX    TP7       SAVE FOR SUBTRACT
00544 4DAF BD 4840          JSR    SUB7      SUBTRACT (EOM-REG ADRS)
00545 4DB2 2F D8            BLE    ERR       ERROR IF PAGE IS 0 OR NEG
00546 4DB4 DE 20            LDX    TP7       GET RESULT
00547 4DB6 DF 1C            STX    TP5       SAVE IT
00548 4DB8 DE 0B            LDX    EOPM      GET OLD PNTR
00549 4DBA DF 1E            STX    TP6       SAVE FOR SUBTRACT
00550 4DBC BD 4840          JSR    SUB7      DO NEW-OLD
00551 4DBF 2B 0C            BMI    NEW       BRANCH IF NEG RESULT
00552 4DC1 DE 0B    OLD     LDX    EOPM      GET OLD EOPM
00553 4DC3 DF 1E            STX    TP6       SAVE IT
00554 4DC5 DE 1C            LDX    TP5       GET NEW EOPM
00555 4DC7 DF 0B            STX    EOPM      SAVE IT
```

```
00556 4DC9 DE 1E          LDX     TP6       GET OLD EOPM
00557 4DCB 20 25          BRA     LOOP      ERASE
00558 4DCD DE 1C   NEW    LDX     TP5       GET NEW EOPM
00559 4DCF 8C 0100        CPX     #$100     USER STATE 0 ?
00560 4DD2 27 B8          BEQ     ERR       ERROR IF YES
00561 4DD4 CE 00CA        LDX     #UIP      GET INST PNTR ADRS
00562 4DD7 D6 09          LDA B   RSFLG     GET R/S FLAG
00563 4DD9 58             ASL B             RUNNING ?
00564 4DDA 2B 02          BMI     RUNING    BRANCH IF YES
00565 4DDC 09             DEX               ELSE GET PRGM PNTR ADRS
00566 4DDD 09             DEX
00567 4DDE A6 01   RUNING LDA A   1,X       DO PNTR-TP5
00568 4DE0 90 1D          SUB A   TP5S
00569 4DE2 A6 00          LDA A   X
00570 4DE4 92 1C          SBC A   TP5
00571 4DE6 2B 08          BMI     OK1       OK IF PNTR<TP5
00572 4DE8 5D             TST B             ELSE CHECK RUN/STOP
00573 4DE9 2B A1          BMI     ERR       ERROR IF RUNNING
00574 4DEB CE 0100        LDX     #256      ELSE SET UPP TO 0
00575 4DEE DF C8          STX     UPP
00576 4DF0 DE 1C   OK1    LDX     TP5       GET NEW EOPM
00577 4DF2 9C 08   LOOP   CPX     EOPM      ALL ERASED ?
00578 4DF4 27 05          BEQ     OUT       BRANCH IF YES
00579 4DF6 6F 00          CLR     X         CLEAR MEMORY
00580 4DF8 08             INX               INC PNTR
00581 4DF9 20 F7          BRA     LOOP      CONTINUE
00582 4DFB DE 1C   OUT    LDX     TP5       GET NEW EOPM
00583 4DFD DF 08          STX     EOPM      SAVE IT
00584 4DFF 09             DEX               BACK UP
00585 4E00 A6 00          LDA A   X         GET INST THERE
00586 4E02 81 8A          CMP A   #SDBB     TWO BYTE INSTRUCTION ?
00587 4E04 23 02          BLS     OUT2      BRANCH IF NOT
00588 4E06 6F 00          CLR     X         ELSE NOP TO LAST STATE
00589 4E08 39      OUT2   RTS               RETURN
00590              *
00591       4848   ADD7   EQU     $4848
00592       4840   SUB7   EQU     $4840
00593 7C70                ORG     $7C70
00594 7C70 4D93           FDB     REGS
00595       49CE   TSFR   EQU     $49CE
00596 7CC8                ORG     $7CC8
00597 7CC8 4C7C           FDB     STOENT,STOENT,STOENT,STOENT,STOENT
00598 7CD2 4C7C           FDB     STOENT,STOENT,STOENT,STOENT,STOENT
00599 7CDC 4C7B           FDB     ADDENT,ADDENT,ADDENT,ADDENT,ADDENT
00600 7CE6 4C7B           FDB     ADDENT,ADDENT,ADDENT,ADDENT,ADDENT
00601 7CF0 4C7A           FDB     SUBENT,SUBENT,SUBENT,SUBENT,SUBENT
00602 7CFA 4C7A           FDB     SUBENT,SUBENT,SUBENT,SUBENT,SUBENT
00603 7D04 4C79           FDB     MLTENT,MLTENT,MLTENT,MLTENT,MLTENT
00604 7D0E 4C79           FDB     MLTENT,MLTENT,MLTENT,MLTENT,MLTENT
00605 7D18 4C78           FDB     DVDENT,DVDENT,DVDENT,DVDENT,DVDENT
00606 7D22 4C78           FDB     DVDENT,DVDENT,DVDENT,DVDENT,DVDENT
00607 7D2C 4C81           FDB     RCLENT,RCLENT,RCLENT,RCLENT,RCLENT
00608 7D36 4C81           FDB     RCLENT,RCLENT,RCLENT,RCLENT,RCLENT
00609 7D84                ORG     $7D84
00610 7D84 4C81           FDB     RCLENT
00611 7D86 4C7C           FDB     STOENT
00612 7D88 4C7B           FDB     ADDENT
00613 7D8A 4C7A           FDB     SUBENT
00614 7D8C 4C79           FDB     MLTENT
00615 7D8E 4C78           FDB     DVDENT
00616 7D90 4C7C           FDB     STOENT,STOENT
00617 7D94 4C7B           FDB     ADDENT,ADDENT
00618 7D98 4C7A           FDB     SUBENT,SUBENT
00619 7D9C 4C79           FDB     MLTENT,MLTENT
00620 7DA0 4C78           FDB     DVDENT,DVDENT
00621 7DA4 4C81           FDB     RCLENT,RCLENT
00622 7DA8 4C81           FDB     RCLENT,RCLENT
00623 7DAC 4C7C           FDB     STOENT,STOENT
00624 7DB0 4C7B           FDB     ADDENT,ADDENT
00625 7DB4 4C7A           FDB     SUBENT,SUBENT
00626 7DB8 4C79           FDB     MLTENT,MLTENT
00627 7DBC 4C78           FDB     DVDENT,DVDENT
00628 7D7E                ORG     $7D7E
00629 7D7E 4D15           FDB     LBLEX
00632                     END
```

SYMBOL TABLE

ADATA	0000	ACTL	0001	BDATA	0002	BCTL	0003	INPUT	0004	IOIN	0005
ERROR	0006	TGL	0007	UFLG	0008	RSFLG	0009	EOM	000A	EOPM	000B
STKFLG	000D	RND	000E	DIGFLG	000F	W2	0010	W1	0011	SFLG	0012
DCNTR	0013	TP1	0014	TP1S	0015	TP2	0016	TP2S	0017	TP3	0018
TP3S	0019	TP4	001A	TP4S	001B	TP5	001C	TP5S	001D	TP6	001E
TP6S	001F	TP7	0020	TP7S	0021	T13	0022	T12	0023	T11	0024
T10	0025	T9	0026	T8	0027	T7	0028	T6	0029	T5	002A
T4	002B	T3	002C	T2	002D	T1	002E	ISTK	002F	ISTACK	0051
TA	0052	SPGM	0054	EXTRA	0056	BUFF	0058	REAL	0068	IMAG	0070
AT1	0078	AT2	0080	W	0088	XR	0090	YR	0098	ZR	00A0
TR	00A8	LSTX	00B0	BKWRT	00B8	BKKC	00BA	SOL7	00C6	UPP	00C8
UIP	00CA	ALPHA	00CC	IO1	00CD	IO2	00D0	IT7	00D3	FLAG	00D5
TPOS	00D6	FILE	00D7	AR	00D8	BR	00E0	CR	00E8	DR	00F0
ER	00F8	SDBB	00BA	MT	7E00	TERMN7	003D	IMED	0040	PARCD	00C0
PAREX	0080	NTBL	0000	DOTS	5EC0	PRTDRV	6020	FRMT	5CA8	BLANK	5D75
LDMSG	57BD	ROLLD	55B2	ROLLU	57F1	PSD	55DA	TXL	55E9	STKUP	55EF
MAD	749B	CMP	74AA	NOR	74D6	TXW	7424	TXXR	7438	EXXR	7452
ARSR	753B	OVUNF	75B6	OVERF	75DD	XR0	740A	XRNINE	75C8	UNDRF	75F1
IMULT	76B9	ODG	7669	FPA	75FC	FPS	75F6	FPM	7735	FPD	7793
FPAEX	763D	FPMEX	7780	LSHIFT	7521	ZEROX	7489	XZEROQ	7416	XZERO2	7417
RECIP	73E6	TXRX	73F3	CONST	6800	FPDBRC	6898	TAN	68A9	ATN	69C3
DSZERO	6A46	NTLN	6A58	EXPN	6AC9	SIN	6894	COS	689A	ASIN	68F2
ACOS	68F7	PH1	6C5D	PH2	6C8D	PH3	6D34	PH4	6DD0	LSFT8	6E47
SQRT	6E65	MAD8	6F2C	CMP8	53E4	IOUPX	6F52	LOG10	6FA7	YUPX	6FE9
RTOP	7328	PTOR	7386	IF0	48A4	NMET	48A8	MET	48B4	IFPLS	48B5
IFMI	48BA	IFXEY	48BF	IFXGEY	48D8	IFXLTY	48D9	MET1	48F7	NMET1	48FC
NEXTC	4C01	NEXTB	4C0D	NEXTA	4C14	NEXTI	4C19	NEXTJ	4C1E	MOREL	4C62
MOREL1	4C66	FEROR	4C73	FORS	4C77	FP1	6838	DVDENT	4C78	MLTENT	4C79
SUBENT	4C7A	ADDENT	4C7B	STOENT	4C7C	RCLENT	4C81	RENT	4C84	TWOBT	4CA4
IADRS	4CA8	USMEM	4CB3	GEN	4CBE	NUMRIC	4CC9	VALID	4CD3	INDR	4CE1
REGO	4CF4	OVFLO	4CF9	INSTYP	4CFE	PCHEK	4D0F	LBLEX	4D15	DDONE	4D1A
STOINS	4D1B	REGMLT	4D39	REGSUB	4D3E	REGADD	4D43	MDONE	4D46	NOARI	4D53
SADRS	4D58	SAD1	4D7F	REGS	4D83	ERR	4D8C	REGZ	4D91	OK	4D97
OLD	4DC1	NEW	4DCD	RUNING	4DDE	OK1	4DF0	LOOP	4DF2	OUT	4DFB
OUT2	4E08	ADD7	4848	SUB7	4840	TSFR	49CE				

TOTAL ERRORS 2

****ERROR 201
218 NAM EDIT
****ERROR 201
472 NAM STEP

```
00215                          OPT    LIST,MEM
00218                          NAM    EDIT
00219                          OPT    LIST,MEM
00220 4EC8                     ORG    $4EC8
00221                *
00222                *THIS ROUTINE CONTAINS ALL CODE FOR ALL EDITTING
00223                *FUNCTIONS ON CJ.
00224                *
00225       4840     SUB7   EQU    $4840
00226       4848     ADD7   EQU    $4848
00227       7978     BWM    EQU    $7978
00228       7B40     ERROR7 EQU    $7B40
00229       0080     T8     EQU    AT2
00230       0082     AFFECT EQU    AT2+2
00231       0084     FIELD  EQU    AT2+4
00232 4EC8 86 16     STCOD7 LDA A  #22       PRESET ACCA FOR POSSIBLE ERROR
00233 4ECA D6 D5            LDA B  FLAG       GET "MEMORY SECURE" INFO
00234 4ECC 54               LSR B             MOVE SECURE BIT TO CARRY
00235 4ECD 25 11            BCS    STC10      BRANCH IF MEMORY SECURED
00236 4ECF DE C8            LDX    UPP        ELSE LOAD PRGM PNTR
00237 4ED1 96 C7            LDA A  SOL7+1     GET INST TO BE STORED
00238 4ED3 16               TAB               SAVE IT IN ACCB
00239 4ED4 2B 0E            BMI    STC1       BRANCH IF ONE BYTE INST
00240 4ED6 08               INX               ELSE BUMP PRGM PNTR
00241 4ED7 9C 0B            CPX    EOPM       END OF MEMORY ?
00242 4ED9 26 08            BNE    STC0       BRANCH IF NOT
00243 4EDB 7F 00CC          CLR    ALPHA      RESET POSSIBLE ALPHA
00244 4EDE 86 07     STC7   LDA A  #7         LOAD MEM OVFLO ERROR
```

```
00245 4EE0 97 06   STC10  STA A  ERROR    SAVE THE ERROR NUMBER
00246 4EE2 39             RTS             RETURN
00247 4EE3 09      STC0   DEX             BACK UP
00248 4EE4 96 52   STC1   LDA A  TA       INSERT MODE SET ?
00249 4EE6 27 2E          BEQ    STC2     BRANCH IF NOT
00250 4EE8 5D             TST B           TWO BYTE INST ?
00251 4EE9 2B 03          BMI    STC3     BRANCH IF NOT
00252 4EEB BD 4FC7        JSR    INSRT    DROP MEMORY
00253 4EEE BD 4FC7 STC3   JSR    INSRT    DROP MEMORY
00254 4EF1 DE C8   STC9   LDX    UPP      GET INST PNTR
00255 4EF3 96 C6          LDA A  SOL7     GET INST
00256 4EF5 A7 00          STA A  X        SAVE FIRST BYTE
00257 4EF7 96 C7          LDA A  SOL7+1   GET 2ND BYTE
00258 4EF9 2B 02          BMI    STC4     BRANCH IF NO 2ND BYTE
00259 4EFB A7 01          STA A  1,X      ELSE STORE 2ND BYTE
00260 4EFD 4F      STC4   CLR A           CLEAR ACCA FOR KEYLOG
00261 4EFE DE C8          LDX    UPP      GET PRGM PNTR
00262 4F00 D6 07          LDA B  TGL      GET TOGGLES
00263 4F02 C5 60          BIT B  #$60     KEYLOG MODE SET ?
00264 4F04 27 03          BEQ    STC5     BRANCH IF NOT
00265 4F06 7E 590A        JMP    KEYLOG   DO KEYLOG AND RETURN
00266 4F09 D6 C7   STC5   LDA B  SOL7+1   CHECK THE CURRENT INST.
00267 4F0B 2B 01          BMI    STC6     BRANCH IF ONE BYTE
00268 4F0D 08             INX             ELSE BUMP PNTR
00269 4F0E 08      STC6   INX             BUMP PNTR
00270 4F0F 9C 0B          CPX    EOPM     END OF MEMORY ?
00271 4F11 27 C8          BEQ    STC7     BRANCH IF YES
00272 4F13 DF C8          STX    UPP      ELSE SAVE NEW PRGM POINTER
00273 4F15 39             RTS             RETURN
00274 4F16 5D      STC2   TST B           2-BYTE INST ?
00275 4F17 2A 12          BPL    STC8     BRANCH IF YES
00276 4F19 A6 00          LDA A  X        ELSE CHECK INST IN MEMORY
00277 4F1B 81 BA          CMP A  #SDBB    TWO BYTE INSTRUCTION ?
00278 4F1D 23 D2          BLS    STC9     BRANCH IF YES
00279 4F1F DF 82          STX    AFFECT   ELSE SAVE DELETE PNTR
00280 4F21 BD 4FEC        JSR    DEL      RAISE MEMORY
00281 4F24 CE FFFF        LDX    #-1      PRESET INDEX
00282 4F27 8D 67          BSR    DMOD0    MODIFY THE ADDRESSES
00283 4F29 20 C6          BRA    STC9     CONTINUE
00284 4F2B A6 00   STC8   LDA A  X        GET CURRENT INST IN MEMORY
00285 4F2D 81 BA          CMP A  #SDBB    TWO BYTE INSTRUCTION ?
00286 4F2F 22 C0          BHI    STC9     BRANCH IF YES
00287 4F31 A6 01          LDA A  1,X      ELSE GET FOLLOWING INST
00288 4F33 27 BC          BEQ    STC9     BRANCH IF IT IS A NOP
00289 4F35 BD 4FC7        JSR    INSRT    ELSE DROP MEMORY
00290 4F38 96 C6          LDA A  SOL7     GET INST TO BE STORED
00291 4F3A A7 00          STA A  X        STORE IT
00292 4F3C 96 C7          LDA A  SOL7+1   GET 2ND BYTE
00293 4F3E A7 01          STA A  1,X      STORE IT
00294 4F40 08             INX             PRESET INDEX
00295 4F41 DF 52          STX    TA       SAVE "INSERT START"
00296 4F43 08             INX             GET "INSERT END"
00297 4F44 DF 80          STX    TB       SAVE IT ALSO
00298 4F46 BD 500C        JSR    AMOD     MODIFY THE ADDRESSES
00299 4F49 20 B2          BRA    STC4     CONTINUE
00300              *
00301              *
00302 4F4B D6 05   INSERT LDA B  FLAG     PROTECTED MEMORY ?
00303 4F4D 54             LSR B
00304 4F4E 25 08          BCS    BACK     RETURN IF YES
00305 4F50 96 52          LDA A  TA       INSERTING ?
00306 4F52 26 07          BNE    TERM     TERMINATE IF SET
00307 4F54 DE C8          LDX    UPP      GET CURRENT USER ADRS
00308 4F56 DF 52          STX    TA       SAVE FOR INSERT START
00309 4F58 7E 7978 BACK   JMP    BWM      RETURN TO SUPV
00310 4F5B BD 52F4 TERM   JSR    STEP1    TERMINATE THE INSERT
00311 4F5E 20 F8          BRA    BACK     RETURN
00312              *
00313              *
00314 4F60 D6 05   DELETE LDA B  FLAG     PROTECTED MEMORY ?
00315 4F62 54             LSR B
00316 4F63 25 F3          BCS    BACK     RETURN IF YES
00317 4F65 96 52          LDA A  TA       INSERTING ?
00318 4F67 26 EF          BNE    BACK     IGNORE IF YES
00319 4F69 DE C8          LDX    UPP      GET PRGM PNTR
00320 4F6B DF 82          STX    AFFECT   SAVE CURRENT ADRS
```

```
00321 4F6D A6 00          LDA A  X        GET INST
00322 4F6F 97 2E          STA A  T1       SAVE CURRENT INST
00323 4F71 81 BA          CMP A  #SDBB    TWO BYTE INSTRUCTION ?
00324 4F73 23 03          BLS    OBYTE    BRANCH IF NOT
00325 4F75 BD 4FEC        JSR    DEL      RAISE MEMORY
00326 4F78 BD 4FEC OBYTE  JSR    DEL      RAISE MEMORY
00327 4F7B CE FFFF        LDX    #-1      INITIALIZE INDEX
00328 4F7E 96 2E          LDA A  T1       GET INST DELETED
00329 4F80 81 BA          CMP A  #SDBB    TWO BYTE INSTRUCTION ?
00330 4F82 23 01          BLS    SBT      BRANCH IF NOT
00331 4F84 09             DEX             2 STATES DELETED
00332 4F85 8D 09    SBT   BSR    DMOD0    MODIFY THE ADDRESSES
00333 4F87 BD 4EFD        JSR    STC4     GO CHECK FOR KEYLOG
00334 4F8A DE 82          LDX    AFFECT   GET OLD ADRS (KEYLOG MOVED IT)
00335 4F8C DF C8          STX    UPP      RESTORE IT
00336 4F8E 20 C8    SBT1  BRA    BACK     RETURN
00337 4F90 DF 1E    DMOD0 STX    TP6      SAVE INDEX FOR MODIFICATION
00338 4F92 CE 00FE        LDX    #254     PRESET INDEX FOR SEARCH
00339 4F95 08       DMOD4 INX             SKIP AHEAD ONE
00340 4F96 08       DMOD1 INX             SKIP AHEAD ONE
00341 4F97 9C 0B          CPX    EOPM     FINISHED YET ?
00342 4F99 26 01          BNE    DMOD8    BRANCH IF NOT
00343 4F9B 39             RTS             ELSE RETURN
00344 4F9C A6 00    DMOD8 LDA A  X        GET NEXT INST
00345 4F9E 81 DF          CMP A  #$DF     ELSE CHECK FOR JMP OR JSR
00346 4FA0 23 F4          BLS    DMOD1    BRANCH IF NOT
00347 4FA2 8D 57          BSR    UNPACK   UNPACK AND CHECK THE ADRS
00348 4FA4 2B EF          BMI    DMOD4    BRANCH IF NO MOD. NECESSARY
00349 4FA6 26 03          BNE    DMOD5    NON-ZERO: GO MODIFY
00350 4FA8 5D             TST B          ELSE TEST LS BITS
00351 4FA9 27 EA          BEQ    DMOD4    NO MOD. IF ZERO
00352 4FAB BD 4848  DMOD5 JSR    ADD7     MODIFY THE OLD ADDRESS
00353 4FAE 2B 13          BMI    DMOD6    BRANCH IF NEG RESULT
00354 4FB0 C6 F0          LDA B  #$F0     GET INST MASK
00355 4FB2 E4 00          AND B  X        REMOVE OLD INST ADRS
00356 4FB4 E7 00          STA B  X        RESTORE OLD INST
00357 4FB6 D6 21          LDA B  TP7S     ACCB=LS BITS OF NEW ADRS
00358 4FB8 58       DMOD7 ASL B           PACK THE NEW ADRS
00359 4FB9 49             ROL A
00360 4FBA 54             LSR B
00361 4FBB E7 01          STA B  1,X      RESTORE NEW LS BITS
00362 4FBD AA 00          ORA A  X        "OR" IN NEW MS BITS
00363 4FBF A7 00          STA A  X        RESTORE NEW MS BITS
00364 4FC1 20 D2          BRA    DMOD4    CONTINUE
00365 4FC3 4F       DMOD6 CLR A           ACCA=ACCB=MIN ADRS
00366 4FC4 5F             CLR B
00367 4FC5 20 F1          BRA    DMOD7    CONTINUE
00368                *
00369 4FC7 DE C8    INSRT LDX    UPP      GET PRGM PNTR
00370 4FC9 A6 00          LDA A  X        GET THE INST
00371 4FCB 08       ILOOP INX             INC POINTER
00372 4FCC 9C 0B          CPX    EOPM     END OF MEMORY ?
00373 4FCE 27 07          BEQ    IDONE    BRANCH IF YES
00374 4FD0 E6 00          LDA B  X        GET CURRENT INST
00375 4FD2 A7 00          STA A  X        SAVE OLD INST
00376 4FD4 17             TBA             TRANSFER NEW INST
00377 4FD5 20 F4          BRA    ILOOP    CONTINUE
00378 4FD7 09       IDONE DEX             BACK UP ONE
00379 4FD8 A6 00          LDA A  X        GET THE INST
00380 4FDA 81 BA          CMP A  #SDBB    TWO BYTE INSTRUCTION ?
00381 4FDC 23 04          BLS    RTN1     BRANCH IF NOT
00382 4FDE 6F 00          CLR    X        ELSE WIPE IT OUT
00383 4FE0 20 03          BRA    ID1      SET ERROR
00384 4FE2 5D       RTN1  TST B           INSTRUCTION SHOVED OFF ?
00385 4FE3 27 04          BEQ    RTN      BRANCH IF NOT
00386 4FE5 86 07    ID1   LDA A  #7       LOAD ERROR
00387 4FE7 97 06          STA A  ERROR    SAVE IT
00388 4FE9 DE C8    RTN   LDX    UPP      RELOAD PRGM PNTR
00389 4FEB 39             RTS             RETURN
00390                *
00391 4FEC DE C8    DEL   LDX    UPP      GET PRGM PNTR
00392 4FEE A6 01    DLOOP LDA A  1,X      GET NEW INST
00393 4FF0 A7 00          STA A  X        SAVE IT IN NEW LOCATION
00394 4FF2 08             INX             INC MEMORY PNTR
00395 4FF3 9C 0B          CPX    EOPM     END OF MEMORY ?
00396 4FF5 26 F7          BNE    DLOOP    BRANCH IF NOT
```

```
00397 4FF7 09                DEX              ELSE BACK UP ONE
00398 4FF8 6F 00             CLR    X         NOP TO LAST STATE
00399 4FFA 39                RTS              RETURN
00400                        *
00401                        *UNPACK AND CHECK THE GOTO/GOSUB ADDRESS
00402                        *
00403 4FFB E6 01    UNPACK   LDA B  1,X       GET 2ND BYTE OF INST
00404 4FFD 58                ASL B            EXTRACT THE ADDRESS
00405 4FFE 46                ROR A            ACCA=MS BITS
00406 4FFF 56                ROR B            ACCB=LS BITS
00407 5000 84 07             AND A  #$7       MASK OFF THE INST. CODE
00408 5002 97 20             STA A  TP7       SAVE MS BITS
00409 5004 D7 21             STA B  TP7S      SAVE LS BITS
00410 5006 4C                INC A            ADD BASE PAGE OFFSET
00411 5007 D0 83             SUB B  AFFECT+1    ADRS MODIFICATION NECESSARY ?
00412 5009 92 82             SBC A  AFFECT
00413 500B 39                RTS
00414                        *
00415                        *
00416                        *
00417 500C DE 52    AMOD     LDX    TA        GET INSERT START
00418 500E DF 82             STX    AFFECT    SAVE FOR "AFFECTED" TEST
00419 5010 DF 84             STX    FIELD     SAVE FOR "FIELD" TEST
00420 5012 DF 1E             STX    TP6       SAVE FOR "B-A"
00421 5014 DE 80             LDX    TB        GET INSERT END
00422 5016 DF 20             STX    TP7       SAVE FOR "B-A"
00423 5018 BD 4840           JSR    SUB7      DO "B-A" TO TP7
00424 501B DE 20             LDX    TP7       GET THE RESULT
00425 501D 26 06             BNE    AMOD1     BRANCH IF NON-ZERO
00426 501F 7F 0052  AMOD4    CLR    TA        RESET ALPHA FLAG/POINTER
00427 5022 DE C8             LDX    UPP       RESTORE INDEX FOR STEP ROUTINE
00428 5024 39                RTS              RETURN
00429 5025 DF 1E    AMOD1    STX    TP6       SAVE RESULT FOR MODIFICATIONS
00430 5027 CE 00FF           LDX    #255      GET STARTING ADRS-1
00431 502A 08       AMOD0    INX              INC MEMORY PNTR
00432 502B 9C 84             CPX    FIELD     FIELD BOUNDARY ?
00433 502D 26 1C             BNE    AMOD2     BRANCH IF NOT
00434 502F 9C 52             CPX    TA        "A" BOUNDARY ?
00435 5031 26 0A             BNE    AMOD3     BRANCH IF NOT
00436 5033 DE 80             LDX    TB        ELSE GET "B" POINTER
00437 5035 DF 84             STX    FIELD     SAVE FOR NEXT FIELD TEST
00438 5037 DF 82             STX    AFFECT    SAVE FOR "AFFECTED" TEST
00439 5039 DE 52             LDX    TA        RESTORE USER ADRS
00440 503B 20 0E             BRA    AMOD2     CONTINUE
00441 503D 9C 08    AMOD3    CPX    EOPM      FINISHED ?
00442 503F 27 DE             BEQ    AMOD4     BRANCH IF YES
00443 5041 DE 0B             LDX    EOPM      ELSE GET NEXT FIELD TEST
00444 5043 DF 84             STX    FIELD     SAVE IT
00445 5045 DE 52             LDX    TA        GET NEW "AFFECTED" VALUE
00446 5047 DF 82             STX    AFFECT    SAVE IT
00447 5049 DE 80             LDX    TB        RESTORE WORK ADRS
00448 504B A6 00    AMOD2    LDA A  X         GET THE USER INST
00449 504D 81 DF             CMP A  #$DF      ELSE CHECK FOR JMP OR JSR
00450 504F 23 D9             BLS    AMOD0     BRANCH IF IT IS NOT
00451 5051 8D A8             BSR    UNPACK    UNPACK AND CHECK THE ADRS
00452 5053 2B D5             BMI    AMOD0     BRANCH IF NOT
00453 5055 BD 484B           JSR    ADD7      ELSE MODIFY THE ADDRESS
00454 5058 85 F8             BIT A  #$F8      NEW ADRS > 2047 ?
00455 505A 26 13             BNE    AMOD8     BRANCH IF YES
00456 505C C6 F0             LDA B  #$F0      ELSE GET INST MASK
00457 505E E4 00             AND B  X         AND IT WITH THE INST
00458 5060 E7 00             STA B  X         RESTORE THE INST
00459 5062 D6 21             LDA B  TP7S      ACCB=LS BITS OF NEW ADRS
00460 5064 58       AMOD10   ASL B            PACK THE NEW ADDRESS
00461 5065 49                ROL A
00462 5066 54                LSR B
00463 5067 E7 01             STA B  1,X       SAVE NEW LS BITS
00464 5069 AA 00             ORA A  X         "OR" IN NEW MS BITS
00465 506B A7 00             STA A  X         RESTORE INST
00466 506D 20 BB             BRA    AMOD0     CONTINUE
00467 506F 86 07    AMOD8    LDA A  #$7       ACCA & ACCB= MAX ADRS
00468 5071 C6 FF             LDA B  #$FF
00469 5073 20 EF             BRA    AMOD10    CONTINUE
00472                        NAM    STEP
00473                        OPT    LIST,MEM
00474 527A                   ORG    $527A
00475                        *
```

```
00476                *THIS ROUTINE HANDLES THE STEP AND BACK STEP
00477                *FUNCTIONS IN BOTH RUN AND PROGRAM MODE.
00478                *
00479                *IN RUN MODE, STEP SIMPLY EXECUTES ONE USER INSTRUCTION.
00480                *
00481                *IN PROGRAM MODE, STEP WILL 1) STEP OVER THE CURRENT
00482                *INSTRUCTION INDICATED BY UPP 2) LIST THAT INST
00483                *IF TRACE MODE IS SET 3) ENTER THE ALPHA MODE IF
00484                *THE INSTRUCTION WAS AN ALPHA INITIATOR.
00485                *
00486                *BACK STEP OPERATES ONLY IN PROGRAM MODE AND
00487                *WILL 1) BACK STEP OVER THE PREVIOUS INSTRUCTION
00488                *2) EXIT THE ALPHA MODE IF THAT INSTRUCTION
00489                *WAS AN ALPHA INITIATOR.
00490                *
00491                *BOTH STEP AND BACK STEP WILL TERMINATE THE
00492                *INSERT MODE IF IT IS SET WHEN THEY ARE CALLED
00493                *
00494                *WRITTEN BY BRAD MILLER
00495                *
00496 527A 96 D5    STEP   LDA A  FLAG     GET PROTECT FLAG
00497 527C 44               LSR A          GET PROTECT BIT
00498 527D 25 2B            BCS    OUT1    RETURN IF PROTECTED
00499 527F 8D 73            BSR    STEP1   CHECK FOR INSERT
00500 5281 D6 07            LDA B  TGL     GET MODE
00501 5283 2B 12            BMI    PGMD    BRANCH IF PROGRAM MODE
00502 5285 DF CA            STX    UIP     SAVE IN INST PNTR
00503 5287 86 C0            LDA A  #$C0    GET R/S FLAG INFO
00504 5289 97 12            STA A  SFLG    SET THE STEP FLAG
00505 528B E6 00            LDA B  X       GET CURRENT USER INST
00506 528D C1 B0            CMP B  #$B0    STOP COMMAND ?
00507 528F 27 01            BEQ    STP8    BRANCH IF YES
00508 5291 44               LSR A          ELSE RSFLG = $40
00509 5292 97 09    STP8   STA A  RSFLG    SAVE THE FLAG
00510 5294 7E 7968          JMP    EXEC7   GO DO IT !
00511 5297 C5 60    PGMD   BIT B  #$60     KEYLOG MODE SET ?
00512 5299 27 12            BEQ    NKEYLG  BRANCH IF NOT
00513 529B 4F               CLR A          PRESET ACCA FOR KEYLOG
00514 529C BD 590A          JSR    KEYLOG  DO THE KEYLOG AND STEP
00515 529F 96 06            LDA A  ERROR   ERROR ?
00516 52A1 26 07            BNE    OUT1    RETURN IF YES
00517 52A3 96 CC    STP6   LDA A  ALPHA    ALPHA MODE ?
00518 52A5 27 03            BEQ    OUT1    BRANCH IF NOT
00519 52A7 7E 519C          JMP    TOALP   ELSE DO "TO ALPHA"
00520 52AA 7E 7978 OUT1     JMP    BWM     RETURN TO SUPV
00521 52AD A6 00    NKEYLG LDA A  X       GET CURRENT INSTRUCTION
00522 52AF 81 BA            CMP A  #$DBB   TWO BYTE INSTRUCTION ?
00523 52B1 23 0D            BLS    STP3    BRANCH IF NOT
00524 52B3 81 BE            CMP A  #$BE    FORMAT INSTRUCTION ?
00525 52B5 26 08            BNE    STP4    BRANCH IF NOT
00526 52B7 E6 01            LDA B  1,X     ELSE GET 2ND BYTE
00527 52B9 C5 0F            BIT B  #$F     ALPHA TYPE FORMAT INST ?
00528 52BB 26 02            BNE    STP4    BRANCH IF NOT
00529 52BD 97 CC            STA A  ALPHA   ELSE SET ALPHA FLAG
00530 52BF 08       STP4   INX            MOVE AHEAD ONE
00531 52C0 08       STP3   INX            MOVE AHEAD ONE
00532 52C1 81 B4            CMP A  #$B4    ALPHA TERMINATOR ?
00533 52C3 27 2A            BEQ    RSTP4   BRANCH IF YES AND RESET ALPHA
00534 52C5 4F               CLR A
00535 52C6 9C 0B    STP7   CPX    EOPM    END OF PROGRAM MEMORY ?
00536 52C8 27 10            BEQ    BSTP3   BRANCH IF YES
00537 52CA DF C8    STP5   STX    UPP     ELSE SAVE NEW PRGM ADRS
00538 52CC 20 D5            BRA    STP6    GO FINISH UP !
00539                *
00540                *BACK STEP BEGINS HERE
00541                *
00542 52CE 8D 24    BSTEP  BSR    STEP1   CHECK FOR INSERT
00543 52D0 09               DEX            BACK UP ONE
00544 52D1 09               DEX            BACK UP ONE
00545 52D2 DF 14            STX    TP1     PRGM PNTR TO TEMP FOR CHECK
00546 52D4 96 14            LDA A  TP1     BACKED BEHIND ADRS ZERO ?
00547 52D6 26 06            BNE    BSTP1   BRANCH IF NOT
00548 52D8 97 C9            STA A  UPP+1   ELSE SET UPP TO STATE ZERO
00549 52DA 97 CC    BSTP3  STA A  ALPHA    CLEAR THE ALPHA MODE
00550 52DC 20 C5            BRA    STP6    GET OUT !
```

```
00551 52DE A6 00    BSTP1   LDA A   X       GET THE INSTRUCTION
00552 52E0 81 BA            CMP A   #$DBB   TWO BYTE INSTRUCTION ?
00553 52E2 22 01            BHI     BSTP2   BRANCH IF YES (UPP OK)
00554 52E4 08               INX             ELSE MOVE AHEAD ONE
00555 52E5 81 BE    BSTP2   CMP A   #$BE    FORMAT INSTRUCTION ?
00556 52E7 26 E1            BNE     STP5    BRANCH IF NOT
00557 52E9 E6 01            LDA B   1,X     ELSE GET 2ND BYTE
00558 52EB C5 0F            BIT B   #$F     ALPHA INST ?
00559 52ED 26 DB            BNE     STP5    BRANCH IF NOT
00560 52EF 4F       BSTP4   CLR A
00561 52F0 97 CC            STA A   ALPHA   CLEAR ALPHA MODE
00562 52F2 20 D2            BRA     STP7    GET OUT !
00563                *
00564                *
00565                *
00566 52F4 DE C8    STEP1   LDX     UPP     GET CURRENT PRGM PNTR
00567 52F6 96 52            LDA A   TA      ARE WE IN THE INSERT MODE ?
00568 52F8 26 01            BNE     STEP2   BRANCH IF YES
00569 52FA 39               RTS             ELSE RETURN
00570 52FB DF 80    STEP2   STX     TB      SAVE FOR "INSERT END"
00571 52FD 7E 500C          JMP     AMOD    MOD. GOTO'S AND RET. UPP IN INDEX
00572                *
00573                *
00574       796B    EXEC7   EQU     $796B
00575       519C    TOALP   EQU     $519C
00576       59DA    KEYLOG  EQU     $59DA
00579               END
```

SYMBOL TABLE

```
ADATA  0000  ACTL   0001  BDATA  0002  BCTL   0003  INPUT  0004  IOIN   0005
ERROR  0006  TGL    0007  UFLG   0008  RSFLG  0009  FOM    000A  EOPM   000B
STKFLG 000D  RND    000E  DIGFLG 000F  W2     0010  W1     0011  SFLG   0012
DCNTR  0013  TP1    0014  TP1S   0015  TP2    0016  TP2S   0017  TP3    0018
TP3S   0019  TP4    001A  TP4S   001B  TP5    001C  TP5S   001D  TP6    001E
TP6S   001F  TP7    0020  TP7S   0021  T13    0022  T12    0023  T11    0024
T10    0025  T9     0026  T8     0027  T7     0028  T6     0029  T5     002A
T4     002B  T3     002C  T2     002D  T1     002E  ISTK   002F  ISTACK 0051
TA     0052  SPGM   0054  EXTRA  0056  BUFF   0058  REAL   0068  IMAG   0070
AT1    0078  AT2    0080  W      0088  XR     0090  YR     0098  ZR     00A0
TR     00A8  LSTX   00B0  BKWRT  00B8  BKKC   00BA  SOL7   00C6  UPP    00C8
UIP    00CA  ALPHA  00CC  IO1    00CD  IO2    00D0  IT7    00D3  FLAG   00D5
TPOS   00D6  FILE   00D7  AR     00D8  BR     00E0  CR     00E8  DR     00F0
EM     00F8  SDBB   00BA  MT     7E00  TERMN7 003D  IMED   0040  PARCD  00C0
PAREX  0080  NTBL   0000  DOTS   5EC0  PRTDRV 602D  FRMT   5CA8  BLANK  5D75
LDMSG  57BD  ROLLD  55B2  ROLLU  57F1  PSD    55DA  TXL    55E9  STKUP  55EF
MAD    7498  CMP    74AA  NOR    74D6  TXW    7424  TXXR   743B  EXXR   7452
ARSR   753B  OVUNF  75B6  OVERF  75DD  XR0    740A  XRNINE 75C8  UNDRF  75F1
IMULT  76B9  QDG    7669  FPA    75FC  FPS    75F6  FPM    7735  FPD    7793
FPAEX  763D  FPMEX  7780  LSHIFT 7521  ZEROX  7489  XZEROQ 7416  XZERO2 7417
RECIP  73E6  TXPX   73F3  CONST  6800  FPDBRC 6B9B  TAN    68A9  ATN    69C3
DSZERO 6A46  NTLN   6A58  EXPN   6AC9  SIN    6B94  COS    689A  ASIN   6BF2
ACOS   6BF7  PH1    6C5D  PH2    6C8D  PH3    6D34  PH4    6DD0  LSFT8  6E47
SQRT   6E65  MAD8   6F2C  CMP8   53E4  IOUPX  6F52  LOG10  6FA7  YUPX   6FE9
RTOP   7328  PTOR   7386  SUB7   4840  ADD7   484B  BWM    797B  ERROR7 7840
T8     0080  AFFECT 0082  FIELD  0084  STCOD7 4EC8  STC7   4EDE  STC10  4EE0
STC0   4EE3  STC1   4EE4  STC3   4EEE  STC9   4EF1  STC4   4EFD  STC5   4F09
STC6   4F0E  STC2   4F16  STC8   4F2B  INSERT 4F4B  BACK   4F58  TERM   4F5B
DELETE 4F60  OBYTE  4F78  SBT    4F85  SBT1   4F8E  DMOD0  4F90  DMOD4  4F95
DMOD1  4F96  DMOD8  4F9C  DMOD5  4FAB  DMOD7  4FB8  DMOD6  4FC3  INSRT  4FC7
ILOOP  4FCB  IDONE  4FD7  RTN1   4FE2  ID1    4FE5  RTN    4FE9  DEL    4FEC
DLOOP  4FEE  UNPACK 4FFB  AMOD   500C  AMOD4  501F  AMOD1  5025  AMOD0  502A
AMOD3  503D  AMOD2  504B  AMOD10 5064  AMOD8  506F  STEP   527A  STP8   5292
PGMD   5297  STP6   52A3  OUT1   52AA  NKEYLG 52AD  STP4   52BF  STP3   52C0
STP7   52C6  STP5   52CA  BSTEP  52CE  BSTP3  52DA  BSTP1  52DE  BSTP2  52E5
BSTP4  52EF  STEP1  52F4  STEP2  52FB  EXEC7  7968  TOALP  519C  KEYLOG 59DA
```

TOTAL ERRORS 2

****ERROR 201
218 NAM FMTFMT

```
00215                          OPT    LIST,MEM
00218                          NAM    FMTFMT
00219                          OPT    LIST,MEM
00220 5075                     ORG    $5075
00221                *  THIS SUBROUTINE ACCEPTS IN ACCA EITHER AN ASCII
00222                *  CHARACTER OR ZERO. IF ACCA=0, CHARACTERS ARE ASSUMED TO BE
00223                *  IN USER MEMORY AND THE OUTPUT IS BUILT FROM THERE.
00224                *  IF ACCA IS A CHARACTER, THE ROUTINE WILL SIMPLY
00225                *  PUT THAT CHARACTER INTO THE CURRENT OUTPUT LINE
00226                *  BEING BUILT AND RETURN TO THE CALLER.
00227                *  ON THE INITIAL ENTRY, T13 MUST BE SET TO ZERO. (T13=FIELD
00228                *  THIS ROUTINE BUILDS THE LINE IN LOCATIONS REAL AND IMAG
00229                *
00230 5075 D6 09    AFMT   LDA B  RSFLG    GET CURRENT RUN/STOP FLAG
00231 5077 D7 27           STA B  T8       SAVE IT FOR EXIT CHECK
00232 5079 97 25           STA A  T10      SINGLE CHARACTER MODE ?
00233 507B 26 14           BNE    SCHAR    BRANCH IF YES.
00234 507D CA C0           ORA B  #$C0     SET IT FOR "STOP" COMMAND CHECK
00235 507F D7 09           STA B  RSFLG    RESTORE THE UPDATE
00236 5081 DE CA           LDX    UIP      ELSE GET PRGM PNTR
00237 5083 08              INX             BUMP IT TO FIRST CHAR
00238 5084 08       AFMT0  INX             BUMP TO THE CHARACTER
00239 5085 9C 0B           CPX    EOPM     END OF MEMORY ?
00240 5087 27 13           BEQ    AFMT15   TERMINATE IF YES
00241 5089 DF CA           STX    UIP      ELSE SAVE UPDATED POINTER
00242 508B A6 00           LDA A  X        GET THE CHARACTER
00243 508D 26 02           BNE    SCHAR    CONTINUE IF NON ZERO
00244 508F 86 20           LDA A  #$20     ELSE LOAD A BLANK
00245 5091 81 84    SCHAR  CMP A  #$84     ALPHA TERMINATOR ?
00246 5093 27 07           BEQ    AFMT15   BRANCH IF YES
00247 5095 81 81           CMP A  #$81     END STATEMENT ?
00248 5097 26 06           BNE    AFMT1    BRANCH IF NOT
00249 5099 09              DEX             ELSE BACK UP ONE
00250 509A DF CA           STX    UIP      RESTORE UIP
00251 509C 7E 513E  AFMT15 JMP    ATERM    TERMINATE THE OUTPUT
00252 509F 81 82    AFMT1  CMP A  #$82     IS IT A PRINT COMMAND ?
00253 50A1 26 47           BNE    AFMT2    BRANCH IF NOT
00254 50A3 D6 22           LDA B  T13      GET FIELD INDICATOR
00255 50A5 26 3C           BNE    AFMT10   IGNORE PRINT IF ONE ALREADY DONE
00256 50A7 5C              INC B           ELSE SET PRINT FLAG
00257 50A8 D7 22           STA B  T13      RESTORE FIELD INDICATOR
00258 50AA 7F 000F  AFMT13 CLR    DIGFLG   TERM. DIGIT ENTRY
00259 50AD BD 5CBC         JSR    FRMT+$14   FORMAT X-REG
00260 50B0 96 67    AFMT4  LDA A  BUFF+15   GET RIGHT-MOST CHAR
00261 50B2 81 20           CMP A  #$20     IS IT A BLANK ?
00262 50B4 26 05           BNE    AFMT3    BRANCH IF NOT
00263 50B6 BD 5184         JSR    SHBUFF   ELSE SHIFT THE BUFFER RIGHT
00264 50B9 20 F5           BRA    AFMT4    CONTINUE REMOVING TRAILING BLANKS
00265 50BB CE 0057  AFMT3  LDX    #BUFF-1    GET LEFT-END CHAR ADRS - 1
00266 50BE 08       AFMT5  INX             GET CHAR ADRS
00267 50BF A6 00           LDA A  X        GET THE CHARACTER
00268 50C1 81 20           CMP A  #$20     IS IT A BLANK ?
00269 50C3 27 F9           BEQ    AFMT5    BRANCH IF YES
00270 50C5 A6 10           LDA A  16,X     ELSE GET CORRESPONDING STRING LOCATIO
00271 50C7 27 03           BEQ    AFMT7    BRANCH IF IT'S EMPTY
00272 50C9 7E 5173         JMP    LOVF1    ELSE TERMINATE THIS LINE
00273 50CC DF 23    AFMT7  STX    T12      SAVE INDEX FOR FULL CHECK
00274 50CE A6 00    AFMT14 LDA A  X        TRANSFER X-REG ASCII
00275 50D0 A7 10           STA A  16,X     THE RIGHT MOST END OF THE
00276 50D2 08              INX             STRING BEING BUILT
00277 50D3 8C 006B         CPX    #BUFF+16   DONE YET ?
00278 50D6 26 F6           BNE    AFMT14   BRANCH IF NOT
00279 50D8 DE 23           LDX    T12      GET INITIAL INDEX VALUE
00280 50DA 8C 0058         CPX    #BUFF    POINTING TO LEFT MOST CHAR ?
00281 50DD 27 39           BEQ    AFMT6    LINE OVERFLOW IF YES
00282 50DF E6 0F           LDA B  15,X     GET STRING CHAR LEFT OF X-REG
00283 50E1 26 35           BNE    AFMT6    LINE FULL IF NON-ZERO
00284 50E3 DE CA    AFMT10 LDX    UIP      ELSE GET PRGM PNTR
00285 50E5 96 25           LDA A  T10      SINGLE CHARACTER MODE ?
00286 50E7 27 9B           BEQ    AFMT0    BRANCH IF NOT
00287 50E9 39              RTS             ELSE RETURN TO CALLER
00288 50EA 81 02    AFMT2  CMP A  #2       "NEW LINE" COMMAND ?
00289 50EC 27 2A           BEQ    AFMT6    BRANCH IF YES
00290 50EE 81 03           CMP A  #3       "SHIFT" COMMAND ?
00291 50F0 26 05           BNE    AFMT16   BRANCH IF NOT
00292 50F2 73 0026         COM    T9       ELSE TOGGLE THE SHIFT FLAG
```

```
00293 50F5 20 EC          BRA     AFMT10    CONTINUE
00294 50F7 D6 26  AFMT16  LDA B   T9        GET THE SHIFT FLAG
00295 50F9 27 0A          BEQ     AFMT17    BRANCH IF SHIFT NOT INVOKED
00296 50FB 81 48          CMP A   #$48      ELSE CHECK CURRENT CHARACTER
00297 50FD 22 06          BHI     AFMT17    BRANCH IF UNAFFECTED BY "SHIFT"
00298 50FF 81 40          CMP A   #$40      ELSE CHECK LOWER LIMIT
00299 5101 23 02          BLS     AFMT17    BRANCH IF UNAFFECTED
00300 5103 8B 1F          ADD A   #$1F      GENERATE SHIFTED CHARACTER
00301 5105 D6 22  AFMT17  LDA B   T13       GET THE FIELD INDICATOR
00302 5107 26 11          BNE     AFMT8     BRANCH IF "PRINT" ALREADY RECEIVED
00303 5109 CE 0067        LDX     #REAL-1    ELSE FIND FIRST AVAILABLE LOCATION
00304 510C 08     AFMT9   INX               BUMP SEARCH ADRS
00305 510D E6 00          LDA B   X         GET CURRENT CONTENTS
00306 510F 26 FB          BNE     AFMT9     CONTINUE SEARCHING IF IT'S FULL
00307 5111 A7 00          STA A   X         ELSE SAVE NEW CHARACTER
00308 5113 8C 0077        CPX     #REAL+15    LAST AVAILABLE LOCATION
00309 5116 26 CB          BNE     AFMT10    BRANCH IF NOT FULL YET
00310 5118 20 56  AFMT6   BRA     LOVF      ELSE LINE OVERFLOW
00311 511A CE 0078 AFMT8  LDX     #REAL+16    GET RIGHT MOST ADRS + 1
00312 511D 09     AFMT12  DEX               BACK INTO THE LINE
00313 511E E6 00          LDA B   X         GET CURRENT CHARACTER
00314 5120 26 FB          BNE     AFMT12    BRANCH IF NOT EMPTY
00315 5122 09             DEX               ELSE BACK UP AGAIN
00316 5123 DF 23          STX     T12       SAVE BUFFER ADRS
00317 5125 E6 02  AFMT11  LDA B   2,X       MOVE THE RIGHT END ASCII
00318 5127 E7 01          STA B   1,X       LEFT ONE POSITION IN THE BUFFER
00319 5129 08             INX
00320 512A 8C 0076        CPX     #REAL+14    DONE SHIFTING YET ?
00321 512D 26 F6          BNE     AFMT11    BRANCH IF NOT
00322 512F A7 01          STA A   1,X       SAVE LATEST CHARACTER
00323 5131 DE 23          LDX     T12       GET INITIAL BUFFER PNTR
00324 5133 8C 0067        CPX     #REAL-1    LINE FULL NOW ?
00325 5136 27 38          BEQ     LOVF      BRANCH IF YES
00326 5138 E6 00          LDA B   X         ELSE GET CHAR LEFT OF X-REG
00327 513A 26 34          BNE     LOVF      OVERFLOW IF NON-ZERO
00328 513C 20 A5  AFMT18  BRA     AFMT10    GO SEE WHAT'S NEXT
00329 513E D6 27  ATERM   LDA B   T8        GET SAVED R/S FLAG
00330 5140 2B 02          BMI     ATERM0    BRANCH IF NO CHANGE NECESSARY
00331 5142 D7 09          STA B   RSFLG     ELSE RESTORE OLD FLAG
00332 5144 CE 0010 ATERM0 LDX     #16        SET CHARACTER COUNT
00333 5147 E6 67   ATERM4 LDA B   REAL-1,X    GET RIGHT END CHAR.
00334 5149 26 04          BNE     ATERM3    BRANCH IF NOT NULL AND PRINT
00335 514B 09             DEX               ELSE CONT. CHECKING FOR NULL STRING
00336 514C 26 F9          BNE     ATERM4    ALL CHAR. CHECKED ?
00337 514E 39             RTS               YES; NULL LINE; NO PRINT; RETURN
00338 514F CE 0057 ATERM3 LDX     #BUFF-1    PRESET INDEX TO MOVE THE LINE
00339 5152 08      ATERM2 INX               THE TEMPORARY BUILDING BUFFER
00340 5153 A6 10          LDA A   16,X      TO THE OUTPUT BUFFER PUTTING
00341 5155 26 02          BNE     ATERM1    BLANKS IN THE OUTPUT BUFFER
00342 5157 86 20          LDA A   #$20      IN THE CORRESPONDING LOCATIONS
00343 5159 A7 00   ATERM1 STA A   X         WHERE ZERO'S WERE FOUND
00344 515B 8C 0067        CPX     #BUFF+15    IN THE TEMPORARY BUFFER
00345 515E 26 F2          BNE     ATERM2
00346 5160 96 25          LDA A   T10       SINGLE CHARACTER MODE ?
00347 5162 26 09          BNE     PRINT     PRINT IF YES
00348 5164 96 09          LDA A   RSFLG     STOP KEY HIT ?
00349 5166 2B 05          BMI     PRINT     PRINT IF NOT
00350 5168 96 12          LDA A   SFLG      STEPPING ?
00351 516A 26 01          BNE     PRINT     PRINT IF YES
00352 516C 39             RTS               ELSE SKIP THE PRINT
00353 516D 7E 5300 PRINT  JMP     DEVICE    CALL THE OUTPUT DEVICE
00354 5170 7F 0022 LOVF   CLR     T13       RESET FIELD POINTER
00355 5173 8D DA   LOVF1  BSR     ATERM3    OUTPUT THE CURRENT STRING
00356 5175 CE 0010        LDX     #16       PRESET INDEX FOR CLEAR
00357 5178 6F 67   LOVF2  CLR     REAL-1,X    CLEAR LINE BUFFER
00358 517A 09             DEX
00359 517B 26 FB          BNE     LOVF2
00360 517D D6 22          LDA B   T13       GET FIELD POINTER
00361 517F 27 B8          BEQ     AFMT18    CONTINUE IF IN ALPHA
00362 5181 7E 50AA        JMP     AFMT13    ELSE RE-DO X-REG
00363 5184 CE 0058 SHBUFF LDX     #BUFF     GET THE BUFFER STARTING ADRS
00364 5187 A6 00          LDA A   X         GET LEFT MOST CHARACTER
00365 5189 08             INX               INC THE BUFFER POINTER
00366 518A 8C 0068 S      CPX     #BUFF+16    FINISHED YET ?
00367 518D 27 08          BEQ     SSS       BRANCH IF YES
00368 518F E6 00          LDA B   X         ELSE SAVE NEXT CHARACTER
```

```
00369 5191 A7 00          STA A  X         REPLACE IT WITH THE OTHER
00370 5193 17             TBA              TRANSFER NEW CHAR TO ACCB
00371 5194 08             INX              ADJUST THE PNTR
00372 5195 20 F3          BRA    S         CONTINUE SHIFTING RIGHT THE BUFFER
00373 5197 86 20    SSS   LDA A  #$20      REPLACE THE LEFT MOST CHAR
00374 5199 97 58          STA A  BUFF      WITH A BLANK AND
00375 519B 39             RTS              RETURN.
00376 519C D6 07    TOALP LDA B  TGL       WHAT MODE ?
00377 519E 2B 0E          BMI    NKLG1     BRANCH IF PROGRAM MODE
00378 51A0 D6 07    ALPHAK LDA B TGL       GET MODE
00379 51A2 2A 3C          BPL    RMOD1     BRANCH IF RUN MODE
00380 51A4 BD 4EC8        JSR    STCOD7    ELSE STORE INST
00381 51A7 7F 00C6        CLR    SOL7      CLEAR THE INSTRUCTION REGISTER
00382 51AA C6 80          LDA B  #$80      PREPARE THE SECOND BYTE
00383 51AC D7 C7          STA B  SOL7+1
00384 51AE D7 CC    NKLG1 STA B  ALPHA     SET THE ALPHA FLAG
00385 51B0 96 06          LDA A  ERROR     ERROR GENERATED ?
00386 51B2 26 29          BNE    AK3+3     BRANCH IF YES
00387 51B4 BD 7A64  AK2   JSR    SDISP1    GET NEXT KEY
00388 51B7 BD 5228        JSR    KTOA      CONVERT TO ASCII
00389 51BA 81 04          CMP A  #4        STEP COMMAND ?
00390 51BC 26 03          BNE    AK8       BRANCH IF NOT
00391 51BE 7E 527A        JMP    STEP      ELSE DO "STEP"
00392 51C1 81 05    AK8   CMP A  #5        BACK STEP COMMAND ?
00393 51C3 26 03          BNE    AK9       BRANCH IF NOT
00394 51C5 7E 52CE        JMP    BSTEP     ELSE DO "BACK STEP"
00395 51C8 97 C6    AK9   STA A  SOL7      SAVE THE ASCII
00396 51CA BD 4EC8        JSR    STCOD7    STORE IT IN MEMORY
00397 51CD 96 06          LDA A  ERROR     ERROR ?
00398 51CF 26 0C          BNE    AK3+3     BRANCH IF YES
00399 51D1 96 C6          LDA A  SOL7      GET THE CHARACTER
00400 51D3 7F 00C6        CLR    SOL7      RESET THE INST REGISTER
00401 51D6 81 B4          CMP A  #$B4      ALPHA TERMINATOR ?
00402 51D8 26 DA          BNE    AK2       BRANCH IF NOT
00403 51DA 7F 00CC  AK3   CLR    ALPHA     ELSE RESET THE ALPHA FLAG
00404 51DD 7E 7978        JMP    BWM       RETURN TO SUPV.
00405 51E0 4F       RMOD1 CLR A            ZERO TO ACCA
00406 51E1 97 22          STA A  T13       CLEAR THE FIELD POINTER
00407 51E3 97 26          STA A  T9        RESET THE SHIFT FLAG
00408 51E5 4C             INC A            SET ACCA
00409 51E6 97 CC          STA A  ALPHA     SET ALPHA FLAG
00410 51E8 CE 0010        LDX    #16       PRESET INDEX
00411 51EB 6F 67    AK7   CLR    REAL-1,X    CLEAR LINE BUFFER
00412 51ED 09             DEX
00413 51EE 26 FB          BNE    AK7
00414 51F0 BD 7A64  AK6   JSR    SDISP1    GO GET NEXT CHAR.
00415 51F3 8D 36          BSR    KTOA      CONVERT ACCA TO ASCII
00416 51F5 D6 C7          LDA B  SOL7+1    GET INST
00417 51F7 C4 70          AND B  #$70      MASK OFF SELECT CODE
00418 51F9 27 16          BEQ    PRT       BRANCH IF PRINTER
00419 51FB 97 68          STA A  REAL      ELSE PUT CHAR IN REAL
00420 51FD BD 532A        JSR    IOFMT     CALL THE DEVICE
00421 5200 96 06          LDA A  ERROR     ERROR GENERATED ?
00422 5202 26 D6          BNE    AK3       BRANCH IF YES
00423 5204 96 68          LDA A  REAL      GET LAST CHAR OUTPUT
00424 5206 81 B4          CMP A  #$B4      ALPHA TERMINATOR ?
00425 5208 27 D0          BEQ    AK3       BRANCH IF YES
00426 520A CE 00C6        LDX    #SOL7     ELSE RESET INST PNTR
00427 520D DF CA          STX    UIP       SAVE IT
00428 520F 20 DF          BRA    AK6       CONTINUE
00429 5211 8D 15    PRT   BSR    AK5       OUTPUT TO BUFFER
00430 5213 96 25          LDA A  T10       GET LAST CHAR OUTPUT
00431 5215 81 B4          CMP A  #$B4      ALPHA TERMINATOR ?
00432 5217 27 C1          BEQ    AK3       BRANCH IF YES
00433 5219 20 D5          BRA    AK6       ELSE CONTINUE
00434 521B 4F       RMPRT CLR A
00435 521C 97 22          STA A  T13       CLEAR FIELD POINTER
00436 521E 97 26          STA A  T9        CLEAR SHIFT INDICATOR
00437 5220 CE 0010        LDX    #16       PRESET INDEX
00438 5223 6F 67    RMPRT1 CLR   REAL-1,X   CLEAR LINE BUFFER
00439 5225 09             DEX
00440 5226 26 FB          BNE    RMPRT1
00441 5228 7E 5075  AK5   JMP    AFMT      OUTPUT THE ASCII
00442 522B CE 5239  KTOA  LDX    #ASCII    GET ASCII TABLE ADDRESS
00443 522E DF 14          STX    TP1       SAVE FOR MODIFICATION
00444 5230 9B 15          ADD A  TP1S      ADD THE KEYCODE TO LSB'S
```

```
00445 5232 97 15          STA A  TP1S     RESTORE RESULT
00446 5234 DE 14          LDX    TP1      RESTORE NEW INDEX
00447 5236 A6 00          LDA A  X        ACCA=ASCII CHARACTER
00448 5238 39             RTS            RETURN
00449                     *
00450        4EC8         STCOD7 EQU    $4EC8
00451        532A         IOFMT  EQU    $532A
00452        7A64         SDISP1 EQU    $7A64
00453        7978         BWM    EQU    $7978
00454        527A         STEP   EQU    $527A
00455        52CE         BSTEP  EQU    $52CE
00457                     *
00458                     *THIS FILE DEFINES THE ASCII SET FOR CJ.
00459                     *ENTRIES ARE IN ORDER OF KEYCODES FROM
00460                     *ZERO TO OCTAL 77.
00461                     *
00462 5239 50             ASCII  FCB    $50      P
00463 523A 51                    FCB    $51      Q
00464 523B 52                    FCB    $52      R
00465 523C 53                    FCB    $53      S
00466 523D B4                    FCB    $B4      TERMINATOR
00467 523E 57                    FCB    $57      W
00468 523F 27                    FCB    $27      '
00469 5240 3E                    FCB    $3E      >
00470 5241 54                    FCB    $54      T
00471 5242 55                    FCB    $55      U
00472 5243 56                    FCB    $56      V
00473 5244 04                    FCB    $4       STEP COMMAND
00474 5245 05                    FCB    $5       BACK STEP COMMAND
00475 5246 58                    FCB    $58      X
00476 5247 37                    FCB    $37      7
00477 5248 38                    FCB    $38      8
00478 5249 41                    FCB    $41      A
00479 524A 42                    FCB    $42      B
00480 524B 43                    FCB    $43      C
00481 524C 44                    FCB    $44      D
00482 524D 45                    FCB    $45      E
00483 524E 59                    FCB    $59      Y
00484 524F 34                    FCB    $34      4
00485 5250 35                    FCB    $35      5
00486 5251 46                    FCB    $46      F
00487 5252 47                    FCB    $47      G
00488 5253 48                    FCB    $48      H
00489 5254 49                    FCB    $49      I
00490 5255 4A                    FCB    $4A      J
00491 5256 5A                    FCB    $5A      Z
00492 5257 31                    FCB    $31      1
00493 5258 32                    FCB    $32      2
00494 5259 4B                    FCB    $4B      K
00495 525A 4C                    FCB    $4C      L
00496 525B 4D                    FCB    $4D      M
00497 525C 4E                    FCB    $4E      N
00498 525D 4F                    FCB    $4F      O
00499 525E 20                    FCB    $20      BLANK
00500 525F 30                    FCB    $30      0
00501 5260 2E                    FCB    $2E      .
00502 5261 24                    FCB    $24      $
00503 5262 23                    FCB    $23      #
00504 5263 3F                    FCB    $3F      ?
00505 5264 25                    FCB    $25      %
00506 5265 5E                    FCB    $5E      ^
00507 5266 3A                    FCB    $3A      :
00508 5267 2A                    FCB    $2A      *
00509 5268 2D                    FCB    $2D      -
00510 5269 40                    FCB    $40      @
00511 526A 03                    FCB    $3       SHIFT
00512 526B B2                    FCB    $B2      PRINT
00513 526C 2F                    FCB    $2F      /
00514 526D 3C                    FCB    $3C      <
00515 526E 39                    FCB    $39      9
00516 526F 36                    FCB    $36      6
00517 5270 33                    FCB    $33      3
00518 5271 2C                    FCB    $2C      ,
00519 5272 29                    FCB    $29      )
00520 5273 3D                    FCB    $3D      =
00521 5274 28                    FCB    $28      (
```

```
00522 5275 01              FCB    $1        TAB
00523 5276 20              FCB    $20       UNUSED KEYCODE (@75)
00524 5277 02              FCB    $2        NEW LINE
00525 5278 28              FCB    $28       +
00526                *
00527                *CODE TO SELECT THE PRINT DEVICE.  IF EOM
00528                *IS NEGATIVE, THE OUTPUT DEVICE IS SPECIFIED BY THE
00529                *CONTENTS OF T6 AND T5.  EOM MUST BE CONTROLLED
00530                *BY THE CALLER AND RESET TO THE CORRECT VALUE
00531                *WHEN FINISHED.
00532                *
00533 53D0                 ORG    $53D0
00534 53D0 D6 0A   DEVICE LDA B  EOM       GET END OF MEMORY WORD
00535 53D2 2B 03           BMI    NOPRT     NOT INTERNAL PRINTER IF NEGATIVE
00536 53D4 7E 6034         JMP    PRTDRV+7    ELSE CALL THE PRINTER
00537 53D7 DE 29   NOPRT  LDX    T6        GET THE DEVICE ADRS
00538 53D9 6E 00           JMP    X         CALL THE DEVICE
00541                      END
```

SYMBOL TABLE

ADATA	0000	ACTL	0001	BDATA	0002	BCTL	0003	INPUT	0004	IOIN	0005
ERROR	0006	TGL	0007	UFLG	0008	RSFLG	0009	EOM	000A	EOPM	000B
STKFLG	000D	RND	000E	DIGFLG	000F	W2	0010	W1	0011	SFLG	0012
DCNTR	0013	TP1	0014	TP1S	0015	TP2	0016	TP2S	0017	TP3	0018
TP3S	0019	TP4	001A	TP4S	001B	TP5	001C	TP5S	001D	TP6	001E
TP6S	001F	TP7	0020	TP7S	0021	T13	0022	T12	0023	T11	0024
T10	0025	T9	0026	T8	0027	T7	0028	T6	0029	T5	002A
T4	002B	T3	002C	T2	002D	T1	002E	ISTK	002F	ISTACK	0051
TA	0052	SPGM	0054	EXTRA	0056	BUFF	0058	REAL	0068	IMAG	0070
AT1	0078	AT2	0080	W	0088	XR	0090	YR	0098	ZR	00A0
TR	00A8	LSTX	00B0	BKWRT	00B8	BKKC	00BA	SOL7	00C6	UPP	00C8
UIP	00CA	ALPHA	00CC	IO1	00CD	IO2	00D0	IT7	00D3	FLAG	00D5
TPOS	00D6	FILE	00D7	AR	00D8	BR	00E0	CR	00E8	DR	00F0
ER	00F8	SDBB	00BA	MT	7E00	TERMN7	003D	IMED	0040	PARCD	00C0
PAREX	0080	NTBL	0000	DOTS	5EC0	PRTDRV	602D	FRMT	5CA8	BLANK	5D75
LDMSG	578D	ROLLD	55B2	ROLLU	57F1	PSD	55DA	TXL	55E9	STKUP	55EF
MAD	749B	CMP	74AA	NOR	74D6	TXW	7424	TXXR	743B	EXXR	7452
ARSR	753B	OVUNF	75B6	OVERF	75DD	XR0	740A	XRNINE	75C8	UNDRF	75F1
IMULT	76B9	QDG	7669	FPA	75FC	FPS	75F6	FPM	7735	FPD	7793
FPAEX	763D	FPMEX	7780	LSHIFT	7521	ZEROX	7489	XZERO0	7416	XZERO2	7417
RECIP	73E6	TXRX	73F3	CONST	6800	FPDBRC	6898	TAN	68A9	ATN	69C3
DSZERO	6446	NTLN	6A58	EXPN	6AC9	SIN	6B94	COS	6B9A	ASIN	6BF2
ACOS	6BF7	PH1	6C5D	PH2	6C8D	PH3	6D34	PH4	6DD0	LSFT8	6E47
SQRT	6E65	MAD8	6F2C	CMP8	53E4	IOUPX	6F52	LOG10	6FA7	YUPX	6FE9
RTOP	7328	PTOR	7386	AFMT	5075	AFMT0	5084	SCHAR	5091	AFMT15	509C
AFMT1	509F	AFMT13	50AA	AFMT4	50B0	AFMT3	50BB	AFMT5	50BE	AFMT7	50CC
AFMT14	50CE	AFMT10	50E3	AFMT2	50EA	AFMT16	50F7	AFMT17	5105	AFMT9	510C
AFMT6	5118	AFMT8	511A	AFMT12	511D	AFMT11	5125	AFMT18	513C	ATERM	513E
ATERM0	5144	ATERM4	5147	ATERM3	514F	ATERM2	5152	ATERM1	5159	PRINT	516D
LOVF	5170	LOVF1	5173	LOVF2	5178	SHBUFF	5184	S	518A	SSS	5197
TOALP	519C	ALPHAK	51A0	NKLG1	51AE	AK2	51B4	AK8	51C1	AK9	51C8
AK3	51DA	RMOD1	51E0	AK7	51EB	AK6	51F0	PRT	5211	RMPRT	5218
RMPRT1	5223	AK5	5228	KTOA	522B	STCOD7	4EC8	IOFMT	532A	SDISP1	7464
BWM	797B	STEP	527A	BSTEP	52CE	ASCII	5239	DEVICE	53D0	NOPRT	53D7

TOTAL ERRORS 1

```
00215                      OPT    LIST.MEM
00218                *
00219                *    THESE ROUTINES RETURN THE INTEGER AND
00220                *    FRACTIONAL PART OF THE ORIGINAL ARGUMENT
00221                *
00222                *    JOHN KEITH    MAY 28, 1974    REV A
00223                *
00224       0014     REGPTR EQU    TP1
00225 5550                 ORG    $5550
00226 5550 BD 55E9 INTFR1 JSR    TXL       SAVE X IN LAST X
00227 5553 CE 0092         LDX    #XR+2     SET INDEX TO MANTISSA
00228 5556 D6 90           LDA B  XR        (EXP+1)/2=WHOLE WORDS TO CLEAR
00229 5558 2B 40           BMI    INTFR9    RET. IF NEG. (ALREADY FRACT.)
00230 555A 5C              INC B
```

```
00231 5558 57          ASR B
00232 555C 5A       INTFR2 DEC B              LOOP TO CLEAR WHOLE WORDS
00233 555D 2B 05           BMI    INTFR3      EXIT FROM LOOP
00234 555F A7 00           STA A  X           CLEAR PAIR OF MANTISSA DIGITS
00235 5561 08              INX                MOVE POINTER
00236 5562 20 F8           BRA    INTFR2      TEST IF DONE
00237 5564 24 06    INTFR3 BCC    INTFR4      CLEAR HALF WORD ?
00238 5566 A6 00           LDA A  X           YES, LOAD WORD
00239 5568 84 0F           AND A  #$F         CLEAR UPPER HALF
00240 556A A7 00           STA A  X           RESTORE RESULT
00241 556C 7E 74D6  INTFR4 JMP    NOR         NOW NORMALIZE RESULT
00242 556F BD 55E9  INTFR5 JSR    TXL         SAVE X IN LAST X
00243 5572 96 90           LDA A  XR          EXAMINE EXPONENT
00244 5574 2A 03           BPL    INTFR6      POSITIVE, CHECK SIZE OF EXPONENT
00245 5576 7E 740A         JMP    XR0         NEGATIVE, NO INTEGER PART, CLEAR AND
00246 5579 81 0B    INTFR6 CMP A  #11         IS EXP > 10 ?
00247 557B 24 1D           BCC    INTFR9      YES, (X) IS INTEGER PART
00248 557D 96 90           LDA A  XR          GET EXPONENT
00249 557F 4C              INC A              ADD 1 SO EXP/2 POINTS TO PROPER WORD
00250 5580 7F 0014         CLR    REGPTR      CLEAR FIRST BYTE OF POINTER
00251 5583 44              LSR A              DIVIDE  EXP+1 BY 2
00252 5584 97 15           STA A  REGPTR+1      SAVE WORD POINTER
00253 5586 DE 14           LDX    REGPTR      LOAD WORD POINTER IN X
00254 5588 24 07           BCC    INTFR7+1      WAS EXP+1 ODD ?
00255 558A A6 92           LDA A  $92,X       YES, CLEAR SECOND HALF OF BYTE
00256 558C 84 F0           AND A  #$F0        CLEAR SECOND HALF
00257 558E A7 92           STA A  $92,X       RESTORE WORD IN X
00258 5590 08       INTFR7 INX                POINT TO NEXT WORD IN X-REG
00259 5591 8C 0006         CPX    #6          IS CLEARING COMPLETE ?
00260 5594 27 04           BEQ    INTFR9      YES, EXIT
00261 5596 6F 92           CLR    $92,X       CLEAR NEXT WORD
00262 5598 20 F6           BRA    INTFR7      CONTINUE OPERATION
00263 559A 39       INTFR9 RTS                RETURN
00265                  *
00266                  *     CLEAR X-REGISTER
00267                  *
00268                  *     JOHN KEITH    APRIL 2, 1974    REV A
00269                  *
00270 559B D7 2E    CLXR1  STA B  T1          SAVE RTS FLAG
00271 559D BD 740A         JSR    XR0         CLEAR X-REG
00272 55A0 4F       LDEX   CLR A              COMMON EXIT POINT FOR STACK LIFT DISA
00273 55A1 97 0F           STA A  DIGFLG      CLEAR FLAG TO START NEW DATA ENTRY
00274 55A3 7E 569C         JMP    DEXIT       RETURN TO SUPV
00276                  *
00277                  *     REGISTER TRANSFER ROUTINES
00278                  *
00279                  *     JOHN KEITH    APRIL 2, 1974    REV B
00280                  *
00281                  *     ENTER KEY
00282                  *     Z->T,Y->Z,X->Y, X UNCHANGED
00283                  *
00284 55A6 D7 2E    ENTER1 STA B  T1          SAVE RTS FLAG
00285 55A8 8D 4A           BSR    STKUP+5     LIFT STACK
00286 55AA 20 F4           BRA    LDEX        RET TO SUPV
00288                  *
00289                  *     X EXCHANGE Y
00290                  *
00291 55AC CE 0098  XEY1   LDX    #YR         GET Y-REG ADRS
00292 55AF 7E 7452         JMP    EXXR        EXCHANGE REGISTERS
00294                  *
00295                  *     ROLL DOWN
00296                  *     T->TEMP,X->T,Y->X,Z->Y,TEMP->Z
00297                  *
00298 55B2 CE 0008 ROLLD1 LDX     #@10        LOAD COUNTER/POINTER
00299 55B5 A6 8F   ROLLD5 LDA A   $8F,X       LOAD WORD FROM X
00300 55B7 E6 A7          LDA B   $A7,X       LOAD WORD FROM T
00301 55B9 A7 A7          STA A   $A7,X       STORE X IN T
00302 55BB A6 9F          LDA A   $9F,X       LOAD WORD FROM Z
00303 55BD E7 9F          STA B   $9F,X       STORE T IN X
00304 55BF E6 97          LDA B   $97,X       LOAD WORD FROM Y
00305 55C1 A7 97          STA A   $97,X       STORE Z IN Y
00306 55C3 E7 8F          STA B   $8F,X       STORE Y IN X
00307 55C5 09             DEX                 DECREMENT COUNTER/POINTER
00308 55C6 26 ED          BNE     ROLLD5      CONTINUE
00309 55C8 39             RTS                 RETURN TO CALLING ROUTINE
00311                  *
```

```
00312                *    RECALL LAST X TO X-REGISTER
00313                *
00314                *    JOHN KEITH    APRIL 3, 1974    REV A
00315                *
00316 55C9 91 0F    RLSTX1 CMP A  DIGFLG   DOING DIGIT ENTRY ?
00317 55CB 27 05           BEQ    RLSTX2   BRANCH IF NOT
00318 55CD 97 0F           STA A  DIGFLG   ELSE TERMINATE IT
00319 55CF 43              COM A           SET MSB
00320 55D0 97 0D           STA A  STKFLG   ENABLE LIFT
00321 55D2 8D 18    RLSTX2 BSR    STKUP    DO STACK UP IF ENABLED
00322 55D4 CE 00B0         LDX    #LSTX    SET INDEX REGISTER TO LAST X
00323 55D7 7E 743B         JMP    TXXR     TRANSFER X TO LAST X
00325                *
00326                *     PSD, PARTIAL STACK DOWN
00327                *   PREFORMS OPERATION Z->Y,T->Z
00328                *
00329                *    JOHN KEITH    APRIL 3, 1974    REV A
00330                *
00331 55DA CE 0008 PSD1   LDX    #8       LOAD COUNTER/POINTER
00332 55DD A6 A7   PSD5   LDA A  $A7,X    LOAD WORD FROM T
00333 55DF E6 9F          LDA B  $9F,X    LOAD WORD FROM Z
00334 55E1 A7 9F          STA A  #9F,X    STORE INTO Z
00335 55E3 E7 97          STA B  $97,X    STORE INTO Y
00336 55E5 09             DEX             DECR COUNTER
00337 55E6 26 F5          BNE    PSD5     NOT COMPLETE
00338 55E8 39             RTS             RETURN TO CALLING ROUTINE
00340                *
00341                *    TXL, TRANSFER X TO LAST X
00342                *
00343                *    JOHN KEITH    APRIL 3, 1974    REV A
00344                *
00345 55E9 CE 00B0         LDX    #LSTX    GET LAST X ADRS
00346 55EC 7E 73F3         JMP    TXRX     TRANSFER X-REG
00348                *
00349                *    STKUP PERFORMS THE FOLLOWING OPERTION:
00350                *
00351                *    Z->T,Y->Z,X->Y, T LOST
00352                *    THIS OPERTION OCCURS ONLY IF THE AUTO
00353                *    STACK UP FLAG IS SET.  IF THE ROUTINE
00354                *    IS ENTERED AT STKUP2 THEN THE OPERATION
00355                *    IS ALWAYS PERFORMED.
00356                *
00357                *    JOHN KEITH    MAR 24, 1974    REV B
00358                *
00359      0020      STEMP  EQU    TP7
00360 55EF 7D 000D STKUP1 TST    STKFLG   TEST CONDITION OF AUTO STACK FLG IN H
00361 55F2 2A 1C          BPL    STKUP9   AUTO LIFT IS NOT ENABLED
00362 55F4 0F      STKUP2 SEI              DISABLE INTERRUPTS WHILE LIFTING STAC
00363 55F5 9F 20          STS    STEMP    SAVE STACK POINTER IN TEMPORARY
00364 55F7 CE 0008        LDX    #8       LOAD LOOP COUNTER
00365 55FA AE 9E   STKUP3 LDS    $9E,X    LOAD TWO WORDS FROM Z
00366 55FC AF A6          STS    $A6,X    STORE WORDS IN T
00367 55FE AF 96          LDS    $96,X    LOAD TWO WORDS FROM Y
00368 5600 AF 9E          STS    $9E,X    STORE WORDS IN Z
00369 5602 AF 8E          LDS    $8E,X    LOAD TWO WORDS FROM X
00370 5604 AF 96          STS    $96,X    STORE WORDS IN Y
00371 5606 09             DEX
00372 5607 09             DEX             DECREMENT COUNTER, POINT TO NEXT LOCA
00373 5608 26 F0          BNE    STKUP3   CONTINUE MOVING REGISTERS
00374 560A 9E 20          LDS    STEMP    RELOAD ORIGINAL STACK POINTER
00375 560C 0E             CLI             ENABLE INTERRUPTS
00376 560D 7F 000D        CLR    STKFLG   CLEAR AUTO STACK FLAG
00377 5610 39      STKUP9 RTS              RETURN TO CALLING POINT
00379                *
00380                *    CLEAR STACK ROUTINE
00381                *
00382                *    JOHN KEITH    JUNE 3, 1974    REV A
00383                *
00384      0014      SAVE   EQU    TP1
00385 5611 0F      CLR1   SEI              DISABLE INTERRUPTS
00386 5612 9F 14          STS    SAVE     SAVE STACK POINTER
00387 5614 8E 0000        LDS    #0       LOAD STACK POINTER WITH CLEAR VALUE
00388 5617 CE 0020        LDX    #32      INITIALIZE LOOP CTR/POINTER
00389 561A AF 8E   CLR3   STS    $8E,X    CLEAR TWO BYTES
00390 561C 09             DEX             DECREMENT LOOP CTR/POINTER
```

```
00391 561D 09              DEX
00392 561E 26 FA           BNE    CLR3      CONTINUE CLEARING
00393 5620 9E 14           LDS    SAVE      RESTORE STACK POINTER
00394 5622 0E              CLI              ENABLE INTERRUPTS
00395 5623 39              RTS              RETURN
00397                      *
00398                      *     PRINT KEY EXECUTION
00399                      *
00400                      *     JOHN KEITH    APRIL 19, 1974    REV A
00401                      *
00402 5624 97 0F    PRT1   STA A  DIGFLG    TERM. DIGIT ENT.
00403 5626 BD 5CBC         JSR    FRMT+$14   FORMAT
00404 5629 7E 6034         JMP    PRTDRV+7   PRINT AND RETURN
00406                      *
00407                      *     PRINT STACK
00408                      *
00409                      *     JOHN KEITH    JUNE 4, 1974    REV A
00410                      *
00411       002E    TEMP   EQU    T1
00412 562C 97 0F    PTSTK1 STA A  DIGFLG    TERM. DIGIT ENT.
00413 562E 86 04           LDA A  #4        SET CNTR
00414 5630 97 2E           STA A  TEMP,     SAVE CTR
00415 5632 BD 57F1  PTSTK3 JSR    ROLLUP     POSITION DATA
00416 5635 BD 5CBC         JSR    FRMT+$14   FORMAT CURRENT DATA IN X-REG
00417 5638 BD 6034         JSR    PRTDRV+7   PRINT THE X-REG
00418 563B 7A 002E         DEC    TEMP       DECREMENT LOOP CTR, DONE ?
00419 563E 26 F2           BNE    PTSTK3     NO, CONTINUE
00420 5640 39              RTS              RETURN
00422                      *
00423                      *     DIGIT ENTRY ROUTINES, KEYS SERVICED IN
00424                      *     THIS PACKAGE INCLUDE DIGIT KEYS (0-9),
00425                      *   . DECIMAL POINT, CHANGE SIGN, AND ENTER EXPONENT
00426                      *
00427                      *
00428       0088    EXP    EQU    W
00429       0089    EXP1   EQU    W+1
00430       0014    DTEMP  EQU    TP1
00431       008A    EXP2   EQU    W+2
00432       7982    BWM2   EQU    $7982
00433                      *
00434                      *     CHANGE SIGN ROUTINE
00435                      *     PART OF ENTER EXPONENT IS INCLUDED
00436                      *
00437                      *
00438                      *     JOHN KEITH    MAY 8,1974    REV D
00439                      *
00440 5641 D7 2E    CHS      STA B  T1        SAVE RTS FLAG
00441 5643 D6 0F             LDA B  DIGFLG    CHECK IF EEX FLAG IS SET
00442 5645 C5 40             BIT B  #%01000000   TEST EEX FLAG, SET ?
00443 5647 26 08             BNE    D1        YES, CHANGE SIGN OF EXPONENT MODIFIER
00444 5649 96 91             LDA A  XR+1      NO, CHANGE MANTISSA SIGN
00445 564B 88 80             EOR A  #$80      CHANGE SIGN
00446 564D 97 91             STA A  XR+1      STORE UPDATED SIGN WORD
00447 564F 20 4E             BRA    DEXIT1    DO NOT MODIFY STACK FLAG
00448 5651 C8 10    D1       EOR B  #%00010000   TOGGLE NEGATIVE BIT (EXPONENT IND
00449 5653 D7 0F             STA B  DIGFLG    SAVE UPDATED VALUE
00450 5655 86 80             LDA A  #$80      LOAD CHANGE SIGN FLAG
00451 5657 D6 90    D3       LDA B  XR        LOAD CURRENT VALUE OF EXPONENT
00452 5659 D0 88             SUB B  EXP       SUBTRACT EXPONENT MODIFIER
00453 565B D7 90             STA B  XR        RESTORE EXP
00454 565D 4D                TST A            IS THIS CHS OR ENTER EXP DIGIT
00455 565E 2A 11             BPL    D9        ENTER DIGIT
00456 5660 70 0088  D5       NEG    EXP       COMPLEMENT EXPONENT MODIFIER (CHS)
00457 5663 D6 88    D7       LDA B  EXP       PRESET B-REG
00458 5665 DB 90             ADD B  XR        ADD MODIFIER TO EXPONENT
00459 5667 BD 75B8           JSR    OVUNF+2    TEST FOR OVER/UNDERFLOW
00460 566A 28 30             BVC    DEXIT     NONE, RETURN
00461 566C 7F 000F  D8       CLR    DIGFLG    OVER/UNDERFLOW OCCURRED, SET NEW ENTR
00462 566F 20 28             BRA    DEXIT     RETURN
00463 5671 D6 89    D9       LDA B  EXP1      LOAD LAST DIGIT USED IN EXP MODIFIER
00464 5673 97 89             STA A  EXP1      STORE NEW DIGIT USED
00465 5675 17                TBA              MULTIPLY OLD DIGIT BY 10 (BINARY)
00466 5676 49                ROL A            PERFORM A *8 BY
00467 5677 49                ROL A            SHIFTING THREE PLACES
00468 5678 49                ROL A
00469 5679 1B                ABA              COMPLETE *10 BY PERFORMING
```

```
00470 567A 1B              ABA              TWO ADDITIONS
00471 567B 9B 89           ADD A  EXP1      NOW ADD NEW DIGIT TO EXP MODIFIER
00472 567D 97 88           STA A  EXP       SAVE NEW EXP MODIFIER
00473 567F D6 0F           LDA B  DIGFLG    CHECK TO SEE WHAT SIGN EXP MOD SHOULD
00474 5681 C5 10           BIT B  #%00010000   IS NEGATIVE BIT SET ?
00475 5683 26 DB           BNE    D5        YES, CHANGE SIGN
00476 5685 20 DC           BRA    D7        NO
00478                 *
00479                 *    DECIMAL POINT ENTRY ROUTINE
00480                 *
00481                 *    JOHN KEITH    MAY 8, 1974    REV F
00482                 *
00483 5687 D7 2E    DECPNT STA B  T1        SAVE RTS FLAG
00484 5689 D6 0F           LDA B  DIGFLG    IS THIS A NEW DIGIT ENTRY ?
00485 568B 26 0B           BNE    D15       NO, SET FLAG
00486 568D BD 55EF D11     JSR    STKUP     DO A STACK UP IF ENABLED
00487 5690 BD 740A         JSR    XR0       CLEAR X-REG FOR NEW ENTRY
00488 5693 4F              CLR A
00489 5694 97 8A           STA A  EXP2      CLEAR EXP FOR 0.000.... CASE
00490 5696 97 88           STA A  EXP       CLEAR EXP FOR FRMT ROUTINE
00491 5698 CA 80    D15    ORA B  #%10000000   SET DECIMAL POINT FLAG
00492 569A D7 0F           STA B  DIGFLG    SAVE NEW FLAG VALUE
00493 569C 7F 000D DEXIT   CLR    STKFLG    DISABLE AUTO STACK LIFT
00494 569F D6 2E   DEXIT1  LDA B  T1        DO RTS ?
00495 56A1 26 01           BNE    FIX1      BRANCH IF NOT
00496 56A3 39              RTS              ELSE DO RETURN
00497 56A4 31       FIX1   INS              INCREMENT STACK POINTER
00498 56A5 31              INS              TO DELETE LAST RETURN VECTOR
00499 56A6 7E 7982         JMP    RWM2      RET. TO SUPERVISOR
00501                 *
00502                 *    ENTER EXPONENT ROUTINE
00503                 *    INCLUDES ROUTINE TO ENTER DIGITS IN MANTISSA
00504                 *
00505                 *    JOHN KEITH    APRIL 2, 1974    REV C
00506                 *
00507 56A9 D7 2E    EEX    STA B  T1        SAVE RTS FLAG
00508 56AB D6 0F           LDA B  DIGFLG    PREPARE TO SET EEX FLAG
00509 56AD C5 40           BIT B  #%01000000   IS EEX FLAG ALREADY SET ?
00510 56AF 26 EB           BNE    DEXIT     YES, EXIT FROM ROUTINE
00511 56B1 CA 40           ORA B  #%01000000   SET EEX FLAG
00512 56B3 D7 0F           STA B  DIGFLG    SAVE NEW FLAG VALUE
00513 56B5 97 88           STA A  EXP       CLEAR EXP MODIFIER LOCATION (A WAS CL
00514 56B7 97 89           STA A  EXP1      CLEAR LAST EXP DIGIT TEMP
00515 56B9 C4 0F           AND B  #%00001111   SAVE COUNTER PORTION OF DIGFLG
00516 56BB 26 DF           BNE    DEXIT     THIS IS NOT A NEW ENTRY, RETURN
00517 56BD BD 55EF         JSR    STKUP     NEW ENTRY, DO A STACK UP
00518 56C0 BD 740A         JSR    XR0       CLEAR X-REG FOR NEW ENTRY
00519 56C3 4C              INC A            SET A TO 1 FOR NEW MANTISSA DIGIT
00520 56C4 D6 0F    D22    LDA B  DIGFLG    LOAD MANTISSA COUNTER
00521 56C6 C4 0F           AND B  #%00001111   SAVE COUNTER
00522 56C8 C1 0A           CMP B  #10       IS COUNTER=10 ?
00523 56CA 24 D0           BCC    DEXIT     YES, IGNORE ENTRY
00524 56CC 54       D23    LSR B            DIVIDE CTR BY 2, SAVE REMAINDER IN CA
00525 56CD 25 04           BCS    D25       CTR WAS ODD, SKIP DIGIT SHIFT
00526 56CF 48              ASL A            POSITION NEW DIGIT
00527 56D0 48              ASL A
00528 56D1 48              ASL A
00529 56D2 48              ASL A
00530 56D3 D7 15    D25    STA B  DTEMP+1   SAVE COUNTER
00531 56D5 7F 0014         CLR    DTEMP     CLEAR FIRST BYTE OF CTR
00532 56D8 DE 14           LDX    DTEMP     LOAD CTR INTO INDEX
00533 56DA AA 92           ORA A  XR+2,X    ADD NEW DIGIT TO MANTISSA
00534 56DC A7 92           STA A  XR+2,X    STORE RESULT
00535 56DE 7C 000F         INC    DIGFLG    INC COUNTER
00536 56E1 20 B9    D27    BRA    DEXIT     RETURN TO SUPERVISOR
00538                 *
00539                 *    DIGIT ACCEPTANCE ROUTINE
00540                 *
00541                 *    JOHN KEITH    APRIL 2, 1974 $  REV D
00542                 *
00543 56E3 4C       DIGIT9 INC A                BUILD APPROPRIATE DIGIT
00544 56E4 4C       DIGIT8 INC A                IN A-REGISTER
00545 56E5 4C       DIGIT7 INC A
00546 56E6 4C       DIGIT6 INC A
00547 56E7 4C       DIGIT5 INC A
00548 56E8 4C       DIGIT4 INC A
```

```
00549 56E9 4C       DIGIT3 INC A
00550 56EA 4C       DIGIT2 INC A
00551 56EB 4C       DIGIT1 INC A
00552 56EC D7 2E    DIGIT0 STA B  T1        SAVE RTS FLAG
00553 56EE D6 0F           LDA B  DIGFLG    CHECK CURRENT STATUS
00554 56F0 26 0F           BNE    D35       IT IS NOT A NEW ENTRY
00555 56F2 BD 55EF         JSR    STKUP     DO A STACK UP IF ENABLED
00556 56F5 BD 740A         JSR    XR0       CLEAR X-REG FOR NEW ENTRY
00557 56F8 D7 88           STA B  EXP       CLEAR EXP FOR FRMT ROUTINE
00558 56FA D7 8A           STA B  EXP2      CLEAR EXP2 FOR FRMT
00559 56FC 4D              TST A            WAS DIGIT A ZERO ?
00560 56FD 27 3F           BEQ    D28       YES: GO SET "LEADING ZERO" BIT
00561 56FF 20 C3           BRA    D22       NO, ENTER DIGIT IN MANTISSA
00562 5701 C1 20    D35    CMP B  #$20      ONLY "LEADING ZERO" BIT SET ?
00563 5703 26 05           BNE    D36       BRANCH IF NOT THE CASE
00564 5705 4D              TST A            ANOTHER LEADING ZERO ?
00565 5706 27 D9           BEQ    D27       RETURN IF YES
00566 5708 20 BA           BRA    D22       ENTER FIRST NON-ZERO DIGIT
00567 570A C5 40    D36    BIT B  #$40      IS EEX FLAG SET ?
00568 570C 27 03           BEQ    D39       NO, CHECK OTHER FLAG
00569 570E 7E 5657         JMP    D3        YES, ENTER NEW DIGIT IN EXPONENT
00570 5711 C4 80    D39    AND B  #%10000000   IS DECIMAL POINT FLAG SET ?
00571 5713 26 0E           BNE    D43       YES
00572 5715 5C              INC B            NO, SET B TO 1 FOR VALUE TO ADD TO EX
00573 5716 DB 90           ADD B  XR        INC EXPONENT
00574 5718 97 8B           STA A  W+3       SAVE DIGIT
00575 571A BD 7588         JSR    OVUNF+2    TEST FOR OVER/UNDERFLOW
00576 571D 29 11           BVS    D45       OVERFLOW EXIT
00577 571F 96 8B           LDA A  W+3       NO OVERFLOW, RELOAD DIGIT
00578 5721 20 A1           BRA    D22       ENTER DIGIT INTO MANTISSA
00579 5723 D6 92    D43    LDA B  XR+2      IS MANTISSA=0 ?
00580 5725 26 9D           BNE    D22       NO, ENTER DIGIT INTO MANTISSA
00581 5727 7A 008A         DEC    EXP2      YES, DECREMENT EXPONENT TEMPORARY
00582 572A D6 8A           LDA B  EXP2      CHECK IF UNDERFLOW OCCURRED
00583 572C C1 9D           CMP B  #@235     UNDERFLOW ?
00584 572E 24 03           BCC    D47       NO, CONTINUE
00585 5730 7E 566C  D45    JMP    D8        SET NEW DIGIT ENTRY AND EXIT
00586 5733 4D       D47    TST A            WAS DIGIT ENTERED A ZERO
00587 5734 27 AB           BEQ    D27       YES, IGNORE
00588 5736 7C 008A         INC    EXP2      RESTORE EXP2 FOR FORMATTER
00589 5739 D7 90           STA B  XR        VALUE OF EXP INTO X-REG
00590 573B 7E 56C4         JMP    D22       AND ENTER DIGIT INTO MNATISSA
00591 573E C6 20    D28    LDA B  #$20      GET "LEADING ZERO" FLAG
00592 5740 D7 0F           STA B  DIGFLG    SAVE IT IN D.E. FLAG
00593 5742 20 9D           BRA    D27       RETURN TO SUPV.
00595 5744 0007            RMB    7
00598                 *
00599                 *    MATH FUNCTION ACCESS   +-*/
00600                 *
00601                 *    JOHN KEITH    APRIL 3, 1974    REV A
00602                 *
00603 5748 8D 0C    M1     BSR    SETUP     SAVE X, LOAD INDEX
00604 574D BD 75FC         JSR    FPA       ADD
00605 5750 20 13           BRA    M28       GO TO COMMON EXIT
00606 5752 8D 05    M10    BSR    SETUP     SAVE X, LOAD INDEX
00607 5754 BD 75F6         JSR    FPS       SUBTRACT
00608 5757 20 0C           BRA    M28       GO TO COMMON EXIT
00609 5759 BD 55E9  SETUP  JSR    TXL       TRANSFER X TO LST X
00610 575C CE 0098         LDX    #YR       SET INDEX REG TO Y
00611 575F 39              RTS              RETURN
00612 5760 8D F7    M20    BSR    SETUP     SAVE X, LOAD INDEX
00613 5762 BD 7735         JSR    FPM       MULTIPLY
00614 5765 7E 55DA  M28    JMP    PSD       DROP STACK
00615 5768 8D EF    M30    BSR    SETUP     SAVE X, LOAD INDEX
00616 576A BD 7793         JSR    FPD       DIVIDE
00617 576D 20 F6           BRA    M28       GO TO COMMON EXIT
00618 576F BD 55E9  M40    JSR    TXL       TRANSFER X TO LST X
00619 5772 7E 6BA9         JMP    TAN       DO TAN X
00620 5775 BD 55E9  M50    JSR    TXL       TRANSFER X TO LST X
00621 5778 7E 69C3         JMP    ATN       DO ATAN X
00622 577B BD 55E9  M60    JSR    TXL       SAVE LAST X
00623 577E 7E 6E65         JMP    SQRT      DO SQRT X
00624 5781 BD 55E9  M70    JSR    TXL       SAVE LAST X
00625 5784 7E 6AC9         JMP    EXPN      DO EXPN X
00626 5787 BD 55E9  M80    JSR    TXL       SAVE LAST X
```

```
00627 578A 7E 6A58          JMP     NTLN        DO LN X
00628 578D BD 55E9 M100     JSR     TXL         SAVE X IN LAST X
00629 5790 7E 73E6          JMP     RECIP       DO 1/X
00630 5793 BD 55E9 M110     JSR     TXL         SAVE X IN LAST X
00631 5796 7E 6B94          JMP     SIN         DO SIN X
00632 5799 BD 55E9 M120     JSR     TXL         SAVE X IN LAST X
00633 579C 7E 6BF2          JMP     ASIN        DO ASIN X
00634 579F BD 55E9 M130     JSR     TXL         SAVE X LAST X.
00635 57A2 7E 6B9A          JMP     COS         DO COS X
00636 57A5 BD 55E9 M140     JSR     TXL         SAVE X IN LAST X
00637 57A8 7E 6BF7          JMP     ACOS        DO ACOS X
00638 57AB BD 55E9 M150     JSR     TXL         X TO LAST X
00639 57AE 7E 6F52          JMP     IOUPX       DO 10**X
00640 57B1 BD 55E9 M160     JSR     TXL         X TO LAST X
00641 57B4 7E 6FE9          JMP     YUPX        DO Y**X
00642 57B7 BD 55E9 M170     JSR     TXL         X TO LAST X
00643 57BA 7E 6FA7          JMP     LOG10       DO LOG X
00645                       *
00646                       *     SUBROUTINE TO LOAD MESSAGES INTO BUFFER
00647                       *
00648                       *     JOHN KEITH    APRIL 23, 1974    REV A
00649                       *
00650       0016    BFPTR   EQU     TP2
00651       0018    MSGPTR  EQU     TP3
00652 57BD E6 00    LDMSG2  LDA B   X           LOAD FIRST CHARACTER
00653 57BF 08       MSGLP   INX                 INCREMENT MESSAGE PTR
00654 57C0 DF 18            STX     MSGPTR      STORE MESSAGE PTR
00655 57C2 DE 16            LDX     BFPTR       LOAD INDEX WITH BUFFER PTR
00656 57C4 C4 7F            AND B   #$7F        ZERO MSB
00657 57C6 4D               TST A               LIST ROUTINE CALL?
00658 57C7 26 02            BNE     NTLST       NO, ERROR MESSAGE
00659 57C9 CB 20            ADD B   #$20        ADD OFFSET BACK TO GET ASCII
00660 57CB E7 00    NTLST   STA B   X           STORE CHARACTER IN BUFFER
00661 57CD 08               INX                 INCR BUFFER PTR
00662 57CE DF 16            STX     BFPTR       SAVE IT
00663 57D0 DE 18            LDX     MSGPTR      RELOAD MESSAGE PTR
00664 57D2 E6 00            LDA B   X           LOAD ANOTHER CHARACTER
00665 57D4 2A E9            BPL     MSGLP       DONE LOADING MESSAGE?
00666 57D6 39               RTS                 YES, RETURN
00668                       *
00669                       *     ENTER PI 3.14159265360
00670                       *
00671                       *     JOHN KEITH    APRIL 3, 1974    REV A
00672                       *
00673 57D7 91 0F    ENTPI1  CMP A   DIGFLG       DOING DIGIT ENTRY ?
00674 57D9 27 05            BEQ     ENTPI2      BRANCH IF NOT
00675 57DB 97 0F            STA A   DIGFLG       ELSE TERMINATE IT
00676 57DD 43               COM A               SET MSB
00677 57DE 97 0D            STA A   STKFLG       ENABLE LIFT
00678 57E0 BD 55EF ENTPI2   JSR     STKUP       DO STACK UP IF ENABLED
00679 57E3 CE 57E9          LDX     #PI         SET INDEX REGISTER TO PI
00680 57E6 7E 7438          JMP     TXXR        TRANSFER PI
00681 57E9 0000     PI      FDB     $0000       FORM PI IN ROM
00682 57EB 3141             FDB     $3141
00683 57ED 5926             FDB     $5926
00684 57EF 5360             FDB     $5360
00686                       *
00687                       *ROLL UP ROUTINE
00688                       *
00689 57F1 BD 55B2 ROLLUP   JSR     ROLLD       DO 3 ROLL DOWNS AND RETURN
00690 57F4 BD 55B2          JSR     ROLLD
00691 57F7 7E 55B2          JMP     ROLLD
00692                       *
00693                       *LINE FEED ROUTINE
00694                       *
00695 57FA BD 5D75 LF       JSR     BLANK       BLANK THE PRINT BUFFER
00696 57FD 7E 6034          JMP     PRTDRV+7    PRINT BLANKS
00698 7C2E                  ORG     $7C2E
00699 7C2E 57D7             FDB     ENTPI1
00700 7C5A                  ORG     $7C5A
00701 7C5A 556F             FDB     INTFR5
00702 7C26                  ORG     $7C26
00703 7C26 5598             FDB     CLXR1
00704 7C20                  ORG     $7C20
00705 7C20 55A6             FDB     ENTER1
00706 7C2A                  ORG     $7C2A
```

```
00707 7C2A 55AC          FDB     XEY1
00708 7C2C               ORG     $7C2C
00709 7C2C 5582          FDB     ROLLD1
00710 7C30               ORG     $7C30
00711 7C30 55C9          FDB     RLSTX1
00712 7C28               ORG     $7C28
00713 7C28 5611          FDB     CLR1
00714 7D64               ORG     $7D64
00715 7D64 5624          FDB     PRT1
00716 7C72               ORG     $7C72
00717 7C72 562C          FDB     PTSTK1
00718 7C22               ORG     $7C22
00719 7C22 56A9          FDB     EEX
00720 7C24 5641          FDB     CHS
00721 7C02               ORG     $7C02
00722 7C02 56EC          FDB     DIGIT0
00723 7C04 56E8          FDB     DIGIT1
00724 7C06 56EA          FDB     DIGIT2
00725 7C08 56E9          FDB     DIGIT3
00726 7C0A 56E8          FDB     DIGIT4
00727 7C0C 56E7          FDB     DIGIT5
00728 7C0E 56E6          FDB     DIGIT6
00729 7C10 56E5          FDB     DIGIT7
00730 7C12 56E4          FDB     DIGIT8
00731 7C14 56E3          FDB     DIGIT9
00732 7C16 5687          FDB     DECPNT
00733 7C18               ORG     $7C18
00734 7C18 574B          FDB     M1
00735 7C1A 5752          FDB     M10
00736 7C1C 5760          FDB     M20
00737 7C1E 5760          FDB     M30
00738 7C32               ORG     $7C32
00739 7C32 5793          FDB     M110,M130,M40,M120,M140,M50
00740 7C3E 5787          FDB     M80
00741 7C40 57B7          FDB     M170
00742 7C4A               ORG     $7C4A
00743 7C4A 5778          FDB     M60
00744 7C4C 578D          FDB     M100
00745 7C42               ORG     $7C42
00746 7C42 5781          FDB     M70
00747 7C44 57AB          FDB     M150
00748 7C46 5781          FDB     M160
00749 7C48               ORG     $7C48
00750 7C48 57F1          FDB     ROLLUP
00751 7C7A               ORG     $7C7A
00752 7C7A 57FA          FDB     LF
00755                    END
```

SYMBOL TABLE

ADATA	0000	ACTL	0001	BDATA	0002	BCTL	0003	INPUT	0004	IOIN	0005
ERROR	0006	TGL	0007	UFLG	0008	RSFLG	0009	EOM	000A	EOPM	000B
STKFLG	000D	RND	000E	DIGFLG	000F	W2	0010	W1	0011	SFLG	0012
DCNTR	0013	TP1	0014	TP1S	0015	TP2	0016	TP2S	0017	TP3	0018
TP3S	0019	TP4	001A	TP4S	001B	TP5	001C	TP5S	001D	TP6	001E
TP6S	001F	TP7	0020	TP7S	0021	T13	0022	T12	0023	T11	0024
T10	0025	T9	0026	T8	0027	T7	0028	T6	0029	T5	002A
T4	002B	T3	002C	T2	002D	T1	002E	ISTK	002F	ISTACK	0051
TA	0052	SPGM	0054	EXTRA	0056	BUFF	0058	REAL	0068	IMAG	0070
AT1	007B	AT2	0080	W	0088	XR	0090	YR	0098	ZR	00A0
TR	00A8	LSTX	00B0	BKWRT	00B8	BKKC	00BA	SOL7	00C6	UPP	00C8
UIP	00CA	ALPHA	00CC	IO1	00CD	IO2	00D0	IT7	00D3	FLAG	00D5
TPOS	00D6	FILE	00D7	AR	00D8	BR	00E0	CR	00E8	DR	00F0
ER	00F8	SDBB	00BA	MT	7E00	TERMN7	003D	IMED	0040	PARCD	00C0
PAREX	0080	NTBL	0000	DOTS	5EC0	PRTDRV	602D	FRMT	5CA8	BLANK	5D75
LDMSG	5780	ROLLD	5582	ROLLU	57F1	PSD	550A	TXL	55E9	STKUP	55EF
MAD	749B	CMP	74AA	NOR	74D6	TXW	7424	TXXR	7438	EXXR	7452
ARSR	753B	OVUNF	75B6	OVERF	75DD	XR0	740A	XRNINE	75C8	UNDRF	75F1
IMULT	7689	QDG	7669	FPA	75FC	FPS	75F6	FPM	7735	FPD	7793
FPAEX	763D	FPMEX	7780	LSHIFT	7521	ZEROX	7489	XZERO0	7416	XZERO2	7417
RECIP	73E6	TXRX	73F3	CONST	6800	FPDBRC	6898	TAN	68A9	ATN	69C3
DSZERO	6A46	NTLN	6A58	EXPN	6AC9	SIN	6894	COS	689A	ASIN	68F2
ACOS	68F7	PH1	6C5D	PH2	6C8D	PH3	6D34	PH4	6DD0	LSFT8	6E47

SQRT	6E65	MAD8	6F2C	CMPR	53E4	IOUPX	6F52	LOG10	6FA7	YUPX	6FE9
RTOP	7328	PTOR	7386	REGPTR	0014	INTFR1	5550	INTFR2	555C	INTFR3	5564
INTFR4	556C	INTFR5	556F	INTFR6	5579	INTFR7	5590	INTFR9	559A	CLXP1	559B
LOEX	55A0	ENTER1	55A6	XEY1	55AC	ROLLD1	5582	ROLLD5	5585	RLSTX1	55C9
RLSTX2	55D2	PSD1	55DA	PSD5	55DD	STEMP	0020	STKUP1	55EF	STKUP2	55F4
STKUP3	55FA	STKUP9	5610	SAVE	0014	CLR1	5611	CLR3	561A	PRT1	5624
TEMP	002E	PTSTK1	562C	PTSTK3	5632	EXP	0098	EXP1	0089	DTEMP	0014
EXP2	008A	BWM2	7982	CHS	5641	D1	5651	D3	5657	D5	5660
D7	5663	D8	566C	D9	5671	DECPNT	5687	D11	568D	D15	5698
DEXIT	569C	DEXIT1	569F	FIX1	56A4	EEX	56A9	D22	56C4	D23	56CC
D25	56D3	D27	56E1	DIGIT9	56E3	DIGIT8	56E4	DIGIT7	56E5	DIGIT6	56E6
DIGIT5	56E7	DIGIT4	56E8	DIGIT3	56E9	DIGIT2	56EA	DIGIT1	56EB	DIGIT0	56EC
D35	5701	D36	570A	D39	5711	D43	5723	D45	5730	D47	5733
D28	573E	M1	5748	M10	5752	SETUP	5759	M20	5760	M28	5765
M30	5768	M40	576F	M50	5775	M60	5779	M70	5781	M80	5787
M100	578D	M110	5793	M120	5799	M130	579F	M140	57A5	M150	57AB
M160	57B1	M170	57B7	BFPTR	0016	MSGPTR	0018	LDMSG2	57BD	MSGLP	57BF
NTLST	57CB	ENTPI1	57D7	ENTPI2	57E0	PI	57E9	ROLLUP	57F1	LF	57FA

```
00215                          OPT    LIST,MEM
00218 5800                     ORG    $5800
00219 5800 2E        TABLE     FCB    $2E       NOP
00220 5801 2F                  FCB    $2F
00221 5802 30                  FCB    $30
00222 5803 90                  FCB    $90       ZERO
00223 5804 91                  FCB    $91       ONE
00224 5805 92                  FCB    $92       TWO
00225 5806 93                  FCB    $93       THREE
00226 5807 94                  FCB    $94       FOUR
00227 5808 95                  FCB    $95       FIVE
00228 5809 96                  FCB    $96       SIX
00229 580A 97                  FCB    $97       SEVEN
00230 580B 98                  FCB    $98       EIGHT
00231 580C 99                  FCB    $99       NINE
00232 580D 8E                  FCB    $8E       DECIMAL POINT
00233 580E 8B                  FCB    $8B       PLUS SIGN
00234 580F 8D                  FCB    $8D       MINUS SIGN
00235 5810 8A                  FCB    $8A       ASTERISK (MULTIPLY)
00236 5811 8B                  FCB    $8B       DIVIDE SIGN
00237 5812 A5                  FCB    $A5       ENTER
00238 5813 2E                  FCB    $2E
00239 5814 34                  FCB    $34
00240 5815 25                  FCB    $25
00241 5816 32                  FCB    $32
00242 5817 3E                  FCB    $3E
00243 5818 A5                  FCB    $A5       ENTER EXPONENT
00244 5819 25                  FCB    $25
00245 581A 38                  FCB    $38
00246 581B 8B                  FCB    $8B       CHANGE SIGN
00247 581C 01                  FCB    $1
00248 581D 0D                  FCB    $D
00249 581E A3                  FCB    $A3       CLEAR X
00250 581F 2C                  FCB    $2C
00251 5820 38                  FCB    $38
00252 5821 A3                  FCB    $A3       CLEAR
00253 5822 2C                  FCB    $2C
00254 5823 25                  FCB    $25
00255 5824 21                  FCB    $21
00256 5825 32                  FCB    $32
00257 5826 B8                  FCB    $B8       X EXCHANGE Y
00258 5827 01                  FCB    $1
00259 5828 39                  FCB    $39
00260 5829 B2                  FCB    $B2       ROLL DOWN
00261 582A 2F                  FCB    $2F
00262 582B 2C                  FCB    $2C
00263 582C 2C                  FCB    $2C
00264 582D 3F                  FCB    $3F
00265 582E 8D                  FCB    $8D       PI
00266 582F AC                  FCB    $AC       LAST X
00267 5830 33                  FCB    $33
00268 5831 34                  FCB    $34
00269 5832 38                  FCB    $38
00270 5833 83                  FCB    $83       SIN
00271 5834 29                  FCB    $29
```

```
00272 5835 2E          FCB     $2E
00273 5836 A3          FCB     $A3       COS
00274 5837 2F          FCB     $2F
00275 5838 33          FCB     $33
00276 5839 B4          FCB     $B4       TAN
00277 583A 21          FCB     $21
00278 583B 2E          FCB     $2E
00279 583C A1          FCB     $A1       ASIN
00280 583D 33          FCB     $33
00281 583E 29          FCB     $29
00282 583F 2E          FCB     $2E
00283 5840 A1          FCB     $A1       ACOS
00284 5841 23          FCB     $23
00285 5842 2F          FCB     $2F
00286 5843 33          FCB     $33
00287 5844 A1          FCB     $A1       ATAN
00288 5845 34          FCB     $34
00289 5846 21          FCB     $21
00290 5847 2E          FCB     $2E
00291 5848 AC          FCB     $AC       LN
00292 5849 2E          FCB     $2E
00293 584A AC          FCB     $AC       LOG
00294 584B 2F          FCB     $2F
00295 584C 27          FCB     $27
00296 584D B2          FCB     $B2       E^X
00297 584E 3E          FCB     $3E
00298 584F 38          FCB     $38
00299 5850 91          FCB     $91       10^X
00300 5851 10          FCB     $10
00301 5852 3E          FCB     $3E
00302 5853 38          FCB     $38
00303 5854 B9          FCB     $B9       Y^X
00304 5855 3E          FCB     $3E
00305 5856 38          FCB     $38
00306 5857 B2          FCB     $B2       ROLL UP
00307 5858 2F          FCB     $2F
00308 5859 2C          FCB     $2C
00309 585A 2C          FCB     $2C
00310 585B 3E          FCB     $3E
00311 585C B3          FCB     $B3       SQUARE ROOT
00312 585D 31          FCB     $31
00313 585E 32          FCB     $32
00314 585F 34          FCB     $34
00315 5860 91          FCB     $91       1/X
00316 5861 0F          FCB     $F
00317 5862 38          FCB     $38
00318 5863 B3          FCB     $B3       SUM+
00319 5864 35          FCB     $35
00320 5865 2D          FCB     $2D
00321 5866 0B          FCB     $B
00322 5867 B3          FCB     $B3       SUM-
00323 5868 35          FCB     $35
00324 5869 2D          FCB     $2D
00325 586A 0D          FCB     $D
00326 586B B2          FCB     $B2       RECT TO POLAR
00327 586C 3C          FCB     $3C
00328 586D 30          FCB     $30
00329 586E B0          FCB     $B0       POLAR TO RECT
00330 586F 3C          FCB     $3C
00331 5870 32          FCB     $32
00332 5871 A1          FCB     $A1       ACC+
00333 5872 23          FCB     $23
00334 5873 23          FCB     $23
00335 5874 0B          FCB     $B
00336 5875 A1          FCB     $A1       ACC-
00337 5876 23          FCB     $23
00338 5877 23          FCB     $23
00339 5878 0D          FCB     $D
00340 5879 A9          FCB     $A9       INTEGER
00341 587A 2E          FCB     $2E
00342 587B 34          FCB     $34
00343 587C A4          FCB     $A4       DEG
00344 587D 25          FCB     $25
00345 587E 27          FCB     $27
00346 587F 33          FCB     $33
00347 5880 BC          FCB     $BC       TO D.MS
```

```
00348 5881 24          FCB    $24
00349 5882 0E          FCB    $E
00350 5883 2D          FCB    $2D
00351 5884 33          FCB    $33
00352 5885 A4          FCB    $A4      FROM D.MS
00353 5886 0E          FCB    $E
00354 5887 2D          FCB    $2D
00355 5888 33          FCB    $33
00356 5889 3C          FCB    $3C
00357 588A A9          FCB    $A9      IF +
00358 588B 26          FCB    $26
00359 588C 00          FCB    $0
00360 588D 0B          FCB    $B
00361 588E A9          FCB    $A9      IF -
00362 588F 26          FCB    $26
00363 5890 00          FCB    $0
00364 5891 0D          FCB    $D
00365 5892 A9          FCB    $A9      IF 0
00366 5893 26          FCB    $26
00367 5894 00          FCB    $0
00368 5895 10          FCB    $10
00369 5896 A9          FCB    $A9      IF X=Y
00370 5897 26          FCB    $26
00371 5898 00          FCB    $0
00372 5899 38          FCB    $38
00373 589A 1D          FCB    $1D
00374 589B 39          FCB    $39
00375 589C A9          FCB    $A9      IF X<Y
00376 589D 26          FCB    $26
00377 589E 00          FCB    $0
00378 589F 38          FCB    $38
00379 58A0 1C          FCB    $1C
00380 58A1 39          FCB    $39
00381 58A2 A9          FCB    $A9      IF X>=Y
00382 58A3 26          FCB    $26
00383 58A4 00          FCB    $0
00384 58A5 38          FCB    $38
00385 58A6 1B          FCB    $1B
00386 58A7 39          FCB    $39
00387 58A8 AD          FCB    $AD      MEAN AND STANDARD DEVIATION
00388 58A9 2E          FCB    $2E
00389 58AA 06          FCB    $6
00390 58AB 33          FCB    $33
00391 58AC 24          FCB    $24
00392 58AD 83          FCB    $83      NO. OF REGISTERS
00393 58AE 32          FCB    $32
00394 58AF 25          FCB    $25
00395 58B0 27          FCB    $27
00396 58B1 33          FCB    $33
00397 58B2 B0          FCB    $B0      PRINT STACK
00398 58B3 32          FCB    $32
00399 58B4 34          FCB    $34
00400 58B5 33          FCB    $33
00401 58B6 34          FCB    $34
00402 58B7 28          FCB    $28
00403 58B8 B2          FCB    $B2      ROUND
00404 58B9 2F          FCB    $2F
00405 58BA 35          FCB    $35
00406 58BB 2E          FCB    $2E
00407 58BC 24          FCB    $24
00408 58BD B2          FCB    $B2      RCL ACC+
00409 58BE 00          FCB    $0
00410 58BF 21          FCB    $21
00411 58C0 23          FCB    $23
00412 58C1 23          FCB    $23
00413 58C2 0B          FCB    $B
00414 58C3 A3          FCB    $A3      CLEAR ALPHA REGISTERS
00415 58C4 2C          FCB    $2C
00416 58C5 32          FCB    $32
00417 58C6 21          FCB    $21
00418 58C7 3C          FCB    $3C
00419 58C8 2A          FCB    $2A
00420 58C9 B3          FCB    $B3      SPACE
00421 58CA 30          FCB    $30
00422 58CB 21          FCB    $21
```

```
00423 58CC 23          FCB   $23
00424 58CD 25          FCB   $25
00425 58CE A7          FCB   $A7      GRADS
00426 58CF 32          FCB   $32
00427 58D0 21          FCB   $21
00428 58D1 24          FCB   $24
00429 58D2 33          FCB   $33
00430 58D3 B2          FCB   $B2      RADS
00431 58D4 21          FCB   $21
00432 58D5 24          FCB   $24
00433 58D6 33          FCB   $33
00434 58D7 C8          FCB   $C8
00435 58D8 33          FCB   $33      SFG
00436 58D9 26          FCB   $26
00437 58DA 27          FCB   $27
00438 58DB C8          FCB   $C8
00439 58DC 29          FCB   $29      IF SFG
00440 58DD 26          FCB   $26
00441 58DE 00          FCB   $0
00442 58DF 33          FCB   $33
00443 58E0 26          FCB   $26
00444 58E1 27          FCB   $27
00445 58E2 C8          FCB   $C8
00446 58E3 23          FCB   $23      CFG
00447 58E4 26          FCB   $26
00448 58E5 27          FCB   $27
00449 58E6 C8          FCB   $C8
00450 58E7 29          FCB   $29      IF CFG
00451 58E8 26          FCB   $26
00452 58E9 00          FCB   $0
00453 58EA 23          FCB   $23
00454 58EB 26          FCB   $26
00455 58EC 27          FCB   $27
00456 58ED A7  TABLE1  FCB   $A7      GO TO
00457 58EE 2F          FCB   $2F
00458 58EF 34          FCB   $34
00459 58F0 2F          FCB   $2F
00460 58F1 A7          FCB   $A7      GOSUB
00461 58F2 2F          FCB   $2F
00462 58F3 33          FCB   $33
00463 58F4 35          FCB   $35
00464 58F5 22          FCB   $22
00465 58F6 C1          FCB   $C1      GOTO
00466 58F7 80          FCB   $80
00467 58F8 C1          FCB   $C1      GOSUB
00468 58F9 81          FCB   $81
00469 58FA CA          FCB   $CA
00470 58FB 33          FCB   $33      STORE
00471 58FC 34          FCB   $34
00472 58FD 2F          FCB   $2F
00473 58FE CA          FCB   $CA
00474 58FF 33          FCB   $33      STORE+
00475 5900 34          FCB   $34
00476 5901 2F          FCB   $2F
00477 5902 0B          FCB   $B
00478 5903 CA          FCB   $CA
00479 5904 33          FCB   $33      STORE-
00480 5905 34          FCB   $34
00481 5906 2F          FCB   $2F
00482 5907 0D          FCB   $D
00483 5908 CA          FCB   $CA
00484 5909 33          FCB   $33      STORE*
00485 590A 34          FCB   $34
00486 590B 2F          FCB   $2F
00487 590C 0A          FCB   $A
00488 590D CA          FCB   $CA
00489 590E 33          FCB   $33      STORE/
00490 590F 34          FCB   $34
00491 5910 2F          FCB   $2F
00492 5911 3B          FCB   $3B
00493 5912 CA          FCB   $CA
00494 5913 32          FCB   $32      RECALL
00495 5914 23          FCB   $23
00496 5915 2C          FCB   $2C
00497 5916 C5          FCB   $C5
00498 5917 23          FCB   $23      5 MAINFRAME CALLS
```

```
00499 5918 21           FCB     $21
00500 5919 2C           FCB     $2C
00501 591A 2C           FCB     $2C
00502 591B B6           FCB     $B6       VERIFY
00503 591C 25           FCB     $25
00504 591D 32           FCB     $32
00505 591E 29           FCB     $29
00506 591F 26           FCB     $26
00507 5920 39           FCB     $39
00508 5921 AC           FCB     $AC       LOAD & GO
00509 5922 24           FCB     $24
00510 5923 06           FCB     $6
00511 5924 27           FCB     $27
00512 5925 2F           FCB     $2F
00513 5926 B2           FCB     $B2       RECORD DATA (RCDATA)
00514 5927 23           FCB     $23
00515 5928 24           FCB     $24
00516 5929 21           FCB     $21
00517 592A 34           FCB     $34
00518 592B 21           FCB     $21
00519 592C B2           FCB     $B2       REWIND
00520 592D 25           FCB     $25
00521 592E 37           FCB     $37
00522 592F 29           FCB     $29
00523 5930 2E           FCB     $2E
00524 5931 24           FCB     $24
00525 5932 B0           FCB     $B0       PAUSE
00526 5933 21           FCB     $21
00527 5934 35           FCB     $35
00528 5935 33           FCB     $33
00529 5936 25           FCB     $25
00530 5937 C3           FCB     $C3
00531 5938 26           FCB     $26       FOR
00532 5939 2F           FCB     $2F
00533 593A 32           FCB     $32
00534 593B C3           FCB     $C3
00535 593C 2E           FCB     $2E       NEXT
00536 593D 25           FCB     $25
00537 593E 38           FCB     $38
00538 593F 34           FCB     $34
00539 5940 B3           FCB     $B3       STOP
00540 5941 34           FCB     $34
00541 5942 2F           FCB     $2F
00542 5943 30           FCB     $30
00543 5944 A5           FCB     $A5       END
00544 5945 2E           FCB     $2E
00545 5946 24           FCB     $24
00546 5947 B0           FCB     $B0       PRINT
00547 5948 32           FCB     $32
00548 5949 29           FCB     $29
00549 594A 2E           FCB     $2E
00550 594B 34           FCB     $34
00551 594C B2           FCB     $B2       RETURN
00552 594D 25           FCB     $25
00553 594E 34           FCB     $34
00554 594F 35           FCB     $35
00555 5950 32           FCB     $32
00556 5951 2E           FCB     $2E
00557 5952 A5           FCB     $A5       END ALPHA
00558 5953 2E           FCB     $2E
00559 5954 24           FCB     $24
00560 5955 48           FCB     $48
00561 5956 B2           FCB     $B2       RECORD PROGRAM (RCPRGM)
00562 5957 23           FCB     $23
00563 5958 30           FCB     $30
00564 5959 32           FCB     $32
00565 595A 27           FCB     $27
00566 595B 2D           FCB     $2D
00567 595C B2           FCB     $B2       RECORD SECURED (RCSCRD)
00568 595D 23           FCB     $23
00569 595E 33           FCB     $33
00570 595F 25           FCB     $25
00571 5960 23           FCB     $23
00572 5961 AD           FCB     $AD       MARK
00573 5962 21           FCB     $21
00574 5963 32           FCB     $32
```

```
00575 5964 28           FCB     $28
00576 5965 A9           FCB     $A9     IDENTIFY
00577 5966 24           FCB     $24
00578 5967 25           FCB     $25
00579 5968 2E           FCB     $2E
00580 5969 34           FCB     $34
00581 596A AC           FCB     $AC     LOAD
00582 596B 2F           FCB     $2F
00583 596C 21           FCB     $21
00584 596D 24           FCB     $24
00585 596E 80           FCB     $80     1 EMPTY SLOT
00586 596F A6           FCB     $A6     FIX
00587 5970 29           FCB     $29
00588 5971 38           FCB     $38
00589 5972 83           FCB     $83     SCI
00590 5973 23           FCB     $23
00591 5974 29           FCB     $29
00592 5975 83           FCB     $83     SCI3
00593 5976 23           FCB     $23
00594 5977 29           FCB     $29
00595 5978 13           FCB     $13
00596 5979 C1           FCB     $C1
00597 597A C0           FCB     $C0
00598 597B AC           FCB     $AC     LABEL
00599 597C 22           FCB     $22
00600 597D 2C           FCB     $2C
00601 597E C1           FCB     $C1     GOSUB LBL
00602 597F 81           FCB     $81
00603 5980 C1           FCB     $C1     GOTO LBL
00604 5981 80           FCB     $80
00605 5982 82           FCB     $82     RECALL INDIRECT
00606 5983 23           FCB     $23
00607 5984 2C           FCB     $2C
00608 5985 00           FCB     $0
00609 5986 29           FCB     $29
00610 5987 83           FCB     $83     STORE INDIRECT
00611 5988 34           FCB     $34
00612 5989 2F           FCB     $2F
00613 598A 00           FCB     $0
00614 598B 29           FCB     $29
00615 598C 83           FCB     $83     STO+ INDIRECT
00616 598D 34           FCB     $34
00617 598E 2F           FCB     $2F
00618 598F 08           FCB     $8
00619 5990 00           FCB     $0
00620 5991 29           FCB     $29
00621 5992 83           FCB     $83     STO- INDIRECT
00622 5993 34           FCB     $34
00623 5994 2F           FCB     $2F
00624 5995 0D           FCB     $D
00625 5996 00           FCB     $0
00626 5997 29           FCB     $29
00627 5998 83           FCB     $83     STO* INDIRECT
00628 5999 34           FCB     $34
00629 599A 2F           FCB     $2F
00630 599B 0A           FCB     $A
00631 599C 00           FCB     $0
00632 599D 29           FCB     $29
00633 599E 83           FCB     $83     STO/ INDIRECT
00634 599F 34           FCB     $34
00635 59A0 2F           FCB     $2F
00636 59A1 38           FCB     $38
00637 59A2 00           FCB     $0
00638 59A3 29           FCB     $29
00639 59A4 C2           FCB     $C2     2 STORES
00640 59A5 85           FCB     $85
00641 59A6 C2           FCB     $C2     2 STO+
00642 59A7 8E           FCB     $8E
00643 59A8 C2           FCB     $C2     2 STO-
00644 59A9 98           FCB     $98
00645 59AA C2           FCB     $C2     2 STO*
00646 59AB A2           FCB     $A2
00647 59AC C2           FCB     $C2     2 STO/
00648 59AD AC           FCB     $AC
00649 59AE C2           FCB     $C2     2 RECALLS
00650 59AF B6           FCB     $B6
```

```
00651 59B0 C2              FCB    $C2       2 RCL IND
00652 59B1 E2              FCB    $E2
00653 59B2 C2              FCB    $C2       2 STO IND
00654 59B3 E3              FCB    $E3
00655 59B4 C2              FCB    $C2       2 STO+ IND
00656 59B5 E4              FCB    $E4
00657 59B6 C2              FCB    $C2       2 STO- IND
00658 59B7 E5              FCB    $E5
00659 59B8 C2              FCB    $C2       2 STO* IND
00660 59B9 E6              FCB    $E6
00661 59BA C2              FCB    $C2       2 STO/ IND
00662 59BB E7              FCB    $E7
00663 59BC D0              FCB    $D0       16 GOSUBS
00664 59BD 81              FCB    $81
00665 59BE D0              FCB    $D0       16 GOTOS
00666 59BF 80              FCB    $80
00667 59C0 D0       FTBL   FCB    $D0       PRINT ALPHA
00668 59C1 52              FCB    $52
00669 59C2 4E              FCB    $4E
00670 59C3 54              FCB    $54
00671 59C4 68              FCB    $68
00672 59C5 D4       TSNL   FCB    $D4       TAB
00673 59C6 41              FCB    $41
00674 59C7 42              FCB    $42
00675 59C8 20              FCB    $20
00676 59C9 20              FCB    $20
00677 59CA CC              FCB    $CC       LINE
00678 59CB 49              FCB    $49
00679 59CC 4E              FCB    $4E
00680 59CD 45              FCB    $45
00681 59CE 20              FCB    $20
00682 59CF D3              FCB    $D3       SHIFT
00683 59D0 48              FCB    $48
00684 59D1 49              FCB    $49
00685 59D2 46              FCB    $46
00686 59D3 54              FCB    $54
00688                      *
00689                      *
00690                      * THIS ROUTINE PROVIDES A PROGRAM LISTING
00691                      * FROM WHERE THE PROGRAM COUNTER IS
00692                      * SETTING UNTIL AN "END" STATEMENT IS REACHED
00693                      * OR UNTIL THE RUN/STOP KEY IS HIT.  IT ALSO
00694                      * PROVIDES A KEYLOG IN THE PROGRAM MODE
00695                      * WHEN THE PRINT ALL SWITHCH IS ON.
00696                      *
00697                      *
00698                      * DAVE UHLRICH   OCTOBER 15,1974   LIST   REV I
00699                      *
00700                      *
00701       797B    BWM    EQU    $797B
00702       001C    SVINS  EQU    TP5
00703       001D    SAVE   EQU    TP5S
00704       0014    GIOFLG EQU    TP1
00705       0015    LSTFLG EQU    TP1S
00706       0018    MSGPTR EQU    TP3
00707       0016    BFPTR  EQU    TP2
00708       48BC    RINBCD EQU    $48BC
00709       4EA4    DGTS   EQU    $4EA4
00710       5300    ROMID  EQU    $5300
00711       57FA    LF     EQU    $57FA
00712       52F4    STEP1  EQU    $52F4
00713 59D4 8D 19    LSTNTR BSR    LIST
00714 59D6 7E 797B         JMP    BWM
00715 59D9 4C       GIONTR INC A             GIO ENTRY
00716 59DA 97 14    KEYLOG STA A  GIOFLG     STORE GIO FLAG
00717 59DC 4F              CLR A
00718 59DD 97 15           STA A  LSTFLG     SET LSTFLG=0
00719 59DF DE C8           LDX    UPP        SET INDEX TO USER PSTN PTR
00720 59E1 DF 8C           STX    W+4        SAVE UPP IN W+4
00721 59E3 A6 00           LDA A  X          LOAD INSTRUCTION
00722 59E5 97 1C           STA A  SVINS      SAVE IT IN SVINS
00723 59E7 97 1D           STA A  SAVE
00724 59E9 96 07           LDA A  TGL        AUDIT TRAIL?
00725 59EB 2A 2A           BPL    OKAY       YES, DON'T WORRY ABOUT SECURED MEMORY
00726 59ED 20 18           BRA    KYLG
```

```
00727 59EF 97 14    LIST    STA A   GIOFLG   STORE GIO FLAG
00728 59F1 BD 52F4          JSR     STEP1
00729 59F4 C6 C0    LSTOK   LDA B   #$C0      LOAD ACCB WITH NEW RSFLG
00730 59F6 D7 15            STA B   LSTFLG   SET LSTFLG .NE. 0
00731 59F8 DE C8            LDX     UPP      SET INDEX TO USER PSTN PTR
00732 59FA DF 8C            STX     W+4      SAVE UPP IN W+4
00733 59FC A6 00            LDA A   X        LOAD INSTRUCTION
00734 59FE 97 1C            STA A   SVINS    SAVE THE INSTRUCTION
00735 5A00 97 1D            STA A   SAVE
00736 5A02 81 B1            CMP A   #$B1     IS IT AN END INSTRUCTION?
00737 5A04 26 02            BNE     NTEND    NO. SET RSFLG=$C0
00738 5A06 C6 40            LDA B   #$40     YES, SET RSFLG=$40
00739 5A08 D7 09    NTEND   STA B   RSFLG    STORE NEW RUN/STOP FLAG
00740 5A0A 96 D5    KYLG    LDA A   FLAG
00741 5A0C 44               LSR A
00742 5A0D 24 08            BCC     OKAY     SECURED PROGRAM?
00743 5A0F 86 16    SPERR   LDA A   #22
00744 5A11 97 06            STA A   ERROR
00745 5A13 4F               CLR A
00746 5A14 97 09            STA A   RSFLG    RSFLG=0
00747 5A16 39               RTS
00748 5A17 BD 5D75  OKAY    JSR     BLANK    BLANK OUT BUFFER
00749 5A1A 7A 00C8          DEC     UPP
00750 5A1D DE C8            LDX     UPP      SET INDEX TO USER PSTN PTR
00751 5A1F 7C 00C8          INC     UPP
00752 5A22 BD 48BC          JSR     BINBCD   CONVERT PC TO BCD
00753 5A25 CE 0058          LDX     #BUFF    SET INDEX TO BEG. OF BUFFER
00754 5A28 86 10            LDA A   #16      SET ACCA=NO. OF DIGITS*4
00755 5A2A BD 4EA4          JSR     DGTS     PUT PC INTO BUFFER
00756 5A2D 96 1C            LDA A   SVINS    LOAD INSTRUCTION
00757 5A2F 08               INX              INCR BUFFER PTR
00758 5A30 08               INX              INCR BUFFER PTR AGAIN
00759 5A31 DF 16            STX     BFPTR    STORE BUFFER PTR
00760 5A33 7D 00CC          TST     ALPHA    ALPHA FLAG SET?
00761 5A36 27 26            BEQ     NTASC    NO, NOT ALPHA STRING
00762 5A38 4D               TST A            MSB SET?
00763 5A39 2F 14            BLE     ENDASC   YES. END ALPHA UNLESS PRINT
00764 5A3B 85 FC            BIT A   #$FC     TAB, SHIFT, OR LINE?
00765 5A3D 27 03            BEQ     TLS
00766 5A3F 7E 5AF1          JMP     STOOP    NO. STORE ASCII
00767 5A42 CE 59C5  TLS     LDX     #TSNL    YES. LOAD TABLE ADDRESS
00768 5A45 DF 18            STX     MSGPTR
00769 5A47 16               TAB
00770 5A48 5A               DEC B            SET TABLE OFFSET IN ACCB
00771 5A49 BD 5B7E          JSR     PRTMSG   PRINT TAB. SHIFT. OR LINE
00772 5A4C 7E 5BF2  END2    JMP     END
00773 5A4F 81 B2    ENDASC  CMP A   #$B2     PRINT COMMAND?
00774 5A51 27 0B            BEQ     NTASC    YES, LIST IT
00775 5A53 81 B4            CMP A   #$B4     ALPHA TERMINATOR?
00776 5A55 27 04            BEQ     NALPHA   YES, END ALPHA
00777 5A57 81 B1            CMP A   #$B1     NO, IS IT AN "END"?
00778 5A59 26 F1            BNE     END2     NO, TREAT IT AS A BLANK
00779 5A5B 7F 00CC  NALPHA  CLR     ALPHA    YES. CLEAR ALPHA FLAG
00780 5A5E CE 5800  NTASC   LDX     #TABLE   SET INDEX TO START OF TABLE
00781 5A61 4F               CLR A            CLEAR INSTRUCTION CTR
00782 5A62 91 1D    CMPCTR  CMP A   SAVE     CTR=INSTR.?
00783 5A64 27 26            BEQ     FOUND    YES, FOUND ASCII
00784 5A66 08       NOCNT   INX              NO, INCR TABLE ADDR
00785 5A67 E6 00            LDA B   X        LOAD NEXT ASCII CHAR
00786 5A69 2A FB            BPL     NOCNT    MSB SET?
00787 5A6B C5 40            BIT B   #$40     BIT 6 SET?
00788 5A6D 26 03            BNE     MRTHN1   YES. MORE THAN ONE OF SPECIFIC INSTR
00789 5A6F 4C               INC A            INCR INSTR CTR
00790 5A70 20 F0            BRA     CMPCTR
00791 5A72 C4 3F    MRTHN1  AND B   #$3F     CLEAR TOP 2 BITS
00792 5A74 08               INX              INX TABLE ADDRESS
00793 5A75 4C       NOTYET  INC A            INCR INSTR CTR
00794 5A76 91 1D            CMP A   SAVE     CTR=INSTR.?
00795 5A78 27 05            BEQ     EQUAL    YES, ADDRESS FOUND
00796 5A7A 5A               DEC B            DECR MULTIPLE INSTR CTR
00797 5A7B 27 E9            BEQ     NOCNT    IF =0, NOT THIS MULTIPLY USED INSTR
00798 5A7D 20 F6            BRA     NOTYET   CONTINUE COUNTING
00799 5A7F E6 00    EQUAL   LDA B   X        LOAD BYTE AFTER BYTE WITH TOP 2 BITS
00800 5A81 2A 09            BPL     FOUND    MSB SET? NO, ADDRESS FOUND
00801 5A83 C4 7F            AND B   #$7F     YES, CLEAR THIS BIT
00802 5A85 D7 1D            STA B   SAVE     STORE NEW INSTR TO BE COUNTED TO
```

```
00803 5A87 CE 58ED        LDX    #TABLE1   LOAD PARTIAL TABLE STARTING ADDRESS
00804 5A8A 20 D5          BRA    CMPCTR-1    START SEARCH OVER AGAIN
00805 5A8C 4F       FOUND CLR A
00806 5A8D BD 57BD        JSR    LDMSG     LOAD MNEMONIC INTO BUFFER
00807 5A90 86 64          LDA A  #$64      SET BUFFER PTR TO OPERAND FIELD
00808 5A92 97 17          STA A  BFPTR+1    STORE IT
00809 5A94 DE 16          LDX    BFPTR     SET INDEX TO BUFFER PTR
00810 5A96 96 1C          LDA A  SVINS     LOAD INSTRUCTION
00811 5A98 81 3F          CMP A  #$3F      NO OPERAND INSTRUCTION?
00812 5A9A 23 57          BLS    DONE      YES. YOU ARE DONE
00813 5A9C 81 5F    TBL20 CMP A  #$5F      FLAG INSTRUCTION?
00814 5A9E 22 07          BHI    TBL21     NO. CHECK FURTHER
00815 5AA0 84 07          AND A  #$7       YES. MASK OFF BOTTOM 3 BITS
00816 5AA2 4C             INC A            ADD ONE TO THEM
00817 5AA3 8A 30          ORA A  #$30      CONVERT RESULT TO ASCII
00818 5AA5 20 4A          BRA    STOOP     STORE OPERAND INTO BUFFER
00819 5AA7 81 61    TBL21 CMP A  #$61      GOSUB/GOTO LBL X?
00820 5AA9 22 05          BHI    TBL22     NO. CHECK FURTHER
00821 5AAB C6 4C          LDA B  #$4C      YES, LOAD ASCII L
00822 5AAD E7 00          STA B  X         STORE IT IN BUFFER
00823 5AAF 08             INX              INCR BUFFER PTR
00824 5AB0 81 63    TBL22 CMP A  #$63      GOSUB/GOTO X?
00825 5AB2 22 04          BHI    TBL23     NO. CHECK FURTHER
00826 5AB4 86 58          LDA A  #$58      YES, LOAD ASCII X
00827 5AB6 20 39          BRA    STOOP     STORE OPERAND
00828 5AB8 81 A4    TBL23 CMP A  #$A4      STO+-*/ A-J OR RCL A-J OR CALL A-E
00829 5ABA 22 0A          BHI    TBL24     NO. CHECK FURTHER
00830 5ABC 80 19          SUB A  #$19      SUBTRACT OFFSET
00831 5ABE 80 0A    SUB10 SUB A  #10       SUBTRACT 10
00832 5AC0 81 4A          CMP A  #$4A      REACHED ASCII CODE FOR A THRU J?
00833 5AC2 22 FA          BHI    SUB10     NO. SUBTRACT 10 MORE
00834 5AC4 20 28          BRA    STOOP     YES. STORE OPERAND
00835 5AC6 81 A9    TBL24 CMP A  #$A9      NO OPERAND INSTRUCTIONS?
00836 5AC8 23 29          BLS    DONE      YES. ALL DONE
00837 5ACA 81 AF    TBL25 CMP A  #$AF      FOR/NEXT?
00838 5ACC 22 28          BHI    TBL30     NO. CHECK FURTHER
00839 5ACE C6 41          LDA B  #$41      LOAD ASCII A INTO ACCB
00840 5AD0 85 02          BIT A  #2        FOR A/NEXT A?
00841 5AD2 27 06          BEQ    LDABC     YES. OPERAND=A
00842 5AD4 5C             INC B            NO, ASCII B IN ACCB
00843 5AD5 85 01          BIT A  #1        FOR B/NEXT B?
00844 5AD7 27 01          BEQ    LDABC     YES. OPERAND=B
00845 5AD9 5C             INC B            NO. OPERAND=C
00846 5ADA E7 00    LDABC STA B  X         STORE OPERAND
00847 5ADC 81 AC          CMP A  #$AC      FOR INSTRUCTION?
00848 5ADE 22 13          BHI    DONE      YES. YOU ARE DONE
00849 5AE0 08             INX
00850 5AE1 86 5C          LDA A  #$5C      LOAD RIGHT ARROW
00851 5AE3 A7 00          STA A  X         STORE IT IN THE BUFFER
00852 5AE5 08             INX
00853 5AE6 CB 05          ADD B  #5        ADD 5 TO GET SECOND LETTER
00854 5AE8 E7 00          STA B  X         STORE SECOND CHARACTER
00855 5AEA C1 48          CMP B  #$48      WAS IT AN "H"?
00856 5AEC 26 05          BNE    DONE      NO. DONE
00857 5AEE 08             INX              YES. INCR BUFFER PTR
00858 5AEF 86 44          LDA A  #$44      LOAD ASCII "D"
00859 5AF1 A7 00    STOOP STA A  X         STORE OPERAND IN BUFFER
00860 5AF3 7E 58F2 DONE   JMP    END
00861 5AF6 81 BA    TBL30 CMP A  #$BA      NO OPERAND INSTRUCTION?
00862 5AF8 23 F9          BLS    DONE      YES. DONE
00863 5AFA 81 BF          CMP A  #$BF      IS INSTRUCTION A LABEL?
00864 5AFC 26 1C          BNE    NOLBL     NO, LOAD OPERAND
00865 5AFE 96 14          LDA A  GIOFLG
00866 5B00 26 52          BNE    ID25-1    GIO? YES. RETURN
00867 5B02 BD 6034        JSR    PRTDRV+7    PRINT LABEL WITHOUT OPERAND
00868 5B05 BD 5D75        JSR    BLANK     BLANK OUT BUFFER
00869 5B08 CE 005B        LDX    #BUFF     SET INDEX TO BEG OF BUFFER
00870 5B0B C6 2D          LDA B  #$2D      LOAD ACCB WITH ASCII MINUS SIGN
00871 5B0D E7 00    DASH  STA B  X         STORE DASHES
00872 5B0F 08             INX              INCR BUFFER PTR
00873 5B10 8C 005C        CPX    #$5C      4 DASHES?
00874 5B13 26 F8          BNE    DASH      NO. LOAD ANOTHER ONE INTO BUFFER
00875 5B15 08             INX              INCR BUFFER PTR
00876 5B16 DF 16          STX    BFPTR     STORE BUFFER PTR
00877 5B18 96 1C          LDA A  SVINS
00878 5B1A DE C8    NOLBL LDX    UPP       SET INDEX TO USER PSTN PTR
```

```
00879 581C 08              INX              POINT TO SECOND BYTE
00880 581D DF 8C           STX    W+4       SAVE 2ND BYTE PTR
00881 581F E6 00           LDA B  X         LOAD 2ND BYTE INTO ACCB
00882 5821 81 BE           CMP A  #$BE      FORMAT INSTRUCTION?
00883 5823 26 6A           BNE    NTFMT     NO. CHECK FURTHER
00884 5825 CE 59C0 FORMAT  LDX    #FTBL
00885 5828 DF 18           STX    MSGPTR
00886 582A 37              PSH B
00887 582B 5D              TST B
00888 582C 27 2F           BEQ    MNEM      PRINT ALPHA?
00889 582E C4 70           AND B  #$70      MASK OFF I/O DIGIT
00890 5830 17              TBA
00891 5831 8D 5300         JSR    ROMID     ROM EXIST?
00892 5834 27 1F           BEQ    ID25      YES, LOAD ASCII
00893 5836 33       PRTFMT PUL B            RELOAD SECOND BYTE
00894 5837 17              TBA              PUT IT IN ACCA ALSO
00895 5838 8D 0F           BSR    DIGIT     PUT ASCII DIGIT INTO BUFFER
00896 583A 08              INX
00897 583B C4 0F           AND B  #$F       MASK OFF LETTER
00898 583D 27 04           BEQ    AMODE     CALL-DIGIT-CALL?
00899 583F CB 40           ADD B  #$40      NO. CONVERT LETTER TO ASCII
00900 5841 20 61           BRA    STRB
00901 5843 86 68    AMODE  LDA A  #$68      LOAD ALPHA
00902 5845 97 CC           STA A  ALPHA     SET ALPHA FLAG
00903 5847 20 A8           BRA    STOOP     STORE ALPHA IN OPERAND
00904 5849 CE 0064 DIGIT   LDX    #$64      POINT TO OPERAND FIELD
00905 584C 44              LSR A
00906 584D 44              LSR A
00907 584E 44              LSR A
00908 584F 44              LSR A
00909 5850 8A 30           ORA A  #$30      CONVERT DIGIT TO ASCII
00910 5852 A7 00           STA A  X         STORE IT IN THE BUFFER
00911 5854 39              RTS
00912 5855 DF 18    ID25   STX    MSGPTR    STORE I/O ROM ADDR
00913 5857 DF 8E           STX    W+6
00914 5859 86 25           LDA A  #$25      ADD $25 TO IT
00915 585B 97 19           STA A  MSGPTR+1    ADD $25 TO GET ADDR OF ASCII MESSAG
00916 585D 7F 00CC MNEM    CLR    ALPHA     CLEAR ALPHA FLAG
00917 5860 33              PUL B            RELOAD SECOND BYTE
00918 5861 C4 0F           AND B  #$F       MASK OFF LETTER
00919 5863 26 02           BNE    LEQ0      LETTER=0?
00920 5865 97 CC           STA A  ALPHA     YES, SET ALPHA FLAG
00921 5867 8D 15    LEQ0   BSR    PRTMSG
00922 5869 DE 8E           LDX    W+6       NO, CHECK IF ID'S THE SAME
00923 586B A6 00           LDA A  X         LOAD ID#1
00924 586D 08              INX
00925 586E A1 00           CMP A  X         COMPARE WITH ID#2
00926 5870 27 34           BEQ    END1      NOT SAME, NO OPERAND
00927 5872 DE 8C           LDX    W+4       SET INDEX TO 2ND BYTE
00928 5874 A6 00           LDA A  X         RELOAD SECOND BYTE
00929 5876 84 70           AND A  #$70      MASK OFF I/O DIGIT
00930 5878 27 78           BEQ    END       DIGIT=0, YOU ARE DONE
00931 587A 8D CD    GENIO  BSR    DIGIT     YES, PUT ASCII DIGIT INTO BUFFER
00932 587C 20 74           BRA    END
00933 587E 17       PRTMSG TBA
00934 587F 58              ASL B
00935 5880 58              ASL B            MULTIPLY BY 4
00936 5881 1B              ABA              ADD TO MULTIPLY BY 5
00937 5882 9B 19           ADD A  MSGPTR+1    ADD RESULT TO BASE ADDR LOW
00938 5884 97 19           STA A  MSGPTR+1    RESTORE IT
00939 5886 86 5D           LDA A  #$5D      SET BFPTR TO MNEMONIC FIELD
00940 5888 97 17           STA A  BFPTR+1     RESTORE IT
00941 588A DE 18           LDX    MSGPTR
00942 588C 7E 578D         JMP    LDMSG
00943 588F D7 19    NTFMT  STA B  MSGPTR+1
00944 5891 16              TAB              PUT INSTR. INTO ACCB ALSO
00945 5892 81 C7           CMP A  #$C7      2-BYTE INSTR WITH 2-DIGIT OPERAND?
00946 5894 22 2F           BHI    REG       NO, CHECK FURTHER
00947 5896 5F              CLR B            OPERAND IN 2ND BYTE ONLY
00948 5897 8D 49           BSR    CNVRT     CONVERT OPERAND TO BCD
00949 5899 DE 16           LDX    BFPTR     SET INDEX BUFFER PTR
00950 589B 7D 002D         TST    T2        OPERAND>=100? (A-0)
00951 589E 27 08           BEQ    LE99      NO, OPERAND A NUMBER
00952 58A0 D6 19           LDA B  MSGPTR+1    YES, LOAD BOTTOM 2 BCD DIGITS
00953 58A2 C0 23           SUB B  #$23      CONVERT TO ASCII A THRU 0
00954 58A4 E7 00    STRB   STA B  X         STORE OPERAND
```

```
00955 5BA6 20 4A    END1    BRA     END         DONE
00956 5BA8 96 1C    LE99    LDA A   SVINS       LOAD INSTRUCTION
00957 5BAA 81 BF            CMP A   #$BF       LABEL IN OPERAND FIELD?
00958 5BAC 23 09            BLS     NTLBL       NO
00959 5BAE 81 C1            CMP A   #$C1       LABEL IN OPERAND FIELD?
00960 5BB0 22 05            BHI     NTLBL       NO
00961 5BB2 C6 4C            LDA B   #$4C       YES, LOAD L INTO ACCB
00962 5BB4 E7 00            STA B   X           PUT IT INTO BUFFER
00963 5BB6 08               INX                 INCR BUFFER PTR
00964 5BB7 16       NTLBL   TAB
00965 5BB8 A6 08            LDA A   #8          LOAD NO. OF DIGITS*4
00966 5BBA C1 BD            CMP B   #$BD        FIX, SCI, OR SCI3?
00967 5BBC 22 02            BHI     DIGITS      NO, DON'T SUPPRESS LEADING 0
00968 5BBE A6 04            LDA A   #4          YES, SUPPRESS LEADING 0 ON OPERAND
00969 5BC0 BD 4EA4  DIGITS  JSR     DGTS        PUT DIGITS INTO BUFFER
00970 5BC3 20 2D            BRA     END         DONE
00971 5BC5 81 0F    REG     CMP A   #$0F        2-BYTE INSTR WITH REG OPERAND?
00972 5BC7 22 0F            BHI     ADDR        NO, OPERAND IS ABSOLUTE ADDRESS
00973 5BC9 C4 01            AND B   #1          LSB 1ST BYTE IS PART OF OPERAND
00974 5BCB BD 15            BSR     CNVRT       CONVERT OPERAND TO BCD
00975 5BCD DE 16            LDX     BFPTR       SET INDEX TO BUFFER PTR
00976 5BCF C6 52            LDA B   #$52        PUT R INTO REG OF OPERAND
00977 5BD1 E7 00            STA B   X           STORE IT IN BUFFER
00978 5BD3 08               INX                 INCR BUFFER PTR
00979 5BD4 86 0C            LDA A   #12         LOAD NO. OF DIGITS*4
00980 5BD6 20 E8            BRA     DIGITS      PUT 3 DIGITS INTO BUFFER
00981 5BD8 C4 0F    ADDR    AND B   #$F         MASK OFF LOWER 4 BITS OF INSTR
00982 5BDA BD 06            BSR     CNVRT       CONVERT 11-BIT OPERAND TO BCD
00983 5BDC DE 16            LDX     BFPTR       SET INDEX TO BUFFER PTR
00984 5BDE 86 10            LDA A   #16         LOAD NO. OF DIGITS*4
00985 5BE0 20 DE            BRA     DIGITS      PUT 4 DIGITS INTO BUFFER
00986 5BE2 D7 18    CNVRT   STA B   MSGPTR      STORE UPPER PART OF OPERAND
00987 5BE4 78 0019          ASL     MSGPTR+1      SHIFT 2ND BYTE LEFT 1 BIT
00988 5BE7 74 0018          LSR     MSGPTR      SHIFT LSB OF 1ST BYTE INTO CARRY
00989 5BEA 76 0019          ROR     MSGPTR+1      ROTOATE IT INTO 2ND BYTE
00990 5BED DE 18            LDX     MSGPTR      SET INDEX TO OPERAND
00991 5BEF 7E 48BC          JMP     BINBCD
00992 5BF2 96 14    END     LDA A   GIOFLG
00993 5BF4 26 3A            BNE     NTLIST      GIO? YES, RETURN
00994 5BF6 BD 6034          JSR     PRTDRV+7      NO, PRINT LINE
00995 5BF9 DE 8C            LDX     W+4
00996 5BFB 08               INX                 INCREMENT IT
00997 5BFC 9C 0B            CPX     EOPM        END OF USER PROGRAM MEMORY?
00998 5BFE 27 0D            BEQ     ABORT1      YES, MEMORY OVERFLOW ABORT
00999 5C00 DF C8            STX     UPP         RESTORE USER PSTN PTR
01000 5C02 96 06            LDA A   ERROR       OUT OF PAPER?
01001 5C04 26 20            BNE     NOEND       YES, STOP LIST OR KEYLOG
01002 5C06 96 09            LDA A   RSFLG       NO, CHECK RSFLG FLAG
01003 5C08 2A 07            BPL     ABORT       CONTINUE LISTING?
01004 5C0A 7E 59F4          JMP     LSTOK       YES, DO ANOTHER LINE
01005 5C0D 86 07    ABORT1  LDA A   #7          MEMORY OVERFLOW ERROR
01006 5C0F 97 06            STA A   ERROR
01007 5C11 96 1C    ABORT   LDA A   SVINS       LOAD INSTRUCTION
01008 5C13 81 81            CMP A   #$81        END INSTRUCTION?
01009 5C15 26 0F            BNE     NOEND       NO, DON'T SPACE
01010 5C17 BD 57FA          JSR     LF
01011 5C1A BD 57FA          JSR     LF
01012 5C1D BD 57FA          JSR     LF
01013 5C20 BD 57FA          JSR     LF
01014 5C23 BD 57FA          JSR     LF
01015 5C26 4F       NOEND   CLR A
01016 5C27 97 09            STA A   RSFLG       NO, CLEAR RUN/STOP FLAG
01017 5C29 96 15            LDA A,  LSTFLG      LIST?
01018 5C2B 27 03            BEQ     NTLIST      NO, KEYLOG
01019 5C2D 7F 00CC          CLR     ALPHA       YES, CLEAR ALPHA FLAG
01020 5C30 39       NTLIST  RTS
01022               *
01023               *     AUDIT TRAIL
01024               *
01025 5C31 85 20    AUDIT   BIT A   #$20
01026 5C33 27 56            BEQ     AUD5
01027 5C35 96 0F            LDA A   DIGFLG
01028 5C37 C1 0B            CMP B   #$B
01029 5C39 23 50            BLS     AUD5        =DIGIT
01030 5C3B C1 11            CMP B   #$11
```

```
01031 5C3D 27 4C          BEQ    AUD5      =EEX.
01032 5C3F C1 BF          CMP B  #$BF
01033 5C41 27 48          BEQ    AUD5      =LABEL
01034 5C43 C1 12          CMP B  #$12
01035 5C45 26 03          BNE    AUD1      #CHS
01036 5C47 4D             TST A
01037 5C48 26 41          BNE    AUD5      STILL DIGIT ENTRY
01038 5C4A 8D 4F    AUD1  BSR    OUTXR
01039 5C4C DE C8          LDX    UPP
01040 5C4E DF 86          STX    AT2+6     SAVE UPP
01041 5C50 CE 00C6        LDX    #SOL7
01042 5C53 DF C8          STX    UPP       SET NEW INSTRUCTION POINTER
01043 5C55 4F             CLR A
01044 5C56 BD 59D9        JSR    GIONTR    GIONTR-INSTRUCTION ASCII
01045 5C59 DE 86          LDX    AT2+6
01046 5C5B DF C8          STX    UPP       RESTORE UPP
01047 5C5D D6 C6          LDA B  SOL7
01048 5C5F C1 B0          CMP B  #$B0
01049 5C61 26 0A          BNE    AUD2      #RUN
01050 5C63 CE 5255        LDX    #$5255
01051 5C66 DF 5D          STX    BUFF+5    STUFF 'RUN '
01052 5C68 CE 4E20        LDX    #$4E20
01053 5C6B DF 5F          STX    BUFF+7
01054 5C6D CE 0058  AUD2  LDX    #BUFF     ADDRESS
01055 5C70 A6 05          LDA A  5,X
01056 5C72 A7 00          STA A  0,X       SHIFT BUFFER LEFT
01057 5C74 08             INX
01058 5C75 8C 0063        CPX    #BUFF+11
01059 5C78 26 F6          BNE    AUD2+3
01060 5C7A 6F 00    AUD3  CLR    0,X
01061 5C7C 08             INX
01062 5C7D 8C 0068        CPX    #BUFF+16
01063 5C80 26 F8          BNE    AUD3
01064 5C82 BD 6034  AUD4  JSR    PRTDRV+7
01065 5C85 96 22          LDA A  T13
01066 5C87 97 0F          STA A  DIGFLG    RESTORE DIGIT ENTRY FLAG
01067 5C89 D6 C6          LDA B  SOL7      RESTORE COMMAND
01068 5C8B 39       AUD5  RTS
01070 5C8C 96 0F    AUDIT1 LDA A  DIGFLG
01071 5C8E 8D 0B          BSR    OUTXR
01072 5C90 BD 5D75        JSR    BLANK
01073 5C93 96 2E          LDA A  T1
01074 5C95 8B 40          ADD A  #$40      ASCII LETTER
01075 5C97 97 58          STA A  BUFF
01076 5C99 20 E7          BRA    AUD4
01077 5C9B 97 22    OUTXR STA A  T13       SAVE DIGIT ENTRY FLAG
01078 5C9D 27 EC          BEQ    AUD5      SKIP IT
01079 5C9F 7F 000F        CLR    DIGFLG
01080 5CA2 BD 5CBC        JSR    FRMT+$14    FORMAT XR
01081 5CA5 7E 6034        JMP    PRTDRV+7    PRINT IT
01085                     END
```

SYMBOL TABLE

ADATA	0000	ACTL	0001	BDATA	0002	BCTL	0003	INPUT	0004	IOIN	0005
ERROR	0006	TGL	0007	UFLG	0008	RSFLG	0009	EOM	000A	EOPM	000B
STKFLG	000D	RND	000E	DIGFLG	000F	W2	0010	W1	0011	SFLG	0012
DCNTR	0013	TP1	0014	TP1S	0015	TP2	0016	TP2S	0017	TP3	0018
TP3S	0019	TP4	001A	TP4S	001B	TP5	001C	TP5S	001D	TP6	001E
TP6S	001F	TP7	0020	TP7S	0021	T13	0022	T12	0023	T11	0024
T10	0025	T9	0026	T8	0027	T7	0028	T6	0029	T5	002A
T4	0028	T3	002C	T2	002D	T1	002E	ISTK	002F	ISTACK	0051
TA	0052	SPGM	0054	EXTRA	0056	BUFF	0058	REAL	0068	IMAG	0070
AT1	0078	AT2	0080	W	0088	XR	0090	YR	0098	ZR	00A0
TR	00A8	LSTX	00B0	BKWRT	00B8	BKKC	00BA	SOL7	00C6	UPP	00C8
UIP	00CA	ALPHA	00CC	IO1	00CD	IO2	00D0	IT7	00D3	FLAG	00D5
TPOS	00D6	FILE	00D7	AR	00D8	BR	00E0	CR	00E8	DR	00F0
PAREX	0080	NTBL	0000	DOTS	5EC0	PRTDRV	602D	FRMT	5C48	BLANK	5D75
LDMSG	578D	ROLLD	5582	ROLLU	57F1	PSD	550A	TXL	55E9	STKUP	55EF
MAD	7498	CMP	74AA	NOR	74D6	TXW	7424	TXXR	7438	EXXR	7452
ARSR	753B	OVUNF	7586	OVERF	75DD	XR0	740A	XRNINE	75C8	UNDRF	75F1
IMULT	76B9	QDG	7669	FPA	75FC	FPS	75F6	FPM	7735	FPD	7793
FPAEX	763D	FPMEX	7780	LSHIFT	7521	ZEROX	7439	XZERO0	7416	XZERO2	7417

```
RECIP   73E6  TXRX    73F3  CONST   6800  FPDBRC 6898  TAN     68A9  ATN     69C3
DSZERO  6A46  NTLN    6A58  EXPN    6AC9  SIN    6894  COS     689A  ASIN    68F2
ACOS    68F7  PH1     6C5D  PH2     6C8D  PH3    6D34  PH4     6DD0  LSFT8   6E47
SQRT    6E65  MADB    6F2C  CMPB    53E4  IOUPX  6F52  LOG10   6FA7  YUPX    6FE9
RTOP    7328  PTOR    7386  TABLE   5800  TABLE1 58ED  FTBL    59C0  TSNL    59C5
BWM     7978  SVINS   001C  SAVE    001D  GIOFLG 0014  LSTFLG  0015  MSGPTR  0018
BFPTR*  0016  BINBCD  488C  DGTS    4EA4  ROMID  5300  LF      57FA  STEP1   52F4
LSTNTR  59D4  GIONTR  59D9  KEYLOG  59DA  LIST   59EF  LSTOK   59F4  NTEND   5A08
KYLG    5A0A  SPERR   5A0F  OKAY    5A17  TLS    5A42  END2    5A4C  ENDASC  5A4F
NALPHA  5A5P  NTASC   5A5E  CMPCTR  5A62  NOCNT  5A66  MRTHN1  5A72  NOTYET  5A75
EQUAL   5A7F  FOUND   5A8C  TBL20   5A9C  TBL21  5AA7  TBL22   5AB0  TBL23   5AB8
SUB10   5ABE  TBL24   5AC6  TBL25   5ACA  LDABC  5ADA  STOOP   5AF1  DONE    5AF3
TBL30   5AF6  DASH    5B0D  NOLRL   5B1A  FORMAT 5B25  PRTFMT  5B36  AMODE   5B43
DIGIT   5B49  ID25    5B55  MNEM    5B5D  LEQ0   5B67  GENIO   5B7A  PRTMSG  5B7E
NTFMT   5B8F  STR8    5BA4  END1    5BA6  LE99   5BA8  NTLBL   5BB7  DIGITS  5BC0
REG     5BC5  ADDR    5BD8  CNVRT   5BE2  END    5BF2  ABORT1  5C0D  ABORT   5C11
NOEND   5C26  NTLIST  5C30  AUDIT   5C31  AUD1   5C4A  AUD2    5C6D  AUD3    5C7A
AUD4    5C82  AUD5    5C8B  AUDIT1  5C8C  OUTXR  5C9B
```

```
00215                       OPT    LIST,MEM
00218        002D    LNGTH  EQU    T2
00219        002C    DECPT  EQU    T3
00220        002B    XPNT   EQU    T4
00221        002A    CARRY  EQU    T5
00222        0029    NUMOVF EQU    T6
00223        0028    FLTFLG EQU    T7
00224        008B    EXP    EQU    W
00225        008A    EXP2   EQU    W+2
00226 5CA8                  ORG    FRMT
00227                *
00228                *
00229                * DAVE UHLRICH   OCTOBER 15,1974   FORMATTER   REV N
00230                *
00231                *
00232                * THIS ROUTINE CONVERTS THE EXPONENT, STORED
00233                * IN TWO'S COMPLEMENT FORM, TO ITS BCD
00234                * EQUIVALENT.  THE TWO'S COMPLEMENT OF THE
00235                * EXPONENT MUST BE IN ACCA BEFORE
00236                * ENTERING THIS ROUTINE.  THE ANSWER WILL
00237                * BE STORED IN XPNT WHEN THE CONVERSION
00238                * IS COMPLETED.
00239                *
00240                *
00241 5CA8 2A 01    EXPNT  BPL    EXPPL     IS SIGN OF EXPNT POSITIVE?
00242 5CAA 40               NEG A           NO, MAKE IT POSITIVE
00243 5CAB C6 FF    EXPPL  LDA B  #$FF      YES. LOAD CTR'S START VALUE
00244 5CAD 5C       XPINC  INC B            INCR TEN'S COLUMN
00245 5CAE 80 0A           SUB A  #10       SUBTRACT 10 FROM ABS(EXPNT)
00246 5CB0 2C FB           BGE    XPINC     IF>=0, EXPNT INCOMPLETE
00247 5CB2 8B 0A           ADD A  #10       IF<0, ADD 10 TO ONE'S COLUMN
00248 5CB4 58              ASL B            CTR VALUE IN ACCB
00249 5CB5 58              ASL B            EQUALS TEN'S COLUMN
00250 5CB6 58              ASL B            OF EXPNT, SO SHIFT
00251 5CB7 58              ASL B            IT LEFT 4 PLACES
00252 5CB8 1B              ABA              OBTAIN WHOLE EXPNT
00253 5CB9 97 2B           STA A  XPNT      STORE EXPNT IN XPNT
00254 5CBB 39              RTS
00256                *
00257                *
00258                * THIS ROUTINE FORMATS THE DATA STORED IN
00259                * THE X REGISTER.  THE FORMAT MODE INFORMA-
00260                * TION IS FOUND IN TGL AND THE DIGIT ENTRY
00261                * FLAG IS LOCATED IN DIGFLG.  THE
00262                * FOLLOWING ARE THE 3 FORMAT MODES:
00263                *      1) FXD N    BBBB-123.456BBBB
00264                *      2) FLT N    BBBBBB-1.234BB02
00265                *      3) PWR3 N   BBBB-123.456BB00
00266                * THE FORMAT MODE USED IN DIGIT ENTRY WILL BE
00267                * FIXED FORMAT.  FIXED FORMAT WILL OVERFLOW
00268                * INTO FLOATING POINT WHEN IT BECOMES
00269                * TOO BIG TO FIT IN THE DISPLAY.  IN THE
00270                * PWR3 MODE, THE ANSWER IS DISPLAYED WITH AN
00271                * EXPONENT WHICH IS THE POWER OF 3 JUST
00272                * BELOW THE ACTUAL EXPONENT.  THIS MODE WILL
00273                * ALSO OVERFLOW INTO ORDINARY FLOATING POINT
00274                * WHEN THE MANTISSA IS TOO LARGE FOR THE
```

```
00275                 * DISPLAY.  THE LARGEST ALLOWABLE ROUND IN
00276                 * PWR3 MODE WILL BE 7; ANY LARGER ROUND
00277                 * FACTOR WILL DEFAULT TO 7.
00278                 *
00279                 *
00280 5C8C 4F        FRMT1  CLR A
00281 5C8D 97 2C             STA A  DECPT    SET DEC PT PTR=0
00282 5C8F 97 2D             STA A  LNGTH    SET LNGTH PTR=0
00283 5CC1 97 29             STA A  NUMOVF   CLEAR NUMERIC OVERFLOW FLAG
00284 5CC3 97 28             STA A  FLTFLG   CLEAR FLT PT OVERFLOW FLAG
00285 5CC5 96 90             LDA A  XR       LOAD EXPNT
00286 5CC7 8D DF             BSR    EXPNT    CONVERT IT TO BCD
00287 5CC9 D6 90             LDA B  XR       LOAD EXPNT INTO ACCB
00288 5CCB 96 0F             LDA A  DIGFLG   LOAD DIGIT ENTRY FLAG
00289 5CCD 81 20             CMP A  #$20     LEADING ZEROES?
00290 5CCF 27 7D             BEQ    NTROS    YES
00291 5CD1 84 DF             AND A  #$DF     DIGIT ENTRY?
00292 5CD3 26 61             BNE    DIGNTR   YES
00293 5CD5 96 07             LDA A  TGL      NO, LOAD TOGGLE SWITCHES
00294 5CD7 84 1C             AND A  #@34     MASK FORMAT MODE BITS
00295 5CD9 81 08             CMP A  #@10
00296 5CDB 2D 28             BLT    FLT      IF<@10, FLOAT PT
00297 5CDD 2E 4D             BGT    FXD      IF>@10, FIXED PT
00298 5CDF 96 0E             LDA A  RND      LOAD ROUND FACTOR
00299 5CE1 81 07             CMP A  #7       ROUND=7?
00300 5CE3 2F 04             BLE    RND7     <=7, LEAVE RND ALONE
00301 5CE5 86 07             LDA A  #7       SET RND TO 7
00302 5CE7 97 0E             STA A  RND      STORE NEW RND=7
00303 5CE9 17         RND7   TBA             PUT BINARY EXPNT INTO ACCA
00304 5CEA 2A 01             BPL    EXPL     POSITIVE?
00305 5CEC 40                NEG A           NO, MAKE IT POSITIVE
00306 5CED 5F         EXPL   CLR B           CLEAR PWR3 CTR
00307 5CEE CB 03      PWR3   ADD B  #3       BUILD PWR3 EXPNT BY 3'S
00308 5CF0 11                CBA             COMPARE REAL AND PWR3 EXPNTS
00309 5CF1 27 12             BEQ    FLT      ALREADY PWR3, USE FLT PT
00310 5CF3 2E F9             BGT    PWR3     REAL>PWR3, CONT BUILDING
00311 5CF5 7D 0090           TST    XR       TEST SIGN OF EXPNT
00312 5CF8 2B 02             BMI    NEG1     MINUS, PWR3 EXPNT BUILT
00313 5CFA C0 03             SUB B  #3       PLUS, SUB TO GET PWR3 EXPNT
00314 5CFC 10         NEG1   SBA             FIND DIFF REAL & PWR3
00315 5CFD 2A 01             BPL    NEG2     POSITIVE?
00316 5CFF 40                NEG A           NO, MAKE IT POSITIVE
00317 5D00 97 2C      NEG2   STA A  DECPT    STORE DIFF IN DEC PT PTR
00318 5D02 17                TBA             TRANSFER PWR3 EXPNT TO ACCA
00319 5D03 8D A3             BSR    EXPNT    CONVERT IT TO BCD
00320 5D05 96 0E      FLT    LDA A  RND      RELOAD ROUND OFF FACTOR
00321 5D07 9B 2C             ADD A  DECPT    ADD DEC PT PTR TO RND
00322 5D09 4C         FLOAT  INC A           RND+DECPT+1=POS CTR
00323 5D0A 7C 0028    FLT1   INC    FLTFLG   SET FLOATING POINT OVERFLOW FLAG
00324 5D0D 20 71             BRA    LDMNT    LOAD THE MANTISSA
00325 5D0F D6 0F      FLT2   LDA B  DIGFLG
00326 5D11 C5 DF             BIT B  #$DF
00327 5D13 27 F4             BEQ    FLOAT    NOT DIGIT ENTRY, FLT PT
00328 5D15 86 0A             LDA A  #10      DIGIT ENTRY, CHECK OVERFLOW
00329 5D17 C4 0F             AND B  #$F      MASK OFF DIGIT CTR
00330 5D19 C1 0A             CMP B  #10      MORE 10 DIGITS HIT?
00331 5D1B 27 03             BEQ    CHKOVF   YES, CHECK OVERFLOW
00332 5D1D 17                TBA             NO, POS CTR=DIGIT CTR
00333 5D1E 20 EA             BRA    FLT1
00334 5D20 D6 90      CHKOVF LDA B  XR       LOAD EXPNT
00335 5D22 D0 88             SUB B  EXP      SUBTRACT EEX EXPNT
00336 5D24 2D E4             BLT    FLT1     EXPNT<0, FLT PT OVRFL
00337 5D26 C1 0A             CMP B  #10      MANT, EXPNT>=10?
00338 5D28 2C E0             BGE    FLT1     YES, FLT PT OVERFLOW
00339 5D2A 20 1A             BRA    FIXED    NO, FIXED PT
00340 5D2C 17         FXD    TBA             SHIFT EXPNT TO ACCA
00341 5D2D 9B 0E             ADD A  RND      RND+EXPNT
00342 5D2F 4C                INC A           RND+EXPNT+1=POS CTR
00343 5D30 81 0A             CMP A  #10      FIXED POINT OVERFLOW?
00344 5D32 2E D1             BGT    FLT      YES, FLT PT
00345 5D34 20 10             BRA    FIXED    NO, FXD PT
00346 5D36 D0 88      DIGNTR SUB B  EXP      XR-EXP=MNT EXPNT
00347 5D38 81 80             CMP A  #$80     DEC PT & 0'S ONLY?
00348 5D3A 27 12             BEQ    NTROS    YES, ENTER ZEROES
00349 5D3C 84 0F             AND A  #$F      GET DIGIT CTR
00350 5D3E 90 8A             SUB A  EXP2     ADD NO. LEADING 0'S TO DIG CTR
```

```
00351 5D40 81 0A          CMP A  #10       NO, MORE THAN 10 DIGITS HIT?
00352 5D42 2C CB          BGE    FLT2      YES, CHECK OVERFLOW
00353 5D44 9B 8A          ADD A  EXP2      NO, GET DIGIT CTR AGAIN
00354 5D46 D7 2C   FIXED  STA B  DECPT     STORE DEC PT PTR
00355 5D48 2C 36          BGE    LDMNT     EXPNT<0? NO, LOAD MANT
00356 5D4A D7 2D          STA B  LNGTH     YES, SET LNGTH PTR=EXPNT
00357 5D4C 20 32          BRA    LDMNT     LOAD THE MANTISSA
00358 5D4E 96 8A   NTROS  LDA A  EXP2      LOAD NO. 0'S AFTER DEC PT
00359 5D50 BD 5CA8        JSR    EXPNT     SET UP EXPNT DUE TO THEM
00360 5D53 96 8A          LDA A  EXP2      GET NUMBER
00361 5D55 40             NEG A            OF ZEROES TO DISPLAY
00362 5D56 4C             INC A
00363 5D57 81 0A          CMP A  #10       10 OF THEM?
00364 5D59 2F 25          BLE    LDMNT     NO, LOAD MANT WITH ZEROES
00365 5D5B 86 0A          LDA A  #10       YES, FILL DISPLAY WITH ZEROES
00366 5D5D 20 AB          BRA    FLT1      YES, FLT POINT OVRFL
00367 5D5F DF 16   LDDGT  STX    TP2       STORE BUFF PTR
00368 5D61 DE 18          LDX    TP3       LOAD X-REG PTR
00369 5D63 A6 00          LDA A  X         LOAD DIGIT INTO ACCA
00370 5D65 C5 01          BIT B  #1        POS CTR EVEN?
00371 5D67 26 05          BNE    STRDGT    NO, DIGIT IN LOWER HALF
00372 5D69 44             LSR A            YES, DIGIT IN
00373 5D6A 44             LSR A            UPPER HALF
00374 5D6B 44             LSR A            BYTE; SHIFT
00375 5D6C 44             LSR A            IT DOWN
00376 5D6D 09             DEX              DECR X-REG PTR
00377 5D6E 84 0F   STRDGT AND A  #$F       MASK LOWER HALF BYTE
00378 5D70 DF 18          STX    TP3       CHANGE INDEX REG FROM
00379 5D72 DE 16          LDX    TP2       X-REG PTR TO BUFF PTR
00380 5D74 39             RTS              DIGIT LOADED; RETURN
00381 5D75 CE 0010 BLANK1 LDX    #$10      SET BUFFER PTR
00382 5D78 C6 20          LDA B  #@40      LOAD BLANK CODE
00383 5D7A E7 57   MORE   STA B  $57,X     STORE BLANK IN BUFF
00384 5D7C 09             DEX              DECR BUFFER PTR
00385 5D7D 26 FB          BNE    MORE      LOAD ANOTHER BLANK
00386 5D7F 39             RTS              RETURN
00387 5D80 BD F3   LDMNT  BSR    BLANK1    LOAD BLANKS INTO DISPLAY BUFFER
00388 5D82 CE 0064        LDX    #@144     SET BUFF PTR AT END OF MANT AREA
00389 5D85 16             TAB              POS PTR IN ACCB ALSO
00390 5D86 44             LSR A            GET POS CTR/2
00391 5D87 8B 92          ADD A  #@222     BUFF PTR=222+POS CTR/2
00392 5D89 97 19          STA A  TP3S      2ND BYTE INDEX REG
00393 5D8B 4F             CLR A
00394 5D8C 97 18          STA A  TP3       1ST BYTE INDEX REG
00395 5D8E 97 2A          STA A  CARRY     CLEAR CARRY
00396 5D90 5D             TST B
00397 5D91 2B 48          BMI    ENDMNT
00398 5D93 BD CA          BSR    LDDGT     LOAD ROUNDING DIGIT
00399 5D95 81 05          CMP A  #5        COMPARE IT TO 5
00400 5D97 2D 03          BLT    CARRY1    IS IT <5?
00401 5D99 7C 002A        INC    CARRY     NO,SET CARRY
00402 5D9C 5D      CARRY1 TST B            TEST POS CTR AGAIN
00403 5D9D 2F 3C          BLE    ENDMNT    IS IT NEGATIVE?
00404 5D9F 09             DEX              NO, DECR BUFF PTR
00405 5DA0 5A             DEC B            DECR POS CTR
00406 5DA1 BD BC   XP99   BSR    LDDGT     LOAD DIGIT
00407 5DA3 D1 2C   LOAD0  CMP B  DECPT     CTR=DEC PT PTR?
00408 5DA5 26 21          BNE    NODPT     NO, NO DEC PT
00409 5DA7 0F             SEI              YES, DISABLE KEY INTERRUPT
00410 5DA8 36             PSH A            SAVE DIGIT ON STACK
00411 5DA9 96 0F          LDA A  DIGFLG
00412 5DAB 2B 14          BMI    DCPT      DEC PT HIT?
00413 5DAD 96 29          LDA A  NUMOVF    NUMERIX OVERFLOW?
00414 5DAF 26 10          BNE    DCPT      IF OVRFL, PUT IN DEC PT
00415 5DB1 96 0F          LDA A  DIGFLG
00416 5DB3 85 DF          BIT A  #$DF      DIGIT ENTRY?
00417 5DB5 27 06          BEQ    CHKRND    NO, CHECK ROUND MODE
00418 5DB7 96 28          LDA A  FLTFLG    YES, FLT PT OVERFLOW
00419 5DB9 27 0B          BEQ    NODPT1    NO, DEC PT SUPPRESSED
00420 5DBB 20 04          BRA    DCPT      YES, DEC PT
00421 5DBD 96 0E   CHKRND LDA A  RND
00422 5DBF 27 05          BEQ    NODPT1    RND=0, NO DEC PT
00423 5DC1 86 2E   DCPT   LDA A  #@56      LOAD DEC PT CODE
00424 5DC3 A7 00          STA A  X         STORE IT IN BUFF
00425 5DC5 09             DEX              DECR BUFF PTR
```

```
00426 5DC6 32       NODPT1 PUL A          RELOAD DIGIT INTO ACCA
00427 5DC7 0E              CLI            ENABLE KEY INTERRUPT
00428 5DC8 9B 2A    NODPT  ADD A  CARRY    ADD CARRY TO DIGIT
00429 5DCA 7F 002A         CLR    CARRY    CLEAR THE CARRY
00430 5DCD 81 0A           CMP A  #10      DIGIT+CARRY<10?
00431 5DCF 2D 04           BLT    CARRY2   YES, NO CARRY
00432 5DD1 7C 002A         INC    CARRY    NO, SET CARRY
00433 5DD4 4F              CLR A           PUT BCD 0 INTO ACCA
00434 5DD5 8A 30    CARRY2 ORA A  #@60      CONVERT BCD TO ASCII
00435 5DD7 A7 00           STA A  X        STORE DIGIT IN BUFF
00436 5DD9 20 C1           BRA    CARRY1
00437 5DDB D1 2D    ENDMNT CMP B  LNGTH     FINISHED MANTISSA?
00438 5DDD 2F 05           BLE    CHKCR,   YES, STILL A CARRY?
00439 5DDF 4F       LOAD1  CLR A           LOAD ACCA WITH BCD 0
00440 5DE0 5A              DEC B           DECR POS CTR
00441 5DE1 09              DEX             DECR BUFF PTR
00442 5DE2 20 8F           BRA    LOAD0    LOAD ZERO INTO BUFF
00443 5DE4 96 2A    CHKCR  LDA A  CARRY    LOAD CARRY
00444 5DE6 85 01           BIT A  #1       IS THERE A CARRY?
00445 5DE8 27 56           BEQ    SGNMT    NO, LOAD SIGN OF MANT
00446 5DEA 7F 002A         CLR    CARRY    CLEAR THE CARRY
00447 5DED 96 28           LDA A  XPNT     LOAD BCD EXPNT
00448 5DEF 81 99           CMP A  #$99     IS IT = 99?
00449 5DF1 26 0D           BNE    NOT99
00450 5DF3 86 96           LDA A  #@226    YES, SET XR PTR=@226
00451 5DF5 97 19           STA A  TP3S     STORE IT
00452 5DF7 C6 09           LDA B  #9       SET POS PTR=9
00453 5DF9 D7 29           STA B  NUMOVF   SET NUMERIC OVERFLOW BIT
00454 5DFB CE 0063         LDX    #@143    RESET BUFF PTR
00455 5DFE 20 A1           BRA    XP99     RELOAD MANTISSA WITHOUT ROUNDING
00456 5E00 96 91    NOT99  LDA A  XR+1     PWR3 OVRFL FLAG SET?
00457 5E02 85 01           BIT A  #1
00458 5E04 26 30           BNE    FLTOVF   YES, FLT PT OVERFLOW
00459 5E06 96 07           LDA A  TGL      LOAD TOGGLE SWITCH REG
00460 5E08 85 08           BIT A  #@10     PWR3?
00461 5E0A 26 22           BNE    POWER3   YES, CHECK PWR3 OVERFLOW
00462 5E0C 96 28           LDA A  FLTFLG   FLT PT OVERFLOW?
00463 5E0E 26 26           BNE    FLTOVF   YES, FLT PT OVERFLOW
00464 5E10 86 01    RGBUF  LDA A  #1       LOAD 1 DUE TO CARRY
00465 5E12 09              DEX             DECR BUFF PTR
00466 5E13 8C 0058         CPX    #@130    REACHED BEG. MANT AREA IN BUFF?
00467 5E16 26 BD           BNE    CARRY2   NO, LOAD CARRY INTO BUFF
00468 5E18 96 90    OVRFL  LDA A  XR       LOAD EXPNT
00469 5E1A BD 5CA8         JSR    EXPNT    CONVERT IT TO BCD
00470 5E1D D6 07           LDA B  TGL
00471 5E1F C5 08           BIT B  #@10     PWR3 MODE?
00472 5E21 27 08           BEQ    FLT3     NO, FIXED POINT OVERFLOW
00473 5E23 7C 0091         INC    XR+1     YES, SET PWR3 OVRFL FLAG
00474 5E26 4F              CLR A
00475 5E27 97 28           STA A  FLTFLG   CLEAR FLT PT OVERFLOW FLAG
00476 5E29 97 2C           STA A  DECPT    SET DECPT=0
00477 5E2B 7E 5D05 FLT3    JMP    FLT
00478 5E2E 96 2C    POWER3 LDA A  DECPT    LOAD POWER3 MANT EXPNT
00479 5E30 81 02           CMP A  #2       MANTISSA OVERFLOW?
00480 5E32 26 DC           BNE    RGBUF    NONE, LOAD CARRY
00481 5E34 20 E2           BRA    OVRFL    YES, REDO IN FLT PT
00482 5E36 86 31    FLTOVF LDA A  #@61      STORE 1 IN BUFF LOC
00483 5E38 A7 00           STA A  X        NEXT TO DEC PT
00484 5E3A 96 90           LDA A  XR       LOAD EXPNT
00485 5E3C 4C              INC A           INCREMENT IT BY 1
00486 5E3D BD 5CA8         JSR    EXPNT    CONVERT IT TO BCD
00487 5E40 96 91    SGNMT  LDA A  XR+1     LOAD MANT SIGN
00488 5E42 84 80           AND A  #$80     CLEAR PWR3 OVFRL FLAG
00489 5E44 97 91           STA A  XR+1     RESTORE MANT SIGN
00490 5E46 2A 05           BPL    SGNPOS   POSITIVE?
00491 5E48 C6 2D           LDA B  #@55     NO, LOAD MINUS SIGN CODE
00492 5E4A 09              DEX             DECR BUFF PTR
00493 5E4B E7 00           STA B  X        STORE MINUS OR BLANK
00494 5E4D CE 0065 SGNPOS  LDX    #@145    SET BUFF PTR AT REG EXPNT AREA
00495 5E50 96 0F    CHKEEX LDA A  DIGFLG
00496 5E52 84 40           AND A  #$40
00497 5E54 9A 28           ORA A  FLTFLG   FLT PT OR EEX?
00498 5E56 27 3D           BEQ    NOEXP    NEITHER, NO EXPMT
00499 5E58 85 01           BIT A  #1       FLT PT?
00500 5E5A 26 12           BNE    SGNXR    YES, EXPNT IN XR
00501 5E5C 96 88           LDA A  EXP      NO, SET EEX EXPNT UP
```

```
00502 5E5E BD 5CA8          JSR     EXPNT
00503 5E61 4D               TST A                   EEX EXPNT ZERO?
00504 5E62 26 06            BNE     EXPN0           .NE. 0, CHECK SIGN
00505 5E64 96 0F            LDA A   DIGFLG
00506 5E66 85 10            BIT A   #$10            CHGSGN FLAG SET?
00507 5E68 26 14            BNE     NEGSGN          YES, SIGN EXPNT NEG
00508 5E6A 96 88    EXPN0   LDA A   EXP             LOAD NON-ZERO EEX EXPNT
00509 5E6C 20 0E            BRA     CHKSGN          CHECK ITS SIGN
00510 5E6E 96 0F    SGNXR   LDA A   DIGFLG          RELOAD DIGIT ENTRY FLAG
00511 5E70 84 DF            AND A   #$DF            CLEAR LEADING ZEROES BIT
00512 5E72 81 80            CMP A   #$80            DEC PT & 0'S ONLY?
00513 5E74 27 08            BEQ     NEGSGN          YES, NEGATIVE EXPNT
00514 5E76 96 28            LDA A   XPNT            LOAD BCD EXPNT
00515 5E78 27 08            BEQ     SGNXP           IF ZERO, SIGN EXPNT PLUS
00516 5E7A 96 90            LDA A   XP              LOAD EXPNT FROM XR
00517 5E7C 2A 04    CHKSGN  BPL     SGNXP           SIGN EXPNT PLUS?
00518 5E7E C6 2D    NEGSGN  LDA B   #@55            NO, LOAD MINUS SIGN CODE
00519 5E80 E7 00            STA B   X               STORE EXPNT SIGN
00520 5E82 08       SGNXP   INX                     INCR BUFF PTR
00521 5E83 96 28            LDA A   XPNT            LOAD BCD EXPNT
00522 5E85 16               TAB                     PUT IT IN ACCB ALSO
00523 5E86 44               LSR A                   TEN'S COLUMN OF
00524 5E87 44               LSR A                   EXPNT IN UPPER
00525 5E88 44               LSR A                   HALF BYTE; SHIFT
00526 5E89 44               LSR A                   IT DOWN
00527 5E8A 8A 30            ORA A   #@60            CONVERT BCD TO ASCII
00528 5E8C A7 00            STA A   X               STORE TEN'S COLUMN
00529 5E8E 08               INX                     INCR BUFF PTR
00530 5E8F C4 0F            AND B   #$F             GET ONE'S COLUMN
00531 5E91 CA 30            ORA B   #@60            CONVERT BCD TO ASCII
00532 5E93 E7 00            STA B   X               STORE ONE'S COLUMN OF EXPNT
00533 5E95 86 20    NOEXP   LDA A   #$20            PUT BLANK CODE IN ACCA
00534 5E97 C6 2E            LDA B   #$2E            PUT DEC PT CODE IN ACCB
00535 5E99 CE 0057          LDX     #BUFF-1          SET INDEX TO BEG OF BUFF-1
00536 5E9C 08       BEGNO   INX
00537 5E9D A1 00            CMP A   X               CHARACTER A BLANK?
00538 5E9F 27 FB            BEQ     BEGNO           YES, NOT BEG OF NO. YET
00539 5EA1 08       DPBLK   INX
00540 5EA2 A1 00            CMP A   X               BLANK?
00541 5EA4 27 04            BEQ     LDCOM           YES, START TO LOAD COMMAS
00542 5EA6 E1 00            CMP B   X               DEC PT?
00543 5EA8 26 F7            BNE     DPBLK           NO, CONTINUE LOOKING
00544 5EAA 09       LDCOM   DEX
00545 5EAB 09       COMMA   DEX
00546 5EAC 09               DEX
00547 5EAD 09               DEX                     FIND COMMA POSITION
00548 5EAE A6 00            LDA A   X               LOAD CHARACTER
00549 5EB0 81 2F            CMP A   #$2F            REACHED END OF NUMBER?
00550 5EB2 23 06            BLS     COMEND          YES, DONE
00551 5EB4 8A 80            ORA A   #$80            NO, PUT IN COMMA
00552 5EB6 A7 00            STA A   X               RESTORE CHAR WITH COMMA
00553 5EB8 20 F1            BRA     COMMA
00554 5EBA 39       COMEND  RTS
00557                       *
00558                       *
00559                       * THIS ROUTINE CONVERTS THE RECTANGULAR
00560                       * COORDINATES IN THE X AND Y REGISTERS
00561                       * TO POLAR FORM.  THE X-COORDINATE MUST BE
00562                       * IN THE X REGISTER AND THE Y-COORDINATE,
00563                       * IN THE Y REGISTER.  THE ANSWER IS
00564                       * RETURNED WITH THE MAGNITUDE IN THE
00565                       * X REGISTER AND THE ANGLE, IN THE Y
00566                       * REGISTER.
00567                       *
00568                       *
00569                       * DAVE UHLRICH   1/3/75   R TO P   REV C
00570                       *
00571                       *
00572       55AC    XEY     EQU     $55AC
00573       57E3    LDPI    EQU     $57E3
00574 7338                  ORG     $7338
00575 7338 BD 55E9 TOPOL    JSR     TXL             SAVE X IN LAST X
00576 733B 96 92            LDA A   XP+2
00577 733D 9A 9A            ORA A   YR+2            BOTH X AND Y = 0?
00578 733F 27 48            BEQ     YONLY           YES, YOU ARE DONE
00579 7341 CE 0098          LDX     #YP             POINT TO 2ND DIVIDE ARG
```

```
00580 7344 BD 7793          JSR    FPD       DIVIDE Y BY X
00581 7347 BD 69C3 NTDIV0   JSR    ATN       FIND ATAN Y/X
00582 734A 96 B1            LDA A  LSTX+1    IS X NEGATIVE?
00583 734C 2A 29            BPL    STOANS    NO, ANGLE IS CORRECT
00584 734E BD 55AC          JSR    XEY       PUT ANGLE INTO Y
00585 7351 BD 740A          JSR    XR0       ZERO OUT X
00586 7354 86 02            LDA A  #2
00587 7356 97 90            STA A  XR        SET XR EXPNT=2
00588 7358 96 99            LDA A  YR+1      SIGN OF ANGLE NEGATIVE?
00589 735A 88 80            EOR A  #$80      YES, CHANGE SIGN
00590 735C 97 91            STA A  XR+1      PUT SIGN ON 180 DEG OFFSET
00591 735E 86 20            LDA A  #$20      LOAD 200 GRADS
00592 7360 D6 07            LDA B  TGL
00593 7362 57               ASR B
00594 7363 25 05            BCS    GRAD      GRADS?
00595 7365 57               ASR B
00596 7366 25 06            BCS    RAD       RADS?
00597 7368 86 18            LDA A  #$18      LOAD 180 DEGREES
00598 736A 97 92    GRAD    STA A  XR+2      STORE IN X
00599 736C 20 03            BRA    ADDOFF
00600 736E BD 57E3 RAD      JSR    LDPI      LOAD PI INTO X
00601 7371 CE 0098 ADDOFF   LDX    #YR
00602 7374 BD 75FC          JSR    FPA       ADD OFFSET TO ANGLE
00603 7377 96 06    STOANS  LDA A  ERROR
00604 7379 26 11            BNE    YONLY     ANGLE=90 DEGREES?
00605 737B CE 0098          LDX    #YR
00606 737E BD 38            BSR    TXRX1     NO, PUT ANGLE INTO Y ALSO
00607 7380 BD 689A          JSR    COS
00608 7383 BD 73E6          JSR    RECIP     1/COS ANGLE
00609 7386 CE 0080          LDX    #LSTX
00610 7389 7E 7735          JMP    FPM       X/COS ANGLE
00611 738C 4F       YONLY   CLR A
00612 738D 97 06            STA A  ERROR
00613 738F 97 99            STA A  YR+1      MAKE MAGNITUDE POSITIVE
00614 7391 7E 55AC          JMP    XEY
00615                *
00616                *
00617                * THIS ROUTINE PERFORMS THE COMPLEMENT
00618                * OPERATION TO THE RECTANGULAR-TO-
00619                * POLAR ROUTINE, DESCRIBED ABOVE.
00620                *
00621                *
00622                * DAVE UHLRICH   6/10/74   P TO R   REV B
00623                *
00624                *
00625 7394 BD 55E9 TORCT  JSR    TXL       SAVE X IN LAST X
00626 7397 BD 25          BSR    YTOX      TRANSFER ANGLE TO X
00627 7399 BD 6B9A        JSR    COS       TAKE COS OF ANGLE
00628 739C CE 0080        LDX    #LSTX     POINT TO 2ND MULT ARG
00629 739F BD 7735        JSR    FPM       MAGNITUDE*SIN OF ANGLE
00630 73A2 CE 0078        LDX    #AT1      POINT TO AT1
00631 73A5 BD 14          BSR    TXRX1     SAVE X COORDINATE IN AT1
00632 73A7 BD 15          BSR    YTOX      TRANSFER ANGLE TO X AGAIN
00633 73A9 BD 6B94        JSR    SIN       TAKE SIN OF ANGLE
00634 73AC CE 0080        LDX    #LSTX     POINT TO 2ND MULT ARG
00635 73AF BD 7735        JSR    FPM       MAGNITUDE*COS OF ANGLE
00636 73B2 BD 55AC        JSR    XEY       PUT Y COORDINATE INTO Y
00637 73B5 CE 0078        LDX    #AT1      POINT TO AT1
00638 73B8 7E 7438 TXXR1  JMP    TXXR
00639 73BB 7E 73F3 TXRX1  JMP    TXRX
00640 73BE CE 0098 YTOX   LDX    #YR
00641 73C1 20 F5          BRA    TXXR1
00642 7C52                ORG    $7C52
00643 7C52 7338           FDB    TOPOL
00644 7C54 7394           FDB    TORCT
00647                     END
```

SYMBOL TABLE

ADATA	0000	ACTL	0001	BDATA	0002	BCTL	0003	INPUT	0004	IOIN	0005
ERROR	0006	TGL	0007	UFLG	0008	RSFLG	0009	EOM	000A	EOPM	000B
STKFLG	000D	RND	000E	DIGFLG	000F	W2	0010	W1	0011	SFLG	0012
DCNTR	0013	TP1	0014	TP1S	0015	TP2	0016	TP2S	0017	TP3	0018
TP3S	0019	TP4	001A	TP4S	001B	TP5	001C	TP5S	001D	TP6	001E

TP6S	001F	TP7	0020	TP7S	0021	T13	0022	T12	0023	T11	0024
T10	0025	T9	0026	T8	0027	T7	0028	T6	0029	T5	002A
T4	002B	T3	002C	T2	002D	T1	002E	ISTK	002F	ISTACK	0051
TA	0052	SPGM	0054	EXTPA	0056	BUFF	0058	REAL	0068	IMAG	0070
AT1	0078	AT2	0080	W	0088	XR	0090	YR	0098	ZR	00A0
TR	00AB	LSTX	00B0	BKWRT	00B8	BKKC	00BA	SOL7	00C6	UPP	00C8
UIP	00CA	ALPHA	00CC	IO1	00CD	IO2	00D0	IT7	00D3	FLAG	00D5
TPOS	00D6	FILE	00D7	AR	00D8	BR	00E0	CR	00E8	DR	00F0
ER	00F8	SDB3	00BA	MT	7E00	TERMN7	003D	IMED	0040	PARCD	00C0
PAREX	0080	NTBL	0000	DOTS	5EC0	PRTDRV	602D	FRMT	5CAB	BLANK	5D75
LDMSG	57BD	ROLLD	55B2	ROLLU	57F1	PSD	55DA	TXL	55E9	STKUP	55EF
MAD	749B	CMP	74AA	NOR	74D6	TXW	7424	TXXR	7438	EXXR	7452
ARSR	753B	OVUNF	75B6	OVERF	75DD	XR0	740A	XRNINE	75C8	UNDRF	75F1
IMULT	7689	ODG	7649	FPA	75FC	FPS	75F6	FPM	7735	FPD	7793
FPAEX	763D	FPMEX	7780	LSHIFT	7521	ZEROX	7489	XZERO0	7416	XZERO2	7417
RECIP	73E6	TXPX	73F3	CONST	6800	FPDBRC	6898	TAN	68A9	ATN	69C3
DSZERO	6A46	NTLN	6A58	EXPN	6AC9	SIN	6894	COS	689A	ASIN	6BF2
ACOS	6BF7	PH1	6C5D	PH2	6C8D	PH3	6D34	PH4	6DD0	LSFT8	6E47
SQRT	6E65	MADR	6F2C	CMPR	53E4	IOUPX	6F52	LOG10	6FA7	YUPX	6FE9
RTOP	7329	PTOR	7386	LNGTH	002D	DECPT	002C	XPNT	0029	CARRY	002A
NUMOVF	0029	FLTFLG	0028	EXP	0088	EXP2	008A	EXPNT	5CA8	EXPPL	5CAB
XPINC	5CAD	FRMT1	5CBC	RND7	5CE9	EXPL	5CED	PWR3	5CEE	NEG1	5CFC
NEG2	5D00	FLT	5D05	FLOAT	5D09	FLT1	5D0A	FLT2	5D0F	CHKOVF	5D20
FXD	502C	DIGNTR	5D36	FIXED	5D46	NTR0S	5D4E	LDDGT	5D5F	STRDGT	5D6E
BLANK1	5D75	MORE	5D7A	LDMNT	5D80	CARRY1	5D9C	XP99	5DA1	LOAD0	5DA3
CHKRND	5DB0	DCPT	5DC1	NODPT1	5DC6	NODPT	5DC8	CARRY2	5DD5	ENDMNT	5DD8
LOAD1	5DDF	CHKCR	5DE4	NOT99	5E00	BGBUF	5E10	OVRFL	5E18	FLT3	5E2B
POWER3	5E2F	FLTOVF	5E36	SGNMT	5E40	SGNPOS	5E4D	CHKEEX	5E50	EXPN0	5E6A
SGNXR	5E6E	CHKSGN	5E7C	NEGSGN	5E7E	SGNXP	5E82	NOEXP	5E95	BEGNO	5E9C
DPBLK	5EA1	LDCOM	5EAA	COMMA	5EAB	COMEND	5EBA	XEY	55AC	LDPI	57E3
TOPOL	7338	NTDIV0	7347	GRAD	736A	RAD	736E	ADDOFF	7371	STOANS	7377
YONLY	738C	TORCT	7394	TXXR1	7388	TXRX1	7388	YTOX	738E		

```
00215                           OPT    LIST,MEM
00218                  *
00219                  *
00220                  * DOT MATRICES FOR PRINTER -- STORED:
00221                  *      75 65 55 45 35 25 15
00222                  *      74 64 54 44 34 24 14
00223                  *      73 63 53 43 33 23 13
00224                  *      72 62 52 42 32 22 12
00225                  *      71 61 51 41 31 21 11
00226                  *
00227                  *
00228 5EC0                      ORG    DOTS
00229                  * SPACE
00230 5EC0 00                   FCB    $0
00231 5EC1 00                   FCB    $0
00232 5EC2 00                   FCB    $0
00233 5EC3 00                   FCB    $0
00234 5EC4 00                   FCB    $0
00235                  * EXCHANGE SIGN
00236 5EC5 14                   FCB    $14
00237 5EC6 16                   FCB    $16
00238 5EC7 55                   FCB    $55
00239 5EC8 34                   FCB    $34
00240 5EC9 14                   FCB    $14
00241                  * LOWER CASE E
00242 5ECA 2C                   FCB    $2C
00243 5ECB 4A                   FCB    $4A
00244 5ECC 4A                   FCB    $4A
00245 5ECD 4A                   FCB    $4A
00246 5ECE 3C                   FCB    $3C
00247                  * POUND SIGN
00248 5ECF 14                   FCB    $14
00249 5ED0 7F                   FCB    $7F
00250 5ED1 14                   FCB    $14
00251 5ED2 7F                   FCB    $7F
00252 5ED3 14                   FCB    $14
00253                  * DOLLAR SIGN
00254 5ED4 12                   FCB    $12
00255 5ED5 2A                   FCB    $2A
00256 5ED6 7F                   FCB    $7F
00257 5ED7 2A                   FCB    $2A
00258 5ED8 24                   FCB    $24
00259                  * PERCENT SIGN
```

```
00260 5ED9 62              FCB     $62
00261 5EDA 64              FCB     $64
00262 5EDB 08              FCB     $8
00263 5EDC 13              FCB     $13
00264 5EDD 23              FCB     $23
00265                   * AND (&) SIGN
00266 5EDE 50              FCB     $50
00267 5EDF 20              FCB     $20
00268 5EE0 56              FCB     $56
00269 5EE1 49              FCB     $49
00270 5EE2 36              FCB     $36
00272                   * APOSTROPHE
00273 5EE3 00              FCB     $0
00274 5EE4 00              FCB     $0
00275 5EE5 07              FCB     $7
00276 5EE6 00              FCB     $0
00277 5EE7 00              FCB     $0
00278                   * LEFT PARENTHESIS
00279 5EE8 00              FCB     $0
00280 5EE9 41              FCB     $41
00281 5EEA 22              FCB     $22
00282 5EEB 1C              FCB     $1C
00283 5EEC 00              FCB     $0
00284                   * RIGHT PARENTHESIS
00285 5EED 00              FCB     $0
00286 5EEE 1C              FCB     $1C
00287 5EEF 22              FCB     $22
00288 5EF0 41              FCB     $41
00289 5EF1 00              FCB     $0
00290                   * ASTERISK
00291 5EF2 08              FCB     $8
00292 5EF3 2A              FCB     $2A
00293 5EF4 1C              FCB     $1C
00294 5EF5 2A              FCB     $2A
00295 5EF6 08              FCB     $8
00296                   * PLUS SIGN
00297 5EF7 08              FCB     $8
00298 5EF8 08              FCB     $8
00299 5EF9 3E              FCB     $3E
00300 5EFA 08              FCB     $8
00301 5EFB 08              FCB     $8
00302                   * COMMA
00303 5EFC 00              FCB     $0
00304 5EFD 00              FCB     $0
00305 5EFE 38              FCB     $38
00306 5EFF 58              FCB     $58
00307 5F00 00              FCB     $0
00308                   * MINUS SIGN
00309 5F01 08              FCB     $8
00310 5F02 08              FCB     $8
00311 5F03 08              FCB     $8
00312 5F04 08              FCB     $8
00313 5F05 08              FCB     $8
00314                   * DECIMAL POINT
00315 5F06 00              FCB     $0
00316 5F07 00              FCB     $0
00317 5F08 60              FCB     $60
00318 5F09 60              FCB     $60
00319 5F0A 00              FCB     $0
00320                   * SLASH (/)
00321 5F0B 02              FCB     $2
00322 5F0C 04              FCB     $4
00323 5F0D 08              FCB     $8
00324 5F0E 10              FCB     $10
00325 5F0F 20              FCB     $20
00327                   * 0
00328 5F10 3E              FCB     $3E
00329 5F11 45              FCB     $45
00330 5F12 49              FCB     $49
00331 5F13 51              FCB     $51
00332 5F14 3E              FCB     $3E
00333                   * 1
00334 5F15 00              FCB     $0
00335 5F16 40              FCB     $40
00336 5F17 7F              FCB     $7F
```

```
00337 5F18 42           FCB     $42
00338 5F19 00           FCB     $0
00339                 * 2
00340 5F1A 46           FCB     $46
00341 5F1B 49           FCB     $49
00342 5F1C 49           FCB     $49
00343 5F1D 51           FCB     $51
00344 5F1E 62           FCB     $62
00345                 * 3
00346 5F1F 36           FCB     $36
00347 5F20 49           FCB     $49
00348 5F21 49           FCB     $49
00349 5F22 41           FCB     $41
00350 5F23 22           FCB     $22
00351                 * 4
00352 5F24 10           FCB     $10
00353 5F25 7F           FCB     $7F
00354 5F26 12           FCB     $12
00355 5F27 14           FCB     $14
00356 5F28 18           FCB     $18
00357                 * 5
00358 5F29 39           FCB     $39
00359 5F2A 45           FCB     $45
00360 5F2B 45           FCB     $45
00361 5F2C 45           FCB     $45
00362 5F2D 27           FCB     $27
00363                 * 6
00364 5F2E 31           FCB     $31
00365 5F2F 49           FCB     $49
00366 5F30 49           FCB     $49
00367 5F31 4A           FCB     $4A
00368 5F32 3C           FCB     $3C
00369                 * 7
00370 5F33 03           FCB     $3
00371 5F34 05           FCB     $5
00372 5F35 09           FCB     $9
00373 5F36 71           FCB     $71
00374 5F37 01           FCB     $1
00375                 * 8
00376 5F38 36           FCB     $36
00377 5F39 49           FCB     $49
00378 5F3A 49           FCB     $49
00379 5F3B 49           FCB     $49
00380 5F3C 36           FCB     $36
00382                 * 9
00383 5F3D 1E           FCB     $1E
00384 5F3E 29           FCB     $29
00385 5F3F 49           FCB     $49
00386 5F40 49           FCB     $49
00387 5F41 06           FCB     $6
00388                 * COLON
00389 5F42 00           FCB     $0
00390 5F43 00           FCB     $0
00391 5F44 36           FCB     $36
00392 5F45 36           FCB     $36
00393 5F46 00           FCB     $0
00394                 * GREATER THAN OR EQUAL TO SIGN
00395 5F47 48           FCB     $48
00396 5F48 54           FCB     $54
00397 5F49 62           FCB     $62
00398 5F4A 41           FCB     $41
00399 5F4B 00           FCB     $0
00400                 * LESS THAN SIGN
00401 5F4C 00           FCB     $0
00402 5F4D 41           FCB     $41
00403 5F4E 22           FCB     $22
00404 5F4F 14           FCB     $14
00405 5F50 08           FCB     $8
00406                 * EQUAL SIGN
00407 5F51 14           FCB     $14
00408 5F52 14           FCB     $14
00409 5F53 14           FCB     $14
00410 5F54 14           FCB     $14
00411 5F55 14           FCB     $14
00412                 * GREATER THAN SIGN
00413 5F56 08           FCB     $8
```

```
00414 5F57 14            FCB     $14
00415 5F58 22            FCB     $22
00416 5F59 41            FCB     $41
00417 5F5A 00            FCB     $0
00418                 * QUESTION MARK
00419 5F5B 06            FCB     $6
00420 5F5C 09            FCB     $9
00421 5F5D 51            FCB     $51
00422 5F5E 01            FCB     $1
00423 5F5F 06            FCB     $6
00424                 * COMMERCIAL AT SIGN
00425 5F60 1E            FCB     $1E
00426 5F61 55            FCB     $55
00427 5F62 5D            FCB     $5D
00428 5F63 41            FCB     $41
00429 5F64 3E            FCB     $3E
00430                 * A
00431 5F65 7E            FCB     $7E
00432 5F66 09            FCB     $9
00433 5F67 09            FCB     $9
00434 5F68 09            FCB     $9
00435 5F69 7E            FCB     $7E
00437                 * B
00438 5F6A 36            FCB     $36
00439 5F6B 49            FCB     $49
00440 5F6C 49            FCB     $49
00441 5F6D 49            FCB     $49
00442 5F6E 7F            FCB     $7F
00443                 * C
00444 5F6F 22            FCB     $22
00445 5F70 41            FCB     $41
00446 5F71 41            FCB     $41
00447 5F72 41            FCB     $41
00448 5F73 3E            FCB     $3E
00449                 * D
00450 5F74 3E            FCB     $3E
00451 5F75 41            FCB     $41
00452 5F76 41            FCB     $41
00453 5F77 7F            FCB     $7F
00454 5F78 41            FCB     $41
00455                 * E
00456 5F79 41            FCB     $41
00457 5F7A 49            FCB     $49
00458 5F7B 49            FCB     $49
00459 5F7C 49            FCB     $49
00460 5F7D 7F            FCB     $7F
00461                 * F
00462 5F7E 01            FCB     $1
00463 5F7F 09            FCB     $9
00464 5F80 09            FCB     $9
00465 5F81 09            FCB     $9
00466 5F82 7F            FCB     $7F
00467                 * G
00468 5F83 71            FCB     $71
00469 5F84 51            FCB     $51
00470 5F85 41            FCB     $41
00471 5F86 41            FCB     $41
00472 5F87 3E            FCB     $3E
00473                 * H
00474 5F88 7F            FCB     $7F
00475 5F89 08            FCB     $8
00476 5F8A 08            FCB     $8
00477 5F8B 08            FCB     $8
00478 5F8C 7F            FCB     $7F
00479                 * I
00480 5F8D 00            FCB     $0
00481 5F8E 41            FCB     $41
00482 5F8F 7F            FCB     $7F
00483 5F90 41            FCB     $41
00484 5F91 00            FCB     $0
00485                 * J
00486 5F92 3F            FCB     $3F
00487 5F93 40            FCB     $40
00488 5F94 40            FCB     $40
00489 5F95 40            FCB     $40
00490 5F96 20            FCB     $20
```

```
00492                  * K
00493 5F97 41                FCB     $41
00494 5F98 22                FCB     $22
00495 5F99 14                FCB     $14
00496 5F9A 08                FCB     $8
00497 5F9B 7F                FCB     $7F
00498                  * L
00499 5F9C 40                FCB     $40
00500 5F9D 40                FCB     $40
00501 5F9E 40                FCB     $40
00502 5F9F 40                FCB     $40
00503 5FA0 7F                FCB     $7F
00504                  * M
00505 5FA1 7F                FCB     $7F
00506 5FA2 02                FCB     $2
00507 5FA3 0C                FCB     $C
00508 5FA4 02                FCB     $2
00509 5FA5 7F                FCB     $7F
00510                  * N
00511 5FA6 7F                FCB     $7F
00512 5FA7 10                FCB     $10
00513 5FA8 08                FCB     $8
00514 5FA9 04                FCB     $4
00515 5FAA 7F                FCB     $7F
00516                  * O
00517 5FAB 3E                FCB     $3E
00518 5FAC 41                FCB     $41
00519 5FAD 41                FCB     $41
00520 5FAE 41                FCB     $41
00521 5FAF 3E                FCB     $3E
00522                  * P
00523 5FB0 06                FCB     $6
00524 5FB1 09                FCB     $9
00525 5FB2 09                FCB     $9
00526 5FB3 09                FCB     $9
00527 5FB4 7F                FCB     $7F
00528                  * Q
00529 5FB5 5E                FCB     $5E
00530 5FB6 21                FCB     $21
00531 5FB7 51                FCB     $51
00532 5FB8 41                FCB     $41
00533 5FB9 3E                FCB     $3E
00534                  * R
00535 5FBA 46                FCB     $46
00536 5FBB 29                FCB     $29
00537 5FBC 19                FCB     $19
00538 5FBD 09                FCB     $9
00539 5FBE 7F                FCB     $7F
00540                  * S
00541 5FBF 32                FCB     $32
00542 5FC0 49                FCB     $49
00543 5FC1 49                FCB     $49
00544 5FC2 49                FCB     $49
00545 5FC3 26                FCB     $26
00547                  * T
00548 5FC4 01                FCB     $1
00549 5FC5 01                FCB     $1
00550 5FC6 7F                FCB     $7F
00551 5FC7 01                FCB     $1
00552 5FC8 01                FCB     $1
00553                  * U
00554 5FC9 3F                FCB     $3F
00555 5FCA 40                FCB     $40
00556 5FCB 40                FCB     $40
00557 5FCC 40                FCB     $40
00558 5FCD 3F                FCB     $3F
00559                  * V
00560 5FCE 07                FCB     $7
00561 5FCF 18                FCB     $18
00562 5FD0 60                FCB     $60
00563 5FD1 18                FCB     $18
00564 5FD2 07                FCB     $7
00565                  * W
00566 5FD3 7F                FCB     $7F
00567 5FD4 20                FCB     $20
00568 5FD5 18                FCB     $18
```

```
00569 5FD6 20            FCB     $20
00570 5FD7 7F            FCB     $7F
00571                 * X
00572 5FD8 63            FCB     $63
00573 5FD9 14            FCB     $14
00574 5FDA 08            FCB     $8
00575 5FDB 14            FCB     $14
00576 5FDC 63            FCB     $63
00577                 * Y
00578 5FDD 03            FCB     $3
00579 5FDE 04            FCB     $4
00580 5FDF 78            FCB     $78
00581 5FE0 04            FCB     $4
00582 5FE1 03            FCB     $3
00583                 * Z
00584 5FE2 43            FCB     $43
00585 5FE3 45            FCB     $45
00586 5FE4 49            FCB     $49
00587 5FE5 51            FCB     $51
00588 5FE6 61            FCB     $61
00589                 * DIVIDE SIGN
00590 5FE7 08            FCB     $8
00591 5FE8 08            FCB     $8
00592 5FE9 2A            FCB     $2A
00593 5FEA 08            FCB     $8
00594 5FEB 08            FCB     $8
00595                 * RIGHT ARROW
00596 5FEC 08            FCB     $8
00597 5FED 1C            FCB     $1C
00598 5FEE 2A            FCB     $2A
00599 5FEF 08            FCB     $8
00600 5FF0 08            FCB     $8
00602                 * PI
00603 5FF1 02            FCB     $2
00604 5FF2 7C            FCB     $7C
00605 5FF3 04            FCB     $4
00606 5FF4 7C            FCB     $7C
00607 5FF5 08            FCB     $8
00608                 * UP ARROW
00609 5FF6 04            FCB     $4
00610 5FF7 02            FCB     $2
00611 5FF8 7F            FCB     $7F
00612 5FF9 02            FCB     $2
00613 5FFA 04            FCB     $4
00614                 * DOWN ARROW
00615 5FFB 10            FCB     $10
00616 5FFC 20            FCB     $20
00617 5FFD 7F            FCB     $7F
00618 5FFE 20            FCB     $20
00619 5FFF 10            FCB     $10
00620                 * GMBH CHARRACTERS
00621 6000 3C            FCB     $3C     O(..)
00622 6001 43            FCB     $43
00623 6002 42            FCB     $42
00624 6003 43            FCB     $43
00625 6004 3C            FCB     $3C
00626 6005 3C            FCB     $3C     U(..)
00627 6006 41            FCB     $41
00628 6007 40            FCB     $40
00629 6008 41            FCB     $41
00630 6009 3C            FCB     $3C
00631 600A 7C            FCB     $7C     A(..)
00632 600B 13            FCB     $13
00633 600C 12            FCB     $12
00634 600D 13            FCB     $13
00635 600E 7C            FCB     $7C
00636 600F 7C            FCB     $7C     A(.)
00637 6010 12            FCB     $12
00638 6011 13            FCB     $13
00639 6012 12            FCB     $12
00640 6013 7C            FCB     $7C
00641 6014 49            FCB     $49     AE
00642 6015 49            FCB     $49
00643 6016 7F            FCB     $7F
00644 6017 09            FCB     $9
```

```
00645 601A 7E              FCB     $7E
00646 601B 42              FCB     $42       L
00647 601A 41              FCB     $41
00648 601B 49              FCB     $49
00649 601C 7E              FCB     $7E
00650 601D 48              FCB     $48
00652                  * SPANISH CHARACTERS
00653 601E 30              FCB     $30       UPSIDE DOWN ?
00654 601F 40              FCB     $40
00655 6020 45              FCB     $45
00656 6021 48              FCB     $48
00657 6022 30              FCB     $30
00658 6023 7D              FCB     $7D       N BAR
00659 6024 21              FCB     $21
00660 6025 11              FCB     $11
00661 6026 09              FCB     $9
00662 6027 7D              FCB     $7D
00663                  * ALPHA
00664 6028 26              FCB     $26
00665 6029 18              FCB     $18
00666 602A 24              FCB     $24
00667 602B 24              FCB     $24
00668 602C 18              FCB     $18
00672       0016   PSTN    EQU     TP2
00673       001A   STACK   EQU     TP4
00674       002D   ROW     EQU     T2
00675       002C   PASS    EQU     T3
00676       002B   DOT     EQU     T4
00677       002A   PULSE   EQU     T5
00678                  *
00679                  *
00680                  * DAVE UHLRICH   AUGUST 14,1974   PRINTER   REV G
00681                  *
00682                  *
00683                  * THIS ROUTINE TAKES THE ASCII DATA IN BUFF,
00684                  * CONVERTS THEM TO THEIR DOT MATRIX
00685                  * EQUIVALENTS, AND PRINTS THEM ON THE
00686                  * PRINTER.  IT DOES THE CONVERSION AND
00687                  * PRINTING ON A ROW-BY-ROW BASIS, USING
00688                  * AN ADAPTIVE FOUR-PASS SYSTEM.  FOR
00689                  * EXAMPLE, IT WILL PRINT ROW 1 OF
00690                  * CHARACTERS 1,5,9,&13 ON THE FIRST
00691                  * PASS, ROW 1 OF 2,6,10,&14 ON THE SECOND,
00692                  * ETC. UNLESS THE NUMBER OF DOTS TO BE
00693                  * PRINTED IN ANY ONE PASS EXCEEDS
00694                  * TEN.  IN THAT CASE, IT WILL PRINT THE
00695                  * ROW DATA TWO CHARACTERS AT A TIME
00696                  * FOR THAT PARTICULAR PASS ONLY.
00697                  *
00698                  * PRINTER SELECT CODE=1101 IN LOWER 4 BITS OF ADATA.
00699                  * PRINT SELECTS IN UPPER 4 BITS OF ADATA:
00700                  *    1) #1=1110
00701                  *    2) #2=1101
00702                  *    3) #3=1011
00703                  *    4) #4=0111
00704                  * PRINTER LOAD CODE IS 101 IN BITS
00705                  * 3,4,&5 OF BCTL.  THE PRINTER SHIFT
00706                  * REGISTER IS CONNECTED TO BIT 0 OF BDATA.
00707                  *
00708                  *
00709 602D 96 07   PRNTR  LDA A  TGL       LOAD TOGGLE SWITCH REG
00710 602F 85 60          BIT A  #$60      PRINT ON OR AUDIT?
00711 6031 26 01          BNE    PRNTR1    YES, PRINT
00712 6033 39             RTS              NO, RETURN
00713 6034 86 0C   PRNTR1 LDA A  #$C
00714 6036 97 00          STA A  ADATA     SEND OUT TOGGLE SWITCH ENABLE
00715 6038 D6 04          LDA B  INPUT     LOAD TOGGLE SWITCHES
00716 603A 4F             CLR A
00717 603B 97 00          STA A  ADATA     CLEAR SELECT CODE
00718 603D C5 02          BIT B  #2        OUT OF PAPER?
00719 603F 27 05          BEQ    PRNTR2    NO, PRINT
00720 6041 86 17          LDA A  #23       YES, LOAD ERROR CODE
00721 6043 97 06          STA A  ERROR
00722 6045 39             RTS
00723 6046 86 F0   PRNTR2 LDA A  #$F0
```

```
00724 6048 BD 6128          JSR     PPRADV   PAPER ADVANCE
00725 604B 4F               CLR A
00726 604C 97 1A            STA A   STACK    SET SPH=0
00727 604E 4C               INC A
00728 604F 97 2D            STA A   ROW      SET ROW PTR=1
00729 6051 86 08   NXTROW   LDA A   #8
00730 6053 97 2C            STA A   PASS     SET PASS CTR
00731 6055 4F               CLR A
00732 6056 97 2B            STA A   DOT
00733 6058 86 68            LDA A   #REAL
00734 605A 97 1B            STA A   STACK+1   SET SPL=REAL
00735 605C CE 0067          LDX     #$67     POINT PSTN PTR AT LAST CHAR
00736 605F 0F               SEI              DISABLE KEY INTERRUPT
00737 6060 A6 00   LDASC    LDA A   X        LOAD ASCII CODE
00738 6062 84 7F            AND A   #$7F     LOP OFF TOP BIT
00739 6064 81 68            CMP A   #$68     IS ASCII CODE > $68?
00740 6066 22 04            BHI     GT68     YES, TREAT IT AS A BLANK
00741 6068 80 20            SUB A   #$20     SUBTRACT OFF ASCII OFFSET
00742 606A 2A 01            BPL     OK       IS ASCII CODE < $20?
00743 606C 4F       GT68    CLR A            YES, TREAT IT AS A BLANK
00744 606D C6 5E    OK      LDA B   #DOTS/$100   LOAD DOT VECTOR ADDR HIGH
00745 606F 97 2A            STA A   PULSE
00746 6071 48               ASL A
00747 6072 48               ASL A            MULTIPLY BY 4
00748 6073 C9 00            ADC B   #0       IF CARRY, INCR ADDR HIGH
00749 6075 9B 2A            ADD A   PULSE    ADD TO MULTIPLY BY 5
00750 6077 C9 00            ADC B   #0       IF CARRY, INCR ADDR HIGH
00751 6079 8B C0            ADD A   #$C0     ADD TABLE ADDR LOW TO OFFSET
00752 607B 97 19            STA A   TP3S     STORE IT IN ADDR LOW
00753 607D C9 00            ADC B   #0       IF CARRY, INCR ADDR HIGH
00754 607F D7 18            STA B   TP3      STORE ADDRESS HIGH
00755 6081 DF 16            STX     PSTN     STORE POS PTR
00756 6083 DE 18            LDX     TP3      LOAD DOT MATRIX ADDR
00757 6085 C6 F8            LDA B   #$F8     LOAD COLUMN CTR
00758 6087 58       LDDOT   ASL B            DECR COLUMN CTR
00759 6088 A6 00            LDA A   X        LOAD DOT COLUMN
00760 608A 94 2D            AND A   ROW      CHOOSE DOT OF ONE ROW
00761 608C 27 04            BEQ     NODOT    IS IT A ZERO?
00762 608E 5C               INC B            NO, LOAD ONE IN DOT ROW
00763 608F 7C 002B          INC     DOT      INCR DOT CTR
00764 6092 08       NODOT   INX              POINT TO NEXT COLUMN
00765 6093 C5 E0            BIT B   #$E0     DONE BUILDING DOT VECTOR?
00766 6095 26 F0            BNE     LDDOT    NO, LOAD NEXT DOT
00767 6097 DE 1A            LDX     STACK    LOAD INDEX WITH STACK PTR
00768 6099 E7 00            STA B   X        STORE DOT VECTOR ON STACK
00769 609B 7C 001B          INC     STACK+1   INCREMENT STACK PTR
00770 609E 96 17            LDA A   PSTN+1   LOAD LOWER HALF PSTN PTR
00771 60A0 80 04            SUB A   #4       DECR IT BY 4
00772 60A2 97 17            STA A   PSTN+1   STORE IT BACK
00773 60A4 DE 16            LDX     PSTN     LOAD INDEX WITH POS PTR
00774 60A6 81 58            CMP A   #$58     IS POS PTR<$58?
00775 60A8 2C B6            BGE     LDASC    NO, CONVERT NEXT ASCII CHAR
00776 60AA 96 2B            LDA A   DOT      LOAD DOT CTR
00777 60AC 16               TAB              PUT IT IN ACCB ALSO
00778 60AD 44               LSR A            SHIFT DOT FLAGS RIGHT ONE BIT
00779 60AE C4 1F            AND B   #$1F     MASK OFF NO. OF DOTS
00780 60B0 C1 0A            CMP B   #10      NO. OF DOTS >10?
00781 60B2 2F 02            BLE     LT10     NO, DO NOT SET DOT FLAG
00782 60B4 8B 80            ADD A   #$80     YES, SET DOT FLAG
00783 60B6 84 F0   LT10     AND A   #$F0     SAVE DOT FLAGS AND
00784 60B8 97 2B            STA A   DOT      CLEAR DOT CTR
00785 60BA 96 17            LDA A   PSTN+1   LOAD POS PTR
00786 60BC 81 54            CMP A   #$54     END OF COMPLETE ROW?
00787 60BE 27 0B            BEQ     ENDROW   YES, PAPER ADVANCE
00788 60C0 8B 0F            ADD A   #15      ADD 15 TO POS PTR
00789 60C2 97 17            STA A   PSTN+1   STORE BEG OF NEXT PASS
00790 60C4 DE 16            LDX     PSTN     LOAD POS PTR INTO INDEX
00791 60C6 74 002C          LSR     PASS     SHIFT PASS CTR
00792 60C9 20 95            BRA     LDASC
00793 60CB CE FE7F ENDROW   LDX     #-385    7.5-4.42=WAIT TO FINISH OFF TIME
00794 60CE BD 613B          JSR     LOOP
00795 60D1 86 FD            LDA A   #$FD     LOAD PRINTER SELECT CODE
00796 60D3 97 00            STA A   ADATA    TURN PRINTER ON
00797 60D5 C6 2C   OUTPT    LDA B   #$2C
00798 60D7 D7 03            STA B   BCTL     ENABLE PRINTER SHIFT REG
00799 60D9 8D 3D            BSR     LDCHAR   SEND DOTS OUT TO PRINTER
```

```
00800 60DB 8D 3B          BSR    LDCHAR   SEND DOTS OUT TO PRINTER
00801 60DD 96 28          LDA A  DOT
00802 60DF 2A 0A          BPL    NOOVFL   MORE THAN 10 DOTS ON THIS PASS?
00803 60E1 5F             CLR B
00804 60E2 86 F6          LDA A  #-10     YES, SET ZERO CTR=-10
00805 60E4 D7 02    LDO   STA B  BDATA    SHIFT 0 INTO PRINTER REG
00806 60E6 4C             INC A            INCR ZERO CTR
00807 60E7 26 FB          BNE    LDO      10 ZEROES LOADED?
00808 60E9 8D 54          BSR    PRINT    YES. PRINT
00809 60EB 8D 28    NOOVFL BSR   LDCHAR   SEND DOTS OUT TO PRINTER
00810 60ED 8D 29          BSR    LDCHAR   SEND DOTS OUT TO PRINTER
00811 60EF 8D 4E          BSR    PRINT    PRINT
00812 60F1 86 3C          LDA A  #$3C
00813 60F3 97 03          STA A  BCTL     TURN PRINTER SHIFT REG OFF
00814 60F5 78 0028        ASL    DOT      POINT TO NEXT DOT FLAG
00815 60F8 96 2C          LDA A  PASS
00816 60FA 48             ASL A
00817 60FB 97 2C          STA A  PASS
00818 60FD 81 10          CMP A  #16      PRINTED ALL FOUR PASSES?
00819 60FF 26 D4          BNE    OUTPT    NO, PRINT NEXT PASS
00820 6101 CE FF06        LDX    #-250    WAIT 2 MSEC AFTER BURN
00821 6104 8D 35          BSR    LOOP
00822 6106 0E             CLI             ENABLE KEY INTERRUPT
00823 6107 86 F0          LDA A  #$F0
00824 6109 97 00          STA A  ADATA    TURN PRINTER OFF
00825 6108 78 002D        ASL    ROW      SHIFT ROW PTR
00826 610E 2B 03          BMI    DONE     MINUS, PRINT COMPLETE
00827 6110 7E 6051        JMP    NXTROW
00828 6113 8D 13    DONE  BSR    PPRADV   PAPER ADVANCE
00829 6115 8D 11          BSR    PPRADV   PAPER ADVANCE
00830 6117 39             RTS
00831 6118 7A 001B LDCHAR DEC    STACK+1   DECR STACK PTR
00832 611B DE 1A          LDX    STACK    LOAD INDEX WITH STACK PTR
00833 611D E6 00          LDA B  X        LOAD DOT VECTOR
00834 611F D7 02    SNDOT STA B  BDATA    SEND DOT OUT TO PRINTER
00835 6121 0D             SEC             SET CARRY
00836 6122 56             ROR B           ROTATE NEXT DOT INTO POSITION
00837 6123 C1 F8          CMP B  #$F8     DONE WITH CHAR?
00838 6125 26 F8          BNE    SNDOT    NO, SEND OUT ANOTHER DOT
00839 6127 39             RTS
00840 6128 0F       PPRADV SEI             DISABLE KEY INTERRUPT
00841 6129 CE FC57        LDX    #-937    LOAD 7.5 MSEC OFF TIME
00842 612C 8D 0D          BSR    LOOP
00843 612E CE F38D        LDX    #-3187   LOAD 25.5 MSEC ON COUNT
00844 6131 C6 FD          LDA B  #$FD
00845 6133 D7 00          STA B  ADATA    TURN PRINTER ON
00846 6135 8D 04          BSR    LOOP
00847 6137 97 00          STA A  ADATA    TURN PRINTER OFF
00848 6139 0E             CLI             ENABLE KEY INTERRUPT
00849 613A 39             RTS
00850 613B 08       LOOP  INX
00851 613C 26 FD          BNE    LOOP
00852 613E 39             RTS
00853 613F DF 16    PRINT STX    PSTN     STORE PSTN PTR
00854 6141 CE FF06        LDX    #-250    LOAD 2.0 MSEC BURN COUNT
00855 6144 86 75          LDA A  #117
00856 6146 97 2A          STA A  PULSE    SET PULSE CTR
00857 6148 96 2C          LDA A  PASS     LOAD PASS CTR
00858 614A 43             COM A           COMPLEMENT IT
00859 614B 48             ASL A
00860 614C 48             ASL A
00861 614D 48             ASL A
00862 614E 48             ASL A
00863 614F 8A 0D          ORA A  #$D      OR WITH PRINTER SELECT CODE
00864 6151 97 00          STA A  ADATA    START PRINT
00865 6153 8D E6          BSR    LOOP     HOLD PRINT FOR 2.0 MSEC
00866 6155 C6 FD    DUTY  LDA B  #$FD
00867 6157 D7 00          STA B  ADATA    TURN OFF PRINT PULSE
00868 6159 7A 002A        DEC    PULSE    HOLD PRINT PULSE OFF 15 USEC
00869 615C 26 03          BNE    MRPLS    PRINT COMPLETE?
00870 615E DE 16          LDX    PSTN     YES, RELOAD PSTN PTR
00871 6160 39             RTS             RETURN
00872 6161 97 00    MRPLS STA A  ADATA    TURN PRINT PULSE BACK ON
00873 6163 87 07FF        STA A  $7FF     HOLD PRINT PULSE ON FOR 15 USEC
00874 6166 20 ED          BRA    DUTY
00877                     END
```

SYMBOL TABLE

ADATA	0000	ACTL	0001	BDATA	0002	BCTL	0003	INPUT	0004	IOIN	0005
ERROR	0006	TGL	0007	UFLG	0008	RSFLG	0009	EOM	000A	EOPM	0008
STKFLG	000D	RND	000E	DIGFLG	000F	W2	0010	W1	0011	SFLG	0012
DCNTR	0013	TP1	0014	TP1S	0015	TP2	0016	TP2S	0017	TP3	0018
TP3S	0019	TP4	001A	TP4S	0018	TP5	001C	TP5S	001D	TP6	001E
TP6S	001F	TP7	0020	TP7S	0021	T13	0022	T12	0023	T11	0024
T10	0025	T9	0026	T8	0027	T7	0028	T6	0029	T5	002A
T4	002B	T3	002C	T2	002D	T1	002E	ISTK	002F	ISTACK	0051
TA	0052	SPGM	0054	EXTRA	0056	BUFF	0058	REAL	0068	IMAG	0070
AT1	0078	AT2	0080	W	0088	XR	0090	YR	0098	ZR	00A0
TR	00A8	LSTX	00B0	BKWRT	00B8	BKKC	00BA	SOL7	00C6	UPP	00C8
UIP	00CA	ALPHA	00CC	IO1	00CD	IO2	00D0	IT7	00D3	FLAG	00D5
TPOS	00D6	FILE	00D7	AR	00D8	BR	00E0	CR	00E8	DR	00F0
ER	00F8	SDBB	00BA	MT	7E00	TERMN7	003D	IMED	0040	PARCD	00C0
PAREX	0080	NTBL	0000	DOTS	5EC0	PRTDRV	602D	FRMT	5CA8	BLANK	5D75
LDMSG	578D	ROLLD	55B2	ROLLU	57F1	PSD	550A	TXL	55E9	STKUP	55EF
MAD	7498	CMP	74AA	NOR	74D6	TXW	7424	TXXR	743B	EXXR	7452
ARSR	753B	OVUNF	75B6	OVERF	75DD	XR0	740A	XRNINE	75C8	UNDRF	75F1
IMULT	76B9	QDG	7669	FPA	75FC	FPS	75F6	FPM	7735	FPD	7793
FPAEX	763D	FPMEX	7780	LSHIFT	7521	ZEROX	7489	XZERO0	7416	XZERO2	7417
RECIP	73E6	TXRX	73F3	CONST	6800	FPDBRC	6898	TAN	68A9	ATN	69C3
DSZERO	6A46	NTLN	6A58	EXPN	6AC9	SIN	6B94	COS	6B9A	ASIN	6BF2
ACOS	6BF7	PH1	6C5D	PH2	6C8D	PH3	6D34	PH4	6DD0	LSFT8	6E47
SQRT	6E65	MAD8	6F2C	CMP8	53E4	IOUPX	6F52	LOG10	6FA7	YUPX	6FE9
RTOP	7328	PTOR	7386	PSTN	0016	STACK	001A	ROW	002D	PASS	002C
DOT	002B	PULSE	002A	PRNTR	602D	PRNTR1	6034	PRNTR2	6046	NXTROW	6051
LDASC	6060	GT68	606C	OK	606D	LDDOT	6087	NODOT	6092	LT10	60H6
ENDROW	60CR	OUTPT	60D5	LD0	60E4	NOOVFL	60EB	DONE	6113	LDCHAR	6118
SNDOT	611F	PPRADV	6128	LOOP	6138	PRINT	613F	DUTY	6155	MRPLS	6161

```
00124                        OPT    LIST,MEM
00127                *
00128                *
00129                * EQUATE TABLE
00130                *
00131                *
00132      002E      SAVE   EQU    T1
00133      002D      FLCTR  EQU    T2
00134      002C      MASK   EQU    T3
00135      002B      REWFLG EQU    T4
00136      002A      KNWALL EQU    T5
00137      007A      MRKBOT EQU    AT1+2
00138      007B      FNDCMD EQU    AT1+3
00139      007C      STACK  EQU    AT1+4
00140      008A      SVINS  EQU    W+2
00141      008D      SAVSTK EQU    W+5
00142      0028      SPADRS EQU    T7
00143      0026      CFSZ   EQU    T9
00144      0024      AFSZ   EQU    T11
00145      0023      TYPE   EQU    T12
00146      0022      CHKSM  EQU    T13
00147      0016      BFPTR  EQU    TP2
00148      00D5      FLAG   EQU    $D5
00149      00D6      TPOS   EQU    $D6
00150      00D7      FILE   EQU    $D7
00151      4D58      SADRS  EQU    $4D58
00152      49CE      TSFR   EQU    $49CE
00153      484B      ADD7   EQU    $484B
00154      4840      SUB7   EQU    $4840
00155      5CA8      FRMT   EQU    $5CA8
00156      5D75      BLANK  EQU    $5D75
00157      578D      LDMSG  EQU    $578D
00158      48BC      BINBCD EQU    $48BC
00159      55EF      STKUP  EQU    $55EF
00160      740A      XR0    EQU    $740A
00161      74D6      NOR    EQU    $74D6
00162      4EA4      DGTS   EQU    $4EA4
00163      602D      PRTDRV EQU    $602D
00164      57F1      ROLLU  EQU    $57F1
00166 6168                   ORG    $6168
00167                *
00168                *
```

```
00169                    * CASSETTE INITIALIZATION AND EXIT ROUTINES.
00170                    *
00171                    *
00172 6168 0F      SETUP SEI              DISABLE KEY INTERRUPT
00173 6169 DF 8A         STX    SVINS    SAVE INSTR. STARTING ADDRESS
00174 616B 30            TSX
00175 616C 08            INX
00176 616D DF 8D         STX    SAVSTK   SAVE STACK POINTER
00177 616F 96 03         LDA A  BCTL     CHECK IF CARTRIDGE HAS BEEN OUT
00178 6171 2A 03         BPL    CRTIN
00179 6173 4F            CLR A
00180 6174 97 D6         STA A  TPOS     TPOS=LOST
00181 6176 BD 6740 CRTIN JSR    COT      CHECK IF CARTRIDGE IS STILL OUT
00182 6179 16            TAB
00183 617A BD 672C       JSR    LIGHT    TAPE SPOOLED OFF?
00184 617D 96 02         LDA A  BDATA    RESET EOT & COT INTERRUPT BIT
00185 617F 4F            CLR A
00186 6180 97 0F         STA A  DIGFLG   CLEAR DIGIT ENTRY FLAG
00187 6182 97 28         STA A  REWFLG
00188 6184 97 2A         STA A  KNWALL
00189 6186 97 7A         STA A  MPKBOT
00190 6188 86 05         LDA A  #5
00191 618A 97 2D         STA A  FLCTR    SET FILE CTR=5
00192 618C 97 7B         STA A  FNDCMD   SET FIND COMMAND FLAG
00193 618E 86 3D         LDA A  #$3D
00194 6190 97 03         STA A  BCTL     ENABLE EOT & COT INTERRUPT
00195 6192 96 D5         LDA A  FLAG
00196 6194 8A 02         ORA A  #2       SET CASSETTE OP FLAG
00197 6196 97 D5         STA A  FLAG
00198 6198 9E 8D         LDS    SAVSTK
00199 619A 09            DEX
00200 619B EE 00         LDX    X        SET INDEX TO RETURN ADDRESS
00201 619D AD 00         JSR    X        DO INSTRUCTION
00202 619F 81 0E ERRMSG  CMP A  #14      CHECKSUM ERROR?
00203 61A1 26 16         BNE    STERR
00204 61A3 DE 8A         LDX    SVINS    LOAD ADDRESS OF INSTR JUST EXECUTED
00205 61A5 8C 6398       CPX    #VERIFY   CHECKSUM ERROR ON VERIFY?
00206 61A8 27 0F         BEQ    STERR    YES, DON'T TRY AGAIN
00207 61AA D6 D5         LDA B  FLAG
00208 61AC CB 20         ADD B  #$20     INCREMENT TRY CTR
00209 61AE D7 D5         STA B  FLAG
00210 61B0 C4 60         AND B  #$60     MASK OFF TRY CTR
00211 61B2 C1 60         CMP B  #$60     FAILED THREE TIMES?
00212 61B4 27 03         BEQ    STERR    YES, PRINT ERROR
00213 61B6 7E 6781       JMP    OVER     NO, TRY AGAIN
00214 61B9 97 06 STERR   STA A  ERROR
00215 61BB BD 66D3       JSR    STOP     MAKE SURE TAPE IS STOPPED
00216 61BE 9E 8D TRNOFF  LDS    SAVSTK   RESET STACK POINTER
00217 61C0 86 3C         LDA A  #$3C
00218 61C2 97 03         STA A  BCTL     DISABLE EOT & COT INTERRUPT
00219 61C4 0E            CLI             ENABLE KEY INTERRUPT
00220 61C5 D6 D5         LDA B  FLAG     LOAD CASSETTE OP FLAG
00221 61C7 C4 9D         AND B  #$9D     CLEAR IT
00222 61C9 D7 D5         STA B  FLAG     RESTORE IT
00223 61CB 39            RTS
00225 61CC E6 00 TOBIN   LDA B  X        LOAD EXPNT
00226 61CE 5C            INC B          INCR IT
00227 61CF 2E 04         BGT    NOOK     IS IT > 0?
00228 61D1 86 06 ERR6    LDA A  #6       ILLEGAL ARGUMENT
00229 61D3 20 CA         BRA    ERRMSG
00230 61D5 C1 05 NOOK    CMP B  #5       IS IT < 5?
00231 61D7 2C F8         BGE    ERR6     NO, ERROR
00232 61D9 7E 49CE       JMP    TSFR     YES, CONVERT NO. TO BINARY
00234                    *
00235                    *
00236                    * THIS ROUTINE MARKS THE NUMBER OF FILES
00237                    * CONTAINED IN THE Y-REGISTER.  THE SIZE
00238                    * OF EACH FILE IS FOUND IN THE Z-REGISTER.
00239                    * THE STARTING FILE NUMBER IS IN THE
00240                    * X-REGISTER.
00241                    *
00242                    *
00243 61DC 8D 8A MARK    BSR    SETUP
00244 61DE 54            LSR B          RECORD INHIBITED?
00245 61DF 25 03         BCS    MRKOK    NO, OKAY TO RECORD
00246 61E1 86 15         LDA A  #21      RECORD INHIBIT
```

```
00247 61E3 39              RTS
00248 61E4 CE 0098 MRKOK   LDX    #YR       SET INDEX TO YR ADDRESS
00249 61E7 8D E3           BSR    TOBIN     CONVERT ITS CONTENTS TO BINARY
00250 61E9 96 20           LDA A  TP7
00251 61EB 26 E4           BNE    ERR6      NO. OF FILES TO BE MARKED>255?
00252 61ED 96 21           LDA A  TP7S
00253 61EF 27 E0           BEQ    ERR6      NO. OF FILES TO BE MARKED=0?
00254 61F1 97 8C           STA A  W+4       SAVE NO. OF FILES TO BE MARKED
00255 61F3 CE 00A0         LDX    #ZR       SET INDEX TO ZR ADDRESS
00256 61F6 8D D4           BSR    TOBIN     CONVERT ITS CONTENTS TO BINARY
00257 61F8 DE 20           LDX    TP7
00258 61FA 27 D5           BEQ    ERR6      FILE SIZE=0. ILLEGAL ARGUMENT
00259 61FC DF 1C           STX    TP5       SAVE FILE SIZE IN TP5
00260 61FE BD 6527         JSR    FILENO    STARTING FILE NO. TO BINARY
00261 6201 96 21           LDA A  TP7S      LOAD NEW FILE NUMBER
00262 6203 27 14           BEQ    BEGIN     IF =0, SET EVERYTHING UP FOR MARK
00263 6205 16              TAB
00264 6206 9B 8C           ADD A  W+4       NEW FILE+NO. OF FILES>255?
00265 6208 25 C7           BCS    ERR6      YES, ILLEGAL ARGUMENT
00266 620A D1 D7           CMP B  FILE      NEW FILE=OLD FILE?
00267 620C 26 05           BNE    FIND1     NO, FIND FILE
00268 620E 96 D6           LDA A  TP0S      YES. CHECK TAPE POSITION
00269 6210 44              LSR A            IN GAP?
00270 6211 25 06           BCS    BEGIN     YES, DON'T SEARCH; BEGIN MARKING
00271 6213 BD 6553 FIND1   JSR    FIND+2    FIND FILE
00272 6216 BD 65AA         JSR    CHECK     CHECK IF RIGHT FILE
00273 6219 4F        BEGIN CLR A
00274 621A 97 7B           STA A  FNDCMD    CLEAR FIND CMMD FLAG
00275 621C 97 26           STA A  CFSZ
00276 621E 97 27           STA A  CFSZ+1    SET CURR FILE SIZE=0
00277 6220 DE 1C           LDX    TP5
00278 6222 DF 24           STX    AFSZ      SET ABS FILE SIZE
00279 6224 96 1C           LDA A  TP5
00280 6226 26 17           BNE    NOSLK     FILE SIZE>255, NO SLACK
00281 6228 D6 1D           LDA B  TP5S
00282 622A C1 50           CMP B  #80       ABS FILE SIZE<80?
00283 622C 22 05           BHI    ADDSLK    NO. ADD SLACK
00284 622E CE 0050         LDX    #80       YES, SET MARKED FILE SIZE TO 80
00285 6231 DF 1C           STX    TP5       SET NEW ABS FILE SIZE=80
00286 6233 C6 19   ADDSLK  LDA B  #25
00287 6235 DB 1D           ADD B  TP5S      ADD 25 BYTES SLACK
00288 6237 24 02           BCC    NOMORE    SIZE+SLACK>255?
00289 6239 4C              INC A
00290 623A 5F              CLR B            YES, ONLY ENOUGH SLACK TO =256
00291 623B D7 1D   NOMORE  STA B  TP5S
00292 623D 97 1C           STA A  TP5
00293 623F C6 05   NOSLK   LDA B  #5
00294 6241 D7 23           STA B  TYPE      SET FILE TYPE=EMPTY
00295 6243 96 21           LDA A  TP7S      LOAD NEW FILE NO.
00296 6245 26 05           BNE    START     IF NOT 0, BEGIN MARKING.
00297 6247 D7 7A           STA B  MRKBOT    IF=0,SET MARK BEG. OF TAPE FLAG
00298 6249 7E 6742         JMP    REWIND+5    REWIND TAPE TO LOAD POINT
00299 624C BD 6668 START   JSR    PART      MARK PART OF PREAMBLE
00300 624F 20 18           BRA    MRKPRE
00301 6251 86 67   FILE0   LDA A  #$67      FORWARD SLOW WRITE DATA=0
00302 6253 97 00           STA A  ADATA
00303 6255 96 00   NXTFL   LDA A  ADATA     RECORD GAP
00304 6257 0E              CLI
00305 6258 02              NOP
00306 6259 0F              SEI
00307 625A 97 00           STA A  ADATA     RESTART TAPE IF INTERRUPTED
00308 625C CE 927C         LDX    #37500    SET INDEX TO MARK 3" GAP
00309 625F BD 66DE         JSR    WAIT
00310 6262 88 80           EOR A  #$80
00311 6264 97 00           STA A  ADATA     YES, CAUSE TRANSITION
00312 6266 BD 6676         JSR    PREMBL    WRITE 1ST PREAMBLE
00313 6269 BD 667F MRKPRE  JSR    RCDHD     RECORD HEADING
00314 626C DE 24           LDX    AFSZ      MARKER FILE?
00315 626E 27 1D           BEQ    FIN1      YES. ALL DONE
00316 6270 DE 1C           LDX    TP5       NO, LOAD INDEX WITH NO. OF BYTES
00317 6272 86 FF           LDA A  #$FF      SET ACCA WITH ALL ONES
00318 6274 BD 663B MRKMR   JSR    RECORD    RECORD BYTE OF ALL ONES
00319 6277 09              DEX              DECR BYTE CTR
00320 6278 26 FA           BNE    MRKMR     DONE? NO, MARK ANOTHER BYTE
00321 627A 7C 00D7         INC    FILE      YES, INCR FILE NO.
00322 627D 7A 008C         DEC    W+4       DECR NO. OF FILES TO BE MARKED
```

```
00323 6280 26 D3          BNE     NXTFL     DONE? NO, MARK NEXT FILE
00324 6282 4F             CLR A
00325 6283 97 24          STA A   AFSZ
00326 6285 97 25          STA A   AFSZ+1    SET ABS FILE SIZE=0
00327 6287 86 06          LDA A   #6
00328 6289 97 23          STA A   TYPE      TYPE=MARKER FILE
00329 628B 20 C8          BRA     NXTFL
00330 628D 96 99    FIN1  LDA A   YR+1
00331 628F 2B 05          BMI     FIN2      NO. OF FILES<0, ERASE TO END OF TAPE
00332 6291 BD 66C9        JSR     RETGP
00333 6294 4F             CLR A
00334 6295 39             RTS
00335 6296 86 80    FIN2  LDA A   #$80
00336 6298 97 7A          STA A   MRKBOT    SET ERASE TO END OF TAPE FLAG
00337 629A 96 00          LDA A   ADATA     LOAD PRESENT TAPE STATUS
00338 629C 84 DF          AND A   #$DF      FORWARD FAST WRITE
00339 629E 7E 6747        JMP     REWIND+10
00341                *
00342                *
00343                * THIS ROUTINE RECORDS INFORMATION IN THE CORE
00344                * OF THE CALCULATOR INTO THE FILE POINTED
00345                * TO BY THE NUMBER IN THE X-REGISTER.  THERE
00346                * ARE FOUR RECORD INSTRUCTIONS:
00347                *   1) FORMAT RECORD
00348                *   2) RECORD PROGRAM
00349                *   3) RECORD DATA
00350                *   4) STORE BINARY
00351                * THE RECORD PROGRAM INSTRUCTION STORES THE
00352                * PROGRAM BEGINNING AT THE PRESENT PROGRAM
00353                * POSITION INTO THE FILE ON THE TAPE.
00354                * THE FORMAT RECORD INSTRUCTION DOES THE SAME
00355                * EXCEPT THAT THE PROGRAM IS STORED AS
00356                * A SECURED PROGRAM.  THE RECORD DATA
00357                * INSTRUCTION USES THE STARTING REGISTER NUMBER IN
00358                * THE Y-REGISTER AND THE NUMBER OF REGISTERS IN THE
00359                * Z-REGISTER TO STORE DATA INTO A FILE ON THE
00360                * TAPE.  THE STORE BINARY INSTRUCTION STORES A
00361                * SPECIAL PROGRAM ONTO THE TAPE.
00362                *
00363                *
00364 62A1 4C       FRCRD  INC A             FORMAT RECORD ENTRY
00365 62A2 4C       RCPRGM INC A             RECORD PROGRAM ENTRY
00366 62A3 D6 D5           LDA B  FLAG
00367 62A5 54              LSR B             SECURED MEMORY?
00368 62A6 24 05           BCC    STBIN      NO, DO THE RECORD
00369 62A8 86 16           LDA A  #22        SECURED MEMORY ERROR
00370 62AA 97 06           STA A  ERROR
00371 62AC 39              RTS
00372 62AD 4C       STBIN  INC A             STORE BINARY ENTRY POINT
00373 62AE 97 88    RCDATA STA A  W
00374 62B0 BD 6168         JSR    SETUP
00375 62B3 54              LSR B             RECORD INHIBITED?
00376 62B4 25 03           BCS    RCDOK      NO, OKAY TO RECORD
00377 62B6 86 15           LDA A  #21        RECORD INHIBITED
00378 62B8 39              RTS
00379 62B9 96 88    RCDOK  LDA A  W
00380 62BB 81 01           CMP A  #1         STORE BINARY?
00381 62BD 26 1E           BNE    NTSBN      NO, CONTINUE
00382 62BF DE 1A           LDX    TP4        SAVE ENDING ADDR IN TP4
00383 62C1 DF 20           STX    TP7
00384 62C3 DE 54           LDX    SPGM       LOAD START OF SPECIAL PRGM
00385 62C5 DF 1E           STX    TP6
00386 62C7 BD 4840         JSR    SUB7       TP7-TP6=CFSZ
00387 62CA DE 20           LDX    TP7
00388 62CC DF 1C           STX    TP5
00389 62CE DE 54           LDX    SPGM
00390 62D0 DF 20           STX    TP7        STARTING ADDRESS IN TP7
00391 62D2 86 03           LDA A  #3         BINARY FILE TYPE
00392 62D4 D6 D5           LDA B  FLAG
00393 62D6 C5 04           BIT B  #4         RECORD SPEC PRGM SECURED?
00394 62D8 27 71           BEQ    FINRCD     NO, TYPE=3
00395 62DA 4C              INC A             YES, TYPE=4
00396 62DB 20 6E           BRA    FINRCD
00397 62DD CE 0000 NTSBN   LDX    #0
00398 62E0 DF 1C           STX    TP5
```

```
00399 62E2 CE 0098          LDX     #YR         SET INDEX TO YR ADDRESS
00400 62E5 BD 61CC          JSR     TOBIN       CONVERT ITS CONTENTS TO BINARY
00401 62E8 96 88            LDA A   W
00402 62EA 27 2A            BEQ     RDATA       W=0, RECORD DATA
00403 62EC 7C 0020          INC     TP7         GET SYSTEM ADDRESS
00404 62EF DE 20            LDX     TP7         PUT ADDR IN INDEX
00405 62F1 BD 6489          JSR     CHADRS      STARTING ADDR OUT OF RANGE?
00406 62F4 2A 38            BPL     ILLARG      YES, ILLEGAL ARGUMENT
00407 62F6 A6 00    CONT    LDA A   X           NO, LOAD CONTENTS OF MEMORY
00408 62F8 08               INX                 INCR ADDRESS
00409 62F9 7C 001D          INC     TP5S        INCR CURR FILE SIZE
00410 62FC 26 03            BNE     CHKEND      CARRY INTO HIGH 8? NO, CHECK FOR "END
00411 62FE 7C 001C          INC     TP5         INCR CURR FILE SIZE HIGH
00412 6301 81 81    CHKEND  CMP A   #$81        "END"?
00413 6303 27 05            BEQ     ENDFND      YES, END FOUND
00414 6305 BD 6489          JSR     CHADRS      END OF MEMORY?
00415 6308 2B EC            BMI     CONT        NO, CONTINUE IN LOOP
00416 630A DF 1A    ENDFND  STX     TP4         SAVE ENDING ADDR IN TP4
00417 630C 4F               CLR A
00418 630D D6 88            LDA B   W
00419 630F C1 03            CMP B   #3          FORMAT RECORD?
00420 6311 26 38            BNE     FINRCD
00421 6313 4C               INC A               SET TO SECURED PROGRAM TYPE
00422 6314 20 35            BRA     FINRCD
00423 6316 96 20    RDATA   LDA A   TP7
00424 6318 27 03            BEQ     RNOK        REG. NO. >255?
00425 631A 86 18    RNERR   LDA A   #24         ILLEGAL ADDRESS
00426 631C 39               RTS
00427 631D BD 646E  RNOK    JSR     ENDADR      FIND ENDING ADDRESS
00428 6320 CE 00A0          LDX     #ZR         Z-REG ADDR INTO INDEX
00429 6323 BD 61CC          JSR     TOBIN       CONVERT ITS CONTENTS TO BINARY
00430 6326 D6 20            LDA B   TP7
00431 6328 26 F0            BNE     RNERR       #REGS>255? YES, ERROR
00432 632A 96 21            LDA A   TP7S        LOAD NO. OF REGS
00433 632C 26 03            BNE     RGNT0       NO. OF REGS=0?
00434 632E 86 06    ILLARG  LDA A   #6          YES,ILLEGAL ARGUMENT
00435 6330 39               RTS
00436 6331 48       RGNT0   ASL A               MULTIPLY NO. OF REGS BY 8
00437 6332 59               ROL B
00438 6333 49               ROL A
00439 6334 59               ROL B
00440 6335 49               ROL A
00441 6336 59               ROL B
00442 6337 97 27            STA A   CFSZ+1      STORE NEW CURRENT
00443 6339 D7 26            STA B   CFSZ        FILE SIZE
00444 6338 97 1D            STA A   TP5S
00445 6330 D7 1C            STA B   TP5
00446 633F DE 1A            LDX     TP4         RELOAD ENDING ADDRESS
00447 6341 BD 6470          JSR     STADR       FIND STARTING ADDRESS
00448 6344 BD 6489          JSR     CHADRS      REGISTER EXIST?
00449 6347 2B D1            BMI     RNERR       REG NOT EXIST, ERROR
00450 6349 86 02            LDA A   #2          SET TO DATA TYPE
00451 634B 97 8C    FINRCD  STA A   W+4         STORE NEW FILE TYPE
00452 634D DE 20            LDX     TP7
00453 634F DF 14            STX     TP1         STARTING ADDRESS INTO TP1
00454 6351 BD 6551          JSR     FIND        FIND FILE
00455 6354 BD 65AA          JSR     CHECK       CHECK IF RIGHT ONE
00456 6357 DE 24            LDX     AFSZ
00457 6359 26 03            BNE     NTMRKR      MARKER FILE?
00458 635B 86 11            LDA A   #17         YES, WRONG FILE TYPE
00459 635D 39               RTS
00460 635E DE 1C    NTMRKR  LDX     TP5
00461 6360 DF 26            STX     CFSZ        SET NEW CURR FILE SIZE
00462 6362 96 8C            LDA A   W+4
00463 6364 97 23            STA A   TYPE        SET NEW FILE TYPE
00464 6366 96 24            LDA A   AFSZ
00465 6368 D6 25            LDA B   AFSZ+1
00466 636A D0 27            SUB B   CFSZ+1
00467 636C 92 26            SBC A   CFSZ        ABS FILE SIZE-CURR FILE SIZE
00468 636E 2A 03            BPL     FLRGE       FILE LARGE ENOUGH?
00469 6370 86 0F            LDA A   #15         FILE TOO SMALL
00470 6372 39               RTS
00471 6373 BD 6668  FLRGE   JSR     PART        WRITE PARTIAL PREAMBLE
00472 6376 BD 667F          JSR     RCDHD       RECORD THE HEADING
00473 6379 7F 0022          CLR     CHKSM       CLEAR CHECKSUM
```

```
00474 637C DE 14          LDX     TP1       STARTING ADDRESS INTO INDEX
00475 637E A6 00    RCDMR LDA A   X         LOAD INFO OUT OF CORE
00476 6380 9B 22          ADD A   CHKSM     ADD IT TO THE CHECKSUM
00477 6382 97 22          STA A   CHKSM     RESTORE CHECKSUM
00478 6384 A6 00          LDA A   X         RELOAD BYTE
00479 6386 BD 663B        JSR     RECORD    RECORD BYTE
00480 6389 08             INX
00481 638A 9C 1A          CPX     TP4       DONE RECORDING BODY?
00482 638C 26 F0          BNE     RCDMR     NO, RECORD MORE
00483 638E 96 22          LDA A   CHKSM     LOAD CHECKSUM INTO ACCA
00484 6390 BD 663B        JSR     RECORD    RECORD CHECKSUM
00485 6393 4F             CLR A
00486 6394 39             RTS
00488                *
00489                *
00490                * THIS ROUTINE LOADS A FILE FROM THE TAPE
00491                * AND EITHER LOADS IT INTO MEMORY OR COMPARES
00492                * IT WITH WHAT IS IN MEMORY.  THERE ARE FOUR
00493                * INSTRUCTIONS HANDLED BY THIS ROUTINE:
00494                *   1) LOAD
00495                *   2) VERIFY
00496                *   3) FORMAT LOAD
00497                *   4) LOAD BINARY
00498                * ALL FOUR INSTRUCTIONS PERFORM THEIR OPERATION
00499                * ON THE FILE POINTED TO BY THE NUMBER IN THE X-REGISTER.
00500                * THE LOAD AND THE VERIFY INSTRUCTIONS REQUIRE
00501                * A STARTING ADDRESS OR STARTING REGISTER NUMBER
00502                * WHICH IS FOUND IN THE Y-REGISTER.  THE
00503                * FILE TYPE, WHICH IS DETERMINED WHILE THE TAPE IS RUNNING,
00504                * TELLS THE ROUTINE WHETHER IT IS A LOAD DATA
00505                * OR A LOAD PROGRAM (VERIFY DATA OR VERIFY PROGRAM).
00506                * THE FORMAT LOAD LOADS THE CONTENTS OF THE FILE INTO
00507                * MEMORY BEGINNING AT THE ADDRESS IMMEDIATELY
00508                * FOLLOWING THE FORMAT LOAD INSTRUCTION
00509                * AND, ONCE THE LOAD IS COMPLETE, EXECUTION OF THE NEW
00510                * CODE IS BEGUN.  THE LOAD BINARY INSTRUCTION LOADS
00511                * A FILE OFF THE TAPE AND STORES IT IN MEMORY AS A
00512                * SPECIAL PROGRAM.
00513                *
00514                *
00515 6395 4C       FLOAD  INC A            LOAD AND RUN ENTRY POINT
00516 6396 4C       LOBIN  INC A            LOAD BINARY ENTRY POINT
00517 6397 4C       LOAD   INC A            LOAD ENTRY POINT
00518 6398 97 88    VERIFY STA A  W         VERIFY ENTRY POINT
00519 639A BD 6168         JSR    SETUP
00520 639D BD 6551         JSR    FIND      FIND FILE
00521 63A0 BD 660E CHKHD   JSR    RDHD      BEGIN READING THE FILE
00522 63A3 BD 658E         JSR    RIGHT     CHECK IF IT IS RIGHT FILE
00523 63A6 27 05           BEQ    FILEOK    YES, CONTINUE
00524 63A8 BD 6566         JSR    RETRY     NO, FIND RIGHT FILE
00525 63AB 20 F3           BRA    CHKHD     CHECK HEADING AGAIN
00526 63AD DE 26    FILEOK LDX    CFSZ
00527 63AF 27 59           BEQ    TYPERR    WRONG FILE TYPE
00528 63B1 CE 0098         LDX    #YR       SET INDEX TO Y-REG ADDRESS
00529 63B4 D6 88           LDA B  W
00530 63B6 96 23           LDA A  TYPE
00531 63B8 81 01           CMP A  #1
00532 63BA 2E 47           BGT    DTFL      DATA FILE?
00533 63BC C1 02           CMP B  #2        LOAD BINARY?
00534 63BE 27 4A           BEQ    TYPERR    YES, WRONG FILE TYPE
00535 63C0 4D              TST A            PRGM FILE?
00536 63C1 27 07           BEQ    NTSCRD    YES, NOT SECURED
00537 63C3 5D              TST B            VERIFY?
00538 63C4 27 04           BEQ    NTSCRD    YES, DON'T SET SCRD PRGM BIT
00539 63C6 9A D5           ORA A  FLAG
00540 63C8 97 D5           STA A  FLAG      SET SECRD PRGM BIT
00541 63CA BD 61CC NTSCRD  JSR    TOBIN     CONVERT ITS STARTING ADDR TO BINARY
00542 63CD 7C 0020         INC    TP7       ADJUST ADDRESS TO SYS ADDR
00543 63D0 DE 20           LDX    TP7
00544 63D2 DF 1E           STX    TP6       STARTING ADDR INTO TP6
00545 63D4 8D 0A           BSR    ENDADD
00546 63D6 BD 6489         JSR    CHADRS    CHECK ADDRESS
00547 63D9 2B 14           BMI    ADROK     ADDRESS OUT OF RANGE?
00548 63DB 27 0F           BEQ    CHKO
00549 63DD 86 07    MEMOVF LDA A  #7        MEMORY OVERFLOW ERROR
00550 63DF 39              RTS
```

```
00551 63E0 DE 26    ENDADD LDX     CFSZ
00552 63E2 DF 20           STX     TP7       NO. OF BYTES INTO TP7
00553 63E4 BD 4848         JSR     ADD7      STARTING ADDR + NO. OF BYTES
00554 63E7 DE 20           LDX     TP7
00555 63E9 DF 1A           STX     TP4       STORE ENDING ADDR INTO TP4
00556 63EB 39              RTS
00557 63EC 5D       CHK0   TST B               EQUAL TO EOPM?
00558 63ED 26 EE           BNE     MEMOVF    NO. GREATER THAN: ERROR
00559 63EF DE 1E    ADROK  LDX     TP6       STARTING INTO INDEX
00560 63F1 D6 88           LDA B   W
00561 63F3 27 4E           BEQ     STRTLD    IF VERIFY, START VERIFYING
00562 63F5 BD 4DFF         JSR     $4DFF     CLEAR 1ST BYTE IF LOAD AT 2ND BYTE
00563 63F8 08              INX
00564 63F9 C1 03           CMP B   #3        LOAD & RUN?
00565 63FB 26 46           BNE     STRTLD    NO, START LOAD
00566 63FD 09              DEX
00567 63FE DF CA           STX     UIP       YES, SET UIP TO PRGM'S BEGINNING
00568 6400 08              INX               STARTING ADDR INTO INDEX AGAIN
00569 6401 20 40           BRA     STRTLD
00570 6403 81 02    DTFL   CMP A   #2        DATA FILE?
00571 6405 26 06           BNE     NTDTFL    NO. BINARY FILE
00572 6407 11              CBA               LOAD BINARY & DATA FILE?
00573 6408 2E 29           BGT     DTFL1     NO, OKAY
00574 640A 86 11    TYPERR LDA A   #17       YES, WRONG FILE TYPE
00575 640C 39              RTS
00576 640D 5D       NTDTFL TST B               VERIFY BINARY?
00577 640E 27 19           BEQ     SPEC      YES, SET UP STARTING AND ENDING ADDRS
00578 6410 C1 02           CMP B   #2        LOAD BINARY?
00579 6412 26 F6           BNE     TYPERR    NO, WRONG FILE TYPE
00580 6414 DE 28           LDX     SPADRS    LOAD STARTING ADDR OF SPGM
00581 6416 BD 71           BSR     CHADRS    SPEC PRGM TOO BIG?
00582 6418 2B C3           BMI     MEMOVF    YES, ERROR
00583 641A DF 54           STX     SPGM      RESET SPGM TO NEW ADDRESS
00584 641C 96 D5           LDA A   FLAG
00585 641E 84 FB           AND A   #$FB      CLEAR SECURED BINARY BIT
00586 6420 D6 23           LDA B   TYPE
00587 6422 C1 04           CMP B   #4        SECURED BINARY?
00588 6424 26 01           BNE     NOTSEC    NO, LOAD
00589 6426 1B              ABA               YES, SET SEC BIN BIT AGAIN
00590 6427 97 D5    NOTSEC STA A   FLAG      STORE NEW FLAG
00591 6429 DE 54    SPEC   LDX     SPGM
00592 642B DF 1E           STX     TP6       STARTING ADDR OF SPGM INTO TP6
00593 642D BD B1           BSR     ENDADD    FIND ENDING ADDRESS
00594 642F DE 54           LDX     SPGM      LOAD STARTING ADDRESS
00595 6431 20 10           BRA     STRTLD
00596 6433 BD 61CC  DTFL1  JSR     TOBIN     CONVERT STARTING REG NO TO BINARY
00597 6436 96 20           LDA A   TP7
00598 6438 27 03           BEQ     LT256     REG. NO.<256?
00599 643A 86 18    RNOERR LDA A   #24        ILLEGAL ADDRESS
00600 643C 39              RTS
00601 643D 8D 2F    LT256  BSR     ENDADR    FIND ENDING ADDRESS
00602 643F 8D 48           BSR     CHADRS    REGISTER EXIST?
00603 6441 2B F7           BMI     RNOERR    REG NOT EXIST, ERROR
00604 6443 7F 0022  STRTLD CLR     CHKSM     CLEAR CHECKSUM
00605 6446 BD 65F9         JSR     SRCH1     FIND END OF SECOND PREAMBLE
00606 6449 BD 65D1  RDBYTE JSR     READ      READ BYTE
00607 644C D6 88           LDA B   W
00608 644E 26 07           BNE     LDBYTE    VERIFY?
00609 6450 A1 00           CMP A   X         YES, COMPARE IT WITH BYTE IN CORE
00610 6452 27 05           BEQ     CMPOK     EQUAL, NO ERROR
00611 6454 86 10           LDA A   #16       VERIFY FAILED
00612 6456 39              RTS
00613 6457 A7 00    LDBYTE STA A   X         LOAD! STORE BYTE INTO CORE
00614 6459 9B 22    CMPOK  ADD A   CHKSM     ADD BYTE TO CHECKSUM
00615 6458 97 22           STA A   CHKSM     RESTORE CHECKSUM
00616 645D 08              INX               INCR ADDRESS
00617 645E 9C 1A           CPX     TP4
00618 6460 26 E7           BNE     RDBYTE    NOT FINISHED, READ ANOTHER BYTE
00619 6462 BD 65D1         JSR     READ      READ CHECKSUM
00620 6465 91 22           CMP A   CHKSM     CHECKSUMS EQUAL?
00621 6467 27 03           BEQ     LDCMPL    YES, LOAD OR VERIFY COMPLETE
00622 6469 86 0E           LDA A   #14       CHECKSUM ERROR
00623 646B 39              RTS
00624 646C 4F       LDCMPL CLR A
00625 646D 39              RTS
```

```
00627 646E 96 21   ENDADR LDA A  TP7S     LOAD REG NO.
00628 6470 BD 4D58        JSR    SADRS    FIND ITS SYSTEM ADDRESS
00629 6473 08             INX
00630 6474 08             INX
00631 6475 08             INX
00632 6476 08             INX
00633 6477 08             INX
00634 6478 08             INX
00635 6479 08             INX
00636 647A 08             INX
00637 647B DF 1A          STX    TP4      ENDING ADDRESS INTO TP4
00638 647D DF 20   STADR   STX    TP7      ENDING ADDRESS INTO TP7
00639 647F DE 26          LDX    CFSZ
00640 6481 DF 1E          STX    TP6      CURR FILE SIZE INTO TP6
00641 6483 BD 4840        JSR    SUB7     ENDING ADDR-CFSZ=STARTING ADDR
00642 6486 DE 20          LDX    TP7      SET INDEX TO STARTING ADDRESS
00643 6488 39             RTS
00645 6489 DF 14   CHADRS STX    TP1
00646 648B D6 15          LDA B  TP1S
00647 648D D0 0C          SUB B  EOPM+1
00648 648F 96 14          LDA A  TP1
00649 6491 92 08          SBC A  EOPM
00650 6493 39             RTS
00652              *
00653              *
00654              * THIS ROUTINE IDENTIFIES THE FILE POINTED
00655              * TO BY THE NUMBER IN THE X-REGISTER.
00656              * WHEN THIS ROUTINE IS FINISHED, THE STACK WILL
00657              * CONTAIN:
00658              *   FILE TYPE IN T
00659              *   CURRENT FILE SIZE IN Z
00660              *   ABSOLUTE FILE SIZE IN Y
00661              *   FILE NUMBER IN X
00662              * THIS INFORMATION WILL ALSO BE PRINTED
00663              * OUT ON THE PRINTER IF THE PRINT SWITCH IS ON.
00664              *
00665              *
00666 6494 BD 6168 IDENT  JSR    SETUP
00667 6497 BD 6551        JSR    FIND     FIND FILE
00668 649A BD 65AA        JSR    CHECK    CHECK IF IT IS RIGHT FILE
00669 649D 96 D5          LDA A  FLAG
00670 649F 84 9D          AND A  #$9D
00671 64A1 97 D5          STA A  FLAG     RESET CASSETTE OP FLAG
00672 64A3 CE 6518        LDX    #IDTBL   LOAD STARTING ADDR OF ID TABLE
00673 64A6 DF 88          STX    W        SAVE IT IN W
00674 64A8 8D 2D          BSR    MSSG
00675 64AA 96 D5          LDA A  FLAG
00676 64AC 2A 04          BPL    TKA      TRACK 8?
00677 64AE 86 2D          LDA A  #$2D
00678 64B0 97 5F          STA A  BUFF+7   YES. PUT MINUS SIGN ON FILE NO.
00679 64B2 BD 48BE TKA    JSR    BINBCD+2
00680 64B5 8D 47          BSR    EQL0     NO, PRINT FILE NO.
00681 64B7 8D 1E          BSR    MSSG
00682 64B9 7F 0022        CLR    CHKSM    CLEAR CHECKSUM WORD
00683 64BC DE 22          LDX    CHKSM    LOAD FILE TYPE
00684 64BE 8D 28          BSR    PRINFO   PRINT FILE TYPE
00685 64C0 8D 15          BSR    MSSG
00686 64C2 DE 26          LDX    CFSZ     LOAD CURRENT FILE SIZE
00687 64C4 8D 22          BSR    PRINFO   PRINT CURRENT FILE SIZE
00688 64C6 8D 0F          BSR    MSSG
00689 64C8 DE 24          LDX    AFSZ     LOAD ABSOLUTE FILE SIZE
00690 64CA 8D 1C          BSR    PRINFO   PRINT ABSOLUTE FILE SIZE
00691 64CC BD 5D75        JSR    BLANK
00692 64CF 8D 38          BSR    IDPRNT
00693 64D1 BD 57F1        JSR    ROLLU    PUT FILE NUMBER IN X
00694 64D4 7E 618E        JMP    TRNOFF
00695 64D7 BD 5D75 MSSG   JSR    BLANK    BLANK OUT BUFFER
00696 64DA CE 0058        LDX    #$58
00697 64DD DF 16          STX    BFPTR    SET BUFF PTR TO BEG OF BUFFER
00698 64DF DE 88          LDX    W        LOAD MESSAGE PTR
00699 64E1 4F             CLR A
00700 64E2 BD 57BD        JSR    LDMSG    LOAD MESSAGE
00701 64E5 DF 88          STX    W        STORE MESSAGE PTR
00702 64E7 39             RTS
00703 64E8 BD 48BC PRINFO JSR    BINBCD   CONVERT BINARY TO BCD
00704 64EB BD 55F4        JSR    STKUP+$5   STACK UP
```

```
00705 64EE BD 740A          JSR     XR0         ZERO OUT X-REGISTER
00706 64F1 DE 20            LDX     T2
00707 64F3 27 09            BEQ     EQL0
00708 64F5 DF 92            STX     XR+2        LOAD 4 BCD DIGITS INTO X
00709 64F7 86 03            LDA A   #3
00710 64F9 97 90            STA A   XR          SET EXPNT TO 3
00711 64FB BD 74D6          JSR     NOR         NORMALIZE THE NUMBER
00712 64FE 96 90    EQL0    LDA A   XR          LOAD NEW EXPNT
00713 6500 4C               INC A
00714 6501 48               ASL A
00715 6502 48               ASL A
00716 6503 CE 0060          LDX     #BUFF+8
00717 6506 BD 4EA4          JSR     DGTS        LOAD DIGITS INTO BUFFER
00718 6509 96 09    IDPRNT  LDA A   RSFLG
00719 650B 2B 0A            BMI     *+12        PROGRAM RUNNING?
00720 650D 48               ASL A
00721 650E 2A 04            BPL     *+6         PRINT IF RSFLG=00?
00722 6510 96 12            LDA A   SFLG
00723 6512 27 03            BEQ     *+5         NO PRINT IF RSFLG=01 AND SFLG=0
00724 6514 7E 602D          JMP     PRTDRV
00725 6517 39               RTS
00727 6518 26       IDTBL   FCB     $26         FILE:
00728 6519 29               FCB     $29
00729 651A 2C               FCB     $2C
00730 651B 25               FCB     $25
00731 651C B4               FCB     $B4         TYPE:
00732 651D 39               FCB     $39
00733 651E 30               FCB     $30
00734 651F 25               FCB     $25
00735 6520 B5               FCB     $B5         USED
00736 6521 33               FCB     $33
00737 6522 25               FCB     $25
00738 6523 24               FCB     $24
00739 6524 AD               FCB     $AD         MAX:
00740 6525 21               FCB     $21
00741 6526 38               FCB     $38
00743               *
00744               *
00745               * THIS ROUTINE CONVERTS THE FILE NUMBER TO
00746               * BINARY AND CHANGES THE CASSETTE TO THE
00747               * PROPER TRACK.
00748               *   1) POSITIVE FILE NO.=TRACK A
00749               *   2) NEGATIVE FILE NO.=TRACK B
00750               *
00751               *
00752 6527 CE 0090 FILENO LDX     #XR         SET INDEX TO X-REG ADDRESS
00753 652A BD 61CC        JSR     TOBIN       CONVERT ITS CONTENTS TO BINARY
00754 652D 96 20          LDA A   TP7
00755 652F 27 03          BEQ     FNOK        FILE NO. > 255?
00756 6531 7E 61D1        JMP     ERR6        YES, ILLEGAL ARGUMENT
00757 6534 D6 D5    FNOK  LDA B   FLAG
00758 6536 17             TBA
00759 6537 98 91          EOR A   XR+1        SAME TRACK?
00760 6539 2A 03          BPL     SAME        YES, DON'T SET TPOS=LOST
00761 653B 4F             CLR A.
00762 653C 97 D6          STA A   TPOS        YES, TPOS=LOST
00763 653E 96 91    SAME  LDA A   XR+1        FILE NO. POSITIVE?
00764 6540 2A 06          BPL     TRACKA      YES, TRACK A
00765 6542 86 12          LDA A   #$12        NO, TRACK B SELECT
00766 6544 CA 80          ORA B   #$80        SET TRACK B
00767 6546 20 04          BRA     TRACK
00768 6548 86 16    TRACKA LDA A  #$16        TRACK A SELECT COED
00769 654A C4 7F          AND B   #$7F        SET TRACK A
00770 654C D7 D5    TRACK STA B   FLAG        RESTORE FLAG
00771 654E 97 00          STA A   ADATA       CHANGE TRACKS
00772 6550 39             RTS
00774               *
00775               *
00776               * THIS SUBROUTINE PERFORMS A FAST FILE
00777               * SEARCH TO THE FILE POINTED TO BY
00778               * THE NUMBER IN THE X-REGISTER.
00779               *
00780               *
00781 6551 8D D4    FIND  BSR     FILENO
00782 6553 9F 7C          STS     STACK
00783 6555 96 D6          LDA A   TPOS
```

```
00784 6557 84 03          AND A  #3
00785 6559 26 0E          BNE    NTLOST   NO. BEGIN FILE SEARCH
00786 655B 86 17          LDA A  #$17     REVERSE FAST READ DATA=0
00787 655D BD 66CD        JSR    RETGP+4   FIND FIRST GAP IN REV DIR
00788 6560 BD 660E        JSR    RDHD     READ ITS HEADING
00789 6563 7C 002A        INC    KNWALL   SET KNOW ALL FLAG
00790 6566 BD 66D3 RETRY  JSR    STOP
00791 6569 96 D7   NTLOST LDA A  FILE
00792 656B 91 21          CMP A  TP7S     FILE=XFILE?
00793 656D 26 10          BNE    FNEX     NO. CHECK DIRECTION TO GO
00794 656F BD 66D3        JSR    STOP
00795 6572 96 2A          LDA A  KNWALL
00796 6574 27 03          BEQ    FAST     IF KNOW ALL ABOUT HEADING. RETURN SLO
00797 6576 BD 66C9        JSR    RETGP    RETURN TO GAP SLOW
00798 6579 96 D6   FAST   LDA A  TPOS
00799 657B 44             LSR A          TPOS=GAP?
00800 657C 24 0F          BCC    DCFN-3   NO. FAST REWIND TO THE GAP
00801 657E 39             RTS
00802 657F 7F 002A FNEX   CLR    KNWALL   CLEAR KNOW ALL FLAG
00803 6582 9F 7C          STS    STACK
00804 6584 91 21          CMP A  TP7S
00805 6586 23 11          BLS    FLTX     FILE < XFILE?
00806 6588 96 D6          LDA A  TPOS     NO. CHECK TAPE POSITION
00807 658A 44             LSR A          TPOS=GAP?
00808 658B 25 03          BCS    DCFN     YES. DON'T CHANGE FILE NO.
00809 658D 7C 00D7        INC    FILE     YES. INCR FILE NO.
00810 6590 86 17   DCFN   LDA A  #$17     REVERSE FAST READ DATA=0
00811 6592 8D 0E          BSR    COUNT
00812 6594 7A 00D7        DEC    FILE     DECR FILE NO.
00813 6597 20 D0          BRA    NTLOST
00814 6599 86 57   FLTX   LDA A  #$57     FORWARD FAST READ DATA=0
00815 659B 8D 05          BSR    COUNT
00816 659D 7C 00D7        INC    FILE     INCR FILE NO.
00817 65A0 20 C7          BRA    NTLOST
00818 65A2 97 00   COUNT  STA A  ADATA    START TAPE
00819 65A4 BD 669B        JSR    DSRCH    DATA SEARCH
00820 65A7 7E 66B1        JMP    GSRCH    GAP SEARCH
00822 65AA 7D 002A CHECK  TST    KNWALL   KNOW ALL FLAG SET?
00823 65AD 26 0E          BNE    FIN      YES. FINISHED
00824 65AF BD 660E        JSR    RDHD     NO. READ HEADING
00825 65B2 BD 66C9        JSR    RETGP    RETURN TO THE GAP
00826 65B5 8D 07          BSR    RIGHT    RIGHT FILE?
00827 65B7 27 04          BEQ    FIN      YES. RETURN
00828 65B9 8D C4          BSR    FNEX     NO. LOOK FOR RIGHT ONE
00829 65BB 20 ED          BRA    CHECK    CHECK NEW FILE
00830 65BD 39      FIN    RTS
00832 65BE 96 D7   RIGHT  LDA A  FILE     LOAD FILE NO.
00833 65C0 91 21          CMP A  TP7S     IS IT THE RIGHT ONE?
00834 65C2 26 04          BNE    NTRFL    NO. DECR FILE CHECK CTR
00835 65C4 5F             CLR B          SET EXIT FLAG
00836 65C5 D7 7B          STA B  FNDCMD   CLEAR FIND CMMD FLAG
00837 65C7 39      AGAIN  RTS
00838 65C8 7A 002D NTRFL  DEC    FLCTR    DECR FILE CHECK CTR
00839 65CB 26 FA          BNE    AGAIN    FAILED CHECK 5 TIMES?
00840 65CD 86 12          LDA A  #18      FILE NOT FOUND
00841 65CF 20 62          BRA    ERR
00843                *
00844                *
00845                * READ SERVICE ROUTINES
00846                *
00847                *
00848 65D1 96 01   READ   LDA A  ACTL     LOOK FOR END OF 9TH BIT
00849 65D3 2A FC          BPL    *-2      FOUND ITS END?
00850 65D5 96 00          LDA A  ADATA    YES. RESET TRANSITION BIT
00851 65D7 C6 FD          LDA B  #-3      YES. SET READ OFFSET
00852 65D9 86 01          LDA A  #1
00853 65DB 97 2C          STA A  MASK     SET MASK=1
00854 65DD 4F             CLR A
00855 65DE 36      TRANS  PSH A           SAVE BYTE BEING READ
00856 65DF CB 18          ADD B  #24      ADD THRESHOLD TO OFFSET (T=216 USEC)
00857 65E1 5A      TRANS1 DEC B           DECR TIMING CTR
00858 65E2 96 01          LDA A  ACTL     TRANSITION OCCURRED?
00859 65E4 2A FB          BPL    TRANS1   NO. CONTINUE IN LOOP
00860 65E6 32             PUL A          RELOAD BYTE BEING READ
00861 65E7 5D             TST B
```

```
00862 65E8 2C 02          BGE    READ0    CTR>=0, BIT=0
00863 65EA 9B 2C          ADD A  MASK     CTR<0, BIT=1
00864 65EC D6 00   READ0  LDA B  ADATA    RESET TRANSITION BIT
00865 65EE C6 FB          LDA B  #-5      SET READ OFFSET
00866 65F0 78 002C        ASL    MASK     SHIFT MASK TO NEXT BIT PSTN
00867 65F3 24 E9          BCC    TRANS    CARRY CLEAR, BYTE NOT FINISHED
00868 65F5 39             RTS
00870 65F6 BD 669B PAST   JSR    DSRCH    GET PAST SWITCH-TO-WRITE GLITCH
00871 65F9 D6 00   SRCH1  LDA B  ADATA    RESET TRANSITION BIT
00872 65FB D6 01          LDA B  ACTL     TRANSITION?
00873 65FD 2A FC          BPL    SRCH1+2
00874 65FF C6 16   NTONE  LDA B  #22      SET RED THRESHOLD
00875 6601 96 00          LDA A  ADATA    RESET TRANSITION BIT
00876 6603 5A      NOTRN  DEC B           DECR TIMING CTR
00877 6604 96 01          LDA A  ACTL     TRANSITION?
00878 6606 2A FB          BPL    NOTRN    NO, CONTINUE WAITING
00879 6608 5D             TST B           BIT=0?
00880 6609 2A F4          BPL    NTONE    YES, CONT LOOKING FOR 1ST ONE
00881 660B 96 00          LDA A  ADATA    RESET TRANSITION BIT
00882 660D 39             RTS
00884 660E 86 77   RDHD   LDA A  #$77     FORWARD SLOW READ DATA=0
00885 6610 97 00          STA A  ADATA    START TAPE
00886 6612 CE FFF9        LDX    #-7      SET INDEX TO HEADING TEMP OFFSET
00887 6615 BD 669B        JSR    DSRCH    DATA SEARCH
00888 6618 8D DC          BSR    PAST     FIND END OF 1ST PREAMBLE
00889 661A 8D B5          BSR    READ     READ FILE NO.
00890 661C 97 89          STA A  W+1      SAVE IT UNTIL HEADING IS READ CORRECT
00891 661E 97 22          STA A  CHKSM    INITIALIZE CHECKSUM
00892 6620 8D AF   RDRST  BSR    READ     READ NEXT BYTE OF HEADING
00893 6622 A7 2A          STA A  $2A,X    STORE IT IN TEMPS
00894 6624 9B 22          ADD A  CHKSM    ADD IT TO CHECKSUM
00895 6626 97 22          STA A  CHKSM    RESTORE CHECKSUM
00896 6628 08             INX
00897 6629 26 F5          BNE    RDRST    NO, READ ANOTHER BYTE
00898 662B 8D A4          BSR    READ     YES. READ CHECKSUM
00899 662D 91 22          CMP A  CHKSM    CHECKSUMS EQUAL?
00900 662F 27 05          BEQ    HDRED    YES, HEADING READ
00901 6631 86 0E          LDA A  #14      CHECKSUM ERROR
00902 6633 7E 619F ERR    JMP    ERRMSG
00903 6636 96 89   HDRED  LDA A  W+1
00904 6638 97 D7          STA A  FILE     STORE NEW FILE NO. IN FILE
00905 663A 39             RTS
00907                *
00908                *
00909                * RECORD SERVICE ROUTINES.
00910                *
00911                *
00912 663B C6 E2   RECORD LDA B  #-30
00913 663D 5C             INC B
00914 663E 26 FD          BNE    *-1      WAIT 180 USEC FOR 9TH BIT
00915 6640 D6 00          LDA B  ADATA
00916 6642 C8 80          EOR B  #$80
00917 6644 D7 00          STA B  ADATA    WRITR 9TH BIT
00918 6646 C6 FA          LDA B  #-6      SET WRITE OFFSET
00919 6648 7F 002C NO9TH  CLR    MASK
00920 664B 7C 002C        INC    MASK     SET MASK=1
00921 664E CB 12   BYTE   ADD B  #18      ADD ZERO COUNT (0=108 USECI
00922 6650 95 2C          BIT A  MASK     IS BIT A ONE OR A ZERO?
00923 6652 27 02          BEQ    BIT
00924 6654 CB 24          ADD B  #36      ADD 0-1 DIFFERENCE (1=324 USEC)
00925 6656 5A      BIT    DEC B           DECR TIMING CTR
00926 6657 26 FD          BNE    BIT      TIMING CTR=0?
00927 6659 D6 00          LDA B  ADATA
00928 665B C8 80          EOR B  #$80
00929 665D D7 00          STA B  ADATA    YES, CHANGE DATA OUT BIT
00930 665F 02             NOP
00931 6660 C6 FA          LDA B  #-6      SET WRITE OFFSET
00932 6662 78 002C        ASL    MASK     SHIFT MASK LEFT ONE
00933 6665 24 E7          BCC    BYTE     DONE WITH BYTE?
00934 6667 39             RTS             YES, RETURN
00936 6668 DE 54   PART   LDX    SPGM
00937 666A DF 28          STX    SPADRS   SAVE SPGM STARTING ADDRESS
00938 666C 86 77          LDA A  #$77     FORWARD SLOW READ DATA=0
00939 666E 97 00          STA A  ADATA    START TAPE
00940 6670 8D 29          BSR    DSRCH    DATA SEARCH
00941 6672 86 67          LDA A  #$67     FORWARD SLOW WRITE DATA=0
```

```
00942 6674 97 00          STA A  ADATA
00943 6676 4F       PREMBL CLR A            SET ACCA TO ALL ZEROES
00944 6677 8D C2          BSR    RECORD     WRITE 8 ZEROES
00945 6679 86 80          LDA A  #$80       LOAD 7 ZEROES AND A ONE
00946 667B C6 F5          LDA B  #-11       SET TIMING CTR
00947 667D 20 C9          BRA    NO9TH      RECORD RESET OF PREAMBLE
00949 667F 96 D7    RCDHD LDA A  FILE       LOAD FILE NO.
00950 6681 97 22          STA A  CHKSM      INITIALIZE CHECKSUM
00951 6683 8D B6          BSR    RECORD     RECORD FILE NO.
00952 6685 CE FFF9        LDX    #-7        SET INDEX TO HEADING TEMP OFFSET
00953 6688 A6 2A    MRHD  LDA A  $2A,X      LOAD OTHER BYTES IN HEADING
00954 668A 9B 22          ADD A  CHKSM      ADD IT TO CHECKSUM
00955 668C 97 22          STA A  CHKSM      RESTORE CHECKSUM
00956 668E A6 2A          LDA A  $2A,X      RELOAD HEADING BYTE
00957 6690 8D A9          BSR    RECORD     RECORD NEXT BYTE OF HEADING
00958 6692 08             INX               INCR HEADING TEMP PTR
00959 6693 26 F3          BNE    MRHD       NO. DO NEXT BYTE
00960 6695 96 22          LDA A  CHKSM      YES. LOAD CHECKSUM
00961 6697 8D A2          BSR    RECORD     RECORD CHECKSUM
00962 6699 20 DB          BRA    PREMBL     RECORD 2ND PREAMBLE
00964                     *
00965                     *
00966                     * SEARCH AND TAPE MOVEMENT ROUTINES.
00967                     *
00968                     *
00969 669B 86 06    DSRCH LDA A  #6         SET NO. OF TRANSITIONS CTR
00970 669D D6 00    BTOCC LDA B  ADATA      RESET TRANSITION BIT
00971 669F 0E       NOBIT CLI
00972 66A0 02             NOP
00973 66A1 0F             SEI
00974 66A2 D7 00          STA B  ADATA      RESTART TAPE IF INTERRUPTED
00975 66A4 7D 0001        TST    ACTL       TRANSITION?
00976 66A7 2A F6          BPL    NOBIT      NO,CHECK AGAIN
00977 66A9 4A             DEC A             YES. DECR NO. OF TRANS CTR
00978 66AA 26 F1          BNE    BTOCC      NO, LOOK FOR ANOTHER
00979 66AC 86 02          LDA A  #2
00980 66AE 97 D6          STA A  TPOS       TPOS =DATA
00981 66B0 39             RTS
00983 66B1 CE 0190  GSRCH LDX    #400       LOAD .35" @ 60 IPS
00984 66B4 96 00          LDA A  ADATA
00985 66B6 85 20          BIT A  #$20       SEARCH SPEED?
00986 66B8 27 03          BEQ    CNTLP      YES. GO TO LOOP
00987 66BA CE 1388        LDX    #5000      LOAD .75" @ 10 IPS
00988 66BD D6 01    CNTLP LDA B  ACTL       TRANSITION?
00989 66BF 2B F0          BMI    GSRCH      YES. START ALL OVER
00990 66C1 09             DEX               NO. DECR CTR
00991 66C2 26 F9          BNE    CNTLP      FOUND GAP? NO. CONTINUE LOOPING
00992 66C4 86 01    ISGAP LDA A  #1
00993 66C6 97 D6          STA A  TPOS       TPOS=GAP
00994 66C8 39             RTS
00996 66C9 8D 08    RETGP BSR    STOP
00997 66CB 86 37          LDA A  #$37       REVERSE SLOW READ DATA=0
00998 66CD 97 00          STA A  ADATA      START TAPE
00999 66CF 8D CA          BSR    DSRCH      DATA SEARCH
01000 66D1 8D DE          BSR    GSRCH      GAP SEARCH
01001 66D3 96 00    STOP  LDA A  ADATA
01002 66D5 84 F0          AND A  #$F0       STOP
01003 66D7 8A 10          ORA A  #$10       READ
01004 66D9 97 00          STA A  ADATA      STOP TAPE
01005 66DB CE 30D4        LDX    #12500     WAIT FOR TAPE TO STOP
01006 66DE 09       WAIT  DEX
01007 66DF 26 FD          BNE    WAIT       WAIT DONE?
01008 66E1 39             RTS
01010 66E2 5F       LOOK  CLR B             FIRST PASS
01011 66E3 8D 28    NTINVT BSR   VALID      LOOK FOR 2 1/2 FT OR HOLE
01012 66E5 27 23          BEQ    FNDVTP     2 1/2 FT. FOUND VALID TAPE
01013 66E7 96 78          LDA A  AT1        HOLE. CHECK DISTANCE
01014 66E9 81 03          CMP A  #3         <=1"?
01015 66EB 23 0A          BLS    BEG        YES. BEGINNING OF TAPE
01016 66ED 81 2F          CMP A  #$2F       <=1 1/2 FT?
01017 66EF 22 03          BHI    PASS2      NO. 2 FT
01018 66F1 5D       FOOT  TST B             FIRST PASS?
01019 66F2 26 08          BNE    END        NO. AT END OF TAPE
01020 66F4 5C       PASS2 INC B             YES. INCR PASS WORD TO 2ND PASS
01021 66F5 20 EC          BRA    NTINVT
01022 66F7 86 57    BEG   LDA A  #$57       FORWARD FAST READ DATA=0
```

```
01023 66F9 5F              CLR B           FIRST PASS
01024 66FA 20 02           BRA     KWE     KNOW WHICH END
01025 66FC 86 17   END     LDA A   #$17     REVERSE FAST READ DATA=0
01026 66FE 91 2E   KWE     CMP A   SAVE     SAME DIRECTION?
01027 6700 27 E1           BEQ     NTINVT  YES, CONT LOOKING FOR 2 1/2 FT
01028 6702 97 2E           STA A   SAVE     NO, SWITCH DIRECTIONS AND CONT LOOKIN
01029 6704 8D CD           BSR     STOP
01030 6706 8D 08           BSR     VALID    BACK TAPE ACROSS HOLE
01031 6708 26 D9           BNE     NTINVT  HOLE HIT, CONTINUE IN NOOP
01032 670A 86 04   FNDVTP  LDA A   #4
01033 670C 97 D6           STA A   TPOS     LOST BUT IN VALID TAPE
01034 670E 20 C3           BRA     STOP
01036 6710 8D 38   VALID   BSR     COT      CHECK IF CARTRIDGE IS OUT
01037 6712 96 02           LDA A   ADATA    RESET EOT & COT INTERRUPT BIT
01038 6714 CE 0000         LDX     #0
01039 6717 96 2E           LDA A   SAVE     LOAD TAPE STATUS
01040 6719 97 00           STA A   ADATA    START TAPE
01041 671B 08      NOHOLE  INX              INCREMENT DISTANCE CTR
01042 671C DF 78           STX     AT1
01043 671E 8C 4E20         CPX     #20000   2 1/2 FT?
01044 6721 27 19           BEQ     VLDTP    YES, CHECK IF IN VALID TAPE
01045 6723 96 03           LDA A   RCTL     NO, HOLE BEEN HIT?
01046 6725 2A F4           BPL     NOHOLE   NO, CONTINUE IN LOOP
01047 6727 CE 00C8         LDX     #200
01048 672A 8D B2           BSR     WAIT     WAIT UNTIL PAST .023" HOLE
01049 672C 37      LIGHT   PSH B            SAVE B ACCUM
01050 672D 86 13           LDA A   #19      END OF TAPE ERROR
01051 672F C6 15           LDA B   #$15
01052 6731 D7 00           STA B   ADATA
01053 6733 D6 04           LDA B   INPUT
01054 6735 C5 04           BIT B   #4       TAPE SPOOLED OFF?
01055 6737 27 1F           BEQ     OFF      YES, GIVE ERROR
01056 6739 33              PUL B            NO, RELOAD ACCB
01057 673A 96 03           LDA A   RCTL     HOLE HIT?
01058 673C 39      VLDTP   RTS
01060 673D 8D 6168 REWIND  JSR     SETUP
01061 6740 97 28           STA A   REWFLG   SET REWIND FLAG
01062 6742 9F 7C           STS     STACK
01063 6744 86 17           LDA A   #$17     REVERSE FAST READ DATA=0
01064 6746 0E              CLI
01065 6747 97 00           STA A   ADATA
01066 6749 20 FC           BRA     *-2
01067 674B 20 86   STOP1   BRA     STOP
01069 674D 86 15   COT     LDA A   #$15
01070 674F 97 00           STA A   ADATA    ENABLE COT CHECK
01071 6751 96 04           LDA A   INPUT
01072 6753 44              LSR A            CARTRIDGE OUT?
01073 6754 24 E6           BCC     VLDTP    NO, IT IS IN
01074 6756 86 14           LDA A   #20      CARTRIDGE OUT ERROR
01075 6758 7E 619F OFF     JMP     ERRMSG
01077                      *
01078                      *
01079                      * THIS ROUTINE IS THE COT AND EOT INTERRUPT
01080                      * ROUTINE.  IT HANDLES THE SITUATIONS WHERE THE
01081                      * CARTRIDGE IS REMOVED IN THE MIDDLE OF
01082                      * A CASSETTE ROUTINE AND WHERE THE TAPE HITS EITHER
01083                      * THE LOAD POINT OR EARLY WARNING HOLES
01084                      * ON THE TAPE.
01085                      *
01086                      *
01087 675B 4F      HOLE    CLR A            SO SEI-SEI WORKS!!!
01088 675C 0F              SEI
01089 675D 96 00           LDA A   ADATA
01090 675F 8A 37           ORA A   #$37     READ SLOW
01091 6761 16              TAB
01092 6762 97 00           STA A   ADATA    START TAPE
01093 6764 CE 61A8         LDX     #25000
01094 6767 8D 66DE         JSR     WAIT     WAIT 2"
01095 676A 8D C0           BSR     LIGHT    TAPE SPOOLED OFF?
01096 676C C4 DF           AND B   #$DF     FAST
01097 676E C8 40           EOR B   #$40     OPPOSITE DIRECTION
01098 6770 D7 2E           STA B   SAVE     SAVE TAPE STATUS
01099 6772 8D D7           BSR     STOP1    STOP TAPE
01100 6774 86 3C           LDA A   #$3C
01101 6776 97 03           STA A   RCTL     DISABLE EOT & COT INTERRUPT
01102 6778 8D 96           BSR     VALID    BACK TAPE ACROSS HOLE
```

```
01103 677A 26 0C          BNE    HIT       HIT HOLE?
01104 677C 8D CD          BSR    STOP1
01105 677E 4F             CLR A            NO, FALSE HOLE
01106 677F 97 D6          STA A  TPOS      TPOS=LOST
01107 6781 9E 8D    OVER  LDS    SAVSTK    RESET STACK PTR
01108 6783 DE 8A          LDX    SVINS     LOAD STARTING ADDR OF INSTR
01109 6785 4F             CLR A
01110 6786 6E 00          JMP    X         DO INSTR OVER AGAIN
01111 6788 9E 7C    HIT   LDS    STACK
01112 678A 96 2E          LDA A  SAVE      RELOAD TAPE STATUS
01113 678C 48             ASL A            FORWARD DIRECTION?
01114 678D 2A 3C          BPL    EOT       NO, END OF TAPE
01115 678F 96 D6          LDA A  TPOS
01116 6791 26 05          BNE    KNOWN     POSITION KNOWN?
01117 6793 8D 66E2        JSR    LOOK      WHICH HOLE?
01118 6796 20 E9          BRA    OVER      DO REWIND AGAIN
01119 6798 8D B1    KNOWN BSR    STOP1
01120 679A 4F             CLR A
01121 679B 97 D7          STA A  FILE      FILE=0
01122 679D 4C             INC A
01123 679E 97 D6          STA A  TPOS      TPOS=GAP
01124 67A0 96 02          LDA A  BDATA     RESET EOT & COT INTERRUPT BIT
01125 67A2 86 3D          LDA A  #$3D
01126 67A4 97 03          STA A  BCTL      ENABLE EOT & COT INTERRUPT
01127 67A6 96 7A          LDA A  MRKBOT
01128 67A8 26 1E          BNE    MARK1     MARK BEG OF TAPE?
01129 67AA 86 77          LDA A  #$77      FORWARD SLOW READ DATA=0
01130 67AC 97 00          STA A  ADATA
01131 67AE 8D 66D8        JSR    WAIT-3    MOVE TAPE 1"
01132 67B1 8D 98          BSR    STOP1
01133 67B3 96 2B          LDA A  REWFLG    CHECK REWIND FLAG
01134 67B5 26 0E          BNE    ISPWND    REWIND, YOU ARE DONE
01135 67B7 7E 6569        JMP    NTLOST    NOT REWIND, CONTINUE WITH INSTRUCTION
01136 67BA 96 D7    ERASE LDA A  FILE
01137 67BC 97 21          STA A  TP7S
01138 67BE 86 3D          LDA A  #$3D
01139 67C0 97 03          STA A  BCTL
01140 67C2 8D 6555        JSR    FIND+4    MOVE TAPE TO GAP OF MARKER FILE
01141 67C5 7E 618E  ISRWND JMP   TRNOFF
01142 67C8 7E 6251  MARK1  JMP   FILE0
01143 67CB 8D 66E2  EOT    JSR   LOOK      DETERMINE IF REALLY EW HOLE
01144 67CE 96 2E          LDA A  SAVE
01145 67D0 48             ASL A            FORWARD DIRECTION?
01146 67D1 2B AE          BMI    OVER      YES, BEG. OF TAPE; DO INSTR OVER
01147 67D3 86 12          LDA A  #18       FILE NOT FOUND
01148 67D5 D6 7B          LDA B  FNDCMD
01149 67D7 26 05          BNE    FNDERR    FIND?
01150 67D9 D6 7A          LDA B  MRKBOT    ERASE TO END OF TPAE?
01151 67DB 2B DD          BMI    ERASE
01152 67DD 4C             INC A            NO, END OF TAPE
01153 67DE 7E 619F  FNDERR JMP   ERRMSG
01155 7FFA                ORG    $7FFA
01156 7FFA 6758           FDB    HOLE
01157 7FFC 6758           FDB    HOLE
01159 7D4A                ORG    $7D4A
01160 7D4A 6398           FDB    VERIFY
01161 7D4C 6395           FDB    FLOAD
01162 7D4E 62AE           FDB    RCDATA
01163 7D50 673D           FDB    REWIND
01165 7D6A                ORG    $7D6A
01166 7D6A 62A2           FDB    PCPRGM
01167 7D6C 62A1           FDB    FRCRD
01168 7D6E 61DC           FDB    MARK
01169 7D70 6494           FDB    IDENT
01170 7D72 6397           FDB    LOAD
01173                     END
```

SYMBOL TABLE

```
ADATA  0000  ACTL   0001  BDATA  0002  BCTL   0003  INPUT  0004  IOIN   0005
ERROR  0006  TGL    0007  UFLG   0008  RSFLG  0009  EOM    000A  EOPM   000B
STKFLG 000D  RND    000E  DIGFLG 000F  W2     0010  W1     0011  SFLG   0012
DCNTR  0013  TP1    0014  TP1S   0015  TP2    0016  TP2S   0017  TP3    0018
TP3S   0019  TP4    001A  TP4S   001B  TP5    001C  TP5S   001D  TP6    001E
```

```
TP6S    001F  TP7     0020  TP7S    0021  T13     0022  T12     0023  T11     0024
T10     0025  T9      0026  T8      0027  T7      0028  T6      0029  T5      002A
T4      002B  T3      002C  T2      002D  T1      002E  ISTK    002F  ISTACK  0051
TA      0052  SPGM    0054  EXTRA   0056  BUFF    0058  REAL    0068  IMAG    0070
AT1     0078  AT2     0080  W       0088  XR      0090  YR      0098  ZR      00A0
TR      00AB  LSTX    00B0  BKWRT   00B8  BKKC    00BA  SOL7    00C6  UPP     00C8
UIP     00CA  SAVE    00EE  FLCTR   00ED  MASK    00EC  REWFLG  00EB  KNWALL  00EA
MRKBOT  007A  FNDCMD  007B  STACK   007C  SVINS   008A  SAVSTK  008D  SPADRS  0028
CFSZ    0026  AFSZ    0024  TYPE    0023  CHKSM   0022  REPTR   0016  FLAG    00D5
TPOS    00D6  FILE    00D7  SADRS   4D58  TSFR    49CE  ADD7    4848  SUB7    4840
FRMT    5CAB  BLANK   5D75  LDMSG   578D  BINBCD  48BC  STKUP   55EF  XR0     740A
NOR     7406  OGTS    4EA4  PRTDRV  602D  ROLLU   57F1  SETUP   6158  CRTIN   6176
ERRMSG  619F  STERR   6189  TRNOFF  61BE  TOBIN   61CC  ERR6    61D1  NOOK    61D5
MARK    61DC  MRKOK   61E4  FIND1   6213  BEGIN   6219  ADDSLK  6233  NOMORE  623B
NOSLK   623F  START   624C  FILE0   6251  NXTFL   6255  MRKPRE  6269  MRKMR   6274
FIN1    6280  FIN2    6296  FRCRD   62A1  RCPRGM  62A2  STBIN   62A0  RCDATA  62AE
RCDOK   62B9  NTSRN   62DD  CONT    62F6  CHKEND  6301  ENDFND  630A  RDATA   6316
RNERR   631A  RNOK    631D  ILLARG  632E  RGNT0   6331  FINRCD  6348  NTMRKR  635E
FLRGE   6373  RCDMR   637E  FLOAD   6395  LDBIN   6396  LOAD    6397  VERIFY  6396
CHKHD   63A0  FILEOK  63AD  NTSCRD  63CA  MEMOVF  63DD  ENDADD  63E0  CHK0    63EC
ADROK   63EF  DTFL    6403  TYPERR  640A  NTDTFL  640D  NOTSEC  6427  SPEC    6429
DTFL1   6433  RNOERR  643A  LT256   643D  STRTLD  6443  RDBYTE  6449  LDBYTE  6457
CMPOK   6459  LOCMPL  646C  ENDADR  646E  STADR   647D  CHADRS  6489  IDENT   6494
TKA     64B2  MSSG    64D7  PRINFO  64E8  EQL0    64FE  IDPRNT  6509  IDTBL   6518
FILEND  6527  FNOK    6534  SAME    653E  TRACKA  6548  TRACK   654C  FIND    6551
RETRY   6566  NTLOST  6569  FAST    6579  FNEX    657F  DCFN    6590  FLTX    6599
COUNT   65A2  CHECK   65AA  FIN     65BD  RIGHT   65BE  AGAIN   65C7  NTRFL   65C8
READ    6501  TRANS   65DE  TRANS1  65E1  READ0   65EC  PAST    65F6  SRCH1   65F9
NTONE   65FF  NOTRN   6603  RDHD    660E  RDRST   6620  ERR     6633  HDRED   6636
RECORD  663B  NO9TH   6648  BYTE    664E  BIT     6655  PART    6668  PREMBL  6676
RCDHD   667F  MRHD    6688  DSRCH   6698  RTOCC   669D  NOBIT   669F  GSRCH   66B1
CNTLP   66BD  ISGAP   66C4  RETGP   66C9  STOP    66D3  WAIT    66DE  LOOK    66E2
NTINVT  66E3  FOOT    66F1  PASS2   66F4  BEG     66F7  END     66FC  KWE     66FE
ENDVTP  670A  VALID   6710  NOHOLE  671B  LIGHT   672C  VLDTP   673C  REWIND  6730
STOP1   6749  COT     674D  OFF     6758  HOLE    6758  OVER    6781  HIT     6788
KNOWN   6798  ERASE   67BA  ISPWND  67C5  MARK1   67C8  EOT     67CB  ENDERR  67DE
****ERROR 201
 216   NAM  EXIO
****ERROR 201
 392   NAM  SIGMA
****ERROR 201
 582  NAM LINK
00001                        NAM    CJRPG
00002                *
00003                *THIS FILE DEFINES THE BASE PAGE READ/WRITE
00004                *ALLOCATION FOR CJ. ADDRESSES ARE HEX 0 THRU FF INCLUSIVE.
00005                *
00006 0000 0001      ADATA  RMB    1          PERIPHERAL ACCESS REGISTER
00007 0001 0001      ACTL   RMB    1          PERIPHERAL CONTROL REGISTER
00008 0002 0001      BDATA  RMB    1          PERIPHERAL ACCESS REGISTER
00009 0003 0001      BCTL   RMB    1          PERIPHERAL CONTROL REGISTER
00010 0004 0001      INPUT  RMB    1          MAINFRAME INPUT PORT
00011 0005 0001      IOIN   RMB    1          I/O ROM INPUT PORT
00012                *
00013                *FOLLOWING IS THE STATUS AREA. IT IS PERMANENT
00014                *READ/WRITE USED TO RETAIN THE STATUS OF THE MACHINE
00015                *
00016 0006 0001      ERROR  RMB    1          ERROR WORD (NON-ZERO IMPLIES ERROR)
00017 0007 0001      TGL    RMB    1          TOGGLE SWITCHES ARE RETAINED HERE
00018 0008 0001      UFLG   RMB    1          USER FLAGS ARE KEPT HERE
00019 0009 0001      RSFLG  RMB    1          USER PRGM RUN/STOP FLAG
00020 000A 0001      EOM    RMB    1          END OF USER R/W WORD (PAGE ADRS ONLY)
00021 000B 0002      EOPM   RMB    2          END OF PROGRAM MEMORY (16 BIT PNTR)
00022 000D 0001      STKFLG RMB    1          AUTO-STACK FLAG (NEG IMPLIES LIFT ENA
00023 000E 0001      RND    RMB    1          DISPLAY ROUND SETTING
00024 000F 0001      DIGFLG RMB    1          DIGIT ENTRY FLAG
00025 0010 0001      W2     RMB    1          BCD ACC. FOR COMPILER
00026 0011 0001      W1     RMB    1          BCD ACC. FOR COMPILER
00027 0012 0001      SFLG   RMB    1          STEP FLAG FOR SINGLE STEP MODE
00028 0013 0001      DCNTR  RMB    1          DIGIT COUNTER FOR COMPILER
00029                *
00030                *FOLLOWING IS THE 2-BYTE TEMPORARY AREA
00031                *USED FOR TEMPORARY STORAGE OF POINTERS, ETC.
00032                *
00033 0014 0001      TP1    RMB    1
00034 0015 0001      TP1S   RMB    1
```

```
00035 0016 0001     TP2     RMB     1
00036 0017 0001     TP2S    RMB     1
00037 0018 0001     TP3     RMB     1
00038 0019 0001     TP3S    RMB     1
00039 001A 0001     TP4     RMB     1
00040 001B 0001     TP4S    RMB     1
00041 001C 0001     TP5     RMB     1
00042 001D 0001     TP5S    RMB     1
00043 001E 0001     TP6     RMB     1
00044 001F 0001     TP6S    RMB     1
00045 0020 0001     TP7     RMB     1
00046 0021 0001     TP7S    RMB     1
00047               *
00048               *FOLLOWING IS THE 1-BYTE TEMPORARY AREA
00049               *USED FOR TEMPORARY STORAGE OF FLAGS, ETC.
00050               *
00051 0022 0001     T13     RMB     1
00052 0023 0001     T12     RMB     1
00053 0024 0001     T11     RMB     1
00054 0025 0001     T10     RMB     1
00055 0026 0001     T9      RMB     1
00056 0027 0001     T8      RMB     1
00057 0028 0001     T7      RMB     1
00058 0029 0001     T6      RMB     1
00059 002A 0001     T5      RMB     1
00060 002B 0001     T4      RMB     1
00061 002C 0001     T3      RMB     1
00062 002D 0001     T2      RMB     1
00063 002E 0001     T1      RMB     1
00064               *
00065               *FOLLOWING IS THE INTERNAL SYSTEM STACK. THE STACK
00066               *ALLOWS FOR 7 USER SUBROUTINES, 7 SYSTEM
00067               *SUBROUTINES, AND ONE INTERRUPT.
00068               *
00069 002F 0022     ISTK    RMB     34
00070 0051 0001     ISTACK  RMB     1       (STARTS HERE; BUILDS DOWN)
00071               *
00072               *FOLLOWING ARE 3 PERMANENT POINTERS
00073               *
00074 0052 0002     TA      RMB     2       INSERT START PNTR/FLAG
00075 0054 0002     SPGM    RMB     2       SPECIAL PRGM POINTER
00076 0056 0002     EXTRA   RMB     2       EXTRA POINTER FOR FUTURE ROMS
00077               *
00078               *FOLLOWING IS THE PRINTER/DISPLAY BUFFER.
00079               *(ALSO USED AS DIGIT STACK BY CORDIC)
00080               *
00081 0058 0010     BUFF    RMB     16
00082               *
00083               *FOLLOWING ARE 4 ARITHMETIC TEMPORARYS
00084               *(ALSO USED BY PRINTER DRIVER FOR DOTS)
00085               *
00086 0068 0008     REAL    RMB     8
00087 0070 0008     IMAG    RMB     8
00088 0078 0008     AT1     RMB     8
00089 0080 0008     AT2     RMB     8
00090               *
00091               *FOLLOWING IS THE WORKING REGISTER FOR THE
00092               *FLOATING POINT MATH SUBROUTINES
00093               *
00094 0088 0008     W       RMB     8
00095               *
00096               *FOLLOWING ARE THE USER STACK REGISTERS
00097               *
00098 0090 0008     XR      RMB     8
00099 0098 0008     YR      RMB     8
00100 00A0 0008     ZR      RMB     8
00101 00A8 0008     TR      RMB     8
00102               *
00103               *FOLLOWING IS THE LAST X REGISTER
00104               *
00105 00B0 0008     LSTX    RMB     8
00106               *
00107               *FOLLOWING IS THE BUFFERED KEYBOARD KEYCODE
00108               *STORAGE AREA. IT WILL BUFFER 12 KEYS
00109               *LOCATION BKWRT MUST ALWAYS BE ZERO. IF BKWRT+1 IS
00110               *NON-ZERO, THE NEXT KEYCODE WILL ALWAYS
```

```
00111                    *BE IN LOCATION BKKC
00112                    *
00113 00B8 0002    BKWRT  RMB    2          BUFFERED KEYBOARD WRITE POINTER
00114 00BA 000C    BKKC   RMB    12         NEXT AVAILABLE KEYCODE
00115                    *
00116                    *FOLLOWING ARE SOME MORE PERMANENT STORAGE
00117                    *LOCATIONS AND POINTERS.
00118                    *
00119 00C6 0002    SOL7   RMB    2          SINGLE OPERATION LOCATION
00120 00C8 0002    UPP    RMB    2          USER PROGRAM POINTER
00121 00CA 0002    UIP    RMB    2          USER INSTRUCTION POINTER
00122 00CC 0001    ALPHA  RMB    1          MEMORY ALPHA FLAG
00123 00CD 0003    IO1    RMB    3          I/O CHANNEL 1 POINTER/FLAG
00124 00D0 0003    IO2    RMB    3          I/O CHANNEL 2 POINTER/FLAG
00125 00D3 0002    IT7    RMB    2          SYNTAX TABLE POINTER
00126 00D5 0001    FLAG   RMB    1          CASSETTE OPERATION FLAG
00127 00D6 0001    TPOS   RMB    1          CASSETTE TAPE POSITION INDICATOR
00128 00D7 0001    FILE   RMB    1          CASSETTE FILE INDICATOR
00129                    *
00130                    *FOLLOWING ARE 5 (A-E) OF THE 10 (A-J) USER
00131                    *ALPHA REGISTERS. THE OTHER 5 (F-J) ARE TAKEN
00132                    *OUT OF USER R/W ON POWER ON.
00133                    *
00134 00D8 0008    AR     RMB    8
00135 00E0 0008    BR     RMB    8
00136 00E8 0008    CR     RMB    8
00137 00F0 0008    DR     RMB    8
00138 00F8 0008    ER     RMB    8
00139                    *
00140                    *THIS CONCLUDES BASE PAGE READ/WRITE
00141                    *USER PROGRAM MEMORY FOLLOWS IMMEDIATELY HEREAFTER
00142                    *
00143                    *FOLLOWING ARE SOME EQUATES DEFINING STARTING
00144                    *ADDRESSES OF COMMONLY USED SYSTEM SUBROUTINES
00145                    *
00146      00BA    SDBB   EQU    @272
00147      7E00    MT     EQU    @77000
00148      003D    TERMN7 EQU    @75
00149      0040    IMED   EQU    @100
00150      00C0    PARCD  EQU    @300
00151      0080    PAREX  EQU    @200
00152      0000    NTBL   EQU    @0
00153      5EC0    DOTS   EQU    $5EC0
00154      602D    PRTDRV EQU    $602D
00155      5CA8    FRMT   EQU    $5CA8
00156      5D75    BLANK  EQU    $5D75
00157      578D    LDMSG  EQU    $578D
00158      5582    ROLLD  EQU    $5582
00159      57F1    ROLLU  EQU    $57F1
00160      55DA    PSD    EQU    $55DA
00161      55E9    TXL    EQU    $55E9
00162      55EF    STKUP  EQU    $55EF
00163      7498    MAD    EQU    $7498
00164      74AA    CMP    EQU    $74AA
00165      74D6    NOR    EQU    $74D6
00166      7424    TXW    EQU    $7424
00167      743B    TXXR   EQU    $743B
00168      7452    EXXR   EQU    $7452
00169      753B    ARSR   EQU    $753B
00170      75B6    OVUNF  EQU    $75B6
00171      75DD    OVERF  EQU    $75DD
00172      740A    XR0    EQU    $740A
00173      75C8    XRNINE EQU    $75C8
00174      75F1    UNDRF  EQU    $75F1
00175      76B9    IMULT  EQU    $76B9
00176      7669    ODG    EQU    $7669
00177      75FC    FPA    EQU    $75FC
00178      75F6    FPS    EQU    $75F6
00179      7735    FPM    EQU    $7735
00180      7793    FPD    EQU    $7793
00181      763D    FPAEX  EQU    $763D
00182      7780    FPMEX  EQU    $7780
00183      7521    LSHIFT EQU    $7521
00184      7489    ZEROX  EQU    $7489
00185      7416    XZEROO EQU    $7416
```

```
00186        7417    XZERO2 EQU    $7417
00187        73E6    RECIP  EQU    $73E6
00188        73F3    TXRX   EQU    $73F3
00189        6800    CONST  EQU    $6800
00190        6898    FPDBRC EQU    $6898
00191        68A9    TAN    EQU    $68A9
00192        69C3    ATN    EQU    $69C3
00193        6A46    DSZERO EQU    $6A46
00194        6A58    NTLN   EQU    $6A58
00195        6AC9    EXPN   EQU    $6AC9
00196        6B94    SIN    EQU    $6B94
00197        6B9A    COS    EQU    $6B9A
00198        6BF2    ASIN   EQU    $6BF2
00199        6BF7    ACOS   EQU    $6BF7
00200        6C5D    PH1    EQU    $6C5D
00201        6C8D    PH2    EQU    $6C8D
00202        6D34    PH3    EQU    $6D34
00203        6DD0    PH4    EQU    $6DD0
00204        6E47    LSFTB  EQU    $6E47
00205        6E65    SQRT   EQU    $6E65
00206        6F2C    MADB   EQU    $6F2C
00207        53F4    CMPB   EQU    $53E4
00208        6F52    IOUPX  EQU    $6F52
00209        6FA7    LOG10  EQU    $6FA7
00210        6FE9    YUPX   EQU    $6FE9
00211        7328    RTOP   EQU    $7328
00212        73B6    PTOR   EQU    $73B6
00213                       OPT    LIST,MEM
00216                       NAM    EXIO
00217 5300                  ORG    $5300
00218                *
00219                *      ROMID
00220                *
00221                *      I/O CODE IS PASSES IN ACCA.  ROUTINE RETURNS
00222                *      ROM ADDRESS IN IX(ALSO IN T11) OR ERROR
00223                *      CODE IN ACCB (5=NO I/O DEVICE).
00224                *
00225 5300 CE 1000 ROMID  LDX    #$1000    FIRST ROM ADDRESS
00226 5303 DF 24          STX    T11       SET IT
00227 5305 A1 00    RM1   CMP A  0,X       CHECK PRIMARY CODE
00228 5307 27 20          BEQ    RM3       FOUND A MATCH
00229 5309 C6 08          LDA B  #8
00230 530B DB 24          ADD B  T11       BUMP ADDRESS TO NEXT ROM
00231 530D D7 24          STA B  T11
00232 530F DE 24          LDX    T11       LOAD NEXT ROM ADDRESS
00233 5311 8C 4800        CPX    #$4800    LAST ONE CHECKED?
00234 5314 26 EF          BNE    RM1       NOT YET
00235 5316 D6 24    RM2   LDA B  T11
00236 5318 C0 08          SUB B  #8        BACK UP TO PREVIOUS ROM
00237 531A D7 24          STA B  T11
00238 531C DE 24          LDX    T11       LOAD ROM POINTER
00239 531E A1 01          CMP A  1,X       CHECK SECONDARY CODE
00240 5320 27 07          BEQ    RM3       FOUND A MATCH
00241 5322 8C 1000        CPX    #$1000    BACK TO ROM ONE?
00242 5325 26 EF          BNE    RM2       NOT YET
00243 5327 C6 05          LDA B  #5        LOAD ERROR CODE
00244 5329 39       RM3   RTS              RETURN
00246                *
00247                *      IOFMT
00248                *
00249                *      MAINFRAME I/O CONTROL
00250                *
00251 532A DE CA    IOFMT LDX    UIP       LOAD INSTR POINTER
00252 532C E6 01          LDA B  1,X       SECOND BYTE OF INSTR
00253 532E C4 70          AND B  #$70      MASK CALL CODE NUMBER
00254 5330 17             TBA
00255 5331 27 1E          BEQ    PNPRT     RUNNING PRINT
00256 5333 8D CB          BSR    ROMID     LOCATE ROM ADDRESS
00257 5335 26 46          BNE    SPS1      ROM NOT FOUND
00258 5337 DE CA    IOF1  LDX    UIP       INSTR POINTER
00259 5339 E6 01          LDA B  1,X       SECOND BYTE
00260 533B C4 0F          AND B  #$F       MASK LETTER CODE
00261 533D 58             ASL B            TIMES TWO
00262 533E D7 25          STA B  T10       SET LOWER HALF OF POINTER
00263 5340 DE 24          LDX    T11       LOAD POINTER
```

```
00264 5342 EE 05          LDX    5,X       LOAD ADDRESS OF ROUTINE
00265 5344 4F             CLR A
00266 5345 AD 00          JSR    0,X       MAKE I/O ROM CALL
00267 5347 DE CA   IOF1A  LDX    UIP
00268 5349 96 06          LDA A  ERROR     LOAD ERROR WORD
00269 534B 26 01          BNE    IOF2      WAS AN ERROR
00270 534D 08             INX              NO ERROR, BUMP INSTR POINTER
00271 534E DF CA   IOF2   STX    UIP       SET INSTR POINTER
00272 5350 39             RTS              RETURN
00274 5351 7E 521B RNPRT  JMP    $521B     CALL PRINT
00276      62AD    STBIN  EQU    $62AD
00277      6396    LDBIN  EQU    $6396
00279             *
00280             *    SPST - SPSTS
00281             *
00282             *    SPECIAL PROGRAM STORE TO TAPE CARTRIGE.
00283             *    SPSTS STORES IT AS A SECURED PROGARM
00284             *    USER PARAMETERS:
00285             *      XR - FILE #
00286             *
00287 5354 86 04  SPSTS  LDA A  #4        SECURE BIT
00288 5356 9A D5  SPST   ORA A  FLAG
00289 5358 8D 26          BSR    LIMIT     FIND LOWER LIMIT
00290 535A 9C 54          CPX    SPGM      NULL PROGRAM?
00291 535C 27 1C          BEQ    SPS0A     YES
00292 535E D6 D5          LDA B  FLAG
00293 5360 C5 04          BIT B  #4        SECURED SPGM?
00294 5362 26 17          BNE    SPS1-2    YES
00295 5364 97 D5          STA A  FLAG      SET NEW FLAG
00296 5366 09       SPS0  DEX
00297 5367 A6 00          LDA A  0,X       FIND END OF SPGM
00298 5369 27 FB          BEQ    SPS0
00299 536B 08             INX
00300 536C DF 1A          STX    TP4       SET END+1 FOR CASSETTE
00301 536E 4F             CLR A
00302 536F CE 62AD        LDX    #STBIN
00303 5372 AD 00          JSR    0,X       CALL ATAPE
00304 5374 96 D5          LDA A  FLAG
00305 5376 84 FB          AND A  #$FB      CLEAR SECURE BIT
00306 5378 97 D5          STA A  FLAG
00307 537A 39       SPS0A RTS
00308 537B C6 16          LDA B  #22       SECURE MEMORY
00309 537D D7 06    SPS1  STA B  ERROR     SET ERROR
00310 537F 39             RTS
00312             *
00313             *    FINDS THE UPPER LIMIT OF MEMORY AND RETURNS
00314             *    THE VALUE IN IX.
00315             *
00316 5380 DE 56  LIMIT  LDX    EXTRA     SPECIAL ROM THERE?
00317 5382 26 0A         BNE    LIM1      YES
00318 5384 DE CD         LDX    I01       LOAD SLOT A
00319 5386 D6 CE         LDA B  I01+1     LOAD TEST PART
00320 5388 D1 D1         CMP B  I02+1     I01<=I02?
00321 538A 23 02         BLS    LIM1      YES, KEEP I01
00322 538C DE D0         LDX    I02       NO, LOAD I02
00323 538E 39       LIM1 RTS
00325             *
00326             *    ERASES PREVIOUS SPGM AND DATA REGISTERS.
00327             *
00328 538F 8D EF  KILL   BSR    LIMIT     MEMORY LIMIT
00329 5391 DF 54         STX    SPGM
00330 5393 9C 08  KIL1   CPX    EOPM      COLLAPSED ALL?
00331 5395 27 F7         BEQ    LIM1      YES
00332 5397 09            DEX
00333 5398 6F 00         CLR    0,X
00334 539A 20 F7         BRA    KIL1
00336             *
00337             *    SPLD
00338             *
00339             *    LOADS A SPECIAL PROGRAM FROM CARTRIDGE.
00340             *    USER PARAMETERS:
00341             *      XR - FILE #
00342             *
00343 539C 8D F1  SPLD   BSR    KILL'     ERASE MEMORY
00344 539E CE 6396       LDX    #LDBIN
```

```
00345 53A1 AD 00          JSR     0,X       CALL TAPE
00346 53A3 96 06          LDA A   ERROR     CASSETT ERROR?
00347 53A5 27 02          BEQ     SPL1      NO
00348 53A7 8D E6          BSR     KILL
00349 53A9 DE 54    SPL1  LDX     SPGM
00350 53AB 0F 0B          STX     EOPM      SET #REGS TO ZERO
00351 53AD 39             RTS
00353               *
00354               *     SPCAL
00355               *
00356               *     CALLS A SPECIAL PROGRAM
00357               *
00358 53AE 8D 00    SPCAL BSR     LIMIT     FIND LOWER LIMIT
00359 53B0 C6 18          LDA B   #24
00360 53B2 9C 54          CPX     SPGM      NULL PROGRAM?
00361 53B4 27 C7          BEQ     SPS1      YES
00362 53B6 DE 54          LDX     SPGM      LOAD STARTING ADDRESS
00363 53B8 6E 00          JMP     0,X       MAKE CALL
00365               *
00366               *     SPTAP
00367               *
00368               *     LOADS A SPGM THROUGH GIO ROM FROM PAPER TAPE.
00369               *     TAPE FORMAT:  ADDRESS,SIZE,DATA,CHECKSUM,SIZE,DATA.....
00370               *
00371 53BA 8D D3    SPTAP BSR     KILL      ERASE DATA REGS
00372 53BC C6 20          LDA B   #$20
00373 53BE F0 3000        SUB B   $3000     GIO ROM PRESENT?
00374 53C1 26 07          BNE     SPT1      NO
00375 53C3 D7 21          STA B   TP7+1     ZERO CHECKSUM
00376 53C5 8D 37FD        JSR     $37FD     GIO ROM JUMP
00377 53C8 20 D9          BRA     SPL1-6    CHECK FOR ERROR
00378 53CA C6 05    SPT1  LDA B   #5        SET ERROR
00379 53CC D7 06          STA B   ERROR
00380 53CE 20 D9          BRA     SPL1
00382 7D40                ORG     $7D40
00383 7D40 539C           FDB     SPLD
00384 7D42 53AE           FDB     SPCAL
00385 7D44 5356           FDB     SPST
00386 7D46 5354           FDB     SPSTS
00387 7D48 53BA           FDB     SPTAP
00388 7D7C                ORG     $7D7C
00389 7D7C 532A           FDB     IOFMT
00392                     NAM     SIGMA
00393 7250                ORG     $7250
00394       4CB3    USMEM EQU     $4CB3
00395       55AC    XEY   EQU     $55AC
00396               *
00397               *     ACCUMULATE ROUTINES
00398               *
00399               *     THIS ROUTINE IMPLEMENTS THE DOUBLE VARRIABLE
00400               *     SUMMATION OF 'X' AND 'Y' INTO REGS 'I' AND 'J'.
00401               *     USER PARAMETERS:
00402               *       XR - NUMBER
00403               *       YR - NUMBER
00404               *     RETURNED VALUES:
00405               *       I  - 'I+XR' OR 'I-XR'
00406               *       J  - 'J+XR' OR 'J-XR'
00407               *
00408 7250 4C       ACCM  INC A
00409 7251 4C       ACCP  INC A
00410 7252 97 2E          STA A  T1         SET '+' OR '-' FLAG
00411 7254 5F             CLR B
00412 7255 D7 2D          STA B  T2
00413 7257 D7 2C          STA B  T3         SET OTHER MACRO FLAGS
00414 7259 86 08          LDA A  #8         'I' REG. INDICATOR
00415 725B 8D 4CB3        JSR    USMEM      STO +,- I
00416 725E 8D 05          BSR    XEY1       EXCHANGE XR AND YR
00417 7260 86 09          LDA A  #9         'J' REG. INDICATOR
00418 7262 8D 4CB3        JSR    USMEM      STO +,- J
00419 7265 7E 55AC  XEY1  JMP    XEY
00421               *
00422               *     RCL-ACC
00423               *
00424               *     RETURNED VALUES:
00425               *       XR - VALUE OF IR
```

```
00426                    *       YR - VALUE OF JR
00427                    *
00428 7268 BD 55EF RACC  JSR     STKUP     STACK OPERATION
00429 726B 96 0A         LDA A   EOM
00430 726D 4A            DEC A             LAST PAGE OF RAM
00431 726E C6 F8         LDA B   #$F8      REG J
00432 7270 D7 0D         STA B   STKFLG    SET FOR AUTO STACK
00433 7272 97 22         STA A   T13
00434 7274 D7 23         STA B   T12       POINTER
00435 7276 DE 22         LDX     T13
00436 7278 8D 61         BSR     TXX       LOAD JR INTO XR
00437 727A BD 55EF       JSR     STKUP     STACK RESULT
00438 727D 86 F0         LDA A   #$F0
00439 727F 97 23         STA A   T12       POINTER TO IR
00440 7281 DE 22         LDX     T13
00441 7283 20 56         BRA     TXX       LOAD IR INTO XR
00443                  *
00444                  *     MEAN - STANDARD DEVIATION
00445                  *
00446                  *     THIS ROUTINE CALCULATES THE MEAN AND STANDARD
00447                  *     DEVIATION GIVEN THE FOLLOWING ALPHA REGISTER
00448                  *     VALUES:
00449                  *       CR - SUM 'X'
00450                  *       DR - SUM 'X^2'
00451                  *       ER - N (#OF TERMS)
00452                  *     RETURNED VALUES:
00453                  *       XR - MEAN
00454                  *       YR - STANDARD DEVIATION
00455                  *     FORMULAS IMPLEMENTED:
00456                  *       MEAN = (SUM 'X')/N
00457                  *       S.D. = SQRT(((SUM 'X^2')-(SUM 'X')*(MEAN))/(N-1))
00458                  *
00459 7285 96 F9   MSDEV  LDA A   ER+1      (+) NUMBER?
00460 7287 2A 05         BPL     *+7       YES
00461 7289 86 06         LDA A   #6
00462 728B 97 06         STA A   ERROR     NO, SET ERROR
00463 728D 39            RTS
00464 728E BD 55EF       JSR     STKUP     STACK OPERATION
00465 7291 CE 00E8       LDX     #CR
00466 7294 8D 45         BSR     TXX       LOAD SUM 'X'
00467 7296 CE 0078       LDX     #AT1
00468 7299 BD 73F3       JSR     TXRX      COPY XR INTO TEMP.
00469 729C CE 00F8       LDX     #ER
00470 729F 8D 3A         BSR     TXX       LOAD 'N' INTO XR
00471 72A1 CE 0078       LDX     #AT1
00472 72A4 BD 7793       JSR     FPD       DIVIDE 'N' INTO SUM'X'
00473 72A7 CE 0078       LDX     #AT1
00474 72AA 8D 32         BSR     TXR       SAVE THE MEAN
00475 72AC CE 00E8       LDX     #CR
00476 72AF BD 7735       JSR     FPM       MEAN*SUM'X'
00477 72B2 CE 00F0       LDX     #DR
00478 72B5 BD 75F6       JSR     FPS       SUM'X^2'-MEAN*SUM'X'
00479 72B8 CE 0080       LDX     #AT2
00480 72BB 8D 21         BSR     TXR       SAVE PARTIAL RESULT
00481 72BD CE 6838       LDX     #$6838
00482 72C0 8D 19         BSR     TXX       LOAD A FLOATING 1.0
00483 72C2 CE 00F8       LDX     #ER
00484 72C5 BD 75F6       JSR     FPS       'N'-1.0
00485 72C8 CE 0080       LDX     #AT2
00486 72CB BD 7793       JSR     FPD       S.D. SQUARED
00487 72CE BD 6E65       JSR     SQRT      FINISH S.D.
00488 72D1 86 80         LDA A   #$80
00489 72D3 97 0D         STA A   STKFLG    SET AUTO STACK
00490 72D5 BD 55EF       JSR     STKUP
00491 72D8 CE 0078       LDX     #AT1
00492 72DB 7E 7438 TXX   JMP     TXXR      LOAD MEAN INTO XR
00493 72DE 7E 73F3 TXR   JMP     TXRX
00495                  *
00496                  *     SIGMA ROUTINES
00497                  *
00498                  *     THIS ROUTINE IMPLEMENTS THE SINGLE VARRIABLE
00499                  *     SUMMATION OF 'X', 'X^2', AND 'N'.
00500                  *     USER PARAMETERS:
00501                  *       XR - NEW 'X' VALUE
00502                  *       CR - CURRENT SUM 'X'
```

```
00503                    *       DR - CURRENT SUM 'XA2'
00504                    *       ER = CURRENT 'N'
00505                    *     RETURNED VALUES:
00506                    *       XR - NEW 'X'
00507                    *       CR - NEW SUM 'X'
00508                    *       DR - NEW SUM 'XA2'
00509                    *       ER - NEW 'N'
00510                    *
00511 72E1 4C           SIGM    INC A
00512 72E2 97 22        SIGP    STA A  T13      SET ENTRY FLAG
00513 72E4 CE 0078              LDX    #AT1
00514 72E7 8D F5                BSR    TXR      SAVE 'X'
00515 72E9 CE 00E8              LDX    #CR
00516 72EC 8D 1F                BSR    OPER     EXECUTE '+' OR '-'
00517 72EE CE 0078              LDX    #AT1
00518 72F1 8D E8                BSR    TXX      RELOAD 'X' INTO XR
00519 72F3 CE 0078              LDX    #AT1
00520 72F6 8D 7735              JSR    FPM      'X'A2
00521 72F9 CE 00F0              LDX    #DR
00522 72FC 8D 0F                BSR    OPER     EXECUTE '+' OR '-'
00523 72FE CE 6838              LDX    #$6838
00524 7301 8D D8                BSR    TXX      LOAD 1.0 INTO XR
00525 7303 CE 00F8              LDX    #ER
00526 7306 8D 05                BSR    OPER     EXECUTE '+' OP '-'
00527 7308 CE 0078              LDX    #AT1
00528 730B 20 CE                BRA    TXX      RESTORE 'X'
00530                    *
00531                    *     PERFORMS '+' OR '-' OPERATION DEPENDING
00532                    *     ON SPECIFIER FLAG T13 (0-ADD).
00533                    *
00534 730D DF 23        OPER    STX    T12      SAVE REG POINTER
00535 730F 96 22                LDA A  T13
00536 7311 27 05                BEQ    OPER1    DO ADD
00537 7313 8D 75F6              JSR    FPS
00538 7316 20 03                BRA    OPER1+3
00539 7318 8D 75FC OPER1        JSR    FPA
00540 731B DE 23                LDX    T12      POINTER
00541 731D 20 BF                BRA    TXR      SAVE RESULT
00543                    *
00544                    *     CREG
00545                    *
00546                    *     CLEARS ALPHA REGISTERS A-J.
00547                    *
00548 731F CE 00D8 CREG         LDX    #AR
00549 7322 8D 0B                BSR    CLG      CLEAR A-E
00550 7324 D6 0A                LDA B  EOM
00551 7326 5A                   DEC B           LAST PAGE OF RAM
00552 7327 D7 14                STA B  TP1
00553 7329 C6 D8                LDA B  #$D8     BUILD ADDRESS
00554 732B D7 15                STA B  TP1+1
00555 732D DE 14                LDX    TP1      ADDRESS OF F-J
00556 732F C6 28        CLG     LDA B  #40      TOTAL COUNT
00557 7331 A7 00                STA A  0,X      ZERO BYTE
00558 7333 08                   INX             BUMP ADDRESS
00559 7334 5A                   DEC B           DONW WITH FIVE REGS YET?
00560 7335 26 FA                BNE    CLG+2    NO
00561 7337 39                   RTS
00563                    *
00564                    *     INSTRUCTION ADDRESS TABLE INSERTS
00565                    *
00566 7C4E                      ORG    $7C4E
00567 7C4E 72E2                 FDB    SIGP     SIGMA +
00568 7C50 72E1                 FDB    SIGM     SIGMA -
00570 7C56                      ORG    $7C56
00571 7C56 7251                 FDB    ACCP     ACCUM +
00572 7C58 7250                 FDB    ACCM     ACCUM -
00574 7C6E                      ORG    $7C6E
00575 7C6E 7285                 FDB    MSDEV    MEAN & S.D.
00577 7C76                      ORG    $7C76
00578 7C76 7268                 FDB    RACC
00579 7C78 731F                 FDB    CREG
00582                           NAM    LINK
00583                           OPT    LIST,NOMEM
00584 7BFF                      ORG    $7BFF
00585                    *
```

```
00586                *THIS ROUTINE DEFINES THE LINKAGE
00587                *TABLE FOR CJ.
00588                *INST*2+$7C00=ROUTINE ADDRESS
00589                *
00590 78FF 39       RRTN   RTS             RETURN FOR DUMMYS
00591 7C00 78FF             FDB    RRTN     NOP
00592 7C02 78FF             FDB    RRTN     0
00593 7C04 78FF             FDB    RRTN     1
00594 7C06 78FF             FDB    RRTN     2
00595 7C08 78FF             FDB    RRTN     3
00596 7C0A 78FF             FDB    RRTN     4
00597 7C0C 78FF             FDB    RRTN     5
00598 7C0E 78FF             FDB    RRTN     6
00599 7C10 78FF             FDB    RRTN     7
00600 7C12 78FF             FDB    RRTN     8
00601 7C14 78FF             FDB    RRTN     9
00602 7C16 78FF             FDB    RRTN     .
00603 7C18 78FF             FDB    RRTN     +
00604 7C1A 78FF             FDB    RRTN     -
00605 7C1C 78FF             FDB    RRTN     *
00606 7C1E 78FF             FDB    RRTN     /
00607 7C20 78FF             FDB    RRTN     ENTER
00608 7C22 78FF             FDB    RRTN     EEX
00609 7C24 78FF             FDB    RRTN     CHS
00610 7C26 78FF             FDB    RRTN     CLX
00611 7C28 78FF             FDB    RRTN     CLEAR
00612 7C2A 78FF             FDB    RRTN     XEY
00613 7C2C 78FF             FDB    RRTN     ROLL DOWN
00614 7C2E 78FF             FDB    RRTN     PI
00615 7C30 78FF             FDB    RRTN     LAST X
00616 7C32 78FF             FDB    RRTN     SIN
00617 7C34 78FF             FDB    RRTN     COS
00618 7C36 78FF             FDB    RRTN     TAN
00619 7C38 78FF             FDB    RRTN     ARCSIN
00620 7C3A 78FF             FDB    RRTN     ARCCOS
00621 7C3C 78FF             FDB    RRTN     ARCTAN
00622 7C3E 78FF             FDB    RRTN     LN
00623 7C40 78FF             FDB    RRTN     LOG
00624 7C42 78FF             FDB    RRTN     E^X
00625 7C44 78FF             FDB    RRTN     10^X
00626 7C46 78FF             FDB    RRTN     Y^X
00627 7C48 78FF             FDB    RRTN     ROLL UP
00628 7C4A 78FF             FDB    RRTN     SQRT X
00629 7C4C 78FF             FDB    RRTN     1/X
00630 7C4E 78FF             FDB    RRTN     SIGMA +
00631 7C50 78FF             FDB    RRTN     SIGMA -
00632 7C52 78FF             FDB    RRTN     R TO P
00633 7C54 78FF             FDB    RRTN     P TO R
00634 7C56 78FF             FDB    RRTN     ACC+
00635 7C58 78FF             FDB    RRTN     ACC-
00636 7C5A 78FF             FDB    RRTN     INT
00637 7C5C 78FF             FDB    RRTN     DEGREES
00638 7C5E 78FF             FDB    RRTN     TO D.MS
00639 7C60 78FF             FDB    RRTN     FROM D.MS
00640 7C62 78FF             FDB    RRTN     IF+
00641 7C64 78FF             FDB    RRTN     IF-
00642 7C66 78FF             FDB    RRTN     IF0
00643 7C68 78FF             FDB    RRTN     IFX=Y
00644 7C6A 78FF             FDB    RRTN     IFX<Y
00645 7C6C 78FF             FDB    RRTN     IFX>=Y
00646 7C6E 78FF             FDB    RRTN     MEAN, STD DEV.
00647 7C70 78FF             FDB    RRTN     # OF R'S
00648 7C72 78FF             FDB    RRTN     LIST STACK
00649 7C74 78FF             FDB    RRTN     ROUND X
00650 7C76 78FF             FDB    RRTN     RCL ACC +
00651 7C78 78FF             FDB    RRTN     CLR A THRU J
00652 7C7A 78FF             FDB    RRTN     LINE FEED
00653 7C7C 78FF             FDB    RRTN     GRADS
00654 7C7E 78FF             FDB    RRTN     RADIANS
00655 7C80 78FF             FDB    RRTN     SFG 1
00656 7C82 78FF             FDB    RRTN     SFG 2
00657 7C84 78FF             FDB    RRTN     SFG 3
00658 7C86 78FF             FDB    RRTN     SFG 4
00659 7C88 78FF             FDB    RRTN     SFG 5
00660 7C8A 78FF             FDB    RRTN     SFG 6
```

```
00661 7C8C 78FF     FDB    RRTN     SFG 7
00662 7C8E 78FF     FDB    RRTN     SFG 8
00663 7C90 78FF     FDB    RRTN     IF SFG 1
00664 7C92 78FF     FDB    RRTN     IF SFG 2
00665 7C94 78FF     FDB    RRTN     IF SFG 3
00666 7C96 78FF     FDB    RRTN     IF SFG 4
00667 7C98 78FF     FDB    RRTN     IF SFG 5
00668 7C9A 78FF     FDB    RRTN     IF SFG 6
00669 7C9C 78FF     FDB    RRTN     IF SFG 7
00670 7C9E 78FF     FDB    RRTN     IF SFG 8
00671 7CA0 78FF     FDB    RRTN     CFG 1
00672 7CA2 78FF     FDB    RRTN     CFG 2
00673 7CA4 78FF     FDB    RRTN     CFG 3
00674 7CA6 78FF     FDB    RRTN     CFG 4
00675 7CA8 78FF     FDB    RRTN     CFG 5
00676 7CAA 78FF     FDB    RRTN     CFG 6
00677 7CAC 78FF     FDB    RRTN     CFG 7
00678 7CAE 78FF     FDB    RRTN     CFG 8
00679 7CB0 78FF     FDB    RRTN     IF CFG 1
00680 7CB2 78FF     FDB    RRTN     IF CFG 2
00681 7CB4 78FF     FDB    RRTN     IF CFG 3
00682 7CB6 78FF     FDB    RRTN     IF CFG 4
00683 7CB8 78FF     FDB    RRTN     IF CFG 5
00684 7CBA 78FF     FDB    RRTN     IF CFG 6
00685 7CBC 78FF     FDB    RRTN     IF CFG 7
00686 7CBE 78FF     FDB    RRTN     IF CFG 8
00687 7CC0 78FF     FDB    RRTN     GOTO LBL X
00688 7CC2 78FF     FDB    RRTN     GOSUB LBL X
00689 7CC4 78FF     FDB    RRTN     GOTO X
00690 7CC6 78FF     FDB    RRTN     GOSUB X
00691 7CC8 78FF     FDB    RRTN     STO A
00692 7CCA 78FF     FDB    RRTN     STO B
00693 7CCC 78FF     FDB    RRTN     STO C
00694 7CCE 78FF     FDB    RRTN     STO D
00695 7CD0 78FF     FDB    RRTN     STO E
00696 7CD2 78FF     FDB    RRTN     STO F
00697 7CD4 78FF     FDB    RRTN     STO G
00698 7CD6 78FF     FDB    RRTN     STO H
00699 7CD8 78FF     FDB    RRTN     STO I
00700 7CDA 78FF     FDB    RRTN     STO J
00701 7CDC 78FF     FDB    RRTN     STO+ A
00702 7CDE 78FF     FDB    RRTN     STO+ B
00703 7CE0 78FF     FDB    RRTN     CTO+ C
00704 7CE2 78FF     FDB    RRTN     STO+ D
00705 7CE4 78FF     FDB    RRTN     STO+ E
00706 7CE6 78FF     FDB    RRTN     STO+ F
00707 7CE8 78FF     FDB    RRTN     STO+ G
00708 7CEA 78FF     FDB    RRTN     STO+ H
00709 7CEC 78FF     FDB    RRTN     STO+ I
00710 7CEE 78FF     FDB    RRTN     STO+ J
00711 7CF0 78FF     FDB    RRTN     STO- A
00712 7CF2 78FF     FDB    RRTN     STO- B
00713 7CF4 78FF     FDB    RRTN     STO- C
00714 7CF6 78FF     FDB    RRTN     STO- D
00715 7CF8 78FF     FDB    RRTN     STO- E
00716 7CFA 78FF     FDB    RRTN     STO- F
00717 7CFC 78FF     FDB    RRTN     STO- G
00718 7CFE 78FF     FDB    RRTN     STO- H
00719 7D00 78FF     FDB    RRTN     STO- I
00720 7D02 78FF     FDB    RRTN     STO- J
00721 7D04 78FF     FDB    RRTN     STO* A
00722 7D06 78FF     FDB    RRTN     STO* B
00723 7D08 78FF     FDB    RRTN     STO* C
00724 7D0A 78FF     FDB    RRTN     STO* D
00725 7D0C 78FF     FDB    RRTN     STO* E
00726 7D0E 78FF     FDB    RRTN     STO* F
00727 7D10 78FF     FDB    RRTN     STO* G
00728 7D12 78FF     FDB    RRTN     STO* H
00729 7D14 78FF     FDB    RRTN     STO* I
00730 7D16 78FF     FDB    RRTN     STO* J
00731 7D18 78FF     FDB    RRTN     STO/ A
00732 7D1A 78FF     FDB    RRTN     STO/ B
00733 7D1C 78FF     FDB    RRTN     STO/ C
00734 7D1E 78FF     FDB    RRTN     STO/ D
00735 7D20 78FF     FDB    RRTN     STO/ E
```

```
00736 7D22 78FF          FDB    RRTN     STO/ F
00737 7D24 78FF          FDB    RRTN     STO/ G
00738 7D26 78FF          FDB    RRTN     STO/ H
00739 7D28 78FF          FDB    RRTN     STO/ I
00740 7D2A 78FF          FDB    RRTN     STO/ J
00741 7D2C 78FF          FDB    RRTN     RCL A
00742 7D2E 78FF          FDB    RRTN     RCL B
00743 7D30 78FF          FDB    RRTN     RCL C
00744 7D32 78FF          FDB    RRTN     RCL D
00745 7D34 78FF          FDB    RRTN     RCL E
00746 7D36 78FF          FDB    RRTN     RCL F
00747 7D38 78FF          FDB    RRTN     RCL G
00748 7D3A 78FF          FDB    RRTN     RCL H
00749 7D3C 78FF          FDB    RRTN     RCL I
00750 7D3E 78FF          FDB    RRTN     RCL J
00751 7D40 78FF          FDB    RRTN     CALL A (LOBIN)
00752 7D42 78FF          FDB    RRTN     CALL B (RUNBIN)
00753 7D44 78FF          FDB    RRTN     CALL C (RCBIN)
00754 7D46 78FF          FDB    RRTN     CALL D (RCBINSC)
00755 7D48 78FF          FDB    RRTN     CALL E (PTAPE)
00756 7D4A 78FF          FDB    RRTN     VERIFY FILE
00757 7D4C 78FF          FDB    RRTN     LOAD & RUN
00758 7D4E 78FF          FDB    RRTN     RECORD DATA
00759 7D50 78FF          FDB    RRTN     REWIND
00760 7D52 78FF          FDB    RRTN     PAUSE
00761 7D54 78FF          FDB    RRTN     FOR B TO G
00762 7D56 78FF          FDB    RRTN     FOR C TO H
00763 7D58 78FF          FDB    RRTN     FOR A TO F
00764 7D5A 78FF          FDB    RRTN     NEXT A
00765 7D5C 78FF          FDB    RRTN     NEXT B
00766 7D5E 78FF          FDB    RRTN     NEXT C
00767 7D60 78FF          FDB    RRTN     RUN/STOP
00768 7D62 78FF          FDB    RRTN     END
00769 7D64 78FF          FDB    RRTN     PRINT
00770 7D66 78FF          FDB    RRTN     RETURN
00771 7D68 78FF          FDB    RRTN     ALPHA TERMINATOR
00772 7D6A 78FF          FDB    RRTN     RECORD PROGRAM
00773 7D6C 78FF          FDB    RRTN     RECORD PROTECTED
00774 7D6E 78FF          FDB    RRTN     MARK FILE
00775 7D70 78FF          FDB    RRTN     IDENTIFY FILE
00776 7D72 78FF          FDB    RRTN     LOAD FILE
00777 7D74 78FF          FDB    RRTN     THIS BIT PATTERN IS UNUSED
00778                    ********************************************************************
00779 7D76 78FF          FDB    RRTN     FIXED
00780 7D78 78FF          FDB    RRTN     SCI
00781 7D7A 78FF          FDB    RRTN     SCI 3
00782 7D7C 78FF          FDB    RRTN     FORMAT
00783 7D7E 78FF          FDB    RRTN     LABEL
00784 7D80 78FF          FDB    RRTN     GOSUB LBL
00785 7D82 78FF          FDB    RRTN     GOTO LBL
00786 7D84 78FF          FDB    RRTN     RCL RCL ALPHA
00787 7D86 78FF          FDB    RRTN     STO RCL ALPHA
00788 7D88 78FF          FDB    RRTN     STO RCL + ALPHA
00789 7D8A 78FF          FDB    RRTN     STO RCL - ALPHA
00790 7D8C 78FF          FDB    RRTN     STO RCL * ALPHA
00791 7D8E 78FF          FDB    RRTN     STO RCL / ALPHA
00792 7D90 78FF          FDB    RRTN     STO 0
00793 7D92 78FF          FDB    RRTN     STO 1
00794 7D94 78FF          FDB    RRTN     STO+ 0
00795 7D96 78FF          FDB    RRTN     STO+ 1
00796 7D98 78FF          FDB    RRTN     STO- 0
00797 7D9A 78FF          FDB    RRTN     STO- 1
00798 7D9C 78FF          FDB    RRTN     STO* 0
00799 7D9E 78FF          FDB    RRTN     STO* 1
00800 7DA0 78FF          FDB    RRTN     STO/ 0
00801 7DA2 78FF          FDB    RRTN     STO/ 1
00802 7DA4 78FF          FDB    RRTN     RCL 0
00803 7DA6 78FF          FDB    RRTN     RCL 1
00804 7DA8 78FF          FDB    RRTN     RCL RCL 0
00805 7DAA 78FF          FDB    RRTN     RCL RCL 1
00806 7DAC 78FF          FDB    RRTN     STO RCL 0
00807 7DAE 78FF          FDB    RRTN     STO RCL 1
00808 7DB0 78FF          FDB    RRTN     STO RCL+ 0
00809 7DB2 78FF          FDB    RRTN     STO RCL+ 1
00810 7DB4 78FF          FDB    RRTN     STO RCL- 0
```

```
00811 7DB6 7BFF          FDB     RRTN      STO RCL- 1
00812 7DB8 7BFF          FDB     RRTN      STO RCL* 0
00813 7DBA 7BFF          FDB     RRTN      STO RCL* 1
00814 7DBC 7BFF          FDB     RRTN      STO RCL/ 0
00815 7DBE 7BFF          FDB     RRTN      STO RCL/ 1
00816 7DC0 7BFF          FDB     RRTN      GOSUB 0
00817 7DC2 7BFF          FDB     RRTN      GOSUB 1
00818 7DC4 7BFF          FDB     RRTN      GOSUB 2
00819 7DC6 7BFF          FDB     RRTN      GOSUB 3
00820 7DC8 7BFF          FDB     RRTN      GOSUB 4
00821 7DCA 7BFF          FDB     RRTN      GOSUB 5
00822 7DCC 7BFF          FDB     RRTN      GOSUB 6
00823 7DCE 7BFF          FDB     RRTN      GOSUB 7
00824 7DD0 7BFF          FDB     RRTN      GOSUB 8
00825 7DD2 7BFF          FDB     RRTN      GOSUB 9
00826 7DD4 7BFF          FDB     RRTN      GOSUB 10
00827 7DD6 7BFF          FDB     RRTN      GOSUB 11
00828 7DD8 7BFF          FDB     RRTN      GOSUB 12
00829 7DDA 7BFF          FDB     RRTN      GOSUB 13
00830 7DDC 7BFF          FDB     RRTN      GOSUB 14
00831 7DDE 7BFF          FDB     RRTN      GOSUB 15
00832 7DE0 7BFF          FDB     RRTN      GOTO 0
00833 7DE2 7BFF          FDB     RRTN      GOTO 1
00834 7DE4 7BFF          FDB     RRTN      GOTO 2
00835 7DE6 7BFF          FDB     RRTN      GOTO 3
00836 7DE8 7BFF          FDB     RRTN      GOTO 4
00837 7DEA 7BFF          FDB     RRTN      GOTO 5
00838 7DEC 7BFF          FDB     RRTN      GOTO 6
00839 7DEE 7BFF          FDB     RRTN      GOTO 7
00840 7DF0 7BFF          FDB     RRTN      GOTO 8
00841 7DF2 7BFF          FDB     RRTN      GOTO 9
00842 7DF4 7BFF          FDB     RRTN      GOTO 10
00843 7DF6 7BFF          FDB     RRTN      GOTO 11
00844 7DF8 7BFF          FDB     RRTN      GOTO 12
00845 7DFA 7BFF          FDB     RRTN      GOTO 13
00846 7DFC 7BFF          FDB     RRTN      GOTO 14
00847 7DFE 7BFF          FDB     RRTN      GOTO 15
```

SYMBOL TABLE

ADATA	0000	ACTL	0001	BDATA	0002	BCTL	0003	INPUT	0004	IOIN	0005
ERROR	0006	TGL	0007	UFLG	0008	RSFLG	0009	EOM	000A	EOPM	000B
STKFLG	000D	RND	000E	DIGFLG	000F	W2	0010	W1	0011	SFLG	0012
DCNTR	0013	TP1	0014	TP1S	0015	TP2	0016	TP2S	0017	TP3	0018
TP3S	0019	TP4	001A	TP4S	001B	TP5	001C	TP5S	001D	TP6	001E
TP6S	001F	TP7	0020	TP7S	0021	T13	0022	T12	0023	T11	0024
T10	0025	T9	0026	T8	0027	T7	0028	T6	0029	T5	002A
T4	002B	T3	002C	T2	002D	T1	002E	ISTK	002F	ISTACK	0051
TA	0052	SPGM	0054	EXTRA	0056	BUFF	0058	REAL	0068	IMAG	0070
AT1	0078	AT2	0080	W	0088	XR	0090	YR	0098	ZR	00A0
TR	00A8	LSTX	00B0	BKWRT	00B8	BKKC	00BA	SOL7	00C6	UPP	00C8
UIP	00CA	ALPHA	00CC	IO1	00CD	IO2	00D0	IT7	00D3	FLAG	00D5
TPOS	00D6	FILE	00D7	AR	00D8	BR	00E0	CR	00E8	DR	00F0
ER	00F8	SDBB	00BA	MT	7E00	TERMN7	0030	IMED	0040	PARCD	00C0
PAREX	0080	NTBL	0000	DOTS	5EC0	PRTDRV	6020	FRMT	5C48	BLANK	5D75
LDMSG	57BD	ROLLD	5582	ROLLU	57F1	PSD	55DA	TXL	55E9	STKUP	55EF
MAD	749B	CMP	74AA	NOR	74D6	TXW	7424	TXXR	7438	EXXR	7452
ARSR	753B	OVUNF	7586	OVERF	75DD	XR0	740A	XRNINE	75C8	UNDRF	75F1
IMULT	7689	QDG	7669	FPA	75FC	FPS	75F6	FPM	7735	FPD	7793
FPAEX	763D	FPMEX	7780	LSHIFT	7521	ZEROX	7489	XZERO0	7416	XZERO2	7417
RECIP	73E6	TXRX	73F3	CONST	6800	FPDBRC	6898	TAN	68A9	ATN	69C3
DSZERO	6A46	NTLN	6A58	EXPN	6AC9	SIN	6894	COS	689A	ASIN	68F2
ACOS	68F7	PH1	6C5D	PH2	6C8D	PH3	6D34	PH4	6D00	LSFTB	6E47
SQRT	6E65	MADB	6F2C	CMPB	53E4	IOUPX	6F52	LOG10	6FA7	YUPX	6FE9
RTOP	7328	PTOR	7386	ROMID	5300	RM1	5305	RM2	5316	RM3	5329
IOFMT	532A	IOF1	5337	IOF1A	5347	IOF2	534F	RNPRT	5351	STBIN	62AD
LDBIN	6396	SPSTS	5354	SPST	5356	SPS0	5366	SPS0A	537A	SPS1	537D
LIMIT	5380	LIM1	538E	KILL	538F	KIL1	5393	SPL0	539C	SPL1	53A9
SPCAL	53AE	SPTAB	53BA	SPT1	53CA	USMEM	4CB3	XEY	55AC	ACCM	7250
ACCP	7251	XEY1	7265	RACC	7268	MSDEV	7285	TXX	7208	TXR	720E
SIGM	72E1	SIGP	72E2	OPER	730D	OPER1	7318	CREG	731F	CLG	732F
RRTN	7BFF										

```
TOTAL ERRORS   3


****ERROR 201
 218  NAM CONST
****ERROR 201
 303  NAM FPDBRC
****ERROR 201
 324  NAM TAN
****ERROR 206
 327 FLAG EQU BUFF+15
****ERROR 206
 333 ADDRST EQU TP7
****ERROR 206
 335 TS EQU REAL
****ERROR 201
 480  NAM ATN
****ERROR 206
 483 FLAG EQU BUFF+15
****ERROR 206
 484 EXP EQU BUFF+14
****ERROR 206
 487 ADDRST EQU TP7
****ERROR 206
 488 SHIFT1 EQU T1
****ERROR 206
 489 SHIFT2 EQU T2
****ERROR 206
 491 TEMP EQU T4
****ERROR 201
 567  NAM DSZERO
****ERROR 206
 570 DIGIT EQU BUFF
****ERROR 201
 589  NAM NTLN
****ERROR 206
 592 FLAG EQU BUFF+15
****ERROR 206
 593 EXP EQU BUFF+14
****ERROR 206
 594 SHIFT1 EQU T1
****ERROR 206
 595 SHIFT3 EQU T3
****ERROR 206
 596 TEMP EQU T4
****ERROR 206
 597 TS EQU REAL
****ERROR 201
 665  NAM EXPN
****ERROR 206
 668 FLAG EQU BUFF+15
****ERROR 206
 669 EXP EQU BUFF+14
****ERROR 206
 671 ADDRST EQU TP7
****ERROR 206
 672 SHIFT1 EQU T1
****ERROR 206
 673 ADDRS EQU TP6
****ERROR 206
 674 ZQFLAG EQU TP5
****ERROR 206
 675 DIGIT EQU BUFF
****ERROR 206
 676 TRIG EQU TRIG1-14
****ERROR 201
 788  NAM SIN
****ERROR 206
 791 FLAG EQU BUFF+15
****ERROR 206
 792 TS EQU REAL
****ERROR 206
 793 QAD EQU BUFF+13
****ERROR 206
 794 TEMP EQU T4
```

```
****ERROR 201
 851  NAM ASIN
****ERROR 206
 854 TS EQU REAL
****ERROR 206                                    5
 855 ADDRST EQU TP7
****ERROR 206
 856 FLAG EQU BUFF+15
****ERROR 206
 857 XTS EQU IMAG                                10
****ERROR 201
 920  NAM PH1
****ERROR 206
 923 DIGIT EQU BUFF
****ERROR 206
 924 ADDRS EQU TP6                               15
****ERROR 206
 925 ADDRST EQU TP7
****ERROR 206
 926 QDIGIT EQU TP3
****ERROR 206                                    20
 927 SHIFT1 EQU T1
****ERROR 206
 928 ZQFLAG EQU TP5
****ERROR 201
 965  NAM PH2
****ERROR 206                                    25
 968 SHIFT3 EQU T3
****ERROR 206
 969 FLAG EQU BUFF+15
****ERROR 206
 970 ADDRS EQU TP6                               30
****ERROR 206
 971 SHIFT2 EQU T2
****ERROR 206
 972 TS EQU REAL
00215                       OPT    LIST,MEM
00218                       NAM    CONST
00219                       OPT    DB16,MEM
00220 6800                  ORG    CONST
00221                  *
00222                  *THIS FILE CONTAINS THE CONVERSION CONSTANTS USED IN
00223                  *THE MATH SUBROUTINES. BOTH THE TRIG AND LN CONSTANTS
00224                  *USED BY THE MATH ALGORITHMS HAVE ONLY THE MANTISSA
00225                  *STORED. ALSO, FOR EACH SET, EACH SUCCESSIVE CONSTANT
00226                  *IS LEFT SHIFTED ONCE FROM THE PRECEEDING CONSTANT.
00227                  *PROPER VALUE ALIGNMENT IS PERFORMED DURING EXECUTION
00228                  *OF EACH ALGORITHM.
00229                  *ALL OTHER CONSTANTS ARE STORED AS FULL
00230                  *FLOATING POINT NUMBERS.
00231                  *
00232                  * R. FUHRMAN  2/28/75  REV B
00233                  *
00234                  *                GRADS/UNIT CIRCLE ( 400 )
00235                  *
00236 6800 02       FIRST  FCB    $02,$00,$40,$00,$00,$00,$00,$00
00237                  *                RADIANS/UNIT CIRCLE ( 2PI )
00238                  *
00239 6808 00       A2PI   FCB    $00,$00,$62,$83,$18,$53,$07,$18
00240                  *                DEGREES/UNIT CIRCLE ( 360 )
00241                  *
00242 6810 02               FCB    $02,$00,$36,$00,$00,$00,$00,$00
00243                  *                GRADS/QUADRANT ( 100 )
00244                  *
00245 6818 02               FCB    $02,$00,$10,$00,$00,$00,$00,$00
00246                  *                RADIANS/QUADRANT ( PI/2 )
00247                  *
00248 6820 00       PI02   FCB    $00,$00,$15,$70,$79,$63,$26,$80
00249                  *                DEGREES/QUADRANT ( 90 )
00250                  *
00251 6828 01               FCB    $01,$00,$90,$00,$00,$00,$00,$00
00252                  *                GRADS/RADIAN ( 200/PI )
00253                  *
00254 6830 01       GRAD   FCB    $01,$00,$63,$66,$19,$77,$23,$68
00255                  *                RADIANS/RADIAN ( 1 )
```

```
00256                    *
00257 6838 00            ONE     FCB     $00,$00,$10,$00,$00,$00,$00,$00
00258                    *                DEGREES/RADIAN ( 180/PI )
00259                    *
00260 6840 01            DEGREE  FCB     $01,$00,$57,$29,$57,$79,$51,$31
00261                    *                ARCTAN( 1 )
00262                    *
00263 6848 78            TRIG1   FCB     $78,$53,$98,$16,$33,$97
00264                    *                ARCTAN( .1 )
00265                    *
00266 684E 99                    FCB     $99,$66,$86,$52,$49,$12
00267                    *                ARCTAN( .01 )
00268                    *
00269 6854 99                    FCB     $99,$99,$66,$66,$86,$67
00270                    *                ARCTAN( .001 )
00271                    *
00272 685A 99                    FCB     $99,$99,$99,$66,$66,$67
00273                    *                ARCTAN( .0001 )
00274                    *
00275 6860 99                    FCB     $99,$99,$99,$99,$66,$67
00276                    *                ARCTAN( .00001 )
00277                    *
00278 6866 99                    FCB     $99,$99,$99,$99,$99,$67
00279                    *                LN( 2 )
00280                    *
00281 686C 69                    FCB     $69,$31,$47,$18,$05,$60
00282                    *                LN( 1.1 )
00283                    *
00284 6872 95                    FCB     $95,$31,$01,$79,$80,$43
00285                    *                LN( 1.01 )
00286                    *
00287 6878 99                    FCB     $99,$50,$33,$08,$53,$17
00288                    *                LN( 1.001 )
00289                    *
00290 687E 99                    FCB     $99,$95,$00,$33,$30,$84
00291                    *                LN( 1.0001 )
00292                    *
00293 6884 99                    FCB     $99,$99,$50,$00,$33,$33
00294                    *                LN( 1.00001 )
00295                    *
00296 688A 99                    FCB     $99,$99,$95,$00,$00,$33
00297                    *                LN( 10 )
00298                    *
00299 6890 00            LN10    FCB     $00,$00,$23,$02,$58,$50,$92,$99
00300                            OPT     LIST
00216                            OPT     LIST,MEM
00219 3000                       ORG     $3000
00220                            OPT     G
00221                    *
00222                    **
00223                    ***     CJ GENERAL I/O ROM
00224                    ***     REV 01
00225                    ***     TIM HICKENLOOPER
00226                    ***     JULY 1975
00227                    **
00228                    *


00230 3000 20                    FCB     $20,$30   ROM ID #'S
      3001 30
00231 3002 7E 3084               JMP     PWRUP     POWER UP JUMP
00232 3005 30E2                  FDB     NULL      WRITE
00233 3007 31FA                  FDB     PRNTX
00234 3009 3359                  FDB     READX
00235 300B 3266                  FDB     PRINT
00236 300D 3172                  FDB     FIELD
00237 300F 33E8                  FDB     DELIM
00238 3011 31A8                  FDB     WBYTE
00239 3013 31C0                  FDB     RBYTE
00240 3015 30E2                  FDB     NULL      AND
00241 3017 30E2                  FDB     NULL      OR
00242 3019 30E2                  FDB     NULL      ROT
00243 301B 30E2                  FDB     NULL      LIST
00244 301D 30E2                  FDB     NULL      LDPGM
00245 301F 30E2                  FDB     NULL      DUPGM
```

```
00246 3021 330D            FDB    FLG
00247 3023 3328            FDB    DATA
00249                 *
00250                 *    ASCII FOR LISTING
00251                 *

00253                      OPT    NG
00254 3025 07              FCB    $D7,$52,$54,$68,$20    WRT 'ALPHA'
00255 302A 07              FCB    $D7,$52,$54,$58,$20    WRTX
00256 302F D2              FCB    $D2,$45,$41,$44,$58    READX
00257 3034 07              FCB    $D7,$52,$49,$54,$45    WRITE
00258 3039 C6              FCB    $C6,$49,$45,$4C,$44    FIELD
00259 303E C4              FCB    $C4,$45,$4C,$49,$4D    DELIM
00260 3043 07              FCB    $D7,$42,$59,$54,$45    WBYTE
00261 3048 D2              FCB    $D2,$42,$59,$54,$45    RBYTE
00262 304D C1              FCB    $C1,$4E,$44,$20,$20    AND
00263 3052 CF              FCB    $CF,$52,$20,$20,$20    OR
00264 3057 D2              FCB    $D2,$4F,$54,$20,$20    ROT
00265 305C CC              FCB    $CC,$49,$53,$54,$20    LIST
00266 3061 CC              FCB    $CC,$44,$50,$47,$4D    LDPGM
00267 3066 C4              FCB    $C4,$55,$50,$47,$4D    DUPGM
00268 306B C6              FCB    $C6,$4C,$41,$47,$20    FLAG
00269 3070 C4              FCB    $C4,$41,$54,$41,$20    DATA
00270                      OPT    G



00272                 *
00273                 *    ERROR MESSAGES
00274                 *

00276                      OPT    NG
00277                 *    SELECT CODE ERR
00278 3075 D3              FCB    $D3,$45,$4C,$45,$43,$54,$20,$43,$4F,$44
00279 307F 45              FCB    $45,$20,$45,$52,$52
00280                      OPT    G



00282       49CE     TSFR  EQU    $49CE
00283       3636     ELIN  EQU    $3636
00285                 *
00286                 *    PWRUP
00287                 *
00288                 *    POWER UP INITIALIZATION ROUTINE.
00289                 *    ALLOCATES ONE REGISTER FOR EACH CHANNEL TO BE
00290                 *    USED.  DISABLES ONE ROM IF THERE ARE TWO.
00291                 *
00292 3084 C6 5B    PWRUP LDA B  #$5B      CHANNEL 1, IO7
00293 3086 8D 51           BSR    RDID     GIO THERE?
00294 3088 26 05           BNE    PWR1     NO
00295 308A CE 00CD         LDX    #IO1
00296 308D 8D 54           BSR    SETUP    INITIALIZE CHANNEL ONE
00297 308F C6 AB    PWR1   LDA B  #$AB     CHANNEL 2, IO7
00298 3091 8D 46           BSR    RDID     GIO THERE?
00299 3093 26 28           BNE    PWR4     NO
00300 3095 CE 00D0         LDX    #IO2
00301 3098 8D 49           BSR    SETUP    INITIALIZE CHANNEL TWO
00302 309A 96 CF           LDA A  IO1+2
00303 309C 91 D2           CMP A  IO2+2    TWO SAME DEVICE?
00304 309E 26 06           BNE    PWR2     NO
00305 30A0 86 A1           LDA A  #$A1
00306 30A2 97 06           STA A  ERROR    SET FATAL ERROR
00307 30A4 20 0C           BRA    PWR3     DISABLE ONE ROM
00308 30A6 84 07    PWR2   AND A  #7
00309 30A8 97 2E           STA A  T1       SAVE ID
00310 30AA 96 D2           LDA A  IO2+2
00311 30AC 84 07           AND A  #7
00312 30AE 91 2E           CMP A  T1       TWO GIO ROMS?
00313 30B0 26 0E           BNE    PWR4     NO
00314 30B2 86 2C    PWR3   LDA A  #@54
00315 30B4 97 03           STA A  BCTL     SET CB2 FUNCTION
00316 30B6 86 E0           LDA A  #$E0
00317 30B8 97 00           STA A  ADATA    DISABLE ROM TWO
00318 30BA 97 02           STA A  BDATA
```

```
00319 30BC A7 00          STA A  0,X      SET IT UP
00320 30BE 20 15          BRA    PWR5
00321 30C0 86 0A   PWR4   LDA A  #$A      DEVICE THREE(GIO)
00322 30C2 91 CF          CMP A  IO1+2    GIO THREE?
00323 30C4 27 04          BEQ    *+6      YES
00324 30C6 91 D2          CMP A  IO2+2    GIO THREE?
00325 30C8 26 0B          BNE    PWR5     NO
00326 30CA B6 2800        LDA A  $2800    MASH ADDRESS
00327 30CD 81 30          CMP A  #$30     MASH IN MACHINE?
00328 30CF 26 04          BNE    PWR5     NO
00329 30D1 86 A1          LDA A  #$A1     YES,
00330 30D3 97 06          STA A  ERROR    SET FATAL ERROR
00331 30D5 0F      PWR5   SEI
00332 30D6 7E 3189        JMP    EX1
00334 30D9 D7 00   RDID   STA B  ADATA    CHANNEL SELECT
00335 30DB 5A             DEC B
00336 30DC D7 00          STA B  ADATA    RELEASE IO7. SET IO1
00337 30DE 96 05          LDA A  IOIN     READ CODE
00338 30E0 85 02          BIT A  #2       TEST GIO BIT
00339 30E2 39      NULL   RTS


00341 30E3 C6 02   SETUP  LDA B  #2       LOAD CODE 2
00342 30E5 85 40          BIT A  #$40     CODE THREE?
00343 30E7 26 02          BNE    *+4      NO
00344 30E9 CA 08          ORA B  #8       SET BIT 3
00345 30EB E7 02          STA B  2,X      SET SELECT CODE
00346 30ED 96 0B          LDA A  EOPM
00347 30EF A7 00          STA A  0,X      SET REG POINTER
00348 30F1 96 0C          LDA A  EOPM+1
00349 30F3 80 08          SUB A  #8       ALLOCATE ONE REGISTER
00350 30F5 A7 01          STA A  1,X
00351 30F7 97 0C          STA A  EOPM+1
00352 30F9 EE 00          LDX    0,X      LOAD POINTER
00353 30FB 86 F0          LDA A  #$F0
00354 30FD A7 00          STA A  0,X
00355 30FF 4C             INC A
00356 3100 A7 01          STA A  1,X      (-) FLAG
00357 3102 A7 02          STA A  2,X      (-) DATA
00358 3104 86 10          LDA A  #16
00359 3106 A7 03          STA A  3,X      FORMAT FIELD = 16
00360 3108 39             RTS


00362                     *
00363                     *    INIT
00364                     *
00365                     *     CALLED BY ALL GIO ROUTINES TO INITIALIZE ALL
00366                     *     POINTER AND FLAGS.
00367                     *       T13 - REGISTER POINTER
00368                     *       T11 - 'STOP' EXECUTED FLAG
00369                     *       T10 - RSFLG SAVE LOC
00370                     *
00371 3109 DE CA   INIT   LDX    UIP
00372 310B A6 01          LDA A  1,X      CALL INSTR NUMBER
00373 310D 84 70          AND A  #$70     MASK IT
00374 310F 81 20          CMP A  #$20     CALL 2?
00375 3111 27 11          BEQ    IN2      YES
00376 3113 86 0A          LDA A  #$A      SELECT THREE
00377 3115 91 CF          CMP A  IO1+2    IN CHANNEL ONE?
00378 3117 27 1B          BEQ    IN4      YES
00379 3119 91 D2          CMP A  IO2+2    IN CHANNEL TWO?
00380 311B 27 11          BEQ    IN3      YES
00381 311D 86 05   IN1    LDA A  #5
00382 311F 97 06          STA A  ERROR    SET IO ERROR
00383 3121 31             INS
00384 3122 31             INS             BUMP RETURN STACK
00385 3123 39             RTS
00386 3124 86 02   IN2    LDA A  #2       SELECT TWO
00387 3126 91 CF          CMP A  IO1+2    IN CHANNEL ONE?
00388 3128 27 0A          BEQ    IN4      YES
00389 312A 91 D2          CMP A  IO2+2    IN CHANNEL TWO?
00390 312C 26 EF          BNE    IN1      NO. ERROR
```

```
00391 312E DE D0    IN3     LDX     IO2
00392 3130 C6 EB            LDA B   #$EB       CHANNEL 2, IO7
00393 3132 20 04            BRA     IN4+4
00394 3134 DE CD    IN4     LDX     IO1
00395 3136 C6 DB            LDA B   #$DB       CHANNEL 1, IO7
00396 3138 DF 22            STX     T13        SET REGISTER POINTER
00397 313A A6 01            LDA A   1,X        (+) FLAG?
00398 313C 2B 02            BMI     *+4        NO
00399 313E C4 3F            AND B   #$3F
00400 3140 84 03            AND A   #3         MASK MODE
00401 3142 97 80            STA A   AT2
00402 3144 47               ASR A              PERIPHERIAL MODE?
00403 3145 27 02            BEQ     IN4A       NO
00404 3147 C8 C0            EOR B   #$C0       OPPOSITE EDGE OF FLAG
00405 3149 0F       IN4A    SEI
00406 314A D7 27            STA B   T8         SET SELECT CODE
00407 314C D7 00            STA B   ADATA      SET CHANNEL, IO7
00408 314E C4 F0            AND B   #$F0
00409 3150 D7 00            STA B   ADATA      RELEASE IO7, LATCH IT
00410 3152 86 2C            LDA A   #@54
00411 3154 97 03            STA A   BCTL       SET CB2 FUNCTION
00412 3156 86 3D            LDA A   #@75
00413 3158 97 01            STA A   ACTL       SET CA1 FUNCTION
00414 315A 4F               CLR A
00415 315B 97 81            STA A   AT2+1      CLEAR STOP KEY FLAG
00416 315D 91 0F            CMP A   DIGFLG     DIGIT ENTRY?
00417 315F 27 05            BEQ     IN4B       NO
00418 3161 97 0F            STA A   DIGFLG
00419 3163 4A               DEC A              TERMINATE IT
00420 3164 97 0D            STA A   STKFLG
00421 3166 96 09    IN4B    LDA A   RSFLG      RUN/STOP FLAG
00422 3168 84 C0            AND A   #$C0
00423 316A 97 82    IN5     STA A   AT2+2      R/S FLAG
00424 316C 86 C0            LDA A   #$C0
00425 316E 97 09            STA A   RSFLG      SET TO RUN
00426 3170 0E               CLI
00427 3171 39               RTS
00429                   *
00430                   *     FIELD
00431                   *
00432                   *
00433                   *     SETS FIELD WIDTH FOR OUTPUTTING NUMBERS.
00434                   *     USER PARAMETERS:
00435                   *       XR - FIELD WIDTH (1<=XR<256)
00436                   *
00437 3172 8D 95    FIELD   BSR     INIT       INITIALIZE
00438 3174 CE 0090          LDX     #XR
00439 3177 8D 1F            BSR     BINRY      CONVERT TO BINARY
00440 3179 26 3A            BNE     WB1        ERROR
00441 317B 4D               TST A              ZERO FIELD?
00442 317C 2F 37            BLE     WB1
00443 317E DE 22            LDX     T13
00444 3180 A7 03            STA A   3,X



00446                   *
00447                   *     EXIT
00448                   *
00449                   *     RESETS RSFLG FOR EXIT, CLEARS CHANNEL SELECT
00450                   *     AND RESETS THE PIA FUNCTION.
00451                   *     ALL ROUTINES EXIT THROUGH HERE!!
00452                   *
00453 3182 0F       EXIT    SEI
00454 3183 96 82            LDA A   AT2+2      ENTRY RSFLG
00455 3185 2B 02            BMI     EX1        EXIT
00456 3187 97 09            STA A   RSFLG      RESTORE IT
00457 3189 C6 04    EX1     LDA B   #4
00458 318B D7 00            STA B   ADATA      IO6, RESET PIA
00459 318D 7F 0000          CLR     ADATA
00460 3190 86 3C            LDA A   #@74
00461 3192 97 01            STA A   ACTL       RESET PIA
00462 3194 97 03            STA A   BCTL
00463 3196 0E               CLI
00464 3197 39               RTS
```

```
00466                    *
00467                    *     BINRY
00468                    *
00469                    *     CONVERTS THE INTEGER PART OF BCD NUMBER TO
00470                    *     ITS 8-BIT BINARY.  IX - REG ADDRESS.
00471                    *     SETS '7' BIT FOR A LEGAL NUMBER (0 TO 256).
00472                    *
00473 3198 A6 01    BINRY  LDA A  1,X       LOAD SIGN, POSITIVE?
00474 319A 2B 0E           BMI    BIN2+2    NO
00475 319C E6 00           LDA B  0,X       LOAD EXPONENT
00476 319E C1 03           CMP B  #3        TEST IT
00477 31A0 2E 08           BGT    BIN2+2    # TOO BIG
00478 31A2 5C              INC B            DIGIT COUNT
00479 31A3 BD 49CE         JSR    TSFR      CONVERT TO BINRY
00480 31A6 96 21           LDA A  TP7S      LOAD 8-BIT CODE
00481 31A8 D6 20    BIN2   LDA B  TP7       LOAD RANGE FLAG
00482 31AA 39              RTS


00484                    *
00485                    *    W-BYTE
00486                    *
00487                    *     TAKES THE INTEGER PART OF THE DECIMAL
00488                    *     NUMBER AND SENDS ITS EQUIVALENT 8-BIT
00489                    *     BINARY BYTE TO THE DEVICE.
00490                    *     USER PARAMETERS:
00491                    *       XR - NUMBER (0<=XR<256)
00492                    *
00493 31AB BD 3109 WBYTE  JSR    INIT      INITIALIZE ALL
00494 31AE CE 0090        LDX    #XR
00495 31B1 8D E5          BSR    BINRY     CONVERT TO BINRY
00496 31B3 27 06          BEQ    WB4       VALID #
00497 31B5 86 06    WB1   LDA A  #6
00498 31B7 97 06    EROR  STA A  ERROR     ERROR CONDITION
00499 31B9 20 C7          BRA    EXIT
00500 31BB BD 3270 WB4    JSR    OTBYT     OUT IT
00501 31BE 20 C2    WB5   BRA    EXIT
00503                    *
00504                    *     RBYTE
00505                    *
00506                    *     READS AN 8-BIT BYTE FROM THE DEVICE AND RETURNS
00507                    *     ITS DECIMAL EQUIVALENT IN THE XR.
00508                    *     USER PARAMETERS:
00509                    *       - NONE
00510                    *
00511 31C0 BD 3109 RBYTE  JSR    INIT
00512 31C3 BD 32DA        JSR    INBYT     INPUT BYTE
00513 31C6 2B BA          BMI    EXIT      STOP KEY, ABORT
00514 31C8 16       BTDEC TAB
00515 31C9 4F             CLR A
00516 31CA 4C       RB1   INC A
00517 31CB C0 64          SUB B  #100      EXTRACT 100'S
00518 31CD 24 FB          BCC    RB1
00519 31CF D7 2D          STA B  T2        SAVE RESIDUE
00520 31D1 8D 14          BSR    BUILD     DIGIT ENTRY
00521 31D3 D6 2D          LDA B  T2
00522 31D5 86 0B          LDA A  #11
00523 31D7 4A       RB2   DEC A
00524 31D8 CB 0A          ADD B  #10       EXTRACT 10'S
00525 31DA 2B FB          BMI    RB2
00526 31DC D7 2D          STA B  T2        SAVE 1'S
00527 31DE 8D 07          BSR    BUILD     DIGIT
00528 31E0 96 2D          LDA A  T2
00529 31E2 4C             INC A            1'S KEY CODE
00530 31E3 8D 02          BSR    BUILD     ENTER IT
00531 31E5 20 D7          BRA    WB5


00533                    *
00534                    *     BUILD
00535                    *
00536                    *     KEY CODE OF NUMBER IS PASSED IN ACCA AND
00537                    *     RUN THROUGH THE DIGIT ENTRY ROUTINE.
```

```
00538                    *
00539 31E7 C6 7C   BUILD  LDA B  #$7C
00540 31E9 D7 14          STA B  TP1       BUILD TABLE ADDRESS
00541 31EB 48             ASL A
00542 31EC 97 15          STA A  TP1+1     INDEX INTO TABLE
00543 31EE DE 14          LDX    TP1
00544 31F0 EE 00          LDX    0,X       ADDRESS OF ROUTINE
00545 31F2 4F             CLR A
00546 31F3 5F             CLR B
00547 31F4 6E 00          JMP    0,X       ENTER DIGIT
00549                    *
00550                    *     PRNT-X
00551                    *
00552                    *     OUTPUTS THE XR ACCORDING TO FIELD AND MACHINE
00553                    *     FORMAT.  EXPONENT IS ALWAYS SENT SIGNED.
00554                    *     IF THE NUMBER IS TOO LARGE FOR THE
00555                    *     FIELD WIDTH, THEN '$'S ARE SENT FOR THE
00556                    *     ENTIRE FIELD.
00557                    *     USER PARAMETERS:
00558                    *       XR - NUMBER TO BE SENT
00559                    *
00560 31F6 86 80   PRNT   LDA A  #$80
00561 31F8 20 04          BRA    PR0
00562 31FA BD 3109 PRNTX  JSR    INIT      INITIALIZE ALL
00563 31FD 4F             CLR A
00564 31FE 97 26   PR0    STA A  T9        SET ENTRY FLAG
00565 3200 BD 5C8C        JSR    FRMT+$14    FORMAT XR
00566 3203 C6 0D          LDA B  #13
00567 3205 86 20          LDA A  #$20      BLANK
00568 3207 CE 0057        LDX    #BUFF-1
00569 320A 5A      PR1    DEC B            COUNT
00570 320B 08             INX
00571 320C A1 00          CMP A  0,X       MANTISSA COUNT
00572 320E 27 FA          BEQ    PR1
00573 3210 91 67          CMP A  BUFF+15
00574 3212 27 0E          BEQ    PR2       NO EXPONENT
00575 3214 91 65          CMP A  BUFF+13
00576 3216 26 04          BNE    *+6       (-) EXPONENT
00577 3218 86 2B          LDA A  #$2B
00578 321A 97 65          STA A  BUFF+13    STUFF A (+)
00579 321C 86 45          LDA A  #$45
00580 321E 97 64          STA A  BUFF+12    STUFF AN 'E'
00581 3220 CB 04          ADD B  #4       FIX TOTAL COUNT
00582 3222 DF 20   PR2    STX    TP7       SET POINTER
00583 3224 D7 2E          STA B  T1        SET CHARACTER COUNT
00584 3226 DE 22          LDX    T13
00585 3228 E0 03          SUB B  3,X       COUNT - FIELD
00586 322A 2F 11          BLE    PR5       WITHIN RANGE
00587 322C E6 03          LDA B  3,X
00588 322E D7 2E          STA B  T1        FIELD TOO SMALL
00589 3230 86 24   PR4    LDA A  #$24
00590 3232 BD 3C          BSR    OTBYT     OUT '$'
00591 3234 2B 28          BMI    PR9       STOP KEY
00592 3236 7A 002E        DEC    T1
00593 3239 26 F5          BNE    PR4
00594 323B 20 21          BRA    PR9
00595 323D D7 2D   PR5    STA B  T2
00596 323F 27 0B          BEQ    PR7       NO BLANKS
00597 3241 86 20   PR6    LDA A  #$20
00598 3243 BD 2B          BSR    OTBYT     LEADING BLANKS
00599 3245 2B 17          BMI    PR9       STOP KEY
00600 3247 7C 002D        INC    T2
00601 324A 2B F5          BMI    PR6
00602 324C 7A 002E PR7    DEC    T1
00603 324F 2B 0D          BMI    PR9       DONE
00604 3251 DE 20          LDX    TP7
00605 3253 A6 00          LDA A  0,X       LOAD CHARACTER
00606 3255 08             INX
00607 3256 DF 20          STX    TP7
00608 3258 84 7F          AND A  #$7F      MASK 7 BITS
00609 325A BD 14          BSR    OTBYT     OUT THE DIGIT
00610 325C 2A EE          BPL    PR7       NEXT CHAR
00611 325E 96 26   PR9    LDA A  T9
00612 3260 2A 01          BPL    PR10      REGULAR ENTRY
```

```
00613 3262 39              RTS
00614 3263 7E 3182 PR10    JMP     EXIT


00616                      *
00617                      *     PRINT
00618                      *
00619                      *     OUTPUTS THE XR IN OUTPUT FORMAT TO THE
00620                      *     DEVICE WITH A 'CR' AND 'LF'.
00621                      *     USER PARAMETERS:
00622                      *       XR - NUMBER TO BE OUTED
00623                      *
00624 3266 BD 3109 PRINT   JSR     INIT
00625 3269 8D 8B           BSR     PRNT
00626 326B BD 3636         JSR     ELIN      CR-LF
00627 326E 20 F3           BRA     PR10


00629                      *
00630                      *     OTBYT
00631                      *
00632                      *     OUTS ACCA TO PERIPHERIAL.  CHECKS FOR 'ECH'
00633                      *     AND SETS 'STP' LINE IF STOP KEY IS HIT.
00634                      *
00635 3270 DE 22   OTBYT   LDX     T13
00636 3272 E6 00           LDA B   0,X
00637 3274 C4 70           AND B   #$70      CLEAR I/O BIT
00638 3276 D7 00           STA B   ADATA
00639 3278 6D 02           TST     2,X       (+) DATA?
00640 327A 2A 01           BPL     OTB1      YES
00641 327C 43              COM A
00642 327D 97 02   OTB1    STA A   BDATA     OUTPUT DATA
00643 327F 8D 0E           BSR     CHECK     WAIT FOR DEVICE READY
00644 3281 8D 44           BSR     HNDSK     HANDSHAKE
00645 3283 D6 81           LDA B   AT2+1     LOAD STOP FLAG
00646 3285 39              RTS
00648                      *
00649                      *       FFLAG
00650                      *
00651 3286 C6 0A   FFLAG   LDA B   #$A       IO1
00652 3288 D7 00           STA B   ADATA     SET IT
00653 328A D6 05           LDA B   IOIN      READ DEVICE STATUS
00654 328C E8 01           EOR B   1,X       TEST FLAG SENCE
00655 328E 39              RTS

00657                      *
00658                      *     CHECK
00659                      *
00660                      *     EXITS IF STOP KEY SEQUENCE EXECUTED.
00661                      *     WAITS FOR FLAG HIGH IF 'ECH' NOT DISABLED.
00662                      *     WAITS FOR FLAG LOW FOR PERIPHERIAL MODE.
00663                      *
00664 328F D6 80   CHECK   LDA B   AT2
00665 3291 27 0B           BEQ     CH2-1     NORMAL, NO ECH
00666 3293 57              ASR B
00667 3294 26 09           BNE     CH2       PERIPHERIAL MODE
00668 3296 D6 09   CH1     LDA B   RSFLG
00669 3298 2A 0E           BPL     STOP      STOP KEY
00670 329A 8D EA           BSR     FFLAG
00671 329C 2A F8           BPL     CH1       DEVICE BUSY
00672 329E 39              RTS
00673 329F D6 09   CH2     LDA B   RSFLG
00674 32A1 2A 05           BPL     STOP      STOP KEY
00675 32A3 8D E1           BSR     FFLAG     TRANSFER BEGUN?
00676 32A5 2B F8           BMI     CH2       NO
00677 32A7 39              RTS

00679                      *
00680                      *     STOP
00681                      *
00682                      *     EXECUTE A STOP KEY. DRIVE 'STP' LOW.
00683                      *     RELEASE 'STP' AFTER 100 US
00684                      *
```

```
00685 32A8 D6 81    STOP    LDA B   AT2+1
00686 32AA 2B 1A            BMI     ST2         STOP EXECUTED
00687 32AC 8D 50            BSR     LATCH       RELEASE CTL
00688 32AE E6 00            LDA B   0,X
00689 32B0 C4 BF            AND B   #$BF        STOP BIT CLEAR
00690 32B2 D7 00            STA B   ADATA       SET IT
00691 32B4 D7 02            STA B   BDATA       LATCH
00692 32B6 D7 81            STA B   AT2+1       SET STOP FLAG
00693 32B8 C6 0F            LDA B   #15
00694 32BA 5A       ST1     DEC B               TIME LOOP
00695 32BB 2A FD            BPL     ST1
00696 32BD E6 00            LDA B   0,X
00697 32BF D7 00            STA B   ADATA       STP HIGH
00698 32C1 D7 02            STA B   BDATA       LATCH
00699 32C3 31               INS
00700 32C4 31               INS                 ABORT
00701 32C5 50               TST B               SET CONDITION CODE
00702 32C6 39       ST2     RTS
00704                   *
00705                   *     HNDSK
00706                   *
00707                   *     DRIVES CTL LINE LOW AND WAITS FOR
00708                   *     THE DDVICE TO RESPOND.
00709                   *
00710 32C7 C6 03    HNDSK   LDA B   #3
00711 32C9 D7 00            STA B   ADATA       SET IO5
00712 32CB D6 00            LDA B   ADATA       CLEAR BIT 7 OF ACTL
00713 32CD 7F 0000          CLR     ADATA       DRIVE CONTROL LINE
00714 32D0 D6 01    HND1    LDA B   ACTL
00715 32D2 2B CA            BMI     CH2-1       EDGE SEEN
00716 32D4 D6 09            LDA B   RSFLG
00717 32D6 2B F8            BMI     HND1        NO STOP KEY
00718 32D8 20 CE            BRA     STOP

00720                   *
00721                   *     INBYT
00722                   *
00723                   *     INPUTS A BYTE FROM PERIPHERIAL.
00724                   *     RETURNS IT IN ACCA.
00725                   *
00726 32DA DE 22    INBYT   LDX     T13
00727 32DC A6 00            LDA A   0,X
00728 32DE 97 00            STA A   ADATA       SET I/O BIT
00729 32E0 97 02            STA A   BDATA
00730 32E2 8D AB            BSR     CHECK       WAIT FOR DEVIDE
00731 32E4 D6 80            LDA B   AT2
00732 32E6 57               ASR B
00733 32E7 26 0C            BNE     INB1        YES
00734 32E9 8D DC            BSR     HNDSK       HANDSHAKE
00735 32EB 96 05    INB0    LDA A   IOIN
00736 32ED 6D 02            TST     2,X         (-) TRUE?
00737 32EF 2B 01            BMI     *+3         YES
00738 32F1 43               COM A
00739 32F2 D6 81            LDA B   AT2+1       STOP FLAG
00740 32F4 39               RTS
00741 32F5 8D 07    INB1    BSR     LATCH       LATCH DATA BUSS
00742 32F7 8D F2            BSR     INB0        INPUT BYTE
00743 32F9 8D CC            BSR     HNDSK       NOW WITH HANDSHAKE
00744 32FB D6 81            LDA B   AT2+1       STOP FLAG
00745 32FD 39               RTS


00747 32FE 96 27    LATCH   LDA A   TR
00748 3300 88 C0            EOR A   #$C0
00749 3302 8D 02            BSR     LAT1
00750 3304 88 CB            EOR A   #$CB
00751 3306 97 00    LAT1    STA A   ADATA
00752 3308 84 F0            AND A   #$F0
00753 330A 97 00            STA A   ADATA
00754 330C 39               RTS
00756                   *
00757                   *     FLG
00758                   *
00759                   *     CHANGES FROM (+) OR (-) TRUE FLAG LEVEL AND
00760                   *     SETS OR CLEARS 'ECH' MODE OF OPERATION.
```

```
00761                *    USER PARAMETERS:
00762                *      XR - '0' FOR ECH CLEAR
00763                *           '1' FOR ECH SET
00764                *           '2' FOR PERIPHERIAL MODE
00765                *           '-' FOR NEGATIVE FLAG
00766                *           '+' FOR POSITIVE FLAG
00767                *
00768 3300 BD 3109 FLG    JSR     INIT
00769 3310 D6 90          LDA B   XR          LOAD EXPONENT
00770 3312 26 11          BNE     EXXT+3
00771 3314 96 92          LDA A   XR+2        MSD OF MANTISSIA
00772 3316 44             LSR A
00773 3317 44             LSR A
00774 3318 44             LSR A               INTEGER PART ONLY
00775 3319 44             LSR A
00776 331A 81 02          CMP A   #2
00777 331C 22 07          BHI     EXXT+3      NOT VALID, TOO BIG
00778 331E 9A 91          ORA A   XR+1        ADD SIGN
00779 3320 A7 01          STA A   1,X         SET IT
00780 3322 7E 3182 EXXT   JMP     EXIT        DONE
00781 3325 7E 3185        JMP     WB1         SET ERROR, EXIT


00783                *
00784                *    DATA
00785                *
00786                *    CHANGES FROM (+) OR (-) TRUE DATA.
00787                *    USER PARAMETERS:
00788                *      XR - (+) FOR POS. DATA
00789                *         - (-) FOR NEG. DATA
00790                *
00791 3328 BD 3109 DATA   JSR     INIT
00792 332B 96 91          LDA A   XR+1        LOAD SIGN
00793 332D A7 02          STA A   2,X
00794 332F 20 F1          BRA     EXXT        DONE
00796                *
00797                *    DIGIT
00798                *
00799                *    CHECKS FOR A DIGIT SENT IN ACCA.  RETURNS
00800                *    KEYCODE FROM ASCII.  ALSO LOOKS FOR A DECIMAL
00801                *    IF IT'S THE FIRST ONE.  SETS 'Z' BIT IF VALID.
00802                *
00803 3331 81 2E   DIGIT  CMP A   #$2E
00804 3333 26 0A          BNE     DIG2        NOT A DECIMAL
00805 3335 D6 2B          LDA B   T4
00806 3337 26 05          BNE     DIG1+1      SECOND DECIMAL
00807 3339 86 0B          LDA A   #$B
00808 333B 97 2B          STA A   T4          SET NON-ZERO
00809 333D 5F         DIG1    CLR B
00810 333E 39             RTS
00811 333F 81 39      DIG2    CMP A   #$39
00812 3341 22 FB          BHI     DIG1+1      EXCEEDS UPPER LIMIT
00813 3343 80 2F          SUB A   #$2F
00814 3345 2E F6          BGT     DIG1        VALID DIGIT
00815 3347 8B 2F          ADD A   #$2F
00816 3349 39             RTS


00818                *
00819                *    SIGN
00820                *
00821                *    CHECKS FOR A '+' OR '-', SETS 'Z' BIT.
00822                *
00823 334A 80 2B      SIGN    SUB A   #$2B        '+'?
00824 334C 27 07          BEQ     SIG1        YES
00825 334E 81 02          CMP A   #2          '-'?
00826 3350 26 04          BNE     SIG1+1      NO
00827 3352 86 12          LDA A   #$12        CHS CODE
00828 3354 5F             CLR B
00829 3355 39         SIG1    RTS
00830 3356 8B 2B          ADD A   #$2B        RESTORE CHAR
00831 3358 39             RTS
```

```
00833                    *
00834                    *     READX
00835                    *
00836                    *     READS AN ASCII NUMBER FROM THE DEVICE INTO
00837                    *     THE XR.  STACK OPERATION DEPENDENT UPON
00838                    *     AUTO-STACK FLAG.  NUMBER MUST START WITH A
00839                    *     DIGIT, DECIMAL, OR SIGN.  TERMINATES WITH
00840                    *     LF, SECOND DECIMAL, TWO DIGIT EXPONENT OR
00841                    *     USER DFFINED DELIMETER.  UPON RECEIT OF
00842                    *     USER 'END' DELIMETER, FLAG #4 IS SET.
00843                    *     USER PARAMFTERS:
00844                    *       - NONE
00845                    *
00846 3359 BD 3109 READX  JSR     INIT
00847 335C 4F             CLR A
00848 335D 97 2D          STA A   T2          STATUS FLAG
00849 335F 97 2B          STA A   T4          DECIMAL FLAG
00850 3361 97 2C          STA A   T3          EXPONENT FLAG
00851 3363 97 2A          STA A   T5          NO-LOAD FKAG
00852 3365 4C             INC A
00853 3366 BD 31E7        JSR     BUILD       ENTER A LEADING ZERO
00854 3369 BD 68    RD1   BSR     INBT        INPUT CHAR
00855 336B 2B 18          BMI     RD3A        STOP KEY
00856 336D 4D             TST A
00857 336E 27 F9          BEQ     RD1         NULL CHARACTER
00858 3370 BD 67          BSR     DELM        DELIMETER?
00859 3372 27 55          BEQ     RD8         YES, ABORT NOW
00860 3374 BD D4          BSR     SIGN
00861 3376 27 04          BEQ     RD3         YES
00862 3378 BD B7          BSR     DIGIT
00863 337A 26 ED          BNE     RD1         NOT A DIGIT
00864 337C D6 2A    RD3   LDA B   T5          NO-LOAD?
00865 337E 26 03          BNE     *+5         YES, IGNOR IT
00866 3380 BD 31E7        JSR     BUILD       ENTER DIGIT
00867 3383 BD 51          BSR     INBT        NEXT CHAR
00868 3385 2B 4C    RD3A  BMI     RD10        STOP KEY
00869 3387 7C 002C        INC     T3          COUNT LAST CHAR
00870 338A 27 37          BEQ     RD7         EXPONENT DONE
00871 338C BD 4B          BSR     DELM        DELIMETER?
00872 338E 27 39          BEQ     RD8         YES
00873 3390 BD 9F          BSR     DIGIT       VALID DIGIT?
00874 3392 27 E8          BEQ     RD3         YES
00875 3394 D6 2D          LDA B   T2          LOAD STAT FLAG
00876 3396 2B 17          BMI     RD5         EXPONENT BEGUN
00877 3398 81 45          CMP A   #$45        AN 'E'?
00878 339A 26 27          BNE     RD7         NO
00879 339C 96 92          LDA A   XR+2        MANTISSA=ZERO?
00880 339E 26 03          BNE     RD4-2       NO
00881 33A0 4C             INC A
00882 33A1 97 2A          STA A   T5          SET NO-LOAD
00883 33A3 CA 80          ORA B   #$80
00884 33A5 86 11    RD4   LDA A   #$11        EEX CODE
00885 33A7 D7 2D          STA B   T2
00886 33A9 C6 FD          LDA B   #-3
00887 33AB D7 2C          STA B   T3          COUNTER
00888 33AD 20 CD          BRA     RD3
00889 33AF D6 2C    RD5   LDA B   T3
00890 33B1 C1 FE          CMP B   #-2         LEGAL POSITION
00891 33B3 26 0E          BNE     RD7         NO
00892 33B5 BD 93          BSR     SIGN        A SIGN?
00893 33B7 26 0A          BNE     RD7         NO
00894 33B9 D6 2D          LDA B   T2
00895 33BB C5 20          BIT B   #$20        SECOND SIGN?
00896 33BD 26 14          BNE     RD10        YES
00897 33BF CA 20          ORA B   #$20
00898 33C1 20 E4          BRA     RD4+2       CHS
00899 33C3 81 0D    RD7   CMP A   #$D         A CR?
00900 33C5 26 02          BNE     RD8         NO
00901 33C7 BD 0D          BSR     INBT
00902 33C9 A1 04    RD8   CMP A   4,X
00903 33CB 26 06          BNE     RD10        NOT USER DEFINED 'END'
00904 33CD 96 08    RD9   LDA A   UFLG
00905 33CF 8A 08          ORA A   #$8         SET FLAG FOUR
00906 33D1 97 08          STA A   UFLG
00907 33D3 7E 3182  RD10  JMP     EXIT
```

```
00909 33D6 7E 32DA INBT   JMP    INBYT


00911                *
00912                *    DELM
00913                *
00914                *    LOOKS FOR A DELIMETER IN ACCA.
00915                *
00916 33D9 A1 04   DELM   CMP A  4,X
00917 33DB 27 0A          BEQ    DEL1+2    'END'
00918 33DD A1 05          CMP A  5,X
00919 33DF 27 06          BEQ    DEL1+2    TYPE1
00920 33E1 A1 06          CMP A  6,X
00921 33E3 27 02          BEQ    DEL1+2    TYPE2
00922 33E5 81 0A   DEL1   CMP A  #$A       TEST FOR 'LF'
00923 33E7 39             RTS
00925                *
00926                *    DELIM
00927                *
00928                *    SETS UP USER SPECIFIED DELIMETERS FOR
00929                *    INPUTING NUMBERS (READX).
00930                *    USER PARAMETERS:
00931                *      XR - FLAG SETTING DELIMETER
00932                *      YR - NUMBER DELIMETER #1
00933                *      ZR - NUMBER DELIMETER #2
00934                *
00935 33E8 BD 3109 DELIM  JSR    INIT
00936 33EB CE 0090        LDX    #XR
00937 33EE BD 3198        JSR    RINRY     CONVERT TO BINARY
00938 33F1 26 1E          BNE    DELM1     RANGE ERROR
00939 33F3 DE 22          LDX    T13
00940 33F5 A7 04          STA A  4,X
00941 33F7 CE 009A        LDX    #YR
00942 33FA BD 3198        JSR    RINRY     CONVERT TO BINARY
00943 33FD 26 12          BNE    DELM1     RANGE ERROR
00944 33FF DE 22          LDX    T13
00945 3401 A7 05          STA A  5,X
00946 3403 CE 00A0        LDX    #ZR
00947 3406 BD 3198        JSR    RINRY     CONVERT TO BINARY
00948 3409 26 06          BNE    DELM1     RANGE ERROR
00949 340B DE 22          LDX    T13
00950 340D A7 06          STA A  6,X
00951 340F 20 C2          BRA    RD10
00952 3411 7E 31B5 DELM1  JMP    WR1       SET ERROR, EXIT
```

SYMBOL TABLE

```
ADATA  0000  ACTL   0001  BDATA  0002  BCTL   0003  INPUT  0004  IOIN   0005
ERROR  0006  TGL    0007  UFLG   0008  RSFLG  0009  EOM    000A  EOPM   000B
STKFLG 000D  RND    000E  DIGFLG 000F  W2     0010  W1     0011  SFLG   0012
DCNTR  0013  TP1    0014  TP1S   0015  TP2    0016  TP2S   0017  TP3    0018
TP3S   0019  TP4    001A  TP4S   001B  TP5    001C  TP5S   001D  TP6    001E
TP6S   001F  TP7    0020  TP7S   0021  T13    0022  T12    0023  T11    0024
T10    0025  T9     0026  T8     0027  T7     0028  T6     0029  T5     002A
T4     002B  T3     002C  T2     002D  T1     002E  ISTK   002F  ISTACK 0051
TA     0052  SPGM   0054  EXTRA  0056  BUFF   0058  REAL   006B  IMAG   0070
AT1    0078  AT2    0080  W      0088  XR     0090  YR     0098  ZR     00A0
TR     00A8  LSTX   00B0  BKWRT  00B8  BKKC   00BA  SOL7   00C6  UPP    00C8
UIP    00CA  ALPHA  00CC  IO1    00CD  IO2    00D0  IT7    00D3  FLAG   0005
TPOS   00D6  FILE   00D7  AR     00D8  BR     00E0  CR     00E8  DR     00F0
ER     00FB  SDBB   00BA  MT     7E00  TERMN7 003D  IMED   0040  PARCD  00C0
PAREX  0080  NTBL   0000  DOTS   5EC0  PRTORV 602D  FRMT   5CA8  BLANK  5D75
LDMSG  57BD  ROLLD  55B2  ROLLU  57F1  PSD    55DA  TXL    55E9  STKUP  55EF
MAD    749B  CMP    74AA  NOR    74D6  TXW    7424  TXXR   7438  EXXR   7452
APSR   7538  OVUNF  75B6  OVERF  75DD  XR0    740A  XRNINE 75C8  UNDRF  75F1
IMULT  7689  QDG    7669  FPA    75FC  FPS    75F6  FPM    7735  FPD    7793
FPAEX  763D  FPMEX  7780  LSHIFT 7521  ZEROX  7489  XZERO0 7416  XZERO2 7417
RECIP  73E6  TXRX   73F3  CONST  6800  FPDBRC  6898  TAN    68A9  ATN    69C3
DSZERO 6A46  NTLN   6A58  EXPN   6AC9  SIN    6894  COS    689A  ASIN   68F2
ACOS   68F7  PH1    6C5D  PH2    6CAD  PH3    6D34  PH4    6DD0  LSFT8  6E47
SQRT   6E65  MAD8   6F2C  CMPR   53E4  IOUPX  6F52  LOG10  6FA7  YUPX   6FE9
RTOP   7328  PTOR   7386  TSFR   49CE  ELIN   3636  PWRUP  3084  PWR1   308F
```

```
PWR2    30A6  PWR3    30B2  PWR4    30C0  PWR5    30D5  RDID    30D9  NULL    30E2
SETUP   30E3  INIT    3109  IN1     3110  IN2     3124  IN3     312E  IN4     3134
IN4A    3149  IN4B    3166  IN5     316A  FIELD   3172  EXIT    3182  EX1     3189
BINRY   3198  BIN2    31A8  WBYTF   31AB  WB1     3195  EROR    3187  WB4     3188
WB5     318E  RBYTE   31C0  BTDEC   31C8  RB1     31CA  RB2     3107  BUILD   31E7
PRNT    31F6  PRNTX   31FA  PR0     31FE  PR1     320A  PR2     3222  PR4     3230
PR5     323D  PR6     3241  PR7     324C  PR9     325E  PR10    3263  PRINT   3266
OTBYT   3270  OTB1    327D  FFLAG   3286  CHECK   328F  CH1     3296  CH2     329F
STOP    32A8  ST1     32BA  ST2     32C6  HNDSK   32C7  HND1    32D0  INBYT   32DA
IN80    32EB  IN81    32F5  LATCH   32FE  LAT1    3306  FLG     330D  EXXT    3322
DATA    3328  DIGIT   3331  DIG1    333D  DIG2    333F  SIGN    334A  SIG1    3355
READX   3359  RD1     3369  RD3     337C  RD3A    3385  RD4     33A5  RD5     33AF
RD7     33C3  RD8     33C9  RD9     33CD  RD10    33D3  INBT    33D6  DELM    33D9
DEL1    33E5  DELIM   33E8  DELM1   3411

TOTAL ERRORS   1

****ERROR 201
 4  NAM CJBPG

 00216                      OPT    LIST,MEM
 00219 3418                 ORG    $3418


 00221                *
 00222                *     EQUATE TABLE
 00223                *

 00225       3109     INIT   EQU    $3109
 00226       3134     IN4    EQU    $3134
 00227       3182     EXIT   EQU    $3182
 00228       3198     BINRY  EQU    $3198
 00229       3185     WB1    EQU    $3185
 00230       31C8     BTDEC  EQU    $31C8
 00231       31F6     PRNT   EQU    $31F6
 00232       3270     OTBYT  EQU    $3270
 00233       32DA     INBYT  EQU    $32DA
 00234       6489     CHADRS EQU    $6489
 00235       59D9     GIONTR EQU    $59D9
 00236       538F     KILL   EQU    $538F
 00238                *
 00239                *     BLIST
 00240                *
 00241                *     LISTS THE PROGRAM IN MACHINE CODE FROM STARTING
 00242                *     ADDRESS (IN XR) TO AN END CODE TO EOPM.
 00243                *     USER MEMORY.  CHECKS FOR PROTECTED MEMORY.
 00244                *     TAPE:  START,FILE LENGTH,FILE,CHECKSUM.
 00245                *     USER PARAMETERS:
 00246                *       XR = STARTING ADDRESS
 00247                *
 00248 3418 BD 3109 BLIST  JSR    INIT
 00249 341B BD 55          BSR    LEGL      LEGAL CALL?
 00250 341D CE 0000        LDX    #0
 00251 3420 DF 18          STX    TP3
 00252 3422 DF 16          STX    TP2       ZERO
 00253 3424 BD 58          BSR    PADRS     STARTING ADDRESS
 00254 3426 A6 B1          LDA A  #$B1
 00255 3428 7C 0019 BL1    INC    TP3S      COUNT RECORD LENGTH
 00256 342B 26 03          BNE    BL1+8
 00257 342D 7C 0018        INC    TP3
 00258 3430 A1 00          CMP A  0,X       CHECK THE ISNTR
 00259 3432 27 05          BEQ    BL2       END CODE
 00260 3434 08             INX
 00261 3435 9C 0B          CPX    EOPM
 00262 3437 26 EF          BNE    BL1       CONTINUE
 00263 3439 86 C3    BL2   LDA A  #$C3
 00264 343B BD 32          BSR    OTBT1     OUT START CODE
 00265 343D 96 18          LDA A  TP3
 00266 343F BD 2E          BSR    OTBT1     OUT RECORD LENGTH
 00267 3441 96 19          LDA A  TP3S
 00268 3443 BD 2A          BSR    OTBT1
 00269 3445 DE 20    BL3.  LDX    TP7       LOAD ADDRESS
```

```
00270 3447 A6 00          LDA A  0,X      NOW THE INSTRUCTION
00271 3449 5F             CLR B
00272 344A 9B 17          ADD A  TP2S     CHECKSUM
00273 344C D9 16          ADC B  TP2
00274 344E 97 17          STA A  TP2S
00275 3450 D7 16          STA B  TP2
00276 3452 A6 00          LDA A  0,X      BYTE AGAIN
00277 3454 8D 19          BSR    OTBT1    OUT IT
00278 3456 2B 14          BMI    EXT
00279 3458 DE 20          LDX    TP7
00280 345A 08             INX             BUMP POINTER
00281 345B DF 20          STX    TP7
00282 345D DE 18          LDX    TP3
00283 345F 09             DEX             COUNT INSTRUCTIONS
00284 3460 DF 18          STX    TP3
00285 3462 26 E1          BNE    BL3      NOT DONE
00286 3464 96 16   BL6     LDA A  TP2
00287 3466 8D 07          BSR    OTBT1
00288 3468 96 17          LDA A  TP2S
00289 346A 8D 03          BSR    OTBT1
00290 346C 7E 3182 EXT    JMP    EXIT
00292 346F 7E 3270 OTBT1  JMP    OTBYT



00294                *
00295                *    LEGL
00296                *
00297 3472 96 D5   LEGL   LDA A  FLAG
00298 3474 46            ROR A
00299 3475 24 1F         BCC    PAD1     NOT DECURED
00300 3477 86 16   LEG1   LDA A  #22
00301 3479 31            INS
00302 347A 31            INS             BUMP STACK
00303 347B 7E 31B7       JMP    WB1+2    SET ERROR, EXIT



00305                *
00306                *    PADRS
00307                *
00308 347E CE 0090 PADRS  LDX    #XR
00309 3481 BD 319C        JSR    RINRY+4   CONVERT TO BINARY
00310 3484 D6 90          LDA B  XR       EXPONENT
00311 3486 C1 03          CMP B  #3       WITHIN RANGE?
00312 3488 2E 0D          BGT    PAD1+1   NO, ILLEGAL
00313 348A D6 20          LDA B  TP7      MSP
00314 348C 5C             INC B
00315 348D D7 20          STA B  TP7
00316 348F DE 20          LDX    TP7      LOAD ADDRESS
00317 3491 BD 6489        JSR    CHADRS   CHECK AGAINST EOPM
00318 3494 2A 01          BPL    PAD1+1   ILLEGAL
00319 3496 39      PAD1   RTS
00320 3497 86 18          LDA A  #24
00321 3499 20 DE          BRA    LEG1+2   SET ERROR
00323                *
00324                *    ASCII
00325                *
00326                *    FORMATS INSTRUCTION IN OUTPUT BUFFER
00327                *    CHECKS FOR SPECIAL CASES AND FIXED ILLEGAL CHARS.
00328                *
00329 349B DE CA   ASCII  LDX    UPP
00330 349D A6 00          LDA A  0,X      LOAD INSTR
00331 349F 97 68          STA A  REAL
00332 34A1 4F             CLR A
00333 34A2 BD 59D9        JSR    GIONTR   BUILD ASCII
00334 34A5 5F             CLR B
00335 34A6 96 CC          LDA A  ALPHA    ASCII STRING?
00336 34A8 26 1F          BNE    ASC1A    YES
00337 34AA 96 68          LDA A  REAL
00338 34AC CE 5049        LDX    #$5049
00339 34AF 81 17          CMP A  #$17
00340 34B1 26 02          BNE    ASC1     NOT PI
00341 34B3 DF 5D          STX    BUFF+5
00342 34B5 81 12   ASC1   CMP A  #$12
```

```
00343 34B7 27 44          BEQ    INSRT+4   CHS
00344 34B9 81 15          CMP A  #$15
00345 34BB 27 3F          BEQ    INSRT+3   XEY
00346 34BD 81 34          CMP A  #$34
00347 34BF 27 3A          BEQ    INSRT+2   X=Y?
00348 34C1 81 35          CMP A  #$35
00349 34C3 27 35          BEQ    INSRT+1   X<Y?
00350 34C5 81 36          CMP A  #$36
00351 34C7 27 30          BEQ    INSRT     X>=Y?
00352 34C9 CE 005D ASC1A  LDX    #BUFF+5
00353 34CC 86 0A          LDA A  #10
00354 34CE 97 14          STA A  TP1       COUNTER
00355 34D0 A6 00    ASC2  LDA A  0,X
00356 34D2 81 5B          CMP A  #$5B
00357 34D4 27 1C          BEQ    ASC3+5    DIVIDE
00358 34D6 81 5C          CMP A  #$5C
00359 34D8 27 1A          BEQ    ASC3+7    RIGHT ARROW
00360 34DA 81 5F          CMP A  #$5F
00361 34DC 27 10          BEQ    ASC3+1    DOWN ARROW
00362 34DE 81 68          CMP A  #$68
00363 34E0 27 0E          BEQ    ASC3+3    'ALPHA'
00364 34E2 81 22          CMP A  #$22
00365 34E4 27 07          BEQ    ASC3      LOWER CASE 'E'
00366 34E6 08             INX
00367 34E7 7A 0014        DEC    TP1
00368 34EA 26 E4          BNE    ASC2
00369 34EC 39             RTS
00370 34ED 5C       ASC3  INC B             STUFF 'E'
00371 34EE CB 04          ADD B  #4         STUFF 'D'
00372 34F0 CB 11          ADD B  #$11       STUFF '@'
00373 34F2 CB 02          ADD B  #2         STUFF '/'
00374 34F4 CB 2D          ADD B  #$2D       STUFF '-'
00375 34F6 E7 00          STA B  0,X
00376 34F8 39             RTS
00378                *
00379                *    INSRT
00380                *
00381                *    INSERTS EXCEPTION ASCII FOR LISTINGS
00382                *
00383 34F9 5C       INSRT INC B
00384 34FA 5C             INC B
00385 34FB 5C             INC B
00386 34FC 5C             INC B
00387 34FD CE 37DF        LDX    #LTAB     TALE BASE ADDRESS
00388 3500 DF 16          STX    TP2
00389 3502 17             TBA
00390 3503 48             ASL A
00391 3504 1B             ABA
00392 3505 48             ASL A
00393 3506 9B 17          ADD A  TP2+1
00394 3508 97 17          STA A  TP2+1     SET POINTER
00395 350A CE 005D        LDX    #BUFF+5
00396 350D DF 14          STX    TP1
00397 350F C6 06          LDA B  #6
00398 3511 DE 16    INS1  LDX    TP2
00399 3513 A6 00          LDA A  0,X
00400 3515 08             INX
00401 3516 DF 16          STX    TP2
00402 3518 DE 14          LDX    TP1
00403 351A A7 00          STA A  0,X
00404 351C 08             INX
00405 351D DF 14          STX    TP1
00406 351F 5A             DEC B
00407 3520 26 EF          BNE    INS1
00408 3522 39             RTS


00410                *
00411                *    LABL
00412                *
00413                *    FORMULATES THE SECOND HALF OF A LABEL
00414                *    FOR LISTING ROUTINES.  DOES BOTH ALPHA
00415                *    AND NUMBER LABELS.
00416                *
```

```
00417 3523 86 2D    LABL   LDA A  #$2D      DASH
00418 3525 8D 1F           BSR    BLNK
00419 3527 DE C8           LDX    UPP
00420 3529 A6 00           LDA A  0,X       INSTRUCTION
00421 352B 80 63           SUB A  #$63      SUBTRACT ASCII BIAS
00422 352D 23 05           BLS    LAB2      A NUMBER LABEL
00423 352F 8B 40           ADD A  #$40      RESTORE
00424 3531 97 5D           STA A  BUFF+5
00425 3533 39              RTS
00426 3534 8B 63    LAB2   ADD A  #$63      RESTORE BIAS
00427 3536 5F              CLR B
00428 3537 5A              DEC B
00429 3538 5C       LAB3   INC B
00430 3539 80 0A           SUB A  #10       TENS
00431 353B 2C FB           BGE    LAB3
00432 353D 8B 3A           ADD A  #$3A
00433 353F CB 30           ADD B  #$30
00434 3541 D7 5D           STA B  BUFF+5
00435 3543 97 5E           STA A  BUFF+6
00436 3545 39              RTS


00438                *
00439                *   BLNK
00440                *
00441                *    STUFFS BLANKS IN THE OUTPUT BUFFER, EXCEPT
00442                *    THE FIRST FOUR CHARACTERS WHICH ARE
00443                *    SPECIFIED BY PASSING THE DESIRED CHARACTER
00444                *    IN ACCA.
00445                *
00446 3546 BD 5D75 BLNK   JSR    BLANK     CLEAR BUFFER
00447 3549 97 58          STA A  BUFF
00448 354B 97 59          STA A  BUFF+1    SPECIAL CHAR
00449 354D 97 5A          STA A  BUFF+2
00450 354F 97 5B          STA A  BUFF+3
00451 3551 39       BLN1  RTS


00453                *
00454                *    LIST
00455                *
00456                *    LISTS THE PROGRAM ON THE PERIPHERIAL IN FOUR
00457                *    COLUMNS, FIFTY LINES PER COLUMN.
00458                *    ROUTINE STOPS AFTER EACH PAGE.
00459                *    -NOTE:  PROGRAM POINTER MAY BE RETURNED
00460                *    ON AN ILLEGAL ADDRESS BOUNDRY.
00461                *    USER PARAMETERS:
00462                *      - NONE
00463                *
00464 3552 BD 3109 LIST   JSR    INIT
00465 3555 96 B2          LDA A  AT2+2     RUNNING?
00466 3557 27 03          BEQ    COL0-3    NO
00467 3559 7E 3633        JMP    XIT       YES, ABORT
00468 355C BD 3472        JSR    LEGL      SECURED?
00469 355F 5F       COL0  CLR B
00470 3560 DE C8          LDX    UPP
00471 3562 A6 00    COL1  LDA A  0,X       INSTRUCTION
00472 3564 5C             INC B
00473 3565 81 B1          CMP A  #$B1
00474 3567 27 0A          BEQ    COL2      END CODE
00475 3569 C1 C8          CMP B  #200
00476 356B 27 06          BEQ    COL2      200 STATES
00477 356D 08             INX
00478 356E 9C 0B          CPX    EOPM
00479 3570 26 F0          BNE    COL1      NO OVERFLOW
00480 3572 09             DEX              BACK UP
00481 3573 D7 6F    COL2  STA B  REAL+7     TOTAL COUNT TO OUT
00482 3575 DF 74          STX    IMAG+4    LAST INSTRUCTION ADDRESS
00483 3577 DE C8          LDX    UPP
00484 3579 09             DEX
00485 357A DF 70          STX    IMAG      COLUMN ONE POINTER
00486 357C 4F       COL6  CLR A
00487 357D 97 6D          STA A  REAL+5    COLUMN FLAG
```

```
00488 357F DE 70          LDX    IMAG
00489 3581 08             INX
00490 3582 DF 70          STX    IMAG       BUMP COLUMN ONE
00491 3584 DF 72          STX    IMAG+2     CUPENT BACK TO ONE
00492 3586 DE 72   COL7   LDX    IMAG+2     GET CURRENT POINTER
00493 3588 DF C8          STX    UPP        SET INSTR POINTER
00494 358A 09             DEX
00495 358B A6 00          LDA A  0,X        PREVIOUS BYTE
00496 358D 81 BA          CMP A  #$BA
00497 358F 23 12          BLS    COL8       SINGLE BYTE INSTR.
00498 3591 97 6C   COL7A  STA A  REAL+4     SAVE
00499 3593 86 2A          LDA A  #$2A        '*'
00500 3595 BD 3546        JSR    BLNK
00501 3598 96 6C          LDA A  REAL+4
00502 359A 81 BF          CMP A  #$BF
00503 359C 26 2A          BNE    COL9       NOT A LABEL
00504 359E BD 3523        JSR    LABL       LABEL
00505 35A1 20 25          BRA    COL9
00506 35A3 7F 00CC COL8   CLR    ALPHA      CLEAR ALPHA FLAG
00507 35A6 A6 01          LDA A  1,X
00508 35A8 81 BE          CMP A  #$BE
00509 35AA 27 19          BEQ    COL8B      FORMAT THIS ONE, OK
00510 35AC A6 01   COL8A  LDA A  1,X        LOAD INSTR
00511 35AE 8C 00FE        CPX    #$FE
00512 35B1 27 12          BEQ    COL8B      START OF PROGRAM, OK
00513 35B3 81 B4          CMP A  #$B4
00514 35B5 27 0E          BEQ    COL8B      ALPHA TERMINATOR, OK
00515 35B7 09             DEX               BACK UP TO PREVIOUS
00516 35B8 81 BE          CMP A  #$BE
00517 35BA 26 F0          BNE    COL8A      NOT A FORMAT, AGAIN
00518 35BC A6 03          LDA A  3,X        SECOND BYTE OF CALL INSTR
00519 35BE 85 0F          BIT A  #$F
00520 35C0 26 03          BNE    COL8B      NOT AN ALPHA STRING
00521 35C2 7C 00CC        INC    ALPHA      SET ALPHA FLAG
00522 35C5 BD 3498 COL8B  JSR    ASCII      INSTRUCTION ASCII
00523 35C8 8D 75   COL9   BSR    STNG       OUT ENTIRE BUFFER
00524 35CA 96 72          LDA A  IMAG+2
00525 35CC D6 73          LDA B  IMAG+3     CURRENT POINTER
00526 35CE CB 32          ADD B  #50        ADD COLUMN LENGTH
00527 35D0 89 00          ADC A  #0         CARRY IF NEEDED
00528 35D2 97 72          STA A  IMAG+2     NEW POINTER
00529 35D4 D7 73          STA B  IMAG+3
00530 35D6 7A 006F        DEC    REAL+7
00531 35D9 27 28          BEQ    COL10A     DONE WITH PAGE
00532 35DB 7C 006D        INC    REAL+5
00533 35DE 96 6D          LDA A  REAL+5
00534 35E0 81 04          CMP A  #4
00535 35E2 27 1D          BEQ    COL10      DONE WITH LINE
00536 35E4 96 72          LDA A  IMAG+2
00537 35E6 91 74          CMP A  IMAG+4     UPPER HALF OF POINTER
00538 35E8 25 0A          BCS    COL9A      LIMIT OK
00539 35EA 26 15          BNE    COL10      TOO FAR, PARTIAL LINE
00540 35EC 96 73          LDA A  IMAG+3
00541 35EE 91 75          CMP A  IMAG+5     CHECK LOWER HALF
00542 35F0 27 02          BEQ    COL9A      WITHIN BOUNDS
00543 35F2 24 0D          BCC    COL10      PARTIAL LINE OUT
00544 35F4 86 20   COL9A  LDA A  #$20
00545 35F6 8D 44          BSR    OTBT2
00546 35F8 86 20          LDA A  #$20       TWO BLANKS
00547 35FA 8D 40          BSR    OTBT2
00548 35FC 2B 21          BMI    COL11      STOP KEY
00549 35FE 7E 3586        JMP    COL7       NEXT COLUMN
00550 3601 8D 33   COL10  BSR    ELIN       END OF LINE
00551 3603 7E 357C        JMP    COL6       NEXT LINE
00552 3606 8D 2E   COL10A BSR    ELIN       EDN OF LINE
00553 3608 DE 74          LDX    IMAG+4
00554 360A A6 00          LDA A  0,X        LAST CODE
00555 360C 81 B1          CMP A  #$B1
00556 360E 27 0F          BEQ    COL11      END CODE, EXIT
00557 3610 08             INX
00558 3611 DF C8          STX    UPP        SET NEW LIST START
00559 3613 9C 0B          CPX    EOPM       END OF MEMORY?
00560 3615 27 16          BEQ    COL13-1     YES
00561 3617 3E             WAI               WAIT FOR CONTINUE KEY
00562 3618 96 09          LDA A  RSFLG      STOP KEY?
```

```
00563 361A 2A 03          BPL    COL11     YES
00564 361C 7E 355F        JMP    COL0      NEXT PAGE
00565 361F DE 74   COL11  LDX    IMAG+4
00566 3621 A6 00          LDA A  0,X
00567 3623 81 BA          CMP A  #$BA
00568 3625 23 01          BLS    COL12     ONE BYTE INSTRUCTION
00569 3627 08             INX
00570 3628 08      COL12  INX
00571 3629 9C 08          CPX    EOPM      END OF MEMORY
00572 362B 26 01          BNE    COL13
00573 362D 09             DEX               YES
00574 362E DF C8   COL13  STX    UPP       SET PROGRAM POINTER
00575 3630 7F 00CC        CLR    ALPHA
00576 3633 7E 3182 XIT    JMP    EXIT      DONE


00578 3636 86 0D   ELIN   LDA A  #$D       CR
00579 3638 8D 02          BSR    OTBT2
00580 363A 86 0A          LDA A  #$A       LF
00581 363C 7E 3270 OTBT2  JMP    OTBYT
00583              *
00584              *     STNG
00585              *
00586              *     OUTPUTS THE ENTIRE BUFFER TO PERIPHERIAL.
00587              *
00588 363F CE 0058 STNG   LDX    #BUFF
00589 3642 DF 14          STX    TP1
00590 3644 A6 00   STN1   LDA A  0,X
00591 3646 08             INX
00592 3647 DF 14          STX    TP1
00593 3649 8D F1          BSR    OTBT2     OUT CHAR
00594 364B DE 14          LDX    TP1
00595 364D 8C 0068        CPX    #BUFF+16   DONE?
00596 3650 26 F2          BNE    STN1      NO
00597 3652 39             RTS


00599              *
00600              *     READB
00601              *
00602              *     READS BINARY TAPE OF PROGRAM INTO MACHINE.  FORMAT
00603              *     SAME OF BINARY LIST.
00604              *     USER PARAMETERS:
00605              *       - NONE
00606              *
00607 3653 BD 3109 READB  JSR    INIT      INITIALIZE
00608 3656 CE 0000        LDX    #0
00609 3659 DF 18          STX    TP3
00610 365B BD 347E        JSR    PADRS     STARTING ADDRESS
00611 365E 8D 46   BRD2   BSR    INBT1
00612 3660 2B 41          BMI    BRD6      STOP KEY
00613 3662 81 C3          CMP A  #$C3
00614 3664 26 F8          BNE    BRD2      LOOK FOR START CODE
00615 3666 8D 3E          BSR    INBT1
00616 3668 97 16          STA A  TP2       SET RECORD LENGTH
00617 366A 8D 3A          BSR    INBT1
00618 366C 97 17          STA A  TP2S
00619 366E 8D 36   BRD3   BSR    INBT1
00620 3670 2B 29          BMI    BRD4A-4    STOP KEY
00621 3672 DE 20          LDX    TP7
00622 3674 9C 08          CPX    EOPM
00623 3676 27 27          BEQ    BRD4A
00624 3678 A7 00          STA A  0,X       STUFF CODE
00625 367A 08             INX               BUMP
00626 367B DF 20          STX    TP7
00627 367D 5F             CLR B
00628 367E 9B 19          ADD A  TP3S
00629 3680 D9 18          ADC B  TP3       CALCULATE CHECKSUM
00630 3682 97 19          STA A  TP3S
00631 3684 D7 18          STA B  TP3
00632 3686 DE 16          LDX    TP2
00633 3688 09             DEX               COUNT RECORD LENGTH
00634 3689 DF 16          STX    TP2
00635 368B 26 E1          BNE    BRD3      NOT DONE YET
```

```
00636 368D BD 17    BRD4    BSR     INBT1
00637 368F 97 1A            STA A   TP4         GET DEVICE CHECKSUM
00638 3691 BD 13            BSR     INBT1
00639 3693 97 1B            STA A   TP4S
00640 3695 DE 1A            LDX     TP4
00641 3697 9C 18            CPX     TP3         COMPARE THEM
00642 3699 27 08            BEQ     BRD6        NO ERROR
00643 369B 86 0E            LDA A   #14
00644 369D 20 02            BRA     BRD4A+2
00645 369F 86 07    BRD4A   LDA A   #7
00646 36A1 97 06            STA A   ERROR       MEMORY OVERFLOW
00647 36A3 7E 3182  BRD6    JMP     EXIT        DONE


00649 36A6 7E 32DA  INBT1   JMP     INBYT


00651                   *
00652                   *     STRING
00653                   *
00654                   *     OUTPUTS AN ASCII STRING TO PERIPHERIAL.  MAY
00655                   *     CONTAIN PRINT KEY CODES FOR OUTING THE XR.
00656                   *     USER PARAMETERS:
00657                   *       - NONE
00658                   *
00659 36A9 BD 3109  STRING  JSR     INIT
00660 36AC 96 68            LDA A   REAL        POSSIBLE CHARACTER
00661 36AE D6 82            LDA B   AT2+2
00662 36B0 26 06            BNE     STR0        PROGRAM EXECUTION
00663 36B2 81 B4            CMP A   #$B4        TERMINATOR?
00664 36B4 27 26            BEQ     STR5        YES
00665 36B6 20 0A            BRA     STR1A       KEYBOARD ENTRY
00666 36B8 DE CA    STR0    LDX     UIP
00667 36BA 08               INX                 STRING POINTER
00668 36BB 08               INX
00669 36BC DF 69            STX     REAL+1
00670 36BE BD 22    STR1    BSR     GETCR       GET A CHAR
00671 36C0 27 16            BEQ     STR4        END OF STRING
00672 36C2 81 B2    STR1A   CMP A   #$B2
00673 36C4 26 05            BNE     STR2        NOT A PRINT
00674 36C6 BD 31F6          JSR     PRNT        OUT XR
00675 36C9 20 02            BRA     STR2A
00676 36CB BD 24    STR2    BSR     OUTCR       OUT IT
00677 36CD 96 82    STR2A   LDA A   AT2+2
00678 36CF 27 D2            BEQ     BRD6        KEYBOARD ENTRY
00679 36D1 5D               TST B
00680 36D2 27 EA            BEQ     STR1        CONTINUE
00681 36D4 BD 0C    STR3    BSR     GETCR       END OF STRING?
00682 36D6 26 FC            BNE     STR3        NOT AN END CODE
00683 36D8 09       STR4    DEX
00684 36D9 09               DEX                 FIX POINTER
00685 36DA DF CA            STX     UIP         SET INSTRUCTION POINTER
00686 36DC DE 22    STR5    LDX     T13
00687 36DE 6F 07            CLR     7,X         CLEAR SHIFT KEY
00688 36E0 20 C1            BRA     BRD6        DONE

00690                   *
00691                   *     GETCR
00692                   *
00693 36E2 DE 69    GETCR   LDX     REAL+1
00694 36E4 A6 00            LDA A   0,X
00695 36E6 08               INX
00696 36E7 DF 69            STX     REAL+1
00697 36E9 81 B4            CMP A   #$B4
00698 36EB 27 03            BEQ     GET1
00699 36ED 09               DEX
00700 36EE 9C 08            CPX     EOPM
00701 36F0 39       GET1    RTS

00703                   *
00704                   *     OUTCR
00705                   *
00706 36F1 DE 22    OUTCR   LDX     T13
```

```
00707 36F3 81 03          CMP A  #3        SHIFT KEY?
00708 36F5 26 08          BNE    OUT1      NO
00709 36F7 86 80          LDA A  #$80
00710 36F9 A8 07          EOR A  7,X       TOGGLE SHIFT BIT
00711 36FB A7 07          STA A  7,X
00712 36FD 5F             CLR B
00713 36FE 39             RTS
00714 36FF 81 02   OUT1   CMP A  #2        LINE?
00715 3701 26 03          BNE    OUT2      NO
00716 3703 7E 3636        JMP    ELIN
00717 3706 81 01   OUT2   CMP A  #1        TAB?
00718 3708 26 02          BNE    OUT2A     NO
00719 370A 8B 08          ADD A  #8
00720 370C 6D 07   OUT2A  TST    7,X       SHIFTED KEYBORD?
00721 370E 2A 18          BPL    OUT6      NO
00722 3710 81 40          CMP A  #@100     LETTER?
00723 3712 23 04          BLS    OUT3      NO
00724 3714 81 5A          CMP A  #@132     LETTER?
00725 3716 23 11          BLS    OUT5+2    YES
00726 3718 CE 378F OUT3   LDX    #TBL      TABLE ADDRESS
00727 371B C6 10          LDA B  #16       TABLE SIZE
00728 371D A1 00   OUT4   CMP A  0,X       FOUND MATCH?
00729 371F 27 06          BEQ    OUT5      YES
00730 3721 08             INX
00731 3722 08             INX              NEXT ENTRY
00732 3723 5A             DEC B            END OF TABLE?
00733 3724 26 F7          BNE    OUT4      NO
00734 3726 39             RTS
00735 3727 A6 01   OUT5   LDA A  1,X       LOAD SHIFTED CHAR
00736 3729 8B 20          ADD A  #@40      ADD BIAS
00737 372B 7E 3270 OUT6   JMP    OTBYT     OUT CHARACTER
00739                     *
00740                     *     AND - OR
00741                     *
00742                     *     PREFORMS THE BINARY OPERATION ON THE X AND Y REG.
00743                     *     CONVERTS TO BINARY, OPERATES, AND BACK TO DECIMAL.
00744                     *     USER PARAMETERS:
00745                     *       XR - NUMBER (0<=#<256)
00746                     *       YR - NUMBER (0<=#<256)
00747                     *
00748 372E 4C       ANDXY INC A
00749 372F 97 2E    ORXY  STA A  T1        SET OPERATION FLAG
00750 3731 BD 3109        JSR    INIT
00751 3734 CE 0090        LDX    #XR
00752 3737 BD 3198        JSR    BINRY     CONVERT XR TO BINARY
00753 373A 26 21          BNE    AN3       ERROR
00754 373C 97 2D          STA A  T2        SAVE
00755 373E CE 0098        LDX    #YR
00756 3741 BD 3198        JSR    BINRY     CONVERT YR TO BINARY
00757 3744 26 17          BNE    AN3       ERROR
00758 3746 97 2C          STA A  T3        SAVE
00759 3748 BD 55DA        JSR    PSD       FIX STACK
00760 374B 96 2D          LDA A  T2        ONE OPERAND
00761 374D D6 2E          LDA B  T1        FLAG
00762 374F 26 04          BNE    AN1       AND OPERATION
00763 3751 9A 2C          ORA A  T3
00764 3753 20 02          BRA    AN1+2
00765 3755 94 2C    AN1   AND A  T3
00766 3757 7F 000D        CLR    STKFLG
00767 375A 7E 31C8        JMP    BTDEC     BACK TO DECIMAL
00768 375D 7E 3185 AN3    JMP    WB1       ERROR, EXIT
00770                     *
00771                     *     ROT
00772                     *
00773                     *     PEFORMS A RIGHT ROTATE ON THE BINARY EQUIVALENT
00774                     *     OF THE NUMBER IN THE XR.
00775                     *     USER PARAMETERS:
00776                     *       XR - NUMBER (0<=#<256)
00777                     *
00778 3760 BD 3109 ROTX   JSR    INIT
00779 3763 CE 0090        LDX    #XR
00780 3766 BD 3198        JSR    BINRY     CONVERT XR TO BINARY
00781 3769 26 F2          BNE    AN3       ERROR
00782 376B 44             LSR A
00783 376C 24 02          BCC    ROT1
```

```
00784 376E 8A 80          ORA A  #$80
00785 3770 20 E5   ROT1   BRA    AN1+2


00787                *
00788                *     INTAP
00789                *
00790                *     ROUTINE CALED BY MAINFRAME FOR LOADING
00791                *     SPECIAL PROGRAMS FROM PAPER TAPE.
00792                *
00793 3772 BD 3134 INTAP  JSR    IN4       CHANNEL A!!
00794 3775 7F 0021        CLR    TP7+1     CLEAR CHECKSUM
00795 3778 8D 2F   INT1   BSR    INCHR     READ CHAR
00796 377A 97 1E          STA A  TP6       SET PAGE
00797 377C 97 54          STA A  SPGM      NULL?
00798 377E 27 F8          BEQ    INT1      YES. LEADER
00799 3780 8D 27          BSR    INCHR
00800 3782 97 1F          STA A  TP6+1     PROGRAM ADDRESS
00801 3784 97 55          STA A  SPGM+1
00802 3786 8D 21   INT2   BSR    INCHR
00803 3788 97 20          STA A  TP7       BLOCK COUNT. ZERO?
00804 378A 27 1A          BEQ    INT5      YES. END OF TAPE
00805 378C 8D 1B   INT3   BSR    INCHR     READ INSTR CODE
00806 378E DE 1E          LDX    TP6
00807 3790 A7 00          STA A  0,X       STUFF MPU INSTR
00808 3792 08             INX
00809 3793 DF 1E          STX    TP6
00810 3795 7A 0020        DEC    TP7       DONE WITH BLOCK?
00811 3798 26 F2          BNE    INT3      NO
00812 379A 8D 0D          BSR    INCHR     READ CHECKSUM
00813 379C 5D             TST B            ERROR?
00814 379D 27 E7          BEQ    INT2      NO
00815 379F BD 538F INT4   JSR    KILL      ERASE MEMORY
00816 37A2 86 0E          LDA A  #14
00817 37A4 97 06          STA A  ERROR     SET ERROR CODE
00818 37A6 7E 3182 INT5   JMP    EXIT      DONE
00820                *
00821                *     INCHR
00822                *
00823 37A9 BD 32DA INCHR  JSR    INBYT     INPUT A BYTE
00824 37AC 2B 06          BMI    INC1      STOP KEY. ABORT
00825 37AE 16             TAB
00826 37AF DB 21          ADD B  TP7+1     UPDATE CHECKSUM
00827 37B1 D7 21          STA B  TP7+1
00828 37B3 39             RTS
00829 37B4 31      INC1   INS              BUMP RETURN STACK
00830 37B5 31             INS
00831 37B6 20 E7          BRA    INT4      OUT ERROR
00833 378F                ORG    $378F
00834                *
00835                *     SHIFTED CHARACTER TABLE
00836                *
00837                     OPT    NG
00838 378F 20      TBL    FCB    @40,@40-@40
00839 37C1 23             FCB    @43,@137-@40
00840 37C3 24             FCB    @44,@134-@40
00841 37C5 25             FCB    @45,@174-@40
00842 37C7 27             FCB    @47,@42-@40
00843 37C9 28             FCB    @50,@133-@40
00844 37CB 29             FCB    @51,@135-@40
00845 37CD 2C             FCB    @54,@140-@40
00846 37CF 2E             FCB    @56,@12-@40
00847 37D1 30             FCB    @60,@15-@40
00848 37D3 3A             FCB    @72,@73-@40
00849 37D5 3C             FCB    @74,@173-@40
00850 37D7 3D             FCB    @75,@176-@40
00851 37D9 3E             FCB    @76,@175-@40
00852 37DB 3F             FCB    @77,@41-@40
00853 37DD 40             FCB    @100,@46-@40
00854                     OPT    G
00856                *
00857                *     LIST EXCEPTION TABLE
00858                *
00859                     OPT    NG
```

```
00860 370F 43       LTAB    FCB     $43,$48,$52,$20,$20,$20    CHS
00861 37E5 58               FCB     $58,$45,$59,$20,$20,$20    XEY
00862 37EB 58               FCB     $58,$3D,$59,$20,$3F,$20    X=Y ?
00863 37F1 58               FCB     $58,$3C,$59,$20,$3F,$20    X<Y ?
00864 37F7 58               FCB     $58,$3E,$3D,$59,$20,$3F    X>=Y ?
00865                       OPT     G


00867                 *
00868                 *     ENTRY FOR SPECIAL PROGRAM LOADING.
00869                 *
00870 37FD                  ORG     $37FD
00871 37FD 7E 3772          JMP     INTAP


00873                 *
00874                 *     ADDRESS TABLE
00875                 *
00876 3005                  ORG     $3005
00877 3005 36A9             FDB     STRING
00878 3015                  ORG     $3015
00879 3015 372E             FDB     ANDXY
00880 3017 372F             FDB     ORXY
00881 3019 3760             FDB     ROTX
00882 301B 3552             FDB     LIST
00883 301D 3653             FDB     READB
00884 301F 3418             FDB     BLIST
```

SYMBOL TABLE

ADATA	0000	ACTL	0001	BDATA	0002	BCTL	0003	INPUT	0004	IOIN	0005
ERROR	0006	TGL	0007	UFLG	0008	RSFLG	0009	EOM	000A	EOPM	000B
STKFLG	000D	RND	000E	DIGFLG	000F	W2	0010	W1	0011	SFLG	0012
DCNTR	0013	TP1	0014	TP1S	0015	TP2	0016	TP2S	0017	TP3	0018
TP3S	0019	TP4	001A	TP4S	001B	TP5	001C	TP5S	001D	TP6	001E
TP6S	001F	TP7	0020	TP7S	0021	T13	0022	T12	0023	T11	0024
T10	0025	T9	0026	T8	0027	T7	0028	T6	0029	T5	002A
T4	002B	T3	002C	T2	002D	T1	002E	ISTK	002F	ISTACK	0051
TA	0052	SPGM	0054	EXTRA	0056	BUFF	0058	REAL	0068	IMAG	0070
AT1	0078	AT2	0080	W	0088	XR	0090	YR	0098	ZR	00A0
TR	00A8	LSTX	00B0	BKWRT	00B8	BKKC	00BA	SOL7	00C6	UPP	00C8
UIP	00CA	ALPHA	00CC	IO1	00CD	IO2	00D0	IT7	00D3	FLAG	00D5
TPOS	00D6	FILE	00D7	AR	00D8	BR	00E0	CR	00E8	DR	00F0
ER	00F8	SDBB	00BA	MT	7E00	TERMN7	003D	IMED	0040	PARCD	00C0
PAREX	0080	NTBL	0000	DOTS	5EC0	PRTDRV	602D	FRMT	5CA8	BLANK	5D75
LDMSG	578D	ROLLD	55B2	ROLLU	57F1	PSD	550A	TXL	55E9	STKUP	55EF
MAD	749B	CMP	74AA	NOR	74D6	TXW	7424	TXXR	7438	EXXR	7452
ARSR	753B	OVUNF	75B6	OVERF	75DD	XR0	740A	XRNINE	75C8	UNDRF	75F1
IMULT	7689	QDG	7669	FPA	75FC	FPS	75F6	FPM	7735	FPD	7793
FPAEX	763D	FPMEX	7780	LSHIFT	7521	ZEROX	7489	XZERO0	7416	XZERO2	7417
RECIP	73E6	TXRX	73F3	CONST	6800	FPDBRC	6898	TAN	68A9	ATN	69C3
DSZERO	6A46	NTLN	6A58	EXPN	6AC9	SIN	6894	COS	689A	ASIN	68F2
ACOS	68F7	PH1	6C5D	PH2	6C8D	PH3	6D34	PH4	6DD0	LSFT8	6E47
SQRT	6E65	MAD8	6F2C	CMP8	53E4	IOUPX	6F52	LOG10	6FA7	YUPX	6FE9
RTOP	7328	PTOR	7386	INIT	3109	IN4	3134	EXIT	3182	BINRY	3198
WB1	31B5	BTDEC	31C8	PRNT	31F6	OTBYT	3270	INBYT	32DA	CHADRS	6489
GIONTR	59D9	KILL	538F	BLIST	3418	BL1	3428	BL2	3439	BL3	3445
BL6	3464	EXT	346C	OTBT1	346F	LEGL	3472	LEG1	3477	PADRS	347E
PAD1	3496	ASCII	3498	ASC1	34B5	ASC1A	34C9	ASC2	34D0	ASC3	34ED
INSRT	34F9	INS1	3511	LAB1	3523	LAB2	3534	LAB3	3538	BLNK	3546
BLN1	3551	LIST	3552	COL0	355F	COL1	3562	COL2	3573	COL6	357C
COL7	3586	COL7A	3591	COL8	35A3	COL8A	35AC	COL8B	35C5	COL9	35C8
COL9A	35F4	COL10	3601	COL10A	3606	COL11	361F	COL12	3628	COL13	362E
XIT	3633	ELIN	3636	OTBT2	363C	STNG	363F	STN1	3644	READB	3653
BRD2	365E	BRD3	366E	BRD4	368D	BRD4A	369F	BRD6	36A3	INBT1	36A6
STRING	36A9	STR0	36B8	STR1	36BE	STR1A	36C2	STR2	36CB	STR2A	36CD
STR3	36D4	STR4	36D8	STR5	36DC	GETCR	36E2	GET1	36F0	OUTCR	36F1
OUT1	36FF	OUT2	3706	OUT2A	370C	OUT3	3718	OUT4	371D	OUT5	3727
OUT6	372B	ANDXY	372E	ORXY	372F	AN1	3755	AN3	375D	ROTX	3760
ROT1	3770	INTAP	3772	INT1	3778	INT2	3786	INT3	378C	INT4	379F
INT5	37A6	INCHR	37A9	INC1	37B4	TBL	37BF	LTAB	370F		

TOTAL ERRORS 1

```
00216                          OPT    LIST,MEM
00219                  *
00220                  **
00221                  ***     CJ PLOTTER ROM
00222                  ***     REV  01
00223                  ***     TIM HICKENLOOPER
00224                  ***     MARCH 1975
00225                  **
00226                  *

00228 3800                     ORG    $3800


00230 3800 10                  FCB    $10,$10   ROM ID #'S
      3801 10
00231 3802 7E 3B3C             JMP    PWRUP
00232 3805 3923                FDB    NULL      WRITE
00233 3807 38D4                FDB    PLOT
00234 3809 3883                FDB    PEN
00235 380B 38E7                FDB    IPLOT
00236 380D 38CA                FDB    MOVE
00237 380F 3923                FDB    NULL      DGTZR - UP
00238 3811 3875                FDB    SCALE
00239 3813 3984                FDB    XAXIS
00240 3815 398C                FDB    YAXIS
00241 3817 38A7                FDB    DRAW
00242 3819 3923                FDB    NULL      DGTZR - DOWN
00243 381B 3923                FDB    NULL      PRNTX
00244 381D 3923                FDB    NULL      CSIZE
00245 381F 3923                FDB    NULL      EXIT
00246 3821 3923                FDB    NULL      DGTZR - LEFT
00247 3823 3923                FDB    NULL      DGTZR - RIGHT


00249                  *
00250                  *     EQUATE TABLE
00251                  *

00253       3FF4       C1     EQU    $3FF4
00254       3F4D       C2     EQU    $3F4D
00256                  *
00257                  *     LISTING ASCII
00258                  *
00260                          OPT    NG
00261 3825 D0                  FCB    $D0,$4C,$4F,$54,$68   PLOT "ALPHA"
00262 382A D0                  FCB    $D0,$4C,$4F,$54,$20   PLOT
00263 382F D0                  FCB    $D0,$45,$4E,$5E,$20   PENA
00264 3834 C9                  FCB    $C9,$50,$4C,$4F,$54   IPLOT
00265 3839 CD                  FCB    $CD,$4F,$56,$45,$20   MOVE
00266 383E C4                  FCB    $C4,$47,$54,$5A,$52   DGTZR
00267 3843 D3                  FCB    $D3,$43,$41,$4C,$45   SCALE
00268 3848 D8                  FCB    $D8,$41,$58,$49,$53   XAXIS
00269 384D D9                  FCB    $D9,$41,$58,$49,$53   YAXIS
00270 3852 C4                  FCB    $C4,$52,$41,$57,$2B   DRAW+
00271 3857 C4                  FCB    $C4,$47,$54,$5A,$52   DGTZR
00272 385C D0                  FCB    $D0,$52,$4E,$54,$58   PRNTX
00273 3861 C3                  FCB    $C3,$53,$49,$5A,$45   CSIZE
00274 3866 C5                  FCB    $C5,$58,$49,$54,$20   EXIT
00275 3868 C4                  FCB    $C4,$47,$54,$5A,$52   DGTZR
00276 3870 C4                  FCB    $C4,$47,$54,$5A,$52   DGTZR
00277                          OPT    G
00279                  *
00280                  *     SCALE
00281                  *
00282                  *     ESTABLISHES FULL SCALE VALUES IN USER UNITS FOR THE
00283                  *     PLOTTER.  THE PLOTTER LIMITS ARE ZERO TO 9999.
00284                  *     SCALE FACTOR = (9999)/((USER MAX)-(USER MIN)).
00285                  *     ORIGIN OFFSET = -(USER MIN)*(SCALE FACTOR).
00286                  *     THESE ARE SAVED IN THE FIRST FOUR PLOTTER REGISTERS:
00287                  *     X-SCALE FACTOR, Y-SCALE FACTOR, X-OFFSET, Y-OFFSET.
00288                  *     USER PARAMETERS:
00289                  *         XR - YMAX
00290                  *         YR - YMIN
00291                  *         ZR - XMAX
00292                  *         TR - XMIN
```

```
00293                    *
00294 3875 BD 3924 SCALE  JSR     INIT      INITIALIZE
00295 3878 8D 49          BSR     ADJST
00296 387A CE 0098 SC1    LDX     #YR
00297 387D BD 75F6        JSR     FPS       MAX-MIN
00298 3880 8D 3A          BSR     NEGXR
00299 3882 96 92          LDA A   XR+2
00300 3884 27 04          BEQ     SC2       ZERO RESULT
00301 3886 96 91          LDA A   XR+1
00302 3888 2A 06          BPL     SC3       POSITIVE NUMBER
00303 388A 86 06    SC2   LDA A   #6
00304 388C 97 06          STA A   ERROR     PARAMETER ERROR
00305 388E 20 1A          BRA     SC4
00306 3890 CE 3FF4 SC3    LDX     #C1
00307 3893 BD 7735        JSR     FPM
00308 3896 BD 73E6        JSR     RECIP     9999/(MAX-MIN)
00309 3899 4F             CLR A
00310 389A BD 39A1        JSR     SET
00311 389D 8D 1D          BSR     NEGXR
00312 389F CE 0098        LDX     #YR
00313 38A2 BD 7735        JSR     FPM       MIN*SCALE
00314 38A5 86 10          LDA A   #$10
00315 38A7 BD 39A1        JSR     SET
00316 38AA BD 396E SC4    JSR     RESXR
00317 38AD BD 55B2        JSR     ROLLD     NEXT PARAMETERS
00318 38B0 BD 55B2        JSR     ROLLD
00319 38B3 BD 3969        JSR     SAVXR
00320 38B6 8D 0B          BSR     ADJST
00321 38B8 27 C0          BEQ     SC1
00322 38BA 20 54    SC5   BRA     EXT1      FIX FSFLG, EXIT
00324                    *
00325                    *    NEGXR
00326                    *
00327 38BC D6 91    NEGXR LDA B   XR+1
00328 38BE CA 80          EOR B   #$80      COMPLEMENT SIGN
00329 38C0 D7 91          STA B   XR+1
00330 38C2 39             RTS


00332                    *
00333                    *    ADJST
00334                    *
00335                    *    TOGGLES T11 BETWEEN 0 AND 8.
00336                    *
00337 38C3 86 08    ADJST LDA A   #8
00338 38C5 90 24          SUB A   T11
00339 38C7 97 24          STA A   T11
00340 38C9 39             RTS


00342                    *
00343                    *    MOVE - PLOT
00344                    *
00345                    *    MOVES THE PEN TO THE POINT SPECIFIED BY THE USER
00346                    *    X AND Y VALUES.
00347                    *    'MOVE' LIFTS THE PEN BEFORE MOVING AND LEAVES IT
00348                    *    RAISED UPON EXITING.  'PLOT' WILL DRAW A LINE FROM
00349                    *    THE CURRENT POINT TO THE NEW POINT IF THE PEN WAS
00350                    *    ALREADY DOWN UPON CALL, ELSE IT WILL MOVE TO THE
00351                    *    POINT AND THEN LOWER THE PEN BEFORE EXITING.
00352                    *    PLOTTER COORDINATE = (USER VALUE)*(SCALE FACTOR)+
00353                    *      (ORIGIN OFFSET).
00354                    *    USER PARAMETERS:
00355                    *        XR - X COORDINATE
00356                    *        YR - Y COORDINATE
00357                    *
00358 38CA 8D 58    MOVE  BSR     INIT      INITIALIZE
00359 38CC 7A 0027        DEC     T8        SET FLAG, LEAVE PEN UP
00360 38CF 8D 38BF        JSR     PENU      RAISE THE PEN
00361 38D2 20 02          BRA     PLT2
00362 38D4 8D 4E    PLOT  BSR     INIT      INITIALIZE
00363 38D6 BD 398C PLT2   JSR     POS       GET PLOTTER COORDINATE
00364 38D9 86 20          LDA A   #$20
```

```
00365 38D8 BD 39A1          JSR    SET       STORE IN PLOTTER POSITION REGISTER
00366 38DE 8D E3            BSR    ADJST
00367 38E0 27 1E            BEQ    IPL2      DONE
00368 38E2 BD 3986          JSR    LDYR
00369 38E5 20 EF            BRA    PLT2      NOW DO Y-COORDINATE
00371                  *
00372                  *    IPLOT
00373                  *
00374                  *    INCREMENTAL PLOT MOVES THE PEN FROM IT'S CURRENT
00375                  *    POSITION THE AMOUNT SPECIFIED BY THE USER INCREMENT
00376                  *    X AND Y VALUES.  DRAWS A LINE IF THE PEN WAS ALREADY
00377                  *    DOWN UPON CALL. ELSE IT MOVES TO THE NEW POINT AND
00378                  *    THEN LOWERS THE PEN.
00379                  *    PLOTTER COORDINATE = (USER VALUE)*(SCALE FACTOR)+
00380                  *      (PRESENT PLOTTER COORDINATE).
00381                  *    USER PARAMETERS:
00382                  *        XR - DELTA X VALUE
00383                  *        YR - DELTA Y VALUE
00384                  *
00385 38E7 9D 3B    IPLOT  BSR    INIT
00386 38E9 BD 3980  IPL1   JSR    MULT      VALUE*SCALE FACTOR
00387 38EC 86 20           LDA A  #$20
00388 38EE BD 3973         JSR    INDX
00389 38F1 BD 75FC         JSR    FPA       ADD DELTA TO COORDINATE
00390 38F4 DE 69           LDX    REAL+1
00391 38F6 BD 73F3         JSR    TXRX      SAVE NEW COORDINATE
00392 38F9 BD 3986         JSR    LDYR
00393 38FC 8D C5           BSR    ADJST
00394 38FE 26 E9           BNE    IPL1
00395 3900 BD 3A94  IPL2   JSR    MOVO      MOVE TO POINT
00396 3903 96 05           LDA A  IOIN      PEN DOWN?
00397 3905 2B 07           BMI    EXIT      YES
00398 3907 96 27           LDA A  T8        'MOVE' COMMAND?
00399 3909 2B 03           BMI    EXIT      YES
00400 3908 BD 3B95         JSR    PEND


00402                  *
00403                  *    EXIT
00404                  *
00405                  *    RESTORES THE XR TO ORIGINAL STATUS. ADJUSTS THE
00406                  *    RSFLG AND RESETS PIA AND CHANNEL SELECT.
00407                  *
00408 390E 8D 5E    EXIT   BSR    RESXR     RESTORE XR
00409 3910 0F       EXT1   SEI
00410 3911 D6 26           LDA B  T9        ENTRY RSFLG STATUS
00411 3913 2B 02           BMI    EXT3      RUNNING AND STOP KEY. EXIT NOW
00412 3915 D7 09    EXT2   STA B  RSFLG     RESTORE ORIGINAL RSFLG
00413 3917 C6 04    EXT3   LDA B  #4        IO6
00414 3919 D7 00           STA B  ADATA     RESET CHANNEL SELECT
00415 391B 7F 0000         CLR    ADATA
00416 391E C6 3C           LDA B  #@74
00417 3920 D7 03           STA B  BCTL      CLEAR CB2 FUNSTION
00418 3922 0F              CLI
00419 3923 39       NULL   RTS
00421                  *
00422                  *    INIT
00423                  *
00424                  *    INITIALIZES POINTERS AND FLAGS FOR ALL PLOTTER
00425                  *    ROUTINE.  SETS CHANNEL SELECT FOR PLOTTER.
00426                  *
00427 3924 86 5B    INIT   LDA A  #$5B      CHANNEL 1. IO7
00428 3926 DE CD           LDX    IO1
00429 3928 D6 CF           LDA B  IO1+2     LOAD SELECT CODE
00430 392A C1 01           CMP B  #1        PLOTTER?
00431 392C 27 04           BEQ    INT1      YES
00432 392E DE D0           LDX    IO2
00433 3930 86 AB           LDA A  #$AB      CHANNEL 2. IO7
00434 3932 97 00    INT1   STA A  ADATA     SET CHANNEL
00435 3934 84 F0           AND A  #$F0
00436 3936 97 00           STA A  ADATA     LATCH IT
00437 3938 86 2C           LDA A  #@54
00438 393A 97 03           STA A  BCTL      SET UP CB2
00439 393C C6 F3           LDA B  #$F3
```

```
00440 393E D7 25          STA B  T10       SAVE IT
00441 3940 DF 22          STX    T13       SAVE FOR FUTURE
00442 3942 4F             CLR A
00443 3943 97 24          STA A  T11       CLEAR XYFLG
00444 3945 97 27          STA A  T8        'MOVE' FLAG
00445 3947 91 0F          CMP A  DIGFLG
00446 3949 27 05          BEQ    INT2      NOT DIGIT ENTRY
00447 394B 97 0F          STA A  DIGFLG
00448 394D 4A             DEC A            TERMINATE DIGIT ENTRY
00449 394E 97 0D          STA A  STKFLG
00450 3950 0F       INT2  SEI
00451 3951 D6 09          LDA B  RSFLG
00452 3953 C4 C0          AND B  #$C0
00453 3955 D7 26          STA B  T9        SAVE RSFLG ENTRY STATUS
00454 3957 C6 C0          LDA B  #$C0
00455 3959 D7 09          STA B  RSFLG     SET TO PROGRAM EXECUTION
00456 395B 0E             CLI
00457 395C D6 05          LDA B  IOIN
00458 395E 59             ROL B            PLOTTER ON?
00459 395F 28 08          BMI    SAVXR     YES
00460 3961 86 05          LDA A  #5
00461 3963 97 06          STA A  ERROR     SET PLOTTER ERROR
00462 3965 31             INS
00463 3966 31             INS              BUMP RETURN STACK
00464 3967 20 A7          BRA    EXIT+2
00466                *
00467                *    SAVXR
00468                *
00469                *    SAVES THE XR IN A TEMPORARY LOCATION (IMAG).
00470                *
00471 3969 CE 0070 SAVXR  LDX    #IMAG
00472 396C 20 35          BRA    TXR

00474                *
00475                *    RESXR
00476                *
00477                *    RESTORES XR FROM TEMPORARY SAVE (IMAG).
00478                *
00479 396E CE 0070 RESXR  LDX    #IMAG
00480 3971 20 16          BRA    TXX

00482                *
00483                *    INDX
00484                *
00485                *    RETURNS SPECIFIED PLOTTER REGISTER ADDRESS IN IX.
00486                *    SPECIFICATION INDEX PASSED IN ACCA AND X OR Y
00487                *    DETERMINED BY XYFLG (T11).  ACCA=0 FOR
00488                *    SCALE FACTOR, ACCA=10(HEX) FOR ORIGEN OFFSET,
00489                *    ACCA=20(HEX) FOR CURRENT PLOTTER COORDINATES.
00490                *
00491 3973 9B 24    INDX   ADD A  T11       ADD XYFLG
00492 3975 9B 23    INDX1  ADD A  T12       ADD TO LOWER HALF OF POINTER
00493 3977 97 6A           STA A  REAL+2    SAVE
00494 3979 96 22           LDA A  T13       UPPER HALF
00495 397B 97 69           STA A  REAL+1
00496 397D DE 69           LDX    REAL+1    LOAD REG. POINTER
00497 397F 39              RTS

00499                *
00500                *    MULT
00501                *
00502                *    MULTIPLIES XR BY SCALE FACTOR.
00503                *
00504 3980 4F       MULT   CLR A
00505 3981 8D F0           BSR    INDX
00506 3983 7E 7735 MLT     JMP    FPM

00508                *
00509                *    LDYR
00510                *
00511                *    LOADS THE YR INTO THE XR.
00512                *
00513 3986 CE 0098 LDYR    LDX    #YR
00514 3989 7E 743B TXX     JMP    TXXR
```

```
00516                    *
00517                    *     POS
00518                    *
00519                    *     CALCULATES THE PLOTTER COORDINATE FROM THE USER
00520                    *     VALUE IN XR.  XYFLAG MUST BE SET BEFORE ENTRY.
00521                    *     XR = (XR)*(SCALE FACTOR)+(ORIGIN OFFSET).
00522                    *
00523 398C 8D F2    POS     BSR     MULT      XR*SCALE FACTOR
00524 398E 86 10            LDA A   #$10
00525 3990 8D E1            BSR     INDX
00526 3992 BD 75FC          JSR     FPA

00528                    *
00529                    *     XRCHK
00530                    *
00531                    *     CHECKS FOR 0 <= XR < 10000. SETS 'Z' BIT
00532                    *     FOR VALID NUMBER (ALSO CLEARS ACCB).
00533                    *
00534 3995 D6 91    XRCHK   LDA B   XR+1
00535 3997 2B 07            BMI     XR1
00536 3999 D6 90            LDA B   XR
00537 399B C1 03            CMP B   #3
00538 399D 2E 01            BGT     XR1
00539 399F 5F               CLR B
00540 39A0 39       XR1     RTS

00542                    *
00543                    *     SET
00544                    *
00545                    *     SAVES THE XR IN SPECIFIED PLOTTER REGISTER.
00546                    *     XYFLAG MUST BE SET BEFORE CALL.
00547                    *
00548 39A1 8D D0    SET     BSR     INDX
00549 39A3 7E 73F3  TXR     JMP     TXRX

00551                    *
00552                    *    GET
00553                    *
00554                    *     LOADS SPECIFIED PLOTTER REGISTER INTO THE
00555                    *     XR.   XYFLAG MUST BE SET BEFORE CALL.
00556 39A6 8D CB    GET     BSR     INDX
00557 39A8 20 DF            BRA     TXX

00559                    *
00560                    *     TEMP1 - TEMP2
00561                    *
00562                    *     SAVES THE XR IN TEMPORARY LOCATIONS.
00563                    *
00564 39AA CE 0058  TEMP1   LDX     #BUFF
00565 39AD 20 F4            BRA     TXR
00566 39AF CE 0060  TEMP2   LDX     #BUFF+8
00567 39B2 20 EF            BRA     TXR
00569                    *
00570                    *     XAXIS - YAXIS
00571                    *
00572                    *     DRAWS AN X OR Y AXIS FROM THE STARTING POINT TO
00573                    *     THE ENDING POINT, INTERCEPTING THE OTHER AXIS AT THE
00574                    *     SPECIFIED POINT, AND SPACING TIC MARKS AS INDICATED.
00575                    *     IF THE SIGN OF THE TIC INTERVAL IS THE SAME AS THE
00576                    *     SIGN OF (ENDING POINT - STARTING POINT), TICS WILL
00577                    *     BE SPACED ALONG THE ENTIRE LENGTH OF THE AXIS. ELSE
00578                    *     ONLY ONE TIC WILL BE DONE AT THE STARTING POINT.
00579                    *     (SEE OPERATION OF 9830 PLOTTER AXIS COMMANDS).
00580                    *     USER PARAMETERS:
00581                    *         XR - AXIS INTERCEPT
00582                    *         YR - TIC MARK INTERVAL
00583                    *         ZR - ENDING POINT
00584                    *         TR - STARTING POINT
00585                    *
00586 39B4 BD 3924  XAXIS   JSR     INIT      INITIALIZE
00587 39B7 BD 38C3          JSR     ADJST
00588 39BA 20 03            BRA     AX1
00589 39BC BD 3924  YAXIS   JSR     INIT
00590 39BF 8D CB    AX1     BSR     POS       SCALE INTERCEPT
00591 39C1 26 21            BNE     AX2       RANGE ERROR
```

```
00592 39C3 86 20          LDA A  #$20
00593 39C5 8D DA          BSR    SET
00594 39C7 BD 38C3        JSR    ADJST
00595 39CA 8D BA          BSR    LDYR
00596 39CC 8D B2          BSR    MULT       SCALE TIC
00597 39CE 8D DA          BSR    TEMP1
00598 39D0 CE 00A0        LDX    #ZR
00599 39D3 8D B4          BSR    TXX
00600 39D5 8D B5          BSR    POS
00601 39D7 26 0B          BNE    AX2        RANGE ERROR
00602 39D9 8D D4          BSR    TEMP2
00603 39DB CE 00A8        LDX    #TR
00604 39DE 8D A9          BSR    TXX
00605 39E0 8D AA          BSR    POS
00606 39E2 27 06          BEQ    AX3
00607 39E4 86 06    AX2   LDA A  #6
00608 39E6 97 06          STA A  ERROR      ERROR EXIT
00609 39E8 20 64          BRA    AX7
00610 39EA 86 20    AX3   LDA A  #$20
00611 39EC BD 3973        JSR    INDX
00612 39EF DF 28          STX    T7         REG POINTER
00613 39F1 8D B0          BSR    TXR
00614 39F3 CE 0060        LDX    #BUFF+8
00615 39F6 BD 75F6        JSR    FPS        END-START
00616 39F9 96 91          LDA A  XR+1
00617 39FB 97 2A          STA A  T5         SET DIRECTION SIGN
00618 39FD BD 3B8F        JSR    PENU
00619 3A00 BD 3AB3        JSR    MOV        MOVE TO START
00620 3A03 BD 3B95        JSR    PEND
00621 3A06 96 58          LDA A  BUFF
00622 3A08 2B 36          BMI    AX5        NEGITIVE EXPONENT
00623 3A0A 96 5A          LDA A  BUFF+2
00624 3A0C 27 32          BEQ    AX5        ZERO TIC
00625 3A0E 96 09    AX4   LDA A  RSFLG
00626 3A10 2A 39          BPL    AX6+2      STOP KEY
00627 3A12 8D 40          BSR    TIC
00628 3A14 96 59          LDA A  BUFF+1
00629 3A16 91 2A          CMP A  T5
00630 3A18 26 26          BNE    AX5        ONE TIC
00631 3A1A CE 0058        LDX    #BUFF
00632 3A1D 8D 72          BSR    TXX1
00633 3A1F DE 28          LDX    T7
00634 3A21 BD 75FC        JSR    FPA        TIC + POSITION
00635 3A24 DE 28          LDX    T7
00636 3A26 8D 29          BSR    TXR1
00637 3A28 CE 0060        LDX    #BUFF+8
00638 3A2B BD 75F6        JSR    FPS        END-POSITION
00639 3A2E 96 91          LDA A  XR+1
00640 3A30 91 2A          CMP A  T5
00641 3A32 27 08          BEQ    AX4A       LEGAL
00642 3A34 96 90          LDA A  XR
00643 3A36 2B 04          BMI    AX4A       SLOP SMALL, DO TIC
00644 3A38 96 92          LDA A  XR+2
00645 3A3A 26 04          BNE    AX5        IF NOT ZERO, DONE
00646 3A3C 8D 75    AX4A  BSR    MOV
00647 3A3E 20 CE          BRA    AX4
00648 3A40 CE 0060  AX5   LDX    #BUFF+8
00649 3A43 8D 4C          BSR    TXX1
00650 3A45 DE 28          LDX    T7
00651 3A47 8D 08          BSR    TXR1       END POINT
00652 3A49 8D 68    AX6   BSR    MOV
00653 3A4B BD 388F        JSR    PENU
00654 3A4E 7E 390E  AX7   JMP    EXIT       DONE


00656 3A51 7E 73F3  TXR1  JMP    TXRX
00658                *
00659                *   TIC
00660                *
00661                *    THIS ROUTINE TAKES THE TOGGLE OF THE XYFLAG AND
00662                *    DRAWS A LINE FROM PLOTTER CURRENT POSITION TO
00663                *    +50,-50, AND THEN RETURNING TO ORIGINAL POINT
00664                *    BEFORE EXITING.
00665                *
00666 3A54 86 08   TIC    LDA A  #8
```

```
00667 3A56 90 24          SUB A  T11      TOGGLE XYFLG
00668 3A58 C6 FF          LDA B  #$FF
00669 3A5A D7 2C          STA B  T3       SET PASS FLAG
00670 3A5C 8B 20          ADD A  #$20
00671 3A5E BD 3975        JSR    INDX1
00672 3A61 DF 1E          STX    TP6      REG ADDRESS
00673 3A63 8D 2C          BSR    TXX1
00674 3A65 CE 0078        LDX    #AT1
00675 3A68 8D E7          BSR    TXR1     SAVE IN TEMPORARY
00676 3A6A CE 3F4D TIC1   LDX    #C2
00677 3A6D BD 75F6        JSR    FPS
00678 3A70 8D 38BC        JSR    NEGXR
00679 3A73 20 06          BRA    TIC3
00680 3A75 CE 3F4D TIC2   LDX    #C2
00681 3A78 BD 75FC        JSR    FPA      ADD 50. THIS TIME
00682 3A7B DE 1E   TIC3   LDX    TP6
00683 3A7D 8D D2          BSR    TXR1     SET TIC END
00684 3A7F 8D 32          BSR    MOV
00685 3A81 CE 0078        LDX    #AT1
00686 3A84 8D 0B          BSR    TXX1
00687 3A86 7C 002C        INC    T3
00688 3A89 27 EA          BEQ    TIC2     NEXT HALF OF TIC
00689 3A8B DE 1E   TIC4   LDX    TP6
00690 3A8D 8D C2          BSR    TXR1     RESTORE
00691 3A8F 20 22   TIC5   BRA    MOV


00693 3A91 7E 7438 TXX1   JMP    TXXR
00695              *
00696              *     MOVQ - MOV
00697              *
00698              *     'MOVQ' CHECKS FOR COORDINATE WITHIN RANGE OF
00699              *     PLOTTER. IF THEY ARE OUT OF RANGE, IT RAISES THE
00700              *     PEN AND MOVES LEAVING PEN RAISED.
00701              *     'MOV' WILL ALWAYS MOVE THE PEN.  IF THE COORDINATES
00702              *     ARE OUT OF RANGE, IT SUBSTITUTES THE CLOSEST LIMIT
00703              *     VALUE AND THEN MAKES THE MOVE.
00704              *
00705 3A94 86 20   MOVQ   LDA A  #$20
00706 3A96 BD 3975        JSR    INDX1
00707 3A99 A6 01          LDA A  1,X
00708 3A9B 2B 10          BMI    PNU
00709 3A9D A6 09          LDA A  9,X
00710 3A9F 2B 0C          BMI    PNU
00711 3AA1 A6 00          LDA A  0,X
00712 3AA3 81 03          CMP A  #3
00713 3AA5 2E 06          BGT    PNU
00714 3AA7 A6 08          LDA A  8,X
00715 3AA9 81 03          CMP A  #3
00716 3AAB 2F 06          BLE    MOV
00717 3AAD BD 388F PNU    JSR    PENU     RAISE THE PEN
00718 3AB0 7A 0027        DEC    T8       SET FLAG, LEAVE PEN UP
00719 3AB3 86 7F   MOV    LDA A  #$7F
00720 3AB5 94 25          AND A  T10      CLEAR SYC BIT
00721 3AB7 97 25          STA A  T10
00722 3AB9 4F             CLR A
00723 3ABA 97 14          STA A  TP1
00724 3ABC 8B 20   MOV1   ADD A  #$20
00725 3ABE BD 3975        JSR    INDX1
00726 3AC1 A6 01          LDA A  1,X
00727 3AC3 2A 05          BPL    MOV3     POSITIVE
00728 3AC5 CE 0000 MOV2   LDX    #0
00729 3AC8 20 10          BRA    MOV5     ZERO DEFAULT
00730 3ACA E6 00   MOV3   LDA B  0,X
00731 3ACC 5C             INC B
00732 3ACD 2F F6          BLE    MOV2     NEGITIVE EXPONENT
00733 3ACF C1 05          CMP B  #5
00734 3AD1 2D 05          BLT    MOV4     WITHIN LIMITS
00735 3AD3 CE 9999        LDX    #$9999
00736 3AD6 20 02          BRA    MOV5
00737 3AD8 8D 13   MOV4   BSR    JUST     FORMAT DATA
00738 3ADA DF 16   MOV5   STX    TP2
00739 3ADC D6 16          LDA B  TP2
00740 3ADE 8D 33          BSR    OUT
00741 3AE0 D6 17          LDA B  TP2S     OUT COORDINATE
```

```
00742 3AE2 8D 2F          BSR    OUT
00743 3AE4 86 08          LDA A  #8
00744 3AE6 90 14          SUB A  TP1
00745 3AE8 97 14          STA A  TP1       FIX FLAG
00746 3AEA 26 D0          BNE    MOV1
00747 3AEC 39             RTS
00749                *
00750                *    JUST
00751                *
00752                *    THIS ROUTINE TAKES THE PLOTTER COORDINATE REGISTER
00753                *    SPECIFIED BY THE IX AND THE NUMBER OF DIGITS
00754                *    SPECIFIED IN ACCB AND FORMS THE RIGHT JUSTIFIED
00755                *    FOUR BCD DIGITS IN TP2 AND TP2S.
00756                *
00757 3AED C0 05   JUST   SUB B  #5
00758 3AEF D7 15          STA B  TP1S      SET COUNTER
00759 3AF1 A6 02          LDA A  2,X
00760 3AF3 E6 03          LDA B  3,X       LOAD FOUR BCD DIGITS
00761 3AF5 7C 0015 JST1   INC    TP1S      NORMALIZED?
00762 3AF8 27 0E          BEQ    JST2      YES
00763 3AFA 0C             CLC
00764 3AFB 46             ROR A
00765 3AFC 56             ROR B
00766 3AFD 0C             CLC
00767 3AFE 46             ROR A
00768 3AFF 56             ROR B            RIGHT SIHFT
00769 3B00 0C             CLC
00770 3B01 46             ROR A
00771 3B02 56             ROR B
00772 3B03 0C             CLC
00773 3B04 46             ROR A
00774 3B05 56             ROR B
00775 3B06 20 ED          BRA    JST1
00776 3B08 97 16   JST2   STA A  TP2
00777 3B0A D7 17          STA B  TP2S      SET JUSTIFIED NUMBER
00778 3B0C DE 16          LDX    TP2
00779 3B0E 39             RTS
00781                *
00782                *   OUT
00783                *
00784                *    THIS ROUTINE OUTS TO THE PLOTTER ACCB.  THE 'SYC'
00785                *    'MVR'. AND 'PNC' BITS OF THE CAHNNEL CODE WORD
00786                *    MUST HAVE BEEN PREVIOUSLY SET.  THE 'SYC' BIT
00787                *    IS ALWAYS RESET BY THE ROUTINE.
00788                *
00789 3B0F 96 09   OUT0   LDA A  RSFLG     STOP KEY?
00790 3B11 2A 25          BPL    OUT3      YES
00791 3B13 86 0A   OUT    LDA A  #$A
00792 3B15 97 00          STA A  ADATA     IO1
00793 3B17 96 05          LDA A  IOIN      FLAG STATUS. BUSY?
00794 3B19 2A F4          BPL    OUT0      YES
00795 3B1B 0F      OUT1   SEI
00796 3B1C 96 25          LDA A  T10       LOAD STATUS BITS
00797 3B1E 97 00          STA A  ADATA     CONTROL BITS SET
00798 3B20 53             COM B
00799 3B21 D7 02          STA B  BDATA     WRITE DATA WORD
00800 3B23 84 F0          AND A  #$F0      RELEASE IO5
00801 3B25 97 00          STA A  ADATA     DRIVE CTL
00802 3B27 8A C3          ORA A  #$C3      RESET SYC,MVR,IO5
00803 3B29 97 25          STA A  T10       SAVE
00804 3B2B 0E             CLI
00805 3B2C 86 0A   OUT2   LDA A  #$A       IO1
00806 3B2E 97 00          STA A  ADATA
00807 3B30 96 05          LDA A  IOIN      READ FLG STATUS
00808 3B32 2A 04          BPL    OUT3      RESPONDED
00809 3B34 96 09          LDA A  RSFLG
00810 3B36 2B F4          BMI    OUT2      NO STOP KEY
00811 3B38 4F      OUT3   CLR A
00812 3B39 97 00          STA A  ADATA     RELEASE IO1
00813 3B3B 39             RTS
00815                *
00816                *    POWER UP
00817                *
00818                *    CALLED BY SUPERVISOR DURING POWER ON.  NINE
00819                *    REGISTERS AND ALLOCATED AND INITIALIZED.
```

```
00820                *       1 - X-SCALE FACTOR
00821                *       2 - Y-SCALE FACTOR
00822                *       3 - X-OFFSET
00823                *       4 - Y-OFFSET
00824                *       5 - X-POSITION
00825                *       6 - Y-POSITION
00826                *       7 - 1/ASPECT RATIO
00827                *       8 - CHARACTER SIZE, ANGLE INFO (SEE CSIZE).
00828                *       9 - PAPER SIZE RATIO
00829                *
00830 3B3C C6 5B    PWRUP   LDA B   #$5B      CHANNEL 1
00831 3B3E D7 00            STA B   ADATA     SET CHANNEL, IO7
00832 3B40 5A               DEC B
00833 3B41 D7 00            STA B   ADATA     RELEASE IO7, SET IO1
00834 3B43 96 05            LDA A   IOIN      READ DEVICE ID
00835 3B45 CE 00CD          LDX     #IO1      SLOT 1 INFO
00836 3B48 85 01            BIT A   #1        PLOTTER THERE?
00837 3B4A 27 03            BEQ     PIN1      YES
00838 3B4C CE 00D0          LDX     #IO2      SLOT 2 INFO
00839 3B4F 86 01    PIN1    LDA A   #1
00840 3B51 A7 02            STA A   2,X       SET PLOTTER ID
00841 3B53 96 0B            LDA A   EOPM
00842 3B55 A7 00            STA A   0,X       SET PAGE
00843 3B57 96 0C            LDA A   EOPM+1
00844 3B59 80 48            SUB A   #$48      ALLOCATE 9 REGISTERS
00845 3B5B A7 01            STA A   1,X
00846 3B5D 97 0C            STA A   EOPM+1
00847 3B5F EE 00            LDX     0,X       LOAD REG. POINTER
00848 3B61 86 10            LDA A   #$10
00849 3B63 A7 02            STA A   2,X       SCALE FACTOR
00850 3B65 A7 0A            STA A   $A,X
00851 3B67 A7 42            STA A   $42,X     PAPER RATIO=1
00852 3B69 6A 30            DEC     $30,X
00853 3B6B 86 60            LDA A   #$60      1/ASPECT=.6
00854 3B6D A7 32            STA A   $32,X
00855 3B6F 86 02            LDA A   #2
00856 3B71 A7 38            STA A   $38,X
00857 3B73 86 20            LDA A   #$20      SIZE 2%
00858 3B75 A7 3A            STA A   $3A,X
00859 3B77 86 08            LDA A   #8
00860 3B79 A7 3E            STA A   $3E,X
00861 3B7B 86 04            LDA A   #4        IO6
00862 3B7D 97 00            STA A   ADATA     RESET CHANNEL SELECT
00863 3B7F 7F 0000          CLR     ADATA
00864 3B82 39       PIN2    RTS
00866                *
00867                *     PEN
00868                *
00869                *     THESE ROUTINES RAISE AND LOWER THE PEN.  'PEN' IS
00870                *     THE MAINFRAME CALL TO RAISE THE PEN.  'PENU' AND
00871                *     'PEND' ARE UTILITY ROUTINES CALLED FROM OTHER
00872                *     PLOTTER ROUTINES.
00873                *
00874 3B83 BD 3924  PEN     JSR     INIT
00875 3B86 96 05            LDA A   IOIN      PEN RAISED?
00876 3B88 2A 02            BPL     PEN1      YES
00877 3B8A 8D 03            BSR     PENU      LIFT IT NOW
00878 3B8C 7E 390F  PEN1    JMP     EXIT
00879                *
00880                *
00881 3B8F D6 25    PENU    LDA B   T10
00882 3B91 CA 20            ORA B   #$20      SET PEN UP BIT
00883 3B93 20 04            BRA     PN1
00884 3B95 D6 25    PEND    LDA B   T10
00885 3B97 C4 DF            AND B   #$DF      CLEAR PEN UP
00886 3B99 C4 3F    PN1     AND B   #$3F      CLEAR SYS, MVR BITS
00887 3B9B D7 25            STA B   T10
00888 3B9D 5F               CLR B
00889 3B9E 7E 3B13          JMP     OUT



00891                *
00892                *     ZERXR
00893                *
```

```
00894                *     ZEROS THE XR.
00895                *
00896 3BA1 CE 0090 ZERXR  LDX    #XR
00897 3BA4 7E 7489 ZERO   JMP    ZEROX



00899                *
00900                *     DRAW+
00901                *
00902                *     OUTS A '+' OVER CURRENT PEN POSITION.
00903                *     LOWERS THE PEN AND LEAVES IT LOWERED.
00904                *
00905 3BA7 BD 3924 DRAW     JSR    INIT
00906 3BAA BD 3AB3          JSR    MOV
00907 3BAD 8D E6            BSR    PEND
00908 3BAF BD 3A54          JSR    TIC
00909 3BB2 BD 38C3          JSR    ADJST
00910 3BB5 BD 3A54          JSR    TIC
00911 3BB8 20 D2            BRA    PEN1      EXIT
```

SYMBOL TABLE

ADATA	0000	ACTL	0001	BDATA	0002	BCTL	0003	INPUT	0004	IOIN	0005
ERROR	0006	TGL	0007	UFLG	0008	RSFLG	0009	EOM	000A	EOPM	000B
STKFLG	000D	RND	000E	DIGFLG	000F	W2	0010	W1	0011	SFLG	0012
DCNTR	0013	TP1	0014	TP1S	0015	TP2	0016	TP2S	0017	TP3	0018
TP3S	0019	TP4	001A	TP4S	001B	TP5	001C	TP5S	001D	TP6	001E
TP6S	001F	TP7	0020	TP7S	0021	T13	0022	T12	0023	T11	0024
T10	0025	T9	0026	T8	0027	T7	0028	T6	0029	T5	002A
T4	002B	T3	002C	T2	002D	T1	002E	ISTK	002F	ISTACK	0051
TA	0052	SPGM	0054	EXTRA	0056	BUFF	0058	REAL	0068	IMAG	0070
AT1	0078	AT2	0080	W	0088	XR	0090	YR	0098	ZR	00A0
TR	00A8	LSTX	00B0	BKWRT	00B8	BKKC	00BA	SOL7	00C6	UPP	00C8
UIP	00CA	ALPHA	00CC	IO1	00CD	IO2	00D0	IT7	00D3	FLAG	00D5
TPOS	00D6	FILE	00D7	AR	00D8	BR	00E0	CR	00E8	DR	00F0
ER	00F8	SDBB	00BA	MT	7E00	TERMN7	003D	IMED	0040	PARCD	00C0
PAREX	0080	NTBL	0000	DOTS	5EC0	PRTDRV	602D	FRMT	5CA8	BLANK	5D75
LDMSG	57BD	ROLLD	55B2	ROLLU	57F1	PSD	55DA	TXL	55E9	STKUP	55EF
MAD	749B	CMP	74AA	NOR	74D6	TXW	7424	TXXR	7438	EXXR	7452
ARSR	753B	OVUNF	75B6	OVERF	75DD	XR0	740A	XRNINE	75C8	UNDRF	75F1
IMULT	7689	QDG	7669	FPA	75FC	FPS	75F6	FPM	7735	FPD	7793
FPAEX	763D	FPMEX	7780	LSHIFT	7521	ZEROX	7489	XZERO0	7416	XZERO2	7417
RECIP	73E6	TXRX	73F3	CONST	6800	FPDBRC	6898	TAN	68A9	ATN	69C3
DSZERO	6A46	NTLN	6A58	EXPN	6AC9	SIN	6B94	COS	6B9A	ASIN	6BF2
ACOS	6BF7	PH1	6C5D	PH2	6C8D	PH3	6D34	PH4	6DD0	LSFTB	6E47
SQRT	6E65	MADB	6F2C	CMPR	53E4	IOUPX	6F52	LOG10	6FA7	YUPX	6FE9
RTOP	7328	PTOR	7386	C1	3FF4	C2	3F4D	SCALE	3B75	SC1	387A
SC2	388A	SC3	3890	SC4	38AA	SC5	38BA	NEGXR	38BC	ADJST	38C3
MOVE	38CA	PLOT	38D4	PLT2	38D6	IPLOT	38E7	IPL1	38E9	IPL2	3900
EXIT	390E	EXT1	3910	EXT2	3915	EXT3	3917	NULL	3923	INIT	3924
INT1	3932	INT2	3950	SAVXR	3969	RESXR	396E	INDX	3973	INDX1	3975
MULT	3980	MLT	3983	LDYR	3986	TXX	3989	POS	398C	XRCHK	3995
XR1	39A0	SET	39A1	TXR	39A3	GET	39A6	TEMP1	39AA	TEMP2	39AF
XAXIS	39B4	YAXIS	39BC	AX1	39BF	AX2	39E4	AX3	39EA	AX4	3A0E
AX4A	3A3C	AX5	3A40	AX6	3A49	AX7	3A4E	TXR1	3A51	TIC	3A54
TIC1	3A6A	TIC2	3A75	TIC3	3A7B	TIC4	3A8B	TIC5	3A8F	TXX1	3A91
MOVQ	3A94	PNU	3AAD	MOV	3AB3	MOV1	3ABC	MOV2	3AC5	MOV3	3ACA
MOV4	3AD8	MOV5	3ADA	JUST	3AED	JST1	3AF5	JST2	3B08	OUT0	3B0F
OUT	3B13	OUT1	3B1B	OUT2	3B2C	OUT3	3B38	PWRUP	3B3C	PIN1	3B4F
PIN2	3B82	PEN	3B83	PEN1	3B8C	PENU	3B8F	PEND	3B95	PN1	3B99
ZERXR	3BA1	ZERO	3BA4	DRAW	3BA7						

TOTAL ERRORS 1

```
****ERROR 201
 4  NAM CJBPG
```

```
00216                          OPT    LIST,MEM
00219                          OPT    G
00220                  *
00221                  *    EQUATE TABLE
00222                  *

00224       38C3       ADJST  EQU    $38C3
00225       390E       EXIT   EQU    $390E
00226       3924       INIT   EQU    $3924
00227       396E       RESXR  EQU    $396E
00228       3973       INDX   EQU    $3973
00229       3975       INDX1  EQU    $3975
00230       39A1       SET    EQU    $39A1
00231       39A6       GET    EQU    $39A6
00232       39AA       TEMP1  EQU    $39AA
00233       3A94       MOVQ   EQU    $3A94
00234       3AB3       MOV    EQU    $3AB3
00235       3B8F       PENU   EQU    $3B8F
00236       3B95       PEND   EQU    $3B95
00237       3BA1       ZERXR  EQU    $3BA1
00239 3BD0             ORG    $3BD0
00240                  *
00241                  *    DGTZR
00242                  *
00243                  *    THIS ENABLES THE USER TO POSITION THE PEN BY
00244                  *    PRESSING DIRECTION STEP KEYS.  BY HOLDING THE
00245                  *    KEY DOWN THE STEP IS REPETED.  STEPS ARE 0.1%
00246                  *    OF FULL SCALE FOR 25 STEPS(REPETED), AND THEN
00247                  *    INCREASED TO 1.0% INCREMENTS.  UPON EXITING
00248                  *    THE USER COORDINATES ARE RETURNED IN THE X AND
00249                  *    Y REGISTERS.
00250                  *
00251 3BD0 86 80    DGTZ5  LDA A  #@200    EXIT
00252 3BD2 4C       DGTZ2  INC A           RIGHT
00253 3BD3 8B 07    DGTZ1  ADD A  #@7      LEFT
00254 3BD5 8B 08    DGTZ3  ADD A  #@10     DOWN
00255 3BD7 8B 14    DGTZ4  ADD A  #@24     UP
00256 3BD9 97 2E           STA A  T1       SAVE KEY CODE
00257 3BDB BD 3924         JSR    INIT
00258 3BDE 96 2E           LDA A  T1       EXIT COMMAN?
00259 3BE0 2A 03           BPL    DT0      NO
00260 3BE2 7E 3C54         JMP    DT7
00261 3BE5 8D A8    DT0    BSR    PENU     LIFT THE PEN
00262 3BE7 BD 55EF         JSR    STKUP
00263 3BEA 4F              CLR A
00264 3BEB 97 09           STA A  RSFLG
00265 3BED 97 B9           STA A  BKWRT+1   RESET FLAGS
00266 3BEF 97 0D           STA A  STKFLG
00267 3BF1 BD 3CA0         JSR    SETUP    INITIALIZE ALL
00268 3BF4 96 B9    DT1    LDA A  BKWRT+1   NEW KEY?
00269 3BF6 26 3C           BNE    DT4A     YES
00270 3BF8 86 0F           LDA A  #$F      KEYBOARD SELECT
00271 3BFA 97 00           STA A  ADATA
00272 3BFC 96 04           LDA A  INPUT    READ KEY BOARD SCAN
00273 3BFE 7F 0000         CLR    ADATA    RESET SELECT CODE
00274 3C01 43              COM A
00275 3C02 44              LSR A           FIX KEY CODE
00276 3C03 44              LSR A
00277 3C04 91 2E           CMP A  T1
00278 3C06 26 28           BNE    DT4      KEY RELEASED
00279 3C08 DF 2B           LDX    T4       LOAD TIME COUNT
00280 3C0A 09              DEX
00281 3C0B DF 2B           STX    T4
00282 3C0D 26 E5           BNE    DT1      TIME NOT UP
00283 3C0F CE 03E8         LDX    #1000
00284 3C12 DF 2B           STX    T4       SET FAST REPETE TIME
00285 3C14 7A 002D         DEC    T2
00286 3C17 26 03           BNE    DT3      SAME STEP
00287 3C19 7C 0078         INC    AT1      1.0 % STEP
00288 3C1C CE 0078  DT3    LDX    #AT1
00289 3C1F BD 75FC         JSR    FPA      ADD DELTA
00290 3C22 BD 3CDD         JSR    RANGE    ADJUST
00291 3C25 7A 0009         DEC    RSFLG
00292 3C28 BD 3AB3         JSR    MOV      MAKE STEP MOVE
00293 3C2B 7F 0009         CLR    RSFLG
```

```
00294 3C2E 20 C4          BRA    DT1        ANOTHER STEP
00295 3C30 7F 00B9 DT4    CLR    BKWRT+1     CLEAR KEY BUFFER
00296 3C33 3E             WAI               WAIT FOR KEY
00297 3C34 96 BA    DT4A  LDA A  BKKC
00298 3C36 97 2E          STA A  T1         SAVE COMMAND
00299 3C38 81 22          CMP A  #@42       EXIT?
00300 3C3A 27 18          BEQ    DT7        YES
00301 3C3C 81 25          CMP A  #@45       STOP KEY?
00302 3C3E 27 14          BEQ    DT7        YES
00303 3C40 81 14    DT5   CMP A  #@24       UP?
00304 3C42 27 0C          BEQ    DT6
00305 3C44 81 1C          CMP A  #@34       DOWN?
00306 3C46 27 08          BEQ    DT6
00307 3C48 81 23          CMP A  #@43       LEFT?
00308 3C4A 27 04          BEQ    DT6
00309 3C4C 81 24          CMP A  #@44       RIGHT?
00310 3C4E 26 E0          BNE    DT4
00311 3C50 BD 4E    DT6   BSR    SETUP      INITIALIZE
00312 3C52 20 C8          BRA    DT3
00313 3C54 86 08    DT7   LDA A  #8
00314 3C56 97 24          STA A  T11        XYFLG=Y
00315 3C58 BD 55EF DT8    JSR    STKUP      PUSH UP STACK
00316 3C5B C6 80          LDA B  #$80
00317 3C5D D7 0D          STA B  STKFLG     SET AUTO STACK
00318 3C5F 86 10          LDA A  #$10
00319 3C61 BD 39A6        JSR    GET        LOAD OFFSET
00320 3C64 86 20          LDA A  #$20
00321 3C66 BD 3973        JSR    INDX
00322 3C69 BD 75F6        JSR    FPS        MINUS POSITION
00323 3C6C BD 39AA        JSR    TEMP1
00324 3C6F 4F             CLR A
00325 3C70 BD 39A6        JSR    GET        LOAD SCALE
00326 3C73 CE 0058        LDX    #BUFF
00327 3C76 BD 7793        JSR    FPD        (POSITION-OFFSET)/SCALE
00328 3C79 CE 0000        LDX    #0
00329 3C7C DF 94          STX    XR+4
00330 3C7E DF 96          STX    XR+6       ZERO NON-SIGNIFICANT DIGITS
00331 3C80 BD 38C3        JSR    ADJST
00332 3C83 27 D3          BEQ    DT8
00333 3C85 D6 26    DT9   LDA B  T9
00334 3C87 D7 09          STA B  PSFLG      RESTORE PSFLG
00335 3C89 96 2E          LDA A  T1         KEY CODE
00336 3C8B 81 22          CMP A  #@42
00337 3C8D 27 06          BEQ    DT10       EXIT KEY
00338 3C8F 5D             TST B
00339 3C90 2A 03          BPL    DT10       NOT PROGRAM EXECUTION
00340 3C92 BD 4800        JSR    $4800      RETURN TO KEYBOARD MODE
00341 3C95 7F 0089 DT10   CLR    BKWRT+1     CLEAR KEY BUFFER
00342 3C98 86 04          LDA A  #4         IO6
00343 3C9A 97 00          STA A  ADATA      RESET CHANNEL
00344 3C9C 7F 0000        CLR    ADATA
00345 3C9F 39             RTS
00347                *
00348                *     SETUP
00349                *
00350 3CA0 CE 0078 SETUP  LDX    #AT1
00351 3CA3 BD 7489        JSR    ZEROX      CLEAR STEP BUFFER
00352 3CA6 CE 1388        LDX    #5000
00353 3CA9 DF 28          STX    T4         SET TIME LOOP
00354 3CAB 86 19          LDA A  #25
00355 3CAD 97 2D          STA A  T2         STEP COUNTER
00356 3CAF 7C 0078        INC    AT1
00357 3CB2 86 10          LDA A  #$10
00358 3CB4 97 7A          STA A  AT1+2      STEP=.1%
00359 3CB6 4F             CLR A
00360 3CB7 97 24          STA A  T11        XYFLG=X
00361 3CB9 97 B9          STA A  BKWRT+1     ZERO KEY BUFFER
00362 3CBB 96 2E          LDA A  T1
00363 3CBD C6 08          LDA B  #8
00364 3CBF 81 1C          CMP A  #@34       VERTICAL COMMAND?
00365 3CC1 22 08          BHI    ST1
00366 3CC3 D7 24          STA B  T11        XYFLG=Y
00367 3CC5 81 14          CMP A  #@24       UP?
00368 3CC7 26 06          BNE    SET2       NO
00369 3CC9 20 08          BRA    SET3
```

```
00370 3CC8 81 23    ST1     CMP A   #@43        LEFT?
00371 3CCD 26 04            BNE     SET3        NO
00372 3CCF 86 80    SET2    LDA A   #$80
00373 3CD1 97 79            STA A   AT1+1       NEGATIVE STEP
00374 3CD3 86 20    SET3    LDA A   #$20
00375 3CD5 BD 3973          JSR     INDX        POINTER TO POINT
00376 3CD8 DF 20            STX     TP7         SET IT
00377 3CDA 7E 743B          JMP     TXXR        LOAD INTO XR


00379               *
00380               *     RANGE
00381               *
00382 3CDD 96 91    RANGE   LDA A   XR+1        NEG NUMBER?
00383 3CDF 27 03            BEQ     PAN1        NO
00384 3CE1 BD 3BA1          JSR     ZERXR
00385 3CE4 96 90    PAN1    LDA A   XR
00386 3CE6 81 03            CMP A   #3          EXPONENT>3?
00387 3CE8 2F 09            BLE     RAN2        NO
00388 3CEA 86 03            LDA A   #3
00389 3CEC 97 90            STA A   XR          SET EXP TO 3
00390 3CEE CE 9999          LDX     #$9999
00391 3CF1 DF 92            STX     XR+2        SET TO MAX
00392 3CF3 DE 20    RAN2    LDX     TP7         POINTER
00393 3CF5 7E 73F3          JMP     TXRX        SAVE CORD.
00396               *
00397               *     WRITE
00398               *
00399               *     THIS ROUTINE DRAWS ASCII CHARACTERS ON THE PLOTTER,
00400               *     ACCORDING TO PARAMETERS PREVIOUSLY SET UP BY THE
00401               *     CSCALE COMMAND.  THE STRING MAY CONTAIN A PRINT
00402               *     CODE IN WHICH CASE THE XR WILL BE OUTED, AFTER
00403               *     WHICH THE STRING WILL CONTINUE.  UNDER KEYBOARD
00404               *     MODE A SINGLE CHARACTER IS PASSED AT EACH CALL.
00405               *     IF THE STOP KEY IS HIT DURING PROGRAM EXECUTION,
00406               *     THE UIP IS ADVANCED TO THE NEXT INSTRUCTION.
00407               *     USER PARAMETERS:
00408               *       XR - POSSIBLE NUMBER
00409               *
00410 3CF8 BD 3924 WRITE   JSR     INIT
00411 3CFB BD 3D9A         JSR     SAVE
00412 3CFE 96 68           LDA A   REAL
00413 3D00 D6 26           LDA B   T9
00414 3D02 26 06           BNE     LET1        PROGRAM EXECUTION
00415 3D04 81 84           CMP A   #$84
00416 3D06 27 2D           BEQ     LET8        TERMINATOR
00417 3D08 20 0A           BRA     LET3
00418 3D0A DE CA    LET1   LDX     UIP
00419 3D0C 08              INX
00420 3D0D 08              INX
00421 3D0E DF 68           STX     REAL+3
00422 3D10 BD 26    LET2   BSR     GETC
00423 3D12 27 1D           BEQ     LET7        TERMINATOR
00424 3D14 81 82    LET3   CMP A   #$82
00425 3D16 26 0A           BNE     LET4
00426 3D18 DF 6E           STX     REAL+6      PRINT KEY, SAVE POINTER
00427 3D1A 8D 30           BSR     NMBR
00428 3D1C DE 6E           LDX     REAL+6      RESTORE
00429 3D1E DF 68           STX     REAL+3
00430 3D20 20 03           BRA     LET5
00431 3D22 BD 3DB2 LET4    JSR     OUTC        OUT THE CHARACTER
00432 3D25 96 26   LET5    LDA A   T9
00433 3D27 27 0C           BEQ     LET8        KEYBOARD
00434 3D29 96 09           LDA A   RSFLG
00435 3D2B 2B E3           BMI     LET2        NO STOP KEY
00436 3D2D 8D 09   LET6    BSR     GETC
00437 3D2F 26 FC           BNE     LET6
00438 3D31 09      LET7    DEX
00439 3D32 09              DEX                 FIX INSTR POINTER
00440 3D33 DF CA           STX     UIP
00441 3D35 7E 390E LET8    JMP     EXIT        DONE
```

```
00443                    *
00444                    *     GETC
00445                    *
00446                    *     LOADS THE CHARACTER POINTED TO BY THE CHARACTER
00447                    *     POINTER (T13) INTO ACCA AND THEN BUMPS THE POINTER.
00448                    *     SETS THE 'Z' BIT IF A TERMINATOR WAS LOADED.
00449                    *     SETS THE 'Z' BIT ON MEMORY OVERFLOW.
00450                    *
00451 3D38 DE 6B   GETC    LDX     REAL+3
00452 3D3A A6 00           LDA A   0,X       CHARACTER
00453 3D3C 08              INX               BUMP
00454 3D3D DF 6B           STX     REAL+3
00455 3D3F 81 84           CMP A   #$84      STRING END
00456 3D41 27 02           BEQ     GET1      YES
00457 3D43 9C 0B           CPX     EOPM
00458 3D45 39       GET1   RTS


00460                    *
00461                    *     PRNTX
00462                    *
00463                    *     OUTPUTS THE CURRENT VALUE OF THE XR, IN MACHINE
00464                    *     FORMAT TO THE PLOTTER.  'NMBR' IS ENTRY POINT
00465                    *     USED BY 'LETTER'.
00466                    *     USER PARAMETERS:
00467                    *       XR - NUMBER TO OUT
00468                    *
00469 3D46 BD 3924 PRNTX  JSR     INIT
00470 3D49 8D 4F          BSR     SAVE
00471 3D4B 4F             CLR A
00472 3D4C 97 68   NMBR   STA A   REAL      ENTRY POINT FLAG
00473 3D4E BD 396E        JSR     RESXR
00474 3D51 BD 5CBC        JSR     FRMT+$14
00475 3D54 CE 0058        LDX     #BUFF
00476 3D57 DF 6B          STX     REAL+3
00477 3D59 86 20          LDA A   #$20      LOAD BLANK FOR TESTS
00478 3D5B C6 11   NUM1   LDA B   #17
00479 3D5D 91 66          CMP A   BUFF+14   EXPONENT NULL?
00480 3D5F 26 02          BNE     NUM2      NO
00481 3D61 C0 04          SUB B   #4        YES, FIX COUNT
00482 3D63 8D D3   NUM2   BSR     GETC
00483 3D65 5A             DEC B
00484 3D66 81 20          CMP A   #$20      LEADING BLANKS
00485 3D68 27 F9          BEQ     NUM2
00486 3D6A 81 2D          CMP A   #$2D      LEADING '-'?
00487 3D6C 27 06          BEQ     NUM2A     YES
00488 3D6E 5C             INC B
00489 3D6F 09             DEX
00490 3D70 DF 6B          STX     REAL+3
00491 3D72 86 20          LDA A   #$20      LEADING BLANK
00492 3D74 D7 1F   NUM2A  STA B   TP6S      LENGTH
00493 3D76 8D 3A   NUM3   BSR     OUTC
00494 3D78 7A 001F        DEC     TP6S
00495 3D7B 27 18          BEQ     NUM4      DONE
00496 3D7D 96 09          LDA A   RSFLG
00497 3D7F 2A 14          BPL     NUM4      STOP KEY, ABORT
00498 3D81 BD 396E        JSR     RESXR
00499 3D84 BD 5CBC        JSR     FRMT+$14
00500 3D87 86 20          LDA A   #$20
00501 3D89 91 65          CMP A   BUFF+13   POSITIVE EXPONENT?
00502 3D8B 26 04          BNE     NUM4-4    NO
00503 3D8D 86 2B          LDA A   #$2B
00504 3D8F 97 65          STA A   BUFF+13   STUFF A '+'
00505 3D91 8D A5          BSR     GETC
00506 3D93 20 E1          BRA     NUM3
00507 3D95 96 68   NUM4   LDA A   REAL      ENTRY FLAG
00508 3D97 2A 9C          BPL     LET8      SINGLE CALL
00509 3D99 39       NUM5  RTS


00511                    *
00512                    *     SAVE
00513                    *
```

```
00514 3D9A 7F 0024 SAVE   CLR     T11
00515 3D9D 86 20          LDA A   #$20
00516 3D9F BD 39A6        JSR     GET
00517 3DA2 CE 0078        LDX     #AT1
00518 3DA5 8D 08          BSR     TXR
00519 3DA7 86 28          LDA A   #$28
00520 3DA9 BD 39A6        JSR     GET
00521 3DAC CE 0080        LDX     #AT2
00522 3DAF 7E 73F3 TXR    JMP     TXRX


00524                     *
00525                     *    OUTC
00526                     *
00527                     *    THIS ROUTINE OUTS THE ASCII CHARACTER SENT IN ACCA.
00528                     *    THE ASCII BIAS IS STRIPPED OFF AND THE RESULT
00529                     *    MULTIPLIED BY THREE TO FORM AN INDEX INTO CTBL.
00530                     *    (SEE CTRL HEADING FOR TABLE LAYOUT).
00531                     *
00532 3DB2 84 7F    OUTC   AND A   #$7F
00533 3DB4 81 03           CMP A   #3          TAB,LINE,SHIFT?
00534 3DB6 2F E1           BLE     NUM5        YES, IGNOR IT
00535 3DB8 CE 3F55         LDX     #CTBL
00536 3DBB 80 24           SUB A   #$24        DOLLAR SIGN?
00537 3DBD 27 07           BEQ     OUT0        YES
00538 3DBF 80 02           SUB A   #2
00539 3DC1 2F 53           BLE     OUT6        OUT OF RANGE
00540 3DC3 16              TAB
00541 3DC4 48              ASL A               TIMES THREE
00542 3DC5 1B              ABA
00543 3DC6 DF 2A    OUT0   STX     T5          BASE POINTER
00544 3DC8 9B 2B           ADD A   T4          SET TABLE POINTER
00545 3DCA 97 2B           STA A   T4
00546 3DCC 86 06           LDA A   #6
00547 3DCE 97 2C           STA A   T3          STEP COUNT
00548 3DD0 97 60           STA A   BUFF+8
00549 3DD2 DE 2A    OUT1   LDX     T5          TABLE POINTER
00550 3DD4 A6 00           LDA A   0,X
00551 3DD6 08              INX
00552 3DD7 D6 60           LDA B   BUFF+8
00553 3DD9 27 07           BEQ     OUT2        RIGHT HALF
00554 3DDB 09              DEX
00555 3DDC 46              ROR A
00556 3DDD 46              ROR A
00557 3DDE 46              ROR A               LEFT HALF
00558 3DDF 46              ROR A
00559 3DE0 C6 FF           LDA B   #$FF
00560 3DE2 5C       OUT2   INC B
00561 3DE3 D7 60           STA B   BUFF+8
00562 3DE5 DF 2A           STX     T5
00563 3DE7 84 0F    OUT3   AND A   #$F         MASK COMMAND
00564 3DE9 27 2B           BEQ     OUT6        END OF CHARACTER
00565 3DEB 5F              CLR B
00566 3DEC 4A              DEC A
00567 3DED 27 20           BEQ     OUT4
00568 3DEF 4A              DEC A
00569 3DF0 27 3D           BEQ     DELB
00570 3DF2 4A              DEC A
00571 3DF3 27 39           BEQ     DELC
00572 3DF5 4A              DEC A
00573 3DF6 27 35           BEQ     DELD
00574 3DF8 4A              DEC A               TRANSLATE AND EXECUTE
00575 3DF9 27 31           BEQ     DELE
00576 3DFB 4A              DEC A
00577 3DFC 27 34           BEQ     DELF
00578 3DFE 4A              DEC A
00579 3DFF 27 32           BEQ     DELG
00580 3E01 4A              DEC A
00581 3E02 27 33           BEQ     DELH
00582 3E04 4A              DEC A
00583 3E05 27 2F           BEQ     DELI
00584 3E07 4A              DEC A
00585 3E08 27 30           BEQ     LIFT
00586 3E0A 4A              DEC A
```

```
00587 3E0B 27 32          BEQ    SEQ1      SPECIAL SEQUNCE 'ACE'
00588 3E0D 20 3B          BRA    SEQ2      SPECIAL SEQUENCE 'GH'
00589 3E0F 8D 13   OUT4   BSR    POINT
00590 3E11 7A 002C OUT4A  DEC    T3
00591 3E14 26 BC          BNE    OUT1      MORE
00592 3E16 86 03   OUT6   LDA A  #3
00593 3E18 5F             CLR B
00594 3E19 8D 38          BSR    VECTR     CHARACTER SPACE
00595 3E1B BD 3B8F        JSR    PENU
00596 3E1E BD 3A94        JSR    MOVQ      NEXT CHAR START POINT
00597 3E21 7E 3D9A        JMP    SAVE
00599                *
00600                *    POINT
00601                *
00602                *    SETS UP NEXT POINT ON CHARACTER AND MAKES MOVE TO IT.
00603                *    IF THE POINT IS OUT OF RANGE. THE PEN IS LIFTED AND
00604                *    THE CHARACTER IS ABORTED.  ACCA AND ACCB MUST BE SET
00605                *    UP BEFORE CALL.
00606                *
00607 3E24 8D 20   POINT  BSR    VECTR
00608 3E26 BD 3A94        JSR    MOVQ
00609 3E29 7E 3B95        JMP    PEND



00611                *
00612                *    VECTOR SET UP ROUTINES
00613                *
00614                *    SETS UP ACCA AND ACCB FOR DESIRED VECTOR TO
00615                *    NEXT POINT OF CHARACTER.
00616                *
00617 3E2C 4C      DELE   INC A
00618 3E2D 4C      DELD   INC A
00619 3E2E 5C      DELC   INC B
00620 3E2F 5C      DELB   INC B
00621 3E30 20 DD          BRA    OUT4
00622 3E32 5C      DELF   INC B
00623 3E33 4C      DELG   INC A
00624 3E34 20 01          BRA    DELH
00625 3E36 5C      DELI   INC B
00626 3E37 4C      DELH   INC A
00627 3E38 20 D5          BRA    OUT4
00628 3E3A BD 3B8F LIFT   JSR    PENU
00629 3E3D 20 D2          BRA    OUT4A
00630 3E3F BD E3   SEQ1   BSR    POINT
00631 3E41 4F             CLR A
00632 3E42 C6 02          LDA B  #2
00633 3E44 8D DE          BSR    POINT
00634 3E46 4F             CLR A
00635 3E47 5F             CLR B
00636 3E48 20 E2          BRA    DELE
00637 3E4A 86 02   SEQ2   LDA A  #2
00638 3E4C 5C             INC B
00639 3E4D 8D D5          BSR    POINT
00640 3E4F 5F             CLR B
00641 3E50 4F             CLR A
00642 3E51 20 E0          BRA    DELG
00644                *
00645                *    VECTR
00646                *
00647                *    CALCULATES THE ACTUAL VECTOR LENGTHS AND ADDS THEM
00648                *    TO THE CHARACTER BASE COORDINATES.
00649                *    VECTOR LENGTH = (NUMBER OF .5 DELTA INCREMENTS)*
00650                *      (0.5)*(HEIGHT)*(DIRECTION SIGN).
00651                *    COORDINATE = (BASE COORDINATE)+(VECTOR LENGTH)*
00652                *      (1/ASPECT: WIDTH ONLY)*(PAPER RATIO: X ONLY).
00653                *
00654 3E53 97 2E    VECTR  STA A  T1        WIDTH
00655 3E55 D7 2D          STA B  T2        HEIGHT
00656 3E57 86 3C          LDA A  #$3C
00657 3E59 8D 67          BSR    INX2
00658 3E5B 0F 20          STX    TP7
00659 3E5D 86 FF          LDA A  #$FF
00660 3E5F 4C      VEC1   INC A
00661 3E60 97 6D          STA A  REAL+5    PASS ID FLG
```

```
00662 3E62 BD 38A1          JSR     ZERXR
00663 3E65 DE 20            LDX     TP7
00664 3E67 A6 00            LDA A   0,X
00665 3E69 97 24            STA A   T11         DIRECTION
00666 3E6B A6 01            LDA A   1,X
00667 3E6D 97 91            STA A   XR+1        SET SIGN
00668 3E6F 08               INX
00669 3E70 08               INX
00670 3E71 DF 20            STX     TP7         UPDATED
00671 3E73 96 2E            LDA A   T1
00672 3E75 D6 6D            LDA B   REAL+5
00673 3E77 27 02            BEQ     VEC2
00674 3E79 96 2D            LDA A   T2
00675 3E7B 4D       VEC2    TST A
00676 3E7C 27 12            BEQ     VEC4        ZERO
00677 3E7E C6 10            LDA B   #$10
00678 3E80 44               LSR A
00679 3E81 24 08            BCC     VEC3        1.0 DELTA
00680 3E83 C6 15            LDA B   #$15
00681 3E85 4D               TST A
00682 3E86 26 06            BNE     VEC3        1.5 DELTA
00683 3E88 C6 FF            LDA B   #$FF
00684 3E8A D7 90            STA B   XR
00685 3E8C C6 50            LDA B   #$50        0.5 DELTA
00686 3E8E D7 92    VEC3    STA B   XR+2
00687 3E90 8D 05    VEC4    BSR     HEIGHT
00688 3E92 96 6D            LDA A   REAL+5
00689 3E94 27 C9            BEQ     VEC1        AGAIN
00690 3E96 39               RTS
00692                *
00693                *      HEIGHT
00694                *
00695                *      MULTIPLIES THE DELTA INCREMENT IN XR BY THE HEIGHT
00696                *      AND ASPECT RATIO IF IT IS A WIDTH CALCULATION.
00697                *      TIMES THE PAPER SIZE RATIO FOR X-VECTORS.
00698                *      THEN ADDS VECTOR TO THE CHARACTER BASE COORDINATE
00699                *      AND SETS THE PLOTTER COORDINATE REGISTER.
00700                *
00701 3E97 86 38    HEIGHT LDA A  #$38
00702 3E99 8D 27           BSR    INX2
00703 3E9B BD 7735         JSR    FPM         TIMES SIZE
00704 3E9E 96 6D           LDA A  REAL+5      WIDTH?
00705 3EA0 26 07           BNE    HEI1        NO
00706 3EA2 86 30           LDA A  #$30
00707 3EA4 8D 1C           BSR    INX2        ASPECT
00708 3EA6 BD 7735         JSR    FPM
00709 3EA9 CE 0080 HEI1    LDX    #AT2
00710 3EAC 96 24           LDA A  T11         X-VECTOR?
00711 3EAE 26 0A           BNE    HEI2        NO
00712 3EB0 86 40           LDA A  #$40
00713 3EB2 8D 0E           BSR    INX2        PAPER SIZE RATIO
00714 3EB4 BD 7735         JSR    FPM
00715 3EB7 CE 0078         LDX    #AT1
00716 3EBA BD 75FC HEI2    JSR    FPA
00717 3EBD 86 20           LDA A  #$20
00718 3EBF 7E 39A1 SET1    JMP    SET


00720 3EC2 7E 3975 INX2    JMP    INDX1
00722                *
00723                *
00724                *      CSIZE
00725                *
00726                *      SETS UP CHARACTER PLOTTING PARAMETERS.
00727                *
00728                *      USER PAZAMETERS:
00729                *          XR - ANGLE (0,1,2,3)
00730                *          YR - HEIGHT (%)
00731                *          ZR - ASPECT RATIO (HEIGHT/WIDTH)
00732                *          TR - PAPER SIZE RATIO (HEIGHT/WIDTH)
00733                *
00734                *      THE ANGLE SPECIFICATION IS 0=ZERO DEGREES,
00735                *      1=90 DEGREES, 2=180 DEGREES, AND 3=270 DEGREES.
00736                *      THE DIRECTION TABLE IS SET UP AS FOLLOWS:
00737                *      WIDTH DIRECTION, WIDTH SIGN, HEIGHT DIRECTION,
```

```
00738                *     AND HEIGHT SIGN.
00739                *     0 - 00,00,08,00 (HEX)
00740                *     1 - 08,00,00,80 (HEX)
00741                *     2 - 00,80,08,80 (HEX)
00742                *     3 - 08,80,00,00 (HEX)
00743                *     THE HEIGHT IS STRICTLY A PERCENTAGE OF PLOTTING
00744                *     AREA.  THE EXPONENT OF VALUE IS INCREMENTED BY TWO
00745                *     AND STORED AS THE ABSOLUTE VALUE.
00746                *     THE ASPECT RATIO IS STORED AS A RECIPROCAL OF THE
00747                *     PASSED VALUE FOR CONVIENCE.
00748                *     ONE, TWO, THREE, OR FOUR PARAMETERS MAY BE CHANGED
00749                *     BY CLEARING THE NEXT HIGHER STACK REGISTER.
00750                *
00751                *
00752 3EC5 BD 3924 CSIZE  JSR     INIT
00753 3EC8 96 90         LDA A  XR
00754 3ECA 27 06         BEQ    CSC2
00755 3ECC 86 06   CSC1  LDA A  #6
00756 3ECE 97 06         STA A  ERROR     ERROR EXIT
00757 3ED0 20 67         BRA    CSC6
00758 3ED2 D6 92   CSC2  LDA B  XR+2
00759 3ED4 54            LSR B
00760 3ED5 54            LSR B
00761 3ED6 54            LSR B
00762 3ED7 54            LSR B
00763 3ED8 C1 03         CMP B  #3
00764 3EDA 2E F0         BGT    CSC1      TOO BIG
00765 3EDC 86 3C         LDA A  #$3C
00766 3EDE BD E2         BSR    INX2
00767 3EE0 4F            CLR A
00768 3EE1 A7 00         STA A  0,X
00769 3EE3 A7 01         STA A  1,X
00770 3EE5 A7 02         STA A  2,X
00771 3EE7 A7 03         STA A  3,X
00772 3EE9 86 08         LDA A  #8
00773 3EEB 54            LSR B
00774 3EEC 24 08         BCC    CSC3      0 OR 2
00775 3EEE A7 00         STA A  0,X
00776 3EF0 86 80         LDA A  #$80
00777 3EF2 5D            TST B
00778 3EF3 27 0D         BEQ    CSC4      1
00779 3EF5 A7 01         STA A  1,X       3
00780 3EF7 20 0B         BRA    CSC5
00781 3EF9 A7 02   CSC3  STA A  2,X
00782 3EFB 86 80         LDA A  #$80
00783 3EFD 5D            TST B
00784 3EFE 27 04         BEQ    CSC5      0
00785 3F00 A7 01         STA A  1,X       2
00786 3F02 A7 03   CSC4  STA A  3,X
00787 3F04 96 9A   CSC5  LDA A  YR+2
00788 3F06 27 31         BEQ    CSC6      EXIT
00789 3F08 86 38         LDA A  #$38
00790 3F0A BD B6         BSR    INX2
00791 3F0C 96 98         LDA A  YR
00792 3F0E 4C            INC A
00793 3F0F 4C            INC A
00794 3F10 A7 00         STA A  0,X
00795 3F12 96 9A         LDA A  YR+2
00796 3F14 A7 02         STA A  2,X       SET HEIGHT (SIZE)
00797 3F16 96 9B         LDA A  YR+3
00798 3F18 A7 03         STA A  3,X
00799 3F1A 96 A2         LDA A  ZP+2
00800 3F1C 27 1B         BEQ    CSC6      EXIT
00801 3F1E CE 00A0       LDX    #ZR
00802 3F21 BD 743B       JSR    TXXR
00803 3F24 BD 73E6       JSR    RECIP     1/ASPECT
00804 3F27 86 30         LDA A  #$30
00805 3F29 BD 94         BSR    SET1
00806 3F2B 96 AA         LDA A  TP+2
00807 3F2D 27 0A         BEQ    CSC6      NULL PARAMETER
00808 3F2F CE 00A8       LDX    #TR
00809 3F32 BD 743B       JSR    TXXR      LOAD PAPER RATIO
00810 3F35 86 40         LDA A  #$40
```

```
00811 3F37 8D 86          BSR    SETI     SAVE
00812 3F39 7E 390E CSC6   JMP    EXIT     DONE



00814 3F4D                ORG    $3F4D
00815                *    FLOATING POINT CONSTANT  C2=50.
00816                      OPT    NG
00817 3F4D 01             FCB    1,0,$50,0,0,0,0,0
00818                      OPT    G
00820                *
00821                *    CTBL
00822                *
00823                *    TABLE ORGANIZED THREE BYTES PER CHARACTER,
00824                *    TWO COMMANDS PER BYTE.
00825                *
00826                *    COMMANDS:
00827                *      0 - END OF CHARACTER
00828                *      1 - MOVE TO 'A'
00829                *      2 - MOVE TO 'B'
00830                *      3 - MOVE TO 'C'
00831                *      4 - MOVE TO 'D'
00832                *      5 - MOVE TO 'E'
00833                *      6 - MOVE TO 'F'
00834                *      7 - MOVE TO 'G'
00835                *      8 - MOVE TO 'H'
00836                *      9 - MOVE TO 'I'
00837                *      A - LIFT PEN
00838                *      B - MOVE IN SEQUENCE 'A','C','E'
00839                *      C - MOVE IN SEQUENCE 'F','G'
00840                *
00841                *      C  D  E
00842                *
00843                *      B  I  F
00844                *
00845                *      A  H  G
00846                *



00848                      OPT    NG
00849                *    DOLLAR SIGN
00850 3F55 17        CTBL  FCB    $17,$35,$48
00851                *    QUOTE
00852 3F58 49             FCB    $49,0,0
00853                *    LEFT PAREN
00854 3F5B 43             FCB    $43,$18,0
00855                *    RIGHT PAREN
00856 3F5E 45             FCB    $45,$78,0
00857                *    MULTIPLY
00858 3F61 15             FCB    $15,$A3,$70
00859                *    ADD-PLUS
00860 3F64 84             FCB    $84,$A2,$60
00861                *    COMMA
00862 3F67 98             FCB    $98,0,0
00863                *    MINUS
00864 3F6A 26             FCB    $26,0,0
00865                *    PERIOD
00866 3F6D 80             FCB    $80,0,0
00867                *    SLASH
00868 3F70 15             FCB    $15,0,0
00869                *    ZERO
00870 3F73 87             FCB    $87,$15,0
00871                *    ONE
00872 3F76 84             FCB    $84,0,0
00873                *    TWO
00874 3F79 35             FCB    $35,$69,$17
00875                *    THREE
00876 3F7C 35             FCB    $35,$69,$C1
00877                *    FOUR
00878 3F7F 32             FCB    $32,$65,$70
00879                *    FIVE
00880 3F82 17             FCB    $17,$92,$35
```

```
00881                *     SIX
00882 3F85 2C                FCB     $2C,$80,0
00883                *     SEVEN
00884 3F88 35                FCB     $35,$80,0
00885                *     EIGHT
00886 3F8B 7B                FCB     $7B,$62,$C0
00887                *     NINE
00888 3F8E 62                FCB     $62,$35,$71
00889                *     COLON
00890 3F91 9A                FCB     $9A,$80,0
00891                *     SEMI-COLON
00892 3F94 4A                FCB     $4A,$98,0
00893                *     LESS THAN
00894 3F97 42                FCB     $42,$80,0
00895                *     EQUAL
00896 3F9A 17                FCB     $17,$A2,$60
00897                *     GREATER THAN
00898 3F9D 46                FCB     $46,$80,0
00899                *     QUESTION MARK
00900 3FA0 35                FCB     $35,$9A,$80
00901                *     '@'
00902 3FA3 B7                FCB     $B7,$89,$60
00903                *     A
00904 3FA6 B6                FCB     $B6,$2C,0
00905                *     B
00906 3FA9 B9                FCB     $B9,$2C,$10
00907                *     C
00908 3FAC 54                FCB     $54,$28,$70
00909                *     D
00910 3FAF 13                FCB     $13,$46,$81
00911                *     E
00912 3FB2 7B                FCB     $7B,$A2,$90
00913                *     F
00914 3FB5 BA                FCB     $BA,$29,0
00915                *     G
00916 3FB8 9C                FCB     $9C,$80,0
00917                *     H
00918 3FBB 13                FCB     $13,$26,$57
00919                *     I
00920 3FBE 48                FCB     $48,0,0
00921                *     J
00922 3FC1 21                FCB     $21,$75,0
00923                *     K
00924 3FC4 13                FCB     $13,$A5,$27
00925                *     L
00926 3FC7 31                FCB     $31,$70,0
00927                *     M
00928 3FCA 13                FCB     $13,$95,$70
00929                *     N
00930 3FCD 13                FCB     $13,$75,0
00931                *     O
00932 3FD0 B7                FCB     $B7,$10,0
00933                *     P
00934 3FD3 B6                FCB     $B6,$20,0
00935                *     Q
00936 3FD6 7B                FCB     $7B,$79,0
00937                *     R
00938 3FD9 B6                FCB     $B6,$29,$70
00939                *     S
00940 3FDC 17                FCB     $17,$62,$35
00941                *     T
00942 3FDF 35                FCB     $35,$48,0
00943                *     U
00944 3FE2 31                FCB     $31,$75,0
00945                *     V
00946 3FE5 38                FCB     $38,$50,0
00947                *     W
00948 3FE8 31                FCB     $31,$97,$50
00949                *     X
00950 3FEB 15                FCB     $15,$A3,$70
00951                *     Y
00952 3FEE 39                FCB     $39,$59,$80
```

```
00953                   *     Z
00954 3FF1 35                   FCB     $35,$17,0
00955                   *     FLOATING POINT CONSTANT  C1=1/9999.
00956 3FF4 FC                   FCB     $FC,0,$10,0,$10,0,$10,0,$10
00957                   *     UP ARROW
00958 3FFD 24                   FCB     $24,$64,$80
00959                           OPT     G
```

```
00961                   *
00962                   *     ADDRESS JUMP TABLE
00963                   *
00964 3805                      ORG     $3805
00965 3805 3CFB                 FDB     WRITE
00966 380F                      ORG     $380F
00967 380F 3807                 FDB     DGTZ4
00968 3819                      ORG     $3819
00969 3819 3805                 FDB     DGTZ3
00970 381B 3D46                 FDB     PRNTX
00971 381D 3EC5                 FDB     CSIZE
00972 381F 3800                 FDB     DGTZ5
00973 3821 3803                 FDB     DGTZ1
00974 3823 3802                 FDB     DGTZ2
00977                           END
```

SYMBOL TABLE

ADATA	0000	ACTL	0001	BDATA	0002	BCTL	0003	INPUT	0004	IOIN	0005
ERROR	0006	TGL	0007	UFLG	0008	RSFLG	0009	EOM	000A	EOPM	000B
STKFLG	000D	RND	000E	DIGFLG	000F	W2	0010	W1	0011	SFLG	0012
DCNTR	0013	TP1	0014	TP1S	0015	TP2	0016	TP2S	0017	TP3	0018
TP3S	0019	TP4	001A	TP4S	001B	TP5	001C	TP5S	001D	TP6	001E
TP6S	001F	TP7	0020	TP7S	0021	T13	0022	T12	0023	T11	0024
T10	0025	T9	0026	T8	0027	T7	0028	T6	0029	T5	002A
T4	002B	T3	002C	T2	002D	T1	002E	ISTK	002F	ISTACK	0051
TA	0052	SPGM	0054	EXTRA	0056	BUFF	0058	REAL	0068	IMAG	0070
AT1	0078	AT2	0080	W	0088	XR	0090	YR	0098	ZR	00A0
TR	00AB	LSTX	00B0	BKWRT	00B8	BKKC	00BA	SOL7	00C6	UPP	00C8
UIP	00CA	ALPHA	00CC	IO1	00CD	IO2	00D0	IT7	00D3	FLAG	00D5
TPOS	00D6	FILE	00D7	AR	00D8	BR	00E0	CR	00E8	DR	00F0
ER	00F8	SDBB	00BA	MT	7E00	TERMN7	003D	IMED	0040	PARCD	00C0
PAREX	0080	NTBL	0000	DOTS	5EC0	PRTDRV	602D	FRMT	5CA8	BLANK	5D75
LOMSG	5780	ROLLD	55B2	ROLLU	57F1	PSD	550A	TXL	55E9	STKUP	55EF
MAD	749B	CMP	74AA	NOR	74D6	TXW	7424	TXXR	743B	EXXR	7452
ARSR	753B	OVUNF	75B6	OVERF	75DD	XR0	740A	XRNINE	75C8	UNDRF	75F1
IMULT	7689	QDG	7669	FPA	75FC	FPS	75F6	FPM	7735	FPD	7793
FPAEX	7630	FPMEX	7780	LSHIFT	7521	ZEROX	7489	XZEROQ	7416	XZERO2	7417
RECIP	73E6	TXPX	73F3	CONST	6800	FPDBRC	6898	TAN	68A9	ATN	69C3
DSZERO	6A46	NTLN	6A58	EXPN	6AC9	SIN	6B94	COS	6B9A	ASIN	6BF2
ACOS	6BF7	PH1	6C5D	PH2	6C8D	PH3	6D34	PH4	6DD0	LSFT8	6E47
SQRT	6E65	MADA	6F2C	CMPR	53E4	IOUPX	6F52	LOG10	6FA7	YUPX	6FE9

RTOP	732R	PTOR	7386	ADJST	38C3	EXIT	390E	INIT	3924	RESXR	396E
INOX	3973	INDX1	3975	SET	39A1	GET	39A6	TEMP1	39AA	MOVQ	3A94
MOV	3AB3	PENU	3B8F	PEND	3B95	ZERXR	3BA1	DGTZ5	3BD0	DGTZ2	3BD2
DGTZ1	3BD3	DGTZ3	3BD5	DGTZ4	3BD7	DT0	3BE5	DT1	3BF4	DT3	3C1C
DT4	3C30	DT4A	3C34	DT5	3C40	DT6	3C50	DT7	3C54	DT8	3C58
DT9	3C85	DT10	3C95	SETUP	3CA0	ST1	3CC8	SET2	3CCF	SET3	3CD3
RANGE	3CDD	RAN1	3CE4	RAN2	3CF3	WRITE	3CF8	LET1	3D0A	LET2	3D10
LET3	3D14	LET4	3D22	LET5	3D25	LET6	3D2D	LET7	3D31	LET8	3D35
GETC	3D38	GET1	3D45	PRNTX	3D46	NMBR	3D4C	NUM1	3D5B	NUM2	3D63
NUM2A	3D74	NUM3	3D76	NUM4	3D95	NUM5	3D99	SAVE	3D9A	TXR	3DAF
OUTC	3DB2	OUT0	3DC6	OUT1	3DD2	OUT2	3DE2	OUT3	3DE7	OUT4	3E0F
OUT4A	3E11	OUT6	3E16	POINT	3E24	DELE	3E2C	DELD	3E2D	DELC	3E2E
DELB	3E2F	DELF	3E32	DELG	3E33	DELI	3E36	DELH	3E37	LIFT	3E3A
SEQ1	3E3F	SEQ2	3E4A	VECTR	3E53	VEC1	3E5F	VEC2	3E78	VEC3	3E8E
VEC4	3E90	HEIGHT	3E97	HEI1	3EA9	HEI2	3EBA	SET1	3EBF	INX2	3EC2
CSIZE	3EC5	CSC1	3ECC	CSC2	3ED2	CSC3	3EF9	CSC4	3F02	CSC5	3F04
CSC6	3F39	CTRL	3F55								

TOTAL ERRORS 1

CALCULATOR OPERATION

GENERAL DESCRIPTION

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 unit, or peripheral I/O units and stored, in modified form, as program steps in the program storage section of the read-write memory. An operational description of the calculator is therefore now set forth with specific reference to the perspective view of the calculator as shown in FIG. **1** and the plan view of the keyboard as shown in FIG. **3**, except as otherwise indicated.

The calculator employs reverse polish notation (RPN) language that involves the use of an operational stack of four registers referred to herein as the X, Y, Z and T registers. Simple arithmetic operations are performed by placing data in the X and Y registers and then actuating one of the arithmetic operator keys. The calculated result is placed in the X register.

The 16-character display **14** shows each number entered from the keyboard **10** and each calculated result. The 16-column thermal printer **16** can be called upon to print the data currently displayed. In addition, the display **14** and printer **16** are valuable programming aids.

The dynamic range of the calculator is from $-9.999999999 \times 10^{99}$ through $9.999999999 \times 10^{99}$. When a calculated result lies outside this range, the message OVERFLOW is printed. All calculations are to twelve places, but the accuracy depends upon the function performed. Ordinary arithmetic functions are accurate to one count in the 12th digit.

In addition to the four working registers X, Y, Z, and T comprising the operational stack, the basic calculator includes ten permanent data storage registers and a 472-step program memory. The program memory may be expanded to 2008 program steps by adding read-write memory to the calculator, as discussed hereinabove. Additional data storage registers may be assigned by the user when needed.

The calculator may be operated by means of a program stored on an external magnetic tape cartridge placed into the magnetic tape cassette reading and recording unit **12**. External magnetic tape cartridges can store either pre-recorded factory programs or programs written by the user.

By inserting optional plug-in I/O ROMs into one or both of the slots provided therefor on the rear panel of the calculator, the calculator may be interfaced to one or more peripheral I/O units. These include, for example, the Hewlett-Packard 9862A X-Y Plotter the Hewlett-Packard 9863A Paper Tape Reader, the Hewlett-Packard 9884A Paper Tape Punch, the Hewlett-Packard 9864A Digitizer, and the Hewlett-Packard 9866A Page Printer. In addition, the calculator may be interfaced to most BCD-compatible instrumentation and, through the use of a universal interface bus manufactured by Hewlett-Packard Company, to nearly all bus-compatible instrumentation.

KEYBOARD OPERATIONS

FIG. **3** illustrates the layout of the calculator keyboard and includes the mnemonic designation or designations associated with each of the keys. Many of the keys have both a primary function designated by the mnemonic inside the key outline and a secondary function designed by the mnemonic above the key outline, with the exception of an ENTER ↑ key, a DECIMAL POINT key, and a group of keys A–O located in the lower left-hand corner of the keyboard, all of whose alternate functions are designated by mnemonics below the key outline. With the exception of keys A–O, these alternate functions may be entered by prefacing actuation of the desired key by actuation of a blank key located in the upper right-hand portion of the keyboard (hereinafter referred to as the BLANK key). The alternate functions indicated below keys A–O all represent programming functions that are entered by merely actu-

ating their associated keys whenever the calculator is in a program mode of operation. No preceding actuation of the BLANK key is required in connection with this group of functions.

Some of these and other keys of the keyboard are associated with characters located below the key outline. These characters may be printed and are automatically entered into the calculator by actuation of their associated keys when the calculator is in an ALPHA mode of operation, described in detail hereinafter.

The two switches on the far right-hand side of the keyboard are used to select the various printer and calculator operating modes. The printer switch is set to the ALL position to automatically print each keyboard operation. To conserve paper, the printer switch may be placed in the OFF position. The printer switch may be placed in the NORMAL position to enable printing during program entry but to suppress printing during execution of functions. The NORMAL position is useful to avoid manually switching the printer off to suppress oftentimes undesired printing during function execution. The calculator will print various messages regardless of the setting of the printer switch. These include messages indicative of peripheral I/O unit status and error messages indicative of incorrect operations. A list of printed error messages is included hereinafter. The operating mode switch is placed in the RUN position when executing functions from the keyboard of when running a program stored in read-write memory.

The 16-character display indicates a calculator busy condition during lengthy keyboard or program executions by displaying a hyphen at each character position of the display. A displayed number may be printed at any time by simply actuating the PRINT key.

The number entry keys of the keyboard are arranged as on an adding machine. Numbers are entered into the calculator one digit at a time from left to right and may include a decimal point. Before entering a second number into the calculator, the first one is saved by actuating the enter ↑ key. To enter a negative number into the calculator, the +/- key is actuated after keying in the number. The +/- key may simply be actuated to change the sign of a calculated result. Large numbers may be entered in scientific notation by actuating the E EX key operationsl entry of the mantissa and the exponent.

The X register may be cleared anytime during number entry by actuating the CLX key. All four registers of the operaational stack may be cleared by actuating the CLEAR key. The RESET key may be used to clear a key sequence that has not been completed.

Calculations involving two numbers and one arithmetic operator are performed by keying in the first number, saving it in the Y register by actuating the ENTER ↑ key, keying in the second number, and finally actuating the selected arithmetic operator key. The result, appearing in the X register, is displayed.

Calculations involving more than one arithmetic operation are performed by keying in the first number and saving it and then keying in subsequent numbers each followed by the appropriate arithmetic operator. Only the first number keyed in need be saved by actuating the ENTER ↑ key. Each subsequent number keyed in after actuation of an arithmetic operator key is automatically saved.

The last number entered into the calculator before actuation of an operator key is automatically stored in a location called LAST X. That number may be recalled into the X register and used after actuation of that operator key by actuating the BLANK key followed by the LAST X key. Such a recall also causes an automatic ENTER ↑ just like keying in a number after actuation of an arithmetic operator key. The LAST X location is not cleared by actuation of either the CLEAR or CLX keys.

Simple arithmetic operations like those discussed above require the use of only the X and Y registers of the operational stack. More complicated functions require the use of either or both of the Z and T registers of the operational stack. The four registers of the operational stack may be thought of as being arranged vertically, the X, Y, Z, and T registers being stacked from bottom to top, respectively. Numbers entered from the keyboard are automatically placed in the X register. A subsequent, actuation of the ENTER ↑ key duplicates that number in the Y register while moving the number previously stored in the Y register to the T register. The number previously stored in the T register is lost. Actuation of the ENTER ↑ key thus moves the contents of the stack registers up. Similarly, when an arithmetic operator key is actuated, the result of the operation is placed in the X register and the previous contents of the Z and T registers are moved down to the Y and Z registers, respectively.

The contents of the stack registers may be manipulated by some control keys of the keyboard. The X⇄Y key exchanges the contents of the X and Y registers without disturbing the Z and T registers. The R ↓ key rolls the contents of each stack register down to the next register, the contents of the X register being placed in the T register. A similar operation is performed in the up direction by the R ↑ key. Actuation of the BLANK key followed by the STACK key prints the contents of each of the stack registers in sequence from T to X.

Printed and displayed numbers normally appear in a fixed format with two digits to the right of the decimal point. To select another fixed format, the BLANK key is actuated and is followed by actuation of the FIX key and one of the numeric keys **0** through **9**. The numeric key indicates the number of digits to the right of the decimal point. When a calculated result of a number entered from the keyboard is too large for the present fixed format, a scientific format is automatically selected by the calculator. The user may select either a standard scientific format or a special scientific format for displayed and printed numbers. The standard scientific format is selected by actuating the BLANK key followed by the SCI key followed by one of the numeric keys. As in the case of a fixed format, the numeric key indicates the number of digits to the right of the decimal point. The special scientific format is selected by actuating the BLANK key followed by the SCI3 key followed by a numeric key. When this format has been selected, printed and displayed numbers appear with exponents that are always even multiples of three.

Ten fixed data storage registers are automatically provided the user for storing numbers representing, for example, intermediate results of calculations. Each of these registers can store one number and is accessed for storage and recall by actuating the STO and RCL keys followed by one of the keys A through J that designate each of the registers. Additional data storage registers may be assigned by the user as needed.

All data storage registers are automatically cleared when the calculator is turned on. To clear fixed data storage registers A through J, without disturbing other

registers, the STO key is actuated and is followed by actuation of the CLEAR key. Additional data storage registers assigned by the user are cleared by storing zero in them or by using a clear routine set forth hereinafter.

Arithmetic operations may be performed directly on the contents of the X register and a data storage register without first recalling the stored number. The result is placed in the data storage register without disturbing the contents of the X register. These operations are performed by actuating the STO key followed by the desired arithmetic operator key followed by the key or keys designating the desired storage register.

Indirect store and recall operations may be performed by specifying a register designation that contains the designation of the register in which the desired data is stored. These operations are performed just like direct store and recall operations, except that the RCL key is actuated before the key designating the intermediate register is actuated. Only the absolute integer value of the contents of the intermediate register is used as the indirect register number. The sign and any fractional part are ignored.

The above-described register arithmetic and indirect storage operations may be combined to perform indirect register arithmetic. The general key sequence is the STO key followed by the desired arithmetic operator key followed by the RCL key followed by the storage register designation key. In each case, the chosen arithmetic operation is performed on the contents of the X register and the contents of the register designated by the contents of the register designated in the key sequence. The result is placed in the register designated in the key sequence and the X register remains unchanged.

Additional data storage registers may be assigned by the user as required. These data storage registers are formed from the portion of user read-write memory not already filled with program instructions. The additional data storage registers are assigned by actuating the numeric keys representing the desired number of additional data storage registers followed by the BLANK key followed by the STO key. If the unfilled portion of user read-write memory is not large enough to accept a particular data storage assignment, an error message MEMORY OVERFLOW will be printed, and the attempted assignment will be ignored. Additional data storage registers remain assigned until the assignment is either changed or the calculator is turned off. The contents of previously assigned data storage registers are not altered by a subsequent assignment so long as they lie within the subsequent assignment. If there is not enough program memory available to accept a specified register assignment, an error message MEMORY OVERFLOW will be printed, and the assignment will be ignored. The user may assign up to 250 additional data storage registers when the calculator is configured with the optional read-write memory **103** shown in FIG. **4**. These registers are labelled from **000** through **249**. The portion of user read-write memory not assigned as data storage registers is available for storing program steps. This arrangement results in more efficient use of the read-write memory than is possible in those calculators and computers having separate fixed areas for program and data storage. The above key sequence for assigning additional data storage registers may be executed either manually or from program control, thus giving the user flexibility in reconfiguring the user read-write memory to accommodate his specific requirements at any point in time. This eliminates the often encountered problem in prior art calculators and computers of having too much program storage and not enough data storage or vice versa. In addition to providing this memory definition flexibility for the user, the calculator provides complete protection for each storage area. That is, the calculator prevents the user from storing data into a storage register that has not been previously assigned as a data storage register and similarly prevents him from storing program steps into the assigned data storage registers. If the user attempts to store data into a data storage register that has not previously been assigned, an error message ILLEGAL ADDRESS will be printed.

A block of assigned data storage registers may be cleared by first deleting the block and by then reassigning it. For example, if data storage registers **000** through **024** have been assigned and it is desired to clear registers **010** through **024**, this may be accomplished by first assigning registers **000** through **009** and by then assigning registers **000** through **024**. The result is that assigned data storage registers **010** through **024** are cleared while registers **000** through **009** remain unaltered.

In addition to the four simple arithmetic functions previously discussed, there are twenty-four scientific functions represented by keys on the keyboard. These keys are grouped in a block and are located near the right-hand side of the keyboard just to the left of the printer and control switches. Each of these scientific function keys has primary, alternate, and inverse functions associated with it. A primary function is designated by merely actuating the desired key. The alternate function associated with a particular key, indicated by nomenclature above the key, is selected as discussed above by first actuating the BLANK key. The inverse function is designated by first actuating the f^{-1} key.

Functions involving angles, such as trigonometric functions and angular conversions, may be performed in either decimal degrees, radians or metric grads. The calculator is automatically set to accept angle values in degrees when it is turned on. Thus, the user may specify any of the three units to angular measurement by actuating the BLANK key followed by the numeric keys, **1**, **2**, and **3** for degrees, radians, and grads, respectively.

A conversion from decimal angular units to degrees, minutes, and seconds is available from the keyboard by actuating the BLANK key followed by the → D.MS key. A conversion from degrees, minutes, and seconds to the equivalent decimal form of the angular units currently selected is available by actuating the BLANK key followed by the D.MS → key. When performing the conversion from decimal degrees, radians or grads to degrees, minutes, and seconds, the result returned to the X register includes the decimal portion of seconds.

PRINTER CONTROL

An ALPHA mode of operation may be selected by actuating the CALL ALPHA key twice. This key is also used, by actuating it just once, to access control of peripheral I/O units connected to the calculator. After actuating this key twice, the message ALPHA appears in the display to indicate that the ALPHA mode has been selected. The user may then selectively actuate the various alphanumeric keys of the keyboard to form a desired message. After the last alphanumeric character of the message has been keyed in, the ALPHA mode may be terminated and the message printed by actuating the CALL ALPHA key once more. In the event the desired message is sixteen or more characters in length,

printing will automatically occur as every sixteenth character is keyed in.

A number of keys of the keyboard take on control functions in the ALPHA mode of operation. For example, the ENTER ↑ key becomes a NEW LINE control key. Actuation of this key causes the calculator to print all alphanumeric characters that have been keyed in and to then advance the printer paper to the next line. Successive actuations of the NEW LINE key cause the calculator to advance the printer paper one line at a time. The X⇄Y key becomes a SPACE control key during the ALPHA mode of operation and is used to insert spaces into an alphanumeric character message. Actuation of the PRINT key at a selected position within an alphanumeric message causes the calculator to print the number currently stored in the X register at that position within the alphanumeric message. The number appears right justified on the line unless alphanumeric characters follow the number. It is also printed in the particular fixed or scientific number format currently selected. This feature is useful for printing labels and calculated results on the same line. The printed number always appears on the same line as the alphanumeric message, provided there is sufficient space for it within the 16-column print field. If the space available is too small, the entire number is printed on the next succeeding line of print. In the event the user makes an error while keying in an alphanumeric message, the ALPHA mode may be cancelled without printing the message by placing the control switch in the lower right-hand corner of the keyboard in the PRGM position and then back to the RUN position. This switch movement has no effect on the numbers stored in the four registers of the operational stack or on the calculator memory.

PROGRAMMING

In addition to manual execution of commands entered from the calculator keybard, the calculator may also be operated automatically by a program stored in the user read-write memory. The program is stored in the form of a modified version of the keycodes associated with each of the keys, as shown in FIG. **29**. Unlike some calculators in which only a portion of the keys are programmable, the present calculator permits the user to include within a program every key sequence associated with manual operation of the calculator. In addition, the calculator keyboard includes a block of keys representing program control functions. These include subroutine branching and labelling keys, qualifier keys, and looping keys for repeating program segments automatically. These program control keys are located in the left-hand third of the keyboard area. They include the alternate functions shown below keys A through O. The basic read-write memory has capacity for storing 472 program steps at locations **0000** through **0471**. By adding an optional read-write memory, program storage can be increased to 2008 program steps. Each step comprises one program instruction that may be either a single key actuation such as the + key or the PRINT key or a combined key sequence such as the STO key followed by the A key or the BLANK key followed by the SIN key. Appropriate key sequences are combined automatically as a program is entered, and a single instruction code representing the entire sequence is stored in user read-write memory. This instruction code is built internally by a series of firmware syntax tables. These tables define a number of key sequences that are valid at any given time along with their corresponding instruction codes. This arrangement is unlike prior art calculators in which each key actuation occupies a separate storage location in memory. The arrangement in the present calculator is advantageous in that a larger program may be stored in the same amount of memory. It is also advantageous in that program execution is more efficient because less syntax checking is required at that time. This results from the fact that a partial syntax check has, in effect, been performed at the time the program was entered to recognize those key sequences resulting in a single internal instruction code.

The program storage portion of the user read-write memory of the calculator may be cleared before entering each new program without altering the contents of any of the data storage registers or the operational stack registers. This is accomplished by placing the control switch in the PRGM position and sequentially actuating the K and N keys. The calculator may be turned off to clear the entire read-write memory.

The calculator includes an internal program counter for determining which program step is displayed, printed or executed. Many programming keys and instructions control the operation of the program counter, allowing the user to enter, edit, run, and record programs. The program counter may be set to any desired step when the calculator is in the PRGM mode by actuating the GO TO key followed by numeric keys representing the program step location in memory. Just as in specifying assigned data storage registers, only the significant digits representative of the memory location need by keyed in if those key actuations are immediately followed by actuation of a non-numeric key or the DECIMAL POINT key. For example, to set the program counter to location **0025**, the calculator is placed in the RUN mode and the key sequence GO TO **25**. is entered, or the key sequence GO TO **0025** is entered. The calculator is then placed in the PRGM mode, and the current step location **0025** together with the number of step locations between the current step location and the end of the program storage portion of the user read-write memory are displayed. The program counter may be manually incremented or decremented while entering a program by actuating the STEP and BKSTEP keys. The program counter is automatically incremented as each program step is entered. It is automatically set to step location **0000** whenever the END key is actuated while the calculator is in the RUN mode, whenever the calculator is turned on, and whenever the program storage portion of user read-write memory is erased.

A program is entered by setting the program counter to the desired beginning step location and by then placing the calculator in the PRGM mode and keying in the program. As each program step is entered the printer lists the step instruction and prints the next step location. The last step instruction of each program must be END.

The printer automatically lists each step location and step instruction during program entry if the printer switch is in the NORM position. When the calculator is in the PRGM mode, the printer also lists each step location as the program counter is manually incremented or decremented. Any portion of a stored program may be listed by setting the program counter to the desired beginning step location and actuating the LIST key. The listing may be stopped by actuating the

RUN STOP key. Listing automatically stops when an END step instruction is encountered.

After the program has been entered, it may be executed by placing the calculator in the RUN mode, setting the program counter to the beginning of the program, and actuating the RUN STOP key. Program execution continues until either a STOP or END instruction is encountered. The user may halt program execution at any time by actuating the RUN STOP key.

Labels may be used as a program aid to name a location in a program. The label instruction is located immediately before the program area to which it refers. The program counter may then be set to the label location by an appropriate branching instruction such as GO TO. Labelling provides a method of addressing program segments independent of step location in memory. A time-saving technique often used when entering and debugging programs is to use labels whenever possible for branching. Then, as program steps are inserted or deleted the branching instructions do not need to be altered. Once the program is operating satisfactorily is the labels originally used in connection with the branching instructions may be replaced with absolute step locations in memory.

A label may be entered by actuating the LABEL key followed by one or more alphanumeric keys to designate the label. For example, if it is desired to establish a label **01** located at step location **0050**, the program counter is first set to that step location and the key sequence LABEL **01** is entered. A labelled program segment may be executed under program control manually from the keyboard. To execute a program segment labelled **06** from the keyboard, it is only necessary to enter the key sequence GO TO LABEL **6** RUN.

Execution of a branching instruction sets the program counter to a designated step location in memory. Program execution then automatically continues from that step location. Both absolute and computed branching instructions are available to the user. An absolute branching instruction causes the program counter to be set to a fixed step location that may be specified as a label. Actuation of the GO TO key followed by a numeric step location or actuation of the GO TO key followed by the LABEL key followed by alphanumeric keys representative of a label are exemplary of absolute branching instructions.

A computed branch instruction results in the program counter being set to a step location indicated by the current contents of the X register. Depending upon the branching instruction, the absolute integer portion of the contents of the X register indicates either a step location or a numeric label. The general sequence for entering computed branching instructions is GO TO X or GO TO LABEL X. This arrangement for computed branching statements represents an advantage over prior art arrangements wherein the user was required to predefine a limited set of destination addresses for each computed branching instruction used and then compute the one address of the set to be used at a given point in time. In the present calculator, the user merely places the destination step location in the X register in advance of execution of the branching instruction. In addition, the user is given added flexibility in that the branch may be to either a computed fixed step location or to a computed label.

IF instructions cause the calculator to make logical comparisons between the contents of the X and Y registers or the current state of some program flags described hereinafter. If the comparison is true, the next program step instruction is executed. However, if the comparison is false, the next program step instruction is skipped. The program step instruction next following an IF instruction usually, but not necessarily, is a branching instruction. Eight IF instructions and their corresponding key sequences are shown in Table 3 below.

Table 3

$X < Y$?	IF $X < Y$
$X = Y$?	IF $X = Y$
$X \geqq Y$?	IF $X \geqq Y$
$X < 0$?	IF −
$X > 0$?	IF +
$X = 0$?	IF 0
IS FLAG N SET?	IF SFG N
IS FLAG N CLEAR?	IF CFG N

A subroutine is a sequence of program instructions that may be used repeatedly, perhaps in several different programs, yet need be stored only once in the memory. A program can branch to, or call, a subroutine at any time through use of a GO SUB instruction. Then, after the subroutine has been executed, a RETURN instuction located at the end of the subroutine causes execution to resume at the step instruction next following the GO SUB instruction. The GO SUB instruction calls a subroutine by specifying either a step location or a label, or it can be in computed branch form similar to the computed GO TO instruction. Subroutines may be nested to a depth of seven. Returns are made on a last in, first out basis, so the returning order is always opposite of the calling order.

FOR-NEXT instructions permit the repetition of any instruction sequence. The FOR and NEXT instructions form a loop with the instruction sequence to be repeated located between them. Each FOR-NEXT instruction is associated with a pair of data storage registers. Data register pairs A & F, B & G, and C & H are available for this purpose. The first register of each pair is specified in the FOR instruction and is the loop counter. The second register of each pair holds the final value. When register pairs A & F an B & G are used, the loop counter is incremented by unity each time the loop is executed. When register pair C & H is used, however, the loop counter is incremented as specified by the contents of register D. FOR-NEXT instructions may be nested but, since there are only three register pairs available, they may be nested only three deep.

Flags may be employed as programmable indicators to allow the calculator to make decisions or to advise the user of certain program conditions. Each of the flags is either set or cleared, and each may be set or cleared either manually from the keyboard or under program control. In addition, all flags are cleared by actuating the END key, by executing an END instruction within a program or by turning the calculator on. Eight flags are available in the calculator. Flags **1** through **4** are for general program use, while flags **5** through **8** have dedicated functions. As generally used, a flag is set by some program sequence or event. Then, later in the program, the state of the flag can be checked to determine a subsequent activity. Flags 1-4 may be set by actuating the SFG CFG key once followed by a numeric key to designate the flag. Those flags may be cleared by actuating the SFG CFG key twice followed by the appropriate numeric designation. Flags **5** and **6** are used to intercept certain error messages. When flag **6** is set, the suppressable error messages such as OVERFLOW will

not be printed. Instead, flag 5 is automatically set whenever a suppressable error occurs. Flag 7 is automatically set whenever a STOP instruction is executed. If data is entered before program execution is continued, flag 7 is cleared. However, if no data is entered before program execution is continued, flag 7 remains set. Flag 8 may be toggled from the set to clear states by successive actuations of the SFG CFG key during program execution.

Several of the keys on the calculator keyboard are useful in performing editing functions on a program stored in the user read-write memory. When a program does not run as expected, the first step usually taken by the user is to examine a listing of the program. To list an entire program, the program counter is set to the first step location of the program, and the LIST key is actuated. A portion of a program may be listed by setting the program counter to the desired step location and then actuating the LIST key. The RUN STOP key may be actuated to halt the listing.

One method often used to check a defective program is to execute it, one instruction at a time. This may be done, while the calculator is in the RUN mode, by setting the program counter to the first step location of the program and then successively actuating the STEP key. Each time the STEP key is actuated, the current instruction is executed, the program counter is advanced to the next step instruction to be executed, and the executed result is displayed.

To change a program step instruction, the program counter is set to the desired step location, the PRGM mode is selected, the new instruction is entered from the keyboard, and the calculator is returned to the RUN mode. If the new instruction requires two program steps, while the old instruction required only one step, the calculator will automatically shift the remainder of the program by one step to accommodate the new instruction and will automatically renumber any affected branching instructions. Similarly, if the new instruction requires only one step, while the old instruction required two steps, the calculator will shift the remainder of the program by one step and renumber any affected branching instructions.

Program instructions may be deleted by setting the program counter to the step location of the unwanted instruction, placing the calculator in the PRGM mode, and then actuating the DELETE key. The calculator automatically moves the remainder of the program to fill the empty step and renumbers any affected branching instructions. An entire block of instructions may be deleted by setting the program counter to the first unwanted step, placing the calculator in the PRGM mode, and then actuating the DELETE key once for each instruction in the sequence. Each time an instruction is deleted the new instruction moved by the calculator to that step location will be printed unless the printer is turned off.

One or more instructions may be inserted into a program by first setting the program counter to the step location at which the first new instruction is to be placed. The calculator is then placed in the PRGM mode, the INSERT key is actuated, and the desired new instruction is keyed in. The insertion operation is terminated by placing the calculator in the RUN mode or actuating any one of the editing keys except MEMORY or DELETE. The calculator automatically renumbers any branching instructions affected during the insertion operation.

The program storage portion of user read-write memory may be cleared by placing the calculator in the PRGM mode and sequentially actuating the MEMORY and DELETE keys. This operation fills the program area with NOP instructions, which designate no operation. An NOP key is available on the keyboard for allowing the user to enter NOP instructions in his program. This arrangement is desirable, for example, in cases wherein the user wishes to presently reserve a step location for possible subsequent entry of an executable instruction.

Instructions forming an alphanumeric message to be printed may also be edited using the various keys just described. The only difference is that the calculator must first be placed in the ALPHA mode of operation, as described in the section above entitled PRINTER CONTROL. One exception is that the calculator must not be in the ALPHA mode when the user is attempting to delete alphanumeric instructions. Otherwise, actuation of the DELETE key will enter the alpha character O.

TAPE OPERATIONS

The magnetic tape cassette unit **12** built into the calculator allows the user to make permanent records on an external magnetic tape cartridge of his programs and data blocks. Each such program or data block may be subsequently read back into the calculator memory as often as desired. Five keys, all programmable, for controlling the operation of magnetic tape cassette unit **12** are provided on the left-hand portion of keyboard **10**. Their primary functions are labelled LOAD, REWIND, RECORD, LIST, and L. Each external tape cartridge has capacity for about 96,000 program steps or the contents of about 12,000 data storage registers. A RECORD slide located on each tape cartridge may be positioned to prevent accidental erasure of information stored on a cartridge by inhibiting execution of a RECORD instruction.

The magnetic tape cassette unit routinely checks to insure that all the information being loaded into the calculator memory from an external tape cartridge corresponds exactly to the information originally recorded. If an error is detected during a data loading or program loading operation, an attempted reloading is made. If the information cannot be successfully loaded after three such automatic attempts, the loading operation is halted and an error message CHECKSUM ERROR is printed. Typical causes for such an error are badly worn or partially erased tapes or a dirty tape head.

Before programs or data can be recorded onto a blank tape cartridge, the cartridge must be initialized by performing one or more MARK TAPE instructions. Each MARK TAPE instruction records a block of empty files onto one track of the tape. Two tracks are available on each tape cartridge, and each track may be initialized and used for information storage and retrieval independent of the other. A primary track may be used by specifying a positive file number in each tape instruction. A secondary track may similarly be used by specifying a negative file number. A blank area is associated with the beginning of each file to serve as a file separator. A file identifier includes information relating to a particular file such as a file number, a file type, an absolute file size, a current file size, etc. A portion of each tape file called the file body is used for actual program or data storage. The absolute file size specified

in the MARK TAPE instruction determines the size of this file body.

Each MARK TAPE instruction, entered by sequentially actuating the BLANK key and the MARK key, initializes one track of a tape cartridge by storing a block of empty files together with appropriate file identifiers. The integer portion of numbers stored in the Z, Y, and X registers specifies, respectively, the size of each file, the number of files in the block, and the number designator for the first file. The size of each file is expressed in program steps. To determine the file size in program steps needed to hold a desired number of data storage registers, the number of data storage registers is merely multiplied by eight.

After the specified number of files has been marked, an extra file is automatically marked, and the tape is positioned in front of the extra file. The extra file is marked to facilitate marking additional files at a later time and hence has no file body. Programs or data may now be stored in each file marked, or more files may be marked beginning with the extra file. Files are marked and designated in numerical order, beginning with file 0 for files marked on the primary track or file −0 for files marked on the secondary track.

The MARK TAPE instruction has the same format for both new and used tape cartridges. However, when marking files on a used tape, it is important to mark over, or erase, all old files. This will prevent unexpected results. Old files may be erased by simply marking new files in sufficient quantity or sufficient size to extend beyond the old files. Or, they may be erased by specifying a negative number of files in any MARK TAPE instruction. For example, if −1 is stored in the Y register at the time a MARK TAPE instruction is executed, a single file will be marked and the remainder of the specified track will automatically be erased.

An IDENTIFY instruction, entered by sequentially actuating the BLANK key and the Indent key, transfers the file identifier information associated with a designated file into the registers of the operational stack. The number of the desired file is stored in the X register prior to execution of the instruction. Following execution of the instruction, a number corresponding to the file type is stored in the T register, the number of steps in use is stored in the Z register, the originally marked file size is stored in the Y register, and, of course, the file number remains stored in the X register. The various file types and their corresponding number designators are shown in Table 4 below.

Table 4

0	PROGRAM FILE
1	SECURED PROGRAM
2	DATA FILE
3	PRE-RECORDED FACTORY PROGRAM
4	SECURED PRE-RECORDED FACTORY PROGRAM
5	EMPTY FILE
6	EXTRA FILE

For the user's convenience, the contents of the four registers of the operational stack together with the alpha labels FILE, TYPE, USED, and MAX are automatically printed when an IDENTIFY instruction is executed from the keyboard.

Execution of a RECORD instruction, entered by actuating the RECORD key, records the contents of the program storage portion of user read-write memory, from a current step location through an END instruction, on a designated tape file. If no END instruction is encountered, the remainder of the program storage portion of user read-write memory is recorded. Before execution of the instruction, the desired beginning step location should be stored in the Y register, and the number of the desired file should be stored in the X register. If the designated file is too small or the tape is protected, the RECORD instruction is cancelled, and an error message is printed.

Execution of a LOAD instruction, entered by actuating the LOAD key, loads programs or data from a desired tape file into the user read-write memory. The file type determines whether programs or data will be loaded. Before execution of a LOAD instruction, the desired beginning step location in memory should be stored in the Y register, and the number of the desired tape file should be stored in the X register. If the file is of the wrong type or there is not enough read-write memory available, the LOAD instruction is cancelled, and an error message is printed.

A LOAD & GO instruction, entered by actuating the LD & GO key, provides a programmable method for automatically loading and executing a specified program. Before execution of the instruction, the beginning step location in memory should be stored in the Y register, and the number of the desired file should be stored in the X register. An extremely long program may be separated into segments, each segment being recorded into a separate tape file. A LOAD & GO instruction may be added to the end of each program segment to automatically call and execute the program segments in succession.

Execution of a RECORD DATA instruction, entered by sequentially actuating the BLANK and RECORD keys, records the content of a block of numbered data storage registers into a specified tape file. Before execution of the instruction, the number of data storage registers to be recorded should be stored in the Z register, the first register number should be stored in the Y register, and the file number should be stored in the X register. If the specified registers have not previously been assigned, if the file is too small or of the wrong type, or if the tape is protected, the RECORD DATA instruction is cancelled, and an error message is printed.

As stated above, the LOAD instruction is used for loading both data and programs into the calculator. The file type determines whether programs or data will be loaded. Before loading data, the starting data storage register number should be stored in the Y register, and the file number should be stored in the X register. The data is loaded, register-by-register, beginning with the starting register. If the file is of the wrong type or if an insufficient number of data storage registers has been assigned, the instruction is cancelled, and an error message is printed.

Execution of a VERIFY instruction, entered by sequential actuation of the BLANK and VERIFY keys, compares the information recorded on a tape file with the program or data presently stored in the calculator memory. To verify a program file, the starting step location should be stored in the Y register and the file number be stored in the X register. To verify a data file, the number of the data storage register should be stored in the Y register and the file number should be stored in the X register. The VERIFY instruction is most easily executed directly after a loading or recording operation, since the proper numbers are already stored in the X and Y registers. If the information in the file is not

identical to that stored in the user read-write memory, one of the error messages VERIFY FAILED or CHECKSUM ERROR is printed. Neither of these two errors will cause program execution to halt when flag **6** is set. In that case, program flag **5** is automatically set by either error.

A RECORD SECURED instruction, entered by sequentially actuating the CALL and RECORD keys, provides a method for recording private programming on tape. Execution of the instruction records a program into a specified file, like the RECORD PROGRAM instruction, except that the file type is designated as type **1**. Before execution of the instruction, the starting step location should be stored in the Y register, and the number of the desired file should be stored in the X register. Execution of the RECORD SECURED instruction does not affect the contents of memory. A secured program can be loaded back into the calculator just as any other program and then executed in the normal manner. However, once a secured program has been loaded into the calculator, any attempt to list, record, or edit the program will result in the error message SECURED MEMORY being printed. When a secured program has been loaded into the calculator memory, all other programs stored in the memory are automatically secured. Data storage registers, however, are not affected. The secured memory may be cleared by erasing the memory or by turning the calculator off.

An AUTOSTART mode of calculator operation is provided to automatically load a program stored in tape file **0** into the calculator memory and initiate execution of that program, all in response to placing the calculator power switch **22** in the ON position. The AUTOSTART mode of operation is selected by positioning the calculator mode switch located in the lower right-hand corner of the keyboard in the AUTOSTART position. This switch is interrogated by the calculator firmware. If the switch is found to be in the AUTOSTART position, the tape is searched for file **0** and the file type is interrogated. If file **0** is of type **0** or **1**, the file is automatically loaded into the calculator memory and execution is initiated at step location **0000**. If any errors occur during loading of this file, the AUTOSTART mode is cancelled, an error message is printed, and the calculator is returned to the RUN mode. The AUTOSTART mode is advantageous in that it provides automatic memory definition without intervention on the part of a possibly unskilled user. In addition, it provides automatic resumption of execution of a program after restoration of operating power following, for example, a power blackout.

The group of keys A through O comprises a group of special function keys that may be defined to call and execute functions defined by the user. Each such defined function is, in effect, a subroutine beginning with a label and ending with a RETURN instruction. Blank overlays are provided for this group of keys to allow the user to identify each identified function. Each defined function may be executed from the keyboard by merely actuating the desired key, or it may be called during program execution through use of a GO SUB instruction.

Each special function key is defined by entering the instructions comprising the defined function into the calculator memory. Each defined function includes a label and a RETURN instruction, just as in the case of subroutines as discussed hereinabove. Each defined function may be stored, beginning at any chosen step location, in the user read-write memory. Special functions may be nested to a depth of seven. Before nested functions are called, the END key should be actuated to reset the nesting counter.

Improper data entries, improper key or program instruction syntax, and improper calculations are all indicated to the user through printed error messages. Unlike prior art calculators that employed numeric error notes and required the use of a look-up table to convert a numeric error note to a meaningful description of the error, the error messages printed by the present calculator are in themselves descriptive of the error that is indicated, thus eliminating the need for a user look-up table. ASCII characters corresponding to each possible error message are stored in the calculator memory. Upon detection of an error and selection of a corresponding error number by the ROM execution routines, an error output routine transmits the ASCII characters forming the error message associated with the selected error number to the printer. A list of the possible error notes that may result from various improper calculator operations together with the corresponding sources of error is provided in Table 5 below. An asterisk to the left of an error message indicates that it may be suppressed through use of the calculator flags described hereinabove.

ERROR MESSAGES

*OVERFLOW — Number or result exceeds calculating range.
*SQRT OF NEG #
*DIVISION BY ZERO
*LOG OF #—ϕ
*NO I/O DEVICE — Peripheral I/O unit not connected.
ILLEGAL ADDRESS — Improper step location or storage register specified.
ILLEGAL ARGUMENT — Mathematically incorrect function argument specified.
MEMORY OVERFLOW — Program instruction, data storage register assignment, or program or data loaded from tape exceeds available memory.
LABEL NOT FOUND
GO SUB OVERFLOW — More than seven subroutines or special functions nested.
MISSING GO SUB
KEY NOT DEFINED — Special function just called is not defined.
IMPROPER SYNTAX
MISSING FOR STMT
*CHECKSUM ERROR — Unrecognizable information being read from tape.
FILE TOO SMALL
*VERIFY FAILED — Program or data in tape file is not identical to that stored in memory.
*WRONG FILE TYPE
FILE NOT FOUND
END OF TAPE — End of tape or tape break has been detected during execution of a MARK FILE instruction.
CARTRIDGE OUT — The magnetic tape cassette unit contains no tape cartridge.
PROTECTED TAPE — The cartridge RECORD slide is positioned to prevent MARK and RECORD operations.
SECURED MEMORY — An attempt has been made to list, edit or record a secured program.
PAPER OUT —≦0 printer paper supply is exhausted.

Table 5

PLOTTER PLUG-IN I/O ROM

By means of a plotter I/O ROM that may be plugged into one of the two peripheral I/O receptacles **18** on the rear panel of the calculator, an X-Y plotter, such as the Hewlett-Packard 9862A, may be interfaced to the calculator. The combination of the calculator and an X-Y plotter provides a system capable of producing hard copy graphic solutions to sophisticated problems. The functions of the plotter are controlled by the calculator through the use of instructions that may be executed from the keyboard or under the program control. The plotter may be used in conventional ways to plot curves representing mathematical functions, to draw histograms or charts, and to draw alphanumeric and special characters. In addition, the plotter/calculator combination may be used as a digitizer to perform functions not previously available in calculator/plotter systems. In the digitizer mode of operation, the calculator/plotter system may be used to digitize lines and figures into scaled coordinate values.

In the digitizer mode of operation, the user may position the plotter pen over various points on the plotter bed by way of calculator keyboard instructions. Once the plotter pen is precisely positioned over the desired point, the plotter transmits the coordinates of that point to the calculator. This information may then be used by the calculator to compute line length, closed area, or other parameters requiring scaled point data.

The plot area, as set by the graph-limit controls on the plotter and a SCALE instruction, is divided into 1000 scaled units in each coordinate direction. For example, a 10-inch square scaled plot has a digitizing resolution of 0.01 inches. The coordinate values resulting from digitizing a point are stored in the registers of the calculator operational stack and are referred to the origin chosen in the SCALE instruction.

The SCALE instruction establishs the full-scale values, in user units, for a given plot area. X_{min}, X_{max}, Y_{min}, and Y_{max} correspond exactly to the respective horizontal and vertical limits of the plotting area established through adjustment of the graph-limit controls on the plotter. This instruction also establishes the point, on or off the plot area, where the origin of the coordinate system is located.

In preparation for executing the SCALE instruction, the chosen values of X_{min}, X_{max}, Y_{min}, and Y_{max} should be stored in the T, Z, Y, and X registers, respectively. The SCALE instruction may then be executed by sequentially actuating the CALL, **1**, and F keys. The scale values selected by the user will remain in effect until either a new SCALE instruction is executed or the calculator is turned off. It is important to be certain that the values X_{min}, X_{max}, Y_{min}, and Y_{max} are entered into the proper stack registers. If they are not, the error message ILLEGAL ARGUMENT will be printed. When the calculator is turned on, an automatic value assignment is made so that $X_{min} = Y_{min} = 0$ and $X_{max} = Y_{max} = 9999$. These automatic limit values may, of course, be altered by subsequent execution of a SCALE instruction.

The digitizing mode of operation may be selected by executing any one of four pen direction key sequences that include CALL 1 E, CALL 1 J, CALL 1 N, and CALL 1 O. Once the digitizing mode has been selected, it is only necessary to actuate any one of the direction keys E, J, N or O to move the plotter pen up, down, left or right, respectively. Each time one of these direction keys is actuated, the plotter pen moves an incremental distance equal to one user unit in the direction specified. By not releasing a direction key the pen may be moved in multiple increments at an increasing speed to more efficiently position the plotter pen over the desired point.

The digitizing mode of operation may be cancelled by actuating either the M key or the RUN STOP key. At this time the coordinate values of the current pen position are entered by the calculator/plotter system into the X and Y registers. In the event the digitizing mode was selected under program control, actuation of the M key restarts the program. Actuation of the RUN STOP key halts execution of the program.

An EXIT instruction, entered by sequentially actuating the CALL, **1**, and M keys, enters the coordinate values of the current pen position into the X and Y registers. This instruction is independent of the digitizing mode of operation but is useful whenever the current X and Y coordinates of the pen position are needed for reference.

The digitizer mode of operation may be understood in detail with reference to the flow charts of FIGS. 74O–Q and the corresponding portions of the firmware listing. The main routine of FIG. 74O has five entry points, called by keys E, J, M, N, and O. These entry points build the equivalent key code for future reference and serve to initialize various pointers. In the event the routine is entered by the M key (EXIT instruction), the routine immediately calculates the X and Y coordinates of the current pen position from information in the plotter registers and returns control of the calculator without altering the status of a flag RSFLG. If entry was via one of the keys E, J, N or O, the plotter pen is lifted, and the operational stack is moved up so that calculations can be done internally in the X register. A routine SETUP is called to set the pen stepping increment to ten plotter absolute units, the initial wait time to about 0.5 seconds, the direction of the step as determined from the key code, and various flags internal to the routine. The SETUP routine then checks for either release of the entry key or a new key actuation. If the key is held down long enough to overcome the 0.5 seconds of wait time then the step increment is added to the current pen position, the wait time is increased to approximately one second, and a count of the number of steps in the chosen direction is started. After 25 steps in the same direction, the step increment is increased to 100 units or 1 percent of the plot area, the pen is moved to a new position, and the loop is continued for as long as the key is held down. If a given step will result in the pen moving out of the plot area, an appropriate boundary coordinate is substituted. When the key is released, the input buffer is cleared, and the routine waits for another direction key actuation. If the next key actuation is the RUN STOP key, the current pen coordinates are calculated and stored in the X and Y registers.

PLUG-IN GENERAL I/O ROM

A general I/O ROM may be plugged into one of the receptacles **18** on the rear panel of the calculator to provide an 8-bit parallel, character serial interface for connecting a wide variety of peripheral I/O units to the calculator. This ROM transfers data in a half-duplex fashion and provides buffer storage for each character or byte of data. Although the calculator itself handles only ASCII-coded information, the general I/O ROM

can transfer data in any 8-bit binary code. These codes are then converted to ASCII code by means of a conversion program.

Several instructions that may be executed either from the calculator keyboard or under program control are associated with the general I/O ROM. These instructions comprise routines and subroutines stored within the general I/O ROM itself and may be understood in detail with reference to pages 187–220 of the calculator firmware listing. The reader may also wish to refer to FIGS. **4**, **20–23**, and **50–53** together with their associated detailed descriptions hereinabove as an aid to understanding the cooperative relationship between the instructions associated with the general I/O ROM and the remainder of the calculator hardware. Each optional plug-in I/O ROM that may be plugged into the rear panel of the calculator is associated with a separate numeric select code that must be specified in each I/O instruction relating to that I/O ROM. The select code associated with the general I/O ROM is **2**. This select code may be altered by those persons skilled in the art.

A DATA instruction, entered into the calculator by sequential actuation of the CALL, **2** and O keys, is employed for selecting either positive true or negative true logic for the I/O data lines. This selection is made by appropriately setting the sign of a number stored in the X register prior to execution of the DATA instruction. Negative true logic is automatically selected when the calculator is turned on.

A FLAG instruction, entered, into the calculator by sequential actuation of the CALL, **2**, and N keys, is employed for setting the logic level and handshake mode for the calculator FLG line. A handshake control line $\overline{\text{ECH}}$ may be disabled by entering zero in the X register and executing the FLAG instruction. Similarly, the $\overline{\text{ECH}}$ line may be enabled by placing the number one in the X register and executing the FLAG instruction. Line $\overline{\text{ECH}}$ is automatically disabled when the calculator is turned on. The logic level of the FLG line is set by the sign of the number entered into the X register prior to execution of the FLAG instruction. Negative true logic is automatically selected when the calculator is turned on.

A WRITE instruction, entered into the calculator by sequential actuation of the CALL, **2**, and C keys, is employed for transmitting the sign, digits, and decimal point of the number currently stored in the X register to the peripheral I/O unit. The number appears right justified in a field set by the current number format of the calculator. Carriage return and line feed characters are automatically transmitted following the number.

A WRITE X, entered into the calculator by sequential actuation of the CALL, **2**, and A keys, performs the same function as the WRITE instruction except that transmission of the carriage return and line feed characters is suppressed.

The carriage return and line feed characters together with a space are employed as delimiters in connection with WRITE instructions. The space characters are used to fill the data field, and the carriage return and line feed characters are used to terminate the data field.

A FIELD instruction, entered into the calculator by sequential actuation of the CALL, and D keys, is employed to set the data field width in effect for WRITE and WRITE X instructions. This data field width is automatically set to sixteen characters when the calculator is turned on. A field width of 1 through 127 characters may be selected by entering the field width into the X register and executing the FIELD instruction. If the number transmitted by a WRITE or WRITE X instruction is too large to be accommodated within the designated field, a field of $ characters is transmitted.

A WRITE ALPHA instruction, entered into the calculator by actuating the CALL key followed by the **2** key followed by the CALL key followed by desired character keys followed by the CALL key, is employed to select an I/O ALPHA mode similar to the ALPHA mode that may be selected when the calculator is operating alone. The ASCII equivalents of the characters specified in the WRITE ALPHA instruction are transmiteed to the peripheral I/O unit.

A READ X instruction, entered into the calculator by sequential actuation of the CALL, **2**, and B keys, is employed to input a number from the peripheral I/O unit to the X register. The number can appear in a free field format or with delimiters specified in a DELIM instruction.

A DELIM instruction entered into the calculator by sequential actuation of the CALL, **2**, and E keys, allows the user to specify any three ASCII characters as delimiters associated with data input to the calculator via the READ X instruction. The specified delimiters, labelled **1** through **3**, must be placed in the X, Y, and Z registers, respectively. When less than three delimiters are specified, the unused stack registers must be filled with zeros. Delimiter **1** performs the additional function of setting program flag **4**. Prior art calculators have had very limited data input capabilities because of the restriction of fixed delimiters. By allowing the user to specify delimiters, considerably more flexibility and control over numeric data input is obtained. Since recognition by the calculator of delimiter **1** sets program flag **4**, the user may input blocks of data of unknown length by simply separating the blocks with that delimiter. This avoids, for instance, delays in calculation while waiting for data that is not available.

A WBYTE instruction, entered into the calculator by sequential actuation of the CALL, **2**, and F keys, is employed to output the 8-bit binary equivalent of an integer number stored in the X register. The integer number must lie between 0 and 255.

An RBYTE instruction, entered into the calculator by sequential actuation of the CALL, **2**, and F keys, is employed to input one 8-bit binary character from a peripheral I/O unit and place its decimal equivalent in the X register.

An AND instruction, entered into the calculator by sequential actuation of the CALL, **2**, and H keys, is employed to combine the 8-bit binary equivalent of the numbers in the X and Y registers by performing a logical AND operation. The result is converted to decimal form and is placed in the X register.

An OR instruction, entered into the calculator by sequential actuation of the CALL, **2**, and I keys, is employed to combine the 8-bit binary equivalent of the numbers in the X and Y registers by performing a logical OR operation. The result is converted to decimal form and is placed in the X register.

A ROTATE instruction, entered into the calculator by sequential actuation of the CALL, **2**, and J keys, is employed to rotate the bits of the 8-bit binary equivalent of the number stored in the X register one place to the right. The result is placed in the X register in decimal form.

A DUMP PROGRAM instruction, entered into the calculator by sequential actuation of the CALL, **2**, and

M keys, outputs program instructions stored in the program portion of the user read-write memory, starting at the step location indicated by the number stored in the X register and ending when an END instruction is encountered.

A LOAD PROGRAM instruction, entered into the calculator by sequential actuation of the CALL, 2 and L keys, is employed to input program instructions from a peripheral I/O unit to the calculator memory, starting at the step location indicated by the number stored in the X register and continuing until an END instruction is encountered. When the LOAD PROGRAM instruction is executed under program control, the calculator automatically continues execution of that program at the next instruction following the LOAD PROGRAM instruction after the new program has been loaded from the peripheral I/O unit.

A LIST instruction, entered into the calculator by sequential actuation of the CALL, 2, and K keys, is employed to output a program listing starting at the step location indicated by the current location of the program counter and ending when an END instruction is encountered. The listing is formatted into four 50-step columns for use with a page-width line printer. This listing halts every 200 steps to allow the operator to insert more printer paper. The listing may then be continued by actuating the K key. The LIST instruction may only be executed from the keyboard and is not programmable.

A REMOTE mode of operation may be selected by entering ±2 into the X register followed by sequential actuation of the CALL, 2, and N keys. This mode is useful because it causes the calculator to wait at each I/O instruction for the peripheral I/O unit to begin the instructed operation. Normally, the calculator issues an I/O instruction and then waits for the peripheral I/O unit to signal completion of the operation. Completion of the operation is indicated by the peripheral I/O unit pulling the FLG line low. This means that the peripheral I/O unit is always waiting for instructions from the calculator. The REMOTE mode is useful in a system involving the use of the one calculator for gathering data and another calculator for performing calculations involving the data. The calculators may be connected via general I/O ROMs and the data gathering calculator would be set to the REMOTE mode each time it is ready to transfer data to the other calculator. This "two-processor" arrangement, with one calculator controlling the I/O operations between them, offers an extremely fast and flexible system for gathering and reducing data.

We claim:

1. An electronic calculator comprising:

memory means for storing instructions and data, said memory means including a program storage area for storing program instructions and a data storage area for storing data;

keyboard input means for entering information including data and instructions into the memory means;

processing means, coupled to said keyboard input means and memory means, for processing data and instructions entered into the memory means to perform selected functions;

output means, coupled to said processing means, for providing an output indication of selected functions performed by the calculator; and

logic means, coupled to said memory means and processing means, for defining a movable boundary between said program storage area and said data storage area of said memory means.

2. An electronic calculator as in claim 1 wherein said logic means is operative for initiating output of an error indication on said output means in response to an attempt by the user to enter program instructions into the data storage area of said memory means and in response to an attempt by the user to enter data into the program storage area of said memory means.

3. An electronic calculator as in claim 1 wherein said logic means includes a pointer word stored in said memory means and said logic means is operative for repositioning said pointer word to define the movable boundary between said program storage area and said data storage area in response to processing by said processing means of a selected instruction stored in said memory means.

4. An electronic calculator as in claim 3 wherein:

said data storage area of said memory means comprises one or more data storage registers; and

5. An electronic calculator comprising:

memory means for storing instructions and data, said memory means including a program storage area for storing program instructions and a data storage area for storing data;

keyboard input means for entering information including data and instructions into said memory means;

processing means, coupled to said keyboard input means and memory means, for processing data and instructions entered into said memory means to perform selected functions, said keyboard input means including control means for initiating processing by said processing means of a program of instructions stored in said program storage area of said memory means;

output means; coupled to said processing means, for providing an output indication of selected functions performed by the calculator; and

logic means, coupled to said memory means and processing means for defining a movable boundary between said program storage area and said data storage area of said memory means, said logic means including a pointer word stored in said memory means and being operative for repositioning said pointer word to define a movable boundary between said program storage area and said data storage area of said memory means when a selected instruction is encountered during processing by said processing means of a program of instructions stored in said program storage area of said memory means.

6. An electronic calculator as in claim 5 wherein said logic means includes means for initiating output of an error indication on said output means in response to an attempt by the user to enter program instructions into the data storage area of said memory means and in response to an attempt by the user to enter data into the program storage area of said memory means.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 4,089,059

DATED : May 9, 1978

INVENTOR(S) : Bradley W. Miller et al

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In the Drawings

The sheets of drawings consisting of Figure 67U, Figure 67V and Figure 67W should be added as per the attached sheets.

Figure 69K should follow sheet 124 rather than sheet 115;
Figure 71L should follow sheet 142 rather than sheet 154;
Figures 71M, 71N, 71Q, 71R, 71S, 71T, 71U, 71V and 71W should follow in sequence.
Figure 74I should follow sheet 154 rather than sheet 164;
Figure 74R should follow sheet 171 rather than sheet 173;
Figures 74S, 74T and 74U should follow in sequence.

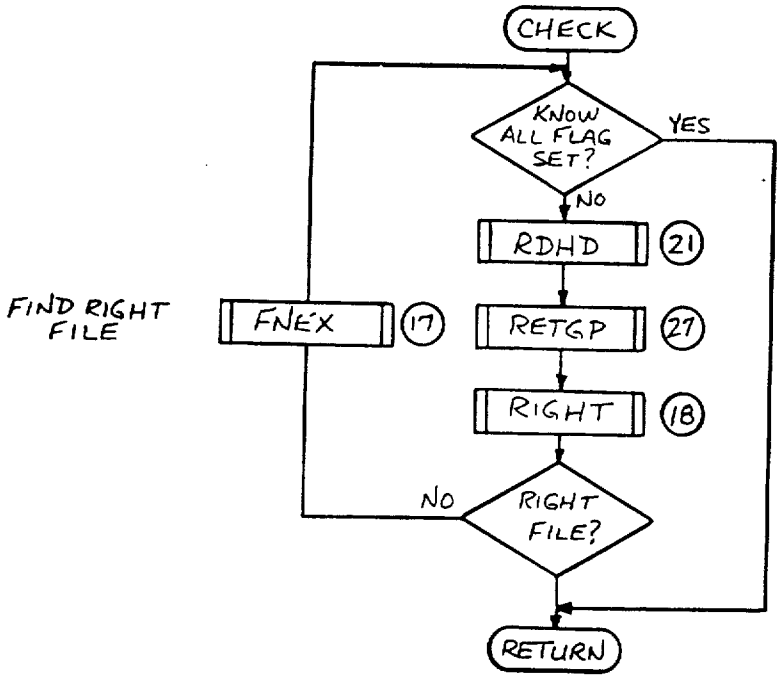


FIG. 67U

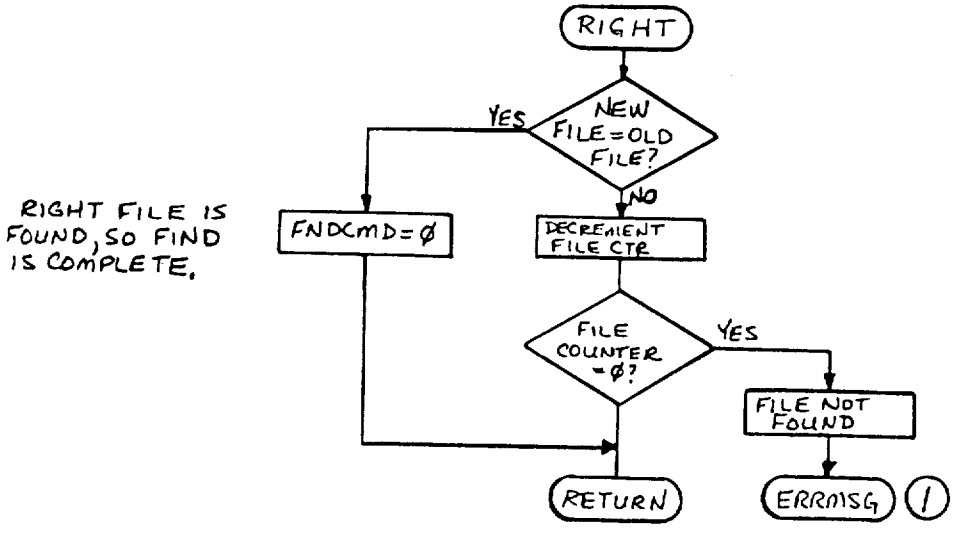


FIG. 67V

ZERO BIT SET ON EXIT MEANS CORRECT FILE WAS FOUND.

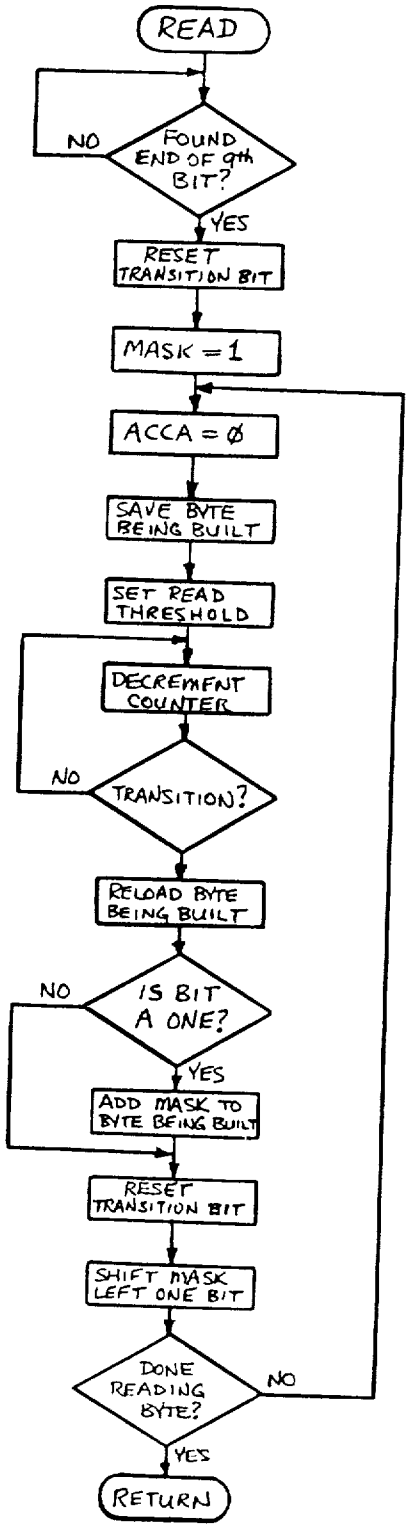

READ
FOUND END OF 9th BIT?
NO
YES
RESET TRANSITION BIT
MASK = 1
ACCA = Ø
SAVE BYTE BEING BUILT
SET READ THRESHOLD
DECREMENT COUNTER
TRANSITION?
NO
RELOAD BYTE BEING BUILT
IS BIT A ONE?
NO
YES
ADD MASK TO BYTE BEING BUILT
RESET TRANSITION BIT
SHIFT MASK LEFT ONE BIT
DONE READING BYTE?
NO
YES
RETURN

FIG. 67W

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 4,089,059

DATED : May 9, 1978

INVENTOR(S) : Bradley W. Miller 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 6, line 50, "i" should be --is--;

Column 6, line 64, "U/O" should be --I/O

Column 7, line 65, after "detailed" insert --schematic--;

Column 8, lines 9 & 10, "antimation" should be --antimotion--;

Column 8, line 56, after "schematic" insert --diagram--;

Column 11, in the Table, at line 13, after "RRAM" delete ".";

Column 13, line 22, "bgy" should be --by--;

Column 16, line 50, "Dll" should be --Dll--;

Column 17, line 25, "I06" should be --$\overline{\text{I06}}$--;

Column 17, line 26, "I07" should be --$\overline{\text{I07}}$--;

Column 17, line 52, "FIG. 53" should be --FIG. 52--;

Column 18, line 28, "<" should be -->--;

Column 18, line 31, ">" should be --<--;

Column 19, line 21, "is" should be --to--;

Column 21, line 7, "DOll" should be --$\overline{\text{DOll}}$--;

Column 21, line 31, after "to" insert --the--;

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 4,089,059

DATED : May 9, 1978

INVENTOR(S) : Bradley W. Miller 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 21, line 53, "Integraror" should be --Integrator--;

Column 21, line 67, "and" (second occurrence) should be --by--;

Column 21, line 67, "CD" should be --DC--;

Column 22, line 48, "-RAW" should be --+RAW--;

Column 23, line 34, "MPWO" should be --$\overline{\text{MPWO}}$--;

Column 331, line 45, "operationsl" should be --between--;

Column 332, line 28, "of" should be --on--;

Column 332, line 43, "of" should be --or--;

Column 333, line 12, after "desired" insert --data--;

Column 337, line 21, after "satisfactorily" delete "is";

Column 337, line 31, after "control" insert --or--;

Column 338, line 42, "an" should be --and--;

Column 342, line 62, after "number" insert --should--;

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 4,089,059

DATED : May 9, 1978

INVENTOR(S) : Bradley W. Miller 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 343, line 58, "identified" should be --defined--;

Column 344, line 33, "*LOG OF #--ø" should be --*LOG OF # $\leq$ ø--;

Column 344, line 68, delete "—$\leq$0" and insert --The--;

Column 347, line 63, after "CALL," insert --2,--;

Column 349, line 45, "the" (first occurrence) should be --that--.

Signed and Sealed this

Twelfth Day of *June 1984*

[SEAL]

Attest:

Attesting Officer

GERALD J. MOSSINGHOFF

Commissioner of Patents and Trademarks