

KD11-Z

11/44 POWER FAIL
CKKACBO

AH-F617B-MC
FICHE 1 OF 1

MAR 1980
COPYRIGHT © 1980
MADE IN USA



Microfiche grid containing multiple frames of data, likely a flight log or maintenance record. The data is too small to read clearly but appears to be organized in columns and rows.



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

.REM %

IDENTIFICATION

PRODUCT CODE:	AC-F615B-MC
PRODUCT NAME:	CKKACBO 11/44 POWER FAIL
DATE CREATED:	1-MAR-79
MAINTAINER:	DIAGNOSTIC ENGINEERING
AUTHORS:	CHUCK ROBINSON

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS MANUAL.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITAL'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

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 BY DIGITAL EQUIPMENT CORPORATION

41
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

CONTENTS

1.	ABSTRACT
2.	REQUIREMENTS
2.1	EQUIPMENT
2.2	STORAGE
2.3	PRELIMINARY PROGRAMS
3.	LOADING PROCEDURE
4.	STARTING PROCEDURE
4.1	STARTING ADDRESS
4.2	CONTROL SWITCH SETTINGS
5.	OPERATING PROCEDURE
5.1	MANUAL
5.2	AUTOMATIC
5.3	APT SETUP
5.4	SUBROUTINE ABSTRAC
6.	ERRORS
7.	RESTRICTIONS
8.	MISCELLANEOUS
8.1	EXECUTION TIME
8.2	STACK POINTER

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

1. ABSTRACT

THIS PROGRAM IS MADE UP OF 11 SUBTESTS TO CHECK OUT THE POWER FAIL ON THE 11/44. THE 2 MSEC. POWER DOWN AND POWER UP TIME IS CHECKED ON EACH POWER FAIL. INITIALLY POWER FAILS ARE TRIED IN ALL PROCESSOR MODES THEN UNDER ERROR CONDITIONS LIKE STACK OVERFLOW, TIME OUT, AND ODD ADDRESS AND MEMORY MANAGEMENT ABORTS. FINALLY A MEMORY VOLATILITY TEST IS RUN ON ALL AVAILABLE MEMORY.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP11/44 STANDARD COMPUTER WITH UP TO 1M WORDS OF CORE MEMORY.

THIS PROGRAM WILL WORK WITH AUTOMATIC POWER FAIL HARDWARE (MANUFACTURING ONLY) OR WITH MANUALLY TRIGGERED POWER FAILURES FOR EACH TEST.

2.2 STORAGE

PROGRAM STORAGE - THE ROUTINES USE MEMORY 0 - 10142

2.3 PRELIMINARY PROGRAMS

IT IS ASSUMED THAT CPU, TRAPS, MEMORY MANAGEMENT AND UNIBUS MAP DIAGNOSTICS HAVE BEEN RUN SUCCESSFULLY.

3. LOADING PROCEDURE

USE STANDARD PROCEDURE FOR 'XXDP' MEDIA.

4. STARTING PROCEDURE

4.1 STARTING ADDRESS

200 IS THE PROGRAM STARTING ADDRESS

4.2 CONTROL SWITCH SETTINGS

SWITCH REGISTER <14> CONTROLS REPEATING OF A FAILING TEST. THIS BIT MUST BE SET TO A 1 (AFTER ERROR HALT) EVERY TIME THE TEST IS TO BE REPEATED.

147
148
149

151
152
153
154
155
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

5. OPERATING PROCEDURE

5.1 MANUAL

START PROGRAM AT ADDRESS 200.
A MESSAGE WILL BE TYPE IDENTIFYING THE NAME OF THE PROGRAM,
THE SIZE OF MEMORY, AND RUNNING INSTRUCTIONS. THE NUMBER OF
EACH TEST WILL BE PRINTED AT THE BEGINNING OF EACH TEST.
MANUALLY POWER DOWN THEN UP AFTER EACH TEST NUMBER APPEARS
ON THE TTY. DO THIS FOR EACH TEST UNTIL THE END OF PASS
MESSAGE IS RECEIVED.

5.2 AUTOMATIC

START THE PROGRAM AT 200, THE PROGRAM SIZES FOR THE PRESENCE OF
AUTOMATIC POWER FAIL HARDWARE. IF FOUND, TESTING WILL BE PROCEED
WITH NO MANUAL INTERVENTION REQUIRED. TEST NUMBERS WILL BE PRINTED
PRIOR TO EACH TEST.

5.3 APT SETUP

THE EXECUTION TIMES PROVIDED IN THE APT SCRIPT THAT FOLLOWS
ARE FOR EXECUTION WITH A 11/44 PROCESSOR, CACHE,
16K CORE MEMORY, AND 300 BAUD.

THE FOLLOWING IS A PROGRAM LOAD FILE USED BY APT:

1. Z TABLE 'A' IS USED FOR APT DUMP MODE.
2. E TABLE 'B' IS USED FOR APT QV AND RUN TIME MODES.
\$ENVM=240 SUPRESSES TYPEOUTS TO THE TERMINAL AND
\$SWREG=4000 INHIBITS TEST ITERATIONS

	1ST PASS RUN TIME	LONGEST TEST TIME	ADDITIONAL RUN TIME
	5	5	0
.....		E TABLES
		A	B
E-MODE/S-MODE (\$ENVM/\$ENV)		000/000	240/001
SWITCH REGISTER 1 (\$SWREG)		000000	004000
SWITCH REGISTER 2 (CPU TYPE/OPTIONS)		000000 00/0000	000000 00/0000
MEMORY MAP CODE 1		000/00000000	000/00000000



208
209
210
211
212
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

5.4 SUBROUTINE ABSTRACT

PDWDWN AND POWUP

THESE ROUTINES ARE USED TO SAVE AND RESTORE VITAL REGISTERS
AND TEST THE TIME ALLOWED FOR POWER FAIL BY THE PROCESSOR.

6. ERRORS

6.1 ERROR PRINTOUT

THE PROGRAM WILL IDENTIFY THE FAILING TEST AND PROVIDE
PERTENANT INFORMATION ON ERROR.

6.2 ERROR HALTS

AFTER AN ERROR PRINTOUT, THE PROGRAM WILL HALT AND RETURN
CONTROL TO THE CONSOLE.
IF USING POWER FAIL HARDWARE AND YOU WISH TO STOP THE PROCESSOR
YOU SHOULD USE THE HALT SWITCH ON THE FRONT PANEL, THE CONSOLE
CTRL P WILL HANG UP TTY. TO GET IN CONSOLE MODE PUT HALT SWITCH
ON, TO GET BACK INTO PROGRAM SET THE SWICTH TO RUN.

IF AN ERROR OCCURS IN TEST 11 (THE MEMORY VOLITILITY TEST)
THE PHYSICAL ADDRESS OF THE BAD MEMORY LOCATION CAN BE
FOUND BY USING PAR.OFF AS BITS <21:6> AND ADDING VIR.ADD
LOCATION <12:0> TO IT.
IF AN ERROR IS DETECTED IN THE POWER DOWN OR POWER UP SUBROUTINE
THEN CONTINUING AFTER THE ERROR HALT, WILL RESTART THE PROGRAM
AT ADDRESS 200.

247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274

6.3 ERROR RECOVERY
CONTINUE OR RESTART AT 200

7. RESTRICTIONS
NONE

8. MISCELLANEOUS

8.1 EXECUTION TIME
5 SEC IF IN AUTOMATIC , N/A IF IN MANUAL

8.2 STACK POINTER
STACK IS INITALLY SET TO 1100

%

276
289

```
.TITLE CKKACAO 11/44 POWER FAIL
:*COPYRIGHT (C) 1979
:*DIGITAL EQUIPMENT CORP.
:*MAYNARD, MASS. 01754
:*
:*PROGRAM BY CHUCK ROBINSON
:*
:*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYS:MAC
:*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
:*
```

290
291
292

```
.SBTTL OPERATIONAL SWITCH SETTINGS
:*
:*      SWITCH          USE
:*      -----          -----
:*      14              LOOP ON TEST
:*      10              INHIBIT BELL AT END-OF-PASS
```

293
325

```
.SBTTL BASIC DEFINITIONS
:*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100 STACK= 1100          ;;FIRST ADDRESS OF THE STACK
001100 KERSTK= STACK      ;;KERNEL STACK
000700 SUPSTK= STACK-200  ;;SUPERVISOR STACK
000600 USESTK= STACK-300   ;;USER STACK
104000 ERROR=EMT
000004 SCOPE=IOT
177776 PS= 177776       ;;PROCESSOR STATUS WORD
177776 PSW=PS
177774 STKLMT= 177774   ;;STACK LIMIT REGISTER
177772 PIRQ= 177772     ;;PROGRAM INTERRUPT REQUEST REGISTER
177570 DSWR= 177570    ;;HARDWARE SWITCH REGISTER
177570 DDISP= 177570   ;;HARDWARE DISPLAY REGISTER
177546 LKS= 177546     ;;LINE CLOCK (KW11-L) STATUS REGISTER
:*MISCELLANEOUS DEFINITIONS
000011 HT= 11          ;;CODE FOR HORIZONTAL TAB
000012 LF= 12          ;;CODE LINE FEED
000015 CR= 15          ;;CODE CARRIAGE RETURN
000200 CRLF= 200         ;;CODE FOR CARRIAGE RETURN-LINE FEED
:*GENERAL PURPOSE REGISTER DEFINITIONS
000000 R0= %0           ;;GENERAL REGISTER
000001 R1= %1           ;;GENERAL REGISTER
000002 R2= %2           ;;GENERAL REGISTER
000003 R3= %3           ;;GENERAL REGISTER
000004 R4= %4           ;;GENERAL REGISTER
000005 R5= %5           ;;GENERAL REGISTER
000006 R6= %6           ;;GENERAL REGISTER
000007 R7= %7           ;;GENERAL REGISTER
000000 R10=R0
000001 R11=R1
000002 R12=R2
000003 R13=R3
000004 R14=R4
000005 R15=R5
000006 SP= %6          ;;STACK POINTER
000006 KSP=SP
```

```
000006 SSP=SP
000006 USP=SP
000007 PC= 27 ::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
000100 LKVEC= 100  ;; LINE CLOCK (KW11-L) VECTOR
000114 CACHVEC=114  ;; CACHE ERROR INTERRUPT VECTOR
000240 PIRQVEC=240  ;; PROGRAM INTERRUPT REQUEST VECTOR
000250 MMVEC= 250  ;; MEMORY MANAGEMENT VECTOR
.SBTTL CACHE REGISTER DEFINITIONS
177740 LOADRS = 177740 ;; LOWER 16 BITS OF ADDRESS THAT CAUSED ERROR
177742 HIADRS = 177742 ;; UPPER SIX BITS OF ADDRESS THAT CAUSED ERROR
177744 MEMERR = 177744 ;; CACHE ERROR REGISTER
177746 CONTRL = 177746 ;; MEMORY CONTROL REGISTER
177750 MAINT = 177750 ;; MEMORY MAINTENANCE REGISTER
177752 HITMIS = 177752 ;; HIT MISS REGISTER '1' IMPLIES HIT IN CACHE
.SBTTL CPU REGISTER DEFINITIONS
177760 SIZELO = 177760 ;; MEMORY SIZE REGISTER NUMBER TO PUT INTO A PAR
;; TO GET TO THE LAST 32 WORDS OF MEMORY
177762 SIZEHI = 177762 ;; HIGH SIZE REGISTER, RESERVED FOR FUTURE USE
;; CURRENTLY ALL ZERO
177764 SYSTID = 177764 ;; SYSTEM ID REGISTER
177766 CPUERR = 177766 ;; CPU ERROR REGISTER HOLDS CONDITION THAT CAUSED
;; THE TRAP TO ERRVEC (000004)
.SBTTL MEMORY MANAGEMENT DEFINITIONS
;*MEMORY MANAGEMENT STATUS REGISTER ADDRESSES
177572 MMR0= 177572
177574 MMR1= 177574
177576 MMR2= 177576
172516 MMR3= 172516
177572 SR0=MMR0
177574 SR1=MMR1
177576 SR2=MMR2
172516 SR3=MMR3
;*USER 'I' PAGE DESCRIPTOR REGISTERS
177600 UIPDR0= 177600
177602 UIPDR1= 177602
177604 UIPDR2= 177604
177606 UIPDR3= 177606
177610 UIPDR4= 177610
177612 UIPDR5= 177612

```

177614	UIPDR6= 177614
177616	UIPDR7= 177616
	:*USER 'D' PAGE DESCRIPTOR REGISTORS
177620	UDPDR0= 177620
177622	UDPDR1= 177622
177624	UDPDR2= 177624
177626	UDPDR3= 177626
177630	UDPDR4= 177630
177632	UDPDR5= 177632
177634	UDPDR6= 177634
177636	UDPDR7= 177636
	:*USER 'I' PAGE ADDRESS REGISTERS
177640	UIPAR0= 177640
177642	UIPAR1= 177642
177644	UIPAR2= 177644
177646	UIPAR3= 177646
177650	UIPAR4= 177650
177652	UIPAR5= 177652
177654	UIPAR6= 177654
177656	UIPAR7= 177656
	:*USER 'D' PAGE ADDRESS REGISTERS
177660	UDPAR0= 177660
177662	UDPAR1= 177662
177664	UDPAR2= 177664
177666	UDPAR3= 177666
177670	UDPAR4= 177670
177672	UDPAR5= 177672
177674	UDPAR6= 177674
177676	UDPAR7= 177676
	:*SUPERVISOR 'I' PAGE DESCRIPTOR REGISTERS
172200	SIPDR0= 172200
172202	SIPDR1= 172202
172204	SIPDR2= 172204
172206	SIPDR3= 172206
172210	SIPDR4= 172210
172212	SIPDR5= 172212
172214	SIPDR6= 172214
172216	SIPDR7= 172216
	:*SUPERVISOR 'D' PAGE DESCRIPTOR REGISTERS
172220	SDPDR0= 172220
172222	SDPDR1= 172222
172224	SDPDR2= 172224
172226	SDPDR3= 172226
172230	SDPDR4= 172230
172232	SDPDR5= 172232
172234	SDPDR6= 172234
172236	SDPDR7= 172236
	:*SUPERVISOR 'I' PAGE ADDRESS REGISTERS
172240	SIPAR0= 172240
172242	SIPAR1= 172242
172244	SIPAR2= 172244
172246	SIPAR3= 172246
172250	SIPAR4= 172250
172252	SIPAR5= 172252
172254	SIPAR6= 172254
172256	SIPAR7= 172256
	:*SUPERVISOR 'D' PAGE ADDRESS REGISTERS

```
172260 SDPAR0= 172260
172262 SDPAR1= 172262
172264 SDPAR2= 172264
172266 SDPAR3= 172266
172270 SDPAR4= 172270
172272 SDPAR5= 172272
172274 SDPAR6= 172274
172276 SDPAR7= 172276
;*KERNEL 'I' PAGE DESCRIPTOR REGISTERS
172300 KIPDR0= 172300
172302 KIPDR1= 172302
172304 KIPDR2= 172304
172306 KIPDR3= 172306
172310 KIPDR4= 172310
172312 KIPDR5= 172312
172314 KIPDR6= 172314
172316 KIPDR7= 172316
;*KERNEL 'D' PAGE DESCRIPTOR REGISTERS
172320 KDPDR0= 172320
172322 KDPDR1= 172322
172324 KDPDR2= 172324
172326 KDPDR3= 172326
172330 KDPDR4= 172330
172332 KDPDR5= 172332
172334 KDPDR6= 172334
172336 KDPDR7= 172336
;*KERNEL 'I' PAGE ADDRESS REGISTERS
172340 KIPAR0= 172340
172342 KIPAR1= 172342
172344 KIPAR2= 172344
172346 KIPAR3= 172346
172350 KIPAR4= 172350
172352 KIPAR5= 172352
172354 KIPAR6= 172354
172356 KIPAR7= 172356
;*KERNEL 'D' PAGE ADDRESS REGISTERS
172360 KDPAR0= 172360
172362 KDPAR1= 172362
172364 KDPAR2= 172364
172366 KDPAR3= 172366
172370 KDPAR4= 172370
172372 KDPAR5= 172372
172374 KDPAR6= 172374
172376 KDPAR7= 172376
.SBTTL UNIBUS MAP REGISTER DEFINITIONS
;*THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED 'MAPLXX'
;*THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED 'MAPHXX'
170200 MAPL00 = 170200
170202 MAPH00 = 170202
170204 MAPL01 = 170204
170206 MAPH01 = 170206
170210 MAPL02 = 170210
170212 MAPH02 = 170212
170214 MAPL03 = 170214
170216 MAPH03 = 170216
170220 MAPL04 = 170220
170222 MAPH04 = 170222
```

170224	MAPL05 = 170224
170226	MAPH05 = 170226
170230	MAPL06 = 170230
170232	MAPH06 = 170232
170234	MAPL07 = 170234
170236	MAPH07 = 170236
170240	MAPL10 = 170240
170242	MAPH10 = 170242
170244	MAPL11 = 170244
170246	MAPH11 = 170246
170250	MAPL12 = 170250
170252	MAPH12 = 170252
170254	MAPL13 = 170254
170256	MAPH13 = 170256
170260	MAPL14 = 170260
170262	MAPH14 = 170262
170264	MAPL15 = 170264
170266	MAPH15 = 170266
170270	MAPL16 = 170270
170272	MAPH16 = 170272
170274	MAPL17 = 170274
170276	MAPH17 = 170276
170300	MAPL20 = 170300
170302	MAPH20 = 170302
170304	MAPL21 = 170304
170306	MAPH21 = 170306
170310	MAPL22 = 170310
170312	MAPH22 = 170312
170314	MAPL23 = 170314
170316	MAPH23 = 170316
170320	MAPL24 = 170320
170320	MAPH24 = 170320
170324	MAPL25 = 170324
170326	MAPH25 = 170326
170330	MAPL26 = 170330
170332	MAPH26 = 170332
170334	MAPL27 = 170334
170336	MAPH27 = 170336
170340	MAPL30 = 170340
170342	MAPH30 = 170342
170344	MAPL31 = 170344
170346	MAPH31 = 170346
170350	MAPL32 = 170350
170352	MAPH32 = 170352
170354	MAPL33 = 170354
170356	MAPH33 = 170356
170360	MAPL34 = 170360
170362	MAPH34 = 170362
170364	MAPL35 = 170364
170366	MAPH35 = 170366
170370	MAPL36 = 170370
170372	MAPH36 = 170372
170374	MAPL37 = 170374
170376	MAPH37 = 170376
170200	MAPL0=MAPL00
170202	MAPH0=MAPH00
170204	MAPL1=MAPL01

```

170206      MAPH1=MAPH01
170210      MAPL2=MAPL02
170212      MAPH2=MAPH02
170214      MAPL3=MAPL03
170216      MAPH3=MAPH03
170220      MAPL4=MAPL04
170222      MAPH4=MAPH04
170224      MAPL5=MAPL05
170226      MAPH5=MAPH05
170230      MAPL6=MAPL06
170232      MAPH6=MAPH06
170234      MAPL7=MAPL07
170236      MAPH7=MAPH07
326         .SBTTL TRAP CATCHER
           .=0
           ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A '+2,HALT'
           ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
           ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
           .=174
000174      000000      DISPREG: .WORD 0           ;;SOFTWARE DISPLAY REGISTER
000176      000000      SWREG:   .WORD 0           ;;SOFTWARE SWITCH REGISTER
           .SBTTL STARTING ADDRESS(ES)
           JMP          @MBEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
327         000200      000137      001424      STACK1=500
328         000500      $SAVPC=.
329         000030      000030      .=30
330         000030      006616      $ERROR
331         000032      000340      340
332         000204      .=$SAVPC
334         .SBTTL ACT11 HOOKS
           ;*****
           ;HOOKS REQUIRED BY ACT11
           $SVPC=.           ;SAVE PC
           .=46
000046      000204      000046      $ENDAD           ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
           .=52
000052      000000      .WORD 0           ;;2)SET LOC.52 TO ZERO
           .=$SVPC           ;; RESTORE PC
335         .=1100
336         .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.
000024      000200      $APTHDR ;;POINT TO APT HEADER BLOCK
000044      001100      .=$X           ;;RESET LOCATION COUNTER
           ;*****
           ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
           ;INTERFACE SPEC.
001100      $APTHD:
001100      000000      $HIBTS: .WORD 0           ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001102      001114      $MBADR: .WORD $MAIL       ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001104      000005      $STMT:  .WORD 5           ;;RUN TIM OF LONGEST TEST
001106      000005      $PASTM: .WORD 5           ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)

```

337
001110 000000
001112 000016

\$UNITM: .WORD 0 ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
\$ETEND: .WORD \$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
\$SBTTL APT MAILBOX-ETABLE

001114
001114 000000
001116 000000
001120 000000
001122 000000
001124 000000
001126 000000
001130 000000
001132 000000
001134
001134 000
001135 000
001136 000000
001140 000000
001142 000000

\$MAIL: .WORD 0 ;;APT MAILBOX
\$MSGTY: .WORD AMSTY ;;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 AMGLG ;;MESSAGE LENGTH
\$ETABLE: .WORD 0 ;;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

001144 000
001145 000

\$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

001146 000000
001150

\$MADR1: .WORD AMADR1 ;;HIGH ADDRESS,BLK#1
MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE

338

001150 112767 000001 000236
001156 112767 000001 000226
001164 000403
001166 112767 000001 000220
001174
001174 010046
001176 010146
001200 105767 000206
001204 001450
001206 122767 000001 177720
001214 001031
001216 132767 000100 177711
001224 001425
001226 017600 000004
001232 062766 000002 000004
001240 005767 177650
001244 001375
001246 010067 177656
001252 105720

\$MEXIT
\$SBTTL APT COMMUNICATIONS ROUTINE

\$SATY1: MOV #1,\$FFLG ;;TO REPORT FATAL ERROR
\$SATY3: MOV #1,\$MFLG ;;TO TYPE A MESSAGE
BR \$ATYC
\$SATY4: MOV #1,\$FFLG ;;TO ONLY REPORT FATAL ERROR
\$ATYC: MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
TSTB \$MFLG ;;SHOULD TYPE A MESSAGE?
BEQ 5\$;;IF NOT: BR
CMPB #APTENV,\$ENV ;;OPERATING UNDER APT?
BNE 3\$;;IF NOT: BR
BITB #APTSPOOL,\$ENVM ;;SHOULD SPOOL MESSAGES?
BEQ 3\$;;IF NOT: BR
MOV @4(SP),R0 ;;GET MESSAGE ADDR.
ADD #2,4(SP) ;;BUMP RETURN ADDR.
1\$: TST \$MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
BNE 1\$;;IF NOT: WAIT
MOV R0,\$MSGAD ;;PUT ADDR IN MAILBOX
2\$: TSTB (R0)+ ;;FIND END OF MESSAGE


```

001254 001376          BNE      2$
001256 166700 177646   SUB      $MSGAD,R0      ;;SUB START OF MESSAGE
001262 006200          ASR      R0              ;;GET MESSAGE LNGTH IN WORDS
001264 010067 177642   MOV      R0,$MSGLGT    ;;PUT LENGTH IN MAILBOX
001270 012767 000004 177616   MOV      #4,$MSGTYPE   ;;TELL APT TO TAKE MSG.
001276 000413          BR       5$
001300 017667 000004 000016 3$:   MOV      @4(SP),4$      ;;PUT MSG ADDR IN JSR LINKAGE
001306 062766 000002 000004          ADD      #2,4(SP)      ;;BUMP RETURN ADDRESS
001314 016746 176456          MOV      177776,-(SP)  ;;PUSH 177776 ON STACK
001320 004767 004646          JSR     PC,$TYPE      ;;CALL TYPE MACRO
001324 000000          4$:   .WORD    0
001326          5$:
001326 105767 000062 10$:   TSTB    $FFLG          ;;SHOULD REPORT FATAL ERROR?
001332 001416          BEQ     12$          ;;IF NOT: BR
001334 005767 177574          TST     $ENV          ;;RUNNING UNDER APT?
001340 001413          BEQ     12$          ;;IF NOT: BR
001342 005767 177546 11$:   TST     $MSGTYPE     ;;FINISHED LAST MESSAGE?
001346 001375          BNE     11$          ;;IF NOT: WAIT
001350 017667 000004 177540   MOV      @4(SP),$FATAL ;;GET ERROR #
001356 062766 000002 000004          ADD      #2,4(SP)      ;;BUMP RETURN ADDR.
001364 005267 177524          INC     $MSGTYPE     ;;TELL APT TO TAKE ERROR
001370 105067 000020 12$:   CLRB    $FFLG          ;;CLEAR FATAL FLAG
001374 105067 000013          CLRB    $LFLG          ;;CLEAR LOG FLAG
001400 105067 000006          CLRB    $MFLG          ;;CLEAR MESSAGE FLAG
001404 012601          MOV     (SP)+,R1      ;;POP STACK INTO R1
001406 012600          MOV     (SP)+,R0      ;;POP STACK INTO R0
001410 000207          RTS     PC           ;;RETURN
001412 000          $MFLG: .BYTE    0      ;;MESSG. FLAG
001413 000          $LFLG: .BYTE    0      ;;LOG FLAG
001414 000          $FFLG: .BYTE    0      ;;FATAL FLAG
                                .EVEN
                                APTSIZE=200
                                APTENV=001
                                APTSPool=100
                                APTCSUP=040
339
340 001416 000000   PFAIL: .WORD 0
341 001420 000000   XYZ:  .WORD 0
342          177570   SWR=177570
343          177570   DISPLAY=SWR
344 001422 000002   LRTI:  RTI
345
346 001424   BEGIN:
                                .SBTTL INITIALIZE THE COMMON TAGS
001424 012706 001100   MOV     #STACK,SP      ;;SETUP THE STACK POINTER
                                ;;INITIALIZE A FEW VECTORS
001430 012737 006544 000034   MOV     #STRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
001436 012737 000340 000036   MOV     #340,@TRAPVEC+2;LEVEL 7
001444 005067 177452          CLR     $PASS          ;;CLEAR THE PASS COUNT
001450 016767 002512 002502   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.
001456 013746 000004          MOV     @ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
001462 012737 001516 000004          MOV     #64,$@ERRVEC  ;;SET UP ERROR VECTOR
001470 012767 177570 176072          MOV     #DSWR,SWR     ;;SETUP FOR A HARDWARE SWICH REGISTER
001476 012767 177570 176064          MOV     #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
001504 022777 177777 176056          CMP     #-1,@SWR      ;;TRY TO REFERENCE HARDWARE SWR

```

```

001512 001012          BNE      66$          ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
001514 000403          BR       65$          ;;AND THE HARDWARE SWR IS NOT - -1
001516 012716 001524   64$:  MOV      #65$, (SP)  ;;BRANCH IF NO TIMEOUT
001522 000002          RTI          ;;SET UP FOR TRAP RETURN
001524 012767 000176 176036 65$:  MOV      #SWREG, SWR  ;;POINT TO SOFTWARE SWR
001532 012767 000174 176030  MOV      #DISPREG, DISPLAY
001540 012637 000004          MOV      (SP)+, @#ERRVEC  ;;RESTORE ERROR VECTOR
001544 005067 177352          CLR      $PASS        ;;CLEAR PASS COUNT
001550 132767 000200 177357  BITB     #APTSIZE, $ENVM  ;;TEST USER SIZE UNDER APT
001556 001403          BEQ      67$          ;;YES, USE NON-APT SWITCH
001560 012767 001136 176002  MOV      #SSWREG, SWR   ;;NO, USE APT SWITCH REGISTER
001566          67$:
347 001566 012777 004654 003300  MOV      #POWDWN, @DVEC  ;;SET UP POWER DOWN VECTOR
348 001574 012737 004554 000114  MOV      #PARERR, @#CACHVEC ;;SET UP PARITY ERROR VECTOR
349 001602 012737 000001 004556  MOV      #1, @#PARFLG   ;;INITIALIZE THE MULTI PARITY ERROR INDICATOR
350
351          ;DETERMINE THE SIZE OF MEMORY IN SYSTEM USING SYSMAC SIZE
352          ;ROUTINE
353
354 001610 005000          CLR      R0           ;MAKE MSB'S = 0
355 001612 052767 000200 003430  BIS      #BIT07, $KT11  ;TURN ON MEMORY MANAGMENT
356 001620 004767 003366          JSR      PC, $SIZE     ;GET PAR VALUE FOR LAST WORD
357 001624 062767 000037 003664  ADD      #37, $LSTBK    ;FINE LAST ADDRESS OF MEMORY BANK
358 001632 016701 003660          MOV      $LSTBK, R1    ;CHECK LAST BANK WITH R1
359 001636 005201          INC      R1           ;SIZE MEMORY
360 001640 071027 000200          DIV      #200, R0      ;FORM THE BLOCK NUMBER OF
361 001644 005300          DEC      R0           ; THE LAST ADDRESS AND
362 001646 010067 003232          MOV      R0, LIMIT    ;SAVE IT.
363 001652 005200          INC      R0           ;MULT BY 4
364 001654 006300          ASL      R0
365 001656 006300          ASL      R0
366
367
368          ;THIS ROUTINE SIZE FOR PFAIL HARDWARE, TO AUTOMATIC POWER FAIL THE SYSTEM
369 001660 012767 177740 177530  MOV      #177740, PFAIL ;SETUP PFAIL ADDRESS
370 001666 012767 001710 176110  MOV      #3$, 4        ;SET PC
371 001674 012767 000340 176104  MOV      #340, 6       ;SET PSW
372 001702 005777 177510          TST      @PFAIL        ;READ PFAIL
373 001706 000412          BR       4$           ;POWER FAIL HARDWARE IS IN SYSTEM
374 001710 012767 000006 176066  3$:  MOV      #6, 4        ;TRAP RETURN
375 001716 005067 176064          CLR      6
376 001722 005726          TST      (SP)+        ;SETUP STACK PC
377 001724 005726          TST      (SP)+        ;SETUP STACK PSW
378 001726 012767 001420 177462  MOV      #XYZ, PFAIL   ;SET UP DUMMY ADDRESS
379 001734          4$:
380 001734 012767 000006 176042  MOV      #6, 4        ;RESTORE TRAP VEC
381 001742 005067 176040          CLR      6
382
383          ;IDENTIFY THE PROGRAM AND TYPE INSTRUCTIONS
384
385 001746 023767 000042 002244  CMP      @#42, $ENDAD
386 001754 001547          BEQ      PRETST
387 001756 104401          TYPE
388 001760 010005          TITLE
389 001762 104401          TYPE
390 001764 010042          BOOT

```

```

391 001766 010046      MOV      RO,-(SP)          ;;SAVE RO FOR TYPEOUT
      001770 104405      TYPDS                    ;;GO TYPE--DECIMAL ASCII WITH SIGN
392 001772 104401 002000  TYPE      ,69$           ;;TYPE ASCIZ STRING
      001776 000415      BR        68$           ;;GET OVER THE ASCIZ
      ;;69$: .ASCIZ  *-K WORDS OF MEMORY EXIST *
      68$:
393 002032 026727 177360 001420  CMP      PFAIL,#XYZ      ;IS AUTOMATIC POWER FAIL
394 002040 001035      BNE      APF            ;BRANCH IF YES
395 002042 104401 002050  TYPE      ,71$           ;;TYPE ASCIZ STRING
      002046 000431      BR        70$           ;;GET OVER THE ASCIZ
      ;;71$: .ASCIZ  <CRLF><LF>* MANUALLY INTERRUPT THE POWER AFTER EACH TEST # *
      70$:
396 002132 000454      BR        FT
397 002134 002134 002142  APF:     TYPE      ,65$           ;;TYPE ASCIZ STRING
      002140 000422      BR        64$           ;;GET OVER THE ASCIZ
      ;;65$: .ASCIZ  <CRLF><LF>* NO MANUAL INTERVENTION REQUIRED*
      64$:
398 002206 104401 002214  TYPE      ,67$           ;;TYPE ASCIZ STRING
      002212 000424      BR        66$           ;;GET OVER THE ASCIZ
      ;;67$: .ASCIZ  <CRLF><LF>* AUTO POWER FAIL TEST HARDWARE EXIST*
      66$:
399 002264
400 002264 104401 002272  FT:     TYPE      ,65$           ;;TYPE ASCIZ STRING
      002270 000401      BR        64$           ;;GET OVER THE ASCIZ
      ;;65$: .ASCIZ  <CRLF>
      64$:
401 002274 012767 000001 176616  PRETST: MOV      #1,$TESTN
402      ;THIS ROUTINE IS FOR CHECKING 2MS TIMING USING SOB OUT OF SIPARS
403      ;BRANCH . OUT OF SIPARO=1.2US
404      ;SOB HAS ONE MORE MICRO STEP THROUGH EQUAL 1.4US
405      ;1024 MUL BY 1.4US EQUAL 1.43MS MIN TIME CHECKED FOR ILLUP,ILLDWN
406 002302 012737 012700 172240  MOV      #12700,@#SIPARO ;LOAD PAR0 & 1 WITH MOV #1777,RO
407 002310 012737 001777 172242  MOV      #1777,@#SIPAR1
408 002316 012737 077001 172244  MOV      #77001,@#SIPAR2 ;LOAD PAR2 WITH SOB RO..
409 002324 012737 000207 172246  MOV      #207,@#SIPAR3 ;LOAD PAR3 WITH RTS PC
410      ;*****
      ;*TEST 1 SIMPLE DOWN/UP TEST (KERNAL)
      ;*****
      TST1:
      002332 104401 002340  TYPE      ,65$           ;;TYPE ASCIZ STRING
      002336 000402      BR        64$           ;;GET OVER THE ASCIZ
      ;;65$: .ASCIZ  *1 *
      64$:
      002344 104401 004234  TYPE ,SENULL
411 002350 005037 177776  CLR      @#PS           ;SET KERNAL MODE
412 002354 012703 002370  MOV      #2$ ,R3        ;SET POWER UP RETURN
413 002360 012777 000001 177030  MOV      #1,@PFAIL      ;SET UP TO START POWER FAIL
414 002366 000001      WAIT                    ;WAIT FOR POWER FAIL
415 002370 010600 2$:     MOV      SP,RO      ;GET SP
416 002372 022700 001074  CMP      #STACK-4,RO    ;CHECK SP
417 002376 001414      BEQ      3$            ;SKIP IF OK
418 002400 012767 001074 005330  MOV      #1074,EXP.SP1
419 002406 010067 005326      MOV      RO,REC.SP1
420 002412 012706 000500      MOV      #STACK1,SP    ;SETUP RESERVED STACK POINTER
                          ;FOR ERROR PRINTOUT

```

```

002416 104000 ERROR
;ERROR
;-----
421 ;SP NOT STACK-4
422 002420 007736 EXP.SP1 ;PRINT EXPECTED STACK POINTER
423 002422 007740 REC.SP1 ;PRINT RECEIVED STACK POINTER
424 002424 000000 0
425 002426 000000 0
426 002430 012706 001100 3$: MOV #STACK,SP ;RESET SP
427 002434 013700 001074 MOV @#STACK-4,R0 ;GET RETURN ADDRESS
428 002440 022700 002370 CMP #2$,R0 ;CHECK ADDRESS
429 002444 001414 BEQ 4$ ;SKIP IF OK
430 002446 012767 002370 005256 MOV #2$,EXP.ADD
431 002454 010067 005254 MOV R0,REC.ADD
432 002460 012706 000500 MOV #STACK1,SP ;SETUP RESERVED STACK POINTER
;FOR ERROR PRINTOUT
002464 104000 ERROR
;ERROR
;-----
433 ;ADDRESS ON STACK IS WRONG
434 002466 007732 EXP.ADD ;PRINT EXPECTED ADDRESS
435 002470 007734 REC.ADD ;PRINT RECEIVED ADDRESS
436 002472 000000 0
437 002474 000000 0
438 002476 013700 001076 4$: MOV @#STACK-2,R0 ;GET OLD PS
439 002502 022700 000000 CMP #0,R0 ;CHECK OLD PS
440 002506 001414 BEQ 5$ ;SKIP IF OK
441 002510 012767 000000 005224 MOV #0,EXP.PSW
442 002516 010067 005222 MOV R0,REC.PSW
443 002522 012706 000500 MOV #STACK1,SP ;SETUP RESERVED STACK POINTER
;FOR ERROR PRINTOUT
002526 104000 ERROR
;ERROR
;-----
444 ;OLD PS IS WRONG
445 002530 007742 EXP.PSW ;PRINT EXPECTED PS
446 002532 007744 REC.PSW ;PRINT RECEIVED PS
447 002534 000000 0
448 002536 000000 0
449 002540 032737 040000 177570 5$: BIT #SW14,@#SWR ;LOOP ON TEST?
002546 001271 BNE TST1 ;LOOP TO TST1
450
451
452

```

```

454
455
456
457
458
459 002550 005267 176344
460
    002554
    002554 104401 002562
    002560 000402
    002566
    002566 104401 004234
461 002572 012706 001100
462 002576 012767 000357 175172
463 002604 012703 002620
464 002610 012777 000001 176600
465 002616 000777
466 002620 010600
467 002622 012706 001100
468 002626 022700 001074
469 002632 001414
470 002634 012767 001074 005074
471 002642 010067 005072
472 002646 012706 000500
    002652 104000
    002652 104000
    002714 104000
    002714 104000
    002716 007736
    002720 007744
    002722 000000
    002724 000000
    002726 013700 001076
    002732 022700 000341
    002736 001414
    002740 012767 000341 005036
    002740 012706 000500
    002742 007742
    002744 007744
    002746 000000
    002748 000000
    002750 013700 001074
    002752 022700 002616
    002754 001414
    002756 012767 002616 004764
  
```

```

:TEST ROUTINE TO CHECK RE-START CAPABILITY
:USING THE BR. INSTRUCTION
:OPERATOR MUST SET HALT SWITCH TO ENABLE POSITION
:
      INC      $TESTN
:*****
:*TEST 2      POWER FAIL WITH BR. INSTRUCTION
:*****
TST2:
      TYPE     ,65$           ;;TYPE ASCIZ STRING
      BR       64$           ;;GET OVER THE ASCIZ
;;65$: .ASCIZ *2 *
64$:
      TYPE , $NULL
      MOV      #STACK,SP      ;SET UP STACK
      MOV      #357,PS        ;SET UP CONDITION CODES
      MOV      #2$,R3         ;SET POWER UP RETURN
      MOV      #1,@PFail     ;SET UP TO START POWER FAIL
1$: BR
2$: MOV      SP,R0           ;WAIT FOR POWER FAIL
      MOV      #STACK,SP     ;GET SP
      CMP      #STACK-4,R0   ;SET STACK
      BEQ      3$           ;CHECK STACK POINTER
      MOV      #1074,EXP.SP1 ;SKIP IF OK
      MOV      R0,REC.SP1
      MOV      #STACK1,SP    ;SETUP RESERVED STACK POINTER
                               ;FOR ERROR PRINTOUT
      ERROR
                               ;ERROR
                               ;-----
                               ;STACK POINTER
                               ;PRINT RECEIVED SP
                               ;PRINT EXPECTED SP
3$: MOV      @#STACK-2,R0    ;GET OLD PS
      CMP      #341,R0       ;CHECK OLD PS
      BEQ      4$           ;SKIP IF OK
      MOV      #341,EXP.PSW
      MOV      R0,REC.PSW
      MOV      #STACK1,SP    ;SETUP RESERVED STACK POINTER
                               ;FOR ERROR PRINTOUT
      ERROR
                               ;ERROR
                               ;-----
                               ;OLD PS IS WRONG
                               ;PRINT EXPECTED OLD PS
                               ;PRINT RECEIVED OLD PS
4$: MOV      @#STACK-4,R0    ;GET RETURN ADDRESS
      CMP      #1$,R0
      BEQ      5$           ;SKIP IF OK
      MOV      #1$,EXP.ADD
  
```

```

493 002746 010067 004762      MOV      R0,REC.ADD
494 002752 012706 000500      MOV      #STACK1,SP          ;SETUP RESERVED STACK POINTER
                                ;FOR ERROR PRINTOUT
                                002756 104000      ERROR
                                ;ERROR
                                ;-----
                                ;ADDRESS ON STACK IS WRONG
                                ;PRINT EXPECTED ADDRESS
                                ;PRINT RECEIVE ADDRESS
495                                ;ERROR
496 002760 007732      EXP.ADD
497 002762 007734      REC.ADD
498 002764 000000      0
499 002766 000000      HALT
500 002770                                5$:
002770 032737 040000 177570      BIT      #SW14,@#SWR        ;LOOP ON TEST?
002776 001266                                BNE     TST2                ;LOOP TO TST2
501
502                                ;TEST ROUTINE TO CHECK RESTART CAPABILITY
503                                ;USING THE EMULATOR TRAP FOR A WAIT
504                                ;OPERATOR MUST SET HALT SWITCH TO ENABLE POSITION
505                                ;
506 003000 005267 176114      INC      $TESTN
507                                ;*****
                                ;*TEST 3      POWER FAIL WITH EMT TRAP
                                ;*****
003004                                TST3:
003004 104401 003012      TYPE    ,65$                ;;TYPE ASCIZ STRING
003010 000402      BR      64$                ;;GET OVER THE ASCIZ
                                ;;65$: .ASCIZ *3 *
64$:
003016                                TYPE,$ENULL
508 003022 005037 177776      CLR     @#PS                ;SET KERNAL MODE
509 003026 012706 001100      MOV     #STACK,SP          ;SET UP STACK
510 003032 012703 003064      MOV     #2$,R3             ;SET POWER UP RETURN
511 003036 012767 001422 174764      MOV     #LRTI,EMTVEC       ;SET UP RETURN FROM EMT TRAP VECTOR
512 003044 012767 000005 174760      MOV     #5,EMTVEC+2       ;SET PSW ON POWER FAIL
513 003052 012777 000001 176336      MOV     #1,@PFAIL         ;SET UP TO START POWER FAIL
514 003060 104000                                3$: EMT                        ;WAIT FOR POWER FAIL
515 003062 000776      BR      3$
516 003064 010600                                2$: MOV     SP,R0                ;GET SP
517 003066 012706 001100      MOV     #STACK,SP
518 003072 012767 006616 174730      MOV     #ERROR,30
519 003100 022700 001074      CMP     #STACK-4,R0        ;RESET EMT VECTOR
520 003104 001461                                BEQ     6$                  ;WAS STACK PUSHED ONLY TWICE
521 003106 022700 001070      CMP     #STACK-10,R0      ;WAS STACK PUSHED 4 TIMES
522 003112 001414                                BEQ     4$
523 003114 012767 001070 004614      MOV     #1070,EXP.SP1
524 003122 010067 004612      MOV     R0,REC.SP1
525 003126 012706 000500      MOV     #STACK1,SP        ;SETUP RESERVED STACK POINTER
                                ;FOR ERROR PRINTOUT
                                003132 104000      ERROR
                                ;ERROR
                                ;-----
                                ;WAS STACK PUSHED 4 TIMES
                                ;PRINT EXPECTED STACK POINTER
                                ;PRINT RECEIVED STACK POINTER
526                                ;ERROR
527 003134 007736      EXP.SP1
528 003136 007740      REC.SP1
529 003140 000000      0

```

```

530 003142 000000          HALT
531 003144 013700 001070    4$:  MOV    @#STACK-10,R0
532 003150 022700 001422    CMP    #LRTI,R0          ;DOES STACK CONTAIN CORRECT INFO
533 003154 001414          BEQ    5$
534 003156 012767 001422 004546  MOV    #LRTI,EXP.ADD
535 003164 010067 004544    MOV    R0,REC.ADD
536 003170 012706 000500    MOV    #STACK1,SP      ;SETUP RESERVED STACK POINTER
                                ;FOR ERROR PRINTOUT
                                ;ERROR
                                ;-----
537                                ;IS STACK PUSHED MORE THAN 2 OR LESS THAN 4
538 003176 007732          EXP.ADD          ;PRINT EXPECTED ADDRESS
539 003200 007734          REC.ADD          ;PRINT RECEIVED ADDRESS
540 003202 000000          0
541 003204 000000          HALT
542 003206 013700 001072    5$:  MOV    @#STACK-6,R0
543 003212 022700 000005    CMP    #5,R0          ;DOES STACK CONTAIN CORRECT INFO
544 003216 001414          BEQ    6$
545 003220 012767 000005 004514  MOV    #5,EXP.PSW
546 003226 000000 007744    RO,REC.PSW
547 003232 012706 000500    MOV    #STACK1,SP      ;SETUP RESERVED STACK POINTER
                                ;FOR ERROR PRINTOUT
                                ;ERROR
                                ;-----
548                                ;STACK CONTIAN WRONG PSW
549 003240 007742          EXP.PSW          ;PRINT EXPECTED PSW
550 003242 007744          REC.PSW          ;PRINT RECEIVED PSW
551 003244 000000          0
552 003246 000000          HALT
553 003250          6$:  BIT    #SW14,@#SWR      ;LOOP ON TEST?
554 003250 032737 040000 177570  BNE    TST3          ;LOOP TO TST3
555 003256 001252
556
557 003260 005267 175634          INC    $TESTN
558                                ;*****
                                ;*TEST 4          POWER FAIL WITH ODD ADDRESS
                                ;*****
                                TST4:
003264          TYPE    ,65$          ;;TYPE ASCIZ STRING
003264 104401 003272          BR     64$          ;;GET OVER THE ASCIZ
003270 000402
;;65$: .ASCIZ  *4 *
64$:
559 003276 104401 004234          TYPE , $ENULL
560 003302 005037 177776          CLR    @#PS          ;SET KERNAL MODE
561 003306 012737 003326 000004  MOV    #3$,@#ERRVEC  ;SET TRAP VECTOR
562 003314 012703 003346          MOV    #1$,R3        ;SET RETURN ADDRESS FOR POWER FAIL
563 003320 012777 000001 176070  MOV    #1,@#PFAIL    ;SET UP TO START POWER FAIL
564 003326 012706 001100    3$:  MOV    #STACK,SP      ;RESET STACK
565 003332 005737 000003    TST    @#3          ;CAUSE ODD ADDRESS TRAP
565 003336 012706 000500    MOV    #STACK1,SP    ;SETUP RESERVED STACK POINTER
                                ;FOR ERROR PRINTOUT

```

```

003342 104000          ERROR

                                :ERROR
                                :-----
566 003344 000000          HALT                                ;ODD ADDRESS TRAP FAILED TO OCCUR
567 003346 012737 000006 000004 1$: MOV #6,@ERRVEC          ;RESET 4
568 003354 032737 040000 177570 BIT #SW14,@SWR          ;LOOP ON TEST?
003362 001340          BNE TST4                                ;LOOP TO TST4
569
570
571 003364 005267 175530          INC $TESTN
572
::*****
:*TEST 5          POWER FAIL WITH TIME OUT (KERNAL)
::*****
TST5:
003370          TYPE ,65$          ;;TYPE ASCIZ STRING
003370 104401 003376          BR 64$          ;;GET OVER THE ASCIZ
003374 000402
::65$: .ASCIZ *5 *
64$:
003402          TYPE,$ENULL
573 003406 012737 003426 000004 MOV #3$,@ERRVEC          ;SET TRAP VECTOR
574 003414 012703 003452 MOV #1$,R3          ;SET UP RETURN ADDRESS FOR POWER FAIL
575 003420 012777 000001 175770 MOV #1,@PFAIL          ;SET UP TO START POWER FAIL
576 003426 012706 001100 3$: MOV #STACK,SP          ;SET STACK
577 003432 005037 177776 CLR @MPS          ;SET KERNAL MODE
578 003436 010037 160000 MOV R0,@#160000          ;CAUSE A TIMEOUT
579 003442 012706 000500 MOV #STACK1,SP          ;SETUP RESERVED STACK POINTER
                                ;FOR ERROR PRINTOUT
003446 104000          ERROR

                                :ERROR
                                :-----
580 003450 000000          HALT                                ;TIME OUT FAILED TO OCCUR
581 003452 012706 001100 1$: MOV #STACK,SP          ;SET STACK
582 003456 012737 000006 000004 MOV #6,@ERRVEC          ;RESET 4
583 003464 032737 040000 177570 BIT #SW14,@SWR          ;LOOP ON TEST?
003472 001336          BNE TST5                                ;LOOP TO TST5

```



```
585 003474 005267 175420          INC    $TESTN
586                                *****
                                ;*TEST 6      POWER FAIL IN THE STACK OVERFLOW (KERNAL)
                                *****
                                TST6:
                                TYPE    ,65$          ;;TYPE ASCIZ STRING
                                BR      64$          ;;GET OVER THE ASCIZ
                                ;;65$: .ASCIZ *6 *
                                64$:
                                TYPE,$ENULL
                                CLR     @MPS        ;SET KERNAL MODE
                                CLR     FLAG        ;CLEAR THE FLAG
                                MOV     #2$,@ERRVEC  ;SET SICK TPAP ADDRESS
                                MOV     #400,SP     ;SET STACK TO YELLOW ZONE
                                MOV     #1$,R3     ;SET RETURN ADDRESS FOR POWER FAIL
                                MOV     #1,@PFAIL   ;SET UP TO START POWER FAIL
                                WAIT    ;WAIT FOR THE POWER FAIL
                                MOV     #STACK1,SP  ;SETUP RESERVED STACK POINTER
                                                ;FOR ERROR PRINTOUT
                                ERROR
                                                ;ERROR
                                                ;-----
595 003562 000000          1$: HALT
596 003564 000407          BR      4$          ;SKIP SP CHECK
597 003566 012737 000006 000004 2$: MOV     #6,@ERRVEC ;RESET 4
598 003574 012703 003604          5$: MOV     #4$,R3   ;SET RETURN
599 003600 000002          RTI
600 003602 000000          HALT          ;GO TO THE POWER FAIL ROUTINE
601 003604          4$:
                                BIT     #SW14,@SWR  ;LOOP ON TEST?
                                BNE    TST6        ;LOOP TO TST6
602
603
604 003614 005267 175300          INC    $TESTN
605                                *****
                                ;*TEST 7      POWER FAIL WITH RESETS
                                *****
                                TST7:
                                TYPE    ,65$          ;;TYPE ASCIZ STRING
                                BR      64$          ;;GET OVER THE ASCIZ
                                ;;65$: .ASCIZ *7 *
                                64$:
                                TYPE,$ENULL
                                CLR     @MPS        ;SET KERNAL MODE
                                MOV     #1$,R3     ;SET RETURN ADDRESS
                                MOV     #STACK,SP   ;RESET STACK
                                MOV     #1,@PFAIL   ;SET UP TO START POWER FAIL
                                3$: RESET        ;RESETS
                                RESET        ;TO WAIT
                                RESET        ;IN
                                BR      3$        ;LOOP
                                MOV     #STACK1,SP  ;SETUP RESERVED STACK POINTER
                                                ;FOR ERROR PRINTOUT
                                ERROR
                                                ;ERROR
```


620
621
622
623
624
625

003714 005267 175200

INC \$TESTN

*TEST 10 MEMORY MANAGEMENT ABORT TEST

003720
003720 104401 003726
003724 000402

TST10:

TYPE ,65\$;:TYPE ASCIZ STRING
BR 64\$;:GET OVER THE ASCIZ
65\$: .ASCIZ *10 *
64\$:
TYPE , \$ENULL
CLR @#PS ;SET KERNAL MODE
MOV #4\$, @#ERRVEC ;SET FOR TIMEOUT
JSR PC, MAP ;MAP THE WORLD
MOV #3\$, @#MMVEC ;SET MEMORY MANAGEMENT VECTOR
MOV #1\$, R3 ;LOAD PF RETURN
MOV #1, @#PFail ;SET UP TO START POWER FAIL
INC @#MMRO ;TURN MEMORY MANAGEMENT ON
3\$: MOV #STACK, SP ;ZAP STACK
INC @#140000 ;ACCESS VIOLATION
MOV #STACK1, SP ;SETUP RESERVED STACK POINTER
;FOR ERROR PRINTOUT

003732
003732 104401 004234
626 003736 005037 177776
627 003742 012737 004016 000004
628 003750 004767 001134
629 003754 012737 004000 000250
630 003762 012703 004020
631 003766 012777 000001 175422
632 003774 005237 177572
633 004000 012706 001100
634 004004 005237 140000
635 004010 012706 000500

004014 104000

ERROR

;FROR

;NO VIOLATION OR TRAP TO 4

636
637 004016 000000
638

4\$: HALT

639 004020 005037 177572
640 004024 012706 001100
641 004030 012737 000006 000004
642 004036 032737 040000 177570
004044 001325

1\$: CLR @#MMRO ;TURN OFF MEMORY MANAGEMENT
2\$: MOV #STACK, SP ;MAKE A NEW STACK
MOV #6, @#ERRVEC ;RESET 4
BIT #SW14, @#SWR ;LOOP ON TEST?
BNE TST10 ;LOOP TO TST10

643
644 004046 005267 175046
645

INC \$TESTN

*TEST 11 MEMORY VOLATILITY TEST

004052 000240
646 004054 005037 177776
647 004060 004767 000170
648 004064 012703 004122
649 004070 104401 004076
004074 000402

TST11:

NOP
CLR @#PS ;SET KERNAL MODE
4\$: JSR PC, LOAD ;LOAD ALL MEMORY WITH 52525
MOV #1\$, R3 ;POWER FAIL RETURN ADDRESS
TYPE ,65\$;:TYPE ASCIZ STRING
BR 64\$;:GET OVER THE ASCIZ
65\$: .ASCIZ *11 *
64\$:
TYPE , \$ENULL
2\$: JSR PC, CHECK ;CHECK FOR THE 52525
MOV #1, @#PFail ;SET UP TO START POWER FAIL
BR 2\$;LOOP FOR EVER OR POWER FAIL
1\$: MOV #STACK, SP ;ZAP THE STACK
JSR PC, CHECK ;CHECK ALL MEMORY
BIT #SW14, @#SWR ;LOOP ON TEST?

004102
004102 104401 004234
650 004106 004767 000246
651 004112 012777 000001 175276
652 004120 000772
653 004122 012706 001100
654 004126 004767 000226
655 004132 032737 040000 177570

JYKACAO 11/44 POWER FAIL
T11 MEMORY VOLATILITY TEST

MACRO M1113 10-DEC-79 10:05 PAGE 10-1

B 3

SEQ 0027

004140 001344

BNE TST11

:LOOP TO TST11

657

```

.SBTTL END OF PASS ROUTINE
:*****
:*INCREMENT THE PASS NUMBER ($PASS)
:*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
:*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
:*IF THERES A MONITOR GO TO IT
:*IF THERE ISN'T JUMP TO FT
$EOP:
004142      NOP
004142 000240      INC      $PASS      ;; INCREMENT THE PASS NUMBER
004144 005267 174752      BIC      #100000,$PASS  ;; DON'T ALLOW A NEG. NUMBER
004150 042767 100000 174744      DEC      (PC)+      ;; LOOP?
004156 005327      $EOPCT: .WORD 1
004160 000001      BGT      $DOAGN      ;; YES
004162 003022      MOV      (PC)+,@(PC)+  ;; RESTORE COUNTER
004164 012737      $ENDCT: .WORD 1
004166 000001      $EOPCT -
004170 004160      TYPE      ,SENDMG      ;; TYPE 'END PASS #'
004172 104401 004237      MOV      $PASS,-(SP)  ;; SAVE $PASS FOR TYPEOUT
004176 016746 174720      TYPDS      ;; GO TYPE--DECIMAL ASCII WITH SIGN
004202 104405      TYPE      ,SENULL      ;; TYPE A NULL CHARACTER
004204 104401 004234      $GET42: MOV      @#42,R0  ;; GET MONITOR ADDRESS
004210 013700 000042      BEQ      $DOAGN      ;; BRANCH IF NO MONITOR
004214 001405      RESET      ;; CLEAR THE WORLD
004216 000005      $ENDAD: JSR      PC,(R0)  ;; GO TO MONITOR
004220 004710      NOP      ;; SAVE ROOM
004222 000240      NOP      ;; FOR
004224 000240      NOP      ;; ACT11
004226 000240      $DOAGN:
004230      JMP      @(PC)+      ;; RETURN
004230 000137      $RTNAD: .WORD  FT
004232 002264      $ENULL: .BYTE  -1,-1,0  ;; NULL CHARACTER STRING
004234      377      377      000      $ENDMG: .ASCIZ  <15><12>/END PASS #/
004237      015      012      105
004242      116      104      040
004245      120      101      123
004250      123      040      043
004253      000

```

```

659
660
661 004254 004767 000630
662 004260 016704 000620
663 004264 016705 000616
664 004270 012737 004320 000250
665 004276 012700 010220
666 004302 062700 120000
667 004306 012737 000001 177572
668 004314 010520
669 004316 000776
670 004320 005304
671 004322 100411
672 004324 012700 120000
673 004330 062737 000200 172352
674 004336 012737 000001 177572
675 004344 000002
676 004346 005037 177572
677 004352 062706 000004
678 004356 000207

.SBTTL LOAD MEMORY WITH DATA PATTERN
LOAD: JSR PC,MAP ;SET UP MEMORY MANAGEMENT REGISTERS
      MOV LIMIT,R4 ;GET BANK COUNT
      MOV DATA,R5 ;GET THE DATA PATTERN
      MOV #2$,@MMVEC ;SET UP FOR MEMORY MANAGEMENT ABORTS
      MOV #END,R0 ;FORM ADDRESS OF WHERE TO START WRITING THE DATA
      ADD #120000,R0 ;USE KIPAR5 FOR ADDRESSING
      MOV #1,@MMRO ;TURN ON MEMORY MANAGEMENT
1$:   MOV R5,(R0)+ ;WRITE THE DATA
      BR 1$
2$:   DEC R4 ;LAST BANK WRITTEN?
      BMI 3$ ;BRANCH IF YES
      MOV #120000,R0 ;RESET ADDRESSING REGISTER
      ADD #200,@KIPAR5 ;SIZE NEXT 4K BANK
      MOV #1,@MMRO ;CLEAR MEMORY MANAGEMENT ERRORS
      RTI ;RETURN TO CONTINUE WRITING
3$:   CLR @MMRO ;TURN OFF MEMORY MANAGEMENT
      ADD #4,SP ;CLEAN UP THE STACK
      RTS PC ;RETURN TO CALLER
  
```

```

680          .SBTTL  CHECK MEMORY FOR CORRECT DATA PATTERN
681
682 004360 004767 000524      CHECK: JSR    PC,MAP          ;SET UP MEMORY MANAGEMENT REGISTERS
683 004364 016704 000514      MOV    LIMIT,R4          ;GET BANK COUNT
684 004370 016705 000512      MOV    DATA,R5         ;GET DATA PATTERN
685 004374 012737 004514 000250  MOV    #2$,@MMVEC       ;SET UP FOR MEMORY MANAGEMENT ABORTS
686 004402 012700 010220      MOV    #END,R0          ;ADDRESS OF WHERE DATA PATTERN STARTS
687 004406 062700 120000      ADD    #120000,R0       ;USE KIPAR5 FOR ADDRESSING
688 004412 012737 000001 177572  MOV    #1,@MMRO         ;TURN ON MEM. MANAGEMENT
689 004420 012001 1$:      MOV    (R0)+,R1         ;READ THE DATA
690 004422 020501      CMP    R5,R1           ;IS IT GOOD?
691 004424 001775      BEQ    1$              ;BRANCH IF YES
692 004426 010067 003314      MOV    R0,VIR.ADD      ;
693 004432 162767 000002 003306  SUB    #2,VIR.ADD      ;
694 004440 042767 160000 003300  BIC    #160000,VIR.ADD ;
695 004446 013767 172352 003274  MOV    @KIPAR5,PAR.OFF ;
696 004454 010567 003246      MOV    R5,EXPDAT      ;
697 004460 010167 003244      MOV    R1,RECDAT     ;
698 004464 012706 000500      MOV    #STACK1,SP    ;SETUP RESERVED STACK POINTER
                                ;FOR ERROR PRINTOUT
                                ;
                                ;ERROR
                                ;-----
004470 104000      ERROR
                                ;
                                ;
699 004472 007746      VIR.ADD                ;PRINT VIRTUAL ADDRESS
700 004474 007750      PAR.OFF               ;PRINT PDR OFFSET
701 004476 007726      EXPDAT                ;PRINT EXPECTED DATA
702 004500 007730      RECDAT                ;PRINT RECEIVED DATA
703 004502 000000      0
704 004504 000000      HALT
705 004506 010560 177776      MOV    R5,-2(R0)      ;WRITE GOOD DATA
706 004512 000742      BR    1$              ;GO READ THE NEXT WORD
707 004514 005304 2$:      DEC    R4              ;LAST BANK COMPARED?
708 004516 100411      BMI    3$             ;BRANCH IF YES
709 004520 012700 120000      MOV    #120000,R0     ;RESET ADDRESSING REGISTER
710 004524 062737 000200 172352  ADD    #200,@KIPAR5   ;SIZE NEXT 4K BANK
711 004532 012737 000001 177572  MOV    #1,@MMRO       ;CLEAR MEMORY MANAGEMENT ERRORS
712 004540 000002      RTI                   ;CONTINUE TESTING
713 004542 005037 177572 3$:      CLR    @MMRO          ;TURN OFF MEM. MANAGEMENT
714 004546 062706 000004      ADD    #4,SP          ;CLEAN UP THE STACK
715 004552 000207      RTS    PC             ;RETURN FOR CALL
    
```

```

717          .SBTTL  PARITY ERROR HANDLER
718
719 004554 005327  PARERR: DEC      (PC)+      ;FIRST TIME IN?
720 004556 000001  PARFLG: .WORD    1
721 004560 002002          BGE      2$      ;BRANCH IF YES
722 004562 000000          1$:  HALT          ;NO--ANOTHER PARITY ERROR OCCURRED WHILE
723 004564 000776          BR       1$      ;PROCESSING THE FIRST ONE.
724 004566 032767 000077 173146 2$:  BIT      #77,HIADRS ;MSB'S OF PARITY ERROR ADDRESS=0?
725 004574 001006          BNE      4$      ;BRANCH IF NO
726 004576 023727 177740 010220  CMP     @LOADRS,#END ;LSB'S ABOVE THE PROGRAM?
727 004604 101002          BHI      4$      ;BRANCH IF YES
728 004606 000000          3$:  HALT          ;PARITY ERROR IS IN THE PROGRAM
729 004610 000776          BR       3$
730 004612 000000          4$:  HALT          ;DATA PARITY ERROR OCCURRED
731 004614 005737 177744          TST     @MEMERR    ;DID AN ABORT OCCUR?
732 004620 100405          BMI      5$      ;BRANCH IF YES
733 004622 162700 000002          SUB     #2,R0      ;BACKUP BY ONE WORD
734 004626 012705 000002          MOV     #BIT01,R5
735 004632 074500          XOR     R5,R0
736 004634 010320          5$:  MOV     R3,(R0)+  ;REWRITE THE BAD DATA
737 004636 013737 177744 177744  MOV     @MEMERR,@MEMERR ;CLEAR ERROR INDICATORS
738 004644 012767 000001 1777C4  MOV     #1,PARFLG  ;INITIALIZE PARITY ERROR FLAG
739 004652 000002          RTI          ;CONTINUE TESTING
  
```



```

741
742          .SBTTL  POWER FAIL ROUTINE
743
744 004654 012767 177777 000220 POWDWN: MOV      #-1,FLAG      ;FIRST INSTRUCTION FLAG
745 004662 005067 000214          CLR      FLAG          ;NOW CLEAR IT
746 004666 012777 005042 000174      MOV      #ILLUP,@UVEC    ;IF TOO FAST
747 004674 011667 000166          MOV      (SP),ERRAD      ;SET THE ERROR ADDRESS
748 004700 022706 000400          CMP      #400,SP        ;YELLOW OR RED?
749 004704 100402          BMI      1$            ;NO
750 004706 012706 001100          MOV      #STACK,SP     ;SET EMERGENCY STACK
751 004712 010046          1$:  MOV      R0,-(6)      ;PUT
752 004714 010146          MOV      R1,-(6)      ;THE
753 004716 010246          MOV      R2,-(6)      ;REGISTERS
754 004720 010346          MOV      R3,-(6)      ;ON
755 004722 010446          MOV      R4,-(6)      ;THE
756 004724 010546          MCV      R5,-(6)      ;STACK
757 004726 010667 000146          MOV      SP,SAV6      ;SAVE THE STACK POINTER
758 004732 004737 172240          JSR      PC,@#SIPARO   ;SET TIME FACTOR
759 004736 012777 004750 000124      MOV      #POWUP,@UVEC  ;RESET THE UP VECTOR
760 004744 000000          HALT
761 004746 000240          NOP
762
763 004750 012777 005054 000116 POWUP:  MOV      #ILLDWN,@DVEC  ;SET TOO FAST DOWN VECTOR
764 004756 016706 000116          MOV      SAV6,SP      ;RESET SP
765 004762 012737 012700 172240      MOV      #12700,@#SIPARO ;LOAD PAR0 & 1 WITH MOV #1777,R0
766 004770 012737 001777 172242      MOV      #1777,@#SIPAR1
767 004776 012737 077001 172244      MOV      #77001,@#SIPAR2 ;LOAD PAR2 WITH SOB R0..
768 005004 012737 000207 172246      MOV      #207,@#SIPAR3  ;LOAD PAR3 WITH RTS PC
769 005012 004737 172240          JSR      PC,@#SIPARO   ;SET TIME FACTOR
770 005016 012605          MOV      (6)+,R5      ;TAKE
771 005020 012604          MOV      (6)+,R4      ;THE
772 005022 012603          MOV      (6)+,R3      ;REGISTERS
773 005024 012602          MOV      (6)+,R2      ;FROM
774 005026 012601          MOV      (6)+,R1      ;THE
775 005030 012600          MOV      (6)+,R0      ;STACK
776 005032 012777 004654 000034      MOV      #POWDWN,@DVEC ;RESET THE DOWN VECTOR
777 005040 000113          JMP      (R3)          ;JUMP INDIRECT TO R3
778
779 005042 104401          ILLUP:  TYPE
780 005044 010105          UPF
781 005046 000000          HALT
782 005050 000167 173124          JMP      200
783 005054 104401          ILLDWN: TYPE
784 005056 010155          DOWN
785 005060 000000          HALT
786 005062 000167 173112          JMP      200
787
788
789 005066 000000          ERRAD:  0
790
791 005070 000024 000026          UVEC:   24,26
792 005074 000024 000026          DVEC:   24,26
793 005100 000000          SAV6:   0
794 005102 000000          FLAG:   0
795 005104 000000          LIMIT:  0
796 005106 052525          DATA: 52525
;RETURN ADDRESS FROM POWER FAIL
;UP ADDRESS PAIR
;DOWN ADDRESS PAIR
;SOME PLACE TO PUT THE SP
;1 INSTRUCTION DOWN FLAG
;TOP OF MEMORY
;WHAT IS TO BE WRITTEN INTO MEMORY

```

798
 799
 800 005110 012737 000000 172340
 801 005116 012737 077406 172300
 802 005124 012737 000000 172352
 803 005132 012737 077406 172312
 804 005140 012737 000200 172354
 805 005146 012737 000000 172314
 806 005154 012737 177600 172356
 807 005162 012737 077406 172316
 808 005170 012737 000020 172516
 809 005176 012737 005206 000250
 810 005204 000207
 811 005206 000000
 812 005210 000776
 813
 814

.SBTTL SETUP MEMORY MANAGMENT REGISTERS

```
MAP:  MOV    #0,@#KIPAR0      ;SETUP PAR0 FOR 1ST 4K
      MOV    #77406,@#KIPDR0 ;4K, R/W, EXPAND UP
      MOV    #0,@#KIPAR5      ;SET UP PAR5 FOR 1ST 4K
      MOV    #77406,@#KIPDR5 ;4K, R/W, ED=UP
      MOV    #200,@#KIPAR6     ;SET UP PAR6 FOR 2ND 4K
      MOV    #0,@#KIPDR6      ;ABORT ALL REFERENCES
      MOV    #177600,@#KIPAR7 ;SET UP PAR7 FOR I/O PAGE
      MOV    #77406,@#KIPDR7 ;4K, R/W, ED=UP
      MOV    #BIT04,@#MMR3     ;SET UP FOR 22-BIT MAPPING
      MOV    #MMERR,@#MMVEC   ;SET UP MEMORY MANAGEMENT VECTOR
      RTS    PC                ;RETURN FROM CALL
MMERR: HALT                   ;MEMORY MANAGEMENT ERROR
      BR     MMERR
```

.SBTTL ROUTINE TO SIZE MEMORY
 .SBTTL ROUTINE TO SIZE MEMORY

```
*****
*CALL:
*   JSR    PC,$SIZE
*   RETURN
*$LSTAD WILL CONTAIN:
*   WITH KT11 OPTION      -- LAST VIRTUAL ADDRESS OF THE LAST BANK
*   WITHOUT KT11 OPTION  -- LAST ABSOLUTE ADDRESS OF AVAILABLE MEMORY
*$LSTBK WILL CONTAIN THE LAST BANK AS A SAF
*$KT11 IS THE MEMORY MANAGEMENT KEY
*$BIT07 = 0 DON'T USE MEMORY MANAGEMENT
*   MUST BE SETUP BEFORE THE CALL
*$BIT15 = 0 DON'T HAVE MEMORY MANAGEMENT OPTION
*   DETERMINED BY ROUTINE
```

005212 010046
 005214 010146
 005216 010246
 005220 010346
 005222 013746 000004
 005226 013746 000006
 005232 010600

 005234 104400
 005236 012637 000006
 005242 012701 003776
 005246 105727
 005250 000200
 005252 100062
 005254 012737 005412 000004
 005262 005737 177572
 005266 052767 100000 177754
 005274 005046
 005276 012702 172340
 005302 012703 000010
 005306 012762 077406 177740 1\$:
 005314 011622
 005316 062716 000200
 005322 077307
 005324 012742 177600
 005330 005042
 005332 012737 005350 000004

```
$SIZE: MOV    R0,-(SP)      ;;SAVE R0 ON THE STACK
      MOV    R1,-(SP)      ;;SAVE R1 ON THE STACK
      MOV    R2,-(SP)      ;;SAVE R2 ON THE STACK
      MOV    R3,-(SP)      ;;SAVE R3 ON THE STACK
      MOV    @#ERRVEC,-(SP) ;;SAVE PRESENT ERROR VECTOR PS & PC
      MOV    @#ERRVEC+2,-(SP)
      MOV    SP,R0         ;;SAVE THE STACK POINTER
      ;;SET THE ERRVEC PS TO THE PRESENT PS
      TRAP
      MOV    (SP)+,@#ERRVEC+2 ;;SAVE THE PSW IN @#ERRVEC+2
      MOV    #3776,R1      ;;SETUP ADDRESS
      TSTB   (PC)+        ;;USE MEMORY MANAGEMENT?
      SKT11: .WORD 200     ;;SET TO USE MEMORY MANAGEMENT
      BPL    $SCORE       ;;BR IF NO
      MOV    #SKTNEX,@#ERRVEC ;;SET FOR TIMEOUT
      TST    @#SRO        ;;KT11 ARE YOU THERE?
      BIS    #100000,$KT11 ;;YES--SET KT11 KEY
      CLR    -(SP)        ;;INITIALIZE FOR 'PAR' LOADING
      MOV    #KIPAR0,R2   ;;ADDRESS OF FIRST 'PAR'
      MOV    #^D8,R3      ;;LOAD EIGHT 'PAR.'S' AND EIGHT 'PDR.'S'
      MOV    #77406,-40(R2) ;;PDR = 4K, UP, READ/WRITE
      MOV    (SP),(R2)+    ;;LOAD 'PAR'
      ADD    #200,(SP)     ;;UPDATE FOR NEXT 'PAR'
      SOB    R3,1$        ;;LOOP UNTIL ALL EIGHT ARE LOADED
      MOV    #177600,-(R2) ;;SETUP KIPAR7 FOR I/O
      CLR    -(R2)        ;;SETUP KIPAR6 FOR TESTING
      MOV    #2$,@#ERRVEC ;;CATCH TIMEOUT IF NO SR3
```

```

005340 012737 000020 172516      MOV    #20,@#SR3      ;;ENABLE 22 BIT MODE
005346 000401                BR     3$             ;;THIS PDP-11 HAS A SR3 REGISTER
005350 022626                2$:  CMP    (SP)+,(SP)+  ;;CLEAN OFF THE STACK--NO SR3
005352 005237 177572                3$:  INC    @#SRO        ;;TURN ON MEMORY MANAGEMENT
005356 012737 005402 000004      MOV    #SKTOUT,@#ERRVEC ;;SET FOR TIME OUT
005364 005737 143776                4$:  TST    @#143776     ;;TRAP ON NON-EX-MEM
005370 062712 000040                ADD    #40,(R2)      ;;MAKE A 1K STEP
005374 023712 172356                CMP    @#KIPAR7,(R2) ;;LAST ONE?
005400 101371                BHI    4$           ;;NO--TRY IT
005402 011202                SKTOUT: MOV   (R2),R2  ;;GET LAST BANK+1
005404 005037 177572                CLR    @#SRO        ;;TURN OFF MEMORY MANAGEMENT
005410 000421                BR     $SIZEX
005412 042767 100000 177630  $KTNEX: BIC   #100000,$KT11 ;;KT11 NON-EXISTENT
005420 012737 005450 000004  $SCORE: MOV   #SCORE,@#ERRVEC ;;SET FOR TIMEOUT
005426 005002                CLR    R2           ;;SET UP BANK
005430 062701 004000                1$:  ADD    #4000,R1    ;;INCREMENT BY 1K
005434 062702 000040                ADD    #40,R2      ;;1K STEP
005440 005711                TST    (R1)        ;;TRAP ON TIME OUT
005442 022701 177776                CMP    #177776,R1  ;;LAST ONE
005446 001370                BNE    1$         ;;NO--TRY AGAIN
005450 162701 004000                $CROUT: SUB  #4000,R1
005454 162702 000040                $SIZEX: SUB  #40,R2  ;;DROP BACK
005460 010006                MOV    R0,SP       ;;RESTORE THE STACK
005462 012637 000006                MOV    (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
005466 012637 000004                MOV    (SP)+,@#ERRVEC
005472 010167 000016                MOV    R1,$LSTAD   ;;LAST ADDRESS
005476 010267 000014                MOV    R2,$LSTBK   ;;LAST BANK
005502 012603                MOV    (SP)+,R3    ;;RESTORE R3
005504 012602                MOV    (SP)+,R2    ;;RESTORE R2
005506 012601                MOV    (SP)+,R1    ;;RESTORE R1
005510 012600                MOV    (SP)+,R0    ;;RESTORE R0
005512 000207                RTS    PC
005514 000000                $LSTAD: .WORD 0     ;;CONTAINS THE LAST ADDRESS
005516 000000                $LSTBK: .WORD 0     ;;CONTAINS THE LAST BANK

```

815
816

818

```

.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
005520 017646 000000      $TYPOS: MOV     @ (SP),-(SP)      ;;PICKUP THE MODE
005524 116667 000001 000211 MOV     1(SP), $OFILL      ;;LOAD ZERO FILL SWITCH
005532 112667 000207      MOV     (SP)+, $OMODE+1    ;;NUMBER OF DIGITS TO TYPE
005536 062716 000002      ADD     #2, (SP)          ;;ADJUST RETURN ADDRESS
005542 000406      BR     $TYPON
005544 112767 000001 000171 $TYPOC: MOV     #1, $OFILL      ;;SET THE ZERO FILL SWITCH
005552 112767 000006 000165      MOV     #6, $OMODE+1      ;;SET FOR SIX(6) DIGITS
005560 112767 000005 000154 $TYPON: MOV     #5, $OCNT      ;;SET THE ITERATION COUNT
005566 010346      MOV     R3, -(SP)        ;;SAVE R3
005570 010446      MOV     R4, -(SP)        ;;SAVE R4
005572 010546      MOV     R5, -(SP)        ;;SAVE R5
005574 116704 000145      MOV     $OMODE+1, R4      ;;GET THE NUMBER OF DIGITS TO TYPE
005600 005404      NEG     R4
005602 062704 000006      ADD     #6, R4            ;;SUBTRACT IT FOR MAX. ALLOWED
005606 110467 000132      MOV     R4, $OMODE        ;;SAVE IT FOR USE
005612 116704 000125      MOV     $OFILL, R4        ;;GET THE ZERO FILL SWITCH
005616 016605 000012      MOV     12(SP), R5        ;;PICKUP THE INPUT NUMBER
005622 005003      CLR     R3                ;;CLEAR THE OUTPUT WORD
005624 006105      1$:   ROL     R5            ;;ROTATE MSB INTO 'C'
005626 000404      BR     3$                ;;GO DO MSB
005630 006105      2$:   ROL     R5            ;;FORM THIS DIGIT
005632 006105      ROL     R5
005634 006105      ROL     R5
005636 010503      MOV     R5, R3
005640 006103      3$:   ROL     R3            ;;GET LSB OF THIS DIGIT
005642 105367 000076      DECB   $OMODE            ;;TYPE THIS DIGIT?
005646 100016      BPL     7$                ;;BR IF NO
005650 042703 177770      BIC     #177770, R3       ;;GET RID OF JUNK
005654 001002      BNE     4$                ;;TEST FOR 0
005656 005704      TST     R4                ;;SUPPRESS THIS 0?
005660 001403      BEQ     5$                ;;BR IF YES
005662 005204      4$:   INC     R4            ;;DON'T SUPPRESS ANYMORE 0'S
005664 052703 000060      BIS     #'0, R3          ;;MAKE THIS DIGIT ASCII
005670 052703 000040      5$:   BIS     #' , R3      ;;MAKE ASCII IF NOT ALREADY

```

819

```

005674 110367 000040          MOVB   R3,8$          ;;SAVE FOR TYPING
005700 104401 005740          TYPE   8$           ;;GO TYPE THIS DIGIT
005704 105367 000032          7$:   DECB   $OCNT    ;;COUNT BY 1
005710 003347                BGT    2$           ;;BR IF MORE TO DO
005712 002402                BLT    6$           ;;BR IF DONE
005714 005204                INC    R4           ;;INSURE LAST DIGIT ISN'T A BLANK
005716 000744                BR     2$           ;;GO DO THE LAST DIGIT
005720 012605          6$:   MOV    (SP)+,R5    ;;RESTORE R5
005722 012604          MOV    (SP)+,R4    ;;RESTORE R4
005724 012603          MOV    (SP)+,R3    ;;RESTORE R3
005726 016666 000002 000004  MOV    2(SP),4(SP)  ;;SET THE STACK FOR RETURNING
005734 012616          MOV    (SP)+,(SP)
005736 000002          RTI                    ;;RETURN
005740 000                8$:   .BYTE  0          ;;STORAGE FOR ASCII DIGIT
005741 000                .BYTE  0          ;;TERMINATOR FOR TYPE ROUTINE
005742 000          $OCNT: .BYTE  0          ;;OCTAL DIGIT COUNTER
005743 000          $OFILL: .BYTE 0          ;;ZERO FILL SWITCH
005744 000000          $OMODE: .WORD 0         ;;NUMBER OF DIGITS TO TYPE
.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
;*****
;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
;SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
;NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
;BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
;REPLACED WITH SPACES.
;CALL:
;*   MOV    NUM,-(SP)          ;;PUT THE BINARY NUMBER ON THE STACK
;*   TYPDS          ;;GO TO THE ROUTINE
$TYPDS:
MOV    R0,-(SP)          ;;PUSH R0 ON STACK
MOV    R1,-(SP)          ;;PUSH R1 ON STACK
MOV    R2,-(SP)          ;;PUSH R2 ON STACK
MOV    R3,-(SP)          ;;PUSH R3 ON STACK
MOV    R5,-(SP)          ;;PUSH R5 ON STACK
MOV    #20200,-(SP)      ;;SET BLANK SWITCH AND SIGN
MOV    20(SP),R5        ;;GET THE INPUT NUMBER
BPL    1$               ;;BR IF INPUT IS POS.
NEG    R5               ;;MAKE THE BINARY NUMBER POS.
MOVB   #'-,1(SP)        ;;MAKE THE ASCII NUMBER NEG.
1$:   CLR    R0          ;;ZERO THE CONSTANTS INDEX
MOV    #5DBLK,R3        ;;SETUP THE OUTPUT POINTER
MOVB   #' ,(R3)+        ;;SET THE FIRST CHARACTER TO A BLANK
2$:   CLR    R2          ;;CLEAR THE BCD NUMBER
MOV    $DTBL(R0),R1     ;;GET THE CONSTANT
3$:   SUB    R1,R5        ;;FORM THIS BCD DIGIT
BLT    4$               ;;BR IF DONE
INC    R2               ;;INCREASE THE BCD DIGIT BY 1
4$:   ADD    R1,R5        ;;ADD BACK THE CONSTANT
TST    R2               ;;CHECK IF BCD DIGIT=0
BNE    5$               ;;FALL THROUGH IF 0
TSTB   (SP)             ;;STILL DOING LEADING 0'S?
BMI    7$               ;;BR IF YES
5$:   ASLB   (SP)        ;;MSD?
BCC    6$               ;;BR IF NO
006050 116663 000001 177777  MOVB   1(SP),-1(R3)  ;;YES--SET THE SIGN
006056 052702 000060          6$:   BIS    #'0,R2    ;;MAKE THE BCD DIGIT ASCII

```

```

006062 052702 000040 7$: BIS #',R2 ;:MAKE IT A SPACE IF NOT ALREADY A DIGIT
006066 110223 MOVB R2,(R3)+ ;:PUT THIS CHARACTER IN THE OUTPUT BUFFER
006070 005720 TST (R0)+ ;:JUST INCREMENTING
006072 020027 000010 CMP R0,#10 ;:CHECK THE TABLE INDEX
006076 002746 BLT 2$ ;:GO DO THE NEXT DIGIT
006100 003002 BGT 8$ ;:GO TO EXIT
006102 010502 MOV R5,R2 ;:GET THE LSD
006104 000764 BR 6$ ;:GO CHANGE TO ASCII
006106 105726 8$: TSTB (SP)+ ;:WAS THE LSD THE FIRST NON-ZERO?
006110 100003 BPL 9$ ;:BR IF NO
006112 116663 177777 177776 MOVB -1(SP),-2(R3) ;:YES--SET THE SIGN FOR TYPING
006120 105013 9$: CLRB (R3) ;:SET THE TERMINATOR
006122 012605 MOV (SP)+,R5 ;:POP STACK INTO R5
006124 012603 MOV (SP)+,R3 ;:POP STACK INTO R3
006126 012602 MOV (SP)+,R2 ;:POP STACK INTO R2
006130 012601 MOV (SP)+,R1 ;:POP STACK INTO R1
006132 012600 MOV (SP)+,R0 ;:POP STACK INTO R0
006134 104401 006162 TYPE $DBLK ;:NOW TYPE THE NUMBER
006140 016666 000002 000004 MOV 2(SP),4(SP) ;:ADJUST THE STACK
006146 012616 MOV (SP)+,(SP)
006150 000002 RTI ;:RETURN TO USER
006152 023420 $DTBL: 10000.
006154 001750 1000.
006156 000144 100.
006160 000012 10.
006162 $DBLK: .BLKW 4

```

820

```

.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
;:*

```

```

006172 105767 000335 $TYPE: TSTB $TPFLG ;:IS THERE A TERMINAL?
006176 100002 BPL 1$ ;:BR IF YES
006200 000000 HALT ;:HALT HERE IF NO TERMINAL
006202 000430 BR 3$ ;:LEAVE
006204 010046 1$: MOV R0,-(SP) ;:SAVE R0
006206 017600 000002 MOV @2(SP),R0 ;:GET ADDRESS OF ASCIZ STRING
006212 122767 000001 172714 CMPB #APTENV,$ENV ;:RUNNING IN APT MODE
006220 001011 BNE 62$ ;:NO,GO CHECK FOR APT CONSOLE
006222 132767 000100 172705 BITB #APTPOOL,$ENVM ;:SPOOL MESSAGE TO APT
006230 001405 BEQ 62$ ;:NO,GO CHECK FOR CONSOLE
006232 010067 000004 MOV R0,61$ ;:SETUP MESSAGE ADDRESS FOR APT
006236 004767 172714 JSR PC,$ATY3 ;:SPOOL MESSAGE TO APT
006242 000000 .WORD 0 ;:MESSAGE ADDRESS
006244 132767 000040 172663 61$: BITB #APTCSUP,$ENVM ;:APT CONSOLE SUPPRESSED
006252 001003 BNE 60$ ;:YES,SKIP TYPE OUT
006254 112046 2$: MOVB (R0)+,-(SP) ;:PUSH CHARACTER TO BE TYPED ONTO STACK

```

```

006256 001005      BNE      4$      ;;BR IF IT ISN'T THE TERMINATOR
006260 005726      TST      (SP)+   ;;IF TERMINATOR POP IT OFF THE STACK
006262 012600      60$:    MOV      (SP)+,R0  ;;RESTORE R0
006264 062716 000002 3$:      ADD      #2,(SP)  ;;ADJUST RETURN PC
006270 000002      RTI                     ;;RETURN
006272 122716 000011 4$:      CMPB     #HT,(SP)  ;;BRANCH IF <HT>
006276 001430      BEQ      8$
006300 122716 000200      CMPB     #CRLF,(SP)  ;;BRANCH IF NOT <CRLF>
006304 001006      BNE      5$
006306 005726      TST      (SP)+   ;;POP <CR><LF> EQUIV
006310 104401      TYPE                     ;;TYPE A CR AND LF
006312 006535      $CRLF
006314 105067 000200      CLRB     $CHARCNT  ;;CLEAR CHARACTER COUNT
006320 000755      BR      2$      ;;GET NEXT CHARACTER
006322 004767 000056 5$:      JSR      PC,$TYPEC  ;;GO TYPE THIS CHARACTER
006326 126726 000200 6$:      CMPB     $FILLC,(SP)+  ;;IS IT TIME FOR FILLER CHARS.?
006332 001350      BNE      2$      ;;IF NO GO GET NEXT CHAR.
006334 016746 000170      MOV      $NULL,-(SP)  ;;GET # OF FILLER CHARS. NEEDED
                                ;;AND THE NULL CHAR.
006340 105366 000001 7$:      DECB     1(SP)      ;;DOES A NULL NEED TO BE TYPED?
006344 002770      BLT      6$      ;;BR IF NO--GO POP THE NULL OFF OF STACK
006346 004767 000032      JSR      PC,$TYPEC  ;;GO TYPE A NULL
006352 105367 000142      DECB     $CHARCNT  ;;DO NOT COUNT AS A COUNT
006356 000770      BR      7$      ;;LOOP
                                ;HORIZONTAL TAB PROCESSOR
006360 112716 000040 8$:      MOVB     #' ,(SP)  ;;REPLACE TAB WITH SPACE
006364 004767 000014 9$:      JSR      PC,$TYPEC  ;;TYPE A SPACE
006370 132767 000007 000122      BITB     #7,$CHARCNT  ;;BRANCH IF NOT AT
006376 001372      BNE      9$      ;;TAB STOP
006400 005726      TST      (SP)+   ;;POP SPACE OFF STACK
006402 000724      BR      2$      ;;GET NEXT CHARACTER
006404 105777 000114 $TYPEC: TSTB     @STPS  ;;WAIT UNTIL PRINTER IS READY
006410 100375      BPL      $TYPEC
006412 116677 000002 000106      MOVB     2(SP),@STPB  ;;LOAD CHAR TO BE TYPED INTO DATA REG.
006420 105777 000114      TSTB     @STKS  ;;SEE IF KEYBOARD IS TALKING.
006424 100021      BPL      2$      ;;BRANCH IF IT ISN'T.
006426 017746 000110      MOV      @STKB,-(SP)  ;;PUSH CHARACTER ONTO STACK.
006432 042716 177600      BIC      #177600,(SP)  ;;BIT CLEAR TOP BYTE AND PARITY BIT.
006436 022726 000023      CMP      #23,(SP)+  ;;SEE IF THIS IS A ^S.
006442 001012      BNE      2$      ;;BRANCH TO CONTINUE IF IT ISN'T.
006444 105777 000070 3$:      TSTB     @STKS  ;;WAIT FOR ANOTHER INPUT.
006450 100375      BPL      3$      ;;BRANCH BACK IF NOT READY.
006452 017746 000064      MOV      @STKB,-(SP)  ;;PUSH NEXT CHARACTER ON STACK.
006456 042716 177600      BIC      #177600,(SP)  ;;BIT CLEAR TOP BYTE AND PARITY BIT.
006462 022726 000021      CMP      #21,(SP)+  ;;SEE IF THIS IS A ^Q.
006466 001366      BNE      3$      ;;BRANCH BACK FOR MORE WAIT IF NOT.
006470 122766 000015 000002 2$:    CMPB     #CR,2(SP)  ;;IS CHARACTER A CARRIAGE RETURN?
006476 001003      BNE      1$      ;;BRANCH IF NO
006500 105067 000014      CLRB     $CHARCNT  ;;YES--CLEAR CHARACTER COUNT
006504 000406      BR      $TYPEX  ;;EXIT
006506 122766 000012 000002 1$:    CMPB     #LF,2(SP)  ;;IS CHARACTER A LINE FEED?
006514 001402      BEQ      $TYPEX  ;;BRANCH IF YES
006516 105227      INCB     (PC)+     ;;COUNT THE CHARACTER
006520 000000      $CHARCNT: .WORD  0  ;;CHARACTER COUNT STORAGE
006522 000207      $TYPEX:  RTS      PC
006524 177564      $TPS:   .WORD  177564  ;;TTY PRINTER STATUS REG. ADDRESS
006526 177566      $TPB:   .WORD  177566  ;;TTY PRINTER BUFFER REG. ADDRESS

```

006530 000
006531 002
006532 012
006533 000
006534 077
006535 015
006536 012 000
821 006540 177560
822 006542 177562
823
824

\$NULL: .BYTE 0 ;:CONTAINS NULL CHARACTER FOR FILLS
\$FILLS: .BYTE 2 ;:CONTAINS # OF FILLER CHARACTERS REQUIRED
\$FILLC: .BYTE 12 ;:INSERT FILL CHARS. AFTER A 'LINE FEED'
\$TPFLG: .BYTE 0 ;:'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
\$QUES: .ASCII '?' ;:QUESTION MARK
\$CRLF: .ASCII <15> ;:CARRIAGE RETURN
\$LF: .ASCIZ <12> ;:LINEFEED
\$TKS: .WORD 177560
\$TKB: .WORD 177562

.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.

006544 010046
006546 016600 000002
006552 005740
006554 111000
006556 006300
006560 016000 006600
006564 000200

\$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

006566 011646
006570 016666 000004 000002
006576 000002

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

.SBTTL TRAP TABLE

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

006600 006566
006602 006172
006604 005544
006606 005520
006610 005560
006612 005746

\$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)
\$TYPDS ;:CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)

825
826
827
828
829
830
831
832
833
834
835
836
837 104000
838
839 006614 000000
840
841
842 006616 010067 000026
843 006622 010167 000024

: ERROR PRINT ROUTINE
:-----

ERROR=104000

ERRPC: .WORD 0

\$ERROR: MOV R0,SAVR0 ;:SAVE R0 THRU R5
MOV R1,SAVR1

844	006626	010267	000022		MOV R2,SAVR2	
845	006632	010367	000020		MOV R3,SAVR3	
846	006636	010467	000016		MOV R4,SAVR4	
847	006642	010567	000014		MOV R5,SAVR5	
848	006646	000406			BR TITL	
849	006650	000000		SAVR0:	.WORD	
850	006652	000000		SAVR1:	.WORD	
851	006654	000000		SAVR2:	.WORD	
852	006656	000000		SAVR3:	.WORD	
853	006660	000000		SAVR4:	.WORD	
854	006662	000000		SAVR5:	.WORD	
855						
856	006664	011601		TITL:	MOV (SP),R1	;R1 CONTAINS ADDRESS FOLLOWING ERRPC ADDRESS
857	006666	162701	000002		SUB #2,R1	
858	006672	010167	177716		MOV R1,ERRPC	;GET ADDRESS OF ERROR MESSAGE:ERRPC
859	006676	104401	006704	TYPE	.65\$::TYPE ASCIZ STRING
	006702	000411		BR	.64\$::GET OVER THE ASCIZ
				::65\$:	.ASCIZ	<CRLF><CRLF>/TESTNO
				64\$:		ERRPC/
860	006726	005721		3\$:	TST (R1)+	;R1 POINTS TO NEXT ARGUMENT
861						
862	006730	012702	007702		MOV #PRTABL,R2	;ADDRESS OF START OF PRINT TABLE LIST
863	006734	012703	007156		MOV #PRTITL,R3	;ADDRESS OF STERT OF TITLES
864	006740	005711			TST (R1)	;END OF ARGUMENTS?
865	006742	001405			BEQ DAT	;YES,GO PRINT DATA
866	006744	005723		2\$:	TST (R3)+	;INDEX THRU TITLES
867	006746	021122			CMP (R1),(R2)+	;SEARCH PRINT TABLE LIST FOR TITLE
868	006750	001375			BNE 2\$;NO; CHECK NEXT LOCATION IN LIST
869						
870	006752	004753			JSR PC,@-(R3)	;FOUND IT; GO PRINT TITLE
871						;CODE ERROR ROUTINE
872	006754	000764			BR 3\$	
873						
874						
875	006756			DAT:		
	006756	104401	006764		TYPE	.65\$
	006762	000401			BR	.64\$
				::65\$:	.ASCIZ	<CRLF>
				64\$:		
876	006766	016746	172126		MOV	\$TESTN,-(SP)
	006772	104403			TYPOS	
	006774	006			.BYTE	6
	006775	000			.BYTE	0
877	006776	104401	007004		TYPE	.67\$
	007002	000402			BR	.66\$
				::67\$:	.ASCIZ	/ /
				66\$:		
878	007010	016746	177600		MOV	ERRPC,-(SP)
	007014	104402			TYPOC	
879	007016	104401	004234		TYPE	,\$ENULL
880						
881	007022	011601			MOV (SP),R1	;R1 CONTAINS ADDRESS FOLLOWING ERRORPC ADDRESS
882	007024	162701	000002		SUB #2,R1	
883	007030	005721		3\$:	TST (R1)+	;R1 POINTS TO NEXT ARGUMENT
884	007032	012702	007702		MOV #PRTABL,R2	;ADDRESS OF START OF PRINT TABLE LIST
885	007036	012703	007536		MOV #PRDATA,R3	;ADDRESS OF STERT OF PRINT DATA
886	007042	005711			TST (R1)	;END OF ARGUMENTS?

```

887 007044 001421          BEQ FIN          ;YES
888 007046 005723          2$: TST (R3)+      ;INDEX THRU DATA PRINTS
889 007050 021122          CMP (R1),(R2)+  ;SEARCH PRINT TABLE LIST FOR TITLE
890 007052 001375          BNE 2$         ;NO; CHECK NEXT LOCATION IN LIST
891 007054 104401 007062  TYPE ,69$      ;;TYPE ASCIZ STRING
      007060 000402          BR 68$        ;;GET OVER THE ASCIZ
      ;;69$: .ASCIZ / /
      68$:

892 007066 004753          JSR PC,@-(R3)   ;
893 007070 104401 004234  TYPE , $NULL
894 007074 104401 007102  TYPE ,71$      ;;TYPE ASCIZ STRING
      007100 000402          BR 70$        ;;GET OVER THE ASCIZ
      ;;71$: .ASCIZ / /
      70$:

895 007106 000750          BR 3$

896
897 007110 005721          FIN: TST (R1)+   ;R1 NOW CONTAINS RETURN ADDRESS
898 007112 010116          MOV R1,(SP)    ;SETUP RETURN ADDRESS IN STACK
899 007114 104401 007122  TYPE ,65$      ;;TYPE ASCIZ STRING
      007120 000401          BR 64$        ;;GET OVER THE ASCIZ
      ;;65$: .ASCIZ <CRLF>
      64$:

900
901 007124 016700 177520  MOV SAVR0,R0   ;RESTORE REGISTERS
902 007130 016701 177516  MOV SAVR1,R1
903 007134 016702 177514  MOV SAVR2,R2
904 007140 016703 177512  MOV SAVR3,R3
905 007144 016704 177510  MOV SAVR4,R4
906 007150 016705 177506  MOV SAVR5,R5
907
908 007154 000002          RTI           ;RETURN
909 007156 007202          PRTITL: 1$
910 007160 007230          2$
911 007162 007256          3$
912 007164 007304          4$
913 007166 007332          5$
914 007170 007360          6$
915 007172 007406          7$
916 007174 007434          10$
917 007176 007462          11$
918 007200 007510          12$
919 007202 007510          1$:
      007202 104401 007210  TYPE ,65$      ;;TYPE ASCIZ STRING
      007206 000407          BR 64$        ;;GET OVER THE ASCIZ
      ;;65$: .ASCIZ / EXPDAT /
      64$:

920 007226 000207          RTS PC
921 007230 000207          2$:
      007230 104401 007236  TYPE ,67$      ;;TYPE ASCIZ STRING
      007234 000407          BR 66$        ;;GET OVER THE ASCIZ
      ;;67$: .ASCIZ / RECDAT /
      66$:

922 007254 000207          RTS PC
923 007256 000207          3$:
      007256 104401 007264  TYPE ,69$      ;;TYPE ASCIZ STRING
      007262 000407          BR 68$        ;;GET OVER THE ASCIZ
      ;;69$: .ASCIZ / EXP.ADD /

```

	007302			68\$:				
924	007302	000207				RTS PC		
925	007304			4\$:				
	007304	104401	007312	TYPE	71\$::TYPE ASCIZ STRING	
	007310	000407		BR	70\$::GET OVER THE ASCIZ	
				::71\$:	.ASCIZ	/ REC.ADD	/	
	007330			70\$:				
926	007330	000207				RTS PC		
927	007332			5\$:				
	007332	104401	007340	TYPE	73\$::TYPE ASCIZ STRING	
	007336	000407		BR	72\$::GET OVER THE ASCIZ	
				::73\$:	.ASCIZ	/ EXP.SP1	/	
	007356			72\$:				
928	007356	000207				RTS PC		
929	007360			6\$:				
	007360	104401	007366	TYPE	75\$::TYPE ASCIZ STRING	
	007364	000407		BR	74\$::GET OVER THE ASCIZ	
				::75\$:	.ASCIZ	/ REC.SP1	/	
	007404			74\$:				
930	007404	000207				RTS PC		
931	007406			7\$:				
	007406	104401	007414	TYPE	77\$::TYPE ASCIZ STRING	
	007412	000407		BR	76\$::GET OVER THE ASCIZ	
				::77\$:	.ASCIZ	/ EXP.PSW	/	
	007432			76\$:				
932	007432	000207				RTS PC		

934 007434
007434 104401 007442
007440 000407

10\$:
TYPE ,79\$
BR 78\$
::TYPE ASCIZ STRING
::GET OVER THE ASCIZ
::79\$: .ASCIZ / REC.PSW /
78\$:

007460
935 007460 000207
936 007462
007462 104401 007470
007466 000407

RTS PC
11\$:
TYPE ,81\$
BR 80\$
::TYPE ASCIZ STRING
::GET OVER THE ASCIZ
::81\$: .ASCIZ / VIR.ADD /
80\$:

007506

CKKACAO 11/44 POWER FAIL
TRAP TABLE

MACRO M1113 10-DEC-79 10:05 PAGE 19^{F 4}

SEQ 0044

938 007506 000207

RTS PC

```

940 007510          12$:
    007510 104401 007516  TYPE      83$      ::TYPE ASCIZ STRING
    007514 000407          BR      82$      ::GET OVER THE ASCIZ
      ::83$: .ASCIZ / PAR.OFF /
      82$:
941 007534          RTS PC
    007534 000207          .EVEN
942
943
944
945
946 007536 007562  PRDATA: 1$
947 007540 007572    2$
948 007542 007602    3$
949 007544 007612    4$
950 007546 007622    5$
951 007550 007632    6$
952 007552 007642    7$
953 007554 007652   10$
954 007556 007662   11$
955 007560 007672   12$
956 007562          1$:
    007562 016746 000140 MOV      EXPDAT,-(SP)  ::SAVE EXPDAT FOR TYPEOUT
    007566 104402          TYPOC  ::GO TYPE--OCTAL ASCII(ALL DIGITS)
957 007570 000207          RTS PC
958 007572          2$:
    007572 016746 000132 MOV      RECDAT,-(SP)  ::SAVE RECDAT FOR TYPEOUT
    007576 104402          TYPOC  ::GO TYPE--OCTAL ASCII(ALL DIGITS)
959 007600 000207          RTS PC
960 007602          3$:
    007602 016746 000124 MOV      EXP.ADD,-(SP)  ::SAVE EXP.ADD FOR TYPEOUT
    007606 104402          TYPOC  ::GO TYPE--OCTAL ASCII(ALL DIGITS)
961 007610 000207          RTS PC
962 007612          4$:
    007612 016746 000116 MOV      REC.ADD,-(SP)  ::SAVE REC.ADD FOR TYPEOUT
    007616 104402          TYPOC  ::GO TYPE--OCTAL ASCII(ALL DIGITS)
963 007620 000207          RTS PC
964 007622          5$:
    007622 016746 000110 MOV      EXP.SP1,-(SP)  ::SAVE EXP.SP1 FOR TYPEOUT
    007626 104402          TYPOC  ::GO TYPE--OCTAL ASCII(ALL DIGITS)
965 007630 000207          RTS PC
966 007632          6$:
    007632 016746 000102 MOV      REC.SP1,-(SP)  ::SAVE REC.SP1 FOR TYPEOUT
    007636 104402          TYPOC  ::GO TYPE--OCTAL ASCII(ALL DIGITS)
967 007640 000207          RTS PC
968 007642          7$:
    007642 016746 000074 MOV      EXP.PSW,-(SP)  ::SAVE EXP.PSW FOR TYPEOUT
    007646 104402          TYPOC  ::GO TYPE--OCTAL ASCII(ALL DIGITS)
969 007650 000207          RTS PC
970 007652          10$:
    007652 016746 000066 MOV      REC.PSW,-(SP)  ::SAVE REC.PSW FOR TYPEOUT
    007656 104402          TYPOC  ::GO TYPE--OCTAL ASCII(ALL DIGITS)
971 007660 000207          RTS PC
972 007662          11$:
    007662 016746 000060 MOV      VIR.ADD,-(SP)  ::SAVE VIR.ADD FOR TYPEOUT
    007666 104402          TYPOC  ::GO TYPE--OCTAL ASCII(ALL DIGITS)
973 007670 000207          RTS PC
974 007672          12$:

```

	007672	016746	000052
	007676	104402	
975	007700	000207	
976			
977			
978	007702		
979	007702	007726	
980	007704	007730	
981	007706	007732	
982	007710	007734	
983	007712	007736	
984	007714	007740	
985	007716	007742	
986	007720	007744	
987	007722	007746	
988	007724	007750	
989			

MOV PAR.OFF, -(SP) ;;SAVE PAR.OFF FOR TYPEOUT
TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
RTS PC

PRTABL:

EXPDAT
RECDAT
EXP.ADD
REC.ADD
EXP.SP1
REC.SP1
EXP.PSW
REC.PSW
VIR.ADD
PAR.OFF

992	007726	000000		
993	007730	000000		
994	007732	000000		
995	007734	000000		
996	007736	000000		
997	007740	000000		
998	007742	000000		
999	007744	000000		
1000	007746	000000		
1001	007750	000000		
1002	007752	015	012	103
	007755	113	113	101
	007760	103	101	060
	007763	040	061	061
	007766	057	064	064
	007771	040	120	117
	007774	127	105	122
	007777	040	106	101
	010002	111	114	000
1003	010005	015	012	103
	010010	113	113	101
	010013	103	101	060
	010016	040	061	061
	010021	057	064	064
	010024	040	120	117
	010027	127	105	122
	010032	040	106	101
	010035	111	114	015
	010040	012	000	
1004	010042	015	012	102
	010045	117	117	124
	010050	040	105	116
	010053	101	102	114
	010056	105	040	123
	010061	127	111	124
	010064	103	110	040
	010067	115	125	123
	010072	124	040	102
	010075	105	040	117
	010100	106	106	015
	010103	012	000	
1005	010105	015	012	120
	010110	117	127	105
	010113	122	040	125
	010116	120	040	102
	010121	105	106	117
	010124	122	105	040
	010127	120	117	127
	010132	105	122	040
	010135	104	117	127
	010140	116	040	103
	010143	117	115	120
	010146	114	105	124
	010151	105	015	012
	010154	000		
1006				
1007	010155	015	012	120

EXPDAT: .WORD 0
 RECDAT: .WORD 0
 EXP.ADD: .WORD 0
 REC.ADD: .WORD 0
 EXP.SP1: .WORD 0
 REC.SP1: .WORD 0
 EXP.PSW: .WORD 0
 REC.PSW: .WORD 0
 VIR.ADD: .WORD 0
 PAR.OFF: .WORD 0
 PNAME: .ASCIZ <15><12>+CKKACAO 11/44 POWER FAIL+

TITLE: .ASCIZ <15><12>+CKKACAO 11/44 POWER FAIL+<15><12>

BOOT: .ASCIZ <15><12>+BOOT ENABLE SWITCH MUST BE OFF+<15><12>

UPF: .ASCIZ <15><12>+POWER UP BEFORE POWER DOWN COMPLETE+<15><12>

DOWN: .ASCIZ <15><12>+POWER DOWN BEFORE UP COMPLETE+<15><12>

010160	117	127	105
010163	122	040	104
010166	117	127	116
010171	040	102	105
010174	106	117	122
010177	105	040	125
010202	120	040	103
010205	117	115	120
010210	114	105	124
010213	105	015	012
010216	000		

1008
1009
1010
1011
1012
1013
1014
1015

010220 000000
000001

.EVEN
END: 0
.END

ABASE = 000000	BIT02 = 000004	IOTVEC = 000020	MAPH13 = 170256	MAPL32 = 170350
ACDW1 = 000000	BIT03 = 000010	KDPAR0 = 172360	MAPH14 = 170262	MAPL33 = 170354
ACDW2 = 000000	BIT04 = 000020	KDPAR1 = 172362	MAPH15 = 170266	MAPL34 = 170360
ACPUOP = 000000	BIT05 = 000040	KDPAR2 = 172364	MAPH16 = 170272	MAPL35 = 170364
ADDW0 = 000000	BIT06 = 000100	KDPAR3 = 172366	MAPH17 = 170276	MAPL36 = 170370
ADDW1 = 000000	BIT07 = 000200	KDPAR4 = 172370	MAPH2 = 170212	MAPL37 = 170374
ADDW10 = 000000	BIT08 = 000400	KDPAR5 = 172372	MAPH20 = 170302	MAPL4 = 170220
ADDW11 = 000000	BIT09 = 001000	KDPAR6 = 172374	MAPH21 = 170306	MAPL5 = 170224
ADDW12 = 000000	BIT1 = 000002	KDPAR7 = 172376	MAPH22 = 170312	MAPL6 = 170230
ADDW13 = 000000	BIT10 = 002000	KDPDR0 = 172320	MAPH23 = 170316	MAPL7 = 170234
ADDW14 = 000000	BIT11 = 004000	KDPDR1 = 172322	MAPH24 = 170320	MEMERR = 177744
ADDW15 = 000000	BIT12 = 010000	KDPDR2 = 172324	MAPH25 = 170326	MMERR = 005206
ADDW2 = 000000	BIT13 = 020000	KDPDR3 = 172326	MAPH26 = 170332	MMR0 = 177572
ADDW3 = 000000	BIT14 = 040000	KDPDR4 = 172330	MAPH27 = 170336	MMR1 = 177574
ADDW4 = 000000	BIT15 = 100000	KDPDR5 = 172332	MAPH3 = 170216	MMR2 = 177576
ADDW5 = 000000	BIT2 = 000004	KDPDR6 = 172334	MAPH30 = 170342	MMR3 = 172516
ADDW6 = 000000	BIT3 = 000010	KDPDR7 = 172336	MAPH31 = 170346	MMVEC = 000250
ADDW7 = 000000	BIT4 = 000020	KERSTK = 001100	MAPH32 = 170352	PARERR = 004554
ADDW8 = 000000	BIT5 = 000040	KIPAR0 = 172340	MAPH33 = 170356	PARFLG = 004556
ADDW9 = 000000	BIT6 = 000100	KIPAR1 = 172342	MAPH34 = 170362	PAR_OF = 007750
ADEVCT = 000000	BIT7 = 000200	KIPAR2 = 172344	MAPH35 = 170366	PFAIL = 001416
ADEVN = 000000	BIT8 = 000400	KIPAR3 = 172346	MAPH36 = 170372	PIRQ = 177772
AENV = 000000	BIT9 = 001000	KIPAR4 = 172350	MAPH37 = 170376	PIRQVE = 000240
AENVN = 000000	BOOT = 010042	KIPAR5 = 172352	MAPH4 = 170222	PNAME = 007752
AFATAL = 000000	BPTVEC = 000014	KIPAR6 = 172354	MAPH5 = 170226	POWDWN = 004654
AMADR1 = 000000	CACHE = 000114	KIPAR7 = 172356	MAPH6 = 170232	POWUP = 004750
AMADR2 = 000000	CHECK = 004360	KIPDR0 = 172300	MAPH7 = 170236	PRDATA = 007536
AMADR3 = 000000	CONTRL = 177746	KIPDR1 = 172302	MAPL0 = 170200	PRETST = 002274
AMADR4 = 000000	CPUERR = 177766	KIPDR2 = 172304	MAPL00 = 170200	PRTABL = 007702
AMAMS1 = 000000	CR = 000015	KIPDR3 = 172306	MAPL01 = 170204	PRTITL = 007156
AMAMS2 = 000000	CRLF = 000200	KIPDR4 = 172310	MAPL02 = 170210	PRO = 000000
AMAMS3 = 000000	DAT = 006756	KIPDR5 = 172312	MAPL03 = 170214	PR1 = 000040
AMAMS4 = 000000	DATA = 005106	KIPDR6 = 172314	MAPL04 = 170220	PR2 = 000100
AMSGAD = 000000	DDISP = 177570	KIPDR7 = 172316	MAPL05 = 170224	PR3 = 000140
AMSGLG = 000000	DISPLA = 177570	KSP = %000006	MAPL06 = 170230	PR4 = 000200
AMSGTY = 000000	DISPRE = 000174	LF = 000012	MAPL07 = 170234	PR5 = 000240
AMTYP1 = 000000	DOWN = 010155	LIMIT = 005104	MAPL1 = 170204	PR6 = 000300
AMTYP2 = 000000	DSWR = 177570	LKS = 177546	MAPL10 = 170240	PR7 = 000340
AMTYP3 = 000000	DVEC = 005074	LKVEC = 000100	MAPL11 = 170244	PS = 177776
AMTYP4 = 000000	EMTVEC = 000030	LOAD = 004254	MAPL12 = 170250	PSW = 177776
APASS = 000000	END = 010220	LOADRS = 177740	MAPL13 = 170254	PWRVEC = 000024
APF = 002134	ERRAD = 005066	LRTI = 001422	MAPL14 = 170260	RECDAT = 007730
APRIOR = 000000	ERROR = 104000	MAINT = 177750	MAPL15 = 170264	REC.AD = 007734
APTCSU = 000040	ERRPC = 006614	MAP = 005110	MAPL16 = 170270	REC.PS = 007744
APTENV = 000001	ERRVEC = 000004	MAPH0 = 170202	MAPL17 = 170274	REC.SP = 007740
APTSIZ = 000200	EXPDAT = 007726	MAPH00 = 170202	MAPL2 = 170210	RESVEC = 000010
APTSPO = 000100	EXP.AD = 007732	MAPH01 = 170206	MAP20 = 170300	R10 = %000000
ASWREG = 000000	EXP.PS = 007742	MAPH02 = 170212	MAPL21 = 170304	R11 = %000001
ATESTN = 000000	EXP.SP = 007736	MAPH03 = 170216	MAPL22 = 170310	R12 = %000002
AUNIT = 000000	FIN = 007110	MAPH04 = 170222	MAPL23 = 170314	R13 = %000003
AUSWR = 000000	FLAG = 005102	MAPH05 = 170226	MAPL24 = 170320	R14 = %000004
AVECT1 = 000000	FT = 002264	MAPH06 = 170232	MAPL25 = 170324	R15 = %000005
AVECT2 = 000000	HIADRS = 177742	MAPH07 = 170236	MAPL26 = 170330	R6 = %000006
BEGIN = 001424	HITMIS = 177752	MAPH1 = 170206	MAPL27 = 170334	R7 = %000007
BIT0 = 000001	HT = 000011	MAPH10 = 170242	MAPL3 = 170214	SAVR0 = 006650
BIT00 = 000001	ILLDWN = 005054	MAPH11 = 170246	MAPL30 = 170340	SAVR1 = 006652
BIT01 = 000002	ILLUP = 005042	MAPH12 = 170252	MAPL31 = 170344	SAVR2 = 006654

SAVR3 006656	STACK1= 000500	TST5 003370	XYZ 001420	\$MSGAD 001130
SAVR4 006660	STKLMT= 177774	TST6 003500	\$APTHD 001100	\$MSGLG 001132
SAVR5 006662	SUPSTK= 000700	TST7 003620	\$ATYC 001174	\$MSGTY 001114
SAV6 005100	SWR = 177570	TYPDS = 104405	\$ATY1 001150	\$MTYP1 001145
SCOPE = 000004	SWREG 000176	TYPE = 104401	\$ATY3 001156	\$NULL 006530
SDPAR0= 172260	SW0 = 000001	TYPOC = 104402	\$ATY4 001166	\$NWTST= 000001
SDPAR1= 172262	SW00 = 000001	TYPON = 104404	\$CHARC 006520	\$OCNT 005742
SDPAR2= 172264	SW01 = 000002	TYPOS = 104403	\$CORE 005420	\$OMODE 005744
SDPAR3= 172266	SW02 = 000004	UDPAR0= 177660	\$CPUOP 001142	\$PASS 001122
SDPAR4= 172270	SW03 = 000010	UDPAR1= 177662	\$CRLF 006535	\$PASTM 001106
SDPAR5= 172272	SW04 = 000020	UDPAR2= 177664	\$CROUT 005450	\$QUES 006534
SDPAR6= 172274	SW05 = 000040	UDPAR3= 177666	\$DBLK 006162	\$RTNAD 004232
SDPAR7= 172276	SW06 = 000100	UDPAR4= 177670	\$DEVCT 001124	\$SAVPC= 000204
SDPDR0= 172220	SW07 = 000200	UDPAR5= 177672	\$DOAGN 004230	\$SETUP= 000024
SDPDR1= 172222	SW08 = 000400	UDPAR6= 177674	\$DTBL 006152	\$SIZE 005212
SDPDR2= 172224	SW09 = 001000	UDPAR7= 177676	\$ENDAD 004220	\$SIZEX 005454
SDPDR3= 172226	SW1 = 000002	UDPDR0= 177620	\$ENDCT 004166	\$STUP = 177777
SDPDR4= 172230	SW10 = 002000	UDPDR1= 177622	\$ENDMG 004237	\$SVPC = 000204
SDPDR5= 172232	SW11 = 004000	UDPDR2= 177624	\$ENULL 004234	\$SWR = 040000
SDPDR6= 172234	SW12 = 010000	UDPDR3= 177626	\$ENV 001134	\$SWREG 001136
SDPDR7= 172236	SW13 = 020000	UDPDR4= 177630	\$ENVM 001135	\$TESTN 001120
SIPAR0= 172240	SW14 = 040000	UDPDR5= 177632	\$EOP 004142	\$TKB 006542
SIPAR1= 172242	SW15 = 100000	UDPDR6= 177634	\$EOPCT 004160	\$TKS 006540
SIPAR2= 172244	SW2 = 000004	UDPDR7= 177636	\$ERROR 006616	\$TN = 000012
SIPAR3= 172246	SW3 = 000010	UIPAR0= 177640	\$ETABL 001134	\$TPB 006526
SIPAR4= 172250	SW4 = 000020	UIPAR1= 177642	\$ETEND 001150	\$TPFLG 006533
SIPAR5= 172252	SW5 = 000040	UIPAR2= 177644	\$FATAL 001116	\$TPS 006524
SIPAR6= 172254	SW6 = 000100	UIPAR3= 177646	\$FFLG 001414	\$TRAP 006544
SIPAR7= 172256	SW7 = 000200	UIPAR4= 177650	\$FILLC 006532	\$TRAP2 006566
SIPDR0= 172200	SW8 = 000400	UIPAR5= 177652	\$FILLS 006531	\$TRP = 000006
SIPDR1= 172202	SW9 = 001000	UIPAR6= 177654	\$GET42 004210	\$TRPAD 006600
SIPDR2= 172204	SYSTID= 177764	UIPAR7= 177656	\$HD = 000000	\$TSTM 001104
SIPDR3= 172206	TBITVE= 000014	UIPDR0= 177600	\$HIBTS 001100	\$TYPDS 005746
SIPDR4= 172210	TITL 006664	UIPDR1= 177602	\$KTNEX 005412	\$TYPE 006172
SIPDR5= 172212	TITLE 010005	UIPDR2= 177604	\$KTOUT 005402	\$TYPEC 006404
SIPDR6= 172214	TKVEC = 000060	UIPDR3= 177606	\$KT11 005250	\$TYPEX 006522
SIPDR7= 172216	TPVEC = 000064	UIPDR4= 177610	\$LF 006536	\$TYPOC 005544
SIZEHI= 177762	TRAPVE= 000034	UIPDR5= 177612	\$LFLG 001413	\$TYPON 005560
SIZELO= 177760	TRTVEC= 000014	UIPDR6= 177614	\$LSTAD 005514	\$TYPOS 005520
SRO = 177572	TST1 002332	UIPDR7= 177616	\$LSTBK 005516	\$UNIT 001126
SR1 = 177574	TST10 003720	UPF 010105	\$MADR1 001146	\$UNITM 001110
SR2 = 177576	TST11 004052	USESTK= 000600	\$MAIL 001114	\$USWR 001140
SR3 = 172516	TST2 002554	USP = %000006	\$MAMS1 001144	\$GET4= 000000
SSP = %000006	TST3 003004	UVEC 005070	\$MBADR 001102	\$OFILL 005743
STACK = 001100	TST4 003264	VIR.AD 007746	\$MFLG 001412	.\$X = 001100

. ABS. 010222 000
000000 001
ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 31896 WORDS (125 PAGES)
DYNAMIC MEMORY: 20536 WORDS (78 PAGES)
ELAPSED TIME: 00:01:18
CKKACA.BIN,CKKACA.SEQ=CKKACA.MLB/ML,CKKACA.P11