

KD11-Z,
M9312

M9312/1144 UBI BOOT
CZM9BE0

AH-E058E-MC
FICHE 1 OF 1

JUL 1982
COPYRIGHT © 78-82
MADE IN USA



.REF !

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-E057E-MC
PRODUCT NAME: CZM9BE0 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 DITITAL 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 EQUIPEMNT CORPORATION

HISTORY SECTION

PBRDAO WAS RELEASED MARCH, 1978.
PBRDBO WAS RELEASED JUNE, 1978.
PBRD440 WAS RELEASED JANUARY, 1979
C7M9BDO WAS RELEASED OCTOBER 1979
C7M9BEO 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 PRECEEDING 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 CHARATERS ":-"
IDENTIFY ALL LINES ADDED OR MODIFIED BY REV. C.

REVISION D WAS CREATED TO ACCOMODATE 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 "FPT 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 ACCOMODATE 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 'DEVCOD:' SUBROUTINES, USED WHEN IN APT MODE WHEN BIT 7 OF \$ENVM 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 UBI 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 PROCEDJRES 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-\$ENVM) 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 COMPARSION
FEATURE THE ETABLE MUST BE SET UP IN THIS ORDER:

\$ENV:	DON'T CARE
\$ENVM:	200 OR 240
\$SWREG:	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 \$MAMS4: NOT USED
 214 \$MTYP4: NOT USED
 215 \$MADR4: NOT USED
 216 \$VECT1: NOT USED
 217 \$VECT2: 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 165020
 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 240
 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
040480----	AO-----	248F1-----
040160	BO	233F1
041460	CO	
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 RECIEVES THE CALCULATED CHECKSUM FROM "CALSUM" AND COMPARES IT AGAINST THE GOOD CHECKSUM TO DETERMINE CRC ERRORS.

5.3 CALSUM

213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269

270 THIS ROUTINE CALCULATES THE CRC16 CHECKSUM OF EACH ROM. IT
271 RECEIVES THE FIRST ADDRESS TO BE CHECKED(FIRSTA), THE LAST
272 ADDRESS TO BE CHECKED(LASTAD) AND THE EXCEPTION ADDRESS (EXCADD)
273 FROM THE "CHECKS" MODULE AND RETURNS TO IT THE CHECKSUM IN R4.
274
275
276 5.4 PROMP
277
278
279 THIS ROUTINE PROCESSES THE ROM PARAMETERS. IT CHECKS THE SIZE/
280 DON'T SIZE BIT IN THE APT ETABLE AND EITHER FORMATS THE SIZING
281 MESSAGES OR COMPARES THE SIZING INFORMATION TO THE APT ETABLE
282 DATA.
283
284
285 5.5 DEVCOD
286
287
288 THIS ROUTINE LOCATES EACH DEVICE CODE AND PASSES IT AND THE
289 ADDRESS IN WHICH IT WAS FOUND BACK TO THE "PROMP" MODULE.
290
291
292 5.6 PPFVAR
293
294
295 THIS ROUTINE DETERMINES THE PSEUDO POWER-FAIL VECTOR ADDRESS AND
296 PASSES IT AND ITS CONTENTS TO THE "PROMP" MODULE
297
298
299 5.7 PUTMES
300
301
302 THIS ROUTINE FORMATS THE PARAMETER AND POWER-FAIL ADDRESS
303 MESSAGES.
304
305
306 5.8 ERRHAN
307
308
309 THIS ROUTINE FORMATS ALL ERROR MESSAGES AND CALLS THE TYPE
310 ROUTINE TO OUTPUT THEM. IT ALSO USES THE OPERATIONAL SWITCHES TO
311 DETERMINE WHETHER TO OUTPUT THE MESSAGE, OR HALT.
312
313
314 5.9 FILBUF
315
316
317 THIS ROUTINE FILLS THE MESSAGE BUFFER WITH ASCII CHARACTERS. IT
318 RECIEVES THE ADDRESS OF THE ASCII IN R5.
319
320
321 5.10 OCASC
322
323
324 THIS ROUTINE TAKES A SIXTEEN BIT BINARY NUMBER AND CONVERTS IT TO
325 6 ASCII CHARACTERS.
326

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

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
-----	-----

384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406

WHERE EXPECTED IS THE CONTENTS OF THE ETABLE AND RECEIVED IS THE VALUE FOUND BY THE PROGRAM.

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

POWER-FAIL DATA ERROR
EXPECTED RECEIVED

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
423
424
425
426
427
428
435
436
437
438
439
440
      .ENABLE ABS
      .ENABL  AMA
      .LIST   ME
      .NLIST  MC
      .NLIST  MD
      .NLIST  CND
      .NLIST  BEX
      .MCALL  .SASTAT,.SSAVE,.SCATCH,.SCMTAG,.SETUP,.SEOP,.STYPE
      .MCALL  STARS,.EQUATE,.SAPTHDR,.SAPTBL,.SAPTYPE
      .MCALL  .SACT11,.HEADER,.SWRHI,REPORT,TYPNAM,PUSH,POP,.STRAP
      .MCALL  .SREAD,.STYPOCT
      .MCALI  .EQUIV
      .TITLE  CZM9BE0 M9312/1144 KT24 UBI BOOT
      ;*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.
      ;*
      $TN=1
      $SWR=160000 ;:HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP OUT
      .SBTTL  OPERATIONAL SWITCH SETTINGS
      ;*
      ;*      SWITCH          USE
      ;*      -----          -----
      ;*      15             HALT ON ERROR
      ;*      14             LOOP ON TEST
      ;*      13             INHIBIT ERROR TYPEOUTS
      .SBTTL  BASIC DEFINITIONS
      ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
      STACK= 1100
      ERROR=EMT
      SCOPE=IOT
      ;*MISCELLANEOUS DEFINITIONS
      HT= 11 ;:CODE FOR HORIZONTAL TAB
      LF= 12 ;:CODE FOR LINE FEED
      CR= 15 ;:CODE FOR CARRIAGE RETURN
      CRLF= 200 ;:CODE FOR CARRIAGE RETURN-LINE FEED
      PS= 177776 ;:PROCESSOR STATUS WORD
      PSW=PS
      STKLMT= 177774 ;:STACK LIMIT REGISTER
      PIRQ= 177772 ;:PROGRAM INTERRUPT REQUEST REGISTER
      DSWR= 177570 ;:HARDWARE SWITCH REGISTER
      DDISP= 177570 ;:HARDWARE DISPLAY REGISTER
      ;*GENERAL PURPOSE REGISTER DEFINITIONS
      R0= %0 ;:GENERAL REGISTER
      R1= %1 ;:GENERAL REGISTER
      R2= %2 ;:GENERAL REGISTER
      R3= %3 ;:GENERAL REGISTER
      R4= %4 ;:GENERAL REGISTER
      R5= %5 ;:GENERAL REGISTER
      R6= %6 ;:GENERAL REGISTER
      R7= %7 ;:GENERAL REGISTER
001100
104000
000004
000011
000012
000015
000200
177776
177776
177774
177772
177570
177570
000000
000001
000002
000003
000004
000005
000006
000007

```

BASIC DEFINITIONS

```

000006 SP= %6 ;;STACK POINTER
000007 PC= %7 ;;PROGRAM COUNTER
;*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
;* 'SWITCH REGISTER' SWITCH DEFINITIONS
100000 SW15= 100000
040000 SW14= 40000
020000 SW13= 20000
010000 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
001000 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
;*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

```



```

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
000004          ERRVEC= 4          ;; TIME OUT AND OTHER ERRORS
000010          RESVEC= 10         ;; RESERVED AND ILLEGAL INSTRUCTIONS
000014          TBITVEC=14        ;; 'T' BIT
000014          TRTVEC= 14         ;; TRACE TRAP
000014          BPTVEC= 14         ;; BREAKPOINT TRAP (BPT)
000020          IOTVEC= 20         ;; INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024          PWRVEC= 24         ;; POWER FAIL
000030          EMTVEC= 30         ;; EMULATOR TRAP (EMT) **ERROR**
000034          TRAPVEC=34         ;; "TRAP" TRAP
000060          TKVEC= 60          ;; TTY KEYBOARD VECTOR
000064          TPVEC= 64          ;; TTY PRINTER VECTOR
000240          PIRQVEC=240        ;; PROGRAM INTERRUPT REQUEST VECTOR

*****
;*
*****
                                PROGRAM EQUATES
*****
443          000001          APTER1 = 1
444          000002          APTER2 = 2
445          000004          APTER3 = 4
446          000010          APTER4 = 10
447          000020          CRCERR = 20
448          000040          PFERR = 40
449          000100          NOROME =100
450          000200          SEQERR =200
451          000400          CONONE =400
452          001000          CONTWO=1000
453          002000          DUPERR=2000
454          0C0000          .=0
455          .SBTTL TRAP CATCHER
456          .=0
457          ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
458          ;*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)
460          000200          000137          001400          JMP @#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
    
```

463 001100
464 001100 000
465 001101 000
466 001102 000000
467 001104 177570
468 001106 177570
469 001110 000000
470 001112 000000
471 001114 173000
472 001116 000000
473 001120 000000
474 001122 000000

. =1100
\$AUTOB: .BYTE 0
\$INTAG: .BYTE 0
.WORD 0
SWR: .WORD DSWR
DISPLAY: .WORD DDISP
ROMERR: 0
MESSAG: 0
FIRSTB: 173000
ROMFIN: 0
ERRCNT: 0
CPUTYP: 0

475

000024 001124
000024 000024
000044 000200
000044 000044
001124 001124
001124 001124

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

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


477

.SBTTL APT MAILBOX-ETABLE

001140
001140 000000
001142 000000
001144 000000
001146 000000
001150 000000
001152 000000
001154 000000
001156 000000
001160
001160 000
001161 000
001162 000000
001164 000000
001166 000000
001170 000
001171 000
001172 000000
001174 000
001175 000
001176 000000
001200 000
001201 000
001202 000000
001204 000
001205 000
001206 000000
001210 000000
001212 000000
001214 000000
001216 000000
001220 000000
001222 000000
001224 000000
001226 000000
001230 000000
001232 000000
001234 000000
001236 000000
001240 000000
001242 000000
001244 000000
001246 000000

.EVEN
\$MAIL: ::APT MAILBOX
\$MSGTY: .WORD AMSGTY ::MESSAGE TYPE CODE
\$FATAL: .WORD AFATAL ::FATAL ERROR NUMBER
\$TESTN: .WORD ATESTN ::TEST NUMBER
\$PASS: .WORD APASS ::PASS COUNT
\$DEVCT: .WORD ADEVCT ::DEVICE COUNT
\$UNIT: .WORD AUNIT ::I/O UNIT NUMBER
\$MSGAD: .WORD AMSGAD ::MESSAGE ADDRESS
\$MSGLG: .WORD AMSGLG ::MESSAGE LENGTH
\$ETABLE: ::APT ENVIRONMENT TABLE
\$ENV: .BYTE AENV ::ENVIRONMENT BYTE
\$ENVM: .BYTE AENVM ::ENVIRONMENT MODE BITS
\$SWREG: .WORD ASWREG ::APT SWITCH REGISTER
\$USWR: .WORD AUSWR ::USER SWITCHES
\$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,PDQ=07,Q=10
BIT 10=REAL TIME CLOCK
BIT 9=FLOATING POINT PROCESSOR
BIT 8=MEMORY MANAGEMENT
\$MAMS1: .BYTE AMAMS1 ::HIGH ADDRESS,M.S. BYTE
\$MTYP1: .BYTE AMTYP1 ::MEM. TYPE,BLK#1
MEM.TYPE BYTE -- (HIGH BYTE)
900 NSEC CORE=001
300 NSEC BIPOLAR=002
500 NSEC MOS=003
\$MADR1: .WORD AMADR1 ::HIGH ADDRESS,BLK#1
MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE
\$MAMS2: .BYTE AMAMS2 ::HIGH ADDRESS,M.S. BYTE
\$MTYP2: .BYTE AMTYP2 ::MEM. TYPE,BLK#2
\$MADR2: .WORD AMADR2 ::MEM.LAST ADDRESS,BLK#2
\$MAMS3: .BYTE AMAMS3 ::HIGH ADDRESS,M.S.BYTE
\$MTYP3: .BYTE AMTYP3 ::MEM. TYPE,BLK#3
\$MADR3: .WORD AMADR3 ::MEM.LAST ADDRESS,BLK#3
\$MAMS4: .BYTE AMAMS4 ::HIGH ADDRESS,M.S.BYTE
\$MTYP4: .BYTE AMTYP4 ::MEM. TYPE,BLK#4
\$MADR4: .WORD AMADR4 ::MEM.LAST ADDRESS,BLK#4
\$VECT1: .WORD AVECT1 ::INTERRUPT VECTOR#1,BUS PRIORITY#1
\$VECT2: .WORD AVECT2 ::INTERRUPT VECTOR#2BUS PRIORITY#2
\$BASE: .WORD ABASE ::BASE ADDRESS OF EQUIPMENT UNDER TEST
\$DEVN: .WORD ADEVN ::DEVICE MAP
\$CDW1: .WORD ACDW1 ::CONTROLLER DESCRIPTION WORD#1
\$CDW2: .WORD ACDW2 ::CONTROLLER DESCRIPTION WORD#2
\$DDW0: .WORD ADDW0 ::DEVICE DESCRIPTOR WORD#0
\$DDW1: .WORD ADDW1 ::DEVICE DESCRIPTOR WORD#1
\$DDW2: .WORD ADDW2 ::DEVICE DESCRIPTOR WORD#2
\$DDW3: .WORD ADDW3 ::DEVICE DESCRIPTOR WORD#3
\$DDW4: .WORD ADDW4 ::DEVICE DESCRIPTOR WORD#4
\$DDW5: .WORD ADDW5 ::DEVICE DESCRIPTOR WORD#5
\$DDW6: .WORD ADDW6 ::DEVICE DESCRIPTOR WORD#6
\$DDW7: .WORD ADDW7 ::DEVICE DESCRIPTOR WORD#7
\$DDW8: .WORD ADDW8 ::DEVICE DESCRIPTOR WORD#8
\$DDW9: .WORD ADDW9 ::DEVICE DESCRIPTOR WORD#9

APT MAILBOX-ETABLE

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

\$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:

478 000007

MFPT=000007

```

479      001400      . =1400
480 001400
      001400 012706 001100      START:
      .SBTTL INITIALIZE THE COMMON TAGS
      MOV #STACK,SP          ;; SETUP THE STACK POINTER
      ;; INITIALIZE A FEW VECTORS
      MOV #STRAP,@#TRAPVEC   ;; TRAP VECTOR FOR TRAP CALLS
      MOV #340,@#TRAPVEC+2;LEVEL 7
      CLR $PASS              ;; CLEAR THE PASS COUNT
      MOV $ENDCT,$EOPCT     ;; 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.
      MOV @#ERRVEC,-(SP)    ;; SAVE ERROR VECTOR
      MOV #64$,@#ERRVEC    ;; SET UP ERROR VECTOR
      MOV #DSWR,SWR        ;; SETUP FOR A HARDWARE SWICH REGISTER
      MOV #DDISP,DISPLAY   ;; AND A HARDWARE DISPLAY REGISTER
      CMP #-1,@SWR         ;; TRY TO REFERENCE HARDWARE SWR
      BNE 66$              ;; BRANCH IF NO TIMEOUT TRAP OCCURRED
      ;; AND THE HARDWARE SWR IS NOT = -1
      BR 65$               ;; BRANCH IF NO TIMEOUT
      64$: MOV #65$,(SP)   ;; SET UP FOR TRAP RETURN
      RTI
      65$: MOV #SWREG,SWR  ;; POINT TO SOFTWARE SWR
      MOV #DISPREG,DISPLAY
      66$: MO' (SP)+,@#ERRVEC ;; RESTORE ERROR VECTOR
      CLR $PASS           ;; CLEAR PASS COUNT
      BITB #APTSIZE,$ENVM ;; TEST USER SIZE UNDER APT
      BEQ 67$             ;; YES,USE NON-APT SWITCH
      MOV #SSWREG,SWR    ;; NO,USE APT SWITCH REGISTER
      67$:
      .SBTTL TYPE PROGRAM NAME
      ;; TYPE THE NAME OF THE PROGRAM IF FIRST PASS
      INC #-1            ;; FIRST TIME?
      BNE 68$           ;; BRANCH IF NO
      CMP #SENDAD,@#42  ;; ACT-11?
      BEQ 68$           ;; BRANCH IF YES
      TYPE ,69$         ;; TYPE ASCIZ STRING
      .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
      TST @#42          ;; ARE WE RUNNING UNDER XXDP/ACT?
      BNE 70$           ;; BRANCH IF YES
      CMPB $ENV,#1     ;; ARE WE RUNNING UNDER APT?
      BEQ 70$           ;; BRANCH IF YES
      CMP SWR,#SWREG   ;; SOFTWARE SWITCH REG SELECTED?
      BNE 71$           ;; BRANCH IF NO
      GTSWR            ;; GET SOFT-SWR SETTINGS
      BR 71$
      70$: MOVB #1,$AUTOB ;; SET AUTO-MODE INDICATOR
      71$:
      BR 68$           ;; GET OVER THE ASCIZ
      ;; 69$: .ASCIZ <CRLF>*CZM9BE0 M9312/1144 KT24 UBI BOOT*<CRLF>
      68$:
      MOV 10,-(SP)     ;; ** SAVE VECTOR ON STACK
      MOV #10$,10     ;; ** SET ERROR VECTOR
      CLR CPUTYP      ;; CLEAR CPU TYPE LOCATION ;DPM001
      MFPT           ;; ** WHAT CPU IS IT
      MOVB R0,CPUTYP  ;; SAVE CPU TYPE IN 'CPUTYP' ;DPM001
      CMPB #1,CPUTYP  ;; ** 11/44?? ;DPM001
      BNE 12$         ;; ** NO USE M9312 ROM NUMBERS
    
```


489	001726	012737	013106	013104		MOV	#MESS44,MESSPT	;	** YES USE UBI ROM NUMBERS
490	001734	012737	007306	007176		MOV	#AROM,ROMPTR	;	** UBI ROM NUMBERS
491	001742	000403				BR	12\$;	** BR IF NO TRAP
492	001744	012716	001776		10\$:	MOV	#15\$, (SP)	;	SET UP RETURN VECTOR
493	001750	000002				RTI		;	RETURN
494	001752	122737	000003	001122	12\$:	CMPB	#3,CPUTYP	;	** 11/24??
495	001760	001006				BNE	15\$;	** 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\$:	MOV	(SP)+,10	;	RESTORE ERROR VECTOR
499	002002	012700	010502			MOV	#BUF1,RO	;	CLR SIZING BUFFERS
500	002006	005020			8\$:	CLR	(RO)+		
501	002010	020027	010566			CMP	RO,#OCTBUF	;	HAVE WE CLEARED THE WHOLE
502	002014	002774				BLT	8\$;	BUFFER,NO,GO BACK
503	002016	005037	001116			CLR	ROMFIN	;	INITIALIZE ROM FOUND INDICATORS
504	002022	005037	004342			CLR	TIMES	;	INITIALIZE ENTRY COUNTER FOR PUTMES SUB.
505	002026	004737	011676			JSR	PC,CLEAR	;	+CLEAR SOME MORE LOCATIONS
506	002032	013746	000004			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		CMPB	#-1,165000	;	IF CONTENTS = -1, OR TRAP
509	002052	001414				BEQ	3\$;	THEN CPU ROM NOT PLUGGED IN
510	002054	012700	165000			MOV	#165000,RO	;	GET FIRST ADDRESS OF ROM SPACE
511	002060	005720			1\$:	TST	(RO)+	;	IF THIS INSTRUCTION TRAPS CPU ROM NOT PLUGGED IN
512	002062	020027	166000			CMP	RO,#166000	;	HAVE WE CHECKED ALL ADDRESSES
513	002066	001374				BNE	1\$;	IF NO THEN CONTINUE LOOPING
514	002070	012737	000001	001116		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 ERRUEL
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			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	#200,@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			BIS	RO,CONFIN	;	+SET CONTINUATION ROM FOUND INDICATOR
526	002146	050037	001116			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			BIS	RO,DEVFIN	;	+SET DEVICE ROM FOUND INDICATOR
530	002164	062737	000200	003644	4\$:	ADD	#200,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	004737	002544			JSR	PC,NOROMS	;	IF NO GO TEST NO ROMS
537	002216	000411				BR	\$EOP	;	GO TO END OF PASS
538	002220	004737	002632		6\$:	JSR	PC,CHECKS	;	GO CALCULATE CHECKSUMS
539	002224	005737	001146			TST	\$PASS	;	ARE WE ON 1ST PASS
540	002230	001004				BNE	\$EOP	;	IF NO SKIP PROCESS OF PARAMETERS
541	002232	004737	003146			JSR	PC,PROMP	;	GO PROCESS ROM PARAMETERS
542	002236	004737	012174			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      NOP
002242 000240      INC      $PASS      ;; INCREMENT THE PASS NUMBER
002244 005237 001146      BIC      #100000,$PASS  ;; DON'T ALLOW A NEG. NUMBER
002250 042737 100000 001146      DEC      (PC)+      ;; LOOP?
002256 005327      $EOPCT: .WORD 1
002260 000001      BGT      $DOAGN      ;; YES
002262 003017      MOV      (PC)+,@(PC)+  ;; RESTORE COUNTER
002264 012737      $ENDCT: .WORD 1
002266 000001      $EOPCT
002270 002260      TYPE      ,SENDMG      ;; TYPE 'END PASS'
002272 104401 002331      TYPE      ,SENULL      ;; TYPE A NULL CHARACTER
002276 104401 002326      $GET42: MOV      @#42,R0      ;; GET MONITOR ADDRESS
002302 013700 000042      BEQ      $DOAGN      ;; BRANCH IF NO MONITOR
002306 001405      RESET      ;; CLEAR THE WORLD
002310 000005      $ENDAD: JSR      PC,(R0)  ;; GO TO MONITOR
002312 004710      NOP      ;; SAVE ROOM
002314 000240      NOP      ;; FOR
002316 000240      NOP      ;; ACT11
002320 000240      $DOAGN:
002322      JMP      @(PC)+      ;; RETURN
002322 000137      $RTNAD: .WORD  RSTART
002324 002204      $ENULL: .BYTE  -1,-1,0
002326      377      377      000      $ENDMG: .ASCII <15><12>/END PASS/
002331      015      012      105
```



```

550 .SBTTL NO ROMS TEST
551 ;THIS ROUTINE IS CALLED ONLY IF NO ROMS WERE FOUND
552 ;DURING SIZING. IT DOES A READ OF ALL BOOTSTRAP ROM
553 ;ADDRESSES AND COMPARES THE CONTENTS TO A KNOWN
554 ;EXPECTED VALUE.
555
556 002544 104401 007350 NOROMS: TYPE ,MNORM ; - TYPE MESSAGE THAT NOROMS FOUND
557 002550 012703 172000 MOV #172000,R3 ; GET FIRST ADDRESS TO BE READ
558 002554 013704 002630 1$: MOV NODATA,R4 ; GET ADDRESS OF EXPECTED VALUE
559 002560 121304 CMPB (R3),R4 ; +DOES RECIEVED VALUE FQUAL EXPECTED
560 002562 001406 BEQ 2$ ; IF YES CONTINUE TESTING
561 002564 011304 MOV (R3),R4 ; +GET BAD VALUE
562 002566 052737 000100 001110 BIS #NOROME,ROMERR ; IF NO SET ERROR INDICATER
563 002574 004737 004344 JSR PC,ERRHAN ; REPORT ERROR
564 002600 062703 000002 2$: ADD #2,R3 ; +GET TO NEXT EVEN ADDRESS
565 002604 022703 173024 CMP #173024,R3 ; +AT A VECTOR ADDRESS?
566 002610 001773 BEQ 2$ ; +BRANCH IF YES
567 002612 022703 173224 CMP #173224,R3 ; +AT A VECTOR ADDRESS?
568 002616 001770 BEQ 2$ ; +BRANCH IF YES
569 002620 022703 174000 CMP #174000,R3 ; HAVE ALL ADDRESSES BEEN TESTED
570 002624 003353 BGT 1$ ; IF NO GO TEST THIS ADDRESS
571 002626 000207 RTS PC
572
573 002630 000377 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 #165775,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 ERROR SET INDICATER
588 002714 004737 004344 JSR PC,ERRHAN ;REPORT ERROR
589 002720 012737 173000 003140 1$: MOV #173000,FIRSTA ;SET UP FIRST ADDRESS TO BE SUMMED
590 002726 012737 173024 003142 MOV #173024,EXCADD ;SET UP EXCEPTION ADDRESS
591 002734 012737 173175 003144 MOV #173175,LASTAD ;SET UP LAST ADDRESS TO BE SUMMED
592 002742 012702 173176 MOV #173176,R2 ;GET ADDRESS OF GOOD DATA
593 002746 006137 003046 2$: 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 INDICATER
601 003002 004737 004344 JSR PC,ERRHAN ;REPORT ERROR
602 003006 062737 000200 003140 3$: ADD #200,FIRSTA ;UPDATE FIRST ADDRESS
603 003014 062737 000200 003142 ADD #200,EXCADD ;UPDATE EXCEPTION ADDRESS
604 003022 062737 000200 003144 ADD #200,LASTAD ;UPDATE LAST ADDRESS
605 003030 062702 000200 ADD #200,R2 ;UPDATE ADDRESS OF GOOD DATA
606 003034 022737 174000 003140 CMP #174000,FIRSTA ;HAVE ALL ROMS BEEN CHECKED
607 003042 001341 BNE 2$ ;IF NO GO CHECKSUM NEXT ONE
608 003044 000207 RTS PC ;IF YES RETURN
609
610 003046 000000 ROMCNT: 0
    
```

```

611                                     .SBTTL  CALCULATE CHECKSUMS MODULE
612                                     ;THIS ROUTINE CALCULATES THE CRC16 CHECKSUM OF THE CONTENTS
613                                     ;OF EVERY LOCATION EXCEPT THE EXCEPTION ADDRESS AND LAST ADDRESS
614                                     ;OF EVERY ROM
615
616 003050 010246                       CALSUM: MOV     R2,-(SP)           ;SAVE R2
617 003052 013700 003140                MOV     FIRSTA,R0       ;GET STARTING ADDRESS
618 003056 005004                        CLR     R4              ;INITIALIZE CRC WORD
619 003060 012005                       LOOP:  MOV     (R0)+,R5   ;GET A BYTE
620 003062 012702 000020                MOV     #16.,R2       ;SET BYTE COUNT
621 003066 000241                       CRCLOP: CLC           ;THE NEXT NINE LINES
622 003070 006004                        ROR     R4              ;DO THE MATH CALCULATIONS
623 003072 006005                        ROR     R5
624 003074 102006                        BVC    1$
625 003076 012701 120001                MOV     #120001,R1
626 003102 040401                        BIC    R4,R1
627 003104 042704 120001                BIC    #120001,R4
628 003110 050104                        BIS    R1,R4
629 003112 005302                       1$:   DEC     R2
630 003114 003364                       BGT    CRCLOP
631 003116 020037 003142                CMP     RO,EXCADD     ;IS NEXT ADDRESS AN EXCEPTION ADDRESS
632 003122 001001                       BNE    2$
633 003124 005720                       TST    (R0)+
634 003126 020037 003144                2$:   CMP     RO,LASTAD  ;IF NO TEST IT
635 003132 101752                       BLOS   LOOP          ;IF YES SKIP ADDRESS
636 003134 012602                       MOV     (SP)+,R2     ;HAVE ALL LOCATIONS BEEN SUMMED
637 003136 000207                       RTS     PC           ;IF NO CONTINUE
638                                     ;RESTORE R2
639 003140 000000                       FIRSTA: 0           ;IF YES RETURN
640 003142 000000                       EXCADD: 0
641 003144 000000                       LASTAD: 0
    
```

```

642          .SBTTL  PROCESS ROM PARAMETERS MODULE
643          ;THIS ROUTINE DETERMINES IF SIZING IS TO BE DONE.  IF IT IS THE
644          ;MODULE WILL GET THE PARAMETERS FROM THE DEVCOD AND PPFVAR
645          ;MODULES AND ASSEMBLE MESSAGES TO BE OUTPUT.  IF SIZING IS
646          ;NOT TO BE DONE THIS MODULE WILL TAKE THE PARAMETERS RECEIVED
647          ;AND COMPARE THEM TO THE VALUES SUPPLIED IN THE ETABLE.
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 003426      JSR    PC,DEVCOD  ;CALL DEVICE CODE MODULE
652 003160 004737 003760      JSR    PC,PUTMES  ;FORMAT PARAMETER MESSAGE
653 003164 010576           MES1
654 003166 004737 003650      JSR    PC,PPFVAR  ;GO GET PSEUDO POWER-FAIL VECTOR ADDRESS
655 003172 032737 000040 001110 BIT    #PFERR,ROMERR ;WAS THERE AN ERR
656 003200 001403           BEQ    10$       ;IF NO GO FORMAT MESSAGE
657 003202 004737 004344      JSR    PC,ERRHAN  ;IF YES GO REPORT MESSAGE
658 003206 000506           BR     9$
659 003210 004737 003760      10$:  JSR    PC,PUTMES  ;FORMAT PSEUDO POWER-FAIL VECTOR
660 003214 011176           MES2      ;ADDRESS MESSAGE
661 003216 005237 001112      INC    MESSAG    ;SET MESSAGE INDICATOR
662 003222 000500           BR     9$        ;GO TO EXIT
663 003224 004737 003426      1$:  JSR    PC,DEVCOD  ;CALL GET DEVICE CODE MODULE
664 003230 012703 010504      MOV    #BUF1+2,R3 ;GET BOARD DEVICE CODES           ;DPM001
665 003234 012704 001224      2$:  MOV    #SDDW0,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 004344      JSR    PC,ERRHAN  ;DOES NOT EXIST IN ETABLE
673 003264 062703 000004      4$:  ADD    #4,R3     ;UPDATE TO NEXT ADDRESS
674 003270 005713           TST    (R3)      ;IF CONTENTS EQUALS ZERO WE'RE DONE
675 003272 001362           BNE   3$        ;IF CONTENTS NO EQUAL ZERO CONTINUE
676 003274 012703 001224      MOV    #SDDW0,R3 ;GET BOARD DEVICE CODES
677 003300 012704 010500      MOV    #BUF1-2,R4 ;GET ETABLE DEVICE CODES
678 003304 062704 000004      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           BNE   6$        ;DEVICE CODES, IF NO CONTINUE
683 003320 052737 000002 001110 BIS    #APTER2,ROMERR ;IF YES ETABLE DEVICE CODE NOT
684 003326 004737 004344      JSR    PC,ERRHAN  ;NOT ON BOARD
685
686 003332 062703 000002      7$:  ADD    #2,R3     ;INCREMENT TO NEXT SDDW LOCATION
687 003336 022703 001254      CMP    #SDDW12,R3 ;HAVE WE CHECKED ALL 12 POSITIONS ;DPM001
688 003342 001360           BNE   6$        ;IF NOT, CONTINUE
689 003344 004737 003650      JSR    PC,PPFVAR  ;GO GET PSEUDO POWER-FAIL VECTOR ADDRESS
690 003350 013704 001214      MOV    $BASE,R4  ;GET ETABLE PPFVA
691 003354 013703 010562      MOV    BUF2,R3   ;GET BOARD PPFVA
692 003360 020403           CMP    R4,R3     ;DO THE TWO PARAMETERS COMPARE
693 003362 001405           BEQ    8$        ;IF YES CONTINUE
694 003364 052737 000004 001110 BIS    #APTER3,ROMERR ;SET APT ERROR INDICATOR
695 003372 004737 004344      JSR    PC,ERRHAN  ;GO TO ERROR ROUTINE
696 003376 013704 001220      8$:  MOV    $CDW1,R4  ;GET ETABLE DATA
697 003402 013703 010564      MOV    BUF2+2,R3 ;GET BOARD DATA
698 003406 020403           CMP    R4,R3     ;DO THE TWO PARAMETER COMPARE

```

PROCESS ROM PARAMETERS MODULE

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

9\$:


```

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 003426 012700 010502
731 003432 012701 000001
732 003436 030137 001116
733 003442 001411
734 003444 012737 165774 003644
735 003452 013720 003644
736 003456 017720 000162
737 003462 005037 003646
738 003466 013737 001114 003644 1$:
739 003474 006101
740 003476 030137 013072 3$:
741 003502 001007
742 003504 004737 011714
743 003510 062737 000200 003644
744 003516 006101
745 003520 000444
746 003522 005737 003646 4$:
747 003526 001010
748 003530 013720 003644
749 003534 017720 000104
750 003540 062737 000002 003644
751 003546 000427
752 003550 013702 003644 5$:
753 003554 067737 000064 003644
754 003562 013703 003644
755 003566 042737 000001 003644
756 003574 042702 170177
757 003600 042703 170177
758 003604 160203
759 003606 005703

```

```

.SBTTL 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 -1 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    1$           :IF NO GO TEST BOOT ROMS
        MOV    #165774,TESTAD :GET ADDRESS OF DIAG. 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    4$          :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    5$          :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	001406				BEQ	8\$:IF YES THEN DONE
761	003612	162703	000200			SUB	#200,R3		:IF NO THEN MOVE POINTER
762	003616	006101				ROL	R1		:ONE BIT FOR EVERY 200
763	003620	004737	011742			JSR	PC,CON2		:+CHECK FOR CONTINUATION CHIP
764	003624	000770				BR	6\$		
765	003626	005137	003646		8\$:	COM	DAFLAG		:CHANGE DATA FLAG TO RIGHT DATA TYPE
766	003632	023727	003644	174000	9\$:	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		

```

772                                     .SBTTL GET PSEUDO POWER-FAIL VECTOR ADDRESS ROUTINE
773                                     :THIS SUBROUTINE TESTS LOCATIONS 173024 AND 173224 TO DETERMINE
774                                     :WHICH VECTOR WILL BE USED IF POWER-FAIL OPTION ENABLED ON THE
775                                     :BOARD. THE OPTION MUST BE ENABLED AND AT LEAST ONE ADDRESS
776                                     :SWITCH MUST BE 'ON' OR AN ERROR WILL BE DETECTED.
777                                     :THE DATA WILL BE RETURNED IN 'BUF2' IN THE FORMAT.
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$,RETURN :+SET UP AN ALTERNATE RETURN
785 003666 004737 012006 JSR PC,HOLCK1 :+CHECK FOR A HOLE
786 003672 012737 173024 010562 MOV #173024,BUF2 :IF IT IS THEN STORE ADDRESS
787 003700 013737 173024 010564 MOV 173024,BUF2+2 :STORE CONTENTS OF LOCATION 173024
788 003706 000423 BR 3$ :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$,RETURN :+SET UP AN ALTERNATE RETURN
792 003726 004737 012020 JSR PC,HOLCK2 :+CHECK FOR A HOLE
793 003732 012737 173224 010562 MOV #173224,BUF2 :IF IT IS THEN STORE ADDRESS
794 003740 013737 173224 010564 MOV 173224,BUF2+2 :STORE CONTENTS OF VECTOR
795 003746 000403 BR 3$ :GET OVER ERROR
796 003750 052737 000040 001110 2$: BIS #PFERR,ROMERR :SET ERROR INDICATOR
797 003756 000207 3$: RTS PC

```

```

798          .SBTTL PUT MESSAGE IN BUFFER ROUTINE
799          :THIS SUBROUTINE FORMATS THE PARAMETER AND POWER-FAIL MESSAGES.
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          BNE 1$          :IF NOT DON'T FORMAT DIAG. ROM MESSAGE
808 004010 105737 001122          TSTB CPUTYP          :SEE IF A NON-11/24/44 CPU          :DPM001
809 004014 001412          BEQ 100$          :BRANCH OVER 11/24/44 SECTION IF SO          :DPM001
810 004016 122737 000001 001122          CMPB #1,CPUTYP          :SEE IF AN 11/44          :DPM001
811 004024 001003          BNE 98$          :BRANCH IF NOT TO SETUP FOR 11/24          :DPM001
812 004026 012705 007465          MOV #DRH44,R5          :GET DIAG. ROM MESSAGE HEADER          :DPM001
813 004032 000405          BR 101$          :BRANCH TO PUT HEADER IN MESSAGE BUFFER          :DPM001
814 004034 012705 007524          98$: MOV #DRH24,R5          :GET DIAG. ROM MESSAGE HEADER          :DPM001
815 004040 000402          BR 101$          :BRANCH TO PUT HEADER IN MESSAGE BUFFER          :DPM001
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 004060 000313          SWAB (R3)          :FORMAT ASCII FOR MESSAGE
821 004062 112324          MOVB (R3)+,(R4)+          :PUT ASCII IN MESSAGE BUFFER
822 004064 112324          MOVB (R3)+,(R4)+
823 004066 012705 007600          1$: MOV #BRHEAD,R5
824 004072 004737 004772          JSR PC,FILBUF          :GO PUT BOOT ROM HEADER IN MESS. BUF.
825 004076 000015          CR
826 004100 012701 000001          MOV #1,R1          :+POINT TO DIAGNOSTIC ROM
827 004104 013702 013104          MOV MESSPT,R2          :+POINT TO MSG TABLE
828 004110 012737 004124 013064          MOV #2$,RETURN          :+SET UP AN ALTERNATE RETURN
829 004116 012737 004310 013102          MOV #5$,FINISH          :+SET UP ANOTHER ALTERNATE RETURN
830 004124 112724 000015          2$: MOVB #CR,(R4)+          :PUT A CR/LF HERE
831 004130 112724 000012          MOVB #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 005140          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          MOVB #40,(R4)+
838 004162 112724 000040          MOVB #40,(R4)+
839 004166 112724 000040          MOVB #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          HT
844 004210 112724 000040          MOVB #40,(R4)+
845 004214 112724 000040          MOVB #40,(R4)+
846 004220 112724 000040          MOVB #40,(R4)+
847 004224 112724 000011          MOVB #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          MOVB (R3)+,(R4)+          :MOV ASCII TO MES1
851 004240 112324          MOVB (R3)+,(R4)+
852 004242 000730          BR 2$          :GO BACK TO START OF LOOP
853 004244 012703 010562          3$: MOV #BUF2,R3          :GET DATA BUFFER ADDRESS
854 004250 012705 007735          MOV #PFHEAD,R5          :GET POWER-FAIL HEADER ADDRESS

```

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	5\$: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			BR	4\$:GO BACK TO START OF LOOP
864	004310	112724	000015	5\$:	MOVB	#CR,(R4)+		:PUT A CR/LF AT END OF MESSAGE
865	004314	112724	000012		MOVB	#LF,(R4)+		
866	004320	105024			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		ADD	#2,(R6)		:GET OVER MESSAGE BUFFER ADDRESS
871	004340	000207			RTS	PC		
872	004342	000000			TIMES:	0		

ERROR HANDLER ROUTINE

```

873          .SBTTL ERROR HANDLER ROUTINE
874          ;THIS SUBROUTINE FORMATS THE ERROR MESSAGES THE TYPES THEM
875          ;OUT.
876 004344 104411          ERRHAN: SAVREG
877 004346 012704 011276      MOV      #ERRMSG,R4
878 004352 005237 001120      INC      ERRCNT          ;INCREMENT ERROR COUNTER
879 004356 032777 002000 174520 BIT      #BIT10,@SWR      ;BELL ON ERROR
880 004364 001402          BEQ      1$              ;BRANCH IF NO
881 004366 104401          TYPE
882 004370 007344          BELL
883 004372 032777 020000 174504 1$: BIT      #BIT13,@SWR      ;INHIBIT ERROR TYPEOUT
884 004400 001402          BEQ      OVER1          ; NO GO THROUGH
885 004402 000137 004722      JMP      OVER10
886 004406 013700 001110      OVER1: MOV      ROMERR,R0      ;GET ERROR CODE
887 004412 013737 001110 004710 MOV      ROMERR,OVER9     ;SAVE ERROR CODE FOR APT
888 004420 012701 004744      MOV      #EHEADT,R1     ;GET ERROR HEADER TABLE
889 004424 000241          CLC              ;C-BIT USED TO STOP STEPPING THROUGH TABLE
890 004426 006000          2$: ROR      R0              ;ROTATE ERROR CODES
891 004430 103403          BCS      3$              ;IF C-BIT SET STOP STEPPING
892 004432 062701 000002      ADD      #2,R1          ;IF C-BIT CLEAR STEP TO NEXT HEADER
893 004436 000773          BR       2$              ;GO STEP
894 004440 011105          3$: MOV      (R1),R5      ;PUT HEADER ADDRESS IN REGISTER.
895 004442 004737 004772      JSR      PC,FILBUF      ;GO PUT HEADER IN ERROR MESSAGE BUF.
896 004446 000015          CR
897 004450 032737 000040 001110 BIT      #PFERR,ROMERR    ;+IS IT A 'PPFVAR' ERROR?
898 004456 001065          BNE      8$              ;+BRANCH IF YES
899 004460 032737 003600 001110 BIT      #3600,ROMERR     ;+IS IT A 'SEQTST' ERROR?
900 004466 001411          BEQ      14$             ;+BRANCH IF NO
901 004470 013737 013076 004506 MOV      ARG2,13$        ;+PUT #CR OR #HT AT 13$
902 004476 013705 013074      MOV      ARG1,R5        ;+ARG1 CONTAINS ADR. OF MSG
903 004502 004737 004772      JSR      PC,FILBUF      ;+PUT DATA INTO BUFFER
904 004506 000000          13$: .WORD      0          ;+CONTAINS CR OR HT
905 004510 000450          BR       8$              ;+
906 004512 032737 000023 001110 14$: BIT      #23,ROMERR     ;+IS ERROR A CRC, APTER1 OR APTER2
907 004520 001426          BEQ      7$              ;IF NO GO FORMAT APTER3, APTER4
908 004522 032737 000020 001110 BIT      #20,ROMERR     ;IS ERROR CRC
909 004530 001417          BEQ      6$              ;IF NO FORMAT APTER1, APTER2
910 004532 013700 003046      MOV      ROMCNT,R0      ;GET ROM COUNTER
911 004536 013701 007176      MOV      ROMPTR,R1      ;GET ROM NUMBER TABLE
912 004542 000241          CLC              ;C-BIT USED TO STOP STEPPING THROUGH TABLE
913 004544 006000          4$: ROR      R0              ;ROTATE ROM NUMBER
914 004546 103403          BCS      5$              ;IF C-BIT SET STOP STEPPING
915 004550 062701 000002      ADD      #2,R1          ;IF C-BIT CLEAR STEP TO NEXT ADRS
916 004554 000773          BR       4$              ;CONTINUE STEPPING
917 004556 011101          5$: MOV      (R1),R1      ;MOVE ADDRESS TO R1 FROM TABLE
918 004560 112124          55$: MOVB     (R1)+,(R4)+    ;PUT ROM NUMBER IN ERROR MESSAGE BUF ;DPM001
919 004562 105711          TSTB     (R1)          ;SEE IF TERMINATOR ;DPM001
920 004564 001375          BNE      55$          ;BRANCH BACK IF NOT ;DPM001
921 004566 000421          BR       8$              ;GO TO END OF ROUTINE
922 004570 112324          6$: MOVB     (R3)+,(R4)+    ;PUT BAD DEVICE CODE IN ERROR MESS. BUF.
923 004572 112324          MOVB     (R3)+,(R4)+
924 004574 000416          BR       8$              ;GO TO END OF ROUTINE
925 004576 016603 000006      7$: MOV      6(R6),R3      ;GET SAVED EXPECTED VALUE
926 004602 004737 005034      JSR      PC,OCASC       ;GO COVERT OCTAL TO ASCII
927 004606 004737 004772      JSR      PC,FILBUF      ;GO PUT DATA IN ERROR MESSAGE BUF.
928 004612 000011          HT
929 004614 016603 000010      MOV      10(R6),R3     ;GET SAVED RECEIVED VALUE

```

```

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    $ENV         ;TST IF ON APT
937 004652 001002              BNE    OVER2        ;IF YES BRANCH
938 004654 000137 004714      JMP    OVER11       ;IF NO BRANCH
939 004660 022737 000001 001116 OVER2:  CMP    #1,ROMFIN    ;- ONLY A CPU ROM?????
940 004666 001005              BNE    AROUND       ;-IF NO BRANCH
941 004670 032737 000040 001110      BIT    #PFERR,ROMERR ;- NOW IF YES ,IS POWER-
942                                ;- FAIL VECTOR THE ERROR????
943 004676 001401              SEQ    AROUND       ;- YES ,WELL THEN IT'S
944 004700 000405              BR     OVER11       ;- NOT A REAL ERROR , CONTINUE
945 004702 004737 005540      AROUND: JSR    PC,$ATY1 ;GO REPORT ERROR
946 004706 011276              ERRMSG
947 004710 000000      OVER9: .WORD    0
948 004712 000000              HALT
949 004714 004737 005546      OVER11: JSR    PC,$ATY3 ;THEN HALT
950 004720 011276              ERRMSG             ;IF NOT ON APT JUST REPORT ERROR
951 004722 032777 100000 174154 OVER10: BIT    #BIT15,@SWR ;IS HALT ON ERROR SET
952 004730 001401              BEQ    12$         ;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      NOROMM
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 000000 FILBUF: CMP    #HT,@(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 0C5006 000404          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          ADD    #2,(R6)    :UPDATE RETURN POINTER TO GET OVER CHARACTER
984 005032 000207          RTS    PC        :THEN RETURN
```

```

985          .SBTTL OCTAL TO ASCII CONVERSION ROUTINE
986          ;THIS SUBROUTINE TAKES A SIXTEEN BIT OCTAL NUMBER AND
987          ;CONVERTS IT TO 6 ASCII CHARACTERS
988
989 005034 012700 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              CLH   (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 152710 000060      1$:  BISB   #60,(R0)    ;MAKE IT ASCII
1000 005074 005200              INC    R0            ;UPDATE BUFFER ADDRESS
1001 005076 020027 010574      CMP    R0,#OCTBUF+6  ;HAVE WE CONVERTED ALL THE NUMBER
1002 005102 001003              BNE   2$            ;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
1014
1015
1016
1017
1018
1019
1020
1021

005140 010346
005142 011303
005144 004737 005034
005150 012603
005152 000207

```
.SBTTL ROUTINE TO SETUP R3 FOR THE OCTAL TO ASCII ROUTINE  
:THIS ROUTINE IS CALLED BY THE PUT MESSAGE ROUTINE TO GET THE RIGHT  
:VALUE IN R3 SO THE OCTAL TO ASCII ROUTINE GETS THE RIGHT NUMBER.  
OCADD: MOV R3, -(R6) :SAVE THE VALUE OF R3  
MOV (R3), R3 :PUT THE DATA TO BE CONVERTED IN R3  
JSR PC, OCASC :GO CONVERT OCTAL TO ASCII  
MOV (R6)+, R3 :RESTORE R3  
RTS PC :RETURN
```


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
:*OR
:* TYPE
:* MESADR
:*
$TYPE: TSTB $TPFLG ;;IS THERE A TERMINAL?
        BPL 1$ ;;BR IF YES
        HALT ;;HALT HERE IF NO TERMINAL
        BR 3$ ;;LEAVE
1$: MOV RO,-(SP) ;;SAVE RO
    MOV @2(SP),RO ;;GET ADDRESS OF ASCIZ STRING
    CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
    BNE 62$ ;;NO,GO CHECK FOR APT CONSOLE
    BITB #APTPOOL,$ENVM ;;SPOOL MESSAGE TO APT
    BEQ 62$ ;;NO,GO CHECK FOR CONSOLE
    MOV RO,61$ ;;SETUP MESSAGE ADDRESS FOR APT
    JSR PC,$ATY3 ;;SPOOL MESSAGE TO APT
61$: .WORD 0 ;;MESSAGE ADDRESS
62$: BITB #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
    BNE 60$ ;;YES,SKIP TYPE OUT
    MOVB (RO)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
    BNE 4$ ;;BR IF IT ISN'T THE TERMINATOR
    TST (SP)+ ;;IF TERMINATOR POP IT OFF THE STACK
60$: MOV (SP)+,RO ;;RESTORE RO
3$: ADD #2,(SP) ;;ADJUST RETURN PC
    RTI ;;RETURN
4$: CMPB #HT,(SP) ;;BRANCH IF <HT>
    BEQ 8$
    CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
    BNE 5$
    TST (SP)+ ;;POP <CR><LF> EQUIV
    TYPE ;;TYPE A CR AND LF
5$: CLRB $CHARCNT ;;CLEAR CHARACTER COUNT
    BR 2$ ;;GET NEXT CHARACTER
5$: JSR PC,$TYPEC ;;GO TYPE THIS CHARACTER
6$: CMPB $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
    BNE 2$ ;;IF NO GO GET NEXT CHAR.
    MOV $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
    ;;AND THE NULL CHAR.
7$: DECB 1(SP) ;;DOES A NULL NEED TO BE TYPED?
    BLT 6$ ;;BR IF NO--GO POP THE NULL OFF OF STACK
    JSR PC,$TYPEC ;;GO TYPE A NULL
    DECB $CHARCNT ;;DO NOT COUNT AS A COUNT
    BR 7$ ;;LOOP
;HORIZONTAL TAB PROCESSOR
8$: MOVB #' ,(SP) ;;REPLACE TAB WITH SPACE

```

```

005154 105737 005533
005160 100002
005162 000000
005164 000430
005166 010046
005170 017600 000002
005174 122737 000001 001160
005202 001011
005204 132737 000100 001161
005212 001405
005214 010037 005224
005220 004737 005546
005224 000000
005226 132737 000040 001161
005234 001003
005236 112046
005240 001005
005242 005726
005244 012600
005246 062716 000002
005252 000002
005254 122716 000011
005260 001430
005262 122716 000200
005266 001006
005270 005726
005272 104401
005274 005535
005276 105037 005514
005302 000755
005304 004737 005366
005310 123726 005532
005314 001350
005316 013746 005530
005322 105366 000001
005326 002770
005330 004737 005366
005334 105337 005514
005340 000770
005342 112716 000040

```

TYPE ROUTINE

```

005346 004737 005366          9$: JSR PC,$TYPEC      ;;TYPE A SPACE
005352 132737 000007 005514 BITB #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 #$XOFF,(SP) ;;WAS CHAR XOFF           ;MJD001
005410 001012          BNE 102$      ;;BR IF NOT                ;MJD001
005412          101$:
005412 105777 000102          TSTB @STKS      ;;WAIT FOR CHAR             ;MJD001
005416 100375          BPL 101$      ;;MJD001
005420 117716 000076          MOVB @STKB,(SP) ;;GET CHAR                  ;MJD001
005424 042716 177600          BIC #177600,(SP) ;;STRIP IT                 ;MJD001
005430 122716 000021          CMPB #$XON,(SP) ;;WAS IT XON?             ;MJD001
005434 001366          BNE 101$      ;;BR IF NOT                ;MJD001
005436          102$:
005436 005726          TST (SP)+       ;;FIX STACK                 ;MJD001
005440          10$:
005440 105777 000060          TSTB @STPS      ;;WAIT UNTIL PRINTER IS READY ;MJD001
005444 100375          BPL 10$
005446 126627 000002 000021  CMPB 2(SP),#$XON ;;IS CHARACTER A RANDOM XON? ;MJD001
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 000406          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 105227          INCB (PC)+    ;;COUNT THE CHARACTER
005514 000000          $CHARCNT: .WORD 0 ;;CHARACTER COUNT STORAGE
005516 000207          $TYPEX: RTS PC
005520 177560          $TKS: .WORD 177560 ;;TTY KDB STATUS           ;MJD001
005522 177562          $TKB: .WORD 177562 ;;TTY KDB 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          $FILLC: .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 012 000          $LF: .ASCII <12> ;;LINEFEED

```

1024

.SBTTL APT COMMUNICATIONS ROUTINE

```

*****
005540 112737 000001 006004 $ATY1: MOVB #1,$FFLG ;;TO REPORT FATAL ERROR
005546 112737 000001 006002 $ATY3: MOVB #1,$MFLG ;;TO TYPE A MESSAGE
005554 000403
005556 112737 000001 006004 $ATY4: MOVB #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
005564 $ATYC:
005564 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
005566 010146 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 CMPB #APTENV,$ENV ;;OPERATING UNDER APT?
005604 001031 BNE 3$ ;;IF NOT: BR
005606 132737 000100 001161 BITB #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
005614 001425 BEQ 3$ ;;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$
005646 163700 001154 SUB $MSGAD,R0 ;;SUB START OF MESSAGE
005652 006200 ASR R0 ;;GET MESSAGE LNTH IN WORDS
005654 010037 001156 MOV R0,$MSGGLT ;;PUT LENGTH IN MAILBOX
005660 012737 000004 001140 MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
005666 000413 BR 5$
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 001160 TST $ENV ;;RUNNING UNDER APT?
005730 001413 BEQ 12$ ;;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 CLRE $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 APTPOOL=100
000040 APTCSUP=040

```

TTY INPUT ROUTINE

1026

```

.SBTTL TTY INPUT ROUTINE
*****
.ENABLE LSB
*****
*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.
$CKSWR: CMP #SWREG,SWR ;; IS THE SOFT-SWR SELECTED?
        BNE 15$ ;; BRANCH IF NO
        TSTB @STKS ;; CHAR THERE?
        BPL 15$ ;; IF NO, DON'T WAIT AROUND
        MOVB @STKB,-(SP) ;; SAVE THE CHAR
        BIC #^C177,(SP) ;; STRIP-OFF THE ASCII
        CMP #7,(SP)+ ;; IS IT A CONTROL G?
        BNE 15$ ;; NO, RETURN TO USER
        CMPB $AUTOB,#1 ;; ARE WE RUNNING IN AUTO-MODE?
        BEQ 15$ ;; BRANCH IF YES
        TYPE ,SCNTLG ;; ECHO THE CONTROL-G (^G)
$GTSWR: TYPE ,SMSWR ;; TYPE CURRENT CONTENTS
        MOV SWREG,-(SP) ;; SAVE SWREG FOR TYPEOUT
        TYPOC ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
        TYPE ,SMNEW ;; PROMPT FOR NEW SWR
19$: CLR -(SP) ;; CLEAR COUNTER
        CLR -(SP) ;; THE NEW SWR
7$: TSTB @STKS ;; CHAR THERE?
        BPL 7$ ;; IF NOT TRY AGAIN
        MOVB @STKB,-(SP) ;; PICK UP CHAR
        BIC #^C177,(SP) ;; MAKE IT 7-BIT ASCII
9$: CMP (SP),#25 ;; IS IT A CONTROL-U?
        BNE 10$ ;; BRANCH IF NOT
        TYPE ,SCNTLU ;; YES, ECHO CONTROL-U (^U)
20$: ADD #6,SP ;; IGNORE PREVIOUS INPUT
        BR 19$ ;; LET'S TRY IT AGAIN
10$: CMP (SP),#15 ;; IS IT A <CR>?
        BNE 16$ ;; BRANCH IF NO
        TTT 4(SP) ;; YES, IS IT THE FIRST CHAR?
        BEQ 11$ ;; BRANCH IF YES
        MOV 2(SP),@SWR ;; SAVE NEW SWR
11$: ADD #6,SP ;; CLEAR UP STACK
14$: TYPE ,SCRLF ;; ECHO <CR> AND <LF>
        CMPB $INTAG,#1 ;; RE-ENABLE TTY KBD INTERRUPTS?
        BNE 15$ ;; BRANCH IF NOT
        MOV #100,@STKS ;; RE-ENABLE TTY KBD INTERRUPTS
15$: RTI ;; RETURN
16$: JSR PC,$TYPEC ;; ECHO CHAR
        CMP (SP),#60 ;; CHAR < 0?
        BLT 18$ ;; BRANCH IF YES
        CMP (SP),#67 ;; CHAR > 7?
        BGT 18$ ;; BRANCH IF YES
        BIC #60,(SP)+ ;; STRIP-OFF ASCII
        TST 2(SP) ;; IS THIS THE FIRST CHAR
        BEQ 17$ ;; BRANCH IF YES
        ASL (SP) ;; NO, SHIFT PRESENT
        ASL (SP) ;; CHAR OVER TO MAKE
        ASL (SP) ;; ROOM FOR NEW ONE.
17$: INC 2(SP) ;; KEEP COUNT OF CHAR
    
```

006006 022737 000176 001104
 006014 001074
 006016 105777 177476
 006022 100071
 006024 117746 177472
 006030 042716 177600
 006034 022726 000007
 006040 001062
 006042 123727 001100 000001
 006050 001456
 006052 104401 006543
 006056 104401 006550
 006062 013746 000176
 006066 104402
 006070 104401 006561
 006074 005046
 006076 005046
 006100 105777 177414
 006104 100375
 006106 117746 177410
 006112 042716 177600
 006116 021627 000025
 006122 001005
 006124 104401 006536
 006130 062706 000006
 006134 000757
 006136 021627 000015
 006142 001022
 006144 005766 000004
 006150 001403
 006152 016677 000002 172724
 006160 062706 000006
 006164 104401 005535
 006170 123727 001101 000001
 006176 001003
 006200 012777 000100 177312
 006206 000002
 006210 004737 005366
 006214 021627 000060
 006220 002420
 006222 021627 000067
 006226 003015
 006230 042726 000060
 006234 005766 000002
 006240 001403
 006242 006316
 006244 006316
 006246 006316
 006250 005266 000002

```
006254 056616 177776          BIS      -2(SP), (SP)      ;; SET IN NEW CHAR
006260 000707                   BR       7$              ;; GET THE NEXT ONE
006262 104401 005534          18$:    TYPE    $QUES      ;; TYPE ?<CR><LF>
006266 000720                   BR       20$              ;; SIMULATE CONTROL-U
.DSABL  LSB
;*****
;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     #^C<177>, 4(SP) ;; GET RID OF JUNK IF ANY
006322 026627 000004 000023    CMP     4(SP), #23      ;; 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 022627 000021          CMP     (SP)+, #21      ;; IS IT A CONTROL-Q?
006354 001366                   BNE     2$              ;; IF NOT DISCARD IT
006356 000750                   BR      1$              ;; YES, 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          4$:    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                   RDCHR  ;; 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 104401 005536          TYPE    ,SLF          ;; TYPE A LINE FEED
```

TTY INPUT ROUTINE

```

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

```


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.
:$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
:CALL:
:*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
:*      TYPOS    ;;CALL FOR TYPEOUT
:*      .BYTE   N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
:*      .BYTE   M              ;;M=1 OR 0
:*                               ;;1=TYPE LEADING ZEROS
:*                               ;;0=SUPPRESS LEADING ZEROS
:$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
:$TYPOS OR $TYPOC
:CALL:
:*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
:*      TYPON    ;;CALL FOR TYPEOUT
:$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
:CALL:
:*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
:*      TYPOC    ;;CALL FOR TYPEOUT
006572 017646 000000      $TYPOS: MOV      @ (SP),-(SP)      ;;PICKUP THE MODE
006576 116637 000001 007015  MOVB    1(SP),SOFILL      ;;LOAD ZERO FILL SWITCH
006604 112637 007017      MOVB    (SP)+,SOMODE+1    ;;NUMBER OF DIGITS TO TYPE
006610 062716 000002      ADD     #2,(SP)          ;;ADJUST RETURN ADDRESS
006614 000406      BR     $TYPON
006616 112737 000001 007015 $TYPOC: MOVB    #1,SOFILL      ;;SET THE ZERO FILL SWITCH
006624 112737 000006 007017  MOVB    #6,SOMODE+1      ;;SET FOR SIX(6) DIGITS
006632 112737 000005 007014 $TYPON: MOVB    #5,SOCNT      ;;SET THE ITERATION COUNT
006640 010346      MOV     R3,-(SP)        ;;SAVE R3
006642 010446      MOV     R4,-(SP)        ;;SAVE R4
006644 010546      MOV     R5,-(SP)        ;;SAVE R5
006646 113704 007017      MOVB    SOMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
006652 005404      NEG     R4
006654 062704 000006      ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
006660 110437 007016      MOVB    R4,SOMODE       ;;SAVE IT FOR USE
006664 113704 007015      MOVB    SOFILL,R4       ;;GET THE ZERO FILL SWITCH
006670 016605 000012      MOV     12(SP),R5      ;;PICKUP THE INPUT NUMBER
006674 005003      CLR     R3              ;;CLEAR THE OUTPUT WORD
006676 006105      1$:    ROL     R5         ;;ROTATE MSB INTO 'C'
006700 00C404      BR     3$              ;;GO DO MSB
006702 006105      2$:    ROL     R5         ;;FORM THIS DIGIT
006704 006105      ROL     R5
006706 006105      ROL     R5
006710 010503      MOV     R5,R3
006712 006103      3$:    ROL     R3         ;;GET LSB OF THIS DIGIT
006714 105337 007016      DECB    SOMODE          ;;TYPE THIS DIGIT?
006720 100016      BPL     7$              ;;BR IF NO
006722 042703 177770      BIC     #177770,R3     ;;GET RID OF JUNK
006726 001002      BNE     4$              ;;TEST FOR 0
006730 005704      TST     R4              ;;SUPPRESS THIS 0?
006732 001403      BEQ     5$              ;;BR IF YES
006734 005204      4$:    INC     R4         ;;DON'T SUPPRESS ANYMORE 0'S
006736 052703 000060      BIS     #'0,R3         ;;MAKE THIS DIGIT ASCII
006742 052703 000040      5$:    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			BLT	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 R0-R5 ROUTINES
:*****
:SAVE R0-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
$SAVREG:
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
MOV R3,-(SP) ;;PUSH R3 ON STACK
MOV R4,-(SP) ;;PUSH R4 ON STACK
MOV R5,-(SP) ;;PUSH R5 ON STACK
MOV 22(SP),-(SP) ;;SAVE PS OF MAIN FLOW
MOV 22(SP),-(SP) ;;SAVE PC OF MAIN FLOW
MOV 22(SP),-(SP) ;;SAVE PS OF CALL
MOV 22(SP),-(SP) ;;SAVE PC OF CALL
RTI
:*RESTORE R0-R5
:CALL:
:* RESREG
$RESREG:
MOV (SP)+,22(SP) ;;RESTORE PC OF CALL
MOV (SP)+,22(SP) ;;RESTORE PS OF CALL
MOV (SP)+,22(SP) ;;RESTORE PC OF MAIN FLOW
MOV (SP)+,22(SP) ;;RESTORE PS OF MAIN FLOW
MOV (SP)+,R5 ;;POP STACK INTO R5
MOV (SP)+,R4 ;;POP STACK INTO R4
MOV (SP)+,R3 ;;POP STACK INTO R3
MOV (SP)+,R2 ;;POP STACK INTO R2
MOV (SP)+,R1 ;;POP STACK INTO R1
MOV (SP)+,R0 ;;POP STACK INTO R0
RTI
    
```

```

007020
007020 010046
007022 010146
007024 010246
007026 010346
007030 010446
007032 010546
007034 016646 000022
007040 016646 000022
007044 016646 000022
007050 016646 000022
007054 000002
    
```

```

007056
007056 012666 000022
007062 012666 000022
007066 012666 000022
007072 012666 000022
007076 012605
007100 012604
007102 012603
007104 012602
007106 012601
007110 012600
007112 000002
    
```

1036

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

007136 011646
 007140 016666 000004 000002
 007146 000002

```
.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.
```

```
$TRAP:  MOV    R0,-(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   $TRPAD(R0),R0 ;;INDEX TO TABLE
        RTS   R0          ;;GO TO ROUTINE
```

```
;;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.
```

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

```
ROUTINE
-----
$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=SAVREG  TRAP+11(104411) SAVE R0-R5 ROUTINE
        $RESREG ;;CALL=RESREG  TRAP+12(104412) RESTORE R0-R5 ROUTINE
```

```
1038
1039 007176 007332 ROMPTR: .SBTTL MESSAGES
1040 007200 065 070 000 AROM11: .WORD ROMNUM
1041 007203 064 070 000 AROM21: .ASCIZ /58/ : **
1042 007206 064 071 000 AROM31: .ASCIZ /48/ : **
1043 007211 065 060 000 AROM41: .ASCIZ /49/ : **
1044 007214 065 071 000 AROM51: .ASCIZ /50/ : **
1045 007217 062 060 000 BROM11: .ASCIZ /59/ : **
1046 007222 063 065 040 BROM21: .ASCIZ /20/ : ** :DPM001
1047 007233 063 063 040 BROM31: .ASCIZ /35 OR 75/ : ** :DPM001
1048 007244 063 064 040 BROM41: .ASCIZ /33 OR 89/ : ** :DPM001
1049 007255 063 062 040 BROM51: .ASCIZ /34 OR 82/ : ** :DPM001
1050 007266 062 060 000 BROM41: .ASCIZ /32 OR 83/ : ** :DPM001
1051 007271 063 065 000 RMNUM1: .ASCIZ /20/
1052 007274 063 063 000 RMNUM2: .ASCIZ /35/
1053 007277 063 064 000 RMNUM3: .ASCIZ /33/
1054 007302 063 062 000 RMNUM4: .ASCIZ /34/
1055 RMNUM5: .ASCIZ /32/
1056 007306 007200 007203 007206 AROM: .EVEN AROM11,AROM21,AROM31,AROM41,AROM51
1057 007320 007217 007222 007233 BROM: .WORD BROM11,BROM21,BROM31,BROM41,BROM51
1058 007332 007266 007271 007274 ROMNUM: .WORD RMNUM1,RMNUM2,RMNUM3,RMNUM4,RMNUM5
1059 007344 207 377 377 BELL: .ASCIZ <207><377><377>
1060 007350 012 015 116 MNORM: .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: .ASCII	<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 PC@
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: .ASCII	/POWER-FAIL VECTOR ERROR/<15><12>
1073	010125	011	105	130	.ASCIZ	/ EXPECTED RECIEVED/<15><12>
1074	010152	120	117	127	ER4MSG: .ASCII	/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/


```

1116 011676 012700 013040 CLEAR: MOV #SEQBUF, R0 ;+SET UP TO CLEAR LOCATIONS
1117 011702 005020 1$. CLR (R0)+ ;+
1118 011704 020027 013102 CMP R0, #FINISH ;+ALL CLEARED?
1119 011710 101774 BLOS 1$ ;+BRANCH IF NO
1120 011712 000207 RTS PC ;+
1121
1122 011714 030137 013070 CON1: BIT R1, CONFIN ;+CONTINUATION CHIP?
1123 011720 001001 BNE 1$ ;+BRANCH IF YES
1124 011722 000207 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 MOV -(R0), CONER2 ;+CHIP IS MISSING-SAVE DEVICE CODE
1138 011762 062700 000002 ADD #2, R0 ;+
1139 011766 000207 RTS PC ;+
1140
1141 011770 012710 000001 2$: MOV #1, (R0) ;+SET CONTINUATION CHIP INDICATOR
1142 011774 014002 MOV -(R0), R2 ;+GET PREVIOUS DEVICE CODE
1143 011776 062700 000004 ADD #4, R0 ;+POINT TO CORRECT LOCATION
1144 012002 010220 MOV R2, (R0)+ ;+PUT IN DEVICE CODE
1145 012004 000207 RTS PC ;+
1146
1147 012006 032737 000002 001116 HOLCK1: BIT #2, ROMFIN ;+A CHIP IN SOCKET #1?
1148 012014 001415 BEQ END ;+BRANCH IF NO
1149 012016 000207 RTS PC ;+
1150
1151 012020 032737 000004 001116 HOLCK2: BIT #4, ROMFIN ;+A CHIP IN SOCKET #2?
1152 012026 001401 BEQ 1$ ;+BRANCH IF NO
1153 012030 000207 RTS PC ;+
1154
1155 012032 122737 000777 173224 1$: CMPB #777, 173224 ;+ARE SWITCHES SET?
1156 012040 001403 BEQ END ;+BRANCH IF NO
1157 012042 052737 000002 013066 BIS #2, FLAG ;+SET 11/60 INDICATOR
1158 012050 013716 013064 END: MOV RETURN, (R6) ;+SET UP ALTERNATE RETURN
1159 012054 000207 RTS PC ;+
1160
1161 012056 005713 ROM1YP: TST (R3) ;+ARE WE FINISHED?
1162 012060 001003 BNE 1$ ;+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 00G170      ROMNM:  3IT      #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$:    MOV      (R2)+, R5      ;+LOAD ROM ID
1185 012156 004737 004772      JSR     PC,   FILBUF      ;+INTO THE MSG BUFFER
1186 012162 000015              CR
1187 012164 000402              BR       4$                ;+
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 ERROR MSG AND EXIT
1200
1201 012174 012737 000011 013076 SEQTST: MOV      #HT,      ARG2      ;+SET UP FOR 'ERRHAN'
1202 012202 005737 013060          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,      ;+POINT TO MSG TABLE
1207 012226 032703 000002          1$: BIT      #2,      R3      ROMERR ;+SET ERROR FLAG
1208 012232 001026          BNE      3$          ;+IDENTIFY THE ROM FOR PRINTOUT
1209 012234 062702 000002          ADD      #2,      R2      ;+BRANCH IF FOUND IT
1210 012240 000241          CLC          ;+SET UP FOR NEXT ROM MSG
1211 012242 006003          ROR      R3          ;+
1212 012244 000770          BR      1$          ;+SET UP TO IDENTIFY NEXT ROM
1213 012246 005737 013062          2$: TST      CONER2      ;+
1214 012252 001423          BEQ      FILTAB      ;+WAS THERE AN ERROR?
1215 012254 012737 001000 001110          MOV      #CONTWO,      ;+BRANCH IF NO
1216 012262 012737 010576 013074          MOV      #MES1,      ARG1      ROMERR ;+SET ERROR FLAG
1217 012270 000337 013062          SWAB     CONER2      ;+PASS THIS ADDRESS TO ERRHAN
1218 012274 013737 013062 010576          MOV      CONER2, MES1      ;+BUILD THE MSG
1219 012302 105037 010600          CLR     MES1+2      ;+
1220 012306 000402          BR      4$          ;+
1221 012310 011237 013074          3$: MOV      (R2),      ARG1      ;+REPORT THE ERROR
1222 012314 004737 004344          4$: JSR      PC,      ERRHAN      ;+PASS MSG ADDRESS TO 'ERRHAN'
1223 012320 000207          RTS      PC          ;+REPORT ERROR
          ;+
    
```

TABLE BUILDING ROUTINE

```

1224          .SBTTL TABLE BUILDING ROUTINE
1225          ;+IF THERE WERE NO CONTINUATION ROM ERRORS, BUILD THE FOLLOWING TABLE:
1226          ;+
1227          ;+   SEQBUF:      FIRST DEVICE CODE
1228          ;+                   # OF CONTINUATION ROMS FOR FIRST DEVICE
1229          ;+                   SECOND DEVICE CODE
1230          ;+                   # OF CONTINUATION ROMS FOR SECOND DEVICE
1231          ;+
1232          ;+
1233          ;+   ENDSEQ:      FOURTH (LAST) DEVICE CODE
1234          ;+
1235          ;+FOR A 'HOLE', THE DEVICE CODE WILL BE A 0.
1236
1237 012322 012703 013040  FILTAB: MOV   #SEQBUF,      R3      ;+SET UP TO FILL A TABLE CALLED 'SEQBUF'
1238 012326 012702 173000  MOV   #173000,   R2      ;+POINT TO FIRST ROM
1239 012332 012700 000001  MOV   #1,       R0      ;+
1240 012336 000241          1$:   CLC                      ;+
1241 012340 006100          ROL   R0                      ;+POINT TO NEXT ROM
1242 012342 030037 001116  BIT   R0,       ROMFIN  ;+IS IT A HOLE?
1243 012346 001404          BEQ   3$,          ;+BRANCH IF YES
1244 012350 030037 013070  2$:   BIT   R0,       CONFIN ;+IS IT A CONTINUATION CHIP?
1245 012354 001011          BNE   4$,          ;+BRANCH IF YES
1246 012356 011213          MOV   (R2),     (R3)    ;+ITS A DEVICE ROM-PUT DEVICE CODE INTO TABLE
1247 012360 062703 000004  3$:   ADD   #4,      R3      ;+POINT TO NEXT DEVICE CODE LOCATION
1248 012364 062702 000200  5$:   ADD   #200,    R2      ;+POINT TO NEXT ROM
1249 012370 030027 000020  BIT   R0,       #20     ;+ALL DONE?
1250 012374 001760          BEQ   1$,          ;+NO-BRANCH
1251 012376 000404          BR    ALPHCK          ;+TABLE COMPLETELY BUILT
1252 012400 005243          4$:   INC   -(R3)        ;+COUNT ONE CONTINUATION CHIP IN LOCATION
1253 012402 062703 000002  ADD   #2,       R3      ;+POINT TO NEXT DEVICE CODE LOCATION
1254 012406 000766          BR    5$,          ;+

```

```

1255
1256
1257
1258
1259
1260
1261
1262 012410 012703 013040
1263 012414 010304
1264 012416 062704 000004
1265 012422 020427 013054
1266 012426 101010
1267 012430 005713
1268 012432 001446
1269 012434 005714
1270 012436 001767
1271 012440 021314
1272 012442 001410
1273 012444 003025
1274 012446 000763
1275 012450 062703 000004
1276 012454 020327 013054
1277 012460 103056
1278 012462 000754
1279 012464 012737 002000 001110
1280 012472 012737 010576 013074
1281 012500 000314
1282 012502 011437 010576
1283 012506 105037 010600
1284 012512 004737 004344
1285 012516 000207
1286 012520 010301
1287 012522 010402
1288 012524 011200
1289 012526 011122
1290 012530 010021
1291 012532 011200
1292 012534 011112
1293 012536 010011
1294 012540 012737 000200 001110
1295 012546 000723
1296 012550 010305
1297 012552 020527 013054
1298 012556 103017
1299 012560 062705 000004
1300 012564 005715
1301 012566 001771
1302 012570 032737 000032 001116
1303 012576 001004
1304 012600 105737 173224
1305 012604 001401
1306 012606 000422
1307 012610 010301
1308 012612 010502
1309 012614 000743
1310 012616 032737 000002 013066
1311 012624 001413

.SBTTL CHECK 'SEQBUF' ROUTINE
:+NOW CHECK DEVICE CODES IN 'SEQBUF' FOR ALPHABETICAL ORDER AND
:+CHECK FOR 'HOLES'. IF THERE IS A DUPLICATE DEVICE CODE, PRINT
:+AN ERROR MESSAGE AND EXIT. IF DEVICE CODES NEED SORTING
:+AND/OR HOLES FILLING, DO IT AND SET THE SEQUENCE ERROR
:+FLAG. ALSO CHECK FOR THE 11/60 SPECIAL CASE.

ALPHCK: MOV #SEQBUF, R3 ;+SET UP TO CHECK ALPHABETIC SEQUENCE IN 'SEQBUF'
1$: MOV R3, R4 ;+WILL BE COMPARING (R3) TO (R4)
2$: ADD #4, R4 ;+WHERE R3 AND R4 POINT TO DEVICE CODE LOCATIONS
CMP R4, #ENDSEQ ;+PAST THE LAST DEVICE CODE?
BHI 3$ ;+BRANCH IF YES
TST (R3) ;+IS THERE A DEVICE CODE HERE?
BEQ 7$ ;+BRANCH IF NO
TST (R4) ;+IS THERE A DEVICE CODE HERE?
BEQ 2$ ;+BRANCH IF NO
CMP (R3), (R4) ;+IS THE SEQUENCE CORRECT?
BEQ 4$ ;+BRANCH IF NO-FOUND A DUPLICATE
BGT 5$ ;+BRANCH IF NO-NEED TO DO A SHIFT
BR 2$ ;+YES-CONTINUE
3$: ADD #4, R3 ;+POINT TO NEXT DEVICE CODE
CMP R3, #ENDSEQ ;+ALL DONE?
BHI 1$ ;+BRANCH IF YES
BR 1$ ;+NOT YET
4$: MOV #DUPERR, ROMERR ;+SET ERROR FLAG
MOV #MES1, ARG1 ;+PASS THIS ADDRESS TO 'ERRHAN'
SWAB (R4) ;+BUILD THE MSG
MOV (R4), MES1
CLRB MES1+2
JSR PC, ERRHAN ;+REPORT THE ERROR
RTS PC
5$: MOV R3, R1 ;+SET UP FOR THE SHUFFLE
6$: MOV R4, R2
MOV (R2), R0 ;+SHIFT THE VALUES
MOV (R1), (R2)+
MOV R0, (R1)+
MOV (R2), R0
MOV (R1), (R2)
MOV R0, (R1)
7$: MOV #SEQERR, ROMERR ;+SET THE SEQUENCE ERROR FLAG
BR 2$ ;+CONTINUE SORTING
8$: MOV R3, R5 ;+SET UP TO SEE IF WE HAVE A 'HOLE'
CMP R5, #ENDSEQ ;+END OF TABLE?
BHI 9$ ;+YES-THIS WAS NOT A 'HOLE'
ADD #4, R5 ;+NO-POINT TO NEXT DEVICE CODE LOCATION
TST (R5) ;+IS THERE A ROM HERE?
BEQ 8$ ;+BRANCH IF NO
BIT #32, ROMFIN ;+A ROM IN SOCKET #2 ONLY?
BNE 11$ ;+BRANCH IF NO
TSTB 173224 ;+11/60 SPECIAL CASE?
BEQ 11$ ;+BRANCH IF NO
BR 10$ ;+THE ONLY ROM MUST STAY IN SOCKET #2
11$: MOV R3, R1 ;+YES-THERE IS A HOLE THAT CAN BE FILLED
MOV R5, R2 ;+GET READY TO SHIFT THE VALUES
BR 6$ ;+FILL IN THE HOLE
9$: BIT #2, FLAG ;+AN 11/60?
BEQ 10$ ;+BRANCH IF NO

```



```

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 013074 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 000004              ADD #4, R0            ;+DO SOCKET #2 ONLY
1337 012740 062702 000002              ADD #2, R2            ;+
1338 012744 012737 011176 013100 1$:   MOV #MES2, BUILD      ;+SET UP THE DEVICE CODE PART
1339 012752 000310                      SWAB (R0)            ;+
1340 012754 012037 011176              MOV (R0)+, MES2      ;+
1341 012760 105037 011200              CLRB MES2+2          ;+
1342 012764 012205 004772 2$:         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 #CONMES, BUILD   ;+SET UP FOR CONTINUATION MSG
1352 013022 000760                      BR 2$                 ;+CONTINUE
1353 013024 005710                      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          CONF IN: .WORD 0     ;+CONTINUATION ROM FOUND INDICATOR
1373 013072 000000          DEVF IN: .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 013126          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: ROM1
1389 013130 013520          ROM2
1390 013132 013534          ROM3
1391 013134 013550          ROM4
1392 013136 111 123 040 CONMES: .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: /<12>
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>
1406 013601 105 063 065 .ASCII /E35- M9312 /<15><12>
1407 013616 105 067 065 .ASCIZ /E75-MAPMOD /
1408 013632 122 117 115 BROM2: .ASCII /ROM 2 - /<15><12>
1409 013647 105 063 063 .ASCII /E33- M9312 /<15><12>
1410 013664 105 070 071 .ASCIZ /E89-MAPMOD /
1411 013700 122 117 115 BROM3: .ASCII /ROM 3 - /<15><12>
1412 013715 105 063 064 .ASCII /E34- M9312 /<15><12>
1413 013732 105 070 062 .ASCIZ /E82-MAPMOD /
1414 013746 122 117 115 BROM4: .ASCII /ROM 4 - /<15><12>
1415 013763 105 063 062 .ASCII /E32- M9312 /<15><12>
1416 014000 105 070 063 .ASCIZ /E83-MAPMOD /
1417 000001 .END
    
```


SYMBOL TABLE

ABASE = 000000	AROM21 007203	CKSWR = 104406	LASTAD 003144	ROMFIN 001116
ACDW1 = 000000	AROM3 013454	CLEAR 011676	LF = 000012	ROMNM 012126
ACDW2 = 000000	AROM31 007206	CONER1 013060	LOOP 003060	ROMNUM 007332
ACPUOP= 000000	AROM4 013470	CONER2 013062	MESSAG 001112	ROMPTR 007176
ADDW0 = 000000	AROM41 007211	CONFIN 013070	MESSPT 013104	ROMTYP 012056
ADDW1 = 000000	AROM51 007214	CONMES 013136	MESS24 013116	ROM1 013504
ADDW10= 000000	AROUND 004702	CONGNE= 000400	MESS44 013106	ROM2 013520
ADDW11= 000000	ASWREG= 000000	CONTWO= 001000	MESTAB 013126	ROM3 013534
ADDW12= 000000	ATESTN= 000000	CON1 011714	MES1 010576	ROM4 013550
ADDW13= 000000	AUNIT = 000000	CON2 011742	MES2 011176	RSTART 002204
ADDW14= 000000	AUSWR = 000000	CPUTYP 001122	MFPT = 000007	R6 = 000006
ADDW15= 000000	AVECT1= 000000	CR = 000015	MNORM 007350	R7 = 000007
ADDW2 = 000000	AVECT2= 000000	CRCERR= 000020	NODATA 002630	SAVREG= 104411
ADDW3 = 000000	BELL 007344	CRCLOP 003066	NOROME= 000100	SCOPE = 000004
ADDW4 = 000000	BIT0 = 000001	CRCMSG 010226	NOROMM 010330	SEQBUF 013040
ADDW5 = 000000	BIT00 = 000001	CRLF = 000200	NOROMS 002544	SEQERR= 000200
ADDW6 = 000000	BIT01 = 000002	DAFLAG 003646	OCADD 005140	SEQMSG 010406
ADDW7 = 000000	BIT02 = 000004	DDISP = 1775 '0	OCASC 005034	SEQTST 012174
ADDW8 = 000000	BIT03 = 000010	DEVCOB 003426	OCTBUF 010566	STACK = 001100
ADDW9 = 000000	BIT04 = 000020	DEVFIN 013072	OUTTAB 012666	START 001400
ADEVCT= 000000	BIT05 = 000040	DISPLA 001106	OVER1 004406	STKLMT= 177774
ADEVN = 000000	BIT06 = 000100	DISPRE 000174	OVER10 004722	SWR 001104
AENV = 000000	BIT07 = 000200	DRHEAD 007444	OVER11 004714	SWREG 000176
AENVN = 000000	BIT08 = 000400	DRH24 007524	OVER2 004660	SW0 = 000001
AFATAL= 000000	BIT09 = 001000	DRH44 007465	OVER9 004710	SW00 = 000001
ALPHCK 012410	BIT1 = 000002	DSWR = 177570	PATCHS 002344	SW01 = 000002
AMADR1= 000000	BIT10 = 002000	DUPERR= 002000	PFERR = 000040	SW02 = 000004
AMADR2= 000000	BIT11 = 004000	EHEADT 004744	PFHEAD 007735	SW03 = 000010
AMADR3= 000000	BIT12 = 010000	EMTVEC= 000030	PFMSG 010252	SW04 = 000020
AMADR4= 000000	BIT13 = 020000	END 012050	PIRQ = 177772	SW05 = 000040
AMAMS1= 000000	BIT14 = 040000	ENDSEQ 013054	PIRQVE= 000240	SW06 = 000100
AMAMS2= 000000	BIT15 = 100000	ERL 007424	PPFVAR 003650	SW07 = 000200
AMAMS3= 000000	BIT2 = 000004	ERMES1 013164	PROMP 003146	SW08 = 000400
AMAMS4= 000000	BIT3 = 000010	ERMES2 013243	PRO = 000000	SW09 = 001000
AMSGAD= 000000	BIT4 = 000020	ERMES3 013323	PR1 = 000040	SW1 = 000002
AMSGLG= 000000	BIT5 = 000040	ERMES4 013377	PR2 = 000100	SW10 = 002000
AMSGTY= 000000	BIT6 = 000100	ERRCNT 001120	PR3 = 000140	SW11 = 004000
AMTYP1= 000000	BIT7 = 000200	ERRHAN 004344	PR4 = 000200	SW12 = 010000
AMTYP2= 000000	BIT8 = 000400	ERRMSG 011276	PR5 = 000240	SW13 = 020000
AMTYP3= 000000	BIT9 = 001000	ERROR = 104000	PR6 = 000300	SW14 = 040000
AMTYP4= 000000	BPTVEC= 000014	ERRVEC= 000004	PR7 = 000340	SW15 = 100000
APASS = 000000	BRHEAD 007600	ER1MSG 010036	PS = 177776	SW2 = 000004
APRIOR= 000000	BROM 007320	ER2MSG 010002	PSW = 177776	SW3 = 000010
APTCSU= 000040	BROM1 013564	ER3MSG 010074	PUTMES 003760	SW4 = 000020
APTENV= 000001	BROM11 007217	ER4MSG 010152	PWRVTC= 000024	SW5 = 000040
APTER1= 000001	BROM2 013632	EXCADD 003142	RDCHK = 104407	SW6 = 000100
APTER2= 000002	BROM21 007222	FILBUF 004772	RDLIN = 104410	SW7 = 000200
APTER3= 000004	BROM3 013700	FILTAB 012322	RESREG= 104412	SW8 = 000400
APTER4= 000010	BROM31 007233	FINISH 013102	RESVEC= 000010	SW9 = 001000
APTSIZ= 000200	BROM4 013746	FIRSTA 003140	RETURN 013064	TBITVE= 000014
APTSP0= 000100	BROM41 007244	FIRSTB 001114	RMNUM1 007266	TEMP 005136
ARG1 013074	BROM51 007255	FLAG 013066	RMNUM2 007271	TESTAD 003644
ARG2 013076	BUF1 010502	GTSWR = 104405	RMNUM3 007274	TIMES 004342
AROM 007306	BUF2 010562	HOLCK1 012006	RMNUM4 007277	TKVEC = 000060
AROM1 013424	BUILD 013100	HOLCK2 012020	RMNUM5 007302	TPVEC = 000064
AROM11 007200	CALSUM 003050	HT = 000011	ROMCNT 003046	TRAPVE= 000034
AROM2 013440	CHECKS 002632	IOTVEC= 000020	ROMERR 001110	TRTVEC= 000014

TYPE = 104401	\$DDW14 001260	\$FILLC 005532	\$MTYP1 001171	\$TPB 005526
TYPOC = 104402	\$DDW15 001262	\$FILLS 005531	\$MTYP2 001175	\$TPFLG 005533
TYPON = 104404	\$DDW2 001230	\$GET42 002302	\$MTYP3 001201	\$TPS 005524
TYPOS = 104403	\$DDW3 001232	\$GTSWR 006056	\$MTYP4 001205	\$TRAP 007114
\$APTHD 001124	\$DDW4 001234	\$HD = 000003	\$NULL 005530	\$TRAP2 007136
\$ATYC 005564	\$DDW5 001236	\$HIBTS 001124	\$OCNT 007014	\$TRP = 000013
\$ATY1 005540	\$DDW6 001240	\$INTAG 001101	\$OMODE 007016	\$TRPAD 007150
\$ATY3 005546	\$DDW7 001242	\$LF 005536	\$PASS 001146	\$STSM 001130
\$ATY4 005556	\$DDW8 001244	\$LFLG 006003	\$PASTM 001132	\$TTYIN 006526
\$AUTOB 001100	\$DDW9 001246	\$MADR1 001172	\$QUES 005534	\$TYPE 005154
\$BASE 001214	\$DEVCT 001150	\$MADR2 001176	\$RDCHR 006270	\$TYPEC 005366
\$CDW1 001220	\$DEVN 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	\$MAMS1 001170	\$RTNAD 002324	\$TYPOS 006572
\$CNTLG 006543	\$ENDMG 002331	\$MAMS2 001174	\$SAVRE 007020	\$UNIT 001152
\$CNTLU 006536	\$ENULL 002326	\$MAMS3 001200	\$SETUP= 000124	\$UNITM 001134
\$CPUOP 001166	\$ENV 001160	\$MAMS4 001204	\$STUP = 177777	\$USWR 001164
\$CRLF 005535	\$ENVM 001161	\$MBADR 001126	\$SVPC = 000210	\$VECT1 001210
\$DDW0 001224	\$EOP 002242	\$MFLG 006002	\$SWR = 160000	\$VECT2 001212
\$DDW1 001226	\$EOPCT 002260	\$MNEW 006561	\$SWREG 001162	\$XOFF = 000023
\$DDW10 001250	\$ETABL 001160	\$MSGAD 001154	\$TESTN 001144	\$XON = 000021
\$DDW11 001252	\$ETEND 001264	\$MSGLG 001156	\$TKB 005522	\$GET4= 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
 CZM9BE.BIN,CZM9BE/CR/-SP/NL:TOC=CZM9BE.MLB/ML,CZM9BE.P11

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES				
ABASE	=	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			
ADEVCT	=	000000	6-477	6-477			
AENV	=	000000	6-477	6-477			
AENV	=	000000	6-477	6-477			
AENV	=	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			
AMAMS1	=	000000	6-477	6-477			
AMAMS2	=	000000	6-477	6-477			
AMAMS3	=	000000	6-477	6-477			
AMAMS4	=	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			
APRIOR	=	000000	6-477				
APTCSU	=	000040	21-1022	#22-1024			
APTENV	=	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			
APTSP0	=	000100	21-1022	22-1024	#22-1024		
ARG1	=	013074	17-902	*34-1216	*34-1221	*36-1280	*37-1326 #37-1374

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
ARG2		013076	17-901 *34-1201 *37-1327 #37-1375
AROM		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
ASWREG	-	000000	6-477 6-477
ATESTN	-	000000	6-477 6-477
AUNIT	-	000000	6-477 6-477
AUSWR	=	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	VALUE	REFERENCES
BROM3		013700	37-1386 #37-1411
BROM31		007233	#29-1047 29-1057
BROM4		013746	37-1387 #37-1414
BROM41		007244	#29-1048 29-1057
BROM51		007255	#29-1049 29-1057
BUF1		010502	7-499 13-664 13-677 14-730 16-805 #32-1088
BUF2		010562	13-691 13-697 *15-786 *15-787 *15-793 *15-794 16-853 16-857 #32-1091
BUILD		013100	*37-1338 37-1345 *37-1351 #37-1376
CALSUM		003050	11-583 11-596 #12-616
CHECKS		002632	7-538 #11-577
CKSWR	=	104406	#28-1036
CLEAR		011676	7-505 #33-1116
CONER1		013060	*33-1126 34-1202 34-1204 #37-1368
CONER2		013062	*33-1137 34-1213 *34-1217 34-1218 #37-1369
CONF IN		013070	*7-525 33-1122 33-1135 35-1244 #37-1372
CONMES		013136	33-1169 37-1351 #37-1392
CONONE	=	000400	#4-454 34-1206
CONTWO	=	001000	#4-455 34-1215
CON1		011714	14-742 #33-1122
CON2		011742	14-763 #33-1131
CPUTYP		001122	#4-474 *7-484 *7-486 7-487 7-494 16-808 16-810
CR	-	000015	#4-442 16-818 16-825 16-830 16-856 16-864 17-896 17-933 18-978
			21-1022 21-1022 33-1186 37-1327 37-1333 37-1344
CRERR	-	000020	#4-450 11-587 11-600
CRCLOP		003066	#12-621 12-630
CRMSG		010226	17-963 #31-1076
CRLF	=	000200	#4-442 7-481 7-481 21-1022 21-1022
DAFLAG		003046	*14-737 14-746 *14-765 #14-771
DDISP	=	177570	#4-442 4-468 7-480
DEVCOB		003426	13-651 13-663 #14-730
DEVFIN		013072	*7-529 14-740 #37-1373
DISPLA		001106	#4-468 *7-480 *7-480
DISPRE		000174	#4-458 7-480
DRHEAD		007444	16-816 #30-1063
DRH24		007524	16-814 #30-1065
DRH44		007465	16-812 #30-1064
DSWR	=	177570	#4-442 4-467 7-480
DUPERR	-	002000	#4-456 36-1279
EHEADT		004744	17-888 #17-959
EMTVEC	-	000030	#4-442
END		012050	33-1148 33-1156 #33-1158
ENDSEQ		013054	36-1265 36-1276 36-1297 #37-1366
ERL		007424	#29-1061
ERMES1		013164	17-967 #37-1393
ERMES2		013243	17-968 #37-1394
ERMES3		013323	17-969 #37-1395
ERMES4		013377	37-1331 #37-1396
ERRCNT		001120	#4-473 *17-878
ERRHAN		004344	10-563 11-588 11-601 13-657 13-672 13-684 13-695 13-701 #17-876
			34-1222 36-1284 37-1356
ERRMSG		011276	17-877 17-946 17-950 #32-1096
ERROR	-	104000	#4-442

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES
ERRVEC	= 000004	#4-442 7-480 *7-480 *7-480 7-506 *7-507 *7-518
ER1MSG	010036	17-959 #31-1071
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 17-927
		17-931 #18-974 33-1170 33-1185 37-1332 37-1343 37-1346
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
GNS	= *****	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 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 18-976
		21-1022 21-1022 33-1171 33-1188 34-1201 37-1347
IOTVEC	= 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 18-979 21-1022 21-1022
LOOP	003060	#12-619 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
MESTAB	013126	37-1379 #37-1388
MES1	010576	13-653 #32-1094 34-1216 *34-1218 *34-1219 36-1280 *36-1282 *36-1283 37-1326
		37-1330
MES2	011176	13-660 #32-1095 37-1338 *37-1340 *37-1341
MFPT	= 000007	#6-478 7-485
MNORM	007350	10-556 #29-1060
NODATA	002630	10-558 #10-573
NOROME	= 000100	#4-452 10-562
NOROMM	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	*17-887 #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	#4-442
PIRQVE	=	000240	#4-442
PPFVAR		003650	13-654 13-689 #15-782
PROMP		003146	7-541 #13-649
PRO	-	000000	#4-442
PR1	-	000040	#4-442
PR2		000100	#4-442
PR3	=	000140	#4-442
PR4	=	000170	#4-442
PR5	=	00040	#4-442
PR6	-	000300	#4-442
PR7	=	000340	#4-442
PS	=	177776	#4-442 4-442
PSW	=	177776	#4-442
PUTMES		003760	13-652 13-659 #16-801
PWRVEC	=	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
RMNUM1		007266	#29-1050 29-1058
RMNUM2		007271	#29-1051 29-1058
RMNUM3		007274	#29-1052 29-1058
RMNUM4		007277	#29-1053 29-1058
RMNUM5		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 *15-796 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
ROMFIN		001116	#4-472 *7-503 *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
ROMNUM		012126	33-1167 33-1168 #33-1176
ROMNUM		007332	29-1039 #29-1058
ROMPTR		007176	*7-490 *7-497 17-911 #29-1039
ROMTYP		012056	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	=	%000006	#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	=	%000007	#4-442
SAVREG	=	104411	17-876 #28-1036
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		REFERENCES								
SYMBOL	VALUE	REFERENCES								
STKLMT	= 177774	#4-442								
SWR	001104	#4-467	*7-480	7-480	*7-480	*7-480	7-481	17-879	17-883	17-951
		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							
TIMES	004342	*7-504	16-802	*16-804	#16-872					
TKVEC	= 000060	#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	16-868	17-881	21-1022	23-1026	23-1026
		23-1026	23-1026	23-1026	23-1026	23-1026	23-1026	23-1026	24-1028	#28-1036
TYPOC	= 104402	23-1026	#28-1036							
TYPON	= 104404	#28-1036								
TYPOS	= 104403	#28-1036								
\$APTHD	001124	5-475	#5-475							
\$ASTAT	= *****	22-1024	22-1024							
\$ATYC	005564	22-1024	#22-1024							
\$ATY1	005540	17-945	#22-1024							
\$ATY3	005546	17-949	21-1022	#22-1024						
\$ATY4	005556	#22-1024								
\$AUTOB	001100	#4-464	*7-481	23-1026	23-1026	23-1026				

SYMBOL	CROSS REFERENCE 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				
\$CMTAG	= *****	7-480	7-480					
\$CNTLG	006543	23-1026	#23-1026					
\$CNTLU	006536	23-1026	#23-1026					
\$CPUOP	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						
\$DEVN	001216	#6-477						
\$DOAGN	002322	8-543	8-543	#8-543				
\$ENDAD	002312	4-462	7-481	#8-543				
\$ENDCT	002266	7-480	#8-543					
\$ENDMG	002331	8-543	#8-543					
\$ENULL	002326	8-543	#8-543					
\$ENV	001160	#6-477	7-481	17-936	21-1022	22-1024	22-1024	
\$ENVN	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	005532	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	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						

MACRO CROSS REFERENCE

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