

KD11-Z

11/44 MEM MGMT PRT B  
CKKT BBO

AH-F614B-MC  
FICHE 1 OF 1

MAR 1980  
COPYRIGHT © 1980  
MADE IN USA



A large grid of approximately 10 columns and 15 rows of data. Each cell contains a small table or list of numbers and text, likely representing a detailed schedule or resource allocation. The data is too small to read accurately but appears to be organized in a structured format.





.REM @

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42

IDENTIFICATION  
-----

PRODUCT CODE: AC-F612B-MC  
PRODUCT NAME: CKKTBB0 11/44 MEM MGMT PRT B  
DATE: DECEMBER, 1979  
MAINTAINER: DIAGNOSTIC ENGINEERING  
AUTHOR: J. PETER BRADY, DAN P. MILLEVILLE

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

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED 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

44  
45  
46  
47  
48  
49  
50  
51

PROGRAM HISTORY

| <u>DATE</u>    | <u>REVISION</u> | <u>REASON FOR REVISION</u> |
|----------------|-----------------|----------------------------|
| OCTOBER, 1979  | A               | FIRST RELEASE              |
| DECEMBER, 1979 | B               | ADDITION OF 'CSM' TEST 35  |

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

TABLE OF CONTENTS  
-----

- 1.0 PROGRAM INFORMATION
  - 1.1 ABSTRACT
  - 1.2 REQUIREMENTS
  - 1.3 RELATED DOCUMENTS AND STANDARDS
  - 1.4 PRELIMINARY PROGRAMS
  
- 2.0 OPERATING INSTRUCTIONS
  - 2.1 LOADING PROCEDURES
  - 2.2 STARTING PROCEDURES
  - 2.3 OPERATIONAL SWITCH SETTINGS
  - 2.4 LOADING THE SWITCH REGISTER
  - 2.5 EXECUTION TIMES
  
- 3.0 ERROR INFORMATION
  - 3.1 ERROR REPORTING PROCEDURES
  - 3.2 INTERPRETING ERROR REPORTS
  - 3.3 SAMPLE ERROR REPORT
  
- 4.0 MISCELLANEOUS INFORMATION
  - 4.1 ACT/APT/XXDP COMPATABILITY
  - 4.2 END-OF-PASS MESSAGE
  - 4.3 T-BIT TRAPPING
  - 4.4 POWER FAILURE HANDLING
  - 4.5 PHYSICAL BUS ADDRESS CONSTRUCTION
  
- 5.0 PROGRAM DESCRIPTION
  - 5.1 SUBROUTINES USED BY THIS PROGRAM
  - 5.2 PROGRAM LISTING
  - 5.3 USING THE PROGRAM TO DIAGNOSE A FAULT

87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143

1.0 PROGRAM INFORMATION  
-----

1.1 ABSTRACT  
-----

THIS PROGRAM WAS DESIGNED USING A 'BOTTOM UP' APPROACH STARTING WITH THE SMALLEST SEGMENT OF MEMORY MANAGEMENT LOGIC POSSIBLE AND BUILDING TO COVER ALL OF THE LOGIC. THE DIAGNOSTIC WILL PROVIDE ENOUGH INFORMATION SUCH THAT BY DEDUCTION, THE FAILURE CAN BE ISOLATED TO A SMALL SEGMENT OF THE MEMORY MANAGEMENT LOGIC.

PART A BEGINS BY TESTING SOME OF THE INTERNAL CPU DATA AND ADDRESS PATHS AND ADDRESS DETECTION LOGIC, THEN WORKS OUTWARD THROUGH THE MEMORY MANAGEMENT REGISTERS. AFTER THE REGISTERS ARE FOUND TO BE USEABLE, RELOCATION (CONSTRUCTION OF PHYSICAL ADDRESSES FROM A VIRTUAL ADDRESS AND THE ASSOCIATED PAR/PDP INFORMATION) IS TESTED. PART B BEGINS BY TESTING THE ABORT AND STATUS SEGMENTS OF LOGIC. PART B THEN CHECKS THE SPECIAL ABORT SEQUENCES, THE MFPI/MTPI INSTRUCTIONS AND THE CSM INSTRUCTION.

1.2 REQUIREMENTS  
-----

A PDP 11/44 PROCESSOR WITH A MINIMUM OF 16K OF MEMORY AND A CONSOLE TERMINAL ARE REQUIRED TO RUN THE PROGRAM UNLESS THE PROGRAM IS RUNNING UNDER APT OR ACT IN WHICH CASE THE CONSOLE TERMINAL IS NOT NECESSARY.

1.3 RELATED DOCUMENTS AND STANDARDS  
-----

1. ACT11/XXDP PROGRAMMING SPECIFICATION
2. STANDARD APT SYSTEM TO A PDP11 DIAGNOSTIC INTERFACE
3. DIAGNOSTIC ENGINEERING STANDARDS AND CONVENTIONS
4. PDP11 MAINDEC SYSMAC PACKAGE
5. XXDP USER'S MANUAL

1.4 PRELIMINARY PROGRAMS  
-----

BEFORE THIS MEMORY MANAGEMENT DIAGNOSTIC IS RUN, THE FOLLOWING DIAGNOSTICS SHOULD BE RUN:

CKKAAA0 11/4 CPU/EIS  
CKKABAO TRAPS

ALSO, THE MAIN MEMORY DIAGNOSTIC (CZMSD) SHOULD BE RUN TO SCAN AT LEAST THE FIRST 16K TO SEE THAT A PROGRAM CAN BE EXECUTED.

2.0 OPERATING INSTRUCTIONS  
-----

2.1 LOADING PROCEDURES  
-----

144  
145  
146  
147  
148  
149  
150  
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

THE PROGRAM IS SUPPLIED ON THE DIAGNOSTIC LOAD MEDIA. REFER TO THE XXDP USER'S MANUAL FOR FURTHER INFORMATION. FOR USE WITH ACT OR APT, REFER TO THEIR RESPECTIVE DOCUMENTS. THE PROGRAM CAN ALSO BE DIRECTLY LOADED USING THE ABSOLUTE LOADER AND THE BINARY PAPER TAPE.

2.2 STARTING PROCEDURES  
-----

THE PROGRAM IS STARTED BY LOADING ADDRESS 200 AND STARTING. THE SWITCH REGISTER SHOULD BE SET ACCORDING TO SECTION 2.3 BEFORE THE PROGRAM IS STARTED. HOWEVER, IF DESIRED, THE PROGRAM WILL USE THE SOFTWARE SWITCH REGISTER AT LOCATION 176 (LOCATION 174 WILL BE USED AS THE SOFTWARE DISPLAY REGISTER). IN THAT CASE, THE PROGRAM WILL ASK FOR THE INITIAL SWITCH REGISTER VALUE BY TYPING 'SWR= XXXXXX NEW= ' AFTER TYPING THE NAME OF THE PROGRAM (XXXXXX = THE OCTAL CONTENTS OF LOCATION 176). (SEE SECTION 2.4)

ALSO THE PROGRAM CAN BE MADE TO USE THE SOFTWARE SWITCH REG. EVEN IF THE CONSOLE SWITCH REG. IS PRESENT BY LOADING '177777' INTO THE CONSOLE SWITCH REG. BEFORE STARTING THE PROGRAM.

2.3 CONTROL SWITCH SETTINGS  
-----

| SWITCH | OCTAL VALUE | USE   |
|--------|-------------|---|
| SW15   | 100000      | HALT ON ERROR<br>THIS SWITCH WHEN SET WILL HALT THE PROCESSOR WHEN AN ERROR IS DETECTED AFTER THE ERROR MESSAGE HAS BEEN TYPED. PRESSING CONTINUE WILL RESUME TESTING (SEE SECTION 3.1 ABOUT LOADING THE SWITCH REG BEFORE CONTINUING). |
| SW14   | 040000      | LOOP ON TEST<br>THIS SWITCH WHEN SET WILL CAUSE THE PROGRAM TO LOOP ON THE CURRENT SUBTEST.   |
| SW13   | 020000      | INHIBIT ERROR TYPEOUTS<br>THIS SWITCH WHEN SET WILL INHIBIT THE TYPING OF ERROR MESSAGES.   |
| SW12   | 010000      | INHIBIT TRACE TRAP<br>THIS SWITCH WHEN SET WILL INHIBIT T-BIT TRAPPING WHICH NORMALLY TAKES PLACE DURING EVERY OTHER PASS STARTING WITH THE THIRD PASS.   |
| SW10   | 002000      | BFL ON ERROR  |

201  
202  
203  
204  
205  
206  
207  
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  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257

THIS SWITCH WHEN SET WILL RING THE CONSOLE TERMINAL BELL WHEN AN ERROR HAS BEEN DETECTED.

SW9 001000 LOOP ON ERROR  
THIS SWITCH WHEN SET WILL CAUSE THE PROGRAM TO LOOP ON THE FIRST FAILURE WHICH IS ENCOUNTERED EVEN IF THE FAILURE IS INTERMITTANT

SW8 000400 LOOP ON TEST IN SWR<7:0>  
THIS SWITCH WHEN SET WILL CAUSE THE PROGRAM TO LOOP ON THE TEST WHOSE TEST NUMBER IS SET IN BITS 7-0 OF THE SWITCH REG.

2.4 LOADING THE SWITCH REGISTER

THE CONSOLE SWITCH REGISTER PROVIDED IS LOADED DIRECTLY FROM THE CONSOLE BY TYPING A CONTROL P (^P), THEN WHEN THE CONSOLE PROMPT IS RECEIVED, TYPE 'D SW XXXXXX', WHERE 'XXXXXX' IS THE INTENDED VALUE OF THE SWITCH REGISTER. THE VALUE OF THE CONSOLE SWITCH REG. CAN BE CHANGED ANY TIME WHETHER THE PROGRAM IS RUNNING OR NOT.

TO LOAD THE SOFTWARE SWITCH REG. WHILE THE PROGRAM IS RUNNING, A CONTROL G (^G) SHOULD BE TYPED ON THE CONSOLE TERMINAL. (THE 'SCOPE' AND 'ERROR' ROUTINES CHECK TO SEE IF A ^G HAS BEEN TYPED.) THE ORIGINAL VALUE OF THE SOFTWARE SWITCH REG. WILL BE REQUESTED AS MENTIONED IN SECTION 2.2.

IN RESPONSE TO A ^G OR AT THE BEGINNING OF THE PROGRAM, THE PROGRAM WILL TYPE:

SWR = XXXXXX NEW -

WHERE 'XXXXXX' IS THE CURRENT OCTAL CONTENTS OF LOC. 176. THE OPERATOR MAY THEN TYPE ANY ONE OF THE FOLLOWING:

- XXXXXX<CR> ONE TO SIX OCTAL DIGITS FOLLOWED BY A CARRIAGE RETURN WHICH WILL BE LOADED AS THE NEW VALUE FOR THE SWITCH REG.
- <CR> JUST A <CR>, LEAVES THE SWITCH REG. AS IT IS.
- XXX^U A CONTROL-U (^U) WILL CAUSE ALL OF THE DIGITS TYPED SO FAR TO BE IGNORED.
- ^C WILL CAUSE THE PROGRAM TO TYPE THE PRESENT TEST AND PASS NUMBERS, REQUEST A NEW VALUE FOR THE SWITCH REG., AND JUMP TO THE END-OF-PASS ROUTINE SO THE PROGRAM WILL GO DIRECTLY TO THE NEXT PASS WITH A NEW SW. REG. VALUE
- <ILL.CHAR> ANY CHARACTER TYPED WHICH IS NOT ANY OF THE ABOVE OR AN OCTAL DIGIT WILL CAUSE THE PROGRAM TO TYPE A '?<CRLF>' AND REACT AS THOUGH A ^U HAD BEEN TYPED.

NOTE: RECOGNITION OF A ^G MAY BE HAMPERED BY

258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314

----- EXECUTION OF A COUPLE 'RESET' INSTRUCTIONS  
WITHIN THE PROGRAM.

2.5 EXECUTION TIMES  
-----

THE RUN TIME FOR A SINGLE PASS WITH TRACE TRAPPING  
ENABLED IS APPROXIMATELY 5 SECONDS WITH CACHE.

3.0 ERROR INFORMATION  
-----

3.1 ERROR REPORTING PROCEDURES  
-----

IF AN ERROR IS DETECTED, THE PROGRAM WILL TRAP TO THE  
ERROR HANDLING ROUTINE (\$ERROR). THE VALUE OF BITS  
15,13,10, AND 9 IN THE SWITCH REGISTER ARE CONSIDERED  
IN REPORTING AN ERROR (SEE SECTION 2.3). THE  
ERROR INFORMATION WILL BE TYPED UNLESS SW13 = 1.

IF SW15 = 1, THE PROCESSOR WILL HALT AFTER THE ERROR IS  
REPORTED. IF THE CONTENTS OF THE SOFTWARE SWITCH REGISTER  
ARE TO BE CHANGED, A ^G SHOULD BE TYPED BEFORE PRESSING  
'CONTINUE' TO RESUME TESTING.

IF SW9 = 1 (LOOP ON ERROR), THE PROGRAM WILL GO TO THE  
ADDRESS CONTAINED IN LOCATION '\$LPERR'. AFTER REPORTING  
THE ERROR, '\$LPERR' IS SET BY EACH 'SCOPE' CALL AND IS  
SET DIRECTLY DURING SOME SUBTESTS TO PROVIDE THE SMALLEST  
LOOP FOR LOOPING ON ERROR. IF SW9 = 0, THE PROGRAM WILL  
RETURN TO THE INSTRUCTION FOLLOWING THE ERROR CALL.  
(SEE SECTION 5.3 FOR MORE ON 'LOOP ON ERROR').

3.2 INTERPRETING ERROR REPORTS  
-----

EVERY ERROR REPORT TYPES THE NUMBER OF THE TEST IN WHICH  
THE ERROR TOOK PLACE (TESTNO) AND THE LOCATION OF THE  
ERROR CALL (ERRORPC). THESE TWO VALUES PINPOINT THE  
PLACE IN THE CODE THAT THE ERROR OCCURRED. BY REFERRING  
TO THE PROGRAM LISTING, THE OPERATOR CAN THEN READ THE  
COMMENTS ASSOCIATED WITH THAT PARTICULAR ERROR AND SUBTEST.  
A DESCRIPTION OF THE TEST FOUND IN THE PROGRAM LISTING  
WILL ALSO PROVIDE THE OPERATOR WITH INFORMATION ON THE LOGIC  
AND FUNCTIONS BEING TESTED.

EVERY ERROR REPORT ALSO TYPES AN ERROR MESSAGE  
GIVING A VERBAL DESCRIPTION OF THE ERROR THAT HAS  
BEEN DETECTED.

BY USING THE COMMENTS AND TEST DESCRIPTION FOUND IN  
THE PROGRAM LISTING TO DETERMINE WHAT FUNCTION OR  
LOGIC WAS BEING TESTED, THE OPERATOR CAN THEN REFER  
TO THE ENGINEERING DRAWINGS TO ISOLATE THE PROBABLE  
CAUSE FOR THE FAILURE.



315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371

3.3 SAMPLE ERROR REPORT  
-----

BELOW IS AN EXAMPLE OF AN ERROR WHICH COULD HAVE OCCURRED DURING EXECUTION OF THE PROGRAM:

```
MEM. MGMT. REG. BITS NOT SET CORRECTLY
REGISTR WROTE  READ  READ-(BINARY)
ADDRESS (OCTAL) (OCTAL) 5432109876543210  TESTNO  ERRORPC
177572  040000  060000  0110000000000000  000012  022060
```

WE SEE THAT THE ERROR OCCURRED IN TEST 12 AT LOCATION 022060. THE 'REGISTER ADDRESS' TELLS US THAT WE WERE TESTING MEMORY MANAGEMENT'S STATUS REGISTER 0 (SRO). IN THE LISTING, THE TEST DESCRIPTION SAYS THAT THE ERROR BITS (BITS <15:13>) OF SRO WERE BEING SET AND CLEARED INDIVIDUALLY. THE ERROR REPORT SAYS WE TRIED TO SET BIT 14 BY WRITING '040000' TO SRO BUT WHEN WE READ IT BACK WE READ '060000'. IT APPEARS THAT BIT 13 IS STUCK AT '1' OR IT IS GETTING SET WHEN BIT 14 IS SET TO '1'. ERROR REPORTS BEFORE AND AFTER THIS ONE COULD TELL US WHICH IS THE CASE.

4.0 MISCELLANEOUS INFORMATION  
-----

4.1 ACT/APT/XXDP COMPATABILITY  
-----

THE PROGRAM IS FULLY ACT AND APT COMPATABLE AND IS SUPPORTED UNDER THE XXDP PACKAGE.

4.2 END-OF-PASS MESSAGE  
-----

AT THE END OF EACH PASS OF THE PROGRAM THE PASS NUMBER AND TOTAL NUMBER OF ERRORS SINCE THE LAST END-OF-PASS ARE REPORTED IN THE END-OF-PASS MESSAGE. FOR EXAMPLE:

END OF PASS #2 TOTAL ERRORS SINCE LAST REPORT 0

THAT WOULD INDICATE THAT PASS TWO WAS JUST COMPLETED AND NO ERRORS WERE DETECTED DURING THAT PASS. BOTH THE PASS NUMBER AND NUMBER OF ERRORS ARE DECIMAL NUMBERS.

4.3 T-BIT TRAPPING  
-----

THE 'T-BIT' (BIT 4) IN THE PROCESSOR STATUS WORD IS SET BY AN 'RTI' IN THE END-OF-PASS ROUTINE FOR EVERY OTHER PASS BEGINNING WITH THE THIRD PASS (PASSES 3,5,7,9...). T-BIT TRAPPING CAN BE INHIBITED BY SETTING BIT 12 = 1 IN THE SWITCH REGISTER (SEE SECTION 2.4).

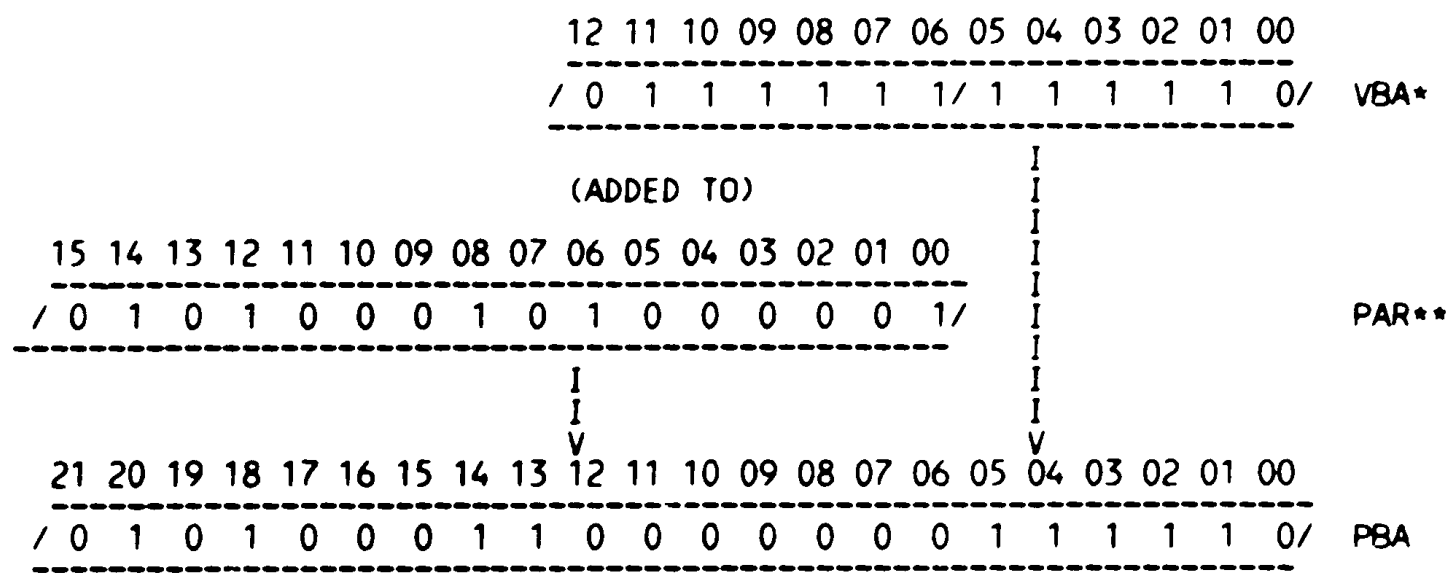
4.4 POWER FAILURE HANDLING  
-----

372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428

IF A POWER FAIL OCCURS (FOLLOWED BY A POWER UP), THE MESSAGE 'POWER FAILURE-RESTARTING' IS TYPED OUT AND THE PROGRAM WILL RESTART EXECUTION AT 'START:' (THE VERY BEGINNING OF THE PROGRAM. IF THE SOFTWARE SWITCH REGISTER IS BEING USED, ITS CONTENTS WILL BE RESTORED.

4.5 PHYSICAL BUS ADDRESS CONSTRUCTION  
-----

BELOW IS A SIMPLIFIED DIAGRAM OF HOW THE MEMORY MANAGEMENT LOGIC CONSTRUCTS A PHYSICAL BUS ADDRESS USING THE VIRTUAL ADDRESS AND THE PAGE ADDRESS REGISTER. THE PAGE DESCRIPTOR REGISTER SELECTED WILL CONTAIN THE PAGE EXPANSION, LENGTH, AND ACCESS INFORMATION.



\* VBA BITS <15:13> SELECT THE APPROPRIATE PAR AND PDR  
 \*\* PSW MODE BITS <15:14> SELECTS THE USER (=11), SUPERVISOR (=01) OR KERNEL (=00) SET OF PAR'S/PDR'S

5.0 PROGRAM DESCRIPTION  
-----

5.1 SUBROUTINES USED BY THIS PROGRAM  
-----

FOLLOWING IS A LIST OF THE SUBROUTINES AND HANDLERS USED BY THIS PROGRAM THAT ARE NOT PROVIDED BY THE 'SYSMAC PACKAGE'. DETAILS OF THE SUBROUTINES UNIQUE TO THIS PROGRAM MAY BE FOUND IN THE PROGRAM LISTING. REFER TO THE 'SYSMAC' DOCUMENT AND PROGRAM LISTING FOR THE OTHER ROUTINES.

1. TURN OFF T-BIT AND SAVE CURRENT PSW
2. TURN ON T-BIT AND RESTORE PREVIOUS PSW
3. SET ALL WRITEABLE BITS IN ALL PAR/PDR'S
4. CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS

429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479

NOTE ALSO THAT THE MACRO LIBRARY USED TO ASSEMBLE THIS PROGRAM HAS OTHER SPECIAL ROUTINES APPENDED TO THE SYSMAC MACRO PACKAGE; THIS LIBRARY MUST BE USED TO ASSEMBLE EITHER PART A OR PART B CORRECTLY.

5.2 PROGRAM LISTING  
-----

A TABLE OF CONTENTS APPEARS AT THE BEGINNING OF THE LISTING WHICH CONTAINS THE NAMES OF EACH SECTION, SUBTEST, AND ROUTINE AND THE LINE NUMBERS CORRESPONDING TO THE START OF EACH.

FOLLOWING THIS SECTION OF DOCUMENTATION IS THE ACTUAL PROGRAM LISTING COMPLETE WITH SUBTEST DESCRIPTIONS AND 'CODING COMMENTS'.

5.3 USING THE PROGRAM TO DIAGNOSE A FAULT  
-----

WHEN AN ERROR OCCURS, ONE OF THE THINGS THAT'S IMPORTANT TO NOTE IS WHAT PASS THE ERROR OCCURRED ON. IF THE PASS NUMBER IS ODD AND IS THREE OR GREATER, THE ERROR MIGHT BE T-BIT SENSITIVE. TRY RUNNING THE PROGRAM AGAIN WITH BIT 12 OF THE SWITCH REG. EQUAL TO '1' TO INHIBIT T-BIT TRAPPING. THIS SHOULD HELP YOU DETERMINE WHAT MAKES THE MACHINE FAIL AND WHEN.

IF YOU HAVE BEEN RUNNING WITH BIT 15 OF THE SWITCH REG. EQUAL TO '0', THEN YOU ARE ABLE TO LOOK AT ALL THE ERRORS THAT MAY BE RELATED TO THE FAULT YOU ARE DIAGNOSING. A FAULT IN AN EARLIER TEST MAY RESULT IN ERRORS DURING LATER TESTS WHICH MAY GIVE YOU MORE CLUES ABOUT THE NATURE OF THE FAULT. NOW USE THE METHOD OUTLINED IN SECTION 3.2 FOR EACH ERROR TO GATHER AS MUCH INFORMATION AS POSSIBLE.

NOW TO TEST YOUR IDEAS ON THE CAUSE OF THE FAILURE, YOU MAY WANT TO SCOPE THIS ERROR CONDITION. SET BIT 09 OF THE SWITCH REG. EQUAL TO '1' TO LOOP ON THE ERROR. FOR AN EVEN TIGHTER SCOPE LOOP THE ERROR CALL CAN BE REPLACED WITH A BRANCH (REFER TO COMMENTS BY ERROR CALLS IN THE PROGRAM LISTING).

OR YOU COULD LOOP ON THE TEST BY EITHER SETTING BIT 14 OF THE SWITCH REG. EQUAL TO '1' OF BY SETTING BIT 08 OF THE SWITCH REG. EQUAL TO '1' AND THEN SETTING THE TEST NUMBER IN BITS 07-00 OF THE SWITCH REG. YOU WILL PROBABLY WANT TO INHIBIT ERROR TYPEOUTS BY SETTING BIT 13 OF THE SWITCH REG. EQUAL TO '1'.



522  
523

```
.TITLE CKKTBB0 11/44 MEM MGMT PRT B
;*COPYRIGHT (C) 1979
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MASS. 01754
;*
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
;*

```

524

```
.SBTTL OPERATIONAL SWITCH SETTINGS
;*
;*      SWITCH          USE
;*      -----          -
;*      15             HALT ON ERROR
;*      14             LOOP ON TEST
;*      13             INHIBIT ERROR TYPEOUTS
;*      12             INHIBIT TRACE TRAP
;*      10             BELL ON ERROR
;*      9              LOOP ON ERROR
;*      8              LOOP ON TEST IN SWR<7:0>

```

525

001100  
104000  
000004

```
.SBTTL BASIC DEFINITIONS
;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100
      ERROR=EMT
      SCOPE=IOT

```

000011  
000012  
000015  
000200  
177776  
177776  
177774  
177772  
177570  
177570

```
;*MISCELLANEOUS DEFINITIONS
HT= 11          ;;CODE FOR HORIZONTAL TAB
LF= 12          ;;CODE FOR LINE FEED
CR= 15          ;;CODE FOR CARRIAGE RETURN
CRLF= 200       ;;CODE FOR CARRIAGE RETURN-LINE FEED
PS= 177776     ;;PROCESSOR STATUS WORD
      PSW=PS
STKLMT= 177774 ;;STACK LIMIT REGISTER
PIRQ= 177772   ;;PROGRAM INTERRUPT REQUEST REGISTER
DSWR= 177570   ;;HARDWARE SWITCH REGISTER
DDISP= 177570  ;;HARDWARE DISPLAY REGISTER

```

000000  
000001  
000002  
000003  
000004  
000005  
000006  
000007  
000006  
000007

```
;*GENERAL PURPOSE REGISTER DEFINITIONS
R0= %0          ;;GENERAL REGISTER
R1= %1          ;;GENERAL REGISTER
R2= %2          ;;GENERAL REGISTER
R3= %3          ;;GENERAL REGISTER
R4= %4          ;;GENERAL REGISTER
R5= %5          ;;GENERAL REGISTER
R6= %6          ;;GENERAL REGISTER
R7= %7          ;;GENERAL REGISTER
SP= %6          ;;STACK POINTER
PC= %7          ;;PROGRAM COUNTER

```

000000  
000040  
000100  
000140  
000200  
000240  
000300  
000340

```
;*PRIORITY LEVEL DEFINITIONS
PR0= 0          ;;PRIORITY LEVEL 0
PR1= 40         ;;PRIORITY LEVEL 1
PR2= 100        ;;PRIORITY LEVEL 2
PR3= 140        ;;PRIORITY LEVEL 3
PR4= 200        ;;PRIORITY LEVEL 4
PR5= 240        ;;PRIORITY LEVEL 5
PR6= 300        ;;PRIORITY LEVEL 6
PR7= 340        ;;PRIORITY LEVEL 7

```

```

100000
040000
020000
010000
004000
002000
001000
000400
000200
000100
000040
000020
000010
000004
000002
000001
001000
000400
000200
000100
000040
000020
000010
000004
000002
000001
100000
040000
020000
010000
004000
002000
001000
000400
000200
000100
000040
000020
000010
000004
000002
000001
001000
000400
000200
000100
000040
000020
000010
000004
000002
000001
000004
000010

;*'SWITCH REGISTER' SWITCH DEFINITIONS
SW15= 100000
SW14= 40000
SW13= 20000
SW12= 10000
SW11= 4000
SW10= 2000
SW09= 1000
SW08= 400
SW07= 200
SW06= 100
SW05= 40
SW04= 20
SW03= 10
SW02= 4
SW01= 2
SW00= 1
SW9=SW09
SW8=SW08
SW7=SW07
SW6=SW06
SW5=SW05
SW4=SW04
SW3=SW03
SW2=SW02
SW1=SW01
SW0=SW00

;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
BIT15= 100000
BIT14= 40000
BIT13= 20000
BIT12= 10000
BIT11= 4000
BIT10= 2000
BIT09= 1000
BIT08= 400
BIT07= 200
BIT06= 100
BIT05= 40
BIT04= 20
BIT03= 10
BIT02= 4
BIT01= 2
BIT00= 1
BIT9=BIT09
BIT8=BIT08
BIT7=BIT07
BIT6=BIT06
BIT5=BIT05
BIT4=BIT04
BIT3=BIT03
BIT2=BIT02
BIT1=BIT01
BIT0=BIT00

;*BASIC 'CPU' TRAP VECTOR ADDRESSES
ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
```

```

000014 TBITVEC=14 ;:'T' BIT
000014 TRTVEC= 14 ;:TRACE TRAP
000014 BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)
000020 IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024 PWRVEC= 24 ;:POWER FAIL
000030 EMTVEC= 30 ;:EMULATOR TRAP (EMT) **ERROR**
000034 TRAPVEC=34 ;:'TRAP' TRAP
000060 TKVEC= 60 ;:TTY KEYBOARD VECTOR
000064 TPVEC= 64 ;:TTY PRINTER VECTOR
000240 PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR
526 .SBTTL MEMORY MANAGEMENT DEFINITIONS
      ;*KT11 VECTOR ADDRESS
000250 MMVEC= 250
      ;*KT11 STATUS REGISTER ADDRESSES
177572 SR0= 177572
177574 SR1= 177574
177576 SR2= 177576
172516 SR3= 172516
      ;*USER 'I' PAGE DESCRIPTOR REGISTERS
177600 UIPDR0= 177600
177602 UIPDR1= 177602
177604 UIPDR2= 177604
177606 UIPDR3= 177606
177610 UIPDR4= 177610
177612 UIPDR5= 177612
177614 JIPDR6= 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                      |
|     |        | :*ADDITIONAL DEFINITIONS            |
|     |        | :*                                  |
| 527 |        |                                     |
| 528 |        |                                     |
| 529 | 177572 | MMR0=SR0                            |
| 530 | 177574 | MMR1=SR1                            |
| 531 | 177576 | MMR2=SR2                            |
| 532 | 172516 | MMR3=SR3                            |
| 533 | 000006 | KSP=SP                              |
| 534 | 000006 | SSP=SP                              |
| 535 | 000006 | USP=SP                              |
| 536 | 000020 | TBIT=BIT4                           |
| 537 | 000100 | WBIT=BIT6                           |
| 538 | 177766 | CPUERR=177766                       |
| 539 | 001100 | KERSTK=STACK                        |
| 540 | 000700 | SUPSTK=STACK-200                    |
| 541 | 000600 | USESTK=STACK-300                    |
| 542 |        |                                     |

```

574          000000          .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      000174          DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
000176      000000          SWREG:   .WORD 0          ;;SOFTWARE SWITCH REGISTER
000200      000137  020000  .SBTTL STARTING ADDRESS(ES)
                                JMP     @START ;;JUMP TO STARTING ADDRESS OF PROGRAM
575          .SBTTL ACT11 HOOKS
                                ;*****
                                ;HOOKS REQUIRED BY ACT11
                                $SVPC=.          ;SAVE PC
                                .=46
000046      037530          $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                                .=52
000052      000000          .WORD 0          ;;2)SET LOC.52 TO ZERO
                                .= $SVPC          ;; RESTORE PC
576          .SBTTL APT PARAMETER BLOCK
                                ;*****
                                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                ;*****
                                .$X=.          ;;SAVE CURRENT LOCATION
                                .=24          ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000024      000200          200          ;;FOR APT START UP
                                .=44          ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044      000204          $APTHDR ;;POINT TO APT HEADER BLOCK
                                --.$X          ;;RESET LOCATION COUNTER
                                ;*****
                                ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                ;INTERFACE SPEC.
000204      $APTHD:
000204      000000          $HIBTS: .WORD 0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
000206      001224          $MBADR: .WORD $MAIL      ;;ADDRESS OF APT MAILBOX (BITS 0-15)
000210      000002          $TSTM:  .WORD 2          ;;RUN TIM OF LONGEST TEST
000212      000005          $PASTM: .WORD 5          ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
000214      000005          $UNITM: .WORD 5          ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
000216      000014          .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)

```



577

.SBTTL COMMON TAGS

\*\*\*\*\*  
\*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS  
\*USED IN THE PROGRAM.

|        |        |            |        |                 |   |
|--------|--------|------------|--------|-----------------|---|
| 001100 | 001100 | \$CMTAG:   | .WORD  | 0               | ::START OF COMMON TAGS                                  |
| 001100 | 000000 | \$TSTNM:   | .BYTE  | 0               | ::CONTAINS THE TEST NUMBER                              |
| 001102 | 000    | \$ERFLG:   | .BYTE  | 0               | ::CONTAINS ERROR FLAG                                   |
| 001103 | 000    | \$ICNT:    | .WORD  | 0               | ::CONTAINS SUBTEST ITERATION COUNT                      |
| 001104 | 000000 | \$LPADR:   | .WORD  | 0               | ::CONTAINS SCOPE LOOP ADDRESS                           |
| 001106 | 000000 | \$LPERR:   | .WORD  | 0               | ::CONTAINS SCOPE RETURN FOR ERRORS                      |
| 001110 | 000000 | \$ERTTL:   | .WORD  | 0               | ::CONTAINS TOTAL ERRORS DETECTED                        |
| 001112 | 000000 | \$ITEMB:   | .BYTE  | 0               | ::CONTAINS ITEM CONTROL BYTE                            |
| 001114 | 000    | \$ERMAX:   | .BYTE  | 1               | ::CONTAINS MAX. ERRORS PER TEST                         |
| 001115 | 001    | \$ERRPC:   | .WORD  | 0               | ::CONTAINS PC OF LAST ERROR INSTRUCTION                 |
| 001116 | 000000 | \$GDADR:   | .WORD  | 0               | ::CONTAINS ADDRESS OF 'GOOD' DATA                       |
| 001120 | 000000 | \$BDADR:   | .WORD  | 0               | ::CONTAINS ADDRESS OF 'BAD' DATA                        |
| 001122 | 000000 | \$GDDAT:   | .WORD  | 0               | ::CONTAINS 'GOOD' DATA                                  |
| 001124 | 000000 | \$BDDAT:   | .WORD  | 0               | ::CONTAINS 'BAD' DATA                                   |
| 001126 | 000000 |            | .WORD  | 0               | ::RESERVED--NOT TO BE USED                              |
| 001130 | 000000 |            | .WORD  | 0               |   |
| 001132 | 000000 |            | .WORD  | 0               |   |
| 001134 | 000    | \$AUTOB:   | .BYTE  | 0               | ::AUTOMATIC MODE INDICATOR                              |
| 001135 | 000    | \$INTAG:   | .BYTE  | 0               | ::INTERRUPT MODE INDICATOR                              |
| 001136 | 000000 |            | .WORD  | 0               |   |
| 001140 | 177570 | \$SWR:     | .WORD  | DSWR            | ::ADDRESS OF SWITCH REGISTER                            |
| 001142 | 177570 | \$DISPLAY: | .WORD  | DDISP           | ::ADDRESS OF DISPLAY REGISTER                           |
| 001144 | 177560 | \$TKS:     | 177560 |                 | ::TTY KBD STATUS  |
| 001146 | 177562 | \$TKB:     | 177562 |                 | ::TTY KBD BUFFER  |
| 001150 | 177564 | \$TPS:     | 177564 |                 | ::TTY PRINTER STATUS REG. ADDRESS                       |
| 001152 | 177566 | \$TPB:     | 177566 |                 | ::TTY PRINTER BUFFER REG. ADDRESS                       |
| 001154 | 000    | \$NULL:    | .BYTE  | 0               | ::CONTAINS NULL CHARACTER FOR FILLS                     |
| 001155 | 002    | \$FILLS:   | .BYTE  | 2               | ::CONTAINS # OF FILLER CHARACTERS REQUIRED              |
| 001156 | 012    | \$FILLC:   | .BYTE  | 12              | ::INSERT FILL CHARS. AFTER A 'LINE FEED'                |
| 001157 | 000    | \$TPFLG:   | .BYTE  | 0               | ::'TERMINAL AVAILABLE' FLAG (BIT<07>-0=YES)             |
| 001160 | 000000 | \$REGAD:   | .WORD  | 0               | ::CONTAINS THE ADDRESS FROM WHICH (\$REG0) WAS OBTAINED |
| 001162 | 000006 |            | .REPT  | \$CM3           |   |
| 001162 | 000000 | \$REG0:    | .WORD  | 0               | ::CONTAINS ((\$REGAD)+0)                                |
| 001164 | 000000 | \$REG1:    | .WORD  | 0               | ::CONTAINS ((\$REGAD)+2)                                |
| 001166 | 000000 | \$REG2:    | .WORD  | 0               | ::CONTAINS ((\$REGAD)+4)                                |
| 001170 | 000000 | \$REG3:    | .WORD  | 0               | ::CONTAINS ((\$REGAD)+6)                                |
| 001172 | 000000 | \$REG4:    | .WORD  | 0               | ::CONTAINS ((\$REGAD)+10)                               |
| 001174 | 000000 | \$REG5:    | .WORD  | 0               | ::CONTAINS ((\$REGAD)+12)                               |
| 001176 | 000006 |            | .REPT  | 6               |   |
| 001200 | 000000 | \$TMP0:    | .WORD  | 0               | ::USER DEFINED  |
| 001202 | 000000 | \$TMP1:    | .WORD  | 0               | ::USER DEFINED  |
| 001204 | 000000 | \$TMP2:    | .WORD  | 0               | ::USER DEFINED  |
| 001206 | 000000 | \$TMP3:    | .WORD  | 0               | ::USER DEFINED  |
| 001210 | 000000 | \$TMP4:    | .WORD  | 0               | ::USER DEFINED  |
| 001212 | 000000 | \$TMP5:    | .WORD  | 0               | ::USER DEFINED  |
| 001214 | 207    | \$ESCAPE:  | 0      |                 | ::ESCAPE ON ERROR ADDRESS                               |
| 001217 | 000    | \$BELL:    | .ASCIZ | <207><377><377> | ::CODE FOR BELL   |
| 001220 | 077    | \$QUES:    | .ASCII | /?/             | ::QUESTION MARK   |
| 001221 | 015    | \$CRLF:    | .ASCII | <15>            | ::CARRIAGE RETURN                                       |
| 001222 | 012    | \$LF:      | .ASCIZ | <12>            | ::LINE FEED   |

```

*****
:SBTTL  APT MAILBOX-ETABLE
*****
.EVEN
001224 $MAIL:          ;; APT MAILBOX
001224 000000 $MSGTY: .WORD  AMSTY  ;; MESSAGE TYPE CODE
001226 000000 $FATAL: .WORD  AFATAL ;; FATAL ERROR NUMBER
001230 000000 $TESTN: .WORD  ATESTN ;; TEST NUMBER
001232 000000 $PASS:  .WORD  APASS   ;; PASS COUNT
001234 000000 $DEVCT: .WORD  ADEVCT  ;; DEVICE COUNT
001236 000000 $UNIT:  .WORD  AUNIT   ;; I/O UNIT NUMBER
001240 000000 $MSGAD: .WORD  AMSGAD  ;; MESSAGE ADDRESS
001242 000000 $MSGLG: .WORD  AMSGLG  ;; MESSAGE LENGTH
001244 $ETABLE:      ;; APT ENVIRONMENT TABLE
001244      000 $ENV:  .BYTE  AENV   ;; ENVIRONMENT BYTE
001245      000 $ENVM: .BYTE  AENVM  ;; ENVIRONMENT MODE BITS
001246 000000 $SWREG: .WORD  ASWREG  ;; APT SWITCH REGISTER
001250 000000 $USWR:  .WORD  AUSWR   ;; USER SWITCHES
001252 000000 $CPUOP: .WORD  ACPUOP  ;; CPU TYPE, OPTIONS
: *
: *          BITS 15-11=CPU TYPE
: *          11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
: *          11/70=06,PDQ=07,Q=10
: *
: *          BIT 10=REAL TIME CLOCK
: *          BIT 9=FLOATING POINT PROCESSOR
: *          BIT 8=MEMORY MANAGEMENT
: *
001254 $ETEND:
.MEXIT

001254 000000 TESTNO: .WORD  0          ; HOLDS TEST NUMBER FOR TYPEOUTS
001256 000000 WASR6:  .WORD  0          ; USED TO STORE THE STACK POINTER AFTER A TRAP
001260 000000 TRAPPL: .WORD  0          ; USED TO STORE THE PC OF A TRAP OR ABORT
001262 000000 TRAPPS: .WORD  0          ; USED TO STORE THE PS OF A TRAP OR ABORT
001264 000000 WASSR0: .WORD  0          ; USED TO STORE CONTENTS OF SR0
001266 000000 WASSR1: .WORD  0          ; USED TO STORE CONTENTS OF SR1
001270 000000 WASSR2: .WORD  0          ; USED TO STORE CONTENTS OF SR2
001272 000000 WASSR3: .WORD  0          ; USED TO STORE CONTENTS OF SR3
001274 000000 TBITPS: .WORD  0          ; SAVES THE PSW THAT MAY HAVE ITS T-BIT ON
001276 000000 VIRT1:  .WORD  0          ; HOLDS VIRTUAL ADDRESS TO BE CONVERTED
001300 000000 PBALO:  .WORD  0          ; HOLDS BITS <15:00> OF PHYSICAL ADDRESS
001302 000000 PBAHI:  .WORD  0          ; HOLDS BITS <21:16> OF PHYSICAL ADDRESS
001304 000000 BADPC:  .WORD  0          ; HOLDS PC FROM ABORT OR TRAP
001306 000200 $MXCNT: .WORD  200        ; HOLD MAX. NUMBER OF LOOP ITERATIONS
001310 000000 $TBIT:  .WORD  0          ; 'T' BIT STATE INDICATOR
001312      136      103      015 $CNTLC: .ASCIZ  /^C/<15><12> ; CONTROL C
001315      012      000
.EVEN

```

```
.SBTTL ERROR POINTER TABLE
;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
;*      EM      ;;POINTS TO THE ERROR MESSAGE
;*      DH      ;;POINTS TO THE DATA HEADER
;*      DT      ;;POINTS TO THE DATA
;*      DF      ;;POINTS TO THE DATA FORMAT
```

```
001320 $ERRTB:
578
579
580 001320 043736      ;*ITEM 1      EM1      ;UNEXPECTED CPU TRAP TO LOC. 004
581 001322 046544      ;              DH1      ;OLD PC OLD PSW R6 WAS CPUERR TESTNO ERRORPC
582 001324 051120      ;              DT1      ;TRAPPC,TRAPPS,WASR6,CPUERR,TESTNO,$ERRPC,0
583 001326 051647      ;              DF12     ;0.0,0.0,0.0
584
585
586 001330 043776      ;*ITEM 2      EM2      ;UNEXPECTED MEM. MGMT. TRAP TO LOC. 250
587 001332 046624      ;              DH2      ;OLD PC OLD PSW R6 WAS SRO SR2 TESTNO ERRORPC
588 001334 051136      ;              DT2      ;TRAPPC, TRAPPS, WASR6, WASSRO, WASSR2, TESTNO, $ERRPC, 0
589 001336 051631      ;              DF2      ;0.0,0.0,0.0,0.0
590
591
592 001340 044045      ;*ITEM 3      EM10     ;MEMORY MGMT. ACCESS ABORT DID NOT OCCUR
593 001342 046714      ;              DH10     ;PDR 4 PSW TESTNO ERRORPC
594 001344 051156      ;              DT10     ;$REG2,$TMP0,TESTNO,$ERRPC,0
595 001346 051640      ;              DF3      ;0.0,0.0
596
597
598 001350 044115      ;*ITEM 4      EM11     ;ACCESS ERROR DID NOT ABORT INSTRUCTION
599 001352 046714      ;              DH10     ;PDR 4 PSW TESTNO ERRORPC
600 001354 051156      ;              DT10     ;$REG2,$TMP0,TESTNO,$ERRPC,0
601 001356 051640      ;              DF3      ;0.0,0.0
602
603
604 001360 044164      ;*ITEM 5      EM12     ;SRO DID NOT REPORT ACCESS ERROR CORRECTLY
605 001362 046754      ;              DH12     ;SRO WAS EXPECTD PDR 4 PSW TESTNO ERRORPC
606 001364 051170      ;              DT12     ;WASSRO,$REG3,$REG2,$TMP0,TESTNO,$ERRPC,0
607 001366 051647      ;              DF12     ;0.0,0.0,0.0
608
609
610 001370 044236      ;*ITEM 6      EM13     ;SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR.
611 001372 047034      ;              DH13     ;SR2 WAS EXPECTD PDR 4 PSW TESTNO ERRORPC
612 001374 051206      ;              DT13     ;WASSR2,$REG4,$REG2,$TMP0,TESTNO,$ERRPC,0
613 001376 051647      ;              DF12     ;0.0,0.0,0.0
614
615
616 001400 044307      ;*ITEM 7      EM14     ;PAGE LGTH. ABORT OCCURRED WHEN IT SHOULDN'T HAVE
617 001402 047114      ;              DH14     ;V.B.A. KIPDR4 SRO WAS SR2 WAS TESTNO ERRORPC
618 001404 051224      ;              DT14     ;$REG0,$REG4,WASSRO,WASSR2,TESTNO,$ERRPC,0
619 001406 051647      ;              DF12     ;0.0,0.0,0.0
620
621
622 001410 044370      ;*ITEM 10     EM15     ;PAGE LGTH. ABORT DID NOT OCCUR WHEN IT SHOULD HAVE
623 001412 047174      ;              DH15     ;V.B.A. KIPDR4 TESTNO ERRORPC
```



|     |        |        |           |   |
|-----|--------|--------|-----------|---|
| 624 | 001414 | 051242 | DT15      | ;\$REG0,\$REG4,TESTNO,\$ERRPC,0                     |
| 625 | 001416 | 051640 | DF3       | :0,0,0,0  |
| 626 |        |        |           |   |
| 627 |        |        | ;*ITEM 11 |   |
| 628 | 001420 | 044453 | EM16      | ;\$R0 DID NOT REPORT PAGE LGTH. ABORT CORRECTLY     |
| 629 | 001422 | 047234 | DH16      | ;\$V.B.A. KIPDR4 \$R0 WAS EXPECTD TESTNO ERRORPC    |
| 630 | 001424 | 051254 | DT16      | ;\$REG0,\$REG4,WASSR0,\$REG2,TESTNO,\$ERRPC,0       |
| 631 | 001426 | 051647 | DF12      | :0,0,0,0,0,0  |
| 632 |        |        |           |   |
| 633 |        |        | ;*ITEM 12 |   |
| 634 | 001430 | 044236 | EM13      | ;\$R2 DID NOT LOCKUP CORRECT VIRUAL ADDR.           |
| 635 | 001432 | 047314 | DH17      | ;\$V.B.A. KIPDR4 \$R2 WAS EXPECTD TESTNO ERRORPC    |
| 636 | 001434 | 051272 | DT17      | ;\$REG0,\$REG4,WASSR2,\$REG3,TESTNO,\$ERRPC,0       |
| 637 | 001436 | 051647 | DF12      | :0,0,0,0,0,0  |
| 638 |        |        |           |   |
| 639 |        |        | ;*ITEM 13 |   |
| 640 | 001440 | 044236 | EM13      | ;\$R2 DID NOT LOCKUP CORRECT VIRUAL ADDR.           |
| 641 | 001442 | 047374 | DH20      | ;\$R2 WAS EXPECTD TESTNO ERRORPC                    |
| 642 | 001444 | 051310 | DT20      | ;\$WASSR2,\$REG1,TESTNO,\$ERRPC,0                   |
| 643 | 001446 | 051640 | DF3       | :0,0,0,0  |
| 644 |        |        |           |   |
| 645 |        |        | ;*ITEM 14 |   |
| 646 | 001450 | 044531 | EM21      | ;\$R0 OR \$R2 CHANGED BY A SECOND ABORT             |
| 647 | 001452 | 047434 | DH21      | ;\$FIRST ABORT SECOND ABORT                         |
| 648 |        |        |           | ;\$R0 WAS \$R2 WAS \$R0 WAS \$R2 WAS TESTNO ERRORPC |
| 649 | 001454 | 051322 | DT21      | ;\$STMP0,\$STMP2,WASSR0,WASSR2,TESTNO,\$ERRPC,0     |
| 650 | 001456 | 051647 | DF12      | :0,0,0,0,0,0  |
| 651 |        |        |           |   |
| 652 |        |        | ;*ITEM 15 |   |
| 653 | 001460 | 044576 | EM22      | ;\$R0 OR \$R2 WAS NOT 'RESET' BY A RESET            |
| 654 | 001462 | 047551 | DH22      | ;\$R0 WAS \$R2 WAS TESTNO ERRORPC                   |
| 655 | 001464 | 051340 | DT22      | ;\$WASSR0,WASSR2,TESTNO,\$ERRPC,0                   |
| 656 | 001466 | 051640 | DF3       | :0,0,0,0  |
| 657 |        |        |           |   |
| 658 |        |        | ;*ITEM 16 |   |
| 659 | 001470 | 044645 | EM23      | ;\$R2 NOT TRACKING CORRECTLY                        |
| 660 | 001472 | 047374 | DH20      | ;\$R2 WAS EXPECTD TESTNO ERRORPC                    |
| 661 | 001474 | 051310 | DT20      | ;\$WASSR2,\$REG1,TESTNO,\$ERRPC,0                   |
| 662 | 001476 | 051640 | DF3       | :0,0,0,0  |
| 663 |        |        |           |   |
| 664 |        |        | ;*ITEM 17 |   |
| 665 | 001500 | 044700 | EM24      | ;\$DID NOT TRAP THRU KERNEL SPACE                   |
| 666 | 001502 | 047611 | DH24      | ;\$PSW WAS R6 WAS TESTNO ERRORPC                    |
| 667 | 001504 | 051352 | DT24      | ;\$REG1,\$REG2,TESTNO,\$ERRPC,0                     |
| 668 | 001506 | 051640 | DF3       | :0,0,0,0  |
| 669 |        |        |           |   |
| 670 |        |        | ;*ITEM 20 |   |
| 671 | 001510 | 044737 | EM25      | ;\$KT ERROR SERVICED ON ODD ADDR. ERROR             |
| 672 | 001512 | 047374 | DH20      | ;\$PDR TESTNO ERRORPC                               |
| 673 | 001514 | 051310 | DT20      | ;\$REG5,TESTNO,\$ERRPC,0                            |
| 674 | 001516 | 051644 | DF5       | :0,0,0  |
| 675 |        |        |           |   |
| 676 |        |        | ;*ITEM 21 |   |
| 677 | 001520 | 045004 | EM26      | ;\$R0 OR \$R2 CHANGED BY ODD ADDR. ERROR            |
| 678 | 001522 | 047651 | DH26      | ;\$EXPECTED RECEIVED                                |
| 679 |        |        |           | ;\$R0 \$R2 \$R0 WAS \$R2 WAS TESTNO ERRORPC         |
| 680 | 001524 | 051364 | DT26      | ;\$REG0,\$REG1,WASSR0,WASSR2,TESTNO,\$ERRPC,0       |

|     |        |        |           |   |
|-----|--------|--------|-----------|---|
| 681 | 001526 | 051647 | DF12      | :0,0,0,0,0,0  |
| 682 |        |        |           |   |
| 683 |        |        | ;*ITEM 22 |   |
| 684 | 001530 | 045052 | EM27      | :ERROR DURING 'DOUBLE ERROR' (KT & ODD ADDR.)         |
| 685 | 001532 | 047763 | DH27      | :EXPECTED:  |
| 686 |        |        |           | :PSW PC SRO SR2                                       |
| 687 |        |        |           | :170017 (3\$+4) 020147 (3\$)                          |
| 688 |        |        |           | :RECEIVED   |
| 689 |        |        |           | :PSW PC SRO SR2 TESTNO ERRORPC                        |
| 690 | 001534 | 051402 | DT27      | :\$REG1,\$REG3,WASSRO,WASSR2,TESTNO,\$ERRPC,0         |
| 691 | 001536 | 051647 | DF12      | :0,0,0,0,0,0  |
| 692 |        |        |           |   |
| 693 |        |        | ;*ITEM 23 |   |
| 694 | 001540 | 045127 | EM30      | :MFPI INSTRUCTION PUSHED WRONG DATA                   |
| 695 | 001542 | 050160 | DH30      | :DATA DATA  |
| 696 |        |        |           | :EXPECTD RECEIVD TESTNO ERRORPC                       |
| 697 | 001544 | 051420 | DT30      | :\$REG0,\$REG1,TESTNO,\$ERRPC,0                       |
| 698 | 001546 | 051640 | DF3       | :0,0,0,0  |
| 699 |        |        |           |   |
| 700 |        |        | ;*ITEM 24 |   |
| 701 | 001550 | 045172 | EM31      | :MTPI INSTRUCTION LOADED WRONG DATA                   |
| 702 | 001552 | 050160 | DH30      | :DATA DATA  |
| 703 |        |        |           | :EXPECTD RECEIVD TESTNO ERRORPC                       |
| 704 | 001554 | 051420 | DT30      | :\$REG0,\$REG1,TESTNO,\$ERRPC,0                       |
| 705 | 001556 | 051640 | DF3       | :0,0,0,0  |
| 706 |        |        |           |   |
| 707 |        |        | ;*ITEM 25 |   |
| 708 | 001560 | 045235 | EM32      | :STACK NOT PUSHED BY MFPI-MTPI                        |
| 709 | 001562 | 050246 | DH32      | :TESTNO ERRORPC                                       |
| 710 | 001564 | 051434 | DT32      | :TESTNO,\$ERRPC,0                                     |
| 711 | 001566 | 051655 | DF32      | :0,0  |
| 712 |        |        |           |   |
| 713 |        |        | ;*ITEM 26 |   |
| 714 | 001570 | 045273 | EM33      | :KERNEL PAGE ACCESSED INSTEAD OF USER: MFPI-MTPI      |
| 715 | 001572 | 050277 | DH33      | :SRO WAS SR2 WAS TESTNO ERRORPC                       |
| 716 | 001574 | 051340 | DT22      | :WASSRO,WASSR2,TESTNO,\$ERRPC,0                       |
| 717 | 001576 | 051640 | DF3       | :0,0,0,0  |
| 718 |        |        |           |   |
| 719 |        |        | ;*ITEM 27 |   |
| 720 | 001600 | 045351 | EM34      | :M.M. ABORT IN KERNAL D-SPACE HAD WRONG CONDITION     |
| 721 | 001602 | 050337 | DH34      | :(MMR0) (MMR1) (MMR2) TESTNO ERRORPC EXPECTING 020031 |
| 722 | 001604 | 051444 | DT34      | :\$REG1,\$REG2,\$REG3,TESTNO,\$ERRPC,0                |
| 723 | 001606 | 051624 | DF1       | :0,0,0,0,0  |
| 724 |        |        |           |   |
| 725 |        |        | ;*ITEM 30 |   |
| 726 | 001610 | 045432 | EM35      | :ILLEGAL MODE 10 NOT ABORTED                          |
| 727 | 001612 | 050246 | DH32      | :TESTNO ERRORPC                                       |
| 728 | 001614 | 051434 | DT32      | :TESTNO,\$ERRPC,0                                     |
| 729 | 001616 | 051655 | DF32      | :0,0  |
| 730 |        |        |           |   |
| 731 |        |        | ;*ITEM 31 |   |
| 732 | 001620 | 045466 | EM36      | :SRO DID NOT REPORT ILLEGAL MODE 10 CORRECTLY         |
| 733 | 001622 | 050430 | DH36      | :SRO WAS EXPECTD TESTNO ERRORPC                       |
| 734 | 001624 | 051460 | DT36      | :WASSRO,\$REG1,TESTNO,\$ERRPC,0                       |
| 735 | 001626 | 051640 | DF3       | :0,0,0,0  |
| 736 |        |        |           |   |
| 737 |        |        | ;*ITEM 32 |   |

|     |        |        |           |  |
|-----|--------|--------|-----------|--|
| 738 | 001630 | 045543 | EM37      | :PSW CHANGED BY AN RTI IN USER MODE                    |
| 739 | 001632 | 050470 | DH37      | :PSW WAS EXPECTD TESTNO ERRORPC                        |
| 740 | 001634 | 051352 | DT24      | :\$REG1,\$REG2,TESTNO,\$ERRPC,0                        |
| 741 | 001636 | 051640 | DF3       | :0,0,0,0   |
| 742 |        |        |           |  |
| 743 |        |        | ;*ITEM 33 |  |
| 744 | 001640 | 045606 | EM40      | :ABORT IN KERNAL D-SPACE PICKED UP VECTOR FROM I-SPACE |
| 745 | 001642 | 050530 | DH40      | :(PSW) TESTNO ERRORPC EXPECTING XXX340                 |
| 746 | 001644 | 051472 | DT40      | :\$REG0,TESTNO,\$ERRPC,0                               |
| 747 | 001646 | 051644 | DF5       | :0,0,0   |
| 748 |        |        |           |  |
| 749 |        |        | ;*ITEM 34 |  |
| 750 | 001650 | 045673 | EM41      | :D SPACE ENABLE CIRCUITRY HAS FAILED                   |
| 751 | 001652 | 050601 | DH41      | :ERROR AUTOI/D VIRTUAL                                 |
| 752 |        |        |           | :REGISTR REGISTR ADDRESS TESTNO PC AT ABORT            |
| 753 | 001654 | 051502 | DT41      | :WASSR0,WASSR1,WASSR2,TESTNO,BADPC,0                   |
| 754 | 001656 | 051624 | DF1       | :0,0,0,0,0   |
| 755 |        |        |           |  |
| 756 |        |        | ;*ITEM 35 |  |
| 757 | 001660 | 045737 | EM42      | :INCORRECT STORE BY MTP INSTRUCTION                    |
| 758 | 001662 | 050705 | DH42      | :GDDATA STORED TESTNO ERRORPC                          |
| 759 | 001664 | 051516 | DT42      | :\$REG3,\$REG4,TESTNO,\$ERRPC,0                        |
| 760 | 001666 | 051640 | DF3       | :0,0,0,0   |
| 761 |        |        |           |  |
| 762 |        |        | ;*ITEM 36 |  |
| 763 | 001670 | 046002 | EM43      | :TRIED TO REFERENCE NON-RESIDENT PAGE                  |
| 764 | 001672 | 050745 | DH43      | :(MMR0) (MMR1) (MMR2) TESTNO ERRORPC                   |
| 765 | 001674 | 051530 | DT43      | :\$REG0,\$REG1,\$REG2,TESTNO,\$ERRPC,0                 |
| 766 | 001676 | 051624 | DF1       |  |
| 767 |        |        |           |  |
| 768 |        |        | ;*ITEM 37 |  |
| 769 | 001700 | 046047 | EM44      | :WRONG DATA FETCHED BY MFP INSTRUCTION                 |
| 770 | 001702 | 050160 | DH30      | :DATA DATA   |
| 771 |        |        |           | :EXPECTD RECEIVD TESTNO ERRORPC                        |
| 772 | 001704 | 051420 | DT30      | :\$REG0,\$REG1,TESTNO,\$ERRPC,0                        |
| 773 | 001706 | 051640 | DF3       | :0,0,0,0   |
| 774 |        |        |           |  |
| 775 |        |        | ;*ITEM 40 |  |
| 776 | 001710 | 046002 | EM43      | :TRIED TO REFERENCE NON-RESIDENT PAGE                  |
| 777 | 001712 | 050745 | DH43      | :(MMR0) (MMR1) (MMR2) TESTNO ERRORPC                   |
| 778 | 001714 | 051544 | DT45      | :WASSR0,WASSR1,WASSR2,TESTNO,\$ERRPC,0                 |
| 779 | 001716 | 051624 | DF1       | :0,0,0,0,0   |
| 780 |        |        |           |  |
| 781 |        |        | ;*ITEM 41 |  |
| 782 | 001720 | 046115 | EM45      | :ILLEGAL CSM DID NOT TRAP TO 10                        |
| 783 | 001722 | 050246 | DH32      | :TESTNO ERRORPC  |
| 784 | 001724 | 051434 | DT32      | :TESTNO,\$ERRPC,0                                      |
| 785 | 001726 | 051631 | DF2       | :0,0   |
| 786 |        |        |           |  |
| 787 |        |        | ;*ITEM 42 |  |
| 788 | 001730 | 046154 | EM46      | :CSM DID NOT ENTER SUPERVISOR MODE                     |
| 789 | 001732 | 051015 | DH44      | :EXPECTD (PSW) TESTNO ERR PC SUB CALL                  |
| 790 | 001734 | 051560 | DT46      | :\$REG3,\$TMP4,TESTNO,\$ERRPC,SRCAL,0                  |
| 791 | 001736 | 051624 | DF1       | :0,0,0,0,0   |
| 792 |        |        |           |  |
| 793 |        |        | ;*ITEM 43 |  |
| 794 | 001740 | 046216 | EM47      | :CSM SET UP WRONG PREVIOUS MODE                        |

|     |        |        |            |  |
|-----|--------|--------|------------|--|
| 795 | 001742 | 051015 | DH44       | : EXPECTD (PSW) TESTNO ERR PC SUB CALL   |
| 796 | 001744 | 051560 | DT46       | : \$REG3,\$TMP4,TESTNO,\$ERRPC,SRCALL,0  |
| 797 | 001746 | 051624 | DF1        | : 0,0,0,0,0                              |
| 798 |        |        |            |  |
| 799 |        |        | : *ITEM 44 |  |
| 800 | 001750 | 046255 | EM50       | : CSM SET UP STACK WRONG                 |
| 801 | 001752 | 050160 | DH30       | : DATA DATA                              |
| 802 |        |        |            | : EXPECTD RECEIVD TESTNO ERR PC SUB CALL |
| 803 | 001754 | 051420 | DT30       | : \$REG5,\$TMP4,TESTNO,\$ERRPC,SRCALL,0  |
| 804 | 001756 | 051624 | DF1        | : 0,0,0,0,0                              |
| 805 |        |        |            |  |
| 806 |        |        | : *ITEM 45 |  |
| 807 | 001760 | 046304 | EM51       | : CSM PUSHED INCORRECT ARGUMENT          |
| 808 | 001762 | 050160 | DH30       | : DATA DATA                              |
| 809 |        |        |            | : EXPECTD RECEIVD TESTNO ERR PC SUB CALL |
| 810 | 001764 | 051574 | DT47       | : \$REG0,\$REG1,TESTNO,\$ERRPC,SRCALL,0  |
| 811 | 001766 | 051624 | DF1        | : 0,0,0,0,0                              |
| 812 |        |        |            |  |
| 813 |        |        | : *ITEM 46 |  |
| 814 | 001770 | 046342 | EM52       | : CSM PUSHED WRONG PC                    |
| 815 | 001772 | 050160 | DH30       | : DATA DATA                              |
| 816 |        |        |            | : EXPECTD RECEIVD TESTNO ERR PC SUB CALL |
| 817 | 001774 | 051610 | DT50       | : \$TMP4,\$REG1,TESTNO,\$ERRPC,SRCALL,0  |
| 818 | 001776 | 051624 | DF1        | : 0,0,0,0,0                              |
| 819 |        |        |            |  |
| 820 |        |        | : *ITEM 47 |  |
| 821 | 002000 | 046366 | EM53       | : CSM DID NOT CLEAR OLD PSW BITS <3:0>   |
| 822 | 002002 | 051015 | DH44       | : OLDPSW TESTNO ERR PC SUB CALL          |
| 823 | 002004 | 051560 | DT46       | : \$REG1,TESTNO,\$ERRPC,SRCALL,0         |
| 824 | 002006 | 051640 | DF3        | : 0,0,0,0                                |
| 825 |        |        |            |  |
| 826 |        |        | : *ITEM 50 |  |
| 827 | 002010 | 046433 | EM54       | : CSM ACCESSED WRONG SUPERVISOR SPACE    |
| 828 | 002012 | 050246 | DH32       | : TESTNO ERR PC SUB CALL                 |
| 829 | 002014 | 051434 | DT32       | : TESTNO,\$ERRPC,SRCALL,0                |
| 830 | 002016 | 051644 | DF5        | : 0,0,0                                  |
| 831 |        |        |            |  |
| 832 |        |        | : *ITEM 51 |  |
| 833 | 002020 | 046477 | EM55       | : CSM ABORTED WHEN IT SHOULD NOT HAVE    |
| 834 | 002022 | 050246 | DH32       | : TESTNO ERR PC SUB CALL                 |
| 835 | 002024 | 051434 | DT32       | : TESTNO,\$ERRPC,SRCALL,0                |
| 836 | 002026 | 051644 | DF5        | : 0,0,0                                  |

```

838      .SBTTL ***** SUBROUTINES UNIQUE TO THIS PROGRAM *****
839
840      .SBTTL INITIALIZE ALL PAR'S AND PDR'S
841      :*****
842      :
843      :   THIS ROUTINE WILL INITIALIZE ALL KERNAL, SUPERVISOR, AND
844      :   USER PAR'S AND PDR'S TO THEIR USUAL INITIAL VALUE
845      :
846      :*****
847      APRINIT:
848      002030 012700 077406      MOV      #77406,R0      ;MAKE ALL PDR'S 4K, READ/WRITE, UPWARDS
849                                     ;EXPANDING, 200 BLOCKS
850      002034 012702 172300      MOV      #KIPDR0,R2    ;LOAD THE ADDRESS OF THE FIRST KERNAL PDR
851      002040 012701 000020      1$:     MOV      #20,R1    ;LOAD R1 WITH 16
852      002044 010022                2$:     MOV      R0,(R2)+    ;LOAD EACH PDR IN TURN
853      002046 077102                SOB      R1,2$         ;LOOP UNTIL ALL ARE LOADED
854      002050 020227 172340      CMP      R2,#KDPDR7+2 ;HAVE WE LOADED ALL KERNAL PDR'S
855      002054 001003                BNE     3$            ;BRANCH IF KERNAL & SUPER HAVE BEEN LOADED
856      002056 012702 172200      MOV      #SIPDR0,R2   ;LOAD ALL SUPERVISOR PDR'S
857      002062 000766                BR      1$           ;BRANCH TO LOOP
858      002064 020227 172240      3$:     CMP      R2,#SDPDR7+2 ;HAVE USER PDR'S BEEN DONE
859      002070 001003                BNE     4$           ;BRANCH IF THEY HAVE
860      002072 012702 177600      MOV      #UIPDR0,R2   ;LOAD ALL USER PDR'S
861      002076 000760                BR      1$           ;BRANCH TO LOOP
862      002100 012701 172340      4$:     MOV      #KIPAR0,R1 ;LOAD R1 WITH ADDRESS OF KIPAR0
863      002104 012702 172360      MOV      #KDPAR0,R2   ;LOAD R2 WITH ADDRESS OF KDPAR0
864      002110 012703 000007      5$:     MOV      #7,R3       ;LOAD LOOP COUNTER WITH 7
865      002114 005000                CLR     R0           ;CLEAR PAR VALUE REGISTER
866      002116 010021                6$:     MOV      R0,(R1)+    ;LOAD AN I-SPACE PAR
867      002120 010022                MOV     R0,(R2)+    ;LOAD A D-SPACE PAR
868      002122 062700 000200      ADD     #200,R0       ;INCREASE THE PAR VALUE BY 200
869      002126 077305                SOB     R3,6$       ;LOOP UNTIL 7 PAR'S ARE LOADED
870      002130 012711 177600      MOV     #177600,(R1)  ;MAP I-SPACE PAR7 TO I/O PAGE
871      002134 012712 177600      MOV     #177600,(R2)  ;MAP D-SPACE PAR7 TO I/O PAGE
872      002140 020127 172356      CMP     R1,#KIPAR7   ;
873      002144 001005                BNE     7$           ;
874      002146 012701 172240      MOV     #SIPAR0,R1   ;
875      002152 012702 172260      MOV     #SDPAR0,R2   ;
876      002156 000754                BR      5$           ;
877      002160 020127 172256      7$:     CMP     R1,#SIPAR7  ;
878      002164 001401                BEQ     8$           ;BRANCH TO USER LOAD ROUTINE
879      002166 000207                RTS     PC           ;RETURN TO CALLING ROUTINE
880      002170 012701 177640      8$:     MOV     #UIPAR0,R1 ;
881      002174 012702 177660      MOV     #UDPAR0,R2   ;
882      002200 000743                BR      5$           ;
883

```



```

885 .SBTTL D-SPACE TESTS MEMORY MANAGEMENT ABORT SERVICE ROUTINE
886 :*****
887 : THIS ROUTINE WILL BE ENTERED IF A MEMORY MANAGEMENT ABORT OCCURS
888 : DURING THE D-SPACE ENABLE TESTS. IF THE ABORT IS A NON-RESIDENT
889 : ABORT, THE PROBLEM IS PROBABLY IN THE D-SPACE ENABLE LOGIC. IN
890 : ALL OF THE D-SPACE ENABLE TESTS, D-SPACE PAGES 1 & 3 ARE MAPPED
891 : NON-RESIDENT AND I-SPACE PAGE 3 IS MAPPED NON-RESIDENT. ALL
892 : OTHER PAGES ARE MAPPED RESIDENT, 4K, READ/WRITE. THEREFORE, IF
893 : THE NON-RESIDENT PAGE IS 1 OR 3 YOU ARE NOT FORCING I-SPACE WHEN
894 : YOU SHOULD. IF THE NON-RESIDENT PAGE IS 3, AND YOU ARE IN TEST
895 : 15, YOU ARE PROBABLY FORCING I-SPACE WHEN YOU SHOULD BE ALLOWING
896 : D-SPACE.
897 :*****
898 002202 NODSPAC: ; STARTING ADDRESS FOR ABORT SERVICE ROUTINE
899 002202 042767 000004 170306 BIC #BIT2,MMR3 ; TURN OFF D-SPACE BEFORE DOING ROUTINE
900 002210 005227 INC (PC)+ ; MAKE FLAG ZERO IF THE FIRST TIME
901 002212 177777 NDFLAG: .WORD -1 ; FLAG SHOULD BE -1
902 002214 001401 BEQ 10$ ; BRANCH IF FIRST TIME IN ROUTINE
903 002216 000000 HALT ; I HAVE ENTERED THIS ROUTINE BEFORE
904 ; THE FIRST ERROR IS REPORTED; THE SECOND
905 ; ENTRY ADDRESS IS ON THE STACK, AND THE
906 ; FIRST ERROR CONDITION IS PROBABLY STILL
907 ; LOCKED UP.
908 002220 011667 177060 10$: MOV (KSP),BADPC ; SAVE PC AT TIME OF ABORT OR TRAP
909 002224 012667 177030 MOV (KSP)+,TRAPPC ; SAVE RETURN ADDRESS IN CASE OF LOOP
910 002230 012667 177026 MOV (KSP)+,TRAPPS ; SAVE OLD PSW IN CASE OF LOOP
911 002234 016767 175332 177022 MOV MMRO,WASSRO ; SAVE STATUS REGISTER
912 002242 016767 175326 177016 MOV MMR1,WASSR1 ; SAVE AUTO INCR/DECR REGISTER
913 002250 016767 175322 177012 MCV MMR2,WASSR2 ; SAVE VIRTUAL ADDRESS REGISTER
914 002256 005767 177002 TST WASSRO ; WAS ABORT NON-RESIDENT?
915 002262 100002 BPL 1$ ; BRANCH IF ABORT NOT EXPECTED
916 002264 104034 ERROR +34 ; D-SPACE ENABLE FAULTY
917 002266 000401 BR 2$ ; BRANCH TO EXIT
918 002270 104002 1$: ERROR +2 ; UNEXPECTED M.M. ABORT
919 002272 042767 177376 175272 2$: BIC #177376,MMRO ; CLEAR ALL BITS EXCEPT 0 AND 8
920 002300 012767 177777 177704 MOV #-1,NDFLAG ; MOVE A -1 TO THE FLAG
921 002306 016746 176750 MOV TRAPPS,-(KSP) ; PUSH OLD PSW ONTO STACK
922 002312 016746 176742 MOV TRAPPC,-(KSP) ; PUSH OLD PC ONTO STACK
923 002316 052767 000004 170172 BIS #BIT2,MMR3 ; TURN D-SPACE BACK ON
924 002324 000006 RTT ; RETURN TO MAIN PROGRAM
925

```

```

927 .SBTTL TURN OFF T-BIT AND SAVE CURRENT PSW
928 :*****
929 :*
930 :* THIS SUBROUTINE IS USED TO TURN OFF THE TRACE TRAP BIT IN
931 :* THE PSW IF IT IS ON. THE PROCESSOR STATUS IS SAVED IN
932 :* 'TBITPS' SO THAT THE PSW CAN BE RESTORED TO ITS PREVIOUS
933 :* CONDITION WHEN CONDITIONS WARRANT T-BIT TRAPPING.
934 :*
935 :*****
936 002326 036727 175444 000020 TOFF: BIT PSW,#TBIT ;IS THE T-BIT SET IN THE PSW?
937 002334 001411 BEQ 1$ ;EXIT IF NO
938 002336 016746 175434 MOV PSW,-(SP) ;PUSH PRESENT PSW ON THE STACK
939 002342 011667 176726 MOV (SP),TBITPS ;ALSO SAVE IT IN 'TBITPS' FOR
940 ;RESTORING LATER
941 002346 042716 000020 BIC #TBIT,(SP) ;CLEAR THE T-BIT (BIT 4) IN THE PSW
942 002352 012746 002360 MOV #1$,-(SP) ;PUSH PC OF 'RTS' ON STACK
943 002356 000006 RTT ;'RETURN' TO 1$ WITH T-BIT OFF
944 002360 000207 1$: RTS PC ;RETURN TO PROGRAM
945
946 .SBTTL TURN ON T-BIT AND RESTORE PREVIOUS PSW
947 :*****
948 :*
949 :* THIS SUBROUTINE IS USED TO RESTORE THE PROCESSOR STATUS
950 :* TO ITS PREVIOUS CONDITION BY RESTORING THE 'T-BIT PSW'
951 :* SAVED BY THE 'TOFF' SUBROUTINE IN THE 'TBITPS' LOCATION.
952 :*
953 :*****
954 002362 036727 176706 000020 TON: BIT TBITPS,#TBIT ;WAS T-BIT ON IN THE PREVIOUS PSW?
955 002370 001410 BEQ 1$ ;EXIT IF NO
956 002372 016746 176676 MOV TBITPS,-(SP) ;PUSH PREVIOUS PSW ON THE STACK
957 002376 012767 000340 176670 MOV #340,TBITPS ;RESET THE 'TBITPS' LOCATION
958 002404 012746 002412 MOV #1$,-(SP) ;PUSH PC OF 'RTS' ON STACK
959 002410 000006 RTT ;'RETURN' TO 1$ WITH T-BIT RESTORED
960 002412 000207 1$: RTS PC ;RETURN TO PROGRAM
961
962
963
    
```

```

965 .SBTTL ***** TRAP HANDLING ROUTINES *****
966
967 .SBTTL CPU TRAP HANDLER ROUTINE
968 :*****
969 :
970 : THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS THRU
971 : 'ERRVEC' (LOC. 004). IF THIS SUBROUTINE IS ENTERED BY A
972 : SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A HALT IS
973 : EXECUTED.
974 :
975 :*****
976 002414 005227 TIMERR: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME THRU
977 002416 177777 TIMFLG: .WORD -1 ;NEGATIVE ONE FOR 'HAVE ENTERED' FLAG
978 002420 001401 BEQ 1$ ;BRANCH IF FIRST TIME IN
979 002422 000000 HALT ;STOP! - I'VE ENTERED THIS ROUTINE
980 ;A SECOND TIME BEFORE I FINISHED
981 ;REPORTING THE FIRST ERROR. THE
982 ;SECOND ENTRY ADDRESS SHOULD BE ON
983 ;THE KERNEL STACK.
984 002424 012667 176630 1$: MOV (KSP)+,TRAPPC ;SAVE PC+2 AT TIME OF ABORT
985 002430 012667 176626 MOV (KSP)+,TRAPPS ;SAVE PS AT TIME OF ABORT
986 002434 010667 176616 MOV KSP,WASR6 ;SAVE STACK POINTER VALUE
987 002440 104001 ERROR +1 ;UNEXPECTED TRAP OR ABORT TO LOC. 4
988 002442 012767 177777 177746 MOV #-1,TIMFLG ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
989 002450 005067 175312 CLR CPUERR ;CLEAR THE CPU ERROR REGISTER
990 002454 016746 176602 MOV TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
991 002460 016746 176574 MOV TRAPPC,-(KSP)
992 002464 000006 RTT ;RETURN FROM INTERRUPT OR ABORT
993
994
995
996
997
998
999
1000
1001
1002
1003
1004 002466 005227
1005 002470 177777
1006 002472 001401
1007 002474 000000
1008
1009
1010
1011
1012 002476 012667 176556 1$: MOV (KSP)+,TRAPPC ;SAVE PC+2 AT TIME OF ABORT
1013 002502 012667 176554 MOV (KSP)+,TRAPPS ;SAVE PS AT TIME OF ABORT
1014 002506 010667 176544 MOV KSP,WASR6 ;SAVE STACK POINTER VALUE
1015 002512 016767 175054 176544 MOV SR0,WASSR0 ;SAVE CONTENTS OF KT STATUS REG. 0
1016 002520 016767 175052 176542 MOV SR2,WASSR2 ;SAVE CONTENTS OF KT STATUS REG. 2
1017 002526 042767 160000 175036 BIC #160000,SR0 ;CLEAR ERROR BITS IN STATUS REG 0
1018 002534 104002 ERROR +2 ;UNEXPECTED TRAP OR ABORT TO LOC. 250
1019 002536 012767 177777 177724 MOV #-1,MGMFLG ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
1020 002544 016746 176512 MOV TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
1021 002550 016746 176504 MOV TRAPPC,-(KSP)
  
```

```

.SBTTL MEMORY MANAGEMENT TRAP HANDLER ROUTINE
:*****
:
: THIS SUBROUTINE WILL HANDLE ALL UNEXPECTED MEMORY MANAGEMENT
: TRAPS AND ABORTS THRU 'MMVEC' (LOC. 250). IF THIS SUBROUTINE IS
: ENTERED BY A SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A
: HALT IS EXECUTED.
:
:*****
MGMERR: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME THRU
MGMFLG: .WORD -1 ;NEGATIVE ONE FOR 'HAVE ENTERED' FLAG
BEQ 1$ ;BRANCH IF FIRST TIME IN
HALT ;STOP! - I'VE ENTERED THIS ROUTINE
;A SECOND TIME BEFORE I FINISHED
;REPORTING THE FIRST ERROR. THE
;SECOND ENTRY ADDRESS SHOULD BE ON
;THE KERNEL STACK.
1$: MOV (KSP)+,TRAPPC ;SAVE PC+2 AT TIME OF ABORT
MOV (KSP)+,TRAPPS ;SAVE PS AT TIME OF ABORT
MOV KSP,WASR6 ;SAVE STACK POINTER VALUE
MOV SR0,WASSR0 ;SAVE CONTENTS OF KT STATUS REG. 0
MOV SR2,WASSR2 ;SAVE CONTENTS OF KT STATUS REG. 2
BIC #160000,SR0 ;CLEAR ERROR BITS IN STATUS REG 0
ERROR +2 ;UNEXPECTED TRAP OR ABORT TO LOC. 250
MOV #-1,MGMFLG ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
MOV TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
MOV TRAPPC,-(KSP)
  
```

1022 002554 000006  
1023

RTT

;RETURN FROM INTERRUPT OR ABORT.

1025  
 1026  
 1027 020000  
 1028  
 1029 02000C

.SBTTL \*\*\*\*\* STARTING POINT OF TEST \*\*\*\*\*  
 .SBTTL \*\*\*\*\* STARTING ADDRESS OF 200 \*\*\*\*\*  
 .=20000

```

START:
.SBTTL INITIALIZE THE COMMON TAGS
::CLEAR THE COMMON TAGS ($CMTAG) AREA
020000 012706 001100 MOV #CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
020004 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
020006 022706 001140 CMP #SWR,R6 ;;DONE?
020012 001374 BNE -6 ;;LOOP BACK IF NO
020014 012706 001100 MOV #STACK,SP ;;SETUP THE STACK POINTER
::INITIALIZE A FEW VECTORS
020020 012737 037606 000020 MOV #SCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
020026 012737 000340 000022 MOV #340,@IOTVEC+2 ;;LEVEL 7
020034 012737 040000 000030 MOV #ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
020042 012737 000340 000032 MOV #340,@EMTVEC+2 ;;LEVEL 7
020050 012737 043422 000034 MOV #TRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
020056 012737 000340 000036 MOV #340,@TRAPVEC+2;LEVEL 7
020064 012737 043510 000024 MOV #PWRDN,@PWRVEC ;;POWER FAILURE VECTOR
020072 012737 000340 000026 MOV #340,@PWRVEC+2 ;;LEVEL 7
020100 016767 017252 017242 MOV $ENDCT,$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
020106 005067 161100 CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
020112 112767 000001 160775 MOVB #1,$ERMAX ;;ALLOW ONE ERROR PER TEST
::INITIALIZE THE 'T-BIT' TRAP VECTOR. THEN LOAD LOCATION '$RTRN', IN
::THE 'END-OF-PASS' ($EOP) ROUTINE, WITH A 'RTI' OR 'RTT'.
020120 012737 037574 000014 MOV #RTRN,@TBITVEC ;;SET 'T' BIT VECTOR TO $RTRN
020126 012737 000340 000016 MOV #340,@TBITVEC+2 ;;LEVEL 7
020134 012767 000002 017432 MOV #RTI,$RTRN ;;SET $RTRN TO A RTI
020142 012737 020170 000010 MOV #65$,@RESVEC ;;TRY TO DO A RTT
020150 005046 CLR -(SP) ;;DUMMY PS
020152 012746 020160 MOV #64$,-(SP) ;;AND PC
020156 000006 RTT ;;TRY THE RTT
020160 012767 000006 017406 64$: MOV #RTT,$RTRN ;;RTT IS LEGAL--SET $RTRN TO A RTT
020166 000402 BR 66$
020170 062706 000010 65$: ADD #10,SP ;;RTT ILLEGAL--CLEAN OFF THE STACK
020174 012737 000012 000010 66$: MOV #RESVEC+2,@RESVEC ;;RESTORE TRAP CATCHER
020202 005067 161102 CLR $TBIT ;;CLEAR 'T' BIT SWITCH
020206 012767 020206 160672 MOV #,$LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
020214 012767 020214 160666 MOV #,$LPERR ;;SETUP THE ERROR LOOP ADDRESS
::SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
::EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
020222 013746 000004 MOV @ERRVEC,-(SP) ;;SAVE ERROR VECTOR
020226 012737 020262 000004 MOV #67$,@ERRVEC ;;SET UP ERROR VECTOR
020234 012767 177570 160676 MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
020242 012767 177570 160672 MOV #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
020250 022777 177777 160662 CMP #-1,@SWR ;;TRY TO REFERENCE HARDWARE SWR
020256 001012 BNE 69$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
;;AND THE HARDWARE SWR IS NOT = -1
020260 000403 BR 68$ ;;BRANCH IF NO TIMEOUT
020262 012716 020270 67$: MOV #68$,(SP) ;;SET UP FOR TRAP RETURN
020266 000002 RTI
020270 012767 000176 160642 68$: MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR
020276 012767 000174 160636 MOV #DISPREG,DISPLAY
020304 012637 000004 69$: MOV (SP)+,@ERRVEC ;;RESTORE ERROR VECTOR
020310 005067 160716 CLR $PASS ;;CLEAR PASS COUNT
020314 132767 000200 160723 BITB #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
  
```



```

020322 001403          BEQ    70$      ;;YES,USE NON-APT SWITCH
020324 012767 001246 160606 70$:  MOV    #$$SWREG,SWR  ;;NO,USE APT SWITCH REGISTER
020332
1030
020332 005227 177777    .SBITL  TYPE PROGRAM NAME
020336 001047          ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
020340 022737 037530 000042  INC    #-1      ;;FIRST TIME?
020346 001443          BNE    71$      ;;BRANCH IF NO
020350 104401 020416    CMP    #SENDAD,@#42  ;;ACT-11?
020354 005737 000042    BEQ    71$      ;;BRANCH IF YES
020360 001012          TYPE    ,72$     ;;TYPE ASCIZ STRING
020362 126727 160656 000001  .SBITL  GET VALUE FOR SOFTWARE SWITCH REGISTER
020370 001406          TST    @#42     ;;ARE WE RUNNING UNDER XXDP/ACT?
020372 026727 160542 000176  BNE    73$      ;;BRANCH IF YES
020400 001005          CMPB   $ENV,#1    ;;ARE WE RUNNING UNDER APT?
020402 104407          BEQ    73$      ;;BRANCH IF YES
020404 000403          CMP    SWR,#SWREG  ;;SOFTWARE SWITCH REG SELECTED?
020406 112767 000001 160520 73$:  BR    74$      ;;BRANCH IF NO
020414 000420          GTSWR          ;;GET SOFT-SWR SETTINGS
020414 000420          BR    74$      ;;SET AUTO-MODE INDICATOR
020456          MOVB   #1,$AUTOB
020456          BR    71$      ;;GET OVER THE ASCIZ
020456          .:72$: .ASCIZ <CRLF>#CKKTBB0 11/44 MEM MGMT PRT B#<CRLF>
020456          .:71$:
1031
1032 020456          LOOP:
1033 020456 012706 001100    MOV    #STACK,KSP  ;;INITIALIZE THE STACK POINTER
1034 020462 012767 040000 157306  MOV    #40000,PSW   ;;TURN ON SUPERVISOR MODE
1035 020470 012706 000700    MOV    #SUPSTK,SSP  ;;INITIALIZE SUPER. STACK POINTER
1036 020474 012767 140000 157274  MOV    #140000,PSW  ;;TURN ON USER MODE
1037 020502 012706 000600    MOV    #USESTK,USP  ;;INITIALIZE USER STACK POINTER
1038 020506 005067 157264    CLR    PSW          ;;RETURN TO KERNAL MODE
1039 020512 012767 002414 157264  MOV    #TIMERR,ERRVEC  ;;LOAD CPU SERVICE ROUTINE INTO TRAP VECTOR
1040 020520 012767 000340 157260  MOV    #340,ERRVEC+2  ;;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
1041 020526 012767 002466 157514  MOV    #MGMERR,MMVEC  ;;LOAD MEMORY MANAGENT ROUTINE INTO VECTOR
1042 020534 012767 000340 157510  MOV    #340,MMVEC+2  ;;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
1043 020542 012700 177777    MOV    #-1,RO       ;;PUT -1 INTO RO TO INITIALIZE FLAGS
1044 020546 010067 161644    MOV    RO,TIMFLG    ;;INITIALIZE CPU ERROR FLAG
1045 020552 010067 161712    MOV    RO,MGMFLG    ;;INITIALIZE MEMORY MANAGEMENT ERROR FLAG
1046 020556 012767 000340 160510  MOV    #340,TBITPS  ;;INITIALIZE LOG THAT HOLDS T-BIT PSW
1047 020564 005067 157002    CLR    MMR0         ;;BE SURE MEM. MGMT IS OFF TO START WITH,
1048 020570 012767 000020 151720  MOV    #BIT4,MMR3   ;;BUT TURN ON 22-BIT ADDRESSING MODE
1049 020576 004767 161226    JSR    PC,APRINIT  ;;JUMP TO PAR/PDR INIT ROUTINE
    
```





1156

```

*****
*TEST 2      READ-ONLY ABORT TEST (ACF=2)
*
*   THIS TEST CHECKS THE ACCESS CONTROL FIELD (ACF) COMPARATOR
*   LOGIC BY CAUSING READ-ONLY ABORTS IN KERNEL, SUPERVISOR AND
*   USER MODES.  PDR 4 IS LOAD WITH ACF=2 AND THEN
*   PHYSICAL ADDR. 60000 IS WRITTEN TO CAUSE THE ABORT.
*
*****
  
```

```

1157 021126 000004
1158 021130
1159 021130 012700 060000      MOV      #60000,R0      ;KERNEL, SUPERVISOR AND USER PAR'S 3 & 4,
1160 021134 012701 100000      MOV      #100000,R1     ;AND PDR 3 ARE SETUP FROM LAST TEST
1161 021140 012703 020011      MOV      #20011,R3     ;LOAD VIRTUAL ADDR. TO REFERENCE PDR3 INTO R0
1162 021144 012702 077402      MOV      #77402,R2     ;LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
1163 021150 012767 021222 157072 2$:  MOV      #5$,MMVEC     ;LOAD R3 WITH WHAT SRO SHOULD READ - R/O, KERNEL, PG.4
1164 021156 010267 151126      MOV      R2,KIPDR4     ;LOAD ACF=2 (READ-ONLY) PDR VALUE IN R2
1165 021162 010267 151022      MOV      R2,SIPDR4     ;POINT MEM. MGMT. TRAP VECTOR TO 5$ BELOW
1166 021166 010267 156416      MOV      R2,UIPDR4     ;LOAD ACF=2 INTO KIPDR4
1167 021172 012767 021204 157710      MOV      #3$,SLPERR    ;LOAD ACF=2 INTO SIPDR4
1168 021200 005267 156366      INC      MMR0          ;LOAD ACF=2 INTO UIPDR4
1169 021204 005010              CLR      (R0)          ;SET LOOP ON ERROR POINTER TO 3$
1170 021206 016767 156564 157762 3$:  MOV      PSW,$TMPO     ;TURN ON MEMORY MANAGEMENT
1171 021214 005211              INC      (R1)          ;CLEAR PHYS. LOC. 60000 USING PDR3
1172 021216 104003              ERROR    +3           ;SAVE PSW IN CASE OF ERROR
1173                                ;TRY TO WRITE USING PDR4 - SHOULD TRAP TO 5$
1174                                ;MEM. MGMT. ABORT DID NOT OCCUR
1175                                ;FOR TIGHTER SCOPE LOOP
1176 021220 000425              BR       8$           ;REPLACE ERROR CALL WITH
1177 021222 062706 000004 5$:  ADD      #4,$SP        ;'BR 3$' = 000772
1178 021226 005710              TST     (R0)          ;BRANCH AROUND STATUS REG. CHECKS IF NO ABORT
1179 021230 001401              BEQ     6$           ;RESTORE STACK POINTER
1180 021232 104004              ERROR    +4           ;DID INSTRUCTION GET ABORTED & NOT EXECUTE
                                ;BRANCH IF YES
                                ;INSTRUCTION WAS NOT ABORTED, LOC. GOT CHANGED
  
```

```

1182 ;FOR TIGHTER SCOPE LOOP
1183 ;REPLACE ERROR CALL WITH
1184 ;'BR 3$' = 000764
1185 021234 016767 156332 160022 6$: MOV SRO,WASSRO ;READ STATUS REG. 0
1186 021242 016767 156330 160020 MOV SR2,WASSR2 ;READ STATUS REG. 2
1187 021250 020367 160010 CMP R3,WASSRO ;DID SRO REPORT READ-ONLY ERROR CORRECTLY?
1188 021254 001401 BEQ 7$ ;BRANCH IF YES
1189 021256 104005 ERROR +5 ;SRO DID NOT REPORT R/O ERROR CORRECTLY
1190 ;FOR TIGHTER SCOPE LOOP
1191 ;REPLACE ERROR CALL WITH
1192 ;'BR 3$' = 000752
1193 021260 012704 021214 7$: MOV #4$,R4 ;LOAD R4 WITH WHAT SR2 SHOULD READ
1194 021264 020467 160000 CMP R4,WASSR2 ;DID SR2 LOCKUP RIGHT VIRTUAL ADDR. (=4$)?
1195 021270 001401 BEQ 8$ ;BRANCH IF YES
1196 021272 104006 ERROR +6 ;SR2 DID NOT LOCKUP VIRTUAL ADDR. OF R/O ERROR
1197 ;FOR TIGHTER SCOPE LOOP
1198 ;REPLACE ERROR CALL WITH
1199 ;'BR 3$' = 000744
1200 021274 005067 156272 8$: CLR MMRO ;TURN OFF MEMORY MANAGEMENT
1201 021300 032767 140000 157670 BIT #140000,$TMPO ;HAS ACF=2 BEEN TESTED IN USER MODE?
1202 021306 001006 BNE 11$ ;BRANCH IF YES
1203 021310 012703 020151 MOV #20151,R3 ;LOAD R3 WITH WHAT SRO SHOULD READ-R/O, USER, PG.4
1204 021314 012767 140000 156454 MOV #140000,PSW ;GO TO USER MODE
1205 021322 000712 BR 2$ ;REPEAT TEST IN USER MODE
1206 021324 022767 040000 157644 11$: CMP #40000,$TMPO ;HAS ACF=2 BEEN TESTED IN SUPERVISOR MODE?
1207 021332 001006 BNE 9$ ;BRANCH IF YES
1208 021334 012703 020051 MOV #20051,R3 ;LOAD R3 WITH WHAT SRO SHOULD READ-R/O, SUPERVISOR, PG.4
1209 021340 012767 040000 156430 MOV #40000,PSW ;GO TO SUPERVISOR MODE
1210 021346 000700 BR 2$ ;REPEAT TEST IN SUPERVISOR MODE
1211 021350 005067 156422 9$: CLR PSW ;GO BACK TO KERNEL MODE BEFORE LEAVING
1212 021354 012767 021130 157526 MOV #1$,$LPERR ;RESET LOOP ON ERROR POINTER TO 1$
1213 021362 012767 002466 156660 MOV #MMGMERR,MMVEC ;RESTORE ADDRESS OF NORMAL MEMORY
1214 ;MANAGEMENT ERROR ROUTINE TO MMVEC.
1225

```

```

:*****
:*TEST 3 TEST ILLIGAL MODE '10'
:*
:* THIS TEST CHECKS TO SEE THAT A 10 IN THE CURRENT MODE BITS OF THE
:* PSW WHILE MEMORY MANAGEMENT IS ON IS ILLEGAL. A
:* MEMORY MANAGEMENT ABORT SHOULD OCCUR AND STATUS REGISTER 0
:* SHOULD REPORT NON-PRESIDENT ABORT, MODE = 10, PAGE = 1 (100103). STATUS
:* REGISTER 2 SHOULD LOCKUP THE ADDRESS OF THE INSTRUCTION
:* THAT LOADED THE PSW.
:*
:*****

```

```

1226 021370 000004 TST3: SCOPE
1227 021372 012767 021414 157510 1$: MOV #2$,$LPERR ;SET LOOP ON ERROR POINTER TO 2$
1228 021400 012767 021426 156642 MOV #3$,MMVEC ;LOAD MEM. MGMT. TRAP VECTOR WITH 3$
1229 021406 012767 000001 156156 MOV #1,MMRO ;TURN ON MEMORY MANAGEMENT
1230 021414 012767 100000 156354 2$: MOV #100000,PSW ;SET 10 IN PSW CURRENT MODE BITS
1231 ERROR +30 ;ILLEGAL MODE 10 NOT ABORTED
1232 ;FOR A TIGHTER SCOPE LOOP
1233 ;REPLACE ERROR CALL WITH
1234 ;'BR 2$' = 000773
1235 021424 000424 BR 5$ ;BRANCH AROUND SRO & SR2 CHECKS
1236 021426 012706 001100 3$: MOV #KERSTK,SP ;RESTORE STACK POINTER
1237 021432 012701 100103 MOV #100103,R1 ;LOAD EXPECTED CONTENTS OR SRO INTO R1
1237 021436 016767 156130 157620 MOV SRO,WASSRO ;READ CONTENTS OF SRO

```



```

1238
1239 021444 020167 157614      CMP      R1,WASSRO      ;DID SRO REPORT ILLEGAL MODE CORRECTLY?
1240 021450 001401              BEQ      4$              ;BRANCH IF YES
1241 021452 104031              ERROR    +31            ;SRO DID NOT REPORT NR ABORT, PG=1, MODE=10
1242                                ;FOR TIGHTER SCOPE LOOP
1243                                ;REPLACE ERROR CALL WITH
1244                                ;'BR 2$' = 000757
1245 021454 012701 021414      4$:   MOV      #2$,R1      ;LOAD EXPECTED CONTENTS OF SR2 INTO R1
1246 021460 016767 156112 157602  MOV      SR2,WASSR2    ;READ CONTENTS OF SR2
1247 021466 020167 157576      CMP      R1,WASSR2    ;DID SR2 LOCKUP VIRT. ADDR OF ABORTED INST.
1248 021472 001401              BEQ      5$              ;BRANCH IF YES
1249 021474 104013              ERROR    +13            ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ILL. MODE INST.
1250                                ;FOR TIGHTER SCOPE LOOP
1251                                ;REPLACE ERROR CALL WITH
1252                                ;'BR 2$' = 000746
1253 021476 042767 160000 156066 5$:   BIC      #160000,SRO   ;CLEAR POSSIBLE ERROR BITS IN SRO
1254 021504 012767 002466 156536  MOV      #MGMERR,MMVEC ;RESTORE MEM. MGMT. ABORT VECTOR
1255 021512 012767 021372 157370  MOV      #1$, $LPERR   ;RESET LOOP ON ERROR POINTER TO 1$
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1287
1288

```

```

:*****
:
: THE NEXT TWO (2) TESTS WILL BE CHECKING THE PAGE LENGTH
: COMPARATORS AND SOME MORE OF THE KT ERROR DETECTION
: AND STATUS LOGIC. THE PAGE LENGTH FIELD (PLF) IN KERNEL
: PDR 4 IS VARIED AND FOR EVERY PLF, THREE (3) VIRTUAL
: ADDRESSES ARE READ. WHILE USING BOTH UPWARD & DOWNWARD PAGE
: EXPANSION, ONE OF THOSE THREE VIRTUAL ADDRESSES WILL CAUSE A
: 'PAGE LENGTH ABORT' WHILE THE OTHER TWO WON'T.
:
: STATUS REGISTER 0 & 2 ARE CHECKED WHEN THE PAGE LENGTH
: ABORT DOES OCCUR TO SEE THAT THE ABORT IS REPORTED AND THAT
: THE VIRTUAL ADDRESS OF THE INSTRUCTION THAT CAUSED THE ABORT
: IS LOCKED UP.
:*****

```

```

:*****
: *TEST 4 PAGE LENGTH FAULTS-UPWARD EXPANSION
:
: THIS TEST VARIES THE PAGE LENGTH FIELD (PLF) IN KERNEL PDR 4
: FROM 1 TO 177 AND FOR EACH PLF, THREE VIRTUAL ADDRESSES (VBA'S)
: ARE ACCESSED. WHEN VBA <12:6> IS LESS THAN OR EQUAL TO PDR <14:8>
: NO ABORT SHOULD OCCUR. WHEN VBA <12:6> IS GREATER THAN PDR <14:8>,
: A PAGE LENGTH ABORT SHOULD OCCUR AND BE REPORTED BY SRO & SR2.
: THE PAGE EXPANSION DIRECTION IN THIS TEST IS UPWARD, (THE ED BIT
: (BIT 3) OF PDR 4 = 0).
:*****

```

```

1289 021520 000004
1289 021522 012767 077406 150556 1$:   TST4:  SCOPE
1290 021530 012767 077406 150554      MOV      #77406,KIPDR3 ;MAKE SURE PDR3 IS DESCRIBED AS R/W
1291 021536 012700 100000              MOV      #77406,KIPDR5 ;MAKE SURE PDR5 IS DESCRIBED AS R/W
1292 021542 012704 000406              MOV      #100000,R0     ;LOAD VIRTUAL ADDR. TO SELECT PDR4 INTO R0
1293 021546 012767 021740 156474 2$:   MOV      #406,R4        ;LOAD FIRST PDR VALUE IN R4 (PLF=1, ACF=6)
                                MOV      #9$,MMVEC      ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS

```

|      |        |        |        |        |       |               |   |                                     |
|------|--------|--------|--------|--------|-------|---------------|---|-------------------------------------|
| 1294 | 021554 | 010467 | 150530 |        | MOV   | R4,KIPDR4     | :LOAD KIPDR4 WITH PAGE LENGTH VALUE                 |                                     |
| 1295 | 021560 | 012767 | 021566 | 157322 | MOV   | #3\$, \$LPERR | :SET LOOP ON ERROR POINTER TO 3\$                   |                                     |
| 1296 | 021566 | 012706 | 001100 | 3\$:   | MOV   | #KERSTK,KSP   | :MAKE SURE STACK POINTER IS ALL SET UP              |                                     |
| 1297 | 021572 | 011001 |        |        | MOV   | (R0),R1       | :ACCESS VIRTUAL ADDR. (VBA < PLF - NO ABORT)        |                                     |
| 1298 | 021574 | 062700 | 000100 |        | ADD   | #100,R0       | :FORM NEXT VIRTUAL ADDRESS IN R0                    |                                     |
| 1299 | 021600 | 012767 | 021606 | 157302 | MOV   | #4\$, \$LPERR | :SET LOOP ON ERROR POINTER TO 4\$                   |                                     |
| 1300 | 021606 | 012706 | 001100 | 4\$:   | MOV   | #KERSTK,KSP   | :MAKE SURE STACK POINTER IS ALL SET UP              |                                     |
| 1301 | 021612 | 011001 |        |        | MOV   | (R0),R1       | :ACCESS VIRTUAL ADDR. (VBA=PLF - NO ABORT)          |                                     |
| 1302 | 021614 | 062700 | 000100 |        | ADD   | #100,R0       | :FORM NEXT VIRTUAL ADDR IN R0                       |                                     |
| 1303 | 021620 | 020027 | 117700 |        | CMP   | R0,#117700    | :HAVE ALL PLF'S BEEN TESTED YET?                    |                                     |
| 1304 | 021624 | 001470 |        |        | BEQ   | 10\$          | :BRANCH IF ALL VBA'S & PLF'S HAVE BEEN USED         |                                     |
| 1305 | 021626 | 012767 | 021642 | 157254 | MOV   | #5\$, \$LPERR | :SET LOOP ON ERROR POINTER TO 5\$                   |                                     |
| 1306 | 021634 | 012767 | 021650 | 156406 | MOV   | #6\$,MMVEC    | :SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT          |                                     |
| 1307 | 021642 | 011001 |        | 5\$:   | MOV   | (R0),R1       | :ACCESS VIRTUAL ADDR. (VBA > PLF - ABORT TO 6\$)    |                                     |
| 1308 | 021644 | 104010 |        |        | ERROR | +10           | :EXPECTED PAGE LENGTH ABORT DID NOT OCCUR           |                                     |
| 1309 |        |        |        |        |       |               | :FOR TIGHTER SCOPE LOOP                             |                                     |
| 1310 |        |        |        |        |       |               | :REPLACE ERROR CALL WITH                            |                                     |
| 1311 |        |        |        |        |       |               | : 'BR 5\$' = 000776                                 |                                     |
| 1312 | 021646 | 000424 |        |        | BR    | 8\$           | :BRANCH AROUND ABORT CHECKS                         |                                     |
| 1313 | 021650 | 012706 | 001100 | 6\$:   | MOV   | #KERSTK,KSP   | :RESTORE STACK POINTER FOLLOWING ABORT              |                                     |
| 1314 | 021654 | 016767 | 155712 | 157402 | MOV   | SRO,WASSRO    | :READ M.M. STATUS REG. 0                            |                                     |
| 1315 | 021662 | 016767 | 155710 | 157400 | MOV   | SR2,WASSR2    | :READ M.M. STATUS REG. 2                            |                                     |
| 1316 | 021670 | 012702 | 040011 |        | MOV   | #40011,R2     | :PUT EXPECTED SRO CONTENTS IN R2                    |                                     |
| 1317 | 021674 | 020267 | 157364 |        | CMP   | R2,WASSRO     | :DID SRO REPORT PG. LENGTH ABORT, PAGE 4, KERNEL?   |                                     |
| 1318 | 021700 | 001401 |        |        | BEQ   | 7\$           | :BRANCH IF YES                                      |                                     |
| 1319 | 021702 | 104011 |        |        | ERROR | +11           | :SRO DID NOT REPORT PG. LENGTH ABORT CORRECTLY      |                                     |
| 1320 |        |        |        |        |       |               | :FOR TIGHTER SCOPE LOOP                             |                                     |
| 1321 |        |        |        |        |       |               | :REPLACE ERROR CALL WITH                            |                                     |
| 1322 |        |        |        |        |       |               | : 'BR 5\$' = 000757                                 |                                     |
| 1323 | 021704 | 012703 | 021642 | 7\$:   | MOV   | #5\$,R3       | :PUT EXPECTED SR2 CONTENTS IN R3                    |                                     |
| 1324 | 021710 | 020367 | 157354 |        | CMP   | R3,WASSR2     | :DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCT'ON? |                                     |
| 1325 | 021714 | 001401 |        |        | BEQ   | 8\$           | :BRANCH IF YES                                      |                                     |
| 1326 | 021716 | 104012 |        |        | ERROR | +12           | :SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY  |                                     |
| 1327 |        |        |        |        |       |               | :FOR TIGHTER SCOPE LOOP                             |                                     |
| 1328 |        |        |        |        |       |               | :REPLACE ERROR CALL WITH                            |                                     |
| 1329 |        |        |        |        |       |               | : 'BR 5\$' = 000751                                 |                                     |
| 1330 | 021720 | 042767 | 160000 | 155644 | 8\$:  | BIC           | #160000,SRO   | :CLEAR ERROR BITS IN SRO            |
| 1331 | 021726 | 062704 | 000400 |        | ADD   | #400,R4       | :FORM NEXT PLF VALUE FOR KIPDR4                     |                                     |
| 1332 | 021732 | 162700 | 000100 |        | SUB   | #100,R0       | :FORM FIRST VIRT. ADDR FOR THAT PLF VALUE           |                                     |
| 1333 | 021736 | 000703 |        |        | BR    | 2\$           | :BRANCH BACK AND ACCESS 3 VBA'S FOR                 |                                     |
| 1334 |        |        |        |        |       |               | :THE PLF VALUE JUST FORMED                          |                                     |
| 1335 | 021740 | 012667 | 157314 | 9\$:   | MOV   | (KSP)+,TRAPPC | :SAVE PC & PS OF TRAP                               |                                     |
| 1336 | 021744 | 012667 | 157312 |        | MOV   | (KSP)+,TRAPPS |   |                                     |
| 1337 | 021750 | 016767 | 155616 | 157306 | MOV   | SRO,WASSRO    | :SAVE CONTENTS OF SRO FOR ERROR                     |                                     |
| 1338 | 021756 | 016767 | 155614 | 157304 | MOV   | SR2,WASSR2    | :SAVE CONTENTS OF SR2 FOR ERROR                     |                                     |
| 1339 | 021764 | 042767 | 160000 | 155600 | BIC   | #160000,SRO   | :CLEAR ERROR BITS IN SRO                            |                                     |
| 1340 | 021772 | 104007 |        |        | ERROR | +7            | :GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED        |                                     |
| 1341 |        |        |        |        |       |               | :FOR TIGHTER SCOPE LOOP                             |                                     |
| 1342 |        |        |        |        |       |               | :REPLACE ERROR CALL WITH                            |                                     |
| 1343 |        |        |        |        |       |               | : A 'NOP' = 000240                                  |                                     |
| 1344 | 021774 | 016746 | 157262 |        | MOV   | TRAPPS,-(KSP) | :PUT PC & PS OF TRAP ON STACK                       |                                     |
| 1345 | 022000 | 016746 | 157254 |        | MOV   | TRAPPC,-(KSP) |   |                                     |
| 1346 | 022004 | 000002 |        |        | RTI   |               | :RETURN FROM UNEXPECTED ABORT                       |                                     |
| 1347 |        |        |        |        |       |               |   |                                     |
| 1348 | 022006 | 012767 | 021522 | 157074 | 10\$: | MOV           | #1\$, \$LPERR                                       | :RESET LOOP ON ERROR POINTER TO 1\$ |
| 1349 | 022014 | 012767 | 002466 | 156226 | MOV   | #MGMERR,MMVEC | :RESTORE NORMAL M.M. TRAP HANDLER                   |                                     |
| 1350 |        |        |        |        |       |               | :ADDRESS TO M.M. TRAP VECTOR                        |                                     |

1351  
 1363  
 1364

```

*****
*TEST 5          PAGE LENGTH FAULTS-DOWNWARD EXPANSION
*
*   THIS TEST VARIES THE PAGE LENGTH FIELD (PLF) IN KERNEL PDR4
*   FROM 176 TO 0 AND FOR EACH PLF, THREE VIRTUAL ADDRESSES (VBA'S)
*   ARE ACCESSED.  WHEN VBA <12:6> IS GREATER THAN OR EQUAL TO PDR <14:8>
*   NO PAGE ABORT SHOULD OCCUR.  WHEN VBA <12:6> IS LESS THAN PDR <14:8>
*   A PAGE LENGTH ABORT SHOULD OCCUR AND BE REPORTED BY SR0 & SR2.
*   THE PAGE EXPANSION DIRECTION IN THIS TEST IS DOWNWARD, (THE ED BIT
*   (BIT 3) OF PDR4=1).
*
*****
  
```

```

TST5:  SCOPE
1365 022022 000004          117700  1$:  MOV      #117700,R0      ;LOAD VIRTUAL ADDR. TO SELECT PDR4 INTO R0
1366 022024 012700          077016  1$:  MOV      #77016,R4      ;LOAD FIRST PDR VALUE IN R4 (PLF=176,ACF=6)
1367 022030 012704          022226 156206 2$:  MOV      #9$,MMVEC     ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS
1368 022034 012767          150242 2$:  MOV      R4,KIPDR4     ;LOAD KIPDR4 WITH PAGE LENGTH VALUE
1369 022042 010467          022054 157034 3$:  MOV      #3$,SLPERR    ;SET LOOP ON ERROR POINTER TO 3$
1370 022046 012767          001100 3$:  MOV      #KERSTK,KSP   ;MAKE SURE STACK POINTER IS ALL SET UP
1371 022060 011001          162700 3$:  MOV      (R0),R1      ;ACCESS VIRTUAL ADDR. (VBA > PLF - NO ABORT)
1372 022062 012700          000100 3$:  SUB      #100,R0       ;FORM NEXT VIRTUAL ADDRESS IN R0
1373 022066 012767          022074 157014 4$:  MOV      #4$,SLPERR    ;SET LOOP ON ERROR POINTER TO 4$
1374 022074 012706          001100 4$:  MOV      #KERSTK,KSP   ;MAKE SURE STACK POINTER IS ALL SET UP
1375 022100 011001          000100 4$:  MOV      (R0),R1      ;ACCESS VIRTUAL ADDR. (VBA=PLF - NO ABORT)
1376 022102 162700          000100 4$:  SUB      #100,R0       ;FORM NEXT VIRTUAL ADDR. IN R0
1377 022106 020027          100000 4$:  CMP      R0,#100000    ;HAVE ALL PLF'S BEEN TESTED YET?
1378 022112 001470          022130 156766 5$:  BEQ      10$           ;BRANCH IF ALL VBA'S & PLF'S HAVE BEEN USED
1379 022114 012767          022136 156120 5$:  MOV      #5$,SLPERR    ;SET LOOP ON ERROR POINTER TO 5$
1380 022122 012767          001100 5$:  MOV      #6$,MMVEC     ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT
1381 022130 011001          104010 5$:  MOV      (R0),R1      ;ACCESS VIRTUAL ADDR. (VBA < PLF - ABORT TO 6$)
1382 022132 104010          001100 5$:  ERROR   +10          ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
1383 022134 000424          001100 5$:  BR      8$           ;FOR TIGHTER SCOPE LOOP
1384 022136 012706          155424 157114 6$:  MOV      #KERSTK,KSP   ;REPLACE ERROR CALL WITH
1385 022142 016767          155422 157112 6$:  MOV      SR0,WASSRO    ;'BR 5$' = 000776
1386 022150 016767          040011 6$:  MOV      SR2,WASSR2    ;BRANCH AROUND ABORT CHECKS
1387 022156 012702          157076 6$:  MOV      #40011,R2     ;RESTORE STACK POINTER FOLLOWING ABORT
1388 022162 020267          001401 6$:  CMP      R2,WASSRO    ;READ M.M. STATUS REG. 0
1389 022166 001401          104011 6$:  BEQ      7$           ;READ M.M. STATUS REG. 2
1390 022170 104011          022130 6$:  ERROR   +11          ;PUT EXPECTED SR0 CONTENTS IN R2
1391 022172 012703          157066 7$:  MOV      #5$,R3        ;DID SR0 REPORT PG. LENGTH ABORT, PG. 4, KERNEL?
1392 022174 020367          001401 7$:  CMP      R3,WASSR2    ;BRANCH IF YES
1393 022176 001401          160000 7$:  BEQ      8$           ;SR0 DID NOT REPORT PG. LENGTH ABORT CORRECTLY
1394 022178 104012          000400 7$:  ERROR   +12          ;FOR TIGHTER SCOPE LOOP
1395 022180 062700          000100 7$:  MOV      #160000,SR0   ;REPLACE ERROR CALL WITH
1396 022182 162704          000400 7$:  SUB      #400,R4       ;'BR 5$' = 000757
1397 022184 062700          000100 7$:  ADD      #100,R0       ;PUT EXPECTED SR2 CONTENTS IN R3
1398 022186 000100          155356 8$:  BIC      #160000,SR0   ;DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?
1399 022188 000400          155356 8$:  SUB      #400,R4       ;BRANCH IF YES
1400 022190 000100          155356 8$:  ADD      #100,R0       ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY
1401 022192 000100          155356 8$:  ERROR   +12          ;FOR TIGHTER SCOPE LOOP
1402 022194 000100          155356 8$:  ERROR   +12          ;REPLACE ERROR CALL WITH
1403 022196 000100          155356 8$:  ERROR   +12          ;'BR 5$' = 000751
1404 022198 000100          155356 8$:  ERROR   +12          ;CLEAR ERROR BITS IN SR0
1405 022200 000100          155356 8$:  ERROR   +12          ;FORM NEXT PLF VALUE FOR KIPDR4
1406 022202 000100          155356 8$:  ERROR   +12          ;FORM FIRST VIRT. ADDR. FOR THAT PLF VALUE
  
```

```

1407 022224 000703 BR 2$ ;BRANCH BACK AND ACCESS 3 VBA'S FOR
1408 ;THE PLF VALUE JUST FORMED
1409 022226 012667 157026 9$: MOV (KSP)+,TRAPPC ;SAVE PC & PS OF TRAP
1410 022232 012667 157024 MOV (KSP)+,TRAPPS
1411 022236 016767 155330 157020 MOV SR0,WASSR0 ;SAVE CONTENTS OF SR0 FOR ERROR
1412 022244 016767 155326 157016 MOV SR2,WASSR2 ;SAVE CONTENTS OF SR2 FOR ERROR
1413 022252 042767 160000 155312 BIC #160000,SR0 ;CLEAR ERROR BITS IN SR0
1414 022260 104007 ERROR +7 ;GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
1415 ;FOR TIGHTER SCOPE LOOP
1416 ;REPLACE ERROR CALL WITH
1417 ;A 'NOP' = 000240
1418 022262 016746 156774 MOV TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
1419 022266 016746 156766 MOV TRAPPC,-(KSP)
1420 022272 000002 RTI ;RETURN FROM UNEXPECTED ABORT
1421
1422 022274 012767 022024 156606 10$: MOV #1$,$LPERR ;RESET LOOP ON ERROR POINTER TO 1$
1423 022302 012767 002466 155740 MOV #MGMERR,MMVEC ;RESTORE NORMAL M.M. TRAP HANDLER
1424 ;ADDRESS TO M.M. TRAP VECTOR
1425
1439
1440
    
```

```

:*****
:TEST 6 SR2 BIT TEST
:
: THIS TEST CHECKS THE BITS IN MEMORY MANAGEMENT REGISTER 2 BY
: CAUSING 'READ-ONLY ABORTS' AT VIRTUAL ADDRESSES BETWEEN 100000
: TO 111000 (PHYSICAL ADDRESSES 060000-071000). KIPDR4 IS USED TO EXECUTE
: THE FOLLOWING FOUR WORDS OF CODE WHICH ARE MOVED THRU MEMORY:
: 010727 MOV PC,(PC)+ ;THIS INSTRUCTION SHOULD CAUSE A R/O ABORT
: 000000 ;ITS VIRTUAL ADDR. SHOULD BE LOCKED UP IN SR2
: 000137 JMP @#3$ ;THIS INSTRUCTION IS ALSO MOVED THRU MEMORY
: (ADDR. OF 3$) ;IN CASE A R/O ABORT DOES NOT OCCUR,
: ;IN WHICH CASE SR2 WILL NOT CONTAIN CORRECT ADDR.
:*****
    
```

```

1441 022310 000004 TST6: SCOPE
1442 022312 012767 000600 150026 1$: MOV #600,KIPAR3 ;BE SURE PAR3 IS MAPPED TO 12-16K
1443 022320 012767 000600 150022 MOV #600,KIPAR4 ;BE SURE PAR4 IS MAPPED TO 12-16K
1444 022326 012767 077406 147752 MOV #77406,KIPDR3 ;MAP PAGE 3 128 BLOCKS, R/W
1445 022334 012767 077402 147746 MOV #77402,KIPDR4 ;MAP PAGE 4 128 BLOCKS, READ-ONLY
1446 022342 012700 060000 MOV #60000,R0 ;LOAD R0 WITH VIRTUAL ADDR. WHICH USES PDR3
1447 022346 012701 100000 MOV #100000,R1 ;LOAD R1 WITH VIRTUAL ADDR. WHICH USES PDR4
1448 022352 012767 022406 155670 MOV #3$,MMVEC ;SET M.M. TRAP VECTOR TO 3$
1449 022360 012767 022366 156522 MOV #2$,$LPERR ;SET LOOP ON ERROR POINTER TO 2$
1450 022372 005020 2$: MOV #010727,(R0)+ ;LOAD 'MOV PC,(PC)+' INSTRUCTION AT ADDR.
1451 022374 012720 000137 CLR (R0)+ ; REACHED THRU PDR/PAR 4.
1452 022400 012710 022406 MOV #000137,(R0)+ ;LOAD 'JMP @#3$' INSTRUCTION AT VIRT. ADDR.
1453 022404 010107 MOV #3$,(R0) ; IN CASE R/O VIOL. DOES NOT ABORT
1454 022406 012706 001100 MOV R1,PC ;TRANSFER PROGRAM EXECUTION TO 'PAGE 4 INSTRUCTIONS'
1455 022412 016767 155160 156650 3$: MOV #KERSTK,KSP ;RESTORE STACK POINTER
1456 022420 020167 156644 MOV SR2,WASSR2 ;READ CONTENTS OF STATUS REG 2
1457 022424 001401 CMP R1,WASSR2 ;WAS ADDR. OF 'RELOCATED - R/O ABORT' LOCKED UP?
1458 022426 104013 BEQ 4$ ;BRANCH IF YES
1459 ;SR2 DID NOT LOCK UP VIRTUAL ADDR. OF R/O VIOL.
1460 ;FOR TIGHTER SCOPE LOOP
1461 ;REPLACE ERROR CALL WITH
1462 022430 042767 160000 155134 4$: BIC #160000,SR0 ;'BR 2$' = 000757
;CLEAR THE ERROR BITS IN SR0
    
```

```

1463 022436 162700 000004 SUB #4,R0 ;RESET R0 TO POINT TO NEXT VIRT. ADDR. TO LOAD
1464 022442 062701 000002 ADD #2,R1 ;FORM VIRTUAL ADDR. THAT SHOULD BE LOCKED UP NEXT
1465 022446 020127 111002 CMP R1,#111002 ;HAVE ALL VBA'S 100000-111000 BEEN TESTED?
1466 022452 103745 BLO 2$ ;BRANCH IF NO
1467
1468 022454 012767 022312 156426 5$: MOV #1$,$LPERR ;RESET LOOP ON ERROR POINTER TO 1$
1469 022462 012767 077406 147620 MOV #77406,KIPDR4 ;RESTORE PDR4 TO R/W ACCESS
1470 022470 012767 002466 155552 MOV #MMGMERR,MMVEC ;RESTORE ADDRESS OF NORMAL M.M.
1471 ;TRAP HANDLER TO M.M. VECTOR
1472
1486
1487
    
```

\*\*\*\*\*  
 \*TEST 7 MORE CHECKS OF SRO & SR2  
 \*\*\*\*\*

THIS TEST PERFORMS SOME ADDITIONAL CHECKS OF THE SRO & SR2 LOGIC. FIRST IT CHECKS THAT SR2 'TRACKS' ALONG ACTING AS A VIRTUAL ADDRESS PROGRAM COUNTER. ALSO SRO & SR2 ARE LOCKED UP BY A PAGE LENGTH ABORT, THEN WITHOUT CLEARING SRO'S ERROR BITS, A R/O ABORT IS CAUSED. SRO & SR2 SHOULD NOT BE CHANGED BY THE SECOND ABORT AND THE INFORMATION ABOUT THE PAGE LENGTH ABORT SHOULD STILL BE LOCKED UP. IN ADDITION A 'RESET' IS EXECUTED TO VERIFY THAT SRO IS CLEARED AND SR2 IS UNLOCKED BY A RESET. AFTER MEMORY MANAGEMENT IS TURNED BACK ON, SR2 IS CHECKED TO SEE THAT IT IS TRACKING AGAIN.

```

022476 000004
1488 022500 012767 000600 147644 TST7: SCOPE
1489 022506 012767 000406 147574 1$: MOV #600,KIPAR5 ;MAP KERNEL PAGE 5 TO 12-16K
1490 022514 012767 077402 147570 MOV #406,KIPDR4 ;SETUP PDR4 FOR PAGE LENGTH ABORT
1491 022522 012767 022530 156360 MOV #77402,KIPDR5 ;SETUP PDR5 FOR R/O ABORT
1492 022530 016767 155042 156532 2$: MOV #2$,$LPERR ;SET LOOP ON ERROR POINTER TO 2$
1493 022536 012701 022530 MOV SR2,WASSR2 ;READ SR2 TO SEE IF ITS TRACKING
1494 022542 020167 156522 MOV #2$,R1 ;PUT EXPECTED VIRTUAL PC IN R1
1495 022546 001401 CMP R1,WASSR2 ;DID SR2 CONTAIN VIRTUAL PC AT 2$?
1496 022550 104016 BEQ 3$ ;BRANCH IF YES
1497 ERROR +16 ;SR2 NOT TRACKING CORRECTLY
1498 ;FOR TIGHTER SCOPE LOOP
1499 ;REPLACE ERROR CALL WITH
; 'BR 2$' = 000767
1500 022552 012767 022560 156330 3$: MOV #4$,$LPERR ;SET LOOP ON ERROR POINTER TO 4$
1501 022560 016767 155012 156502 4$: MOV SR2,WASSR2 ;READ SR2 TO SEE IF ITS TRACKING
1502 022566 012701 022560 MOV #4$,R1 ;PUT EXPECTED VIRTUAL PC IN R1
1503 022572 020167 156472 CMP R1,WASSR2 ;DID SR2 CONTAIN VIRTUAL PC AT 4$?
1504 022576 001401 BEQ 5$ ;BRANCH IF YES
1505 022600 104016 ERROR +16 ;SR2 NOT TRACKING CORRECTLY
1506 ;FOR TIGHTER SCOPE LOOP
1507 ;REPLACE ERROR CALL WITH
; 'BR 4$' = 000767
1508
1509 022602 012767 022610 156300 5$: MOV #6$,$LPERR ;SET LOOP ON ERROR POINTER TO 6$
1510 022610 012767 022626 155432 6$: MOV #7$,MMVEC ;PUT ADDRESS OF 7$ IN M.M. TRAP VECTOR
1511 022616 005067 156356 CLR $TMP1 ;CLEAR ERROR INDICATOR
1512 022622 005237 100500 INC @#100500 ;CAUSE PAGE LENGTH ABORT - TRAP TO 7$
1513 022626 012706 001100 7$: MOV #KERSTK,KSP ;RESTORE STACK POINTER AFTER ABORT
1514 022632 016767 154734 156336 MOV SRO,$TMP0 ;SAVE SRO'S INFORMATION ON PG. LGTH. ABORT
1515 022640 016767 154732 156334 MOV SR2,$TMP2 ;SAVE SR2'S INFORMATION ON PG. LGTH. ABORT
1516 022646 012767 022660 155374 MOV #8$,MMVEC ;PUT ADDRESS OF 8$ IN M.M. TRAP VECTOR
1517 022654 005237 120000 INC @#120000 ;CAUSE R/O ABORT - TRAP TO 8$
1518 022660 012706 001100 8$: MOV #KERSTK,KSP ;RESTORE STACK POINTER AFTER ABORT
    
```

```

1519 022664 016767 154702 156372      MOV      SRO,WASSRO      ;READ SRO FOLLOWING SECOND KT ABORT?
1520 022672 016767 154700 156370      MOV      SR2,WASSR2     ;READ SR2 FOLLOWING SECOND KT ABORT
1521 022700 026767 156272 156356      CMP      $TMP0,WASSRO   ;IS SRO STILL HOLDING INFO ON FIRST ABORT?
1522 022706 001402                BEQ      9$              ;BRANCH IF YES
1523 022710 005267 156264                INC      $TMP1          ;SET ERROR INDICATOR
1524 022714 026767 156262 156346 9$:   CMP      $TMP2,WASSR2   ;DOES SR2 STILL HOLD PC OF FIRST ABORT?
1525 022722 001402                BEQ      10$            ;BRANCH IF YES
1526 022724 005267 156250                INC      $TMP1          ;SET ERROR INDICATOR
1527 022730 005767 156244                TST     $TMP1           ;WERE SRO OR SR2 CHANGED BY A SECOND ABORT?
1528 022734 001401                BEQ      11$            ;BRANCH IF NO
1529 022736 104014                ERROR   +14            ;ONE OF STATUS REGS. CHANGED BY SECOND ABORT
1530                                ;FOR TIGHTER SCOPE LOOP
1531                                ;REPLACE ERROR CALL WITH
1532                                ;'BR 6$' = 000726
1533 022740 005067 156234                11$:   CLR      $TMP1          ;CLEAR ERROR INDICATOR
1534 022744 000005                RESET                                ;EXECUTE A RESET, APPLYING AN 'INIT'
1535 022746 016767 154620 156310      MOV      SRO,WASSRO     ;READ SRO
1536 022754 005767 156304                TST     WASSRO          ;WAS SRO CLEARED BY THE RESET?
1537 022760 001402                BEQ      12$            ;BRANCH IF YES
1538 022762 005267 156212                INC      $TMP1          ;SRO NOT CLEARED BY A RESET
1539 022766 016767 154604 156274 12$:  MOV      SR2,WASSR2     ;READ SR2
1540 022774 022767 022766 156266      CMP      #12$,WASSR2    ;WAS SR2 UNLOCKED BY A RESET?
1541 023002 001402                BEQ      13$            ;BRANCH IF YES
1542 023004 005267 156170                INC      $TMP1          ;SR2 NOT UNLOCKED BY A RESET
1543 023010 005767 156164                13$:   TST     $TMP1           ;WERE SRO & SR2 BOTH 'RESET' BY A RESET?
1544 023014 001401                BEQ      14$            ;BRANCH IF YES
1545 023016 104015                ERROR   +15            ;SRO OR SR2 NOT 'RESET' BY A RESET
1546                                ;FOR TIGHTER SCOPE LOOP
1547                                ;REPLACE ERROR CALL WITH
1548                                ;'BR 6$' = 000676
1549 023020 005267 154546                14$:   INC      SRO           ;TURN MEMORY MANAGEMENT BACK ON
1550 023024 016767 154546 156236 15$:  MOV      SR2,WASSR2     ;READ SR2 TO SEE IF ITS TRACKING AGAIN
1551 023032 012701 023024                MOV      #15$,R1        ;PUT EXPECTED VIRTUAL PC IN R1
1552 023036 020167 156226                CMP      R1,WASSR2      ;DID SR2 CONTAIN VIRTUAL PC AT 15$
1553 023042 001401                BEQ      16$            ;BRANCH IF YES
1554 023044 104016                ERROR   +16            ;SR2 NOT TRACKING CORRECTLY
1555                                ;FOR TIGHTER SCOPE LOOP
1556                                ;REPLACE ERROR CALL WITH
1557                                ;'BR 6$' = 000663
1558 023046 012767 022500 156034 16$:  MOV      #1$, $LPERR     ;RESET LOOP ON ERROR POINTER TO 1$
1559 023054 012767 077406 147226      MOV      #77406,KIPDR4 ;RESET PDR4 TO 128 BLKS, R/W
1560 023062 012767 077406 147222      MOV      #77406,KIPDR5 ;RESET PDR5 TO 128 BLKS, R/W
1561 023070 012767 002466 155152      MOV      #MMGMERR,MMVEC ;RESTORE ADDRESS OF NORMAL MEMORY
1562                                ;MANAGEMENT TRAP ROUTINE TO M.M. VECTOR
1563
1577
  
```



1579

\*\*\*\*\*  
\*TEST 10 SUPER/USER ABORT PICKS UP KERNEL VECTOR  
\*\*\*\*\*

THIS TEST CHECKS TO BE SURE THAT WHEN AN ABORT OCCURS WHILE  
IN SUPERVISOR OR USER MODE, THE TRAP VECTOR INFORMATION  
FETCHED IS TAKEN FROM KERNEL SPACE. USER PAGE 0 IS MAPPED  
TO 12K (60000-77776) SO THAT IF USER SPACE IS USED INSTEAD  
OF KERNEL, THE NEW PC THAT WAS LOADED AT LOC. 060004 IS USED  
INSTEAD OF THE NEW PC THAT SHOULD BE PICKED UP FROM LOC. 000004.  
THE SUPERVISOR PAGE 0 IS THEN MAPPED TO 12K, AND THE TEST  
IS REPEATED FOR SUPERVISOR MODE. AN ODD ADDRESS ERROR IS  
USED TO CAUSE A TRAP TO '4'.

\*\*\*\*\*  
TST10: SCOPE  
\*\*\*\*\*

1580 023076 000004  
1581 023100 004767 157222  
1582 023104 012767 023112 155776  
1583 023112 005067 154660  
1584 023116 012706 001100  
1585  
1586  
1587 023122 012767 000600 154510  
1588 023130 012737 023212 000004  
1589 023136 012737 000340 000006  
1590 023144 012767 140000 154624  
1591 023152 012706 000600  
1592 023156 012737 023176 000004  
1593 023164 012737 000340 000006  
1594 023172 005767 000001  
1595  
1596  
1597 023176 016701 154574  
1598 023202 010602  
1599 023204 005067 154566  
1600 023210 104017  
1601  
1602  
1603  
1604 023212 005067 154560  
1605 023216 012706 001100  
1606 023222 005067 154412  
1607 023226 012767 140000 154542  
1608 023234 012706 000600  
1609 023240 005067 154532  
1610  
1611  
1612  
1613 023244 012767 000600 146766  
1614 023252 012737 023334 000004  
1615 023260 012737 000340 000006  
1616 023266 012767 040000 154502  
1617 023274 012706 000700  
1618 023300 012737 023320 000004  
1619 023306 012737 000340 000006  
1620 023314 005767 000001  
162

```

1$: JSR PC,TOFF ;TURN OFF T-BIT TRAPPING FOR THIS TEST
    MOV #2$, $LPIERR ;SET LOOP ON ERROR POINTER TO 2$
2$: CLR PSW ;GO TO KERNEL MODE
    MOV #KERSTK, KSP ;SETUP KERNEL STACK PTR.

;
; TEST USER MODE ABORT
;
    MOV #600, UIPARO ;MAP USER PAGE 0 TO 12K
    MOV #4$, @#4 ;LOAD KERNEL VECTOR 4 (LOC.4) WITH 4$
    MOV #340, @#6 ;LOAD VECTOR+2 WITH NEW PSW
    MOV #140000, PSW ;GO TO USER MODE
    MOV #USESTK, USP ;SETUP USER STACK PTR.
    MOV #3$, @#4 ;LOAD USER VECTOR 4 (LOC. 60004) WITH 3$
    MOV #340, @#6 ;LOAD VECTOR+2 WITH NEW PSW
    TST 3$+1 ;CAUSE ODD ADDR. ERROR TRAP TO '4'
                    ;SHOULD PICK UP NEW PC=4$ FROM KERNEL
                    ;LOC. 4, NOT PC=3$ FROM USER LOC. 4 (=60004)
3$: MOV PSW, R1 ;SAVE PSW FOR ERROR
    MOV SP, R2 ;SAVE VALUE OF STACK POINTER FOR ERROR
    CLR PSW ;BE SURE BACK IN KERNEL MODE
    ERROR +17 ;DID NOT TRAP THRU KERNEL SPACE
                    ;FOR TIGHTER SCOPE LOOP
                    ;REPLACE ERROR CALL WITH
                    ;'BR 2$' = 000740
4$: CLR PSW ;BE SURE BACK IN KERNEL MODE
    MOV #KERSTK, KSP ;RESTORE KERNEL S.P. IN CASE IT CHANGED
    CLR UIPARO ;REMAP USER PAGE 0 TO 0-4K
    MOV #140000, PSW ;GO TO USER MODE
    MOV #USESTK, USP ;RESTORE USER STACK POINTER
    CLR PSW ;GO BACK TO KERNEL MODE

;
; NOW TEST THE SUPERVISOR MODE ABORT
;
    MOV #600, SIPARO ;MAP SUPERVISOR PAGE 0 TO 12K
    MOV #6$, @#4 ;LOAD KERNAL VECTOR 4 WITH 6$
    MOV #340, @#6 ;LOAD VECTOR+2 WITH NEW PSW
    MOV #40000, PSW ;GO TO SUPERVISOR MODE
    MOV #SUPSTK, SSP ;SETUP SUPERVISOR STACK PTR.
    MOV #5$, @#4 ;LOAD SUPERVISOR VECTOR 4 (LOC. 60004) WITH 5$
    MOV #340, @#6 ;LOAD VECTOR+2 WITH NEW PSW
    TST 5$+1 ;CAUSE ODD ADDR. ERROR TRAP TO '4'
                    ;SHOULD PICK UP NEW PC=6$ FROM KERNEL

```

```

1622                                     ;LOC. 4, NOT PC=5$ FROM SUPERVISOR LOC. 4 (=60004)
1623 023320 016701 154452 5$: MOV PSW,R1 ;SAVE PSW FOR ERROR
1624 023324 010602         MOV SP,R2 ;SAVE VALUE OF STACK POINTER FOR ERROR
1625 023326 005067 154444   CLR PSW ;BE SURE BACK IN KERNEL MODE
1626 023332 104017         ERROR +17 ;DID NOT TRAP THRU KERNEL SPACE
1627                                     ;FOR TIGHTER SCOPE LOOP
1628                                     ;REPLACE ERROR CALL WITH
1629                                     ;'BR 2$' = 000740
1630 023334 005067 154436 6$: CLR PSW ;BE SURE BACK IN KERNEL MODE
1631 023340 012706 001100   MOV #KERSTK,KSP ;RESTORE KERNEL S.P. IN CASE IT CHANGED
1632 023344 005067 146670   CLR SIPARO ;REMAP SUPERVISOR PAGE 0 TO 0-4K
1633 023350 012767 040000 154420 MOV #40000,PSW ;GO TO SUPERVISOR MODE
1634 023356 012706 000700   MOV #SUPSTK,SSP ;RESTORE SUPERVISOR STACK POINTER
1635 023362 005067 154410   CLR PSW ;GO BACK TO KERNEL MODE
1636 023366 012737 002414 000004 MOV #TIMERR,@#4 ;RESTORE ADDR. OF NORMAL CPU TRAP HANDLER TO 4
1637 023374 012767 023100 155506 MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
1638 023402 004767 156754   JSR PC,T0N ;TURN T-BIT TRAPPING BACK ON
1639
1646

```

```

:*****
:*TEST 11 RTI IN SUPER/USER MODE DOES NOT CHANGE PSW
:*
:* THIS TEST CHECKS TO SEE THAT WHEN AN RTI IS EXECUTED IN SUPERVISOR
:* OR USER MODE, THE MODE OR PRIORITY BITS OF THE PSW ARE NOT CHANGED.
:*
:*****

```

```

023406 000004
1647
1648 023410 012767 023422 155472 1$: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
1649 023416 012702 170000   MOV #170000,R2 ;LOAD 'PRESENT & EXPECTED' PSW VALUE INTO R2
1650 023422 010267 154350 2$: MOV R2,PSW ;GO TO PRESENT MODE-PRIORITY 0
1651 023426 012746 000340   MOV #340,-(SP) ;PUT A NEW PSW (PRIORITY=7) ON STACK
1652 023432 012746 023440   MOV #3$,-(SP) ;PUT NEW PC ON THE STACK
1653 023436 000002         RTI ;DO AN RTI FROM PRESENT MODE
1654 023440 016701 154332 3$: MOV PSW,R1 ;READ NEW PSW INTO R1
1655 023444 042701 007437   BIC #7437,R1 ;MASK OFF COND. CODE, T-BIT, AND UNUSED BITS
1656 023450 005067 154322   CLR PSW ;GO BACK TO KERNEL MODE
1657 023454 020201         CMP R2,R1 ;DID PSW STAY IN PRESENT MODE, PRIORITY=0?
1658 023456 001401         BEQ 4$ ;BRANCH IF YES
1659 023460 104032         ERROR +32 ;PSW CHANGED BY AN RTI FROM USER
1660                                     ;FOR A TIGHTER SCOPE LOOP
1661                                     ;REPLACE ERROR CALL WITH
1662                                     ;'BR=2$' = 000760
1663 023462 022702 050000 4$: CMP #50000,R2 ;IF SUPERVISOR MODE HAS BEEN CHECKED,
1664 023466 001403         BEQ 5$ ;GO TO END OF TEST
1665 023470 012702 050000   MOV #50000,R2 ;ELSE, SET SUPERVISOR MODE,
1666 023474 000752         BR 2$ ;AND BRANCH BACK TO TEST IT.
1667 023476 012767 023410 155404 5$: MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
1668

```

1681

\*\*\*\*\*  
 \*TEST 12 KT ERROR NOT SERVICED IF ODD ADDR. ERROR  
 \*

THIS TEST CHECKS TO SEE THAT IF A CERTAIN VIRTUAL ADDRESS THAT  
 WOULD CAUSE A MEMORY MANAGEMENT ERROR CAUSES AN ODD ADDRESS  
 ERROR FIRST, THE ODD ADDRESS ERROR IS SERVICED BUT THE MEMORY  
 MANAGEMENT ERROR ISN'T. THIS MEANS THAT SRO AND SR2  
 SHOULD NOT REPORT THE ERROR OR LOCK UP ITS VIRTUAL ADDRESS.  
 A READ-ONLY VIOLATION IS USED AS THE POTENTIAL MEMORY MANAGEMENT  
 ERROR

\*\*\*\*\*

|      |        |        |        |        |                      |  |  |
|------|--------|--------|--------|--------|----------------------|--|--|
| 1682 | 023504 | 000004 |        |        | TST12: SCOPE         |  |  |
| 1683 | 023506 | 012767 | 000600 | 146634 | 1\$: MOV #600,KIPDR4 |  | :MAP KERNEL PAGE 4 TO 12-16K                 |
| 1684 | 023514 | 012705 | 077402 |        | MOV #77402,R5        |  | :LOAD PDR4 DATA INTO R5                      |
| 1685 | 023520 | 010567 | 146564 |        | MOV R5,KIPDR4        |  | :MAP PAGE 4 READ-ONLY                        |
| 1686 | 023524 | 012737 | 023554 | 000004 | MOV #4\$,@#4         |  | :SET CPU TRAP VECTOR TO ADDRESS OF 4\$       |
| 1687 | 023532 | 012737 | 023552 | 000250 | MOV #3\$,@#250       |  | :SET M.M. TRAP VECTOR TO ADDRESS OF 3\$      |
| 1688 | 023540 | 012767 | 023546 | 155342 | MOV #2\$, \$LPERR    |  | :SET LOOP ON ERROR POINTER TO 2\$            |
| 1689 | 023546 | 005267 | 034227 |        | 2\$: INC 60001       |  | :CAUSE ODD ADDR. ERROR & POTENTIAL R/O ABORT |
| 1690 | 023552 | 104020 |        |        | 3\$: ERROR +20       |  | :TRAPPED THRU M.M. VECTOR BUT SHOULDN'T HAVE |
| 1691 |        |        |        |        |                      |  | :FOR TIGHTER SCOPE LOOP                      |
| 1692 |        |        |        |        |                      |  | :REPLACE ERROR CALL WITH                     |
| 1693 | 023554 | 012706 | 001100 |        | 4\$: MOV #KERSTK,KSP |  | :RESTORE STACK POINTER AFTER TRAPPING        |
| 1694 | 023560 | 005067 | 155414 |        | CLR \$TMP1           |  | :CLEAR ERROR INDICATOR                       |
| 1695 | 023564 | 016767 | 154002 | 155472 | MOV SRO,WASSRO       |  | :READ STATUS REG. 0                          |
| 1696 | 023572 | 016767 | 154000 | 155470 | 5\$: MOV SR2,WASSR2  |  | :READ STATUS REG. 2                          |
| 1697 | 023600 | 012700 | 000017 |        | MOV #17,R0           |  | :LOAD EXPECTED SRO CONTENTS INTO R0          |
| 1698 | 023604 | 020067 | 155454 |        | CMP R0,WASSRO        |  | :SRO ERROR BITS LEFT CLEAR BY TRAPPING?      |
| 1699 | 023610 | 001402 |        |        | BEQ 6\$              |  | :BRANCH IF YES                               |
| 1700 | 023612 | 005267 | 155362 |        | INC \$TMP1           |  | :SRO ERROR BITS SET WHEN ODD ADDR. SERVICED  |
| 1701 | 023616 | 012701 | 023572 |        | 6\$: MOV #5\$,R1     |  | :LOAD EXPECTED SR2 CONTENTS INTO R1          |
| 1702 | 023622 | 020167 | 155442 |        | CMP R1,WASSR2        |  | :WAS SR2 LEFT UNLOCKED BY TRAPPING?          |
| 1703 | 023626 | 001402 |        |        | BEQ 7\$              |  | :BRANCH IF YES                               |
| 1704 | 023630 | 005267 | 155344 |        | INC \$TMP1           |  | :SR2 LOCKED UP BY ODD ADDR. ERROR            |
| 1705 | 023634 | 005767 | 155340 |        | 7\$: TST \$TMP1      |  | :WHERE SRO OR SR2 EFFECTED?                  |
| 1706 | 023640 | 001404 |        |        | BEQ 8\$              |  | :BRANCH IF NO                                |
| 1707 | 023642 | 104021 |        |        | ERROR +21            |  | :SRO OR SR2 CHANGED BY ODD ADDR. ERROR       |
| 1708 |        |        |        |        |                      |  | :FOR TIGHTER SCOPE LOOP                      |
| 1709 |        |        |        |        |                      |  | :REPLACE ERROR CALL WITH                     |
| 1710 |        |        |        |        |                      |  | : 'BR 2\$' = 000741                          |
| 1711 | 023644 | 042767 | 160000 | 153720 | 8\$: BIC #160000,SRO |  | :CLEAR ERROR BITS THAT MAY BE SET IN SRO     |
| 1712 | 023652 | 012737 | 002414 | 000004 | MOV #TIMERR,@#4      |  | :RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER  |
| 1713 | 023660 | 012737 | 002466 | 000250 | MOV #MGMERR,@#250    |  | :RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER |
| 1714 | 023666 | 012767 | 077406 | 146414 | MOV #77406,KIPDR4    |  | :REMAP PAGE 4 TO READ/WRITE                  |
| 1715 | 023674 | 012767 | 023506 | 155206 | MOV #1\$, \$LPERR    |  | :RESET LOOP ON ERROR POINTER TO 1\$          |
| 1716 |        |        |        |        |                      |  |  |
| 1731 |        |        |        |        |                      |  |  |
| 1732 |        |        |        |        |                      |  |  |

\*\*\*\*\*  
 \*TEST 13 PC & PSW SAVED FOR KT ERROR ON ODD ADDR.  
 \*

THIS TEST CHECKS THE PC AND PROCESSOR STATUS WORD SAVED WHEN  
 A KT ERROR OCCURS DURING THE SECOND PUSH ON THE STACK DURING  
 SERVICING OF AN ODD ADDR. ERROR. DURING A 'DOUBLE ERROR'  
 SEQUENCE SUCH AS THIS, THE PSW SAVED WILL BE THE ONE PICKED UP  
 FROM VECTOR+2 (LOC. 6 IN THIS CASE) AFTER THE FIRST TRAP.

NOT THE PSW PRESENT BEFORE THE FIRST TRAP. SRO AND SR2 SHOULD RECORD THE KT ERROR (A R/O VIOLATION BY THE USER STACK PTR.)  
 NOTE THAT THE PREVIOUS MODE BITS <13:12> OF THE PSW WILL BE SET IN THE PSW THAT IS SAVED.

```

*****
TST13: SCOPE
1733 023702 000004 156416 153730 1$: JSR PC,TOFF ;TURN T-BIT TRAPPING OFF FOR THIS TEST
1734 023704 004767 000600 153724 MOV #600,UIPAR3 ;MAP USER PAGE 3 TO 12-16K
1735 023710 012767 000600 153724 MOV #600,UIPAR4 ;MAP USER PAGE 4 TO 12-16K
1736 023716 012767 077402 153654 MOV #77402,UIPDR3 ;MAP USER PAGE 3 READ-ONLY
1737 023722 012767 077406 153650 MOV #77406,UIPDR4 ;MAP USER PAGE 4 READ/WRITE
1738 023728 012737 024014 000004 MOV #4$,a#4 ;LOAD ADDRESS OF 4$ IN CPU (ODD ADDR.) VECTOR
1739 023734 012737 140017 000006 MOV #140017,a#6 ;LOAD PSW THAT SHOULD BE PUT ON STACK IN VECTOR+2
1740 023740 012737 024014 000250 MOV #4$,a#250 ;LOAD ADDRESS OF 4$ IN M.M. TRAP VECTOR
1741 023746 012737 000340 000252 MOV #340,a#252 ;LOAD A KERNEL PSW IN MMVEC+2
1742 023752 012767 023776 155112 MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
1743 023758 012767 140000 153772 2$: MOV #140000,PSW ;GO TO USER MODE
1744 024004 012706 100002 3$: MOV #100002,USP ;SET USER STACK PTR. SO SECOND PUSH IS IN PG. 3
1745 024010 005267 053771 3$: INC 100005 ;CAUSE ODD ADDRESS ERROR THAT WILL CAUSE
1746 1746 ;R/O ERROR WHEN TRY TO SAVE OLD PC
1747 024014 016601 000002 4$: MOV 2(KSP),R1 ;PUT PSW SAVED ON KERNEL STACK INTO R1
1748 024020 011603 MOV (KSP),R3 ;PUT PC SAVED ON KERNEL STACK INTO R3
1749 024022 016767 153544 155234 MOV SRO,WASSRO ;READ THE CONTENTS OF M.M. STATUS REG. 0
1750 024028 016767 153542 155232 MOV SR2,WASSR2 ;READ THE CONTENTS OF M.M. STATUS REG. 2
1751 024034 042767 160000 153526 BIC #160000,SRO ;CLEAR THE ERROR BITS IN SRO
1752 024040 005067 153726 CLR PSW ;BE SURE IN KERNEL MODE
1753 024046 012706 001100 MOV #KERSTK,KSP ;RESTORE KERNEL STACK POINTER
1754 024052 012767 140000 153714 MOV #140000,PSW ;GO TO USER MODE
1755 024058 012706 000600 MOV #USESTK,USP ;RESTORE USER STACK POINTER
1756 024064 005067 153704 CLR PSW ;GO BACK TO KERNEL MODE
1757 024070 005067 155100 CLR $TMP0 ;CLEAR ERROR INDICATOR
1758 024076 020127 170017 CMP R1,#170017 ;WAS THE PSW SAVED THE ONE PICKED UP BY THE
1759 ;ODD ADDR. TRAP FROM ERRVEC+2?
1760 ;VALUE 170017 = PSW FROM LOC. 6 WITH
1761 ;PREVIOUS MODE BITS = USER
1762 024102 001402 BEQ 5$ ;BRANCH IF YES
1763 024104 005267 155066 INC $TMP0 ;WRONG PSW SAVED DURING 'DOUBLE ERROR' SEQUENCE
1764 024110 020327 024014 5$: CMP R3,#3$+4 ;WAS THE PC AT THE TIME OF THE ODD ADDR. ERROR
1765 ;SAVED ON THE STACK?
1766 024114 001402 BEQ 6$ ;BRANCH IF YES
1767 024116 005267 155054 INC $TMP0 ;WRONG PC SAVED DURING TRAP SEQUENCE
1768 024122 026727 155136 020147 6$: CMP WASSRO,#20147 ;DID SRO REPORT - USER, PAGE 3, R/O ABORT?
1769 024128 001402 BEQ 7$ ;BRANCH IF YES
1770 024134 005267 155040 INC $TMP0 ;SRO DID NOT REPORT R/O ABORT
1771 024140 026727 155126 024010 7$: CMP WASSR2,#3$ ;DID SR2 LOCK UP VIRTUAL ADDR. OF LAST
1772 ;INSTRUCTION SUCCESSFULLY FETCHED?
1773 024144 001402 BEQ 8$ ;BRANCH IF YES
1774 024146 005267 155024 INC $TMP0 ;SR2 DID NOT LOCK UP ADDR. OF ODD ADDR. INST.
1775 024152 005767 155020 8$: TST $TMP0 ;ANY 'ERRORS' DURING TRAP SEQUENCE?
1776 024158 001401 BEQ 9$ ;BRANCH IF NO
1777 024160 104022 ERROR +22 ;THE WRONG PC OR PSW WERE SAVED
1778 ;OR SRO OR SR2 DID NOT REPORT R/O
1779 ;ERROR DURING ODD ADDR. - KT TRAP
1780 ;SEQUENCE
1781 ;FOR TIGHTER SCOPE LOOP

```

|      |        |        |        |        |      |     |               |  |  |  |  |
|------|--------|--------|--------|--------|------|-----|---------------|--|--|--|--|
| 1782 |        |        |        |        |      |     |               |  |  |  | :REPLACE ERROR CALL WITH                     |
| 1783 |        |        |        |        |      |     |               |  |  |  | : 'BR 2\$' = 000710                          |
| 1784 | 024162 | 012737 | 002414 | 000004 | 9\$: | MOV | #TIMERR,@#4   |  |  |  | :RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER  |
| 1785 | 024170 | 012737 | 000340 | 000006 |      | MOV | #340,@#6      |  |  |  | :RELOAD ERRVEC+2 WITH KERNEL PSW             |
| 1786 | 024176 | 012737 | 002466 | 000250 |      | MOV | #MGMERR,@#250 |  |  |  | :RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER |
| 1787 | 024204 | 012767 | 077406 | 153374 |      | MOV | #77406,U1PDR3 |  |  |  | :REMAP USER PAGE 3 READ/WRITE                |
| 1788 | 024212 | 012767 | 023704 | 154670 |      | MOV | #1\$,SLPERR   |  |  |  | :RESET LOOP ON ERROR POINTER TO 1\$          |
| 1789 | 024220 | 004767 | 156136 |        |      | JSR | PC,TON        |  |  |  | :TURN T-BIT TRAPPING BACK ON                 |



|      |        |        |        |        |      |                                       |                                    |
|------|--------|--------|--------|--------|------|---------------------------------------|------------------------------------|
| 1838 | 024334 | 005300 |        |        | DEC  | R0                                    |                                    |
| 1839 | 024336 | 006000 |        |        | ROR  | R0                                    |                                    |
| 1840 | 024340 | 006100 |        |        | ROL  | R0                                    |                                    |
| 1841 | 024342 | 005067 | 153224 |        | CLR  | MMRO                                  | ;TURN OFF MEMORY MANAGEMENT        |
| 1842 |        |        |        | ::*    | TEST | SOB NON-MOD WITH DSTM=2,3,6,7; DSTF=7 |                                    |
| 1843 |        |        |        | ::*    |      |                                       |                                    |
| 1844 | 024346 | 012737 | 100002 | 060000 | MOV  | #100002,@#60000                       | ;SET UP TEST LOCATION              |
| 1845 | 024354 | 012767 | 024366 | 154526 | MOV  | #11\$, \$LPERR                        | ;SET LOOP ON ERROR POINTER TO 11\$ |
| 1846 | 024362 | 005267 | 153204 |        | INC  | MMRO                                  | ;TURN ON MEMORY MANAGEMENT         |



```

1848 024366 005727 060000      11$:  TST      #60000      :DSTM=2 SOP NON-MOD
1849 024372 005737 100000      TST      @#100000    :DSTM=3 SOP NON-MOD
1850 024376 005767 053376      TST      100000     :DSTM=6 SOP NON-MOD
1851 024402 005777 053372      TST      @100000    :DSTM=7 SOP NON-MOD
1852 024406 005067 153160      CLR      MMRO       :TURN OFF MEMORY MANAGEMENT
1853          :*      TEST SOB MOD WITH DSTM=2,3,6,7; DSTF=7
1854          :*
1855 024412 012767 024432 154470      MOV      #12$, $LPERR :SET LOOP ON ERROR POINTER TO 12$
1856 024420 012737 100000 060002      MOV      #100000, @#60002 :SET UP TEST VALUES
1857 024426 005267 153140      INC      MMRO       :TURN ON MEMORY MANAGEMENT
1858 024432 005027 060000      12$:  CLR      #60000     :DSTM=2 SOP MOD
1859 024436 005037 100000      CLR      @#100000    :DSTM=3 SOP MOD
1860 024442 005067 053332      CLR      100000     :DSTM=6 SOP MOD
1861 024446 005077 053330      CLR      @100002    :DSTM=7 SOP MOD
1862 024452 005067 153114      CLR      MMRO       :TURN OFF MEMORY MANAGEMENT
1863          :*
1864          :*      THE NEXT THREE TESTS ARE CONCERNED WITH TESTING
1865          :*      DOP AND NOT(SRCM=DSTM=0)
1866          :*
1867          :*      TEST DOP DEST NON-MOD SRCM=DSTM=2,3,6,7; SRCF=DSTF=7
1868          :*
1869 024456 012767 024476 154424      MOV      #13$, $LPERR :SET LOOP ON ERROR POINTER TO 13$
1870 024464 012737 100000 060002      MOV      #100000, @#60002 :SET UP TEST VALUE
1871 024472 005267 153074      INC      MMRO       :TURN ON MEMORY MANAGEMENT
1872 024476 022727 060000 060000      13$:  CMP      #60000, #60000 :SRCM=2 DSTM=2 DOP NON-MOD
1873 024504 023737 100000 100000      CMP      @#100000, @#100000 :SRCM=3 DSTM=3 DOP NON-MOD
1874 024512 026767 053262 053260      CMP      100000, 100000 :SRCM=6 DSTM=6 DOP NON-MOD
1875 024520 027777 053256 053254      CMP      @100002, @100002 :SRCM=7 DSTM=7 DOP NON-MOD
1876 024526 005067 153040      CLR      MMRO       :TURN OFF MEMORY MANAGEMENT
1877          :*      TEST MOV DEST AND NOT(SRCM=DSTM=0)
1878          :*      SRCM=DSTM=2,3,6,7; SRCF=DSTF=7
1879          :*
1880 024532 012767 024552 154350      MOV      #14$, $LPERR :SET LOOP ON ERROR POINTER TO 14$
1881 024540 012737 100000 060002      MOV      #100000, @#60002 :SET UP TEST VALUE
1882 024546 005267 153020      INC      MMRO       :TURN ON MEMORY MANAGEMENT
1883 024552 012727 060002 060002      14$:  MOV      #60002, #60002 :SRCM=2 DSTM=2 MOV
1884 024560 012737 060000 100000      MOV      #60000, @#100000 :SRCM=2 DSTM=3 MOV
1885 024566 012767 060000 053204      MOV      #60000, 100000 :SRCM=2 DSTM=6 MOV
1886 024574 012777 060000 053200      MOV      #60000, @100002 :SRCM=2 DSTM=7 MOV
1887 024602 005067 152764      CLR      MMRO       :TURN OFF MEMORY MANAGEMENT
1888          :*      TEST DOP DEST MOD AND NOT SUB
1889          :*      SRCM=DSTM=2,3,6,7; SRCF=DSTF=7
1890          :*
1891 024606 012767 024620 154274      MOV      #15$, $LPERR :SET LOOP ON ERROR POINTER TO 15$
1892 024614 005267 152752      INC      MMRO       :TURN ON MEMORY MANAGEMENT
1893 024620 052727 060000 060000      15$:  BIS      #60000, #60000 :SRCM=2 DSTM=2 DOP MOD
1894 024626 052737 000000 100000      BIS      #00000, @#100000 :SRCM=2 DSTM=3 DOP MOD
1895 024634 052767 000000 053136      BIS      #00000, 100000 :SRCM=2 DSTM=6 DOP MOD
1896 024642 052777 000000 053132      BIS      #00000, @100002 :SRCM=2 DSTM=7 DOP MOD
1897 024650 005067 152716      CLR      MMRO       :TURN OFF MEMORY MANAGEMENT
1898          :*      TEST SWAB WITH DSTM=2,3,6,7; DSTF=7
1899          :*
1900 024654 012767 024674 154226      MOV      #16$, $LPERR :SET LOOP ON ERROR POINTER TO 16$
1901 024662 012737 100002 060000      MOV      #100002, @#60000 :SET UP TEST VALUES
1902 024670 005267 152676      INC      MMRO       :TURN ON MEMORY MANAGEMENT
1903 024674 000327 060000      16$:  SWAB     #60000     :DSTM=2 SWAB
1904 024700 000337 100002      SWAB     @#100002    :DSTM=3 SWAB
  
```

```

1905 024704 000367 053072          SWAB 100002          :DSTM=6 SWAB
1906 024710 000377 053064          SWAB @100000        :DSTM=7 SWAB
1907 024714 005067 152652          CLR  MMRO           :TURN OFF MEMORY MANAGEMENT
1908          :*          TEST ROT/SHFT WITH DSTM=2,3,6,7; DSTF=7
1909          :*
1910 024720 012767 024740 154162    MOV  #17$, $LPERR    :SET LOOP ON ERROR POINTER TO 17$
1911 024726 012737 100000 060002    MOV  #100000, @#60002 :SET UP TEST VALUES
1912 024734 005267 152632          INC  MMRO           :TURN ON MEMORY MANAGEMENT
1913 024740 006127 060000 17$:   ROL  #60000          :DSTM=2 ROT/SHFT
1914 024744 006137 100000          ROL  @#100000        :DSTM=3 ROT/SHFT
1915 024750 006167 053024          ROL  100000          :DSTM=6 ROT/SHFT
1916 024754 006177 053022          ROL  @100002        :DSTM=7 ROT/SHFT
1917 024760 005067 152606          CLR  MMRO           :TURN OFF MEMORY MANAGEMENT
1918          :*          TEST ASH/ASHC WITH DSTM=2,3,6,7; DSTF=7
1919          :*
1920 024764 012767 025012 154116    MOV  #18$, $LPERR    :SET LOOP ON ERROR POINTER TO 18$
1921 024772 012737 000001 060000    MOV  #1, @#60000     :SET UP TEST VALUES
1922 025000 012737 100000 060002    MOV  #100000, @#60002
1923 025006 005267 152560          INC  MMRO           :TURN ON MEMORY MANAGEMENT
1924 025012 072027 000001 18$:   ASH  #1, R0          :DSTM=2 ASH/ASHC
1925 025016 072037 100000          ASH  @#100000, R0    :DSTM=3 ASH/ASHC
1926 025022 072067 052752          ASH  100000, R0     :DSTM=6 ASH/ASHC
1927 025026 072077 052750          ASH  @100002, R0    :DSTM=7 ASH/ASHC
1928 025032 005067 152534          CLR  MMRO           :TURN OFF MEMORY MANAGEMENT
1929          :*          TEST MUL/DIV WITH DSTM=2,3,6,7; DSTF=7
1930          :*
1931 025036 012767 025050 154044    MOV  #19$, $LPERR    :SET LOOP ON ERROR POINTER TO 19$
1932 025044 005267 152522          INC  MMRO           :TURN ON MEMORY MANAGEMENT
1933 025050 070027 000002 19$:   MUL  #2, R0          :DSTM=2 MUL/DIV
1934 025054 070037 100000          MUL  @#100000, R0    :DSTM=3 MUL/DIV
1935 025060 070067 052714          MUL  100000, R0     :DSTM=6 MUL/DIV
1936 025064 070077 052712          MUL  @100002, R0    :DSTM=7 MUL/DIV
1937 025070 005067 152476          CLR  MMRO           :TURN OFF MEMORY MANAGEMENT
1938          :*          TEST JMP WITH DSTM=3,6,7; DSTF=7
1939          :*
1940 025074 012767 025114 154006    MOV  #20$, $LPERR    :SET LOOP ON ERROR POINTER TO 20$
1941 025102 012737 025130 060000    MOV  #23$, @#60000   :SET UP TEST VALUES
1942 025110 005267 152456          INC  MMRO           :TURN ON MEMORY MANAGEMENT
1943 025114 000137 025120 20$:   JMP  @#21$          :DSTM=3 JMP
1944 025120 000167 000000 21$:   JMP  22$           :DSTM=6 JMP
1945 025124 000177 052650 22$:   JMP  @100000        :DSTM=7 JMP
1946 025130 005067 152436 23$:   CLR  MMRO           :TURN OFF MEMORY MANAGEMENT
1947          :*          TEST SUB WITH DSTM=2,3,6,7; DSTF=7
1948          :*
1949 025134 012767 025156 153746    MOV  #28$, $LPERR    :SET LOOP ON ERROR POINTER TO 28$
1950 025142 005000          CLP  R0             :SET UP TEST VALUES
1951 025144 012737 100000 060002    MOV  #100000, @#60002
1952 025152 005267 152414          INC  MMRO           :TURN ON MEMORY MANAGEMENT
1953 025156 160027 060002 28$:   SUB  R0, #60002     :DSTM=2 SUB
1954 025162 160037 100000          SUB  R0, @#100000    :DSTM=3 SUB
1955 025166 160067 052606          SUB  R0, 100000     :DSTM=6 SUB
1956 025172 160077 052604          SUB  R0, @100002    :DSTM=7 SUB
1957 025176 005067 152370          CLR  MMRO           :TURN OFF MEMORY MANAGEMENT
1958 025202 012767 024312 153700    MOV  #10$, $LPERR    :SET LOOP ON ERROR POINTER TO START OF TEST
1969          :*****
          :*TEST 15          ENABLE D-SPACE AND SEE I-SPACE IS NOT FORCED
          :*
  
```

THIS TEST SHOWS THAT I-SPACE IS NOT FORCED IF THE REGISTER FIELD IS NOT 7, BUT THE OTHER CONDITIONS ARE MET.

ALL ERRORS FOUND IN THIS TEST ARE REPORTED WHEN THE CPU ABORTS THROUGH 'MMVEC' TO SUBROUTINE 'MODSPAC'. THIS SUBROUTINE WILL REPORT THAT D-SPACE WAS NOT ENABLED PROPERLY.

\*\*\*\*\*

```

025210 000004
1970 025212 012700 077406
1971 025216 010067 145066
1972 025222 010067 145074
1973 025226 010067 145072
1974 025232 010067 145070
1975 025236 105067 145044
1976
1977
1978 025242 012700 060000
1979 025246 010037 060000
1980 025252 012737 060002 060002
1981 025260 012737 060004 060004
1982 025266 012737 060006 060006
1983 025274 012767 025336 153606
1984 025302 012737 060000 060000
1985 025310 012737 060002 060002
1986 025316 012737 060004 060004
1987 025324 012737 060006 060006
1988 025332 005267 152234
1989 025336 005710
1990 025340 005720
1991 025342 005730
1992 025344 005750
1993 025346 005770 000000
1994 025352 005067 152214
1995
1996
1997 025356 012767 025370 153524
1998 025364 005267 152202
1999 025370 005010
2000 025372 005020
2001 025374 005030
2002 025376 005050
2003 025400 005070 000000
2004 025404 005067 152162
2005
2006
2007
2008 025410 012767 025452 153472
2009 025416 012702 000032
2010 025422 012700 060000
2011 025426 012701 060000
2012 025432 010021
2013 025434 062700 000002
2014 025440 077204
2015 025442 012700 060000
2016 025446 005267 152120
2017 025452 021010
  
```

```

TST15: SCOPE
MOV #77406,R0
MOV R0,KIPDR4 ;MAKE KIPDR4 R/W,4K,200 BLOCKS
MOV R0,KDPDR1 ;MAKE KDPDR1 R/W,4K,200 BLOCKS
MOV R0,KDPDR2 ;MAKE KDPDR2 R/W,4K,200 BLOCKS
MOV R0,KDPDR3 ;MAKE KDPDR3 R/W,4K,200 BLOCKS
CLRB KIPDR3 ;MAKE KIPDR3 NON-RESIDENT
TEST SOP NON-MOD; DSTM=1,2,3,5,7

20$: MOV #60000,R0 ;SET UP CONSTANTS FOR TEST
MOV R0,@#60000
MOV #60002,@#60002
MOV #60004,@#60004
MOV #60006,@#60006
MOV #1$, $LPERR ;SET LOOP ON ERROR POINTER TO 1$
MOV #60000,@#60000
MOV #60002,@#60002
MOV #60004,@#60004
MOV #60006,@#60006
INC MMR0 ;TURN ON MEMORY MANAGEMENT
1$: TST (R0) ;DSTM=1 SOP NON-MOD
TST (R0)+ ;DSTM=2 SOP NON-MOD
TST @ (R0)+ ;DSTM=3 SOP NON-MOD
TST @-(R0) ;DSTM=5 SOP NON-MOD
TST @0(R0) ;DSTM=7 SOP NON-MOD
CLR MMR0 ;TURN OFF MEMORY MANAGEMENT
TEST SOP MOD; DSTM=1,2,3,5,7

2$: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
INC MMR0 ;TURN ON MEMORY MANAGEMENT
CLR (R0) ;DSTM=1 SOP MOD
CLR (R0)+ ;DSTM=2 SOP MOD
CLR @ (R0)+ ;DSTM=3 SOP MOD
CLR @-(R0) ;DSTM=5 SOP MOD
CLR @0(R0) ;DSTM=7 SOP MOD
CLR MMR0 ;TURN OFF MEMORY MANAGEMENT
TEST DOP DEST NON-MOD WITH SRCM=1,2,3,5,7 AND DSTM=1,2,3,5,7
ALL SOURCE MODES TO BE TESTED ARE TESTED HERE

3$: MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
MOV #32,R2 ;SET UP ADDRESSES 60000-60064 FOR TEST
MOV #60000,R0
MOV #60000,R1
21$: MOV R0,(R1)+
ADD #2,R0
SOB R2,21$
MOV #60000,R0
INC MMR0 ;TURN ON MEMORY MANAGEMENT
3$: CMP (R0),(R0) ;SRCM=1 DSTM=1 DOP DEST NON-MOD
  
```

|      |        |        |        |        |      |      |                |  |
|------|--------|--------|--------|--------|------|------|----------------|--|
| 2018 | 025454 | 021020 |        |        |      | CMP  | (R0),(R0)+     | ;SRCM=1 DSTM=2 DOP DEST NON-MOD                |
| 2019 | 025456 | 021030 |        |        |      | CMP  | (R0),a-(R0)+   | ;SRCM=1 DSTM=3 DOP DEST NON-MOD                |
| 2020 | 025460 | 021050 |        |        |      | CMP  | (R0),a-(R0)    | ;SRCM=1 DSTM=5 DOP DEST NON-MOD                |
| 2021 | 025462 | 021070 | 000000 |        |      | CMP  | (R0),a0(R0)    | ;SRCM=1 DSTM=7 DOP DEST NON-MOD                |
| 2022 | 025466 | 022010 |        |        |      | CMP  | (R0)+,(R0)     | ;SRCM=2 DSTM=1 DOP DEST NON-MOD                |
| 2023 | 025470 | 022020 |        |        |      | CMP  | (R0)+,(R0)+    | ;SRCM=2 DSTM=2 DOP DEST NON-MOD                |
| 2024 | 025472 | 022030 |        |        |      | CMP  | (R0)+,a(R0)+   | ;SRCM=2 DSTM=3 DOP DEST NON-MOD                |
| 2025 | 025474 | 022050 |        |        |      | CMP  | (R0)+,a-(R0)   | ;SRCM=2 DSTM=5 DOP DEST NON-MOD                |
| 2026 | 025476 | 022070 | 000000 |        |      | CMP  | (R0)+,a0(R0)   | ;SRCM=2 DSTM=7 DOP DEST NON-MOD                |
| 2027 | 025502 | 023010 |        |        |      | CMP  | a(R0)+,(R0)    | ;SRCM=3 DSTM=1 DOP DEST NON-MOD                |
| 2028 | 025504 | 023020 |        |        |      | CMP  | a(R0)+,(R0)+   | ;SRCM=3 DSTM=2 DOP DEST NON-MOD                |
| 2029 | 025506 | 023030 |        |        |      | CMP  | a(R0)+,a(R0)+  | ;SRCM=3 DSTM=3 DOP DEST NON-MOD                |
| 2030 | 025510 | 023050 |        |        |      | CMP  | a(R0)+,a-(R0)  | ;SRCM=3 DSTM=5 DOP DEST NON-MOD                |
| 2031 | 025512 | 023070 | 000000 |        |      | CMP  | a(R0)+,a0(R0)  | ;SRCM=3 DSTM=7 DOP DEST NON-MOD                |
| 2032 | 025516 | 025010 |        |        |      | CMP  | a-(R0),(R0)    | ;SRCM=5 DSTM=1 DOP DEST NON-MOD                |
| 2033 | 025520 | 025020 |        |        |      | CMP  | a-(R0),(R0)+   | ;SRCM=5 DSTM=2 DOP DEST NON-MOD                |
| 2034 | 025522 | 025030 |        |        |      | CMP  | a-(R0),a(R0)+  | ;SRCM=5 DSTM=3 DOP DEST NON-MOD                |
| 2035 | 025524 | 025050 |        |        |      | CMP  | a-(R0),a-(R0)  | ;SRCM=5 DSTM=5 DOP DEST NON-MOD                |
| 2036 | 025526 | 025070 | 000000 |        |      | CMP  | a-(R0),a0(R0)  | ;SRCM=5 DSTM=7 DOP DEST NON-MOD                |
| 2037 | 025532 | 027010 | 000000 |        |      | CMP  | a0(R0),(R0)    | ;SRCM=7 DSTM=1 DOP DEST NON-MOD                |
| 2038 | 025536 | 027020 | 000000 |        |      | CMP  | a0(R0),(R0)+   | ;SRCM=7 DSTM=2 DOP DEST NON-MOD                |
| 2039 | 025542 | 027030 | 000000 |        |      | CMP  | a0(R0),a(R0)+  | ;SRCM=7 DSTM=3 DOP DEST NON-MOD                |
| 2040 | 025546 | 027050 | 000000 |        |      | CMP  | a0(R0),a-(R0)  | ;SRCM=7 DSTM=5 DOP DEST NON-MOD                |
| 2041 | 025552 | 027070 | 000000 | 000000 |      | CMP  | a0(R0),a0(R0)  | ;SRCM=7 DSTM=7 DOP DEST NON-MOD                |
| 2042 | 025560 | 005067 | 152006 |        |      | CLR  | MMR0           | ;TURN OFF MEMORY MANAGEMENT                    |
| 2043 |        |        |        |        |      | ;    | *              | TEST DOP DEST MOD AND NOT SUB; DSTM=1,2,3,5,7  |
| 2044 |        |        |        |        |      | ;    | *              |  |
| 2045 | 025564 | 005000 |        |        |      | CLR  | R0             | ;SET UP CONSTANTS FOR TEST                     |
| 2046 | 025566 | 012701 | 060000 |        |      | MOV  | #60000,R1      |  |
| 2047 | 025572 | 012767 | 025604 | 153310 |      | MOV  | #4\$, \$LPERR  | ;SET LOOP ON ERROR POINTER TO 4\$              |
| 2048 | 025600 | 005267 | 151766 |        |      | INC  | MMR0           | ;TURN ON MEMORY MANAGEMENT                     |
| 2049 | 025604 | 050011 |        |        | 4\$: | BIS  | R0,(R1)        | ;DSTM=1 DOP DEST MOD                           |
| 2050 | 025606 | 050021 |        |        |      | BIS  | R0,(R1)+       | ;DSTM=2 DOP DEST MOD                           |
| 2051 | 025610 | 050031 |        |        |      | BIS  | R0,a(R1)+      | ;DSTM=3 DOP DEST MOD                           |
| 2052 | 025612 | 050051 |        |        |      | BIS  | R0,a-(R1)      | ;DSTM=5 DOP DEST MOD                           |
| 2053 | 025614 | 050071 | 000000 |        |      | BIS  | R0,a0(R1)      | ;DSTM=7 DOP DEST MOD                           |
| 2054 | 025620 | 005067 | 151746 |        |      | CLR  | MMR0           | ;TURN OFF MEMORY MANAGEMENT                    |
| 2055 |        |        |        |        |      | ;    | *              | TEST MOV DEST AND NOT(SMO AND DMO); DSTM=3,5,7 |
| 2056 |        |        |        |        |      | ;    | *              |  |
| 2057 | 025624 | 012701 | 060000 |        |      | MOV  | #60000,R1      | ;SET UP CONSTANTS FOR TEST                     |
| 2058 | 025630 | 012737 | 060002 | 060000 |      | MOV  | #60002,a#60000 |  |
| 2059 | 025636 | 012767 | 025650 | 153244 |      | MOV  | #5\$, \$LPERR  | ;SET LOOP ON ERROR POINTER TO 5\$              |
| 2060 | 025644 | 005267 | 151722 |        |      | INC  | MMR0           | ;TURN ON MEMORY MANAGEMENT                     |
| 2061 | 025650 | 010031 |        |        | 5\$: | MOV  | R0,a(R1)+      | ;DSTM=3 MOV DEST                               |
| 2062 | 025652 | 010051 |        |        |      | MOV  | R0,a-(R1)      | ;DSTM=5 MOV DEST                               |
| 2063 | 025654 | 010071 | 000000 |        |      | MOV  | R0,a0(R1)      | ;DSTM=7 MOV DEST                               |
| 2064 | 025660 | 005067 | 151706 |        |      | CLR  | MMR0           | ;TURN OFF MEMORY MANAGEMENT                    |
| 2065 |        |        |        |        |      | ;    | *              | TEST SWAB; DSTM=1,2,3,5,7                      |
| 2066 |        |        |        |        |      | ;    | *              |  |
| 2067 | 025664 | 012701 | 060000 |        |      | MOV  | #60000,R1      | ;SET UP CONSTANTS FOR TEST                     |
| 2068 | 025670 | 012767 | 025702 | 153212 |      | MOV  | #6\$, \$LPERR  | ;SET LOOP ON ERROR POINTER TO 6\$              |
| 2069 | 025676 | 005267 | 151670 |        |      | INC  | MMR0           | ;TURN ON MEMORY MANAGEMENT                     |
| 2070 | 025702 | 000311 |        |        | 6\$: | SWAB | (R1)           | ;DSTM=1 SWAB                                   |
| 2071 | 025704 | 000321 |        |        |      | SWAB | (R1)+          | ;DSTM=2 SWAB                                   |
| 2072 | 025706 | 000331 |        |        |      | SWAB | a(R1)+         | ;DSTM=3 SWAB                                   |
| 2073 | 025710 | 000351 |        |        |      | SWAB | a-(R1)         | ;DSTM=5 SWAB                                   |
| 2074 | 025712 | 000371 | 000000 |        |      | SWAB | a0(R1)         | ;DSTM=7 SWAB                                   |

```

2075 025716 005067 151650          CLR      MMR0          ;TURN OFF MEMORY MANAGEMENT
2076          :*          TEST ROT/SHFT; DSTM=1,2,3,5,7
2077          :*
2078 025722 012701 060000          MOV      #60000,R1     ;SET UP CONSTANTS FOR TEST
2079 025726 012702 060006          MOV      #60006,R2
2080 025732 010203          MOV      R2,R3
2081 025734 012737 060000 060000  MOV      #60000,@#60000
2082 025742 012737 060002 060002  MOV      #60002,@#60002
2083 025750 012737 060004 060004  MOV      #60004,@#60004
2084 025756 012737 060006 060006  MOV      #60006,@#60006
2085 025764 012767 025776 153116  MOV      #7$, $LPERR   ;SET LOOP ON ERROR POINTER TO 7$
2086 025772 005267 151574          INC      MMR0          ;TURN ON MEMORY MANAGEMENT
2087 025776 006111          7$: ROL      (R1)       ;DSTM=1 ROT/SHFT
2088 026000 006121          ROL      (R1)+        ;DSTM=2 ROT/SHFT
2089 026002 006131          ROL      @ (R1)+      ;DSTM=3 ROT/SHFT
2090 026004 006152          ROL      @-(R2)       ;DSTM=5 ROT/SHFT
2091 026006 006173 000000          ROL      @0(R3)       ;DSTM=7 ROT/SHFT
2092 026012 005067 151554          CLR      MMR0          ;TURN OFF MEMORY MANAGEMENT
2093          :*          TEST MUL/DIV; DSTM=1,2,3,5,7
2094          :*
2095 026016 012767 026032 153064  MOV      #8$, $LPERR   ;SET LOOP ON ERROR POINTER TO 8$
2096 026024 005003          CLR      R3           ;SET UP CONSTANT FOR TEST
2097 026026 005267 151540          INC      MMR0          ;TURN ON MEMORY MANAGEMENT
2098 026032 070310          8$: MUL      (R0),R3    ;DSTM=1 MUL/DIV
2099 026034 070320          MUL      (R0)+,R3     ;DSTM=2 MUL/DIV
2100 026036 070330          MUL      @ (R0)+,R3   ;DSTM=3 MUL/DIV
2101 026040 070350          MUL      @-(R0),R3    ;DSTM=5 MUL/DIV
2102 026042 070370 000000          MUL      @0(R0),R3    ;DSTM=7 MUL/DIV
2103 026046 005067 151520          CLR      MMR0          ;TURN OFF MEMORY MANAGEMENT
2104          :*          TEST ASH/ASHC; DSTM=1,2,3,5,7
2105          :*
2106 026052 012737 000001 060000  MOV      #1,@#60000    ;SET UP CONSTANTS FOR THE TEST
2107 026060 012737 060000 060002  MOV      #60000,@#60002
2108 026066 012700 060000          MOV      #60000,R0
2109 026072 012767 026104 153010  MOV      #9$, $LPERR   ;SET LOOP ON ERROR POINTER TO 9$
2110 026100 005267 151466          INC      MMR0          ;TURN ON MEMORY MANAGEMENT
2111 026104 072310          9$: ASH      (R0),R3    ;DSTM=1 ASH/ASHC
2112 026106 072320          ASH      (R0)+,R3     ;DSTM=2 ASH/ASHC
2113 026110 072330          ASH      @ (R0)+,R3   ;DSTM=3 ASH/ASHC
2114 026112 072350          ASH      @-(R0),R3    ;DSTM=5 ASH/ASHC
2115 026114 072370 000000          ASH      @0(R0),R3    ;DSTM=7 ASH/ASHC
2116 026120 005067 151446          CLR      MMR0          ;TURN OFF MEMORY MANAGEMENT
2117          :*          TEST JMP; DSTM=3,7
2118          :*
2119 026124 012767 026156 152756  MOV      #10$, $LPERR  ;SET LOOP ON ERROR POINTER TO 10$
2120 026132 012737 026160 060000  MOV      #11$,@#60000  ;SET UP CONSTANTS FOR THE TEST
2121 026140 012737 026164 060002  MOV      #12$,@#60002
2122 026146 012701 060000          MOV      #60000,R1
2123 026152 005267 151414          INC      MMR0          ;TURN ON MEMORY MANAGEMENT
2124 026156 000131          10$: JMP      @ (R1)+     ;DSTM=3 JMP
2125 026160 000171 000000          11$: JMP      @0(R1)    ;DSTM=7 JMP
2126 026164 005067 151402          12$: CLR      MMR0          ;TURN OFF MEMORY MANAGEMENT
2127          :*          TEST JSR; DSTM=3,7
2128          :*
2129 026170 012767 026222 152712  MOV      #13$, $LPERR  ;SET LOOP ON ERROR POINTER TO 13$
2130 026176 012737 026224 060000  MOV      #14$,@#60000  ;SET UP CONSTANTS FOR THE TEST
2131 026204 012737 026230 060002  MOV      #15$,@#60002
  
```

```

2132 026212 012701 060000      MOV    #60000,R1
2133 026216 005267 151350      INC    MMR0           ;TURN ON MEMORY MANAGEMENT
2134 026222 004731          13$:   JSR    PC,@(R1)+   ;DSTM=3 JSR
2135 026224 004771 000000      14$:   JSR    PC,@(R1)  ;DSTM=7 JSR
2136 026230 005067 151336      15$:   CLR    MMR0           ;TURN OFF MEMORY MANAGEMENT
2137 026234 112767 000006 144044  MOVB   #6,KIPDR3     ;MAKE KIPDR3 RESIDENT
2138 026242 012706 001100      MOV    #KERSTK,KSP   ;RESET STACK POINTER
2139 026246 012767 025242 152634  MOV    #20$,$LPERR   ;SET LOOP ON ERROR POINTER TO START OF TEST
2140
2149
  
```

```

*****
*TEST 16      PROPER ENABLING OF SUPER. D-SPACE
*
*      THIS TEST CHECKS FOR PROPER ENABLING OF THE SUPERVISOR D-SPACE.
*
*      ANY ERRORS ENCOUNTERED WILL BE REPORTED THROUGH 'MMVEC' TO
*      SUBROUTINE 'NODSPAC'.
*****
  
```

```

026254 000004
2150 026256 105067 143740      TST16: SCOPE
2151 026262 012767 026310 152620 20$:   CLRB   SDPDR1       ;MAKE SDPDR1 NON-RESIDENT
2152 026270 012767 000022 144220  MOV    #1$,$LPERR   ;SET LOOP ON ERROR POINTER TO 1$
2153 026276 052767 040000 151472  MOV    #22,MMR3     ;ENABLE 22-BIT SUPERVISOR D-SPACE
2154 026304 005267 151262      BIS    #40000,PSW   ;ENABLE SUPERVISOR MODE
2155      INC    MMR0           ;TURN ON MEMORY MANAGEMENT
2156      ;*      THE NEXT INSTRUCTIONS SHOULD NEVER INVOKE D-SPACE
2157 026310 000400      1$:   BR     2$
2158 026312 005700      2$:   TST    R0
2159 026314 005200      INC    R0
2160 026316 005067 151250      CLR    MMR0           ;TURN OFF MEMORY MANAGEMENT
2161 026322 112767 000006 143672  MOVB   #6,SDPDR1    ;MAKE SDPDR1 RESIDENT
2162 026330 105067 143652      CLRB   SIPDR3       ;MAKE SIPDR3 NON-RESIDENT
2163      ;*      TEST SOP INSTRUCTIONS
2164      ;*
2165 026334 012767 026346 152546  MOV    #3$,$LPERR   ;SET LOOP ON ERROR POINTER TO 3$
2166 026342 005267 151224      INC    MMR0           ;TURN ON MEMORY MANAGEMENT
2167 026346 005737 060000      3$:   TST    @#60000    ;DSTM=3 DSTF=7 SOP NON-MOD
2168 026352 005037 060000      CLR    @#60000      ;DSTM=3 DSTF=7 SOP MOD
2169 026356 005067 151210      CLR    MMR0           ;TURN OFF MEMORY MANAGEMENT
2170      ;*      TEST DOP INSTRUCTIONS
2171      ;*
2172 026362 012767 026400 152520  MOV    #4$,$LPERR   ;SET LOOP ON ERROR POINTER TO 4$
2173 026370 012700 060000      MOV    #60000,R0    ;SET UP CONSTANT FOR TEST
2174 026374 005267 151172      INC    MMR0           ;TURN ON MEMORY MANAGEMENT
2175 026400 023710 060000      4$:   CMP    @#60000,(R0) ;SRCM=3 DSTM=1 DOP DEST NON-MOD
2176 026404 052730 000000      BIS    #0,@(R0)+    ;SRCM=2 DSTM=3 DOP DEST MOD
2177 026410 013737 060002 060002  MOV    @#60002,@#60002 ;SRCM=3 DSTM=3 MOV DEST
2178 026416 005067 151150      CLR    MMR0           ;TURN OFF MEMORY MANAGEMENT
2179      ;*      TEST ROT/SHFT AND ASH/ASHC INSTRUCTIONS
2180      ;*
2181 026422 012767 026446 152460  MOV    #5$,$LPERR   ;SET LOOP ON ERROR POINTER TO 5$
2182 026430 012737 000001 060000  MOV    #1,@#60000   ;SET CONSTANT FOR TEST
2183 026436 012700 000001      MOV    #1,R0        ;SET UP CONSTANT FOR TEST
2184 026442 005267 151124      INC    MMR0           ;TURN ON MEMORY MANAGEMENT
2185 026446 006137 060000      5$:   ROL    @#60000    ;DSTM=3 ROT/SHFT
2186 026452 072337 060000      ASH    @#60000,R3   ;DSTM=3 ASH/ASHC
2187 026456 005067 151110      CLR    MMR0           ;TURN OFF MEMORY MANAGEMENT
  
```

```

2188          : *          TEST MUL/DIV AND SWAB INSTRUCTIONS
2189          : *
2190 026462 012767 026474 152420      MOV    #6$, $LPERR      ;SET LOOP ON ERROR POINTER TO 6$
2191 026470 005267 151076            INC    MMR0             ;TURN ON MEMORY MANAGEMENT
2192 026474 070037 060000      6$:   MUL    @#60000,R0      ;DSTM=3 MUL/DIV
2193 026500 000337 060004      SWAB   @#60004         ;DSTM=3 SWAB
2194 026504 005067 151062            CLR    MMR0             ;TURN OFF MEMORY MANAGEMENT
2195 026510 042767 000002 144000     BIC    #2,MMR3         ;DISABLE SUPERVISOR D-SPACE
2196 026516 112767 000006 143462     MOVB   #6,SIPDR3      ;MAKE SIPDR3 RESIDENT
2197 026524 012767 026256 152356     MOV    #20$, $LPERR   ;SET LOOP ON ERROR POINTER TO START OF TEST
2198
2207
  
```

```

*****
*TEST 17          PROPER ENABLING OF USER D-SPACE
*
*          THIS TEST CHECKS FOR PROPER ENABLING OF THE USER D-SPACE.
*
*          ANY ERRORS ENCOUNTERED WILL BE REPORTED THROUGH 'MMVEC' TO
*          SUBROUTINE 'MODSPAC'.
*****
  
```

```

          026532 000004
2208 026534 105067 151062      TST17: SCOPE
2209 026540 012767 026566 152342  20$:  CLRB   UDPDR1      ;MAKE UDPDR1 NON-RESIDENT
2210 026546 012767 000021 143742     MOV    #1$, $LPERR   ;SET LOOP ON ERROR POINTER TO 1$
2211 026554 052767 140000 151214     MOV    #21,MMR3      ;ENABLE 22-BIT USER D-SPACE
2212 026562 005267 151004      BIS    #140000,PSW   ;ENABLE USER MODE
2213          : *          THE NEXT INSTRUCTIONS SHOULD NEVER INVOKE D-SPACE
2214          : *
2215 026566 000400      1$:  BR     2$
2216 026570 005700      2$:  TST    R0
2217 026572 005200      INC    R0
2218 026574 005067 150772      CLR    MMR0          ;TURN OFF MEMORY MANAGEMENT
2219 026600 112767 000006 151014     MOVB   #6,JDPDR1     ;MAKE UDPDR1 RESIDENT
2220 026606 105067 150774      CLRB   UIPDR3        ;MAKE UIPDR3 NON-RESIDENT
2221          : *          TEST SOP INSTRUCTIONS
2222          : *
2223 026612 012767 026624 152270      MOV    #3$, $LPERR   ;SET LOOP ON ERROR POINTER TO 3$
2224 026620 005267 150746      INC    MMR0          ;TURN ON MEMORY MANAGEMENT
2225 026624 005737 060000      3$:  TST    @#60000     ;DSTM=3 DSTF=7 SOP NON-MOD
2226 026630 005037 060000      CLR    @#60000     ;DSTM=3 DSTF=7 SOP MOD
2227 026634 005067 150732      CLR    MMR0          ;TURN OFF MEMORY MANAGEMENT
2228          : *          TEST DOP INSTRUCTIONS
2229          : *
2230 026640 012767 026656 152242      MOV    #4$, $LPERR   ;SET LOOP ON ERROR POINTER TO 4$
2231 026646 012700 060000      MOV    #60000,R0     ;SET UP CONSTANT FOR TEST
2232 026652 005267 150714      INC    MMR0          ;TURN ON MEMORY MANAGEMENT
2233 026656 023710 060000      4$:  CMP    @#60000,(R0) ;SRCM=3 DSTM=1 DOP DEST NON-MOD
2234 026662 052730 000000      BIS    #0,@(R0)+    ;SRCM=2 DSTM=3 DOP DEST MOD
2235 026666 013737 060002 060002     MOV    @#60002,@#60002 ;SRCM=3 DSTM=3 MOV DEST
2236 026674 005067 150672      CLR    MMR0          ;TURN OFF MEMORY MANAGEMENT
2237          : *          TEST ROT/SHFT AND ASH/ASHC INSTRUCTIONS
2238          : *
2239 026700 012767 026720 152202      MOV    #5$, $LPERR   ;SET LOOP ON ERROR POINTER TO 5$
2240 026706 012737 000001 060000     MOV    #1,@#60000    ;SET CONSTANT FOR TEST
2241 026714 005267 150652      INC    MMR0          ;TURN ON MEMORY MANAGEMENT
2242 026720 006137 060000      5$:  ROL    @#60000     ;DSTM=3 ROT/SHFT
2243 026724 072337 060000      ASH    @#60000,R3    ;DSTM=3 ASH/ASHC
  
```



```

2244 026730 005067 150636 CLR MMR0 ;TURN OFF MEMORY MANAGEMENT
2245 ;* TEST MUL/DIV AND SWAB INSTRUCTIONS
2246 ;*
2247 026734 012767 026752 152146 MOV #6$,SLPERR ;SET LOOP ON ERROR POINTER TO 6$
2248 026742 012700 000001 MOV #1,R0 ;SET UP CONSTANT FOR TEST
2249 026746 005267 150620 INC MMR0 ;TURN ON MEMORY MANAGEMENT
2250 026752 070037 060000 6$: MUL @#60000,R0 ;DSTM=3 MUL/DIV
2251 026756 000337 060004 SWAB @#60004 ;DSTM=3 SWAB
2252 026762 005067 150604 CLR MMR0 ;TURN OFF MEMORY MANAGEMENT
2253 026766 042767 000001 143522 BIC #1,MMR3 ;DISABLE USER D-SPACE
2254 026774 112767 000006 150604 MOVB #6,UIPDR3 ;MAKE UIPDR3 RESIDENT
2255 027002 012767 026534 152100 MOV #20$,SLPERR ;SET LOOP ON ERROR POINTER TO START OF TEST
2256 027010 005067 150762 CLR PSW ;RESET TO KERNAL SPACE
2257
2266
  
```

```

:*****
:TEST 20 TRAPPING IN D-SPACE KERNAL MODE
:
: THIS TEST VERIFIES THAT THE ABORT VECTOR IS TAKEN FROM
: D-SPACE AND NOT I-SPACE. THE I-SPACE VECTOR POINTS TO
: 10$ AND THE D-SPACE VECTOR POINTS TO 15$. EACH PSW IN
: VIRTUAL 252 IS DIFFERENT SO THE PROGRAM CAN TELL WHICH
: AREA IT IS PICKED UP FROM.
:*****
  
```

```

027014 000004
2267 027016 004767 153304 JSR PC,TOFF ;TURN OF T-BIT FOR THIS TEST
2268 027022 012767 027056 152060 20$: MOV #1$,SLPERR ;SET LOOP ON ERROR POINTER TO 1$
2269 027030 012737 027124 000250 MOV #10$,@MMVEC ;SET M.M. VEC. TO HOLD BAD VECTOR
2270 027036 005037 000252 CLR @MMVEC+2 ;PSW IN 252 HAS PRIORITY OF ZERO
2271 027042 012737 027132 000350 MOV #15$,@#350 ;SET D-SPACE M.M VECTOR TO 350
2272 027050 012737 000340 000352 MOV #340,@#352 ;SET PSW PRIORITY TO ?
2273 027056 012706 001000 1$: MOV #1000,KSP ;SET UP KERNAL VECTOR
2274 027062 005267 150504 INC MMR0 ;TURN ON MEMORY MANAGEMENT
2275 ;NOW SET UP FOR AN ABORT IN KERNAL MODE WITH D-SPACE ENABLED
2276
2277 027066 012767 077402 143234 MOV #77402,KDPDR4 ;KERNAL D-SPACE PAGE 4 IS READ ONLY
2278 027074 012767 000600 143266 MOV #600,KDPAR4 ;MAP D-SPACE PAGE 4 TO 12K
2279 027102 052767 000004 143406 BIS #BIT2,MMR3 ;ENABLE KERNAL D-SPACE MAPPING
2280 027110 012767 000001 143242 MOV #1,KDPAR0 ;MAP KERNAL D PAGE 0 TO 000100
2281 027116 012737 177777 100000 MOV #-1,@#100000 ;TRY TO WRITE TO PAGE 4
2282 027124 016700 150646 10$: MOV PSW,R0 ;SAVE PSW FOR COMPARE
2283 027130 000402 BR '6$ ;BRANCH TO D-SPACE READ CODE
2284 027132 016700 150640 15$: MOV PSW,R0 ;SAVE PSW FOR COMPARE
2285 027136 005067 143216 16$: CLR KDPAR0 ;REMAP KERNAL D PAGE 0 TO PHYSICAL 0
2286 027142 012706 001100 MOV #KERSTK,KSP ;RESET STACK POINTER AFTER D-SPACE ABORT
2287 027146 042767 000004 143342 BIC #BIT2,MMR3 ;TURN OFF KERNAL D-SPACE ENABLE
2288 027154 016701 150412 MOV MMR0,R1 ;SAVE MMR0 FOR COMPARE
2289 027160 016702 150410 MOV MMR1,R2 ;SAVE MMR1 FOR COMPARE
2290 027164 016703 150406 MOV MMR2,R3 ;SAVE MMR2 FOR COMPARE
2291 027170 122700 000340 CMPB #340,R0 ;DID YOU PICK CORRECT PSW
2292 027174 001401 BEQ 2$ ;BRANCH IF PSW IS 340
2293 027176 104033 ERROR +33 ;WRONG PSW PICKED IN ABORT SEQUENCE
2294 027200 022701 020031 2$: CMP #020031,R1 ;EXPECTING READ ONLY ABORT
2295 ;KERNAL MODE D-SPACE PAGE 4
2296 027204 001401 BEQ 3$ ;BRANCH IF CONDITION IS CORRECT
2297 027206 104027 ERROR +27 ;WRONG M.M. ABORT CONDITION
2298 027210 005067 150356 3$: CLR MMR0 ;CLEAR OFF MMR0 FOR EXIT OF TEST
2299 027214 012767 002466 151026 MOV #MGMERR,MMVEC ;RESTORE NORMAL M.M. TRAP VECTOR
  
```

|      |        |        |        |        |     |              |   |
|------|--------|--------|--------|--------|-----|--------------|---|
| 2300 | 027222 | 012767 | 000340 | 151022 | MOV | #340,MMVEC+2 | ;RESTORE TRAP PSW (PRIORITY=7)              |
| 2301 | 027230 | 012767 | 027022 | 151652 | MOV | #20\$,SLPERR | ;SET LOOP ON ERROR PCINTER TO START OF TEST |
| 2302 |        |        |        |        |     |              |   |



```

2360 027366 012767 010340 150402 9$: MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
2361 027374 012702 100000 MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
2362 027400 006512 MFPI (R2) ;READ FROM PHYSICAL 60000
2363 027402 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
2364 027404 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
2365 027406 001401 BEQ 10$ ;BRANCH IF CORRECT DATA WAS FETCHED
2366 027410 104023 ERROR +23 ;WRONG DATA WAS FETCHED
2367 ;FOR TIGHTER SCOPE LOOP
2368 ;REPLACE ERROR CALL WITH
2369 ;'BR 9$' = 000766
2370 027412 10$: ;THE FOLLOWING WILL TEST DSTM=2 MFPI.
2371 027412 012767 027420 151470 MOV #11$,SLPERR ;SET LOOP ON ERROR POINTER TO 11$
2372 027420 012767 010340 150350 11$: MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
2373 027426 012702 100000 MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
2374 027432 006522 MFPI (R2)+ ;READ FROM PHYSICAL 60000
2375 027434 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
2376 027436 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
2377 027440 001401 BEQ 12$ ;BRANCH IF CORRECT DATA WAS FETCHED
2378 027442 104023 ERROR +23 ;WRONG DATA WAS FETCHED
2379 ;FOR TIGHTER SCOPE LOOP
2380 ;REPLACE ERROR CALL WITH
2381 ;'BR 11$' = 000766
2382 027444 12$: ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
2383 027444 012767 027452 151436 MOV #13$,SLPERR ;SET LOOP ON ERROR POINTER TO 13$
2384 027452 012767 010340 150316 13$: MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
2385 027460 006537 100000 MFPI @#100000 ;READ FROM PHYSICAL 60000
2386 027464 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
2387 027466 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
2388 027470 001401 BEQ 14$ ;BRANCH IF CORRECT DATA WAS FETCHED
2389 027472 104023 ERROR +23 ;WRONG DATA WAS FETCHED
2390 ;FOR TIGHTER SCOPE LOOP
2391 ;REPLACE ERROR CALL WITH
2392 ;'BR 13$' = 000767
2393 027474 14$: ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
2394 027474 012767 027502 151406 MOV #15$,SLPERR ;SET LOOP ON ERROR POINTER TO 15$
2395 027502 012767 010340 150266 15$: MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
2396 027510 012702 100002 MOV #100002,R2 ;LOAD VIRTUAL ADDRESS INTO R2
2397 027514 006542 MFPI -(R2) ;READ FROM PHYSICAL 60000
2398 027516 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
2399 027520 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
2400 027522 001401 BEQ 16$ ;BRANCH IF CORRECT DATA WAS FETCHED
2401 027524 104023 ERROR +23 ;WRONG DATA WAS FETCHED
2402 ;FOR TIGHTER SCOPE LOOP
2403 ;REPLACE ERROR CALL WITH
2404 ;'BR 15$' = 000766
2405 027526 16$: ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
2406 ;
2407 ;
2408 027526 012767 027534 151354 MOV #17$,SLPERR ;SET LOOP ON ERROR POINTER TO 17$
2409 027534 012767 010340 150234 17$: MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
2410 027542 012767 100000 151432 MOV #100000,$TMP2 ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
2411 027550 012702 001204 MOV #<$TMP2+2>,R2 ;LOAD ADDR. OF $TMP2+2 INTO R2
2412 027554 006552 MFPI @-(R2) ;READ FROM PHYSICAL 60000
2413 027556 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
2414 027560 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
2415 027562 001401 BEQ 18$ ;BRANCH IF CORRECT DATA WAS FETCHED
2416 027564 104023 ERROR +23 ;WRONG DATA WAS FETCHED
  
```

```

2417                                     ;FOR TIGHTER SCOPE LOOP
2418                                     ;REPLACE ERROR CALL WITH
2419                                     ;'BR 17$' = 000763
2420 U27566      18$:      ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
2421                                     ;
2422 027566 012767 027574 151314      MOV      #19$, $LPERR      ;SET LOOP ON ERROR POINTER TO 19$
2423 027574 012767 010340 150174 19$:  MOV      #010340, PSW      ;MAKE PREVIOUS MODE SUPERVISOR
2424 027602 005002                                     CLR      R2              ;MAKE REGISTER 2 A ZERO
2425 027604 006562 100000      MFPI     100000(R2)      ;READ FROM PHYSICAL 60000
2426 027610 012601                                     MOV      (KSP)+, R1      ;POP KERNEL STACK INTO R1
2427 027612 020001                                     CMP      R0, R1          ;WAS DATA FETCHED SAME AS STORED
2428 027614 001401      BEQ      20$              ;BRANCH IF CORRECT DATA WAS FETCHED
2429 027616 104023      ERROR     +23            ;WRONG DATA WAS FETCHED
2430                                     ;FOR TIGHTER SCOPE LOOP
2431                                     ;REPLACE ERROR CALL WITH
2432                                     ;'BR 19$' = 000766
2433 027620      20$:      ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
2434                                     ;
2435 027620 012767 027626 151262      MOV      #21$, $LPERR      ;SET LOOP ON ERROR POINTER TO 21$
2436 027626 012767 010340 150142 21$:  MOV      #010340, PSW      ;MAKE PREVIOUS MODE SUPERVISOR
2437 027634 012767 100000 151340      MOV      #100000, $TMP2   ;LOAD TEST LOC. V.A. INTO $TMP2
2438 027642 012702 001202                                     MOV      #$TMP2, R2      ;LOAD ADDRESS OF $TMP2 INTO R2
2439 027646 006572 000000      MFPI     @0(R2)          ;USE $TMP2 TO FETCH VIRTUAL
2440                                     ;ADDRESS OF 60000
2441 027652 012601                                     MOV      (KSP)+, R1      ;POP KERNEL STACK INTO R1
2442 027654 020001                                     CMP      R0, R1          ;WAS DATA FETCHED SAME AS STORED
2443 027656 001401      BEQ      22$              ;BRANCH IF CORRECT DATA WAS FETCHED
2444 027660 104023      ERROR     +23            ;WRONG DATA WAS FETCHED
2445                                     ;FOR TIGHTER SCOPE LOOP
2446                                     ;REPLACE ERROR CALL WITH
2447                                     ;'BR 21$' = 000762
2448 027662 012767 002466 150360 22$:  MOV      #MGMERR, MMVEC   ;SET M.M. VECTOR TO NORMAL ROUTINE
2449 027670 012767 027252 151212      MOV      #1$, $LPERR      ;SET LOOP POINTER TO START OF TEST
2450 027676 112767 000006 142404      MOV      #6, KIPDR4      ;MAKE KIPDR4 RESIDENT
2451 027704 000423      BR       TST22          ;BRANCH TO NEXT TEST
2452
2453
2454 027706 012667 151346      23$:  MOV      (KSP)+, TRAPPC   ;SAVE PC & PS OF TRAP
2455 027712 012667 151344      MOV      (KSP)+, TRAPPS   ;
2456 027716 016767 147650 151340      MOV      SRO, WASSRO     ;SAVE SRO FOR ERROR TYPEOUT
2457 027724 016767 147646 151336      MOV      SR2, WASSR2     ;SAVE SR2 FOR ERROR TYPEOUT
2458 027732 042767 160000 147632      BIC      #160000, SRO     ;CLEAR ERROR BITS IN SRO AND LEAVE
2459 027740 104026      ERROR     +26            ;TRIED TO READ NON-RESIDENT PAGE
2460                                     ;FOR TIGHTER SCOPE LOOP
2461                                     ;REPLACE ERROR CALL WITH
2462                                     ;A 'NOP' = 000240
2463 027742 016746 151314      MOV      TRAPPS, -(KSP)   ;PUT PC & PS OF TRAP ON STACK
2464 027746 016746 151306      MOV      TRAPPC, -(KSP)
2465 027752 000002      RTI
2466

```

2479

\*\*\*\*\*  
:TEST 22 MOVE FROM PREVIOUS (USER) I-SPACE

: THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT THE  
: PREVIOUS MODE IS CLOKED CORRECTLY  
: THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,

: IF THE CORRECT MODE (USER) IS NOT ENABLED A NON-RESIDENT ABORT  
: WILL OCCUR AND TRAP TO 23\$, WHERE THE ERRORS ARE REPORTED.

\*\*\*\*\*

2480 027754 000004  
2481 027756 012700 036514 147660  
2482 027762 012767 000600  
2483 027770 010037 100000  
2484 027774 012767 030404 150246  
2485 030002 105067 142302  
2486  
2487 030006 012767 030014 151074  
2488 030014 012767 030340 147754  
2489 030022 006506  
2490 030024 022706 001100

TST22: SCOPE  
1\$: MOV #36514,R0 ;LOAD DATA PATTERN INTO R0  
MOV #600,UIPAR4 ;MAP UIPAR4 TO 12K  
MOV R0,@#100000 ;LOAD DATA PATTERN INTO PHY 60000  
MOV #23\$,MMVEC ;SET M.M. VECTOR TO 23\$  
CLRB KIPDR4 ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT  
;THE FOLLOWING WILL TEST DSTM=0 MFPI  
:  
MOV #5\$,SLPERR ;SET LOOP ON ERROR POINTER TO 5\$  
5\$: MOV #030340,PSW ;MAKE PREVIOUS MODE USER  
6\$: MFPI USP ;PUT USER STACK POINTER ON KERNEL STACK  
CMP #KERSTK,KSP ;WAS SOMETHING PUSHED ON STACK AT 6\$

|      |        |        |       |            |                                  |
|------|--------|--------|-------|------------|----------------------------------|
| 2492 | 030030 | 001407 | BEQ   | 7\$        | :BRANCH IF NOTHING WAS PUSHED    |
| 2493 | 030032 | 012600 | MOV   | (KSP)+,R0  | :POP KERNEL STACK INTO R0        |
| 2494 | 030034 | 012701 | MOV   | #USESTK,R1 | :EXPECTING TO GET 600 AS USP     |
| 2495 | 030040 | 020001 | CMP   | R0,R1      | :DID YOU GET THE RIGHT POINTER?  |
| 2496 | 030042 | 001403 | BEQ   | 8\$        | :BRANCH IF YOU DID               |
| 2497 | 030044 | 104023 | ERROR | +23        | :WRONG THING WAS PUSHED ON STACK |
| 2498 |        |        |       |            | :FOR TIGHTER SCOPE LOOP          |
| 2499 |        |        |       |            | :REPLACE ERROR CALL WITH         |



```

2501
2502 030046 000401
2503 030050 104025 7$: BR 8$
2504 ERROR +25
2505
2506
2507 030052 8$: ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
2508 030052 012767 030064 151030 MOV #9$, $LPERR ;SET LOOP ON ERROR POINTER TO 9$
2509 030060 012700 036514 MOV #36514, R0 ;RELOAD DATA PATTERN IN R0
2510 030064 012767 030340 147704 9$: MOV #030340, PSW ;MAKE PREVIOUS MODE USER
2511 030072 012702 100000 MOV #100000, R2 ;LOAD VIRTUAL ADDRESS INTO R2
2512 030076 006512 MFPI (R2) ;READ FROM PHYSICAL 60000
2513 030100 012601 MOV (KSP)+, R1 ;POP KERNEL STACK INTO R1
2514 030102 020001 CMP R0, R1 ;WAS DATA FETCHED SAME AS STORED
2515 030104 001401 BEQ 10$ ;BRANCH IF CORRECT DATA WAS FETCHED
2516 030106 104023 ERROR +23 ;WRONG DATA WAS FETCHED
2517 ;FOR TIGHTER SCOPE LOOP
2518 ;REPLACE ERROR CALL WITH
2519 ;'BR 9$' = 000766
2520 030110 10$: ;THE FOLLOWING WILL TEST DSTM=2 MFPI.
2521 030110 012767 030116 150772 MOV #11$, $LPERR ;SET LOOP ON ERROR POINTER TO 11$
2522 030116 012767 030340 147652 11$: MOV #030340, PSW ;MAKE PREVIOUS MODE USER
2523 030124 012702 100000 MOV #100000, R2 ;LOAD VIRTUAL ADDRESS INTO R2
  
```

```
2525 030130 006522 MFPI (R2)+ ;READ FROM PHYSICAL 60000
2526 030132 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
2527 030134 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
2528 030136 001401 BEQ 12$ ;BRANCH IF CORRECT DATA WAS FETCHED
2529 030140 104023 ERROR +23 ;WRONG DATA WAS FETCHED
2530 ;FOR TIGHTER SCOPE LOOP
2531 ;REPLACE ERROR CALL WITH
2532 ;'BR 11$' = 000766
2533 030142 12$: ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
2534 030142 012767 030150 150740 MOV #13$,SLPERR ;SET LOOP ON ERROR POINTER TO 13$
2535 030150 012767 030340 147620 13$: MOV #030340,PSW ;MAKE PREVIOUS MODF USER
```

```
2537 030156 006537 100000 MFPI @100000 ;READ FROM PHYSICAL 60000
2538 030162 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
2539 030164 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
2540 030166 001401 BEQ 14$ ;BRANCH IF CORRECT DATA WAS FETCHED
2541 030170 104023 ERROR +23 ;WRONG DATA WAS FETCHED
2542 ;FOR TIGHTER SCOPE LOOP
2543 ;REPLACE ERROR CALL WITH
2544 ;'BR 13$' = 000767
2545 030172 14$: ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
```

```
2547 030172 012767 030200 150710      MOV    #15$, $LPERR      ;SET LOOP ON ERROR POINTER TO 15$
2548 030200 012767 030340 147570 15$:  MOV    #030340,PSW      ;MAKE PREVIOUS MODE USER
2549 030206 012702 100002                MOV    #100002,R2       ;LOAD VIRTUAL ADDRESS INTO R2
2550 030212 006542                        MFPI   -(R2)            ;READ FROM PHYSICAL 60000
2551 030214 012601                        MOV    (KSP)+,R1       ;POP KERNEL STACK INTO R1
2552 030216 020001                        CMP    R0,R1           ;WAS DATA FETCHED SAME AS STORED
2553 030220 001401                        BEQ    16$             ;BRANCH IF CORRECT DATA WAS FETCHED
2554 030222 104023                        ERROR  +23             ;WRONG DATA WAS FETCHED
2555                                     ;FOR TIGHTER SCOPE LOOP
2556                                     ;REPLACE ERROR CALL WITH
2557                                     ;'BR 15$' = 000766
2558 030224                                16$:
2559                                     ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
2560                                     ;
2561 030224 012767 030232 150656      MOV    #17$, $LPERR      ;SET LOOP ON ERROR POINTER TO 17$
2562 030232 012767 030340 147536 17$:  MOV    #030340,PSW      ;MAKE PREVIOUS MODE USER
```

2564 030240 012767 100000 150734           MOV    #100000,\$TMP2   ;LOAD TEST LOC. VIRT. ADDR INTO LOC. \$TMP2

2566 030246 012702 001204  
2567 030252 006552  
2568 030254 012601

MOV #<\$TMP2+2>,R2 ;LOAD ADDR. OF \$TMP2+2 INTO R2  
MFPI @-(R2) ;READ FROM PHYSICAL 60000  
MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1

```

2570 030256 020001      CMP      R0,R1      ;WAS DATA FETCHED SAME AS STORED
2571 030260 001401      BEQ      18$        ;BRANCH IF CORRECT DATA WAS FETLHED
2572 030262 104023      ERROR    +23       ;WRONG DATA WAS FETCHED
2573                                     ;FOR TIGHTER SCOPE LOOP
2574                                     ;REPLACE ERROR CALL WITH
2575                                     ;'BR 17$' = 000763
2576 030264          18$:      ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
2577                                     ;
2578 030264 012767 030272 150616      MOV      #19$,SLPERR ;SET LOOP ON ERROR POINTER TO 19$
2579 030272 012767 030340 147476 19$:  MOV      #030340,PSW ;MAKE PREVIOUS MODE USER
2580 030300 005002      CLR      R2         ;MAKE REGISTER 2 A ZERO
2581 030302 006562 100000      MFPI    100000(R2)  ;READ FROM PHYSICAL 60000
2582 030306 012601      MOV      (KSP)+,R1 ;POP KERNEL STACK INTO R1
2583 030310 020001      CMP      R0,R1      ;WAS DATA FETCHED SAME AS STORED
2584 030312 001401      BEQ      20$        ;BRANCH IF CORRECT DATA WAS FETCHED
2585 030314 104023      ERROR    +23       ;WRONG DATA WAS FETCHED
2586                                     ;FOR TIGHTER SCOPE LOOP
2587                                     ;REPLACE ERROR CALL WITH
2588                                     ;'BR 19$' = 000766
2589 030316          20$:      ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
2590                                     ;
2591 030316 012767 030324 150564      MOV      #21$,SLPERR ;SET LOOP ON ERROR POINTER TO 21$
2592 030324 012767 030340 147444 21$:  MOV      #030340,PSW ;MAKE PREVIOUS MODE USER
2593 030332 012767 100000 150642      MOV      #100000,$TMP2 ;LOAD TEST LOC. V.A. INTO $TMP2
2594 030340 012702 001202      MOV      #$TMP2,R2  ;LOAD ADDRESS OF $TMP2 INTO R2
2595 030344 006572 000000      MFPI    @0(R2)     ;USE $TMP2 TO FETCH VIRTUAL
2596                                     ;ADDRESS OF 60000
2597 030350 012601      MOV      (KSP)+,R1 ;POP KERNEL STACK INTO R1
2598 030352 020001      CMP      R0,R1      ;WAS DATA FETCHED SAME AS STORED
2599 030354 001401      BEQ      22$        ;BRANCH IF CORRECT DATA WAS FETCHED
2600 030356 104023      ERROR    +23       ;WRONG DATA WAS FETCHED
2601                                     ;FOR TIGHTER SCOPE LOOP
2602                                     ;REPLACE ERROR CALL WITH
2603                                     ;'BR 21$' = 000762
2604 030360 012767 002466 147662 22$:  MOV      #MGMERR,MMVEC ;SET M.M. VECTOR TO NORMAL ROUTINE
2605 030366 012767 027756 150514      MOV      #1$ ,      ;SET LOOP POINTER TO START OF TEST
2606 030374 112767 000006 141706      MOVVB   #6,KIPDR4  ;MAKE KIPDR4 RESIDENT
2607 030402 000423      BR      TST23      ;:BRANCH TO NEXT TEST
2608
2609
2610 030404 012667 150650          23$:  MOV      (KSP)+,TRAPPC ;SAVE PC & PS OF TRAP
2611 030410 012667 150646      MOV      (KSP)+,TRAPPS
2612 030414 016767 147152 150642      MOV      SR0,WASSRO ;SAVE SR0 FOR ERROR TYPEOUT
2613 030422 016767 147150 150640      MOV      SR2,WASSR2 ;SAVE SR2 FOR ERROR TYPEOUT
2614 030430 042767 160000 147134      BIC     #160000,SR0 ;CLEAR ERROR BITS IN SR0 AND LEAVE
2615 030436 104026      ERROR    +26       ;TRIED TO READ NON-RESIDENT PAGE
2616                                     ;FOR TIGHTER SCOPE LOOP
2617                                     ;REPLACE ERROR CALL WITH
2618                                     ;A 'NOP' = 000240
2619 030440 016746 150616      MOV      TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
2620 030444 016746 150610      MOV      TRAPPC,-(KSP)
2621 030450 000002      RTI
  
```



2623  
2624  
2636

\*\*\*\*\*  
\*TEST 23 MOVE TO PREVIOUS(SUPERVISOR) I-SPACE  
\*\*\*\*\*

\* THIS TEST USES THE 'MTPI' INSTRUCTION TO ENSURE THAT THE  
\* PREVIOUS MODE IS CLOKED CORRECTLY  
\* THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.  
\*\*\*\*\*

\* IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT  
\* WILL OCCUR AND TRAP TO 20\$, WHERE THE ERRORS ARE REPORTED.  
\*\*\*\*\*

|      |        |        |        |        |  |   |
|------|--------|--------|--------|--------|--|---|
| 2637 | 030452 | 000004 |        |        | TST23: SCOPE                               |   |
| 2638 | 030454 | 012767 | 077406 | 141526 | 1\$: MOV #77406,SIPDR4                     | ;SUPERVISOR I-SPACE PAGE 4 READ/WRITE     |
| 2639 | 030462 | 012767 | 031242 | 147560 | MOV #20\$,MVEC                             | ;SET M.M. VECTOR TO 20\$                  |
| 2640 |        |        |        |        |  | ;THE FOLLOWING WILL TEST DSTM=0 MTPI      |
| 2641 | 030470 | 012767 | 010340 | 147300 | 2\$: MOV #010340,PSW                       | ;MAKE PREVIOUS MODE SUPERVISOR            |
| 2642 | 030476 | 012746 | 007777 |        | MOV #7777,-(KSP)                           | ;PUSH DATA ON KERNEL STACK                |
| 2643 | 030502 | 006606 |        |        | MTPI SSP                                   | ;LOAD SUPERVISOR STACK POINTER            |
| 2644 | 030504 | 006506 |        |        | MFPI SSP                                   | ;READ SUPERVISOR STACK POINTER            |
| 2645 | 030506 | 012601 |        |        | MOV (KSP)+,R1                              | ;POP KERNEL STACK INTO R1                 |
| 2646 | 030510 | 022701 | 007777 |        | CMP #7777,R1                               | ;WAS SUPERVISOR STACK POINTER CHANGED     |
| 2647 | 030514 | 001401 |        |        | BEQ 3\$                                    | ;BRANCH IF IT WAS                         |
| 2648 | 030516 | 104025 |        |        | ERROR +25                                  | ;SUPERVISOR STACK POINTER NOT CHANGED     |
| 2649 |        |        |        |        |  | ;FOR TIGHTER SCOPE LOOP                   |
| 2650 |        |        |        |        |  | ;REPLACE ERROR CALL WITH                  |
| 2651 |        |        |        |        |  | ;'BR 2\$' = 000764                        |
| 2652 | 030520 | 012767 | 010340 | 147250 | 3\$: MOV #010340,PSW                       | ;MAKE PREVIOUS MODE SUPERVISOR            |
| 2653 | 030526 | 012746 | 000700 |        | MOV #SUPSTK,-(KSP)                         | ;GET READY TO RESTORE SUPERVISOR S. POINT |
| 2654 | 030532 | 006606 |        |        | MTPI SSP                                   | ;RESTORE SUPERVISOR STACK POINTER         |
| 2655 | 030534 |        |        |        |  | ;THIS WILL TEST DSTM = 1 MTPI.            |
| 2656 | 030534 | 012767 | 030552 | 150346 | MOV #5\$,SLPERR                            | ;SET LOOP ON ERROR POINTER TO 5\$         |
| 2657 | 030542 | 012702 | 100000 |        | MOV #100000,R2                             | ;LOAD VIRTUAL ADDRESS INTO R2             |
| 2658 | 030546 | 012700 | 125252 |        | MOV #125252,R0                             | ;LOAD TEST DATA INTO R0                   |
| 2659 | 030552 | 010046 |        |        | 5\$: MOV R0,-(KSP)                         | ;PUSH TEST DATA ON KERNEL STACK           |
| 2660 | 030554 | 105067 | 141530 |        | CLRB KIPDR4                                | ;MAKE KERNEL I PAGE 4 NON-RESIDENT        |
| 2661 | 030560 | 006612 |        |        | MTPI (R2)                                  | ;LOAD TEST DATA INTO PHYSICAL 60000       |
| 2662 | 030562 | 112767 | 000006 | 141520 | MOVB #006,KIPDR4                           | ;MAKE KERNEL PAGE 4 RESIDENT              |
| 2663 | 030570 | 011201 |        |        | MOV (R2),R1                                | ;READ FROM ADDRESS 60000                  |
| 2664 | 030572 | 020001 |        |        | CMP R0,R1                                  | ;SEE IF DATA WAS STORED AT CORRECT PLACE  |
| 2665 | 030574 | 001401 |        |        | BEQ 6\$                                    | ;BRANCH IF STORE WAS CORRECT              |
| 2666 | 030576 | 104024 |        |        | ERROR +24                                  | ;INCORRECT STORE                          |
| 2667 |        |        |        |        |  | ;FOR TIGHTER SCOPE LOOP                   |
| 2668 |        |        |        |        |  | ;REPLACE ERROR CALL WITH                  |
| 2669 |        |        |        |        |  | ;'BR 5\$' = 000765                        |
| 2670 | 030600 |        |        |        | 6\$: ;THE FOLLOWING WILL TEST DSTM=2 MTPI. |   |
| 2671 |        |        |        |        |  |   |
| 2672 | 030600 | 012767 | 030624 | 150302 | MOV #8\$,SLPERR                            | ;SET LOOP ON ERROR POINTER TO 8\$         |
| 2673 | 030606 | 012767 | 010340 | 147162 | MOV #010340,PSW                            | ;MAKE PREVIOUS MODE SUPERVISOR            |
| 2674 | 030614 | 012700 | 125252 |        | MOV #125252,R0                             | ;LOAD TEST DATA INTO R0                   |
| 2675 | 030620 | 012702 | 100000 |        | MOV #100000,R2                             | ;LOAD VIRTUAL ADDRESS INTO R2             |
| 2676 | 030624 | 010046 |        |        | 8\$: MOV R0,-(KSP)                         | ;PUSH TEST DATA ON KERNEL STACK           |
| 2677 | 030626 | 105067 | 141456 |        | CLRB KIPDR4                                | ;MAKE KERNEL PAGE 4 NON-RESIDENT          |
| 2678 | 030632 | 006612 |        |        | MTPI (R2)                                  | ;LOAD TEST DATA INTO PHYSICAL 60000       |



```

2736 031064          15$: ;THIS WILL TEST DSTM = 6 MTPI.
2737                :
2738 031064 012767 031106 150016      MOV    #16$, $LPERR      ;SET LOOP ON ERROR POINTER TO 16$
2739 031072 012767 010340 146676      MOV    #010340, PSW     ;MAKE PREVIOUS MODE SUPERVISOR
2740 031100 012700 052525                MOV    #52525, R0       ;LOAD TEST DATA INTO R0
2741 031104 005002                CLR    R2               ;MAKE REGISTER 2 ZERO
2742 031106 010046          16$:      MOV    R0, -(KSP)       ;PUSH TEST DATA ON KERNEL STACK
2743 031110 105067 141174      CLR    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
2744 031114 006662 100000      MTP    100000(R2)      ;LOAD TEST DATA INTO PHYSICAL 60000
2745 031120 112767 000006 141162      MOV    #006, KIPDR4    ;MAKE KERNEL PAGE 4 RESIDENT
2746 031126 013701 100000      MOV    @#100000, R1    ;READ FROM ADDRESS 60000
2747 031132 020001                CMP    R0, R1          ;SEE IF DATA WAS STORED CORRECTLY
2748 031134 001401                BEQ    17$             ;BRANCH IF STORE WAS CORRECT
2749 031136 104024                ERROR  +24            ;INCORRECT STORE
2750                :
2751                :FOR TIGHTER SCOPE LOOP
2752                :REPLACE ERROR CALL WITH
2753                :'BR 16$' = 000763
2753 031140          17$: ;THE FOLLOWING WILL TEST DSTM=7 MTPI.
2754                :
2755 031140 012767 031172 147742      MOV    #18$, $LPERR    ;SET LOOP ON ERROR POINTER TO 18$
2756 031146 012767 010340 146622      MOV    #010340, PSW    ;MAKE PREVIOUS MODE SUPERVISOR
2757 031154 012700 125252                MOV    #125252, R0     ;LOAD TEST DATA INTO R0
2758 031160 012767 100000 150014      MOV    #100000, $TMP2  ;LOAD VIRT. ADDR. OF TEST LOCATION
2759                :INTO LOCATION $TMP2
2760 031166 012702 001202          18$:      MOV    #$TMP2, R2      ;LOAD ADDRESS OF $TMP2 INTO R2
2761 031172 010046                MOV    R0, -(KSP)     ;PUSH TEST DATA ON KERNEL STACK
2762 031174 105067 141110      CLR    KIPDR4          ;MAKE KERNEL PAGE 4 NON-RESIDENT
2763 031200 006672 000000      MTP    @0(R2)         ;LOAD TEST DATA INTO PHYSICAL 60000
2764 031204 112767 000006 141076      MOV    #006, KIPDR4    ;MAKE KERNEL PAGE 4 RESIDENT
2765 031212 013701 100000      MOV    @#100000, R1    ;READ FROM ADDRESS 60000
2766 031216 020001                CMP    R0, R1          ;SEE IF DATA WAS STORED CORRECTLY
2767 031220 001401                BEQ    19$             ;BRANCH IF STORE WAS CORRECT
2768 031222 104024                ERROR  +24            ;INCORRECT STORE
2769                :
2770                :FOR TIGHTER SCOPE LOOP
2771                :REPLACE ERROR CALL WITH
2772                :'BR 18$' = 000763
2772 031224 012767 030454 147656      19$:      MOV    #1$, $LPERR     ;SET LOOP POINTER TO START OF TEST
2773 031232 012767 002466 147010      MOV    #MGMERR, MMVEC  ;RESTORE M.M. VECTOR TO NORMAL ROUTINE
2774 031240 000423                BR     TST24           ;BRANCH TO NEXT TEST
2775                :
2776                :
2777 031242 012667 150012          20$:      MOV    (KSP)+, TRAPPC  ;SAVE PC & PS OF TRAP
2778 031246 012667 150010      MOV    (KSP)+, TRAPPS  ;SAVE PC & PS OF TRAP
2779 031252 016767 146314 150004      MOV    SR0, WASSR0     ;SAVE SR0 FOR ERROR TYPEOUT
2780 031260 016767 146312 150002      MOV    SR2, WASSR2     ;SAVE SR2 FOR ERROR TYPEOUT
2781 031266 042767 160000 146276      BIC    #160000, SR0    ;CLEAR ERROR BITS IN SR0
2782 031274 104024                ERROR  +24            ;TRIED TO LOAD A N.R. PAGE 4
2783                :FOR TIGHTER SCOPE LOOP
2784                :REPLACE ERROR CALL WITH
2785                :A 'NOP' = 000240
2786 031276 016746 147760                MOV    TRAPPS, -(KSP)  ;PUT PC & PS OF TRAP ON STACK
2787 031302 016746 147752                MOV    TRAPPC, -(KSP)
2788 031306 000002                RTI                    ;RETURN TO TEST
  
```

2790

```

*****
*TEST 24 MOVE TO PREVIOUS (USER) I-SPACE
*
* THIS TEST USES THE 'MTPI' INSTRUCTION TO ENSURE THAT THE
* PREVIOUS MODE IS CLOKED CORRECTLY
* THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.
*
* IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
* WILL OCCUR AND TRAP TO 20$, WHERE THE ERRORS ARE REPORTED.
*
*****
  
```

```

TST24: SCOPE
1$: MOV #77406,UIPDR4 ;USER I-SPACE PAGE 4 READ/WRITE
MOV #600,UIPAR4 ;MAP USER I PAGE 4 TO 12K
MOV #20$,MMVEC ;SET M.M. VECTOR TO 20$
;THE FOLLOWING WILL TEST DSTM=0 MTPI
2$: MOV #030340,PSW ;MAKE PREVIOUS MODE USER
MOV #7777,-(KSP) ;PUSH DATA ON KERNEL STACK
MTPI USP ;LOAD USER STACK POINTER
MFPI USP ;READ USER STACK POINTER
MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
CMP #7777,R1 ;WAS USER STACK POINTER CHANGED
BEQ 3$ ;BRANCH IF IT WAS
ERROR +25 ;USER STACK POINTER NOT CHANGED
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
; 'BR 2$' = 000764
3$: MOV #030340,PSW ;MAKE PREVIOUS MODE USER
MOV #USESTK,-(KSP) ;GET READY TO RESTORE USER S. POINT
MTPI USP ;RESTORE USER STACK POINTER
4$: ;THIS WILL TEST DSTM = 1 MTPI.
MOV #5$, $LPERR ;SET LOOP ON ERROR POINTER TO 5$
MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
MOV #125252,R0 ;LOAD TEST DATA INTO R0
5$: MOV R0,-(KSP) ;PUSH TEST DATA ON KERNEL STACK
CLRB KIPDR4 ;MAKE KERNEL I PAGE 4 NON-RESIDENT
MTPI (R2) ;LOAD TEST DATA INTO PHYSICAL 60000
MOVB #006,KIPDR4 ;MAKE KERNEL PAGE 4 RESIDENT
MOV (R2),R1 ;READ FROM ADDRESS 60000
CMP R0,R1 ;SEE IF DATA WAS STORED AT CORRECT PLACE
BEQ 6$ ;BRANCH IF STORE WAS CORRECT
ERROR +24 ;INCORRECT STORE
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
; 'BR 5$' = 000765
6$: ;THE FOLLOWING WILL TEST DSTM=2 MTPI.
MOV #8$, $LPERR ;SET LOOP ON ERROR POINTER TO 8$
MOV #030340,PSW ;MAKE PREVIOUS MODE USER
MOV #125252,R0 ;LOAD TEST DATA INTO R0
MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
8$: MOV R0,-(KSP) ;PUSH TEST DATA ON KERNEL STACK
CLRB KIPDR4 ;MAKE KERNEL PAGE 4 NON-RESIDENT
MTPI (R2) ;LOAD TEST DATA INTO PHYSICAL 60000
MOVB #006,KIPDR4 ;MAKE KERNEL PAGE 4 RESIDENT
  
```

```

031310 000004
2791 031312 012767 077406 146270
2792 031320 012767 000600 146322
2793 031326 012767 032106 146714
2794
2795
2796 031334 012767 030340 146434
2797 031342 012746 007777
2798 031346 006606
2799 031350 006506
2800 031352 012601
2801 031354 022701 007777
2802 031360 001401
2803 031362 104025
2804
2805
2806
2807 031364 012767 030340 146404
2808 031372 012746 000600
2809 031376 006606
2810 031400
2811 031400 012767 031416 147502
2812 031406 012702 100000
2813 031412 012700 125252
2814 031416 010046
2815 031420 105067 140664
2816 031424 006612
2817 031426 112767 000006 140654
2818 031434 011201
2819 031436 020001
2820 031440 001401
2821 031442 104024
2822
2823
2824
2825 031444
2826
2827 031444 012767 031470 147436
2828 031452 012767 030340 146316
2829 031460 012700 125252
2830 031464 012702 100000
2831 031470 010046
2832 031472 105067 140612
2833 031476 006612
2834 031500 112767 000006 140602
  
```

```

2835 031506 013701 100000      MOV      @#100000,R1      ;READ FROM ADDRESS 60000
2836 031512 020001      CMP      R0,R1          ;SEE IF DATA WAS STORED CORRECTLY
2837 031514 001401      BEQ     9$             ;BRANCH IF STORE WAS CORRECT
2838 031516 104024      ERROR   +24           ;INCORRECT STORE
2839                                ;FOR TIGHTER SCOPE LOOP
2840                                ;REPLACE ERROR CALL WITH
2841                                ;'BR 8$' = 000764
2842 031520      9$:      ;THIS WILL TEST DSTM = 3 MTPI.
2843 031520 012767 031540 147362      MOV      #10$, $LPERR    ;SET LOOP ON ERROR POINTER TO 10$
2844 031526 012767 030340 146242      MOV      #030340,PSW    ;MAKE PREVIOUS MODE USER
2845 031534 012700 052525      MOV      #52525,R0      ;LOAD TEST DATA INTO R0
2846 031540 010046      10$:     MOV      R0,-(KSP)     ;PUSH TEST DATA ON KERNEL STACK
2847 031542 105067 140542      CLRB    KIPDR4         ;MAKE KERNEL I PAGE 4 NON-RESIDENT
2848 031546 006637 100000      MTPI    @#100000       ;LOAD TEST DATA INTO PHYSICAL 60000
2849 031552 112767 000006 140530      MOV     #006,KIPDR4    ;MAKE KERNEL PAGE 4 RESIDENT
2850 031560 013701 100000      MOV      @#100000,R1    ;READ FROM ADDRESS 60000
2851 031564 020001      CMP      R0,R1          ;SEE IF DATA WAS STORED CORRECTLY
2852 031566 001401      BEQ     11$           ;BRANCH IF STORE WAS CORRECT
2853 031570 104024      ERROR   +24           ;INCORRECT STORE
2854                                ;FOR TIGHTER SCOPE LOOP
2855                                ;REPLACE ERROR CALL WITH
2856                                ;'BR 10$' = 000763
2857 031572      11$:     ;THIS WILL TEST DSTM = 4 MTPI.
2858 031572 012767 031612 147310      MOV      #12$, $LPERR    ;SET LOOP ON ERROR POINTER TO 12$
2859 031600 012767 030340 146170      MOV      #030340,PSW    ;MAKE PREVIOUS MODE USER
2860 031606 012700 125252      MOV      #125252,R0     ;LOAD TEST DATA INTO R0
2861 031612 010046      12$:     MOV      R0,-(KSP)     ;PUSH TEST DATA ON KERNEL STACK
2862 031614 012702 100002      MOV      #100002,R2    ;LOAD VIRTUAL ADDRESS INTO R2
2863 031620 105067 140464      CLRB    KIPDR4         ;MAKE KERNEL I PAGE 4 NON-RESIDENT
2864 031624 006642      MTPI    -(R2)          ;LOAD TEST DATA INTO PHYSICAL 60000
2865 031626 112767 000006 140454      MOV     #006,KIPDR4    ;MAKE KERNEL PAGE 4 RESIDENT
2866 031634 013701 100000      MOV      @#100000,R1    ;READ FROM ADDRESS 60000
2867 031640 020001      CMP      R0,R1          ;SEE IF DATA WAS STORED CORRECTLY
2868 031642 001401      BEQ     13$           ;BRANCH IF STORE WAS CORRECT
2869 031644 104024      ERROR   +24           ;INCORRECT STORE
2870                                ;FOR TIGHTER SCOPE LOOP
2871                                ;REPLACE ERROR CALL WITH
2872                                ;'BR 12$' = 000762
2873 031646      13$:     ;THE FOLLOWING WILL TEST DSTM=5 MTPI.
2874                                ;
2875 031646 012767 031700 147234      MOV      #14$, $LPERR    ;SET LOOP ON ERROR POINTER TO 14$
2876 031654 012767 030340 146114      MOV      #030340,PSW    ;MAKE PREVIOUS MODE USER
2877 031662 012700 052525      MOV      #52525,R0      ;LOAD TEST DATA INTO R0
2878 031666 012702 001204      MOV      #<$TMP2+2>,R2  ;LOAD ADDR. OF LOC. $TMP2+2 INTO R2
2879 031672 012767 100000 147302      MOV      #100000,$TMP2  ;LOAD VIRT. ADDR. OF TEST LOC. INTO $TMP2
2880 031700 010046      14$:     MOV      R0,-(KSP)     ;PUSH TEST DATA ON KERNEL STACK
2881 031702 105067 140402      CLRB    KIPDR4         ;MAKE KERNEL PAGE 4 NON-RESIDENT
2882 031706 006652      MTPI    @-(R2)         ;LOAD TEST DATA INTO PHYSICAL 60000
2883 031710 112767 000006 140372      MOV     #006,KIPDR4    ;MAKE KERNEL PAGE 4 RESIDENT
2884 031716 013701 100000      MOV      @#100000,R1    ;READ FROM ADDRESS 60000
2885 031722 020001      CMP      R0,R1          ;SEE IF DATA WAS STORED CORRECTLY
2886 031724 001401      BEQ     15$           ;BRANCH IF STORE WAS CORRECT
2887 031726 104024      ERROR   +24           ;INCORRECT STORE
2888                                ;FOR TIGHTER SCOPE LOOP
2889                                ;REPLACE ERROR CALL WITH
2890                                ;'BR 14$' = 000764
2891 031730      15$:     ;THIS WILL TEST DSTM = 6 MTPI.

```

```

2892
2893 031730 012767 031752 147152      : MOV #16$, $LPERR      ;SET LOOP ON ERROR POINTER TO 16$
2894 031736 012767 030340 146032      : MOV #030340, PSW     ;MAKE PREVIOUS MODE USER
2895 031744 012700 052525              : MOV #52525, R0       ;LOAD TEST DATA INTO R0
2896 031750 005002                      : CLR R2               ;MAKE REGISTER 2 ZERO
2897 031752 010046 16$: MOV R0, -(KSP) ;PUSH TEST DATA ON KERNEL STACK
2898 031754 105067 140330              : CLRB KIPDR4         ;MAKE KERNEL I PAGE 4 NON-RESIDENT
2899 031760 006662 100000              : MTPJ 100000(R2)     ;LOAD TEST DATA INTO PHYSICAL 60000
2900 031764 112767 000006 140316      : MOV #006, KIPDR4    ;MAKE KERNEL PAGE 4 RESIDENT
2901 031772 013701 100000              : MOV @#100000, R1    ;READ FROM ADDRESS 60000
2902 031776 020001                      : CMP R0, R1          ;SEE IF DATA WAS STORED CORRECTLY
2903 032000 001401                      : BEQ 17$             ;BRANCH IF STORE WAS CORRECT
2904 032002 104024                      : ERROR +24          ;INCORRECT STORE
2905
2906
2907
2908 032004 17$: ;THE FOLLOWING WILL TEST DSTM=7 MTPJ.
2909
2910 032004 012767 032036 147076      : MOV #18$, $LPERR     ;SET LOOP ON ERROR POINTER TO 18$
2911 032012 012767 030340 145756      : MOV #030340, PSW     ;MAKE PREVIOUS MODE USER
2912 032020 012700 125252              : MOV #125252, R0      ;LOAD TEST DATA INTO R0
2913 032024 012767 100000 147150      : MOV #100000, $TMP2   ;LOAD VIRT. ADDR. OF TEST LOCATION
2914
2915 032032 012702 001202              : MOV $TMP2, R2        ;LOAD ADDRESS OF $TMP2 INTO R2
2916 032036 010046 18$: MOV R0, -(KSP) ;PUSH TEST DATA ON KERNEL STACK
2917 032040 105067 140244              : CLRB KIPDR4         ;MAKE KERNEL PAGE 4 NON-RESIDENT
2918 032044 006672 000000              : MTPJ @0(R2)         ;LOAD TEST DATA INTO PHYSICAL 60000
2919 032050 112767 000006 140232      : MOV #006, KIPDR4    ;MAKE KERNEL PAGE 4 RESIDENT
2920 032056 013701 100000              : MOV @#100000, R1    ;READ FROM ADDRESS 60000
2921 032062 020001                      : CMP R0, R1          ;SEE IF DATA WAS STORED CORRECTLY
2922 032064 001401                      : BEQ 19$             ;BRANCH IF STORE WAS CORRECT
2923 032066 104024                      : ERROR +24          ;INCORRECT STORE
2924
2925
2926
2927 032070 012767 031312 147012 19$: MOV #1$, $LPERR ;SET LOOP POINTER TO START OF TEST
2928 032076 012767 002466 146144      : MOV #MGMERR, MMVEC  ;RESTORE M.M. VECTOR TO NORMAL ROUTINE
2929 032104 000423                      : BR TST25            ;BRANCH TO NEXT TEST
2930
2931
2932 032106 012667 147146 20$: MOV (KSP)+, TRAPPC ;SAVE PC & PS OF TRAP
2933 032112 012667 147144              : MOV (KSP)+, TRAPPS
2934 032116 016767 145450 147140      : MOV SR0, WASSRO     ;SAVE SR0 FOR ERROR TYPEOUT
2935 032124 016767 145446 147136      : MOV SR2, WASSR2     ;SAVE SR2 FOR ERROR TYPEOUT
2936 032132 042767 160000 145432      : BIC #160000, SR0    ;CLEAR ERROR BITS IN SR0
2937 032140 104024                      : ERROR +24          ;TRIED TO LOAD A N.R. PAGE 4
2938
2939
2940
2941 032142 016746 147114              : MOV TRAPPS, -(KSP)  ;PUT PC & PS OF TRAP ON STACK
2942 032146 016746 147106              : MOV TRAPPC, -(KSP)
2943 032152 000002                      : RTI                ;RETURN TO TEST
2944
  
```

2957

```

*****
*TEST 25 MFPI (KERNAL) TO SUPERVISOR MODE
*
* THIS TEST CHECKS THAT IF THE PREVIOUS MODE IS KERNEL THE
* FETCH IS FROM KERNEL SPACE.
* THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.
*
* IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
* WILL OCCUR AND TRAP TO 21$, WHERE THE ERRORS ARE REPORTED.
*****
    
```

```

TST25: SCOPE
2958 032154 000004 032164 146724 1$: MOV #1$, $LPERR ;SET LOOP ON ERROR TO 1$
2959 032156 012767 040340 145604 1$: MOV #040340,PSW ;GO TO SUPERVISOR MODE FOR THIS TEST
2960 032172 012700 036514 1$: MOV #36514,R0 ;LOAD DATA PATTERN INTO R0
2961 032176 010037 100000 1$: MOV R0,#100000 ;LOAD DATA PATTERN INTO PHY 60000
2962 032202 012702 100000 1$: MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
2963 ;THE FOLLOWING WILL TEST DSTM=0 MFPI
2964
2965 032206 012767 032616 146034 4$: MOV #21$,MMVEC ;SET M.M. VECTOR TO 21$
2966 032214 105067 137770 4$: CLRB SIPDR4 ;MAKE SUPERVISOR I-SPACE PAGE 4 NON-RESIDENT
2967 032220 012767 040340 145550 4$: MOV #040340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT SUPERVISOR
2968 032226 006506 4$: MFPI KSP ;PUT KERNEL STACK POINTER ON SUPERVISOR STACK
2969 032230 022706 000700 4$: CMP #SUPSTK,SSP ;WAS SOMETHING PUSHED ON STACK AT 1$
2970 032234 001407 4$: BEQ 5$ ;BRANCH IF NOTHING WAS PUSHED
2971 032236 012600 4$: MOV (SSP)+,R0 ;POP SUPERVISOR STACK INTO R0
2972 032240 012701 001100 4$: MOV #KERSTK,R1 ;EXPECTING 1100 AS KSP
2973 032244 020001 4$: CMP R0,R1 ;DID YOU GET THE RIGHT POINTER?
2974 032246 001403 4$: BEQ 6$ ;BRANCH IF YOU DID
2975 032250 104023 4$: ERROR +23 ;WRONG THING WAS PUSHED ON STACK
2976 ;FOR TIGHTER SCOPE LOOP
2977 ;REPLACE ERROR CALL WITH
2978 ;'BR 4$' = 000766
2979 032252 000401 5$: BR 6$ ;BRANCH TO NEXT TRY
2980 032254 104025 5$: ERROR +25 ;NOTHING PUSHED ON STACK
2981 ;FOR TIGHTER SCOPE LOOP
2982 ;REPLACF ERROR CALL WITH
2983 ;'BR 4$' = 000764
2984 032256 6$: ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
2985 032256 012767 032264 146624 7$: MOV #7$, $LPERR ;SET LOOP ON ERROR POINTER TO 7$
2986 032264 012767 040340 145504 7$: MOV #040340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT SUPERVISOR
2987 032272 012700 036514 7$: MOV #36514,R0 ;LOAD DATA EXPECTED INTO R0
2988 032276 012702 100000 7$: MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
2989 032302 006512 7$: MFPI (R2) ;READ FROM PHYSICAL 60000
2990 032304 012601 7$: MOV (SSP)+,R1 ;POP SUPERVISOR STACK INTO R1
2991 032306 020001 7$: CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
2992 032310 001401 7$: BEQ 8$ ;BRANCH IF CORRECT DATA WAS FETCHED
2993 032312 104023 7$: ERROR +23 ;WRONG DATA WAS FETCHED
2994 ;FOR TIGHTER SCOPE LOOP
2995 ;REPLACE ERROR CALL WITH
2996 ;'BR 7$' = 000764
2997 032314 8$: ;THE FOLLOWING WILL TEST DSM=2 MFPI.
2998 032314 012767 032322 146566 9$: MOV #9$, $LPERR ;SET LOOP ON ERROR POINTER TO 9$
2999 032322 012767 040340 145446 9$: MOV #040340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT SUPERVISOR
3000 032330 012702 100000 9$: MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
3001 032334 006522 9$: MFPI (R2)+ ;READ FROM PHYSICAL 60000
    
```







```

3059 032522      18$:      ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
3060
3061 032522      012767 032530 146360      .
3062 032530      012767 040340 145240      19$:      MOV      #19$, $LPERR      ;SET LOOP ON ERROR POINTER TO 19$
3063 032536      012767 100000 146436      MOV      #040340, PSW      ;MAKE PREVIOUS MODE KERNEL PRESENT SUPERVISOR
3064 032544      012702 001202      MOV      #100000, $TMP2    ;LOAD TEST LOC. VIRT. ADDR. INTO $TMP2
3065 032550      006572 000000      MOV      #$TMP2, R2        ;LOAD ADDRESS OF $TMP2 INTO R2
3066 032554      012601      MFPI      @0(R2)           ;READ FROM PHYSICAL 60000
3067 032556      020001      MOV      (SSP)+, R1        ;POP SUPERVISOR STACK INTO R1
3068 032560      001401      CMP      R0, R1           ;WAS DATA FETCHED SAME AS STORED
3069 032562      104023      BEQ      20$              ;BRANCH IF CORRECT DATA FETCHED
3070      ERROR      +23        ;WRONG DATA WAS FETCHED
3071      ;FOR TIGHTER SCOPE LOOP
3072      ;REPLACE ERROR CALL WITH
3073      ;'BR 19$' = 000762
3074 032564      012767 002466 145456      20$:      MOV      #MMGMERR, MMVEC    ;SET M.M. VECTOR TO NORMAL ROUTINE
3075 032572      012767 000340 145176      MOV      #00340, PSW       ;GO BACK TO KERNEL MODE, PREVIOUS KERNEL
3076 032600      012767 032164 146302      MOV      #1$, $LPERR       ;SET LOOP POINTER TO START OF TEST
3077 032606      112767 000006 137374      MOV      #6, $IPDR4        ;MAKE $IPDR4 RESIDENT
3078      BR      TST26        ;BRANCH TO NEXT TEXT
3079
3080 032616      012667 146436      21$:      MOV      (KSP)+, TRAPPC    ;SAVE PC & PS OF TRAP
3081 032622      012667 146434      MOV      (KSP)+, TRAPPS    ;SAVE PC & PS OF TRAP
3082 032626      016767 144740 146430      MOV      SRO, WASSRO       ;SAVE SRO FOR ERROR TYPEOUT
3083 032634      016767 144736 146426      MOV      SR2, WASSR2       ;SAVE SR2 FOR ERROR TYPEOUT
3084 032642      042767 160000 144722      BIC      #160000, SRO      ;CLEAR ERROR BITS IN SRO
3085 032650      104026      ERROR      +26            ;TRIED TO READ NON-RESIDENT PAGE
3086      ;FOR TIGHTER SCOPE LOOP
3087      ;REPLACE ERROR CALL WITH
3088      ;A 'NOP' = 000240
3089 032652      016746 146404      MOV      TRAPPS, -(KSP)    ;PUT PC & PS OF TRAP ON STACK
3090 032656      016746 146376      MOV      TRAPPC, -(KSP)
3091 032662      000002      RTI                          ;RETURN TO TEST
3092

```

```

*****
*TEST 26      MFPI (KERNAL) TO USER MODE
*
*      THIS TEST CHECKS THAT IF THE PREVIOUS MODE IS KERNEL THE
*      FETCH IS FROM KERNEL SPACE.
*      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.
*
*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
*      WILL OCCUR AND TRAP TO 21$, WHERE THE ERRORS ARE REPORTED.
*****

```

```

3093 032664      000004      TST26:  SCOPE
3094 032666      012767 032674 146214      MOV      #1$, $LPERR      ;SET LOOP ON ERROR TO 1$
3095 032702      012700 036514 145074      1$:      MOV      #140340, PSW      ;GO TO USER MODE FOR THIS TEST
3096 032706      010037 100000      MOV      #36514, R0        ;LOAD DATA PATTERN INTO R0
3097 032712      012702 100000      MOV      R0, @#100000     ;LOAD DATA PATTERN INTO PHY 60000
3098      MOV      #100000, R2    ;LOAD VIRTUAL ADDRESS INTO R2
3099      ;THE FOLLOWING WILL TEST DSTM=0 MFPI
3100 032716      012767 033326 145324      MOV      #21$, MMVEC      ;SET M.M. VECTOR TO 21$
3101 032724      105067 144660      CLR      UIPDR4           ;MAKE USER I-SPACE PAGE 4 NON-RESIDENT
3102 032730      012767 140340 145040      MOV      #140340, PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3103 032736      006506      4$:      MFPI      KSP              ;PUT KERNEL STACK POINTER ON USER STACK

```

```

3104 032740 022706 000600      CMP      #USESTK,USP      ;WAS SOMETHING PUSHED ON STACK AT 1$
3105 032744 001407      BEQ      5$              ;BRANCH IF NOTHING WAS PUSHED
3106 032746 012600      MOV      (USP)+,R0      ;POP USER STACK INTO R0
3107 032750 012701 001100      MOV      #KERSTK,R1     ;EXPECTING 1100 AS KSP
3108 032754 020001      CMP      R0,R1          ;DID YOU GET THE RIGHT POINTER?
3109 032756 001403      BEQ      6$              ;BRANCH IF YOU DID
3110 032760 104023      ERROR   +23            ;WRONG THING WAS PUSHED ON STACK
3111                                ;FOR TIGHTER SCOPE LOOP
3112                                ;REPLACE ERROR CALL WITH
3113                                ;'BR 4$' = 000766
3114 032762 000401      BR      6$              ;BRANCH TO NEXT TRY
3115 032764 104025      5$:      ERROR   +25            ;NOTHING PUSHED ON STACK
3116                                ;FOR TIGHTER SCOPE LOOP
3117                                ;REPLACE ERROR CALL WITH
3118                                ;'BR 4$' = 000764
3119 032766      6$:      ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
3120 032766 012767 032774 146114      MOV      #7$, $LPERR     ;SET LOOP ON ERROR POINTER TO 7$
3121 032774 012767 140340 144774      7$:      MOV      #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3122 033002 012700 036514      MOV      #36514,R0      ;LOAD DATA EXPECTED INTO R0
3123 033006 012702 100000      MOV      #100000,R2     ;LOAD VIRTUAL ADDRESS INTO R2
3124 033012 006512      MFPI    (R2)            ;READ FROM PHYSICAL 60000
3125 033014 012601      MOV      (USP)+,R1     ;POP USER STACK INTO R1
3126 033016 020001      CMP      R0,R1          ;WAS DATA FETCHED SAME AS STORED
3127 033020 001401      BEQ      8$              ;BRANCH IF CORRECT DATA WAS FETCHED
3128 033022 104023      ERROR   +23            ;WRONG DATA WAS FETCHED
3129                                ;FOR TIGHTER SCOPE LOOP
3130                                ;REPLACE ERROR CALL WITH
3131                                ;'BR 7$' = 000764
3132 033024      8$:      ;THE FOLLOWING WILL TEST DSM=2 MFPI.
3133 033024 012767 033032 146056      MOV      #9$, $LPERR     ;SET LOOP ON ERROR POINTER TO 9$
3134 033032 012767 140340 144736      9$:      MOV      #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3135 033040 012702 100000      MOV      #100000,R2     ;LOAD VIRTUAL ADDRESS INTO R2
3136 033044 006522      MFPI    (R2)+           ;READ FROM PHYSICAL 60000
3137 033046 012601      MOV      (USP)+,R1     ;POP USER STACK INTO R1
3138 033050 020001      CMP      R0,R1          ;WAS DATA FETCHED SAME AS STORED
3139 033052 001401      BEQ      10$             ;BRANCH IF CORRECT DATA WAS FETCHED
3140 033054 104023      ERROR   +23            ;WRONG DATA WAS FETCHED
3141                                ;FOR TIGHTER SCOPE LOOP
3142                                ;REPLACE ERROR CALL WITH
3143                                ;'BR 9$' = 000766
3144 033056      10$:     ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
3145 033056 012767 033064 146024      MOV      #11$, $LPERR    ;SET LOOP ON ERROR POINTER TO 11$
3146 033064 012767 140340 144704      11$:     MOV      #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3147 033072 006537 100000      MFPI    @#100000        ;READ FROM PHYSICAL 60000
3148 033076 012601      MOV      (USP)+,R1     ;POP USER STACK INTO R1
3149 033100 020001      CMP      R0,R1          ;WAS DATA FETCHED SAME AS STORED
3150 033102 001401      BEQ      12$             ;BRANCH IF CORRECT DATA WAS FETCHED
3151 033104 104023      ERROR   +23            ;WRONG DATA WAS FETCHED
3152                                ;FOR TIGHTER SCOPE LOOP
3153                                ;REPLACE ERROR CALL WITH
3154                                ;'BR 11$' = 000767
3155 033106      12$:     ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
3156 033106 012767 033114 145774      MOV      #13$, $LPERR    ;SET LOOP ON ERROR POINTER TO 13$
3157 033114 012767 140340 144654      13$:     MOV      #140340,PSW     ;MAKE PREVIOUS MODE DERNEL PRESENT USER
3158 033122 012702 100002      MOV      #100002,R2     ;LOAD VIRTUAL ADDRESS INTO R2
3159 033126 006542      MFPI    -(R2)           ;READ FROM PHYSICAL 60000
3160 033130 012601      MOV      (USP)+,R1     ;POP USER STACK INTO R1
  
```

```

3161 033132 020001      CMP      R0,R1      ;WAS DATA FETCHED SAME AS STORED
3162 033134 001401      BEQ      14$        ;BRANCH IF CORRECT DATA WAS FETCHED
3163 033136 104023      ERROR   +23        ;WRONG DATA WAS FETCHED
3164                                     ;FOR TIGHTER SCOPE LOOP
3165                                     ;REPLACE ERROR CALL WITH
3166                                     ;'BR 13$' = 000766
3167 033140          14$:  ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
3168
3169 033140 012767 033146 145742      MOV      #15$, $LPERR ;SET LOOP ON ERROR POINTER TO 15$
3170 033146 012767 140340 144622      15$:  MOV      #140340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3171 033154 012767 100000 146020      MOV      #100000,$TMP2 ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
3172 033162 012702 001204      MOV      #<$TMP2+2>,R2 ;LOAD ADDRESS OF $TMP2+2 INTO R2
3173 033166 006552      MFPI    @-(R2)      ;READ FROM PHYSICAL 60000
3174 033170 012601      MOV      (USP)+,R1   ;POP USER STACK INTO R1
3175 033172 020001      CMP      R0,R1      ;WAS DATA FETCHED SAME AS STORED
3176 033174 001401      BEQ      16$        ;BRANCH IF CORRECT DATA FETCHED
3177 033176 104023      ERROR   +23        ;WRONG DATA WAS FETCHED
3178                                     ;FOR TIGHTER SCOPE LOOP
3179                                     ;REPLACE ERROR CALL WITH
3180                                     ;'BR 15$' = 000763
3181 033200          16$:  ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
3182
3183 033200 012767 033206 145702      MOV      #17$, $LPERR ;SET LOOP ON ERROR POINTER TO 17$.
3184 033206 012767 140340 144562      17$:  MOV      #140340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3185 033214 005002      CLR      R2         ;MAKE REGISTER 2 A ZERO
3186 033216 006562 100000      MFPI    100000(R2) ;READ FROM PHYSICAL 60000
3187 033222 012601      MOV      (USP)+,R1   ;POP USER STACK INTO R1
3188 033224 020001      CMP      R0,R1      ;WAS DATA FETCHED SAME AS STORED
3189 033226 001401      BEQ      18$        ;BRANCH IF CORRECT DATA FETCHED
3190 033230 104023      ERROR   +23        ;WRONG DATA WAS FETCHED
3191                                     ;FOR TIGHTER SCOPE LOOP
3192                                     ;REPLACE ERROR CALL WITH
3193                                     ;'BR 17$' = 000766
3194 033232          18$:  ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
3195
3196 033232 012767 033240 145650      MOV      #19$, $LPERR ;SET LOOP ON ERROR POINTER TO 19$
3197 033240 012767 140340 144530      19$:  MOV      #140340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3198 033246 012767 100000 145726      MOV      #100000,$TMP2 ;LOAD TEST LOC. VIRT. ADDR. INTO $TMP2
3199 033254 012702 001202      MOV      # $TMP2,R2  ;LOAD ADDRESS OF $TMP2 INTO R2
3200 033260 006572 000000      MFPI    @0(R2)      ;READ FROM PHYSICAL 60000
3201 033264 012601      MOV      (USP)+,R1   ;POP USER STACK INTO R1
3202 033266 020001      CMP      R0,R1      ;WAS DATA FETCHED SAME AS STORED
3203 033270 001401      BEQ      20$        ;BRANCH IF CORRECT DATA FETCHED
3204 033272 104023      ERROR   +23        ;WRONG DATA WAS FETCHED
3205                                     ;FOR TIGHTER SCOPE LOOP
3206                                     ;REPLACE ERROR CALL WITH
3207                                     ;'BR 19$' = 000762
3208 033274 012767 002466 144746      20$:  MOV      #MGMERR,MMVEC ;SET M.M. VECTOR TO NORMAL ROUTINE
3209 033302 012767 000340 144466      MOV      #00340,PSW ;GO BACK TO KERNEL MODE, PREVIOUS KERNEL
3210 033310 012767 032674 145572      MOV      #1$, $LPERR ;SET LOOP POINTER TO START OF TEST
3211 033316 112767 000006 144264      MOVB    #6,UIPDR4   ;MAKE UIPDR4 RESIDENT
3212 033324 000423      BR      TST27      ;BRANCH TO NEXT TEXT
3213
3214
3215 033326 012667 145726          21$:  MOV      (KSP)+,TRAPPC ;SAVE PC & PS OF TRAP
3216 033332 012667 145724      MOV      (KSP)+,TRAPPS
3217 033336 016767 144230 145720      MOV      SRO,WASSRO ;SAVE SRO FOR ERROR TYPEOUT
  
```

```

3218 033344 016767 144226 145716      MOV      SR2,WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
3219 033352 042767 160000 144212      BIC      #160000,SRO     ;CLEAR ERROR BITS IN SRO
3220 033360 104026                ERROR  +26              ;TRIED TO READ NON-RESIDENT PAGE
3221                                ;FOR TIGHTER SCOPE LOOP
3222                                ;REPLACE ERROR CALL WITH
3223                                ;A 'NOP' = 000240
3224 033362 016746 145674      MOV      TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
3225 033366 016746 145666      MOV      TRAPPC,-(KSP)
3226 033372 000002                RTI                    ;RETURN TO TEST
3227
3228
3238

```

```

*****
*TEST 27      MFPI (SUPERVISOR) WITH SUPER D-SPACE ENABLED
*
*      THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT PREVIOUS
*      MODE IS CLOKED CORRECTLY, AND THAT D-SPACE IS NOT ENABLED.
*
*      IF THE CORRECT MODE IS NOT ENABLD, A NON-RESIDENT ABORT WILL
*      OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
*
*****

```

```

033374 000004
3239 033376 012767 033412 145502      MOV      #20$, $LPADR   ;SET LOOP POINTER TO 20$
3240 033404 012767 033412 145476      MOV      #20$, $LPERR   ;SET LOOP ON ERROR POINTER TO 20$
3241 033412 012700 077400      20$:    MOV      #77400,R0     ;MAKE PAGE 4 IN ALL BUT SUPERVISOR I
3242                                ;AND KERNAL I NON-RESIDENT
3243 033416 010067 136706      MOV      R0,KDPDR4      ;KERNAL D-SPACE PAGE 4
3244 033422 010067 136602      MOV      R0,SDPDR4      ;SUPERVISOR D-SPACE PAGE 4
3245 033426 010067 144176      MOV      R0,UDPDR4      ;USER D-SPACE PAGE 4
3246 033432 010067 144152      MOV      R0,UIPDR4      ;USER I-SPACE PAGE 4
3247 033436 012700 036514      MOV      #36514,R0      ;LOAD DATA PATTERN INTO R0
3248 033442 010037 060000      MOV      R0,#60000      ;LOAD DATA PATTERN INTO PHYS. 60000
3249 033446 012767 033672 144574      MOV      #10$,MMVEC     ;SET M.M. VECTOR TO 10$
3250 033454 052767 000002 137034      BIS      #BIT1,MMR3     ;ENABLE SUPERVISOR D-SPACE
3251 033462 105067 136622      CLRB    KIPDR4         ;MAKE KERNAL I-SPACE PAGE 4 NON-RESIDENT
3252                                ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
3253
3254 033466 012767 033474 145414      MOV      #12$, $LPERR   ;SET LOOP ON ERROR POINTER TO 12$
3255 033474 012767 010340 144274      12$:    MOV      #010340,PSW   ;MAKE PREVIOUS MODE SUPERVISOR
3256 033502 012702 100000      MOV      #100000,R2     ;LOAD VIRTUAL ADDRESS INTO R2
3257 033506 006512                MFPI    (R2)           ;READ FROM PHYSICAL 60000
3258 033510 012601                MOV      (KSP)+,R1     ;POP KERNAL STACK INTO R1
3259 033512 020001                CMP      R0,R1         ;WAS DATA FETCHED SAME AS DATA STORED
3260 033514 001401                BEQ      4$            ;BRANCH IF CORRECT DATA FETCHED
3261 033516 104037                ERROR   +37           ;WRONG DATA FETCHED
3262                                ;THE FOLLOWING WILL TEST DSTM=2 MFPI.
3263
3264 033520 012767 033526 145362      4$:    MOV      #14$, $LPERR   ;SET LOOP ON ERROR POINTER TO 14$
3265 033526 012767 010340 144242      14$:    MOV      #010340,PSW   ;MAKE PREVIOUS MODE SUPERVISOR
3266 033534 012702 100000      MOV      #100000,R2     ;LOAD VIRTUAL ADDRESS INTO R2
3267 033540 006522                MFPI    (R2)+         ;READ FROM PHYSICAL 100000
3268 033542 012601                MOV      (KSP)+,R1     ;POP KERNAL STACK INTO R1
3269 033544 020001                CMP      R0,R1         ;WAS DATA FETCHED SAME AS DATA STORED
3270 033546 001401                BEQ      5$            ;BRANCH IF CORRECTDATA FETCHED
3271 033550 104037                ERROR   +37           ;WRONG DATA FETCHED
3272                                ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
3273

```

|      |        |        |        |        |       |      |              |                                    |
|------|--------|--------|--------|--------|-------|------|--------------|------------------------------------|
| 3274 | 033552 | 012767 | 033560 | 145330 | 5\$:  | MOV  | #15\$,SLPERR | :SET LOOP ON ERROR POINTER TO 15\$ |
| 3275 | 033560 | 012767 | 010340 | 144210 | 15\$: | MOV  | #010340,PSW  | :MAKE PREVIOUS MODE SUPERVISOR     |
| 3276 | 033566 | 012702 | 100000 |        |       | MOV  | #100000,R2   | :LOAD VIRTUAL ADDRESS INTO R2      |
| 3277 | 033572 | 006537 | 100000 |        |       | MFPI | @100000      | :READ FROM PHYSICAL 100000         |

```

3279 033576 012601      MOV      (KSP)+,R1      ;POP KERNAL STACK INTO R1
3280 033600 020001      CMP      R0,R1         ;WAS DATA FETCHED SAME AS DATA STORED
3281 033602 001401      BEQ      6$           ;BRANCH IF CORRECTDATA FETCHED
3282 033604 104037      ERROR   +37          ;WRONG DATA FETCHED
3283                      ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
3284                      ;
3285 033606 012767 033614 145274 6$:      MOV      #16$, $LPERR   ;SET LOOP ON ERROR POINTER TO 16$
3286 033614 012767 010340 144154 16$:     MOV      #010340,PSW    ;MAKE PREVIOUS MODE SUPERVISOR
3287 033622 012702 100002      MOV      #100002,R2    ;LOAD VIRTUAL ADDRESS INTO R2
3288 033626 006542      MFPI    -(R2)         ;READ FROM PHYSICAL 100000
3289 033630 012601      MOV      (KSP)+,R1    ;POP KERNAL STACK INTO R1
3290 033632 020001      CMP      R0,R1         ;WAS DATA FETCHED SAME AS DATA STORED
3291 033634 001401      BEQ      7$           ;BRANCH IF CORRECTDATA FETCHED
3292 033636 104037      ERROR   +37          ;WRONG DATA FETCHED
3293 033640 012767 002466 144402 7$:      MOV      #MMGMERR,MMVEC ;SET M.M. VECTOR TO NORMAL ROUTINE
3294 033646 012767 033412 145234      MOV      #20$, $LPERR   ;SET LOOP ON ERROR POINTER TO START OF TEST
3295 033654 112767 000006 136426      MOV      #6, KIPDR4     ;MAKE KIPDR4 RESIDENT
3296 033662 042767 000002 136626      BIC      #BIT1,MMR3     ;DISABLE SUPERVISOR D-SPACE
3297 033670 000426      BR      TST30        ;BRANCH TO NEXT TEST
3298                      ;
3299 033672 016767 143674 145364 10$:     MOV      MMR0,WASSR0   ;SAVE MMR0 FOR ERROR TYPEOUT
3300 033700 016767 143670 145360      MOV      MMR1,WASSR1   ;SAVE MMR1 FOR ERROR TYPEOUT
3301 033706 016767 143664 145354      MOV      MMR2,WASSR2   ;SAVE MMR2 FOR ERROR TYPEOUT
3302 033714 016767 136576 145350      MOV      MMR3,WASSR3   ;SAVE MMR3 FOR ERROR TYPEOUT
3303 033722 012667 145332      MOV      (KSP)+,TRAPPC ;SAVE PC & PS OF TRAP
3304 033726 012667 145330      MOV      (KSP)+,TRAPPS
3305 033732 104040      ERROR   +40          ;TRIED TO READ NON-RESIDENT PAGE
3306 033734 016746 145322      MOV      TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
3307 033740 016746 145314      MOV      TRAPPC,-(KSP)
3308 033744 000002      RTI
3309
3319

```

```

*****
*TEST 30      MTP1(SUPERVISOR) WITH SUPER. D-SPACE ENABLED
*
*      THIS TEST USES THE 'MTP1' INSTRUCTION TO ENSURE THAT THE
*      PREVIOUS MODE IS CLOKED CORRECTLY, AND THAT D-SPACE IS NOT
*      ENABLED.
*      IF THE CORRECT MODE IS NOT ENABLED, A NON-RESIDENT ABORT
*      WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
*****

```

```

3320 033746 000004      TST30: SCOPE
3321 033750 012767 034264 144272 20$:      MOV      #10$,MMVEC    ;SET M.M. VECTOR TO 10$
3322 033756 052767 000002 136532      BIS      #BIT1,MMR3    ;ENABLE SUPERVISOR D-SPACE
3323                      ;THIS WILL TEST DSTM = 1 MTP1
3324 033764 012767 034010 145116      MOV      #13$, $LPERR   ;SET LOOP ON ERROR POINTER TO 13$
3325 033772 012767 010340 143776      MOV      #010340,PSW    ;MAKE PREVIOUS MODE SUPERVISOR
3326 034000 012705 100000      MOV      #100000,R5    ;LOAD VIRTUAL ADDRESS INTO R5
3327 034004 012703 125252      MOV      #125252,R3    ;LOAD TEST DATA INTO R3
3328 034010 010346 13$:      MOV      R3,-(KSP)     ;PUSH TEST DATA ONTO KERNAL STACK
3329 034012 105067 136272      CLRB    KIPDR4        ;MAKE KERNAL PAGE 4 NON-RESIDENT
3330 034016 006615      MTP1   (R5)          ;LOAD TEST DATA INTO PHYSICAL 100000
3331 034020 112767 000006 136262      MOV      #006, KIPDR4  ;MAKE KERNAL I PAGE 4 RESIDENT
3332 034026 011504      MOV      (R5),R4      ;READ FROM ADDRESS 100000
3333 034030 020304      CMP      R3,R4        ;SEE IF DATA WAS STORED AT CORRECT PLACE
3334 034032 001401      BEQ      4$           ;BRANCH IF STORE WAS CORRECT

```

```

3335 034034 104035          ERROR +35          :INCORRECT STORE
3336          :THIS WILL TEST DSTM = 3 MTPI
3337 034036 012767 034056 145044 4$:  MOV #14$, $LPERR      ;SET LOOP ON ERROR POINTER TO 14$
3338 034044 012767 010340 143724      MOV #010340, PSW      ;MAKE PREVIOUS TYPE SUPERVISOR
3339 034052 012703 052525          MOV #52525, R3        ;LOAD TEST DATA INTO R3
3340 034056 010346          14$:  MOV R3, -(KSP)        ;PUSH TEST DATA ON KERNAL STACK
3341 034060 105067 136224          CLR# KIPDR4           ;MAKE KERNAL I PAGE 4 NON-RESIDENT
3342 034064 006637 100000          MTPI @#100000         ;LOAD TEST DATA INTO PHYSICAL 100000
3343 034070 112767 000006 136212      MOV# #006, KIPDR4     ;MAKE KERNAL I PAGE 4 RESIDENT
3344 034076 013704 100000          MOV @#100000, R4      ;READ FROM ADDRESS 100000
3345 034102 020304          CMP R3, R4           ;SEE IF DATA WAS STORED CORRECTLY
3346 034104 001401          BEQ 5$               ;BRANCH IF STORED CORRECTLY
3347 034106 104035          ERROR +35          :INCORRECT STORE
3348          :THIS WILL TEST DSTM = 4 MTPI
3349 034110 012767 034130 144772 5$:  MOV #15$, $LPERR      ;SET LOOP ON ERROR POINTER TO 15$
3350 034116 012767 010340 143652      MOV #010340, PSW      ;MAKE PREVIOUS TYPE SUPERVISOR
3351 034124 012703 125252          MOV #125252, R3        ;LOAD TEST DATA INTO R3
3352 034130 010346          15$:  MOV R3, -(KSP)        ;PUSH TEST DATA ON KERNAL STACK
3353 034132 012705 100002          MOV #100002, R5        ;LOAD VIRTUAL ADDRESS INTO R5
3354 034136 105067 136146          CLR# KIPDR4           ;MAKE KERNAL I PAGE 4 NON-RESIDENT
3355 034142 006645          MTPI -(R5)           ;LOAD TEST DATA INTO PHYSICAL 100000
3356 034144 112767 000006 136136      MOV# #006, KIPDR4     ;MAKE KERNAL I PAGE 4 RESIDENT
3357 034152 013704 100000          MOV @#100000, R4      ;READ FROM ADDRESS 100000
3358 034156 020304          CMP R3, R4           ;SEE IF DATA WAS STORED CORRECTLY
3359 034160 001401          BEQ 6$               ;BRANCH IF STORED CORRECTLY
3360 034162 104035          ERROR +35          :INCORRECT STORE
3361          :THIS WILL TEST DSTM = 6 MTPI
3362 034164 012767 034206 144716 6$:  MOV #16$, $LPERR      ;SET LOOP ON ERROR POINTER TO 16$
3363 034172 012767 010340 143576      MOV #010340, PSW      ;MAKE PREVIOUS TYPE SUPERVISOR
3364 034200 012703 052525          MOV #52525, R3        ;LOAD TEST DATA INTO R3
3365 034204 005005          CLR R5               ;MAKE R5 ZERO
3366 034206 010346          16$:  MOV R3, -(KSP)        ;PUSH TEST DATA ON KERNAL STACK
3367 034210 105067 136074          CLR# KIPDR4           ;MAKE KERNAL I PAGE 4 NON-RESIDENT
3368 034214 006665 100000          MTPI 100000(R5)       ;LOAD TEST DATA INTO PHYSICAL 100000
3369 034220 112767 000006 136062      MOV# #006, KIPDR4     ;MAKE KERNAL I PAGE 4 RESIDENT
3370 034226 013704 100000          MOV @#100000, R4      ;READ FROM ADDRESS 100000
3371 034232 020304          CMP R3, R4           ;SEE IF DATA WAS STORED CORRECTLY
3372 034234 001401          BEQ 7$               ;BRANCH IF STORED CORRECTLY
3373 034236 104035          ERROR +35          :INCORRECT STORE
3374 034240 012767 033750 144642 7$:  MOV #20$, $LPERR      ;SET LOOP ON ERROR POINTER TO 20; START OF TEST
3375 034246 012767 002466 143774      MOV #MMGMERR, MMVEC    ;RESTORE M.M. VECTOR TO NORMAL ROUTINE
3376 034254 042767 000002 136234      BIC #BIT1, MMR3        ;DISABLE SUPERVISOR D-SPACE
3377 034262 000410          BR TST31             ;BRANCH TO NEXT TEST
3378 034264 016700 143302          10$:  MOV MMR0, R0          ;SAVE MMR0 FOR ERROR TYPEOUT
3379 034270 016701 143300          MOV MMR1, R1          ;SAVE MMR1 FOR ERROR TYPEOUT
3380 034274 016702 143276          MOV MMR2, R2          ;SAVE MMR2 FOR ERROR TYPEOUT
3381 034300 104036          ERROR +36          ;TRIED TO LOAD A NON-RESIDENT PAGE 4
3382 034302 000002          RTI                  ;RETURN TO TEST
3383
3384

```

\*\*\*\*\*  
\*TEST 31 MTPI (USER) WITH USER D-SPACE ENABLED

\*  
\* THIS TEST USES THE 'MTPI' INSTRUCTION TO ENSURE THAT THE  
\* PREVIOUS MODE IS CLOCKED CORRECTLY, AND THAT D-SPACE IS NOT  
\* ENABLED.  
\* IF THE CORRECT MODE IS NOT ENABLED, A NON-RESIDENT ABORT  
\* WILL OCCUR AND TRAP TO 10\$, WHERE THE ERRORS ARE REPORTED.  
\*

```

:*****
:
TST31: SCOPE
20$: MOV #77400,SIPDR4 ;MAKE SIPDR4 NON-RESIDENT
MOV #77406,UIPDR4 ;MAKE UIPDR4 RESIDENT
MOV #10$,MMVEC ;SET M.M. VECTOR TO 10$
BIS #BIT0,MMR3 ;ENABLE USER D-SPACE
;THIS WILL TEST DSTM = 1 MTPJ
3385 034304 000004
3386 034306 012767 077400 135674
3387 034314 012767 077406 143266
3388 034322 012767 034636 143720
3388 034330 052767 000001 136160
3389
3390 034336 012767 034362 144544
3391 034344 012767 030340 143424
3392 034352 012705 100000
3393 034356 012703 125252
3394 034362 010346
3395 034364 105067 135720
3396 034370 006615
3397 034372 112767 000006 135710
3398 034400 011504
3399 034402 020304
3400 034404 001401
3401 034406 104035
3402
;THIS WILL TEST DSTM = 3 MTPJ
3403 034410 012767 034430 144472
3404 034416 012767 030340 143352
3405 034424 012703 052525
3406 034430 010346
3407 034432 105067 135652
3408 034436 006637 100000
3409 034442 112767 000006 135640
3410 034450 013704 100000
3411 034454 020304
3412 034456 001401
3413 034460 104035
3414
;THIS WILL TEST DSTM = 4 MTPJ
3415 034462 012767 034502 144420
3416 034470 012767 030340 143300
3417 034476 012703 125252
3418 034502 010346
3419 034504 012705 100002
3420 034510 105067 135574
3421 034514 006645
3422 034516 112767 000006 135564
3423 034524 013704 100000
3424 034530 020304
3425 034532 001401
3426 034534 104035
3427
;THIS WILL TEST DSTM = 6 MTPJ
3428 034536 012767 034560 144344
3429 034544 012767 030340 143224
3430 034552 012703 052525
3431 034556 005005
3432 034560 010346
3433 034562 105067 135522
3434 034566 006665 100000
3435 034572 112767 000006 135510
3436 034600 013704 100000
3437 034604 020304
3438 034606 001401
;*****

```



```

3439 034610 104035          ERROR +35          ;INCORRECT STORE
3440 034612 012767 034306 144270 7$:  MOV #20$, $LPERR ;SET LOOP ON ERROR POINTER TO START OF TEST
3441 034620 012767 002466 143422      MOV #MMGMERR, MMVEC ;RESTORE M.M. VECTOR TO NORMAL ROUTINE
3442 034626 042767 000001 135662      BIC #BIT0, MMR3    ;DISABLE USER D-SPACE
3443 034634 000410          BR TST32          ;BRANCH TO NEXT TEST
3444 034636 016700 142730          10$: MOV MMR0, R0      ;SAVE MMR0 FOR ERROR TYPEOUT
3445 034642 016701 142726          MOV MMR1, R1      ;SAVE MMR1 FOR ERROR TYPEOUT
3446 034646 016702 142724          MOV MMR2, R2      ;SAVE MMR2 FOR ERROR TYPEOUT
3447 034652 104036          ERROR +36        ;TRIED TO LOAD A NON-RESIDENT PAGE 4
3448 034654 000002          RTI              ;RETURN TO TEST
3457

```

\*\*\*\*\*  
\*TEST 32 MFPI(PREVIOUS=CURRENT=KERNEL)  
\*\*\*\*\*

\* THIS TEST CHECKS THAT IF BOTH PREVIOUS AND CURRENT MODES  
\* ARE KERNEL, AND THE SOURCE MODE IS 0, THE DESTINATION  
\* STACK IS NOT DECREMENTED BEFORE ACCESS.  
\* 'MFPI KSP' SHOULD PUSH THE NON-DECREMENTED VALUE  
\* OF KSP (1100) ONTO THE STACK (AT LOC. 1076).  
\*\*\*\*\*

```

3458 034656 000004          TST32: SCOPE
3458 034660 112767 000006 135442 1$: MOVB #6, KDPDR4 ;MAKE KDPDR4 RESIDENT
3459 034666 005037 177776      CLR #MPSW        ;SET PREVIOUS = CURRENT = KERNEL
3460 034672 012700 001100      MOV #STACK, R0  ;SETUP VALUE FOR STACK POINTER
3461 034674 010006          MOV R0, KSP     ;LOAD STACK POINTER
3462 034700 006506          MFPI KSP       ;THE VALUE 'STACK' SHOULD BE PUSHED
3463                                ;BEFORE BEING DECREMENTED
3464 034702 011601          MOV (KSP), R1  ;READ DATA WHICH WAS PUSHED
3465 034704 020001          CMP R0, R1     ;WAS THE ORIGINAL VALUE OF THE
3466                                ;STACK POINTER PUSHED?
3467 034706 001401          BEQ 2$        ;BRANCH IF YES
3468 034710 104037          ERROR +37     ;MFPI FETCHED WRONG DATA
3469                                ;FOR TIGHTER SCOPE LOOP
3470                                ;REPLACE ERROR CALL WITH
3471                                ;'BR 1$' = 000766
3472 034712 005740          2$: TST -(R0)    ;SETUP EXPECTED STACK POINTER VALUE
3473 034714 020600          CMP KSP, R0   ;WAS THE STACK POINTER DECREMENTED?
3474 034716 001401          BEQ 3$        ;BRANCH IF YES
3475 034720 104025          ERROR +25     ;STACK NOT PUSHED BY THE MFPI
3476                                ;FOR TIGHTER SCOPE LOOP
3477                                ;REPLACE ERROR CALL WITH
3478                                ;'BR 1$' = 000762
3479 034722 012706 001100          3$: MOV #STACK, KSP ;RESTORE STACK POINTER
3489

```

\*\*\*\*\*  
\*TEST 33 MFPI (SUPERVISOR) WITH SUPER D-SPACE ENABLED  
\*\*\*\*\*

\* THIS TEST CHECKS TO SEE THAT THE REFERENCE IS TO D-SPACE  
\* IF THE INSTRUCTION IS AN MFPI.  
\* IF THE CORRECT MODE IS NOT ENABLED, A NON-RESIDENT ABORT  
\* WILL OCCUR AND TRAP TO 10\$, WHERE THE ERRORS ARE REPORTED.  
\*\*\*\*\*

```

3490 034726 000004          TST33: SCOPE
3490 034730 012767 034752 144150      MOV #20$, $LPADR ;SET LOOP POINTER TO 20$
3491 034736 012767 034752 144144      MOV #20$, $LPERR ;SET LOOP ON ERROR POINTER TO 20$
3492 034744 012767 000600 135316      MOV #600, SDPAR4 ;MAP SDPAR4 TO 12K
3493 034752 012700 077400          20$: MOV #77400, R0  ;MAKE PAGE 4 IN ALL BUT SUPERVISOR D

```

```

3494                                     :AND KERNAL I NON-RESIDENT
3495 034756 010067 135346                MOV    R0,KDPDR4                :KERNAL D-SPACE PAGE 4
3496 034762 010067 135222                MOV    R0,SIPDR4                :SUPERVISOR I-SPACE PAGE 4
3497 034766 010067 142636                MOV    R0,UDPDR4                :USER D-SPACE PAGE 4
3498 034772 010067 142612                MOV    R0,UIPDR4                :USER I-SPACE PAGE 4
3499 034776 012767 077406 135224        MOV    #77406,SDPDR4           :MAKE SDPDR4 RESIDENT
3500 035004 012700 036514                MOV    #36514,R0               :LOAD DATA PATTERN INTO R0
3501 035010 010037 100000                MOV    R0,#100000              :LOAD DATA PATTERN INTO PHYS. 100000
3502 035014 012767 035246 143226        MOV    #10$,MMVEC              :SET M.M. VECTOR TO 10$
3503 035022 052767 000002 135466        BIS    #BIT1,MMR3              :ENABLE SUPERVISOR D-SPACE
3504 035030 105067 135254                CLR    KIPDR4                  :MAKE KERNAL I-SPACE PAGE 4 NON-RESIDENT
3505                                     :THE FOLLOWING WILL TEST DSTM=1 MFPD
3506                                     :
3507 035034 012767 035042 144046 2$:    MOV    #12$,SLPERR             :SET LOOP ON ERROR POINTER TO 12$
3508 035042 012767 010340 142726 12$:    MOV    #010340,PSW            :MAKE PREVIOUS MODE SUPERVISOR
3509 035050 012702 100000                MOV    #100000,R2              :LOAD VIRTUAL ADDRESS INTO R2
3510 035054 106512                MFPD    (R2)                   :READ FROM PHYSICAL 100000
3511 035056 012601                MOV    (KSP)+,R1              :POP KERNAL STACK INTO R1
3512 035060 020001                CMP    R0,R1                  :WAS DATA FETCHED SAME AS DATA STORED
3513 035062 001401                BEQ    4$                      :BRANCH IF CORRECT DATA WAS FETCHED
3514 035064 104037                ERROR  +37                    :WRONG DATA WAS FETCHED
3515                                     :THE FOLLOWING WILL TEST DSTM=2 MFPD
3516                                     :
3517 035066 012767 035074 144014 4$:    MOV    #14$,SLPERR             :SET LOOP ON ERROR POINTER TO 14$
3518 035074 012767 010340 142674 14$:    MOV    #010340,PSW            :MAKE PREVIOUS MODE SUPERVISOR
3519 035102 012702 100000                MOV    #100000,R2              :LOAD VIRTUAL ADDRESS INTO R2
3520 035106 106522                MFPD    (R2)+                  :READ FROM PHYSICAL 100000
3521 035110 012601                MOV    (KSP)+,R1              :POP KERNAL STACK INTO R1
3522 035112 020001                CMP    R0,R1                  :WAS DATA FETCHED SAME AS DATA STORED
3523 035114 001401                BEQ    5$                      :BRANCH IF CORRECT DATA WAS FETCHED
3524 035116 104037                ERROR  +37                    :WRONG DATA WAS FETCHED
3525                                     :THE FOLLOWING WILL TEST DSTM=3 MFPD
3526                                     :
3527 035120 012767 035126 143762 5$:    MOV    #15$,SLPERR             :SET LOOP ON ERROR POINTER TO 15$
3528 035126 012767 010340 142642 15$:    MOV    #010340,PSW            :MAKE PREVIOUS MODE SUPERVISOR
3529 035134 012702 100000                MOV    #100000,R2              :LOAD VIRTUAL ADDRESS INTO R2
3530 035140 106537 100000                MFPD    #100000                :READ FROM PHYSICAL 100000
3531 035144 012601                MOV    (KSP)+,R1              :POP KERNAL STACK INTO R1
3532 035146 020001                CMP    R0,R1                  :WAS DATA FETCHED SAME AS DATA STORED
3533 035150 001401                BEQ    6$                      :BRANCH IF CORRECT DATA WAS FETCHED
3534 035152 104037                ERROR  +37                    :WRONG DATA WAS FETCHED
3535                                     :THE FOLLOWING WILL TEST DSTM=4 MFPD
3536                                     :
3537 035154 012767 035162 143726 6$:    MOV    #16$,SLPERR             :SET LOOP ON ERROR POINTER TO 16$
3538 035162 012767 010340 142606 16$:    MOV    #010340,PSW            :MAKE PREVIOUS MODE SUPERVISOR
3539 035170 012702 100002                MOV    #100002,R2              :LOAD VIRTUAL ADDRESS INTO R2
3540 035174 106542                MFPD    -(R2)                  :READ FROM PHYSICAL 100000
3541 035176 012601                MOV    (KSP)+,R1              :POP KERNAL STACK INTO R1
3542 035200 020001                CMP    R0,R1                  :WAS DATA FETCHED SAME AS DATA STORED
3543 035202 001401                BEQ    7$                      :BRANCH IF CORRECT DATA WAS FETCHED
3544 035204 104037                ERROR  +37                    :WRONG DATA WAS FETCHED
3545 035206 012767 002466 143034 7$:    MOV    #MGMEERR,MMVEC          :RESTORE M.M. VECTOR
3546 035214 012767 034752 143666        MOV    #20$,SLPERR             :SET LOOP ON ERROR POINTER TO START OF ROUTINE
3547 035222 042767 000002 135266        BIC    #BIT1,MMR3              :DISABLE SUPERVISOR D-SPACE
3548 035230 012700 077406                MOV    #77406,R0               :SET UP R0 FOR 4K RESIDENT R/W
3549 035234 010067 135050                MOV    R0,KIPDR4              :MAKE KIPDR4 RESIDENT
3550 035240 010067 142364                MOV    R0,UDPDR4              :MAKE UDPDR4 RESIDENT

```

```

3551 035244 000423 BR TST34 ;:BRANCH TO NEXT TEST
3552 035246 016767 142320 144010 10$: MOV MMR0,WASSRO ;:SAVE MMR0 FOR ERROR TYPEOUT
3553 035254 016767 142314 144004 MOV MMR1,WASSR1 ;:SAVE MMR1 FOR ERROR TYPEOUT
3554 035262 016767 142310 144000 MOV MMR2,WASSR2 ;:SAVE MMR2 FOR ERROR TYPEOUT
3555 035270 012667 143764 MOV (KSP)+,TRAPPC ;:SAVE PC & PS OF TRAP
3556 035274 012667 143762 MOV (KSP)+,TRAPPS
3557 035300 104040 ERROR +40 ;:TRIED TO READ NON-RESIDNT PAGE
3558 035302 016746 143754 MOV TRAPPS,-(KSP) ;:PUT PC & PS OF TRAP ON STACK
3559 035306 016746 143746 MOV TRAPPC,-(KSP)
3560 035312 000002 RTI ;:RETURN TO TEST
3561
3573
  
```

\*\*\*\*\*  
 :\*TEST 34 MFPI (USER/PREV USER) WITH USER D-SPACE ENABLED

THIS TEST CHECKS THAT IF THE INSTRUCTION IS EITHER MFPI OR  
 MTPPI AND BOTH THE PRESENT AND PREVIOUS MODES ARE USER, THEN  
 D-SPACE IS USED IF IT IS ENABLED. IN THIS WAY AN OPERATING  
 SYSTEM CAN MAKE PROPRIETARY CODE "EXECUTE ONLY" FOR THE USER.

IF THE CORRECT MODE IS NOT ENABLED, A NON-RESIDENT ABORT WILL  
 OCCUR AND TRAP TO 10\$, WHERE THE ERRORS ARE REPORTED.

\*\*\*\*\*

```

035314 000004 TST34: SCOPE
3574 035316 012767 035332 143562 MOV #20$, $LPADR ;:SET LOOP POINTER TO 20$
3575 035324 012767 035332 143556 MOV #20$, $LPERR ;:SET LOOP ON ERROR POINTER TO 20$
3576 035332 012767 000600 142330 20$: MOV #600,UDPAR4 ;:MAP UDPAR4 TO 12K
3577 035340 012700 036514 MOV #36514,R0 ;:LOAD DATA PATTERN INTO R0
3578 035344 010037 100000 MOV R0,#100000 ;:LOAD DATA PATTERN INTO PHYS. 100000
3579 035350 012767 035606 142672 MOV #10$,MMVEC ;:SET M.M. VECTOR TO 10$
3580 035356 052767 000001 135132 BIS #BIT0,MMR3 ;:ENABLE USER D-SPACE
3581 035364 105067 134640 CLRB SDPDR4 ;:MAKE SDPDR4 NON-RESIDENT
3582 035370 105067 134714 CLRB KIPDR4 ;:MAKE KERNAL I-SPACE PAGE 4 NON-RESIDENT
3583 ;:THE FOLLOWING WILL TEST DSTM=1 MFPI
3584
3585 035374 012767 035402 143506 2$: MOV #12$, $LPERR ;:SET LOOP ON ERROR POINTER TO 12$
3586 035402 012767 170340 142366 12$: MOV #170340,PSW ;:SET PREVIOUS MODE TO USER
3587 035410 012702 100000 MOV #100000,R2 ;:LOAD VIRTUAL ADDRESS INTO R2
3588 035414 006512 MFPI (R2) ;:READ FROM PHYSICAL 100000
3589 035416 012601 MOV (USP)+,R1 ;:POP USER STACK INTO R1
3590 035420 020001 CMP R0,R1 ;:WAS DATA FETCHED SAME AS DATA STORED
3591 035422 001401 BEQ 4$ ;:BRANCH IF CORRECT DATA WAS FETCHED
3592 035424 104037 ERROR +37 ;:WRONG DATA WAS FETCHED
3593 ;:THE FOLLOWING WILL TEST DSTM=2 MFPI
3594
3595 035426 012767 035434 143454 4$: MOV #14$, $LPERR ;:SET LOOP ON ERROR POINTER TO 14$
3596 035434 012767 170340 142334 14$: MOV #170340,PSW ;:SET PREVIOUS MODE TO USER
3597 035442 012702 100000 MOV #100000,R2 ;:LOAD VIRTUAL ADDRESS INTO R2
3598 035446 006522 MFPI (R2)+ ;:READ FROM PHYSICAL 100000
3599 035450 012601 MOV (USP)+,R1 ;:POP USER STACK INTO R1
3600 035452 020001 CMP R0,R1 ;:WAS DATA FETCHED SAME AS DATA STORED
3601 035454 001401 BEQ 5$ ;:BRANCH IF CORRECT DATA WAS FETCHED
3602 035456 104037 ERROR +37 ;:WRONG DATA WAS FETCHED
3603 ;:THE FOLLOWING WILL TEST DSTM=3 MFPI
3604
3605 035460 012767 035466 143422 5$: MOV #15$, $LPERR ;:SET LOOP ON ERROR POINTER TO 15$
3606 035466 012767 170340 142302 15$: MOV #170340,PSW ;:SET PREVIOUS MODE TO USER
  
```

```

3607 035474 012702 100000      MOV      #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3608 035500 006537 100000      MFPI     @#100000      ;READ FROM PHYSICAL 100000
3609 035504 012601              MOV      (USP)+,R1      ;POP USER STACK INTO R1
3610 035506 020001              CMP      R0,R1          ;WAS DATA FETCHED SAME AS DATA STORED
3611 035510 001401              BEQ      6$             ;BRANCH IF CORRECT DATA WAS FETCHED
3612 035512 104037              ERROR   +37            ;WRONG DATA WAS FETCHED
3613                               ;THE FOLLOWING WILL TEST DSTM=4 MFPI
3614                               ;
3615 035514 012767 035522 143366 6$:  MOV      #16$, $LPERR   ;SET LOOP ON ERROR POINTER TO 16$
3616 035522 012767 170340 142246 16$:  MOV      #170340,PSW    ;SET PREVIOUS MODE TO USER
3617 035530 012702 100002              MOV      #100002,R2    ;LOAD VIRTUAL ADDRESS INTO R2
3618 035534 006542              MFPI     -(R2)         ;READ FROM PHYSICAL 100000
3619 035536 012601              MOV      (USP)+,R1      ;POP USER STACK INTO R1
3620 035540 020001              CMP      R0,R1          ;WAS DATA FETCHED SAME AS DATA STORED
3621 035542 001401              BEQ      7$             ;BRANCH IF CORRECT DATA WAS FETCHED
3622 035544 104037              ERROR   +37            ;WRONG DATA WAS FETCHED
3623 035546 012767 002466 142474 7$:  MOV      #MGMERR,MMVEC  ;RESTORE M.M. VECTOR
3624 035554 012767 035332 143326              MOV      #20$, $LPERR   ;SET LOOP ON ERROR POINTER TO START OF ROUTINE
3625 035562 042767 000001 134726              BIC      #BIT0,MMR3     ;DISABLE USER D-SPACE
3626 035570 012767 000340 142200              MOV      #340,PSW      ;MAKE PRESENT MODE KERNAL
3627 035576 112767 000006 134504              MOV      #6,KIPDR4     ;MAKE KIPDR4 RESIDENT
3628 035604 000423              BR       TST35         ;BRANCH TO NEXT TEST
3629 035606 016767 141760 143450 10$:  MOV      MMR0,WASSRO    ;SAVE MMR0 FOR ERROR TYPEOUT
3630 035614 016767 141754 143444              MOV      MMR1,WASSR1    ;SAVE MMR1 FOR ERROR TYPEOUT
3631 035622 016767 141750 143440              MOV      MMR2,WASSR2    ;SAVE MMR2 FOR ERROR TYPEOUT
3632 035630 012667 143424              MOV      (KSP)+,TRAPPC ;SAVE PC & PS OF TRAP
3633 035634 012667 143422              MOV      (KSP)+,TRAPPS ;
3634 035640 104040              ERROR   +40            ;TRIED TO READ NON-RESIDENT PAGE
3635 035642 016746 143414              MOV      TRAPPS,-(KSP)  ;PUT PC & PS OF TRAP ON STACK
3636 035646 016746 143406              MOV      TRAPPC,-(KSP) ;
3637 035652 000002              RTI                    ;RETURN TO TEST
  
```

3647

```

*****
*TEST 35      TEST CSM INSTRUCTION
*****
*          THIS TEST CHECKS OUT THE CSM (CALL SUPERVISOR MODE)
*          INSTRUCTION TO MAKE SURE IT CAN BE ENABLED AND
*          DISABLED, AND THAT IT WORKS PROPERLY IN THE PROPER
*          MODES WHEN ENABLED.
*****

```

```

035654 000004
3648 035656 005067 134634
3649 035662 005067 142110
3650 035666 016767 142116 143310
3651 035674 012767 035710 142106
3652 035702 005267 141664
3653

```

```

TST35: SCOPE
CLR      MMR3      ;MAKE SURE MMR3 IS CLEARED
CLR      PSW       ;MAKE SURE PSW PREVIOUS=CURRENT=KERNEL
MOV      RESVEC,$TMP3 ;SAVE TRAPS TO 10 VECTOR
MOV      #1$,RESVEC ;TRAPS TO 10 GO TO 1$
INC      MMR0      ;TURN ON MEMORY MANAGEMENT
;THE FOLLOWING ".WORD 007000" IS ACTUALLY THE "CSM" INSTRUCTION. AT THE TIME
;THIS WAS COMPILED, COMPILER WAS UNABLE TO INTERPRET THE "CSM" INSTRUCTION
;IN THE SOURCE FILE, SO A MACRO PRINTS THIS COMMENT AND THE ".WORD 007000"
;LINE WHENEVER IT SEES THE "CSM" IN THE SOURCE FILE.

```

```

035706 007000
3654 035710 005067 141656 1$:
3655 035714 016767 143264 142066
3656 035722 022726 035710
3657 035726 001403
3658 035730 022626
3659 035732 104041
3660 035734 000401
3661 035736 005726 15$:
3662 035740 012767 035762 142042 16$:
3663 035746 052767 040000 142022
3664 035754 005267 141612
3665

```

```

.WORD 007000 ;THE "CSM" INSTRUCTION
CLR      MMR0      ;TURN OFF MEMORY MANAGEMENT
MOV      $TMP3,RESVEC ;RESTORE TRAPS TO 10 VECTOR
CMP      #1$,(SP)+ ;SEE IF IT WAS AN EXPECTED TRAP
BEQ      15$      ;BRANCH AROUND ERROR CALL IF IT WAS
CMP      (SP)+,(SP)+ ;CORRECT STACK
ERROR    +41      ;ILLEGAL CSM DID NOT TRAP TO 10
BR       16$      ;BRANCH AROUND STACK CORRECTION
TST      (SP)+    ;CORRECT STACK
MOV      #2$,RESVEC ;TRAPS TO 10 GO TO 2$
BIS      #40000,PSW ;NOW IN SUPERVISOR MODE
INC      MMR0      ;TURN ON MEMORY MANAGEMENT
;THE FOLLOWING ".WORD 007000" IS ACTUALLY THE "CSM" INSTRUCTION. AT THE TIME
;THIS WAS COMPILED, COMPILER WAS UNABLE TO INTERPRET THE "CSM" INSTRUCTION
;IN THE SOURCE FILE, SO A MACRO PRINTS THIS COMMENT AND THE ".WORD 007000"
;LINE WHENEVER IT SEES THE "CSM" IN THE SOURCE FILE.

```

```

035760 007000
3666 035762 005067 141604 2$:
3667 035766 016767 143212 142014
3668 035774 022726 035762
3669 036000 001403
3670 036002 022626
3671 036004 104041
3672 036006 000401
3673 036010 005726 25$:
3674 036012 012767 036034 141770 26$:
3675 036020 052767 140000 141750
3676 036026 005267 141540
3677

```

```

.WORD 007000 ;THE "CSM" INSTRUCTION
CLR      MMR0      ;TURN OFF MEMORY MANAGEMENT
MOV      $TMP3,RESVEC ;RESTORE TRAPS TO 10 VECTOR
CMP      #2$,(SP)+ ;SEE IF IT WAS AN EXPECTED TRAP
BEQ      25$      ;BRANCH AROUND ERROR CALL IF IT WAS
CMP      (SP)+,(SP)+ ;CORRECT STACK
ERROR    +41      ;ILLEGAL CSM DID NOT TRAP TO 10
BR       26$      ;BRANCH AROUND STACK CORRECTION
TST      (SP)+    ;CORRECT STACK
MOV      #3$,RESVEC ;TRAPS TO 10 GO TO 3$
BIS      #140000,PSW ;NOW IN USER MODE
INC      MMR0      ;TURN ON MEMORY MANAGEMENT
;THE FOLLOWING ".WORD 007000" IS ACTUALLY THE "CSM" INSTRUCTION. AT THE TIME
;THIS WAS COMPILED, COMPILER WAS UNABLE TO INTERPRET THE "CSM" INSTRUCTION
;IN THE SOURCE FILE, SO A MACRO PRINTS THIS COMMENT AND THE ".WORD 007000"
;LINE WHENEVER IT SEES THE "CSM" IN THE SOURCE FILE.

```

```

036032 007000
3678 036034 005067 141532 3$:
3679 036040 016767 143140 141742
3680 036046 022726 036034
3681 036052 001403
3682 036054 022626

```

```

.WORD 007000 ;THE "CSM" INSTRUCTION
CLR      MMR0      ;TURN OFF MEMORY MANAGEMEN'
MOV      $TMP3,RESVEC ;RESTORE TRAPS TO 10 VECTOR
CMP      #3$,(SP)+ ;SEE IF IT WAS AN EXPECTED TRAP
BEQ      35$      ;BRANCH AROUND ERROR CALL IF IT WAS
CMP      (SP)+,(SP)+ ;CORRECT STACK

```

```

3683 036056 104041          ERROR +41          ;ILLEGAL CSM DID NOT TRAP TO 10
3684 036060 000401          BR 36$           ;BRANCH AROUND STACK CORRECTION
3685 036062 005726          TST (SP)+        ;CORRECT STACK
3686 036064 052767 000010 134424 35$:  BIS #BIT3,MMR3   ;TURN ON CSM ENABLE
3687 036072 005067 141700          CLR PSW          ;MAKE SURE WE ARE IN KERNEL MODE
3688 036076 012767 036112 141704  MOV #4$,RESVEC   ;TRAPS TO 10 GO TO 4$
3689 036104 005267 141462          INC MMR0         ;TURN ON MEMORY MANAGEMENT
3690                                     ;THE FOLLOWING ".WORD 007000" IS ACTUALLY THE "CSM" INSTRUCTION. AT THE TIME
                                     ;THIS WAS COMPILED, COMPILER WAS UNABLE TO INTERPRET THE "CSM" INSTRUCTION
                                     ;IN THE SOURCE FILE, SO A MACRO PRINTS THIS COMMENT AND THE ".WORD 007000"
                                     ;LINE WHENEVER IT SEES THE "CSM" IN THE SOURCE FILE.
                                     .WORD 007000      ;THE "CSM" INSTRUCTION
3691 036110 007000          CLR MMR0         ;TURN OFF MEMORY MANAGEMENT
3692 036112 005067 141454          MOV $TMP3,RESVEC ;RESTORE TRAPS TO 10 VECTOR
3693 036116 016767 143062 141664  CMP #4$, (SP)+   ;SEE IF IT WAS AN EXPECTED TRAP
3694 036124 022726 036112          BEQ 45$         ;BRANCH AROUND ERROR CALL IF IT WAS
3695 036130 001403          CMP (SP)+, (SP)+ ;CORRECT STACK
3696 036132 022626          ERROR +41      ;ILLEGAL CSM DID NOT TRAP TO 10
3697 036134 104041          BR 46$         ;BRANCH AROUND STACK CORRECTION
3698 036136 000401          TST (SP)+      ;CORRECT STACK
3699 036140 005726          JSR PC,APRINIT ;RESET ALL MEMORY MANAGEMENT REGISTERS
3700 036142 004767 143662          ;*
3701                                     ;* CSM FROM SUPERVISOR I-SPACE
3702                                     ;*
3703 036146 012700 052525          MOV #52525,R0   ;SET UP R0
3704 036152 012737 037146 060000  MOV #CSMWSE,@#60000 ;SET UP CSM WRONG SPACE ERROR
3705 036160 012767 000600 134072  MOV #600,SDPAR0 ;IF WE GO TO D-SPACE, FLAG AS AN ERROR
3706 036166 012767 000032 134322  MOV #32,MMR3    ;ENABLE CSM, 22-BIT, SUPERVISOR SPACE
3707 036174 012702 040000          MOV #40000,R2   ;PUT SUPERVISOR MODE IN R2
3708 036200 012767 036460 141602  MOV #CSMPC,RESVEC ;SET UP TRAPS TO 10 VECTOR
3709 036206 050267 141564          BIS R2,PSW      ;GO TO SUPERVISOR MODE
3710 036212 012706 000700          MOV #SUPSTK,SSP ;SET THE SUPERVISOR STACK
3711 036216 012767 036224 001000  MOV #80$,SRCALL ;MOVE ADDRESS OF SUBROUTINE CALL TO SRCALL
3712 036224 004767 000210          JSR PC,CSMSUB   ;TEST CSM INSTRUCTION SUBROUTINE
3713 036230 005067 141542          CLR PSW         ;RETURN TO KERNEL MODE
3714 036234 005067 134020          CLR SDPAR0     ;RESET SDPAR0 TO FIRST 4K
3715                                     ;*
3716                                     ;* CSM FROM SUPERVISOR D-SPACE
3717                                     ;*
3718 036240 012767 000600 133772  MOV #600,SIPARO ;IF WE GO TO I-SPACE, FLAG IT AS AN ERROR
3719 036246 012737 036460 060010  MOV #CSMPC,@#60010 ;SET UP TRAPS TO 10 + 60000 AS MAPPED BY SIPARO
3720 036254 050267 141516          BIS R2,PSW      ;GO TO SUPERVISOR MODE
3721 036260 012767 036266 000736  MOV #81$,SRCALL ;MOVE ADDRESS OF SUBROUTINE CALL TO SRCALL
3722 036266 004767 000146          JSR PC,CSMSUB   ;TEST CSM INSTRUCTION SUBROUTINE
3723 036272 005067 141500          CLR PSW         ;RETURN TO KERNEL MODE
3724 036276 005067 133736          CLR SIPARO     ;RESET SIPARO TO FIRST 4K
3725                                     ;*
3726                                     ;* CSM FROM USER I-SPACE
3727                                     ;*
3728 036302 012767 000031 134206  MOV #31,MMR3    ;TURN ON USER, 22-BIT, CSM
3729 036310 012767 000600 141342  MOV #600,UDPAR0 ;IF WE GO TO D-SPACE, FLAG AS AN ERROR
3730 036316 012702 140000          MOV #140000,R2  ;MOVE USER MODE TO R2
3731 036322 012767 036460 141460  MOV #CSMPC,RESVEC ;SET UP TRAPS TO 10 VECTOR
3732 036330 050267 141442          BIS R2,PSW      ;GO TO USER MODE
3733 036334 012706 000600          MOV #USESTK,USP ;SET USER STACK
3734 036340 012767 036346 000656  MOV #82$,SRCALL ;MOVE ADDRESS OF SUBROUTINE CALL TO SRCALL
3735 036346 004767 000066          JSR PC,CSMSUB   ;TEST CSM INSTRUCTION SUBROUTINE

```

```

3736 036352 005067 141420 CLR PSW ;RETURN TO KERNEL MODE
3737 036356 005067 141276 CLR UDPARO ;RESET UDPARO TO FIRST 4K
3738
3739 ;*
3740 ;* CSM FROM USER D-SPACE
3741 036362 012767 000600 141250 MOV #600,UIPARO ;IF WE GO TO I-SPACE, FLAG AS AN ERROR
3742 036370 012767 036460 141412 MOV #CSMPC,RESVEC ;SET UP TRAPS TO 10 VECTOR
3743 036376 050267 141374 BIS R2,PSW ;GO TO USER MODE
3744 036402 012706 000600 MOV #USESTK,USP ;RESET USER STACK
3745 036406 012767 036414 000610 MOV #83$,SRCALL ;MOVE ADDRESS OF SUBROUTINE CALL TO SRCALL
3746 036414 004767 000020 83$: JSR PC,CSMSUB ;TEST CSM INSTRUCTION SUBROUTINE
3747 036420 005067 141352 CLR PSW ;RETURN TO KERNEL MODE
3748 036424 005067 141210 CLR UI,AR0 ;RESET UIPARO TO FIRST 4K
3749 036430 005067 134062 CLR MMR3 ;TURN OFF 22-BIT, CSM INSTRUCTION D-SPACE
3750 036434 000167 000666 JMP $EOP ;JUMP TO END OF PASS ROUTINE
    
```

\*\*\*\*\*NOTE\*\*\*\*\*  
 ; IN ALL ERROR CALLS IN THIS SUBROUTINE, THE 'SUB CALL' FIGURE IS THE  
 ; ADDRESS OF THE JSR THAT CALLED THE SUBROUTINE THAT RESULTED IN THE  
 ; ERROR. BECAUSE THE SUBROUTINE IS USED 4 TIMES IN THE TEST, IT IS  
 ; IMPORTANT TO KNOW WHICH JSR WAS ACTIVE WHEN THE ERROR WAS CALLED.

```

3751
3752
3753
3754
3755
3756
3757
3758 036440 016703 141332 CSMSUB: MOV PSW,R3 ;STORE PRE-CSM PSW IN R3
3759 036444 010605 MOV SP,R5 ;MOVE STACK POINTER VALUE TO R5
3760 036446 162705 000006 SUB #6,R5 ;SUBTRACT 6 FROM R5 - EXPECT 3 PUSHES ON STACK
3761 036452 005267 141114 INC MMR0 ;TURN ON MEMORY MANAGEMENT
3762
;THE FOLLOWING '.WORD 007000' IS ACTUALLY THE 'CSM' INSTRUCTION. AT THE TIME
;THIS WAS COMPILED, COMPILER WAS UNABLE TO INTERPRET THE 'CSM' INSTRUCTION
;IN THE SOURCE FILE, SO A MACRO PRINTS THIS COMMENT AND THE '.WORD 007000'
;LINE WHENEVER IT SEES THE 'CSM' IN THE SOURCE FILE.
3763 036456 007000 .WORD 007000 ;THE 'CSM' INSTRUCTION
3764 036460 016767 142520 141322 CSMP3: MOV $TMP3,RESVEC ;RESTORE TRAPS TO 10 VECTOR
3765 036466 022716 036460 CMP #CSMPC,(SP) ;SEE IF CSM EXECUTED
3766 036472 001006 BNE 20$ ;BRANCH IF IT DID
3767 036474 005067 141072 CLR MMR0 ;TURN OFF MEMORY MANAGEMENT
3768 036500 022626 CMP (SP)+,(SP)+ ;CORRECT STACK
3769 036502 104051 ERROR +51 ;CSM ABORTED WHEN IT SHOULD NOT HAVE
3770 036504 000167 000470 JMP CSMRET ;JUMP TO SUBROUTINE RETURN
3771 036510 010346 20$: MOV R3,-(SSP) ;PUSH PRE-CSM PSW ON STACK
3772 036512 042703 037777 BIC #3777,R3 ;WIPE OUT ALL BITS EXCEPT <15:14>,
3773 036516 000241 CLC ;CLEAR THE CARRY BIT,
3774 036520 006003 ROR R3 ;MOVE <15:14> OVER TO
3775 036522 006003 ROR R3 ;THE RIGHT TO <13:12>,
3776 036524 052703 040000 BIS #40000,R3 ;AND SET ,14>, PUTTING '01' IN <15:14>
3777 036530 112767 000005 000465 MOV #5,LCNT+1 ;MOVE 5 (LOOP COUNTER FOR STACK SUB RTNS) TO LCNT+1
3778 036536 016704 141234 MOV PSW,R4 ;MOVE CURRENT PSW TO R4
3779 036542 042704 007777 BIC #7777,R4 ;WIPE OUT ALL BITS BUT <15:12>
3780 036550 001416 CMP R3,R4 ;SEE IF PSW STATUS BITS MATCHES CALCULATED VALUE
3781 036552 005067 141014 BEQ 21$ ;BRANCH AROUND ERROR CALL IF OK
3782 036556 016767 141214 142422 CLR MMR0 ;TURN OFF MEMORY MANAGEMENT
3783 036564 010367 142400 MOV PSW,$TMP4 ;STORE PSW FOR PRINTOUT
3784 036570 004767 000434 MOV R3,$REG3 ;STORE R3 FOR PRINTOUT
3785 036574 104042 JSR PC,POPSTK ;GO POP STACK 5 TIMES
3786 036576 004767 000474 ERROR +42 ;NOT SUPERVISOR MODE
3787 036602 005267 140764 JSR PC,PSHSTK ;GO RESTORE STACK
3788 036606 016704 141164 21$: INC MMR0 ;TURN MEMORY MANAGEMENT BACK ON
MOV PSW,R4 ;MOVE CURRENT PSW TO R4
    
```



|      |        |        |        |        |       |               |   |
|------|--------|--------|--------|--------|-------|---------------|---|
| 3789 | 036612 | 042704 | 037777 |        | BIC   | #37777,R4     | :CLEAR ALL BITS EXCEPT <15:14>                  |
| 3790 | 036616 | 022704 | 040000 |        | CMP   | #40000,R4     | :SEE IF SUPERVISOR MODE                         |
| 3791 | 036622 | 001416 |        |        | BEQ   | 1\$           | :BRANCH AROUND ERROR CALL IF OK                 |
| 3792 | 036624 | 005067 | 140742 |        | CLR   | MMRO          | :TURN OFF MEMORY MANAGEMENT                     |
| 3793 | 036630 | 010467 | 142352 |        | MOV   | R4,\$TMP4     | :STORE R4 FOR PRINTOUT                          |
| 3794 | 036634 | 012767 | 040000 | 142326 | MOV   | #40000,\$REG3 | :STORE 40000 FOR PRINTOUT                       |
| 3795 | 036642 | 004767 | 000362 |        | JSR   | PC,POPSTK     | :GO POP STACK 5 TIMES                           |
| 3796 | 036646 | 104042 |        |        | ERROR | +42           | :NOT SUPERVISOR MODE                            |
| 3797 | 036650 | 004767 | 000422 |        | JSR   | PC,PSHSTK     | :GO RESTORE STACK                               |
| 3798 | 036654 | 005267 | 140712 |        | INC   | MMRO          | :TURN MEMORY MANAGEMENT BACK ON                 |
| 3799 | 036660 | 016704 | 141112 | 1\$:   | MOV   | PSW,R4        | :MOVE CURRENT PSW TO R4                         |
| 3800 | 036664 | 042704 | 147777 |        | BIC   | #147777,R4    | :CLEAR ALL BITS EXCEPT <13:12>                  |
| 3801 | 036670 | 006304 |        |        | ASL   | R4            | :MOVE <13:12> BITS IN R4                        |
| 3802 | 036672 | 006304 |        |        | ASL   | R4            | :TO THE <15:14> POSITION                        |
| 3803 | 036674 | 020204 |        |        | CMP   | R2,R4         | :SEE IF PREVIOUS MODE IS OK                     |
| 3804 | 036676 | 001415 |        |        | BEQ   | 2\$           | :BRANCH AROUND ERROR CALL IF OK                 |
| 3805 | 036700 | 005067 | 140666 |        | CLR   | MMRO          | :TURN OFF MEMORY MANAGEMENT                     |
| 3806 | 036704 | 010267 | 142256 |        | MOV   | R2,\$REG2     | :STORE EXPECTED PREVIOUS MODE FOR PRINTOUT      |
| 3807 | 036710 | 010467 | 142272 |        | MOV   | R4,\$TMP4     | :STORE OBTAINED PREVIOUS MODE FOR PRINTOUT      |
| 3808 | 036714 | 004767 | 000310 |        | JSR   | PC,POPSTK     | :GO POP STACK 5 TIMES                           |
| 3809 | 036720 | 104043 |        |        | ERROR | +43           | :WRONG PREVIOUS MODE                            |
| 3810 | 036722 | 004767 | 000350 |        | JSR   | PC,PSHSTK     | :GO RESTORE STACK                               |
| 3811 | 036726 | 005267 | 140640 |        | INC   | MMRO          | :TURN MEMORY MANAGEMENT BACK ON                 |
| 3812 | 036732 | 012603 |        | 2\$:   | MOV   | (SSP)+,R3     | :RESTORE PRE-CSM PSW IN R3                      |
| 3813 | 036734 | 105367 | 000263 |        | DECB  | LCNT+1        | :DECREMENT LCNT+1 TO 4                          |
| 3814 | 036740 | 020506 |        |        | CMP   | R5,SSP        | :SEE IF STACK POINTER VALUE WAS TRANSFERED      |
| 3815 | 036742 | 001415 |        |        | BEQ   | 3\$           | :BRANCH AROUND ERROR CALL IF SO                 |
| 3816 | 036744 | 005067 | 140622 |        | CLR   | MMRO          | :TURN OFF MEMORY MANAGEMENT                     |
| 3817 | 036750 | 010567 | 142220 |        | MOV   | R5,\$REG5     | :STORE R5 FOR PRINTOUT                          |
| 3818 | 036754 | 010667 | 142226 |        | MOV   | SSP,\$TMP4    | :STORE SSP FOR PRINTOUT                         |
| 3819 | 036760 | 004767 | 000244 |        | JSR   | PC,POPSTK     | :GO POP STACK 4 TIMES                           |
| 3820 | 036764 | 104044 |        |        | ERROR | +44           | :INCORRECT STACK                                |
| 3821 | 036766 | 004767 | 000304 |        | JSR   | PC,PSHSTK     | :GO RESTORE STACK                               |
| 3822 | 036772 | 005267 | 140574 |        | INC   | MMRO          | :TURN MEMORY MANAGEMENT BACK ON                 |
| 3823 | 036776 | 012601 |        | 3\$:   | MOV   | (SSP)+,R1     | :POP ARGUMENT OFF OF STACK                      |
| 3824 | 037000 | 105367 | 000217 |        | DECB  | LCNT+1        | :DECREMENT LCNT+1 TO 3                          |
| 3825 | 037004 | 020001 |        |        | CMP   | R0,R1         | :COMPARE R0 WITH THE ARGUMENT THAT WAS ON STACK |
| 3826 | 037006 | 001415 |        |        | BEQ   | 4\$           | :IF EQUAL, BRANCH AROUND ERROR                  |
| 3827 | 037010 | 005067 | 140556 |        | CLR   | MMRO          | :TURN OFF MEMORY MANAGEMENT                     |
| 3828 | 037014 | 010067 | 142142 |        | MOV   | R0,\$REG0     | :STORE R0 FOR PRINTOUT                          |
| 3829 | 037020 | 010167 | 142140 |        | MOV   | R1,\$REG1     | :STORE R1 FOR PRINTOUT                          |
| 3830 | 037024 | 004767 | 000200 |        | JSR   | PC,POPSTK     | :GO POP STACK 3 TIMES                           |
| 3831 | 037030 | 104045 |        |        | ERROR | +45           | :INCORRECT ARGUMENT                             |
| 3832 | 037032 | 004767 | 000240 |        | JSR   | PC,PSHSTK     | :GO RESTORE STACK                               |
| 3833 | 037036 | 005267 | 140530 |        | INC   | MMRO          | :TURN MEMORY MANAGEMENT BACK ON                 |
| 3834 | 037042 | 012601 |        | 4\$:   | MOV   | (SSP)+,R1     | :POP UPDATED PC PUSHED ON STACK                 |
| 3835 | 037044 | 105367 | 000153 |        | DECB  | LCNT+1        | :DECREMENT LCNT+1 TO 2                          |
| 3836 | 037050 | 022701 | 036460 |        | CMP   | #CSMPC,R1     | :SEE IF UPDATED PC WAS CORRECT                  |
| 3837 | 037054 | 001416 |        |        | BEQ   | 5\$           | :BRANCH AROUND ERROR IF OK                      |
| 3838 | 037056 | 005067 | 140510 |        | CLR   | MMRO          | :TURN OFF MEMORY MANAGEMENT                     |
| 3839 | 037062 | 012767 | 036460 | 142116 | MOV   | #CSMPC,\$TMP4 | :STORE EXPECTED FOR PRINTOUT                    |
| 3840 | 037070 | 010167 | 142070 |        | MOV   | R1,\$REG1     | :STORE R1 FOR PRINTOUT                          |
| 3841 | 037074 | 004767 | 000130 |        | JSR   | PC,POPSTK     | :GO POP STACK 2 TIMES                           |
| 3842 | 037100 | 104046 |        |        | ERROR | +46           | :WRONG PC                                       |
| 3843 | 037102 | 004767 | 000170 |        | JSR   | PC,PSHSTK     | :GO RESTORE STACK                               |
| 3844 | 037106 | 005267 | 140460 |        | INC   | MMRO          | :TURN MEMORY MANAGEMENT BACK ON                 |
| 3845 | 037112 | 012601 |        | 5\$:   | MOV   | (SSP)+,R1     | :MOVE TEMP PUSHED INTO R1                       |



```

3846 037114 032701 000017          BIT      #17,R1          ;SEE IF PSW <3:0> WERE CLEARED
3847 037120 001427          BEQ      CSMRET        ;BRANCH AROUND ERROR CALL IF OK
3848 037122 005067 140444          CLR      MMRO         ;TURN OFF MEMORY MANAGEMENT
3849 037126 010167 142032          MOV      R1,$REG1     ;STORE R1 FOR PRINTOUT
3850 037132 012667 000052          MOV      (SSP)+,TEMP+2 ;POP STACK ONCE
3851 037136 104047          ERROR   +47          ;BITS <3:0> SET IN PSW
3852 037140 016746 000044          MOV      TEMP+2,-(SSP) ;RESTORE STACK ONCE
3853 037144 000415          BR       CSMRET       ;BRANCH TO RETURN
3854 037146 005067 140420          CSMWSE: CLR      MMRO ;TURN OFF MEMORY MANAGEMENT
3855 037152 016701 000046          MOV      SRCALL,R1    ;MOVE RETURN ADDRESS TO R1
3856 037156 022121          CMP      (R1)+,(R1)+  ;INCREMENT R1 BY 4
3857 037160 012667 000024          1$:     MOV      (SSP)+,TEMP+2 ;POP STACK
3858 037164 026701 000020          CMP      TEMP+2,R1    ;SEE IF POPPED NUMBER IS RETURN ADDRESS
3859 037170 001373          BNE     1$           ;BRANCH FOR ANOTHER POP IF NOT
3860 037172 104050          ERROR   +50          ;WRONG SUPERVISOR SPACE ERROR
3861 037174 016746 000010          MOV      TEMP+2,-(SSP) ;RESTORE STACK
3862 037200 005067 140366          CSMRET: CLR      MMRO ;TURN OFF MEMORY MANAGEMENT
3863 037204 000207          RTS      PC          ;RETURN FROM THIS SUBROUTINE
3864
3865 037206          TEMP:   .BLKW      6   ;LOCATIONS TO STORE STACK CONTENTS
3866 037222 000000          LCNT:   .WORD      0   ;LOOP COUNTER STORAGE
3867 037224 000000          SRCALL: .WORD      0   ;LOCATION FOR SUBROUTINE CALL ADDRESS
3868 037226 000000          PNTR:   .WORD      0   ;POINTER
3869
3870 037230 116767 177767 177764          POPSTK: MOV      LCNT+1,LCNT ;MOVE UPPER BYTE START VALUE TO LOWER BYTE
3871 037236 012667 177744          MOV      (SSP)+,TEMP  ;SAVE RETURN PC OF THIS SUBROUTINE
3872 037242 012767 037210 177756          MOV      #TEMP+2,PNTR ;MOVE ADDRESS OF 1ST STORAGE LOCATION TO POINTER
3873 037250 012677 177752          PPLOOP: MOV      (SSP)+,@PNTR ;POP STACK INTO LOCATION PNTR IS POINTING TO
3874 037254 062767 000002 177744          ADD      #2,PNTR     ;ADD 2 TO THE POINTER
3875 037262 105367 177734          DECB    LCNT        ;DECREMENT THE LOOP COUNTER
3876 037266 001370          BNE     PPLOOP      ;BRANCH BACK FOR ANOTHER LOOP IF NOT ZERO
3877 037270 016746 177712          RETURN: MOV      TEMP,-(SSP) ;PUSH PC RETURN OF THIS SUBROUTINE BACK ON STACK
3878 037274 000207          RTS      PC          ;RETURN
3879
3880 037276 012667 177704          PSHSTK: MOV      (SSP)+,TEMP ;SAVE RETURN PC OF THIS SUBROUTINE
3881 037302 162767 000002 177716          PSLOOP: SUB      #2,PNTR  ;SUBTRACT 2 FROM THE POINTER
3882 037310 017746 177712          MOV      @PNTR,-(SSP) ;PUSH LOCATION PNTR IS POINTING TO ONTO STACK
3883 037314 022767 037210 177704          CMP      #TEMP+2,PNTR ;SEE IF LAST ITEM WAS PUSHED
3884 037322 001367          BNE     PSLOOP      ;BRANCH BACK FOR ANOTHER LOOP IF NOT
3885 037324 000761          BR      RETURN      ;BRANCH TO STACK RESTORE AND RETURN

```

3887

```

.SBTTL END OF PASS ROUTINE
:*****
:*INCREMENT THE PASS NUMBER ($PASS)
:*TYPE 'END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYYY'
:*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
:*IF SW12=1 INHIBIT TRACE TRAP
:*IF THERES A MONITOR GO TO IT
:*IF THERE ISN'T JUMP TO LOOP
$EOP:
037326          SCOPE
037326 000004   CLR      $STNM      ;;ZERO THE TEST NUMBER
037330 005067 141546  INC      $PASS      ;;INCREMENT THE PASS NUMBER
037334 005267 141672  BIC     #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
037340 042767 100000 141664  DEC     (PC)+      ;;LOOP?
037346 005327  $EOPCT: .WORD    1
037350 000001  BGT     $DOAGN     ;;YES
037352 003072  MOV     (PC)+,@(PC)+ ;;RESTORE COUNTER
037354 012737  $ENDCT: .WORD    1
037356 000001  $EOPCT
037360 037350  TYPE    ,65$      ;;TYPE ASCIZ STRING
037362 104401 037370  BR      64$      ;;GET OVER THE ASCIZ
037366 000407  ;;65$: .ASCIZ <12><15>/END PASS #/
64$:
037406          MOV     $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT
037406 016746 141620  ;;TYPE PASS NUMBER
037412 104405  TYPDS   ;;GO TYPE--DECIMAL ASCII WITH SIGN
037414 104401 037422  TYPE    ,67$      ;;TYPE ASCIZ STRING
037420 000421  BR      66$      ;;GET OVER THE ASCIZ
66$:
037464          .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
037464 016746 141422  MOV     $ERTTL,-(SP) ;;SAVE $ERTTL FOR TYPEOUT
037470          ;;TOTAL NUMBER OF ERRORS
037472 104405  TYPDS   ;;GO TYPE--DECIMAL ASCII WITH SIGN
037476 104401 001221  TYPE    ,SCLRF    ;;TYPE CARRIAGE RETURN, LINE FEED
037476 005067 141410  CLR     $ERTTL    ;;CLEAR ERROR TOTAL
037502 013700 000042  $GET42: MOV    @#42,R0 ;;GET MONITOR ADDRESS
037506 001414  BEQ     $DOAGN    ;;BRANCH IF NO MONITOR
037510 005046  CLR     -(SP)    ;;INSURE THE 'T' BIT IS CLEAR
037512 012746 037520  MOV     #$CLR.T,-(SP) ;;SETUP FOR AN RTI OR RTT
037516 000426  BR      $RTN     ;;GO DO AN RTI OR RTT TO LOAD THE PSW
037520          ;;WITH A CLEARED 'T' BIT
$CLR.T:
037520 013700 000042  MOV     @#42,R0  ;;INSURE R0 CONTAINS THE MONITORS
037524 001405  BEQ     $DOAGN   ;;RETURN ADDRESS
037526 000005  RESET  ;;CLEAR THE WORLD
037530 004710  $ENDAD: JSR    PC,(R0) ;;GO TO MONITOR
037532 000240  NOP    ;;SAVE ROOM
037534 000240  NOP    ;;FOR
037536 000240  NOP    ;;ACT11
037540          $DOAGN:
037540 104400  TRAP   ;;PUSH OLD PSW AND PC ON STACK
037542 042716 000020  BIC     #20,(SP)  ;;CLEAR THE 'T' BIT
037546 032777 010000 141364  BIT     #BIT12,@SWR ;;RUN WITH TRACE TRAP?
037554 001005  BNE    1$        ;;BR IF NO
037556 005167 141526  COM     $TBIT     ;;IS IT TIME FOR TRACE TRAP
037562 100402  BMI    1$        ;;BR IF NO
037564 052716 000020  BIS     #20,(SP)  ;;SET TRACE TRAP

```

3888

```

037570 012746 037576 1$: MOV #SLOOP,-(SP) ;;JUMP TO START OF TEST
037574 000002 $RTN: RTI ;;RETURN--THIS IS CHANGED TO
;;AN 'RTT' IF 'RTT' IS A LEGAL
;;INSTRUCTION

037576 $LOOP: JMP @PC+ ;;RETURN
037576 000137 $RTNAD: .WORD LOOP
037600 020456 $ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
037602 377 000 .EVEN

.SBTTL SCOPE HANDLER ROUTINE
*****
*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
*AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW14=1 LOOP ON TEST
*SW09=1 LOOP ON ERROR
*SW08=1 LOOP ON TEST IN SWR<7:0>
*CALL
* SCOPE ;;SCOPE=IOT
$SCOPE:
037606 104410 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
037610 032777 040000 141322 1$: BIT #BIT14,@SWR ;;LOOP ON PRESENT TEST?
037616 001062 BNE $OVER ;;YES IF SW14=1
;#####START OF CODE FOR THE XOR TESTER#####
037620 000416 $XTSTR: BR 6$ ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
;;THIS INSTRUCTION TO A 'NOP' (NOP 240)
037622 013746 000004 MOV @ERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
037626 012737 037646 000004 MOV #5,$@ERRVEC ;;SET FOR TIMEOUT
037634 005737 177060 TST @177060 ;;TIME OUT ON XOR?
037640 012637 000004 MOV (SP)+,@ERRVEC ;;RESTORE THE ERROR VECTOR
037644 000431 BR $$VLAD ;;GO TO THE NEXT TEST
037646 022626 5$: CMP (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
037650 012637 000004 MOV (SP)+,@ERRVEC ;;RESTORE THE ERROR VECTOR
037654 000417 BR 7$ ;;LOOP ON THE PRESENT TEST
037656 6$:;#####END OF CODE FOR THE XOR TESTER#####
037656 032777 000400 141254 BIT #BIT08,@SWR ;;LOOP ON SPEC. TEST?
037664 001404 BEQ 2$ ;;BR IF NO
037666 127767 141246 141206 CMPB @SWR,$STNM ;;ON THE RIGHT TEST? SWR<7:0>
037674 001433 BEQ $OVER ;;BR IF YES
037676 105767 141201 2$: TSTB $ERFLG ;;HAS AN ERROR OCCURRED?
037702 001412 BEQ $$VLAD ;;BR IF NO
037704 032777 001000 141226 BIT #BIT09,@SWR ;;LOOP ON ERROR?
037712 001404 BEQ 4$ ;;BR IF NO
037714 016767 141170 141164 7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
037722 000420 BR $OVER
037724 105067 141153 4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
037730 105267 141146 $$VLAD: INCB $STNM ;;COUNT TEST NUMBERS
037734 116767 141142 141266 MOVB $STNM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX
037742 011667 141140 MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
037746 011667 141136 MOV (SP),$LPERR ;;SAVE ERROR LOOP ADDRESS
037752 005067 141234 CLR $ESCAPE ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
037756 112767 000001 141131 MOVB #1,$ERMAX ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
037764 016777 141112 141150 $OVER: MOV $STNM,@DISPLAY ;;DISPLAY TEST NUMBER
037772 016716 141110 MOV $LPADR,(SP) ;;FUDGE RETURN ADDRESS
037776 000002 RTI ;;FIXES PS

.SBTTL ERROR HANDLER ROUTINE

```

3889

```

*****
*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
*AND GO TO ERRYP ON ERROR
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW15=1      HALT ON ERROR
*SW13=1      INHIBIT ERROR TYPEOUTS
*SW10=1      BELL ON ERROR
*SW09=1      LOOP ON ERROR
*CALL
*
ERROR N      ;;ERROR-EMT AND N=ERROR ITEM NUMBER
$ERROR:
040000
040000 104410
040002 010067 141154
040006 010167 141152
040012 010267 141150
040016 010367 141146
040022 010467 141144
040026 010567 141142
040032 116767 141044 141214
040040 105267 141037 7$:
040044 001775
040046 016777 141030 141066
040054 032777 002000 141056
040062 001402
040064 104401 001214
040070 005267 141016 1$:
040074 011667 141016
040100 162767 000002 141010
040106 117767 141004 141000
040114 032777 020000 141016
040122 001004
040124 004767 000106
040130 104401 001221
040134
040134 122767 000001 141102 20$:
040142 001007
040144 116767 140744 000004
040152 004767 002052
040156 000
040157 000 21$:
040160 000777 22$:
040162 005777 140752 2$:
040166 100002
040170 000000
040172 104410
040174 032777 001000 140736 3$:
040202 001402
040204 016716 140700
040210 005767 140776 4$:
040214 001402
040216 016716 140770
040222
040222 022737 037530 000042 5$:
040230 001001
040232 000000
040234 6$:
CKSWR
MOV R0,$REG0 ;;TEST FOR CHANGE IN SOFT-SWR
MOV R1,$REG1 ;;SAVE THE CONTENTS OF R0
MOV R2,$REG2 ;;SAVE THE CONTENTS OF R1
MOV R3,$REG3 ;;SAVE THE CONTENTS OF R2
MOV R4,$REG4 ;;SAVE THE CONTENTS OF R3
MOV R5,$REG5 ;;SAVE THE CONTENTS OF R4
MOVB $STNM,TESTNO ;;SAVE THE TEST NUMBER
INCB $ERFLG ;;SET THE ERROR FLAG
BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO
MOV $STNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
BIT #BIT10,@SWR ;;BELL ON ERROR?
BEQ 1$ ;;NO - SKIP
TYPE $BELL ;;RING BELL
INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
MOV (SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
SUB #2,$ERRPC
MOVB @ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
BIT #BIT13,@SWR ;;SKIP TYPEOUT IF SET
BNE 20$ ;;SKIP TYPEOUTS
JSR PC,ERRYP ;;GO TO USER ERROR ROUTINE
TYPE $CRLF
CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
BNE 2$ ;;NO,SKIP APT ERROR REPORT
MOVB $ITEMB,21$ ;;SET ITEM NUMBER AS ERROR NUMBER
JSR PC,$ATY4 ;;REPORT FATAL ERROR TO APT
.BYTE 0
.BYTE 0
BR 22$ ;;APT ERROR LOOP
TST @SWR ;;HALT ON ERROR
BPL 3$ ;;SKIP IF CONTINUE
HALT ;;HALT ON ERROR!
CKSWR
BIT #BIT09,@SWR ;;TEST FOR CHANGE IN SOFT-SWR
BEQ 4$ ;;BR IF NO
MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
BEQ 5$ ;;BR IF NONE
MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
CMP #SENDAD,@#42 ;;ACT-11 AUTO-ACCEPT?
BNE 6$ ;;BRANCH IF NO
HALT ;;YES

```

```

040234 000002          RTI          ;;RETURN
3890
3891 040236 104401 001221  ERRYP: TYPE ,SRLF          ;'CARRIAGE RETURN' & 'LINE FEED'
3892 040242 010046          MOV      R0,-(KSP)      ;SAVE R0.
3893 040244 005000          CLR      R0          ;PICKUP THE ITEM INDEX
3894 040246 153700 001114  BISB    @#SITEMB,R0
3895 040252 001004          BNE     1$          ;IF ITEM NUMBER IS ZERO, JUST
3896                                ;TYPE THE PC OF THE ERROR
3897 040254 016746 140636  MOV      $ERRPC,-(SP)  ;SAVE $ERRPC FOR TYPEOUT
3898                                ;ERROR ADDRESS
3899 040260 104402          TYPDC          ;GO TYPE--OCTAL ASCII(ALL DIGITS)
3900 040262 000522          BR      13$        ;GET OUT
3901 040264 005300 1$:    DEC      R0          ;ADJUST THE INDEX SO THAT IT WILL
3902 040266 006300          ASL     R0          ;WORK FOR THE ERROR TABLE.
3903 040270 006300          ASL     R0
3904 040272 006300          ASL     R0
3905 040274 062700 001320  ADD     #SERRTB,R0  ;FORM TABLE POINTER
3906 040300 012067 000004  MOV     (R0)+,2$   ;PICKUP 'ERROR MESSAGE' POINTER
3907 040304 001404          BEQ     3$          ;SKIP TYPEOUT IF NO POINTER
3908 040306 104401          TYPE          ;TYPE THE 'ERROR MESSAGE'
3909 040310 000000 2$:    .WORD   0          ;'ERROR MESSAGE' POINTER GOES HERE
3910 040312 104401 001221  TYPE    ,SRLF      ;'CARRIAGE RETURN' & 'LINE FEED'
3911 040316 012067 000004 3$:    MOV     (R0)+,4$   ;PICKUP 'DATA HEADER' POINTER
3912 040322 001404          BEQ     5$          ;SKIP TYPEOUT IF 0
3913 040324 104401          TYPE          ;TYPE THE 'DATA HEADER'
3914 040326 000000 4$:    .WORD   0          ;'DATA HEADER' POINTER GOES HERE
3915 040330 104401 001221  TYPE    ,SRLF      ;'CARRIAGE RETURN' & 'LINE FEED'
3916 040334 010146 5$:    MOV     R1,-(KSP)  ;SAVE R1
3917 040336 012001          MOV     (R0)+,R1   ;PICKUP 'DATA TABLE' POINTER
3918 040340 001472          BEQ     12$         ;BR IF NO DATA TO BE TYPED
3919 040342 012000          MOV     (R0)+,R0   ;PICKUP 'DATA FORMAT' POINTER
3920 040344 105710 6$:    TSTB   (R0)        ;IS IT FORMAT 0?
3921 040346 001003          BNE     7$          ;BR IF NO
3922                                ;*THIS CODE IS FOR OCTAL (16-BIT) FORMAT (DF=0)
3923 040350 013146          MOV     @R1+,-(SP) ;SAVE @R1)+ FOR TYPEOUT
3924 040352 104402          TYPDC          ;GO TYPE--OCTAL ASCII(ALL DIGITS)
3925 040354 000456          BR      11$
3926                                ;*THIS CODE IS FOR DECIMAL FORMAT (DF=1)
3927 040356 121027 000001 7$:    CMPB   (R0),#1     ;IS IT FORMAT 1?
3928 040362 001003          BNE     8$          ;BRANCH IF NO
3929 040364 013146          MOV     @R1+,-(SP) ;SAVE @R1)+ FOR TYPEOUT
3930 040366 104405          TYPDS          ;GO TYPE--DECIMAL ASCII WITH SIGN
3931 040370 000450          BR      11$
3932                                ;*THIS CODE IS FOR BINARY FORMAT (DF=2)
3933 040372 121027 000002 8$:    CMPB   (R0),#2     ;IS IT FORMAT 2?
3934 040376 001003          BNE     9$          ;BRANCH IF NO
3935 040400 013146          MOV     @R1+,-(SP) ;SAVE @R1)+ FOR TYPEOUT
3936 040402 104406          TYPBN          ;GO TYPE--BINARY ASCII
3937 040404 000442          BR      11$
3938                                ;*THIS CODE IS FOR OCTAL (22-BIT) FORMAT (DF=3)
3939 040406 121027 000003 9$:    CMPB   (R0),#3     ;IS IT FORMAT 3?
3940 040412 001011          BNE     15$         ;BRANCH IF NO
3941 040414 012146          MOV     (R1)+,-(KSP) ;PUT ADDRESS OF FIRST LOC. ON STACK
3942 040416 004767 002660  JSR     PC,$DB20    ;CONVERT TWO LOCS. TO AN ASCII STRING
3943 040422 062716 000003  ADD     #3,(KSP)    ;ONLY NEED 8 CHARACTERS NOT 11
3944 040426 012667 000002  MOV     (KSP)+,10$  ;PUT ADDRESS OF ASCII CHARS. AT 10$
3945 040432 104401          TYPE          ;TYPE OCTAL VALUE OF 22-BIT BINARY NO.

```

3946 040434 000000  
3947  
3948 040436 010246  
3949 040440 010346  
3950 040442 013103  
3951 040444 005002  
3952 040446 073227 000006  
3953 040452 010267 140530  
3954 040456 010367 140526  
3955 040462 012746 001206  
3956 040466 004767 002610  
3957 040472 062716 000003  
3958 040476 012667 000002  
3959 040502 104401  
3960 040504 000000  
3961 040506 012603  
3962 040510 012602  
3963 040512 005711  
3964 040514 001404  
3965 040516 104401 040540  
3966 040522 105720  
3967 040524 000707  
3968 040526 012601  
3969 040530 012600  
3970 040532 104401 001221  
3971 040536 000207  
3972 040540 040 040  
3973 040543 000  
3974

```
10$: .WORD 0  
: *THIS CODE IS FOR OCTAL (22-BIT) FORMAT FOR A PAR LEFT SHIFTED 6 (DF 4)  
15$: MOV R2, -(KSP) ;SAVE R2 ON STACK  
MOV R3, -(KSP) ;SAVE R3 ON STACK  
MOV @ (R1)+, R3 ;LOAD DATA WORD INTO R3  
CLR R2 ;R2 HOLDS UPPER SIX BITS OF NUMBER  
ASHC #6, R2 ;SHIFT VALUE LEFT 6 TIMES  
MOV R2, $TMP4 ;HOLDS LOWER 16 BITS OF ADDRESS  
MOV R3, $TMP5 ;HOLDS UPPER 6 BITS OF ADDRESS  
MOV # $TMP4, -(KSP) ;PUT ADDRESS OF LOWER BITS ONTO STACK  
JSR PC, $DB20 ;CONVERT TWO LOCS. TO AN ASCII STRING  
ADD #3, (KSP) ;ONLY NEED 8 CHARACTERS NOT 11  
MOV (KSP)+, 16$ ;PUT ADDRESS OF ASCII CHARS. AT 16$  
TYPE ;TYPE OCTAL VALUE OF 22-BIT BINARY NO.  
16$: .WORD 0  
MOV (KSP)+, R3 ;RESTORE R3  
MOV (KSP)+, R2 ;RESTORE R2  
11$: TST (R1) ;IS THERE ANOTHER NUMBER?  
BEQ 12$ ;BR IF NO  
TYPE ,14$ ;TYPE TWO(2) SPACES  
TSTB (R0)+ ;POINT TO NEW 'DATA FORMAT'  
BR 6$ ;LOOP  
12$: MOV (KSP)+, R1 ;RESTORE R1  
13$: MOV (KSP)+, R0 ;RESTORE R0  
TYPE , $CRLF ;'CARRIAGE RETURN' & 'LINE FEED'  
RTS PC ;RETURN  
14$: .ASCIZ / / ;TWO(2) SPACES  
.BYTE 0
```

```
.SBTTL TTY INPUT ROUTINE  
: *****  
.ENABL LSB  
: *****  
: *SOFTWARE SWITCH REGISTER CHANGE ROUTINE.  
: *ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL  
: *SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL  
: *WHEN OPERATING IN TTY FLAG MODE.
```

040544 022767 000176 140366  
040552 001114  
040554 105777 140364  
040560 100111  
040562 117746 140360  
040566 042716 177600  
040572 022726 000007  
040576 001102  
040600 126727 140330 000001  
040606 001476  
040610 104401 041501  
040614 104401 041506  
040620 016746 137352  
040624 104402  
040626 104401 041517  
040632 005046  
040634 005046  
040636 105777 140302  
040642 100375  
040644 117746 140276  
040650 042716 177600

```
$CKSWR: CMP #SWREG, SWR ;IS THE SOFT-SWR SELECTED?  
BNE 15$ ;BRANCH IF NO  
TSTB @ $TKS ;CHAR THERE?  
BPL 15$ ;IF NO, DON'T WAIT AROUND  
MOVB @ $TKB, -(SP) ;SAVE THE CHAR  
BIC #^C177, (SP) ;STRIP-OFF THE ASCII  
CMP #7, (SP)+ ;IS IT A CONTROL G?  
BNE 15$ ;NO, RETURN TO USER  
CMPB $AUTOB, #1 ;ARE WE RUNNING IN AUTO-MODE?  
BEQ 15$ ;BRANCH IF YES  
TYPE , $CNTLG ;ECHO THE CONTROL-G (^G)  
$GTSWR: TYPE , $MSWR ;TYPE CURRENT CONTENTS  
MOV SWREG, -(SP) ;SAVE SWREG FOR TYPEOUT  
TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)  
TYPE , $MNEW ;PROMPT FOR NEW SWR  
19$: CLR -(SP) ;CLEAR COUNTER  
CLR -(SP) ;THE NEW SWR  
7$: TSTB @ $TKS ;CHAR THERE?  
BPL 7$ ;IF NOT TRY AGAIN  
MOVB @ $TKB, -(SP) ;PICK UP CHAR  
BIC #^C177, (SP) ;MAKE IT 7-BIT ASCII
```

```

040654 021627 000003      CMP      (SP),#3      :: IS IT A CONTROL-C?
040660 001015             BNE      9$          :: BRANCH IF NOT
040662 104401 001312      TYPE    ,%CNTLC     :: YES, ECHO CONTROL-C (^C)
040666 062706 000006      ADD     #6,SP       :: CLEAN UP STACK
040672 126727 140237 000001  CMPB    $INTAG,#1   :: REENABLE TTY KEYBOARD INTERRUPTS?
040700 001003             BNE      8$          :: BRANCH IF NO
040702 012777 000100 140234  MOV     #100,@$TKS  :: ALLOW TTY KEYBOARD INTERRUPTS
040710 000167 000614      JMP     CNTRLC      :: CONTROL-C RESTART
040714 021627 000025 8$:      CMP     (SP),#25   :: IS IT A CONTROL-U?
040720 001005             BNE      10$         :: BRANCH IF NOT
040722 104401 041474      TYPE    ,%CNTLU     :: YES, ECHO CONTROL-U (^U)
040726 062706 000006 20$:     ADD     #6,SP       :: IGNORE PREVIOUS INPUT
040732 000737             BR      19$         :: LET'S TRY IT AGAIN
040734 021627 000015 10$:     CMP     (SP),#15   :: IS IT A <CR>?
040740 001022             BNE      16$         :: BRANCH IF NO
040742 005766 000004      TST     4(SP)       :: YES, IS IT THE FIRST CHAR?
040746 001403             BEQ     11$         :: BRANCH IF YES
040750 016677 000002 140162  MOV     2(SP),@$SWR  :: SAVE NEW SWR
040756 062706 000006 11$:     ADD     #6,SP       :: CLEAR UP STACK
040762 104401 001221 14$:     TYPE    ,%CRLF     :: ECHO <CR> AND <LF>
040766 126727 140143 000001  CMPB    $INTAG,#1   :: RE-ENABLE TTY KBD INTERRUPTS?
040774 001003             BNE      15$         :: BRANCH IF NOT
040776 012777 000100 140140  MOV     #100,@$TKS  :: RE-ENABLE TTY KBD INTERRUPTS
041004 000002             RTI                    :: RETURN
041006 004767 001060 15$:     JSR     PC,$TYPEC   :: ECHO CHAR
041012 021627 000060 16$:     CMP     (SP),#60   :: CHAR < 0?
041016 002420             BLT     18$         :: BRANCH IF YES
041020 021627 000067      CMP     (SP),#67   :: CHAR > 7?
041024 003015             BGT     18$         :: BRANCH IF YES
041026 042726 000060      BIC     #60,(SP)+  :: STRIP-OFF ASCII
041032 005766 000002      TST     2(SP)       :: IS THIS THE FIRST CHAR
041036 001403             BEQ     17$         :: BRANCH IF YES
041040 006316             ASL     (SP)         :: NO, SHIFT PRESENT
041042 006316             ASL     (SP)         :: CHAR OVER TO MAKE
041044 006316             ASL     (SP)         :: ROOM FOR NEW ONE.
041046 005266 000002 17$:     INC     2(SP)       :: KEEP COUNT OF CHAR
041052 056616 177776      BIS     -2(SP), (SP) :: SET IN NEW CHAR
041056 000667             BR      7$          :: GET THE NEXT ONE
041060 104401 001220 18$:     TYPE    ,%QUES     :: TYPE ?<CR><LF>
041064 000720             BR      20$         :: SIMULATE CONTROL-U
.DSABL  LSB
*****
*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
*CALL:
*      RDCHR          :: INPUT A SINGLE CHARACTER FROM THE TTY
*      RETURN HERE    :: CHARACTER IS ON THE STACK
*                    :: WITH PARITY BIT STRIPPED OFF
$RDCHR: MOV     (SP),-(SP)  :: PUSH DOWN THE PC
041070 016666 000004 000002  MOV     4(SP),2(SP)  :: SAVE THE PS
041076 105777 140042 1$:      TSTB   @$TKS       :: WAIT FOR
041102 100375             BPL     1$          :: A CHARACTER
041104 117766 140036 000004  MOVB    @$TKB,4(SP)  :: READ THE TTY
041112 042766 177600 000004  BIC     #^C<177>,4(SP) :: GET RID OF JUNK IF ANY
041120 026627 000004 000023  CMP     4(SP),#23   :: IS IT A CONTROL-S?
041126 001013             BNE      3$         :: BRANCH IF NO
041130 105777 140010 2$:      TSTB   @$TKS       :: WAIT FOR A CHARACTER

```

```

041134 100375          BFL      2$          ;; LOOP UNTIL ITS THERE
041136 117746 140004  MOVB    @STKB,-(SP)    ;; GET CHARACTER
041142 042716 177600  BIC    #^C177,(SP)    ;; MAKE IT 7-BIT ASCII
041146 022627 000021  CMP    (SP)+,#21      ;; IS IT A CONTROL-Q?
041152 001366          BNE    2$             ;; IF NOT DISCARD IT
041154 000750          BR     1$             ;; YES, RESUME
041156 026627 000004 000140 3$:    CMP    4(SP),#140    ;; IS IT UPPER CASE?
041164 002407          BLT    4$             ;; BRANCH IF YES
041166 026627 000004 000175  CMP    4(SP),#175    ;; IS IT A SPECIAL CHAR?
041174 003003          BGT    4$             ;; BRANCH IF YES
041176 042766 000040 000004  BIC    #40,4(SP)     ;; MAKE IT UPPER CASE
041204 000002          RTI           ;; GO BACK TO USER
*****
*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
*CALL:
*      RDLIN          ;; INPUT A STRING FROM THE TTY
*      RETURN HERE   ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
*                      ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
$RDLIN: MOV    R3,-(SP)    ;; SAVE R3
        CLR    -(SP)     ;; CLEAR THE RUBOUT KEY
1$:    MOV    #TTYIN,R3  ;; GET ADDRESS
2$:    CMP    #TTYIN+8.,R3 ;; BUFFER FULL?
        BLOS  4$         ;; BR IF YES
        RDCHR          ;; GO READ ONE CHARACTER FROM THE TTY
        MOVB   (SP)+,(R3) ;; GET CHARACTER
        CMPB  #3,(R3)    ;; IS IT A CONTROL-C?
        BNE  10$        ;; BRANCH IF NO
        TYPE  ,SCNTLC    ;; TYPE A CONTROL-C (^C)
        TST  (SP)+      ;; CLEAN RUBOUT KEY OFF OF THE STACK
        MOV  (SP)+,R3    ;; RESTORE R3
        JMP  CNTRLC     ;; GOTO CONTROL-C RESTART
10$:   CMPB  #177,(R3)   ;; IS IT A RUBOUT
        BNE  5$         ;; BR IF NO
        TST  (SP)       ;; IS THIS THE FIRST RUBOUT?
        BNE  6$         ;; BR IF NO
        MOVB #'\,9$     ;; TYPE A BACK SLASH
        TYPE ,9$
        MOV  #-1,(SP)   ;; SET THE RUBOUT KEY
6$:   DEC  R3           ;; BACKUP BY ONE
        CMP  R3,#TTYIN  ;; STACK EMPTY?
        BLO  4$         ;; BR IF YES
        MOVB (R3),9$    ;; SETUP TO TYPEOUT THE DELETED CHAR.
        TYPE ,9$
        BR   2$         ;; GO TYPE
5$:   TST  (SP)       ;; GO READ ANOTHER CHAR.
        BEQ  7$         ;; RUBOUT KEY SET?
        MOVB #'\,9$     ;; BR IF NO
        TYPE ,9$       ;; TYPE A BACK SLASH
        CLR  (SP)      ;; CLEAR THE RUBOUT KEY
7$:   CMPB  #25,(R3)    ;; IS CHARACTER A CTRL U?
        BNE  8$         ;; BR IF NO
        TYPE ,SCNTLU    ;; TYPE A CONTROL 'U'
        BR   1$         ;; GO START OVER
8$:   CMPB  #22,(R3)    ;; IS CHARACTER A '^R'?
        BNE  3$         ;; BRANCH IF NO
        CLRB (R3)       ;; CLEAR THE CHARACTER
        TYPE ,SCRLF     ;; TYPE A 'CR' & 'LF'
  
```



```

041374 104401 041464          TYPE      , $TTYIN      ;; TYPE THE INPUT STRING
041400 000706          BR          2$          ;; GO PICKUP ANOTHER CHACTER
041402 104401 001220      4$:      TYPE      , $QUES      ;; TYPE A '?'
041406 000701          BR          1$          ;; CLEAR THE BUFFER AND LOOP
041410 111367 000046      3$:      MOVVB    (R3), 9$      ;; ECHO THE CHARACTER
041414 104401 041462          TYPE      , 9$
041420 122723 000015      CMPB    #15, (R3)+    ;; CHECK FOR RETURN
041424 001274          BNE          2$          ;; LOOP IF NOT RETURN
041426 105063 177777      CLRB    -1(R3)      ;; CLEAR RETURN (THE 15)
041432 104401 001222          TYPE      , $LF        ;; TYPE A LINE FEED
041436 005726          TST      (SP)+      ;; CLEAN RUBOUT KEY FROM THE STACK
041440 012603          MOV      (SP)+, R3      ;; RESTORE R3
041442 011646          MOV      (SP), -(SP)    ;; ADJUST THE STACK AND PUT ADDRESS OF THE
041444 016666 000004 000002      MOV      4(SP), 2(SP)    ;; FIRST ASCII CHARACTER ON IT
041452 012766 041464 000004      MOV      # $TTYIN, 4(SP)
041460 000002          RTI
041462 000          9$:      .BYTE    0          ;; RETURN
041463 000          .BYTE    0          ;; STORAGE FOR ASCII CHAR. TO TYPE
041464          $TTYIN: .BLKB    8.          ;; TERMINATOR
041474 136 125 015      $CNTLU: .ASCIZ  / ^U/<15><12>    ;; RESERVE 8 BYTES FOR TTY INPUT
041477 012 000          ;; CONTROL 'U'
041501 136 107 015      $CNTLG: .ASCIZ  / ^G/<15><12>    ;; CONTROL 'G'
041504 012 000
041506 015 012 123      $MSWR: .ASCIZ  <15><12>/SWR - /
041511 127 122 040
041514 075 040 000
041517 040 040 116      $MNEW: .ASCIZ  / NEW - /
041522 105 127 040
041525 075 040 000

3975
3976          .SBTTL  CONTROL-C SERVICING ROUTINE
3977
3978 041530 016767 137476 137452 CNTRLC: MOV      $PASS, $TMP5    ;; GET THE VALUE OF '$PASS'
3979 041536 005267 137446          INC      $TMP5      ;; FORM CURRENT PASS #
3980 041542 104401 041607          TYPE      , $MSG      ;; TYPE THE TEST STOPS HERE
3981 041546 116767 137330 000026      MOVVB    $TSTNM, 1$    ;; SAVE TEST NUMBER
3982 041554 016746 000022          MOV      1$, -(SP)    ;; SAVE 1$ FO TYPEOUT
3983 041560 104402          TYPDC
3984 041562 104401 041604          TYPE      , 2$
3985 041566 016746 137416          MOV      $TMP5, -(SP)    ;; SAVE $TMP5 FOR TYPEOUT
3986 041572 104405          TYPDS      ;; TYPE ASCII DECIMAL WITH SIGN
3987 041574 104407          GTSWR      ;; ASK FOR NEW SWR VALUE
3988 041576 000167 175526          JMP      $EOP+2        ;; JUMP TO END OF PASS + 2
3989 041602 000000          1$:      .WORD    0          ;; TEST # BUFFER
3990 041604 040 040 000      2$:      .ASCIZ  / /          ;; 2 SPACES & STOP MESSAGE
3991 041607 112 125 115      MSG:      .ASCII  /JUMPING TO END OF PASS/<15><12>
041612 120 111 116
041615 107 040 124
041620 117 040 105
041623 116 104 040
041626 117 106 040
041631 120 101 123
041634 123 015 012
3992 041637 124 105 123          .ASCIZ  /TESTNO PASSNU/<15><12>
041642 124 116 117
041645 011 120 101
041650 123 123 116
  
```

|        |     |     |     |
|--------|-----|-----|-----|
| 041653 | 117 | 015 | 012 |
| 041656 | 000 |     |     |

3994 041657 000  
3995

```

        .BYTE 0
        .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
        *
        041660 105767 137273 $TYPE: TSTB $TPFLG      ;;IS THERE A TERMINAL?
        041664 100002      BPL 1$      ;;BR IF YES
        041666 000000      HALT      ;;HALT HERE IF NO TERMINAL
        041670 000430      BR 3$      ;;LEAVE
        041672 010046      1$: MOV RO,-(SP)      ;;SAVE RO
        041674 017600 000002 MOV @2(SP),RO      ;;GET ADDRESS OF ASCIZ STRING
        041700 122767 000001 137336 CMPB #APTENV,$ENV      ;;RUNNING IN APT MODE
        041706 001011      BNE 62$      ;;NO,GO CHECK FOR APT CONSOLE
        041710 132767 000100 137327 BITB #APTSPOOL,$ENVM      ;;SPOOL MESSAGE TO APT
        041716 001405      BEQ 62$      ;;NO,GO CHECK FOR CONSOLE
        041720 010067 000004      MOV RO,61$      ;;SETUP MESSAGE ADDRESS FOR APT
        041724 004767 000270      JSR PC,$ATY3      ;;SPOOL MESSAGE TO APT
        041730 000000      61$: .WORD 0      ;;MESSAGE ADDRESS
        041732 132767 000040 137305 62$: BITB #APTCSUP,$ENVM      ;;APT CONSOLE SUPPRESSED
        041740 001003      BNE 60$      ;;YES,SKIP TYPE OUT
        041742 112046      2$: MOVB (RO)+,-(SP)      ;;PUSH CHARACTER TO BE TYPED ONTO STACK
        041744 001005      BNE 4$      ;;BR IF IT ISN'T THE TERMINATOR
        041746 005726      TST (SP)+      ;;IF TERMINATOR POP IT OFF THE STACK
        041750 012600      60$: MOV (SP)+,RO      ;;RESTORE RO
        041752 062716 000002      3$: ADD #2,(SP)      ;;ADJUST RETURN PC
        041756 000002      RTI      ;;RETURN
        041760 122716 000011      4$: CMPB #HT,(SP)      ;;BRANCH IF <HT>
        041764 001430      BEQ 8$
        041766 122716 000200      CMPB #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
        041772 001006      BNE 5$
        041774 005726      TST (SP)+      ;;POP <CR><LF> EQUIV
        041776 104401      TYPE      ;;TYPE A CR AND LF
        042000 001221      $CRLF
        042002 105067 000200      CLRB $CHARCNT      ;;CLEAR CHARACTER COUNT
        042006 000755      BR 2$      ;;GET NEXT CHARACTER
        042010 004767 000056      5$: JSR PC,$TYPEC      ;;GO TYPE THIS CHARACTER
        042014 126726 137136      6$: CMPB $FILLC,(SP)+      ;;IS IT TIME FOR FILLER CHARS.?
        042020 001350      BNE 2$      ;;IF NO GO GET NEXT CHAR.
        042022 016746 137126      MOV $NULL,-(SP)      ;;GET # OF FILLER CHARS. NEEDED
        ;;AND THE NULL CHAR.
        042026 105366 000001      7$: DECB 1(SP)      ;;DOES A NULL NEED TO BE TYPED?
        042032 002770      BLT 6$      ;;BR IF NO--GO POP THE NULL OFF OF STACK
        042034 004767 000032      JSR PC,$TYPEC      ;;GO TYPE A NULL
        042040 105367 000142      DECB $CHARCNT      ;;DO NOT COUNT AS A COUNT
        042044 000770      BR 7$      ;;LOOP
        ;HORIZONTAL TAB PROCESSOR
    
```

```

042046 112716 000040      8$:  MOVB  #' (SP)      ;; REPLACE TAB WITH SPACE
042052 004767 000014      9$:  JSR  PC,$TYPEC    ;; TYPE A SPACE
042056 132767 000007 000122 BITB  #7,$CHARCNT  ;; BRANCH IF NOT AT
042064 001372           BNE  9$           ;; TAB STOP
042066 005726           TST  (SP)+        ;; POP SPACE OFF STACK
042070 000724           BR   2$           ;; GET NEXT CHARACTER
042072 105777 137052      $TYPEC: TSTB @STPS    ;; WAIT UNTIL PRINTER IS READY
042076 100375           BPL  $TYPEC
042100 116677 000002 137044 MOVB  2(SP),@STPB  ;; LOAD CHAR TO BE TYPED INTO DATA REG.
042106 105777 137032      TSTB @STKS        ;; SEE IF KEYBOARD IS TALKING.
042112 100021           BPL  2$           ;; BRANCH IF IT ISN'T.
042114 017746 137026      MOV  @STKB,-(SP)   ;; PUSH CHARACTER ONTO STACK.
042120 042716 177600      BIC  #177600,(SP) ;; BIT CLEAR TOP BYTE AND PARITY BIT.
042124 022726 000023      CMP  #23,(SP)+    ;; SEE IF THIS IS A ^S.
042130 001012           BNE  2$           ;; BRANCH TO CONTINUE IF IT ISN'T.
042132 105777 137006      3$:  TSTB @STKS        ;; WAIT FOR ANOTHER INPUT.
042136 100375           BPL  3$           ;; BRANCH BACK IF NOT READY.
042140 017746 137002      MOV  @STKB,-(SP)   ;; PUSH NEXT CHARACTER ON STACK.
042144 042716 177600      BIC  #177600,(SP) ;; BIT CLEAR TOP BYTE AND PARITY BIT.
042150 022726 000021      CMP  #21,(SP)+    ;; SEE IF THIS IS A ^Q.
042154 001366           BNE  3$           ;; BRANCH BACK FOR MORE WAIT IF NOT.
042156 122766 000015 000002 2$:  CMPB #CR,2(SP)    ;; IS CHARACTER A CARRIAGE RETURN?
042164 001003           BNE  1$           ;; BRANCH IF NO
042166 105067 000014      CLRB $CHARCNT    ;; YES--CLEAR CHARACTER COUNT
042172 000406           BR   $TYPEX      ;; EXIT
042174 122766 000012 000002 1$:  CMPB #LF,2(SP)    ;; IS CHARACTER A LINE FEED?
042202 001402           BEQ  $TYPEX      ;; BRANCH IF YES
042204 105227           INCB (PC)+       ;; COUNT THE CHARACTER
042206 000000      $CHARCNT: .WORD 0 ;; CHARACTER COUNT STORAGE
042210 000207      $TYPEX: RTS  PC
  
```

3996

.SBTTL APT COMMUNICATIONS ROUTINE  
 \*\*\*\*\*

```

042212 112767 000001 000236 $ATY1: MOVB  #1,$FFLG  ;; TO REPORT FATAL ERROR
042220 112767 000001 000226 $ATY3: MOVB  #1,$MFLG  ;; TO TYPE A MESSAGE
042226 000403           BR   $ATYC
042230 112767 000001 000220 $ATY4: MOVB  #1,$FFLG  ;; TO ONLY REPORT FATAL ERROR
042236           $ATYC:
042236 010046           MOV  R0,-(SP)      ;; PUSH R0 ON STACK
042240 010146           MOV  R1,-(SP)      ;; PUSH R1 ON STACK
042242 105767 000206      TSTB $MFLG        ;; SHOULD TYPE A MESSAGE?
042246 001450           BEQ  5$           ;; IF NOT: BR
042250 122767 000001 136766      (MPB #APTENV,$ENV  ;; OPERATING UNDER APT?
042256 001031           BNE  3$           ;; IF NOT: BR
042260 132767 000100 136757      BITB #APTPOOL,$ENVM ;; SHOULD SPOOL MESSAGES?
042266 001425           BEQ  3$           ;; IF NOT: BR
042270 017600 000004           MOV  @4(SP),R0    ;; GET MESSAGE ADDR.
042274 062766 000002 000004      ADD  #2,4(SP)     ;; BUMP RETURN ADDR.
042302 005767 136716      1$:  TST  $MSGTYPE    ;; SEE IF DONE W/ LAST XMISSION?
042306 001375           BNE  1$           ;; IF NOT: WAIT
042310 010067 136724      MOV  R0,$MSGAD    ;; PUT ADDR IN MAILBOX
042314 105720      2$:  TSTB (R0)+       ;; FIND END OF MESSAGE
042316 001376           BNE  2$
042320 166700 136714      SUB  $MSGAD,R0    ;; SUB START OF MESSAGE
042324 006200           ASR  R0           ;; GET MESSAGE LNTH IN WORDS
042326 010067 136710      MOV  R0,$MSGGLT  ;; PUT LENGTH IN MAILBOX
042332 012767 000004 136664      MOV  #4,$MSGTYPE ;; TELL APT TC TAKE MSG.
042340 000413           BR   5$
  
```

```

042342 017667 000004 000016 3$:  MOV @4(SP),4$      ;;PUT MSG ADDR IN JSR LINKAGE
042350 062766 000002 000004      ADD #2,4(SP)      ;;BUMP RETURN ADDRESS
042356 016746 135414      MOV 177776,-(SP)  ;;PUSH 177776 ON STACK
042362 004767 177272      JSR PC,$TYPE     ;;CALL TYPE MACRO
042366 000000      4$:  .WORD 0
042370      5$:
042370 105767 000062 10$:  TSTB $FFLG      ;;SHOULD REPORT FATAL ERROR?
042374 001416      BEQ 12$          ;;IF NOT: BR
042376 005767 136642      TST $ENV        ;;RUNNING UNDER APT?
042402 001413      BEQ 12$          ;;IF NOT: BR
042404 005767 136614 11$:  TST $MSGTYPE    ;;FINISHED LAST MESSAGE?
042410 001375      BNE 11$         ;;IF NOT: WAIT
042412 017667 000004 136606  MOV @4(SP),$FATAL ;;GET ERROR #
042420 062766 000002 000004  ADD #2,4(SP)      ;;BUMP RETURN ADDR.
042426 005267 136572      INC $MSGTYPE     ;;TELL APT TO TAKE ERROR
042432 105067 000020 12$:  CLRB $FFLG      ;;CLEAR FATAL FLAG
042436 105067 000013      CLRB $LFLG      ;;CLEAR LOG FLAG
042442 105067 000006      CLRB $MFLG      ;;CLEAR MESSAGE FLAG
042446 012601      MOV (SP)+,R1    ;;POP STACK INTO R1
042450 012600      MOV (SP)+,R0    ;;POP STACK INTO R0
042452 000207      RTS PC         ;;RETURN
042454 000      $MFLG: .BYTE 0  ;;MESSG. FLAG
042455 000      $LFLG: .BYTE 0  ;;LOG FLAG
042456 000      $FFLG: .BYTE 0  ;;FATAL FLAG

```

```

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

```

3997

```

.SBTTL BINARY TO ASCII AND TYPE ROUTINE
*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
*BINARY-ASCII NUMBER AND TYPE IT.
*CALL:

```

```

*      MOV NUMBER,-(SP)  ;;NUMBER TO BE TYPED
*      TYPBN            ;;TYPE IT
$TYPBN: MOV R1,-(SP)     ;;SAVE R1 ON THE STACK
        MOV 6(SP),R1    ;;GET THE INPUT NUMBER
        SEC             ;;SET 'C' SO CAN KEEP TRACK OF THE NUMBER OF BITS
042460 010146 000006 1$:  MOV #0,$BIN        ;;SET CHARACTER TO AN ASCII '0'.
042462 016601      ROL R1          ;;GET THIS BIT
042466 000261      BEQ 2$          ;;DONE?
042470 112767 000060 000034  ADCB $BIN         ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
042476 006101      TYPE , $BIN    ;;GO TYPE THIS BIT
042500 001406      CLC          ;;CLEAR 'C' SO CAN KEEP TRACK OF BITS
042502 105567 000024      BR 1$          ;;GO DO THE NEXT BIT
042506 104401 042532 2$:  MOV (SP)+,R1    ;;POP THE STACK INTO R1
042512 000241      MOV 2(SP),4(SP)  ;;ADJUST THE STACK
042514 000765      MOV (SP)+,(SP)
042516 012601      RTI          ;;RETURN TO USER
042520 016666 000002 000004  $BIN: .BYTE 0,0  ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
042526 012616
042530 000002
042532 000      000

```

3998

```

.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
    042534 017646 000000 $TYPOS: MOV @ (SP),-(SP) ;;PICKUP THE MODE
    042540 116667 000001 000211 MOVB 1(SP), $OFILL ;;LOAD ZERO FILL SWITCH
    042546 112667 000207 MOVB (SP)+, $OMODE+1 ;;NUMBER OF DIGITS TO TYPE
    042552 062716 000002 ADD #2, (SP) ;;ADJUST RETURN ADDRESS
    042556 000406 BR $TYPON
    042560 112767 000001 000171 $TYPOC: MOVB #1, $OFILL ;;SET THE ZERO FILL SWITCH
    042566 112767 000006 000165 MOVB #6, $OMODE+1 ;;SET FOR SIX(6) DIGITS
    042574 112767 000005 000154 $TYPON: MOVB #5, $OCNT ;;SET THE ITERATION COUNT
    042602 010346 MOV .3, -(SP) ;;SAVE R3
    042604 010446 MOV R4, -(SP) ;;SAVE R4
    042606 010546 MOV R5, -(SP) ;;SAVE R5
    042610 116704 000145 MOVB $OMODE+1, R4 ;;GET THE NUMBER OF DIGITS TO TYPE
    042614 005404 NEG R4
    042616 062704 000006 ADD #6, R4 ;;SUBTRACT IT FOR MAX. ALLOWED
    042622 110467 000132 MOVB R4, $OMODE ;;SAVE IT FOR USE
    042626 116704 000125 MOVB $OFILL, R4 ;;GET THE ZERO FILL SWITCH
    042632 016605 000012 MOV 12(SP), R5 ;;PICKUP THE INPUT NUMBER
    042636 005003 CLR R3 ;;CLEAR THE OUTPUT WORD
    042640 006105 1$: ROL R5 ;;ROTATE MSB INTO 'C'
    042642 000404 BR 3$ ;;GO DO MSB
    042644 006105 2$: ROL R5 ;;FORM THIS DIGIT
    042646 006105 ROL R5
    042650 006105 ROL R5
    042652 010503 MOV R5, R3
    042654 006103 3$: ROL R3 ;;GET LSB OF THIS DIGIT
    042656 105367 000076 DECB $OMODE ;;TYPE THIS DIGIT?
    042662 100016 BPL 7$ ;;BR IF NO
    042664 042703 177770 BIC #177770, R3 ;;GET RID OF JUNK
    042670 001002 BNE 4$ ;;TEST FOR 0
    042672 005704 TST R4 ;;SUPPRESS THIS 0?
    042674 001403 BEQ 5$ ;;BR IF YES
    042676 005204 4$: INC R4 ;;DON'T SUPPRESS ANYMORE 0'S
    042700 052703 000060 BIS #'0, R3 ;;MAKE THIS DIGIT ASCII
    042704 052703 000040 5$: BIS #' , R3 ;;MAKE ASCII IF NOT ALREADY
    042710 110367 000040 MOVB R3, 8$ ;;SAVE FOR TYPING
    042714 104401 042754 TYPE , 8$ ;;GO TYPE THIS DIGIT
    042720 105367 000032 7$: DECB $OCNT ;;COUNT BY 1
    042724 003347 BGT 2$ ;;BR IF MORE TO DO
    042726 002402 BLT 6$ ;;BR IF DONE
    042730 005204 INC R4 ;;INSURE LAST DIGIT ISN'T A BLANK
    
```

|        |        |               |       |             |                               |
|--------|--------|---------------|-------|-------------|-------------------------------|
| 042732 | 000744 |               | BR    | 2\$         | ::GO DO THE LAST DIGIT        |
| 042734 | 012605 |               | MOV   | (SP)+,R5    | ::RESTORE R5                  |
| 042736 | 012604 |               | MOV   | (SP)+,R4    | ::RESTORE R4                  |
| 042740 | 012603 |               | MOV   | (SP)+,R3    | ::RESTORE R3                  |
| 042742 | 016666 | 000002 000004 | MOV   | 2(SP),4(SP) | ::SET THE STACK FOR RETURNING |
| 042750 | 012616 |               | MOV   | (SP)+,(SP)  |                               |
| 042752 | 000002 |               | RTI   |             | ::RETURN                      |
| 042754 | 000    | 8\$:          | .BYTE | 0           | ::STORAGE FOR ASCII DIGIT     |
| 042755 | 000    |               | .BYTE | 0           | ::TERMINATOR FOR TYPE ROUTINE |
| 042756 | 000    | \$OCNT:       | .BYTE | 0           | ::OCTAL DIGIT COUNTER         |
| 042757 | 000    | \$OFILL:      | .BYTE | 0           | ::ZERO FILL SWITCH            |
| 042760 | 000000 | \$OMODE:      | .WORD | 0           | ::NUMBER OF DIGITS TO TYPE    |

4000

```

.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.
      CLR      R0           ;;ZERO THE CONSTANTS INDEX
      MOV      #SDBLK,R3    ;;SETUP THE OUTPUT POINTER
      MOVB    #' ,(R3)+     ;;SET THE FIRST CHARACTER TO A BLANK
      CLR      R2           ;;CLEAR THE BCD NUMBER
      MOV      $DTBL(R0),R1 ;;GET THE CONSTANT
      SUB     R1,R5         ;;FORM THIS BCD DIGIT
      BLT     4$           ;;BR IF DONE
      INC     R2           ;;INCREASE THE BCD DIGIT BY 1
      BR      3$
      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
      ASLB   (SP)         ;;MSD?
      BCC     6$           ;;BR IF NO
      MOVB   1(SP),-1(R3)  ;;YES--SET THE SIGN
      BIS    #'0,R2       ;;MAKE THE BCD DIGIT ASCII
      BIS    #' ,R2       ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
      MOVB   R2,(R3)+     ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
      TST    (R0)+       ;;JUST INCREMENTING
      CMP    R0,#10      ;;CHECK THE TABLE INDEX
      BLT    2$           ;;GO DO THE NEXT DIGIT
      BGT    8$           ;;GO TO EXIT
      MOV    R5,R2       ;;GET THE LSD
      BR     6$         ;;GO CHANGE TO ASCII
      TSTB  (SP)+       ;;WAS THE LSD THE FIRST NON-ZERO?
      BPL   9$         ;;BR IF NO
      MOVB  -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
      CLRB  (R3)        ;;SET THE TERMINATOR
      MOV   (SP)+,R5    ;;POP STACK INTO R5
      MOV   (SP)+,R3    ;;POP STACK INTO R3
      MOV   (SP)+,R2    ;;POP STACK INTO R2
      MOV   (SP)+,R1    ;;POP STACK INTO R1
      MOV   (SP)+,R0    ;;POP STACK INTO R0
      TYPE  ,SDBLK     ;;NOW TYPE THE NUMBER
    
```

```

042762
042762 010046
042764 010146
042766 010246
042770 010346
042772 010546
042774 012746 020200
043000 016605 000020
043004 100004
043006 005405
043010 112766 000055 000001
043016 005000 1$:
043020 012703 043176
043024 112723 000040
043030 005002 2$:
043032 016001 043166
043036 160105 3$:
043040 002402
043042 005202
043044 000774
043046 060105 4$:
043050 005702
043052 001002
043054 105716
043056 100407
043060 106316 5$:
043062 103003
043064 116663 000001 177777
043072 052702 000060 6$:
043076 052702 000040 7$:
043102 110223
043104 005720
043106 020027 000010
043112 002746
043114 003002
043116 010502
043120 000764
043122 105726 8$:
043124 100003
043126 116663 177777 177776
043134 105013 9$:
043136 012605
043140 012603
043142 012602
043144 012601
043146 012600
043150 104401 043176
    
```



4001

043154 016666 000002 000004  
 043162 012616  
 043164 000002  
 043166 023420  
 043170 001750  
 043172 000144  
 043174 000012  
 043176

```

MOV 2(SP),4(SP) ;;ADJUST THE STACK
MOV (SP)+,(SP)
RTI ;;RETURN TO USER
$DTBL: 10000.
      1000.
      100.
      10.
$DBLK: .BLKW 4
.SBTTL SAVE AND RESTORE R0-R5 ROUTINES
*****
*SAVE R0-R5
*CALL:
* SAVREG
*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
*
*TOP---(+16)
* +2---(+18)
* +4---R5
* +6---R4
* +8---R3
*+10---R2
*+12---R1
*+14---R0

```

043206  
 043206 010046  
 043210 010146  
 043212 010246  
 043214 010346  
 043216 010446  
 043220 010546  
 043222 016646 000022  
 043226 016646 000022  
 043232 016646 000022  
 043236 016646 000022  
 043242 000002

```

$SAVREG:
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
MOV R3,-(SP) ;;PUSH R3 ON STACK
MOV R4,-(SP) ;;PUSH R4 ON STACK
MOV R5,-(SP) ;;PUSH R5 ON STACK
MOV 22(SP),-(SP) ;;SAVE PS OF MAIN FLOW
MOV 22(SP),-(SP) ;;SAVE PC OF MAIN FLOW
MOV 22(SP),-(SP) ;;SAVE PS OF CALL
MOV 22(SP),-(SP) ;;SAVE PC OF CALL
RTI
*RESTORE R0-R5
*CALL:
* RESREG
$RESREG:
MOV (SP)+,22(SP) ;;RESTORE PC OF CALL
MOV (SP)+,22(SP) ;;RESTORE PS OF CALL
MOV (SP)+,22(SP) ;;RESTORE PC OF MAIN FLOW
MOV (SP)+,22(SP) ;;RESTORE PS OF MAIN FLOW
MOV (SP)+,R5 ;;POP STACK INTO R5
MOV (SP)+,R4 ;;POP STACK INTO R4
MOV (SP)+,R3 ;;POP STACK INTO R3
MOV (SP)+,R2 ;;POP STACK INTO R2
MOV (SP)+,R1 ;;POP STACK INTO R1
MOV (SP)+,R0 ;;POP STACK INTO R0
RTI

```

4002

043244  
 043244 012666 000022  
 043250 012666 000022  
 043254 012666 000022  
 043260 012666 000022  
 043264 012605  
 043266 012604  
 043270 012603  
 043272 012602  
 043274 012601  
 043276 012600  
 043300 000002

```

.SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
*****
*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
*UNSIGNED OCTAL ASCII NUMBER.
*CALL
* MOV #PNTR,-(SP) ;;POINTER TO LOW WORD OF BINARY NUMBER
* JSR PC,@#$DB20 ;;CALL THE ROUTINE

```

```

043302 104413          ;* RETURN          ;;THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK
043304 016601 000002  $DB20: SAVREG          ;;SAVE ALL REGISTERS
043310 012705 043421  MOV 2(SP),R1          ;;PICKUP THE POINTER TO LOW WORD
043314 012704 000014  MOV #SOCTVL+13.,R5      ;;POINTER TO DATA TABLE
043320 012703 177770  MOV #12.,R4           ;;DO ELEVEN CHARACTERS
043324 012100  MOV #^C7,R3          ;;MASK
043326 012101  MOV (R1)+,R0          ;;LOWER WORD
043330 005002  MOV (R1)+,R1          ;;HIGH WORD
043332 110245  CLR R2              ;;TERMINATOR
043334 010002  1$: MOVB R2,-(R5)    ;;PUT CHARACTER IN DATA TABLE
043336 005304  MOV RC,R2           ;;GET THIS DIGIT
043340 003007  DEC R4              ;;COUNT THIS CHARACTER
043342 001405  BGT 3$              ;;BR IF NOT THE LAST DIGIT
043344 005205  BEQ 2$              ;;BR IF IT IS THE LAST DIGIT
043346 010566 000002  INC R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
043352 104414  MOV R5,2(SP)        ;;ASCII CHAR. & PUT IT ON THE STACK
043354 000207  RESREG             ;;RESTORE ALL REGISTERS
043356 006203  2$: RTS PC          ;;RETURN TO USER
043360 006001  3$: ASR R3           ;;POSITION THE MASK FOR THE LAST DIGIT
043362 006000  ROR R1              ;;POSITION THE BINARY NUMBER FOR
043364 006001  ROR R0              ;; THE NEXT OCTAL DIGIT
043366 006000  ROR R1
043370 006001  ROR R0
043372 006000  ROR R1
043374 040302  BIC R3,R2           ;;MASK OUT ALL JUNK
043376 062702 000060  ADD #'0,R2          ;;MAKE THIS CHAR. ASCII
043402 000753  BR 1$               ;;GO PUT IT IN THE DATA TABLE
043404          $OCTVL: .BLKB 14.          ;;RESERVE DATA TABLE
    
```

4004

```
.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.
```

```
043422 010046
043424 016600 000002
043430 005740
043432 111000
043434 006300
043436 016000 043456
043442 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
```

```
043444 011646
043446 016666 000004 000002
043454 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.
```

```
043456 043444
043460 041660
043462 042560
043464 042534
043466 042574
043470 042762
043472 042460
043474 040614
043476 040544
043500 041066
043502 041206
043504 043206
043506 043244
```

```
ROUTINE
:-----
$TRPAD: .WORD $TRAP2
$TYPE ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
$TYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
$TYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
$TYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
$TYPDS ;;CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
$TYPBN ;;CALL=TYPBN TRAP+6(104406) TYPE BINARY (ASCII) NUMBER
$GTSWR ;;CALL=GTSWR TRAP+7(104407) GET SOFT-SWR SETTING
$CKSWR ;;CALL=CKSWR TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
$RDCHR ;;CALL=RDCHR TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
$RDLIN ;;CALL=RDLIN TRAP+12(104412) TTY TYPEIN STRING ROUTINE
$SAVREG ;;CALL=SAVREG TRAP+13(104413) SAVE R0-R5 ROUTINE
$RESREG ;;CALL=RESREG TRAP+14(104414) RESTORE R0-R5 ROUTINE
```

4005

```
.SBTTL POWER DOWN AND UP ROUTINES
:*****
```

```
043510 012737 043666 000024
043516 012737 000340 000026
043524 010046
043526 010146
043530 010246
043532 010346
043534 010446
043536 010546
043540 017746 135374
043544 010667 000122
043550 012737 043562 000024
043556 000000
043560 000776
```

```
:POWER DOWN ROUTINE
$PWRDN: MOV # $ILLUP,@#PWRVEC ;;SET FOR FAST UP
MOV #340,@#PWRVEC+2 ;;PRIO:7
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
MOV R3,-(SP) ;;PUSH R3 ON STACK
MOV R4,-(SP) ;;PUSH R4 ON STACK
MOV R5,-(SP) ;;PUSH R5 ON STACK
MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
MOV SP,$SAVR6 ;;SAVE SP
MOV # $PWRUP,@#PWRVEC ;;SET UP VECTOR
HALT
BR -2 ;;HANG UP
```

```
043562 012737 043666 000024
043570 016706 000076
043574 005067 000072
043600 005267 000066
```

```
:POWER UP ROUTINE
$PWRUP: MOV # $ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
MOV $SAVR6,SP ;;GET SP
CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
1$: INC $SAVR6 ;;WAIT FOR THE INC
```

```

043604 001375      BNE      1$          ;;OF WORD
043606 012677 135326  MOV      (SP)+,@SWR    ;;POP STACK INTO @SWR
043612 012605      MOV      (SP)+,R5      ;;POP STACK INTO R5
043614 012604      MOV      (SP)+,R4      ;;POP STACK INTO R4
043616 012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
043620 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
043622 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
043624 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
043626 012737 043510 000024  MOV      #PWRDN,@PWRVEC ;;SET UP THE POWER DOWN VECTOR
043634 012737 000340 000026  MOV      #340,@PWRVEC+2 ;;PRIO:7
043642 104401      TYPE                                ;;REPORT THE POWER FAILURE
043644 043674      $PWRMG: .WORD PWRMSG        ;;POWER FAIL MESSAGE POINTER
043646 012716      MOV      (PC)+,(SP)    ;;RESTART AT START
043650 020000      $PWRAD: .WORD START      ;;RESTART ADDRESS
043652 042766 000020 000002  BIC      #20,2(SP)     ;;CLEAR 'T' BIT
043660 005067 135424      CLR      $TBIT        ;;CLEAR THE 'T' BIT FLAG
043664 000002      RTI
043666 000000      $ILLUP: HALT          ;;THE POWER UP SEQUENCE WAS STARTED
043670 000776      BR      .-2          ;; BEFORE THE POWER DOWN WAS COMPLETE
043672 000000      $SAVR6: 0            ;;PUT THE SP HERE
4006 043674 012 015 040  PWRMSG: .ASCIZ <12><15>? POWER FAILURE - RESTARTING ?<12><15>
043677 120 117 127
043702 105 122 040
043705 106 101 111
043710 114 125 122
043713 105 040 055
043716 040 122 105
043721 123 124 101
043724 122 124 111
043727 116 107 040
4007 043732 012 015 000      .EVEN
    
```

```

4009          .SBTTL  ERROR MESSAGES, DATA HEADERS-TABLES & FORMATS
4010          .NLIST  BEX
4011 043736    125      116      105  EM1:  .ASCIZ  /UNEXPECTED CPU TRAP TO LOC. 004/
4012 043776    125      116      105  EM2:  .ASCIZ  /UNEXPECTED MEM. MGMT. TRAP TO LOC. 250/
4013 044045    115      105      115  EM10: .ASCIZ  /MEMORY MGMT. ACCESS ABORT DID NOT OCCUR/
4014 044115    101      103      103  EM11: .ASCIZ  /ACCESS ERROR DID NOT ABORT INSTRUCTION/
4015 044164    123      122      060  EM12: .ASCIZ  /SR0 DID NOT REPORT ACCESS ERROR CORRECTLY/
4016 044236    123      122      062  EM13: .ASCIZ  /SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR./
4017 044307    120      101      107  EM14: .ASCIZ  /PAGE LGTH. ABORT OCCURRED WHEN IT SHOULDN'T HAVE/
4018 044370    120      101      107  EM15: .ASCIZ  /PAGE LGTH. ABORT DID NOT OCCUR WHEN IT SHOULD HAVE/
4019 044453    123      122      060  EM16: .ASCIZ  /SR0 DID NOT REPORT PAGE LGTH. ABORT CORRECTLY/
4020 044531    123      122      060  EM21: .ASCIZ  /SR0 OR SR2 CHANGED BY A SECOND ABORT/
4021 044576    123      122      060  EM22: .ASCIZ  /SR0 OR SR2 WERE NOT 'RESET' BY A RESET/
4022 044645    123      122      062  EM23: .ASCIZ  /SR2 NOT TRACKING CORRECTLY/
4023 044700    104      111      104  EM24: .ASCIZ  /DID NOT TRAP THRU KERNEL SPACE/
4024 044737    113      124      040  EM25: .ASCIZ  /KT ERROR SERVICED ON ODD ADDR. ERROR/
4025 045004    123      122      060  EM26: .ASCIZ  /SR0 OR SR2 CHANGED BY ODD ADDR. ERROR/
4026 045052    105      122      122  EM27: .ASCIZ  /ERROR DURING 'DOUBLE ERROR' (KT & ODD ADDR.)/
4027 045127    115      106      120  EM30: .ASCIZ  /MFPI INSTRUCTION PUSHED WRONG DATA/
4028 045172    115      124      120  EM31: .ASCIZ  /MTPI INSTRUCTION LOADED WRONG DATA/
4029 045235    123      124      101  EM32: .ASCIZ  /STACK NOT PUSHED BY MFPI-MTPI/
4030 045273    113      105      122  EM33: .ASCIZ  /KERNEL PAGE ACCESS INSTEAD OF USER: MFPI-MTPI/
4031 045351    115      056      115  EM34: .ASCIZ  /M.M. ABORT IN KERNAL D-SPACE HAD WRONG CONDITION/
4032 045432    111      114      114  EM35: .ASCIZ  /ILLEGAL MODE 10 NOT ABORTED/
4033 045466    123      122      060  EM36: .ASCIZ  /SR0 DID NOT REPORT ILLEGAL MODE 10 CORRECTLY/
4034 045543    120      123      127  EM37: .ASCIZ  /PSW CHANGED BY AN RTI IN USER MODE/
4035 045606    101      102      117  EM40: .ASCIZ  /ABORT I KERNAL D-SPACE PICKED UP VECTOR FROM I-SPACE/
4036 045673    104      040      123  EM41: .ASCIZ  /D SPACE ENABLE CIRCUITRY HAS FAILED/
4037 045737    111      116      103  EM42: .ASCIZ  /INCORRECT STORE BY MTP INSTRUCTION/
4038 046002    124      122      111  EM43: .ASCIZ  /TRIED TO REFERENCE NON-RESIDENT PAGE/
4039 046047    127      122      117  EM44: .ASCIZ  /WRONG DATA FETCHED BY MFP INSTRUCTION/
4040 046115    111      114      114  EM45: .ASCIZ  /ILLEGAL CSM DID NOT TRAP TO 10/
4041 046154    103      123      115  EM46: .ASCIZ  /CSM DID NOT ENTER SUPERVISOR MODE/
4042 046216    103      123      115  EM47: .ASCIZ  /CSM SET UP WRONG PREVIOUS MODE/
4043 046255    103      123      115  EM50: .ASCIZ  /CSM SET UP STACK WRONG/
4044 046304    103      123      115  EM51: .ASCIZ  /CSM PUSHED INCORRECT ARGUMENT/
4045 046342    103      123      115  EM52: .ASCIZ  /CSM PUSHED WRONG PC/
4046 046366    103      123      115  EM53: .ASCIZ  /CSM DID NOT CLEAR OLD PSW BITS <3:0>/
4047 046433    103      123      115  EM54: .ASCIZ  /CSM ACCESSED WRONG SUPERVISOR SPACE/
4048 046477    103      123      115  EM55: .ASCIZ  /CSM ABORTED WHEN IT SHOULD NOT HAVE/
4049          .EVEN

```

|      |        |     |     |     |       |        |              |               |           |           |              |                   |
|------|--------|-----|-----|-----|-------|--------|--------------|---------------|-----------|-----------|--------------|-------------------|
| 4051 | 046544 | 117 | 114 | 104 | DH1:  | .ASCIZ | /OLD PC      | OLD PSW       | R6 WAS    | CPUERR    | TESTNO       | ERRORPC/          |
| 4052 | 046624 | 117 | 114 | 104 | DH2:  | .ASCIZ | /OLD PC      | OLD PSW       | R6 WAS    | SR0       | SR2          | TESTNO ERRORPC/   |
| 4053 | 046714 | 120 | 104 | 122 | DH10: | .ASCIZ | /PDR 4       | PSW           | TESTNO    | ERRORPC/  |              |                   |
| 4054 | 046754 | 123 | 122 | 060 | DH12: | .ASCIZ | /SR0 WAS     | EXPECTED      | PDR 4     | PSW       | TESTNO       | ERRORPC/          |
| 4055 | 047034 | 123 | 122 | 062 | DH13: | .ASCIZ | /SR2 WAS     | EXPECTED      | PDR 4     | PSW       | TESTNO       | ERRORPC/          |
| 4056 | 047114 | 126 | 056 | 102 | DH14: | .ASCIZ | /V.B.A.      | KIPDR4        | SR0 WAS   | SR2 WAS   | TESTNO       | ERRORPC/          |
| 4057 | 047174 | 126 | 056 | 102 | DH15: | .ASCIZ | /V.B.A.      | KIPDR4        | TESTNO    | ERRORPC/  |              |                   |
| 4058 | 047234 | 126 | 056 | 102 | DH16: | .ASCIZ | /V.B.A.      | KIPDR4        | SR0 WAS   | EXPECTED  | TESTNO       | ERRORPC/          |
| 4059 | 047314 | 126 | 056 | 102 | DH17: | .ASCIZ | /V.B.A.      | KIPDR4        | SR2 WAS   | EXPECTED  | TESTNO       | ERRORPC/          |
| 4060 | 047374 | 123 | 122 | 062 | DH20: | .ASCIZ | /SR2 WAS     | EXPECTED      | TESTNO    | ERRORPC/  |              |                   |
| 4061 | 047434 | 106 | 111 | 122 | DH21: | .ASCII | /FIRST ABORT | SECOND ABORT/ | <CP_r>    |           |              |                   |
| 4062 | 047471 | 123 | 122 | 060 |       | .ASCIZ | /SR0 WAS     | SR2 WAS       | SR0 WAS   | SR2 WAS   | TESTNO       | ERRORPC/          |
| 4063 | 047551 | 123 | 122 | 060 | DH22: | .ASCIZ | /SR0 WAS     | SR2 WAS       | TESTNO    | ERRORPC/  |              |                   |
| 4064 | 047611 | 120 | 123 | 127 | DH24: | .ASCIZ | /PSW WAS     | R6 WAS        | TESTNO    | ERRORPC/  |              |                   |
| 4065 | 047651 | 105 | 130 | 120 | DH26: | .ASCII | /EXPECTED    | RECEIVED/     | <CRLF>    |           |              |                   |
| 4066 | 047703 | 123 | 122 | 060 |       | .ASCIZ | /SR0         | SR2           | SR0 WAS   | SR2 WAS   | TESTNO       | ERRORPC/          |
| 4067 | 047763 | 105 | 130 | 120 | DH27: | .ASCII | /EXPECTED:   | <CRLF>        |           |           |              |                   |
| 4068 | 047775 | 120 | 123 | 127 |       | .ASCII | /PSW         | PC            | SR0       | SR2/      | <CRLF>       |                   |
| 4069 | 050031 | 061 | 067 | 060 |       | .ASCII | /170017      | (3\$+4)       | 020147    | (3\$)     | <CRLF>       |                   |
| 4070 | 050066 | 122 | 105 | 103 |       | .ASCII | /RECEIVED:   | <CRLF>        |           |           |              |                   |
| 4071 | 050100 | 120 | 123 | 127 |       | .ASCIZ | /PSW         | PC            | SR0       | SR2       | TESTNO       | ERRORPC/          |
| 4072 | 050160 | 104 | 101 | 124 | DH30: | .ASCII | /DATA        | DATA/         | <CRLF>    |           |              |                   |
| 4073 | 050175 | 105 | 130 | 120 |       | .ASCIZ | /EXPECTED    | RECEIVED      | TESTNO    | ERR PC    | SUB CALL/    |                   |
| 4074 | 050246 | 124 | 105 | 123 | DH32: | .ASCIZ | /TESTNO      | ERR PC        | SUB CALL/ |           |              |                   |
| 4075 | 050277 | 123 | 122 | 060 | DH33: | .ASCIZ | /SR0 WAS     | SR2 WAS       | TESTNO    | ERRORPC/  |              |                   |
| 4076 | 050337 | 050 | 115 | 115 | DH34: | .ASCIZ | /(MMR0)      | (MMR1)        | (MMR2)    | TESTNO    | ERRORPC      | EXPECTING 020031/ |
| 4077 | 050430 | 123 | 122 | 060 | DH36: | .ASCIZ | /SR0 WAS     | EXPECTED      | TESTNO    | ERRORPC/  |              |                   |
| 4078 | 050470 | 120 | 123 | 127 | DH37: | .ASCIZ | /PSW WAS     | EXPECTED      | TESTNO    | ERRORPC/  |              |                   |
| 4079 | 050530 | 050 | 120 | 123 | DH40: | .ASCIZ | /(PSW)       | TESTNO        | ERRORPC   | EXPECTING | XXX340/      |                   |
| 4080 | 050601 | 105 | 122 | 122 | DH41: | .ASCII | /ERROR       | AUTOI-D       | VIRTUAL/  | <CRLF>    |              |                   |
| 4081 | 050631 | 122 | 105 | 107 |       | .ASCIZ | /REGISTR     | REGISTR       | ADDRESS   | TESTNO    | PC AT ABORT/ |                   |
| 4082 | 050705 | 107 | 104 | 104 | DH42: | .ASCIZ | /GDDATA      | STORED        | TESTNO    | ERRORPC/  |              |                   |
| 4083 | 050745 | 050 | 115 | 115 | DH43: | .ASCIZ | /(MMR0)      | (MMR1)        | (MMR2)    | TESTNO    | ERRORPC/     |                   |
| 4084 | 051015 | 105 | 130 | 120 | DH44: | .ASCIZ | /EXPECTED    | (PSW)         | TESTNO    | ERRORPC   | SUB CALL/    |                   |
| 4085 | 051067 | 117 | 114 | 104 | DH45: | .ASCIZ | /OLDPSW      | TE TNO        | ERRORPC/  |           |              |                   |
| 4086 |        |     |     |     |       | .EVEN  |              |               |           |           |              |                   |

|      |        |        |        |        |       |       |   |
|------|--------|--------|--------|--------|-------|-------|---|
| 4088 | 051120 | 001260 | 001262 | 001256 | DT1:  | .WORD | TRAPPC, TRAPPS, WASR6, CPUERR, TESTNO, \$ERRPC, 0         |
| 4089 | 051136 | 001260 | 001262 | 001256 | DT2:  | .WORD | TRAPPC, TRAPPS, WASR6, WASSR0, WASSR2, TESTNO, \$ERRPC, 0 |
| 4090 | 051156 | 001166 | 001176 | 001254 | DT10: | .WORD | \$REG2, \$TMP0, TESTNO, \$ERRPC, 0                        |
| 4091 | 051170 | 001264 | 001170 | 001166 | DT12: | .WORD | WASSR0, \$REG3, \$REG2, \$TMP0, TESTNO, \$ERRPC, 0        |
| 4092 | 051206 | 001270 | 001172 | 001166 | DT13: | .WORD | WASSR2, \$REG4, \$REG2, \$TMP0, TESTNO, \$ERRPC, 0        |
| 4093 | 051224 | 001162 | 001172 | 001264 | DT14: | .WORD | \$REG0, \$REG4, WASSR0, WASSR2, TESTNO, \$ERRPC, 0        |
| 4094 | 051242 | 001162 | 001172 | 001254 | DT15: | .WORD | \$REG0, \$REG4, TESTNO, \$ERRPC, 0                        |
| 4095 | 051254 | 001162 | 001172 | 001264 | DT16: | .WORD | \$REG0, \$REG4, WASSR0, \$REG2, TESTNO, \$ERRPC, 0        |
| 4096 | 051272 | 001162 | 001172 | 001270 | DT17: | .WORD | \$REG0, \$REG4, WASSR2, \$REG3, TESTNO, \$ERRPC, 0        |
| 4097 | 051310 | 001270 | 001164 | 001254 | DT20: | .WORD | WASSR2, \$REG1, TESTNO, \$ERRPC, 0                        |
| 4098 | 051322 | 001176 | 001202 | 001264 | DT21: | .WORD | \$TMP0, \$TMP2, WASSR0, WASSR2, TESTNO, \$ERRPC, 0        |
| 4099 | 051340 | 001264 | 001270 | 001254 | DT22: | .WORD | WASSR0, WASSR2, TESTNO, \$ERRPC, 0                        |
| 4100 | 051352 | 001164 | 001166 | 001254 | DT24: | .WORD | \$REG1, \$REG2, TESTNO, \$ERRPC, 0                        |
| 4101 | 051364 | 001162 | 001164 | 001264 | DT26: | .WORD | \$REG0, \$REG1, WASSR0, WASSR2, TESTNO, \$ERRPC, 0        |
| 4102 | 051402 | 001164 | 001170 | 001264 | DT27: | .WORD | \$REG1, \$REG3, WASSR0, WASSR2, TESTNO, \$ERRPC, 0        |
| 4103 | 051420 | 001174 | 001206 | 001254 | DT30: | .WORD | \$REG5, \$TMP4, TESTNO, \$ERRPC, SRCALL, 0                |
| 4104 | 051434 | 001254 | 001116 | 037224 | DT32: | .WORD | TESTNO, \$ERRPC, SRCALL, 0                                |
| 4105 | 051444 | 001164 | 001166 | 001170 | DT34: | .WORD | \$REG1, \$REG2, \$REG3, TESTNO, \$ERRPC, 0                |
| 4106 | 051460 | 001264 | 001164 | 001254 | DT36: | .WORD | WASSR0, \$REG1, TESTNO, \$ERRPC, 0                        |
| 4107 | 051472 | 001162 | 001254 | 001116 | DT40: | .WORD | \$REG0, TESTNO, \$ERRPC, 0                                |
| 4108 | 051502 | 001264 | 001266 | 001270 | DT41: | .WORD | WASSR0, WASSR1, WASSR2, TESTNO, BADPC, 0                  |
| 4109 | 051516 | 001162 | 001164 | 001254 | DT42: | .WORD | \$REG0, \$REG1, TESTNO, \$ERRPC, 0                        |
| 4110 | 051530 | 001162 | 001164 | 001166 | DT43: | .WORD | \$REG0, \$REG1, \$REG2, TESTNO, \$ERRPC, 0                |
| 4111 | 051544 | 001264 | 001266 | 001270 | DT45: | .WORD | WASSR0, WASSR1, WASSR2, TESTNO, \$ERRPC, 0                |
| 4112 | 051560 | 001170 | 001206 | 001254 | DT46: | .WORD | \$REG3, \$TMP4, TESTNO, \$ERRPC, SRCALL, 0                |
| 4113 | 051574 | 001162 | 001164 | 001254 | DT47: | .WORD | \$REG0, \$REG1, TESTNO, \$ERRPC, SRCALL, 0                |
| 4114 | 051610 | 001206 | 001164 | 001254 | DT50: | .WORD | \$TMP4, \$REG1, TESTNO, \$ERRPC, SRCALL, 0                |

```
4116 051624      000      000      000 DF1:  .BYTE  0,0,0,0,0
4117 051631      000      000      000 DF2:  .BYTE  0,0,0,0,0,0,0
4118 051640      000      000      000 DF3:  .BYTE  0,0,0,0
4119 051644      000      000      000 DF5:  .BYTE  0,0,0
4120 051647      000      000      000 DF12: .BYTE  0,0,0,0,0,0
4121 051655      000      000      000 DF32: .BYTE  0,0
4122
4123
4124
4125
4126      000001
                                .EVEN
                                .LIST  BEX
                                .END
```



|                 |                 |                 |                 |                 |
|-----------------|-----------------|-----------------|-----------------|-----------------|
| ABASE = 000000  | BIT02 = 000004  | DH32 = 050246   | EM27 = 045052   | KIPDR2= 172304  |
| ACDW1 = 000000  | BIT03 = 000010  | DH33 = 0227     | EM30 = 045127   | KIPDR3= 172306  |
| ACDW2 = 000000  | BIT04 = 000020  | DH34 = 0337     | EM31 = 045172   | KIPDR4= 172310  |
| ACPUOP= 000000  | BIT05 = 000040  | DH36 = 050430   | EM32 = 045235   | KIPDR5= 172312  |
| ADDW0 = 000000  | BIT06 = 000100  | DH37 = 050470   | EM33 = 045273   | KIPDR6= 172314  |
| ADDW1 = 000000  | BIT07 = 000200  | DH40 = 050530   | EM34 = 045351   | KIPDR7= 172316  |
| ADDW10= 000000  | BIT08 = 000400  | DH41 = 050601   | EM35 = 045432   | KSP = %000006   |
| ADDW11= 000000  | BIT09 = 001000  | DH42 = 050705   | EM36 = 045466   | LCNT = 037222   |
| ADDW12= 000000  | BIT1 = 000002   | DH43 = 050745   | EM37 = 045543   | LF = 000012     |
| ADDW13= 000000  | BIT10 = 002000  | DH44 = 051015   | EM40 = 045606   | LOOP = 020456   |
| ADDW14= 000000  | BIT11 = 004000  | DH45 = 051067   | EM41 = 045673   | MGMERR = 002466 |
| ADDW15= 000000  | BIT12 = 010000  | DISPLA = 001142 | EM42 = 045737   | MGMFLG = 002470 |
| ADDW2 = 000000  | BIT13 = 020000  | DISPRE = 000174 | EM43 = 046002   | MMRO = 177572   |
| ADDW3 = 000000  | BIT14 = 040000  | DSWR = 177570   | EM44 = 046047   | MMR1 = 177574   |
| ADDW4 = 000000  | BIT15 = 100000  | DT1 = 051120    | EM45 = 046115   | MMR2 = 177576   |
| ADDW5 = 000000  | BIT2 = 000004   | DT10 = 051156   | EM46 = 046154   | MMR3 = 172516   |
| ADDW6 = 000000  | BIT3 = 000010   | DT12 = 051170   | EM47 = 046216   | MMVEC = 000250  |
| ADDW7 = 000000  | BIT4 = 000020   | DT13 = 051206   | EM50 = 046255   | NDFLAG = 002212 |
| ADDW8 = 000000  | BIT5 = 000040   | DT14 = 051224   | EM51 = 046304   | NODSPA = 002202 |
| ADDW9 = 000000  | BIT6 = 000100   | DT15 = 051242   | EM52 = 046342   | PBAHI = 001302  |
| ADEVCT= 000000  | BIT7 = 000200   | DT16 = 051254   | EM53 = 046366   | PBALO = 001300  |
| ADEVVM = 000000 | BIT8 = 000400   | DT17 = 051272   | EM54 = 046433   | PIRQ = 177772   |
| AENV = 000000   | BIT9 = 001000   | DT2 = 051136    | EM55 = 046477   | PIRQVE= 000240  |
| AENVM = 000000  | BPTVEC= 000014  | DT20 = 051310   | ERROR = 104000  | PNTR = 037226   |
| AFATAL = 000000 | CKSWR = 104410  | DT21 = 051322   | ERRTYP = 040236 | POPSTK = 037230 |
| AMADR1 = 000000 | CMMSG = 041607  | DT22 = 051340   | ERRVEC = 000004 | PLOOP = 037250  |
| AMADR2 = 000000 | CNTRLC = 041530 | DT24 = 051352   | GTSWR = 104407  | PRO = 000000    |
| AMADR3 = 000000 | CPUERR= 177766  | DT26 = 051364   | HT = 000011     | PR1 = 000040    |
| AMADR4 = 000000 | CR = 000015     | DT27 = 051402   | INIT = 020564   | PR2 = 000100    |
| AMAMS1 = 000000 | CRLF = 000200   | DT30 = 051420   | IOTVEC = 000020 | PR3 = 000140    |
| AMAMS2 = 000000 | CSMPC = 036460  | DT32 = 051434   | KDPAR0 = 172360 | PR4 = 000200    |
| AMAMS3 = 000000 | CSMRET = 037200 | DT34 = 051444   | KDPAR1 = 172362 | PR5 = 000240    |
| AMAMS4 = 000000 | CSMSUB = 036440 | DT36 = 051460   | KDPAR2 = 172364 | PR6 = 000300    |
| AMSGAD = 000000 | CSMWSE = 037146 | DT40 = 051472   | KDPAR3 = 172366 | PR7 = 000340    |
| AMSGLG = 000000 | DDISP = 177570  | DT41 = 051502   | KDPAR4 = 172370 | PS = 177776     |
| AMSGTY = 000000 | DF1 = 051624    | DT42 = 051516   | KDPAR5 = 172372 | PSHSTK = 037276 |
| AMTYP1 = 000000 | DF12 = 051647   | DT43 = 051530   | KDPAR6 = 172374 | PSLOOP = 037302 |
| AMTYP2 = 000000 | DF2 = 051631    | DT45 = 051544   | KDPAR7 = 172376 | PSW = 177776    |
| AMTYP3 = 000000 | DF3 = 051640    | DT46 = 051560   | KDPDR0 = 172320 | PWRMSG = 043674 |
| AMTYP4 = 000000 | DF32 = 051655   | DT47 = 051574   | KDPDR1 = 172322 | PWRVEC = 000024 |
| APASS = 000000  | DF5 = 051644    | DT50 = 051610   | KDPDR2 = 172324 | RDCHR = 104411  |
| APRINI = 002030 | DH1 = 046544    | EMTVEC = 000030 | KDPDR3 = 172326 | RDLIN = 104412  |
| APRIOR = 000000 | DH10 = 046714   | EM1 = 043736    | KDPDR4 = 172330 | RESREG = 104414 |
| APTCSU = 000040 | DH12 = 046754   | EM10 = 044045   | KDPDR5 = 172332 | RESVEC = 000010 |
| APTENV = 000001 | DH13 = 047034   | EM11 = 044115   | KDPDR6 = 172334 | RETURN = 037270 |
| APTSIZ = 000200 | DH14 = 047114   | EM12 = 044164   | KDPDR7 = 172336 | R6 = %000006    |
| APTSPO = 000100 | DH15 = 047174   | EM13 = 044236   | KERSTK = 001100 | R7 = %000007    |
| ASWREG = 000000 | DH16 = 047234   | EM14 = 044307   | KIPAR0 = 172340 | SAVREG = 104413 |
| ATESTN = 000000 | DH17 = 047314   | EM15 = 044370   | KIPAR1 = 172342 | SCOPE = 000004  |
| ALUNIT = 000000 | DH2 = 046624    | EM16 = 044453   | KIPAR2 = 172344 | SDPAR0 = 172260 |
| AUSWR = 000000  | DH20 = 047374   | EM2 = 043776    | KIPAR3 = 172346 | SDPAR1 = 172262 |
| AVECT1 = 000000 | DH21 = 047434   | EM21 = 044531   | KIPAR4 = 172350 | SDPAR2 = 172264 |
| AVECT2 = 000000 | DH22 = 047551   | EM22 = 044576   | KIPAR5 = 172352 | SDPAR3 = 172266 |
| BADPC = 001304  | DH24 = 047611   | EM23 = 044645   | KIPAR6 = 172354 | SDPAR4 = 172270 |
| BIT0 = 000001   | DH26 = 047651   | EM24 = 044700   | KIPAR7 = 172356 | SDPAR5 = 172272 |
| BIT00 = 000001  | DH27 = 047763   | EM25 = 044737   | KIPDR0 = 172300 | SDPAR6 = 172274 |
| BIT01 = 000002  | DH30 = 050160   | EM26 = 045004   | KIPDR1 = 172302 | SDPAR7 = 172276 |

|                |                |                |                 |                 |
|----------------|----------------|----------------|-----------------|-----------------|
| SDPDR0= 172220 | SW5 = 000040   | UDPAR2= 177664 | \$CNTLC 001312  | \$OCTVL 043404  |
| SDPDR1= 172222 | SW6 = 000100   | UDPAR3= 177666 | \$CNTLG 041501  | \$OMODE 042760  |
| SDPDR2= 172224 | SW7 = 000200   | UDPAR4= 177670 | \$CNTLU 041474  | \$OVER 037764   |
| SDPDR3= 172226 | SW8 = 000400   | UDPAR5= 177672 | \$CPUOP 001252  | \$PASS 001232   |
| SDPDR4= 172230 | SW9 = 001000   | UDPAR6= 177674 | \$CRLF 001221   | \$PASTM 000212  |
| SDPDR5= 172232 | TBIT = 000020  | UDPAR7= 177676 | \$DBLK 043176   | \$PWRAD 043650  |
| SDPDR6= 172234 | TBITPS 001274  | UDPDR0= 177620 | \$DB20 043302   | \$PWRDN 043510  |
| SDPDR7= 172236 | TBITVE= 000014 | UDPDR1= 177622 | \$DEVCT 001234  | \$PWRMG 043644  |
| SIPAR0= 172240 | TEMP 037206    | UDPDR2= 177624 | \$DOAGN 037540  | \$PWRUP 043562  |
| SIPAR1= 172242 | TESTNO 001254  | UDPDR3= 177626 | \$DTBL 043166   | \$QUES 001220   |
| SIPAR2= 172244 | TIMERR 002414  | UDPDR4= 177630 | \$ENDAD 037530  | \$RDCHR 041066  |
| SIPAR3= 172246 | TIMFLG 002416  | UDPDR5= 177632 | \$ENDCT 037356  | \$RDLIN 041206  |
| SIPAR4= 172250 | TKVER = 000060 | UDPDR6= 177634 | \$ENULL 037602  | \$RDSZ = 000010 |
| SIPAR5= 172252 | TOFF 002326    | UDPDR7= 177636 | \$ENV 001244    | \$REGAD 001160  |
| SIPAR6= 172254 | TON 002362     | UIPAR0= 177640 | \$ENVM 001245   | \$REGO 001162   |
| SIPAR7= 172256 | TPVEC = 000064 | UIPAR1= 177642 | \$EOP 037326    | \$REG1 001164   |
| SIPDR0= 172200 | TRAPPC 001260  | UIPAR2= 177644 | \$EOPCT 037350  | \$REG2 001166   |
| SIPDR1= 172202 | TRAPPS 001262  | UIPAR3= 177646 | \$ERFLG 001103  | \$REG3 001170   |
| SIPDR2= 172204 | TRAPVE= 000034 | UIPAR4= 177650 | \$ERMAX 001115  | \$REG4 001172   |
| SIPDR3= 172206 | TRTVEC= 000014 | UIPAR5= 177652 | \$ERROR 040000  | \$REG5 001174   |
| SIPDR4= 172210 | TST1 020602    | UIPAR6= 177654 | \$ERRPC 001116  | \$RESRE 043244  |
| SIPDR5= 172212 | TST10 023076   | UIPAR7= 177656 | \$ERRTB 001320  | \$RTNAD 037600  |
| SIPDR6= 172214 | TST11 023406   | UIPDR0= 177600 | \$ERTTL 001112  | \$RTRN 037574   |
| SIPDR7= 172216 | TST12 023504   | UIPDR1= 177602 | \$ESCAP 001212  | \$SAVRE 043206  |
| SRCALL 037224  | TST13 023702   | UIPDR2= 177604 | \$ETABL 001244  | \$SAVR6 043672  |
| SRO = 177572   | TST14 024224   | UIPDR3= 177606 | \$ETEND 001254  | \$SCOPE 037606  |
| SR1 = 177574   | TST15 025210   | UIPDR4= 177610 | \$FATAL 001226  | \$SETUP= 000137 |
| SR2 = 177576   | TST16 026254   | UIPDR5= 177612 | \$FFLG 042456   | \$STUP = 177777 |
| SR3 = 172516   | TST17 026532   | UIPDR6= 177614 | \$FILLC 001156  | \$SVLAD 037730  |
| SSP = %000006  | TST2 021126    | UIPDR7= 177616 | \$FILLS 001155  | \$SVPC = 000204 |
| STACK = 001100 | TST20 027014   | USESTK= 000600 | \$GDADR 001120  | \$SWR = 173400  |
| START 020000   | TST21 027236   | USP = %000006  | \$GDDAT 001124  | \$SWREG 001246  |
| STKLMT= 177774 | TST22 027754   | VIRT1 001276   | \$GET42 037502  | \$SWRMK= 000000 |
| SUPSTK= 000700 | TST23 030452   | WASR6 001256   | \$GTSWR 040614  | \$TBIT 001310   |
| SWR 001140     | TST24 031310   | WASSR0 001264  | \$HD = 000000   | \$TESTN 001230  |
| SWREG 000176   | TST25 032154   | WASSR1 001266  | \$HIBTS 000204  | \$TKB 001146    |
| SWO = 000001   | TST26 032664   | WASSR2 001270  | \$ICNT 001104   | \$TKS 001144    |
| SW00 = 000001  | TST27 033374   | WASSR3 001272  | \$ILLUP 043666  | \$TMP0 001176   |
| SW01 = 000002  | TST3 021370    | WBIT = 000100  | \$INTAG 001135  | \$TMP1 001200   |
| SW02 = 000004  | TST30 033746   | \$APTHD 000204 | \$ITEMB 001114  | \$TMP2 001202   |
| SW03 = 000010  | TST31 034304   | \$ATYC 042236  | \$LF 001222     | \$TMP3 001204   |
| SW04 = 000020  | TST32 034656   | \$ATY1 042212  | \$LFLG 042455   | \$TMP4 001206   |
| SW05 = 000040  | TST33 034726   | \$ATY3 042220  | \$LOOP 037576   | \$TMP5 001210   |
| SW06 = 000100  | TST34 035314   | \$ATY4 042230  | \$LPADR 001106  | \$TN = 000036   |
| SW07 = 000200  | TST35 035654   | \$AUTOB 001134 | \$LPERR 001110  | \$TPB 001152    |
| SW08 = 000400  | TST4 021520    | \$BDADR 001122 | \$MAIL 001224   | \$TPFLG 001157  |
| SW09 = 001000  | TST5 022022    | \$BDDAT 001126 | \$MBADR 000206  | \$TPS 001150    |
| SW1 = 000002   | TST6 022310    | \$BELL 001214  | \$MFLG 042454   | \$TRAP 043422   |
| SW10 = 002000  | TST7 022476    | \$BIN 042532   | \$MNEW 041517   | \$TRAP2 043444  |
| SW11 = 004000  | TYPBN = 104406 | \$CHARC 042206 | \$MSGAD 001240  | \$TRP = 000015  |
| SW12 = 010000  | TYPDS = 104405 | \$CKSWR 040544 | \$MSGLG 001242  | \$TRPAD 043456  |
| SW13 = 020000  | TYPE = 104401  | \$CLR.T 037520 | \$MSGTY 001224  | \$TSTM 000210   |
| SW14 = 040000  | TYPOC = 104402 | \$CMTAG 001100 | \$MSWR 041506   | \$TSTNM 001102  |
| SW15 = 100000  | TYPON = 104404 | \$CM1 = 000006 | \$MXCNT 001306  | \$TTYIN 041464  |
| SW2 = 000004   | TYPOS = 104403 | \$CM2 = 000014 | \$NULL 001154   | \$TYPBN 042460  |
| SW3 = 000010   | UDPAR0= 177660 | \$CM3 = 000006 | \$NWTST= 000001 | \$TYPDS 042762  |
| SW4 = 000020   | UDPAR1= 177662 | \$CM4 = 000006 | \$OCNT 042756   | \$TYPE 041660   |

\$TYPEC 042072  
\$TYPEX 042210  
\$TYPOC 042560

\$TYPON 042574  
\$TYPOS 042534  
\$UNIT 001236

\$UNITM 000214  
\$USWR 001250

\$XTSTR 037620  
\$\$GET4= 000001

\$OFILL 042757  
\$.SX = 000204

. ABS. 051660    000  
         000000    001

ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 51968 WORDS ( 203 PAGES)  
DYNAMIC MEMORY: 20224 WORDS ( 77 PAGES)  
ELAPSED TIME: 00:05:35  
CKKTBB.BIN,CKKTBB.SEQ/CR/-SP-CKKTBB0.MLB/ML,CKKTBB0.P11