

.REM 1

2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40

IDENTIFICATION

PRODUCT CODE: AC-E157E-MC  
 PRODUCT NAME: CZM980C M9312/1144 UBI BOOT  
 DATE: JANUARY, 1982  
 MAINTAINER: DIAGNOSTIC ENGINEERING

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT. THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED UNDER A LICENSE AND MAY ONLY BE SUED OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1979,1982 BY DIGITAL EQUIPMENT CORPORATION

## HISTORY SECTION

PBRDA0 WAS RELEASED MARCH, 1978.  
PBRDB0 WAS RELEASED JUNE, 1978.  
PBRD440 WAS RELEASED JANUARY, 1979  
CZM9BDO WAS RELEASED OCTOBER 1979  
CZM9BEO WAS RELEASED JANUARY, 1982

REVISION B WAS CREATED TO PROVIDE THE FOLLOWING ENHANCEMENTS:

1. PROPERLY CHECK THE BOOT ROM'S ALPHABETIC SEQUENCE AND, IF NOT IN CORRECT SEQUENCE, PRINT THE CORRECT SEQUENCE AS AN ERROR MESSAGE. ALSO CHECK FOR NO HOLES AND CHECK FOR ROM IN SOCKET #2 IF 11/60 AND ONLY ONE ROM EXISTS, ELSE PRINT THE CORRECT SEQUENCE.
2. THE DIAGNOSTIC CANNOT DETERMINE THE DEVICE CODE FOR A CONTINUATION ROM. THEREFORE, CONTINUATION ROMS ARE TREATED AS EXTENSIONS OF THE PRECEDING DEVICE CODE ROM. ILLEGAL PLACEMENT OF CONTINUATION ROMS ARE REPORTED IN ERROR MESSAGES. A DUPLICATE DEVICE ROM IS ALSO REPORTED IN AN ERROR MESSAGE.

COMMENT FIELDS BEGINNING WITH THE CHARACTERS ":-+" IDENTIFY ALL LINES ADDED OR MODIFIED BY REV B0.

REVISION C WAS CREATED TO PROVIDE THE FOLLOWING ENHANCEMENTS:

1. TO UPDATE THE DIAGNOSTIC BASED ON FAULT INSERTION OF THE M9312 MODULE.
2. TO ADD A ' NO ROMS FOUND ' MESSAGE.
3. TO SIZE FOR A FALSE ERROR CAUSED BY A REAL FAILURE TO FIND THE POWER FAIL VECTOR WHEN ONLY THE CPU ROM IS PRESENT. THIS SHOULD NOT BE TREATED AS A FATAL ERROR (WHICH IT WAS IN REV. B). PREVIOUSLY, APT WOULD NOT RUN IN PRINTING MODE DISABLED DUE TO THIS ERROR.

COMMENT FIELDS BEGINNING WITH THE CHARACTERS ":-+" IDENTIFY ALL LINES ADDED OR MODIFIED BY REV. C.

REVISION D WAS CREATED TO ACCOMMODATE TESTING OF 11/44 UBI MODULE. THE UBI MODULE HAS LOGIC WHICH IS FUNCTIONALLY EQUIVALENT TO THE M9312.

1. SINCE THE DIAGNOSTIC CALLS OUT E NUMBERS REFERENCING THE 4 BOOTSTRAP & 1 BOOT DIAGNOSTIC ROM IC'S, USE MPT INSTRUCTION TO DETERMINE IF CPU IS 11/44. IF IT IS THEN USE SET OF E NUMBERS CONSISTENT WITH UBI MODULE.

COMMENT FIELDS BEGINNING WITH THE CHARACTERS ":-+" IDENTIFY ALL LINES ADDED OR MODIFIED BY REV. D.

REVISION E WAS CREATED TO ACCOMMODATE TESTING OF 11/24 WITH BOOT ROMS ON EITHER THE MAP MODULE OR THE CPU MODULE. ALSO, CORRECTIONS WERE MADE TO SOME OF THE

42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98

99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155

MESSAGE PRINTOUTS WHEN RUN ON AN 11/24. IT ALSO CORRECTED CODING ERRORS IN THE 'PROMP:' AND 'DEVCO:' SUBROUTINES, USED WHEN IN APT MODE WHEN BIT 7 OF SENVM IS SET (THE 200 BIT). ATTENTION: PROPER SETUP OF THE APT TABLES IS NECESSARY TO RUN IN THAT MODE. SEE SECTION UNDER APT (4.3). WHILE RUNNING WITH 4 ROMS, DIAG FAILED. CODE WAS CHANGED TO CORRECT THIS PROBLEM. WHILE RUNNING WITH CONTINUATION ROMS, DIAGNOSTIC FAILED. CODE WAS CHANGED TO CORRECT THIS PROBLEM. WRONG ROM NUMBER WAS PRINTED ON A ROM FAILURE, CODE CORRECTED TO POINT TO THE CORRECT ROM ASCII STREAM.

COMMENT FIELDS ENDING WITH ";DPM001" IDENTIFY ALL LINES ADDED OR MODIFIED BY REV E.

1.0 ABSTRACT

THIS PROGRAM VERIFIES THE ROM INFORMATION FOR THE M9312 BOOTSTRAP TERMINATOR OR 11/44 J31 TERMINATOR. IT HAS TWO MODES OF OPERATION; STAND-ALONE MO WHICH REQUIRES OPERATOR INTERVENTION AND APT-MODE.

2.0 REQUIREMENTS

2.1 HARDWARE

ANY PDP-11 UNIBUS PROCESSOR WITH CONSOLE TERMINAL AND/OR HARDWARE SWITCH REGISTER  
M9312 BOOT STRAP TERMINATOR  
4K MEMORY

NOTE: IF 11/44 CPU, M9312 BOOTSTRAP TERMINATOR IS NOT NEEDED

2.2 SOFTWARE

THIS PROGRAM REQUIRES THAT THE CORRECT OPERATION OF THE PROCESSOR, MEMORY AND CONSOLE TERMINAL HAVE BEEN VERIFIED BY THE APPROPRIATE DIAGNOSTICS.

3.0 LOADING AND STARTING PROCEDURES

3.1 LOAD THE PROGRAM BY ANY OF THE STANDARD PROCEDURES FOR ABSOLUTE PROGRAM FORMATS.

3.2 STARTING ADDRESS

200- DO CRC VERIFICATION AND SIZING

3.3 RESTART ADDRESS

204- RESTART WITHOUT SIZING

156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212

3.4 SPECIAL ENVIRONMENTS

THIS PROGRAM IS APT COMPATIBLE SEE SECTION FOR ETABLE SET-UP.  
FOLLOW STANDARD APT PROCEDURES FOR LOADING AND STARTING.

4.0 OPERATING PROCEDURES

4.1 OPERATIONAL SWITCH SETTINGS

THE PROGRAM IS DESIGNED TO USE THE HARDWARE SWITCH REGISTER,  
HOWEVER IF THIS REGISTER IS NOT AVAILABLE THE PROGRAM WILL USE  
LOCATION 176 AS THE SWITCH REGISTER.

SW 15=1 OR UP-- HALT ON ERROR  
SW 13=1 OR UP-- INHIBIT ERROR TYPEOUTS  
SW 10=1 OR UP-- BELL ON ERROR

4.2 EXECUTION TIMES

EXECUTION TIME IS DEPENDENT ON THE CONSOLE TERMINAL DURING THE  
FIRST PASS. ALL OTHER PASSES TAKE LESS THEN 1 SECOND.

4.3 APT PROCEDURES

THERE ARE TWO CHOICES WHEN RUNNING UNDER APT. IF THE SIZE BIT  
(BIT 7-SENVM) IS CLEAR THE PROGRAM WILL OPERATE IN NORMAL  
STAND-ALONE MODE. IF THE SIZE BIT IS SET THE PROGRAM WILL  
COMPARE PARAMETERS FROM THE BOARD TO THE CONTENTS OF THE ETABLE.  
THIS TABLE MUST BE SET UP PROPERLY AS TO THE HARDWARE CONFIGURA-  
TION, OTHERWISE ERROR(S) WILL RESULT. TO USE THE APT COMPASSION  
FEATURE THE ETABLE MUST BE SET UP IN THIS ORDER:

SENV:	DON'T CARE
SENVM:	200 OR 240
SSWREG:	DON'T CARE
\$USWR:	NOT USED
\$CPUOP:	NOT USED
\$MAMS1:	NOT USED
\$MTYP1:	NOT USED
\$MADR1:	NOT USED
\$MAMS2:	NOT USED
\$MTYP2:	NOT USED
\$MADR2:	NOT USED
\$MAMS3:	NOT USED
\$MTYP3:	NOT USED
\$MADR3:	NOT USED
\$MAMS4:	NOT USED

213 SMAMS4: NOT USED  
 214 SMYYP4: NOT USED  
 215 SMADR4: NOT USED  
 216 SVECT1: NOT USED  
 217 SVECT2: NOT USED  
 218 \$BASE: PSEUDO POWER-FAIL VECTOR ADDRESS - MUST BE LOADED IF BIT 7  
 219 OF \$ENVM IS SET (THE 200 BIT) (DIAGNOSTIC ASSUMES 173024  
 220 IF BIT 7 OF \$ENVM IS CLEAR)  
 221 \$DEVN: NOT USED  
 222 \$CDW1: CONTENTS OF ADDRESS IN \$BASE - MUST BE LOADED IF BIT 7  
 223 OF \$ENVM IS SET (THE 200 BIT) (DIAGNOSTIC ASSUMES 1650)  
 224 IF BIT 7 OF \$ENVM IS CLEAR)  
 225 \$CDW2: NOT USED  
 226 \$DDW0-\$DDW11: UP TO 12 DEVICE CODES EXPECTED - MUST BE LOADED IF \$ENVM=200 OR 2 0  
 227 \$DDW12: NOT USED  
 228 \$DDW13: NOT USED  
 229 \$DDW14: NOT USED  
 230 \$DDW15: NOT USED

NOTE: THE ORDER FOR LOADING \$DDW0 IS IMPORTANT. THE FIRST VALUE SHOULD BE FOR THE DIAGNOSTIC / CPU ROM, FOLLOWED BY THE APPROPRIATE OCTAL VALUES FOR THE BOOT ROMS PRESENT. SOME OCTAL VALUES USED ARE:

OCTAL DATA	MNEMONIC	PIN LABELING
040460	AD	248FT
040160	BO	233F1
041460	CG	
042113	DK	756A9
042114	DL	751A9
042130	DX	753A9
046524	MT	757A9

5.0 SUBROUTINE ABSTRACTS

5.1 NOROMS

THIS ROUTINE IS CALLED ONLY IF NO ROMS WERE FOUND DURING SIZING IT DOES A READ OF ALL BOOTSTRAP ROM ADDRESSES AND COMPARES THE CONTENTS TO A KNOWN EXPECTED VALUE.

5.2 CHECKS

THIS ROUTINE SETS UP THE FIRST, LAST AND EXCEPTION ADDRESSES FOR THE "CALSUM" SUBROUTINE. IT RECEIVES THE CALCULATED CHECKSUM FROM "CALSUM" AND COMPARES IT AGAINST THE GOOD CHECKSUM TO DETERMINE CRC ERRORS.

5.3 CALSUM

THIS ROUTINE CALCULATES THE CRC16 CHECKSUM OF EACH ROM. IT RECEIVES THE FIRST ADDRESS TO BE CHECKED (FIRSTA), THE LAST ADDRESS TO BE CHECKED (LASTAD) AND THE EXCEPTION ADDRESS (EXCADD) FROM THE "CHECKS" MODULE AND RETURNS TO IT THE CHECKSUM IN R4.

5.4 PROMP

THIS ROUTINE PROCESSES THE ROM PARAMETERS. IT CHECKS THE SIZE/DON'T SIZE BIT IN THE APT ETABLE AND EITHER FORMATS THE SIZING MESSAGES OR COMPARES THE SIZING INFORMATION TO THE APT ETABLE DATA.

5.5 DEVCOD

THIS ROUTINE LOCATES EACH DEVICE CODE AND PASSES IT AND THE ADDRESS IN WHICH IT WAS FOUND BACK TO THE "PROMP" MODULE.

5.6 PPFVAR

THIS ROUTINE DETERMINES THE PSEUDO POWER-FAIL VECTOR ADDRESS AND PASSES IT AND ITS CONTENTS TO THE "PROMP" MODULE

5.7 PUTMES

THIS ROUTINE FORMATS THE PARAMETER AND POWER-FAIL ADDRESS MESSAGES.

5.8 ERRHAN

THIS ROUTINE FORMATS ALL ERROR MESSAGES AND CALLS THE TYPE ROUTINE TO OUTPUT THEM. IT ALSO USES THE OPERATIONAL SWITCHES TO DETERMINE WHETHER TO OUTPUT THE MESSAGE, OR HALT.

5.9 FILBUF

THIS ROUTINE FILLS THE MESSAGE BUFFER WITH ASCII CHARACTERS. IT RECIEVES THE ADDRESS OF THE ASCII IN R5.

5.10 OCASC

THIS ROUTINE TAXES A SIXTEEN BIT BINARY NUMBER AND CONVERTS IT TO 6 ASCII CHARACTERS.

270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326

5.11 OCADD

THIS ROUTINE IS USED BY THE 'PUTMES' MODULE WHEN THE DATA IN R3 IS NOT IN THE RIGHT MODE TO BE HANDLED BY THE 'OCASC' MODULE. IT MOVES THE ADDRESSING MODE OF R3 UP ONE LEVEL OF DEFERMENT.

6.0 RELIABILITY//AVAILABILITY//SERVICEABILITY

WHEN RUNNING IN ANY ENVIRONMENT BUT APT THERE IS ONLY ONE ERROR DETECTED BY THE PROGRAM. THIS ERROR IS A CHECKSUM ERROR. THE MESSAGE WILL BE:

CRC ERROR IN ROM EXX

THE FOLLOWING ERROR MESSAGES WERE ADDED IN REV B0:

1. A CONTINUATION ROM IS INCORRECTLY LOCATED IN ROM X(EXX)
2. A CONTINUATION ROM IS MISSING FOR DEVICE CODE XX
3. THERE IS A DUPLICATE ROM WITH DEVICE CODE XX
4. ROM SEQUENCE IS INCORRECT AS PER INSTALLATION PROCEDURE.

SEQUENCE SHOULD BE:

ROM 1(E35)	XX
ROM 2(E33)	XX
ROM 3(E34)	XX
ROM 4(E32)	XX

WHEN RUNNING IN THE APT ENVIRONMENT THREE OTHER ERRORS MAY OCCUR.

1. AN ERROR WILL OCCUR WHEN THE DEVICE CODES IN THE ETABLE DO NOT MATCH THE DEVICE CODES FOUND IN THE ROMS. THE ERROR MESSAGE WILL BE EITHER:

1. COULD NOT FIND DEVICE CODE XX.
2. FOUND UNEXPECTED DEVICE CODE XX.

THE FIRST MESSAGE WILL BE PASSED TO APT IF A DEVICE CODE LISTED IN THE ETABLE CANNOT BE FOUND IN THE EXISTING ROMS. THE SECOND MESSAGE WILL BE SENT IF A DEVICE CODE NOT LISTED IN THE ETABLE IS FOUND IN A ROM.

2. THE SECOND ERROR WILL BE IF THE PSEUDO POWER-FAIL VECTOR ADDRESS IN THE ETABLE DOES NOT MATCH THE ADDRESS DETERMINED BY THE PROGRAM. TO BE IN THE BOARDS' SWITCHES. THE MESSAGE WILL BE:

POWER-FAIL VECTOR	
EXPECTED	RECEIVED
-----	-----

327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383

384  
385 WHERE EXPECTED IS THE CONTENTS OF THE ETABLE AND RECEIVED IS  
386 THE VALUE FOUND BY THE PROGRAM.  
387

388 THE THIRD ERROR WILL BE IF THE CONTENTS OF BOARD'S PSEUDO  
389 POWER-FAIL VECTOR ADDRESS DOES NOT MATCH THE VALUE EXPECTED  
390 FROM THE ETABLE. THE MESSAGE WILL BE:  
391

392 POWER-FAIL DATA ERROR  
393 EXPECTED RECEIVED  
394 -----

397  
398  
399  
400  
401  
402  
403  
404  
405  
406

7.0 NOTE: DIAGNOSTIC COVERAGE

THIS DIAGNOSTIC DOESNOT PERFORM ANY EXAMINATION OF  
THE BOOT CIRCUITRY ON THE M9312. IF YOU WISH TO CHECK THIS  
CIRCUITRY YOU MUST FIRST, DETERMINE WHICH DEVICE YOU WILL BOOT  
AND SECOND, YOU MUST SET THE SWITCHES ON THE M9312  
TO THE APPROPRIATE SETTINGS. YOU MAY NOW ATTEMPT TO BOOT  
THE M9312. IF THE DEVICE DOES BOOT THIS SHOULD BE  
EVIDENT FROM THE CONSOLE TERMINAL, IF PRESENT, OR FROM THE RUN  
LIGHT ON THE FRONT PANEL, WHICH SHOULD BE LIT. RIGHT? RIGHT!



```

421
422 000000      .ENABLE ABS
423             .ENABL AMA
424             .LIST ME
425             .NLIST MC
426             .NLIST MD
427             .NLIST CND
428             .NLIST BEA
435             .MCALL .$ASTAT,.$SAVE,.$SCATCH,.$CMTAG,.$SETUP,.$SEOP,.$STYPE
436             .MCALL STARS,.$EQUATE,.$SAPTHR,.$SAPTBL,.$SAPTYE
437             .MCALL .$ACT11,.$HEADER,.$SWRHI,.$REPORT,.$TYPNAM,.$PUSH,.$POP,.$STRAP
438             .MCALL .$READ,.$STYPOCT
439             .MCALL .EQUIV
440             .TITLE CZM98EO M9312/1144 KT24 UBI 800T
                :*COPYRIGHT (C) 1982
                :*DIGITAL EQUIPMENT CORP.
                :*MAYNARD, MASS. 01754
                :*
                :*PROGRAM BY DAN MILLEVILLE
                :*
                :*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                :*PACKAGE (MAINDEC-11-DZQAC-C5), JAN. 1981.
                :*
006001          $TN=1
160000          $$SWR=160000      ;;HALT ON ERROR, LOOP ON TEST, INHIEIT ERROR TYP0UT
441             .SBTTL OPERATIONAL SWITCH SETTINGS
                :*
                :*          SWITCH          USE
                :*          -----          -----
                :*          15          HALT ON ERROR
                :*          14          LOOP ON TEST
                :*          13          INHIBIT ERROR TYPEOUTS
442             .SBTTL BASIC DEFINITIONS
                :*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100          STACK= 1100
104000          ERROR=EMT
000004          SCOPE=IOT
                :*MISCELLANEOUS DEFINITIONS
000G11          HT= 11          ;;CODE FOR HORIZONTAL TAG
000012          LF= 12          ;;CODE FOR LINE FEED
000015          CR= 15          ;;CODE FOR CARRIAGE RETURN
000200          CRLF= 200       ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776          PS= 177776     ;;PROCESSOR STATUS WORD
177777          PSW=PS
177774          STKLMT= 177774  ;;STACK LIMIT REGISTER
177772          PIRQ= 177772    ;;PROGRAM INTERRUPT REQUEST REGISTER
177570          DSWR= 177570    ;;HARDWARE SWITCH REGISTER
177570          DDISP= 177570   ;;HARDWARE DISPLAY REGISTER
                :*GENERAL PURPOSE REGISTER DEFINITIONS
000000          R0= X0          ;;GENERAL REGISTER
000001          R1= X1          ;;GENERAL REGISTER
000002          R2= X2          ;;GENERAL REGISTER
000003          R3= X3          ;;GENERAL REGISTER
000004          R4= X4          ;;GENERAL REGISTER
000005          R5= X5          ;;GENERAL REGISTER
000006          R6= X6          ;;GENERAL REGISTER
000007          R7= X7          ;;GENERAL REGISTER

```

CZM98EC M9312/1144 KT24 L81 B00 MALPO M1115 22-JAN-82 09:18 PAGE 4-1  
 BASIC DEFINITIONS

```

000006      SP=      X6          ;;STACK POINTER
000007      PC=      X7          ;;PROGRAM COUNTER
000000      ;;*PRIORITY LEVEL DEFINITIONS
000000      PR0=     0           ;;PRIORITY LEVEL 0
000040      PR1=    40          ;;PRIORITY LEVEL 1
000100      PR2=   100         ;;PRIORITY LEVEL 2
000140      PR3=   140         ;;PRIORITY LEVEL 3
000200      PR4=   200         ;;PRIORITY LEVEL 4
000240      PR5=   240         ;;PRIORITY LEVEL 5
000300      PR6=   300         ;;PRIORITY LEVEL 6
000340      PR7=   340         ;;PRIORITY LEVEL 7
000000      ;;* SWITCH REGISTER" SWITCH DEFINITIONS
000000      SW15=  10000
000000      SW14=   4000
000000      SW13=  20000
0010000     SW12=  10000
004000     SW11=   4000
002000     SW10=  2000
001000     SW09=  1000
000400     SW08=   400
000200     SW07=   200
000100     SW06=   100
000040     SW05=   40
000020     SW04=   20
000010     SW03=   10
000004     SW02=    4
000002     SW01=    2
000001     SW00=    1
000000     SW9=SW09
000400     SW8=SW08
000200     SW7=SW07
000100     SW6=SW06
000040     SW5=SW05
000020     SW4=SW04
000010     SW3=SW03
000004     SW2=SW02
000002     SW1=SW01
000001     SW0=SW00
000000      ;;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000     BIT15= 100000
040000     BIT14=  40000
020000     BIT13=  20000
010000     BIT12=  10000
004000     BIT11=   4000
002000     BIT10=   2000
001000     BIT09=   1000
000400     BIT08=   400
000200     BIT07=   200
000100     BIT06=   100
000040     BIT05=   40
000020     BIT04=   20
000010     BIT03=   10
000004     BIT02=    4
000002     BIT01=    2
000001     BIT00=    1
001000     BIT9=BIT09
000400     BIT8=BIT08

```

BASIC DEFINITIONS

000200 BIT7=BIT07  
 000100 BIT6=BIT06  
 000040 BIT5=BIT05  
 000020 BIT4=BIT04  
 000010 BIT3=BIT03  
 000004 BIT2=BIT02  
 000002 BIT1=BIT01  
 000001 BIT0=BIT00

```

;*BASIC "CPU" TRAP VECTOR ADDRESSES
ERRVEC= 4          ;; TIME OUT AND OTHER ERRORS
RESVEC= 10         ;; RESERVED AND ILLEGAL INSTRUCTIONS
TBITVEC=14        ;; "T" BIT
TRIVEC= 14        ;; TRACE TRAP
BPTVEC= 14        ;; BREAKPOINT TRAP (BPT)
IOTVEC= 20        ;; INPUT/OUTPUT TRAP (IOT) **SCCPE**
PWRVEC= 24        ;; POWER FAIL
EMTVEC= 30        ;; EMULATOR TRAP (EMT) **ERROR**
TRAPVEC=34        ;; "TRAP" TRAP
TKVEC= 60         ;; TTY KEYBOARD VECTOR
TPVEC= 64         ;; TTY PRINTER VECTOR
PIRQVEC=240       ;; PROGRAM INTERRUPT REQUEST VECTOR
    
```

\*\*\*\*\*  
 ;; PROGRAM EQUATES  
 ;\*

```

*****
;*****
APTR1 = 1
APTR2 = 2
APTR3 = 4
APTR4 = 10
CRCERR = 20
PFERR = 40
NOROME =100
SEQERR =200
CONONE =400      ;+
CONTWO=1000     ;+
DUPERR=2000     ;+
=0
    
```

.SBTTL TRAP CATCHER  
 =0

```

; *ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A "+2,HALT"
; *SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
; *LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
    
```

```

=174
DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
SWREG:   .WORD 0          ;;SOFTWARE SWITCH REGISTER
.SBTTL STARTING ADDRESS(ES)
JMP R#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
=204
JMP RSTART
    
```

.SBTTL ACT11 HOOKS  
 ;\*\*\*\*\*  
 ;HOOKS REQUIRED BY ACT11

```

$SVPC=.          ;SAVE PC
=46
$ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN $.EOP
=52
.WORD 0         ;;2)SET LOC.52 TO ZERO
.$SVPC          ;; RESTORE PC
    
```

443  
 444  
 445  
 446 000001  
 447 000002  
 448 000004  
 449 000010  
 450 000020  
 451 000040  
 452 000100  
 453 000200  
 454 000400  
 455 001000  
 456 002000  
 457 000000  
 458  
  
 000174 000174  
 000176 000000  
  
 000137 001400  
 000204 000137 002204  
 460  
 461 000204 000137 002204  
 462  
  
 000210  
 000046  
 000046 002312  
 000052  
 000052 000000  
 000010

463	001100		.=1100
464	001100	000	\$AUTOB: .BYTE 0
465	001101	000	\$INTAG: .BYTE 0
466	001102	000000	.WORD C
467	001104	177570	SWR: .WORD DSWR
468	001106	177570	DISPLAY: .WORD DDISP
469	001110	000000	ROMERR: 0
470	001112	000000	MESSAG: 0
471	001114	173000	FIRSTB: 173000
472	001116	000000	ROMFIN: 0
473	001120	000000	ERRCNT: 0
474	001122	000000	CPUTYP: 0

APT PARAMETER BLOCK

475

001124  
000024  
000024 000200  
000044  
000044 001124  
001124

001124  
001124 000000  
001126 001140  
001130 000052  
001132 000002  
001134 000000  
001136 000052

```

.SBTTL APT PARAMETER BLOCK
:*****
:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
:*****
.$X=      ;;SAVE CURRENT LOCATION
.=24     ;;SET POWER FAIL TO POINT TO START OF PRO
200      ;;FOR APT START UP
.=44     ;;POINT TO APT INDIRECT ADDRESS PNTR.
$APTHDR  ;;POINT TO APT HEADER BLOCK
.=.$X    ;;RESET LOCATION COUNTER
:*****
:SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
:INTERFACE SPEC.
$APTHDR:
$HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR: .WORD $MAIL  ;;ADDRESS OF APT MAILBOX (BITS 0-15)
$TSTM:  .WORD 2.     ;;RUN TIM OF LONGEST TEST
$PASTM: .WORD 2.     ;;RUN TIME IN SECS. OF 1ST PASS C, 1 UNIT (QUICK VERIFY)
$UNITM: .WORD 0      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR FACIAL ADDITIONAL UNIT
        .WORD $TEND-$MAIL/2 ;;LENGTH MAILBOX-STABLE (WORDS)

```

477

.SBTTL APT MAILBOX-ETABLE

```

:*****
:EVEN
001:40      000000  $MAIL:          ::APT MAILBOX
001:40      000000  $MSGTY: .WORD   ARMSGTY ::MESSAGE TYPE CODE
001:42      000000  $FATAL: .WORD   AFATAL  ::FATAL ERROR NUMBER
001:44      000000  $TESTM: .WORD   ATESTM  ::TEST NUMBER
001:46      000000  $PASS:  .WORD   APASS    ::PASS COUNT
001:50      000000  $DEVCT: .WORD   ADEVCT  ::DEVICE COUNT
001:52      000000  $UNIT:  .WORD   AUNIT    ::I/O UNIT NUMBER
001:54      000000  $MSGAD: .WORD   ARMSGAD  ::MESSAGE ADDRESS
001:56      000000  $MSGLG: .WORD   ARMSGLG  ::MESSAGE LENGTH
001:60      000000  $ETABLE:       ::APT ENVIRONMENT TABLE
001:60      000    $ENV:  .BYTE   AENV      ::ENVIRONMENT BYTE
001:61      000    $ENVM: .BYTE   AENVM     ::ENVIRONMENT MODE BITS
001:62      000000  $SUREG: .WORD   ASUREG   ::APT SWITCH REGISTER
001:64      000000  $USWR:  .WORD   AUSWR    ::USER SWITCHES
001:66      000000  $CPUOP: .WORD   ACPUOP   ::CPU TYPE,OPTIONS
:          *          BITS 15-11=CPU TYPE
:          *          11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
:          *          11/70=06,P00=07,Q=10
:          *          BIT 10=REAL TIME CLOCK
:          *          BIT 9=FLOATING POINT PROCESSOR
:          *          BIT 8=MEMORY MANAGEMENT
001:70      000    $MAMS1: .BYTE   AMAMS1   ::HIGH ADDRESS,M.S. BYTE
001:71      000    $MTYP1: .BYTE   ARTYP1  ::MEM. TYPE,BLK#1
:          *          MEM. TYPE BYTE -- (HIGH BYTE)
:          *          900 NSEC CORE=001
:          *          300 NSEC BIPOLAR=002
:          *          500 NSEC MOS=003
001:72      000000  $MADR1: .WORD   AMADR1   ::HIGH ADDRESS,BLK#1
:          *          MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE
001:74      000    $MAMS2: .BYTE   AMAMS2   ::HIGH ADDRESS,M.S. BYTE
001:75      000    $MTYP2: .BYTE   ARTYP2  ::MEM. TYPE,BLK#2
001:76      000000  $MADR2: .WORD   AMADR2   ::MEM.LAST ADDRESS,BLK#2
001:80      000    $MAMS3: .BYTE   AMAMS3   ::HIGH ADDRESS,M.S. BYTE
001:81      000    $MTYP3: .BYTE   ARTYP3  ::MEM. TYPE,BLK#3
001:82      000000  $MADR3: .WORD   AMADR3   ::MEM.LAST ADDRESS,BLK#3
001:84      000    $MAMS4: .BYTE   AMAMS4   ::HIGH ADDRESS,M.S. BYTE
001:85      000    $MTYP4: .BYTE   ARTYP4  ::MEM. TYPE,BLK#4
001:86      000000  $MADR4: .WORD   AMADR4   ::MEM.LAST ADDRESS,BLK#4
001:90      000000  $SVECT1: .WORD   AVECT1  ::INTERRUPT VECTOR#1,BUS PRIORITY#1
001:92      000000  $SVECT2: .WORD   AVECT2  ::INTERRUPT VECTOR#2,BUS PRIORITY#2
001:94      000000  $BASE:  .WORD   ABASE    ::BASE ADDRESS OF EQUIPMENT UNDER TEST
001:96      000000  $DEVN:  .WORD   ADEVN    ::DEVICE MAP
001:98      000000  $CDW1:  .WORD   ACDW1    ::CONTROLLER DESCRIPTION WORD#1
001:99      000000  $CDW2:  .WORD   ACDW2    ::CONTROLLER DESCRIPTION WORD#2
001:100     000000  $DDW0:  .WORD   ADDW0    ::DEVICE DESCRIPTOR WORD#0
001:101     000000  $DDW1:  .WORD   ADDW1    ::DEVICE DESCRIPTOR WORD#1
001:102     000000  $DDW2:  .WORD   ADDW2    ::DEVICE DESCRIPTOR WORD#2
001:103     000000  $DDW3:  .WORD   ADDW3    ::DEVICE DESCRIPTOR WORD#3
001:104     000000  $DDW4:  .WORD   ADDW4    ::DEVICE DESCRIPTOR WORD#4
001:105     000000  $DDW5:  .WORD   ADDW5    ::DEVICE DESCRIPTOR WORD#5
001:106     000000  $DDW6:  .WORD   ADDW6    ::DEVICE DESCRIPTOR WORD#6
001:107     000000  $DDW7:  .WORD   ADDW7    ::DEVICE DESCRIPTOR WORD#7
001:108     000000  $DDW8:  .WORD   ADDW8    ::DEVICE DESCRIPTOR WORD#8
001:109     000000  $DDW9:  .WORD   ADDW9    ::DEVICE DESCRIPTOR WORD#9

```

001250 000000  
001252 000000  
001254 000000  
001256 000000  
001260 000000  
001262 000000  
001264

\$DDW10: .WORD ADDW10 ::DEVICE DESCRIPTOR WORD#10  
\$DDW11: .WORD ADDW11 ::DEVICE DESCRIPTOR WORD#11  
\$DDW12: .WORD ADDW12 ::DEVICE DESCRIPTOR WORD#12  
\$DDW13: .WORD ADDW13 ::DEVICE DESCRIPTOR WORD#13  
\$DDW14: .WORD ADDW14 ::DEVICE DESCRIPTOR WORD#14  
\$DDW15: .WORD ADDW15 ::DEVICE DESCRIPTOR WORD#15  
\$ETEND:

MFPT::000007

478 000007

```

479          .=1400
480 001400    001400
          START:
          .SBTTL INITIALIZE THE COMMON TAGS
001400 012706 001100      MOV #STACK,SP          ;;SETUP THE STACK POINTER
          ;;INITIALIZE A FEW VECTORS
001404 012737 007114 000034      MOV #STRAP,@TRAPVEC    ;;TRAP VECTOR FOR TRAP CALL
001412 012737 000340 000036      MOV #340,@TRAPVEC+2;LEVEL 7
001420 005037 001146          CLR $PASS              ;;CLEAR THE PASS COUNT
001424 013737 002266 002260      MOV $ENDCT,$EOPTCT    ;;SETUP END-OF-PROGRAM COUNTER
          ;;SIZE FOR A HARDWARE SWITCH REGISTER, IF NOT FOUND OR IT IS
          ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
001432 013746 000404          MOV @ERRVEC,-(SP)     ;;SAVE ERROR VECTOR
001436 012737 001472 000004      MOV #64,@ERRVEC      ;;SET UP ERROR VECTOR
001444 012737 177570          MOV #DSWR,SWR        ;;SETUP FOR A HARDWARE SWICH REGISTER
001452 012737 177570          MOV #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
001460 022777 177777          CMP #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
001466 001012          BNE          ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
          ;;AND THE HARDWARE SWR IS NOT = -1
001470 000403          BR 65$          ;;BRANCH IF NO TIMEOUT
001472 012716 001500          MOV #65$,(SP)        ;;SET UP FOR TRAP RETURN
001476 000002          RTI
001500 012737 000176 001104          MOV #SWREG,SWR      ;;POINT TO SOFTWARE SWR
001504 012737 000174 001106          MOV #DISPREG,DISPLAY
001514 012637 000004          MOV (SP)+,@ERRVEC  ;;RESTORE ERROR VECTOR
001520 005037 001146          CLR $PASS            ;;CLEAR PASS COUNT
001524 132737 000200 001161          BITB #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
001532 001403          BEQ 67$          ;;YES,USE NON-APT SWITCH
001534 012737 001162 001104          MOV #SSWREG,SWR   ;;NO,USE APT SWITCH REGISTER
001542
481
          67$:
          .SBTTL TYPE PROGRAM NAME
          ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
001542 005227 177777          INC #-1              ;;FIRST TIME?
001546 001051          BNE 68$          ;;BRANCH IF NO
001550 022737 002312 000042          CMP $ENDAD,@#42   ;;ACT-11?
001556 001445          BEQ 63$          ;;BRANCH IF YES
001560 104401 001626          TYPE 65$          ;;TYPE ASCIZ STRING
          .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
001564 005737 000042          TST #42          ;;ARE WE RUNNING UNDER XXDP/ACT?
001570 001012          BNE 70$          ;;BRANCH IF YES
001572 123727 001160 000001          CMPB $ENV,#1     ;;ARE WE RUNNING UNDER APT?
001600 001406          BEQ 70$          ;;BRANCH IF YES
001602 023727 001104 000176          CMP SWR,#SWREG   ;;SOFTWARE SWITCH REG SELECTED?
001610 001005          BNE 71$          ;;BRANCH IF NO
001612 104405          GTSWR          ;;GET SOFT-SWR SETTINGS
001614 000403          BR 71$
001616 12737 000001 001100          MOVB #1,$AUTOB   ;;SET AUTO-MODE INDICATOR
001624          BEQ 71$
001624 000422          BR 68$
          ;;69$:
001672          .ASCIZ <CRLF>*CZM980 M9312/1144 KT24 UBI BOOT*<CRLF>
          68$:
482 001672 013746 000010          MOV -(SP)          ;; SAVE VECTOR ON STACK
483 001676 012737 001744 000010          MOV #10$,10      ;; SET ERROR VECTOR
484 001704 005037 001122          CLP CPU1TYPE    ;; CLEAR CPU TYPE LOCATION      :DPM001
485 001710 000007          MFPT           ;; WHAT CPU IS IT
486 001712 110037 001122          MOVB RC,CPUTYP  ;; SAVE CPU TYPE IN 'CPUTYP'    :DPM001
487 001716 122737 000001 001122          CMPB #1,CPUTYP  ;; ** 11/44??                  :DPM001
488 001724 001012          BNE 12$          ;; NO USE M9312 ROM NUMBERS

```



```

489 001726 012737 013106 013104      MOV      #MESS44,MESSPT      **: YES USE UBI ROM NUMBERS
490 001734 012737 007306 007176      MOV      #BROM,ROMPTR     **: UBI ROM NUMBERS
491 001742 000403                    BR       #2$              **: BR IF NO TRAP
492 001744 012716 001776          10$:    MOV      #15$, (SP)      :SET UP RETURN VECTOR
493 001750 000002                    RTI                       :RETURN
494 001752 122737 003003 001122 12$:    CMPSB  #3,CPUTYP         **: 11/24? :DPM001
495 001760 001006                    BNE     #1$              **: NO USE M9312 ROM NUMBERS :DPM001
496 001762 012737 013116 013104      MOV      #MESS24,MESSPT   **: YES USE 11/24 UBI ROM NUMS :DPM001
497 001770 012737 007320 007176      MOV      #BROM,ROMPTR     **: UBI 11/24 ROM NUMBERS :DPM001
498 001776 012637 000010          15$:    MO      (SP)+,10         :RESTORE ERROR VECTOR
499 002002 012700 010502          8$:    MOV      #BUF1,RO        :CLR SIZING BUFFERS
500 002006 005020                    CLR     RO+              :
501 002010 029077 010566                    CMP     RO,#OCTBUF       :HAVE WE CLEARED THE WHOLE
502 002014 002774                    BLT     #5$              :BUFFER,NO,GO BACK
503 002016 005037 001116          8$:    CLR     ROMFIN          :INITIALIZE ROM FOUND INDICATOR$
504 002022 005037 004342          8$:    CLR     TIMES          :INITIALIZE ENTRY COUNTER FOR PUTMES SUB.
505 002026 004737 011676                    JC      PC,EAR           :+CLEAR SOME MORE LOCATIONS
506 002032 013746 000004          8$:    MOV      ERRVEC,-(R6)     :SAVE CONTENTS OF LOCATION 4
507 002036 012737 002100 000004      MOV      #2$,ERRVEC       :+SET UP FOR POSSIBLE TRAP
508 002044 122737 177777 165000      CMPSB  #-1,165000        :IF CONTENTS = -1, OR TRAP
509 002052 001414                    BEQ     #3$              :THEN CPU ROM NOT PLUGGED IN
510 002054 012700 165000          1$:    MOV      #165000,RO      :GET FIRST ADDRESS OF ROM SPACE
511 002060 005720                    TST     RO+              :IF THIS INSTRUCTION TRAPS CPU ROM NOT PLUGGED IN
512 002062 023027 166000          1$:    CMP     RO,#166000      :HAVE WE CHECKED ALL ADDRESSES
513 002066 001374                    BNE     #1$              :IF NO THEN CONTINUE LOOPING
514 002070 012737 000001 001116          1$:    MOV      #1,ROMFIN      :IF NO TRAPS OR LOW BYTE OF FIRST
515                                     :ADDRESS NOT EQUAL -1 ASSUME ROM PRESENT
516 002076 000402                    BR      #3$              :NO TRAP JUST RESTORE ERRVEC
517 002100 062706 000004          2$:    ADD     #4,R6           :+IF TRAP CLEAN OFF PC, PSW
518 002104 012637 000004          3$:    MOV      (R6)+,ERRVEC    :RESTORE ERRVEC
519 002110 012700 000002          3$:    MOV      #2,RO         :INITIALIZE ROM FOUND INDICATOR
520 002114 013737 001114 003644      MOV      #FIRSTB,TESTAD   :GET FIRST BOOT
521 002122 132777 000200 001514 5$:    BITB   #230,TESTAD       :IF LOW BYTE HAS BIT 7 SET
522 002130 001411                    BEQ     #7$              :DO NOT SET ROM FOUND INDICATOR
523 002132 022777 177776 001504      CMP     #-2,TESTAD       :UNLESS, CONTENTS OF ADDRESS
524 002140 001011                    BNE     #4$              :EQUAL TO -2.(CONTINUATION ROM)
525 002142 050037 013070          4$:    BIS     RO,CONFIN       :+SET CONTINUATION ROM FOUND INDICATOR
526 002146 050037 001116          4$:    BIS     RO,ROMFIN      :+SET ROM FOUND INDICATOR
527 002152 000404                    BR      #4$              :+CONTINUE
528 002154 050037 001116          7$:    BIS     RO,ROMFIN      :SET ROM FOUND INDICATOR
529 002160 050037 013072          7$:    BIS     RO,DEVFIN      :+SET DEVICE ROM FOUND INDICATOR
530 002164 062737 000200 003644 4$:    ADD     #20,TESTAD       :UPDATE TEST ADDRESS TO NEXT ROM
531 002172 006100                    ROL     RO               :UPDATE ROM FOUND INDICATOR
532 002174 022737 174000 003644      CMP     #174000,TESTAD   :HAVE CHECKED ALL THE ROMS
533 002202 001347                    BNE     #5$              :IF NO CONTINUE CHECKING
534 002204 005737 001116          RSTART: TST     ROMFIN      :ARE THERE ANY ROMS AVAILABLE
535 002210 001003                    BNE     #6$              :IF YES GO TO ROM TEST
536 002212 0037 002544          6$:    JSR     PC,NOROMS      :IF NO GO TEST NO ROMS
537 002216 0037 002544          6$:    BR      #EOP           :GO TO END OF PASS
538 002220 004757 002632          6$:    JSR     PC,CHECKS     :GO CALCULATE CHECKSUMS
539 002224 005737 001146          6$:    TST     $PASS         :ARE WE ON 1ST PASS
540 002230 001004                    BNE     #EOP           :IF NO SKIP PROCESS OF PARAMETERS
541 002232 00737 003146          6$:    JSR     PC,PROMP      :GO PROCESS ROM PARAMETERS
542 002236 004737 012174          6$:    JSR     PC,SEQTST     :GO CHECK SEQUENCE OF DEVICE CODES

```

543

```

.SBTTL END OF PASS ROUTINE
:*****
:INCREMENT THE PASS NUMBER ($PASS)
:*TYPE 'END PASS'
:*IF THERES A MONITOR GO TO IT
:*IF THERE ISN'T JUMP TO RSTART
:*IF IT IS DESIRED TO HAVE A BELL INDICATE THE 'END OF PASS' LOCATION
:*SENDMG CAN BE CHANGED TO 7.
$EOP:
002242      000240      NOP
002242      005237      001146      INC      $PASS      ::INCREMENT THE PASS NUMBER
002250      042737      100000      001.46      BIC      #1J0000,$PASS  ::DON'T ALLOW A NEG. NUMBER
002256      005327      DEC      (PC)+      ::LOOP?
002260      000001      $EOPCT- .WORD      1
002262      003017      SGT      $D0AGN      ::YES
002264      012737      MOV      (PC)+,@(PC)+  ::RESTORE COUNTER
002266      000001      SENDCT: .WORD      1
002270      002260      $EOPCT
002272      104401      002331      TYPE      ,SENDMG      ::TYPE 'END PASS'
002276      104401      002326      TYPE      ,SEMULL      ::TYPE A NULL CHARACTER
002302      013700      000042      $GET42: MOV      @#42,RO  ::GET MONITOR ADDRESS
002306      001405      BEQ      $D0AGN      ::BRANCH IF NO MONITOR
002310      000005      RESET      ::CLEAR THE WORLD
002312      004710      $ENDAD: JSR      PC,(RO)  ::GO TO MONITOR
002314      000240      NOP      ::SAVE ROOM
002316      000240      NOP      ::FOR
002320      000240      NOP      ::ACT11
002322      $D0AGN:
002322      000137      JMP      @(PC)+      ::RETURN
002324      002204      $RTNAD: .WORD      RSTART
002326      377      377      000      $EMULL: .BYTE      -1,-1,0  ::NULL CHARACTER STR:::
002331      015      012      105      $ENDMG: .ASCII <15><12>/END PASS/
  
```



```

550
551
552
553
554
555
556 002544 104401 007350
557 002550 012703 172000
558 002554 013704 002630
559 002560 121304
560 002562 001406
561 002564 011304
562 002566 052737 000100 001110
563 002574 004337 004344
564 002600 062703 000002
565 002604 022703 173024
566 002610 001773
567 002612 022703 173224
568 002616 001770
569 002620 022703 174000
570 002624 003353
571 002626 000207
572
573 002630 000377
  
```

```

.SBTTL NO ROMS TEST
:THIS ROUTINE IS CALLED ONLY IF NO ROMS WERE FOUND
:DURING SIZING. IT DOES A READ OF ALL BOOTSTRAP ROM
:ADDRESSES AND COMPARES THE CONTENTS TO A KNOWN
:EXPECTED VALUE.
  
```

```

NOROMS: TYPE .NOROM
          MOV #172000,R3      :- TYPE MESSAGE THAT NOROMS FOUND
1$:      MOV NODATA, R4      :- GET FIRST ADDRESS TO BE READ
          CMPB (R3),R4      :- GET ADDRESS OF EXPECTED VALUE
          BEQ 2$            :- DOES RECEIVED VALUE EQUAL EXPECTED
          MOV (R3),R4       :- IF YES CONTINUE TESTING
          BIS #NOROME,ROMERR :- GET BAD VALUE
          JSR PC,ERRHAN     :- IF NO SET ERROR INDICATER
2$:      ADD #2,R3          :- REPORT ERROR
          CMP #173024,R3    :- GET TO NEXT EVEN ADDRESS
          BEQ 2$            :- AT A VECTOR ADDRESS?
          CMP #173224,R3    :- BRANCH IF YES
          BEQ 2$            :- AT A VECTOR ADDRESS?
          CMP #174000,R3    :- BRANCH IF YES
          BGT 1$            :- HAVE ALL ADDRESSES BEEN TESTED
          RTS PC           :- IF NO GO TEST THIS ADDRESS

NODATA: .WORD 377          :- A HOLE LOOKS LIKE XXX377
  
```

```

574
575
576
577 002632 012737 000001 003046 CHECKS: MOV #1,ROMCNT :INITIALIZE ROM COUNTER
578 002640 033737 003046 001116 BIT ROMCNT,ROMFIN :IS DIAGNOSTIC ROM PRESENT
579 002646 001424 BEQ 1$ :IF NO GO CHECKSUM BOOT ROMS
580 002650 012737 165775 003144 MOV #16<775,LASTAD :SET UP LAST ADDRESS TO BE SUMMED
581 002656 012737 000000 003142 MOV #0,EXCADD :SET UP EXCEPTION ADDRESS
582 002664 012737 165000 003140 MOV #165000,FIRSTA :SET UP FIRST ADDRESS TO BE SUMMED
583 002672 004737 003050 JSR PC,CALSUM :GO CALCULATE CHECKSUM
584 002676 013703 165776 MOV 165776,R3 :GET EXPECTED CHECKSUM
585 002702 020304 CMP R3,R4 :COMPARE CHECKSUMS
586 002704 001405 BEQ 1$ :IF CHECKSUMS COMPARE CONTINUE
587 002706 052737 000020 001110 BIS #CRCERR,ROMERR :IF ERRORS SET INDICATOR
588 002714 004737 004344 JSR PC,ERRHAN :REPORT ERROR
589 002720 012737 173000 003140 1$: MOV #173000,FIRSTA :SET UP FIRST ADDRESS TO BE SUMMED
590 002726 012737 173024 003142 MOV #173024,EXCADD :SET UP EXCEPTION ADDRESS
591 002734 012737 173175 003144 MOV #173175,LASTA :SET UP LAST ADDRESS TO BE SUMMED
592 002742 012702 173176 MOV #173176,R2 :GET ADDRESS OF GOOD DATA
593 002746 006137 00304 2$: ROL ROMCNT :UPDATE ROM COUNTER
594 002752 033737 003046 001116 BIT ROMCNT,ROMFIN :IS ROM PRESENT
595 002760 001412 BEQ 3$ :IF NO, GO UPDATE TO NEXT ROM
596 002762 004737 003050 JSR PC,CALSUM :IF YES GO CALCULATE CHECKSUM
597 002766 011203 MOV (R2),R3 :GET EXPECTED CHECKSUM
598 002770 020304 CMP R3,R4 :COMPARE CHECKSUMS
599 002772 001405 BEQ 3$ :IF CHECKSUMS EQUAL CONTINUE
600 002774 052737 000020 001110 BIS #CRCERR,ROMERR :IF ERROR SET INDICATOR
601 003002 004737 004344 JSR PC,ERRHAN :REPORT ERROR
602 003006 062737 000200 003140 3$: ADD #200,FIRSTA :UPDATE FIRST ADDRESS
603 003014 062737 000200 003142 ADC #200,EXCADD :UPDATE EXCEPTION ADDRESS
604 003022 062737 000200 003144 ADD #200,LASTAD :UPDATE LAST ADDRESS
605 003030 062702 000200 ADD #200,R2 :UPDATE ADDRESS OF GOOD DATA
606 003034 022737 174000 003140 CMP #174000,FIRSTA :HAVE ALL ROMS BEEN CHECKED
607 003042 001341 RNE 2$ :IF NO GO CHECKSUM NEXT ONE
608 003044 000207 RTS PC :IF YES RETURN
609
610 003046 000000 ROMCNT: 0
    
```

CALCULATE CHECKSUMS MODULE

.SBTTL CALCULATE CHECKSUMS MODULE  
 :THIS ROUTINE CALCULATES THE CRC16 CHECKSUM OF THE CONTENTS  
 :OF EVERY LOCATION EXCEPT THE EXCEPTION ADDRESS AND LAST ADDRESS  
 :OF EVERY ROM

611  
 612  
 613  
 614  
 615  
 616 003050 010246  
 617 003052 013700 003140  
 618 003056 005004  
 619 003060 012005  
 620 003062 012702 000020  
 621 003066 000241  
 622 003070 006004  
 623 003072 006005  
 624 003074 102006  
 625 003076 012701 120001  
 626 003102 040401  
 627 003104 042704 120001  
 628 003110 050104  
 629 003112 005302  
 630 003114 003364  
 631 003116 020037 003142  
 632 003122 001001  
 633 003124 005720  
 634 003126 020037 003144  
 635 003132 101752  
 636 003134 012602  
 637 003136 000207  
 638  
 639 003140 000000  
 640 003142 000000  
 641 003144 000000

```

CALSUM: MOV R2, -(SP) ;SAVE R2
        MOV FIRSTA, R0 ;GET STARTING ADDRESS
        CLR R4 ;INITIALIZE CRC WORD
LCOP: MOV (R0)+, R5 ;GET A BYTE
      MOV #16., R2 ;SET BYTE COUNT
CRCLOP: CLC ;THE NEXT NINE LINES
        ROR R4 ;DO THE MATH CALCULATIONS
        ROR R5
        BVC 1$
        MOV #120001, R1
        BIC R4, R1
        BIC #120001, R4
        BIS R1, R4
1$: DEC R2
     BGT CRCLOP
     CMP R0, EXCADD ;IS NEXT ADDRESS AN EXCEPTION ADDRESS
     BNE 2$ ;IF NO TEST IT
     TST (R0)+ ;IF YES SKIP ADDRESS
     CMP R0, LASTAD ;HAVE ALL LOCATIONS BEEN SUMMED
     BLOS LOOP ;IF NO CONTINUE
     MOV (SP)+, R2 ;RESTORE R2
     RTS PC ;IF YES RETURN
    
```

FIRSTA: 0  
 EXCADD: 0  
 LASTAD: 0

PROCESS ROM PARAMETERS MODULE

.SBTTL PROCESS ROM PARAMETERS MODULE  
 ;THIS ROUTINE DETERMINES IF SIZING IS TO BE DONE. IF IT IS THE  
 ;MODULE WILL GET THE PARAMETERS FROM THE DEVCOD AND PPFVAR  
 ;MODULES AND ASSEMBLE MESSAGES TO BE OUTPUT. IF SIZING IS  
 ;NOT TO BE DONE THIS MODULE WILL TAKE THE PARAMETERS RECEIVED  
 ;AND COMPARE THEM TO THE VALUES SUPPLIED IN THE ETABLE.

```

642
643
644
645
646
647
648
649 003146 105737 001161 PROMP: TSTB $ENVM ;IF BIT IS CLEAR SIZE
650 003152 100424 BMI 1$ ;IF BIT IS SET DON'T SIZE.
651 003154 004737 JSR PC,DEVCOD ;CALL DEVICE CODE MODULE
652 003160 004737 JSR PC,PUMTES ;FORMAT PARAMETER MESSAGE
653 003164 010576 MES1
654 003166 004737 JSR PC,PPFVAR ;GO GET PSEUDO POWER-FAIL VECTOR ADDRESS
655 003172 032737 0000'0 001110 BIT #PFERR,ROMERR ;WAS THERE AN ERR
656 003200 001403 BEQ 10$ ;IF NO GO FORMAT MESSAGE
657 003202 004737 JSR PC,ERRHAN ;IF YES GO REPORT MESSAGE
658 003206 000506 BR 9$
659 003210 004737 10$: JSR PC,PUMTES ;FORMAT PSEUDO POWER-FAIL VECTOR
660 003214 011176 MES2 ;ADDRESS MESSAGE
661 003216 005237 001112 INC MESSAG ;SET MESSAGE INDICATOR
662 003222 000500 BR 9$ ;GO TO EXIT
663 003224 004737 1$: JSR PC,DEVCOD ;CALL GET DEVICE CODE MODULE
664 003230 012703 MOV #BUF1+2,R3 ;GET BOARD DEVICE CODES ;DPM001
665 003234 012704 2$: MOV $DDW0,R4 ;GET ETABLE DEVICE CODE PARAMETERS
666 003240 012402 3$: MOV (R4)+,R2 ;GET ETABLE DEVICE CODE
667 003242 020213 CMP R2,(R3) ;DO THE TWO DEVICE CODES COMPARE
668 003244 001407 BEQ 4$ ;IF YES GET NEXT BOARD DEVICE CODE
669 003246 005714 TST (R4) ;IF NO HAVE WE CHECKED THE WHOLE ETABLE
670 003250 001373 BNE 3$ ;IF WE HAVEN'T CHECK NEXT ETABLE ENTRY
671 003252 052737 000001 001110 BIS #APTER1,ROMERR ;IF WE HAVE THEN DEVICE CODE ON BOARD
672 003260 004737 JSR PC,ERRHAN ;DOES NOT EXIST IN ETABLE
673 003264 062703 4$: ADD #4,R3 ;UPDATE TO NEXT ADDRESS
674 003270 005713 TSF (R3) ;IF CONTENTS EQUALS ZERO WE'RE DONE
675 003272 001362 BNE 3$ ;IF CONTENTS NO EQUAL ZERO CONTINUE
676 003274 012703 001224 MOV $DDW0,R3 ;GET BOARD DEVICE CODES
677 003300 012704 010500 MOV #BUF1-2,R4 ;GET ETABLE DEVICE CODES
678 003304 062704 6$: ADD #4,R4 ;GET BOARD DEVICE CODE ;DPM001
679 003310 021314 CMP (R3),(R4) ;DO THE TWO DEVICE CODES COMPARE
680 003312 001407 BEQ 7$ ;IF YES GET NEXT ETABLE DEVICE CODE
681 003314 005714 TST (R4) ;IF NO HAVE WE CHECKED ALL THE BOARD
682 003316 001372 PNE 6$ ;DEVICE CODES, IF NO CONTINUE
683 003320 052737 000002 001110 BIS #APTER2,ROMERR ;IF YES ETABLE DEVICE CODE NOT
684 003326 004737 JSR PC,ERRHAN ;NOT ON BOARD
685
686 003332 062703 7$: ADD #2,R3 ;INCREMENT TO NEXT $DDW LOCATION
687 003336 022703 CMP $DDW12,R3 ;HAVE WE CHECKED ALL 12 POSITIONS ;DPM001
688 003342 001360 BNE 6$ ;IF NOT, CONTINUE
689 003344 004737 JSR PC,PPFVAR ;GO GET PSEUDO POWER-FAIL VECTOR ADDRESS
690 003350 013704 MOV $BASE,R4 ;GET ETABLE PPFVA
691 003354 013703 BUF2,R3 ;GET BOARD PPFVA
692 003360 020403 CMP R4,R3 ;DO THE TWO PARAMETERS COMPARE
693 003362 001405 BEQ 3$ ;IF YES CONTINUE
694 003364 052737 000004 001110 BIS #APTER3,ROMERR ;SET APT ERROR INDICATOR
695 003372 004737 JSR PC,ERRHAN ;GO TO ERROR ROUTINE
696 003376 013704 8$: MOV $CDW1,R4 ;GET ETABLE DATA
697 003402 013703 MOV BUF2+2,R3 ;GET BOARD DATA
698 003406 020403 CMP R4,R3 ;DO THE TWO PARAMETER COMPARE
    
```

699	003410	001405			BEQ	9%		:IF YES THEN DONE
700	003412	052737	000010	001110	BIS	#APTER4,RCMERR		:SET APT ERROR INDICATOR
701	003420	004737	004344		JSR	PC,ERRHAN		:GO TO ERROR ROUTINE
702	003424	000207		9%	RTS	PC		



GET DEVICE CODES MODULE

703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759

```

.SBTL GET DEVICE CODES MODULE
:THIS SUBROUTINE LOCATES EACH DEVICE CODE AND PASSES IT AND THE
:ADDRESS IN WHICH IT WAS FOUND BACK TO THE CALLING ROUTINE.
:DATA IS STORED IN BUFFER 'BUF1' IN THIS FORMAT:

      BUF1:  ADDRESS OF FIRST DEVICE CODE
             DEVICE CODE
             ADDRESS OF SECOND DEVICE CODE
             DEVICE CODE
             .
             .
             .
             ADDRESS OF NTH DEVICE CODE
             DEVICE CODE

+CONTINUATION ROM DATA IS STORED IN 'BUF1' AS FOLLOWS:
+
+
+CONTINUATION CHIP IDENTIFIER (ALWAYS A #1)
+DEVICE CODE (PREVIOUS ROM'S DEVICE CODE,OR -; IF ILLEGAL ROM)
+
+
:IF DIAGNOSTIC ROM PRESENT IT WILL BE THE FIRST DEVICE CODE.
:THERE IS ROOM FOR UP TO 12 DEVICE CODES IN TEMP
DEVCOD: MOV #BUF1,R0
        MOV #1,R1          :INITIALIZE ROM POINTER
        BIT R1,ROMFIN      :IS DIAGNOSTIC ROM PRESENT
        BEQ $S             :IF NO GO TEST BOOT ROMS
        MOV #165774,TESTAD :GET ADDRESS OF DI.G. ROM DEVICE CODE.
        MOV TESTAD,(R0)+   :STORE ADDRESS OF DEVICE CODE
        MOV @TESTAD,(R0)+ :STORE DEVICE CODE
        CLR DAFLAG         :CLR DATA TYPE FLAG
        MOV FIRSTB,TESTAD :GET ADDRESS OF FIRST BOOT ROM
        ROL R1             :UPDATE POINTER TO NEXT ROM
        BIT R1,DEVFIN      :+IS DEVICE ROM PRESENT
        BNE $S            :IF YES GO GET PARAMETERS
        JSR PC,CON1        :+CHECK FOR CONTINUATION CHIP
        ADD #200,TESTAD    :IF NO UPDATE TEST ADDRESS
        ROL R1             :UPDATE POINTER TO NEXT ROM
        BR 9$             :GO SEE IF ALL ROM CHECKED
        TST DAFLAG        :IF DATAFLAG=0 THEN DATA IS DEVICE CODE
        BNE $S            :IF DATAFLAG=1 THE DATA IS OFFSET TO NEXT
        MOV TESTAD,(R0)+  :STORE ADDRESS OF DEVICE CODE
        MOV @TESTAD,(R0)+ :STORE DEVICE CODE
        ADD #2,TESTAD     :UPDATE TEST ADDRESS
        BR 8$             :GET OVER SOME CODE
        MOV TESTAD,R2     :SAVE OLD TEST ADDRESS
        ADD @TESTAD,TESTAD :UPDATE TO NEW TEST ADDRESS
        MOV TESTAD,R3     :GET THE NEW ADDRESS
        BIC #1,TESTAD     :MAKE SURE ADDRESS IS EVEN
        BIC #170177,R2    :+SAVE ONLY BITS 7,8,9,10,11
        BIC #170177,R3    :+IN R3 ALSO
        SUB R2,R3         :CALCULATE DISTANCE BETWEEN ADD
        TST R3            :IS R3 EQUAL TO 0
    
```

:DPM001

760	003610	001'06				BEQ	%\$		:IF YES THEN DONE
761	003612	16'03	009200			SUB	#200,R3		:IF NO THEN MOVE POINTER
762	003616	006'01				ROL	R1		:ONE BIT FOR EVERY 200
763	003620	004'37	011742			JSR	PC,CON2		:+CHECK FOR CONTINUATION CHIP
764	003624	006'79				BR	6\$		
765	003626	005'37	003646	8\$:		COM	DAFLAG		:CHANGE DATA FLAG TO RIGHT DATA TYPE
766	003632	023727	003644	7\$:		CMP	TESTAD,#174000		:HAVE WE CHECKED ALL THE ROM
767	003640	002716				BLT	3\$		:IF NO CONTINUE
768	003642	000207				RTS	PC		:IF YES RETURN
769									
770	003644	000000				TESTAD:	0		
771	003646	000000				DAFLAG:	0		

GET PSEUDO POWER-FAIL VECTOR ADDRESS ROUTINE

```

.SBTTL GET PSEUDO POWER-FAIL VECTOR ADDRESS ROUTINE
:THIS SUBROUTINE TESTS LOCATIONS 173024 AND 173224 TO DETERMINE
:WHICH VECTOR WILL BE USED IF POWER-FAIL OPTION ENABLED ON THE
:BOARD. THE OPTION MUST BE ENABLED AND AT LEAST ONE ADDRESS
:SWITCH MUST BE 'ON' OR AN ERROR WILL BE DETECTED.
:THE DATA WILL BE RETURNED IN 'BUF2' IN THE FORMAT.

```

```

:
:   BUF2:  PSEUDO POWER-FAIL VECTOR ADDRESS
:          CONTENTS OF VECTOR ADDRESS
:

```

```

772
773
774
775
776
777
778
779
780
781
782 003650 022737 173000 173024 PPFVAR: CMP #173000,173024 :- TEST IF LOCATION 173024 SELECTED
783 003656 001414 BEQ 1$ :- IF NOT THEN GO TEST LOCATION 173224
784 003660 012737 003710 013064 MOV #1$,RETURN :-SET UP AN ALTERNATE RETURN
785 003666 004737 012006 JSR PC,MOLCK1 :-CHECK FOR A HOLE
786 003672 012737 173024 010562 MOV #173024,BUF2 :- IF IT IS THEN STORE ADDRESS
787 003700 013737 173024 01G564 MOV 173024,BUF2+2 :- STORE CONTENTS OF LOCATION 173024
788 003706 000423 BR 3$ :- GO TO RETURN
789 003710 022737 173000 173224 1$: CMP #173000,173224 :-TEST IF LOCATION 173224 SELECTED
790 003716 001414 BEQ 2$ :- IF NOT THEN SET ERROR INDICATOR
791 003720 012737 003750 013064 MOV #2$,RETURN :-SET UP AN ALTERNATE RETURN
792 003726 004737 012020 JSR PC,MGLCK2 :-CHECK FOR A HOLE
793 003732 012737 173224 010562 MOV #173224,BUF2 :- IF IT IS THEN STORE ADDRESS
794 003740 013737 173224 010564 MOV 173224,BUF2+2 :- STORE CONTENTS OF VECTOR
795 003746 000403 BR 3$ :- GET OVER ERROR
796 003750 052737 000040 001110 2$: BIS #PFERR,ROMERR :- SET ERROR INDICATOR
797 003756 000207 3$: RTS PC

```

.SBTTL PUT MESSAGE IN BUFFER ROUTINE  
 ;THIS SUBROUTINE FORMATS THE PARAMETER AND POWER-FAIL MESSAGES.

```

798
799
800
801 003760 017604 000000      PUTMES: MOV      @ (R6),R4      ;GET MESSAGE BUFFER ADDRESS
802 003764 005737 004342      TST      TIMES           ;IS THIS FIRST TIME THROUGH
803 003770 001125             BNE      3$              ;IF NOT FIRST TIME THEN FORMAT POWER-FAIL MESSAGE
804 003772 005237 004342      INC      TIMES           ;TOGGLE WATCHDOG
805 003776 012703 010502      MOV      #BUF1,R3        ;GET DATA BUFFER ADDRESS
806 004002 022713 165774      CMP      #165774,(R3)    ;IS DIAGNOSTIC ROM PRESENT
807 004006 001027             ENE      1$              ;IF NOT DON'T FORMAT DIAG. ROM MESSAGE
808 004010 105737 001122      TSTB    CPUITYP          ;SEE IF A NON-11/24/44 CPU
809 004014 001412             BEQ      100$            ;BRANCH OVER 11/24/44 SECTION IF SO
810 004016 122737 000001 001122 CMPB     #1,CPUITYP      ;SEE IF AN 11/44
811 004024 001003             BNE      98$             ;BRANCH IF NOT TO SETUP FOR 11/24
812 004026 012705 007465      MOV      #DRH44,R5       ;GET DIAG. ROM MESSAGE HEADER
813 004032 000405             BR       101$           ;BRANCH TO PUT HEADER IN MESSAGE BUFFER
814 004034 012705 007524      98$:   MOV      #DRH24,R5    ;GET DIAG. ROM MESSAGE HEADER
815 004040 000402             BR       101$           ;BRANCH TO PUT HEADER IN MESSAGE BUFFER
816 004042 012705 007444      100$:  MOV      #DRHEAD,R5    ;IF IT IS GET DIAG. ROM MESSAGE HEADER
817 004046 004737 004772      101$:  JSR      PC,FILBUF     ;GO PUT HEADER IN MESSAGE BUFFER
818 004052 000015             CR
819 004054 062703 000002      ADD      #2,R3           ;SKIP OVER ADDRESS OF ASCII
820 004050 000313             SWAB    (R3)             ;FORMAT ASCII FOR MESSAGE
821 004062 112324             MOVVB   (R3)+,(R4)+     ;PUT ASCII IN MESSAGE BUFFER
822 004064 112324             MOVVB   (R3)+,(R4)+
823 004066 012705 007600      1$:   MOVVB   #BRHEAD,R5     ;
824 004072 004737 004772      JSR      PC,FILBUF     ;GO PUT BOOT ROM HEADER IN MESS. BUF.
825 004076 000015             CR
826 004100 012701 000001      MOV      #1,R1           ;+POINT TO DIAGNOSTIC ROM
827 004104 013702 013104      MOV      MESSPT,R2       ;+POINT TO MSG TABLE
828 004110 012737 004124 013064 MOV      #2$,RETURN      ;+SET UP AN ALTERNATE RETURN
829 004116 012737 004310 013102 MOV      #5$,FINISH      ;+SET UP ANOTHER ALTERNATE RETURN
830 004124 112724 000015      2$:   MOVVB   #CR,(R4)+     ;PUT A CR/LF HERE
831 004130 112724 000012      MOVVB   #LF,(R4)+
832 004134 004737 012056      JSR      PC,ROMTYP      ;+FIND THE ROM TYPE
833 004140 062713 000004      ADD      #4,(R3)        ;GET FIRST ENTRY POINT
834 004144 004737 005149      JSR      PC,OCADD       ;GO CONVERT OCTAL TO ASCII
835 004150 004737 004772      JSR      PC,FILBUF     ;GO PUT ENTRY POINT IN MESSAGE BUF
836 004154 000011             HT
837 004156 112724 000040      MOVVB   #40,(R4)+
838 004162 112724 000040      MOVVB   #40,(R4)+
839 004166 112724 000040      MOVVB   #40,(R4)+
840 004172 062713 000002      ADD      #2,(R3)        ;GET SECOND ENTRY POINT
841 004176 004737 005140      JSR      PC,OCADD       ;GO CONVERT OCTAL TO ASCII
842 004202 004737 004772      JSR      PC,FILBUF     ;GO PUT ENTRY POINT IN MESSAGE BUF.
843 004206 000011             H:
844 004210 112724 000040      MOVVB   #40,(R4)+
845 004214 112724 000040      MOVVB   #40,(R4)+
846 004220 112724 000040      MOVVB   #40,(R4)+
847 004224 112724 000011      MOVVB   #HT,(R4)+      ;PUT TAB IN HERE
848 004230 062703 000002      ADD      #2,R3          ;UPDATE DATA BUFFER ADDRESS
849 004234 000313             SWAB    (R3)            ;FORMAT ASCII FOR MESSAGE
850 004236 112324             MOVVB   (R3)+,(R4)+     ;MOV ASCII TO MEST
851 004240 112324             MOVVB   (R3)+,(R4)+
852 004242 000730             BR       2$             ;GO BACK TO START OF LOOP
853 004244 012703 010562      3$:   MOV      #BU:2,R3        ;GET DATA BUFFER ADDRESS
854 004250 012705 007735      MOV      #PFHEAD,R5     ;GET POWER-FAIL HEADER ADDRESS
    
```

0 3

PUT MESSAGE IN BUFFER ROUTINE

855	004254	004737	004772		JSR	PC,FILBUF		:GO PUT POWER-FAIL HEADER IN MESSAGE BUF.
856	004260	000015			CR			
857	004262	022703	010566	4\$:	CMP	#BUF2+4,R3		:ARE WE DONE
858	004266	001410			BEQ	4\$		:IF YES THEN GO RETURN
859	004270	004737	005140		JSR	PC,OCADD		:GO CONVERT OCTAL TO ASCII
860	004274	004737	004772		JSR	PC,FILBUF		
861	004300	000011			HT			
862	004302	062703	000002		ADD	#2,R3		:UPDATE DATA BUFFER ADDRESS
863	004306	000765			SR	4\$		:GO BACK TO START OF LOOP
864	004310	112724	000015	5\$:	MOVB	#CR,(R4)+		:PUT / R/LF AT END OF MESSAGE
865	004314	112724	000012		MOVB	#LF,(R4)+		
866	004320	115034			CLRB	(R4)+		:PUT ZERO TERMINATOR AT END OF MESSAGE
867	004322	017637	000000	004332	MOV	@(R6),6\$		:GET MESSAGE BUFFER ADDRESS
868	004330	104401			TYPE			
869	004332	000000		6\$:	.WORD	0		
870	004334	062716	000002		ADR	#2,(R6)		:GET OVER MESSAGE BUFFER ADDRESS
871	004340	000207			RTS	PC		
872	004342	000000		TIMES:		0		

ERROR HANDLER ROUTINE

.SBTTL ERROR HANDLER ROUTINE  
 :THIS SUBROUTINE FORMATS THE ERROR MESSAGES THE TYPES THEM  
 :OUT.

ERRHAN: SAVREG

873										
874										
875										
876	004344	104411								
877	004346	012704	011276							
878	004352	005237	001120							
879	004356	032777	002000	174520						
880	004364	001402								
881	004366	004401								
882	004370	007344								
883	004372	032777	020000	174504	1\$:					
884	004400	001402								
885	004402	000137	004722							
886	004406	013700	001110		OVER1:					
887	004412	013737	001110	004710						
888	004420	012701	004744							
889	004424	000241								
890	004426	006000			2\$:					
891	004430	103403								
892	004432	062701	000002							
893	004436	000773								
894	004440	011105			3\$:					
895	004442	004737	004772							
896	004446	000015								
897	004450	032737	000040	001110						
898	004456	001065								
899	004460	032737	003600	001110						
900	004466	001411								
901	004470	013737	013076	004506						
902	004476	013705	013074							
903	004502	004737	004772							
904	004506	000000			13\$:					
905	004510	000450								
906	004512	032737	000023	001110	14\$:					
907	004520	001426								
908	004522	032737	000020	001110						
909	004530	001417								
910	004532	013700	003046							
911	004536	013701	007176							
912	004542	000241								
913	004544	006000			4\$:					
914	004546	103403								
915	004550	062701	000002							
916	004554	000773								
917	004556	011101			5\$:					
918	004560	112124			55\$:					
919	004562	105711								
920	004564	001375								
921	004566	000421								
922	004570	112324			6\$:					
923	004572	112324								
924	004574	000416								
925	004576	016603	000006		7\$:					
926	004602	004737	005034							
927	004606	004737	004772							
928	004612	000011								
929	004614	016603	000010							

:DPM001  
 :DPM001  
 :DPM001

```

930 004620 004737 005034 JSR PC,OCASC ;GO COVERT OCTAL TO ASCII
931 004624 004737 004772 JSR PC,FILBUF ;GO PUT DATA IN ERROR MESSAGE BUFFER.
932 004630 000011 HT
933 004632 112724 000015 8$: MOVB #CR,(R4)+ ;PUT CR/LF AT END OF MESSAGE
934 004636 112724 000012 MOVB #LF,(R4)+
935 004642 112724 000000 MOVB #0,(R4)+ ;PUT TERMINATOR AT END OF MESSAGE
936 004646 005737 001160 TST $ENV ;TST IF ON APT
937 004652 001002 BNE OVER2 ;IF YES BRANCH
938 004654 000137 004714 JMP OVER11 ;IF NO BRANCH
939 004660 022737 000001 OVER2: CMP #!,ROMFIN ;- ONLY A CPU ROM?????
940 004666 001005 BNE AROUND ;- IF NO BRANCH
941 004670 032737 000040 J01110 BIT #!FERR,ROMERR ;- NOW IF YES, IS POWER-
942 ;- FAIL VECTOR THE ERROR????
943 004676 001401 BEQ AROUND ;- YES ,WELL THEN IT'S
944 004700 000405 BR OVER11 ;- NOT A REAL ERROR , CONTINUE
945 004702 004737 005540 AROUND: JSR PC,$ATY1 ;GO REPORT ERROR
946 004706 011276 ERRMSG
947 004710 000000 OVER9: .WORD 0
948 004712 000000 HALT ;THEN HALT
949 004714 004737 005546 OVER11: JSR PC,$ATY3 ;IF NOT ON APT JUST REPORT ERROR
950 004720 011276 ERRMSG
951 004722 032777 1000J0 174154 OVER10: BIT #BIT15,$SWR ;IS HALT ON ERROR SET
952 004730 001401 BEQ 12$ ;IF NO SKIP OVER HALT
953 004732 000000 HALT
954 004734 005037 001110 12$: CLR ROMERR ;CLEAR ERROR FLAGS
955 004740 104412 RESREG
956 004742 000207 RTS PC
957
958
959 004744 010036 EHEADT: ER1MSG
960 004746 010002 ER2MSG
961 004750 010074 ER3MSG
962 004752 010152 ER4MSG
963 004754 010226 CRCMSG
964 004756 010252 PFMSG
965 004760 010330 NOROPM
966 004762 010406 SEQMSG ;+
967 004764 013164 ERMES1 ;+
968 004766 013243 ERMES2 ;+
969 004770 013323 ERMES3 ;+
    
```

```

970          .SBTTL FILL BUFFER ROUTINE
971          :THIS SUBROUTINE FILLS THE MESSAGE BUFFER WITH ASCII
972          :CHARACTERS.
973
974 004772 022776 000011 030000 FILBUF: CMP    #HT,0(R6)    :IS FIRST CHARACTER A TAB OR CR
975 005000 001003          BNE    1$          :IF CR THEN GO PUT CR/LF IN BUFFER
976 005002 112724 000011          MOVB   #HT,(R4)+    :IF TAB THEN PUT TAB IN BUFFER
977 005006 002404          BR     2$          :GET OVER NEXT LINE
978 005010 112724 000015 1$:  MOVB   #CR,(R4)+    :MOV CR/LF TO BUFFER
979 005014 112724 000012          MOVB   #LF,(R4)+
980 005020 112524 2$:  MOVB   (R5)+,(R4)+    :PUT A CHARACTER IN MESSAGE BUFFER
981 005022 105715          TSTB   (R5)          :IS NEXT CHARACTER ZERO
982 005024 001375          BNE    2$          :IF NOT PUT IT IN MESSAGE BUFFER AND GET NEXT
983 005026 062716 000002          ADI    #2,(R6)    :UPDATE RETURN POINTER TO GET OVER CHARACTER
984 005032 000207          RTI    PC          :THEN RETURN
    
```



SBTTL OCTAL TO ASCII CONVERSION ROUTINE  
 :THIS SUBROUTINE TAKES A SIXTEEN BIT OCTAL NUMBER AND  
 :CONVERTS IT TO 6 ASCII CHARACTERS

985					
986					
987					
988					
989	005034	012730	010566	OCASC:	MOV #OCTBUF,R0 ;GET BUFFER ADDRESS
990	005040	005020			CLR (R0)+ ;CLEAR BUFFER
991	005042	005020			CLR (R0)+
992	005044	005020			CLR (R0)+
993	005046	005020			CLR (R0)+
994	005050	012700	010566		MOV #OCTBUF,R0 ;GET BUFFER ADDRESS
995	005054	010337	005136		MOV R3,TEMP ;GET OCTAL NUMBER
996	005060	000241			CLC ;CLEAR CARRY
997	005062	006137	005136		ROL TEMP ;ROTATE BIT INTO CARRY BIT
998	005066	006110			ROL (R0) ;ROTATE CARRY BIT INTO BUFFER
999	005070	12710	000060	1\$:	BISB #60,(R0) ;MAKE IT ASCII
1000	005074	005200			INC R0 ;UPDATE BUFFER ADDRESS
1001	005076	020027	010574		CMR R0,#OCTBUF+6 ;HAVE WE CONVERTED ALL THE NUMBER
1002	005102	001003			BNE 2\$ ;IF NO CONTINUE
1003	005104	012705	010566		MOV #OCTBUF,R5 ;IF YES PUT BUFFER ADDRESS IN REGISTER
1004	005110	000207			RTS PC ;RETURN
1005	005112	006137	005136	2\$:	ROL TEMP ;ROTATE BIT INTO CARRY BIT
1006	005116	106110			ROLB (R0) ;ROTATE CARRY BIT INTO BUFFER
1007	005120	006137	005136		ROL TEMP ;
1008	005124	106110			ROLB (R0) ;
1009	005126	006137	005136		ROL TEMP ;
1010	005132	106110			ROLB (R0) ;
1011	005134	000755			BR 1\$ ;GO TO START OF LOOP
1012	005136	000000		TEMP:	0

```
1013          .SBTTL ROUTINE TO SETUP R3 FOR THE OCTAL TO ASCII ROUTINE
1014          ;THIS ROUTINE IS CALLED BY THE PUT MESSAGE ROUTINE TO GET THE RIGHT
1015          ;VALUE IN R3 SO THE OCTAL TO ASCII ROUTINE GETS THE RIGHT NUMBER.
1016
1017 005140 010346          OCADD:  MOV   R3, -(R6)          ;SAVE THE VALUE OF R3
1018 005142 011303          MOV   (R3), R3          ;PUT THE DATA TO BE CONVERTED IN R3
1019 005144 004737 005034  JSP   PC, OCASC          ;GO CONVERT OCTAL TO ASCII
1020 005150 012603          MOV   (R6)+, R3          ;RESTORE R3
1021 005152 000207          RTS    PC              ;RETURN
```

TYPE ROUTINE

1022

```

.SBTTL TYPE ROUTINE
*****
*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
*
*CALL:
*1) USING A TRAP INSTRUCTION
* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
*
*CR TYPE MESADR
*
*
005154 105737 005533 $TYPE: TSTB $TPFLG ;;IS THERE A TERMINAL?
005160 100002 BPL 1$ ;;BR IF YES
005162 000000 HALT ;;HALT HERE IF NO TERMINAL
005164 000430 BR 3$ ;;LEAVE
005166 010046 1$: MOV RO,-(SP) ;;SAVE RO
005170 017600 009002 MOV @2(SP),RO ;;GET ADDRESS OF ASCIZ STRING
005174 12273. 000001 001160 CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
005202 001011 62$ BNE 62$ ;;NO,GO CHECK FOR APT CONSOLE
005204 132737 000100 001161 BITB #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
005212 001405 BEQ 62$ ;;NO,GO CHECK FOR CONSOLE
005214 010037 005224 MOV RO,61$ ;;SETUP MESSAGE ADDRESS FOR APT
005220 004737 065246 JSR PC,$ATY3 ;;SPOOL MESSAGE TO APT
005224 132737 61$: .WORD 0 ;;MESSAGE ADDRESS
005226 000000 000040 001161 62$: BITB #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
00523. 201003 60$ BNE 60$ ;;YES,SKIP TYPE OUT
005236 112046 2$: MOVB (RO)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
005240 001005 4$ BNE 4$ ;;BR IF IT ISN'T THE TERMINATOR
005242 005726 TST (SP)+ ;;IF TERMINATOR POP IT OFF THE STACK
005244 012600 60$: MOV (SP)+,RO ;;RESTORE RO
005246 062716 000002 3$: ADD #2,(SP) ;;ADJUST RETURN PC
005252 000002 RTI ;;RETURN
005254 122716 000011 4$: CMPB #HT,(SP) ;;BRANCH IF <HT>
005260 001430 BEQ 8$ ;;BRANCH IF NOT <CR>
005262 122716 000200 CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
005266 001006 BNE 5$
005270 005726 TST (SP)+ ;;POP <CR><LF> EQUIV
005272 104401 $TYPE ;;TYPE A CR AND LF
005274 005535 $CRLF
005276 105037 005514 CLRB $CHARCNT ;;CLEAR CHARACTER COUNT
005302 007755 BR 2$ ;;GET NEXT CHARACTER
005304 064737 005366 5$: JSR PC,$TYPEC ;;GO TYPE THIS CHARACTER
005310 122726 005522 6$: CMPB $FILLC,(SP) ;;IS IT TIME FOR FILLER CHARS.?
005314 001350 BNE 2$ ;;IF NO GO GET NEXT CHAR.
005316 013746 005520 MOV $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
;;AND THE NULL CHAR.
005322 105366 000001 7$: DECB 1(SP) ;;DOES A NULL NEED TO BE TYPED?
005326 002770 BLT 6$ ;;BR IF NO--GO POP THE NULL OF: OF STACK
00533. 004737 005366 JSR PC,$TYPEC ;;GO TYPE A NULL
005337 105337 005514 DECB $CHARCNT ;;DO NOT COUNT AS A COUNT
005340 000770 BR 7$ ;;LOOP
-HORIZONTAL TAB PROCESSOR
005342 112716 000040 8$: MOVB #' ,(SP) ;;REPLACE TAB WITH SPACE

```

```

005346 004737 005366          9$: JSR PC,$TYPEC      ::TYPE A SPACE
005352 132737 000907 005514 BTR #7,$CHARCNT    ::BRANCH IF NOT AT
005360 001372          BNE 9$          ::TAB STOP
005362 005726          TST (SP)+          ::POP SPACE OFF STACK
005364 000724          BR 2$          ::GET NEXT CHARACTER
005366          $TYPEC:
005366 105777 000126          TSTB @STKS        ::CHAR IN KYBD BUFFER? :MJD001
005372 100022          BPL 10$          ::BR IF NOT :MJD001
005374 017746 000122          MOV @STKB,-(SP)   ::GET CHAR :MJD001
005400 042716 177600          BIC #177600,(SP)  ::STRIP EXTRANEIOUS BITS :MJD001
005404 122716 000023          CMPB #SXOFF,(SP) ::WAS CHAR XOFF :MJD001
005410 001012          BNE 102$        ::BR IF NOT :MJD001
005412          101$:
005412 105777 000102          TSTB @STKS        ::WAIT FOR CHAR :MJD001
005416 100375          BPL 101$        ::WAIT FOR CHAR :MJD001
005420 117716 000376          MOVB @STKB,(SP)  ::GET CHAR :MJD001
005424 042716 177600          BIC #177600,(SP) ::STRIP IT :MJD001
005430 122716 000021          CMPB #SXON,(SP)  ::WAS IT XON? :MJD001
005434 001366          BNE 101$        ::BR IF NOT :MJD001
005436          102$:
005436          TST (SP)+          ::FIX STACK :MJD001
005440          10$:
005440 105777 000060          TSTB @STPS        ::WAIT UNTIL PRINTER IS READY :MJD001
005444 100375          BPL 10$          ::WAIT UNTIL PRINTER IS READY :MJD001
005446 126627 000002 000021  CMPB 2(SP),#$XON  ::IS CHARACTER A RANDOM XON? :RAN001
005454 001420          BEQ $TYPEX        ::BRANCH IF YES :RAN001
005456 116677 000002 000042  MOVB 2(SP),@STPB  ::LOAD CHAR TO BE TYPED INTO DATA REG.
005464 122766 000015 000002  CMPB #CR,2(SP)    ::IS CHARACTER A CARRIAGE RETURN?
005472 001003          BNE 1$          ::BRANCH IF NO
005474 105037 005514          CLRB $CHARCNT    ::YES--CLEAR CHARACTER COUNT
005500 00406          BR $TYPEX        ::EXIT
005502 122766 000012 000002 1$: CMPB #LF,2(SP)    ::IS CHARACTER A LINE FEED?
005510 001402          BEQ $TYPEX        ::BRANCH IF YES
005512 103227          INCB (PC)+        ::COUNT THE CHARACTER
005514 000000          $CHARCNT: WORD 0 ::CHARACTER COUNT STORAGE
005516 000207          $TYPEX: RTS PC
005520 177560          $TKS: .WORD 177560 ::TTY KBD STATUS :MJD001
005522 177562          $TKB: .WORD 177562 ::TTY KBD BUFFER :MJD001
005524 177564          $TPS: .WORD 177564 ::TTY PRINTER STATUS REG. ADDRESS
005526 177566          $TPB: .WORD 177566 ::TTY PRINTER BUFFER REG. ADDRESS
005530 000          $NULL: .BYTE 0 ::CONTAINS NULL CHARACTER FOR FILLS
005531 002          $FILLS: .BYTE 2 ::CONTAINS # OF FILLER CHARACTERS REQUIRED
005532 012          $FILL: .BYTE 12 ::INSERT FILL CHARS. AFTER A "LINE FEED"
005533 000          $TPFLG: .BYTE 0 ::"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
005534 077          $QUES: .ASCII "?" ::QUESTION MARK
005535 015          $CRLF: .ASCII <15> ::CARRIAGE RETURN
005536 012          $LF: .ASCII <12> ::LINEFEED
  
```

1024

..BTLL APT COMMUNICATIONS ROUTINE

```

005540 112737 000001 006004 $ATY1: MOV #1,$FFLG ::TO REPORT FATAL ERROR
005546 112737 000001 006002 $ATY3: MOV #1,$MFLG ::TO TYPE A MESSAGE
005554 000403 BR $ATYC
005556 112737 000001 006004 $ATY4: MOV #1,$FFLG ::TO ONLY REPORT FATAL ERROR
005564 $ATYC:
005564 010046 MOV R0,-(SP) ::PUSH R0 ON STACK
005566 010046 MOV R1,-(SP) ::PUSH R1 ON STACK
005570 105737 006002 TSTB $MFLG ::SHOULD TYPE A MESSAGE?
005574 001450 BEQ 5$ ::IF NOT: BR
005576 122737 000001 001160 #APTENV,$ENVV ::OPERATING UNDER APT?
005604 001031 BNE 3$ ::IF NOT: BR
005606 132737 000100 001161 BITB #APTSPOOL,$ENVM ::SHOULD SPOOL MESSAGES?
005614 001425 BEQ 3$ ::IF NOT: BR
005616 017600 000004 MOV #4(SP),R0 ::GET MESSAGE ADDR.
005622 062766 000002 000004 ADD #2,4(SP) ::BUMP RETURN ADDR.
005630 005737 001140 1$: TST $MSGTYPE ::SEE IF DONE W/ LAST XMISSION?
005634 001375 BNE 1$ ::IF NOT: WAIT
005636 010037 001154 MOV R0,$MSGAD ::PUT ADDR IN MAILBOX
005642 105720 2$: TSTB (R0)+ ::FIND END OF MESSAGE
005644 001376 BNE 2$
005652 006200 SLB $MSGAD,R0 ::SLB START OF MESSAGE
005654 010037 001156 ASR R0 ::GET MESSAGE LGTH IN WORDS
005660 012737 000004 001140 MOV R0,$MSGLGT ::PUT LENGTH IN MAILBOX
005666 000413 BR #4,$MSGTYPE ::TELL APT TO TAKE MSG.
005670 017637 000004 005714 3$: MOV #4(SP),4$ ::PUT MSG ADDR IN JSR LINKAGE
005676 062766 000002 000004 ADD #2,4(SP) ::BUMP RETURN ADDRESS
005704 013746 177776 MOV 177776,-(SP) ::PUSH 177776 ON STACK
005710 004737 005154 JSR PC,$TYPE ::CALL TYPE MACRO
005714 000000 4$: .WORD 0
005716 5$:
005716 105737 006004 10$: TSTB $FFLG ::SHOULD REPORT FATAL ERROR?
005722 001416 BEQ 12$ ::IF NOT: BR
005724 005737 301160 TST $ENVV ::RUNNING UNDER APT?
005730 001413 BEQ 12$ ::IF NOT: BR
005732 005737 001140 11$: TST $MSGTYPE ::FINISHED LAST MESSAGE?
005736 001375 BNE 11$ ::IF NOT: WAIT
005740 017637 000004 001142 MOV #4(SP),$FATAL ::GET ERROR #
005746 062766 000002 000004 ADD #2,4(SP) ::BUMP RETURN ADDR.
005754 005237 001140 INC $MSGTYPE ::TELL APT TO TAKE ERROR
005760 105037 006004 12$: CLRB $FFLG ::CLEAR FATAL FLAG
005764 105037 006003 CLRB $LFLG ::CLEAR LOG FLAG
005770 105037 006002 CLRB $MFLG ::CLEAR MESSAGE FLAG
005774 012601 MOV (SP)+,R1 ::POP STACK INTO R1
005776 012600 MOV (SP)+,R0 ::POP STACK INTO R0
006000 000207 RTS PC ::RETURN
006002 000 $MFLG: .BYTE 0 ::MESSG. FLAG
006003 000 $LFLG: .BYTE 0 ::LOG FLAG
006004 000 $FFLG: .BYTE 0 ::FATAL FLAG
::EVEN

000200 APTSIZE=200
000001 APTENV=001
000100 APTSPool=100
000040 APTCSUP=040
  
```

T0c0

..SBTTL TTY INPUT ROUTINE

```

*****
ENABL LSS
*****
*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
*WHEN OPERATING IN TTY FLAG MODE.
006006 022737 000176 001104 $K$SWR: CMP #SWREG,SWR ;;IS THE SOFT-SWR SELECTED?
006014 001074 BNE 15$ ;;BRANCH IF NO
006016 135777 177476 TSTB @STKS ;;CHAR THERE?
006022 100071 BPL 15$ ;;IF NO, DON'T WAIT AROUND
006024 117746 177472 MOVB @STKB,-(SP) ;;SAVE THE CHAR
006030 042716 177660 BIC #'C177,(SP) ;;STRIP-OFF THE ASCII
006034 022726 000007 CMP #7,(SP)+ ;;IS IT A CONTROL G?
006040 001062 BNE 15$ ;;NO, RETURN TO USER
006042 123727 001100 000001 CMPSB #AUTOB,#1 ;;ARE WE RUNNING IN AUTO-MODE?
006050 001456 BEQ 15$ ;;BRANCH IF YES
006052 104401 006543 TYPE ,SCNTLG ;;ECHO THE CONTROL-G (^G)
006056 104401 006550 $G$SWR: TYPE ,SMSUR ;;TYPE CURRENT CONTENTS
006062 013746 000176 MOV SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
006066 104402 TYPE,C TYPE--OCTAL ASCII(ALL DIGITS)
006070 104401 TYPE ,SMNEU ;;PROMPT FOR NEW SWR
006074 005046 19$: CLR -(SP) ;;CLEAR COUNTER
006076 005046 CLR -(SP) ;;THE NEW SWR
006100 105777 177414 7$: TSTB @STKS ;;CHAR THERE?
006104 100375 BPL 7$ ;;IF NOT TRY AGAIN
006106 117746 177410 MOVB @STKB,-(SP) ;;PICK UP CHAR
006112 042716 177600 BIC #'C177,(SP) ;;MAKE IT 7-BIT ASCII
006116 021627 000025 9$: CMP (SP),#25 ;;IS IT A CONTROL-U?
006122 001005 BNE 10$ ;;BRANCH IF NOT
006124 104401 006536 TYPE ,SCNTLU ;;YES, ECHO CONTROL-U (^U)
006130 062706 000006 20$: ADD #6,SP ;;IGNORE PREVIOUS INPUT
006134 000757 SR 19$ ;;LET'S TRY IT AGAIN
006136 021627 300015 10$: CMP (SP),#15 ;;IS IT A <CR>?
006142 001022 BNE 16$ ;;BRANCH IF NO
006144 005766 000004 TST 4(SP) ;;YES, IS IT THE FIRST CHAR?
006150 001413 BEQ 11$ ;;BRANCH IF YES
006152 016677 000002 177224 MOV 2(SP),@SWR ;;SAVE NEW SWR
006160 062706 000006 11$: ADD #6,SP ;;CLEAR UP STACK
006164 104401 005535 14$: TYPE ,SCRLF ;;ECHO <CR> AND <LF>
006170 123727 001101 000001 CMPSB $INTAG,#1 ;;RE-ENABLE TTY KBD INTERRUPTS?
006176 001003 BNE 15$ ;;BRANCH IF NOT
006200 012777 000100 177312 MOV #100,@STKS ;;RE-ENABLE TTY KBD INTERRUPTS
006206 000002 RTI ;;RETURN
006210 004737 005366 16$: JSR PC,$TYPEC ;;ECHO CHAR
006214 021627 000060 CMP (SP),#60 ;;CHAR < 0?
006220 002420 BLT 18$ ;;BRANCH IF YES
006222 021627 000067 CMP (SP),#67 ;;CHAR > ??
006226 003015 BGT 18$ ;;BRANCH IF YES
006230 042726 000060 BIC #60,(SP)+ ;;STRIP-OFF ASCII
006234 005766 000002 TST 2(SP) ;;IS THIS THE FIRST CHAR
006240 001403 BEQ 17$ ;;BRANCH IF YES
006242 006316 ASL (SP) ;;NO, SHIFT PRESENT
006244 006316 ASL (SP) ;; CHAR OVER TO MAKE
006246 006316 ASL (SP) ;; ROOM FOR NEW ONE.
006250 005266 000002 17$: INC 2(SP) ;;KEEP COUNT OF CHAR

```

TTY INPUT ROUTINE

```

006254 056616 177776          BIS      -2(SP), (SP)      ;; SET IN NEW CHAR
006260 000707                BR       7$              ;; GET THE NEXT ONE
006262 104401 005534      18$:     TYPE    $WUES      ;; TYPE ?<CR><LF>
006266 000720                BR       20$              ;; SIMULATE CONTROL-U

.DSABL  L$B
*****
*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
*CALL:
*
* RDCHR      ;; INPUT A SINGLE CHARACTER FROM THE TTY
* RETURN HERE ;; CHARACTER IS ON THE STACK
*           ;; WITH PARITY BIT STRIPPED OFF

006270 011646          $RDCHR:  MOV      (SP), -(SP)      ;; PUSH DOWN THE PC
006272 016666 000004 000002  MOV      4(SP), 2(SP)      ;; SAVE THE PS
006300 105777 177214      1$:     TSTB    @STKS      ;; WAIT FOR
006304 100375                BPL     1$              ;; A CHARACTER
006306 117766 177210 000004  MOVB    @STKB, 4(SP)      ;; READ THE TTY
006314 042766 177600 000004  BIC     #'<C177>, 4(SP)  ;; GET RID OF JUNK IF ANY
006322 026627 000004 000023  CMP     4(SP), #2$      ;; IS IT A CONTROL-S?
006330 001013                BNE    3$              ;; BRANCH IF NO
006332 105777 177162      2$:     TSTB    @STKS      ;; WAIT FOR A CHARACTER
006336 100375                BPL     2$              ;; LOOP UNTIL ITS THERE
006340 117746 177156      MOVB    @STKB, -(SP)      ;; GET CHARACTER
006344 042716 177600      BIC     #'<C177>, (SP)  ;; MAKE IT 7-BIT ASCII
006350 027627 000021      CMP     (SP)+, #21      ;; IS IT A CONTROL-Q?
006354 061366                BNE    2$              ;; IF NOT DISCARD IT
006356 000750                BR     1$              ;; Y... RESUME
006360 026627 000004 000021  3$:     CMP     4(SP), #XON  ;; IS IT A RANDOM XON?
006366 001744                BEQ    1$              ;; BRANCH IF YES
006370 026627 000004 000140  CMP     4(SP), #140    ;; IS IT UPPER CASE?
006376 002407                BLT    4$              ;; BRANCH IF YES
006400 026627 000004 000175  CMP     4(SP), #175    ;; IS IT A SPECIAL CHAR?
006406 003003                BGT    4$              ;; BRANCH IF YES
006410 042766 000040 000004  BIC     #40, 4(SP)     ;; MAKE IT UPPER CASE
006416 000002                RTI     ;; GO BACK TO USER

*****
*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
*CALL:
*
* RDLIN      ;; INPUT A STRING FROM THE TTY
* RETURN HERE ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
*           ;; TERMINATOR WILL BE A BYTE OF ALL 0'S

006420 010346          $RDLIN:  MOV      R3, -(SP)      ;; SAVE R3
006422 012703 006526      1$:     MOV     @TTYIN, R3    ;; GET ADDRESS
006426 022703 006536      2$:     CMP     @TTYIN+8, R3    ;; BUFFER FULL?
006432 101405                BLOS   4$              ;; BR IF YES
006434 104407                BLOSRDCHR      ;; GO READ ONE CHARACTER FROM THE TTY
006436 112613      MOVB    (SP)+, (R3)      ;; GET CHARACTER
006440 122713 000177      10$:    CMPB    #177, (R3)      ;; IS IT A RUBOUT
006444 001003                BNE    3$              ;; SKIP IF NOT
006446 104401 005534      4$:     TYPE    $QUES      ;; TYPE A '?'
006452 000763                BR     1$              ;; CLEAR THE BUFFER AND LOOP
006454 111337 006524      3$:     MOVB    (R3), 9$      ;; ECHO THE CHARACTER
006460 104401 006524      TYPE    9$
006464 122723 000015      CMPB    #15, (R3)+     ;; CHECK FOR RETURN
006470 001356                BNE    2$              ;; LOOP IF NOT RETURN
006472 105063 177777      CLRB   -1(R3)         ;; CLEAR RETURN (THE 15)
006476 10101 005536      TYPE    , $LF        ;; TYPE A LINE FEED

```

006502	012603			MOV	(SP)+,R3	::RESTORE R3
006504	011646			MOV	(SP)-,(SP)	::ADJUST THE STACK AND PUT ADDRESS OF THE
006506	016666	000004	000002	MOV	4(SP),2(SP)	:: FIRST ASCII CHARACTER ON IT
006514	012766	006526	000004	MOV	#TTYIN,4(SP)	::
006522	000002			RTI		::RETURN
006524	000			9%:	.BYTE 0	::STORAGE FOR ASCII CHAR. TO TYPE
006525	000				.BYTE 0	::TERMINATOR
006526				STTYIN:	.BLKB 8	::RESERVE 8 BYTES FOR TTY INPUT
006536	136	125	015	\$CNTLU:	.ASCIIZ /"U"/<15><12>	::CONTROL 'U'
006543	136	107	015	\$CNTLG:	.ASCIIZ /"G"/<15><12>	::CONTROL 'G'
006550	015	012	123	\$PSWR:	.ASCIIZ <15><12>/SWR = /	::CONTROL 'G'
006561	040	040	116	\$PNEW:	.ASCIIZ / NEW = /	::CONTROL 'G'



1028

```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE
*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPOS    N              ;;CALL FOR TYPEOUT
*      .BYTE   N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*      .BYTE   M              ;;M=1 OR 0
*                               ;;1=TYPE LEADING ZEROS
*                               ;;0=SUPPRESS LEADING ZEROS
*
*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPON    N              ;;CALL FOR TYPEOUT
*
*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPOC    N              ;;CALL FOR TYPEOUT
006572 017646 000000 007015 $TYPOS: MOV      @ (SP),-(SP)      ;;PICKUP THE MODE
006576 116637 000001 007015 MOV      1(SP),%$OFILL    ;;LOAD ZERO FILL SWITCH
006604 112637 007017 007015 MOV      (%SP)+,%$MODE+1  ;;NUMBER OF DIGITS TO TYPE
006610 062716 000002 007015 ADD      #2,(SP)          ;;ADJUST RETURN ADDRESS
006614 000406 000000 007015 BR        $TYPON
006616 112737 000001 007015 $TYPOC: MOV      #1,%$OFILL    ;;SET THE ZERO FILL SWITCH
006624 112737 000006 007017 MOV      #6,%$MODE+1     ;;SET FOR SIX(6) DIGITS
006632 112737 000005 007014 $TYPON: MOV      #5,%$CNT    ;;SET THE ITERATION COUNT
006640 010346 000000 007015 MOV      R3,-(SP)        ;;SAVE R3
006642 010446 000000 007015 MOV      R4,-(SP)        ;;SAVE R4
006644 010546 000000 007015 MOV      R5,-(SP)        ;;SAVE R5
006646 113704 007017 007015 MOV      #%MODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
006652 005404 000000 007015 NEG      R4
006654 062704 000006 007015 ADD      #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
006660 110437 007016 007015 MOV      R4,%$MODE       ;;SAVE IT FOR USE
006664 113704 007015 007015 MOV      %$OFILL,R4      ;;GET THE ZERO FILL SWITCH
006670 016605 000012 007015 MOV      12(SP),R5       ;;PICKUP THE INPUT NUMBER
006674 005003 000000 007015 CLR      R3              ;;CLEAR THE OUTPUT WORD
006676 006105 000000 007015 1$: ROL     R5              ;;ROTATE MSB INTO 'C'
006700 000404 000000 007015 BR        R5              ;;GO DC MSB
006702 006105 000000 007015 2$: ROL     R5              ;;FORM THIS DIGIT
006704 006105 000000 007015 ROL     R5
006706 006105 000000 007015 ROL     R5
006710 010503 000000 007015 MOV      R5,R3
006712 006103 000000 007015 3$: ROL     R3              ;;GET LSB OF THIS DIGIT
006714 105337 007016 007015 DECB    %$MODE          ;;TYPE THIS DIGIT?
006720 100016 000000 007015 BPL     R5              ;;BR IF NO
006722 042703 177770 007015 BIC     #177770,R3      ;;GET RID OF JUNK
006726 001002 000000 007015 BNE     R4              ;;TEST FOR 0
006730 005704 000000 007015 TST     R4              ;;SUPPRESS THIS 0?
006732 001403 000000 007015 REQ     R4              ;;BR IF YES
006734 005204 000000 007015 4$: INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
006736 052703 000060 007015 BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
006742 052703 000040 007015 5$: BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY

```

006746	110337	007012		MOVB	R3,8\$	::SAVE FOR TYPING
006752	104401	007012		TYPE	8\$	::GO TYPE THIS DIGIT
006756	105337	007014	7\$:	DECB	\$OCNT	::COUNT BY 1
006762	003347			BGT	2\$	::BR IF MORE TO DO
006764	002402			BGT	6\$	::BR IF DONE
006766	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
006770	000744			BR	2\$	::GO DO THE LAST DIGIT
006772	012605		6\$:	MOV	(SP)+,R5	::RESTORE R5
006774	012604			MOV	(SP)+,R4	::RESTORE R4
006776	012603			MOV	(SP)+,R3	::RESTORE R3
007000	016666	000002	000004	MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
007006	012616			MOV	(SP)+,(SP)	
007010	000002			RTI		::RETURN
007012	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
007013	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
007014	000			\$OCNT:	.BYTE 0	::OCTAL DIGIT COUNTER
007015	000			\$OFILL:	.BYTE 0	::ZERO FILL SWITCH
007016	000000			\$OMODE:	.WORD 0	::NUMBER OF DIGITS TO TYPE

1030

..SBTTL SAVE AND RESTORE RO-R5 ROUTINES  
 ..\*\*\*\*\*

..\*SAVE RO-R5  
 ..\*CALL:  
 ..\* SAVREG  
 ..\*UPON RETURN FROM \$SAVREG THE STACK WILL LOOK LIKE:  
 ..\*  
 ..\*TOP---(+16)  
 ..\* +2---(+18)  
 ..\* +4---R5  
 ..\* +6---R4  
 ..\* +8---R3  
 ..\* +10---R2  
 ..\* +12---R1  
 ..\* +14---R0

007020	010046		MOV	R0,-(SP)	::PUSH R0 ON STACK
007022	010146		MOV	R1,-(SP)	::PUSH R1 ON STACK
007024	010246		MOV	R2,-(SP)	::PUSH R2 ON STACK
007026	010346		MOV	R3,-(SP)	::PUSH R3 ON STACK
007030	010446		MOV	R4,-(SP)	::PUSH R4 ON STACK
007032	010546		MOV	R5,-(SP)	::PUSH R5 ON STACK
007034	016646	000022	MOV	22(SP),-(SP)	::SAVE PS OF MAIN FLOW
007040	016646	000022	MOV	22(SP),-(SP)	::SAVE PC OF MAIN FLOW
007044	016646	000022	MOV	22(SP),-(SP)	::SAVE PS OF CALL
007050	016646	000022	MOV	22(SP),-(SP)	::SAVE PC OF CALL
007054	000002		RTI		

..\*RESTORE RO-R5

..\*CALL:  
 ..\* RESREG  
 ..\* \$RESREG:

007056	012666	000022	MOV	(SP)+,22(SP)	::RESTORE PC OF CALL
007062	012666	000022	MOV	(SP)+,22(SP)	::RESTORE PS OF CALL
007066	012666	000022	MOV	(SP)+,22(SP)	::RESTORE PC OF MAIN FLOW
007072	012666	000022	MOV	(SP)+,22(SP)	::RESTORE PS OF MAIN FLOW
007076	012605		MOV	(SP)+,R5	::POP STACK INTO R5
007100	012604		MOV	(SP)+,R4	::POP STACK INTO R4
007102	012603		MOV	(SP)+,R3	::POP STACK INTO R3
007104	012602		MOV	(SP)+,R2	::POP STACK INTO R2
007106	012601		MOV	(SP)+,R1	::POP STACK INTO R1
007110	012600		MOV	(SP)+,R0	::POP STACK INTO R0
007112	000002		RTI		

1036

.SBTTL TRAP DECODER

\*\*\*\*\*  
 \*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION  
 \*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS  
 \*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL  
 \*GO TO THAT ROUTINE.

007114 010046  
 007116 016600 000002  
 007122 005740  
 007124 111000  
 007126 006300  
 007130 016000 007150  
 007134 000200

```
$TRAP:  MOV  RO,-(SP)      ;;SAVE R0
        MOV  2(SP),R0    ;;GET TRAP ADDRESS
        TST  -(R0)      ;;BACKUP BY 2
        MOVB (R0),R0    ;;GET RIGHT BYTE OF TRAP
        ASL  R0          ;;POSITION FOR INDEXING
        MOV  $,RPAD(R0),R0 ;;INDEX TO TABLE
        PTS  R0          ;;GO TO ROUTINE
```

007136 011646  
 007140 016666 000004 000002  
 007146 000002

```
;;THIS IS USE TO HANDLE THE 'GETPRI' MACRO
$TRAP2: MOV  (SP),-(SP)  ;;MOVE THE PC DOWN
        MOV  4(SP),2(SP) ;;MOVE THE PSW DOWN
        RTI              ;;RESTORE THE PSW
```

.SBTTL TRAP TABLE

\*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED  
 \*BY THE 'TRAP' INSTRUCTION.  
 : ROUTINE  
 :

007150 007136  
 007152 005154  
 007154 006616  
 007156 006572  
 007160 006632  
 007162 006056  
 007164 006006  
 007166 006270  
 007170 006420  
 007172 007020  
 007174 007056

```
$TRPAD: .WORD  $TRAP2
        $TYPE  ;;CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
        $TYPOC ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
        $TYPOS ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
        $TYPON ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
        $GTSWR ;;CALL=GTSWR    TRAP+5(104405)  GET SOFT-SWR SETTING
        $CKSWR ;;CALL=CKSWR    TRAP+6(104406)  TEST FOR CHANGE IN SOFT-SWR
        $RDCHR ;;CALL=RDCHR    TRAP+7(104407)  TTY TYPEIN CHARACTER ROUTINE
        $RDLIN ;;CALL=RDLIN    TRAP+10(104410) TTY TYPEIN STRING ROUTINE
        $$AVREG ;;CALL=$AVREG  TRAP+11(104411) SAVE R0-R5 ROUTINE
        $RESREG ;;CALL=$RESREG TRAP+12(104412) RESTORE R0-R5 ROUTINE
```

MESSAGES

1038					.SBTTL	MESSAGES	
1039	007176	007332			ROMPTR: .WORD	ROMNUM	
1040	007200	065	070	000	AROM11: .ASCIZ	/58/	***
1041	007203	064	070	000	AROM21: .ASCIZ	:/48/	***
1042	007206	064	071	000	AROM31: .ASCIZ	/49/	***
1043	007211	065	060	000	AROM41: .ASCIZ	/50/	***
1044	007214	065	071	000	AROM51: .ASCIZ	/50/	***
1045	007217	062	060	000	BROM11: .ASCIZ	/2C/	***
1046	007222	063	065	040	BROM21: .ASCIZ	/35 OR 75/	** :DPM001
1047	007233	063	063	040	BROM31: .ASCIZ	/33 OR 89/	** :DPM001
1048	007244	063	064	040	BROM41: .ASCIZ	/34 OR 82/	** :DPM001
1049	007255	063	062	040	BROM51: .ASCIZ	/32 OR 83/	** :DPM001
1050	007266	062	060	000	RPMNUM1: .ASCIZ	/20/	***
1051	007271	063	065	000	RPMNUM2: .ASCIZ	/35/	***
1052	007274	063	063	000	RPMNUM3: .ASCIZ	/33/	***
1053	007277	063	061	000	RPMNUM4: .ASCIZ	/34/	***
1054	007302	063	062	000	RPMNUM5: .ASCIZ	/32/	***
1055					.EVEN		
1056	007306	007200	007203	007206	AROM: .WORD	AROM11,AROM21,AROM31,AROM41,AROM51	
1057	007320	007217	007222	007233	BROM: .WORD	BROM11,BROM21,BROM31,BROM41,BROM51	
1053	007332	007266	007271	007274	ROMNUM: .WORD	RPMNUM1,RPMNUM2,RPMNUM3,RPMNUM4,RPMNUM5	
1059	007344	207	377	377	BELL: .ASCIZ	<207><377><377>	
1060	007350	012	015	116	PNORM: .ASCIZ	<12><15>/NO ROMS RESPONDED TO THE SIZING ROUTINE/<12><15>	
1061	007424	054	040	105	ERL: .ASCIZ	/, ERROR LINE # /	

1062				.SBTTL	ASCIZ MESSAGES				
1063	007444	104	111	101	DRHEAD: .ASCIZ	/DIAG. ROM: E20	/		
1064	007465	104	111	101	DRH44: .ASCIZ	/DIAG. ROM FOR 11-44 UBI: E58	/	***	:DPM001
1065	007524	104	111	101	DRH24: .ASCIZ	/DIAG. ROM FOR 11-24 KT24: E74	/	M9312: E58	:DPM001
1066	007600	012	102	117	BRHEAD: .ASCIZ	<12>/BOOTSTRAP ROM ENTRY POINTS	AND DEVICE CODES/	<15><12>	
1067	007656	114	117	103	.ASCIZ	/LOC.	NO DIAG.	RUN DIAG.	DEVICE CODE/

## ASCIZ MESSAGES

1069	007735	120	123	105	PFHEAD: .ASCIZ	@PSEUDO POWER-FAIL VECTOR ADR./NEW PCB
1070	010002	103	117	125	ER2MSG: .ASCIZ	/COULD NOT FIND DEVICE CODE /
1071	010036	106	117	125	ER1MSG: .ASCIZ	/FOUND UNEXPECTED DEVICE CODE /
1072	010074	120	117	127	ER3MSG: .ASCIZ	/POWER-FAIL VECTOR ERROR/<15><12>
1073	010125	011	105	130	.ASCIZ	/ EXPECTED RECIEVED/<15><12>
1074	010152	120	117	127	ER4MSG: .ASCIZ	/POWER-FAIL DATA ERROR/<15><12>
1075	010201	011	105	130	.ASCIZ	/ EXPECTED RECIEVED/<15><12>
1076	010226	103	122	103	CRCMSG: .ASCIZ	/CRC ERROR IN ROM E-/
1077	010252	103	117	125	PFMSG: .ASCIZ	/COULD NOT DETERMINE POWER-FAIL VECTOR ADDRESS/

```

1079 010330      116    117    040  NOROMM: .ASCII /NO ROMS TEST ERROR/<15><12>
1080 010354      040    040    040  .ASCII / VALUE ADDRESS/<15><12>
1081 010406      015    012    122  SEQMSG: .ASCII <15><12>/ROM SEQUENCE IS INCORRECT AS PER /
1082 010451      111    116    125  .ASCII /INSTALLATION PROCEDURE./

```

1083  
1084  
1085  
1086  
1087

.EVEN  
:BUFFERS

```

1088 010502 000030  BUF1:  .REPT 30
1090 010502 000000      .WORD 0
      010504 000000      .WORD 0
      010506 000000      .WORD 0
      010510 000000      .WORD 0
      010512 000000      .WORD 0
      010514 000000      .WORD 0
      010516 000000      .WORD 0
      010520 000000      .WORD 0
      010522 000000      .WORD 0
      010524 000000      .WORD 0
      010526 000000      .WORD 0
      010530 000000      .WORD 0
      010532 000000      .WORD 0
      010534 000000      .WORD 0
      010536 000000      .WORD 0
      010540 000000      .WORD 0
      010542 000000      .WORD 0
      010544 000000      .WORD 0
      010546 000000      .WORD 0
      010550 000000      .WORD 0
      010552 000000      .WORD 0
      010554 000000      .WORD 0
      010556 000000      .WORD 0
      010560 000000      .WORD 0
1091 010562 000000  BUF2:  .WORD 0
1092 010564 000000      .WORD 0
1093 010566      OCTBUF: .BLKB 10
1094 010576      MES1:  .BLKB 400
1095 011176      MES2:  .BLKB 100
1096 011276      ERRMSG: .BLKB 400

```

1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112

```

;+
;+THE FOLLOWING CODE WAS ADDED IN REV B0:
;+
;+"CLEAR" CLEARS THE NEW LABEL AND BUFFER LOCATIONS
;+
;+"CON1" AND "CON2" FILL "BUF1" WITH THE CONTINUATION ROM DATA AND
;+SET ERROR FLAGS WHEN NECESSARY
;+
;+"MOLCK1" AND "MOLCK2" CHECK FOR HOLES TO VERIFY THE POWER FAIL
;+VECTOR ADDRESS; ALSO SET 11/50 INDICATOR WHEN APPLICABLE
;+
;+"ROMTYP" AND "ROMM1" FILL THE MESSAGE BUFFER WITH A CONTINUATION ROM
;+MESSAGE AND WITH A ROM IDENTIFICATION MESSAGE
;+

```



```

1116 011676 012700 013000 CLEAR: MOV #SEQBUF, R0 ;+SET UP TO CLEAR LOCATIONS
1117 011702 005020 1$: CLR (R0)+ ;+
1118 011704 020027 013102 CMP RO, #FINISH ;+ALL CLEARED
1119 011710 101774 BLOS 1$ ;+BRANCH IF NO
1120 011712 0002C7 RTS PC ;+
1121
1122 011714 030137 013070 CON1: BIT R1, CONFIN ;+CONTINUATION CHIP?
1123 011720 001001 BNE 1$ ;+BRANCH IF YES
1124 011722 0002C7 RTS PC ;+
1125
1126 011724 050137 013060 1$: BIS R1, CONER1 ;+CHIP DOES NOT BELONG HERE-SET ERROR FLAG
1127 011730 012720 000001 MOV #1, (R0)+ ;+SET CONTINUATION ROM INDICATOR
1128 011734 012720 177777 MOV #-1, (R0)+ ;+ILLEGAL ROM'S DEVICE CODE =-1
1129 011740 000207 RTS PC ;+
1130
1131 011742 005703 CON2: TST R3 ;+POINTING OVER THIS CHIP?
1132 011744 001001 BNE 1$ ;+BRANCH IF YES
1133 011746 000207 RTS PC ;+
1134
1135 011750 030137 013070 1$: BIT R1, CONFIN ;+CONTINUATION CHIP?
1136 011754 001005 BNE 2$ ;+BRANCH IF YES
1137 011756 014037 013062 MCV -(R0), CONER2 ;+CHIP IS MISSING-SAVE DEVICE CODE
1138 011762 062700 000002 ADD #2, RO ;+
1139 011766 000207 RTS PC ;+
1140
1141 011770 012710 000001 2$: MOV #1, (R0) ;+SET CONTINUATION CHIP INDICATOR
1142 011774 014002 MOV -(R0), R2 ;+GET PREVIOUS DEVICE CODE
1143 011776 062700 000004 ADD #4, RO ;+POINT TO CORRECT LOCATION
1144 012002 010220 MGV R2, (R0)+ ;+PUT IN DEVICE CODE
1145 012004 000207 RTS PC ;+
1146
1147 012006 032737 000002 001116 HOLCK1: BIT #2, ROMFIN ;+A CHIP IN SOCKET #1?
1148 012014 001415 BEQ END ;+BRANCH IF NO
1149 012016 000207 RTS PC ;+
1150
1151 012020 032737 000004 001116 HOLCK2: BIT #4, ROMFIN ;+A CHIP IN SOCKET #2?
1152 012026 001401 BEQ 1$ ;+BRANCH IF NO
1153 012030 000207 RTS PC ;+
1154
1155 012032 122737 000777 173224 1$: CMPB #777, 173224 ;+ARE SWITCHES SET?
1156 012040 001403 BEQ END ;+BRANCH IF NO
1157 012042 052737 000002 013066 BIS #2, FLAG ;+SET 11/60 INDICATOR
1158 012050 013716 013064 END: MOV RETURN, (R6) ;+SET UP ALTERNATE RETURN
1159 012054 000207 RTS PC ;+
1160
1161 012056 005713 ROMTYP: TST (R3) ;+ARE WE FINISHED?
1162 012060 001003 BNE 1$ ;+BRANCH IF NO
1163 012062 013716 013102 MOV FINISH, (R6) ;+SET UP ALTERNATE RETURN
1164 012066 000207 RTS PC ;+
1165
1166 012070 022713 000001 1$: CMP #1, (R3) ;+CONTINUATION CHIP?
1167 012074 001014 BNE ROMNM ;+ITS A DEVICE CHIP-FIND SOCKET #
1168 012076 004737 012126 2$: JSR PC, ROMNM ;+FIND SOCKET #
1169 012102 012705 013136 MOV #CONMES, R5 ;+LOAD MSG INTO BUFFER
1170 012106 004737 004772 JSR PC, FILBUF ;+
1171 012112 000011 HT ;+
1172 012114 062703 000004 ADD #4, R3 ;+POP OVER CONTINUATION CHIP DATA

```

1173	012120	013716	013064		MOV	RETURN. (R6)		:+SET UP ALTERNATE RETURN
1174	012124	000207			RTS	PC		:+
1175								
1176	012126	032713	000170	ROMM:	BIT	#170, (R3)		:+BEGINNING A NEW ROM?
1177	012132	001015			BNE	3\$		:+BRANCH IF NOT
1178	012134	000241		1\$:	CLC			:+
1179	012136	006101			ROL	R1		:+POINT TO NEXT SOCKET
1180	012140	030137	001116		BIT	R1, ROMFIN		:+IS A ROM THERE?
1181	012144	001003			BNE	2\$		:+BRANCH IF YES
1182	012146	062702	000002		ADD	#2, R2		:+POINT TO NEXT MSG
1183	012152	000770			BR	1\$		:+CONTINUE
1184	012154	012205		2\$:	MOV	(R2)+, R5		:+LOAD ROM ID
1185	012156	004737	00'772		JSR	PC, FILBUF		:+INTO THE MSG BUFFER
1186	012162	003015			CR			:+
1187	012164	000402			BR	4\$		:+
1188	012166	112724	000011	3\$:	MOVB	#HT, (R4)+		:+LOAD A TAB
1189	012172	000207		4\$:	RTS	PC		:+

```

1190          SBTTL SEQUENCE TEST
1191          :+THIS TEST VERIFIES THAT CONTINUATION CHIPS ARE LOCATED WHERE
1192          :+THEY BELONG, THAT THERE ARE NO DUPLICATE DEVICE ROMS,
1193          :+THAT THE ROMS ARE IN ALPHABETICAL ORDER, THAT THERE ARE NO
1194          :+HOLES, AND THAT SOCKET #2 HAS A CHIP IF THE PROCESSOR IS AN 11/60.
1195          :+IF THE ONLY PROBLEM IS ALPHABETICAL ORDER AND/OR HOLES,
1196          :+THEN A CORRECT SEQUENCE IS DETERMINED AND PRINTED OUT.
1197
1198          :+FIRST, CHECK FOR ILLEGAL CONTINUATION ROM ERRORS
1199          :+IF ANY ARE FOUND, PRINT ERRGR MSG AND EXIT
1200
1201 012174 012737 000G11 013076 SEQTST: MOV    #HT,    ARG2          :+SET UP FOR 'ERRHAN'
1202 012202 005737 013660          TST    CONER1          :+WAS THERE AN ERROR?
1203 012206 001417          BEQ    2$,          :+BRANCH IF NO
1204 012210 013703 013060          MOV    CONER1, R3          :+THE BIT IDENTIFIES THE CHIP
1205 012214 013702 013104          MOV    MESSPT, R2          :+POINT TO MSG TABLE
1206 012220 012737 000400 001110          MOV    #CONONE, ROMERR          :+SET ERROR FLAG
1207 012226 032793 000002          BIT    #2,    R3          :+IDENTIFY THE ROM FOR PRINTOUT
1208 012232 001926          BNE    3$,          :+BRANCH IF FOUND IT
1209 012234 062702 000002          ADD    #2,    R2          :+SET UP FOR NEXT ROM MSG
1210 012240 000241          CLC          :+
1211 012242 006003          ROR    R3          :+SET UP TO IDENTIFY NEXT ROM
1212 012244 000770          BR    1$,          :+
1213 012246 005737 013062          2$:  TST    CONER2          :+WAS THERE AN ERROR?
1214 012252 001423          BEQ    FILTAB          :+BRANCH IF NO
1215 012254 012737 001000 001110          MOV    #CONTWO, ROMERR          :+SET ERROR FLAG
1216 012262 012737 010576 013074          MOV    #MES1, ARG1          :+PASS THIS ADDRESS TO ERRHAN
1217 012270 000337 013062          SWAB  CONER2          :+BUILD THE MSG
1218 012274 013737 013062 010576          MOV    CONER2, MES1          :+
1219 012302 105037 010600          CLRB  MES1+2          :+
1220 012306 000402          BR    4$,          :+REPORT THE ERROR
1221 012310 011237 013074          3$:  MOV    (R2), ARG1          :+PASS MSG ADDRESS TO 'ERRHAN'
1222 012314 004737 004344          4$:  JSR    PC,    ERRHAN          :+REPORT ERROR
1223 012320 000207          RTS    PC          :+
    
```

1224  
 1225  
 1226  
 1227  
 1228  
 1229  
 1230  
 1231  
 1232  
 1233  
 1234  
 1235  
 1236  
 1237 012322 012703 013040  
 1238 012326 012702 173000  
 1239 012332 012700 000601  
 1240 012336 000241  
 1241 012340 006100  
 1242 012342 030037 001116  
 1243 012346 001404  
 1244 012350 030037 013070  
 1245 012354 001011  
 1246 012356 011213  
 1247 012360 062703 000004  
 1248 012364 062702 000200  
 1249 012370 030027 000020  
 1250 012374 001760  
 1251 012376 000404  
 1252 012400 005243  
 1253 012402 062703 000002  
 1254 012406 000766

```

        .SBTTL TABLE BUILDING ROUTINE
        ;+IF THERE WERE NO CONTINUATION ROM ERRORS, BUILD THE FOLLOWING TABLE:
        ;+
        SEQBUF:      FIRST DEVICE CODE
        ;+              # OF CONTINUATION ROMS FOR FIRST DEVICE
        ;+              SECOND DEVICE CODE
        ;+              # OF CONTINUATION ROMS FOR SECOND DEVICE
        ;+
        ;+
        ;+
        ;+
        ;+
        ;+
        ENDSEQ:      FOURTH (LAST) DEVICE CODE
        ;+
        ;+FOR A 'HOLE', THE DEVICE CODE WILL BE A 0.

FILTAB: MOV      #SEQBUF,      R3      ;+SET UP TO FILL A TABLE CALLED 'SEQBUF'
        MOV      #173000,     R2      ;+POINT TO FIRST ROM
        MOV      #1,         R0      ;+
        1$: CLC
        ROL      RO
        BIT      RO,          ROMFIN
        BEQ      3$
        2$: BIT      RO,          CONFIN
        LNE      4$
        MOV      (R2),        (R3)
        3$: ADD      #4,         R3      ;+POINT TO NEXT DEVICE CODE LOCATION
        5$: ADD      #200,       R2      ;+POINT TO NEXT ROM
        BIT      RO,          #20
        BEQ      1$
        BR      ALPHCK
        4$: IMC      -(R3)
        ADD      #2,         R3      ;+POINT TO NEXT DEVICE CODE LOCATION
        BR      5$
    
```

```

1255          .SBTTL CHECK 'SEQBUF' ROUTINE
1256          .+NOW CHECK DEVICE CODES IN 'SEQBUF' FOR ALPHABETICAL ORDER AND
1257          .+CHECK FOR 'HOLES'. IF THERE IS A DUPLICATE DEVICE CODE, PRINT
1258          .+AN ERROR MESSAGE AND EXIT. IF DEVICE CODES NEED SORTING
1259          .+AND/OR HOLES FILLING, DO IT AND SET THE SEQUENCE ERROR
1260          .+FLAG. ALSO CHECK FOR THE 11/60 SPECIAL CASE.
1261
1262 012410 012703 013040 ALPHCK: MOV #SEQBUF, R3 .+SET UP TO CHECK ALPHABETIC SEQUENCE IN 'SEQBUF'
1263 012414 013004 18: MOV R3, R4 .+WILL BE COMPARING (R3) TO (R4)
1264 012416 062704 000004 ADD #4, R4 .+WHERE R3 AND R4 POINT TO DEVICE CODE LOCATIONS
1265 012422 020427 013054 CMP R4, #ENDSEQ .+PAST THE LAST DEVICE CODE?
1266 012426 101010 BHI 38 .+BRANCH IF YES
1267 012430 005713 TST (R3) .+IS THERE A DEVICE CODE HERE?
1268 012432 001446 BEQ 78 .+BRANCH IF NO
1269 012434 005714 TST (R4) .+IS THERE A DEVICE CODE HERE?
1270 012436 001767 BEQ 28 .+BRANCH IF NO
1271 012440 021314 CMP (R3), (R4) .+IS THE SEQUENCE CORRECT?
1272 012442 001410 BEQ 48 .+BRANCH IF NO-FOUND A DUPLICATE
1273 012444 003025 BGT 58 .+BRANCH IF NO-NEED TO DO A SHIFT
1274 012446 000763 BR 28 .+YES-CONTINUE
1275 012450 062703 000004 38: ADD #4, R3 .+POINT TO NEXT DEVICE CODE
1276 012454 020327 013054 CMP R3, #ENDSEQ .+ALL DONE?
1277 012460 103056 BHS 98 .+BRANCH IF YES
1278 012462 000754 BR 18 .+NOT YET
1279 012464 012737 002000 001110 48: MOV #DUPERR, ROMERR .+SET ERROR FLAG
1280 012472 012737 010576 013074 MOV #MES1, ARG1 .+PASS THIS ADDRESS TO 'ERRHAN'
1281 012500 000314 SWAB (R4) .+BUILD THE MSG
1282 012502 011437 010576 MOV (R4), MES1 .+
1283 012506 105037 010600 CLRB MES1+2 .+
1284 012512 004737 004344 JSR PC, ERRHAN .+REPORT THE ERROR
1285 012516 000207 RTS PC .+
1286 012520 010301 58: MOV R3, R1 .+SET UP FOR THE SHUFFLE
1287 012522 010402 MOV R4, R2 .+
1288 012524 011200 MOV (R2), R0 .+SHIFT THE VALUES
1289 012526 011122 MOV (R1), (R2)+ .+
1290 012530 010021 MOV R0, (R1)+ .+
1291 012532 011200 MOV (R2), R0 .+
1292 012534 011112 MOV (R1), (R2) .+
1293 012536 010011 MOV R0, (R1) .+
1294 012540 012737 000200 001110 MOV #SEGERR, ROMERR .+SET THE SEQUENCE ERROR FLAG
1295 012546 000723 BR 28 .+CONTINUE SORTING
1296 012550 010305 78: MOV R3, R5 .+SET UP TO SEE IF WE HAVE A 'HOLE'
1297 012552 020527 013054 88: CMP R5, #ENDSEQ .+END OF TABLE?
1298 012556 103017 BHS 98 .+YES-THIS WAS NOT A 'HOLE'
1299 012560 062705 000004 ADD #4, R5 .+NO-POINT TO NEXT DEVICE CODE LOCATION
1300 012564 005715 TST (R5) .+IS THERE A ROM HERE?
1301 012566 001771 DEQ 88 .+BRANCH IF NO
1302 012570 032737 000032 001116 BIT #32, ROMFIN .+A ROM IN SOCKET #2 ONLY?
1303 012576 001004 BNE 118 .+BRANCH IF NO
1304 012600 105737 173224 TSTB 173224 .+11/60 SPECIAL CASE?
1305 012604 001401 BEQ 118 .+BRANCH IF NO
1306 012606 000422 BR 108 .+THE ONLY ROM MUST STAY IN SOCKET #2
1307 012610 010301 118: MOV R3, R1 .+YES-THERE IS A HOLE THAT CAN BE FILLED
1308 012612 010502 MOV R5, R2 .+GET READY TO SHIFT THE VALUES
1309 012614 000743 BR 68 .+FILL IN THE HOLE
1310 012616 032737 0C0002 013066 98: BIT #2, FLAG .+AN 11/60?
1311 012624 001713 BEQ 108 .+BRANCH IF NO

```

CHK 'SEQBUF' ROUTINE

```

1312 012626 005737 013044       TST   SEQBUF+4      ;+IS SOCKET #2 EMPT'?
1313 012632 001366             BNE   11$          ;+BRANCH IF NO
1314 012634 005737 013040       TST   SEQBUF       ;+IS SOCKET #1 EMPT?Y?
1315 012640 001405             BEQ   10$          ;+BRANCH IF YES
1316 012642 012701 013040       MOV   #SEQBUF, R1 ;+SET UP TO SHIFT
1317 012646 012702 013044       MOV   #SEQBUF+4, R2 ;+CHIPS #1 AND #2
1318 012652 000724             BR    6$          ;+DO THE SHIFT
1319 012654 032737 000200 001110 10$:   BIT   #SEQERR, ROMERR ;+WAS THERE A SEQUENCE ERROR?
1320 012662 001001             BNE   OUTTAB     ;+BRANCH IF YES
1321 012664 000207             RTS   PC         ;+

```

BUILDING CORRECT SEQUENCE MSG FOR 'ERRHAN' ROUTINE

```

1322          .SBTTL BUILDING CORRECT SEQUENCE MSG FOR 'ERRHAN' ROUTINE
1323          :+THE FOLLOWING ROUTINE BUILDS THE CORRECT SEQUENCE MSG TO BE
1324          :+PRINTED AS AN ERROR MSG BY 'ERRHAN'
1325
1326 012666 012737 010576 015074 OUTTAB: MOV  #MES1, ARG1      ;+PASS MSG ADR. TO 'ERRHAN'
1327 012674 012737 000015 013076      MOV  #CR, ARG2        ;+PASS THIS TOO
1328 012702 012700 013040              MOV  #SEQBUF, R0     ;+
1329 012706 013702 013104              MOV  MESSPT, R2      ;+
1330 012712 012704 010576              MOV  #MES1, R4       ;+START BUILDING THE ERROR MSG
1331 012716 012705 013377              MOV  #ERMES4, R5     ;+PUT IN THE FIRST PART
1332 012722 004737 004772              JSR  PC, FILBUF      ;+
1333 012726 000015                      CR                      ;+
1334 012730 005710                      TST  (R0)            ;+IS A ROM IN SOCKET #1?
1335 012732 001004                      BNE  1$              ;+BRANCH IF YES
1336 012734 062700                      ADD  #4, R0          ;+DO SOCKET #2 ONLY
1337 012740 062702 000002              ADD  #2, R2         ;+
1338 012744 012737 011176 013100 1$:  MOV  #MES2, BUILD     ;+SET UP THE DEVICE CODE PART
1339 012752 000310                      SWAB (R0)           ;+
1340 012754 012037 011176              MOV  (R0)+, MES2    ;+
1341 012760 105037 011200              CLRB MES2+2        ;+
1342 012764 012205 012205 2$:  MOV  (R2)+, R5       ;+IDENTIFY THE ROM #
1343 012766 004737 004772              JSR  PC, FILBUF     ;+
1344 012772 000015                      CR                      ;+
1345 012774 013705 013100              MOV  BUILD, R5      ;+PUT IN EITHER DEVICE CODE OR CONTINUATION MSG
1346 013000 004737 004772              JSR  PC, FILBUF     ;+
1347 013004 000011                      HT                      ;+
1348 013006 005720                      TST  (R0)+          ;+ANY CONTINUATION ROMS?
1349 013010 001405                      BEQ  3$              ;+BRANCH IF NO
1350 013012 005340                      DEC  -(R0)           ;+COUNT DOWN ONE ROM
1351 013014 012737 013136 013100      MOV  #COMMES, BUILD ;+SET UP FOR CONTINUATION MSG
1352 013022 000766                      BR   2$              ;+CONTINUE
1353 013024 005710 012737 3$:  TST  (R0)           ;+ANY MORE DEVICE ROMS?
1354 013026 001346                      BNE  1$              ;+BRANCH IF YES
1355 013030 105014                      CLRB (R4)           ;+MUST END WITH 0 BYTE
1356 013032 004737 004344              JSR  PC, ERRHAN     ;+REPORT THE ERROR
1357 013036 000207                      RTS  PC              ;+
1358
1359
1360 013040 000000          SEQBUF: .WORD 0      ;+FIRST DEVICE CODE
1361 013042 000000          .WORD 0      ;+# OF CONTINUATION CHIPS FOR FIRST DEVICE
1362 013044 000000          .WORD 0      ;+SECOND DEVICE CODE, ETC
1363 013046 000000          .WORD 0      ;+
1364 013050 000000          .WORD 0      ;+
1365 013052 000000          .WORD 0      ;+
1366 013054 000000          ENDSEQ: .WORD 0     ;+FOURTH (LAST) DEVICE CODE
1367 013056 000000          .WORD 0      ;+
1368 013060 000000          CONER1: .WORD 0    ;+EXTRA CONTINUATION CHIP ERROR FLAG
1369 013062 000000          CONER2: .WORD 0    ;+MISSING CONTINUATION CHIP FOR THIS DEVICE
1370 013064 000000          RETURN: .WORD 0   ;+ALTERNATE RETURN ADDRESS
1371 013066 000000          FLAG: .WORD 0    ;+BIT 1:11/60
1372 013070 000000          CONFIN: .WORD 0   ;+CONTINUATION ROM FOUND INDICATOR
1373 013072 000000          DEVIN: .WORD 0    ;+DEVICE ROM FOUND INDICATOR
1374 013074 000000          ARG1: .WORD 0    ;+PASSES MSG ADR. TO 'ERRHAN'
1375 013076 000000          ARG2: .WORD 0    ;+PASSES CR OR HT TO 'ERRHAN'
1376 013100 000000          BUILD: .WORD 0   ;+
1377 013102 000000          FINISH: .WORD 0  ;+ALTERNATE RETURN ADDRESS
1378

```

1379 013104 013176 MESSPT: .WORD MESTAB  
1380 013106 013424 MESS44: AROM1 : \*\*  
1381 013110 013440 AROM2 : \*\*  
1382 013112 013454 AROM3 : \*\*  
1383 013114 013470 AROM4 : \*\*  
1384 013116 013564 MESS24: BROM1 : \*\*  
1385 013120 013632 BROM2 : \*\*  
1386 013122 013700 BROM3 : \*\*  
1387 013124 013746 BROM4 : \*\*  
1388 013126 013504 MESTAB: RC11 :+ASCII MSG TABLE  
1389 013130 013520 ROM2 :+  
1390 013132 013534 ROM3 :+  
1391 013134 013550 ROM4 :+  
1392 013136 111 123 040 COMPRES: .ASCIZ /IS A CONTINUATION ROM/  
1393 013164 015 012 101 ERMES1: .ASCIZ <15><12>/A CONTINUATION ROM IS INCORRECTLY LOCATED IN/  
1394 013243 015 012 101 ERMES2: .ASCIZ <15><12>/A CONTINUATION ROM IS MISSING FOR DEVICE CODE/  
1395 013323 015 012 124 ERMES3: .ASCIZ <15><12>/THERE IS A DUPLICATE ROM WITH DEVICE CODE/  
1396 013377 123 105 121 ERMES4: .ASCIZ /SEQUENCE SHOULD BE: /<15>  
1397 013424 122 117 115 AROM1: .ASCIZ /ROM 1(E48) / : \*\*  
1398 013440 122 117 115 AROM2: .ASCIZ /ROM 2(E49) / : \*\*  
1399 013454 122 117 115 AROM3: .ASCIZ /ROM 3(E50) / : \*\*  
1400 013470 122 117 115 AROM4: .ASCIZ /ROM 4(E59) / : \*\*  
1401 013504 122 117 115 ROM1: .ASCIZ /ROM 1(E35) / : \*\*  
1402 013520 122 117 115 ROM2: .ASCIZ /ROM 2(E33) / : \*\*  
1403 013534 122 117 115 ROM3: .ASCIZ /ROM 3(E34) / : \*\*  
1404 013550 122 117 115 ROM4: .ASCIZ /ROM 4(E32) / : \*\*  
1405 013564 122 117 115 BROM1: .ASCII /ROM 1 - /<15><12> : \*\* :DPM001  
1406 013601 105 063 065 .ASCII /E35-M9312 /<15><12> : \*\* :DPM001  
1407 013616 105 067 065 .ASCII /E75-MAPMOD / : \*\* :DPM001  
1408 013632 122 117 115 BROM2: .ASCII /ROM 2 - /<15><12> : \*\* :DPM001  
1409 013647 105 063 063 .ASCII /E33-M9312 /<15><12> : \*\* :DPM001  
1410 013664 105 070 071 .ASCII /E89-MAPMOD / : \*\* :DPM001  
1411 013700 122 117 115 BROM3: .ASCII /ROM 3 - /<15><12> : \*\* :DPM001  
1412 013715 105 063 064 .ASCII /E34-M9312 /<15><12> : \*\* :DPM001  
1413 013732 105 070 062 .ASCII /E82-MAPMOD / : \*\* :DPM001  
1414 013746 122 117 115 BROM4: .ASCII /ROM 4 - /<15><12> : \*\* :DPM001  
1415 013763 105 063 062 .ASCII /E32-M9312 /<15><12> : \*\* :DPM001  
1416 014030 105 070 063 .ASCII /E83-MAPMOD / : \*\* :DPM001  
1417 000001 .END



SYMBOL TABLE

ABASE = 000000  
 ACW1 = 000000  
 ACW2 = 000000  
 ACPUOP = 000000  
 ADDW0 = 000000  
 ADDW1 = 000000  
 ADDW10 = 000000  
 ADDW11 = 000000  
 ADDW12 = 000000  
 ADDW13 = 000000  
 ADDW14 = 000000  
 ADDW15 = 000000  
 ADDW2 = 000000  
 ADDW3 = 000000  
 ADDW4 = 000000  
 ADDW5 = 000000  
 ADDW6 = 000000  
 ADDW7 = 000000  
 ADDW8 = 000000  
 ADDW9 = 000000  
 ADEVCT = 000000  
 ADEVN = 000000  
 AENV = 000000  
 AFATAL = 000000  
 ALPHCK 012410  
 AMADR1 = 000000  
 AMADR2 = 000000  
 AMADR3 = 000000  
 AMADR4 = 000000  
 AMARS1 = 000000  
 AMARS2 = 000000  
 AMARS3 = 000000  
 AMARS4 = 000000  
 AMSGAD = 000000  
 AMSGLG = 000000  
 AMSGTY = 000000  
 AMTYP1 = 000000  
 AMTYP2 = 000000  
 AMTYP3 = 000000  
 AMTYP4 = 000000  
 APASS = 000000  
 APRTOR = 000000  
 APTCSU = 000000  
 APTENV = 000001  
 APTER1 = 000001  
 APTER2 = 000002  
 APTER3 = 000004  
 APTER4 = 000010  
 APTSIZ = 000200  
 APTSPO = 000100  
 ARG1 013074  
 ARG2 013076  
 AROM 007306  
 AROM1 013424  
 AROM11 007200  
 AROM2 013440

AROM21 007203  
 AROM3 013454  
 AROM31 007256  
 AROM4 013470  
 AROM41 007211  
 AROM51 007214  
 AROUND 004702  
 ASUREG = 000000  
 ATESTN = 000000  
 AUNIT = 000000  
 AUSWR = 000000  
 AVECT1 = 000000  
 AVECT2 = 000000  
 BELL 007344  
 BIT0 = 000001  
 BIT00 = 000001  
 BIT01 = 000002  
 BIT02 = 000004  
 BIT03 = 000010  
 BIT04 = 000020  
 BIT05 = 000040  
 BIT06 = 000100  
 BIT07 = 000200  
 BIT08 = 000400  
 BIT09 = 001000  
 BIT11 = 000002  
 BIT110 = 002000  
 BIT111 = 004000  
 BIT112 = 010000  
 BIT113 = 020000  
 BIT114 = 040000  
 BIT115 = 100000  
 BIT2 = 000004  
 BIT3 = 000010  
 BIT4 = 000020  
 BIT5 = 000040  
 BIT6 = 000100  
 BIT7 = 000200  
 BIT8 = 000400  
 BIT9 = 001000  
 BPTVEC = 000014  
 BRHEAD 007600  
 BROM 007326  
 BROM1 013534  
 BROM11 007217  
 BROM2 013632  
 BROM21 007220  
 BROM3 013700  
 BROM31 007233  
 BROM4 013744  
 BROM41 007244  
 BROM51 007255  
 BUF1 010502  
 BUF2 010562  
 BUILD 013106  
 CALSUM 003050  
 CHECKS 002632

CKSWR = 104406  
 CLEAR 011676  
 CONER1 013060  
 CONE12 013062  
 CONF IN 013070  
 CONPES 013136  
 CONONE = 000400  
 CCATWO = 001000  
 CON1 011714  
 CON2 011742  
 CPUTYP 001122  
 CR = 000015  
 CRCERR = 000020  
 CRCLOP 003066  
 CRCMSG 010226  
 CRLF = 000200  
 DAF LAG 003646  
 DDISE = 177570  
 DEVCOD 003426  
 DEVFIN 013072  
 DIGPLA 001106  
 DISPRE 000174  
 DRHEAD 007444  
 DRH24 007524  
 DRH44 007465  
 DSWR = 177570  
 DUPERR = 002000  
 EHEAD 004744  
 EMTVEC = 000030  
 END 012050  
 ENDSEQ 013054  
 ERL 007424  
 ERMS1 013164  
 ERMS2 013243  
 ERMS3 013323  
 ERMS4 013377  
 ERRCNT 001120  
 ERPHAN 004344  
 ERRMSG 011276  
 ERROR = 104000  
 ERRVEC = 000004  
 ER1MSG 010036  
 ER2MSG 010022  
 ER3MSG 010074  
 ER4MSG 010152  
 EXCADD 003142  
 FILBUF 004772  
 FILTAB 012322  
 FINISH 013102  
 FIRSTA 003140  
 FIRSTB 001114  
 FLAG 013066  
 GTSWR = 104405  
 HOLCK1 012306  
 HOLCK2 012020  
 I = 000011  
 IOTVEC = 000020

LASTAD 003144  
 LF = 000012  
 LOOP 003060  
 MESSAG 001112  
 MESSPT 013104  
 MESS24 013110  
 MESS44 013106  
 MESTAB 013126  
 MES1 010576  
 MES2 011176  
 MFPT = 000007  
 MNORM 007350  
 NODATA 002630  
 NOROME = 000100  
 NOROMP 010330  
 NOROMS 002544  
 OCADD 005140  
 GCASC 005034  
 OCTBUF 010566  
 OUTTAB 012666  
 OVER1 004406  
 OVER10 004722  
 OVER11 004714  
 OVER2 004660  
 OVER9 004710  
 PATCHS 002344  
 PFERR = 000040  
 PFHEAD 007735  
 PFMSG 010252  
 PIRQ = 177772  
 PIRQVE = 000240  
 PPFVAR 003650  
 PROMP 003146  
 PRO = 000000  
 PR1 = 000040  
 PR2 = 000100  
 PR3 = 000140  
 PR4 = 000200  
 PR5 = 000240  
 PR6 = 000300  
 PR7 = 000340  
 PS = 177776  
 PSW = 177776  
 PUTMES 003760  
 PURVEC = 000024  
 RDCHR = 104407  
 RDLIN = 104410  
 RESREG = 104412  
 RESVEC = 000010  
 RETURN 013064  
 RPNUM1 007266  
 RPNUM2 007271  
 RPNUM3 007274  
 RPNUM4 007277  
 RPNUM5 007302  
 ROMCNT 003046  
 ROMERA 001110

POMFIN 001116  
 ROMNUM 012126  
 ROMNUM 007332  
 ROMPTR 007176  
 ROPTYP 012056  
 ROM1 013504  
 ROM2 013520  
 ROM3 013534  
 ROM4 013550  
 RSTART 002204  
 R6 = 000006  
 R7 = 000007  
 SAVREG = 104411  
 SCOPE = 000004  
 SEQBUF 013040  
 SEQERR = 002000  
 SEQMSG 010406  
 SEQT = 012174  
 STACK = 001100  
 START 001400  
 TDKLMT = 177774  
 SWR 001104  
 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  
 SW12 = 010000  
 SW13 = 020000  
 SW14 = 040000  
 SW15 = 100000  
 SW2 = 000004  
 SW3 = 000010  
 SW4 = 000020  
 SW5 = 000040  
 SW6 = 000100  
 SW7 = 000200  
 SW8 = 000400  
 SW9 = 001000  
 TBI TVE = 000014  
 TEMP 005136  
 TESTAD 003644  
 TIMES 004342  
 KVL C = 000060  
 TPVEC = 000064  
 TRAPVE = 000034  
 TRITVE = 000014

TYPE = 104401	\$DDW14 0C1250	\$FILLC 005532	SMTYP1 001171	\$TPB 005526
TYPOC = 104402	\$DDW15 001262	\$FILLS 005531	SMTYP2 001175	\$TPFLG 005533
TYPON = 104404	\$DDJ2 001230	\$GET42 002302	SMTYP3 001201	\$TPS 005524
TYPOS = 104403	\$DDW3 001232	\$GT5WR 006056	SMTYP4 001205	\$TRAP 007114
\$APTHD 001124	\$DDW4 001234	\$HD = 000003	\$NULL 005530	\$TRAP2 007136
\$ATYC 005564	\$DDW5 001236	\$HSTS 001124	\$OCNT 007014	\$TRP = 000013
\$ATY1 005540	\$DDW6 001240	\$INTAG 001101	\$OMODE 007016	\$TRPAD 007150
\$ATY3 005546	\$DDW7 001242	\$LF 005536	\$PASS 001146	\$TSTM 001130
\$ATY4 005556	\$DDW8 001244	\$LFLG 006003	\$PASTM 001132	\$TTYIN 006526
\$AUTOB 001100	\$DDW9 001246	\$MADR1 001172	\$QUES 005534	\$TYPE 005154
\$BASE 001214	\$DEVCY 001150	\$MADR2 001176	\$RDCHR 006270	\$TYPEC 005366
\$CDW1 001220	\$DEVMI 001216	\$MADR3 001202	\$RDLIN 006420	\$TYPEX 005516
\$CDW2 001222	\$DOAGN 002322	\$MADR4 001206	\$RDSZ = 000010	\$TYPOC 006616
\$CHARC 005514	\$ENDAD 002312	\$MAIL 001140	\$RESRE 007056	\$TYPON 006632
\$CKSWR 006006	\$ENDCT 002266	\$MARS1 001170	\$RTMAD 002324	\$TYPOS 005572
\$CNTLG 006543	\$ENDPG 002331	\$MARS2 001174	\$SAVRE 007020	\$UNIT 001152
\$CNTLU 006536	\$EMULL 002326	\$MARS3 001200	\$SETUP= 000124	\$UNITM 001134
\$CPUDP 001166	\$MV 001160	\$MARS4 001204	\$STUP = 177777	\$USWR 001164
\$CRLF 005535	\$ENVMI 001161	\$MBADR 001126	\$SVPC = 000210	\$SVECT1 001210
\$DDW0 001224	\$EOP 002242	\$PFLG 006002	\$SWR = 160000	\$SVECT2 001212
\$DDW1 001226	\$EOPCT 002260	\$PNEW 006561	\$SWRE: 001162	\$XOFF = 000023
\$DDW10 001250	\$ETAGL 001160	\$MSGAD 001154	\$TESTN 001144	\$XON = 000021
\$DDW11 001252	\$ETEND 001264	\$MSGGL 001156	\$TKB 005522	\$XGET4= 000000
\$DDW12 001254	\$FATAL 001142	\$MSGTY 001140	\$TKS 005520	\$OFILL 007015
\$DDW13 001256	\$FFLG 006004	\$MSWR 006550	\$TN = 000001	.\$X = 001124

. ABS. 014014 000  
 000000 001  
 ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 33328 WORDS ( 131 PAGES)  
 DYNAMIC MEMORY: 20034 WORDS ( 77 PAGES)  
 ELAPSED TIME: 00:01:23  
 CZM98E.BIN,CZM98E/CR/-SP/NL:TOC=CZM98E.MLB/ML,CZM98E.P11

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
APASE	=	000000	6-477 6-477
ACDW1	=	000000	6-477 6-477
ACDW2	=	000000	6-477 6-477
ACPUOP	=	000000	6-477 6-477
ADDW0	=	000000	6-477 6-477
ADDW1	=	000000	6-477 6-477
ADDW10	=	000000	6-477 6-477
ADDW11	=	000000	6-477 6-477
ADDW12	=	000000	6-477 6-477
ADDW13	=	000000	6-477 6-477
ADDW14	=	000000	6-477 6-477
ADDW15	=	000000	6-477 6-477
ADDW2	=	000000	6-477 6-477
ADDW3	=	000000	6-477 6-477
ADDW4	=	000000	6-477 6-477
ADDW5	=	000000	6-477 6-477
ADDW6	=	000000	6-477 6-477
ADDW7	=	000000	6-477 6-477
ADDW8	=	000000	6-477 6-477
ADDW9	=	000000	6-477 6-477
ADEVCT	=	000000	6-477 6-477
ADEVW	=	000000	6-477 6-477
AENV	=	000000	6-477 6-477
AENVW	=	000000	6-477 6-477
AFATAL	=	000000	6-477 6-477
ALPHCK	=	012410	35-1251 #36-1262
AMADR1	=	000000	6-477 6-477
AMADR2	=	000000	6-477 6-477
AMADR3	=	000000	6-477 6-477
AMADR4	=	000000	6-477 6-477
AMARS1	=	000000	6-477 6-477
AMARS2	=	000000	6-477 6-477
AMARS3	=	000000	6-477 6-477
AMARS4	=	000000	6-477 6-477
AMSGAD	=	000000	6-477 6-477
AMSGLG	=	000000	6-477 6-477
AMSGTY	=	000000	6-477 6-477
AMTYP1	=	000000	6-477 6-477
AMTYP2	=	000000	6-477 6-477
AMTYP3	=	000000	6-477 6-477
AMTYP4	=	000000	6-477 6-477
APASS	=	000000	6-477 6-477
APROR	=	000000	6-477
APTCSU	=	000040	21-1022 #22-1024
APTEW	=	000001	21-1022 22-1024 #22-1024
APTER1	=	000001	#4-446 13-671
APTER2	=	000002	#4-447 13-683
APTER3	=	000004	#4-448 13-694
APTER4	=	000010	#4-449 13-700
APTSIZ	=	000200	7-480 #22-1024
APTSPO	=	000100	21-1022 22-1024 #22-1024
ARG1	=	013074	17-902 *34-1216 *34-1221 *36-1280 *37-1326 #37-1374

## SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES
ARG2	013476	17-901 #34-1201 #37-1327 #37-1375
AROR	007306	7-490 #29-1056
AROM1	013424	37-1380 #37-1397
AROM11	007200	#29-1040 29-1056
AROM2	013440	37-1381 #37-1398
AROM21	007203	#29-1041 29-1056
AROM3	013454	37-1382 #37-1399
AROM31	007206	#29-1042 29-1056
AROM4	013470	37-1383 #37-1400
AROM41	007211	#29-1043 29-1056
AROM51	007214	#29-1044 29-1056
AROUND	004702	17-940 17-943 #17-945
ASUREG	= 000000	6-477 6-477
ATESTN	= 000000	6-477 6-477
AUMIT	= 000000	6-477 6-477
AUSMR	= 000000	6-477 6-477
AVECT1	= 000000	6-477 6-477
AVECT2	= 000000	6-477 6-477
BELL	007344	17-882 #29-1059
BIT0	= 000001	#4-442
BIT00	= 000001	#4-442 4-442
BIT01	= 000002	#4-442 4-442
BIT02	= 000004	#4-442 4-442
BIT03	= 000010	#4-442 4-442
BIT04	= 000020	#4-442 4-442
BIT05	= 000040	#4-442 4-442
BIT06	= 000100	#4-442 4-442
BIT07	= 000200	#4-442 4-442
BIT08	= 000400	#4-442 4-442
BIT09	= 001000	#4-442 4-442
BIT1	= 000002	#4-442
BIT10	= 002000	#4-442 17-879
BIT11	= 004000	#4-442
BIT12	= 010000	#4-442
BIT13	= 020000	#4-442 17-883
BIT14	= 040000	#4-442
BIT15	= 100000	#4-442 17-951
BIT2	= 000004	#4-442
BIT3	= 000010	#4-442
BIT4	= 000020	#4-442
BIT5	= 000040	#4-442
BIT6	= 000100	#4-442
BIT7	= 000200	#4-442
BIT8	= 000400	#4-442
BIT9	= 001000	#4-442
BPTVEC	= 000014	#4-442
BRHEAD	007600	16-823 #30-1066
BROM	007320	7-497 #29-1057
BROM1	013564	37-1384 #37-1405
BROM11	007217	#29-1045 29-1057
BROM2	013632	37-1385 #37-1408
BROM21	007222	#29-1046 29-1057



## SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES							
ERRVEC	= 000004	#4-442	-480	*7-480	*7-480	7-506	*7-507	*7-518	
ER1MSG	010036	17-959	#1-1071						
ER2MSG	010002	17-960	#31-1070						
ER3MSG	010074	17-961	#31-1072						
ER4MSG	010152	17-962	#31-1074						
EXCADD	003142	*11-581	*11-590	*11-603	12-631	#12-640			
FILBUF	004772	16-817	16-824	16-835	16-842	16-855	16-860	17-895	17-903
		17-931	#18-974	33-1170	33-1185	37-1332	37-1343	37-1345	17-927
FILTAB	012322	34-1214	#35-1237						
FINISH	013102	*16-829	33-1118	33-1163	#37-1377				
FIRSTA	003140	*11-582	*11-589	*11-602	11-606	12-617	#12-639		
FIRSTB	001114	#4-471	7-520	14-738					
FLAG	013066	*33-1157	36-1310	#37-1371					
GWS	= *****	4-458	4-458	7-481	28-1036	28-1036	28-1036	28-1036	28-1036
		28-1036	28-1036	28-1036	28-1036	28-1036	28-1036	28-1036	28-1036
		28-1036	28-1036	28-1036	28-1036	28-1036	28-1036	28-1036	28-1036
GTSWR	= 104405	7-481	#28-1036						
HOLCK1	012006	15-785	#33-1147						
HOLCK2	012020	15-792	#33-1151						
HT	= 000011	#4-442	16-836	16-843	16-847	16-861	17-928	17-932	18-974
		21-1022	21-1022	33-1171	33-1188	34-1201	37-1347		18-976
10TVEC	= 000020	#4-442							
LASTAD	003144	*11-580	*11-591	*11-604	12-634	#12-641			
LF	= 000012	#4-442	16-831	16-865	17-934	18-970	21-1022	21-1022	
LOOP	003060	#12-617	12-635						
MESSAG	001112	#4-470	*13-661						
MESSPT	013104	*7-489	*7-496	16-827	34-1205	37-1329	#37-1379		
MESS24	013116	7-496	#37-1384						
MESS44	013106	7-489	#37-1380						
MES1AB	013126	37-1379	#37-1388						
MES1	010576	13-653	#32-1094	34-1216	*34-1218	*34-1219	36-1280	*36-1282	*36-1283
		37-1330							37-1326
MES2	011176	13-660	#32-1095	37-1338	*37-1340	*37-1341			
MFPT	= 000007	#6-478	7-485						
MINORM	007350	10-556	#29-1060						
MODATA	002630	1-558	#10-573						
NOROME	= 000100	#4-452	10-562						
NOROPM	010330	17-965	#32-1079						
NOROMS	002544	7-536	#10-556						
OCADD	005140	16-834	16-841	16-859	#20-1017				
OCASC	005034	17-926	17-930	#19-989	20-1019				
OCTBUF	010566	7-501	19-989	19-994	19-1001	19-1003	#32-1093		
OUTTAB	012666	36-1320	#37-1326						
OVER1	004406	17-884	#17-886						
OVER10	004722	17-885	#17-951						
OVER11	004714	17-938	17-944	#17-949					
OVER2	004660	17-937	#17-939						
OVER9	004710	*11-887	#17-947						
PATCHS	002344	#9-548							
PFERR	= 000040	#4-451	13-655	15-796	17-897	17-941			
PFHEAD	007735	16-854	#31-1069						
PFMSG	010252	17-964	#31-1077						

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
PIRQ	=	177772	#4-412
PIRQVE	=	000240	#4-442
PPFVAR	=	003650	13-654 13-689 #15-762
PROMP	=	003145	7-541 #13-649
PRO	=	000000	#4-412
PR1	=	000040	#4-442
PR2	=	000100	#4-442
PR3	=	000140	#4-442
PR4	=	000200	#4-442
PR5	=	000240	#4-442
PR6	=	000300	#4-442
PR7	=	000340	#4-442
PS	=	177776	#4-442 4-412
PSW	=	177776	#4-442
PUMES	=	003760	13-652 13-659 #16-801
PURVEC	=	000024	#4-442
RDCHR	=	104407	23-1026 #28-1036
RDLIN	=	104410	#28-1036
RESREG	=	104412	17-955 #28-1036
RESVEC	=	000010	#4-442
RETURN	=	013064	*15-784 *15-791 *16-828 33-1158 33-1173 #37-1370
ROMNUM1	=	007266	#29-1050 29-1058
ROMNUM2	=	007271	#29-1051 29-1058
ROMNUM3	=	007274	#29-1052 29-1058
ROMNUM4	=	007277	#29-1053 29-1058
ROMNUM5	=	007302	#29-1054 29-1058
ROMCNT	=	003046	*11-577 11-578 *11-593 11-594 #11-610 17-910
ROMERR	=	001110	#4-469 *10-562 *11-587 *11-600 13-655 *13-671 *13-683 *13-694 *13-700 *15-795 17-886 17-887 17-897 17-899 17-906 17-908 17-941 *17-954 *34-1206 *34-1215 *36-1279 *36-1294 36-1319 #4-472 *7-502 *7-514 *7-526 *7-528 7-534 11-578 11-594 14-732 17-939 33-1147 33-1151 33-1180 35-1242 36-1302
ROMFIN	=	001116	17-939 33-1147 33-1168 #33-1176
ROMINH	=	012126	33-1167
ROMINUM	=	007332	29-1039 #29-1058
ROMPTR	=	007176	*7-490 *7-497 17-911 #29-1039
ROMTYP	=	012046	16-832 #33-1161
ROM1	=	013504	37-1388 #37-1401
ROM2	=	013520	37-1389 #37-1402
ROM3	=	013534	37-1390 #37-1403
ROM4	=	013550	37-1391 #37-1404
RSTART	=	002204	4-461 #7-534 8-543
R6	=	X000006	#4-442 *7-506 *7-517 *7-518 16-801 16-867 16-870 17-925 17-929 18-974 18-983 *20-1017 *20-1020 33-1158 33-1163 33-1173
R7	=	X000007	#4-442 17-876 #28-1036
SAVREG	=	104411	#4-442
SCOPE	=	000004	#4-442
SEQBUF	=	013040	33-1116 35-1237 36-1262 36-1312 36-1314 36-1316 36-1317 37-1328 #37-1360
SEQERR	=	000200	#4-453 36-1294 36-1319
SEQMSG	=	010406	17-966 #32-1081
SEQTST	=	012174	7-542 #34-1201
STACK	=	001100	#4-442 7-480
START	=	001400	4-458 #7-480

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES								
SYMBOL			#4-442								
STK_MT	=	177774	#4-467	*7-480	7-480	*7-480	*7-480	7-481	17-879	17-823	17-951
SWR	=	001104	23-1026	23-1026							
SWREG	=	000176	#4-458	7-480	7-481	23-1026	23-1026				
SW0	=	000001	#4-442								
SW00	=	000001	#4-442	4-442							
SW01	=	000002	#4-442	4-442							
SW02	=	000004	#4-442	4-442							
SW03	=	000010	#4-442	4-442							
SW04	=	000020	#4-442	4-442							
SW05	=	000040	#4-442	4-442							
SW06	=	000100	#4-442	4-442							
SW07	=	000200	#4-442	4-442							
SW08	=	000400	#4-442	4-442							
SW09	=	001000	#4-442	4-442							
SW1	=	000002	#4-442								
SW10	=	002000	#4-442								
SW11	=	004000	#4-442								
SW12	=	010000	#4-442								
SW13	=	020000	#4-442								
SW14	=	040000	#4-442								
SW15	=	100000	#4-442								
SW2	=	000004	#4-442								
SW3	=	000010	#4-442								
SW4	=	000020	#4-442								
SW5	=	000040	#4-442								
SW6	=	000100	#4-442								
SW7	=	000200	#4-442								
SW8	=	000400	#4-442								
SW9	=	001000	#4-442								
TBITVE	=	000014	#4-442								
TEMP	=	005136	*19-995	*19-997	*19-1005	*19-1007	*19-1009	#19-1012			
TESTAD	=	003644	*7-520	7-521	7-523	*7-530	7-532	*14-734	14-735	14-736	*14-738
			*14-743	14-748	14-749	*14-750	14-752	14-753	*14-753	14-754	*14-755
			14-766	#14-770							
			*7-504	16-802	*16-804	#16-772					
TIMES	=	004342	*7-504	16-802	*16-804	#16-772					
TKVEC	=	000040	#4-442								
TPVEC	=	000064	#4-442								
TRAPVE	=	000034	#4-442	*7-480	*7-480						
TRTVEC	=	000014	#4-442								
TYPE	=	104401	7-481	8-543	8-543	10-556	14-868	17-881	21-1022	23-1026	23-1026
			23-1026	23-1026	23-1026	23-1026	23-1026	21-1022	23-1026	24-1028	#28-1036
TYPGC	=	104402	23-1026	#28-1036							
TYPON	=	104404	#28-1036								
TYPOS	=	104403	#28-1036								
SAPTHD	=	001124	5-475	#5-475							
SASTAT	=	*****	22-1024	22-1024	22-1024						
SATYC	=	005564	22-1024	#22-1024							
SATY1	=	005540	17-945	#22-1024							
SATY3	=	005546	17-949	21-1022	#22-1024						
SATY4	=	005556	#22-1024								
SAUTOB	=	001100	#4-464	*7-481	23-1026	23-1026	23-1026				



SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES
SPAL23	001202	#6-477
SPADR4	001206	#6-477
SPAIL	001140	5-475 #6-477 7-480 7-481 21-1022
SPARS1	001170	#6-477
SPARS2	001174	#6-477
SPARS3	001200	#6-477
SPARS4	001204	#6-477
SPADR	001126	#5-475
SPFLG	006002	*22-1024 22-1024 #22-1024
SPNEW	006561	23-1026 #23-1026
SPRGAD	001154	#6-477 *22-1024 22-1024
SPRGLG	001156	#6-477 *22-1024
SPSGTY	001140	#6-477 22-1024 *22-1024 22-1024 *22-1024
SPSMR	006550	23-1026 #23-1026
SPITY1	001171	#6-477
SPITY2	001175	#6-477
SPITY3	001201	#6-477
SPITY4	001205	#6-477
SNULL	005530	21-1022 21-1022 #21-1022
SOCNT	007014	*24-1028 #24-1028
SOWIDE	007016	*24-1028 24-1028 *24-1028 *24-1028 #24-1028
SPASS	001146	#6-477 *7-480 *7-480 7-539 *8-54 *8-543 8-543 8-543
SPASTM	001132	#5-475
SOUES	005534	21-1022 21-1022 #21-1022 23-1026 23-1026 23-1026 23-1026
SRDCHR	006270	#23-1026 28-1036
SRDDEC	= *****	28-1036
SRDLIN	006420	#23-1026 28-1036
SRDOCT	= *****	28-1036
SRDSZ	= 000010	#22-1026 23-1026
SRESRE	007056	#25-1030 28-1036
SRTNAD	002324	#8-543
SR2A	= *****	28-1036
SSAVRE	007020	#2-1030 28-1036 28-1036
SETUP	= 000124	#4-459 4-459 #4-459 4-459 #4-459 4-459 #4-459 7-480 7-480 7-480 #4-459 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480
STUP	= 177777	7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480
SVPC	= 000210	7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480
SMR	= 160000	7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480 7-480
SMREG	001162	#4-459 #4-459 4-459 #4-459 #4-459 #4-459 #4-459 #4-459 #4-459 #4-459 #4-459 #4-459 #4-459 #4-459 #4-459 #4-459 #4-459 #4-459 #4-459 #4-459
STESTN	001144	4-462 4-462 4-462 4-462 4-462 4-462 4-462 4-462 4-462 4-462 4-462 4-462 4-462 4-462 4-462 4-462 4-462 4-462 4-462
STKB	005522	4-440 #4-440 4-440 4-440 4-440 4-440 4-440 4-440 4-440 4-440 4-440 4-440 4-440 4-440 4-440 4-440 4-440 4-440 4-440 4-440
STKS	005520	#6-477 7-480 8-543 8-543 8-543 8-543 8-543 8-543 8-543 8-543 8-543 8-543 8-543 8-543 8-543 8-543 8-543 8-543 8-543 8-543
STN	= 000001	21-1022 21-1022 21-1022 21-1022 #21-1022 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026
STPB	005526	23-1026 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026 23-1026
STPFLG	005533	21-1022 21-1022 21-1022 21-1022 #21-1022 21-1022 21-1022 21-1022 21-1022 21-1022 21-1022 21-1022 21-1022 21-1022 21-1022 21-1022 21-1022 21-1022 21-1022 21-1022
STPS	005524	21-1022 21-1022 21-1022 21-1022 #21-1022 21-1022 21-1022 21-1022 21-1022 21-1022 21-1022 21-1022 21-1022 21-1022 21-1022 21-1022 21-1022 21-1022 21-1022 21-1022
STRAP	007114	7-480 #28-1036
STRAP2	007136	#28-1036 28-1036

SYMBOL CROSS REFERENCE SEQUENCE 67  
 SYMBOL VALUE REFERENCES CREF V01

SYMBOL	CROSS REFERENCE	REFERENCES	CREF	V01
*TRP	= 000013	#28-1036 28-1036	28-1036	28-1036
		28-1036 #28-1036	28-1036	28-1036
		28-1036 28-1036	#28-1036	28-1036
		28-1036 28-1036	28-1036	28-1036
		28-1036 28-1036	#28-1036	28-1036
		28-1036 28-1036	28-1036	28-1036
		#28-1036 28-1036	28-1036	#28-1036
		28-1036 #28-1036	28-1036	28-1036
\$TRPAD	007150	#28-1036	28-1036	28-1036
\$TSTM	001130	#5-475		
\$TTYIN	006526	23-1026	23-1026	#23-1026
\$TYBWN	= *****	28-1036		
\$TYPDS	= *****	28-1036		
\$TYPE	005154	#21-1022 22-1024	28-1036	28-1036
\$TYPEC	005366	21-1022 21-1022	21-1022	#21-1022 23-1026
\$TYPEX	005516	21-1022 21-1022	21-1022	#21-1022
\$TYPOC	006616	#24-1028 28-1036	28-1036	
\$TYPON	006632	24-1028 #24-1028	28-1036	
\$TYPOS	006572	#24-1028 28-1036		
\$UNIT	001152	#6-477		
\$UNITM	001134	#5-475		
\$USWR	001164	#6-477		
\$VECT1	001210	#6-477		
\$VECT2	001212	#6-477		
\$XOFF	= 000023	21-1022 21-1022		
\$XON	= 000021	21-1022	21-1022	23-1026
\$SGE74	= 000000	#8-543 8-543		
\$OFILL	= 007015	*24-1028 *24-1028	24-1028	#24-1028
.\$ASTA	= *****	22-1024 22-1024		
.\$X	= 001124	#5-475 5-475		



MACRO CROSS REFERENCE

MACRO NAME	REFERENCES
STRAP	#4-437 #28-1036
STYPE	#4-435 #21-1022
STYPO	#4-438 24-1028

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES						
\$BASE	001214	#6-477	13-690					
\$CDW1	001220	#6-477	13-696					
\$CDW2	001222	#6-477						
\$CHARC	005514	*21-1022	*21-1022	21-1022	*21-1022	#21-1022		
\$CKSWR	006006	#23-1026	28-1036	28-1036				
\$CRTAG	*****	7-480	7-480					
\$CNTLG	006543	23-1026	#23-1026					
\$CNTLU	006536	23-1026	#23-1026					
\$CPUOFF	001166	#6-477						
\$CRLF	005535	21-1022	21-1022	21-1022	#21-1022	23-1026	23-1026	23-1026
\$DDW0	001224	#6-477	13-665	13-676				
\$DDW1	001226	#6-477						
\$DDW10	001250	#6-477						
\$DDW11	001252	#6-477						
\$DDW12	001254	#6-477	13-687					
\$DDW13	001256	#6-477						
\$DDW14	001260	#6-477						
\$DDW15	001262	#6-477						
\$DDW2	001230	#6-477						
\$DDW3	001232	#6-477						
\$DDW4	001234	#6-477						
\$DDW5	001236	#6-477						
\$DDW6	001240	#6-477						
\$DDW7	001242	#6-477						
\$DDW8	001244	#6-477						
\$DDW9	001246	#6-477						
\$DEVCT	001150	#6-477						
\$DEVPM	001216	#6-477						
\$DOAGN	002322	8-543	P-543	#8-543				
\$ENDAD	002312	4-462	7-481	#8-543				
\$ENDCT	002266	7-480	#8-543					
\$ENDMG	002331	8-543	#8-543					
\$ENJLL	002326	8-543	#8-543					
\$ENV	001160	#6-477	7-481	17-236	21-1022	22-1024	22-1024	
\$ENVPM	001161	#6-477	7-480	13-649	21-1022	21-1022	22-1024	
\$EOP	002242	7-537	7-540	#8-543				
\$EOPCT	002260	*7-480	#8-543	8-543				
\$ETABL	001160	#6-477						
\$ETEND	001264	5-475	#6-477					
\$FATAL	001142	#6-477	*22-1024					
\$FFLG	006004	*22-1024	*22-1024	22-1024	*22-1024	#22-1024		
\$FILLC	005512	21-1022	21-1022	21-1022	#21-1022			
\$FILLS	005531	21-1022	21-1022	#21-1022				
\$GET42	002302	#8-543						
\$GTSWR	006056	#23-1026	28-1036	28-1036				
\$HD	000003	4-440	4-440	4-440				
\$HIBTS	001124	#5-475						
\$INTAG	001101	#4-465	23-1026	23-1026				
\$LF	005536	21-1022	21-1022	#21-1022	23-1026	23-1026	23-1026	
\$LFLG	006003	*22-1024	#22-1024					
\$MADR1	001172	#6-477						
\$MADR2	001176	#6-477						