

NCV-11

EXERCISER
CZNCDA0

AH-E777A-MC

COPYRIGHT © 73-78

FICHE 1 OF 1

DEC 1978

digital

MADE IN USA

Product Code: AC-E776A-MC
Product Name: CZNCDAO NCV11 EXERCISER
Date: AUGUST 1978
MAintainer: Diagnostic Engineering

Copyright (C) 1973,1978
Digital Equipment Corporation, MAynard, Mass.

This software is furnished under a license for use only on a single computer system and may be copied only with the inclusion of the above copyright notice. This software, or any other copies thereof, may not be provided or otherwise made available to any other person except for use on such system and to one who agrees to these license terms. Title to and ownership of the software shall at all times remain in DEC.

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation.

DEC assumes no responsibility for the use or reliability of its software on equipment which is not supplied by DEC.

C.0 TABLE OF CONTENTS

- 1.0 ABSTRACT
- 2.0 REQUIREMENTS
 - 2.1 Equipment
 - 2.2 Equipment - Optional
 - 2.3 PRELIMINARY PROGRAMS
 - 2.4 Storage
- 3.0 LOADING PROCEDURE
- 4.0 STARTING PROCEDURE
- 5.0 OPERATOR OPTIONS
 - 5.1 OPERATOR KEYBOARD OPTIONS
 - 5.2 OPERATOR SWITCH REGISTER OPTION(S)
- 6.0 RECOMMENDED OPERATOR ACTION
 - 6.1 Camera Setup
 - 6.2 Joystick Setup
- 7.0 MISCELLANEOUS
 - 7.1 Device Bus Address Modifications
 - 7.2 Free Run Mode (Type 'F')
 - 7.3 Error Reporting
 - 7.4 Execution Time
- 8.0 PROGRAM DESCRIPTION
 - 8.1 Data Connection & Display
 - 8.2 Joystick Mode
- 9.0 LISTING

1.0 ABSTRACT

This is a self contained program designed to convert data from the GAMMA camera via the NCV11 interface and display it on a VSV01 display. It can be used for camera/interface and joystick setup or for practice in image manipulation. It is meant as a exerciser rather than a diagnostic. TOTAL PROGRAM CONTROL IS ACCOMPLISHED THRU THE CONSOLE KEYBOARD. THE PROGRAM IS NOT CHAINABLE/SCRIPTABLE UNDER XXDP/APT.

2.0 REQUIREMENTS

2.1 Equipment

1. PDP-11 FAMILY OR LSI-11 FAMILY Computer
2. I/O Terminal (i.e., LA36)
3. NCV11 Interface
4. Gamma Camera (OR SUBSTITUTE)

2.2 Equipment - Optional

1. H3060 Joystick
2. VSV01 DISPLAY

2.3 PRELIMINARY PROGRAMS

MAINDEC-11-DVNCA SHOULD HAVE PREVIOUSLY BEEN RUN

2.4 Storage

This program uses all of lower 12K of memory.

3.0 LOADING PROCEDURE

Normal procedure for loading a binary program into memory SHOULD BE followed.

4.0 STARTING PROCEDURE

Loading address 200 and starting will initialize the system,
IDENTIFY THE PROGRAM and wait a keyboARd command.

5.0 OPERATOR OPTIONS

5.1 OPERATOR KEYBOARD OPTIONS

TYPE 'H' - HELP OPERATOR AND DISPLAY OR TYPE THIS LIST
Type 'N' - Collect new data from camera - clears core matrix FIRST
Type 'C' - Collect data from camera - starts NCV11, previous core
matrix data not cleared
TYPE 'X' - SELECT ANOTHER CAMERA CHANNEL
Type 'S' - Stop NCV11 data collection - also terminates free RUN
MODE
Type 'Z' - Zoom - Set NCV11 TO GAIN 2
Type 'R' - Regular - SET NCV11 TO GAIN 1 (DEFAULT COND)
Type 'W' - Select VSV01 display using 1st bit map
Type 'M' - Select VSV01 display USING 2nd bit map
Type 'A' - Display ISOTOPE A (default cond)
Type 'B' - Display ISOTOPE B (B GAMMA)
Type 'D' - Display data - display selected ISOTOPE
Type 'L' - Get lower threshold from keyboard (default=0) - all
matrix values less than typed value are not displayed
Type 'U' - Get upper threshold from KEYBOARD (default=177777) -
All matrix values greater than typed value are NOT
DISPLAYED
Type 'F' - Free run mode - collect and display new data
continuously - Type 'S' to terminate this mode
TYPE 'X' - SELECT ANOTHER CAMERA CHANNEL
Type 'G' - Initialize everything - start fresh
Type 'J' - Joystick calibration NCV11 - use 'Cntrl C' to terminate mode
TYPE 'T' - DISPLAY INTENSITY TEST PATTERN ON SELECTED DISPLAY -
THIS FEATURE IS FOR DISPLAY VERIFCATION ONLY
TYPE 'O' - OTHER TERMINAL TO CONTROL THE PROGRAM

Type 'Cntrl & C' - Abort whatever - back to keyboard monitor

5.2 OPERATOR SWITCH REGISTER OPTION(S)

NONE

6.0 RECOMMENDED OPERATOR ACTION

6.1 Camera/A017 Setup

1. Place radioactive flood source IN FRONT OF CAMERA DETECTOR, OR USE A WEAK RADIOACTIVE SAMPLE ABOUT 3 FEET FROM A DETECTOR WITH THE COLLIMATOR REMOVED.
2. Collect data (C) and display (D)
3. Adjust the X and Y gain controls ON THE A017 until a round circle* IS SEEN and centered within the box on screen.
4. Increase the circle* size until the edges just begin to TOUCH THE BOX.
5. Collect data for a 3 minute period and observe the image. If there is an artifact on the center of the screen that looks like a right angle bracket, the 'Z' delay circuit should be adjusted.
6. IF THE MATRIX AND Z COUNTS EQUAL 0 THEN NO DATA WAS COLLECTED, THEREFORE, NO DISPLAY.

* THIS MAY BE A HEXAGON FOR SOME MAKES OF CAMERA.

6.2 Joystick Setup

1. Select Joystick configuration by typing 'J'.
2. Adjust the Y Pot on the H3060 Joystick so that the HORIZONTAL cross hair provide UNIFORM access within the box drawn on the screen. ADJUST THE X POT ON THE H3060 SO THAT THE VERTICAL CROSS HAIR WILL ALSO BE UNIFORM EXCEPT THAT DUE TO A HARDWARE OFFSET WILL NOT REACH THE RIGHT EDGE AND EXCEED THE LEFT EDGE BY THE SAME AMOUNT. Ideally, a physical center of the Joystick will provide a point approximately centered in the box.
3. Check that when the Joystick interrupt SWITCH is depressed the CROSS HAIRS DO not follow the Joystick.
4. A 'Cntrl C' will return user back to the keyboard monitor.

7.0 MISCELLANEOUS

7.1 Device Bus Address Modifications

NCV11 Modify location 'NCVADR' if base bus address is not 772760

VSV01 Modify location 'VTVADR' if base bus address (CHARACTER GENERATOR) IS NOT 772600
MODIFY LOCATION 'VTMADR' IF BASE BUS ADDRESS (BIT MAP) IS NOT 772620

Note: A restart is required after any of the above address modifications.

7.2 Free Run Mode (Type 'F')

Modification of location 'TIME' will alter the data collection time in the free run mode. The default value of 20(8) provides about 10 SECONDS ON AN LSI-11 CPU.

7.3 Error Reporting

1. INCORRECT KEYBOARD COMMANDS ARE RESPONDED WITH '?'
2. A MESSAGE WILL BE REPORTED IF ANY SELECTED DEVICE (SEE SECTION 7.1) DOES NOT RESPOND.

7.4 Execution Time

The EXECUTION TIME is completely dependent upon the user for whatever option selected.

8.0 PROGRAM DESCRIPTION

8.1 Data Collection & Display

The user may collect data from the gamma camera system via the NCV11 interface and display this data on the VSV01 (bitmap) display. All functions are ENTERED thru the keyboard as defined in section 5.1. When the data collection mode is selected the NCV11 is receiving X & Y data from the gamma camera (looking AT a radioactive source). This information is transformed via THE address maker LOGIC which forms a unique address within a defined matrix relative to the scan position of the camera. The NCV11 preforms an NPR increment to memory to this UNIQUE address. The program selects the address MATRIX 64x64x16 (RESOLUTION 2) OFFSET TO address 20000(8) FOR THE 'A' Isotope, AND FOR THE 'B' ISOTOPE, (CAMERAS EQUIPPED WITH THE DUAL ISOTOPE ATTACHMENT) INFORMATION VIA B GAMMA LOGIC IS STORED STARTING AT ADDRESS 40000 (8). The intensity of each cell (memory location) of the image in memory is adjusted with the upper and lower thresholds and scaled to ONE OF 16 intensity levels. The cell with the highest count represents THE HIGHEST intensity level, therefore appears brightest on the display. At the completion of the image display, the following parameters are displayed:

Lower Threshold -	All values greater than this value are displayed
Upper Threshold -	All values above this value are omitted
CAMERA -	SELECTED CAMERA CHANNEL
PB -	UP/DN INDICATOR OF THE PUSH BUTTON STATE DURING LAST COLLECTION
JB -	UP/DN INDICATOR OF THE JOY BUTTON STATE DURING LAST COLLECTION
Z Count -	32 bit counter of EVENT pulses
MatRix Count -	Total count of the contents of each cell in the 64x64 matrix
Zoom -	Displayed if gain was selected
B-Gamma -	Displayed if 'B' Isotope selected

8.2 Joystick Mode

Selecting the TEST takes advantage of the Joystick mode provided by the NCV11. This mode selects the JOYSTICK inputs and the NCV11 performs like an A/D with the X & Y CONVERTED values available in X-Y holding register. The X and Y data is applied to the display and should conform to the requirements in section 6.2. THE VSV01 uses the X & Y cross hairs as the DISPLAY indicator.

9.0 LISTING

CZNCDA NCV11 EXERCISER MACY11 27(654) 8-AUG-78 15:16
CZNCDA.P11

SEQ 0008

17	BASIC DEFINITIONS
133	TRAP CATCHER
142	STARTING ADDRESS(ES)
146	COMMON TAGS
187	ERROR POINTER TABLE
274	MAIN PROGRAM
276	INITIALIZE THE COMMON TAGS
380	KEYBOARD DISPATCH TABLE
408	COMMAND DECODER
556	PROGRAM ROUTINES
818	PROGRAM SUBROUTINES
1040	TTY INPUT ROUTINE
1209	TYPE ROUTINE
1280	BINARY TO OCTAL (ASCII) AND TYPE
1358	READ AN OCTAL NUMBER FROM THE TTY
1412	TRAP DECODER
1435	TRAP TABLE
1455	POWER DOWN AND UP ROUTINES
1497	DISPLAY MESSAGES
1526	ASCII MESSAGES

```
1      .TITLE CZNCDA NCV11 EXERCISER
2      .;COPYRIGHT (C) 1978
3      .;DIGITAL EQUIPMENT CORP.
4      .;MAYNARD, MASS. 01754
5      .;
6      .;PROGRAM BY R.SHOOP
7      .;
8      .;THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
9      .;PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
10     .;
11     000001 $TN=1
12     160000 $SWR=160000      .;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT
13     172760      ABASE=172760      .;DEFAULT NCV11 ADDRESS
14     .SBTTL BASIC DEFINITIONS
15
16     .;INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
17     001100 STACK= 1100
18     .EQUIV EMT,ERROR      .;BASIC DEFINITION OF ERROR CALL
19     .EQUIV IOT,SCOPE      .;BASIC DEFINITION OF SCOPE CALL
20
21     .;MISCELLANEOUS DEFINITIONS
22     000011 HT= 11      .;CODE FOR HORIZONTAL TAB
23     000012 LF= 12      .;CODE FOR LINE FEED
24     000015 CR= 15      .;CODE FOR CARRIAGE RETURN
25     000200 CRLF= 200    .;CODE FOR CARRIAGE RETURN-LINE FEED
26     177776 PS= 177776  .;PROCESSOR STATUS WORD
27     .EQUIV PS,PSW
28     177774 STKLMT= 177774 .;STACK LIMIT REGISTER
29     177772 PIRQ= 177772 .;PROGRAM INTERRUPT REQUEST REGISTER
30     177570 DSWR= 177570 .;HARDWARE SWITCH REGISTER
31     177570 DDISP= 177570 .;HARDWARE DISPLAY REGISTER
32
33     .;GENERAL PURPOSE REGISTER DEFINITIONS
34     000000 R0= %0      .;GENERAL REGISTER
35     000001 R1= %1      .;GENERAL REGISTER
36     000002 R2= %2      .;GENERAL REGISTER
37     000003 R3= %3      .;GENERAL REGISTER
38     000004 R4= %4      .;GENERAL REGISTER
39     000005 R5= %5      .;GENERAL REGISTER
40     000006 R6= %6      .;GENERAL REGISTER
41     000007 R7= %7      .;GENERAL REGISTER
42     000006 SP= %6      .;STACK POINTER
43     000007 PC= %7      .;PROGRAM COUNTER
44
45     .;PRIORITY LEVEL DEFINITIONS
46     000000 PR0= 0      .;PRIORITY LEVEL 0
47     000040 PR1= 40     .;PRIORITY LEVEL 1
48     000100 PR2= 100    .;PRIORITY LEVEL 2
49     000140 PR3= 140    .;PRIORITY LEVEL 3
50     000200 PR4= 200    .;PRIORITY LEVEL 4
51     000240 PR5= 240    .;PRIORITY LEVEL 5
52     000300 PR6= 300    .;PRIORITY LEVEL 6
53     000340 PR7= 340    .;PRIORITY LEVEL 7
54
```

```

55      ;*'SWITCH REGISTER' SWITCH DEFINITIONS
56      100000 SW15= 100000
57      040000 SW14= 40000
58      020000 SW13= 20000
59      010000 SW12= 10000
60      004000 SW11= 4000
61      002000 SW10= 2000
62      001000 SW09= 1000
63      000400 SW08= 400
64      000200 SW07= 200
65      000100 SW06= 100
66      000040 SW05= 40
67      000020 SW04= 20
68      000010 SW03= 10
69      000004 SW02= 4
70      000002 SW01= 2
71      000001 SW00= 1
72      .EQUIV SW09,SW9
73      .EQUIV SW08,SW8
74      .EQUIV SW07,SW7
75      .EQUIV SW06,SW6
76      .EQUIV SW05,SW5
77      .EQUIV SW04,SW4
78      .EQUIV SW03,SW3
79      .EQUIV SW02,SW2
80      .EQUIV SW01,SW1
81      .EQUIV SW00,SW0
82
83      ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
84      100000 BIT15= 100000
85      040000 BIT14= 40000
86      020000 BIT13= 20000
87      010000 BIT12= 10000
88      004000 BIT11= 4000
89      002000 BIT10= 2000
90      001000 BIT09= 1000
91      000400 BIT08= 400
92      000200 BIT07= 200
93      000100 BIT06= 100
94      000040 BIT05= 40
95      000020 BIT04= 20
96      000010 BIT03= 10
97      000004 BIT02= 4
98      000002 BIT01= 2
99      000001 BIT00= 1
100     .EQUIV BIT09,BIT9
101     .EQUIV BIT08,BIT8
102     .EQUIV BIT07,BIT7
103     .EQUIV BIT06,BIT6
104     .EQUIV BIT05,BIT5
105     .EQUIV BIT04,BIT4
106     .EQUIV BIT03,BIT3
107     .EQUIV BIT02,BIT2
108     .EQUIV BIT01,BIT1
  
```

```

109      .EQUIV BIT00,BIT0
110
111      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
112      000004      ERRVEC= 4      ;;TIME OUT AND OTHER ERRORS
113      000010      RESVEC= 10     ;;RESERVED AND ILLEGAL INSTRUCTIONS
114      000014      TBITVEC=14     ;;'T' BIT
115      000014      TRTVEC= 14     ;;TRACE TRAP
116      000014      BPTVEC= 14     ;;BREAKPOINT TRAP (BPT)
117      000020      IOTVEC= 20     ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
118      000024      PWRVEC= 24     ;;POWER FAIL
119      000030      EMTVEC= 30     ;;EMULATOR TRAP (EMT) **ERROR**
120      000034      TRAPVEC=34     ;;'TRAP' TRAP
121      000060      TKVEC= 60      ;;TTY KEYBOARD VECTOR
122      000064      TPVEC= 64      ;;TTY PRINTER VECTOR
123      000240      PIRQVEC=240    ;;PROGRAM INTERRUPT REQUEST VECTOR
124
125
126      ;SOME COMMON PROGRAM VALUES AND EQUATES
127
128      020000      MATRIX=20000    ;STARTING ADRS OF MATRIX DATA
129
130      .SBTTL TRAP CATCHER
131
132      000000      .=0
133      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A '".+2,HALT"'
134      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
135      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
136      000174      .=174
137      000174      000000      DISPREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
138      000176      000000      SWREG:   .WORD 0      ;;SOFTWARE SWITCH REGISTER
139      .SBTTL STARTING ADDRESS(ES)
140      000200      000137      001322      JMP @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
141      000100      000100      .-100
142      000100      000104      000340      000002      104,340,RTI ;LSI-11 B EVENT

```

```

143      .SBTTL COMMON TAGS
144
145      ;*****
146      ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
147      ;*USED IN THE PROGRAM.
148
149      .=1100
150      $CMTAG:      ;:START OF COMMON TAGS
151      $PASS: .WORD 0      ;:CONTAINS PASS COUNT
152      $TSTNM: .BYTE 0      ;:CONTAINS THE TEST NUMBER
153      $ERFLG: .BYTE 0      ;:CONTAINS ERROR FLAG
154      $ICNT: .WORD 0      ;:CONTAINS SUBTEST ITERATION COUNT
155      $LPADR: .WORD 0      ;:CONTAINS SCOPE LOOP ADDRESS
156      $LPERR: .WORD 0      ;:CONTAINS SCOPE RETURN FOR ERRORS
157      $ERTTL: .WORD 0      ;:CONTAINS TOTAL ERRORS DETECTED
158      $ITEMB: .BYTE 0      ;:CONTAINS ITEM CONTROL BYTE
159      $ERMAX: .BYTE 1      ;:CONTAINS MAX. ERRORS PER TEST
160      $ERRPC: .WORD 0      ;:CONTAINS PC OF LAST ERROR INSTRUCTION
161      $GDADR: .WORD 0      ;:CONTAINS ADDRESS OF 'GOOD' DATA
162      $BDADR: .WORD 0      ;:CONTAINS ADDRESS OF 'BAD' DATA
163      $GDDAT: .WORD 0      ;:CONTAINS 'GOOD' DATA
164      $BDDAT: .WORD 0      ;:CONTAINS 'BAD' DATA
165      .WORD 0      ;:RESERVED--NOT TO BE USED
166      .WORD 0
167      $AUTOB: .BYTE 0      ;:AUTOMATIC MODE INDICATOR
168      $INTAG: .BYTE 0      ;:INTERRUPT MODE INDICATOR
169      .WORD 0
170      $SWR: .WORD DSWR      ;:ADDRESS OF SWITCH REGISTER
171      $DISPLAY: .WORD DDISP      ;:ADDRESS OF DISPLAY REGISTER
172      $TKS: 177560      ;:TTY KBD STATUS
173      $TKB: 177562      ;:TTY KBD BUFFER
174      $TPS: 177564      ;:TTY PRINTER STATUS REG. ADDRESS
175      $TPB: 177566      ;:TTY PRINTER BUFFER REG. ADDRESS
176      $NULL: .BYTE 0      ;:CONTAINS NULL CHARACTER FOR FILLS
177      $FILLS: .BYTE 2      ;:CONTAINS # OF FILLER CHARACTERS REQUIRED
178      $FILLC: .BYTE 12      ;:INSERT FILL CHARS. AFTER A 'LINE FEED'
179      $TPFLG: .BYTE 0      ;: 'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
180      $QUES: .ASCII /?/      ;:QUESTION MARK
181      $CRLF: .ASCII <15>      ;:CARRIAGE RETURN
182      $LF: .ASCII <12>      ;:LINE FEED
183      ;*****

```

```
184 .SBTTL ERROR POINTER TABLE
185
186 ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
187 ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
188 ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
189 ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
190 ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
191
192 ;* EM ;;POINTS TO THE ERROR MESSAGE
193 ;* DH ;;POINTS TO THE DATA HEADER
194 ;* DT ;;POINTS TO THE DATA
195 ;* DF ;;POINTS TO THE DATA FORMAT
196
197
198 001164 $ERRTB:
199 ;THIS PROGRAM DOES NOT USE THE ABOVE ERROR TABLE
200
201
202
203
204
205
206
207 ;NCV11 BUS ADRS ASSIGNMENTS - MODS ARE TO BE MADE HERE
208
209 001164 172760 NCADR: ABASE
210
211 ;VSV01 BUS ADRS ASSIGNMENTS (CHAR GEN) - MODS ARE TO BE MADE HERE
212
213 001166 172600 VTVADR: 172600
214
215 ;VSV01 BUS ADRS ASSIGNMENTS (BIT MAP) - MODS ARE TO BE MADE HERE
216
217 001170 172620 VTMADR: 172620 ;FIRST MAP 2ND MAP IS 20 HIGHER
218
219 ;COUNTER FOR FREE RUN MODE
220
221 001172 000024 TIME: 20. ;LOOP COUNTER FOR 'F' MODE
222
```

223				:NCV11 REGISTER ADDRESS POINTERS	
224	001174	172760	NCCSR:	ABASE	:NCV11 STATUS CONTROL REGISTER
225	001176	172762	NCOFF:	ABASE+2	:NCV11 OFFSET REGISTER
226	001200	172764	NCWCR:	ABASE+4	:NCV11 WORD COUNT REGISTER
227	001202	172766	NCBAR:	ABASE+6	:NCV11 BUS ADDRESS REGISTER / LOW Z COUNT
228	001204	172770	NCSFR:	ABASE+10	:NCV11 SPECIAL FUNCTION / JOYSTICK STATUS REGISTER
229	001206	172772	NCADM:	ABASE+12	:NCV11 ADDRESS MAKER REGISTER
230	001210	172774	NCJOY:	ABASE+14	:NCV11 JOYSTICK DATA REGISTER
231	001212	172776	NCBAR1:	ABASE+16	:NCV11 SPARE <SAME AS NCBAR>
232				:VS 01 REGISTER ADDRESS POINTERS (CHARACTER GENERATOR)	
233	001214	172600	VTVCRG:	172600	:CHAR/STATUS
234	001216	172602	VTVCHP:	172602	:CROSS HAIR POS
235	001220	172604	VTVPOS:	172604	:CHAR POS
236				:VSV01 REGISTER ADDRESS POINTERS (BIT MAP)	
237	001222	172620	VTVCSR:	172620	:COMMAND/STATUS
238	001224	172622	VTVMAP:	172622	:MAP ADRS
239	001226	172624	VTVPX:	172624	:PIXEL WD
240	001230	172626	VTVPX1:	172626	:PIXEL BYTE
241	001232	172630	VTVINT:	172630	:INTENSITY LOOK UP
242					
243				:COMMON PROGRAM TAGS AND STORAGE LOCATIONS	
244	001234	000000	TEMPO:	0	:COMMON UTILITY LOC
245	001236	000000	TEMP1:	0	:COMMON UTILITY LOC
246	001240	000000	TEMP2:	0	:COMMON UTILITY LOC
247	001242	000000	DUMMY1:	0	:OCTAL TEMP LOC.
248	001244	000000	DUMMY2:	0	:OCTAL TEMP LOC.
249	001246	000000	INTLUT:	0	
250	001250	006400	VTVSAV:	6400	:VSV01 SET UP - FULL SCREEN, MONO(BLK/WHT), ENA DISPLAY
251	001252	000000	MRXADR:	0	:CURRENT CORE ADRS OF CELL BEING DISPLAYED
252	001254	000000	MAPADR:	0	:CURRENT ADRS OF BIT MAP LD
253	001256	000003	PIXCNT:	3	:CURRENT PIXEL BYTE COUNT
254	001260	000000	PIXASM:	0	:4 PIXELS ASSEMBLED HERE BEFORE BIT MAP LD
255	001262	000000	KBUFF:	0	:CONTAINS KEYBOARD CHAR TYPED
256	001264	000000	TTYOUT:	0	:OUTPUT CHAR TO PRINTER
257	001266	000000	THLO:	0	:LOW THRESHOLD VALUE APPLIED TO MATRIX CELLS
258	001270	177777	THHI:	177777	:HIGH THRESHOLD VALUE APPLIED TO MATRIX CELLS
259	001272	000000	COMSAV:	0	:SAVED NCV11 CSR CONTENTS WHEN DISPLAYING
260	001274	000000	CAMERA:	0	:CAMERA SELECTION IN BITS 8-9
261	001276	000000	GAIN:	0	:GAIN 1=0, GAIN 2=BIT10
262	001300	010000	TOTSIZ:	4096.	:TOTAL # OF CELLS IN MATRIX (ISOTOPE A OR B)
263	001302	000000	CHARCT:	0	:COUNTS TOTAL # OF CELLS IN MATRIX (ISOTOPE A OR B)
264	001304	000000	CPERL:	0	:CONTAINS LARGEST CELL OF MATRIX WITHIN THRESHOLDS
265	001306	000000	ROWCNT:	0	:COUNTS 64 CELLS EACH ROW OF CORE MATRIX
266	001310	020000	TABLEX:	MATRIX	:CONTAINS STARTING ADRS OF MATRIX OF SELECTED ISOTOPE
267	001312	000000	FREERN:	0	:NON-ZERO SAYS COLLECT NEW DATA & DISPLAY IN FREE RUN MO
268	001314	000000	MSELCT:	0	:0 SAYS 1ST BIT MAP, -1 SAYS 2ND BIT MAP(VSV01)
269	001316	000000	BELLEN:	0	:0 SAYS RING BELL ON NO CONVERT, -1 SAYS DON'T
270	001320	000000	NOVSV:	0	:NON-ZERO INDICATES NO VSV01 DISPLAY

CZNCDA.P11 MAIN PROGRAM

```
271 001322 START:
272 .SBTTL INITIALIZE THE COMMON TAGS
273 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
274 001322 012706 001100 MOV #CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
275 001326 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
276 001330 022706 001140 CMP #SWR,R6 ;;DONE?
277 001334 001374 BNE -6 ;;LOOP BACK IF NO
278 001336 012706 001100 MOV #STACK,SP ;;SETUP THE STACK POINTER
279 ;;INITIALIZE A FEW VECTORS
280 001342 012737 007316 000034 MOV #STRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
281 001350 012737 000340 000036 MOV #340,@#TRAPVEC+2;LEVEL 7
282 001356 012737 007376 000024 MOV #SPWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
283 001364 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;LEVEL 7
284 ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
285 ;;EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
286 001372 013746 000004 MOV @#ERRVEC,-(SP) ;;SAVE ERROR VECTOR
287 001376 012737 001432 000004 MOV #64$,@#ERRVEC ;;SET UP ERROR VECTOR
288 001404 012737 177570 001140 MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
289 001412 012737 177570 001142 MOV #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
290 001420 022777 177777 177512 CMP #-1,@SWR ;;TRY TO REFERENCE HARDWARE SWR
291 001426 001012 BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
292 ;;AND THE HARDWARE SWR IS NOT = -1
293 001430 000403 BR 65$ ;;BRANCH IF NO TIMEOUT
294 001432 012716 001440 64$: MOV #65$,(SP) ;;SET UP FOR TRAP RETURN
295 001436 000002 RTI
296 001440 012737 000176 001140 65$: MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR
297 001446 012737 000174 001142 MOV #DISPREG,DISPLAY
298 001454 012637 000004 66$: MOV (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
299
300 001460 104401 010003 TYPE ,MSG1 ;;GO IDENTIFY PROGRAM
301 001464 004737 004440 JSR PC,SELCTA ;;DEFAULT TO VSV01 DISPLAY IF THERE
302 001470 103005 BCC 3$ ;;BR IF DISPLAY VSV01 IS THERE
303 001472 005237 001320 INC NOVSV ;;SAVE THE FACT NO VSV01 CONNECTED
304 001476 104401 010121 TYPE ,MSG3 ;;GO TYPE 'BUS TIMEOUT ER - VSV01 DISPLAY'
305 001502 000000 HALT ;;NO DISPLAY SEEN AT ASSIGNED BUS ADRS'S
306 001504 013700 001164 3$: MOV NCADR,R0 ;;GET NCV11 BASE ADRS
307 001510 012701 001174 MOV #NCCSR,R1 ;;GET POINTER ADRS
308 001514 010021 NCSET: MOV R0,(R1)+ ;;SET UP NCV11 REG ADRS PTRS
309 001516 062700 000002 ADD #2,R0 ;;BUMP REG ADRS
310 001522 022701 001214 CMP #VTVCRG,R1 ;;ALL SET UP?
311 001526 001372 BNE NCSET ;;BR IF NOT
312 001530 012737 001544 000004 MOV #1$,ERRVEC ;;SET UP TIMEOUT ADRS
313 001536 005777 177432 RST @NCCSR ;;WILL TRAP TO LOC 4 IF NOT THERE
314 001542 000404 BR START1 ;;BR IF THERE
315 001544 022626 1$: CMP (SP)+,(SP)+ ;;FIX STACK SINCE NO RTI
316 001546 104401 010241 TYPE ,MSG5 ;;GO TYPE 'BUS TIMEOUT ER - NCV11'
317 001552 000000 HALT ;;NCV11 NOT SEEN AT ASSIGNED BUS ADRS
318 001554 012737 000006 000004 START1: MOV #ERRVEC+2,@#ERRVEC ;;RESTORE ERR TRAP LOC TO PT TO 6
319 001562 000005 RESET ;;CLR WORLD
320 001564 004737 004660 JSR PC,NCSTP1 ;;GO STOP NCV11 & CLR CORE MATRIX
321 001570 012777 020000 177400 MOV #MATRIX,@NCOFF ;;SET OFFSET(ADRS 20000)
322 001576 005037 001276 CLR GAIN ;;REGULAR GAIN
323 001602 012737 020000 001310 MOV #MATRIX,TABLEX ;;ISOTOPE A
324 001610 005037 001266 CLR THLO ;;CLEAR LOWER THRESHOLD
```

```

325 001614 012737 177777 001270      MOV      #177777,THHI      ;CEILING THRESHOLD
326 001622 012737 005104 000060      MOV      #KBINT,TKVEC    ;SET KEYBOARD INTR RETURN ADRS
327 001630 012737 000340 000062      MOV      #340,TKVEC+2    ;SET UP NEW PRIORITY ON INTR
328 001636 012777 000100 177300      MOV      #100,@$TKS      ;SET INTR ENABLE
329 001644 104401 010275              TYPE      ,MSG6          ;GO ASK FOR KEYBOARD COMMAND(S)
330
331      ;THIS IS A LIST OF KEYBOARD COMMANDS FOR THE NCV11/DISPLAY/JOYSTICK
332
333      ;H      HELP FRAME
334      ;D      DISPLAY DATA
335      ;E      ERASE THE SCOPE
336      ;L      GET LOWER THRESHOLD FROM KEYBOARD & DISPLAY DATA
337      ;U      GET UPPER THRESHOLD FROM KEYBOARD & DISPLAY DATA
338      ;N      COLLECT NEW DATA FROM CAMERA (CLEAR CORE MATRIX)
339      ;C      COLLECT DATA FROM CAMERA
340      ;X      CHANGE CAMERA
341      ;S      STOP COLLECTION
342      ;G      INITIALIZE EVERYTHING
343      ;Z      ZOOM - SET GAIN TO 2
344      ;R      REGULAR - SET GAIN TO 1
345      ;A      DISPLAY ISOTOPE A
346      ;B      DISPLAY ISOTOPE B
347      ;W      SELECT VSV01 DISPLAY USING 1ST BIT MAP
348      ;M      SELECT VSV01 DISPLAY USING 2ND BIT MAP
349      ;F      FREE RUN MODE - COLLECT & DISPLAY NEW DATA CONTINUALLY
350      ;J      JOYSTICK CALIBRATION
351      ;T      DISPLAY INTENSITY TEST IMAGE ON SELECTED DISPLAY
352      ;O      OTHER TERMINAL TO CONTROL PROGRAM
353      ;CNTRL C  ABORT WHATEVER - BACK TO KEYBOARD MONITOR
354
355      ;THIS CODE WILL DISPATCH PROGRAM TO THE PROPER
356      ;ROUTINE VIA THE DISPATCH TABLE 'RTABLE'
357 001650 005037 001312      LISEN:  CLR      FREERN      ;KNOCK DOWN FREE RUN MODE IF SET
358 001654 012706 001100      MOV      #STACK,SP      ;RESET STACK PTR
359 001660 005037 001262      CLR      KBUFF          ;INSURE NO KEYBOARD GARBAGE
360 001664 005046              CLR      -(SP)          ;PUSH LEVEL 0 ONTO STACK
361 001666 012746 001674      MOV      #LISN,-(SP)    ;TO BE LSI-11 COMPATABLE
362 001672 000002              RTI              ;FAKE RTI TO LOWER PRIORITY
363 001674 013700 001262      LISN:  MOV      KBUFF,R0      ;LOOK FOR CHAR
364 001700 001775              BEQ      LISN          ;WAIT FOR ONE
365 001702 005037 001262      CLR      KBUFF          ;
366 001706 020027 000101      CMP      R0,#101        ;WAS IT A CHAR?
367 001712 103445              BCS      BOOB00        ;NOT A GOOD CHAR
368 001714 042700 177740      BIC      #177740,R0      ;ELIMINATE LOWER CASE
369 001720 020027 000033      CMP      R0,#33         ;IS IT AN ALPHA CHAR?
370 001724 103040              BCC      BOOB00        ;BR IF NOT
371 001726 006300              ASL      R0            ;MAKE UP WORD OFFSET
372 001730 005760 001740      TST      RTABLE-2(R0)    ;IS IT A LEGAL COMMAND?
373 001734 001434              BEQ      BOOB00        ;BR IF NOT
374 001736 000170 001740      JMP      @RTABLE-2(R0)  ;GO DO IT

```

```
375
376
377
378 001742 002046
379 001744 002060
380 001746 002072
381 001750 002102
382 001752 002106
383 001754 002116
384 001756 002212
385 001760 002222
386 001762 000000
387 001764 002274
388 001766 000000
389 001770 002300
390 001772 002340
391 001774 002372
392 001776 002406
393 002000 000000
394 002002 000000
395 002004 002526
396 002006 002544
397 002010 002562
398 002012 002576
399 002014 000000
400 002016 002636
401 002020 002666
402 002022 000000
403 002024 003006
```

.SBTTL KEYBOARD DISPATCH TABLE

RTABLE: ROUTA	: 'A' DISPLAY ISOTOPE A
ROUTB	: 'B' DISPLAY ISOTOPE B
ROUTC	: 'C' COLLECT DATA
ROUTD	: 'D' DISPLAY DATA
ROUTE	: 'E' ERASE SCOPE
ROUTF	: 'F' FREE RUN MODE(COLLECT NEW DATA & DISPLAY CONTINOUSL
ROUTG	: 'G' INITIALIZE EVERYTHING
ROUTH	: 'H' HELP THE OPERATOR
0	: 'I' BOO800
ROUTJ	: 'J' GO TO NCV11 JOYSTICK CALIBRATION
0	: 'K' BOO800
ROUTL	: 'L' GET LOWER THRESHOLD FROM KEYBOARD
ROUTM	: 'M' SELECT VSV01 DISPLAY USING 2ND BIT MAP
ROUTN	: 'N' GET NEW DATA FROM CAMERA
ROUTO	: 'O' GET ADDRESS OF OTHER TERMINAL
0	: 'P' BOO800
0	: 'Q' BOO800
ROUTR	: 'R' REGULAR GAIN =1
ROUTS	: 'S' STOP COLLECTION
ROUTT	: 'T' DISPLAY INTENSITY TEST IMAGE ON SELECTED DISPLAY
ROUTU	: 'U' GET UPPER THRESHOLD FROM KEYBOARD
0	: 'V' BOO800
ROUTW	: 'W' SELECT VSV01 DISPLAY USING 1ST BIT MAP
ROUTX	: 'X' CHANGE CAMERA CHANNEL
0	: 'Y' BOO800
ROUTZ	: 'Z' ZOOM - GAIN = 2

```

404 .SBTTL COMMAND DECODER
405 ;REPORTS ILLEGAL KEYBOARD CHARACTERS
406
407 002026 012737 000077 001264 BOOB00: MOV #77,TTYOUT ;SET UP '?'
408 002034 004737 005164 JSR PC,TYPO ;TYPE IT
409 002040 004737 005144 JSR PC,TPCR ;DO A 'CR'
410 002044 000713 BR LISN ;GO LOOK FOR GOOD CHAR
411
412 002046 012737 020000 001310 ROUTA: MOV #MATRIX,TABLEX ;WILL DISPLAY ISOTOPE A
413 002054 000137 003026 JMP CHANGE ;GO DO IT
414 002060 012737 040000 001310 ROUTB: MOV #MATRIX+20000,TABLEX ;WILL DISPLAY ISOTOPE B
415 002066 000137 003026 JMP CHANGE ;GO DO IT
416 002072 004737 004576 ROUTC: JSR PC,NCSTRT ;GO START NCV11
417 002076 000137 001674 JMP LISN ;COLLECT DATA UNTIL KEYBRD COMMAND
418 002102 000137 003026 ROUTD: JMP CHANGE ;GO DISPLAY DATA
419 002106 004737 004712 ROUTE: JSR PC,ERASE ;GO ERASE SCOPE
420 002112 000137 001674 JMP LISN ;GO WAIT ON NEXT COMMAND
421 002116 012737 177777 001312 ROUTF: MOV #-1,FREERN ;SELECT FREE RUN MODE
422 002124 004737 004660 JSR PC,NCSTP1 ;GO STOP NCV11 & CLR CORE MATRIX AREA
423 002130 004737 004576 JSR PC,NCSTRT ;GO START NCV11
424 002134 005000 CLR RO ;SET UP TIMER
425 002136 013701 001172 MOV TIME,R1 ;SET UP GROSS TIMER VALUE
426 002142 005300 1$: DEC RO ;COUNT
427 002144 001376 BNE 1$ ;WAIT FOR ZERO
428 002146 042737 000040 001262 BIC #BIT5,KBUFF ;LOOK FOR POSSIBLE STOP KEY - RID LOWER CASE
429 002154 022737 000123 001262 CMP #123,KBUFF ;STOP BEEN STRUCK?
430 002162 001007 BNE 2$ ;BR IF NOT
431 002164 052777 000400 177012 BIS #BIT8,@NCSFR ;STOP NCV11
432 002172 005077 176776 CLR @NCCSR ;ZERO NCV CSR
433 002176 000137 001650 JMP LISN ;GO AWAIT NEXT COMMAND
434 002202 005301 2$: DEC R1 ;DO LOOP AGAIN?
435 002204 001356 BNE 1$ ;BR IF SO
436 002206 000137 003026 JMP CHANGE ;NOW GO DISPLAY DATA JUST COLLECTED
437 002212 004737 004712 ROUTG: JSR PC,ERASE ;GO ERASE SELECTED DISPLAY
438 002216 000137 001554 JMP START1 ;INITIALIZE AND START FRESH
439 002222 005737 001320 ROUTH: TST NOVSV ;CHECK IF DISPLAY
440 002226 001016 BNE 1$ ;BR IF NO VSV01
441 002230 004737 004712 JSR PC,ERASE ;CLEAR THE SCREEN
442 002234 000240 NOP
443 002236 000240 NOP
444 002240 012777 002000 176752 MOV #2000,@VTVP0S ;POSITION THE READ-OUT
445 002246 004537 005202 JSR R5,VTWRIT ;TELL THE OPERATOR VIA VSV01
446 002252 010422 HELPO
447 002254 004537 005202 JSR R5,VTWRIT
448 002260 011120 HELP1
449 002262 000402 BR 2$
450 002264 104401 011120 1$: TYPE, HELP1 ;NO VSV01 - TELL OPERATOR SHORT LIST
451 002270 000137 001674 2$: JMP LISN
452 002274 000137 004224 ROUTJ: JMP TJOY ;CALB. THE JOYSTICK GO TO IT
453 002300 012746 000340 ROUTL: MOV #340,-(SP) ;PUSH HIGH PSW ON STACK
454 002304 012746 002312 MOV #1$,-(SP) ;PUSH RETURN ADDRESS
455 002310 000002 RTI ;FAKE RTI TO RAISE PSW
456 002312 104401 010040 1$: TYPE ,MSG2 ;ASK FOR OCTAL DATA
457 002316 104411 RDOCT ;GO GET IT

```

458	002320	012637	001266		MOV	(SP)+,THLO	;SAVE IT
459	002324	005046			CLR	-(SP)	;PUSH LOW PSW
460	002326	012746	002334		MOV	#2\$,-(SP)	
461	002332	000002			RTI		
462	002334	000137	003026		JMP	CHANGE	;GO DISPLAY
463	002340	012737	177777	001314	ROUTM: MOV	#-1,MSELCT	;SELECT 2ND BIT MAP
464	002346	004737	004440		JSR	PC,SELCTA	;RELOAD ADDRESSES
465	002352	000240			NOP		
466	002354	013700	001222		MOV	VTVC SR,R0	;GET 2ND BIT MAP ADRS
467	002360	042760	000400	177760	BIC	#BIT8,-20(R0)	;TURN OFF 1ST BIT MAP
468	002366	000137	001674		JMP	LISN	;GO AWAIT NEXT COMMAND
469	002372	004737	004660		ROUTN: JSR	PC,NCSTP1	;GO STOP NCV11 & CLR CORE MATRIX
470	002376	004737	004576		JSR	PC,NCSTRT	;START THE NCV11
471	002402	000137	001674		JMP	LISN	;COLLECT DATA & AWAIT NEXT KEYBRD COMMAND
472	002406	012746	000340		ROUTO: MOV	#340,-(SP)	;RAISE PS
473	002412	012746	002420		MOV	#1\$,-(SP)	
474	002416	000002			RTI		
475	002420	104401	010165		1\$: TYPE,	MSG4	;ASK OPR FOR ADDRESS
476	002424	104411			RDOCT		;GET HIS INPUT
477	002426	012600			MOV	(SP)+,R0	;GET ADDRESS
478	002430	001002			BNE	2\$	;BR IF NOT 'CR' OR 0
479	002432	000137	002026		JMP	BOOB00	;FAT FINGER OPERATOR
480	002436	013746	000004		2\$: MOV	@ERRVEC,-(SP)	;SAVE LOC 4
481	002442	012737	002510	000004	MOV	#3\$,@ERRVEC	;SAVE IF WRONG ADDRESS BY OPR.
482	002450	005710			TST	(R0)	;REF. THE NEW CONSOLE ADDR.
483	002452	012701	001144		MOV	#STKS,R1	;GET OLD ADDR. POINTER
484	002456	010021			MOV	R0,(R1)+	;LOAD NEW ADDRESS
485	002460	005720			TST	(R0)+	;BUMP ADDRESS
486	002462	010021			MOV	R0,(R1)+	;LOAD NEW ADDRESS
487	002464	005720			TST	(R0)+	;BUMP ADDRESS
488	002466	010021			MOV	R0,(R1)+	;LOAD NEW ADDRESS
489	002470	005720			TST	(R0)+	;BUMP ADDRESS
490	002472	010021			MOV	R0,(R1)+	;LOAD NEW ADDRESS
491	002474	012637	000004		MOV	(SP)+,@ERRVEC	;RESTORE LOC 4.
492	002500	005046			CLR	-(SP)	
493	002502	012746	001322		MOV	#START,-(SP)	;LOWER PS AND START PROG AGAIN
494	002506	000002			RTI		
495	002510	022626			3\$: CMP	(SP)+,(SP)+	;CLEAN STACK
496	002512	012637	000004		MOV	(SP)+,@ERRVEC	;RESTORE LOC. 4
497	002516	005046			CLR	-(SP)	;LOWER PS
498	002520	012746	002026		MOV	#BOOB00,-(SP)	;RETURN
499	002524	000002			RTI		
500	002526	005037	001276		ROUTR: CLR	GAIN	;WANT REGULAR GAIN
501	002532	042777	002000	176434	BIC	#BIT10,@NCCSR	;INSURE REGULAR GAIN
502	002540	000137	001674		JMP	LISN	;LOOK FOR NEXT COMMAND
503	002544	052777	000400	176432	ROUTS: BIS	#BIT8,@NCSFR	;STOP NCV11
504	002552	005077	176416		CLR	@NCCSR	;ZERO NCV CSR
505	002556	000137	001674		JMP	LISN	;LOOK FOR NEXT COMMAND
506	002562	004737	004636		ROUTT: JSR	PC,NCSTP	;GO STOP THE NCV11 IF RUNNING
507	002566	004737	005036		JSR	PC,LDIMGE	;GO SET UP TEST CORE IMAGE
508	002572	000137	003026		JMP	CHANGE	;GO DISPLAY IT
509	002576	012746	000340		ROUTU: MOV	#340,-(SP)	
510	002602	012746	002610		MOV	#1\$,-(SP)	
511	002606	000002			RTI		

512	002610	104401	010040		1\$:	TYPE	,MSG2		;ASK FOR OCTAL DATA
513	002614	104411				RDOCT			;GO GET IT
514	002616	012637	001270			MOV	(SP)+,THHI		;SAVE IT
515	002622	005046				CLR	-(SP)		
516	002624	012746	002632			MOV	#2\$,-(SP)		
517	002630	000002				RTI			
518	002632	000137	003026		2\$:	JMP	CHANGE		;GO DISPLAY
519	002636	005037	001314		ROUTW:	CLR	MSELECT		;SELECT 1ST BIT MAP
520	002642	004737	004440			JSR	PC,SELCTA		;CHANGE ADDRESSES
521	002646	000240				NOP			
522	002650	013700	001222			MOV	VTVCSR,R0		;GET 1ST BIT MAP ADRS
523	002654	042760	000400	000020		BIC	#BIT8,20(R0)		;TURN OFF 2ND BIT MAP
524	002662	000137	001674			JMP	LISN		;GO AWAIT NEXT COMMAND
525	002666	012746	000340		ROUTX:	MOV	#340,-(SP)		
526	002672	012746	002700			MOV	#1\$,-(SP)		
527	002676	000002				RTI			
528	002700	104401	010361		1\$:	TYPE	,MSG7		;TELL OPERATOR TO SELECT CAMERA
529	002704	104411				RDOCT			;WAIT FOR HIS INPUT
530	002706	012637	003002			MOV	(SP)+,10\$		;GET CHAR.
531	002712	000240				NOP			
532	002714	000240				NOP			
533	002716	000240				NOP			
534	002720	042737	177774	003002		BIC	#177774,10\$		;MASK OFF BITS
535	002726	005237	003002			INC	10\$		
536	002732	005037	003004			CLR	11\$		;CLEAR ACTUAL VALUE
537	002736	005337	003002		2\$:	DEC	10\$		
538	002742	001403				BEQ	3\$		
539	002744	005237	003004			INC	11\$		;UPDATE ACUTAL
540	002750	000772				BR	2\$		
541	002752	113737	003004	001275	3\$:	MOVW	11\$,CAMERA+1		;UPDATE CAMERA SAVE LOC.
542	002760	053777	001274	176206		BIS	CAMERA,@NCCSR		;AND SELECT THAT CAMERA
543	002766	005046				CLR	-(SP)		
544	002770	012746	002776			MOV	#4\$,-(SP)		
545	002774	000002				RTI			
546	002776	000137	003026		4\$:	JMP	CHANGE		;RETURN
547	003002	000000			10\$:	0			
548	003004	000000			11\$:	0			
549	003006	012737	002000	001276	ROUTZ:	MOV	#BIT10,GAIN		;ENABLE GAIN
550	003014	053777	001276	176152		BIS	GAIN,@NCCSR		;SET IT AT NCV CSR
551	003022	000137	001674			JMP	LISN		;GO AWAIT NEXT COMMAND

```

552          .SBTTL PROGRAM ROUTINES
553          ;GO CLEAR SCREEN OF SELECTED DISPLAY
554
555 003026 004737 004712      CHANGE: JSR      PC,ERASE          ;GO ERASE SCOPE
556
557          ;STOP NCV11 AND GET SET TO DISPLAY
558
559 003032 004737 004636      JSR      PC,NCSTP          ;GO STOP NCV11
560
561          ;NOW DRAW A BOX AROUND POTENTIAL MATRIX DISPLAY
562          ;AREA ON THE SELECTED DISPLAY
563 003036 012700 001750      BOX:  MOV      #1750,R0          ;SET UP BIT MAP ADRS - TOP LINE
564 003042 012701 073567      MOV      #73567,R1          ;SET UP PIXEL DATA IN R1 - INT 7
565 003046 004737 004744      JSR      PC,DISPY          ;GO LOAD BIT MAP
566 003052 005200              1$:  INC      R0          ;ADVANCE ADRS
567 003054 022700 001770      CMP      #1770,R0          ;TOP LINE DONE?
568 003060 001403              BEQ      2$              ;BR IF SO
569 003062 004737 004756      JSR      PC,DCONT          ;LOAD NEXT PIXEL WD
570 003066 000771              BR      1$              ;NEXT MAP LOAD
571 003070 012700 006010      2$:  MOV      #6010,R0          ;SET UP BIT MAP ADRS - BOT LINE
572 003074 004737 004744      JSR      PC,DISPY          ;GO LOAD BIT MAP
573 003100 005200              3$:  INC      R0          ;ADVANCE ADRS
574 003102 022700 006030      CMP      #6030,R0          ;BOT LINE DONE?
575 003106 001403              BEQ      4$              ;BR IF SO
576 003110 004737 004756      JSR      PC,DCONT          ;LOAD NEXT PIXEL WD
577 003114 000771              BR      3$              ;NEXT MAP LOAD
578 003116 012700 001730      4$:  MOV      #1730,R0          ;SET UP BIT MAP ADRS SIDE LINES
579 003122 062700 000017      5$:  ADD      #17,R0          ;OFFSET NEXT ROW
580 003126 012701 070000      MOV      #70000,R1          ;SET UP PIXEL 3 DATA IN R1 - INT 7
581 003132 022700 006047      CMP      #6047,R0          ;AT BOTTOM OF SCREEN?
582 003136 001411              BEQ      6$              ;GO START VSV01 DISPLAY
583 003140 004737 004744      JSR      PC,DISPY          ;GO LOAD BIT MAP
584 003144 012701 000007      MO.      #7,R1          ;SET UP PIXEL 0 DATA IN R1 - INT 7
585 003150 062700 000021      ADD      #21,R0          ;OFFSET TO RIGHT LINE
586 003154 004737 004744      JSR      PC,DISPY          ;GO LOAD BIT MAP
587 003160 000760              BR      5$              ;DO NEXT ROW
588 003162 013777 001250 176032 6$:  MOV      VTVSAV,@VTVCSR          ;START DISPLAY
589 003170 012737 002010 001254      MOV      #2010,MAPADR          ;OFFSET BIT MAP ADRS
590 003176 012737 000003 001256      MOV      #3,PIXCNT          ;SET UP PIXEL BYTE COUNT
591 003204 005037 001260              CLR      PIXASM          ;CLR PIXEL ASSEMBLY WORD
592
593          ;NOW LETS FIND THE LARGEST CELL
594 003210 013737 001300 001302      LARGES: MOV      TOTSIZ,CHARCT          ;NUMBER OF ELEMENTS TO CONSIDER
595 003216 017700 176066      MOV      @TABLEX,R0          ;THIS IS FIRST GUESS
596 003222 013701 001310      MOV      TABLEX,R1          ;R1 POINTS TO TABLE
597 003226 020021              1$:  CMP      R0,(R1)+          ;COMPARE GUESS AGAINST NEW
598 003230 103005              BHS      3$              ;GUESS STILL GOOD
599 003232 024137 001270      CMP      -(R1),THHI          ;GUESS SMALLER BUT CHECK NEW
600 003236 101001              BHI      2$              ;HI AGAINST UPPER THRESHOLD
601 003240 011100              MOV      (R1),R0          ;WITHIN BOUNDS. MAKE THIS NEW HI
602 003242 005721              2$:  *ST      (R1)+          ;INCREASE REGISTER BY 2
603 003244 005337 001302      3$:  DEC      CHARCT          ;COUNT THIS LAST COMPARISON
604 003250 001366              BNE      1$

```

```

605 ;THE LARGEST CELL WITHIN THE UPPER THRESHOLD IS NOW IN R0
606 ;NOW ACCOUNT FOR LOWER THRESHOLD - BOW OUT IF TOO SMALL
607 003252 163700 001266 SUB THLO,R0 ;SUBTRACT LOWER THRESHOLD
608 003256 101002 BHI 4$ ;BR IF CELL VALUE GREATER THAN LO THRESHOLD
609 003260 000137 003612 JMP PATWRT ;DON'T DISPLAY-ALL VALUES BELOW LO THRESHOLD
610 003264 010037 001304 4$: MOV R0,CPERL ;SAVE LARGEST CELL OF MATRIX
611
612 ;NOW PICK OUT EACH VALUE IN CORE MATRIX AND SCALE TO THE
613 ;PROPER INTENSITY LEVEL. THEN DISPLAY IT ON THE SELECTED DISPLAY
614
615 003270 013737 001310 001252 DMATRX: MOV TABLEX,MRXADR ;BEGIN AT TOP LEFT ROW
616 003276 062737 017600 001252 ADD #8064.,MRXADR ;OFFSET TO BOTTOM OF CORE MATRIX
617 003304 012737 000100 001306 MOV #64.,ROWCNT ;THERE ARE 64 CELLS PER ROW
618 003312 017702 175734 DISLOP: MOV @MRXADR,R2 ;GET A CELL VALUE FROM MATRIX
619 003316 062737 000002 001252 ADD #2,MRXADR ;BUMP MATRIX ADRS
620 003324 005337 001306 DEC ROWCNT ;COUNT CELL THIS ROW
621 003330 001006 BNE 1$ ;BR IF ROW NOT FINISHED
622 003332 012737 000100 001306 MOV #64.,ROWCNT ;RESET NEXT ROW COUNT
623 003340 162737 000400 001252 SUB #256.,MRXADR ;SET UP FOR NEXT ROW IN MATRIX
624 003346 020237 001270 1$: CMP R2,THHI ;CELL WITHIN HI THRESHOLD?
625 003352 101003 BHI 2$ ;BR IF NOT
626 003354 163702 001266 SUB THLO,R2 ;SUB LOW THRESHOLD
627 003360 101002 BHI SCLCEL ;BR IF CELL ABOVE LOW THRESHOLD
628 003362 005004 2$: CLR R4 ;SET CELL TO LOWEST INTENSITY
629 003364 000423 BR MAPLD
630 003366 013701 001304 SCLCEL: MOV CPERL,R1 ;NOW ESTABLISH THE INTENSITY LEVEL
631 003372 012703 000005 MOV #5,R3 ;SCALE TO 16 LEVELS
632 003376 005004 CLR R4
633 003400 006304 2$: ASL R4 ;MUL BY 2
634 003402 020201 CMP R2,R1 ;COMPARE THIS CELL TO LARGEST
635 003404 103404 BLO 4$ ;BR IF SMALLER
636 003406 005701 TST R1 ;CHECK CASE WHERE 0 DEVISOR
637 003410 001401 BEQ 3$ ;BR IF SO
638 003412 005204 INC R4 ;ACCOUNT FOR GOOD SUBTRACTION
639 003414 160102 3$: SUB R1,R2 ;NO DO THE SUBTRACTION
640 003416 000241 4$: CLC
641 003420 006001 ROR R1 ;MAKE DEVISOR SMALLER
642 003422 005303 DEC R3 ;COUNT POSITION
643 003424 001365 BNE 2$ ;AGAIN IF NOT SCALED TO 16 LEVELS YET
644 003426 160102 SUB R1,R2 ;ACCOUNT FOR REMAINDER
645 003430 101401 BLOS MAPLD ;BR IF TOO SMALL
646 003432 005204 INC R4 ;ROUND UP

```

```

647                                     ;THIS CODE DISPLAYS EACH SCALED CELL ON THE VSV01 (ONE OF 16 LEVELS)
648
649 003434 013700 001254 MAPLD: MOV MAPADR,R0 ;SET UP BIT MAP ADRS
650 003440 005704 TST R4 ;LOOK FOR NO INTENSITY
651 003442 001401 BEQ 1$ ;BR IF NO INTENSITY
652 003444 005304 DEC R4 ;OFFSET TO 0-17
653 003446 000304 1$: SWAB R4 ;PREPARE FOR PIXEL LOC
654 003450 006304 ASL R4 ;MOVE TO TOP 4 BITS
655 003452 006304 ASL R4
656 003454 006304 ASL R4
657 003456 006304 ASL R4
658 003460 000241 CLC ;NOW ASSEMBLE THIS PIXEL INTO PIXEL WORD
659 003462 006037 001260 ROR PIXASM ;NOW MAKE ROOM IN PIXEL WORD
660 003466 006037 001260 ROR PIXASM
661 003472 006037 001260 ROR PIXASM
662 003476 006037 001260 ROR PIXASM
663 003502 060437 001260 ADD R4,PIXASM ;ADD THIS PIXEL TO OTHERS
664 003506 005737 001256 TST PIXCNT ;ALL 4 PIXELS DONE FOR THIS WORD?
665 003512 001004 BNE 2$ ;BR IF NOT
666 003514 013701 001260 MOV PIXASM,R1 ;LD PIXEL WORD INTO R1
667 003520 004737 004744 JSR PC,DISPY ;LOAD BIT MAP
668 003524 005337 001256 2$: DEC PIXCNT ;COUNT PIXEL
669 003530 100270 BPL DISLOP ;BR IF 4 PIXELS NOT ASSEMBLED YET
670 003532 005037 001260 CLR PIXASM ;CLR PIXEL ASSEMBLY WORD
671 003536 012737 000003 001256 MOV #3,PIXCNT ;RESET PIXEL COUNT
672 003544 005237 001254 INC MAPADR ;ADVANCE MAP ADRS
673 003550 013700 001254 MOV MAPADR,R0 ;LOAD INTO R0
674 003554 022700 005770 CMP #5770,R0 ;HAVE ALL 64 ROWS BEEN DONE?
675 003560 001002 BNE 3$ ;BR IF NOT
676 003562 000137 003612 JMP PATWRT ;NOW GO DISPLAY PARAMETERS
677 003566 042700 177740 3$: BIC #177740,R0 ;SAVE ROW POSITION BITS
678 003572 022700 000030 CMP #30,R0 ;NOW LOOK FOR END OF ROW
679 003576 001003 BNE 4$ ;BR IF NOT AT END
680 003600 062737 000020 001254 ADD #20,MAPADR ;ADVANCE MAP ADRS TO NEXT ROW
681 003606 000137 003312 4$: JMP DISLOP ;GO GET NEXT MATRIX DATUM

```

```

682                                     ;CODE TO WRITE PARAMETER DATA AT BOTTOM OF SCREEN
683
684 003612 012777 012000 175400 PATWRT: MOV    #12000,@VTVPOS ;VSV01 - SET CHAR POS,LINE 20, LEFT MARGIN
685 003620 004537 005202          JSR      R5,VTWRIT
686 003624 007550                WD001          ;'CR,LOWER THRESHOLD'
687 003626 004537 005226          JSR      R5,AWRIT          ;DISPAY OCTAL
688 003632 001266                THLO
689 003634 004537 005202          JSR      R5,VTWRIT
690 003640 007573                WD002          ;'CR, UPPER THRESHOLD'
691 003642 004537 005226          JSR      R5,AWRIT          ;DISPAY OCTAL
692 003646 001270                THHI
693 003650 113737 001275 001242    MOVB     CAMERA+1,DUMMY1 ;GET CAMERA VALUE
694 003656 062737 000060 001242    ADD      #60,DUMMY1      ;MAKE ASCII
695 003664 113737 001242 007706    MOVB     DUMMY1,CAMOUT ;SAVE FOR READOUT
696                                     ;TEST THE 'PB' FLAG
697 003672 032777 000040 175304    BIT      #BIT5,@NCSFR ;TEST FOR 'PB' FLAG
698 003700 001007                BNE      1$              ;BR IF SET
699 003702 112737 000125 007722    MOVB     #'U,PBUP          ;NO- TELL OPER. IT WAS UP
700 003710 112737 000120 007723    MOVB     #'P,PBUP+1
701 003716 000406                BR       2$
702 003720 112737 000104 007722    1$: MOVB     #'D,PBUP          ;TELL OPER. IT WAS DN
703 003726 112737 000116 007723    MOVB     #'N,PBUP+1
704                                     ;TEST FOR JOY-STICK BUTTON FLAG
705 003734 032777 000100 175242    2$: BIT      #BIT6,@NCSFR ;TEST FOR 'JOY-BUTTON' FLAG
706 003742 001007                BNE      3$              ;BR IF SET
707 003744 112737 000125 007737    MOVB     #'U,JBUP          ;TELL OPER. IT WAS UP
708 003752 112737 000120 007740    MOVB     #'P,JBUP+1
709 003760 000406                BR       4$
710 003762 112737 000104 007737    3$: MOVB     #'D,JBUP          ;TELL OPER. IT WAS DN
711 003770 112737 000116 007740    MOVB     #'N,JBUP+1
712 003776 004537 005202          4$: JSR      R5,VTWRIT          ;DISPLAY CAMERA VALUE AND BUTTON STATUS
713 004002 007673                WD007
714 004004 004537 005202          JSR      R5,VTWRIT
715 004010 007621                WD003          ;'Z COUNT ='
716 004012 017737 175164 001242    MOV      @NCBAR,DUMMY1
717 004020 017737 175154 001244    MOV      @NCWCR,DUMMY2
718 004026 004537 005226          JSR      R5,AWRIT          ;DISPALY OCTAL
719 004032 001244                DUMMY2
720 004034 004537 005202          JSR      R5,VTWRIT          ;INSERT '-'
721 004040 010416                DASH
722 004042 004537 005226          JSR      R5,AWRIT          ;DISPLAY OCTAL
723 004046 001242                DUMMY1
724 004050 004537 005202          JSR      R5,VTWRIT
725 004054 007634                WD004          ;'MATRIX COUNT='
726

```

```

727      ;DETERMINE THE # OF COUNTS IN THE MATRIX
728
729 004056 013737 001300 001234      MOV      TOTSIZ,TEMPO
730 004064 005002      CLR      R2
731 004066 005003      CLR      R3
732 004070 013701 001310      MOV      TABLEX,R1
733 004074 062103      MXSML: ADD      (R1)+,R3      ;NOW ADD UP ALL VALUES IN MATRIX
734 004076 005502      ADC      R2
735 004100 005337 001234      DEC      TEMPO
736 004104 001373      BNE      MXSML
737 004106 010337 001242      MOV      R3,DUMMY1
738 004112 010237 001244      MOV      R2,DUMMY2
739 004116 004537 005226      JSR      R5,AWRIT      ;TELL OPER. OCTAL
740 004122 001244      DUMMY2
741 004124 004537 005202      JSR      R5,VTWRIT      ;INSERT DASH
742 004130 010416      DASH
743 004132 004537 005226      JSR      R5,AWRIT
744 004136 001242      DUMMY1
745 004140 004537 005202      JSR      R5,VTWRIT
746 004144 011500      BCRLF
747
748      ;TEST FOR GAIN 2 AND B GAMMA STATES
749
750 004146 005737 001276      TST      GAIN
751 004152 001403      BEQ      1$
752 004154 004537 005202      JSR      R5,VTWRIT
753 004160 007653      WD005      ;'ZOOM'
754 004162 023727 001310 ^,0000 1$: CMP      TABLEX,#MATRIX+20000 ;ISOTOPE B?
755 004170 001003      BNE      2$      ;NO
756 004172 004537 005202      JSR      R5,VTWRIT      ;YES
757 004176 007662      WD006      ;'B-GAMMA'
758 004200 005737 001312      2$: TST      FREERN      ;IN FREE RUN MODE?
759 004204 001402      BEQ      3$      ;BR IF NOT
760 004206 000137 002116      JMP      ROUTF      ;YES, GO GET NEW DATA
761 004212 013777 001272 174754 3$: MOV      COMSAV,@NCCSR ;RESUME THE NCV11
762 004220 000137 001674      JMP      LISN      ;DONE WITH MESSAGES
  
```

```

763 ;THIS CODE DRAWS CROSS HAIRS(VSV01) WITH THE X & Y JOYSTICK DATA
764 ;THE BUG WILL FOLLOW THE JOYSTICK WHEN THE INTERRUPT BAR IS NOT DEPRESSED -
765 ;ALL PARTS WITHIN THE DISPLAYED BOX ON THE SELECTED DISPLAY SHOULD
766 ;BE ACCESSIBLE IN A UNIFORM MANNER
767 ;A CNTRL 'C' WILL GET USER BACK TO KEYBOARD MONITOR
768 004224 004737 004712 TJOY: JSR PC,ERASE ;START FRESH
769
770 ;DRAW A BOX UP ON VSV01 SCREEN FOR NCV11 JOYSTICK CALIBRATION
771 004230 005000 CLR R0 ;SET UP BIT MAP ADRS - TOP LINE
772 004232 012701 073567 MOV #73567,R1 ;SET UP PIXEL DATA IN R1 - INT 7
773 004236 004737 004744 JSR PC,DISPY ;GO LOAD BIT MAP
774 004242 005200 1$: INC R0 ;ADVANCE BIT MAP ADRS
775 004244 022700 000040 CMP #40,R0 ;TOP LINE DONE?
776 004250 001403 BEQ 2$ ;BR IF SO
777 004252 004737 004756 JSR PC,DCONT ;LOAD NEXT PIXEL
778 004256 000771 BR 1$ ;NEXT MAP LOAD
779 004260 012700 007740 2$: MOV #7740,R0 ;SET UP BIT MAP ADRS - BOT LINE
780 004264 004737 004744 JSR PC,DISPY ;GO LOAD BIT MAP
781 004270 005200 3$: INC R0 ;ADVANCE ADRS
782 004272 022700 010000 CMP #10000,R0 ;BOT LINE DONE?
783 004276 001403 BEQ 4$ ;BR IF SO
784 004300 004737 004756 JSR PC,DCONT ;LOAD NEXT PIXEL WORD
785 004304 000771 BR 3$ ;NEXT MAP LOAD
786 004306 012700 000040 4$: MOV #40,R0 ;SET UP BIT MAP ADRS SIDE LINES
787 004312 012701 000007 5$: MOV #7,R1 ;SET UP PIXEL 0 DATA IN R1 - INT 7
788 004316 004737 004744 JSR PC,DISPY ;GO LOAD BIT MAP
789 004322 012701 070000 MOV #70000,R1 ;SET UP PIXEL 3 DATA IN R1 - INT 7
790 004326 062700 000037 ADD #37,R0 ;OFFSET TO RIGHT SIDE LINE
791 004332 004737 004744 JSR PC,DISPY ;GO LOAD BIT MAP
792 004336 005200 INC R0 ;GO TO NEXT ROW ON LEFT
793 004340 022700 007740 CMP #7740,R0 ;TO BOTTOM YET?
794 004344 001362 BNE 5$ ;BR IF NOT
795 004346 013777 001250 174646 MOV VTVSAV,@VTVCSR ;ENABLE BIT MAP
796
797 ;NOW COLLECT ANALOG DATA FROM NCV11 JOYSTICK
798
799 004354 012777 004000 174622 DISBG: MOV #BIT11,@NCSFR ;CLEAR THE DEVICE
800 004362 052777 000001 174614 1$: BIS #BIT0,@NCSFR ;CONVERT JOYSTICK
801 004370 105777 174610 2$: TSTB @NCSFR ;DONE?
802 004374 100375 BPL 2$ ;BR IF NOT
803 004376 017737 174606 001234 MOV @NCJOY,TEMPO ;STORE CONVERSION VALUES
804 004404 000240 NOP
805 004406 000240 NOP
806 004410 000240 NOP
807 004412 004737 005356 JSR PC,XYAVE ;GO AVG LAST 32. X-Y JOYSTICK VALUES
808 004416 032777 000040 174560 BIT #BIT5,@NCSFR ;LOOK FOR JOY BOTTON DOWN
809 004424 001356 BNE 1$ ;DON'T MOVE THE BUG
810 004426 000240 NOP
811 004430 000240 NOP
812 004432 004737 005006 JSR PC,DISPY2 ;GO DISPLAY CROSS HAIRS - VSV01
813 004436 000751 BR 1$ ;DO ANOTHER CONVERSION

```

```

814
815
816
817
818
819 004440 013700 001166
820 004444 012701 001214
821 004450 010021
822 004452 062700 000002
823 004456 022701 001222
824 004462 001372
825 004464 013700 001170
826 004470 005737 001314
827 004474 001402
828 004476 062700 000020
829 004502 010021
830 004504 062700 000002
831 004510 022701 001234
832 004514 001372
833 004516 012737 004570 000004
834 004524 005777 174464
835 004530 005777 174466
836 004534 013700 001246
837 004540 010077 174466
838 004544 062700 000401
839 004550 032700 010000
840 004554 001771
841 004556 000241
842 004560 012737 000006 000004
843 004566 000207
844 004570 022626
845 004572 000261
846 004574 000771
847
848
849
850
851 004576 012777 004030 174370
852 004604 012777 020000 174364
853 004612 053777 001276 174354
854 004620 053777 001274 174346
855 004626 052777 000001 174340
856 004634 000207
857
858
859
860
861 004636 017737 174332 001272
862 004644 052777 000400 174332
863 004652 005077 174316
864 004656 000207

;*****
;ROUTINE SELECTS VSV01 DISPLAY - SETS UP BUS ADRS AND INTENSITY LEVEL
;AND INTENSITY LOOK-UP TABLE - THE CARRY BIT IS SET ON EXIT IF THE
;VSV01 IS NOT SEEN AT THE ASSIGNED BUS ADDRESS
;*****
SELCTA: MOV VTVADR,R0 ;GET VSV01 BASE ADRS
MOV #VTVCRG,R1 ;GET PTR ADRS
1$: MOV R0,(R1)+ ;SET UP REG ADRS PTRS
ADD #2,R0 ;BUMP REG ADRS
CMP #VTVCSR,R1 ;CHAR GEN REGS ALL SET UP?
BNE 1$ ;BR IF NOT
MOV VTMADR,R0 ;GET BASE ADRS OF BIT MAP
TST MSELECT ;USING SECOND MAP?
BEQ 2$ ;BR IF NOT
ADD #20,R0 ;POINT TO 2ND BIT MAP ADRS'S
2$: MOV R0,(R1)+ ;CONTINUE TO BIT MAP ADRS'S
ADD #2,R0 ;BUMP REG ADRS
CMP #VTVINT+2,R1 ;ALL SET UP?
BNE 2$ ;BR IF NOT
MOV #5$,@#ERRVEC ;SET UP BUS TIMEOUT RETURN ADRS IF NO VSV01
TST @VTVCRG ;IS CHARACTER GENERATOR THERE?
TST @VTVCSR ;IS BIT MAP THERE?
MOV INTLUT,R0 ;SET UP ADRS & DATA OF INTENSITY LOOK-UP TABLE
3$: MOV R0,@VTVINT ;SET UP TABLE
ADD #401,R0 ;ADVANCE ADRS & INTENSITY
BIT #10000,R0 ;TABLE LOADED?
BEQ 3$ ;BR IF NOT
CLC ;ZERO CARRY SAYS VSV01 THERE
4$: MOV #ERRVEC+2,@#ERRVEC ;RESTORE ER TRAP LOC TO PT TO 6
RTS PC ;EXIT
5$: CMP (SP)+,(SP)+ ;FIX STACK SINCE NO RTI
SEC ;CARRY ON EXIT SAYS NO VSV01
BR 4$ ;GO EXIT

;*****
;ROUTINE STARTS NCV11 AT SELECTED GAIN AND CAMERA
;*****
NCSTRT: MOV #4030,@NCCSR ;SET UP ZB ENABLE AND 64*64 WORD MATRIX
MOV #MATRIX,@NCOFF ;ENSURE OFFSET REG. IS SET
BIS GAIN,@NCCSR ;SET UP GAIN
BIS CAMERA,@NCCSR ;SET UP CAMERA
BIS #BIT0,@NCCSR ;SET GO BIT
RTS PC ;RETURN

;*****
;ROUTINE STOPS NCV11 AND SAVES NCV11 STATUS
;*****
NCSTP: MOV @NCCSR,COMSAV ;SAVE THE INTERFACE ACTION
BIS #BIT8,@NCSFR ;DISABLE NPR'S
CLR @NCCSR ;ZERO ALL STATUS
RTS PC ;RETURN

```

```

865                                     ;:*****
866                                     ;ROUTINE STOPS NCV11, SAVES STATUS AND CLEARS MATRIX CORE AREA
867                                     ;:*****
868 004660 004737 004636 NCSTP1: JSR      PC,NCSTP      ;GO STOP NCV11 AND SAVE STATUS
869 004664 005077 174310      CLR      @NCWCR      ;CLEAR HIGH WORD
870 004670 005077 174306      CLR      @NCBAR      ;CLEAR LOW WORD
871 004674 012700 020000      MOV      #MATRIX,R0    ;GET SET TO ZERO CORE MATRIX AREA
872 004700 005020 1$:        CLR      (R0)+          ;ZERO LOC
873 004702 020027 060000      CMP      R0,#MATRIX+40000 ;ALL DONE?
874 004706 001374      BNE      1$                ;BR IF MORE
875 004710 000207      RTS      PC                ;RETURN
876
877                                     ;:*****
878                                     ;ROUTINE WILL ERASE DISPLAY
879                                     ;:*****
880 004712 052777 001000 174302 ERASE: BIS      #1000,@VTVCSR ;ERASE DISPLAY (CLR BIT MAP)
881 004720 105777 174276 1$:    TSTB     @VTVCSR      ;LOOK FOR READY
882 004724 100375      BPL      1$                ;WAIT FOR IT
883 004726 042777 001000 174266 BIC      #1000,@VTVCSR ;TURN OFF ERASE DISPLAY
884 004734 012777 002035 174252 MOV      #2035,@VTVCRG ;CLR CHAR SCREEN & DISABLE CURSOR
885 004742 000207      RTS      PC                ;EXIT
886
887                                     ;:*****
888                                     ;ROUTINE WILL LOAD BIT MAP (VSV01)
889                                     ;R0 CONTAINS BIT MAP ADRS AND R1 THE BIT MAP DATA
890                                     ;:*****
891 004744 042777 000400 174250 DISPY: BIC      #400,@VTVCSR ;STOP DISPLAY
892 004752 010077 174246      MOV      R0,@VTVMAP    ;LOAD BIT MAP ADRS
893 004756 042777 000400 174236 DCONT: BIC      #400,@VTVCSR ;AGAIN IF ENTERING HERE
894 004764 105777 174232 1$:    TSTB     @VTVCSR      ;READY?
895 004770 100375      BPL      1$                ;WAIT THEN
896 004772 010177 174230      MOV      R1,@VTVPX     ;LOAD BIT MAP
897 004776 052777 000400 174216 BIS      #400,@VTVCSR ;RESUME DISPLAY
898 005004 000207      RTS      PC                ;RETURN
899
900                                     ;:*****
901                                     ;ROUTINE WILL DISPLAY X & Y CROSS HAIRS ON VSV01
902                                     ;:*****
903 005006 005777 174202 DISPY2: TST      @VTVCRG      ;READY?
904 005012 100375      BPL      DISPY2            ;WAIT IF NOT
905 005014 105137 001235      COMB     TEMPO+1      ;X NEEDS TO BE INVERTED
906 005020 013777 001234 174170 MOV      TEMPO,@VTVCHP ;LOAD X & Y CROSS HAIRS
907 005026 052777 016000 174160 BIS      #16000,@VTVCRG ;ENABLE THE CROSS HAIRS
908 005034 000207      RTS      PC                ;RETURN

```

```

910      ;:*****
911      ;ROUTINE WILL FILL CORE WITH AN IMAGE THAT WHEN
912      ;DISPLAYED WILL CONTAIN ALL THE INTENSITY LEVELS -
913      ;ROWS AT THE BOTTOM OF THE SCREEN WILL APPEAR BRIGHTEST -
914      ;NOTE THAT THIS IS ONLY A DISPLAY TEST PATTERN FOR
915      ;POSSIBLE DISPLAY ADJUSTMENTS BY THE USER
916      ;:*****
917      LDIMGE: MOV      #100,R0          ;COUNT 64 ROWS
918      MOV      TABLEX,R1          ;GET SELECTED ISOTOPE
919      MOV      #100,R2          ;COUNT 64 DATA POINTS PER ROW
920      MOV      #100,R3          ;100 WILL REPRESENT HIGHEST INTENSITY LEVEL(100-0)
921      1$: MOV      R3,(R1)+        ;LOAD CORE IMAGE
922      DEC      R2                ;DONE ROW?
923      BNE      1$              ;BR IF NOT
924      DEC      R0                ;DONE ROWS?
925      BEQ      2$              ;BR IF SO
926      SUB      #1,R3            ;LOWER NEXT CELL VALUE
927      MOV      #100,R2          ;RESET ROW LENGTH COUNTER
928      BR       1$              ;LOAD THIS ROW IMAGE
929      2$: RTS      PC            ;RETURN FOR DISPLAY
930
931      ;:*****
932      ;KEYBOARD INTERRUPT SERVICE ROUTINE
933      ;:*****
934      KBINT: MOV      @STKB,KBUFF    ;READ KEY BOARD
935      MOV      KBUFF,@STPB          ;ECHO CHAR.
936      BIC      #177600,KBUFF        ;RID PARITY
937      CMP      #3,KBUFF            ;CNTRL 'C'?
938      BNE      1$                  ;BR IF NOT
939      JMP      LISEN                ;ABORT WHATEVER & LOOK FOR NEXT COMMAND
940      1$: RTI
941
942      ;:*****
943      ;ROUTINE TYPES 'CR' AND 'LF' OR CHAR IN TTYOUT
944      ;:*****
945      TYPGR: MOV      #15,TTYOUT      ;SET UP FOR A 'CR'
946      JSR      PC,TYP0
947      MOV      #12,TTYOUT            ;SET UP FOR LF
948      TYP0: TSTB      @STPS          ;WAIT FOR LAST CHARACTER
949      BPL      TYP0
950      MOV      TTYOUT,@STPB          ;SEND IT OUT
951      RTS      PC
952
910      005036 012700 000100
911      005042 013701 001310
912      005046 012702 000100
913      005052 012703 000100
914      005056 010321
915      005060 005302
916      005062 001375
917      005064 005300
918      005066 001405
919      005070 162703 000001
920      005074 012702 000100
921      005100 000766
922      005102 000207
923
935      005104 017737 174036 001262
936      005112 013777 001262 174032
937      005120 042737 177600 001262
938      005126 022737 000003 001262
939      005134 001002
940      005136 000137 001650
941      005142 000002
942
946      005144 012737 000015 001264
947      005152 004737 005164
948      005156 012737 000012 001264
949      005164 105777 173760
950      005170 100375
951      005172 013777 001264 173752
952      005200 000207

```

```

953      ;*****
954      ;THIS SUBROUTINE IS CALLED WITH THE ADDRESS OF A MESSAGE TERM WITH A 0 BYTE
955      ;*****
956
957 005202 012504      VTWRIT: MOV      (R5)+,R4      ;GET ADDRESS OF MESSAGE IN R4
958 005204 005777 174004 1$:      TST      @VTVCRG      ;WAIT FOR READY
959 005210 100375      BPL      1$
960 005212 105714      TSTB     (R4)      ,TERM ?
961 005214 001403      BEQ      2$      ;BR IF YES
962 005216 112477 173772      MOVB     (R4)+,@VTVCRG ;LOAD THE CHARACTER
963 005222 000770      BR       1$
964 005224 000205      2$:      RTS      R5      ;EXIT
965
966      ;*****
967      ;THIS ROUTINE CONVERTS OCTAL INTO ASCII FOR DISPLAY
968      ;*****
969 005226 012537 005340      AWRIT: MOV      (R5)+,10$      ;GET VALUE'S ADDRESS
970 005232 017737 000102 005340      MOV      @10$,10$      ;GET ACTUAL VALUE
971 005240 010046      MOV      R0,-(SP)      ;SAVE R0
972 005242 012700 005354      MOV      #NUMEND,R0      ;LOAD LAST ADDRESS OF OCTAL TYPEOUT
973 005246 012737 000006 005344      MOV      #6,12$      ;LOAD LOOP COUNTER
974 005254 000406      BR       2$
975 005256 006237 005340      1$:      ASR      10$      ;SHIFT DATA
976 005262 006237 005340      ASR      10$
977 005266 006237 005340      ASR      10$
978 005272 013737 005340 005342      2$:      MOV      10$,11$      ;COPY THE VALUE
979 005300 042737 177770 005342      BIC      #177770,11$      ;MASK OFF UNWANTED BITS
980 005306 062737 000060 005342      ADD      #60,11$      ;MAKE ASCII OCTAL
981 005314 113740 005342      MOVB     11$,-(R0)      ;SAVE THE CHAR.
982 005320 005337 005344      DEC      12$      ;FINISHED ?
983 005324 001354      BNE      1$
984 005326 012600      MOV      (SP)+,R0
985 005330 004537 005202      JSR      R5,VTWRIT      ;DISPAY THE OCTAL #
986 005334 005346      NUMBEG
987 005336 000205      RTS      R5      ;EXIT
988 005340 000000      10$:      0
989 005342 000000      11$:      0
990 005344 000005      12$:      5
991 005346      060      060      060      NUMBEG: .BYTE      60,60,60,60,60,60
992 005351      060      060
993 005354      000      NUMEND: .BYTE      0
994      005356      .EVEN

```

```

995
996
997
998 005356 010046
999 005360 013700 001234
1000 005364 013702 005604
1001 005370 010022
1002 005372 020227 005604
1003 005376 103402
1004 005400 012702 005504
1005 005404 010237 005604
1006 005410 012702 005504
1007 005414 004737 005446
1008 005420 110046
1009 005422 012702 005505
1010 005426 004737 005446
1011 005432 000300
1012 005434 152600
1013 005436 010037 001234
1014 005442 012600
1015 005444 000207
1016
1017 005446 005000
1018 005450 005005
1019 005452 152205
1020 005454 060500
1021 005456 005202
1022 005460 020227 005604
1023 005464 103771
1024 005466 006300
1025 005470 006300
1026 005472 006300
1027 005474 000300
1028 005476 042700 177400
1029 005502 000207
1030
1031 005504
1032 005604
1033 005604 005504

::*****
:THIS ROUTINE AVERAGES THE LAST 32. X-Y JOYSTICK VALUES
:*****
XYAVE: MOV R0,-(SP) ;SAVE R0
      MOV TEMP0,R0 ;GET X & Y
      MOV XYBUFP,R2 ;GET CURRENT BUFFER POINTER
      MOV R0,(R2)+ ;SAVE NEW X-Y VALUE
      CMP R2,#XYBUFE ;END OF BUFFER?
      BLO 1$ ;NO
      MOV #XYBUF,R2 ;YES, GO BACK TO BEGINING OF BUFFER
1$: MOV R2,XYBUFP ;SAVE NEW BUFFER POINTER
     MOV #XYBUF,R2 ;CALC AVE X
     JSR PC,10$
     MOVB R0,-(SP) ;SAVE IT
     MOV #XYBUF+1,R2 ;CALC AVE Y
     JSR PC,10$ ;GO DO IT
     SWAB R0 ;GET X-Y IN R0
     BISB (SP)+,R0 ;GET SAVED X
     MOV R0,TEMP0 ;PUT IN TEMP0
     MOV (SP)+,R0 ;RESTORE R0
     RTS PC ;EXIT WITH AVE X-Y IN TEMP0

10$: CLR R0 ;ZERO SUM
11$: CLR R5 ;DO A MOVB TO A REG (UNSIGNED)
     BISB (R2)+,R5
     ADD R5,R0 ;ADD IT IN
     INC R2 ;SKIP OTHER VALUE
     CMP R2,#XYBUFE ;END OF BUFFER?
     BLO 11$ ;NO
     ASL R0 ;DIVIDE BY 32.
     ASL R0
     ASL R0
     SWAB R0
     BIC #177400,R0 ;CLR HI BYTE
     RTS PC

XYBUF:
XYBUFE=.
XYBUFP: XYBUF

```

```

1034      ::*****
1035      ::SBTTL  TTY INPUT ROUTINE
1036
1037      ::*****
1038      ::ENABL  LSB
1039
1040      ::*****
1041      ::*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
1042      ::*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
1043      ::*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
1044      ::*WHEN OPERATING IN TTY FLAG MODE.
1045      005606 022737 000176 001140 $CKSWR: CMP      #SWREG,SWR      ::IS THE SOFT-SWR SELECTED?
1046      005614 001074          BNE      15$      ::BRANCH IF NO
1047      005616 105777 173322          TSTB     @STKS      ::CHAR THERE?
1048      005622 100071          BPL      15$      ::IF NO, DON'T WAIT AROUND
1049      005624 117746 173316          MOVB     @STKB,-(SP)  ::SAVE THE CHAR
1050      005630 042716 177600          BIC      #^C177,(SP)  ::STRIP-OFF THE ASCII
1051      005634 022726 000007          CMP      #7,(SP)+    ::IS IT A CONTROL G?
1052      005640 001062          BNE      15$      ::NO, RETURN TO USER
1053      005642 123727 001134 000001          CMPSB    $AUTOB,#1  ::ARE WE RUNNING IN AUTO-MODE?
1054      005650 001456          BEQ      15$      ::BRANCH IF YES
1055
1056      005652 104401 006460          $GTSWR: TYPE     , $CNTLG      ::ECHO THE CONTROL-G (^G)
1057      005656 104401 006465          TYPE     , $MSWR      ::TYPE CURRENT CONTENTS
1058      005662 013746 000176          MOV      SWREG,-(SP)  ::SAVE SWREG FOR TYPEOUT
1059      005666 104402          TYPOC      ::GO TYPE--OCTAL ASCII(ALL DIGITS)
1060      005670 104401 006476          TYPE     , $MNEW      ::PROMPT FOR NEW SWR
1061      005674 005046          19$:  CLR      -(SP)      ::CLEAR COUNTER
1062      005676 005046          CLR      -(SP)      ::THE NEW SWR
1063      005700 105777 173240          7$:  TSTB     @STKS      ::CHAR THERE?
1064      005704 100375          BPL      7$      ::IF NOT TRY AGAIN
1065
1066      005706 117746 173234          MOVB     @STKB,-(SP)  ::PICK UP CHAR
1067      005712 042716 177600          BIC      #^C177,(SP)  ::MAKE IT 7-BIT ASCII
1068
1069
1070
1071      005716 021627 000025          9$:  CMP      (SP),#25      ::IS IT A CONTROL-U?
1072      005722 001005          BNE      10$      ::BRANCH IF NOT
1073      005724 104401 006453          TYPE     , $CNTLU      ::YES, ECHO CONTROL-U (^U)
1074      005730 062706 000006          20$:  ADD      #6,SP      ::IGNORE PREVIOUS INPUT
1075      005734 000757          BR      19$      ::LET'S TRY IT AGAIN
1076
1077
1078      005736 021627 000015          10$:  CMP      (SP),#15      ::IS IT A <CR>?
1079      005742 001022          BNE      16$      ::BRANCH IF NO
1080      005744 005766 000004          TST      4(SP)      ::YES, IS IT THE FIRST CHAR?
1081      005750 001403          BEQ      11$      ::BRANCH IF YES
1082      005752 016677 000002 173160          MOV      2(SP),@SWR  ::SAVE NEW SWR
1083      005760 062706 000006          11$:  ADD      #6,SP      ::CLEAR UP STACK
1084      005764 104401 001161          14$:  TYPE     , $CRLF      ::ECHO <CR> AND <LF>
1085      005770 123727 001135 000001          CMPSB    $INTAG,#1  ::RE-ENABLE TTY KBD INTERRUPTS?
1086      005776 001003          BNE      15$      ::BRANCH IF NOT
1087      006000 012777 000100 173136          MOV      #100,@STKS  ::RE-ENABLE TTY KBD INTERRUPTS

```

```
1088 006006 000002          15$: RTI          ;;RETURN
1089 006010 004737 006660    16$: JSR      PC,$TYPEC  ;;ECHO CHAR
1090 006014 021627 000060    CMP      (SP),#60  ;;CHAR < 0?
1091 006020 002420          BLT      18$      ;;BRANCH IF YES
1092 006022 021627 000067    CMP      (SP),#67  ;;CHAR > 7?
1093 006026 003015          BGT      18$      ;;BRANCH IF YES
1094 006030 042726 000060    BIC      #60,(SP)+  ;;STRIP-OFF ASCII
1095 006034 005766 000002    TST      2(SP)    ;;IS THIS THE FIRST CHAR
1096 006040 001403          BEQ      17$      ;;BRANCH IF YES
1097 006042 006316          ASL      (SP)    ;;NO, SHIFT PRESENT
1098 006044 006316          ASL      (SP)    ;;CHAR OVER TO MAKE
1099 006046 006316          ASL      (SP)    ;;ROOM FOR NEW ONE.
1100 006050 005266 000002    17$: INC      2(SP)    ;;KEEP COUNT OF CHAR
1101 006054 056616 177776    BIS      -2(SP),(SP) ;;SET IN NEW CHAR
1102 006060 000707          BR       7$      ;;GET THE NEXT ONE
1103 006062 104401 001160    18$: TYPE    ,$QUES  ;;TYPE ?<CR><LF>
1104 006066 000720          BR       20$      ;;SIMULATE CONTROL-U
1105          .DSABL  LSB
1106
1107
1108          ;*****
1109          ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
1110          ;*CALL:
1111          ;*      RDCHR          ;;INPUT A SINGLE CHARACTER FROM THE TTY
1112          ;*      RETURN HERE   ;;CHARACTER IS ON THE STACK
1113          ;*                  ;;WITH PARITY BIT STRIPPED OFF
1114          ;*
1115          ;*
1116 006070 011646          $RDCHR: MOV      (SP),-(SP) ;;PUSH DOWN THE PC
1117 006072 016666 000004 000002 MOV      4(SP),2(SP) ;;SAVE THE PS
1118 006100 105777 173040    1$: TSTB     @TKS     ;;WAIT FOR
1119 006104 100375          BPL      1$      ;;A CHARACTER
1120 006106 117766 173034 000004 MOVB     @TKB,4(SP) ;;READ THE TTY
1121 006114 042766 177600 000004 BIC      #^C<177>,4(SP) ;;GET RID OF JUNK IF ANY
1122 006122 026627 000004 000023 CMP      4(SP),#23  ;;IS IT A CONTROL-S?
1123 006130 001013          BNE      3$      ;;BRANCH IF NO
1124 006132 105777 173006    2$: TSTB     @TKS     ;;WAIT FOR A CHARACTER
1125 006136 100375          BPL      2$      ;;LOOP UNTIL ITS THERE
1126 006140 117746 173002 MOVB     @TKB,-(SP) ;;GET CHARACTER
1127 006144 042716 177600 BIC      #^C177,(SP) ;;MAKE IT 7-BIT ASCII
1128 006150 022627 000021 CMP      (SP)+,#21  ;;IS IT A CONTROL-Q?
1129 006154 001366          BNE      2$      ;;IF NOT DISCARD IT
1130 006156 000750          BR       1$      ;;YES, RESUME
1131 006160 026627 000004 000140 3$: CMP      4(SP),#140  ;;IS IT UPPER CASE?
1132 006166 002407          BLT      4$      ;;BRANCH IF YES
1133 006170 026627 000004 000175 CMP      4(SP),#175  ;;IS IT A SPECIAL CHAR?
1134 006176 003003          BGT      4$      ;;BRANCH IF YES
1135 006200 042766 000040 000004 BIC      #40,4(SP)  ;;MAKE IT UPPER CASE
1136 006206 000002          4$: RTI          ;;GO BACK TO USER
1137          ;*****
1138          ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
1139          ;*CALL:
1140          ;*      RDLIN          ;;INPUT A STRING FROM THE TTY
1141          ;*      RETURN HERE   ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
```

1142				;	*		:: TERMINATOR WILL BE A BYTE OF ALL 0'S
1143							
1144	006210	010346		\$RDLIN:	MOV	R3,-(SP)	::SAVE R3
1145	006212	005046			CLR	-(SP)	::CLEAR THE RUBOUT KEY
1146	006214	012703	006444	1\$:	MOV	#STTYIN,R3	::GET ADDRESS
1147	006220	022703	006453	2\$:	CMP	#STTYIN+7,R3	::BUFFER FULL?
1148	006224	101456			BLOS	4\$	::BR IF YES
1149	006226	104407			RDCHR		::GO READ ONE CHARACTER FROM THE TTY
1150	006230	112613			MOVB	(SP)+,(R3)	::GET CHARACTER
1151	006232	122713	000177	10\$:	CMPS	#177,(R3)	::IS IT A RUBOUT
1152	006236	001022			BNF	5\$	::BR IF NO
1153	006240	005716			TST	(SP)	::IS THIS THE FIRST RUBOUT?
1154	006242	001007			BNE	6\$	::BR IF NO
1155	006244	112737	000134 006442		MOVB	#'\,9\$	::TYPE A BACK SLASH
1156	006252	104401	006442		TYPE	,9\$	
1157	006256	012716	177777		MOV	#-1,(SP)	::SET THE RUBOUT KEY
1158	006262	005303		6\$:	DEC	R3	::BACKUP BY ONE
1159	006264	020327	006444		CMP	R3,#STTYIN	::STACK EMPTY?
1160	006270	103434			BLO	4\$	::BR IF YES
1161	006272	111337	006442		MOVB	(R3),9\$	::SETUP TO TYPEOUT THE DELETED CHAR.
1162	006276	104401	006442		TYPE	,9\$	::GO TYPE
1163	006302	000746			BR	2\$	::GO READ ANOTHER CHAR.
1164	006304	005716		5\$:	TST	(SP)	::RUBOUT KEY SET?
1165	006306	001406			BEQ	7\$	::BR IF NO
1166	006310	112737	000134 006442		MOVB	#'\,9\$	::TYPE A BACK SLASH
1167	006316	104401	006442		TYPE	,9\$	
1168	006322	005016			CLR	(SP)	::CLEAR THE RUBOUT KEY
1169	006324	122713	000025	7\$:	CMPS	#25,(R3)	::IS CHARACTER A CTRL U?
1170	006330	001003			BNE	8\$	::BR IF NO
1171	006332	104401	006453		TYPE	,\$CNTLU	::TYPE A CONTROL 'U'
1172	006336	000726			BR	1\$	::GO START OVER
1173	006340	122713	000022	8\$:	CMPS	#22,(R3)	::IS CHARACTER A '^R'?
1174	006344	001011			BNE	3\$	::BRANCH IF NO
1175	006346	105013			CLRB	(R3)	::CLEAR THE CHARACTER
1176	006350	104401	001161		TYPE	,\$CRLF	::TYPE A 'CR' & 'LF'
1177	006354	104401	006444		TYPE	,\$TTYIN	::TYPE THE INPUT STRING
1178	006360	000717			BR	2\$	::GO PICKUP ANOTHER CHARACTER
1179	006362	104401	001160	4\$:	TYPE	,\$QUES	::TYPE A '?'
1180	006366	000712			BR	1\$	::CLEAR THE BUFFER AND LOOP
1181	006370	111337	006442	3\$:	MOVB	(R3),9\$	::ECHO THE CHARACTER
1182	006374	104401	006442		TYPE	,9\$	
1183	006400	122723	000015		CMPS	#15,(R3)+	::CHECK FOR RETURN
1184	006404	001305			BNE	2\$	::LOOP IF NOT RETURN
1185	006406	105063	177777		CLRB	-1(R3)	::CLEAR RETURN (THE 15)
1186	006412	104401	001162		TYPE	,\$LF	::TYPE A LINE FEED
1187	006416	005726			TST	(SP)+	::CLEAN RUBOUT KEY FROM THE STACK
1188	006420	012603			MOV	(SP)+,R3	::RESTORE R3
1189	006422	011646			MOV	(SP),-(SP)	::ADJUST THE STACK AND PUT ADDRESS OF THE
1190	006424	016666	000004 000002		MOV	4(SP),2(SP)	::FIRST ASCII CHARACTER ON IT
1191	006432	012766	006444 000004		MOV	#STTYIN,4(SP)	
1192	006440	000002			RTI		::RETURN
1193	006442	000		9\$:	.BYTE	0	::STORAGE FOR ASCII CHAR. TO TYPE
1194	006443	000			.BYTE	0	::TERMINATOR
1195	006444	000007		\$TTYIN:	.BLKB	7	::RESERVE 7 BYTES FOR TTY INPUT

```

1196 006453 136 006525 000012 $CNTLU: .ASCIZ / ^U/<15><12> ::CONTROL 'U'
1197 006460 043536 005015 000 $CNTLG: .ASCIZ / ^G/<15><12> ::CONTROL 'G'
1198 006465 015 051412 051127 $MSWR: .ASCIZ <15><12>/SWR - /
1199 006472 036440 000040
1200 006476 020040 042516 020127 $MNEW: .ASCIZ / NEW = /
1201 006504 020075 000
1202 006510
1203 .EVEN
1204 ::*****
1205 .SBTTL TYPE ROUTINE
1206 ::*****
1207 *ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
1208 *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
1209 *NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
1210 *NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
1211 *NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
1212 *
1213 *CALL:
1214 *1) USING A TRAP INSTRUCTION
1215 * TYPE ,MESADR ::MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
1216 *OR
1217 * TYPE
1218 * MESADR
1219 *
1220
1221 006510 105737 001157 $TYPE: TSTB $TPFLG ::IS THERE A TERMINAL?
1222 006514 100002 BPL 1$ ::BR IF YES
1223 006516 000000 HALT ::HALT HERE IF NO TERMINAL
1224 006520 000407 BR 3$ ::LEAVE
1225 006522 010046 1$: MOV R0,-(SP) ::SAVE R0
1226 006524 017600 000002 MOV @2(SP),R0 ::GET ADDRESS OF ASCIZ STRING
1227 006530 112046 2$: MOVB (R0)+,-(SP) ::PUSH CHARACTER TO BE TYPED ONTO STACK
1228 006532 001005 BNE 4$ ::BR IF IT ISN'T THE TERMINATOR
1229 006534 005726 TST (SP)+ ::IF TERMINATOR POP IT OFF THE STACK
1230 006536 012600 60$: MCV (SP)+,R0 ::RESTORE R0
1231 006540 062716 000002 3$: ADD #2,(SP) ::ADJUST RETURN PC
1232 006544 000002 RTI ::RETURN
1233 006546 122716 000011 4$: CMPB #HT,(SP) ::BRANCH IF <HT>
1234 006552 001430 BEQ 8$
1235 006554 122716 000200 CMPB #CRLF,(SP) ::BRANCH IF NOT <CRLF>
1236 006560 001006 BNE 5$
1237 006562 005726 TST (SP)+ ::POP <CR><LF> EQUIV
1238 006564 104401 TYPE ::TYPE A CR AND LF
1239 006566 001161 $CRLF
1240 006570 105037 006724 CLRB $CHARCNT ::CLEAR CHARACTER COUNT
1241 006574 000755 BR 2$ ::GET NEXT CHARACTER
1242 006576 004737 006660 5$: JSR PC,$TYPEC ::GO TYPE THIS CHARACTER
1243 006602 123726 001156 6$: CMPB $FILLC,(SP)+ ::IS IT TIME FOR FILLER CHARS.?
1244 006606 001350 BNE 2$ ::IF NO GO GET NEXT CHAR.
1245 006610 013746 001154 MOV $NULL,-(SP) ::GET # OF FILLER CHARS. NEEDED
1246 AND THE NULL CHAR.
1247 006614 105366 000001 7$: DECB 1(SP) ::DOES A NULL NEED TO BE TYPED?
1248 006620 002770 BLT 6$ ::BR IF NO--GO POP THE NULL OFF OF STACK
1249 006622 004737 006660 JSR PC,$TYPEC ::GO TYPE A NULL

```

```
1250 006626 105337 006724          DECB  $CHARCNT      ;;DO NOT COUNT AS A COUNT
1251 006632 000770                  BR      7$          ;;LOOP
1252
1253          ;HORIZONTAL TAB PROCESSOR
1254
1255 006634 112716 000040      8$:  MOVB  #' (SP)          ;;REPLACE TAB WITH SPACE
1256 006640 004737 006660      9$:  JSR   PC,$TYPEC      ;;TYPE A SPACE
1257 006644 132737 000007 006724  BITB  #7,$CHARCNT      ;;BRANCH IF NOT AT
1258 006652 001372          BNE   9$          ;;TAB STOP
1259 006654 005726          TST   (SP)+          ;;POP SPACE OFF STACK
1260 006656 000724          BR     2$          ;;GET NEXT CHARACTER
1261 006660 105777 172264      $TYPEC: TSTB  @ $TPS      ;;WAIT UNTIL PRINTER IS READY
1262 006664 100375          BPL   $TYPEC
1263 006666 116677 000002 172256  MOVB  2(SP),@ $TPB      ;;LOAD CHAR TO BE TYPED INTO DATA REG.
1264 006674 122766 000015 000002  CMPB  #CR,2(SP)      ;;IS CHARACTER A CARRIAGE RETURN?
1265 006702 001003          BNE   1$          ;;BRANCH IF NO
1266 006704 105037 006724          CLRB  $CHARCNT      ;;YES--CLEAR CHARACTER COUNT
1267 006710 000406          BR     $TYPEX      ;;EXIT
1268 006712 122766 000012 000002  1$:  CMPB  #LF,2(SP)      ;;IS CHARACTER A LINE FEED?
1269 006720 001402          BEQ   $TYPEX      ;;BRANCH IF YES
1270 006722 105227          INCB  (PC)+          ;;COUNT THE CHARACTER
1271 006724 000000      $CHARCNT: .WORD  0          ;;CHARACTER COUNT STORAGE
1272 006726 000207      $TYPEX: RTS   PC
1273
1274          ;*****
1275          .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
1276
1277          ;*****
1278          ;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
1279          ;OCTAL (ASCII) NUMBER AND TYPE IT.
1280          ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
1281          ;*CALL:
1282          ;*      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
1283          ;*      TYPOS          ;;CALL FOR TYPEOUT
1284          ;*      .BYTE   N          ;;N-1 TO 6 FOR NUMBER OF DIGITS TO TYPE
1285          ;*      .BYTE   M          ;;M=1 OR 0
1286          ;*                      ;;1=TYPE LEADING ZEROS
1287          ;*                      ;;0=SUPPRESS LEADING ZEROS
1288          ;*
1289          ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
1290          ;*$TYPOS OR $TYPOC
1291          ;*CALL:
1292          ;*      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
1293          ;*      TYPON          ;;CALL FOR TYPEOUT
1294          ;*
1295          ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
1296          ;*CALL:
1297          ;*      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
1298          ;*      TYPOC          ;;CALL FOR TYPEOUT
1299
1300 006730 017646 000000 007153  $TYPOS: MOV  @ (SP),-(SP)      ;;PICKUP THE MODE
1301 006734 116637 000001          MOVB  1(SP),$OFILL      ;;LOAD ZERO FILL SWITCH
1302 006742 112637 007155          MOVB  (SP)+,$OMODE+1      ;;NUMBER OF DIGITS TO TYPE
1303 006746 062716 000002          ADD   #2,(SP)          ;;ADJUST RETURN ADDRESS
```

```

1304 006752 000406 BR $TYPON
1305 006754 112737 000001 007153 $TYPON: MOVB #1,$OFILL ;;SET THE ZERO FILL SWITCH
1306 006762 112737 000006 007155 MOVB #6,$OMODE+1 ;;SET FOR SIX(6) DIGITS
1307 006770 112737 000005 007152 $TYPON: MOVB #5,$OCNT ;;SET THE ITERATION COUNT
1308 006776 010346 MOV R3,-(SP) ;;SAVE R3
1309 007000 010446 MOV R4,-(SP) ;;SAVE R4
1310 007002 010546 MOV R5,-(SP) ;;SAVE R5
1311 007004 113704 007155 MOVB $OMODE+1,R4 ;;GET THE NUMBER OF DIGITS TO TYPE
1312 007010 005404 NEG R4
1313 007012 062704 000006 ADD #6,R4 ;;SUBTRACT IT FOR MAX. ALLOWED
1314 007016 110437 007154 MOVB R4,$OMODE ;;SAVE IT FOR USE
1315 007022 113704 007153 MOVB $OFILL,R4 ;;GET THE ZERO FILL SWITCH
1316 007026 016605 000012 MOV 12(SP),R5 ;;PICKUP THE INPUT NUMBER
1317 007032 005003 CLR R3 ;;CLEAR THE OUTPUT WORD
1318 007034 006105 1$: ROL R5 ;;ROTATE MSB INTO 'C'
1319 007036 000404 BR 3$ ;;GO DO MSB
1320 007040 006105 2$: ROL R5 ;;FORM THIS DIGIT
1321 007042 006105 ROL R5
1322 007044 006105 ROL R5
1323 007046 010503 MOV R5,R3
1324 007050 006103 3$: ROL R3 ;;GET LSB OF THIS DIGIT
1325 007052 105337 007154 DECB $OMODE ;;TYPE THIS DIGIT?
1326 007056 100016 BPL 7$ ;;BR IF NO
1327 007060 042703 177770 BIC #177770,R3 ;;GET RID OF JUNK
1328 007064 001002 BNE 4$ ;;TEST FOR 0
1329 007066 005704 TST R4 ;;SUPPRESS THIS 0?
1330 007070 001403 BEQ 5$ ;;BR IF YES
1331 007072 005204 4$: INC R4 ;;DON'T SUPPRESS ANYMORE 0'S
1332 007074 052703 000060 BIS #'0,R3 ;;MAKE THIS DIGIT ASCII
1333 007100 052703 000040 5$: BIS #' ,R3 ;;MAKE ASCII IF NOT ALREADY
1334 007104 110337 007150 MOVB R3,8$ ;;SAVE FOR TYPING
1335 007110 104401 007150 TYPE ,8$ ;;GO TYPE THIS DIGIT
1336 007114 105337 007152 7$: DECB $OCNT ;;COUNT BY 1
1337 007120 003347 BGT 2$ ;;BR IF MORE TO DO
1338 007122 002402 BLT 6$ ;;BR IF DONE
1339 007124 005204 INC R4 ;;INSURE LAST DIGIT ISN'T A BLANK
1340 007126 000744 BR 2$ ;;GO DO THE LAST DIGIT
1341 007130 012605 6$: MOV (SP)+,R5 ;;RESTORE R5
1342 007132 012604 MOV (SP)+,R4 ;;RESTORE R4
1343 007134 012603 MOV (SP)+,R3 ;;RESTORE R3
1344 007136 016666 000002 000004 MOV 2(SP),4(SP) ;;SET THE STACK FOR RETURNING
1345 007144 012616 MOV (SP)+,(SP)
1346 007146 000002 RTI ;;RETURN
1347 007150 000 8$: .BYTE 0 ;;STORAGE FOR ASCII DIGIT
1348 007151 000 .BYTE 0 ;;TERMINATOR FOR TYPE ROUTINE
1349 007152 000 $OCNT: .BYTE 0 ;;OCTAL DIGIT COUNTER
1350 007153 000 $OFILL: .BYTE 0 ;;ZERO FILL SWITCH
1351 007154 000000 $OMODE: .WORD 0 ;;NUMBER OF DIGITS TO TYPE
1352 *****
1353 .SBTTL READ AN OCTAL NUMBER FROM THE TTY
1354 *****
1355 *THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
1356 *CHANGE IT TO BINARY.
1357

```

```

1358      ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
1359      ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A '?' WILL BE TYPED
1360      ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
1361      ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
1362      ;*CALL:
1363      ;*      RDOCT          ;;READ AN OCTAL NUMBER
1364      ;*      RETURN HERE   ;;LOW ORDER BITS ARE ON TOP OF THE STACK
1365      ;*                    ;;HIGH ORDER BITS ARE IN $HIOCT
1366
1367 007156 011646          $RDOCT: MOV      (SP),-(SP)      ;;PROVIDE SPACE FOR THE
1368 007160 016666 000004 000002      MOV      4(SP),2(SP)      ;;INPUT NUMBER
1369 007166 010046          MOV      R0,-(SP)      ;;PUSH R0 ON STACK
1370 007170 010146          MOV      R1,-(SP)      ;;PUSH R1 ON STACK
1371 007172 010246          MOV      R2,-(SP)      ;;PUSH R2 ON STACK
1372 007174 104410      1$:  RDLIN          ;;READ AN ASCII LINE
1373 007176 012600          MOV      (SP)+,R0      ;;GET ADDRESS OF 1ST CHARACTER
1374 007200 010037 007304          MOV      R0,5$      ;;AND SAVE IT
1375 007204 005001          CLR      R1      ;;CLEAR DATA WORD
1376 007206 005002          CLR      R2
1377 007210 112046      2$:  MOVB      (R0)+,-(SP)      ;;PICKUP THIS CHARACTER
1378 007212 001420          BEQ      3$      ;;IF ZERO GET OUT
1379 007214 122716 000060          CMPB      #'0,(SP)      ;;MAKE SURE THIS CHARACTER
1380 007220 003026          BGT      4$      ;;IS AN OCTAL DIGIT
1381 007222 122716 000067          CMPB      #'7,(SP)
1382 007226 002423          BLT      4$
1383 007230 006301          ASL      R1      ;;*2
1384 007232 006102          ROL      R2
1385 007234 006301          ASL      R1      ;;*4
1386 007236 006102          ROL      R2
1387 007240 006301          ASL      R1      ;;*8
1388 007242 006102          ROL      R2
1389 007244 042716 177770          BIC      #'C7,(SP)      ;;STRIP THE ASCII JUNK
1390 007250 062601          ADD      (SP)+,R1      ;;ADD IN THIS DIGIT
1391 007252 000756          BR      2$      ;;LOOP
1392 007254 005726      3$:  TST      (SP)+      ;;CLEAN TERMINATOR FROM STACK
1393 007256 010166 000012          MOV      R1,12(SP)      ;;SAVE THE RESULT
1394 007262 010237 007314          MOV      R2,$HIOCT
1395 007266 012602          MOV      (SP)+,R2      ;;POP STACK INTO R2
1396 007270 012601          MOV      (SP)+,R1      ;;POP STACK INTO R1
1397 007272 012600          MOV      (SP)+,R0      ;;POP STACK INTO R0
1398 007274 000002          RTI          ;;RETURN
1399 007276 005726      4$:  TST      (SP)+      ;;CLEAN PARTIAL FROM STACK
1400 007300 105010          CLRB      (R0)      ;;SET A TERMINATOR
1401 007302 104401          TYPE          ;;TYPE UP THRU THE BAD CHAR.
1402 007304 000000      5$:  .WORD      0
1403 007306 104401 001160          TYPE      $QUES      ;; '?' 'CR' & 'LF'
1404 007312 000730          BR      1$      ;;TRY AGAIN
1405 007314 000000      $HIOCT: .WORD      0      ;;HIGH ORDER BITS GO HERE

```

```

1406      ;:*****
1407      .SBTTL TRAP DECODER
1408
1409      ;:*****
1410      ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
1411      ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
1412      ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
1413      ;*GO TO THAT ROUTINE.
1414
1415 007316 010046      $TRAP: MOV      R0,-(SP)      ;;SAVE R0
1416 007320 016600 000002      MOV      2(SP),R0      ;;GET TRAP ADDRESS
1417 007324 005740      TST      -(R0)      ;;BACKUP BY 2
1418 007326 111000      MOVB     (R0),R0      ;;GET RIGHT BYTE OF TRAP
1419 007330 006300      ASL      R0      ;;POSITION FOR INDEXING
1420 007332 016000 007352      MOV      $TRPAD(R0),R0      ;;INDEX TO TABLE
1421 007336 000200      RTS      R0      ;;GO TO ROUTINE
1422
1423
1424      ;:THIS IS USE TO HANDLE THE 'GETPRI' MACRO
1425
1426 007340 011646      $TRAP2: MOV      (SP),-(SP)      ;;MOVE THE PC DOWN
1427 007342 016666 000004 000002      MOV      4(SP),2(SP)      ;;MOVE THE PSW DOWN
1428 007350 000002      RTI      ;;RESTORE THE PSW
1429
1430      .SBTTL TRAP TABLE
1431
1432      ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
1433      ;*BY THE 'TRAP' INSTRUCTION.
1434
1435      :      ROUTINE
1436      :      -----
1437 007352 007340      $TRPAD: .WORD      $TRAP2
1438 007354 006510      $TYPE      ;;CALL=TYPE      TRAP+1(104401) TTY TYPEOUT ROUTINE
1439 007356 006754      $TYPOC     ;;CALL=TYPOC      TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
1440 007360 006730      $TYPOS     ;;CALL=TYPOS      TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
1441 007362 006770      $TYPON     ;;CALL=TYPON      TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
1442
1443 007364 005656      $GTSWR     ;;CALL=GTSWR      TRAP+5(104405) GET SOFT-SWR SETTING
1444
1445 007366 005606      $CKSWR     ;;CALL=CKSWR      TRAP+6(104406) TEST FOR CHANGE IN SOFT-SWR
1446 007370 006070      $RDCHR     ;;CALL=RDCHR      TRAP+7(104407) TTY TYPEIN CHARACTER ROUTINE
1447 007372 006210      $RDLIN     ;;CALL=RDLIN      TRAP+10(104410) TTY TYPEIN STRING ROUTINE
1448 007374 007156      $RDOCT     ;;CALL=RDOCT      TRAP+11(104411) READ AN OCTAL NUMBER FROM TTY

```

```
1449 ;:*****
1450 ;SBTTL POWER DOWN AND UP ROUTINES
1451 ;:*****
1452 ;POWER DOWN ROUTINE
1453 $PWRDN: MOV $ILLUP,@#PWRVEC ;;SET FOR FAST UP
1454 007376 012737 007542 000024 MOV #340,@#PWRVEC+2 ;;PRIO:7
1455 007404 012737 000340 000026 MOV R0,-(SP) ;;PUSH R0 ON STACK
1456 007412 010046 MOV R1,-(SP) ;;PUSH R1 ON STACK
1457 007414 010146 MOV R2,-(SP) ;;PUSH R2 ON STACK
1458 007416 010246 MOV R3,-(SP) ;;PUSH R3 ON STACK
1459 007420 010346 MOV R4,-(SP) ;;PUSH R4 ON STACK
1460 007422 010446 MOV R5,-(SP) ;;PUSH R5 ON STACK
1461 007424 010546 MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
1462 007426 017746 171506 MOV SP,$SAVR6 ;;SAVE SP
1463 007432 010637 007546 MOV #PWRUP,@#PWRVEC ;;SET UP VECTOR
1464 007436 012737 007450 000024 HALT
1465 007444 000000 BR -2 ;;HANG UP
1466 007446 000776
1467
1468 ;:*****
1469 ;POWER UP ROUTINE
1470 007450 012737 007542 000024 $PWRUP: MOV $ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
1471 007456 013706 007546 MOV $SAVR6,SP ;;GET SP
1472 007462 005037 007546 CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
1473 007466 005237 007546 1$: INC $SAVR6 ;;WAIT FOR THE INC
1474 007472 001375 BNE 1$ ;;OF WORD
1475 007474 012677 171440 MOV (SP)+,@SWR ;;POP STACK INTO @SWR
1476 007500 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
1477 007502 012604 MOV (SP)+,R4 ;;POP STACK INTO R4
1478 007504 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
1479 007506 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
1480 007510 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
1481 007512 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
1482 007514 012737 007376 000024 MOV #PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
1483 007522 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;PRIO:7
1484 007530 104401 TYPE ;;REPORT THE POWER FAILURE
1485 007532 007742 $PWRMG: .WORD PWRMSG ;;POWER FAIL MESSAGE POINTER
1486 007534 012716 MOV (PC)+,(SP) ;;RESTART AT START1
1487 007536 001554 $PWRAD: .WORD START1 ;;RESTART ADDRESS
1488 007540 000002 RTI
1489 007542 000000 $ILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
1490 007544 000776 BR -2 ;; BEFORE THE POWER DOWN WAS COMPLETE
1491 007546 000000 $SAVR6: 0 ;;PUT THE SP HERE
```

```

1492 .SBTTL DISPLAY MESSAGES
1493 ;THIS IS THE MESSAGE FOR THE BOTTOM OF THE SCOPE
1494 007550 005015 047514 042527 WD001: .ASCIZ <15><12>/LOWER THRESHOLD /
1495 007556 020122 044124 042522
1496 007564 044123 046117 020104
1497 007572 000
1498 007573 040 020040 020040 WD002: .ASCIZ / UPPER THRESHOLD /
1499 007600 050125 042520 020122
1500 007606 044124 042522 044123
1501 007614 046117 020104 000
1502 007621 015 055012 041440 WD003: .ASCIZ <15><12>/Z COUNT=/
1503 007626 052517 052116 000075
1504 007634 046440 052101 044522 WD004: .ASCIZ / MATRIX COUNT=/
1505 007642 020130 047503 047125
1506 007650 036524 000
1507 007653 040 055040 047517 WD005: .ASCIZ / ZOOM/
1508 007660 000115
1509 007662 041040 043455 046501 WD006: .ASCIZ / B-GAMMA/
1510 007670 040515 000
1511 007673 015 041412 046501 WD007: .ASCII <15><12>/CAMERA # /
1512 007700 051105 020101 020043
1513 007706 060 CAMOUT: .BYTE 60
1514 007707 040 020040 020040 .ASCII / PB IS /
1515 007714 041120 044440 020123
1516 007722 050125 020040 020040 PBUP: .ASCII /UP JB IS /
1517 007730 045040 020102 051511
1518 007736 040
1519 007737 125 000120 JBUP: .ASCIZ /UP/
1520
1521 .SBTTL ASCII MESSAGES
1522
1523 007742 005015 051012 051505 PWRMSG: .ASCIZ <15><12><12>/RESTARTED AFTER POWER FAILURE/
1524 007750 040524 052122 042105
1525 007756 040440 052106 051105
1526 007764 050040 053517 051105
1527 007772 043040 044501 052514
1528 010000 042522 000
1529 010003 015 005012 055103 MSG1: .ASCIZ <15><12><12>/CZNCDA NCV-11 EXERCISER/
1530 010010 041516 040504 020040
1531 010016 047040 053103 030455
1532 010024 020061 054105 051105
1533 010032 044503 042523 000122
1534 010040 005015 047105 042524 MSG2: .ASCIZ <15><12>/ENTER THRESHOLD VALUE IN OCTAL - THEN RETURN =/
1535 010046 020122 044124 042522
1536 010054 044123 046117 020104
1537 010062 040526 052514 020105
1538 010070 047111 047440 052103
1539 010076 046101 026440 052040
1540 010104 042510 020116 042522
1541 010112 052524 047122 036440
1542 010120 000
1543 010121 015 041012 051525 MSG3: .ASCIZ <15><12>/BUS TIMEOUT ERROR - VSV01 DISPLAY/
1544 010126 052040 046511 047505
1545 010134 052125 042440 051122

```

CZNCDA NCV11 EXERCISER MACY11 27(654) 8-AUG-78 15:16 PAGE 34
 CZNCDA.P11 ASCII MESSAGES

SEQ 0042

1546	010142	051117	026440	053040	
1547	010150	053123	030460	042040	
1548	010156	051511	046120	054501	
1549	010164	000			
1550	010165	015	042412	052116	MSG4: .ASCIIZ <15><12>/ENTER CONSOLE BUS ADDRESS - THEN RETURN =/
1551	010172	051105	041440	047117	
1552	010200	047523	042514	041040	
1553	010206	051525	040440	042104	
1554	010214	042522	051523	026440	
1555	010222	052040	042510	020116	
1556	010230	042522	052524	047122	
1557	010236	036440	000		
1558	010241	015	041012	051525	MSG5: .ASCIIZ <15><12>/BUS TIMEOUT ERROR - NCV11/
1559	010246	052040	046511	047505	
1560	010254	052125	042440	051122	
1561	010262	051117	026440	047040	
1562	010270	053103	030461	000	
1563	010275	015	052012	050131	MSG6: .ASCIIZ <15><12>/TYPE 'H' FOR HELP INFO ENTER KEYBOARD COMMAND(S)/
1564	010302	020105	044042	020042	
1565	010310	047506	020122	042510	
1566	010316	050114	044440	043116	
1567	010324	020117	042440	052116	
1568	010332	051105	045440	054505	
1569	010340	047502	051101	020104	
1570	010346	047503	046515	047101	
1571	010354	024104	024523	000	
1572	010361	015	042412	052116	MSG7: .ASCIIZ <15><12>/ENTER CAMERA NUMBER 0-3 ? /
1573	010366	051105	041440	046501	
1574	010374	051105	020101	052516	
1575	010402	041115	051105	030040	
1576	010410	031455	037440	000040	
1577	010416	026440	000040		DASH: .ASCIIZ / - /
1578	010422	005015	042115	030455	HELPO: .ASCII <15><12>/MD-11-DVNCB-A NCV11 EXERCISER/<15><12><12>
1579	010430	026461	053104	041516	
1580	010436	026502	020101	041516	
1581	010444	030526	020061	054105	
1582	010452	051105	044503	042523	
1583	010460	006522	005012		
1584	010464	020104	020040	020040	.ASCII /D DISPLAY DATA/<15><12>
1585	010472	042040	051511	046120	
1586	010500	054501	042040	052101	
1587	010506	006501	012		
1588	010511	105	020040	020040	.ASCII /E ERASE SCOPE/<15><12>
1589	010516	020040	051105	051501	
1590	010524	020105	041523	050117	
1591	010532	006505	012		
1592	010535	114	020040	020040	.ASCII /L GET LOWER THRESHOLD/<15><12>
1593	010542	020040	042507	020124	
1594	010550	047514	042527	020122	
1595	010556	044124	042522	044123	
1596	010564	046117	006504	012	
1597	010571	125	020040	020040	.ASCII /U GET UPPER THRESHOLD/<15><12>
1598	010576	020040	042507	020124	
1599	010604	050125	042520	020122	

1600	010612	044124	042522	044123	
1601	010620	046117	006504	012	
1602	010625	101	020040	020040	.ASCII /A DISPLAY ISOTOPE A/<15><12>
1603	010632	020040	044504	050123	
1604	010640	040514	020131	051511	
1605	010646	052117	050117	020105	
1606	010654	006501	012		
1607	010657	102	020040	020040	.ASCII /B DISPLAY ISOTOPE B/<15><12>
1608	010664	020040	044504	050123	
1609	010672	040514	020131	051511	
1610	010700	052117	050117	020105	
1611	010706	006502	012		
1612	010711	127	020040	020040	.ASCII /W SELECT FIRST BIT MAP/<15><12>
1613	010716	020040	042523	042514	
1614	010724	052103	043040	051111	
1615	010732	052123	041040	052111	
1616	010740	046440	050101	005015	
1617	010746	020115	020040	020040	.ASCII /M SELECT SECOND BIT MAP/<15><12>
1618	010754	051440	046105	041505	
1619	010762	020124	042523	047503	
1620	010770	042116	041040	052111	
1621	010776	046440	050101	005015	
1622	011004	020106	020040	020040	.ASCII /F FREE RUN MODE/<15><12>
1623	011012	043040	042522	020105	
1624	011020	052522	020116	047515	
1625	011026	042504	005015		
1626	011032	020112	020040	020040	.ASCII /J JOYSTICK CALIB./<15><12>
1627	011040	045040	054517	052123	
1628	011046	041511	020113	040503	
1629	011054	044514	027102	005015	
1630	011062	020124	020040	020040	.ASCIIZ /T DISPLAY INTENSITY TEST/
1631	011070	042040	051511	046120	
1632	011076	054501	044440	052116	
1633	011104	047105	044523	054524	
1634	011112	052040	051505	000124	
1635	011120	005015	020116	020040	HELP1: .ASCII <15><12>/N COLLECT NEW DATA/<15><12>
1636	011126	020040	041440	046117	
1637	011134	042514	052103	047040	
1638	011142	053505	042040	052101	
1639	011150	006501	012		
1640	011153	103	020040	020040	.ASCII /C COLLECT MORE DATA/<15><12>
1641	011160	020040	047503	046114	
1642	011166	041505	020124	047515	
1643	011174	042522	042040	052101	
1644	011202	006501	012		
1645	011205	130	020040	020040	.ASCII /X CHANGE CAMERA CHANNEL/<15><12>
1646	011212	020040	044103	047101	
1647	011220	042507	041440	046501	
1648	011226	051105	020101	044103	
1649	011234	047101	042516	006514	
1650	011242	012			
1651	011243	123	020040	020040	.ASCII /S STOP COLLECTION/<15><12>
1652	011250	020040	052123	050117	
1653	011256	041440	046117	042514	

CZNCDA NCV11 EXERCISER MACY11 27(654) 8-AUG-78 15:16 PAGE 36
CZNCDA.P11 ASCII MESSAGES

SEQ 0044

1654	011264	052103	047511	006516		
1655	011272	012				
1656	011273	107	020040	020040	.ASCII /G	INITILIZE/<15><12>
1657	011300	020040	047111	052111		
1658	011306	046111	055111	006505		
1659	011314	012				
1660	011315	132	020040	020040	.ASCII /Z	ZOOM GAIN/<15><12>
1661	011322	020040	047532	046517		
1662	011330	043440	044501	006516		
1663	011336	012				
1664	011337	122	020040	020040	.ASCII /R	REGULAR GAIN/<15><12>
1665	011344	020040	042522	052507		
1666	011352	040514	020122	040507		
1667	011360	047111	005015			
1668	011364	020117	020040	020040	.ASCII /O	OTHER CONSOLE TERMINAL/<15><12>
1669	011372	047440	044124	051105		
1670	011400	041440	047117	047523		
1671	011406	042514	052040	051105		
1672	011414	044515	040516	006514		
1673	011422	012				
1674	011423	110	020040	020040	.ASCII /H	HELP THE OPERATOR AND REPEAT THIS LIST/
1675	011430	020040	042510	050114		
1676	011436	052040	042510	047440		
1677	011444	042520	040522	047524		
1678	011452	020122	047101	020104		
1679	011460	042522	042520	052101		
1680	011466	052040	044510	020123		
1681	011474	044514	052123			
1682	011500	015	012	000	BCRLF: .BYTE	15,12,0
1683						
1684					:LOC. 20000 THRU 57776 ARE BUFFER AREA	
1685						
1686	000001				.END	

		13#	209	224	225	226	227	228	229	230	231
ABASE =	172760										
AWRIT	005226	687	691	718	722	739	743	969#			
BCRLF	011500	746	1682#								
BELLEN	001316	269#									
BIT0 =	000001	109#	800	855							
BIT00 =	000001	99#	109								
BIT01 =	000002	98#	108								
BIT02 =	000004	97#	107								
BIT03 =	000010	96#	106								
BIT04 =	000020	95#	105								
BIT05 =	000040	94#	104								
BIT06 =	000100	93#	103								
BIT07 =	000200	92#	102								
BIT08 =	000400	91#	101								
BIT09 =	001000	90#	100								
BIT1 =	000002	108#									
BIT10 =	002000	89#	501	549							
BIT11 =	004000	88#	799								
BIT12 =	010000	87#									
BIT13 =	020000	86#									
BIT14 =	040000	85#									
BIT15 =	100000	84#									
BIT2 =	000004	107#									
BIT3 =	000010	106#									
BIT4 =	000020	105#									
BIT5 =	000040	104#	428	697	808						
BIT6 =	000100	103#	705								
BIT7 =	000200	102#									
BIT8 =	000400	101#	431	467	503	523	862				
BIT9 =	001000	100#									
BOOB00	002026	367	370	373	407#	479	498				
BOX	003036	563#									
BPTVEC=	000014	116#									
CAMERA	001274	260#	541*	542	693	854					
CAMOUT	007706	695*	1513#								
CHANGE	003026	413	415	418	436	462	508	518	546	555#	
CHARCT	001302	263#	594*	603*							
CKSWR =	104406	1445#									
COMSAV	001272	259#	761	861*							
CPERL	001304	264#	610*	630							
CR =	000015	24#	1264	1274							
CRLF =	000200	25#	1235	1274							
DASH	010416	721	742	1577#							
DCONT	004756	569	576	777	784	893#					
DDISP =	177570	31#	171	289							
DISBG	004354	799#									
DISLOP	003312	618#	669	681							
DISPLA	001142	171#	289*	297*							
DISPRE	000174	137#	297								
DISPY	004744	565	572	583	586	667	773	780	788	791	891#
DISPY2	005006	812	904#	905							
DMATRX	003270	615#									
DSWR =	177570	30#	170	288							
DUMMY1	001242	247#	693*	694*	695	716*	723	737*	744		

[illegible]

PC = %000007	43#	301*	320*	408*	409*	416*	419*	422*	423*	437*	441*	464*	469*
	470*	506*	507*	520*	555*	559*	565*	569*	572*	576*	583*	586*	657*
	768*	773*	777*	780*	784*	788*	791*	807*	812*	843*	856*	864*	868*
	875*	885*	898*	909*	929*	947*	952*	1007*	1010*	1015*	1029*	1089*	1242*
	1249*	1256*	1270*	1272*	1486								
PIRQ = 177772	29#												
PIRQVE = 000240	123#												
PXASM 001260	254#	591*	659*	660*	661*	662*	663*	666	670*				
PIXCNT 001256	253#	590*	664	668*	671*								
PR0 = 000000	46#												
PR1 = 000040	47#												
FR2 = 000100	48#												
FR3 = 000140	49#												
FR4 = 000200	50#												
FR5 = 000240	51#												
PR6 = 000300	52#												
PR7 = 000340	53#												
PS = 177776	26#	27											
PSW = 177776	27#												
PWRMSG 007742	1485	1523#											
PWRVEC = 000024	118#	282*	283*	1454*	1455*	1464*	1470*	1482*	1483*				
RDCHR = 104407	1149	1446#											
RDLIN = 104410	1372	1447#											
RDOCT = 104411	457	476	513	529	1448#								
RESVEC = 000010	113#												
ROUTA 002046	378	412#											
ROUTB 002060	379	414#											
ROUTC 002072	380	416#											
ROUTD 002102	381	418#											
ROUTE 002106	382	419#											
ROUTF 002116	383	421#	760										
ROUTG 002212	384	437#											
ROUTH 002222	385	439#											
ROUTJ 002274	387	452#											
ROUTL 002300	389	453#											
ROUTM 002340	390	463#											
ROUTN 002372	391	469#											
ROUTO 002406	392	472#											
ROUTR 002526	395	500#											
ROUTS 002544	396	503#											
ROUTT 002562	397	506#											
ROUTU 002576	398	509#											
ROUTW 002636	400	519#											
ROUTX 002666	401	525#											
ROUTZ 003006	403	549#											
ROWCNT 001306	265#	617*	620*	622*									
RTABLE 001742	372	374	378#										
RO - %000000	34#	306*	308	309*	363*	366	368*	369	371*	372	374	424*	426*
	466*	467*	477*	482	484	485	486	487	488	489	490	522*	523*
	563*	566*	567	571*	573*	574	578*	579*	581	585*	595*	597	601*
	607*	610	649*	673*	674	677*	678	771*	774*	775	779*	781*	782
	786*	790*	792*	793	819*	821	822*	825*	828*	829	830*	836*	837
	838*	839	871*	872*	873	892	917*	924*	971	972*	981*	984*	998
	999*	1001	1008	1011*	1012*	1013	1014*	1017*	1020*	1024*	1025*	1026*	1027*

SEQ 0048

```

STACK = 001100
START 001322
START1 001554
STKLMT= 177774
SWR    001140
SWREG  000176
SW0     = 000001
SW00    = 000001
SW01    = 000002
SW02    = 000004
SW03    = 000010
SW04    = 000020
SW05    = 000040
SW06    = 000100
SW07    = 000200
SW08    = 000400
SW09    = 001000
SW1     = 000002
SW10    = 002000
SW11    = 004000

```

[illegible]

WD006	007662	757	1509#					
WD007	007673	713	1511#					
XYAVE	005356	807	998#					
XYBUF	005504	1004	1006	1009	1031#	1033		
XYBUFE=	005604	1002	1022	1032#				
XYBUFP	005604	1000	1005*	1033#				
\$AUTOB	001134	167#	1053	1202				
\$BDADR	001122	162#						
\$BDDAT	001126	164#						
\$CHARC	006724	1240*	1250*	1257	1266*	1271#		
\$CKSWR	005606	1045#	1445					
\$CMTAG	001100	150#	273	274				
\$CM3 =	000000	180#						
\$CNTLG	006460	1056	1197#					
\$CNTLU	006453	1073	1171	1196#				
\$CRLF	001161	181#	1084	1176	1196	1239	1274	1406
\$ERFLG	001103	153#						
\$ERMAX	001115	159#						
\$ERRPC	001116	160#						
\$ERRTB	001164	198#						
\$ERTTL	001112	157#						
\$FILLC	001156	178#	1243	1274				
\$FILLS	001155	177#	1274					
\$GDADR	001120	161#						
\$GDDAT	001124	163#						
\$GTSWR	005656	1057#	1443					
\$HD =	000003	11	12					
\$HIOCT	007314	1394*	1405#					
\$ICNT	001104	154#						
\$ILLUP	007542	1454	1470	1489#				
\$INTAG	001135	168#	1085	1202				
\$ITEMB	001114	158#						
\$LF	001162	182#	1186	1196	1274	1406		
\$LPADR	001106	155#						
\$LPERR	001110	156#						
\$MAIL =	***** U	300	1227					
\$MNEW	006476	1060	1200#					
\$MSWR	006465	1057	1198#					
\$NULL	001154	176#	1245	1274				
\$OCNT	007152	1307*	1336*	1349#				
\$OMODE	007154	1302*	1306*	1311	1314*	1325*	1351#	
\$PASS	001100	151#						
\$PWRAD	007536	1487#						
\$PWRDN	007376	282	1454#	1482				
\$PWRMG	007532	1485#						
\$PWRUP	007450	1464	1470#					
\$QUES	001160	180#	1103	1179	1196	1274	1403	1406
\$RDCHR	006070	1116#	1446					
\$RDDEC=	***** U	1449						
\$RDLIN	006210	1144#	1447					
\$RDOCT	007156	1367#	1448					
\$RDSZ =	000007	1137#						
\$R2A =	***** U	1449						
\$SAVRE=	***** U	1449						

[illegible]

ADC	734														
ADD	309	579	585	616	619	663	680	694	733	790	822	828	830	838	990
	1020	1074	1083	1231	1303	1313	1390								
ASL	371	633	654	655	656	657	1024	1025	1026	1097	1098	1099	1383	1385	1387
	1419														
ASR	975	976	977												
BCC	302	370													
BCS	367														
BEQ	364	373	538	568	575	582	637	651	751	759	776	783	827	840	925
	961	1054	1081	1096	1165	1234	1269	1330	1378						
BGT	1093	1134	1337	1380											
BHI	600	608	625	627											
BHIS	598														
BIC	368	428	467	501	523	534	677	883	891	893	937	979	1028	1050	1067
	1094	1121	1127	1135	1327	1389									
BIS	431	503	542	550	800	853	854	855	862	880	897	908	1101	1332	1333
BISB	1012	1019													
BIT	697	705	808	839											
BITB	1257														
BLO	635	1003	1023	1160											
BLOS	645	1148													
BLT	1091	1132	1248	1338	1382										
BNE	277	291	311	427	430	435	440	478	604	621	643	665	675	679	698
	706	736	755	794	809	824	832	874	923	939	983	1046	1052	1072	1079
	1086	1123	1129	1152	1154	1170	1174	1184	1228	1236	1244	1258	1265	1328	1474
BPL	669	802	882	895	905	950	959	1048	1064	1119	1125	1222	1262	1326	
BR	293	314	410	449	540	570	577	587	629	701	709	778	785	813	846
	928	963	974	1075	1102	1104	1130	1163	1172	1178	1180	1224	1241	1251	1260
	1267	1304	1319	1340	1391	1404	1466	1490							
CLC	640	658	841												
CLR	275	322	324	357	359	360	365	424	432	459	492	497	500	504	515
	519	536	543	591	628	632	670	730	731	771	863	869	870	872	1017
	1018	1061	1062	1145	1168	1317	1375	1376	1472						
CLRB	1175	1185	1240	1266	1400										
CMP	276	290	310	315	366	369	429	495	567	574	581	597	599	624	634
	674	678	754	775	782	793	823	831	844	873	938	1002	1022	1045	1051
	1071	1078	1090	1092	1122	1128	1131	1133	1147	1159					
CMPB	1053	1085	1151	1169	1173	1183	1233	1235	1243	1264	1268	1379	1381		
COMB	906														
DEC	426	434	537	603	620	642	652	668	735	922	924	982	1158		
DECB	1247	1250	1325	1336											
EMT	18														
HALT	136	305	317	1223	1465	1489									
INC	303	535	539	566	573	638	646	672	774	781	792	1021	1100	1331	1339
	1473														
INCB	1270														
IOT	19														
JMP	140	374	413	415	417	418	420	433	436	438	451	452	462	468	471
	479	502	505	508	518	524	546	551	609	676	681	760	762	940	
JSR	301	320	408	409	416	419	422	423	437	441	445	447	464	469	470
	506	507	520	555	559	565	569	572	576	583	586	667	685	687	689
	691	712	714	718	720	722	724	739	741	743	745	752	756	768	773
	777	780	784	788	791	807	812	868	947	985	1007	1010	1089	1242	1249
	1256														

MOV	274	278	280	281	282	283	286	287	288	289	294	296	297	298	306
	307	308	312	318	321	323	325	326	327	328	358	361	363	407	412
	414	421	425	444	453	454	458	460	463	466	472	473	477	480	481
	483	484	486	488	490	491	493	496	498	509	510	514	516	522	525
	526	530	544	549	563	564	571	578	580	584	588	589	590	594	595
	596	601	610	615	617	618	622	630	631	649	666	671	673	684	716
	717	729	732	737	738	761	772	779	786	787	789	795	799	803	819
	820	821	825	829	833	836	837	842	851	852	861	871	884	892	896
	907	917	918	919	920	921	927	935	936	946	948	951	957	969	970
	971	972	973	978	984	998	999	1000	1001	1004	1005	1006	1009	1013	1014
	1058	1082	1087	1116	1117	1144	1146	1157	1188	1189	1190	1191	1225	1226	1230
	1245	1300	1308	1309	1310	1316	1323	1341	1342	1343	1344	1345	1367	1368	1369
	1370	1371	1373	1374	1393	1394	1395	1396	1397	1415	1416	1420	1426	1427	1454
	1455	1456	1457	1458	1459	1460	1461	1462	1463	1464	1470	1471	1475	1476	1477
	1478	1479	1480	1481	1482	1483	1486								
MOV8	541	693	695	699	700	702	703	707	708	710	711	962	981	1008	1049
	1066	1120	1126	1150	1155	1161	1166	1181	1227	1255	1263	1301	1302	1305	1306
	1307	1311	1314	1315	1334	1377	1418								
NEG	1312														
NOP	442	443	465	521	531	532	533	804	805	806	810	811			
RESET	319														
ROL	1318	1320	1321	1322	1324	1384	1386	1388							
ROR	641	659	660	661	662										
RTI	142	295	362	455	461	474	494	499	511	517	527	545	941	1088	1136
	1192	1232	1346	1398	1428	1488									
RTS	843	856	864	875	885	898	909	929	952	964	987	1015	1029	1272	1421
SEC	845														
SUB	607	623	626	639	644	926									
SWAB	653	1011	1027												
TRAP	1430	1439	1440	1441	1443	1445	1446	1447	1448						
TST	313	372	439	482	485	487	489	602	636	650	664	750	758	826	834
	835	904	958	1080	1095	1153	1164	1187	1229	1237	1259	1329	1392	1399	1417
TSTB	801	881	894	949	960	1047	1063	1118	1124	1221	1261				
.ASCII	180	181	1511	1514	1516	1578	1584	1588	1592	1597	1602	1607	1612	1617	1622
	1626	1635	1640	1645	1651	1656	1660	1664	1668	1674					
.ASCIIZ	182	1196	1197	1198	1200	1494	1498	1502	1504	1507	1509	1519	1523	1529	1534
	1543	1550	1558	1563	1572	1577	1630								
.BLKB	1195														
.BYTE	152	153	158	159	167	168	176	177	178	179	991	993	1193	1194	1347
	1348	1349	1350	1513	1682										
.DSABL	1105														
.ENABL	1	1038													
.END	1686														
.ENDC	6	18	110	124	141	146	150	152	180	184	271	278	279	280	282
	284	300	815	819	849	851	859	861	866	868	878	880	888	891	902
	904	911	917	933	935	944	946	954	956	967	969	996	998	1035	1038
	1039	1041	1069	1105	1109	1137	1138	1146	1148	1151	1179	1196	1202	1204	1207
	1227	1275	1278	1353	1356	1362	1406	1407	1410	1416	1419	1438	1439	1440	1441
	1442	1443	1444	1445	1446	1447	1448	1449	1450	1453	1462	1463	1469	1475	1476
	1486	1488	1492												
.EQUIV	18	19	27	72	73	74	75	76	77	78	79	80	81	100	101
	102	103	104	105	106	107	108	109							
.EVEN	994	1202													
.IF	2	16	82	110	139	145	149	151	180	183	184	271	273	278	280

	282	284	300	814	818	848	850	858	860	865	867	877	879	887	890
	901	903	910	916	932	934	943	945	953	955	966	968	995	997	1034
	1037	1039	1040	1041	1069	1108	1109	1137	1145	1147	1151	1152	1195	1196	1202
	1203	1206	1227	1274	1277	1352	1355	1358	1374	1406	1409	1415	1419	1430	1439
	1440	1441	1442	1443	1445	1446	1447	1448	1449	1452	1462	1463	1468	1475	1476
	1484	1486	1488	1492											
.IFF	16	146	149	151	180	184	278	815	819	849	851	859	861	866	868
	878	880	888	891	902	904	911	917	933	935	944	946	954	956	967
	969	996	998	1035	1038	1041	1109	1111	1116	1137	1138	1147	1179	1195	1204
	1207	1275	1278	1353	1356	1407	1410	1416	1450	1453	1469	1484			
.IFT	1111	1116	1379	1399	1406										
.IFTF	1056	1109	1112	1375	1383	1405									
.IIF	1	6	11	12	136	183	279	1038	1059	1187	1196	1202	1274	1406	1438
	1439	1440	1441	1443	1445	1446	1447	1448							
.IRP	271	1369	1395	1456	1462	1475	1476								
.LIST	1	124	136	180	271	284	814	1032	1137	1430	1438	1439	1440	1441	1442
	1443	1444	1445	1446	1447	1448	1449								
.MACRO	143	1430													
.MCALL	1	124	284												
.NLIST	1	124	136	180	271	284	814	1032	1137	1430	1438	1439	1440	1441	1442
	1443	1444	1445	1446	1447	1448	1449								
.PAGE	143	184	223	552	1492										
.REPT	136	485	1031												
.SBTTL	14	130	139	143	184	271	272	376	404	552	814	1035	1204	1275	1353
	1407	1430	1450	1492	1521										
.TITLE	1														
.WORD	136	137	138	151	154	155	156	157	160	161	162	163	164	165	166
	169	170	171	1271	1351	1402	1405	1437	1485	1487					

ERRORS DETECTED: 0

CZNCDA NCV11 EXERCISER MACY11 27(654) 8-AUG-78 15:16 PAGE 48 E 5
CZNCDA.P11

SEQ 0056

*CZNCDA,CZNCDA/CRF/SOL=CZNCDA
RUN-TIME: 12 6 1 SECONDS
CORE USED: 18K