

# KD11-Z

11/44 MEM MGMT PRT A  
CKKTA00

AH-F611A-MC  
COPYRIGHT 1980  
FICHE 1 OF 1

JAN 1980  
**digital**  
MADE IN USA



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

.REM    @

IDENTIFICATION

PRODUCT CODE:    AC-F609A-MC  
PRODUCT NAME:    CKKTAA0 11/44 MEM MGMT PRT A  
DATE:            OCTOBER, 1979  
MAINTAINER:      DIAGNOSTIC ENGINEERING  
AUTHOR:          J. PETER BRADY

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  
52  
53  
54

PROGRAM HISTORY

<u>DATE</u>	<u>REVISION</u>	<u>REASON FOR REVISION</u>
OCTOBER, 1979	A	FIRST RELEASE

56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91

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

93 1.0 PROGRAM INFORMATION  
94 -----  
95  
96 1.1 ABSTRACT  
97 -----  
98  
99 THIS PROGRAM WAS DESIGNED USING A 'BOTTOM UP' APPROACH  
100 STARTING WITH THE SMALLEST SEGMENT OF MEMORY MANAGEMENT  
101 LOGIC POSSIBLE AND BUILDING TO COVER ALL OF THE LOGIC.  
102 THE DIAGNOSTIC WILL PROVIDE ENOUGH INFORMATION SUCH THAT  
103 BY DEDUCTION, THE FAILURE CAN BE ISOLATED TO A SMALL  
104 SEGMENT OF THE MEMORY MANAGEMENT LOGIC.  
105  
106 PART A BEGINS BY TESTING SOME OF THE INTERNAL CPU DATA  
107 AND ADDRESS PATHS AND ADDRESS DETECTION LOGIC, THEN  
108 WORKS OUTWARD THROUGH THE MEMORY MANAGEMENT REGISTERS.  
109 AFTER THE REGISTERS ARE FOUND TO BE USEABLE, RELOCATION  
110 (CONSTRUCTION OF PHYSICAL ADDRESSES FROM A VIRTUAL ADDRESS  
111 AND THE ASSOCIATED PAR/PDR INFORMATION) IS TESTED. PART B  
112 BEGINS BY TESTING THE ABORT AND STATUS SEGMENTS OF LOGIC.  
113 PART B THEN CHECKS THE SPECIAL ABORT SEQUENCES, THE MFPI/MTPI  
114 INSTRUCTIONS AND THE CSM INSTRUCTION.  
115  
116 1.2 REQUIREMENTS  
117 -----  
118  
119 A PDP 11/44 PROCESSOR WITH A MINIMUM OF 16K OF MEMORY  
120 AND A CONSOLE TERMINAL ARE REQUIRED TO RUN THE PROGRAM  
121 UNLESS THE PROGRAM IS RUNNING UNDER APT OR ACT IN WHICH  
122 CASE THE CONSOLE TERMINAL IS NOT NECESSARY.  
123  
124 1.3 RELATED DOCUMENTS AND STANDARDS  
125 -----  
126  
127 1. ACT11/XXDP PROGRAMMING SPECIFICATION  
128 2. STANDARD APT SYSTEM TO A PDP11 DIAGNOSTIC INTERFACE  
129 3. DIAGNOSTIC ENGINEERING STANDARDS AND CONVENTIONS  
130 4. PDP11 MAINDEC SYSMAC PACKAGE  
131 5. XXDP USER'S MANUAL  
132  
133 1.4 PRELIMINARY PROGRAMS  
134 -----  
135  
136 BEFORE THIS MEMORY MANAGEMENT DIAGNOSTIC IS RUN, THE  
137 FOLLOWING DIAGNOSTICS SHOULD BE RUN:  
138  
139 CKKAAA0 1/44 CPU/EIS  
140 CKKABAO 11/44 TRAPS  
141  
142 ALSO, THE MAIN MEMORY DIAGNOSTIC (CZMSD) SHOULD BE RUN TO SCAN  
143 AT LEAST THE FIRST 16K TO SEE THAT A PROGRAM CAN BE EXECUTED.  
144  
145 2.0 OPERATING INSTRUCTIONS  
146 -----  
147  
148 2.1 LOADING PROCEDURES  
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  
201  
202  
203  
204  
205  
206

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 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 THIS SWITCH WHEN SET WILL CAUSE THE PROGRAM TO LOOP ON THE CURRENT SUBTEST.
SW13	020000	INHIBIT ERROR TYPEOUTS THIS SWITCH WHEN SET WILL INHIBIT THE TYPING OF ERROR MESSAGES.
SW12	010000	INHIBIT TRACE TRAP THIS SWITCH WHEN SET WILL INHIBIT T-BIT TRAPPING WHICH NORMALLY TAKES PLACE DURING EVERY OTHER PASS STARTING WITH THE THIRD PASS.
SW10	002000	BELL ON ERROR

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  
258  
259  
260  
261  
262  
263

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

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  
315  
316  
317  
318  
319  
320

----- 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 8 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.



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  
372  
373  
374  
375  
376  
377

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' (F.T 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).

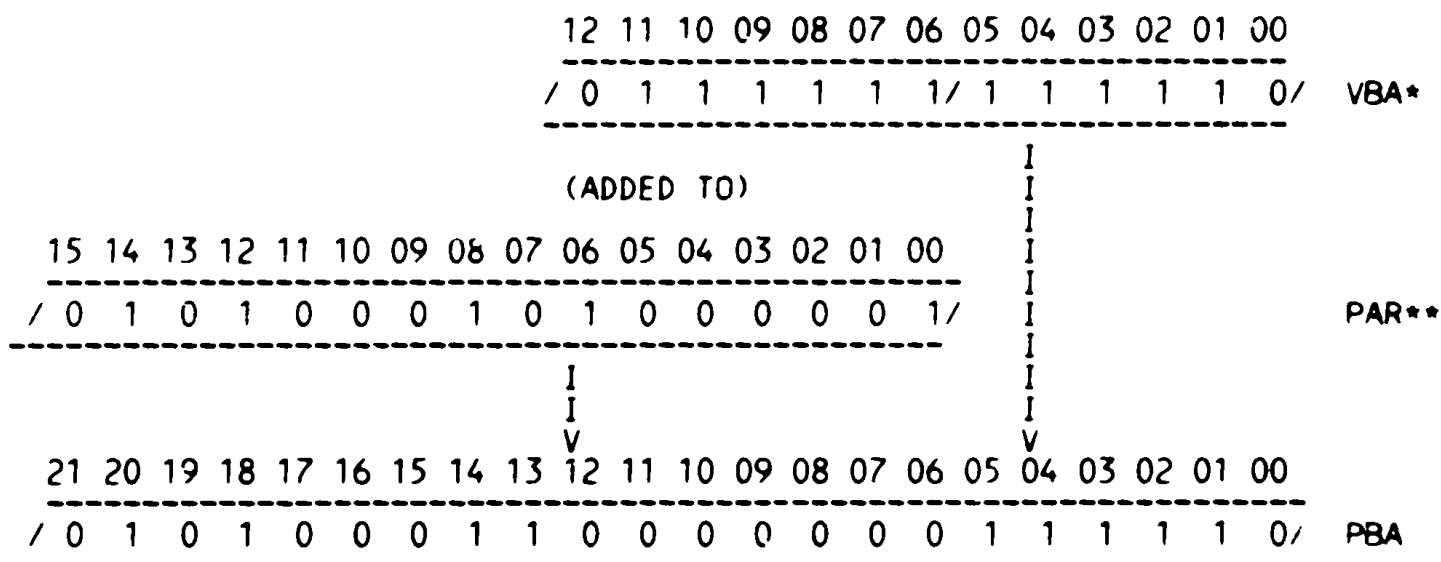
4.4      POWER FAILURE HANDLING

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  
429  
430  
431  
432  
433  
434

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

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  
480  
481  
482  
483  
484  
485  
486

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'.



804  
805

```
.TITLE CKKTAAD 11/44 MEM MGMT PRT A
:*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.
:*
```

806

```
.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>
```

807

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
```

```

                                ;*'SWITCH REGISTER' SWITCH DEFINITIONS
100000 SW15= 100000
040000 SW14= 40000
020000 SW13= 20000
010000 SW12= 10000
004000 SW11= 4000
002000 SW10= 2000
001000 SW09= 1000
000400 SW08= 400
000200 SW07= 200
000100 SW06= 100
000040 SW05= 40
000020 SW04= 20
000010 SW03= 10
000004 SW02= 4
000002 SW01= 2
000001 SW00= 1
                                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)
100000 BIT15= 100000
040000 BIT14= 40000
020000 BIT13= 20000
010000 BIT12= 10000
004000 BIT11= 4000
002000 BIT10= 2000
001000 BIT09= 1000
000400 BIT08= 400
000200 BIT07= 200
000100 BIT06= 100
000040 BIT05= 40
000020 BIT04= 20
000010 BIT03= 10
000004 BIT02= 4
000002 BIT01= 2
000001 BIT00= 1
                                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
000004 ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
000010 RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
```

808

```
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
J00030  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

.SBTTL MEMORY MANAGEMENT DEFINITIONS
;*KT11 VECTOR ADDRESS
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 UIPDR6= 177614
177616 UIPDR7= 177616
;*USER 'D' PAGE DESCRIPTOR REGISTORS
177620 UDPDR0= 177620
177622 UDPDR1= 177622
177624 UDPDR2= 177624
177626 UDPDR3= 177626
177630 UDPDR4= 177630
177632 UDPDR5= 177632
177634 UDPDR6= 177634
177636 UDPDR7= 177636
;*USER 'I' PAGE ADDRESS REGISTERS
177640 UIPAR0= 177640
177642 UIPAR1= 177642
177644 UIPAR2= 177644
177646 UIPAR3= 177646
177650 UIPAR4= 177650
177652 UIPAR5= 177652
177654 UIPAR6= 177654
177656 UIPAR7= 177656
;*USER 'D' PAGE ADDRESS REGISTERS
177660 UDPAR0= 177660
177662 UDPAR1= 177662
177664 UDPAR2= 177664
177666 UDPAR3= 177666
177670 UDPAR4= 177670
177672 UDPAR5= 177672
177674 UDPAR6= 177674
177676 UDPAR7= 177676
;*SUPERVISOR '' 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
;*
809
810
811 177572 MMR0=SR0
812 177574 MMR1=SR1
813 177576 MMR2=SR2
814 172516 MMR3=SR3
815 000006 KSP=SP
816 000006 SSP=SP
817 000006 USP=SP
818 000020 TBIT=BIT4
819 000100 WBIT=BIT6
820 001100 KERSTK=STACK
821 000700 SUPSTK=STACK-200
822 000600 USESTK=STACK-300
823 177766 CPUERR=177766
824 177770 RMIREG=177770
879
881 .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
000174 000174 .=174
000174 000000 DISPREG: .WORD 0 ;:SOFTWARE DISPLAY REGISTER
000176 000000 SWREG: .WORD 0 ;:SOFTWARE SWITCH REGISTER
882 000200 000137 020000 .SBTTL STARTING ADDRESS(ES)
      JMP @START ;:JUMP TO STARTING ADDRESS OF PROGRAM
      .SBTTL ACT11 HOOKS
      ;:*****
      ;HOOKS REQUIRED BY ACT11
000204 000204 $SVPC=. ;SAVE PC
000046 000046 .-46
035116 $ENDAD ;:1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
000052 000052 .=52
000052 000000 .WORD 0 ;:2)SET LOC.52 TO ZERO
883 000204 . $SVPC ;: RESTORE PC
      .SBTTL APT PARAMETER BLOCK
      ;:*****
      ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
      ;:*****
000204 000204 .SX=. ;:SAVE CURRENT LOCATION
000024 000024 .=24 ;:SET POWER FAIL TO POINT TO START OF PROGRAM
000200 200 ;:FOR APT START UP
000044 . 44 ;:POINT TO APT INDIRECT ADDRESS PNTR.
000044 000204 $APTHDR ;:POINT TO APT HEADER BLOCK
```

000204

.=.\$X ;;RESET LOCATION COUNTER

\*\*\*\*\*  
:SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC  
:INTERFACE SPEC.

000204  
000204 000000  
000206 001224  
000210 000010  
000212 000020  
000214 000005  
000216 000014

\$APTHD:  
\$HIBTS: .WORD 0 ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.  
\$MBADR: .WORD \$MAIL ;;ADDRESS OF APT MAILBOX (BITS 0-15)  
\$TSTM: .WORD 10 ;;RUN TIM OF LONGEST TEST  
\$PASTM: .WORD 20 ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)  
\$UNITM: .WORD 5 ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT  
.WORD \$ETEND-\$MAIL/2 ;;LENGTH MAILBOX-ETABLE (WORDS)



884

001100 001100  
001100 000000  
001102 000  
001103 000  
001104 000000  
001106 000000  
001110 000000  
001112 000000  
001114 000  
001115 001  
001116 000000  
001120 000000  
001122 000000  
001124 000000  
001126 000000  
001130 000000  
001132 000000  
001134 000  
001135 000  
001136 000000  
001140 177570  
001142 177570  
001144 177560  
001146 177562  
001150 177564  
001152 177566  
001154 000  
001155 002  
001156 012  
001157 000  
001160 000000  
  
001162 000006  
001164 000000  
001166 000000  
001170 000000  
001172 000000  
001174 000000  
001176 000006  
001200 000000  
001202 000000  
001204 000000  
001206 000000  
001210 000000  
001212 000000  
001214 207 377 377  
001217 000  
001220 077  
001221 015  
001222 012 000

```
.SBTTL COMMON TAGS
:*****
:*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
:*USED IN THE PROGRAM.
.=1100

$CMTAG:                ;; START OF COMMON TAGS
                        .WORD 0
$STNM: .BYTE 0        ;; CONTAINS THE TEST NUMBER
$ERFLG: .BYTE 0       ;; CONTAINS ERROR FLAG
$ICNT: .WORD 0        ;; CONTAINS SUBTEST ITERATION COUNT
$LPADR: .WORD 0       ;; CONTAINS SCOPE LOOP ADDRESS
$LPERR: .WORD 0       ;; CONTAINS SCOPE RETURN FOR ERRORS
$ERTTL: .WORD 0       ;; CONTAINS TOTAL ERRORS DETECTED
$ITEMB: .BYTE 0       ;; CONTAINS ITEM CONTROL BYTE
$ERMAX: .BYTE 1       ;; CONTAINS MAX. ERRORS PER TEST
$ERRPC: .WORD 0       ;; CONTAINS PC OF LAST ERROR INSTRUCTION
$GDADR: .WORD 0       ;; CONTAINS ADDRESS OF 'GOOD' DATA
$BDADR: .WORD 0       ;; CONTAINS ADDRESS OF 'BAD' DATA
$GDDAT: .WORD 0       ;; CONTAINS 'GOOD' DATA
$BDDAT: .WORD 0       ;; CONTAINS 'BAD' DATA
                        .WORD 0
                        .WORD 0
                        .WORD 0
$AUTOB: .BYTE 0       ;; AUTOMATIC MODE INDICATOR
$INTAG: .BYTE 0       ;; INTERRUPT MODE INDICATOR
                        .WORD 0
$SWR: .WORD DSWR      ;; ADDRESS OF SWITCH REGISTER
$DISPLAY: .WORD DDISP ;; ADDRESS OF DISPLAY REGISTER
$TKS: 177560          ;; TTY KBD STATUS
$TKB: 177562          ;; TTY KBD BUFFER
$TPS: 177564          ;; TTY PRINTER STATUS REG. ADDRESS
$TPB: 177566          ;; TTY PRINTER BUFFER REG. ADDRESS
$NULL: .BYTE 0        ;; CONTAINS NULL CHARACTER FOR FILLS
$FILLS: .BYTE 2       ;; CONTAINS # OF FILLER CHARACTERS REQUIRED
$FILLC: .BYTE 12      ;; INSERT FILL CHARS. AFTER A 'LINE FEED'
$TPFLG: .BYTE 0       ;; 'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
$REGAD: .WORD 0       ;; CONTAINS THE ADDRESS FROM
                        ;; WHICH ($REGO) WAS OBTAINED

$REGO: .REPT $CM3     .WORD 0 ;; CONTAINS (($REGAD)+0)
$REG1: .WORD 0        ;; CONTAINS (($REGAD)+2)
$REG2: .WORD 0        ;; CONTAINS (($REGAD)+4)
$REG3: .WORD 0        ;; CONTAINS (($REGAD)+6)
$REG4: .WORD 0        ;; CONTAINS (($REGAD)+10)
$REG5: .WORD 0        ;; CONTAINS (($REGAD)+12)
.REPT 6
$TMP0: .WORD 0        ;; USER DEFINED
$TMP1: .WORD 0        ;; USER DEFINED
$TMP2: .WORD 0        ;; USER DEFINED
$TMP3: .WORD 0        ;; USER DEFINED
$TMP4: .WORD 0        ;; USER DEFINED
$TMP5: .WORD 0        ;; USER DEFINED
$ESCAPE: 0           ;; ESCAPE ON ERROR ADDRESS
$BELL: .ASCIIZ <207><377><377> ;; CODE FOR BELL
$QUES: .ASCII /?/    ;; QUESTION MARK
$CRLF: .ASCII <15>   ;; CARRIAGE RETURN
$LF: .ASCIIZ <12>    ;; LINE FEED
```

\*\*\*\*\*  
:SBTTL APT MAILBOX-ETABLE  
:\*\*\*\*\*

001224  
001224 000000  
001226 000000  
001230 000000  
001232 000000  
001234 000000  
001236 000000  
001240 000000  
001242 000000  
001244  
001244 000  
001245 000  
001246 000000  
001250 000000  
001252 000000

.EVEN  
\$MAIL: APT MAILBOX  
\$MSGTY: .WORD AMSGTY :MESSAGE TYPE CODE  
\$FATAL: .WORD AFATAL :FATAL ERROR NUMBER  
\$TESTN: .WORD ATESTN :TEST NUMBER  
\$PASS: .WORD APASS :PASS COUNT  
\$DEVCT: .WORD ADEVCT :DEVICE COUNT  
\$UNIT: .WORD AUNIT :I/O UNIT NUMBER  
\$MSGAD: .WORD AMSGAD :MESSAGE ADDRESS  
\$MSGLG: .WORD AMSGLG :MESSAGE LENGTH  
\$ETABLE: APT ENVIRONMENT TABLE  
\$ENV: .BYTE AENV :ENVIRONMENT BYTE  
\$ENVM: .BYTE AENVM :ENVIRONMENT MODE BITS  
\$SWREG: .WORD ASWREG :APT SWITCH REGISTER  
\$USWR: .WORD AUSWR :USER SWITCHES  
\$CPUOP: .WORD ACPUOP :CPU TYPE,OPTIONS

BIT 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  
001256 000000  
001260 000000  
001262 000000  
001264 000000  
001266 000000  
001270 000000  
001272 000000  
001274 000000  
001276 000000  
001300 000000  
001302 000000  
001304 000000  
001306 000000  
001310 000000  
001312 000000  
001314 000000  
001316 000000  
001320 000000  
001322 000000  
001324 000000  
001326 000000  
001330 000000  
001332 000000  
001334 000000  
001336 000000  
001340 000000  
001342 020000  
001344 000200

TESTNO: .WORD 0 :HOLDS TEST NUMBER FOR TYPEOUTS  
WASR6: .WORD 0 :USED TO STORE THE STACK POINTER AFTER A TRAP  
TRAPPC: .WORD 0 :USED TO STORE THE PC OF A TRAP OR ABORT  
TRAPPS: .WORD 0 :USED TO STORE THE PS OF A TRAP OR ABORT  
WASSR0: .WORD 0 :USED TO STORE CONTENTS OF SR0  
WASSR1: .WORD 0 :USED TO STORE CONTENTS OF SR1  
WASSR2: .WORD 0 :USED TO STORE CONTENTS OF SR2  
TBITPS: .WORD 0 :SAVES THE PSW THAT MAY HAVE ITS T-BIT ON  
TONUM: .WORD 0 :HOLDS NUMBER OF TIME-OUTS  
VIRT1: .WORD 0 :HOLDS VIRTUAL ADDRESS TO BE CONVERTED  
VIRT2: .WORD 0 :  
PEALO: .WORD 0 :HOLDS BITS <15:00> OF PHYSICAL ADDRESS  
PBAHI: .WORD 0 :HOLDS BITS <21:16> OF PHYSICAL ADDRESS  
DATAOR: .WORD 0 :HOLDS LOGICAL OR OF BAD DATA  
DATAND: .WORD 0 :HOLDS LOGICAL AND OF BAD DATA  
PATTOR: .WORD 0 :HOLDS LOGICAL OR OF PATTERN LOADED  
PATAND: .WORD 0 :HOLDS LOGICAL AND OF PATTERN LOADED  
ADDROR: .WORD 0 :HOLDS LOGICAL OR OF ADDRESS  
ADRAND: .WORD 0 :HOLDS LOGICAL AND OF ADDRESS  
ERRCNT: .WORD 0 :HOLDS NUMBER OF ERRORS ON TEST  
BADPC: .WORD 0 :HOLDS PC FROM ABORT OR TRAP  
OLDPC: .WORD 0 :HOLDS RETURN ADDRESS IN CASE OF LOOP ON ERROR  
OLDPS: .WORD 0 :HOLDS OLD PSW IN CASE OF LOOP ON ERROR  
PCPUER: .WORD 0 :HOLDS VALUE OF CPU ERROR REGISTER  
CPUEXP: .WORD 0 :HOLDS EXPECTED CPU ERROR CONDITION  
HOLFLG: .WORD 0 :HOLDS NUMBER OF TIMEOUTS FOR CARRY PROPAGATION TEST  
HDWFLAG: .WORD C :FLAGS APT SPECIAL HARDWARE FOR T47 & T50  
READON: .WORD 20000 :READ ONLY BIT IN MMRO  
\$MXCNT: .WORD 200 :HOLD MAX. NUMBER OF LOOP ITERATIONS

```
001346 000000
001350   136   103   015 $TBIT: .WORD 0      ;'T' BIT STATE INDICATOR
001353   012   000   $CNTLC: .ASCIZ ;'^C/<15><12> ;CONTROL C

001356                                PARTAB: .EVEN ;THIS IS THE TABLE OF THE FIRST PAR OR PDR
                                        ;OF EACH GROUP. THEY ARE USED FOR THE DUAL
                                        ;ADDRESSING TEST.

001356 172200                                .WORD 172200 ;SIPDRO
001360 172240                                .WORD 172240 ;SIPARO
001362 172300                                .WORD 172300 ;KIPDRO
001364 172340                                .WORD 172340 ;KIPARO
001366 177600                                .WORD 177600 ;UIPDRO
001370 177640                                .WORD 177640 ;UIPARO
```



.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

\$ERRTB:

885	001372				
886					
887	001372	041304	EM1		:UNEXPECTED CPU TRAP TO LOC. 004
888	001374	044647	DH1		:OLD PC OLD PSW R6 WAS CPUERR TESTNO ERRORPC
889	001376	047606	DT1		:TRAPPC,TRAPPS,WASR6,CPUERR,TESTNO,\$ERRPC,0
890	001400	050360	DF1		:0,0,0,0,0
891					
892					
893	001402	041344	EM2		:UNEXPECTED MEM. MGMT. TRAP TO LOC. 250
894	001404	044727	DH2		:OLD PC OLD PSW R6 WAS SR0 SR1 SR2 TESTNO ERR
895	001406	047624	DT2		:TRAPPC,TRAPPS,WASR6,WASSR0,WASSR1,WASSR2,TESTNO,\$ERRPC,0
896	001410	050365	DF2		:0,0,0,0,0,0,0,0
897					
898					
899	001412	041413	EM3		:PRIORITY BITS SET WRONG IN PSW
900	001414	045026	DH3		:WROTE READ TESTNO ERRORPC
901	001416	047646	DT3		;\$REG0,\$REG1,TESTNO,\$ERRPC,0
902	001420	050375	DF3		:0,0,0,0
903					
904					
905	001422	041452	EM4		:MODE BITS SET WRONG IN PSW
906	001424	045026	DH3		:WROTE READ TESTNO ERRORPC
907	001426	047646	DT3		;\$REG0,\$REG1,TESTNO,\$ERRPC,0
908	001430	050375	DF3		:0,0,0,0
909					
910					
911	001432	041505	EM5		:DUAL ADDRESSING BETWEEN HI&LO BYTES OF PSW
912	001434	045026	DH3		:WROTE READ TESTNO ERRORPC
913	001436	047646	DT3		;\$REG0,\$REG1,TESTNO,\$ERRPC,0
914	001440	050375	DF3		:0,0,0,0
915					
916					
917	001442	041560	EM6		:KERNEL R6 CHANGED BY WRITING SUPERVISOR/USER R6
918	001444	045026	DH3		:WROTE READ TESTNO ERRORPC
919	001446	047646	DT3		;\$REG0,\$REG1,TESTNO,\$ERRPC,0
920	001450	050375	DF3		:0,0,0,0
921					
922					
923	001452	041643	EM7		:A MEMORY MGMT. REG. TIMED OUT
924	001454	045066	DH7		:ADDRESS TESTNO ERRORPC
925	001456	047660	DT7		;\$REG0,TESTNO,\$ERRPC,0
926	001460	050401	DF7		:0,0,0
927					
928					
929	001462	041701	EM10		:SUMMARY OF MEM. MGMT. REG. TIMEOUTS
930	001464	045116	DH10		:REGISTER-ADDRS NUM. OF

931				:AND-ED OR-ED TIMOUTS TESTNO ERRORPC
932	001466	047670	DT10	:ADRAND,ADDROR,TONUM,TESTNO,\$ERRPC,0
933	001470	050404	DF10	:0,0,1,0,0
934				
935			:*ITEM 11	
936	001472	041745	EM11	:MEM. MGMT. REG. WOULD NOT CLEAR
937	001474	045216	DH11	:REGISTR READ READ-(BINARY)
938				:ADDRESS (OCTAL) 5432109876543210 TESTNO ERRORPC
939	001476	047704	DT11	:\$REG0,\$REG1,\$REG1,TESTNO,\$ERRPC,0
940	001500	050411	DF11	:0,0,2,0,0
941				
942			:*ITEM 12	
943	001502	042005	EM12	:MEM. MGMT. REG. BITS NOT SET CORRECTLY
944	001504	045336	DH12	:REGISTR WROTE READ READ
945				:ADDRESS (OCTAL) (OCTAL) (BINARY) TESTNO ERRORPC
946	001506	047720	DT12	:\$REG0,\$REG1,\$REG2,\$REG2,TESTNO,\$ERRPC,0
947	001510	050416	DF12	:0,0,0,2,0,0
948				
949			:*ITEM 13	
950	001512	042054	EM13	:SRO EFFECTED BY WRITE TO PSW
951	001514	045476	DH13	:READ TESTNO ERRORPC
952	001516	047660	DT7	:\$REG0,TESTNO,\$ERRPC,0
953	001520	050401	DF7	:0,0,0
954				
955			:*ITEM 14	
956	001522	042111	EM14	:MMR1 DID NOT TRACK PROPERLY
957	001524	045526	DH14	:EXPECTD (MMR1) TESTNO ERRORPC
958	001526	047736	DT14	:\$REG3,\$REG1,TESTNO,\$ERRPC,0
959	001530	050375	DF3	:0,0,0,0
960				
961			:*ITEM 15	
962	001532	042145	EM15	:MMR3 IS HOLDING THE WRONG DATA
963	001534	045566	DH15	:LOADED (MMR3) TESTNO ERRORPC
964	001536	047750	DT15	:\$REG2,\$REG1,TESTNO,\$ERRPC,0
965	001540	050375	DF3	:0,0,0,0
966				
967			:*ITEM 16	
968	001542	042204	EM16	:DUAL ADDRESSING BETWEEN PAR-PDR GROUPS
969	001544	045626	DH16	:INDEX INDEX PAR-PDR
970				:EXPECTD RECEIVD ADDRREAD TESTNO ERRORPC
971	001546	047762	DT16	:\$REG0,\$REG1,\$REG2,TESTNO,\$ERRPC,0
972	001550	050360	DF1	:0,0,0,0,0
973				
974			:*ITEM 17	
975	001552	042253	EM17	:PHYS. ADDR. FORMED READ WRONG IN MAINT. MODE
976	001554	045727	DH17	:VIR ADDR KIPAR4 GDDATA BADDATA TESTNO ERRORPC
977	001556	047776	DT17	:\$REG0,KIPAR4,\$REG1,\$REG2,TESTNO,\$ERRPC,0
978	001560	050365	DF2	:0,0,0,0,0,0
979				
980			:*ITEM 20	
981	001562	042331	EM20	:PHYS. ADDR. FORMED READ WRONG IN RELOCATE MODE
982	001564	046010	DH20	:PHYSICL PAR 4 PAR 5
983				:ADDRESS VBA VBA PAR 4 PAR 5 PSW TESTNO ERR
984	001566	050014	DT20	:PBALO,VIRT1,VIRT2,\$REG4,\$REG5,\$TMP0,TESTNO,\$ERRPC,0
985	001570	050432	DF20	:3,0,0,0,0,0,0,0
986				
987			:*ITEM 21	





Address	Offset	Value	Register	Description
1102	002012	043621	EM43	:18-BIT MAPPING POSSIBLE HOLE AT TOP OF MEMORY :STARTADR TESTNO ERRORPC :\$TMP1,TESTNO,\$ERRPC,0 :4,0,0
1103	002014	047153	DH43	
1104	002016	050264	DT43	
1105	002020	050462	DF43	
1106				
1107			:*ITEM 44	
1108	002022	043677	EM44	:NO TRAP THRU ERRVEC, AT 18-BIT ADDR. 760000 :TESTNO ERRORPC :TESTNO,\$ERRPC,0 :0,0
1109	002024	046314	DH24	
1110	002026	050072	DT24	
1111	002030	050442	DF24	
1112				
1113			:*ITEM 45	
1114	002032	043752	EM45	:DIDN'T GET WRAP AROUND TO ADDRESS ZERO :DATA TESTNO ERRORPC :\$REG1,TESTNO,\$ERRPC,0 :0,0,0
1115	002034	047203	DH45	
1116	002036	050274	DT45	
1117	002040	050401	DF7	
1118				
1119			:*ITEM 46	
1120	002042	044021	EM46	:NO TRAP THRU ERRVEC, ON NON-EXISTANT ADDR. :NEADDR TESTNO ERRORPC :\$REG0,TESTNO,\$ERRPC,0 :0,0,0
1121	002044	047233	DH46	
1122	002046	047660	DT7	
1123	002050	050401	DF7	
1124				
1125			:*ITEM 47	
1126	002052	044074	EM47	:PREMATURE END OF MEMORY FOUND :KIPAR4 LASTBK TESTNO ERRORPC :KIPAR4,\$LSTBK,TESTNO,\$ERRPC,0 :0,0,0,0
1127	002054	047262	DH47	
1128	002056	050304	DT47	
1129	002060	050365	DF2	
1130				
1131			:*ITEM 50	
1132	002062	044132	EM50	:22-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM :STARTBK FINISHBK TESTNO ERRORPC :\$TMP1,\$TMP2,TESTNO,\$ERRPC,0 :0,0,0,0
1133	002064	047010	DH41	
1134	002066	050236	DT41	
1135	002070	050365	DF2	
1136				
1137			:*ITEM 51	
1138	002072	044213	EM51	:BAD RELOCATION, CARRY PROPAGATION 22-BIT MAPPING :PATTERN DATA ADDRESS :LOADED FETCHED INTENDED TESTNO ERRORPC :\$REG2,\$REG3,\$REG0,TESTNO,\$ERRPC,0 :0,0,0,0,0
1139	002074	047051	DH42	
1140				
1141	002076	050250	DT42	
1142	002100	050365	DF2	
1143				
1144			:*ITEM 52	
1145	002102	044273	EM52	:22-BIT MAPPING POSSIBLE HOLE AT TOP OF MEMORY :STARTADR TESTNO ERRORPC :\$TMP1,TESTNO,\$ERRPC,0 :0,0,0
1146	002104	047153	DH43	
1147	002106	050264	DT43	
1148	002110	050365	DF2	
1149				
1150			:*ITEM 53	
1151	002112	044351	EM53	:DID NOT GET UNIBUS ADDRESS :PATTERN DATA ADDRESS :LOADED FETCHED INTENDED TESTNO ERRORPC :\$REG2,\$REG3,\$REG0,TESTNO,\$ERRPC,0 :0,0,0,0,0
1152	002114	047051	DH42	
1153				
1154	002116	050250	DT42	
1155	002120	050365	DF2	
1156				
1157			:*ITEM 54	
1158	002122	044404	EM54	:W-BIT DID NOT GET SET IN PDR



Line	Address	Code	Register	Description
1159	002124	047326	DH54	:PDR VIRTUAL
1160				:TESTED ADDRESS TESTNO ERRORPC
1161	002126	050316	DT54	:\$REG5,\$REG3,TESTNO,\$ERRPC,0
1162	002130	050375	DF3	:0,0,0,0
1163				
1164				:*ITEM 55
1165	002132	044441	EM55	:W-BIT SET IN MORE THAN ONE PDR
1166	002134	047406	DH55	:PDR IN PDR VIRTUAL
1167				:ERROR TESTED ADDRESS TESTNO ERRORPC
1168	002136	050330	DT55	:\$REG0,\$REG5,\$REG3,TESTNO,\$ERRPC,0
1169	002140	050360	DF1	:0,0,0,0,0
1170				
1171				:*ITEM 56
1172	002142	044500	EM56	:W-BIT NOT CLEARED BY WRITING TO PDR
1173	002144	047506	DH56	:PDR TESTNO ERRORPC
1174	002146	050344	DT56	:\$REG5,TESTNO,\$ERRPC,0
1175	002150	050401	DF7	:0,0,0
1176				
1177				:*ITEM 57
1178	002152	044544	EM57	:W-BIT GOT SET DURING ODD ADDR. ABORT
1179	002154	047536	DH57	:PDR WAS EXPECTD TESTNO ERRORPC
1180	002156	047750	DT15	:\$REG2,\$REG1,TESTNO,\$ERRPC,0
1181	002160	050375	DF3	:0,0,0,0
1182				
1183				:*ITEM 60
1184	002162	044611	EM60	:NO APT SPECIAL HARDWARE FOUND
1185	002164	047576	DH60	:ERRORPC
1186	002166	050354	DT60	:\$ERRPC,0
1187	002170	050360	DF1	:0

```

1189
1190 .SBTTL ***** SUBROUTINES UNIQUE TO THIS PROGRAM *****
1191
1192 .SBTTL TURN OFF T-BIT AND SAVE CURRENT PSW
1193 :*****
1194 :*
1195 :* THIS SUBROUTINE IS USED TO TURN OFF THE TRACE TRAP BIT IN
1196 :* THE PSW IF IT IS ON. THE PROCESSOR STATUS IS SAVED IN
1197 :* 'TBITPS' SO THAT THE PSW CAN BE RESTORED TO ITS PREVIOUS
1198 :* CONDITION WHEN CONDITIONS WARRANT T-BIT TRAPPING.
1199 :*
1200 :*****
1201 002172 036727 175600 000020 TOFF: BIT PSW,#TBIT ;IS THE T-BIT SET IN THE PSW?
1202 002200 001411 BEQ 1$ ;EXIT IF NO
1203 002202 016746 175570 MOV PSW,-(SP) ;PUSH PRESENT PSW ON THE STACK
1204 002206 011667 177060 MOV (SP),TBITPS ;ALSO SAVE IT IN 'TBITPS' FOR
1205 ;RESTORING LATER
1206 002212 042716 000020 BIC #TBIT,(SP) ;CLEAR THE T-BIT (BIT 4) IN THE PSW
1207 002216 012746 002224 MOV #1$,-(SP) ;PUSH PC OF 'RTS' ON STACK
1208 002222 000006 RTT ;'RETURN' TO 1$ WITH T-BIT OFF
1209 002224 000207 1$: RTS PC ;RETURN TO PROGRAM
1210
1211 .SBTTL TURN ON T-BIT AND RESTORE PREVIOUS PSW
1212 :*****
1213 :*
1214 :* THIS SUBROUTINE IS USED TO RESTORE THE PROCESSOR STATUS
1215 :* TO ITS PREVIOUS CONDITION BY RESTORING THE 'T-BIT PSW'
1216 :* SAVED BY THE 'TOFF' SUBROUTINE IN THE 'TBITPS' LOCATION.
1217 :*
1218 :*****
1219 002226 036727 177040 000020 TON: BIT TBITPS,#TBIT ;WAS T-BIT ON IN THE PREVIOUS PSW?
1220 002234 001410 BEQ 1$ ;EXIT IF NO
1221 002236 016746 177030 MOV TBITPS,-(SP) ;PUSH PREVIOUS PSW ON THE STACK
1222 002242 012767 000340 177022 MOV #340,TBITPS ;RESET THE 'TBITPS' LOCATION
1223 002250 012746 002256 MOV #1$,-(SP) ;PUSH PC OF 'RTS' ON STACK
1224 002254 000006 RTT ;'RETURN' TO 1$ WITH T-BIT RESTORED
1225 002256 000207 1$: RTS PC ;RETURN TO PROGRAM
1226
1227 .SBTTL CLEAR 16 PAR'S OR PDR'S STARTING FROM ADDRESS IN R5
1228 :*****
1229 :* THIS ROUTINE CLEARS 16 CONSECUTIVE MEMORY MANAGEMENT
1230 :* REGISTERS STARTING WITH THE REGISTER POINTED TO BY R5.
1231 :* IT SAVES R0 ON THE STACK, AND LOADS A COUNT IN R0,
1232 :* CLEARS THE PAR'S OR PDR'S, AND THEN RESTORES R0
1233 :* BEFORE RETURNING.
1234 :*
1235 :* CALL: MOV #KIPAR0,R5 ;PUT ADDRESS OF FIRST KERNAL PAR INTO R5
1236 :* JSR PC,CLRREG ;CLEAR ALL THE KERNAL PAR'S
1237 :*****
1238
1239 002260 010046 CLRREG: MOV R0,-(KSP) ;SAVE R0 ON THE STACK
1240 002262 012700 000020 MOV #20,R0 ;PUT COUNT IN R0
1241 002266 005025 1$: CLR (R5)+ ;CLEAR PAR OR PDR POINTED TO BY R5
1242 002270 077002 SOB R0,1$ ;BRANCH BACK 15 DECIMAL TIMES
1243 002272 012600 MOV (KSP)+,R0 ;POP R0 FROM STACK
1244 002274 000207 RTS PC ;RETURN TO TEST
1245

```

1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253 002276  
1254 002276 005067 177004  
1255 002302 005067 177010  
1256 002306 005067 177000  
1257 002312 012700 177777  
1258 002316 010067 176766  
1259 002322 010067 176772  
1260 002326 010067 176762  
1261 002332 000207  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272 002334  
1273 002334 012667 176766  
1274 002340 050067 176752  
1275 002344 030067 176750  
1276 002350 050167 176732  
1277 002354 030167 176730  
1278 002360 105767 176517  
1279 002364 001002  
1280 002366 104034  
1281 002370 000401  
1282 002372 104035  
1283 002374 000177 176726  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295 002400  
1296 002400 012667 176722  
1297 002404 050067 176706  
1298 002410 030067 176704  
1299 002414 050167 176672  
1300 002420 030167 176670  
1301 002424 050267 176656  
1302 002430 030267 176654

```
.SBTTL CLEANUP LOCATIONS THAT HOLD LOGICAL 'AND' AND 'OR'
:*****
:* THIS SUBROUTINE IS USED TO INITIALIZE ALL LOCATIONS THAT
:* HOLD THE 'LOGICAL AND' AND 'LOGICAL OR' OF THE DATA AND
:* ADDRESSES THAT FAILED DURING THE EXECUTION OF A TEST.
:*****
CLEANUP:
CLR DATAOR ;LOCATION OF LOGICAL OR OF BAD DATA
CLR ADDROR ;LOCATION OF LOGICAL OR OF ADDRESS
CLR PATTOR ;LOCATION OF LOGICAL OR OF PATTERN LOADED
MOV #-1,R0 ;LOAD -1 INTO R0 TO INITIALIZE
MOV R0,DATAND ;LOCATION OF LOGICAL AND OF BAD DATA
MOV R0,ADRAND ;LOCATION OF LOGICAL AND OF ADDRESS
MOV R0,PATAND ;LOCATION OF LOGICAL AND OF PATTERN LOADED
RTS PC ;RETURN TO TEST
```

```
.SBTTL DUAL ADDRESSING WHEN LOADING A PAR OR PDR
:*****
:* THIS SUBROUTINE WILL LOG AND REPORT ALL DUAL ADDRESSING ERRORS
:* FOUND IN BOTH PAR'S AND PDR'S. A 'LOGICAL OR' AND A 'LOGICAL
:* AND' OF THE WRITTEN ADDRESSES WILL BE MAINTAINED IN 'ADDROR',
:* AND 'ADRAND'. THE LOG ON THE ADDITIONAL OR FAILING ADDRESSES
:* WILL BE MAINTAINED IN 'DATAOR' AND 'DATAND'.
:*****
```

```
DUALADR:
MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF LOOP
BIS R0,ADDROR ;LOGICAL OR OF WRITTEN ADDRESS
BIT R0,ADRAND ;LOGICAL AND OF WRITTEN ADDRESS
BIS R1,DATAOR ;LOGICAL OR OF DUALED DATA
BIT R1,DATAND ;LOGICAL AND OF DUALED DATA
TSTB $ERFLG ;SEE IF THIS IS FIRST ERROR
BNE 1$ ;BRANCH IF NOT FIRST ERROR
ERROR +34
BR 2$ ;BRANCH TO EXIT
1$: ERROR +35
2$: JMP @OLDPC ;RETURN TO TEST
```

```
.SBTTL COUNT PATTERN ERRORS IN PAR'S OR PDR'S
:*****
:* THIS SUBROUTINE IS USED TO LOG AND REPORT THE COUNT PATTERN
:* ERRORS OCCURRING WHEN TESTING THE PAR'S AND PDR'S. THE
:* 'LOGICAL OR' AND 'LOGICAL AND' OF VARIOUS DATA WILL BE
:* MAINTAINED A FOLLOWS:
:* 1. ADDRESSES OF FAILED REGISTERS IN 'ADDROR' AND 'ADRAND'
:* 2. DATA FETCHED FROM REGISTERS IN 'DATAOR' AND 'DATAND'
:* 3. PATTERN LOADED INTO REGISTERS IN 'PATTOR' AND 'PATAND'.
:*****
```

```
PARCOUNT:
MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF LOOP
BIS R0,ADDROR ;LOGICAL OR OF FAILING ADDRESS
BIT R0,ADRAND ;LOGICAL AND OF FAILING ADDRESS
BIS R1,PATTOR ;LOGICAL OR OF PATTERN LOADED
BIT R1,PATAND ;LOGICAL AND OF PATTERN LOADED
BIS R2,DATAOR ;LOGICAL OR OF DATA FETCHED
BIT R2,DATAND ;LOGICAL AND OF DATA FETCHED
```

1303 002434 105767 176443  
 1304 002440 001002  
 1305 002442 104036  
 1306 002444 000401  
 1307 002446 104037  
 1308 002450 000177 176652  
 1309  
 1310

TSTB \$ERFLG ;SEE IF THIS IS THE FIRST ERROR  
 BNE 1\$ ;BRANCH IF NOT FIRST ERROR  
 ERROR +36  
 BR 2\$ ;BRANCH TO EXIT  
 ERROR +37  
 1\$:  
 2\$: JMP @OLDPC ;RETURN TO TEST

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

002454 010046  
 002456 010146  
 002460 010246  
 002462 010346  
 002464 013746 000004  
 002470 013746 000006  
 002474 010600

\$SIZE: MOV R0,-(SP) ;:SAVE R0 ON THE STACK  
 MOV R1,-(SP) ;:SAVE R1 ON THE STACK  
 MOV R2,-(SP) ;:SAVE R2 ON THE STACK  
 MOV R3,-(SP) ;:SAVE R3 ON THE STACK  
 MOV @ERRVEC,-(SP) ;:SAVE PRESENT ERROR VECTOR PS & PC  
 MOV @ERRVEC+2,-(SP)  
 MOV SP,R0 ;:SAVE THE STACK POINTER

;;SET THE ERRVEC PS TO THE PRESENT PS  
 TRAP ;:PUSH OLD PSW AND PC ON STACK  
 MOV (SP)+,@ERRVEC+2 ;:SAVE THE PSW IN @ERRVEC+2  
 MOV #3776,R1 ;:SETUP ADDRESS  
 TSTB (PC)+ ;:USE MEMORY MANAGEMENT?  
 \$KT11: .WORD 200 ;:SET TO USE MEMORY MANAGEMENT  
 BPL \$CORE ;:BR IF NO  
 MOV # \$SKTNEX,@ERRVEC ;:SET FOR TIMEOUT  
 TST @SR0 ;:KT11 ARE YOU THERE?  
 BIS #10000,\$KT11 ;:YES--SET KT11 KEY  
 MOV @MGMERR,@ERRVEC ;:SET IN CASE OF ERROR  
 MOV #340,@ERRVEC+2  
 CLR -(SP) ;:INITIALIZE FOR 'PAR' LOADING  
 MOV #KIPAR0,R2 ;:ADDRESS OF FIRST 'PAR'  
 MOV #\*D8,R3 ;:LOAD EIGHT 'PAR.'S' AND EIGHT 'PDR.'S'  
 1\$: MOV #77406,-40(R2) ;:PDR = 4K, UP, READ/WRITE  
 MOV (SP),(R2)+ ;:LOAD 'PAR'  
 ADD #200,(SP) ;:UPDATE FOR NEXT 'PAR'  
 SOB R3,1\$ ;:LOOP UNTIL ALL EIGHT ARE LOADED  
 MOV #177600,-(R2) ;:SETUP KIPAR7 FOR I/O  
 CLR -(R2) ;:SETUP KIPAR6 FOR TESTING.  
 MOV #2\$,@ERRVEC ;:CATCH TIMEOUT IF NO SR3  
 MOV #20,@SR3 ;:ENABLE 22 BIT MODE  
 BR 3\$ ;:THIS PDP-11 HAS A SR3 REGISTER  
 2\$: CMP (SP)+,(SP)+ ;:CLEAN OFF TE STAC--NO SR3  
 3\$: INC @SR0 ;:TURN ON MEMORY MANAGEMENT  
 MOV # \$SKTOUT,@ERRVEC ;:SET FOR TIME OUT  
 4\$: TST @143776 ;:TRAP ON NON-EX-MEM  
 ADD #40,(R2) ;:MAKE A 1K STFP  
 CMP @KIPAR7,(R2) ;:LAST ONE?  
 BHI 4\$ ;:NO--TRY IT

002476 104400  
 002500 012637 000006  
 002504 012701 003776  
 002510 105727  
 002512 000200  
 002514 100070  
 002516 012737 002670 000004  
 002524 005737 177572  
 002530 052767 100000 177754  
 002536 012737 003122 000250  
 002544 012737 000340 000252  
 002552 005046  
 002554 012702 172340  
 002560 012703 000010  
 002564 012762 077406 177740 1\$:  
 002572 011622  
 002574 062716 000200  
 002600 077307  
 002602 012742 177600  
 002606 005042  
 002610 012737 002626 000004  
 002616 012737 000020 172516  
 002624 000401  
 002626 022626 2\$:  
 002630 005237 177572 3\$:  
 002634 012737 002660 000004  
 002642 005737 143776 4\$:  
 002646 062712 000040  
 002652 023712 172356  
 002656 101371

```

002660 011202          SKTOUT: MOV      (R2),R2          ;;GET LAST BANK+1
002662 005037 177572          CLR      @#SR0          ;;TURN OFF MEMORY MANAGEMENT
002666 000421          BR        $SIZEX
002670 042767 100000 177614 SKTNEX: BIC      #100000,$KT11      ;;KT11 NON-EXISTENT
002676 012737 002726 000004 SCORE:  MOV      #SCROUT,@#ERRVEC ;;SET FOR TIMEOUT
002704 005002          CLR      R2          ;;SET UP BANK
002706 062701 004000          1$:  ADD      #4000,R1      ;;INCREMENT BY 1K
002712 062702 000040          ADD      #40,R2         ;;1K STEP
002716 005711          TST      (R1)         ;;TRAP ON TIME OUT
002720 022701 177776          CMP      #177776,R1    ;;LAST ONE
002724 001370          BNE     1$          ;;NO--TRY AGAIN
002726 162701 004000          SCROUT: SUB     #4000,R1
002732 162702 000040          $SIZEX: SUB    #40,R2      ;;DROP BACK
002736 012737 002754 000004          MOV      #2$,@#ERRVEC ;;SET FOR TIMEOUT
002744 012701 020000          MOV      #20000,R1    ;;FIRST ADDRESS
002750 005721          1$:  TST      (R1)+      ;;TEST AND STEP TO NEXT ADDRESS
002752 000776          BR        1$          ;;TRY ANOTHER
002754 162701 000002          2$:  SUB      #2,R1      ;;DROP BACK
002760 010006          MOV      R0,SP        ;;RESTORE THE STACK
002762 012637 000006          MOV      (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
002766 012637 000004          MOV      (SP)+,@#ERRVEC
002772 010167 000022          MOV      R1,$LSTAD    ;;LAST ADDRESS
002776 010267 000020          MOV      R2,$LSTBK    ;;LAST BANK
003002 012603          MOV      (SP)+,R3     ;;RESTORE R3
003004 012602          MOV      (SP)+,R2     ;;RESTORE R2
003006 012601          MOV      (SP)+,R1     ;;RESTORE R1
003010 012600          MOV      (SP)+,R0     ;;RESTORE R0
003012 005067 174750          CLR      CPUERR      ;;CLEAR THE ERROR REGISTER
003016 000207          RTS      PC
003020 000000          $LSTAD: .WORD    0      ;;CONTAINS THE LAST ADDRESS
003022 000000          $LSTBK: .WORD    0      ;;CONTAINS THE LAST BANK

1311
1312          .SBTTI ***** TRAP HANDLING ROUTINES *****
1313
1314          .SBTTL CPU TRAP HANDLER ROUTINE
1315          ;*****
1316          ;*
1317          ;* THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS THRU
1318          ;* 'ERRVEC' (LOC. 004). IF THIS SUBROUTINE IS ENTERED BY A
1319          ;* SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A HALT IS
1320          ;* EXECUTED.
1321          ;*
1322          ;*****
1323 003024 005227          TIMERR: INC      (PC)+      ;;MAKE FLAG ZERO IF FIRST TIME THRU
1324 003026 177777          TIMFLG: .WORD    -1      ;;NEGATIVE ONE FOR 'HAVE ENTERED' FLAG
1325 003030 001401          BEQ     1$          ;;BRANCH IF FIRST TIME IN
1326 003032 000000          HALT          ;;STOP! - I'VE ENTERED THIS ROUTINE
1327          ;;A SECOND TIME BEFORE I FINISHED
1328          ;;REPORTING THE FIRST ERROR. THE
1329          ;;SECOND ENTRY ADDRESS SHOULD BE ON
1330          ;;THE KERNEL STACK.
1331 003034 012667 176220          1$:  MOV      (KSP)+,TRAPPC ;;SAVE PC+2 AT TIME OF ABORT
1332 003040 012667 176216          MOV      (KSP)+,TRAPPS ;;SAVE PS AT TIME OF ABORT
1333 003044 016767 174716 176260          MOV      CPUERR,PCPUER ;;SAVE CPU ERROR REGISTER
1334 003052 010667 176200          MOV      KSP,WASR6    ;;SAVE STACK POINTER VALUE
1335 003056 005767 176252          TST      CPUEXP      ;;SEE IF ANY ERROR CONDITION WAS EXPECTED
1336 003062 001404          BEQ     2$          ;;BRANCH IF NO TRAP EXPECTED
    
```



```

1337 003064 026767 176242 176242      CMP      PCPUER, CPUEXP      ;SEE IF EXPECTED CONDITION OCCURED
1338 003072 001401                                BEQ      3$                  ;BRANCH IF ERROR CODES MATCH
1339 003074 104001                                2$:     ERROR      +1        ;UNEXPECTED TRAP OR ABORT TO LOC. 4
1340 003076 005067 174664                                3$:     CLR      CPUERR      ;CLEAR CPU ERROR REGISTER
1341 003102 012767 177777 177716      MOV      #-1, TIMFLG        ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
1342 003110 016746 176146      MOV      TRAPPS, -(KSP)     ;PUT PC & PS OF TRAP ON STACK
1343 003114 016746 176140      MOV      TRAPPC, -(KSP)
1344 003120 000006                                RTT                          ;RETURN FROM INTERRUPT OR ABORT
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
    
```

.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.
:
:*****
    
```

```

1356 003122 005227      MGMERR: INC      (PC)+        ;MAKE FLAG ZERO IF FIRST TIME THRU
1357 003124 177777      MGMFLG: .WORD   -1          ;NEGATIVE ONE FOR 'HAVE ENTERED' FLAG
1358 003126 001401      BEQ      1$              ;BRANCH IF FIRST TIME IN
1359 003130 000000      HALT                    ;STOP! - I'VE ENTERED THIS ROUTINE
1360                                ;A SECOND TIME BEFORE I FINISHED
1361                                ;REPORTING THE FIRST ERROR. THE
1362                                ;SECOND ENTRY ADDRESS SHOULD BE ON
1363                                ;THE KERNEL STACK.
1364 003132 012667 176122      1$:     MOV      (KSP)+, TRAPPC ;SAVE PC+2 AT TIME OF ABORT
1365 003136 012667 176120      MOV      (KSP)+, TRAPPS    ;SAVE PS AT TIME OF ABORT
1366 003142 010667 176110      MOV      KSP, WASR6        ;SAVE STACK POINTER VALUE
1367 003146 016767 174420 176110      MOV      SR0, WASSR0       ;SAVE CONTENTS OF KT STATUS REG. 0
1368 003154 016767 174414 176104      MOV      SR1, WASSR1       ;SAVE CONTENTS OF KT STATUS REG. 1
1369 003162 016767 174410 176100      MOV      SR2, WASSR2       ;SAVE CONTENTS OF KT STATUS REG. 2
1370 003170 042767 160000 174374      BIC      #160000, SR0      ;CLEAR ERROR BITS IN STATUS REG 0
1371 003176 104002      ERROR      +2            ;UNEXPECTED TRAP OR ABORT TO LOC. 250
1372 003200 012767 177777 177716      MOV      #-1, MGMFLG      ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
1373 003206 016746 176050      MOV      TRAPPS, -(KSP)   ;PUT PC & PS OF TRAP ON STACK
1374 003212 016746 176042      MOV      TRAPPC, -(KSP)
1375 003216 000006                                RTT                          ;RETURN FROM INTERRUPT OR ABORT.
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
    
```

.SBTTL CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS

```

:*****
:
: THIS SUBROUTINE IS USED TO FORM AN 22-BIT PHYSICAL ADDRESS
: (PBA) FROM THE 16-BIT VIRTUAL ADDRESS (VBA) AND THE APPROPRIATE
: PAGE ADDRESS REGISTER (PAR). THE SAME METHOD USED BY THE MEMORY
: MANAGEMENT LOGIC IS USED. VBA <15:13> SELECTS WHICH PAR/PDR
: IS TO BE USED, VBA <5:0>+PBA <5:0>, AND VBA <12:6> IS ADDED
: TO PAR <15:00> TO GIVE PBA <21:6>. BITS <21:16> OF THE
: PHYSICAL ADDRESS ARE LEFT IN LOC. 'PBAHI' AND BITS <15:00>
: ARE LEFT IN LOC. 'PBALO'. THE PSW'S 'CURRENT MODE' BITS
: ARE USED TO SELECT THE KERNEL, SUPERVISOR OR USER PAR/PDR'S.
: THE ROUTINE IS ENTERED WITH LOC. 'VIRT1' CONTAINING THE 16-BIT
: VIRTUAL ADDRESS.
:
:*****
    
```

```

1394 003220 012702 172340 FORMPA: MOV #KIPAR0,R2 ;LOAD ADDRESS OF FIRST KERNEL PAR IN R2
1395 003224 032767 140000 174544 BIT #140000,PSW ;IN USER MODE?
1396 003232 001403 BEQ 1$ ;BRANCH IF NO
1397 003234 012702 177640 MOV #UIPAR0,R2 ;LOAD ADDRESS OF FIRST USER PAR IN R2
1398 003240 000406 BR 2$ ;BRANCH WHEN DONE
1399 003242 032767 040000 174526 1$: BIT #40000,PSW ;IN SUPERVISOR MODE?
1400 003250 001402 BEQ 2$ ;BRANCH IF NO
1401 003252 012702 172240 MOV #SIPAR0,R2 ;LOAD ADDRESS OF FIRST SUPERVISOR PAR IN R2
1402 003256 016700 176014 2$: MOV VIRT1,R0 ;LOAD VIRTUAL ADDR. (VBA) INTO R0
1403 003262 072027 177764 ASH #-14,R0 ;GET BITS <15:13> DOWN TO BITS <3:1>
1404 003266 042700 177761 BIC #177761,R0 ;MASK OF ALL BITS BUT BITS <3:1>
1405 003272 060002 ADD R0,R2 ;ADD OFFSET TO BASE PAR ADDRESS
1406 003274 011200 MOV (R2),R0 ;GET BITS <15:00> FROM APPROPRIATE PAR
1407 003276 010002 MOV R0,R2 ;COPY PAR BITS <15:00> INTO R2
1408 003300 016767 175772 175774 MOV VIRT1,PBALO ;PUT VIRTUAL ADDR. IN LOC. 'PBALO'
1409 003306 042767 160000 175766 BIC #160000,PBALO ;CLEAR OFF BITS <15:13> OF ORIGINAL VBA
1410 003314 072227 177766 ASH #-12,R2 ;GET PAR <15:00> DOWN TO BITS <1:0> OF R2
1411 003320 042702 177774 BIC #177774,R2 ;CLEAR OFF ALL BITS BUT BITS <1:0>
1412 003324 072027 000006 ASH #6,R0 ;SHIFT PAR<9:0> TO <15:6> OF R0
1413 003330 042700 000077 BIC #77,R0 ;CLEAR BITS <5:0> OF R0
1414 003334 060007 175742 ADD R0,PBALO ;IN EFFECT, ADD VBA<12:0> TO PAR<9:0>
1415 ;(PAR<9:0> IN BITS <15:6> OF R0)
1416 003340 005502 ADC R2 ;ADD ANY CARRY TO R2
1417 003342 010267 175736 MOV R2,PBAHI ;PUT BITS <21:16> OF PHYSICAL ADDR. IN PBAHI
1418 003346 000207 RTS PC ;RETURN TO PROGRAM
  
```

1420  
 1421  
 1422 020000  
 1423  
 1424 020000

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

020000 012706 001100  
 020004 005026  
 020006 022706 001140  
 020012 001374  
 020014 012706 001100  
  
 020020 012737 035174 000020  
 020026 012737 000340 000022  
 020034 012737 035366 000030  
 020042 012737 000340 000032  
 020050 012737 040770 000034  
 020056 012737 000340 000036  
 020064 012737 041056 000024  
 020072 012737 000340 000026  
 020100 016767 014640 014630  
 020106 005067 161100  
 020112 112767 000001 160775  
  
 020120 012737 035162 000014  
 020126 012737 000340 000016  
 020134 012767 000002 015020  
 020142 012737 020170 000010  
 020150 005046  
 020152 012746 020160  
 020156 000006  
 020160 012767 000006 014774 64\$:  
 020166 000402  
 020170 062706 000010 65\$:  
 020174 012737 000012 000010 66\$:  
 020202 005067 161140  
 020206 012767 020206 160672  
 020214 012767 020214 160666  
  
 020222 013746 000004  
 020226 012737 020262 000004  
 020234 012767 177570 160676  
 020242 012767 177570 160672  
 020250 022777 177777 160662  
 020256 001012  
  
 020260 000403  
 020262 012716 020270 67\$:  
 020266 000002  
 020270 012767 000176 160642 68\$:  
 020276 012767 000174 160636  
 020304 012637 000004 69\$:  
 020310 005067 160716  
 020314 132767 000200 160723

```

START:
.SBTTL INITIALIZE THE COMMON TAGS
;;CLEAR THE COMMON TAGS ($CMTAG) AREA
MOV #CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
CLR (R6)+ ;;CLEAR MEMORY LOCATION
CMP #SWR,R6 ;;DONE?
BNE -6 ;;LOOP BACK IF NO
MOV #STACK,SP ;;SETUP THE STACK POINTER
;;INITIALIZE A FEW VECTORS
MOV #SCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
MOV #340,@IOTVEC+2 ;;LEVEL 7
MOV #ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
MOV #340,@EMTVEC+2 ;;LEVEL 7
MOV #STRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
MOV #340,@TRAPVEC+2;LEVEL 7
MOV #SPURDN,@PWRVEC ;;POWER FAILURE VECTOR
MOV #340,@PWRVEC+2 ;;LEVEL 7
MOV $ENDCT,$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
MOVB #1,$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'.
MOV #RTRN,@TBITVEC ;;SET 'T' BIT VECTOR TO $RTRN
MOV #340,@TBITVEC+2 ;;LEVEL 7
MOV #RTI,$RTRN ;;SET $RTRN TO A RTI
MOV #65$,@RESVEC ;;TRY TO DO A RTT
CLR -(SP) ;;DUMMY PS
MOV #64$,-(SP) ;;AND PC
RTT ;;TRY THE RTT
64$: MOV #RTT,$RTRN ;;RTT IS LEGAL--SET $RTRN TO A RTT
BR 66$
65$: ADD #10,SP ;;RTT ILLEGAL--CLEAN OFF THE STACK
66$: MOV #RESVEC+2,@RESVEC ;;RESTORE TRAP CATCHER
CLR $TBIT ;;CLEAR 'T' BIT SWITCH
MOV #,$SLPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
MOV #,$SLPERR ;;SETUP THE ERROR LOOP ADDRESS
;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
;;EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
MOV @ERRVEC,-(SP) ;;SAVE ERROR VECTOR
MOV #67$,@ERRVEC ;;SET UP ERROR VECTOR
MOV #DSWR,$SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
MOV #DDISP,$DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
CMP #-1,$SWR ;;TRY TO REFERENCE HARDWARE SWR
BNE 69$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
;;AND THE HARDWARE SWR IS NOT -1
BR 68$ ;;BRANCH IF NO TIMEOUT
67$: MOV #68$,(SP) ;;SET UP FOR TRAP RETURN
RTI
68$: MOV #SWREG,$SWR ;;POINT TO SOFTWARE SWR
MOV #DISPREG,$DISPLAY
69$: MOV (SP)+,@ERRVEC ;;RESTORE ERROR VECTOR
CLR $PASS ;;CLEAR PASS COUNT
BITB #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
  
```

```

020322 001403          BEQ    70$      ::YES,USE NON-APT SWITCH
020324 012767 001246 160606  MOV    #$$SWREG,SWR  ::NO,USE APT SWITCH REGISTER
020332
1425 70$:
.SBTTL TYPE PROGRAM NAME
::TYPE THE NAME OF THE PROGRAM IF FIRST PASS
020332 005227 177777      INC    #-1          ::FIRST TIME?
020336 001047          BNE    71$          ::BRANCH IF NO
020340 022737 035116 000042  CMP    #SENDAD,@#42  ::ACT-11?
020346 001443          BEQ    71$          ::BRANCH IF YES
020350 104401 020416      TYPE    ,72$        ::TYPE ASCIZ STRING
.SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
020354 005737 000042      TST    @#42        ::ARE WE RUNNING UNDER XXDP/ACT?
020360 001012          BNE    73$          ::BRANCH IF YES
020362 126727 160656 000001  CMPB   $ENV,#1      ::ARE WE RUNNING UNDER APT?
020370 001406          BEQ    73$          ::BRANCH IF YES
020372 026727 160542 000176  CMP    SWR,#SWREG   ::SOFTWARE SWITCH REG SELECTED?
020400 001005          BNE    74$          ::BRANCH IF NO
020402 104407          GTSWR                ::GET SOFT-SWR SETTINGS
020404 000403          BR     74$
020406 112767 000001 160520 73$:  MOVB   #1,$AUTOB    ::SET AUTO-MODE INDICATOR
020414          74$:
020414 000420          BR     71$          ::GET OVER THE ASCIZ
020456          ::72$: .ASCIZ <CRLF>#CKKTAAO 11/44 MEM MGMT PRT A#<CRLF>
          71$:
426
1427 020456          LOOP:
1428 020456 012706 001100      MOV    #STACK,KSP   :INITIALIZE THE STACK POINTER
1429 020462 012767 003024 157314  MOV    #TIMERR,ERRVEC :LOAD CPU SERVICE ROUTINE INTO TRAP VECTOR
1430 020470 012767 000340 157310  MOV    #340,ERRVEC+2 :SET NEW PS TO PRIORITY LEVEL 7-KERNE
1431 020476 012767 003122 157544  MOV    #MGMEERR,MMVEC :LOAD MEMORY MANAGENT ROUTINE INTO VECTOR
1432 020504 012767 000340 157540  MOV    #340,MMVEC+2  :SET NEW PS TO PRIORITY LEVEL 7-KERNEL
1433 020512 012700 177777      MOV    #-1,R0       :PUT -1 INTO R0 TO INITIALIZE FLAGS
1434 020516 010067 162304      MOV    R0,TIMFLG    :INITIALIZE CPU ERROR FLAG
1435 020522 010067 162376      MOV    R0,MGMFLG    :INITIALIZE MEMORY MANAGEMENT ERROR FLAG
1436 020526 012767 000340 160536  MOV    #340,TBITPS  :INITIALIZE LOG THAT HOLDS T-BIT PSW

```

1438 020534 005067 157032  
1439 020540 005067 151752

CLR MMRO  
CLR MMR3

;BE SURE MEM. MGMT IS OFF TO START WITH  
;MAKE SURE ALL MAPPING IS OFF

1441	020544	005067	157216	CLR	CPUERR	:MAKE SURE CPU ERROR REG. IS CLEAR
1442	020550	052767	000200 161734	BIS	#BIT7,\$KT11	:SET M.M. FLAG FOR SIZING ROUTINE
1443	020556	004767	161672	JSR	PC,\$SIZE	:RUN SIZING ROUTINE
1444	020562	005067	157200	CLR	CPUERR	:CLEAR CPU ERROR REG. AFTER SIZING
1445						





1509

```

*****
*TEST 3          BYTE ADDRESSING TEST FOR PSW
*
*          THIS TEST WRITES THE HIGH AND LOW BYTES OF THE PROCESSOR STATUS WORD
*          AND READS THEM BACK TO BE SURE THEY CAN BE WRITTEN INDEPENDENTLY.
*          THIS CHECKS THE PSW PORTION OF THE ADDRESS DETECTION LOGIC.
*
*****
  
```

```

1510 020730 000004
1510 020732 012767 020740 160150
1511 020740 005067 157032
1512 020744 012700 000360
1513 020750 110067 157023
1514 020754 016701 157016
1515 020760 042701 007437
1516 020764 000300
1517 020766 020001
1518 020770 001403
1519 020772 005067 157000
1520 020776 104005
1521
1522
1523
1524 021000 012767 021006 160102 3$:
1525 021006 005067 156764 4$:
1526 021012 012700 000340
1527 021016 110067 156754
1528 021022 016701 156750
1529 021026 042701 007437
1530 021032 020001
1531 021034 001403
1532 021036 005067 156734
1533 021042 104005
1534
1535
1536
1537 021044 012767 020732 160036 5$:
  
```

```

TST3: SCOPE
1$: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
2$: CLR PSW ;CLEAR THE PSW
MOV #360,R0 ;PUT THE HIGH BYTE DATA INTO R0
MOVB R0,PSW+1 ;WRITE THE HIGH BYTE OF THE PSW
MOV PSW,R1 ;READ BACK THE ENTIRE PSW
BIC #007437,R1 ;MASK OFF THE T & CC BITS
SWAB R0 ;GET DATA WRITTEN IN HIGH BYTE OF R0
CMP R0,R1 ;WAS THE PSW WRITTEN TO CORRECTLY
BEQ 3$ ;BRANCH IF YES
CLR PSW ;CLEAR PSW FOR ERROR REPORT
ERROR +5 ;LOW BYTE EFFECTED BY WRITE TO HIGH BYTE OF PSW
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 2$' = 000760
3$: MOV #4$, $LPERR ;SET LOOP ON ERROR POINTER TO 4$
4$: CLR PSW ;CLEAR THE PSW
MOV #340,R0 ;PUT THE LOW BYTE DATA INTO R0
MOVB R0,PSW ;WRITE THE LOW BYTE OF THE PSW
MOV PSW,R1 ;READ BACK THE ENTIRE PSW
BIC #007437,R1 ;MASK OFF THE T&CC BITS
CMP R0,R1 ;WAS PSW WRITTEN TO CORRECTLY
BEQ 5$ ;BRANCH IF YES
CLR PSW ;CLEAR PSW FOR ERROR REPORT
ERROR +5 ;HIGH BYTE EFFECTED BY WRITE TO LOW BYTE OF PSW
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 2$' = 000736
5$: MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
  
```

1550

```

*****
*TEST 4          TEST AND SETUP OF STACK POINTERS
*
*   THIS TEST SETS THE USER AND KERNEL STACK POINTERS FOR THE
*   REST OF THE PROGRAM AND MAKES SURE THEY ARE INDEPENDENT OF
*   EACH OTHER.  KERNEL R6 IS SET TO 1100, SUPERVISOR R6 IS SET TO 700,
*   USER R6 IS SET TO 600, THEN KERNEL R6 IS READ TO BE SURE
*   IT'S STILL 1100.  THE SECOND PART OF THE TEST CHECKS TO SEE
*   THAT THE ILLEGAL MODE('10') STACK POINTER IS MAPPED TO THE
*   USER STACK POINTER, WITH MEMORY MANAGEMENT OFF.
*
*****

```

```

1551 021052 000004
1551 021054 005037 177572
1552 021060 005067 156712
1553 021064 012706 001100
1554 021070 012767 040000 156700
1555 021076 012706 000700
1556 021102 012767 140000 156666
1557 021110 012706 000600
1558 021114 005067 156656
1559 021120 022706 001100
1560 021124 000404
1561 021126 012700 001100
1562 021132 010601
1563 021134 104006
1564
1565
1566
1567 021136 012767 021136 157744 1$:
1568 021144 005067 156626
1569 021150 012746 177777
1570 021154 012767 040000 156614
1571 021162 012746 000001
1572 021166 012767 140000 156602
1573 021174 005046
1574 021176 012767 100000 156572
1575 021204 011600
1576 021206 001401
1577 021210 104040

```

```

TST4:  SCOPE
      CLR      @MMRO           ;MAKE SURE M.M. IS OFF
      CLR      PSW            ;GO TO KERNEL MODE
      MOV      #KERSTK,KSP    ;SET KERNEL STACK POINTER TO 1100
      MOV      #40000,PSW     ;GO TO SUPERVISOR MODE
      MOV      #SUPSTK,SSP    ;SET SUPERVISOR STACK POINTER TO 700
      MOV      #140000,PSW   ;GO TO USER MODE
      MOV      #USESTK,USP    ;SET USER STACK POINTER TO 600
      CLR      PSW            ;BACK TO KERNEL MODE
      CMP      #KERSTK,KSP    ;IS KERNEL R6 STILL 1100?
      BR       1$            ;BRANCH TO NEXT PART OF TEST
      MOV      #KERSTK,R0     ;SAVE DATA WRITTEN FOR ERROR REPORT
      MOV      KSP,R1        ;SAVE DATA READ AFTER USER R6 WAS WRITTEN
      ERROR   +6             ;KERNEL R6 CHANGED BY WRITING USER R6
                               ;FOR TIGHTER SCOPE LOOP
                               ;REPLACE ERROR CALL WITH
                               ;000756
                               ;SET LOOP ON ERROR POINTER TO 1$
      MOV      #1$,SLPERR    ;SET LOOP ON ERROR POINTER TO 1$
      CLR      PSW            ;GO TO KERNEL MODE
      MOV      #-1,-(KSP)    ;PUSH A -1 ONTO STACK
      MOV      #40000,PSW    ;GO TO SUPERVISOR MODE
      MOV      #1,-(SSP)     ;PUSH A 1 ONTO THE STACK
      MOV      #140000,PSW   ;GO TO USER MODE
      CLR      -(USP)        ;PUSH A ZERO ONTO THE STACK
      MOV      #100000,PSW   ;PUT ILLEGAL MODE IN PSW
      MOV      (SP),R0       ;PUT TOP OF STACK INTO R0
      BEQ     TST5          ;WE'RE OK-GO TO NEXT TEST
      ERROR   +40          ;ILLEGAL MODE NOT MAPPED TO USER MODE

```

1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1600  
1601

```

:*****
:
: THE NEXT SEVEN (7) TESTS WILL TRY TO ADDRESS ALL OF THE
: MEMORY MANAGEMENT REGISTERS (SR0->SR3; I&D SPACE KERNEL,
: SUPERVISOR & USER PAR/PDR'S).
: EVERY TIME A REGISTER TIMES OUT ITS ADDRESS WILL BE REPORTED.
: AT THE END OF EACH TEST A SUMMARY OF THE ADDRESSES THAT TIMED
: OUT DURING THAT TEST IS GIVEN. THE RESULTS OF 'AND-ING' AND 'OR-ING'
: THEIR ADDRESSES IS GIVEN TO SHOW WHICH ADDRESS LINES MAY BE
: STUCK AT 0 OR 1. THE PAR/PDR ADDRESS AND KT MUX'S ARE THE
: THINGS BEING CHECKED.
:*****

```

```

:*****
:TEST 5 SR0,SR1,SR2,SR3 TIMEOUT TEST
:
: THIS TEST ADDRESSES THE MEMORY MANAGEMENT STATUS REGISTERS
: 0,1,2, AND 3. DATA WILL BE WRITTEN OR READ FROM THESE REGISTERS
: IN LATER TESTS, THIS TEST JUST CHECKS FOR A RESPONSE.
:*****

```

1602 021212 000004

TST5: SCOPE

```

1603 021214 005067 156556          CLR     PSW           ;MAKE SURE KERNAL MODE IS SELECTED
1604 021220 012767 021262 157662 1$:  MOV     #2$, $LPERR  ;SET LOOP ON ERROR POINTER TO 2$
1605 021226 012737 021326 000004    MOV     #6$, @#4     ;SET TIMEOUT VECTOR TO 6$
1606 021234 012700 177572          MOV     #SR0, R0     ;LOAD R0 WITH ADDRESS OF FIRST REG.
1607 021240 012701 000003          MOV     #3, R1       ;LOAD R1 WITH THE LOOP COUNT
1608 021244 012767 177777 160046    MOV     #-1,ADRAND   ;INITIALIZE 'AND' OF ADDRS. LOC.
1609 021252 005067 160040          CLR     ADDROR       ;INITIALIZE 'OR' OF ADDRS. LOC.
1610 021256 005067 160012          CLR     TONUM        ;INITIALIZE 'TIMEOUTS' COUNTER
1611 021262 005710          2$:  TST     (R0)     ;TRY ADDRESSING A STATUS REGISTER
1612                                     ;IF IT TIMES OUT GO TO 6$
1613 021264 062700 000002          3$:  ADD     #2, R0     ;PUT NEXT ADDRESS IN R0
1614 021270 077104          SOB     R1, 2$       ;LOOP BACK TO 2$ UNTIL ALL TESTED
1615 021272 012700 172516          MOV     #SR3, R0     ;LOAD R0 WITH THE ADDRESS OF SR3
1616 021276 005710          TST     (R0)        ;TRY ADDRESSING SR3
1617 021300 012767 021220 157602 4$:  MOV     #1$, $LPERR  ;RESET LOOP ON ERROR POINTER TO 1$
1618 021306 005767 157762          TST     TONUM        ;DID ANY OF THE STATUS REG.S TIMEOUT?
1619 021312 001401          BEQ     5$           ;BRANCH IF NO
1620 021314 104010          ERROR  +10          ;SUMMARY OF STATUS REG. TIMEOUTS
1621 021316 012737 003024 000004 5$:  MOV     #TIMERR, @#4 ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1622 021324 000417          BR     TST6         ;;GO TO NEXT TEST
1623
1624 021326 062706 000004          6$:  ADD     #4, KSP     ;CLEAN UP THE STACK
1625 021332 104007          ERROR  +7           ;ONE OF THE STATUS REGS. TIMED OUT
1626                                     ;FOR TIGHTER SCOPE LOOP
1627                                     ;REPLACE ERROR CALL WITH
1628                                     ;'BR 2$' = 000756
1629 021334 010002          MOV     R0, R2       ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
1630 021336 050267 157754          BIS     R2, ADDROR   ;'OR' IT WITH OTHER ADDRS. THAT TIMED OUT
1631 021342 005102          COM     R2           ;'AND' IT WITH OTHER ADDRS. THAT TIMED OUT
1632 021344 040267 157750          BIC     R2, ADRAND   ;
1633 021350 005267 157720          INC     TONUM        ;INCREMENT THE TIMEOUT COUNTER
1634 021354 032700 172516          BIT     #SR3, R0     ;TEST FOR LAST REGISTER

```

1635 021360 001347 BNE 4\$ ;SR3 IS LAST;GO TO FINAL ERROR REPORT  
 1636 021362 000740 BR 3\$ ;BRANCH BACK TO TEST THE NEXT ADDR.  
 1637  
 1641

\*\*\*\*\*  
 \*TEST 6 KERNEL I&D SPACE PAR'S TIMEOUT TEST  
 \* THIS TEST ADDRESSES THE SIXTEEN (16) KERNEL PAGE ADDRESS  
 \* REGISTERS (KIPAR0-KIPAR7 AND KDPAR0-KDPAR7)  
 \* AND CHECKS THAT SOMETHING RESPONDS TO THEIR ADDRESSES.  
 \*\*\*\*\*

1642 021364 000004  
 021366 012767 021430 157514 1\$: SCOPE  
 021374 012737 021466 000004 1\$: MOV #2\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 2\$  
 021402 012700 172340 MOV #5\$, @#4 ;SET TIMEOUT VECTOR TO 5\$  
 021406 012701 000020 MOV #KIPAR0,R0 ;LOAD R0 WITH ADDRESS OF FIRST REG  
 021412 012767 177777 157700 MOV #20,R1 ;LOAD R1 WITH LOOP COUNT (16)  
 021420 005067 157672 MOV #-1,ADRAND ;INITIALIZE 'AND' OF ADDR. LOC  
 021424 005067 157644 CLR ADDROR ;INITIALIZE 'OR' OF ADDR. LOC.  
 021430 005710 2\$: CLR TONUM ;INITIALIZE 'TIMEOUTS' COUNTER  
 TST (R0) ;TRY ADDRESSING A REGISTER  
 ;IF IT TIMES OUT, WILL GO TO 5\$  
 021432 062700 000002 3\$: ADD #2,R0 ;PUT NEXT REGISTER ADDRESS IN R0  
 021436 077104 SOB R1,2\$ ;LOOP BACK TO 2\$ UNTIL ALL TESTED  
 021440 012767 021366 157442 MOV #1\$, \$LPERR ;RESET LOOP ON ERROR POINTER TO 1\$  
 021446 005767 157622 TST TONUM ;DID ANY OF THE REGISTERS TIME OUT?  
 021452 001401 BEQ 4\$ ;BRANCH IF NO  
 021454 104010 ERROR +10 ;SUMMARY OF REGISTER TIMEOUTS  
 021456 012737 003024 000004 4\$: MOV #TIMERR,@#4 ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS  
 021464 000414 BR TST7 ;BRANCH TO NEXT TEST  
 021466 062706 000004 5\$: ADD #4,KSP ;CLEAN UP THE STACK  
 021472 104007 ERROR +7 ;ONE OF THE REGISTER TIMED OUT  
 ;FOR TIGHTER SCOPE LOOP, REPLACE ERROR  
 ;CALL WITH 'BR 2\$' = 000756  
 021474 010002 MOV R0,R2 ;LOAD THE ADDRESS THAT TIMED OUT INTO R2  
 021476 050267 157614 BIS R2,ADDROR ;'OR' IT WITH OTHER ADDRS. THAT TIMED OUT  
 021502 005102 COM R2 ;'AND' IT WITH OTHER ADDRS. THAT TIMED OUT  
 021504 040267 157610 BIC R2,ADRAND  
 021510 005267 157560 INC TONUM ;INCREMENT THE TIMEOUT COUNTER  
 021514 000746 BR 3\$ ;BRANCH BACK TO TEST THE NEXT REGISTER

1643  
 1647

\*\*\*\*\*  
 \*TEST 7 KERNEL I&D SPACE PDR'S TIMEOUT TEST  
 \* THIS TEST ADDRESSES THE SIXTEEN (16) KERNEL PAGE DESCRIPTOR  
 \* REGISTERS (KIPDR0-KIPDR7 AND KDPR0-KDPR7)  
 \* AND CHECKS THAT SOMETHING RESPONDS TO THEIR ADDRESSES.  
 \*\*\*\*\*

1648 021516 000004  
 021520 012767 021562 157362 1\$: SCOPE  
 021526 012737 021620 000004 1\$: MOV #2\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 2\$  
 021534 012700 172300 MOV #5\$, @#4 ;SET TIMEOUT VECTOR TO 5\$  
 021540 012701 000020 MOV #KIPDR0,R0 ;LOAD R0 WITH ADDRESS OF FIRST REG  
 021544 012767 177777 157546 MOV #20,R1 ;LOAD R1 WITH LOOP COUNT (16)  
 021552 005067 157540 MOV #-1,ADRAND ;INITIALIZE 'AND' OF ADDR. LOC  
 021556 005067 157512 CLR ADDROR ;INITIALIZE 'OR' OF ADDR. LOC.  
 021562 005710 2\$: CLR TONUM ;INITIALIZE 'TIMEOUTS' COUNTER  
 TST (R0) ;TRY ADDRESSING A REGISTER  
 ;IF IT TIMES OUT, WILL GO TO 5\$  
 021564 062700 000002 3\$: ADD #2,R0 ;PUT NEXT REGISTER ADDRESS IN R0  
 021570 077104 SOB R1,2\$ ;LOOP BACK TO 2\$ UNTIL ALL TESTED  
 021572 012767 021520 157310 MOV #1\$, \$LPERR ;RESET LOOP ON ERROR POINTER TO 1\$

```
021600 005767 157470      TST      TONUM      ;DID ANY OF THE REGISTES TIME OUT?
021604 001401              BEQ      4$         ;BRANCH IF NO
021606 104010              ERROR    +10        ;SUMMARY OF REGISTER TIMEOUTS
021610 012737 003024 000004 4$:  MOV      #TIMERR,@#4 ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
021616 000414              BR       TST10     ;BRANCH TO NEXT TEST
021620 062706 000004      5$:  ADD      #4,KSP    ;CLEAN UP THE STACK
021624 104007              ERROR    +7         ;ONE OF THE REGISTER TIMED OUT
                                ;FOR TIGHTER SCOPE LOOP, REPLACE ERROR
                                ;CALL WITH 'BR 2$' = 000756
021626 010002              MOV      R0,R2     ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
021630 050267 157462      BIS      R2,ADDROR ;'OR' IT WITH OTHER ADDRS. THAT TIMED OUT
021634 005102              COM      R2        ;'AND' IT WITH OTHER ADDRS. THAT TIMED OUT
021636 040267 157456      BIC      R2,ADRAND
021642 005267 157426      INC      TONUM     ;INCREMENT THE TIMEOUT COUNTER
021646 000746              BR       3$         ;BRANCH BACK TO TEST THE NEXT REGISTER
```

1649  
1653

```
::*****  
:*TEST 10      SUPERVISOR I&D SPACE PAR'S TIMEOUT TEST  
:*      THIS TEST ADDRESSES THE SIXTEEN (16) SUPERVISOR PAGE ADDRESS  
:*      REGISTERS (SIPAR0-SIPAR7 AND SDPAR0-SDPAR7)  
:*      AND CHECKS THAT SOMETHING RESPONDS TO THEIR ADDRESSES.  
:*****
```

```
1654 021650 000004      TST10: SCOPE
021652 012767 021714 157230 1$:  MOV      #2$,SLPERR ;SET LOOP ON ERROR POINTER TO 2$
021660 012737 021752 000004      MOV      #5$,@#4    ;SET TIMEOUT VECTOR TO 5$
021666 012700 172240      MOV      #SIPAR0,R0 ;LOAD R0 WITH ADDRESS OF FIRST REG
021672 012701 000020      MOV      #20,R1    ;LOAD R1 WITH LOOP COUNT (16)
021676 012767 177777 157414      MOV      #-1,ADRAND ;INITIALIZE 'AND' OF ADDR. LOC
021704 005067 157406      CLR      ADDROR    ;INITIALIZE 'OR' OF ADR. LOC.
021710 005067 157360      CLR      TONUM     ;INITIALIZE 'TIMEOUTS' COUNTER
021714 005710      2$:  TST      (R0)     ;TRY ADDRESSING A REGISTER
                                ;IF IT TIMES OUT, WILL GO TO 5$
021716 062700 000002      3$:  ADD      #2,R0     ;PUT NEXT REGISTER ADDRESS IN R0
021722 077104      SOB      R1,2$     ;LOOP BACK TO 2$ UNTIL ALL TESTED
021724 012767 021652 157156      MOV      #1$,SLPERR ;RESET LOOP ON ERROR POINTER TO 1$
021732 005767 157336      TST      TONUM     ;DID ANY OF THE REGISTES TIME OUT?
021736 001401              BEQ      4$         ;BRANCH IF NO
021740 104010              ERROR    +10        ;SUMMARY OF REGISTER TIMEOUTS
021742 012737 003024 000004 4$:  MOV      #TIMERR,@#4 ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
021750 000414              BR       TST11     ;BRANCH TO NEXT TEST
021752 062706 000004      5$:  ADD      #4,KSP    ;CLEAN UP THE STACK
021756 104007              ERROR    +7         ;ONE OF THE REGISTER TIMED OUT
                                ;FOR TIGHTER SCOPE LOOP, REPLACE ERROR
                                ;CALL WITH 'BR 2$' = 000756
021760 010002              MOV      R0,R2     ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
021762 050267 157330      BIS      R2,ADDROR ;'OR' IT WITH OTHER ADDRS. THAT TIMED OUT
021766 005102              COM      R2        ;'AND' IT WITH OTHER ADDRS. THAT TIMED OUT
021770 040267 157324      BIC      R2,ADRAND
021774 005267 157274      INC      TONUM     ;INCREMENT THE TIMEOUT COUNTER
022000 000746              BR       3$         ;BRANCH BACK TO TEST THE NEXT REGISTER
```

1655  
1659

```
::*****  
:*TEST 11      SUPERVISOR I&D SPACE PDR'S TIMEOUT TEST  
:*      THIS TEST ADDRESSES THE SIXTEEN (16) SUPERVISOR PAGE DESCRIPTOR  
:*      REGISTERS (SIPAR0-SIPAR7 AND SDPAR0-SDPAR7)  
:*      AND CHECKS THAT SOMETHING RESPONDS TO THEIR ADDRESSES.  
:*****
```



```

1660 022002 000004 TST11: SCOPE
022004 012767 022046 157076 1$: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
022012 012737 022104 000004 MOV #5$, @#4 ;SET TIMEOUT VECTOR TO 5$
022020 012700 172200 MOV #SIPDR0,R0 ;LOAD R0 WITH ADDRESS OF FIRST REG
022024 012701 000020 MOV #20,R1 ;LOAD R1 WITH LOOP COUNT (16)
022030 012767 177777 157262 MOV #-1,ADRAND ;INITIALIZE 'AND' OF ADDR. LOC
022036 005067 157254 CLR ADDROR ;INITIALIZE 'OR' OF ADR. LOC.
022042 005067 157226 CLR TONUM ;INITIALIZE 'TIMEOUTS' COUNTER
022046 005710 2$: TST (R0) ;TRY ADDRESSING A REGISTER
;IF IT TIMES OUT, WILL GO TO 5$
022050 062700 000002 3$: ADD #2,R0 ;PUT NEXT REGISTER ADDRESS IN R0
022054 077104 SOB R1,2$ ;LOOP BACK TO 2$ UNTIL ALL TESTED
022056 012767 022004 157024 MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
022064 005767 157204 TST TONUM ;DID ANY OF THE REGISTERS TIME OUT?
022070 001401 BEQ 4$ ;BRANCH IF NO
022072 104010 ERROR +10 ;SUMMARY OF REGISTER TIMEOUTS
022074 012737 003024 000004 4$: MOV #TIMERR,@#4 ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
022102 000414 BR TST12 ;BRANCH TO NEXT TEST
022104 062706 000004 5$: ADD #4,KSP ;CLEAN UP THE STACK
022110 104007 ERROR +7 ;ONE OF THE REGISTER TIMED OUT
;FOR TIGHTER SCOPE LOOP, REPLACE ERROR
;CALL WITH 'BR 2$' = 000756
022112 010002 MOV R0,R2 ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
022114 050267 157176 BIS R2,ADDROR ;'OR' IT WITH OTHER ADDRS. THAT TIMED OUT
022120 005102 COM R2 ;'AND' IT WITH OTHER ADDRS. THAT TIMED OUT
022122 040267 157172 BIC R2,ADRAND
022126 005267 157142 INC TONUM ;INCREMENT THE TIMEOUT COUNTER
022132 000746 BR 3$ ;BRANCH BACK TO TEST THE NEXT REGISTER
  
```

1661  
1665

```

:*****
:*TEST 12 USER I&D SPACE PAR'S TIMEOUT TEST
:* THIS TEST ADDRESSES THE SIXTEEN (16) USER PAGE ADDRESS
:* REGISTERS (UIPAR0-UIPAR7 AND UDPAR0-UDPAR7)
:* AND CHECKS THAT SOMETHING RESPONDS TO THEIR ADDRESSES.
:*****
  
```

```

1666 022134 000004 TST12: SCOPE
022136 012767 022200 156744 1$: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
022144 012737 022236 000004 MOV #5$, @#4 ;SET TIMEOUT VECTOR TO 5$
022152 012700 177640 MOV #UIPAR0,R0 ;LOAD R0 WITH ADDRESS OF FIRST REG
022156 012701 000020 MOV #20,R1 ;LOAD R1 WITH LOOP COUNT (16)
022162 012767 177777 157130 MOV #-1,ADRAND ;INITIALIZE 'AND' OF ADDR. LOC
022170 005067 157122 CLR ADDROR ;INITIALIZE 'OR' OF ADR. LOC.
022174 005067 157074 CLR TONUM ;INITIALIZE 'TIMEOUTS' COUNTER
022200 005710 2$: TST (R0) ;TRY ADDRESSING A REGISTER
;IF IT TIMES OUT, WILL GO TO 5$
022202 062700 000002 3$: ADD #2,R0 ;PUT NEXT REGISTER ADDRESS IN R0
022206 077104 SOB R1,2$ ;LOOP BACK TO 2$ UNTIL ALL TESTED
022210 012767 022136 156672 MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
022216 005767 157052 TST TONUM ;DID ANY OF THE REGISTERS TIME OUT?
022222 001401 BEQ 4$ ;BRANCH IF NO
022224 104010 ERROR +10 ;SUMMARY OF REGISTER TIMEOUTS
022226 012737 003024 000004 4$: MOV #TIMERR,@#4 ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
022234 000414 BR TST13 ;BRANCH TO NEXT TEST
022236 062706 000004 5$: ADD #4,KSP ;CLEAN UP THE STACK
022242 104007 ERROR +7 ;ONE OF THE REGISTER TIMED OUT
;FOR TIGHTER SCOPE LOOP, REPLACE ERROR
;CALL WITH 'BR 2$' = 000756
  
```

```

022244 010002      MOV    R0,R2      ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
022246 050267 157044  BIS    R2,ADDROR  ;'OR' IT WITH OTHER ADDRS. THAT TIMED OUT
022252 005102      COM    R2          ;'AND' IT WITH OTHER ADDRS. THAT TIMED OUT
022254 040267 157040  BIC    R2,ADRAND
022260 005267 157010  INC    TONUM      ;INCREMENT THE TIMEOUT COUNTER
022264 000746      BR     3$         ;BRANCH BACK TO TEST THE NEXT REGISTER
  
```

1667  
1671

```

*****
*TEST 13      USER I&D SPACE PDR'S TIMEOUT TEST
*      THIS TEST ADDRESSES THE SIXTEEN (16) USER PAGE DESCRIPTOR
*      REGISTERS (UIPAR0-UIPAR7 AND UDPAR0-UDPAR7)
*      AND CHECKS THAT SOMETHING RESPONDS TO THEIR ADDRESSES.
*****
  
```

```

1672 022266 000004      TST13: SCOPE
022270 012767 022332 156612 1$:  MOV    #2$,SLPERR  ;SET LOOP ON ERROR POINTER TO 2$
022276 012737 022370 000004      MOV    #5$,@#4     ;SET TIMEOUT VECTOR TO 5$
022304 012700 177600      MOV    #UIPDR0,R0  ;LOAD R0 WITH ADDRESS OF FIRST REG
022310 012701 000020      MOV    #20,R1      ;LOAD R1 WITH LOOP COUNT (16)
022314 012767 177777 156776      MOV    #-1,ADRAND  ;INITIALIZE 'AND' OF ADDR. LOC
022322 005067 156770      CLR    ADDROR      ;INITIALIZE 'OR' OF ADR. LOC.
022326 005067 156742      CLR    TONUM       ;INITIALIZE 'TIMEOUTS' COUNTER
022332 005710      2$:  TST    (R0)       ;TRY ADDRESSING A REGISTER
                                ;IF IT TIMES OUT, WILL GO TO 5$
022334 062700 000002      3$:  ADD    #2,R0       ;PUT NEXT REGISTER ADDRESS IN R0
022340 077104      SOB    R1,2$       ;LOOP BACK TO 2$ UNTIL ALL TESTED
022342 012767 022270 156540      MOV    #1$,SLPERR  ;RESET LOOP ON ERROR POINTER TO 1$
022350 005767 156720      TST    TONUM       ;DID ANY OF THE REGISTES TIME OUT?
022354 001401      BEQ    4$          ;BRANCH IF NO
022356 104010      ERROR  +10        ;SUMMARY OF REGISTER TIMEOUTS
022360 012737 003024 000004 4$:  MOV    #TIMERR,@#4 ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
022366 000414      BR     TST14      ;BRANCH TO NEXT TEST
022370 062706 000004 5$:  ADD    #4,KSP      ;CLEAN UP THE STACK
022374 104007      ERROR  +7         ;ONE OF THE REGISTER TIMED OUT
                                ;FOR TIGHTER SCOPE LOOP, REPLACE ERROR
                                ;CALL WITH 'BR 2$' = 000756
022376 010002      MOV    R0,R2      ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
022400 050267 156712  BIS    R2,ADDROR  ;'OR' IT WITH OTHER ADDRS. THAT TIMED OUT
022404 005102      COM    R2          ;'AND' IT WITH OTHER ADDRS. THAT TIMED OUT
022406 040267 156706  BIC    R2,ADRAND
022412 005267 156656  INC    TONUM       ;INCREMENT THE TIMEOUT COUNTER
022416 000746      BR     3$         ;BRANCH BACK TO TEST THE NEXT REGISTER
  
```

1673  
1686

```

*****
*TEST 14      MMR0(15:13) BIT TEST & MMR2 TEST
*
*      THIS TEST CHECKS BITS <15:13> OF MEMORY MANAGEMENT
*      REGISTER 0 TO SEE THAT EACH CAN BE SET AND CLEARED
*      AND THAT A 'RESET' WILL CLEAR ALL OF THEM. A TEST
*      OF THESE THREE ERROR BITS CHECKS PART OF SRO, THE
*      SRO MUX AND THE KTMUX. THE REST OF THE BITS IN
*      SRO WILL BE CHECKED LATER.
*      ALSO CHECK THAT SR2 IS TRACKING WITH MEM. MGMT.
*      OFF BUT LOCKS UP WHEN ANY OF SRO ERROR BITS SET.
*****
  
```

1687 022420 000004

TST14: SCOPE

```

1688 022422 012700 177572      1$:  MOV    #SRO,R0      ;LOAD ADDRESS OF SRO INTO R0
1689 022426 012710 160000      MOV    #160000,(R0) ;SET BITS <15:13> IN SRO (ERROR BITS)
1690 022432 000005      RESET  ;ISSUE AND 'INIT' SIGNAL
1691 022434 011001      MOV    (R0),R1      ;READ SRO INTO R1 TO SEE IF CLEAR
1692 022436 001404      BEQ    2$           ;BRANCH IF SRO<15:13> CLEARED BY 'INIT'
1693 022440 104011      ERROR  +11         ;SRO<15:13> NOT CLEARED BY A 'RESET'
1694                                     ;FOR TIGHTER SCOPE LOOP
1695                                     ;REPLACE ERROR CALL WITH
1696                                     ;'BR 1$' = 000770
1697 022442 012767 022450 156440  MOV    #2$,SLPERR   ;SET LOOP ON ERROR POINTER TO 2$
1698 022450 016767 155122 156612 2$:  MOV    SR2,WASSR2   ;READ CONTENTS OF SR2
1699 022456 012701 022450      MOV    #2$,R1       ;LOAD EXPECTED CONTENTS INTO R1
1700 022462 020167 156602      CMP    R1,WASSR2    ;IS SR2 TRACKING?
1701 022466 001401      BEQ    3$           ;BRANCH IF YES
1702 022470 104021      ERROR  +21         ;SR2 NOT 'TRACKING' VIRTUAL ADDRESSES
1703                                     ;FOR TIGHTER SCOPE LOOP
1704                                     ;REPLACE ERROR CALL WITH
1705                                     ;'BR 2$' = 000767
1706 022472 012767 022510 156410 3$:  MOV    #4$,SLPERR   ;SET LOOP ON ERROR POINTER TO 4$
1707 022500 012701 100000      MOV    #BIT15,R1    ;PUT DATA TO BE WRITTEN IN R1
1708 022504 012703 000003      MOV    #3,R3        ;SETUP R3 AS A LOOP COUNTER
1709 022510 005010      CLR    (R0)         ;CLEAR SRO
1710 022512 050110      4$:  BIS    R1,(R0)     ;SET ONE OF THE ERROR BITS IN SRO
1711 022514 011002      5$:  MOV    (R0),R2     ;READ SRO INTO R2
1712 022516 020102      CMP    R1,R2        ;DID RIGHT ERROR BIT GET SET?
1713 022520 001401      BEQ    6$           ;BRANCH IF YES
1714 022522 104012      ERROR  +12         ;BITS WERE SET WRONG IN SRO
1715                                     ;FOR TIGHTER SCOPE LOOP
1716                                     ;REPLACE ERROR CALL WITH
1717                                     ;'BR 4$' = 000772
1718 022524 012704 022512 156532 6$:  MOV    #5$,R4       ;LOAD EXPECTED CONTENTS OF SR2 IN R4
1719 022530 016767 155042      MOV    SR2,WASSR2   ;READ SR2
1720 022536 020467 156526      CMP    R4,WASSR2    ;DID SR2 LOCK UP WHEN ERROR
1721                                     ;BIT SET IN SR1?
1722 022542 001401      BEQ    7$           ;BRANCH IF YES
1723 022544 104027      ERROR  +27         ;SR2 DID NOT LOCK UP
1724                                     ;FOR TIGHTER SCOPE LOOP
1725                                     ;REPLACE ERROR CALL WITH
1726                                     ;'BR 4$' = 000761
1727 022546 006001      7$:  ROR    R1          ;CHANGE DATA TO CHECK NEXT ERROR BIT
1728 022550 077321      SOB    R3,4$       ;LOOP BACK UNTIL <15:13> ALL TESTED
1729 022552 005010      CLR    (R0)        ;CLEAR SRO BEFORE LEAVING
1730 022554 012767 022422 156326  MOV    #1$,SLPERR   ;RESET LOOP ON ERROR POINTER TO 1$
1731
1741

```

```

:*****
:*TEST 15      SRO & PSW DUAL ADDRESSING TEST
:*
:*      THIS TEST CHECKS MORE OF THE ADDRESS DETECTION LOGIC BY
:*      VERIFYING THAT STATUS REGISTER 0 IS NOT EFFECTED BY WRITING
:*      TO THE PSW AND THAT THE LOW BYTE OF STATUS REGISTER 0
:*      IS NOT EFFECTED BY WRITING TO ITS HIGH BYTE. THIS IS TO
:*      SEE IF ADJACENT OUTPUTS ARE SHORTED ON THE ADDRESS DET. LOGIC.
:*
:*****

```

```

1742 022562 000004      TST15: SCOPE
1743 022564 005067 155206 1$:  CLR    PSW          ;CLEAR THE PSW

```

1744	022570	005067	154776		CLR	SRO	:CLEAR STATUS REGISTER 0
1745	022574	012737	000340	177776	MOV	#340,24PSW	:SET PRIORITY 7 IN LOW BYTE OF PSW
1746	022602	016700	154764		MOV	SRO,R0	:READ STATUS REGISTER 0
1747	022606	001401			BEQ	2\$	:BRANCH IF IT WAS STILL 0
1748	022610	104013			ERROR	+13	:SRO EFFECTED BY A WRITE TO THE PSW
1749							:FOR TIGHTER SCOPE LOOP
1750							:REPLACE ERROR CALL WITH
1751							: 'BR 1\$' = 000767
1752	022612	005067	154754	2\$:	CLR	SRO	:BE SURE SRO IS 0 BEFORE LEAVING
1753	022616	005067	155154		CLR	PSW	:BE SURE PSW IS 0 BEFORE LEAVING

1767  
 1768

```

:*****
:*TEST 16 BIT TEST OF MEMORY MANAGEMENT REGISTER 1
:*THIS TEST WILL CAUSE BITS IN MMR1 TO BE LOCKED BY SETTING BITS <15:13>
:*IN MMRO. THE REGISTER ONLY GETS CLOKED ON AN INSTRUCTION THAT AUTO
:*INCREMENTS OR AUTO DECREMENTS A REGISTER. THE LOWER BYTE IS ALWAYS CLOKED
:*FIRST AND THE UPPER BYTE IS ONLY CLOKED IF BOTH SOURCE AND DESTINATION
:*AUTO INCREMENT OR DECREMENT A REGISTER.
:*ALL COUNTS (+1,-1,+2,-2) THAT CAN BE GENERATED ARE CLOKED INTO THE HIGH
:*AND LOW BYTES. ALL REGISTER NUMBERS ARE GENERATED, INCLUDING ALL THREE
:*R6'S, AND ALL THE BITS THAT HOLD THE REGISTER NUMBER IN BOTH BYTES
:*ARE TESTED; HOWEVER, ALL REGISTER NUMBERS ARE CLOKED INTO BOTH HIGH AND
:*LOW BYTES, BUT ENOUGH ARE USED TO TEST THE LOGIC.
:*****
  
```

```

TST16: SCOPE
20$: MOV #MMR1,R0 ;PUT ADDRESS OF MMR1 INTO R0
MOV #BIT13,R4 ;SET READ ONLY BIT IN R4
MOV #11$,SLPERR ;SET LOOP ON ERROR POINTER TO 11$
11$: MOV #BIT15,@MMRO ;LOCK UP MMR1 LOW BYTE 027,HIGH BYTE 027
;WORD-013427
MOV #013427,R3 ;PUT EXPECTED DATA IN R3
MOV (R0),R1 ;READ MMR1 INTO R1
CMP R1,R3 ;SEE IF DATA MATCHES
BEQ 10$ ;BRANCH IF DATA MATCHES
ERROR +14 ;MMR1 DID NOT TRACK PROPERLY
10$: MOV #12$,SLPERR ;SET LOOP ON ERROR POINTER TO 12$
12$: RESET ;ISSUE INIT
MOV (R0),R1 ;SEE IF MMR1 IS CLEARED
BEQ 1$ ;BRANCH IF MMR1 IS ZERO
ERROR +11 ;CAN'T CLEAR MMR1
1769 022622 000004 177574
1770 022624 012700 020000
1771 022630 012704 022642 156246
1772 022634 012767 022642 177572
1773 022642 012737 100000
1774 022650 012703 013427
1775 022654 011001
1776 022656 020103
1777 022660 001401
1778 022662 104014
1779 022664 012767 022672 156216
1780 022672 000005
1781 022674 011001
1782 022676 001401
1783 022700 104011
1784 022702 012767 022710 156200
1785 022710 012702 177572
1786 022714 012722 040000
1787
1788 022720 011001
1789 022722 012703 011027
1790 022726 020103
1791 022730 001401
1792 022732 104014
1793 022734 005037 177572
1794 022740 012767 022746 156142
1795 022746 012702 177574
1796 022752 010442
1797
1798 022754 011001
1799 022756 012703 171000
1800 022762 020103
1801 022764 001401
1802 022766 104014
1803 022770 005012
1804 022772 012767 023000 156110
1805 023000 012705 177573
1806 023004 012701 001343
1807 023010 112125
1808
1809 023012 011001
1810 023014 012703 006411
  
```

1811	023020	020103				CMP	R1,R3		:SEE IF DATA MATCHES
1812	023022	001401				BEQ	4\$		:BRANCH IF DATA MATCHES
1813	023024	104014				ERROR	+14		:MMR1 DID NOT TRACK PROPERLY
1814	023026	005012			4\$:	CLR	(R2)		:CLEAR MMR0
1815	023030	012767	023036	156052		MOV	#16\$, \$LPERR		:SET LOOP ON ERROR POINTER TO 16\$
1816	023036	012701	177574		16\$:	MOV	#MMR0+2,R1		:PUT ADDRESS OF MMR0'S UPPER BYTE
1817									:PLUS 1 IN R1
1818	023042	012705	001344			MOV	#READON+2,R5		:PUT ADDRESS OF READ ONLY BIT <13>
1819									:PLUS 2 IN R5
1820	023046	114541				MOVB	-(R5),-(R1)		:LOCK UP MMR1-LOWER BYTE 375
1821									:UPPER BYTE 371-WORD 174775
1822	023050	011001				MOV	(R0),R1		:READ MMR1
1823	023052	012703	174775			MOV	#174775,R3		:PUT EXPECTED DATA IN R3
1824	023056	020103				CMP	R1,R3		:SEE IF DATA MATCHES
1825	023060	001401				BEQ	5\$		:BRANCH IF DATA MATCHES
1826	023062	104014				ERROR	+14		:MMR1 DID NOT TRACK PROPERLY
1827	023064	005012			5\$:	CLR	(R2)		:CLEAR MMR0
1828	023066	012767	023106	156014		MOV	#17\$, \$LPERR		:SET LOOP ON ERROR POINTER TO 17\$
1829	023074	012737	040000	177776		MOV	#40000,@#PSW		:SET SUPERVISOR MODE
1830	023102	010637	001176			MOV	SSP,@#STMP0		:SAVE SUPERVISOR STACK POINTER
1831	023106	012706	177574		17\$:	MOV	#MMR0+2,SSP		:PUT ADDRESS OF MMR0+2 IN SSP
1832	023112	012746	040000			MOV	#40000,-(SSP)		:LOCK UP MMR1-LOWER BYTE 027
1833									:UPPER BYTE 366-WORD 173027
1834	023116	011001				MOV	(R0),R1		:READ MMR1
1835	023120	012703	173027			MOV	#173027,R3		:PUT EXPECTED DATA IN R3
1836	023124	020103				CMP	R1,R3		:SEE IF DATA MATCHES
1837	023126	001401				BEQ	6\$		:BRANCH IF DATA MATCHES
1838	023130	104014				ERROR	+14		:MMR1 DID NOT TRACK PROPERLY
1839	023132	016706	156040		6\$:	MOV	\$TMP0,SSP		:RESTORE THE SUPERVISOR STACK POINTER
1840	023136	005012				CLR	(R2)		:CLEAR MMR0
1841	023140	012767	023160	155742		MOV	#18\$, \$LPERR		:SET LOOP ON ERROR POINTER TO 18\$
1842	023146	012737	140000	177776		MOV	#140000,@#PSW		:SET USER MODE
1843	023154	010637	001176			MOV	USP,@#STMP0		:SAVE USER STACK POINTER
1844	023160	012706	177572		18\$:	MOV	#MMR0,USP		:PUT ADDRESS OF MMR0 IN USP
1845	023164	012726	100000			MOV	#100000,(USP)+		:LOCK UP MMR1-LOWER BYTE 027
1846									:UPPER BYTE 026-WORD 013027
1847	023170	011001				MOV	(R0),R1		:READ MMR1
1848	023172	012703	013027			MOV	#013027,R3		:PUT EXPECTED DATA IN R3
1849	023176	020103				CMP	R1,R3		:SEE IF DATA MATCHES
1850	023200	001401				BEQ	7\$		:BRANCH IF DATA MATCHES
1851	023202	104014				ERROR	+14		:MMR1 DID NOT TRACK PROPERLY
1852	023204	016706	155766		7\$:	MOV	\$TMP0,USP		:RESTORE THE USER STACK POINTER
1853	023210	005037	177776			CLR	@#PSW		:GO BACK TO KERNAL MODE
1854	023214	005037	177572			CLR	@#MMR0		:LET ALL MEM MGT REG'S TRACK
1855	023220	012767	022624	155662		MOV	#20\$, \$LPERR		:SET LOOP ON ERROR POINTER TO STAR OF TEST

1864

```

*****
*TEST 17 BIT TEST OF MEMORY MANAGEMENT REGISTER 3
* THIS TEST SETS AND CLEARS BITS <05:04> AND <02:00> OF MMR3. IT DOES
* NOT TEST THAT THE BITS FUNCTION PROPERLY SINCE THAT REQUIRES MORE
* LOGIC; HOWEVER, IT TESTS THE REGISTER AND THE DATA PATH TO AND FROM
* THE REGISTER. THE PROPER FUNCTIONING OF THESE BITS IS TESTED LATER
* IN SEVERAL TESTS.
*****
  
```

```

TST17: SCOPE
1865 023226 000004 172516 20$: MOV #MMR3,R0 ;PUT ADDRESS OF MMR3 IN R0
1866 023230 012700 000001 MOV #1,R2 ;SET BIT TO FLOAT THRU MMR3
1867 023234 012702 CLR (R0) ;CLEAR MMR3
1868 023240 005010 MOV (R0),R1 ;READ MMR3 INTO R1
1869 023242 011001 BEQ 5$ ;BRANCH IF ZERO
1870 023244 001401 ERROR +11 ;CAN'T CLEAR MMR3
1871 023246 104011 5$: MOV #6,R3 ;SET UP LOOP COUNT
1872 023250 012703 000006 MOV #1$,$LPERR ;SET LOOP ON ERROR POINTER TO 1$
1873 023254 012767 023262 155626 1$: MOV R2,(R0) ;LOAD MMR3
1874 023262 010210 MOV (R0),R1 ;READ MMR3 INTO R1
1875 023264 011001 CMP R2,R1 ;DOES DATA MATCH PATTERN?
1876 023266 020201 BEQ 2$ ;BRANCH IF IT MATCHES
1877 023270 001401 ERROR +15 ;MMR3 HAS WRONG DATA
1878 023272 104015 2$: ASL R2 ;LEFT SHIFT DATA PATTERN
1879 023274 006302 SOB R3,1$ ;BRANCH BACK IF R3 NOT 0
1880 023276 077307 MOV #77,R2 ;PUT DATA PATTERN IN R2
1881 023300 012702 000077 MOV #12$,$LPERR ;SET LOOP ON ERROR POINTER TO 12$
1882 023304 012767 023312 155576 12$: MOV R2,(R0) ;TRY TO SET ALL BITS IN MMR3
1883 023312 010210 MOV (R0),R1 ;READ MMR3 INTO R1
1884 023314 011001 CMP R2,R1 ;SEE IF ALL BITS GOT SET
1885 023316 020201 BEQ 3$ ;BRANCH IF ALL WERE SET
1886 023320 001401 ERROR +15 ;MMR3 HAS WRONG DATA
1887 023322 104015 3$: MOV #13$,$LPERR ;SET LOOP ON ERROR POINTER TO 13$
1888 023324 012767 023332 155556 13$: RESET ;ISSUE AN INIT
1889 023332 000005 MOV (R0),R2 ;READ MMR3 INTO R2
1890 023334 011002 BEQ 4$ ;BRANCH IF MMR3 CLEARED
1891 023336 001401 ERROR +11 ;CAN'T CLEAR MMR3
1892 023340 104011 4$: MOV #20$,$LPERR ;SET LOOP ON ERROR POINTER TO START OF TEST
1892 023342 012767 023230 155540
  
```





```

; *TEST 21      DUAL ADDRESS SUPERVISOR PAR'S,ON LOADING
; *
; *THIS TEST FIRST CLEARS ALL THE SUPERVISOR PAGE ADDRESS REGISTERS,
; *AND CHECKS TO SEE THAT THEY EACH HOLD ZERO. THEN, STARTING
; *WITH I-SPACE ADDRESS REGISTER ZERO, ONE REGISTER AT A TIME IS
; *LOADED WITH A NEGATIVE ONE. ALL SUPERVISOR ADDRESS REGISTERS ARE
; *NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
; *INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN
; *AT THE END OF THIS TEST.
; *****

```

```

1918 023546 000004
023550 004767 156522
023554 012767 023602 155326
023562 012705 172240
023566 004767 156466
023572 012700 172240
023576 012701 000020
023602 011002
023604 001401
023606 104030
023610 062700 000002
023614 077106

TST21: SCOPE
20$: JSR PC,CLEANUP ;INITIALIZE THE ERROR LOCATIONS
      MOV #1$, $LPERR ;SET LOOP ON ERROR POINTER TO 1$
      MOV #SIPAR0,R5 ;PUT ADDRESS OF FIRST PAR IN R5
      JSR PC,CLRREG ;CLEAR 16 REGISTERS POINTED TO BY R5
      MOV #SIPAR0,R0 ;PUT ADDRESS OF FIRST PAR IN R0
      MOV #20,R1 ;BRANCH COUNT IS 16 DECIMAL
1$: MOV (R0),R2 ;READ PAR TO R2
   BEQ 2$ ;BRANCH IF PAR IS 0
   ERROR +30 ;PAR NOT ZERO
2$: ADD #2,R0 ;POINT TO NEXT REGISTER
   SOB R1,1$ ;BRANCH BACK TO 1$ 15 TIMES
;NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE REGISTER AND
;READING THE REST TO SEE ONLY THAT REGISTER IS NON ZERO. (ANY
;DROPPED BITS WILL BE FOUND IN THE NEXT TEST).
      MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
      MOV #SIPAR0,R0 ;PUT ADDRESS OF FIRST PAR IN R0
3$: MOV #SIPAR0,R5 ;LOAD STARTING ADDRESS INTO R5
   JSR PC,CLRREG ;CLEAR 16 REGISTERS POINTED TO BY R5
   MOV #SDPAR7,R1 ;PUT SDPAR7 ADDRESS INTO R1
   MOV #-1,(R0) ;LOAD REGISTER UNDER TEST
4$: CLR $TMP0 ;FLAG TO INDICATE THERE WAS A MATCH
   MOV (R1),R2 ;READ ALL REGISTERS
   BEQ 6$ ;BRANCH IF REGISTER IS 0
   CMP R0,R1 ;IS THE ADDRESS OF NON-ZERO REGISTER THE SAME
   ;AS THE REGISTER UNDER TEST?
   BEQ 5$ ;BRANCH IF ADDRESSES MATCH
   JSR PC,DUALADR ;LOG AND REPORT ERRORS
5$: INC $TMP0 ;SET FLAG WHEN ADDRESSES MATCH
6$: SUB #2,R1 ;POINT TO NEXT REGISTER
   CMP #SIPAR0,R1 ;SEE IF ALL REGISTERS HAVE BEEN READ
   BLOS 4$ ;BRANCH IF MORE TO READ
   ADD #2,R0 ;NOW LOAD THE NEXT REGISTER
   CMP #SDPAR7,R0 ;SEE IF THERE ARE MORE REGISTERS TO TEST
   BHIS 3$ ;BRANCH IF MORE REGISTERS TO TEST
   MOV #20$, $LPERR ;SET LOOP ON ERROR POINTER TO START OF TEST
   TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
   BR TST22 ;BRANCH TO NEXT TEST IF NO ERRORS
   MOV ERRCNT,$TMP5 ;SAVE # OF ERRORS FOR PAR OUT
   ERROR +31 ;SUMMARY OF DUAL ADDRESS TEST

023616 012767 023630 155264
023624 012700 172240
023630 012705 172240
023634 004767 156420
023640 012701 172276
023644 012710 177777
023650 005067 155322
023654 011102
023656 001406
023660 020001

023662 001402
023664 004767 156444
023670 005267 155302
023674 162701 000002
023700 022701 172240
023704 101761
023706 062700 000002
023712 022700 172276
023716 103344
023720 012767 023550 155162
023726 005767 155370
023732 000404
023734 016767 155362 155246
023742 104031

```

1923

```

:*****
:*TEST 22      DUAL ADDRESS USER PAR'S,ON LOADING
:*
:*THIS TEST FIRST CLEARS ALL THE USER PAGE ADDRESS REGISTERS,
:*AND CHECKS TO SEE THAT THEY EACH HOLD ZERO. THEN, STARTING
:*WITH I-SPACE ADDRESS REGISTER ZERO, ONE REGISTER AT A TIME IS
:*LOADED WITH A NEGATIVE ONE. ALL USER ADDRESS REGISTERS ARE
:*NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
:*INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN
:*AT THE END OF THIS TEST.
:*****

```

1924 023744 000004  
023746 004767 156324  
023752 012767 024000 155130  
023760 012705 177640  
023764 004767 156270  
023770 012700 177640  
023774 012701 000020  
024000 011002  
024002 001401  
024004 104030  
024006 062700 000002  
024012 077106

```

TST22: SCOPE
20$: JSR PC,CLEANUP      ;INITIALIZE THE ERROR LOCATIONS
      MOV #1$, $LPERR    ;SET LOOP ON ERROR POINTER TO 1$
      MOV #UIPAR0,R5     ;PUT ADDRESS OF FIRST PAR IN R5
      JSR PC,CLRREG      ;CLEAR 16 REGISTERS POINTED TO BY R5
      MOV #UIPAR0,R0     ;PUT ADDRESS OF FIRST PAR IN R0
      MOV #20,R1         ;BRANCH COUNT IS 16 DECIMAL
1$:   MOV (R0),R2        ;READ PAR TO R2
      BEQ 2$             ;BRANCH IF PAR IS 0
      ERROR +30         ;PAR NOT ZERO
2$:   ADD #2,R0          ;POINT TO NEXT REGISTER
      SOB R1,1$        ;BRANCH BACK TO 1$ 15 TIMES
;NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE REGISTER AND
;READING THE REST TO SEE ONLY THAT REGISTER IS NON ZERO. (ANY
;DROPPED BITS WILL BE FOUND IN THE NEXT TEST).
      MOV #3$, $LPERR    ;SET LOOP ON ERROR POINTER TO 3$
      MOV #UIPAR0,R0     ;PUT ADDRESS OF FIRST PAR IN R0
3$:   MOV #UIPAR0,R5     ;LOAD STARTING ADDRESS INTO R5
      JSR PC,CLRREG      ;CLEAR 16 REGISTERS POINTED TO BY R5
      MOV #UDPAR7,R1     ;PUT UDPAR7 ADDRESS INTO R1
      MOV #-1,(R0)       ;LOAD REGISTER UNDER TEST
4$:   CLR $TMP0         ;FLAG TO INDICATE THERE WAS A MATCH
      MOV (R1),R2        ;READ ALL REGISTERS
      BEQ 6$             ;BRANCH IF REGISTER IS 0
      CMP R0,R1          ;IS THE ADDRESS OF NON-ZERO REGISTER THE SAME
                        ;AS THE REGISTER UNDER TEST?
      BEQ 5$             ;BRANCH IF ADDRESSES MATCH
5$:   JSR PC,DUALADR     ;LOG AND REPORT ERRORS
      INC $TMP0         ;SET FLAG WHEN ADDRESSES MATCH
6$:   SUB #2,R1          ;POINT TO NEXT REGISTER
      CMP #UIPAR0,R1     ;SEE IF ALL REGISTERS HAVE BEEN READ
      BLOS 4$           ;BRANCH IF MORE TO READ
      ADD #2,R0          ;NOW LOAD THE NEXT REGISTER
      CMP #UDPAR7,R0     ;SEE IF THERE ARE MORE REGISTERS TO TEST
      BHIS 3$           ;BRANCH IF MORE REGISTERS TO TEST
      MOV #20$, $LPERR   ;SET LOOP ON ERROR POINTER TO START OF TEST
      TST ERRCNT        ;SEE IF THERE WERE ANY ERRORS
      BR TST23          ;BRANCH TO NEXT TEST IF NO ERRORS
      MOV ERRCNT,$TMP5   ;SAVE # OF ERRORS FOR PAR OUT
      ERROR +31         ;SUMMARY OF DUAL ADDRESS TEST

```

024014 012767 024026 155066  
024022 012700 177640  
024026 012705 177640  
024032 004767 156222  
024036 012701 177676  
024042 012710 177777  
024046 005067 155124  
024052 011102  
024054 001406  
024056 020001  
024060 001402  
024062 004767 156246  
024066 005267 155104  
024072 162701 000002  
024076 022701 177640  
024102 101761  
024104 062700 000002  
024110 022700 177676  
024114 103344  
024116 012767 023746 154764  
024124 005767 155172  
024130 000404  
024132 016767 155164 155050  
024140 104031

1929

```

:*****
:*TEST 23      DUAL ADDRESS KERNAL PDR'S,ON LOADING
:*
:*THIS TEST FIRST CLEARS ALL THE KERNEL PAGE DESCRIPTOR REGISTERS,
:*AND CHECKS TO SEE THAT THEY EACH HOLD ZERO. THEN, STARTING
:*WITH I-SPACE DESCRIPTOR REGISTER ZERO, ONE REGISTER AT A TIME IS
:*LOADED WITH A NEGATIVE ONE. ALL KERNEL DESCRIPTOR REGISTERS ARE
:*NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
:*INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN
:*AT THE END OF THIS TEST.
:*****

```

```

1930 024142 000004
024144 004767 156126
024150 012767 024176 154732
024156 012705 172300
024162 004767 156072
024166 012700 172300
024172 012701 000020
024176 011002
024200 001401
024202 104030
024204 062700 000002
024210 077106

```

```

TST23: SCOPE
20$: JSR PC,CLEANUP      ;INITIALIZE THE ERROR LOCATIONS
      MOV #1$, $LPERR    ;SET LOOP ON ERROR POINTER TO 1$
      MOV #KIPDRO,R5     ;PUT ADDRESS OF FIRST PDR IN R5
      JSR PC,CLRREG      ;CLEAR 16 REGISTERS POINTED TO BY R5
      MOV #KIPDRO,R0     ;PUT ADDRESS OF FIRST PDR IN R0
      MOV #20,R1         ;BRANCH COUNT IS 16 DECIMAL
1$:   MOV (R0),R2        ;READ PDR TO R2
      BEQ 2$             ;BRANCH IF PDR IS 0
      ERROR +30         ;PDR NOT ZERO
2$:   ADD #2,R0          ;POINT TO NEXT REGISTER
      SOB R1,1$        ;BRANCH BACK TO 1$ 15 TIMES

```

```

;NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE REGISTER AND
;READING THE REST TO SEE ONLY THAT REGISTER IS NON ZERO. (ANY
;DROPPED BITS WILL BE FOUND IN THE NEXT TEST).

```

```

024212 012767 024224 154670
024220 012700 172300
024224 012705 172300
024230 004767 156024
024234 012701 172336
024240 012710 177777
024244 005067 154726
024250 011102
024252 001406
024254 020001

```

```

      MOV #3$, $LPERR    ;SET LOOP ON ERROR POINTER TO 3$
      MOV #KIPDRO,R0     ;PUT ADDRESS OF FIRST PDR IN R0
3$:   MOV #KIPDRO,R5     ;LOAD STARTING ADDRESS INTO R5
      JSR PC,CLRREG      ;CLEAR 16 REGISTERS POINTED TO BY R5
      MOV #KDPDR7,R1     ;PUT KDPDR7 ADDRESS INTO R1
      MOV #-1,(R0)       ;LOAD REGISTER UNDER TEST
4$:   CLR $TMP0          ;FLAG TO INDICATE THERE WAS A MATCH
      MOV (R1),R2        ;READ ALL REGISTERS
      BEQ 6$             ;BRANCH IF REGISTER IS 0
      CMP R0,R1         ;IS THE ADDRESS OF NON-ZERO REGISTER THE SAME
                        ;AS THE REGISTER UNDER TEST?

```

```

024256 001402
024260 004767 156050
024264 005267 154706
024270 162701 000002
024274 022701 172300
024300 101761
024302 062700 000002
024306 022700 172336
024312 103344
024314 012767 024144 154566
024322 005767 154774
024326 000404
024330 016767 154766 154652
024336 104031

```

```

      BEQ 5$             ;BRANCH IF ADDRESSES MATCH
      JSR PC,DUALADR     ;LOG AND REPORT ERRORS
5$:   INC $TMP0          ;SET FLAG WHEN ADDRESSES MATCH
6$:   SUB #2,R1         ;POINT TO NEXT REGISTER
      CMP #KIPDRO,R1     ;SEE IF ALL REGISTERS HAVE BEEN READ
      BLOS 4$           ;BRANCH IF MORE TO READ
      ADD #2,R0          ;NOW LOAD THE NEXT REGISTER
      CMP #KDPDR7,R0     ;SEE IF THERE ARE MORE REGISTERS TO TEST
      BHS 3$            ;BRANCH IF MORE REGISTERS TO TEST
      MOV #20$, $LPERR   ;SET LOOP ON ERROR POINTER TO START OF TEST
      TST ERRCNT         ;SEE IF THERE WERE ANY ERRORS
      BR TST24          ;BRANCH TO NEXT TEST IF NO ERRORS
      MOV ERRCNT,$TMP5   ;SAVE # OF ERRORS FOR PDR OUT
      ERROR +31         ;SUMMARY OF DUAL ADDRESS TEST

```

1935

```

:*****
:*TEST 24      DUAL ADDRESS SUPERVISOR PDR'S,ON LOADING
:*
:*THIS TEST FIRST CLEARS ALL THE SUPERVISOR PAGE DESCRIPTOR REGISTERS,
:*AND CHECKS TO SEE THAT THEY EACH HOLD ZERO. THEN, STARTING
:*WITH I-SPACE DESCRIPTOR REGISTER ZERO, ONE REGISTER AT A TIME IS
:*LOADED WITH A NEGATIVE ONE. ALL SUPERVISOR DESCRIPTOR REGISTERS ARE
:*NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
:*INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN
:*AT THE END OF THIS TEST.
:*****
  
```

```

1936 024340 000004
024342 004767 155730
024346 012767 024374 154534
024354 012705 172200
024360 004767 155674
024364 012700 172200
024370 012701 000020
024374 011002
024376 001401
024400 104030
024402 062700 000002
024406 077106

024410 012767 024422 154472
024416 012700 172200
024422 012705 172200
024426 004767 155626
024432 012701 172236
024436 012710 177777
024442 005067 154530
024446 011102
024450 001406
024452 020001

024454 001402
024456 004767 155652
024462 005267 154510
024466 162701 000002
024472 022701 172200
024476 101761
024500 062700 000002
024504 022700 172236
024510 103344
024512 012767 024342 154370
024520 005767 154576
024524 000404
024526 016767 154570 154454
024534 104031
  
```

```

TST24: SCOPE
20$: JSR PC,CLEANUP ;INITIALIZE THE ERROR LOCATIONS
      MOV #1$, $LPERR ;SET LOOP ON ERROR POINTER TO 1$
      MOV #SIPDR0,R5 ;PUT ADDRESS OF FIRST PDR IN R5
      JSR PC,CLRREG ;CLEAR 16 REGISTERS POINTED TO BY R5
      MOV #SIPDR0,R0 ;PUT ADDRESS OF FIRST PDR IN R0
      MOV #20,R1 ;BRANCH COUNT IS 16 DECIMAL
1$: MOV (R0),R2 ;READ PDR TO R2
     BEQ 2$ ;BRANCH IF PDR IS 0
     ERROR +30 ;PDR NOT ZERO
2$: ADD #2,R0 ;POINT TO NEXT REGISTER
     SOB R1,1$ ;BRANCH BACK TO 1$ 15 TIMES
;NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE REGISTER AND
;READING THE REST TO SEE ONLY THAT REGISTER IS NON ZERO. (ANY
;DROPPED BITS WILL BE FOUND IN THE NEXT TEST).
      MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
      MOV #SIPDR0,R0 ;PUT ADDRESS OF FIRST PDR IN R0
3$: MOV #SIPDR0,R5 ;LOAD STARTING ADDRESS INTO R5
     JSR PC,CLRREG ;CLEAR 16 REGISTERS POINTED TO BY R5
     MOV #SDPDR7,R1 ;PUT SDPDR7 ADDRESS INTO R1
     MOV #-1,(R0) ;LOAD REGISTER UNDER TEST
4$: CLR $TMP0 ;FLAG TO INDICATE THERE WAS A MATCH
     MOV (R1),R2 ;READ ALL REGISTERS
     BEQ 6$ ;BRANCH IF REGISTER IS 0
     CMP R0,R1 ;IS THE ADDRESS OF NON-ZERO REGISTER THE SAME
     ;AS THE REGISTER UNDER TEST?
     BEQ 5$ ;BRANCH IF ADDRESSES MATCH
     JSR PC,DUALADR ;LOG AND REPORT ERRORS
5$: INC $TMP0 ;SET FLAG WHEN ADDRESSES MATCH
6$: SUB #2,R1 ;POINT TO NEXT REGISTER
     CMP #SIPDR0,R1 ;SEE IF ALL REGISTERS HAVE BEEN READ
     BLOS 4$ ;BRANCH IF MORE TO READ
     ADD #2,R0 ;NOW LOAD THE NEXT REGISTER
     CMP #SDPDR7,R0 ;SEE IF THERE ARE MORE REGISTERS TO TEST
     BHIS 3$ ;BRANCH IF MORE REGISTERS TO TEST
     MOV #20$, $LPERR ;SET LOOP ON ERROR POINTER TO START OF TEST
     TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
     BR TST25 ;BRANCH TO NEXT TEST IF NO ERRORS
     MOV ERRCNT,$TMP5 ;SAVE # OF ERRORS FOR PDR OUT
     ERROR +31 ;SUMMARY OF DUAL ADDRESS TEST
  
```

1941

```

*****
*TEST 25      DUAL ADDRESS USER PDR'S,ON LOADING
*
*THIS TEST FIRST CLEARS ALL THE USER PAGE DESCRIPTOR REGISTERS,
*AND CHECKS TO SEE THAT THEY EACH HOLD ZERO. THEN, STARTING
*WITH I-SPACE DESCRIPTOR REGISTER ZERO, ONE REGISTER AT A TIME IS
*LOADED WITH A NEGATIVE ONE. ALL USER DESCRIPTOR REGISTERS ARE
*NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
*INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN
*AT THE END OF THIS TEST.
*****
    
```

1942 024536 000004  
 024540 004767 155532  
 024544 012767 024572 154336  
 024552 012705 177600  
 024556 004767 155476  
 024562 012700 177600  
 024566 012701 000020  
 024572 011002  
 024574 001401  
 024576 104030  
 024600 062700 000002  
 024604 077106

```

TST25: SCOPE
20$: JSR PC,CLEANUP ;INITIALIZE THE ERROR LOCATIONS
      MOV #1$, $LPERR ;SET LOOP ON ERROR POINTER TO 1$
      MOV #UIPDR0,R5 ;PUT ADDRESS OF FIRST PDR IN R5
      JSR PC,CLRREG ;CLEAR 16 REGISTERS POINTED TO BY R5
      MOV #UIPDR0,R0 ;PUT ADDRESS OF FIRST PDR IN R0
      MOV #20,R1 ;BRANCH COUNT IS 16 DECIMAL
1$: MOV (R0),R2 ;READ PDR TO R2
     BEQ 2$ ;BRANCH IF PDR IS 0
     ERROR +30 ;PDR NOT ZERO
2$: ADD #2,R0 ;POINT TO NEXT REGISTER
     SOB R1,1$ ;BRANCH BACK TO 1$ 15 TIMES
;NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE REGISTER AND
;READING THE REST TO SEE ONLY THAT REGISTER IS NON ZERO. (ANY
;DROPPED BITS WILL BE FOUND IN THE NEXT TEST)
      MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
      MOV #UIPDR0,R0 ;PUT ADDRESS OF FIRST PDR IN R0
3$: MOV #UIPDR0,R5 ;LOAD STARTING ADDRESS INTO R5
     JSR PC,CLRREG ;CLEAR 16 REGISTERS POINTED TO BY R5
     MOV #UDPDR7,R1 ;PUT UDPDR7 ADDRESS INTO R1
     MOV #-1,(R0) ;LOAD REGISTER UNDER TEST
4$: CLR $TMP0 ;FLAG TO INDICATE THERE WAS A MATCH
     MOV (R1),R2 ;READ ALL REGISTERS
     BEQ 6$ ;BRANCH IF REGISTER IS 0
     CMP R0,R1 ;IS THE ADDRESS OF NON-ZERO REGISTER THE SAME
     ;AS THE REGISTER UNDER TEST?
     BEQ 5$ ;BRANCH IF ADDRESSES MATCH
     JSR PC,DUALADR ;LOG AND REPORT ERRORS
5$: INC $TMP0 ;SET FLAG WHEN ADDRESSES MATCH
6$: SUB #2,R1 ;POINT TO NEXT REGISTER
     CMP #UIPDR0,R1 ;SEE IF ALL REGISTERS HAVE BEEN READ
     BLOS 4$ ;BRANCH IF MORE TO READ
     ADD #2,R0 ;NOW LOAD THE NEXT REGISTER
     CMP #UDPDR7,R0 ;SEE IF THERE ARE MORE REGISTERS TO TEST
     BHIS 3$ ;BRANCH IF MORE REGISTERS TO TEST
     MOV #20$, $LPERR ;SET LOOP ON ERROR POINTER TO START OF TEST
     TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
     BR TST26 ;BRANCH TO NEXT TEST IF NO ERRORS
     MOV ERRCNT,$TMP5 ;SAVE # OF ERRORS FOR PDR OUT
     ERROR +31 ;SUMMARY OF DUAL ADDRESS TEST
    
```

1943  
 1959

```

*****
*TEST 26      COUNT PATTERN IN KERNAL PAR'S
*
*THIS TEST RUNS A COUNT PATTERN THROUGH THE KERNAL PAGE ADDRESS REGISTERS.
*IF THE COUNT PATTERN DOES NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS,
    
```

;\*DATA PATTERN, AND BAD DATA ARE REPORTED. AT THE END OF THE TEST A SUMMARY  
 :\*OF ERRORS IS GIVEN.

\*\*\*\*\*

1960 024734 000004  
 024736 004767 155334  
 024742 012767 024752 154140  
 024750 005001  
 024752 012700 172340  
 024756 010110  
 024760 011002  
 024762 010104  
 024764 020402  
 024766 001402  
 024770 004767 155404  
 024774 062700 000002  
 025000 022700 172376  
 025004 103364  
 025006 022701 177777  
 025012 001403  
 025014 062701 000401  
 025020 000754  
 025022 012767 024736 154060  
 025030 005767 154266  
 025034 000404  
 025036 016767 154260 154144  
 025044 104032

TST26: SCOPE  
 20\$: JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS  
 MOV #1\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 1\$  
 CLR R1 ;CLEAR REGISTER TO HOLD COUNT PATTERN  
 1\$: MOV #KIPAR0,R0 ;PUT ADDRESS OF FIRST REGISTER IN'0 R0  
 2\$: MOV R1,(R0) ;LOAD COUNT INTO REGISTER  
 MOV (R0),R2 ;READ REGISTER BACK TO R2  
 MOV R1,R4 ;PUT PATTERN IN R4  
 CMP R4,R2 ;SEE IF DATA MATCHES PATTERN  
 BEQ 3\$ ;BRANCH IF DATA IS GOOD  
 JSR PC,PARCOUNT ;LOG AND REPORT COUNT ERROR  
 3\$: ADD #2,R0 ;POINT TO NEXT REGISTER  
 CMP #KDPAR7,R0 ;SEE IF YOU PASSED THE KDPAR7 PAR  
 BHIS 2\$ ;BRANCH IF MORE TEST  
 CMP #177777,R1 ;SEE IF COUNT HAS REACHED 177777  
 BEQ 4\$ ;BRANCH IF SO  
 ADD #401,R1 ;INCREASE COUNT PATTERN  
 BR 1\$ ;BRANCH TO CONTINUE TEST  
 4\$: MOV #20\$, \$LPERR ;SET LOOP POINTER TO START OF TEST  
 TST ERRCNT ;SEE IF THERE WERE ANY ERRORS  
 BR TST27 ;BRANCH TO NEXT TEST IF NO ERRORS  
 MOV ERRCNT,\$TMP5 ;SAVE # OF ERRORS FOR TYPEOUT  
 FRROR +32 ;SUMMARY OF COUNT PATTERN FAILURES

1961  
 1965

\*\*\*\*\*

;\*TEST 27 COUNT PATTERN IN SUPERVISOR PAR'S

;\*THIS TEST RUNS A COUNT PATTERN THROUGH THE SUPERVISOR PAGE ADDRESS REGISTERS.  
 :\*IF THE COUNT PATTERN DOES NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS,  
 :\*DATA PATTERN, AND BAD DATA ARE REPORTED. AT THE END OF THE TEST A SUMMARY  
 :\*OF ERRORS IS GIVEN.

\*\*\*\*\*

1966 025046 000004  
 025050 004767 155222  
 025054 012767 025064 154026  
 025062 005001  
 025064 012700 172240  
 025070 010110  
 025072 011002  
 025074 010104  
 025076 020402  
 025100 001402  
 025102 004767 155272  
 025106 062700 000002  
 025112 022700 172276  
 025116 103364  
 025120 022701 177777  
 025124 001403  
 025126 062701 000401  
 025132 000754  
 025134 012767 025050 153746  
 025142 005767 154154  
 025146 000404  
 025150 016767 154146 154032

TST27: SCOPE  
 20\$: JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS  
 MOV #1\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 1\$  
 CLR R1 ;CLEAR REGISTER TO HOLD COUNT PATTERN  
 1\$: MOV #SIPAR0,R0 ;PUT ADDRESS OF FIRST REGISTER INTO R0  
 2\$: MOV R1,(R0) ;LOAD COUNT INTO REGISTER  
 MOV (R0),R2 ;READ REGISTER BACK TO R2  
 MOV R1,R4 ;PUT PATTERN IN R4  
 CMP R4,R2 ;SEE IF DATA MATCHES PATTERN  
 BEQ 3\$ ;BRANCH IF DATA IS GOOD  
 JSR PC,PARCOUNT ;LOG AND REPORT COUNT ERROR  
 3\$: ADD #2,R0 ;POINT TO NEXT REGISTER  
 CMP #SDPAR7,R0 ;SEE IF YOU PASSED THE SDPAR7 PAR  
 BHIS 2\$ ;BRANCH IF MORE TEST  
 CMP #177777,R1 ;SEE IF COUNT HAS REACHED 177777  
 BEQ 4\$ ;BRANCH IF SO  
 ADD #401,R1 ;INCREASE COUNT PATTERN  
 BR 1\$ ;BRANCH TO CONTINUE TEST  
 4\$: MOV #20\$, \$LPERR ;SET LOOP POINTER TO START OF TEST  
 TST ERRCNT ;SEE IF THERE WERE ANY ERRORS  
 BR TST30 ;BRANCH TO NEXT TEST IF NO ERRORS  
 MOV ERRCNT,\$TMP5 ;SAVE # OF ERRORS FOR TYPEOUT



1967 025156 104032  
 1971

ERROR +32 ; SUMMARY OF COUNT PATTERN FAILURES

```

:*****
:*TEST 30      COUNT PATTERN IN USER PAR'S
:*
:*THIS TEST RUNS A COUNT PATTERN THROUGH THE USER PAGE ADDRESS REGISTERS.
:*IF THE COUNT PATTERN DOES NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS,
:*DATA PATTERN, AND BAD DATA ARE REPORTED.  AT THE END OF THE TEST A SUMMARY
:*OF ERRORS IS GIVEN.
:*****
  
```

1972 025160 000004  
 025162 004767 155110  
 025166 012767 025176 153714  
 025174 005001  
 025176 012700 172340  
 025202 010110  
 025204 011002  
 025206 010104  
 025210 020402  
 025212 001402  
 025214 004767 155160  
 025220 062700 000002  
 025224 022700 172376  
 025230 103364  
 025232 022701 177777  
 025236 001403  
 025240 062701 000401  
 025244 000754  
 025246 012767 025162 153634  
 025254 005767 154042  
 025260 000404  
 025262 016767 154034 153720  
 025270 104032

```

TST30:  SCOPE
20$:   JSR    PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
        MOV    #1$, $LPERR    ;SET LOOP ON ERROR POINTER TO 1$
        CLR    R1             ;CLEAR REGISTER TO HOLD COUNT PATTERN
1$:    MOV    #KIPAR0,R0      ;PUT ADDRESS OF FIRST REGISTER INTO R0
2$:    MOV    R1,(R0)         ;LOAD COUNT INTO REGISTER
        MOV    (R0),R2        ;READ REGISTER BACK TO R2
        MOV    R1,R4         ;PUT PATTERN IN R4
        CMP    R4,R2         ;SEE IF DATA MATCHES PATTERN
        BEQ    3$            ;BRANCH IF DATA IS GOOD
        JSR    PC,PARCOUNT  ;LOG AND REPORT COUNT ERROR
3$:    ADD    #2,R0           ;POINT TO NEXT REGISTER
        CMP    #KDPAR7,R0    ;SEE IF YOU PASSED THE KDPAR7 PAR
        BHS    2$           ;BRANCH IF MORE TEST
        CMP    #177777,R1    ;SEE IF COUNT HAS REACHED 177777
        BEQ    4$           ;BRANCH IF SO
        ADD    #401,R1       ;INCREASE COUNT PATTERN
        BR    1$            ;BRANCH TO CONTINUE TEST
4$:    MOV    #20$, $LPERR    ;SET LOOP POINTER TO START OF TEST
        TST   ERRCNT         ;SEE IF THERE WERE ANY ERRORS
        BR    TST31        ;BRANCH TO NEXT TEST IF NO ERRORS
        MOV    ERRCNT,$TMP5  ;SAVE # OF ERRORS FOR TYPEOUT
        ERROR +32          ;SUMMARY OF COUNT PATTERN FAILURES
  
```

1973  
 1977

```

:*****
:*TEST 31      COUNT PATTERN IN KERNAL PDR'S
:*
:*THIS TEST RUNS A COUNT PATTERN THROUGH THE KERNAL PAGE DESCRIPTOR REGISTERS.
:*SINCE BITS <05:04> AND <0> ARE NOT IMPLEMENTED, AND BITS <07:06>
:*CANNOT BE SET BY DIRECT LOAD, THEY ARE MASKED OUT OF THE DATA COMPARE.
:*THESE BITS ARE STILL SENT TO THE DESCRIPTOR REGISTER TO FIND BAD ETCHES.
:*IF THE COUNT PATTERN DOES NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS,
:*DATA PATTERN, AND BAD DATA ARE REPORTED.  AT THE END OF THE TEST A SUMMARY
:*OF ERRORS IS GIVEN.
:*****
  
```

1978 025272 000004  
 025274 004767 154776  
 025300 012767 025310 153602  
 025306 005001  
 025310 012700 172300  
 025314 010110  
 025316 011002  
 025320 010104  
 025322 042704 000361  
 025326 020402  
 025330 001402  
 025332 004767 155042

```

TST31:  SCOPE
20$:   JSR    PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
        MOV    #1$, $LPERR    ;SET LOOP ON ERROR POINTER TO 1$
        CLR    R1             ;CLEAR REGISTER TO HOLD COUNT PATTERN
1$:    MOV    #KIPDR0,R0      ;PUT ADDRESS OF FIRST REGISTER INTO R0
2$:    MOV    R1,(R0)         ;LOAD COUNT INTO REGISTER
        MOV    (R0),R2        ;READ REGISTER BACK TO R2
        MOV    R1,R4         ;PUT PATTERN IN R4
        BIC    #000361,R4    ;CLEAR BITS NOT FOUND IN REGISTER
        CMP    R4,R2         ;SEE IF DATA MATCHES PATTERN
        BEQ    3$            ;BRANCH IF DATA IS GOOD
        JSR    PC,PARCOUNT  ;LOG AND REPORT COUNT ERROR
  
```

```
025336 062700 000002      3$:  ADD    #2,R0      ;POINT TO NEXT REGISTER
025342 022700 172336      CMP    #KDPDR7,R0 ;SEE IF YOU PASSED THE KDPDR7 PDR
025346 103362      BHIS   2$         ;BRANCH IF MORE TEST
025350 022701 177777      CMP    #177777,R1 ;SEE IF COUNT HAS REACHED 177777
025354 001403      BEQ   4$         ;BRANCH IF SO
025356 062701 000401      ADD    #401,R1    ;INCREASE COUNT PATTERN
025362 000752      BR    1$         ;BRANCH TO CONTINUE TEST
025364 012767 025274 153516 4$:  MOV    #20$,$LPERR ;SET LOOP POINTER TO START OF TEST
025372 005767 153724      TST   ERRCNT     ;SEE IF THERE WERE ANY ERRORS
025376 000404      BR    TST32     ;:BRANCH TO NEXT TEST IF NO ERRORS
025400 016767 153716 153602  MOV    ERRCNT,$TMP5 ;SAVE # OF ERRORS FOR TYPEOUT
025406 104032      ERROR +32       ;SUMMARY OF COUNT PATTERN FAILURES
```

1979

1984

```

:*****
:*TEST 32      COUNT PATTERN IN SUPERVISOR PDR'S
:*
:*THIS TEST RUNS A COUNT PATTERN THROUGH THE SUPERVISOR PAGE DESCRIPTOR REGISTERS.
:*SINCE BITS <05:04> AND <0> ARE NOT IMPLEMENTED, AND BITS <07:06>
:*CANNOT BE SET BY DIRECT LOAD, THEY ARE MASKED OUT OF THE DATA COMPARE.
:*THESE BITS ARE STILL SENT TO THE DESCRIPTOR REGISTER TO FIND BAD ETCHES.
:*IF THE COUNT PATTERN DOES NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS,
:*DATA PATTERN, AND BAD DATA ARE REPORTED.  AT THE END OF THE TEST A SUMMARY
:*OF ERRORS IS GIVEN.
:*****
    
```

```

1985 025410 000004
      025412 004767 154660
      025416 012767 025426 153464
      025424 005001
      025426 012700 172200
      025432 010110
      025434 011002
      025436 010104
      025440 042704 000361
      025444 020402
      025446 001402
      025450 004767 154724
      025454 062700 000002
      025460 022700 172236
      025464 103362
      025466 022701 177777
      025472 001403
      025474 062701 000401
      025500 000752
      025502 012767 025412 153400
      025510 005767 153606
      025514 000404
      025516 016767 153600 153464
      025524 104032
    
```

```

TST32: SCOPE
20$: JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
      MOV #1$, $LPERR ;SET LOOP ON ERROR POINTER TO 1$
      CLR R1 ;CLEAR REGISTER TO HOLD COUNT PATTERN
1$: MOV #SIPDR0,R0 ;PUT ADDRESS OF FIRST REGISTER INTO R0
2$: MOV R1,(R0) ;LOAD COUNT INTO REGISTER
      MOV (R0),R2 ;READ REGISTER BACK TO R2
      MOV R1,R4 ;PUT PATTERN IN R4
      BIC #000361,R4 ;CLEAR BITS NOT FOUND IN REGISTER
      CMP R4,R2 ;SEE IF DATA MATCHES PATTERN
      BEQ 3$ ;BRANCH IF DATA IS GOOD
      JSR PC,PARCOUNT ;LOG AND REPORT COUNT ERROR
3$: ADD #2,R0 ;POINT TO NEXT REGISTER
      CMP #SDPDR7,R0 ;SEE IF YOU PASSED THE SDPDR7 PDR
      BHIS 2$ ;BRANCH IF MORE TEST
      CMP #177777,R1 ;SEE IF COUNT HAS REACHED 177777
      BEQ 4$ ;BRANCH IF SO
      ADD #401,R1 ;INCREASE COUNT PATTERN
      BR 1$ ;BRANCH TO CONTINUE TEST
4$: MOV #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
      TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
      BR ;BRANCH TO NEXT TEST IF NO ERRORS
      MOV ERRCNT,$TMP5 ;SAVE # OF ERRORS FOR TYPEOUT
      ERROR +32 ;SUMMARY OF COUNT PATTERN FAILURES
    
```

1986  
1990

```

:*****
:*TEST 33      COUNT PATTERN IN USER PDR'S
:*
:*THIS TEST RUNS A COUNT PATTERN THROUGH THE USER PAGE ADDRESS REGISTERS.
:*SINCE BITS <05:04> AND <0> ARE NOT IMPLEMENTED, AND BITS <07:06>
:*CANNOT BE SET BY DIRECT LOAD, THEY ARE MASKED OUT OF THE DATA COMPARE.
:*THESE BITS ARE STILL SENT TO THE DESCRIPTOR REGISTER TO FIND BAD ETCHES.
:*IF THE COUNT PATTERN DOES NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS,
:*DATA PATTERN, AND BAD DATA ARE REPORTED.  AT THE END OF THE TEST A SUMMARY
:*OF ERRORS IS GIVEN.
:*****
    
```

```

1991 025526 000004
      025530 004767 154542
      025534 012767 025544 153346
      025542 005001
      025544 012700 177600
      025550 010110
      025552 011002
      025554 010104
      025556 042704 000361
      025562 020402
    
```

```

TST33: SCOPE
20$: JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
      MOV #1$, $LPERR ;SET LOOP ON ERROR POINTER TO 1$
      CLR R1 ;CLEAR REGISTER TO HOLD COUNT PATTERN
1$: MOV #UIPDR0,R0 ;PUT ADDRESS OF FIRST REGISTER INTO R0
2$: MOV R1,(R0) ;LOAD COUNT INTO REGISTER
      MOV (R0),R2 ;READ REGISTER BACK TO R2
      MOV R1,R4 ;PUT PATTERN IN R4
      BIC #000361,R4 ;CLEAR BITS NOT FOUND IN REGISTER
      CMP R4,R2 ;SEE IF DATA MATCHES PATTERN
    
```

```

025564 001402          BEQ      3$          ;BRANCH IF DATA IS GOOD
025566 004767 154606   JSR      PC,PARCOUNT ;LOG AND REPORT COUNT ERROR
025572 062700 000002   3$:    ADD      #2,R0       ;POINT TO NEXT REGISTER
025576 022700 177636   CMP      #UDPDR7,RC     ;SEE IF YOU PASSED THE UDPDR7 PDR
025602 103362          BH,S      2$          ;BRANCH IF MORE TEST
025604 022701 177777   CMP      #177777,R1    ;SEE IF COUNT HAS REACHED 177777
025610 001403          BEQ      4$          ;BRANCH IF SO
025612 062701 000401   ADD      #401,R1       ;INCREASE COUNT PATTERN
025616 000752          BR       1$          ;BRANCH TO CONTINUE TEST
025620 012767 025530 153262 4$:    MOV      #20$,$LPERR   ;SET LOOP POINTER TO START OF TEST
025626 005767 153470   TST      ERRCNT        ;SEE IF THERE WERE ANY ERRORS
025632 000404          BR       TST34        ;BRANCH TO NEXT TEST IF NO ERRORS
025634 016767 153462 153346 MOV      ERRCNT,$TMP5  ;SAVE # OF ERRORS FOR TYPEOUT
025642 104032          ERROR    +32         ;SUMMARY OF COUNT PATTERN FAILURES
  
```

1992  
2000  
2004

```

*****
*TEST 34          BYTE ADDRESSING OF KERNAL PAR'S
*
*   THIS TEST CHECK THE LOGIC ASSOCIATED WITH BYTE ADDRESSING
*   OF KERNAL PAGE ADDRESS REGISTERS.  IT CAN BE ASSUMED THAT IF
*   ONE REGISTER WORKS, THEY ALL WORK BECAUSE THE REGISTERS
*   HAVE ALL BEEN TESTED.
*****
  
```

```

2005 025644 000004          TST34: SCOPE
025646 012767 025664 153234 20$:  MOV      #11$,$LPERR   ;SET LOOP ON ERROR POINTER TO 11$
025654 012702 000014          MOV      #14,R2       ;PUT EXPECTED DATA IN R2
025660 012700 172340          MOV      #KIPAR0,R0   ;PUT ADDRESS OF REGISTER IN R0
025664 005067 144450          1$:    CLR      KIPAR0       ;CLEAR THE REGISTER UNDER TEST
025670 112710 172014          MOVVB   #172014,(R0)  ;LOAD LOWER BYTE OF REGISTER
025674 016701 144440          MOV      KIPAR0,R1   ;READ REGISTER INTO R1
025700 020102          CMP      R1,R2       ;SEE IF ONLY LOWER BYTE WAS WRITTEN
025702 001401          BEQ      1$          ;BRANCH IF DATA MATCHES
025704 104033          ERROR    +33        ;DIDN'T LOAD CORRECT BYTE
025706 012767 025724 153174 1$:    MOV      #12$,$LPERR   ;SET LOOP ON ERROR POINTER TO 12$
025714 012700 172341          MOV      #KIPAR0+1,R0 ;POINT TO UPPER BYTE OF REGISTER
025720 012702 052014          MOV      #52014,R2   ;LOAD EXPECTED DATA INTO R2
025724 112710 000124          12$:  MOVVB   #124,(R0)    ;WRITE UPPER BYTE OF REGISTER
025730 016701 144404          MOV      KIPAR0,R1   ;READ REGISTER INTO R1
025734 020102          CMP      R1,R2       ;SEE IF ONLY UPPER BYTE WAS WRITTEN
025736 001401          BEQ      2$          ;BRANCH TO EXIT IF CORRECT
025740 104033          ERROR    +33        ;DIDN'T LOAD CORRECT BYTE
2009 025742 012767 025646 153140 2$:    MOV      #20$,$LPERR   ;SET LOOP POINTER TO START OF TEST
  
```

2009

```

*****
*TEST 35          BYTE ADDRESSING OF SUPERVISOR PAR'S
*
*   THIS TEST CHECK THE LOGIC ASSOCIATED WITH BYTE ADDRESSING
*   OF SUPERVISOR PAGE ADDRESS REGISTERS.  IT CAN BE ASSUMED THA :F
*   ONE REGISTER WORKS, THEY ALL WORK BECAUSE THE REGISTERS
*   HAVE ALL BEEN TESTED.
*****
  
```

```

2010 025750 000004          TST35: SCOPE
025752 012767 025770 153130 20$:  MOV      #11$,$LPERR   ;SET LOOP ON ERROR POINTER TO 11$
025760 012702 000014          MOV      #14,R2       ;PUT EXPECTED DATA IN R2
025764 012700 172240          MOV      #SIPAR0,R0   ;PUT ADDRESS OF REGISTER IN R0
025770 005067 144244          11$:  CLR      SIPAR0       ;CLEAR THE REGISTER UNDER TEST
025774 112710 172014          MOVVB   #172014,(R0)  ;LOAD LOWER BYTE OF REGISTER
  
```

```

026000 016701 144234      MOV     SIPARO,R1      ;READ REGISTER INTO R1
026004 020102           CMP     R1,R2         ;SEE IF ONLY LOWER BYTE WAS WRITTEN
026006 001401           BEQ    1$             ;BRANCH IF DATA MATCHES
026010 104033           ERROR  +33           ;DIDN'T LOAD CORRECT BYTE
026012 012767 026030 153070 1$:  MOV     #12$,$LPERR   ;SET LOOP ON ERROR POINTER TO 12$
026020 0127C0 172241     MOV     #SIPARO+1,R0  ;POINT TO UPPER BYTE OF REGISTER
026024 012702 052014     MOV     #52014,R2     ;LOAD EXPECTED DATA INTO R2
026030 112710 000124     12$:  MOVVB  #124,(R0)     ;WRITE UPPER BYTE OF REGISTER
026034 016701 144200     MOV     SIPARO,R1     ;READ REGISTER INTO R1
026040 020102           CMP     R1,R2         ;SEE IF ONLY UPPER BYTE WAS WRITTEN
026042 001401           BEQ    2$             ;BRANCH TO EXIT IF CORRECT
026044 104033           ERROR  +33           ;DIDN'T LOAD CORRECT BYTE
2014 026046 012767 025752 153034 2$:  MOV     #20$,$LPERR   ;SET LOOP POINTER TO START OF TEST
  
```

```

:*****
:*TEST 36      BYTE ADDRESSING OF USER PAR'S
:*
:*      THIS TEST CHECK THE LOGIC ASSOCIATED WITH BYTE ADDRESSING
:*      OF USER PAGE ADDRESS REGISTERS.  IT CAN BE ASSUMED THAT IF
:*      ONE REGISTER WORKS, THEY ALL WORK BECAUSE THE REGISTERS
:*      HAVE ALL BEEN TESTED.
  
```

```

2015 026054 000004      TST36: SCOPE
026056 012767 026074 153024 20$:  MOV     #11$,$LPERR   ;SET LOOP ON ERROR POINTER TO 11$
026064 012702 000014     MOV     #14,R2        ;PUT EXPECTED DATA IN R2
026070 012700 177640     MOV     #UIPARO,R0    ;PUT ADDRESS OF REGISTER IN R0
026074 005067 151540     11$:  CLR     UIPARO        ;CLEAR THE REGISTER UNDER TEST
026100 112710 172014     MOVVB  #172014,(R0)   ;LOAD LOWER BYTE OF REGISTER
026104 016701 151530     MOV     UIPARO,R1     ;READ REGISTER INTO R1
026110 020102           CMP     R1,R2         ;SEE IF ONLY LOWER BYTE WAS WRITTEN
026112 001401           BEQ    1$             ;BRANCH IF DATA MATCHES
026114 104033           ERROR  +33           ;DIDN'T LOAD CORRECT BYTE
026116 012767 026134 152764 1$:  MOV     #12$,$LPERR   ;SET LOOP ON ERROR POINTER TO 12$
026124 012700 177641     MOV     #UIPARO+1,R0  ;POINT TO UPPER BYTE OF REGISTER
026130 012702 052014     MOV     #52014,R2     ;LOAD EXPECTED DATA INTO R2
026134 112710 000124     12$:  MOVVB  #124,(R0)     ;WRITE UPPER BYTE OF REGISTER
026140 016701 151474     MOV     UIPARO,R1     ;READ REGISTER INTO R1
026144 020102           CMP     R1,R2         ;SEE IF ONLY UPPER BYTE WAS WRITTEN
026146 001401           BEQ    2$             ;BRANCH TO EXIT IF CORRECT
026150 104033           ERROR  +33           ;DIDN'T LOAD CORRECT BYTE
2019 026152 012767 026056 152730 2$:  MOV     #20$,$LPERR   ;SET LOOP POINTER TO START OF TEST
  
```

```

:*****
:*TEST 37      BYTE ADDRESSING OF KERNAL PDR'S
:*
:*      THIS TEST CHECK THE LOGIC ASSOCIATED WITH BYTE ADDRESSING
:*      OF KERNAL PAGE DESCRIPTOR REGISTERS.  IT CAN BE ASSUMED THAT IF
:*      ONE REGISTER WORKS, THEY ALL WORK BECAUSE THE REGISTERS
:*      HAVE ALL BEEN TESTED.
  
```

```

2020 026160 000004      TST37: SCOPE
026162 012767 026200 152720 20$:  MOV     #11$,$LPERR   ;SET LOOP ON ERROR POINTER TO 11$
026170 012702 000014     MOV     #14,R2        ;PUT EXPECTED DATA IN R2
026174 012700 172300     MOV     #KIPDR0,R0    ;PUT ADDRESS OF REGISTER IN R0
026200 005067 144074     11$:  CLR     KIPDR0        ;CLEAR THE REGISTER UNDER TEST
026204 112710 172014     MOVVB  #172014,(R0)   ;LOAD LOWER BYTE OF REGISTER
026210 016701 144064     MOV     KIPDR0,R1     ;READ REGISTER INTO R1
026214 020102           CMP     R1,R2         ;SEE IF ONLY LOWER BYTE WAS WRITTEN
026216 001401           BEQ    1$             ;BRANCH IF DATA MATCHES
  
```

```

026220 104033          ERROR +33          ;DIDN'T LOAD CORRECT BYTE
026222 012767 026240 152660 1$:  MOV #12$, $LPERR ;SET LOOP ON ERROR POINTER TO 12$
026230 012700 172301          MOV #KIPDR0+1, R0 ;POINT TO UPPER BYTE OF REGISTER
026234 012702 052014          MOV #52014, R2    ;LOAD EXPECTED DATA INTO R2
026240 112710 000124          12$: MOVB #124, (R0)   ;WRITE UPPER BYTE OF REGISTER
026244 016701 144030          MOV KIPDR0, R1   ;READ REGISTER INTO R1
026250 020102          CMP R1, R2      ;SEE IF ONLY UPPER BYTE WAS WRITTEN
026252 001401          BEQ 2$         ;BRANCH TO EXIT IF CORRECT
026254 104033          ERROR +33          ;DIDN'T LOAD CORRECT BYTE
026256 012767 026162 152624 2$:  MOV #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
    
```

2024

```

:*****
: *TEST 40          BYTE ADDRESSING OF SUPERVISOR PDR'S
: *
    
```

```

: * THIS TEST CHECK THE LOGIC ASSOCIATED WITH BYTE ADDRESSING
: * OF KERNAL PAGE DESCRIPTOR REGISTERS. IT CAN BE ASSUMED THAT IF
: * ONE REGISTER WORKS, THEY ALL WORK BECAUSE THE REGISTERS
: * HAVE ALL BEEN TESTED.
: *****
    
```

```

2025 026264 000004          TST40: SCOPE
026266 012767 026304 152614 20$:  MOV #11$, $LPERR ;SET LOOP ON ERROR POINTER TO 11$
026274 012702 000014          MOV #14, R2     ;PUT EXPECTED DATA IN R2
026300 012700 172200          MOV #SIPDR0, R0 ;PUT ADDRESS OF REGISTER IN R0
026304 005067 143670          11$: CLR SIPDR0     ;CLEAR THE REGISTER UNDER TEST
026310 112710 172014          MOVB #172014, (R0) ;LOAD LOWER BYTE OF REGISTER
026314 016701 143660          MOV SIPDR0, R1 ;READ REGISTER INTO R1
026320 020102          CMP R1, R2    ;SEE IF ONLY LOWER BYTE WAS WRITTEN
026322 001401          BEQ 1$       ;BRANCH IF DATA MATCHES
026324 104033          ERROR +33    ;DIDN'T LOAD CORRECT BYTE
026326 012767 026344 152554 1$:  MOV #12$, $LPERR ;SET LOOP ON ERROR POINTER TO 12$
026334 012700 172201          MOV #SIPDR0+1, R0 ;POINT TO UPPER BYTE OF REGISTER
026340 012702 052014          MOV #52014, R2    ;LOAD EXPECTED DATA INTO R2
026344 112710 000124          12$: MOVB #124, (R0)   ;WRITE UPPER BYTE OF REGISTER
026350 016701 143624          MOV SIPDR0, R1   ;READ REGISTER INTO R1
026354 020102          CMP R1, R2    ;SEE IF ONLY UPPER BYTE WAS WRITTEN
026356 001401          BEQ 2$       ;BRANCH TO EXIT IF CORRECT
026360 104033          ERROR +33    ;DIDN'T LOAD CORRECT BYTE
026362 012767 026266 152520 2$:  MOV #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
    
```

2029

```

:*****
: *TEST 41          BYTE ADDRESSING OF USER PDR'S
: *
    
```

```

: * THIS TEST CHECK THE LOGIC ASSOCIATED WITH BYTE ADDRESSING
: * OF USER PAGE DESCRIPTOR REGISTERS. IT CAN BE ASSUMED THAT IF
: * ONE REGISTER WORKS, THEY ALL WORK BECAUSE THE REGISTERS
: * HAVE ALL BEEN TESTED.
: *****
    
```

```

2030 026370 000004          TST41: SCOPE
026372 012767 026410 152510 20$:  MOV #11$, $LPERR ;SET LOOP ON ERROR POINTER TO 11$
026400 012702 000014          MOV #14, R2     ;PUT EXPECTED DATA IN R2
026404 012700 177600          MOV #UIPDR0, R0 ;PUT ADDRESS OF REGISTER IN R0
026410 005067 151164          11$: CLR UIPDR0     ;CLEAR THE REGISTER UNDER TEST
026414 112710 172014          MOVB #172014, (R0) ;LOAD LOWER BYTE OF REGISTER
026420 016701 151154          MOV UIPDR0, R1 ;READ REGISTER INTO R1
026424 020102          CMP R1, R2    ;SEE IF ONLY LOWER BYTE WAS WRITTEN
026426 001401          BEQ 1$       ;BRANCH IF DATA MATCHES
026430 104033          ERROR +33    ;DIDN'T LOAD CORRECT BYTE
026432 012767 026450 152450 1$:  MOV #12$, $LPERR ;SET LOOP ON ERROR POINTER TO 12$
026440 012700 177601          MOV #UIPDR0+1, R0 ;POINT TO UPPER BYTE OF REGISTER
    
```

```
026444 012702 052014          MOV      #52014,R2      ;LOAD EXPECTED DATA INTO R2
026450 112710 000124      12$:  MOVB   #124,(R0)      ;WRITE UPPER BYTE OF REGISTER
026454 016701 151120          MOV     UIPDR0,R1      ;READ REGISTER INTO R1
026460 020102              CMP     R1,R2          ;SEE IF ONLY UPPER BYTE WAS WRITTEN
026462 001401              BEQ    2$              ;BRANCH TO EXIT IF CORRECT
026464 104033              ERROR  +33              ;DIDN'T LOAD CORRECT BYTE
026466 012767 026372 152414 2$:  MOV     #20$,$LPERR    ;SET LOOP POINTER TO START OF TEST
```

2045

```

:*****
:*TEST 42      DUAL ADDRESSING FOR ALL PAR'S & PDR'S
:*
:*      THIS TEST WILL ENSURE THAT THERE IS NO DUAL ADDRESSING
:*      BETWEEN GROUPS OF PAR'S AND PDR'S.  THAT IS, WHEN YOU
:*      REFERENCE A KERNAL I-SPACE PAR YOU ARE REALLY REFERENCING
:*      THAT PAR.  FIRST EACH I-SPACE PAR0 OR PDR0 IS LOADED WITH
:*      A UNIQUE NUMBER 0-12 AND THEN THEY ARE EACH CHECKED FOR
:*      THE CORRECT DATA.
:*      EACH GROUP HAS ALREADY BEEN TESTED FOR DUAL ADDRESSING
:*      WITHIN ITS OWN GROUP, SO ONLY ONE REGISTER FROM EACH
:*      GROUP NEEDS TO BE TESTED.
:*****
  
```

```

TST42: SCOPE
2046 026474 000004      20$:  MOV      #1$, $LPERR      ;SET LOOP ON ERROR POINTER TO 1$
2047 026476 012767      CLR      R0                  ;THIS WILL HOLD THE INDEX OF EACH
2048 026504 005000                                ;REGISTER, AND THE COUNT LOADED.
2049 026506 012704 000006                                ;NUMBER OF TIMES TO DO THE LOOP
2050 026512 010070 001356      1$:  MOV      R0, @PARTAB(R0)    ;LOAD PAR OR PDR WITH INDEX NUMBER
2051 026516 062700 000002      ADD      #2, R0              ;CHANGE INDEX TO POINT TO NEXT REGISTER
2052 026522 077405                                SOB      R4, 1$             ;BRANCH BACK 5 TIMES
2053 026524 012704 000006                                MOV      #6, R4             ;DO NEXT LOOP 6 TIMES
2054 026530 016767 000006      152352  MOV      10$, $LPERR        ;SET LOOP ON ERROR POINTER TO 10$
2055 026536 162700 000002      2$:  SUB      #2, R0           ;ADJUST INDEX FOR REGISTER DESIRED
2056 026542 017001 001356      10$:  MOV      @PARTAB(R0), R1  ;READ PAR OR PDR INTO R1
2057 026546 020001                                CMP      R0, R1             ;SEE IF INDEX EQUALS DATA
2058 026550 001403                                BEQ      3$                 ;BRANCH IF INCORRECT REGISTER WAS READ
2059 026552 016002 001356      MOV      PARTAB(R0), R2     ;SAVE ADDRESS OF BAD REGISTER
2060 026556 104016                                ERROR   +16                 ;DUAL ADDRESSING ERROR
2061 026560 077412                                SOB      R4, 2$             ;BRANCH BACK 5 TIMES
2062 026562 012767 026476 152320 3$:  MOV      #20$, $LPERR      ;SET LOOP ON ERROR POINTER TO START OF TEST
2063
2073
  
```

```

:*****
:*TEST 43      TEST THAT PAR-PDR'S NOT AFFECTED BY RESET
:*
:*      THIS TEST CHECKS TO SEE THAT THE KERNEL, SUPERVISOR OR USER
:*      PAR/PDR'S ARE NOT AFFECTED BY THE EXECUTION OF A 'RESET'
:*      INSTRUCTION.  ALL WRITEABLE BITS ARE SET TO A '1' IN THE
:*      PAR/PDR'S; THEY ARE THEN READ TO SEE THAT THEY REMAINED
:*      UNCHANGED.
:*****
  
```

```

TST43: SCOPE
2074 026570 000004
2075
2076 026572 012703 000020      1$:  MOV      #20, R3          ;LOAD LOOP COUNTER WITH A 16
2077 026576 012701 172300      MOV      #KIPDR0, R1        ;LOAD FIRST ADDRESS OF PDR INTO R1
2078 026602 012702 172340      MOV      #KIPAR0, R2        ;LOAD FIRST ADDRESS OF PAR INTO R2
2079 026606 012721 177777      21$:  MOV      #-1, (R1)+     ;SET BITS IN PDR TO 1'S
2080 026612 012722 177777      MOV      #-1, (R2)+     ;SET BITS IN PAR'S TO 1'S
2081 026616 077305                                SOB      R3, 21$         ;LOOP UNTIL ALL KERNAL PAR/PDR'S LOADED
2082 026620 012703 000020      MOV      #20, R3          ;LOAD LOOP COUNTER WITH A 16
2083 026624 012701 172200      MOV      #SIPDR0, R1        ;LOAD FIRST ADDRESS OF PDR INTO R1
2084 026630 012702 172240      MOV      #SIPAR0, R2        ;LOAD FIRST ADDRESS OF PAR INTO R2
2085 026634 012721 177777      22$:  MOV      #-1, (R1)+     ;SET BITS IN PDR TO 1'S
2086 026640 012722 177777      MOV      #-1, (R2)+     ;SET BITS IN PAR'S TO 1'S
  
```



```
2087 026644 077305 SOB R3,22$ ;LOOP UNTIL ALL SUPERVISOR PAR/PDR'S LOADED
2088 026646 012703 000020 MOV #20,R3 ;LOAD LOOP COUNTER WITH A 16
2089 026652 012701 177600 MOV #UIPDRO,R1 ;LOAD FIRST ADDRESS OF PDR INTO R1
2090 026656 012702 177640 MOV #UIPARO,R2 ;LOAD FIRST ADDRESS OF PAR INTO R2
2091 026662 012721 177777 23$: MOV #-1,(R1)+ ;SET BITS IN PDR TO 1'S
2092 026666 012722 177777 MOV #-1,(R2)+ ;SET BITS IN PAR'S TO 1'S
2093 026672 077305 SOB R3,23$ ;LOOP UNTIL ALL USER PAR/PDR'S LOADED
2094 026674 000005 RESET ;ISSUE AN 'INIT' BY EXECUTING A RESET
2095 026676 012700 172300 MOV #KIPDRO,R0 ;LOAD ADDRESS OF FIRST KERNEL PDR IN R0
```

```

2097 026702 012704 000020      MOV      #20,R4      ;LOAD LOOP COUNTER WITH A 16
2098 026706 011001      MOV      (R0),R1    ;READ A KERNEL PDR INTO R1
2099 026710 022701 177416      CMP      #177416,R1 ;ARE ALL THE BITS STILL SET?
2100 026714 001401      BEQ      3$         ;BRANCH IF YES
2101 026716 104023      ERROR   +23        ;KERNEL PDR AFFECTED BY A RESET
2102                                ;FOR TIGHTER SCOPE LOOP
2103                                ;REPLACE ERROR CALL WITH
2104                                ;'BR 2$' = 000773
2105 026720 062700 000002      3$:      ADD      #2,R0      ;FORM ADDRESS OF NEXT KERNEL PDR
2106 026724 077410      SOB     R4,2$      ;LOOP TO 2$ UNTIL ALL KERNEL PDR'S CHECKED
2107 026726 012700 172340      MOV      #KIPAR0,R0 ;LOAD ADDRESS OF FIRST KERNEL PAR IN R0
2108 026732 012704 000020      MOV      #20,R4      ;LOAD LOOP COUNTER WITH A 16
2109 026736 011001      4$:      MOV      (R0),R1    ;READ A KERNEL PAR INTO R1
2110 026740 022701 177777      CMP      #177777,R1 ;ARE ALL THE BITS STILL SET?
2111 026744 001401      BEQ      5$         ;BRANCH IF YES
2112 026746 104023      ERROR   +23        ;KERNEL PAR AFFECTED BY A RESET
2113                                ;FOR TIGHTER SCOPE LOOP
2114                                ;REPLACE ERROR CALL WITH
2115                                ;'BR 4$' = 000773
2116 026750 062700 000002      5$:      ADD      #2,R0      ;FORM ADDRESS OF NEXT KERNEL PAR
2117 026754 077410      SOB     R4,4$      ;LOOP TO 4$ UNTIL ALL KERNEL PAR'S CHECKED
2118
2119 026756 012700 172200      MOV      #SIPDR0,R0 ;LOAD ADDRESS OF FIRST SUPERVISOR PDR IN R0
2120 026762 012704 000020      MOV      #20,R4      ;LOAD LOOP COUNTER WITH A 16
2121 026766 011001      6$:      MOV      (R0),R1    ;READ A SUPERVISOR PDR INTO R1
2122 026770 022701 177416      CMP      #177416,R1 ;ARE ALL THE BITS STILL SET?
2123 026774 001401      BEQ      7$         ;BRANCH IF YES
2124 026776 104023      ERROR   +23        ;SUPERVISOR PDR AFFECTED BY A RESET
2125                                ;FOR TIGHTER SCOPE LOOP
2126                                ;REPLACE ERROR CALL WITH
2127                                ;'BR 6$' = 000773
2128 027000 062700 000002      7$:      ADD      #2,R0      ;FORM ADDRESS OF NEXT SUPERVISOR PDR
2129 027004 077410      SOB     R4,6$      ;LOOP TO 6$ UNTIL ALL SUPERVISOR PDR'S CHECKED
2130
2131 027006 012700 172240      MOV      #SIPAR0,R0 ;LOAD ADDRESS OF FIRST SUPERVISOR PAR IN R0
2132 027012 012704 000020      MOV      #20,R4      ;LOAD LOOP COUNTER WITH A 16
2133 027016 011001      8$:      MOV      (R0),R1    ;READ A SUPERVISOR PAR INTO R1
2134 027020 022701 177777      CMP      #177777,R1 ;ARE ALL THE BITS STILL SET?
2135 027024 001401      BEQ      9$         ;BRANCH IF YES
2136 027026 104023      ERROR   +23        ;SUPERVISOR PAR AFFECTED BY A RESET
2137                                ;FOR TIGHTER SCOPE LOOP
2138                                ;REPLACE ERROR CALL WITH
2139                                ;'BR 8$' = 000773
2140 027030 062700 000002      9$:      ADD      #2,R0      ;FORM ADDRESS OF NEXT SUPERVISOR PAR
2141 027034 077410      SOB     R4,8$      ;LOOP TO 8$ UNTIL ALL SUPERVISOR PAR'S CHECKED
2142
2143 027036 012700 177600      MOV      #UIPDR0,R0 ;LOAD ADDRESS OF FIRST USER PDR WITH R0
2144 027042 012704 000020      MOV      #20,R4      ;LOAD LOOP COUNTER WITH A 16
2145 027046 011001      10$:     MOV      (R0),R1    ;READ A USER PDR INTO R1
2146 027050 022701 177416      CMP      #177416,R1 ;ARE ALL THE BITS STILL SET?
2147 027054 001401      BEQ      11$        ;BRANCH IF YES
2148 027056 104023      ERROR   +23        ;USER PDR AFFECTED BY A RESET
2149                                ;FOR TIGHTER SCOPE LOOP
2150                                ;REPLACE ERROR CALL WITH
2151                                ;'BR 10$' = 000773
2152 027060 062700 000002      11$:     ADD      #2,R0      ;FORM ADDRESS OF NEXT USER PDR
2153 027064 077410      SOB     R4,10$     ;LOOP TO 10$ UNTIL ALL USER PDR'S CHECKED
  
```

```
2154  
2155 027066 012700 177640          MOV    #UIPAR0,R0      ;LOAD ADDRESS OF FIRST USER PAR IN R0  
2156 027072 012704 000020          MOV    #20,R4         ;LOAD LOOP COUNTER WITH A 16  
2157 027076 011001          12$:  MOV    (R0),R1      ;READ A USER PAR INTO R1  
2158 027100 022701 177777          CMP    #177777,R1    ;ARE ALL THE BITS STILL SET?  
2159 027104 001401          BEQ    13$           ;BRANCH IF YES  
2160 027106 104023          ERROR  +23          ;USER PAR AFFECTED BY A RESET  
2161  
2162  
2163  
2164 027110 062700 000002          13$:  ADD    #2,R0         ;FORM ADDRESS OF NEXT USER PAR  
2165 027114 077410          SOB    R4,12$       ;LOOP TO 12$ UNTIL ALL USER PAR'S CHECKED  
2166  
2167
```

2181

```
*****
*TEST 44 INST. FETCH NOT RELOCATED IN MAINT. MODE
*
* THIS TEST CHECKS TO SEE THAT WHEN MEMORY MANAGEMENT IS IN
* MAINTENANCE MODE (DESTINATION-ONLY-RELOCATION), AN INSTRUCTION
* FETCH IS NOT RELOCATED AND A RESET CLEARS THE MAINTENANCE BIT
* (BIT 08) IN SRO. IF THE 'FETCH' IS RELOCATED, A PG.LENGTH ABORT
* SHOULD OCCUR, CAUSING A HALT SINCE TRAP CATCHER IS PLACED IN VECTOR 250
*
* NOTE: A HALT MAY OCCUR IF MAINT. MODE NOT DISABLED BY RESET
*****
```

```
2182 027116 000004
2183 027120 004767 153046
2184 027124 012700 172300
2185 027130 012704 000010
2186 027134 005020
2187 027136 077402
2188 027140 012767 000252 151102
2189 027146 005067 151100
2190
2191 027152 012767 001006 143120
2192 027160 012767 027166 151722
2193 027166 012767 000400 150376
2194 027174 000005
2195 027176 032767 000400 150366
2196 027204 001403
2197 027206 005067 150360
2198 027212 104024
2199
2200
2201
2202 027214 012706 001100
2203 027220 005067 150346
2204 027224 012767 003122 151016
2205 027232 012767 000340 151012
2206 027240 012767 027120 151642
2207 027246 004767 152754
2208
2222
```

```
TST44: SCOPE
1$: JSR PC,TOFF ;TURN T-BIT TRAPPING OFF FOR THIS TEST
MOV #KIPDR0,R0 ;LOAD ADDRESS OF FIRST KERNEL PDR INTO R0
MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
2$: CLR (R0)+ ;CLEAR PDR - MAPPING PAGE NON-RES, 0 BLKS.
SOB R4,2$ ;LOOP TO 2$ UNTIL ALL KERNEL PDR'S CLEARED
MOV #MMVEC+2,MMVEC ;LOAD TRAP CATCHER INTO MEM MGMT. VECTOR
CLR MMVEC+2
;
; A HALT WILL OCCUR IF RESET FAILS
; TO DISABLE DEST.-ONLY RELOCATION
; MAP KERNEL PG 0 R/W, 3 BLOCKS LONG.
MOV #1006,KIPDR0 ;MAP KERNEL PG 0 R/W, 3 BLOCKS LONG.
MOV #3$,SLPERR ;SET LOOP ON ERROR POINTER TO 3$
3$: MOV #BIT8,SRO ;TURN ON DEST-ONLY-RELOCATION
RESET ;SHOULD CLEAR MAINT. BIT - WILL ABORT IF RELOCATED
BIT #BIT8,SRO ;WAS MAINT. BIT (BIT 8) OF SRO CLEARED?
BEQ 4$ ;BRANCH IF YES
CLR SRO ;CLEAR SRO SO ERROR CAN BE REPORTED
ERROR +24 ;MAINT. MODE NOT DISABLED BY A RESET
; FOR A TIGHTER SCOPE LOOP
; REPLACE ERROR CALL WITH
; 'BR 3$' = 000765
4$: MOV #KERSTK,KSP ;RESTORE STACK POINTER
CLR SRO ;BE SURE SRO IS CLEAR
MOV #MMGMERR,MMVEC ;RESTORE MEM. MGMT. TRAP VECTOR
MOV #340,MMVEC+2 ;RESTORE MEM. MGMT VECTOR+2
MOV #1$,SLPERR ;RESET LOOP ON ERROR POINTER TO 1$
JSR PC,TON ;TURN T-BIT TRAPPING BACK ON
```

```
2223 027252 000004
2224 027254 004767 152712
```

```
*****
*TEST 45 TEST THAT SOURCE NOT RELOCATED IN MAINT. MODE
*
* THIS TEST CHECKS TO SEE THAT WHEN MEMORY MANAGEMENT IS IN
* MAINTENANCE MODE, THE SOURCE IS NOT RELOCATED; ONLY THE
* DESTINATION IS RELOCATED. KERNEL PAR'S 3 & 4 ARE MAPPED TO
* PHYSICAL ADDRESS 60000-77776. PDR4 IS SET TO ALLOW FULL
* READ/WRITE BUT PDR3 IS CLEAR ALLOWING NO ACCESS. VIRTUAL
* ADDRESSES REFERENCING PAR/PDR'S 4 & 3 ARE USED AS
* DESTINATION AND SOURCE RESPECTIVELY. IF THE SOURCE IS
* RELOCATED IN MAINTENANCE MODE A MEM. MGMT. TRAP WILL OCCUR
* AND THE ERROR WILL BE REPORTED. KERNEL PG. 7 IS MAPPED R/W.
*****
TST45: SCOPE
1$: JSR PC,TOFF ;TURN OFF T-BIT TRAPPING FOR THIS TEST
```

```

2224 027260 012767 027350 151622      MOV      #2$, $LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2225 027266 012767 000600 143052      MOV      #600, KIPAR3    ;MAP KERNAL PAGE 3 TO 12-16K
2226 027274 012767 000600 143046      MOV      #600, KIPAR4    ;MAP KERNAL PAGE 4 TO 12-16K
2227 027302 012767 007600 143046      MOV      #7600, KIPAR7   ;MAP KERNAL PAGE 7 TO THE I/O PAGE
2228 027310 005067 142772                CLR      KIPDR3          ;MAP KERNAL PAGE 3 'NO ACCESS'
2229 027314 012767 077406 142766      MOV      #77406, KIPDR4  ;MAP KERNAL PAGE 4 R/W, 128 BLKS
2230 027322 012767 077406 142766      MOV      #77406, KIPDR7  ;MAP KERNAL PAGE 7 R/W, 128 BLKS
2231 027330 012767 027430 150712      MOV      #4$, MMVEC      ;SET M.M. VECTOR TO 4$ IN CASE OF ABORT
2232 027336 012737 000377 060000      MOV      #377, @#60000   ;LOAD ALL 1'S IN LOW BYTE OF TEST LOC.
2233 027344 012700 177777                MOV      #-1, R0         ;LOAD EXPECTED CONTENTS OF TEST LOC. IN R0
2234 027350 052767 000400 150214 2$:   BIS      #BIT8, SRO      ;TURN ON DEST.-ONLY-RELOCATION
2235 027356 113737 060000 100001      MOV      @#60000, @#100001 ;LOAD HI BYTE OF TEST LOC.-ABORT IF SOURCE RELOCATED
2236 027364 000005                RESET                    ;TURN OFF DEST.-ONLY-RELOCATION
2237 027366 013701 060000                MOV      @#60000, R1     ;READ CONTENTS OF TEST LOC.
2238 027372 020001                CMP      R0, R1          ;WAS TEST LOCATION LOADED PROPERLY?
2239 027374 001401                BEQ     3$              ;BRANCH IF YES
2240 027376 104025                ERROR   +25             ;TEST LOC. NOT LOADED - INST. WAS ABORTED
2241                                ;WHEN SOURCE WAS RELOCATED
2242                                ;FOR TIGHTER SCOPE LOOP REPLACE
2243                                ;ERROR CALL WITH
2244                                ;'BR 2$' = 000764
2245 027400 012767 003122 150642 3$:   MOV      #MGMERR, MMVEC  ;RESTORE MEM. MGMT. VECTOR
2246 027406 012767 077406 142672      MOV      #77406, KIPDR3  ;MAP KERNAL PAGE 3 R/W, 128 BLKS
2247 027414 012767 027254 151466      MOV      #1$, $LPERR     ;RESET LOOP ON ERROR POINTER TO 1$
2248 027422 004767 152600                JSR     PC, TON         ;TURN T-BIT TRAPPING BACK ON
2249 027426 000430                BR      TST46          ;SKIP TO NEXT TEST
2250                                ;* IF THE PROGRAM TRIES TO RELOCATE THE SOURCE, IT SHOULD TRAP TO 4$
2251                                ;
2252 027430 042767 000400 150134 4$:   BIC      #BIT8, SRO      ;TURN OFF DEST.-ONLY-RELOCATION THRU PAR/PDR 7
2253 027436 016767 150130 151620      MOV      SRO, WASSRO     ;SAVE REST OF SRO CONTENTS
2254 027444 016767 150126 151616      MOV      SR2, WASSR2     ;SAVE CONTENTS OF SR2
2255 027452 010667 151600                MOV      SP, WASR6       ;SAVE VALUE OF THE STACK POINTER
2256 027456 012667 151576                MOV      (SP)+, TRAPPC   ;SAVE PC OF TRAP
2257 027462 012667 151574                MOV      (SP)+, TRAPPS   ;SAVE PSW OF TRAP
2258 027466 042767 160000 150076      BIC      #160000, SRO    ;CLEAR ERROR BITS IN SRO
2259 027474 104026                ERROR   +26             ;SOURCE APPARENTLY RELOCATED IN MAINT. MODE
2260                                ;FOR TIGHTER SCOPE LOOP
2261                                ;REPLACE ERROR CALL WITH A
2262                                ;'NOP' = 000240
2263 027476 016746 151560                MOV      TRAPPS, -(SP)   ;PUT PSW OF TRAP BACK ON THE STACK
2264 027502 016746 151552                MOV      TRAPPC, -(SP)  ;PUT PC OF TRAP BACK ON THE STACK
2265 027506 000002                RTI                    ;RETURN TO TEST
  
```



```

2311 027734 012767 027766 151146      MOV      #3$, $LPERR      ;SET LOOP ON ERROR POINTER TO 3$
      027742 013767 067776 151226      MOV      @#67776,$TMP0   ;SAVE DATA AT TEST LOCATION
      027750 012737 000625 172350      MOV      #625,@#KIPAR4   ;LOAD PAR4 WITH 625
      027756 012700 105276           MOV      #105276,R0      ;PUT VIRTUAL ADDRESS IN R0
      027762 012701 125201           MOV      #125201,R1      ;PUT DATA PATTERN INTO R1
      027766 052737 000400 177572 3$:  BIS      #BIT8,@MMRO     ;TURN ON DESTINATION ONLY RELOCATION
      027774 010110           MOV      R1,(R0)        ;TRY TO LOAD DATA PATTERN INTO 105276
      027776 013702 067776           MOV      @#67776,R2     ;READ (67776) INTO R2
      030002 000005           RESET                     ;CLEAR MMRO
      030004 020102           CMP      R1,R2          ;SEE IF DATA MATCHES PATTERN
      030006 001401           BEQ      4$             ;BRANCH IF DATA MATCHES
      030010 104017           ERROR    +17            ;RELOCATION FAILED
      030012 016737 151160 067776 4$:  MOV      $TMP0,@#67776   ;RESTORE ORIGINAL DATA TO TEST LOCATION
2312 030020 012767 030052 151062      MOV      #5$, $LPERR     ;SET LOOP ON ERROR POINTER TO 5$
      030026 013767 071000 151142      MOV      @#71000,$TMP0   ;SAVE DATA AT TEST LOCATION
      030034 012737 000633 172350      MOV      #633,@#KIPAR4   ;LOAD PAR4 WITH 633
      030042 012700 105500           MOV      #105500,R0      ;PUT VIRTUAL ADDRESS IN R0
      030046 012701 125202           MOV      #125202,R1      ;PUT DATA PATTERN INTO R1
      030052 052737 000400 177572 5$:  BIS      #BIT8,@MMRO     ;TURN ON DESTINATION ONLY RELOCATION
      030060 010110           MOV      R1,(R0)        ;TRY TO LOAD DATA PATTERN INTO 105500
      030062 013702 071000           MOV      @#71000,R2     ;READ (71000) INTO R2
      030066 000005           RESET                     ;CLEAR MMRO
      030070 020102           CMP      R1,R2          ;SEE IF DATA MATCHES PATTERN
      030072 001401           BEQ      6$             ;BRANCH IF DATA MATCHES
      030074 104017           ERROR    +17            ;RELOCATION FAILED
      030076 016737 151074 071000 6$:  MOV      $TMP0,@#71000   ;RESTORE ORIGINAL DATA TO TEST LOCATION
2313 030104 012767 030136 150776      MOV      #7$, $LPERR     ;SET LOOP ON ERROR POINTER TO 7$
      030112 013767 072076 151056      MOV      @#72076,$TMP0   ;SAVE DATA AT TEST LOCATION
      030120 012737 000604 172350      MOV      #604,@#KIPAR4   ;LOAD PAR4 WITH 604
      030126 012700 111476           MOV      #111476,R0      ;PUT VIRTUAL ADDRESS IN R0
      030132 012701 125203           MOV      #125203,R1      ;PUT DATA PATTERN INTO R1
      030136 052737 000400 177572 7$:  BIS      #BIT8,@MMRO     ;TURN ON DESTINATION ONLY RELOCATION
      030144 010110           MOV      R1,(R0)        ;TRY TO LOAD DATA PATTERN INTO 111476
      030146 013702 072076           MOV      @#72076,R2     ;READ (72076) INTO R2
      030152 000005           RESET                     ;CLEAR MMRO
      030154 020102           CMP      R1,R2          ;SEE IF DATA MATCHES PATTERN
      030156 001401           BEQ      8$             ;BRANCH IF DATA MATCHES
      030160 104017           ERROR    +17            ;RELOCATION FAILED
      030162 016737 151010 072076 8$:  MOV      $TMP0,@#72076   ;RESTORE ORIGINAL DATA TO TEST LOCATION
2314 030170 026727 152626 001000      CMP      $LSTBK,#1000    ;DO WE HAVE OVER 16K ON THIS SYSTEM?
2315 030176 003002           BGT      9$             ;BRANCH IF WE DO
2316 030200 000167 001166           JMP      APTTST          ;JUMP TO THE NEXT TEST IF WE DO NOT
2317 030204           9$:
      030204 012767 030236 150676      MOV      #11$, $LPERR    ;SET LOOP ON ERROR POINTER TO 11$
      030212 013767 100000 150756      MOV      @#100000,$TMP0  ;SAVE DATA AT TEST LOCATION
      030220 012737 001000 172350      MOV      #1000,@#KIPAR4  ;LOAD PAR4 WITH 1000
      030226 012700 100000           MOV      #100000,R0      ;PUT VIRTUAL ADDRESS IN R0
      030232 012701 125204           MOV      #125204,R1      ;PUT DATA PATTERN INTO R1
      030236 052737 000400 177572 11$: BIS      #BIT8,@MMRO     ;TURN ON DESTINATION ONLY RELOCATION
      030244 010110           MOV      R1,(R0)        ;TRY TO LOAD DATA PATTERN INTO 100000
      030246 013702 100000           MOV      @#100000,R2     ;READ (100000) INTO R2
      030252 000005           RESET                     ;CLEAR MMRO
      030254 020102           CMP      R1,R2          ;SEE IF DATA MATCHES PATTERN
      030256 001401           BEQ      12$           ;BRANCH IF DATA MATCHES
      030260 104017           ERROR    +17            ;RELOCATION FAILED
      030262 016737 150710 100000 12$: MOV      $TMP0,@#100000  ;RESTORE ORIGINAL DATA TO TEST LOCATION
2318 030270 012767 030322 150612      MOV      #13$, $LPERR    ;SET LOOP ON ERROR POINTER TO 13$
  
```

	030276	013767	137776	150672	MOV	@#137776,\$TMP0	:SAVE DATA AT TEST LOCATION
	030304	012737	001252	172350	MOV	#1252,@#KIPAR4	:LOAD PAR4 WITH 1252
	030312	012700	112576		MOV	#112576,R0	:PUT VIRTUAL ADDRESS IN R0
	030316	012701	125205		MOV	#125205,R1	:PUT DATA PATTERN INTO R1
	030322	052737	000400	177572	13\$: BIS	#BIT8,@#MMRO	:TURN ON DESTINATION ONLY RELOCATION
	030330	010110			MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 112576
	030332	013702	137776		MOV	@#137776,R2	:READ (137776) INTO R2
	030336	000005			RESET		:CLEAR MMRO
	030340	020102			CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
	030342	001401			BEQ	14\$	:BRANCH IF DATA MATCHES
	030344	104017			ERROR	+17	:RELOCATION FAILED
2319	030346	016737	150624	137776	14\$: MOV	\$TMP0,@#137776	:RESTORE ORIGINAL DATA TO TEST LOCATION
	030354	012767	030406	150526	MOV	#15\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 15\$
	030362	013767	117776	150606	MOV	@#117776,\$TMP0	:SAVE DATA AT TEST LOCATION
	030370	012737	001125	172350	MOV	#1125,@#KIPAR4	:LOAD PAR4 WITH 1125
	030376	012700	105276		MOV	#105276,R0	:PUT VIRTUAL ADDRESS IN R0
	030402	012701	125206		MOV	#125206,R1	:PUT DATA PATTERN INTO R1
	030406	052737	000400	177572	15\$: BIS	#BIT8,@#MMRO	:TURN ON DESTINATION ONLY RELOCATION
	030414	010110			MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 105276
	030416	013702	117776		MOV	@#117776,R2	:READ (117776) INTO R2
	030422	000005			RESET		:CLEAR MMRO
	030424	020102			CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
	030426	001401			BEQ	16\$	:BRANCH IF DATA MATCHES
	030430	104017			ERROR	+17	:RELOCATION FAILED
2320	030432	016737	150540	117776	16\$: MOV	\$TMP0,@#117776	:RESTORE ORIGINAL DATA TO TEST LOCATION
	030440	012767	030472	150442	MOV	#17\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 17\$
	030446	013767	102000	150522	MOV	@#102000,\$TMP0	:SAVE DATA AT TEST LOCATION
	030454	012737	001010	172350	MOV	#1010,@#KIPAR4	:LOAD PAR4 WITH 1010
	030462	012700	101000		MOV	#101000,R0	:PUT VIRTUAL ADDRESS IN R0
	030466	012701	125207		MOV	#125207,R1	:PUT DATA PATTERN INTO R1
	030472	052737	000400	177572	17\$: BIS	#BIT8,@#MMRO	:TURN ON DESTINATION ONLY RELOCATION
	030500	010110			MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 101000
	030502	013702	102000		MOV	@#102000,R2	:READ (102000) INTO R2
	030506	000005			RESET		:CLEAR MMRO
	030510	020102			CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
	030512	001401			BEQ	18\$	:BRANCH IF DATA MATCHES
	030514	104017			ERROR	+17	:RELOCATION FAILED
2321	030516	016737	150454	102000	18\$: MOV	\$TMP0,@#102000	:RESTORE ORIGINAL DATA TO TEST LOCATION
	030524	012767	030556	150356	MOV	#19\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 19\$
	030532	013767	142000	150436	MOV	@#142000,\$TMP0	:SAVE DATA AT TEST LOCATION
	030540	012737	001314	172350	MOV	#1314,@#KIPAR4	:LOAD PAR4 WITH 1314
	030546	012700	110400		MOV	#110400,R0	:PUT VIRTUAL ADDRESS IN R0
	030552	012701	125210		MOV	#125210,R1	:PUT DATA PATTERN INTO R1
	030556	052737	000400	177572	19\$: BIS	#BIT8,@#MMRO	:TURN ON DESTINATION ONLY RELOCATION
	030564	010110			MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 110400
	030566	013702	142000		MOV	@#142000,R2	:READ (142000) INTO R2
	030572	000005			RESET		:CLEAR MMRO
	030574	020102			CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
	030576	001401			BEQ	20\$	:BRANCH IF DATA MATCHES
	030600	104017			ERROR	+17	:RELOCATION FAILED
2322	030602	016737	150370	142000	20\$: MOV	\$TMP0,@#142000	:RESTORE ORIGINAL DATA TO TEST LOCATION
	030610	012767	030642	150272	MOV	#21\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 21\$
	030616	013767	122000	150352	MOV	@#122000,\$TMP0	:SAVE DATA AT TEST LOCATION
	030624	012737	001104	172350	MOV	#1104,@#KIPAR4	:LOAD PAR4 WITH 1104
	030632	012700	111400		MOV	#111400,R0	:PUT VIRTUAL ADDRESS IN R0
	030636	012701	125211		MOV	#125211,R1	:PUT DATA PATTERN INTO R1
	030642	052737	000400	177572	21\$: BIS	#BIT8,@#MMRO	:TURN ON DESTINATION ONLY RELOCATION



	030650	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 111400
	030652	013702	122000			MOV	@#122000,R2	:READ (122000) INTO R2
	030656	000005				RFSET		:CLEAR MMRO
	030660	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
	030662	001401				BEQ	22\$	:BRANCH IF DATA MATCHES
	030664	104017				ERROR	+17	:RELOCATION FAILED
	030666	016737	150304	122000	22\$:	MOV	\$TMP0,@#122000	:RESTORE ORIGINAL DATA TO TEST LOCATION
2323	030674	012767	030726	150206		MOV	#23\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 23\$
	030702	013767	142000	150266		MOV	@#142000,\$TMP0	:SAVE DATA AT TEST LOCATION
	030710	012737	001356	172350		MOV	#1356,@#KIPAR4	:LOAD PAR4 WITH 1356
	030716	012700	104200			MOV	#104200,R0	:PUT VIRTUAL ADDRESS IN R0
	030722	012701	125212			MOV	#125212,R1	:PUT DATA PATTERN INTO R1
	030726	052737	000400	177572	23\$:	BIS	#BIT8,@#MMRO	:TURN ON DESTINATION ONLY RELOCATION
	030734	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 104200
	030736	013702	142000			MOV	@#142000,R2	:READ (142000) INTO R2
	030742	000005				RESET		:CLEAR MMRO
	030744	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
	030746	001401				BEQ	24\$	:BRANCH IF DATA MATCHES
	030750	104017				ERROR	+17	:RELOCATION FAILED
	030752	016737	150220	142000	24\$:	MOV	\$TMP0,@#142000	:RESTORE ORIGINAL DATA TO TEST LOCATION
2324	030760	012767	031012	150122		MOV	#25\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 25\$
	030766	013767	142000	150202		MOV	@#142000,\$TMP0	:SAVE DATA AT TEST LOCATION
	030774	012737	001242	172350		MOV	#1242,@#KIPAR4	:LOAD PAR4 WITH 1242
	031002	012700	115600			MOV	#115600,R0	:PUT VIRTUAL ADDRESS IN R0
	031006	012701	125213			MOV	#125213,R1	:PUT DATA PATTERN INTO R1
	031012	052737	000400	177572	25\$:	BIS	#BIT8,@#MMRO	:TURN ON DESTINATION ONLY RELOCATION
	031020	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 115600
	031022	013702	142000			MOV	@#142000,R2	:READ (142000) INTO R2
	031026	000005				RESET		:CLEAR MMRO
	031030	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
	031032	001401				BEQ	26\$	:BRANCH IF DATA MATCHES
	031034	104017				ERROR	+17	:RELOCATION FAILED
	031036	016737	150134	142000	26\$:	MOV	\$TMP0,@#142000	:RESTORE ORIGINAL DATA TO TEST LOCATION
2325	031044	012767	031076	150036		MOV	#27\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 27\$
	031052	013767	142000	150116		MOV	@#142000,\$TMP0	:SAVE DATA AT TEST LOCATION
	031060	012737	001377	172350		MOV	#1377,@#KIPAR4	:LOAD PAR4 WITH 1377
	031066	012700	102100			MOV	#102100,R0	:PUT VIRTUAL ADDRESS IN R0
	031072	012701	125214			MOV	#125214,R1	:PUT DATA PATTERN INTO R1
	031076	052737	000400	177572	27\$:	BIS	#BIT8,@#MMRO	:TURN ON DESTINATION ONLY RELOCATION
	031104	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 102100
	031106	013702	142000			MOV	@#142000,R2	:READ (142000) INTO R2
	031112	000005				RESET		:CLEAR MMRO
	031114	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
	031116	001401				BEQ	28\$	:BRANCH IF DATA MATCHES
	031120	104017				ERROR	+17	:RELOCATION FAILED
	031122	016737	150050	142000	28\$:	MOV	\$TMP0,@#142000	:RESTORE ORIGINAL DATA TO TEST LOCATION
2326	031130	012767	031162	147752		MOV	#29\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 29\$
	031136	013767	142000	150032		MOV	@#142000,\$TMP0	:SAVE DATA AT TEST LOCATION
	031144	012737	001221	172350		MOV	#1221,@#KIPAR4	:LOAD PAR4 WITH 1221
	031152	012700	117700			MOV	#117700,R0	:PUT VIRTUAL ADDRESS IN R0
	031156	012701	125215			MOV	#125215,R1	:PUT DATA PATTERN INTO R1
	031162	052737	000400	177572	29\$:	BIS	#BIT8,@#MMRO	:TURN ON DESTINATION ONLY RELOCATION
	031170	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 117700
	031172	013702	142000			MOV	@#142000,R2	:READ (142000) INTO R2
	031176	000005				RESET		:CLEAR MMRO
	031200	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
	031202	001401				BEQ	30\$	:BRANCH IF DATA MATCHES

```

031204 104017 ERROR +17 ;RELOCATION FAILED
2327 031206 016737 147764 142000 30$: MOV $TMP0,@#142000 ;RESTORE ORIGINAL DATA TO TEST LOCATION
031214 012767 031246 147666 MOV #31$, $LPERR ;SET LOOP ON ERROR POINTER TO 31$
031222 013767 140000 147746 MOV @#140000,$TMP0 ;SAVE DATA AT TEST LOCATION
031230 012737 001377 172350 MOV #1377,@#KIPAR4 ;LOAD PAR4 WITH 1377
031236 012700 100100 MOV #100100,R0 ;PUT VIRTUAL ADDRESS IN R0
031242 012701 125216 MOV #125216,R1 ;PUT DATA PATTERN INTO R1
031246 052737 000400 177572 31$: BIS #BIT8,@#MMRO ;TURN ON DESTINATION ONLY RELOCATION
031254 010110 MOV R1,(R0) ;TRY TO LOAD DATA PATTERN INTO '00100
031256 013702 140000 MOV @#140000,R2 ;READ (140000) INTO R2
031262 000005 RESET ;CLEAR MMRO
031264 020102 CMP R1,R2 ;SEE IF DATA MATCHES PATTERN
031266 001401 BEQ 32$ ;BRANCH IF DATA MATCHES
031270 104017 ERROR +17 ;RELOCATION FAILED
2328 031272 016737 147700 140000 32$: MOV $TMP0,@#140000 ;RESTORE ORIGINAL DATA TO TEST LOCATION
031300 012767 031332 147602 MOV #33$, $LPERR ;SET LOOP ON ERROR POINTER TO 33$
031306 013767 140000 147662 MOV @#140000,$TMP0 ;SAVE DATA AT TEST LOCATION
031314 012737 001370 172350 MOV #1370,@#KIPAR4 ;LOAD PAR4 WITH 1370
031322 012700 101000 MOV #101000,R0 ;PUT VIRTUAL ADDRESS IN R0
031326 012701 125217 MOV #125217,R1 ;PUT DATA PATTERN INTO R1
031332 052737 000400 177572 33$: BIS #BIT8,@#MMRO ;TURN ON DESTINATION ONLY RELOCATION
031340 010110 MOV R1,(R0) ;TRY TO LOAD DATA PATTERN INTO 101000
031342 013702 140000 MOV @#140000,R2 ;READ (140000) INTO R2
031346 000005 RESET ;CLEAR MMRO
031350 020102 CMP R1,R2 ;SEE IF DATA MATCHES PATTERN
031352 001401 BEQ 34$ ;BRANCH IF DATA MATCHES
031354 104017 ERROR +17 ;RELOCATION FAILED
2329 031356 016737 147614 140000 34$: MOV $TMP0,@#140000 ;RESTORE ORIGINAL DATA TO TEST LOCATION
031364 012767 027572 147516 MOV #40$, $LPERR ;SET LOOP ON ERROR POINTER TO 40$
  
```

```

2330
2331
2332 ;*****
2333 ;* THIS PIECE OF CODE CHECKS TO SEE IF THE PROGRAM IS RUNNING UNDER
2334 ;* APT, AND THEN LOOKS FOR THE APT SPECIAL ADDRESSING HARDWARE. IF
2335 ;* THE HARDWARE IS FOUND, THE NEXT TWO TESTS OPERATE AS IF THERE
2336 ;* WERE A FULL COMPLEMENT OF MEMORY ON THE SYSTEM.
  
```

```

2337 031372 005067 147742 APTTST: CLR HDWFLAG ;RESET THE HARDWARE PRESENT FLAG
2338 031376 105767 147642 TSTB $ENV ;ARE WE UNDER APT?
2339 031402 001002 BNE 1$ ;BRANCH IF WE ARE
2340 031404 000167 000042 JMP TST47 ;ELSE GO TO TEST 47
2341 031410 032767 000200 147632 1$: BIT #BIT7,$USWR ;IS THE HARDWARE SUPPOSED TO BE THERE?
2342 031416 001002 BNE 2$ ;BRANCH IF TRUE
2343 031420 000167 000026 JMP TST47 ;ELSE GO TO TEST 47
2344 031424 012767 031440 146352 2$: MOV #3$,ERRVEC ;TRAPS TO 4 GO TO 3$
2345 031432 005767 146332 TST RMIREG ;SPECIAL HARDWARE PRESENT?
2346 031436 000403 BR 4$ ;IF WE GOT HERE, THE HARDWARE IS THERE
2347 031440 104060 3$: ERROR +60 ;APT SPECIAL HARDWARE NOT FOUND
2348 031442 000167 000004 JMP TST47 ;GO TO TEST 47
2349 031446 005267 147666 4$: INC HDWFLAG ;SET SPECIAL HARDWARE PRESENT FLAG
  
```

```

2350
2365 ;*****
;*TEST 47 18-BIT MAPPING CARRY PROPAGATION
;*
;* THIS TEST USES FULL 18-BIT RELOCATION TO CHECK THE CARRY
;* PROPAGATION OF THE RELOCATION ADDER. SINCE THIS TEST SCANS
;* MEMORY FROM 00100000 TO 00740000 ON 8K BOUNDARIES, IT WILL
;* REPORT ANY HOLES THAT IT FINDS IN MEMORY UP TO THE LAST
  
```

.\* BLOCK. THE INFORMATION GIVEN WILL BE THE ADDRESS WHERE  
 .\* THE HOLE WAS DISCOVERED AND THE FIRST GOOD ADDRESS AFTER  
 .\* THE HOLE.  
 .\*

.\* NOTE - PART OF THIS TEST WILL NOT BE RUN IF THE SYSTEM  
 .\* MEMORY SIZE IS LESS THAN 16K WORDS.  
 .\*

\*\*\*\*\*  
 TST47: SCOPE

```

031452 000004
2366
2367 031454 012700 100100      MOV      #100100,R0      ;LOAD VIRTUAL ADDR FOR PAR4 INTO R0
2368 031460 012737 000001 177572  MOV      #1,2AMMRO      ;TURN ON 18-BIT MAPPING
2369 031466 005767 147646      TST      HDWFLAG        ;SPECIAL HARDWARE?
2370 031472 001004      BNE      20$            ;BRANCH IF TRUE
2371 031474 026727 151322 001000  CMP      $LSTBK,#1000   ;IS THERE AT LEAST 16K ON THE SYSTEM?
2372 031502 002510      BLT      4$            ;BRANCH IF LESS THAN 16K TO 4$
2373 031504 005067 147626      20$: CLR      HOLFLG        ;MAKE SURE HOLE FLAG STARTS AT ZERO
2374 031510 012767 032146 146266  MOV      #10$,ERRVEC    ;SET ERROR VECTOR POINTER TO 10$
2375 031516 012767 000777 140624  MOV      #777,KIPAR4    ;LOAD PAR4 WITH STARTING BASE
2376 031524 012767 001000 140620  MOV      #1000,KIPAR5   ;START ADDRESSING WITH 16K
2377 031532 012701 120000      MOV      #120000,R1     ;LOAD VIRTUAL ADDR FOR PAR5 INTO R1
2378 031536 012702 000375      MOV      #375,R2        ;LOAD DATA PATTERN INTO R2
2379 031542 012767 031550 147340  MOV      #1$, $LPERR    ;SET LOOP ON ERROR POINTER TO 1$
2380 031550 005067 147556      1$: CLR      PCPUER      ;CLEAR CPU ERROR FLAG
2381 031554 005767 147560      TST      HDWFLAG        ;SPECIAL HARDWARE?
2382 031560 001404      BEQ      2$            ;BRANCH IF FALSE
2383 031562 026767 140564 151232  CMP      KIPAR5,$LSTBK  ;ARE WE OUT OF EXISTING MEMORY?
2384 031570 003023      BGT      21$           ;BRANCH IF TRUE
2385 031572 011167 147400      2$: MOV      (R1),$TMP0    ;SAVE DATA AT TEST LOCATION
2386 031576 005767 147530      TST      PCPUER        ;SEE IF THERE WAS A CPU TRAP
2387 031602 001014      BNE      3$            ;BRANCH IF TRAP OCCURRED
2388 031604 005767 147526      TST      HOLFLG        ;SEE IF HOLE WAS FOUND IN MEMORY
2389 031610 001411      BEQ      3$            ;BRANCH IF NO HOLE WAS FOUND
2390 031612 016767 140534 147362  MOV      KIPAR5,$TMP2   ;SAVE PAR THAT POINTS TO END OF HOLE
2391 031620 012767 031504 147262  MOV      #20$, $LPERR   ;SET LOOP ON ERROR POINTER TO 20$
2392 031626 104041      ERROR    +41           ;HOLE IN MEMORY FROM $TMP1 TO $TMP2
2393 031630 005067 147502      CLR      HOLFLG        ;CLEAR HOLE FLAG IN CASE THERE ARE MORE
2394 031634 010210      3$: MOV      R2,(R0)     ;LOAD TEST PATTERN INTO TEST LOCATION
2395 031636 000405      BR       23$           ;BRANCH OVER NEXT INSTRUCTIONS
2396 031640 010204      21$: MOV      R2,R4       ;GET COPY OF DATA
2397 031642 052704 000400      BIS      #BIT8,R4      ;ADD SPECIAL HARDWARE ENABLE BIT
2398 031646 010467 146116      MOV      R4,RMIREG     ;PUT DATA IN SPECIAL HARDWARE
2399 031652 011103      23$: MOV      (R1),R3    ;READ TEST LOCATION VIA DIFFERENT VIRT. ADDR.
2400 031654 020203      CMP      R2,R3         ;SEE IF THE CORRECT LOCATION WAS REFERENCED
2401 031656 001401      BEQ      22$           ;BRANCH IF CORRECT DATA WAS OBTAINED
2402 031660 104042      ERROR    +42           ;BAD RELOCATION
2403 031662 016711 147310      22$: MOV      $TMP0,(R1) ;RESTORE ORIGINAL DATA TO TEST LOCATION
2404 031666 062767 000400 140454  ADD      #400,KIPAR4   ;CHANGE BASE ADDRESS
2405 031674 062767 000400 140450  ADD      #400,KIPAR5   ;CHANGE BASE ADDRESS
2406 031702 005302      DEC      R2            ;CHANGE DATA PATTERN
2407 031704 022767 007400 140440  CMP      #7400,KIPAR5  ;SEE IF PAST LAST ADDRESS
2408 031712 103316      BHIS    1$            ;BRANCH IF NOT PAST LAST ADDRESS
2409 031714 005767 147416      TST      HOLFLG        ;SEE IF YOU ARE IN MIDDLE OF HOLE
2410 031720 001401      BEQ      4$            ;BRANCH IF NOT IN MIDDLE OF HOLE
2411 031722 104043      ERROR    +43           ;IN MIDDLE OF HOLE IN MEMORY
2412      HOLFLG HAS NUMBER OF TIMEOUTS
2413      $TMP1 HAS PAR OF FIRST TIMEOUT
  
```

```

2414 :*
2415 :* ADDRESS 760000 IS GENERATED BY CARRY PROPAGATION WHILE THE
2416 :* PROGRAM IS EXPECTING A TIME OUT. (760000 IS THE FIRST I/O
2417 :* PAGE ADDRESS). KIPAR4 WILL CONTAIN 007577 AND R0 HAS 100100
2418 :* (WHICH SELECTS KIPAR4 AND GENERATES ADDRESS 760000). IF THE
2419 :* WRONG BIT IN THE CPU ERROR REGISTER IS SET, OR IF NO TRAP
2420 :* TO 'ERRVEC' OCCURS AN ERROR IS REPORTED.
2421 :*
2422 031724 005767 147410 4$: TST HDWFLAG ;SPECIAL HARDWARE?
2423 031730 001402 BEQ 45$ ;BRANCH IF NOT
2424 031732 005067 146032 CLR RMIREG ;MAKE SURE SPECIAL HARDWARE IS OFF
2425 031736 012767 003024 146040 45$: MOV #TIMERR,ERRVEC ;RESTORE NORMAL ROUTINE FOR TRAPS THRU 4
2426 031744 012767 031776 147136 MOV #40$,SLPERR ;SET LOOP ON ERROR POINTER TO 40$
2427 031752 012767 007577 140370 MOV #7577,KIPAR4 ;LOAD KIPAR4 WITH 007577
2428 031760 012702 007600 MOV #7600,R2 ;LOAD DATA PATTERN INTO R2
2429 031764 012767 000020 147342 MOV #20,CPUEXP ;EXPECTING UNIBUS TIMEOUT
2430 031772 005067 147334 CLR PCPUER ;CLEAR CPU TRAP FLAG
2431 031776 010210 40$: MOV R2,(R0) ;LOAD 760000 THRU PAGE 4
2432 :* ;THIS INSTRUCTION SHOULD TIMEOUT
2433 :* ;OVER THE UNIBUS.
2434 032000 005767 147326 TST PCPUER ;SEE IF TRAP OCCURRED
2435 032004 001001 BNE 6$ ;BRANCH IF TRAP
2436 032006 104044 ERROR +44 ;NO CPU TRAP
2437 :*
2438 :* ADDRESS 000000 IS GENERATED BY CARRY PROPAGATION. THEN, IF
2439 :* THERE IS LESS THAN 120K OF MEMORY ON THE SYSTEM, THE LAST
2440 :* BLOCK ($LSTBK) IS USED AS A PAR AND A CARRY IS PROPAGATED TO
2441 :* CAUSE AN 18-BIT OVERFLOW.
2442 :*
2443 032010 012767 032034 147072 6$: MOV #16$,SLPERR ;SET LOOP ON ERROR POINTER TO 16$
2444 032016 005067 147312 CLR CPUEXP ;NO TRAPS THRU ERRVEC EXPECTED HERE
2445 032022 012767 007777 140320 MOV #7777,KIPAR4 ;LOAD PAR4 WITH HIGHEST VALUE POSSIBLE
2446 032030 005037 000000 CLR @#000000 ;CLEAR ADDRESS ZERO
2447 032034 013701 100100 16$: MOV @#100100,R1 ;THIS SHOULD READ ADDRESS ZERO INTO R1
2448 032040 005701 TST R1 ;SEE IF YOU READ ADDRESS ZERO
2449 032042 001401 BEQ 7$ ;BRANCH IF ADDRESS ZERO WAS READ
2450 032044 104045 ERROR +45 ;DIDN'T READ ADDRESS ZERO
2451 032046 012767 032112 147034 7$: MOV #17$,SLPERR ;SET LOOP ON ERROR POINTER TO 17$
2452 032054 022767 007377 150740 CMP #7377,$LSTBK ;IS MEMORY BLOCK SIZE < 7377
2453 032062 101423 BLOS 8$ ;BRANCH IF MORE THAN 120K ON SYSTEM
2454 032064 012767 000040 147242 MOV #40,CPUEXP ;EXPECTING NON-EXISTANT MEMORY ERROR
2455 032072 005067 147234 CLR PCPUER ;CLEAR TRAP THRU ERRVEC FLAG
2456 032076 016767 150720 140244 MOV $LSTBK,KIPAR4 ;GET READY TO GENERATE NON-EXISTANT ADDR.
2457 032104 062767 000100 140236 ADD #100,KIPAR4 ;MAKE SURE WE'RE IN NON-EXISTANT MEMORY.
2458 032112 013701 100100 17$: MOV @#100100,R1 ;READ FROM NON-EXISTANT ADDRESS
2459 032116 005767 147210 TST PCPUER ;SEE IF TRAP THRU ERRVEC OCCURRED
2460 032122 001003 BNE 8$ ;BRANCH TO EXIT IF TRAP OCCURED
2461 032124 012700 100100 MOV #100100,R0 ;SAVE VIRTUAL ADDRESS FOR ERROR TYPEOUT
2462 032130 104046 ERROR +46 ;NO TRAP THRU ERRVEC
2463 032132 005067 147176 8$: CLR CPUEXP ;NO CPU TRAPS EXPECTED
2464 032136 012767 031504 146744 MOV #20$,SLPERR ;SET LOOP ON ERROR POINTER TO START OF TEST
2465 032144 000462 BR TST50 ;BRANCH TO NEXT TEST
2466 :*
2467 :* ***** TRAP TO HERE THRU ERRVEC *****
2468 :*
2469 032146 012667 147154 10$: MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS
2470 032152 012667 147152 MOV (KSP)+,OLDPS ;SAVE RETURN PSW
  
```

```

2471 032156 016767 145604 147146      MOV    CPUERR,PCPUER    ;SAVE CPU ERROR REGISTER FOR TYPING
2472 032164 022767 000040 147140      CMP    #40,PCPUER      ;WAS TRAP NON-EXISTANT MEMORY
2473 032172 001012                    BNE    12$             ;BRANCH IF MEMORY EXISTS
2474 032174 026767 140150 150620      CMP    KIPAR4,$LSTBK   ;SEE IF PAR4 MATCHES LAST BLOCK IN MEMORY
2475 032202 002404                    BLT    11$             ;BRANCH IF NO MATCH-ERROR IN COMPARE CIRCUITS
2476 032204 012767 031724 147114      MOV    #4$,OLDPC       ;CHANGE RETURN ADDRESS IF AT TOP OF MEMORY
2477 032212 000430                    BR     14$             ;BRANCH TO EXIT
2478 032214 104047                    11$:  ERROR +47        ;PREMATURE END OF MEMORY FOUND
2479 032216 000426                    BR     14$             ;BRANCH TO EXIT
2480 032220 022767 000020 147104      12$:  CMP    #20,PCPUER   ;SEE IF ADDRESS TIMED OUT
2481 032226 001011                    BNE    13$             ;BRANCH IF NO TIME OUT,UNEXPECTED ERROR
2482 032230 005767 147102                    TST    HOLFLG         ;HAS THIS HAPPENED BEFORE?
2483 032234 001003                    BNE    15$             ;BRANCH IF NOT FIRST TIMEOUT
2484 032236 016767 140110 146734      MOV    KIPAR5,$TMP1    ;SAVE PAR WHERE HOLE FIRST DISCOVERED
2485 032244 005267 147066                    15$:  INC    HOLFLG     ;KEEP COUNT OF SUCCESSIVE TIMEOUTS
2486 032250 000411                    BR     14$             ;BRANCH TO EXIT
2487 032252 016767 147050 147044      13$:  MOV    OLDPC,BADPC  ;MOVE PC OF UNEXPECTED ERROR FOR TYPEOUT
2488 032260 104001                    ERROR +1              ;UNEXPECTED TRAP THROUGH ERRVEC
2489 032262 016767 146620 147036      MOV    $LPADR,OLDPC    ;RETURN TO BEGINNING OF TEST
2490 032270 005067 147042                    CLR    HOLFLG         ;CLEAR FLAG IN CASE IT WAS SET
2491 032274 005067 145466                    14$:  CLR    CPUERR      ;CLEAR CPU ERROR REGISTER
2492 032300 016746 147024                    MOV    OLDPS,-(KSP)   ;PUSH OLD PSW ONTO STACK
2493 032304 016746 147016                    MOV    OLDPC,-(KSP)   ;PUSH RETURN ADDRESS ONTO STACK
2494 032310 000002                    RTI                    ;RETURN TO TEST AND CONTINUE
2508
2509

```

```

:*****
:*TEST 50      22-BIT MAPPING CARRY PROPAGATION
:*
:*      THIS TEST USES FULL 22-BIT RELOCATION TO CHECK THE CARRY
:*      PROPAGATION THAT PERTAINS TO 22 BIT ADDRESSES. THIS TEST
:*      SCANS MEMORY FROM 00740000 TO 16740000 OR THE LAST BLOCK,
:*      ON 8K BOUNDARIES. IF ANY HOLES ARE FOUND, THE ADDRESS
:*      WHERE THE HOLE WAS DISCOVERED AND THE FIRST GOOD ADDRESS
:*      AFTER THE HOLE WILL BE REPORTED.
:*
:*      NOTE - PART OF THIS TEST WILL NOT BE RUN IF THERE IS LESS
:*      THAN 120K ON THE SYSTEM.
:*
:*****

```

```

032312 000004
2510 032314 012700 100100      TST50: SCOPE
2511 032320 012701 120000      MOV    #100100,R0      ;LOAD VIRTUAL ADDR FOR PAR4 INTO R0
2512 032324 012767 000020 140164      MOV    #120000,R1      ;LOAD VIRTUAL ADDR FOR PAR5 INTO R1
2513 032332 005767 147002      MOV    #BIT4,MMR3     ;ENABLE 22 BIT MAPPING
2514 032336 001004                    TST    HDWFLAG        ;IS THERE SPECIAL HARDWARE?
2515 032340 022767 007377 150454      BNE    20$             ;BRANCH IF THERE IS
2516 032346 003112                    CMP    #7377,$LSTBK   ;IS THERE AT LEAST 120K ON THE SYSTEM?
2517 032350 012767 033030 145426      20$:  BGT    4$             ;BRANCH IF LESS THAN 120K
2518 032356 005067 146754                    MOV    #10$,ERRVEC    ;SET ERROR VECTOR POINTER TO 10$
2519 032362 012767 007377 137760      CLR    HOLFLG         ;MAKE SURE HOLE FLAG STARTS AT ZERO
2520 032370 012767 007400 137754      MOV    #7377,KIPAR4   ;LOAD PAR4 WITH STARTING BASE
2521 032376 012702 000360                    MOV    #7400,KIPAR5   ;LOAD BASE ADDRESS +100 INTO KIPARS
2522 032402 012767 032410 146500      1$:  MOV    #360,R2       ;LOAD DATA PATTERN INTO R2
2523 032410 005767 146724      21$:  MOV    #21,$LPERR   ;SET LOOP ON ERROR POINTER TO 21$
2524 032414 001404                    TST    HDWFLAG        ;IS THERE SPECIAL HARDWARE?
2525 032416 026767 137730 150376      BEQ    22$             ;BRANCH IF NOT
2526 032424 003025                    CMP    KIPAR5,$LSTBK  ;ARE WE OUT OF MEMORY SPACE?
                                BGT    23$             ;BRANCH IF WE ARE

```

```

2527 032426 005067 146700      22$: CLR      PCPUER      ;CLEAR CPU ERROR FLAG
2528 032432 011167 146540      MOV      (R1), $TMP0 ;SAVE DATA AT TEST LOCATION
2529 032436 005767 146670      TST      PCPUER      ;SEE IF THERE WAS A CPU TRAP
2530 032442 001014                BNE      2$          ;BRANCH IF TRAP OCCURRED
2531 032444 005767 146666      TST      HOLFLG      ;SEE IF HOLE WAS FOUND IN MEMORY
2532 032450 001411                BEQ      2$          ;BRANCH IF NO HOLE WAS FOUND
2533 032452 016767 137674 146522  MOV      KIPAR5, $TMP2 ;SAVE PAR THAT POINTS TO END OF HOLE
2534 032460 012767 032350 146422  MOV      #20$, $LPERR ;SET LOOP ON ERROR POINTER TO 20$
2535 032466 104050                ERKOR    +50         ;HOLE IN MEMORY FROM $TMP1 TO $TMP2
2536 032470 005067 146642      CLR      HOLFLG      ;CLEAR HOLE FLAG IN CASE THERE ARE MORE
2537 032474 010210                2$: MOV      R2, (R0)   ;LOAD TEST PATTERN INTO TEST LOCATION
2538 032476 000405                BR       24$        ;BRANCH OVER NEXT FEW INSTRUCTIONS
2539 032500 010204                23$: MOV      R2, R4    ;GET CURRENT DATA VALUE
2540 032502 052704 000400      BIS      #BIT8, R4   ;SET SPECIAL HARDWARE ENABLE BIT
2541 032506 010467 145256      MOV      R4, RMIREG  ;SETUP SPECIAL HARDWARE
2542 032512 011103                24$: MOV      (R1), R3  ;READ TEST LOCATION VIA DIFFERENT VIRT. ADDR.
2543 032514 020203                CMP      R2, R3     ;SEE IF THE CORRECT LOCATION WAS REFERENCED
2544 032516 001401                BEQ      3$          ;BRANCH IF CORRECT DATA WAS OBTAINED
2545 032520 104051                ERKOR    +51         ;BAD RELOCATION 22-BIT MAPPING
2546 032522 016711 146450                3$: MOV      $TMP0, (R1) ;RESTORE ORIGINAL DATA TO TEST LOCATION
2547 032526 062767 000400 137614  ADD      #400, KIPAR4 ;CHANGE BASE ADDRESS
2548 032534 062767 000400 137610  ADD      #400, KIPAR5 ;CHANGE BASE ADDRESS
2549 032542 005302                DEC      R2          ;CHANGE DATA PATTERN
2550 032544 026767 137600 150250  CMP      KIPAR4, $LSTBK ;IS THE PAR ABOVE THE LAST BLOCK OF MEMORY
2551 032552 003010                BGT      4$          ;BRANCH IF ABOVE LAST MEM. LOC.
2552 032554 022767 167400 137570  CMP      #167400, KIPAR5 ;MAKE SURE YOU DON'T GO ON THE UNIBUS
2553 032562 103307                BHIS     1$          ;BRANCH IF NOT PAST LAST ADDRESS
2554 032564 005767 146546      TST      HOLFLG      ;SEE IF MEMORY ENDS WITH A HOLE
2555 032570 001401                BEQ      4$          ;BRANCH IF NO HOLE AT END OF MEMORY
2556 032572 104052                ERKOR    +52         ;HOLE AT END OF MEMORY
2557
2558      ;*
2559      ;* UNIBUS ADDRESS 000000 IS GENERATED BY CARRY PROPAGATION
2560      ;* SINCE THE MAP IS DISABLED, THIS SHOULD REFERENCE PHYSICAL
2561      ;* ADDRESS 000000.
2562 032574 012767 000020 137714  4$: MOV      #BIT4, MMR3 ;ENABLE 22-BIT MAPPING
2563 032602 012767 003024 145174  MOV      #TIMERR, ERRVEC ;RESTORE NORMAL ROUTINE FOR TRAPS THRU 4
2564 032610 012767 167777 137532  MOV      #167777, KIPAR4 ;GET READY TO TEST U.B. ADDRESS 0
2565 032616 012767 000000 137526  MOV      #000000, KIPAR5 ;SHOULD GO TO PHYSICAL ADDRESS 0
2566 032624 012702 017000                MOV      #17000, R2   ;LOAD DATA PATTERN INTO R2
2567 032630 012767 032650 146252  MOV      #5$, $LPERR  ;SET LOOP ON ERROR POINTER TO 5$
2568 032636 012767 000020 146470  MOV      #20, CPUEXP  ;EXPECTING UNIBUS TIMEOUT
2569 032644 011067 146326                MOV      (R0), $TMP0 ;SAVE DATA IN LOCATION 0 USING PAGE 4
2570 032650 010210                5$: MOV      R2, (R0)   ;LOAD DATA PATTERN INTO TEST LOCATION
2571 032652 011103                MOV      (R1), R3    ;READ TEST LOCATION VIA DIFFERENT VIRT ADDR
2572 032654 016710 146316                MOV      $TMP0, (R0) ;RESTORE ORIGINAL DATA USING PAGE 4
2573 032660 020203                CMP      R2, R3     ;SEE IF DATA MATCHES
2574 032662 001401                BEQ      6$          ;BRANCH IF TRAP
2575 032664 104053                ERKOR    +53         ;BAD RELOCATION, UNIBUS ADDRESS
2576
2577      ;*
2578      ;* ADDRESS 000000 IS GENERATED BY CARRY PROPAGATION. THEN, IF
2579      ;* THERE IS LESS THAN 120K OF MEMORY ON THE SYSTEM, THE LAST
2580      ;* BLOCK ($LSTBK) IS USED AS A PAR AND A CARRY IS PROPAGATED TO
2581      ;* CAUSE A 22-BIT OVERFLOW.
2582 032666 012767 032712 146214  6$: MOV      #16$, $LPERR ;SET LOOP ON ERROR POINTER TO 16$
2583 032674 005067 146434                CLR      CPUEXP     ;NO TRAPS THRU ERRVEC EXPECTED HERE
  
```



```

2584 032700 012767 177777 137442      MOV      #177777,KIPAR4 ;LOAD PAR4 WITH HIGHEST VALUE POSSIBLE
2585 032706 005037 000000              CLR      @#000000      ;CLEAR ADDRESS ZERO
2586 032712 013701 100100      16$:    MOV      @#100100,R1 ;THIS SHOULD READ ADDRESS ZERO INTO R1
2587 032716 005701              TST      R1           ;SEE IF YOU READ ADDRESS ZERO
2588 032720 001401              BEQ      7$          ;BRANCH IF ADDRESS ZERO WAS READ
2589 032722 104045              ERROR   +45         ;DIDN'T READ ADDRESS ZERO
2590 032724 012767 032770 146156 7$:    MOV      #17$, $LPERR ;SET LOOP ON ERROR POINTER TO 17$
2591 032732 022767 167777 150062      CMP      #167777,$LSTBK ;IS MEMORY BLOCK SIZE < 167777
2592 032740 101423              BLOS    8$          ;BRANCH IF MORE THAN 120K ON SYSTEM
2593 032742 012767 000040 146364      MOV      #40,CPUEXP   ;EXPECTING NON-EXISTANT MEMORY ERROR
2594 032750 005067 146356              CLR      PCPUER      ;CLEAR TRAP THRU ERRVEC FLAG
2595 032754 016767 150042 137366      MOV      $LSTBK,KIPAR4 ;GET READY TO GENERATE NON-EXISTANT ADDR.
2596 032762 062767 000100 137360      ADD      #100,KIPAR4 ;MAKE SURE WE ARE IN NON-EXISTANT MEMORY
2597 032770 013701 100100      17$:    MOV      @#100100,R1 ;READ FROM NON-EXISTANT ADDRESS
2598 032774 005767 146332              TST      PCPUER      ;SEE IF TRAP THRU ERRVEC OCCURRED
2599 033000 001003              BNE     8$          ;BRANCH TO EXIT IF TRAP OCCURED
2600 033002 012700 100100      MOV      #100100,R0   ;SAVE VIRTUAL ADDRESS FOR ERROR TYPEOUT
2601 033006 104046              ERROR   +46         ;NO TRAP THRU ERRVEC
2602 033010 005067 146320      8$:    CLR      CPUEXP     ;NO CPU TRAPS EXPECTED
2603 033014 012767 032350 146066      MOV      #20$, $LPERR ;SET LOOP ON ERROR POINTER TO START OF TEST
2604 033022 005067 137470              CLR      MMR3       ;RETURN TO 18-BIT MAPPING
2605 033026 000462              BR      TST51       ;:BRANCH TO NEXT TEST

```

;\*\*\*\*\* TRAP TO HERE THRU ERRVEC \*\*\*\*\*

```

2609 033030 012667 146272      10$:    MOV      (KSP)+,OLDPC ;SAVE RETURN ADDRESS
2610 033034 012667 146270      MOV      (KSP)+,OLDPS ;SAVE RETURN PSW
2611 033040 016767 144722 146264      MOV      CPUERR,PCPUER ;SAVE CPU ERROR REGISTER FOR TYPING
2612 033046 022767 000040 146256      CMP      #40,PCPUER  ;WAS TRAP NON-EXISTANT MEMORY
2613 033054 001012              BNE     12$         ;BRANCH IF MEMORY EXISTS
2614 033056 026767 137266 147736      CMP      KIPAR4,$LSTBK ;SEE IF PAR4 MATCHES LAST BLOCK IN MEMORY
2615 033064 002404              BLT     11$         ;BRANCH IF NO MATCH-ERROR IN COMPARE CIRCUITS
2616 033066 012767 032574 146232      MOV      #4$,OLDPC   ;CHANGE RETURN ADDRESS IF AT TOP OF MEMORY
2617 033074 000430              BR      14$         ;BRANCH TO EXIT
2618 033076 104047      11$:    ERROR   +47         ;PREMATURE END OF MEMORY FOUND
2619 033100 000426              BR      14$         ;BRANCH TO EXIT
2620 033102 022767 000020 146222 12$:    CMP      #20,PCPUER  ;SEE IF ADDRESS TIMED OUT
2621 033110 001011              BNE     13$         ;BRANCH IF NO TIME OUT,UNEXPECTED ERROR
2622 033112 005767 146220              TST     HOLFLG     ;HAS THIS HAPPENED BEFORE?
2623 033116 001003              BNE     15$         ;BRANCH IF NOT FIRST TIMEOUT
2624 033120 016767 137226 146052      MOV      KIPAR5,$TMP1 ;SAVE PAR WHERE HOLE FIRST DISCOVERED
2625 033126 005267 146204      15$:    INC      HOLFLG     ;KEEP COUNT OF SUCCESSIVE TIMEOUTS
2626 033132 000411              BR      14$         ;BRANCH TO EXIT
2627 033134 016767 146166 146162 13$:    MOV      OLDPC,BADPC ;MOVE PC OF UNEXPECTED ERROR FOR TYPEOUT
2628 033142 104001              ERROR   +1         ;UNEXPECTED TRAP THROUGH ERRVEC
2629 033144 016767 145736 146154      MOV      $LPADR,OLDPC ;RETURN TO BEGINNING OF TEST
2630 033152 005067 146160              CLR     HOLFLG     ;CLEAR FLAG IN CASE IT WAS SET
2631 033156 005067 144604      14$:    CLR     CPUERR     ;CLEAR CPU ERROR REGISTER
2632 033162 016746 146142      MOV     OLDPS,-(KSP) ;PUSH OLD PSW ONTO STACK
2633 033166 016746 146134      MOV     OLDPC,-(KSP) ;PUSH RETURN ADDRESS ONTO STACK
2634 033172 000002              RTI              ;RETURN TO TEST AND CONTINUE

```

```

;*****
;*TEST 51      READ AND WRITE WHILE IN RELOCATE MODE
;*
;*      THE FOLLOWING TEST TURNS ON MEMORY MANAGEMENT AND THEN
;*      READS AND WRITES LOCATIONS BETWEEN PHYSICAL ADDRESSES

```

```

: * 060000-067600. ONE LOCATION IN EVERY BLOCK (32. WORDS)
: * IS WRITTEN USING PAR4 AND READ USING PAR5. THIS IS
: * DONE IN BOTH USER AND KERNEL MODES. THE 'MODE' INPUT TO
: * THE PAR/PDR ADDRESS MUX IS CHECKED BY READING AND WRITING
: * IN USER MODE. REMEMBER ALSO, THAT SINCE MEMORY MANAGEMENT
: * IS ON (IN RELOCATE MODE) THE PROGRAM ITSELF IS USING ITS
: * VIRTUAL ADDRESSES AND THE PAR/PDR'S TO EXECUTE.

: * WHILE TESTING IN KERNEL MODE, USER PAGES 4 & 5 ARE MAPPED
: * NON-RESIDENT WITH DIFFERENT PAR VALUES THAN THE KERNEL
: * PAR'S TO BE SURE THAT THE KERNEL PAR'S AND PDR'S ARE BEING
: * USED WHEN IN KERNEL MODE (AND VICE VERSA WHILE TESTING IN
: * USER MODE). IF A MEM. MGMT. TRAP OCCURS, THE PROGRAM GOES
: * TO 8$ WHERE THE TRAP IS REPORTED.
    
```

\*\*\*\*\*  
 TST51: SCOPE

2657	033174	000004			
2658	033176	005067	144574	1\$:	CLR PSW ;START IN KERNEL MODE
2659	033202	012704	000577		MOV #577,R4 ;LOAD R4 WITH VALUE FOR PAR4
2660	033206	012705	000600		MOV #600,R5 ;LOAD R5 WITH VALUE FOR PAR5
2661	033212	010467	137132		MOV R4,KIPAR4 ;LOAD KERNEL PAR4
2662	033216	010567	137130		MOV R5,KIPAR5 ;LOAD KERNEL PAR5
2663	033222	012700	177640		MOV #UIPAR0,R0 ;LOAD ADDRESS OF FIRST USER PAR IN R0
2664	033226	012703	172240		MOV #SIPAR0,R3 ;LOAD ADDRESS OF FIRST SUPERVISOR PAR IN R3
2665	033232	005001			CLR R1 ;CLEAR R1
2666	033234	012702	000007		MOV #7,R2 ;LOAD LOOP COUNTER WITH A 7
2667	033240	010120		2\$:	MOV R1,(R0)+ ;MAP USER PAR'S TO PAGES 0-6 (4K EACH)
2668	033242	010123			MOV R1,(R3)+ ;MAP SUPERVISOR PAR'S TO PAGES 0-6 (4K EACH)
2669	033244	062701	000200		ADD #200,R1
2670	033250	077205			SOB R2,2\$ ;LOOP UNTIL UIPAR0-UIPAR6 ARE LOADED
2671	033252	012710	007600		MOV #7600,(R0) ;MAP USER PAR7 TO THE I/O PAGE
2672	033256	012713	007600		MOV #7600,(R3) ;MAP SUPERVISOR PAR7 TO THE I/O PAGE
2673	033262	012700	177600		MOV #UIPDR0,R0 ;LOAD ADDRESS OF FIRST USER PDR IN R0
2674	033266	012703	172200		MOV #SIPDR0,R3 ;LOAD ADDRESS OF FIRST SUPERVISOR PDR IN R3
2675	033272	012701	077406		MOV #77406,R1 ;LOAD PDR DATA INTO R1
2676	033276	012702	000010		MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8
2677	033302	010120		3\$:	MOV R1,(R0)+ ;MAP ALL 8 PAGES 128 BLOCKS, UPWARD
2678	033304	010123			MOV R1,(R3)+ ;EXPANDABLE, READ/WRITE FOR
2679	033306	077203			SOB R2,3\$ ;USER AND SUPERVISOR MODES.
2680	033310	105067	144274		CLRB UIPDR4 ;MAP USER SPACE NON-RESIDENT WHILE
2681	033314	105067	144272		CLRB UIPDR5 ;TESTING KERNEL SPACE
2682	033320	105067	136664		CLRB SIPDR4 ;MAP SUPERVISOR SPACE NON-RESIDENT WHILE
2683	033324	105067	136662		CLRB SIPDR5 ;TESTING KERNEL SPACE
2684	033330	010567	144314		MOV R5,UIPAR4 ;MAP USER PAR'S OPPOSITE OF KIPAR'S
2685	033334	010467	144312		MOV R4,UIPAR5
2686	033340	010567	136704		MOV R5,SIPAR4 ;MAP SUPERVISOR PAR'S OPPOSITE OF KIPAR'S
2687	033344	010467	136702		MOV R4,SIPAR5
2688	033350	012767	000001 144214		MOV #1,SRO ;TURN ON MEMORY MANAGEMENT (RELOCATE MODE)
2689	033356	012767	033410 145524		MOV #5\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 5\$
2690	033364	012767	033732 144656		MOV #8\$, \$MVEC ;SET M. M. TRAP VECTOR TO 8\$
2691	033372	016767	144400 145576	4\$:	MOV PSW,\$TMP0 ;SAVE PSW IN CASE OF ERROR
2692	033400	012700	100100		MOV #100100,R0 ;PUT VIRTUAL ADDR. THAT USES PAR4 IN R0
2693	033404	012701	120000		MOV #120000,R1 ;PUT VIRTUAL ADDR. THAT USES PAR5 IN R1
2694	033410	010010		5\$:	MOV R0,(R0) ;WRITE TO TEST LOC. USING PAR4
2695	033412	011102			MOV (R1),R2 ;READ THE SAME LOC., BUT USING PAR5
2696	033414	020002			CMP R0,R2 ;DID WE READ WHAT WE WROTE?



```

2697 033416 001411          BEQ      6$          ;BRANCH IF YES
2698 033420 010167 145654    MOV      R1,VIRT2    ;SAVE VIRTUAL ADDR. THAT SELECTED PAR5
2699 033424 010067 145646    MOV      R0,VIRT1    ;SAVE VIRTUAL ADDR. THAT SELECTED PAR4
2700 033430 004767 147564    JSR     PC,FORMPA    ;GO FORM PHYSICAL ADDRESS BEING USED
2701 033434 104020          ERROR   +20         ;READING LOC. USING PAR5 AND A VIRT.
2702                                ;ADDR. DID NOT FIND DATA WRITTEN WHEN USING
2703                                ;PAR4 AND VIRT. ADDRESS.
2704                                ;FOR TIGHTER SCOPE LOOP
2705                                ;REPLACE ERROR CALL WITH
2706                                ;'BR 5$' = 000765
2707 033436 016700 145634          MOV      VIRT1,R0    ;RESTORE VBA IN R0
2708 033442 062700 000100      6$:    ADD      #100,R0    ;CHANGE VIRTUAL ADDRS. TO POINT TO NEXT BLOCK
2709 033446 062701 000100      ADD      #100,R1
2710 033452 020127 127700      CMP     R1,#127700   ;WERE BLOCKS FROM 60000-67600 ALL TRIED?
2711 033456 001354          BNE     5$          ;BRANCH IF NO
2712 033460 032767 140000 144310 BIT     #140000,PSW   ;HAVE WE DONE TEST IN USER MODE YET?
2713 033466 001026          BNE     7$          ;BRANCH IF YES
2714 033470 010467 144154      MOV     R4,UIPAR4    ;LOAD USER PAR4
2715 033474 010567 144152      MOV     R5,UIPAR5    ;LOAD USER PAR5
2716 033500 112767 000006 144102 MOVB    #6,UIPDR4     ;MAP USER SPACE R/W TO TEST IT
2717 033506 112767 000006 144076 MOVB    #6,UIPDR5
2718 033514 105067 136570      CLRB   KIPDR4        ;MAP KERNEL SPACE NON-RESIDENT WHILE
2719 033520 105067 136566      CLRB   KIPDR5        ; TESTING USER SPACE
2720 033524 010567 136620      MOV     R5,KIPAR4    ;MAP KERNEL PAR'S OPPOSITE UIPAR'S
2721 033530 010467 136616      MOV     R4,KIPAR5
2722 033534 012767 140000 144234 MOV     #140000,PSW   ;GO TO USER MODE
2723 033542 000713          BR      4$          ;GO BACK AND READ/WRITE IN USER MODE
2724 033544 032767 040000 144224 7$:    BIT     #40000,PSW   ;HAVE WE DONE TEST IN SUPERVISOR MODE YET?
2725 033552 001026          BNE     10$         ;BRANCH IF YES
2726 033554 010467 136470      MOV     R4,SIPAR4    ;LOAD SUPERVISOR PAR4
2727 033560 010567 136466      MOV     R5,SIPAR5    ;LOAD SUPERVISOR PAR5
2728 033564 112767 000006 136416 MOVB    #6,SIPDR4     ;MAP SUPERVISOR SPACE R/W TO TEST IT
2729 033572 112767 000006 136412 MOVB    #6,SIPDR5
2730 033600 105067 144004      CLRB   UIPDR4        ;MAP USER SPACE NON-RESIDENT WHILE
2731 033604 105067 144002      CLRB   UIPDR5        ; TESTING USER SPACE
2732 033610 010567 144034      MOV     R5,UIPAR4    ;MAP USER PAR'S OPPOSITE SIPAR'S
2733 033614 010467 144032      MOV     R4,UIPAR5
2734 033620 012767 040000 144150 MOV     #40000,PSW   ;GO TO SUPERVISOR MODE
2735 033626 000661          BR      4$          ;GO BACK AND READ/WRITE IN SUPERVISOR MODE
2736 033630 005067 144142      10$:  CLR     PSW          ;GO BACK TO KERNEL MODE BEFORE LEAVING
2737 033634 012767 077406 136446 MOV     #77406,KIPDR4 ;REMAP KERNEL PAGES READ/WRITE
2738 033642 012767 077406 136442 MOV     #77406,KIPDR5
2739 033650 012767 077406 143732 MOV     #77406,UIPDR4 ;REMAP USER PAGES READ/WRITE
2740 033656 012767 077406 143726 MOV     #77406,UIPDR5
2741 033664 010567 136460      MOV     R5,KIPAR4    ;MAP KERNEL, SUPERVISOR AND USER PAR'S 4 & 5
2742 033670 010567 136456      MOV     R5,KIPAR5    ; BACK TO 12-16K
2743 033674 010567 143750      MOV     R5,UIPAR4
2744 033700 010567 143746      MOV     R5,UIPAR5
2745 033704 010567 136340      MOV     R5,SIPAR4
2746 033710 010567 136336      MOV     R5,SIPAR5
2747 033714 012767 003122 144326 MOV     #MMGMERR,MMVEC ;RESTORE ADDR. OF NORMAL M.M. TRAP ROUTINE
2748 033722 012767 033176 145160 MOV     #1$,SLPERR    ;RESET LOOP ON ERROR POINTER TO 1$
2749 033730 000427          BR      TST52       ;BRANCH TO NEXT TEST
2750
2751                                ;***** TRAP TO HERE THRU ERRVEC *****

```

```

2753
2754 033732 012667 145322      8$:  MOV      (KSP)+,TRAPPC  ;SAVE PC & PS OF TRAP
2755 033736 012667 145320      MOV      (KSP)+,TRAPPS
2756                                     ;PROGRAM WILL TRAP TO HERE IF TRY
2757                                     ;TO USE USER/SUPERVISOR PDR'S WHEN IN KERNEL MODE
2758                                     ;OR KERNEL PDR'S WHEN IN USER/SUPERVISOR MODE
2759 033742 010067 145330      MOV      RO,VIRT1      ;SAVE VIRTUAL ADDRESS FOR ERROR REPORT
2760 033746 004767 147246      JSR      PC,FORMPA     ;GO FORM THE PHYSICAL ADDRESS BEING USED
2761 033752 016767 143614 145304  MOV      SR0,WASSR0    ;SAVE SR0 & SR2 FOR ERROR REPORT
2762 033760 016767 143612 145302  MOV      SR2,WASSR2
2763 033766 042767 160000 143576  BIC      #160000,SR0   ;CLEAR ERROR BITS IN SR0
2764 033774 104022      ERROR +22             ;M.M. TRAP WHILE IN RELOCATE MODF -
2765                                     ;REFERENCED WRONG SET OF PDR'S
2766                                     ;FOR TIGHTER SCOPE LOOP
2767                                     ;REPLACE ERROR CALL WITH
2768                                     ;A 'NOP' = 000240
2769 033776 016746 145260      MOV      TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
2770 034002 016746 145252      MOV      TRAPPC,-(KSP)
2771 034006 000002      RTI                    ;RETURN TO TEST
  
```



```

034150 031527 000100 BIT (R5),#WBIT ;DID WRITING PDR CLEAR THE W-BIT?
034154 001401 BEQ 8$ ;BRANCH IF YES
034156 104056 ERROR +56 ;W-BIT DID NOT CLEAR BY WRITNG THE PDR
;FOR TIGHTER SCOPE LOOP, REPLACE ERROR CALL
;WITH 'BR 3$' = 000740
034160 062705 000002 8$: ADD #2,R5 ;POINT R5 TO THE NEXT PDR TO BE TESTED
034164 062703 020000 ADD #20000,R3 ;CHANGE VIRT. ADDR TO REF. NEXT PDR
034170 077445 SOB R4,3$ ;LOOP BACK TO 3$ UNTIL ALL 8 PDR'S TESTED
034172 012767 034012 144710 MOV #1$,$LPERR ;RESET LOOP ON ERROR POINTER TO 1$
034200 004767 146022 JSR PC,TON ;TURN T-BIT BACK ON FOR NEXT TEST
  
```

2783  
2787

```

*****
*TEST 53 W-BIT LOGIC TEST, SUPERVISOR PDR'S
* THIS TEST WRITES TO EIGHT (8) DIFFERENT VIRTUAL ADDRESSES
* (VBA'S = 17776,37776,57776,77776,117776,137776,157776 & 177776
* & PBA'S CONSTRUCTED = 17776, 37776, 57776, 77776, 77776,
* 77776, 77776, & 77776 RESPECTIVELY).
* WHICH SHOULD CAUSE THE 'W-BIT' TO SET IN EACH OF THE
* EIGHT (8) SUPERVISOR PAGE DESCRIPTOR REGISTERS. THE PDR'S
* ARE CHECKED TO SEE THAT IT'S W-BIT DOES SET WHEN THE
* PAGE IT IS MAPPED TO IS WRITTEN TO AND THAT THE W-BIT
* DOES NOT SET IN ANY OF THE OTHER PDR'S. SUPERVISOR PDR'S 3,4,5 & 6
* ARE MAPPED TO 12-16K FOR THIS TEST. ALSO THE W-BIT
* SHOULD BE CLEARED WHEN THE PDR IS WRITTEN TO. THE
* W-BIT PORTION OF THE PDR'S AND THE PAR/PDR ADRS MUX
* ARE BEING CHECKED.
*****
  
```

```

2788 034204 000004 TST53: SCOPE
2789 034206 012767 040000 143562 1$: MOV #40000,PSW ;GO TO SUPERVISOR MODE FOR THIS TEST
034214 004767 145752 JSR PC,TOFF ;TURN T-BIT TRAPPING OFF FOR THIS TEST
034220 012702 000004 MOV #4,R2 ;SET LOOP COUNTER TO 4
034224 012700 172246 MOV #SIPAR3,R0 ;LOAD ADDRESS OF SIPAR3 INTO R0
034230 012701 000600 MOV #600,R1 ;LOAD '12-16K' PAR VALUE INTO R1
034234 010120 2$: MOV R1,(R0)+ ;MAP PARS 3-6 TO 12-16K
034236 077202 SOB R2,2$ ;LOOP UNTIL ALL 4 OF THEM ARE LOADED
034240 012705 172200 MOV #SIPDRO,R5 ;LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
034244 012704 000010 MOV #8,R4 ;SET LOOP COUNTER TO 8
034250 012703 017776 MOV #17776,R3 ;INITIALIZE VIRTUAL ADDRESS TO BE IN R3
034254 012767 034262 144526 3$: MOV #3$,$LPERR ;SET LOOP ON ERROR POINTER TO 3$
034262 012700 172200 MOV #SIPDRO,R0 ;LOAD ADDR. OF FIRST PDR TO BE SETUP IN R0
034266 012702 000010 MOV #8,R2 ;SET LOOP COUNTER TO 8
034272 012701 077406 MOV #77406,R1 ;PUT 'W-BIT OFF DATA' INTO R1
034276 010120 4$: MOV R1,(R0)+ ;CLEAR ALL W-BITS BY WRITING TO ALL PDRS
034300 077202 SOB R2,4$ ;LOOP UNTIL ALL OF THEM ARE SET UP
034302 011313 MOV (R3),(R3) ;DO 'DATO' TO VIRTUAL ADDR.-SETTING A W-BIT
034304 031527 000100 BIT (R5),#WBIT ;DID THAT CAUSE W-BIT TO BE SET?
034310 001002 BNE 5$ ;BRANCH IF YES
034312 104056 ERROR +54 ;W-BIT DID NOT GET SET IN PDR
;FOR TIGHTER SCOPE LOOP, REPLACE ERROR
;CALL WITH 'BR 3$' = 000763
034314 000422 5$: BR 8$ ;SKIP CHECKING OTHER PDR'S-ERROR WILL SET W-BITS
034316 012702 000010 MOV #8,R2 ;SET LOOP COUNTER TO 8
034322 012700 172200 MOV #SIPDRO,R0 ;LOAD ADDR. OF FIRST PDR TO BE CHECKED IN R0
034326 031027 000100 6$: BIT (R0),#WBIT ;DID W-BIT IN OTHER PDRS REMAIN CLEAR?
034332 001403 BEQ 7$ ;BRANCH IF YES
034334 020500 CMP R5,R0 ;IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
034336 001401 BEQ 7$ ;BRANCH IF YES
  
```

```

034340 104055          ERROR +55          ;W-BIT GOT SET IN MORE THAN ONE PDR
;FOR TIGHTER SCOPE LOOP, REPLACE ERROR
;CALL WITH 'BR 3$' = 000750
034342 062700 000002 7$:  ADD #2,R0          ;POINT R0 TO NEXT PDR TO BE CHECKED
034346 077211          SOB R2,6$          ;LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
034350 010115          MOV R1,(R5)        ;WRITE TO THE PDR TESTED TO CLEAR W-BIT
034352 031527 000100  BIT (R5),#WBIT    ;DID WRITING PDR CLEAR THE W-BIT?
034356 001401          BEQ 8$              ;BRANCH IF YES
034360 104056          ERROR +56          ;W-BIT DID NOT CLEAR BY WRITING THE PDR
;FOR TIGHTER SCOPE LOOP, REPLACE ERROR CALL
;WITH 'BR 3$' = 000740
034362 062705 000002 8$:  ADD #2,R5          ;POINT R5 TO THE NEXT PDR TO BE TESTED
034366 062703 020000  ADD #20000,R3     ;CHANGE VIRT. ADDR TO REF. NEXT PDR
034372 077445          SOB R4,3$          ;LOOP BACK TO 3$ UNTIL ALL 8 PDR'S TESTED
034374 012767 034206 144506 MOV #1$,SLPERR    ;RESET LOOP ON ERROR POINTER TO 1$
034402 004767 145620  JSR PC,TON        ;TURN T-BIT BACK ON FOR NEXT TEST
2790 034406 005067 143364 CLR PSW           ;BACK TO KERNAL MODE BEFORE LEAVING
2791
2795
    
```

```

*****
*TEST 54      W-BIT LOGIC TEST, USER PDR'S
* THIS TEST WRITES TO EIGHT (8) DIFFERENT VIRTUAL ADDRESSES
* (VBA'S = 17776,37776,57776,77776,117776,137776,157776 & 177776
* & PBA'S CONSTRUCTED = 17776, 37776, 57776, 77776, 17776,
* 77776, 77776, & 77776 RESPECTIVELY).
* WHICH SHOULD CAUSE THE 'W-BIT' TO SET IN EACH OF THE
* EIGHT (8) USER PAGE DESCRIPTOR REGISTERS. THE PDR'S
* ARE CHECKED TO SEE THAT IT'S W-BIT DOES SET WHEN THE
* PAGE IT IS MAPPED TO IS WRITTEN TO AND THAT THE W-BIT
* DOES NOT SET IN ANY OF THE OTHER PDR'S. USER PDR'S 3,4,5 & 6
* ARE MAPPED TO 12-16K FOR THIS TEST. ALSO THE W-BIT
* SHOULD BE CLEARED WHEN THE PDR IS WRITTEN TO. THE
* W-BIT PORTION OF THE PDR'S AND THE PAR/PDR ADRS MUX
* ARE BEING CHECKED.
*****
    
```

```

2796 034412 000004          ST54: SCOPE
2797 034414 012767 140000 143354 1$:  MOV #140000,PSW    ;GO TO USER MODE FOR THIS TEST
;TURN T-BIT TRAPPING OFF FOR THIS TEST
034422 004767 145544          JSR PC,TOFF
034426 012702 000004          MOV #4,R2          ;SET LOOP COUNTER TO 4
034432 012700 177646          MOV #UIPAR3,R0     ;LOAD ADDRESS OF UIPAR3 INTO R0
034436 012701 000600          MOV #600,R1        ;LOAD '12-16K' PAR VALUE INTO R1
034442 010120          2$:  MOV R1,(R0)+        ;MAP PARS 3-6 TO 12-16K
034444 077202          SOB R2,2$          ;LOOP UNTIL ALL 4 OF THEM ARE LOADED
034446 012705 177600          MOV #UIPDR0,R5     ;LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
034452 012704 000010          MOV #8,R4          ;SET LOOP COUNTER TO 8
034456 012703 017776          MOV #17776,R3      ;INITIALIZE VIRTUAL ADDRESS TO BE IN R3
034462 012767 034470 144420 3$:  MOV #3$,SLPERR    ;SET LOOP ON ERROR POINTER TO 3$
034470 012700 177600          MOV #UIPDR0,R0     ;LOAD ADDR. OF FIRST PDR TO BE SETUP IN R0
034474 012702 000010          MOV #8,R2          ;SET LOOP COUNTER TO 8
034500 012701 077406          MOV #77406,R1     ;PUT 'W-BIT OFF DATA' INTO R1
034504 010120          4$:  MOV R1,(R0)+        ;CLEAR ALL W-BITS BY WRITING TO ALL PDRS
034506 077202          SOB R2,4$          ;LOOP UNTIL ALL OF THEM ARE SET UP
034510 011313          MOV (R3),(R3)     ;DO 'DATA' TO VIRTUAL ADDR.-SETTING A W-BIT
034512 031527 000100  BIT (R5),#WBIT    ;DID THAT CAUSE W-BIT TO BE SET?
034516 001002          BNE 5$           ;BRANCH IF YES
034520 104054          ERROR +54          ;W-BIT DID NOT GET SET IN PDR
;FOR TIGHTER SCOPE LOOP, REPLACE ERROR
;CALL WITH 'BR 3$' = 000763
    
```

034522	000422			BR	8\$			:SKIP CHECKING OTHER PDR'S-ERROR WILL SET W-BITS
034524	012702	000010		5\$: MOV	#8,R2			:SET LOOP COUNTER TO 8
034530	012700	177600		MOV	#UIPDR0,R0			:LOAD ADDR. OF FIRST PDR TO BE CHECKED IN R0
034534	031027	000100		6\$: BIT	(R0),#WBIT			:DID W-BIT IN OTHER PDRS REMAIN CLEAR?
034540	001403			BEQ	7\$			:BRANCH IF YES
034542	020500			CMP	R5,R0			:IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
034544	001401			BEQ	7\$			:BRANCH IF YES
034546	104055			ERROR	+55			:W-BIT GOT SET IN MORE THAN ONE PDR
								:FOR TIGHTER SCOPE LOOP, REPLACE ERROR
								:CALL WITH 'BR 3\$' = 000750
034550	062700	000002		7\$: ADD	#2,R0			:POINT R0 TO NEXT PDR TO BE CHECKED
034554	077211			SOB	R2,6\$			:LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
034556	010115			MOV	R1,(R5)			:WRITE TO THE PDR TESTED TO CLEAR W-BIT
034560	031527	000100		BIT	(R5),#WBIT			:DID WRITING PDR CLEAR THE W-BIT?
034564	001401			BEQ	8\$			:BRANCH IF YES
034566	104056			ERROR	+56			:W-BIT DID NOT CLEAR BY WRITING THE PDR
								:FOR TIGHTER SCOPE LOOP, REPLACE ERROR CALL
								:WITH 'BR 3\$' = 000740
034570	062705	000002		8\$: ADD	#2,R5			:POINT R5 TO THE NEXT PDR TO BE TESTED
034574	062703	020000		ADD	#20000,R3			:CHANGE VIRT. ADDR TO REF. NEXT PDR
034600	077445			SOB	R4,3\$			:LOOP BACK TO 3\$ UNTIL ALL 8 PDR'S TESTED
034602	012767	034414	144300	MOV	#1\$,\$LPERR			:RESET LOOP ON ERROR POINTER TO 1\$
034610	004767	145412		JSR	PC,TON			:TURN T-BIT BACK ON FOR NEXT TEST
2798 034614	005067	143156		CLR	PSW			:BACK TO KERNEL MODE BEFORE LEAVING

2810

```

*****
*TEST 55      TEST 'W-BIT NOT SET' CASES
*
*   THIS TEST CHECKS TWO SPECIAL CASES WHERE THE W-BIT DOES
*   NOT GET SET ON A WRITE.  FIRST CASE IS THAT THE W-BIT
*   SHOULD NOT SET IN PAGE DESCRIPTOR REG. 7 WHEN WRITING TO
*   STATUS REG SR0 (KERNEL PDR 7 IS USED).  SECOND CASE IS THAT
*   THE W-BIT IS NOT SET IF THE 'DATO' IS ABORTED DUE TO AN
*   ODD ADDRESS ERROR (KERNEL PDR3 & VIRTUAL ADDR 60001 ARE USED).
*
*****
  
```

```

034620 000004
2811
2812 034622 004767 145344
2813 034626 012701 077406
2814 034632 012767 034632 144250
2815 034640 010167 135442
2816 034644 012767 034656 143132
2817 034652 005267 023123
2818 034656 012706 001100
2819 034662 016702 135420
2820 034666 020102
2821 034670 001401
2822 034672 104057
2823
2824
2825
2826 034674 012767 003024 143102
2827 034702 012767 034622 144200
2828 034710 004767 145312
  
```

```

TST55: SCOPE
1$: JSR PC,TOFF ;TURN OFF T-BIT TRAPPING FOR THIS TEST
    MOV #77406,R1 ;PUT 'W-BIT OFF' VALUE FOR PDR IN R1
2$: MOV #2$,SLPERR ;SET LOOP ON ERROR POINTER TO 2$
    MOV R1,KIPDR3 ;LOAD KERNEL PDR3 WITH 77406 TO CLEAR W-BIT
    MOV #3$,ERRVEC ;SET UP LOC. 4 TO 3$ FOR ODD ADDR. ABORT
    INC 60001 ;CAUSE ODD ADDRESS ABORT THRU LOC. 4
3$: MOV #KERSTK,KSP ;RESTORE THE STACK POINTER
    MOV KIPDR3,R2 ;READ KIPDR3 INTO R2
    CMP R1,R2 ;WAS W-BIT LEFT CLEARED?
    BEQ 4$ ;BRANCH IF YES
    ERROR +57 ;W-BIT GOT SET DURING AN ODD ADDR. ABORT
    ;FOR TIGHTER SCOPE LOOP
    ;REPLACE ERROR CALL WITH
    ;'BR 2$' = 000757
4$: MOV #TIMERR,ERRVEC ;RESTORE NORMAL CPU TRAP ROUTINE TO LOC.4
    MOV #1$,SLPERR ;RESET LOOP ON ERROR POINTER TO 1$
    JSR PC,TON ;TURN T-BIT TRAPPING BACK ON
  
```

2830  
2831  
2832

```

.SBTTL *****
.SBTTL END OF PASS ROUTINE
:*****
:*INCREMENT THE PASS NUMBER ($PASS)
:*TYPE 'END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYY'
:*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
:*IF SW12=1 INHIBIT TRACE TRAP
:*IF THERE'S A MONITOR GO TO IT
:*IF THERE ISN'T JUMP TO LOOP
$EOP:
034714          SCOPE
034714 000004   CLR      $STNM      ;;ZERO THE TEST NUMBER
034716 005067 144160   INC      $PASS      ;;INCREMENT THE PASS NUMBER
034722 005267 144304   BIC      #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
034726 042767 100000 144276   DEC      (PC)+      ;;LOOP?
034734 005327   $EOPCT: .WORD    1
034736 000001   BGT      $DOAGN     ;;YES
034740 003072   MOV      (PC)+,@(PC)+ ;;RESTORE COUNTER
034742 012737   $ENDCT: .WORD    1
034744 000001   $EOPCT
034746 034736   TYPE    ,65$        ;;TYPE ASCIZ STRING
034750 104401 034756   BR      64$         ;;GET OVER THE ASCIZ
034754 000407   ;;65$: .ASCIZ <12><15>/END PASS #/
64$:
034774          MOV      $PASS,-(SP)   ;;SAVE $PASS FOR TYPEOUT
034774 016746 144232   ;;TYPE PASS NUMBER
035000 104405   TYPDS   ;;GO TYPE--DECIMAL ASCII WITH SIGN
035002 104401 035010   TYPE    ,67$        ;;TYPE ASCIZ STRING
035006 000421   BR      66$         ;;GET OVER THE ASCIZ
66$:
035052          .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
035052 016746 144034   MOV      $ERTTL,-(SP) ;;SAVE $ERTTL FOR TYPEOUT
035056 104405   ;;TOTAL NUMBER OF ERRORS
035060 104401 001221   TYPDS   ;;GO TYPE--DECIMAL ASCII WITH SIGN
035064 005067 144022   TYPE    , $CRLF     ;;TYPE CARRIAGE RETURN, LINE FEED
035070 013700 000042   $GET42: MOV      @#42,R0 ;;GET MONITOR ADDRESS
035074 001414   BEQ     $DOAGN     ;;BRANCH IF NO MONITOR
035076 005046   CLR    -(SP)      ;;INSURE THE 'T' BIT IS CLEAR
035100 012746 035106   MOV     # $CLR.T,-(SP) ;;SETUP FOR AN RTI OR RTT
035104 000426   BR     $RTN       ;;GO DO AN RTI OR RTT TO LOAD THE PSW
;;WITH A CLEARED 'T' BIT
$CLR.T:
035106          MOV     @#42,R0 ;;INSURE R0 CONTAINS THE MONITORS
035112          BEQ     $DOAGN ;;RETURN ADDRESS
035114          RESET  ;;CLEAR THE WORLD
035116          $ENDAD: JSR   PC,(R0) ;;GO TO MONITOR
035120          NOP    ;;SAVE ROOM
035122          NOP    ;;FOR
035124          NOP    ;;ACT11
035126          $DOAGN: TRAP  ;;PUSH OLD PSW AND PC ON STACK
035130          BIC   #20,(SP) ;;CLEAR THE 'T' BIT
035134          BIT   #BIT12,@SWR ;;RUN WITH TRACE TRAP?
035142          BNE   1$     ;;BR IF NO
035144          COM   $TBIT  ;;IS IT TIME FOR TRACE TRAP

```



```
035150 100402          BMI 1$          ;;BR IF NO
035152 052716 000020   BIS #20,(SP)     ;;SET TRACE TRAP
035156 012746 035164   1$: MOV #SLOOP,-(SP) ;;JUMP TO START OF TEST
035162 000002          $RTRN: RTI          ;;RETURN--THIS IS CHANGED TO
                                ;;AN 'RTT' IF 'RTT' IS A LEGAL
                                ;;INSTRUCTION
```

```
035164          $LOOP:          ;;RETURN
035164 000137          JMP @ (PC)+
035166 020456          $RTNAD: .WORD LOOP
035170 377 377 000 $ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
                                .EVEN
```

2833

.SBTTL SCOPE HANDLER ROUTINE

```
*****
*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
*AND LOAD THE ERROR FLAG ($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
```

```
035174          $SCOPE:          ;;SCOPE=IOT
035174 104410          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
035176 032777 040000 143734 1$: BIT #BIT14,@SWR ;;LOOP ON PRESENT TEST?
035204 001062          BNE $OVER          ;;YES IF SW14=1
                                ;#####START OF CODE FOR THE XOR TESTER#####
035206 000416          $XTSTR: BR 6$          ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
                                ;;THIS INSTRUCTION TO A 'NOP' (NOP 240)
035210 013746 000004          MOV @ERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
035214 012737 035234 000004          MOV #5$,@ERRVEC ;;SET FOR TIMEOUT
035222 005737 177060          TST @177060 ;;TIME OUT ON XOR?
035226 012637 000004          MOV (SP)+,@ERRVEC ;;RESTORE THE ERROR VECTOR
035232 000431          BR $SVLAD ;;GO TO THE NEXT TEST
035234 022626          5$: CMP (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
035236 012637 000004          MOV (SP)+,@ERRVEC ;;RESTORE THE ERROR VECTOR
035242 000417          BR 7$          ;;LOOP ON THE PRESENT TEST
035244          6$:;#####END OF CODE FOR THE XOR TESTER#####
035244 032777 000400 143666          BIT #BIT08,@SWR ;;LOOP ON SPEC. TEST?
035252 001404          BEQ 2$          ;;BR IF NO
035254 127767 143660 143620          CMPB @SWR,$TSTNM ;;ON THE RIGHT TEST? SWR<7:0>
035262 001433          BEQ $OVER          ;;BR IF YES
035264 105767 143613          2$: TSTB $ERFLG ;;HAS AN ERROR OCCURRED?
035270 001412          BEQ $SVLAD ;;BR IF NO
035272 032777 001000 143640          BIT #BIT09,@SWR ;;LOOP ON ERROR?
035300 001404          BEQ 4$          ;;BR IF NO
035302 016767 143602 143576          7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
035310 000420          BR $OVER
035312 105067 143565          4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
035316 105267 143560          $SVLAD: INCB $TSTNM ;;COUNT TEST NUMBERS
035322 116767 143554 143700          MOVB $TSTNM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX
035330 011667 143552          MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
035334 011667 143550          MOV (SP),$LPERR ;;SAVE ERROR LOOP ADDRESS
035340 005067 143646          CLR $ESCAPE ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
035344 112767 000001 143543          MOVB #1,$ERMAX ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
035352 016777 143524 143562          $OVER: MOV $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
035360 016716 143522          MOV $LPADR,(SP) ;;FUDGE RETURN ADDRESS
```



```

035620 000000          HALT          ;;YES
035622          6$: RTI              ;;RETURN
035622 000002
2835
2836 035624 104401 001221 ERRRTYP: TYPE    ,SCLRF      ;'CARRIAGE RETURN' & 'LINE FEED'
2837 035630 010046          MOV      RO,-(KSP)  ;SAVE RO.
2838 035632 005000          CLR      RO          ;PICKUP THE ITEM INDEX
2839 035634 153700 001114          BISB    @#SITEMB,RO
2840 035640 001004          BNE     1$          ;IF ITEM NUMBER IS ZERO, JUST
2841          ;TYPE THE PC OF THE ERROR
2842 035642 016746 143250          MOV     $ERRPC,-(SP) ;SAVE $ERRPC FOR TYPEOUT
2843          ;ERROR ADDRESS
2844 035646 104402          TYPCC   ;GO TYPE--OCTAL ASCII(ALL DIGITS)
2845 035650 000522          BR      13$        ;GET OUT
2846 035652 005300          1$: DEC     RO          ;ADJUST THE INDEX SO THAT IT WILL
2847 035654 006300          ASL     RO          ;WORK FOR THE ERROR TABLE.
2848 035656 006300          ASL     RO
2849 035660 006300          ASL     RO
2850 035662 062700 001372          ADD     #SERRTB,RO  ;FORM TABLE POINTER
2851 035666 012067 000004          MOV     (RO)+,2$   ;PICKUP 'ERROR MESSAGE' POINTER
2852 035672 001404          BEQ     3$          ;SKIP TYPEOUT IF NO POINTER
2853 035674 104401          TYPE   ;TYPE THE 'ERROR MESSAGE'
2854 035676 000000          2$: .WORD   0        ;'ERROR MESSAGE' POINTER GOES HERE
2855 035700 104401 001221          TYPE   ,SCLRF      ;'CARRIAGE RETURN' & 'LINE FEED'
2856 035704 012067 000004          3$: MOV     (RO)+,4$ ;PICKUP 'DATA HEADER' POINTER
2857 035710 001404          BEQ     5$          ;SKIP TYPEOUT IF 0
2858 035712 104401          TYPE   ;TYPE THE 'DATA HEADER'
2859 035714 000000          4$: .WORD   0        ;'DATA HEADER' POINTER GOES HERE
2860 035716 104401 001221          TYPE   ,SCLRF      ;'CARRIAGE RETURN' & 'LINE FEED'
2861 035722 010146          5$: MOV     R1,-(KSP) ;SAVE R1
2862 035724 012001          MOV     (R0)+,R1   ;PICKUP 'DATA TABLE' POINTER
2863 035726 001472          BEQ     12$        ;BR IF NO DATA TO BE TYPED
2864 035730 012000          MOV     (R0)+,RO   ;PICKUP 'DATA FORMAT' POINTER
2865 035732 105710          6$: TSTB    (RO)      ;IS IT FORMAT 0?
2866 035734 001003          BNE     7$          ;BR IF NO
2867          ;*THIS CODE IS FOR OCTAL (16-BIT) FORMAT (DF=0)
2868 035736 013146          MOV     @ (R1)+,-(SP) ;SAVE @ (R1)+ FOR TYPEOUT
2869 035740 104402          TYPCC   ;GO TYPE--OCTAL ASCII(ALL DIGITS)
2870 035742 000456          BR      11$
2871          ;*THIS CODE IS FOR DECIMAL FORMAT (DF=1)
2872 035744 121027 000001          7$: CMPB    (RO),#1   ;IS IT FORMAT 1?
2873 035750 001003          BNE     8$          ;BRANCH IF NO
2874 035752 013146          MOV     @ (R1)+,-(SP) ;SAVE @ (R1)+ FOR TYPEOUT
2875 035754 104405          TYPDS   ;GO TYPE--DECIMAL ASCII WITH SIGN
2876 035756 000450          BR      11$
2877          ;*THIS CODE IS FOR BINARY FORMAT (DF=2)
2878 035760 121027 000002          8$: CMPB    (RO),#2   ;IS IT FORMAT 2
2879 035764 001003          BNE     9$          ;BRANCH IF NO
2880 035766 013146          MOV     @ (R1)+,-(SP) ;SAVE @ (R1)+ FOR TYPEOUT
2881 035770 104406          TYPBN   ;GO TYPE--BINARY ASCII
2882 035772 000442          BR      11$
2883          ;*THIS CODE IS FOR OCTAL (22-BIT) FORMAT (DF=3)
2884 035774 121027 000003          9$: CMPB    (RO),#3   ;IS IT FORMAT 3?
2885 036000 001011          BNE     15$        ;BRANCH IF NO
2886 036002 012146          MOV     (R1)+,-(KSP) ;PUT ADDRESS OF FIRST LOC. ON STACK
2887 036004 004767 002640          JSR     PC,$DB20   ;CONVERT TWO LOCS. TO AN ASCII STRING
2888 036010 062716 000003          ADD     #3,(KSP)   ;ONLY NEED 8 CHARACTERS NOT 11

```

```

2889 036014 012667 000002      MOV      (KSP)+,10$      ;PUT ADDRESS OF ASCII CHARS. AT 10$
2890 036020 104401              TYPE                    ;TYPE OCTAL VALUE OF 22-BIT BINARY NO.
2891 036022 000000      10$: .WORD      0
2892                          ;*THIS CODE IS FOR OCTAL (22-BIT) FORMAT FOR A PAR LEFT SHIFTED 6 (DF=4)
2893 036024 010246      15$: MOV      R2,-(KSP)    ;SAVE R2 ON STACK
2894 036026 010346      MOV      R3,-(KSP)    ;SAVE R3 ON STACK
2895 036030 013103      MOV      @R1)+,R3     ;LOAD DATA WORD INTO R3
2896 036032 005002      CLR      R2           ;R2 HOLDS UPPER SIX BITS OF NUMBER
2897 036034 073227 000006      ASHC     #6,R2        ;SHIFT VALUE LEFT 6 TIMES
2898 036040 010267 143142      MOV      R2,$TMP4     ;HOLDS LOWER 16 BITS OF ADDRESS
2899 036044 010367 143140      MOV      R3,$TMP5     ;HOLDS UPPER 6 BITS OF ADDRESS
2900 036050 012746 001206      MOV      #$TMP4,-(KSP) ;PUT ADDRESS OF LOWER BITS ONTO STACK
2901 036054 004767 002570      JSR      PC,$DB20     ;CONVERT TWO LOCS. TO AN ASCII STRING
2902 036060 062716 000003      ADD      #3,(KSP)     ;ONLY NEED 8 CHARACTERS NOT 11
2903 036064 012667 000002      MOV      (KSP)+,16$   ;PUT ADDRESS OF ASCII CHARS. AT 16$
2904 036070 104401              TYPE                    ;TYPE OCTAL VALUE OF 22-BIT BINARY NO.
2905 036072 000000      16$: .WORD      0
2906 036074 012603      MOV      (KSP)+,R3    ;RESTORE R3
2907 036076 012602      MOV      (KSP)+,R2    ;RESTORE R2
2908 036100 005711      11$: TST      (R1)      ;IS THERE ANOTHER NUMBER?
2909 036102 001404      BEQ      12$          ;BR IF NO
2910 036104 104401 036126      TYPE     ,14$        ;TYPE TWO(2) SPACES
2911 036110 105720      TSTB     (R0)+        ;POINT TO NEW 'DATA FORMAT'
2912 036112 000707      BR       6$          ;LOOP
2913 036114 012601      12$: MOV      (KSP)+,R1  ;RESTORE R1
2914 036116 012600      13$: MOV      (KSP)+,R0  ;RESTORE R0
2915 036120 104401 001221      TYPE     ,$CRLF      ;'CARRIAGE RETURN' & 'LINE FEED'
2916 036124 000207      RTS      PC          ;RETURN
2917 036126 040 040 000 14$: .ASCIZ / / ;TWO(2) SPACES
2918 036131 000
2919
2920

```

```

.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.

```

```

036132 022767 000176 143000 $CKSWR: CMP      #SWREG,SWR    ;IS THE SOFT-SWR SELECTED?
036140 001114      BNE      15$          ;BRANCH IF NO
036142 105777 142776      TSTB     @TKS        ;CHAR THERE?
036146 100111      BPL      15$          ;IF NO, DON'T WAIT AROUND
036150 117746 142772      MOVB     @TKB,-(SP)   ;SAVE THE CHAR
036154 042716 177600      BIC      #^C177,(SP) ;STRIP-OFF THE ASCII
036160 022726 000007      CMP      #7,(SP)+    ;IS IT A CONTROL G?
036164 001102      BNE      15$          ;NO, RETURN TO USER
036166 126727 142742 000001      CMPB     $AUTOB,#1   ;ARE WE RUNNING IN AUTO-MODE?
036174 001476      BEQ      15$          ;BRANCH IF YES
036176 104401 037067      TYPE     ,SCNTLG     ;ECHO THE CONTROL-G (^G)
036202 104401 037074      $GTSWR: TYPE     ,SMSWR ;TYPE CURRENT CONTENTS
036206 016746 141764      MOV      SWREG,-(SP) ;SAVE SWREG FOR TYPEOUT
036212 104402      TYPOC
036214 104401 037105      TYPE     ,SMNEW      ;PROMPT FOR NEW SWR
036220 005046      19$: CLR      -(SP)     ;CLEAR COUNTER
036222 005046      CLR      -(SP)     ;THE NEW SWR
036224 105777 142714      7$: TSTB     @TKS        ;CHAR THERE?

```

```

036230 100375          BPL      7$          ;; IF NOT TRY AGAIN
036232 117746 142710  MOVB    @STKB,-(SP)  ;; PICK UP CHAR
036236 042716 177600  BIC    #^C177,(SP)  ;; MAKE IT 7-BIT ASCII
036242 021627 000003  CMP    (SP),#3      ;; IS IT A CONTROL-C?
036246 001015          BNE    9$          ;; BRANCH IF NOT
036250 104401 001350  TYPE   ,SCNTLC      ;; YES, ECHO CONTROL-C (^C)
036254 062706 000006  ADD    #6,SP        ;; CLEAN UP STACK
036260 126727 142651 000001  CMPB   $INTAG,#1    ;; REENABLE TTY KEYBOARD INTERRUPTS?
036266 001003          BNE    8$          ;; BRANCH IF NO
036270 012777 000100 142646  MOV    #100,@STKS   ;; ALLOW TTY KEYBOARD INTERRUPTS
036276 000167 000614 8$:    JMP    CNTRLC      ;; CONTROL-C RESTART
036302 021627 000025 9$:    CMP    (SP),#25    ;; IS IT A CONTROL-U?
036306 001005          BNE    10$         ;; BRANCH IF NOT
036310 104401 037062  TYPE   ,SCNTLU      ;; YES, ECHO CONTROL-U (^U)
036314 062706 000006 20$:   ADD    #6,SP        ;; IGNORE PREVIOUS INPUT
036320 000737          BR     19$         ;; LET'S TRY IT AGAIN
036322 021627 000015 10$:   CMP    (SP),#15    ;; IS IT A <CR>?
036326 001022          BNE    16$         ;; BRANCH IF NO
036330 005766 000004  TST    4(SP)        ;; YES, IS IT THE FIRST CHAR?
036334 001403          BEQ    11$         ;; BRANCH IF YES
036336 016677 000002 142574  MOV    2(SP),@SWR   ;; SAVE NEW SWR
036344 062706 000006 11$:   ADD    #6,SP        ;; CLEAN UP STACK
036350 104401 001221 14$:   TYPE   ,$CRLF      ;; ECHO <CR> AND <LF>
036354 126727 142555 000001  CMPB   $INTAG,#1    ;; RE-ENABLE TTY KBD INTERRUPTS?
036362 001003          BNE    15$         ;; BRANCH IF NOT
036364 012777 000100 142552  MOV    #100,@STKS   ;; RE-ENABLE TTY KBD INTERRUPTS
036372 000002          RTI                    ;; RETURN
036374 004767 001040 15$:   JSR    PC,$TYPEC    ;; ECHO CHAR
036400 021627 000060 16$:   CMP    (SP),#60    ;; CHAR < 0?
036404 002420          BLT    18$         ;; BRANCH IF YES
036406 021627 000067  CMP    (SP),#67    ;; CHAR > 7?
036412 003015          BGT    18$         ;; BRANCH IF YES
036414 042726 000060  BIC    #60,(SP)+    ;; STRIP-OFF ASCII
036420 005766 000002  TST    2(SP)        ;; IS THIS THE FIRST CHAR
036424 001403          BEQ    17$         ;; BRANCH IF YES
036426 006316          ASL    (SP)         ;; NO, SHIFT PRESENT
036430 006316          ASL    (SP)         ;; CHAR OVER TO MAKE
036432 006316          ASL    (SP)         ;; ROOM FOR NEW ONE.
036434 005266 000002 17$:   INC    2(SP)        ;; KEEP COUNT OF CHAR
036440 056616 177776  BIS    -2(SP),(SP)  ;; SET IN NEW CHAR
036444 000667          BR     7$          ;; GET THE NEXT ONE
036446 104401 001220 18$:   TYPE   ,SQUES      ;; TYPE ?<CR><LF>
036452 000720          BR     20$         ;; 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
*
$RDCHR: MOV    (SP),-(SP)  ;; PUSH DOWN THE PC
036454 011646          MOV    4(SP),2(SP)   ;; SAVE THE PS
036456 016666 000004 000002 1$:    TSTB   @STKS        ;; WAIT FOR
036464 105777 142454          BPL    1$          ;; A CHARACTER
036470 100375          MOVB   @STKB,4(SP)   ;; READ THE TTY
036472 117766 142450 000004          BIC    #^C<177>,4(SP) ;; GET RID OF JUNK IF ANY
036500 042766 177600 000004

```

```
036506 026627 000004 000023      CMP      4(SP),#23      ;; IS IT A CONTROL-S?
036514 001013                    BNE      3$           ;; BRANCH IF NO
036516 105777 142422          2$:      TSTB     @STKS        ;; WAIT FOR A CHARACTER
036522 100375                    BPL      2$           ;; LOOP UNTIL ITS THERE
036524 117746 142416      MOVB     @STKB,-(SP)    ;; GET CHARACTER
036530 042716 177600      BIC      #^C177,(SP)   ;; MAKE IT 7-BIT ASCII
036534 022627 000021      CMP      (SP)+,#21    ;; IS IT A CONTROL-Q?
036540 001366                    BNE      2$           ;; IF NOT DISCARD IT
036542 000750                    BR        1$           ;; YES, RESUME
036544 026627 000004 000140      3$:      CMP      4(SP),#140  ;; IS IT UPPER CASE?
036552 002407                    BLT      4$           ;; BRANCH IF YES
036554 026627 000004 000175      CMP      4(SP),#175  ;; IS IT A SPECIAL CHAR?
036562 003003                    BGT      4$           ;; BRANCH IF YES
036564 042766 000040 000004      BIC      #40,4(SP)   ;; MAKE IT UPPER CASE
036572 000002          4$:      RTI          ;; GO BACK TO USER
;*****
;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
;*CALL:
;*      RDLIN          ;; INPUT A STRING FROM THE TTY
;*      RETURN HERE   ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
;*                  ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
$RDLIN: MOV      R3,-(SP)    ;; SAVE R3
        CLR      -(SP)     ;; CLEAR THE RUBOUT KEY
036574 010346          1$:      MOV      #$TTYIN,R3  ;; GET ADDRESS
036576 005046          2$:      CMP      #$TTYIN+8.,R3  ;; BUFFER FULL?
036600 012703 037052      BLOS     4$           ;; BR IF YES
036604 022703 037062      RDCHR   (SP)+,(R3)    ;; GO READ ONE CHARACTER FROM THE TTY
036610 101467      MOVB     #3,(R3)   ;; GET CHARACTER
036612 104411      BNE      10$          ;; IS IT A CONTROL-C?
036614 112613      TYPE     ,SCNTLC  ;; TYPE A CONTROL-C (^C)
036616 122713 000003      TST     (SP)+        ;; CLEAN RUBOUT KEY OFF OF THE STACK
036622 001006      MOV      (SP)+,R3    ;; RESTORE R3
036624 104401 001350      JMP     CNTRLC       ;; GOTO CONTROL-C RESTART
036630 005726          10$:     CMPB     #177,(R3)    ;; IS IT A RUBOUT
036632 012603      BNE      5$           ;; BR IF NO
036634 000167 000256      TST     (SP)         ;; IS THIS THE FIRST RUBOUT?
036640 122713 000177      BNE      6$           ;; BR IF NO
036644 001022      MOVB     #' \,9$    ;; TYPE A BACK SLASH
036646 005716          6$:      TYPE     ,9$
036650 001007      MOV      #-1,(SP)   ;; SET THE RUBOUT KEY
036652 112767 000134 000170      DEC     R3           ;; BACKUP BY ONE
036660 104401 037050      CMP     R3,$$TYIN   ;; STACK EMPTY?
036664 012716 177777      BLO     4$           ;; BR IF YES
036670 005303          5$:      SETUP  TO TYPEOUT THE DELETED CHAR.
036672 020327 037052      BR      2$           ;; GO READ ANOTHER CHAR.
036676 103434          5$:      TST     (SP)         ;; RUBOUT KEY SET?
036700 111367 000144      BEQ     7$           ;; BR IF NO
036704 104401 037050      MOVB     #' \,9$    ;; TYPE A BACK SLASH
036710 000735          7$:      TYPE     ,9$
036712 005716          7$:      CLR     (SP)         ;; CLEAR THE RUBOUT KEY
036714 001406          7$:      CMPB     #25,(R3)    ;; IS CHARACTER A CTRL U?
036716 112767 000134 000124      BNE      8$           ;; BR IF NO
036724 104401 037050      TYPE     ,SCNTLU   ;; TYPE A CONTROL 'U'
036730 005016          8$:      BR      1$           ;; GO START OVER
036732 122713 000025      BR      1$           ;; IS CHARACTER A 'R'?
036736 001003          8$:      CMPB     #22,(R3)
036740 104401 037062
036744 000715
036746 122713 000022
```

```

036752 001011      BNE      3$      ;;BRANCH IF NO
036754 105013      CLRB     (R3)     ;;CLEAR THE CHARACTER
036756 104401 001221 TYPE     ,SCLRF  ;;TYPE A 'CR' & 'LF'
036762 104401 037052 TYPE     ,STTYIN  ;;TYPE THE INPUT STRING
036766 000706      BR       2$      ;;GO PICKUP ANOTHER CHACTER
036770 104401 001220 4$:     TYPE     ,SQUES  ;;TYPE A '?'
036774 000701      BR       1$      ;;CLEAR THE BUFFER AND LOOP
036776 111367 000046 3$:     MOVB     (R3),9$  ;;ECHO THE CHARACTER
037002 104401 037050 TYPE     ,9$
037006 122723 000015 CMPB     #15,(R3)+ ;;CHECK FOR RETURN
037012 001274      BNE     2$      ;;LOOP IF NOT RETURN
037014 105063 177777 CLRB     -1(R3)   ;;CLEAR RETURN (THE 15)
037020 104401 001222 TYPE     ,SLF     ;;TYPE A LINE FEED
037024 00572C      TST     (SP)+   ;;CLEAN RUBOUT KEY FROM THE STACK
037026 012603      MOV     (SP)+,R3  ;;RESTORE R3
037030 011646      MOV     (SP),-(SP) ;;ADJUST THE STACK AND PUT ADDRESS OF THE
037032 016666 000004 000002 MOV     4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
037040 012766 037052 000004 MOV     #STTYIN,4(SP)
037046 000002      RTI
037050          000      9$:     .BYTE    0      ;;STORAGE FOR ASCII CHAR. TO TYPE
037051          000      .BYTE    0      ;;TERMINATOR
037052          .BLKB   8.      ;;RESERVE 8 BYTES FOR TTY INPUT
037062          136      125      015  $CNTLU: .ASCIZ  /^U/<15><12> ;;CONTROL 'U'
037065          012      000
037067          136      107      015  $CNTLG: .ASCIZ  /^G/<15><12> ;;CONTROL 'G'
037072          012      000
037074          015      012      123  $MSWR:  .ASCIZ  <15><12>/SWR - /
037077          127      122      040
037102          075      040      000
037105          040      040      116  $MNEW:  .ASCIZ  / NEW - /
037110          105      127      040
037113          075      040      000

2921
2922          .SBTTL  CONTROL-C SERVICING ROUTINE
2923
2924 037116 016767 142110 142064 CNTRLC: MOV     $PASS,$TMP5  ;;GET THE VALUE OF '$PASS'
2925 037124 005267 142060      INC     $TMP5      ;;FORM CURRENT PASS #
2926 037130 104401 037175      TYPE     ,CMMSG    ;;TYPE THE TEST STOPS HERE
2927 037134 116767 141742 000026 MOVB     $STNM,1$   ;;SAVE TEST NUMBER
2928 037142 016746 000022      MOV     1$,-(SP)  ;;SAVE 1$ FO TYPEOUT
2929 037146 104402      TYPOC
2930 037150 104401 037172      TYPE     ,2$
2931 037154 016746 142030      MOV     $TMP5,-(SP) ;;SAVE $TMP5 FOR TYPEOUT
2932 037160 104405      TYPDS      ;;TYPE ASCII DECIMAL WITH SIGN
2933 037162 104407      GTSWR      ;;ASK FOR NEW SWR VALUE
2934 037164 000167 175526      JMP     $EOP+2     ;;JUMP TO END OF PASS + 2
2935 037170 000000      1$:     .WORD    0      ;;TEST # BUFFER
2936 037172          040      040      000 2$:     .ASCIZ  / /      ;;2 SPACES & STOP MESSAGE
2937 037175          112      125      115  CMMSG:  .ASCII  /JUMPING TO END OF PASS/<15><12>
037200          120      111      116
037203          107      040      124
037206          117      040      105
037211          116      104      040
037214          117      106      040
037217          120      101      123
037222          123      015      012
2938 037225          000      .BYTE    0

```

2939

```

.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
*
037226 105767 141725 $TYPE: TSTB $TPFLG ;:IS THERE A TERMINAL?
037232 100002 BPL 1$ ;:BR IF YES
037234 000000 HALT ;:HALT HERE IF NO TERMINAL
037236 000430 BR 3$ ;:LEAVE
037240 010046 1$: MOV RO,-(SP) ;:SAVE RO
037242 017600 000002 MOV @2(SP),RO ;:GET ADDRESS OF ASCIZ STRING
037246 122767 000001 141770 CMPB #APTENV,$ENV ;:RUNNING IN APT MODE
037254 001011 BNE 62$ ;:NO,GO CHECK FOR APT CONSOLE
037256 132767 000100 141761 BITB #APTSPool,$ENV ;:SPOOL MESSAGE TO APT
037264 001405 BEQ 62$ ;:NO,GO CHECK FOR CONSOLE
037266 010067 000004 MOV RO,61$ ;:SETUP MESSAGE ADDRESS FOR APT
037272 004767 000270 JSR PC,$ATY3 ;:SPOOL MESSAGE TO APT
037276 000000 61$: .WORD 0 ;:MESSAGE ADDRESS
037300 132767 000040 141737 62$: BITB #APTCSUP,$ENV ;:APT CONSOLE SUPPRESSED
037306 001003 BNE 60$ ;:YES,SKIP TYPE OUT
037310 112046 2$: MOVB (RO)+,-(SP) ;:PUSH CHARACTER TO BE TYPED ONTO STACK
037312 001005 BNE 4$ ;:BR IF IT ISN'T THE TERMINATOR
037314 005726 TST (SP)+ ;:IF TERMINATOR POP IT OFF THE STACK
037316 012600 60$: MOV (SP)+,RO ;:RESTORE RO
037320 062716 000002 3$: ADD #2,(SP) ;:ADJUST RETURN PC
037324 000002 RTI ;:RETURN
037326 122716 000011 4$: CMPB #HT,(SP) ;:BRANCH IF <HT>
037332 001430 BEQ 8$ ;:BRANCH IF NOT <CRLF>
037334 122716 000200 CMPB #CRLF,(SP) ;:BRANCH IF NOT <CRLF>
037340 001006 BNE 5$ ;:POP <CR><LF> EQUIV
037342 005726 TST (SP)+ ;:TYPE A CR AND LF
037344 104401 TYPE ;:TYPE A CR AND LF
037346 001221 $CRLF ;:TYPE A CR AND LF
037350 105067 000200 CLRB $CHARCNT ;:CLEAR CHARACTER COUNT
037354 000755 BR 2$ ;:GET NEXT CHARACTER
037356 004767 000056 5$: JSR PC,$TYPEC ;:GO TYPE THIS CHARACTER
037362 126726 141570 6$: CMPB $FILLC,(SP)+ ;:IS IT TIME FOR FILLER CHARS.?
037366 001350 BNE 2$ ;:IF NO GO GET NEXT CHAR.
037370 016746 141560 MOV $NULL,-(SP) ;:GET # OF FILLER CHARS. NEEDED
;:AND THE NULL CHAR.
037374 105366 000001 7$: DECB 1(SP) ;:DOES A NULL NEED TO BE TYPED?
037400 002770 BLT 6$ ;:BR IF NO--GO POP THE NULL OFF OF STACK
037402 004767 000032 JSR PC,$TYPEC ;:GO TYPE A NULL
037406 105367 000142 DECB $CHARCNT ;:DO NOT COUNT AS A COUNT
037412 000770 BR 7$ ;:LOOP
;HORIZONTAL TAB PROCESSOR
037414 112716 000040 8$: MOVB #' ,(SP) ;:REPLACE TAB WITH SPACE

```



```

037420 004767 000014          9$: JSR PC,$TYPEC      ;;TYPE A SPACE
037424 132767 000007 000122 BITB #7,$CHARCNT    ;;BRANCH IF NOT AT
037432 001372          BNC 9$              ;;TAB STOP
037434 005726          TST (SP)+          ;;POP SPACE OFF STACK
037436 000724          BR 2$              ;;GET NEXT CHARACTER
037440 105777 141504 $TYPEC: TSTB @STPS          ;;WAIT UNTIL PRINTER IS READY
037444 100375          BPL $TYPEC
037446 116677 000002 141476 MOVB 2(SP),@STPB    ;;LOAD CHAR TO BE TYPED INTO DATA REG.
037454 105777 141464 TSTB @STKS          ;;SEE IF KEYBOARD IS TALKING.
037460 100021          BPL 2$              ;;BRANCH IF IT ISN'T.
037462 017746 141460 MOV @STKB,-(SP)    ;;PUSH CHARACTER ONTO STACK.
037466 042716 177600 BIC #177600,(SP)  ;;BIT CLEAR TOP BYTE AND PARITY BIT.
037472 022726 000023 CMP #23,(SP)+      ;;SEE IF THIS IS A ^S.
037476 001012          BNE 2$              ;;BRANCH TO CONTINUE IF IT ISN'T.
037500 105777 141440 3$: TSTB @STKS          ;;WAIT FOR ANOTHER INPUT.
037504 100375          BPL 3$              ;;BRANCH BACK IF NOT READY.
037506 017746 141434 MOV @STKB,-(SP)    ;;PUSH NEXT CHARACTER ON STACK.
037512 042716 177600 BIC #177600,(SP)  ;;BIT CLEAR TOP BYTE AND PARITY BIT.
037516 022726 000021 CMP #21,(SP)+      ;;SEE IF THIS IS A ^Q.
037522 001366          BNE 3$              ;;BRANCH BACK FOR MORE WAIT IF NOT.
037524 122766 000015 000002 2$: CMPB #CR,2(SP)      ;;IS CHARACTER A CARRIAGE RETURN?
037532 001003          BNE 1$              ;;BRANCH IF NO
037534 105067 000014 CLRB $CHARCNT     ;;YES--CLEAR CHARACTER COUNT
037540 000406          BR $TYPEX          ;;EXIT
037542 122766 000012 000002 1$: CMPB #LF,2(SP)      ;;IS CHARACTER A LINE FEED?
037550 001402          BEQ $TYPEX          ;;BRANCH IF YES
037552 105227          INCB (PC)+          ;;COUNT THE CHARACTER
037554 000000          $CHARCNT: .WORD 0          ;;CHARACTER COUNT STORAGE
037556 000207          $TYPEX: RTS PC

```

2940

.SBTTL APT COMMUNICATIONS ROUTINE

```

*****
037560 112767 000001 000236 $ATY1: MOVB #1,$FFLG    ;;TO REPORT FATAL ERROR
037566 112767 000001 000226 $ATY3: MOVB #1,$MFLG    ;;TO TYPE A MESSAGE
037574 000403          BR $ATYC
037576 112767 000001 000220 $ATY4: MOVB #1,$FFLG    ;;TO ONLY REPORT FATAL ERROR
037604          $ATYC:
037604 010046          MOV R0,-(SP)          ;;PUSH R0 ON STACK
037606 010146          MOV R1,-(SP)          ;;PUSH R1 ON STACK
037610 105767 000206 TSTB $MFLG          ;;SHOULD TYPE A MESSAGE?
037614 001450          BEQ 5$              ;;IF NOT: BR
037616 122767 000001 141420 CMPB #APTENV,$ENV  ;;OPERATING UNDER APT?
037624 001031          BNE 3$              ;;IF NOT: BR
037626 132767 000100 141411 BITB #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
037634 001425          BEQ 3$              ;;IF NOT: BR
037636 017600 000004 MOV @4(SP),R0      ;;GET MESSAGE ADDR.
037642 062766 000002 000004 ADD #2,4(SP)       ;;BUMP RETURN ADDR.
037650 005767 141350 1$: TST $MSGTYPE          ;;SEE IF DONE W/ LAST XMISSION?
037654 001375          BNE 1$              ;;IF NOT: WAIT
037656 010067 141356 MOV R0,$MSGAD      ;;PUT ADDR IN MAILBOX
037662 105720          2$: TSTB (R0)+          ;;FIND END OF MESSAGE
037664 001376          BNE 2$              ;;SUB START OF MESSAGE
037666 166700 141346 SUB $MSGAD,R0      ;;GET MESSAGE LNTH IN WORDS
037672 006200          ASR R0              ;;PUT LENGTH IN MAILBOX
037674 010067 141342 MOV R0,$MSGGLT    ;;TELL APT TO TAKE MSG.
037700 012767 000004 141316 MOV #4,$MSGTYPE
037706 000413          BR 5$
037710 017667 000004 000016 3$: MOV @4(SP),4$      ;;PUT MSG ADDR IN JSR LINKAGE

```

```

037716 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS
037724 016746 140046 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
037730 004767 177272 JSR PC,$TYPE ;;CALL TYPE MACRO
037734 000000 4$: .WORD 0
037736 5$:
037736 105767 000062 10$: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?
037742 001416 BEQ 12$ ;;IF NOT: BR
037744 005767 141274 TST $ENV ;;RUNNING UNDER APT?
037750 001413 BEQ 12$ ;;IF NOT: BR
037752 005767 141246 11$: TST $MSGTYPE ;;FINISHED LAST MESSAGE?
037756 001375 BNE 11$ ;;IF NOT: WAIT
037760 017667 000004 141240 MOV @4(SP),$FATAL ;;GET ERROR #
037766 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
037774 005267 141224 INC $MSGTYPE ;;TELL APT TO TAKE ERROR
040000 105067 000020 12$: CLRB $FFLG ;;CLEAR FATAL FLAG
040004 105067 000013 CLRB $LFLG ;;CLEAR LOG FLAG
040010 105067 000006 CLRB $MFLG ;;CLEAR MESSAGE FLAG
040014 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
040016 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
040020 000207 RTS PC ;;RETURN
040022 000 $MFLG: .BYTE 0 ;;MESSG. FLAG
040023 000 $LFLG: .BYTE 0 ;;LOG FLAG
040024 000 $FFLG: .BYTE 0 ;;FATAL FLAG
                .EVEN

```

000200  
000001  
000100  
000040

2941

```

APTSIZE=200
APTENV=001
APTSPOOL=100
APTCSUP=040
.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
$TYPBN: MOV    R1,-(SP)    ;;SAVE R1 ON THE STACK
        MOV    6(SP),R1   ;;GET THE INPUT NUMBER
        SEC
1$:     MOVB   #'0,$BIN    ;;SET 'C' SO CAN KEEP TRACK OF THE NUMBER OF BITS
        ROL   R1          ;;GET THIS BIT
        BEQ   2$          ;;DONE?
        ADCB  $BIN        ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
        TYPE  ,$BIN       ;;GO TYPE THIS BIT
        CLC
        BR   1$          ;;CLEAR 'C' SO CAN KEEP TRACK OF BITS
        BR   1$          ;;GO DO THE NEXT BIT
2$:     MOV    (SP)+,R1    ;;POP THE STACK INTO R1
        MOV    2(SP),4(SP) ;;ADJUST THE STACK
        MOV    (SP)+,(SP)
        RTI              ;;RETURN TO USER
$BIN:   .BYTE  0,0       ;;STORAGE FOR ASCII CHAR. AND TERMINATOR

```

2942

```

.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
        ;*
040102 017646 000000      $TYPOS: MOV      @ (SP),-(SP)      ;;PICKUP THE MODE
040106 116667 000001 000211  MOVB     1(SP), $OFILL      ;;LOAD ZERO FILL SWITCH
040114 112667 000207      MOVB     (SP)+, $OMODE+1    ;;NUMBER OF DIGITS TO TYPE
040120 062716 000002      ADD      #2, (SP)        ;;ADJUST RETURN ADDRESS
040124 000406      BR      $TYPON
040126 112767 000001 000171 $TYPOC: MOVB     #1, $OFILL      ;;SET THE ZERO FILL SWITCH
040134 112767 000006 000165  MOVB     #6, $OMODE+1    ;;SET FOR SIX(6) DIGITS
040142 112767 000005 000154 $TYPON: MOVB     #5, $OCNT      ;;SET THE ITERATION COUNT
040150 010346      MOV      R3,-(SP)        ;;SAVE R3
040152 010446      MOV      R4,-(SP)        ;;SAVE R4
040154 010546      MCV      R5,-(SP)        ;;SAVE R5
040156 116704 000145      MOVB     $OMODE+1, R4    ;;GET THE NUMBER OF DIGITS TO TYPE
040162 005404      NEG      R4
040164 062704 000006      ADD      #6, R4          ;;SUBTRACT IT FOR MAX. ALLOWED
040170 110467 000132      MOVB     R4, $OMODE      ;;SAVE IT FOR USE
040174 116704 000125      MOVB     $OFILL, R4      ;;GET THE ZERO FILL SWITCH
040200 016605 000012      MOV      12(SP), R5     ;;PICKUP THE INPUT NUMBER
040204 005003      CLR      R3            ;;CLEAR THE OUTPUT WORD
040206 006105      1$:    ROL      R5          ;;ROTATE MSB INTO 'C'
040210 000404      BR      3$            ;;GO DC MSB
040212 006105      2$:    ROL      R5          ;;FORM THIS DIGIT
040214 006105      ROL      R5
040216 006105      ROL      R5
040220 010503      MOV      R5, R3
040222 006103      3$:    ROL      R3          ;;GET LSB OF THIS DIGIT
040224 105367 000076      DECB     $OMODE          ;;TYPE THIS DIGIT?
040230 100016      BPL      7$            ;;BR IF NO
040232 042703 177770      BIC      #177770, R3     ;;GET RID OF JUNK
040236 001002      BNE      4$            ;;TEST FOR 0
040240 005704      TST      R4            ;;SUPPRESS THIS 0?
040242 001403      BEQ      5$            ;;BR IF YES
040244 005204      4$:    INC      R4          ;;DON'T SUPPRESS ANYMORE 0'S
040246 052703 000060      BIS      #'0, R3        ;;MAKE THIS DIGIT ASCII
040252 052703 000040      5$:    BIS      #' , R3        ;;MAKE ASCII IF NOT ALREADY
040256 110367 000040      MOVB     R3, 8$        ;;SAVE FOR TYPING
040262 104401 040322      TYPE     , 8$          ;;GO TYPE THIS DIGIT
040266 105367 000032      7$:    DECB     $OCNT      ;;COUNT BY 1
040272 003347      BGT      2$            ;;BR IF MORE TO DO
040274 002402      BLT      6$            ;;BR IF DONE
040276 005204      INC      R4            ;;INSURE LAST DIGIT ISN'T A BLANK
040300 000744      BR      2$            ;;GO DO THE LAST DIGIT
    
```

```

040302 012605      6$:  MOV      (SP)+,R5      ;;RESTORE R5
040304 012604      MOV      (SP)+,R4      ;;RESTORE R4
040306 012603      MOV      (SP)+,R3      ;;RESTORE R3
040310 016666 000002 000004  MOV      2(SP),4(SP)    ;;SET THE STACK FOR RETURNING
040316 012616      MOV      (SP)+,(SP)
040320 000002      RTI                          ;;RETURN
040322      000      8$:  .BYTE 0          ;;STORAGE FOR ASCII DIGIT
040323      000      .BYTE 0          ;;TERMINATOR FOR TYPE ROUTINE
040324      000      $OCNT: .BYTE 0        ;;OCTAL DIGIT COUNTER
040325      000      $OFILL: .BYTE 0       ;;ZERO FILL SWITCH
2943 040326 000000  $OMODE: .WORD 0         ;;NUMBER OF DIGITS TO TYPE
                                .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
                                ;*****
                                ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
                                ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
                                ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
                                ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
                                ;*REPLACED WITH SPACES.
                                ;*CALL:
                                ;*  MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
                                ;*  TYPDS      ;;GO TO THE ROUTINE
040330      $TYPDS:
040330 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
040332 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
040334 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
040336 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
040340 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
040342 012746 020200      MOV      #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
040346 016605 000020      MOV      20(SP),R5      ;;GET THE INPUT NUMBER
040352 100004      BPL      1$          ;;BR IF INPUT IS POS.
040354 005405      NEG      R5          ;;MAKE THE BINARY NUMBER POS.
040356 112766 000055 000001  MOVB     #'-,1(SP)     ;;MAKE THE ASCII NUMBER NEG.
040364 005000      1$:  CLR      R0          ;;ZERO THE CONSTANTS INDEX
040366 012703 040544      MOV      #SDBLK,R3     ;;SETUP THE OUTPUT POINTER
040372 112723 000040      MOVB     #' ,(R3)+    ;;SET THE FIRST CHARACTER TO A BLANK
040376 005002      2$:  CLR      R2          ;;CLEAR THE BCD NUMBER
040400 016001 040534      MOV      $DTBL(R0),R1  ;;GET THE CONSTANT
040404 160105      3$:  SUB      R1,R5      ;;FORM THIS BCD DIGIT
040406 002402      BLT     4$          ;;BR IF DONE
040410 005202      INC     R2          ;;INCREASE THE BCD DIGIT BY 1
040412 000774      BR      3$
040414 060105      4$:  ADD      R1,R5      ;;ADD BACK THE CONSTANT
040416 005702      TST     R2          ;;CHECK IF BCD DIGIT 0
040420 001002      BNE     5$          ;;FALL THROUGH IF 0
040422 105716      TSTB    (SP)        ;;STILL DOING LEADING 0'S?
040424 100407      BMI     7$          ;;BR IF YES
040426 106316      5$:  ASLB    (SP)        ;;MSD?
040430 103003      BCC     6$          ;;BR IF NO
040432 116663 000001 177777  MOVB     1(SP),-1(R3)  ;;YES--SET THE SIGN
040440 052702 000060      6$:  BIS      #'0,R2     ;;MAKE THE BCD DIGIT ASCII!
040444 052702 000040      7$:  BIS      #' ,R2     ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
040450 110223      MOVB    R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
040452 005720      TST     (R0)+      ;;JUST INCREMENTING
040454 020027 000010      CMP     R0,#10     ;;CHECK THE TABLE INDEX
040460 002746      BLT     2$          ;;GO DO THE NEXT DIGIT
040462 003002      BGT     8$          ;;GO TO EXIT
040464 010502      MOV     R5,R2      ;;GET THE LSD
    
```

2944

```

040466 000764          BR      6$          ;;GO CHANGE TO ASCII
040470 105726          8$:  TSTB   (SP)+        ;;WAS THE LSD THE FIRST NON-ZERO?
040472 100003          BPL    9$          ;;BR IF NO
040474 116663 177777 177776  MOVB  -1(SP),-2(R3)  ;;YES--SET THE SIGN FOR TYPING
040502 105013          9$:  CLRB   (R3)          ;;SET THE TERMINATOR
040504 012605          MOV   (SP)+,R5       ;;POP STACK INTO R5
040506 012603          MOV   (SP)+,R3       ;;POP STACK INTO R3
040510 012602          MOV   (SP)+,R2       ;;POP STACK INTO R2
040512 012601          MOV   (SP)+,R1       ;;POP STACK INTO R1
040514 012600          MOV   (SP)+,R0       ;;POP STACK INTO R0
040516 104401 040544  TYPE  $DBLK          ;;NOW TYPE THE NUMBER
040522 016666 000002 000004  MOV   2(SP),4(SP)    ;;ADJUST THE STACK
040530 012616          MOV   (SP)+,(SP)
040532 000002          RTI                    ;;RETURN TO USER
040534 023420          $DTBL: 10000.
040536 001750          1000.
040540 000144          100.
040542 000012          10.
040544          $DBLK: .BLKW 4
                $BTTL  SAVE AND RESTORE R0-R5 ROUTINES
                ;*****
                ;*SAVE R0-R5
                ;*CALL:
                ;*   SAVREG
                ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
                ;*
                ;*TOP---(+16)
                ;* +2---(+18)
                ;* +4---R5
                ;* +6---R4
                ;* +8---R3
                ;*+10---R2
                ;*+12---R1
                ;*+14---R0
                $SAVREG:
040554          MOV   R0,-(SP)          ;;PUSH R0 ON STACK
040556          MOV   R1,-(SP)          ;;PUSH R1 ON STACK
040560          MOV   R2,-(SP)          ;;PUSH R2 ON STACK
040562          MOV   R3,-(SP)          ;;PUSH R3 ON STACK
040564          MOV   R4,-(SP)          ;;PUSH R4 ON STACK
040566          MOV   R5,-(SP)          ;;PUSH R5 ON STACK
040570          MOV   22(SP),-(SP)      ;;SAVE PS OF MAIN FLOW
040574          MOV   22(SP),-(SP)      ;;SAVE PC OF MAIN FLOW
040600          MOV   22(SP),-(SP)      ;;SAVE PS OF CALL
040604          MOV   22(SP),-(SP)      ;;SAVE PC OF CALL
040610          RTI
                ;*RESTORE R0-R5
                ;*CALL:
                ;*   RESREG
                $RESREG:
040612          MOV   (SP)+,22(SP)      ;;RESTORE PC OF CALL
040616          MOV   (SP)+,22(SP)      ;;RESTORE PS OF CALL
040622          MOV   (SP)+,22(SP)      ;;RESTORE PC OF MAIN FLOW
040626          MOV   (SP)+,22(SP)      ;;RESTORE PS OF MAIN FLOW
040632          MOV   (SP)+,R5          ;;POP STACK INTO R5
040634          MOV   (SP)+,R4          ;;POP STACK INTO R4
040636          MOV   (SP)+,R3          ;;POP STACK INTO R3
    
```

040640 012602  
 040642 012601  
 040644 012600  
 040646 000002  
 2945

```

MOV (SP)+,R2 ;;POP STACK INTO R2
MOV (SP)+,R1 ;;POP STACK INTO R1
MOV (SP)+,R0 ;;POP STACK INTO R0
RTI

.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
* RETURN ;; THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK
$DB20: SAVREG ;; SAVE ALL REGISTERS
MOV 2(SP),R1 ;; PICKUP THE POINTER TO LOW WORD
MOV #SOCTVL+13.,R5 ;; POINTER TO DATA TABLE
MOV #12.,R4 ;; DO ELEVEN CHARACTERS
MOV #^C7,R3 ;; MASK
MOV (R1)+,R0 ;; LOWER WORD
MOV (R1)+,R1 ;; HIGH WORD
CLR R2 ;; TERMINATOR
1$: MOV B R2,-(R5) ;; PUT CHARACTER IN DATA TABLE
MOV R0,R2 ;; GET THIS DIGIT
DEC R4 ;; COUNT THIS CHARACTER
BGT 3$ ;; BR IF NOT THE LAST DIGIT
BEQ 2$ ;; BR IF IT IS THE LAST DIGIT
INC R5 ;; ALL DIGITS DONE-ADJUST POINTER FOR FIRST
MOV R5,2(SP) ;; ASCII CHAR. & PUT IT ON THE STACK
RESREG ;; RESTORE ALL REGISTERS
RTS PC ;; RETURN TO USER
2$: ASR R3 ;; POSITION THE MASK FOR THE LAST DIGIT
3$: ROR R1 ;; POSITION THE BINARY NUMBER FOR
ROR R0 ;; THE NEXT OCTAL DIGIT
ROR R1
ROR R0
ROR R1
ROR R0
BIC R3,R2 ;; MASK OUT ALL JUNK
ADD #'0,R2 ;; MAKE THIS CHAR. ASCII
BR 1$ ;; GO PUT IT IN THE DATA TABLE
$SOCTVL: .BLKB 14. ;; RESERVE DATA TABLE
    
```

040650 104413  
 040652 016601 000002  
 040656 012705 040767  
 040662 012704 000014  
 040666 012703 177770  
 040672 012100  
 040674 012101  
 040676 005002  
 040700 110245  
 040702 010002  
 040704 005304  
 040706 003007  
 040710 001405  
 040712 005205  
 040714 010566 000002  
 040720 104414  
 040722 000207  
 040724 006203  
 040726 006001  
 040730 006000  
 040732 006001  
 040734 006000  
 040736 006001  
 040740 006000  
 040742 040302  
 040744 062702 000060  
 040750 000753  
 040752

2947

040770 010046  
040772 016600 000002  
040776 005740  
041000 111000  
041002 006300  
041004 016000 041024  
041010 000200  
  
041012 011646  
041014 016666 000004 000002  
041022 000002

```
.SBTTL TRAP DECODER
:*****
:THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
:AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
:OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
:GO TO THAT ROUTINE.
$TRAP: MOV R0,-(SP) ;;SAVE R0
MOV 2(SP),R0 ;;GET TRAP ADDRESS
TST -(R0) ;;BACKUP BY 2
MOVB (R0),R0 ;;GET RIGHT BYTE OF TRAP
ASL R0 ;;POSITION FOR INDEXING
MOV $TRPAD(R0),R0 ;;INDEX TO TABLE
RTS R0 ;;GO TO ROUTINE
;;THIS IS USE TO HANDLE THE 'GETPRI' MACRO
$TRAP2: MOV (SP),-(SP) ;;MOVE THE PC DOWN
MOV 4(SP),2(SP) ;;MOVE THE PSW DOWN
RTI ;;RESTORE THE PSW
```

2948

041024 041012  
041026 037226  
041030 040126  
041032 040102  
041034 040142  
041036 040330  
041040 040026  
041042 036202  
041044 036132  
041046 036454  
041050 036574  
041052 040554  
041054 040612

```
.SBTTL TRAP TABLE
:THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
:BY THE 'TRAP' INSTRUCTION.
: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
```

041056 012737 041234 000024  
041064 012737 000340 000026  
041072 010046  
041074 010146  
041076 010246  
041100 010346  
041102 010446  
041104 010546  
041106 017746 140026  
041112 010667 000122  
041116 012737 041130 000024  
041124 000000  
041126 000776

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

041130 012737 041234 000024  
041136 016706 000076  
041142 005067 000072  
041146 005267 000066

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

```

041152 001375          BNE      1$          ;;OF WORD
041154 012677 137760  MOV      (SP)+,@SWR    ;;POP STACK INTO @SWR
041160 012605          MOV      (SP)+,R5     ;;POP STACK INTO R5
041162 012604          MOV      (SP)+,R4     ;;POP STACK INTO R4
041164 012603          MOV      (SP)+,R3     ;;POP STACK INTO R3
041166 012602          MOV      (SP)+,R2     ;;POP STACK INTO R2
041170 012601          MOV      (SP)+,R1     ;;POP STACK INTO R1
041172 012600          MOV      (SP)+,R0     ;;POP STACK INTO R0
041174 012737 041056 000024 MOV      #SPWRDN,@PWRVEC ;;SET UP THE POWER DOWN VECTOR
041202 012737 000340 000026 MOVL     #340,@PWRVEC+2 ;;PRIO:7
041210 104401          TYPE          ;;REPORT THE POWER FAILURE
041212 041242          $PWRMG: .WORD PWRMSG ;;POWER FAIL MESSAGE POINTER
041214 012716          MOV      (PC)+,(SP) ;;RESTART AT START
041216 020000          $PWRAD: .WORD START ;;RESTART ADDRESS
041220 042766 000020 000002 BIC      #20,2(SP)    ;;CLEAR 'T' BIT
041226 005067 140114          CLR      $TBIT      ;;CLEAR THE 'T' BIT FLAG
041232 000002          RT!
041234 000000          $ILLUP: HALT      ;;THE POWER UP SEQUENCE WAS STARTED
041236 000776          BR      .-2        ;; BEFORE THE POWER DOWN WAS COMPLETE
041240 000000          $SAVR6: 0          ;;PUT THE SP HERE
2949 041242 012 015 040 PWRMSG: .ASCIIZ <12><15>? POWER FAILURE - RESTARTING ?<12><15>
041245 120 117 127
041250 105 122 040
041253 106 101 111
041256 114 125 122
041261 105 040 055
041264 040 122 105
041267 123 124 101
041272 122 124 111
041275 116 107 040
041300 012 015 000

2950          .EVEN
2951
2952
    
```



```

2954          .SBTTL  ERROR MESSAGES, DATA HEADERS-TABLES & FORMATS
2955          .NLIST  BEX
2956 041304    125    116    105  EM1:  .ASCIZ  /UNEXPECTED CPU TRAP TO LOC. 004/
2957 041344    125    116    105  EM2:  .ASCIZ  /UNEXPECTED MEM. MGMT. TRAP TO LOC. 250/
2958 041413    120    122    111  EM3:  .ASCIZ  /PRIORITY BITS SET WRONG IN PSW/
2959 041452    115    117    104  EM4:  .ASCIZ  /MODE BITS SET WRONG IN PSW/
2960 041505    104    125    101  EM5:  .ASCIZ  /DUAL ADDRESSING BETWEEN HIBLO BYTES OF PSW/
2961 041560    113    105    122  EM6:  .ASCIZ  /KERNEL R6 CHANGED BY WRITING SUPERVISOR OR USER R6/
2962 041643    101    040    115  EM7:  .ASCIZ  /A MEMORY MGMT. REG. TIMED OUT/
2963 041701    123    125    115  EM10: .ASCIZ  /SUMMARY OF MEM. MGMT. REG. TIMEOUTS/
2964 041745    115    105    115  EM11: .ASCIZ  /MEM. MGMT. REG. WOULD NOT CLEAR/
2965 042005    115    105    115  EM12: .ASCIZ  /MEM. MGMT. REG. BITS NOT SET CORRECTLY/
2966 042054    123    122    060  EM13: .ASCIZ  /SRO EFFECTED BY WRITE TO PSW/
2967 042111    115    115    122  EM14: .ASCIZ  /MMR1 DID NOT TRACK PROPERLY/
2968 042145    115    115    122  EM15: .ASCIZ  /MMR3 IS HOLDING THE WRONG DATA/
2969 042204    104    125    101  EM16: .ASCIZ  /DUAL ADDRESSING BETWEEN PAR-PDR GROUPS/
2970 042253    102    101    104  EM17: .ASCIZ  /BAD RELOCATION, ON STORING DATA 18-BIT MAPPING/
2971 042331    120    110    131  EM20: .ASCIZ  /PHYS. ADDR. FORMED WRONG IN RELOCATE MODE/
2972 042403    123    122    062  EM21: .ASCIZ  /SR2 NOT TRACKING CORRECTLY/
2973 042436    127    122    117  EM22: .ASCIZ  /WRONG PDR'S REFERENCED WHILE IN RELOCATE MODE/
2974 042514    120    101    122  EM23: .ASCIZ  /PAR OR PDR CHANGED BY A RESET/
2975 042552    115    101    111  EM24: .ASCIZ  /MAINT. MODE (SRO <8>) NOT DISABLED BY A RESET/
2976 042630    104    101    124  EM25: .ASCIZ  /DATA INCORRECT AFTER A MAINT. MODE WRITE/
2977 042701    123    117    125  EM26: .ASCIZ  /SOURCE RELOCATED IN MAINT. MODE/
2978 042741    123    122    062  EM27: .ASCIZ  /SR2 DID NOT LOCK UP CORRECT VIRTUAL ADDRESS/
2979 043015    106    117    114  EM30: .ASCIZ  /FOLLOWING PAR-PDR WILL NOT ZERO/
2980 043055    123    125    115  EM31: .ASCIZ  /SUMMARY OF DUAL ADDRESSING ERRORS/
2981 043117    123    125    115  EM32: .ASCIZ  /SUMMARY OF COUNT PATTERN FAILURES/
2982 043161    105    122    122  EM33: .ASCIZ  /ERROR IN BYTE ADDRESSING OF PAR-PDR/
2983 043225    124    110    105  EM34: .ASCIZ  /THE FOLLOWING ARE DUAL ADDRESSING ERRORS FOR PAR'S-PDR'S/
2984 043316    124    110    105  EM36: .ASCIZ  /THE FOLLOWING ARE COUNT PATTERN ERRORS FOR PAR'S-PDR'S/
2985 043405    111    114    114  EM40: .ASCIZ  /ILLEGAL (MODE 10) STACK POINTER NOT MAPPED TO USER/
2986 043470    061    070    055  EM41: .ASCIZ  /18-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM/
2987 043551    106    101    125  EM42: .ASCIZ  /FAULTY CARRY PROPAGATION 18-BIT MAPPING/
2988 043621    061    070    055  EM43: .ASCIZ  /18-BIT MAPPING POSSIBLE HOLE AT TOP OF MEMORY/
2989 043677    116    117    040  EM44: .ASCIZ  /NO TRAP THRU ERRVEC, AT 18-BIT ADDR. 760000/
2990 043752    104    111    104  EM45: .ASCIZ  /DIDN'T GET WRAP AROUND TO ADDRESS ZERO/
2991 044021    116    117    040  EM46: .ASCIZ  /NO TRAP THRU ERRVEC, ON NON-EXISTANT ADDR./
2992 044074    120    122    105  EM47: .ASCIZ  /PREMATURE END OF MEMORY FOUND/
2993 044132    062    062    055  EM50: .ASCIZ  /22-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM/
2994 044213    102    101    104  EM51: .ASCIZ  /BAD RELOCATION, CARRY PROPAGATION 22-BIT MAPPING/
2995 044273    062    062    055  EM52: .ASCIZ  /22-BIT MAPPING POSSIBLE HOLE AT TOP OF MEMORY/
2996 044351    104    111    104  EM53: .ASCIZ  /DID NOT GET UNIBUS ADDRESS/
2997 044404    127    055    102  EM54: .ASCIZ  /W-BIT DID NOT GET SET IN PDR/
2998 044441    127    055    102  EM55: .ASCIZ  /W-BIT SET IN MORE THAN ONE PDR/
2999 044500    127    055    102  EM56: .ASCIZ  /W-BIT NOT CLEARED BY WRITING TO PDR/
3000 044544    127    055    102  EM57: .ASCIZ  /W-BIT GOT SET DURING ODD ADDR. ABORT/
3001 044611    116    117    040  EM60: .ASCIZ  /NO APT SPECIAL HARDWARE FOUND/
3002
3003 044647    117    114    104  DH1:  .ASCIZ  /OLD PC OLD PSW R6 WAS CPUERR TESTNO ERRORPC/
3004 044727    117    114    104  DH2:  .ASCIZ  /OLD PC OLD PSW R6 WAS SRO SR1 SR2 TESTNO ERRORPC/
3005 045026    127    122    117  DH3:  .ASCIZ  /WROTE READ TESTNO ERRORPC/
3006 045066    101    104    104  DH7:  .ASCIZ  /ADDRESS TESTNO ERRORPC/
3007 045116    122    105    107  DH10: .ASCII  /REGISTER-ADDRS NUM OF/<CRLF>
3008 045146    101    116    104  .ASCIZ  /AND-ED OR-ED TIMOUTS TESTNO ERRORPC/
3009 045216    122    105    107  DH11: .ASCII  /REGISTR READ READ-(BINARY)/<CRLF>
3010 045254    101    104    104  .ASCIZ  /ADDRESS (OCTAL) 5432109876543210 TESTNO ERRORPC/

```



```

3068 050136 001316 001320 001306 DT31: .WORD ADDROR,ADRAND,DATAOR,DATAND,TESTNO,ERRCNT,0
3069 050154 001316 001320 001312 DT32: .WORD ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,TESTNO,ERRCNT,0
3070 050176 001162 001166 001164 DT33: .WORD $REG0,$REG2,$REG1,TESTNO,$ERRPC,0
3071 050212 001162 001164 001254 DT35: .WORD $REG0,$REG1,TESTNO,0
3072 050222 001162 001166 001172 DT37: .WORD $REG0,$REG2,$REG4,$REG1,TESTNO,0
3073 050236 001200 001202 001254 DT41: .WORD $TMP1,$TMP2,TESTNO,$ERRPC,0
3074 050250 001166 001170 001162 DT42: .WORD $REG2,$REG3,$REG0,TESTNO,$ERRPC,0
3075 050264 001200 001254 001116 DT43: .WORD $TMP1,TESTNO,$ERRPC,0
3076 050274 001164 001254 001116 DT45: .WORD $REG1,TESTNO,$ERRPC,0
3077 050304 172350 003022 001254 DT47: .WORD KIPAR4,$LSTBK,TESTNO,$ERRPC,0
3078 050316 001174 001170 001254 DT54: .WORD $REG5,$REG3,TESTNO,$ERRPC,0
3079 050330 001162 001174 001170 DT55: .WORD $REG0,$REG5,$REG3,TESTNO,$ERRPC,0
3080 050344 001174 001254 001116 DT56: .WORD $REG5,TESTNO,$ERRPC,0
3081 050354 001116 000000 DT60: .WORD $ERRPC,0
3082
3083 050360 000 000 000 DF1: .BYTE 0,0,0,0,0
3084 050365 000 000 000 DF2: .BYTE 0,0,0,0,0,0,0,0
3085 050375 000 000 000 DF3: .BYTE 0,0,0,0
3086 050401 000 000 000 DF7: .BYTE 0,0,0
3087 050404 000 000 001 DF10: .BYTE 0,0,1,0,0
3088 050411 000 000 002 DF11: .BYTE 0,0,2,0,0
3089 050416 000 000 000 DF12: .BYTE 0,0,0,2,0,0
3090 050424 000 000 000 DF16: .BYTE 0,0,0,0,0,0
3091 050432 003 000 000 DF20: .BYTE 3,0,0,0,0,0,0,0
3092 050442 000 000 000 DF24: .BYTE 0,0
3093 050444 000 000 000 DF32: .BYTE 0,0,0,0,0,1
3094 050452 000 000 000 DF33: .BYTE 0,0,0,0,0,0,0,1
3095 050462 004 000 000 DF43: .BYTE 4,0,0
3096 .LIST BEX
3097 .EVEN
3098
3099 000001 .END

```

ABASE = 000000	BIT00 = 000001	DH15 = 045566	DT47 = 050304	ERRVEC = 000004
ACDW1 = 000000	BIT01 = 000002	DH16 = 045626	DT54 = 050316	FORMPA = 003220
ACDW2 = 000000	BIT02 = 000004	DH17 = 045727	DT55 = 050330	GTSWR = 104407
ACPUOP = 000000	BIT03 = 000010	DH2 = 044727	DT56 = 050344	HDWFLA = 001340
ADJROR = 001316	BIT04 = 000020	DH20 = 046010	DT60 = 050354	HOLFLG = 001336
ADDW0 = 000000	BIT05 = 000040	DH21 = 046136	DT7 = 047660	HT = 000011
ADDW1 = 000000	BIT06 = 000100	DH22 = 046176	DUALAD = 002334	IOTVEC = 000020
ADDW10 = 000000	BIT07 = 000200	DH24 = 046314	EMTVEC = 000030	KDPAR0 = 172360
ADDW11 = 000000	BIT08 = 000400	DH3 = 045026	EM1 = 041304	KDPAR1 = 172362
ADDW12 = 000000	BIT09 = 001000	DH30 = 046334	EM10 = 041701	KDPAR2 = 172364
ADDW13 = 000000	BIT10 = 002000	DH31 = 046374	EM11 = 041745	KDPAR3 = 172366
ADDW14 = 000000	BIT11 = 004000	DH32 = 046513	EM12 = 042005	KDPAR4 = 172370
ADDW15 = 000000	BIT12 = 010000	DH33 = 046613	EM13 = 042054	KDPAR5 = 172372
ADDW2 = 000000	BIT13 = 020000	DH34 = 046663	EM14 = 042111	KDPAR6 = 172374
ADDW3 = 000000	BIT14 = 040000	DH36 = 046703	EM15 = 042145	KDPAR7 = 172376
ADDW4 = 000000	BIT15 = 100000	DH40 = 046752	EM16 = 042204	KDPDR0 = 172320
ADDW5 = 000000	BIT2 = 000004	DH41 = 047010	EM17 = 042253	KDPDR1 = 172322
ADDW6 = 000000	BIT3 = 000010	DH42 = 047051	EM2 = 041344	KDPDR2 = 172324
ADDW7 = 000000	BIT4 = 000020	DH43 = 047153	EM20 = 042331	KDPDR3 = 172326
ADDW8 = 000000	BIT5 = 000040	DH45 = 047203	EM21 = 042403	KDPDR4 = 172330
ADDW9 = 000000	BIT6 = 000100	DH46 = 047233	EM22 = 042436	KDPDR5 = 172332
ADEVCT = 000000	BIT7 = 000200	DH47 = 047262	EM23 = 042514	KDPDR6 = 172334
ADEVM = 000000	BIT8 = 000400	DH54 = 047326	EM24 = 042552	KDPDR7 = 172336
ADRAND = 001320	BIT9 = 001000	DH55 = 047406	EM25 = 042630	KERSTK = 001100
AENV = 000000	BPTVEC = 000014	DH56 = 047506	EM26 = 042701	KIPAR0 = 172340
AENVM = 000000	CKSWR = 104410	DH57 = 047536	EM27 = 042741	KIPAR1 = 172342
AFATAL = 000000	CLEANU = 002275	DH60 = 047576	EM3 = 041413	KIPAR2 = 172344
AMADR1 = 000000	CLRREG = 002260	DH7 = 045066	EM30 = 043015	KIPAR3 = 172346
AMADR2 = 000000	CMSG = 037175	DISPLA = 001142	EM31 = 043055	KIPAR4 = 172350
AMADR3 = 000000	CNTRLC = 037116	DISPRE = 000174	EM32 = 043117	KIPAR5 = 172352
AMADR4 = 000000	CPUERR = 177766	DSWR = 177570	EM33 = 043161	KIPAR6 = 172354
AMAMS1 = 000000	CPUEXP = 001334	DT1 = 047606	EM34 = 043225	KIPAR7 = 172356
AMAMS2 = 000000	CR = 000015	DT10 = 047670	EM36 = 043316	KIPDR0 = 172300
AMAMS3 = 000000	CRLF = 000200	DT11 = 047704	EM4 = 041452	KIPDR1 = 172302
AMAMS4 = 000000	DATAAND = 001310	DT12 = 047720	EM40 = 043405	KIPDR2 = 172304
AMSGAD = 000000	DATAOR = 001306	DT14 = 047736	EM41 = 043470	KIPDR3 = 172306
AMSGLG = 000000	DDISP = 177570	DT15 = 047750	EM42 = 043551	KIPDR4 = 172310
AMSGTY = 000000	DF1 = 050360	DT16 = 047762	EM43 = 043621	KIPDR5 = 172312
AMTYP1 = 000000	DF10 = 050404	DT17 = 047776	EM44 = 043677	KIPDR6 = 172314
AMTYP2 = 000000	DF11 = 050411	DT2 = 047624	EM45 = 043752	KIPDR7 = 172316
AMTYP3 = 000000	DF12 = 050416	DT20 = 050014	EM46 = 044021	KSP = 000006
AMTYP4 = 000000	DF16 = 050424	DT21 = 050036	EM47 = 044074	LF = 000012
APASS = 000000	DF2 = 050365	DT22 = 050050	EM5 = 041505	LOOP = 020456
APRIOR = 000000	DF20 = 050432	DT24 = 050072	EM50 = 044132	MGMERR = 003122
APTCSU = 000040	DF24 = 050442	DT25 = 050100	EM51 = 044213	MGMFLG = 003124
APTENV = 000001	DF3 = 050375	DT27 = 050112	EM52 = 044273	MMRO = 177572
APTSIZ = 000200	DF32 = 050444	DT3 = 047646	EM53 = 044351	MMR1 = 177574
APTSPO = 000100	DF33 = 050452	DT30 = 050124	EM54 = 044404	MMR2 = 177576
APTTST = 031372	DF43 = 050462	DT31 = 050136	EM55 = 044441	MMR3 = 172516
ASWREG = 000000	DF7 = 050401	DT32 = 050154	EM56 = 044500	MMVEC = 000250
ATESTN = 000000	DH1 = 044647	DT33 = 050176	EM57 = 044544	OLDPC = 001326
AUNIT = 000000	DH10 = 045116	DT35 = 050212	FM6 = 041560	OLDPS = 001330
AUSWR = 000000	DH11 = 045216	DT37 = 050222	EM60 = 044611	PARCOU = 002400
AVECT1 = 000000	DH12 = 045336	DT41 = 050236	EM7 = 041643	PARTAB = 001356
AVECT2 = 000000	DH13 = 045476	DT42 = 050250	ERRCNT = 001322	PATAND = 001314
BADPC = 001324	DH14 = 045526	DT43 = 050264	ERROR = 104000	PATTOR = 001312
BITO = 000001		DT45 = 050274	ERTYP = 035624	PBAHI = 001304

PBALO 001302	SIPDR7= 172216	TST13 022266	UDPDR2= 177624	SDB20 040650
PCPUER 001332	SR0 = 177572	TST14 022420	UDPDR3= 177626	SDEVCT 001234
PIRQ = 177772	SR1 = 177574	TST15 022562	UDPDR4= 177630	SDOAGN 035126
PIRQVE= 000240	SR2 = 177576	TST16 022622	UDPDR5= 177632	SDTBL 040534
PRO = 000000	SR3 = 172516	TST17 023226	UDPDR6= 177634	SENDAD 035116
PR1 = 000040	SSP =%000006	TST2 020644	UDPDR7= 177636	SENDCT 034744
PR2 = 000100	STACK = 001100	TST20 023350	UIPAR0= 177640	SENULL 035170
PR3 = 000140	START 020000	TST21 023546	UIPAR1= 177642	SENV 001244
PR4 = 000200	STKLMT= 177774	TST22 023744	UIPAR2= 177644	SENVN 001245
PR5 = 000240	SUPSTK= 000700	TST23 024142	UIPAR3= 177646	SEOP 034714
PR6 = 000300	SWR 001140	TST24 024340	UIPAR4= 177650	SEOPCT 034736
PR7 = 000340	SWREG 000176	TST25 024536	UIPAR5= 177652	SERFLG 001103
PS = 177776	SW0 = 000001	TST26 024734	UIPAR6= 177654	SERMAX 001115
PSW = 177776	SW00 = 000001	TST27 025046	UIPAR7= 177656	SERROR 035366
PWRMSG 041242	SW01 = 000002	TST3 020730	UIPDR0= 177600	SERRPC 001116
PWRVEC= 000024	SW02 = 000004	TST30 025160	UIPDR1= 177602	SERRTB 001372
RDCHR = 104411	SW03 = 000010	TST31 025272	UIPDR2= 177604	SERTTL 001112
RDLIN = 104412	SW04 = 000020	TST32 025410	UIPDR3= 177606	SESCAP 001212
READON 001342	SW05 = 000040	TST33 025526	UIPDR4= 177610	SETABL 001244
RESREG= 104414	SW06 = 000100	TST34 025644	UIPDR5= 177612	SETEND 001254
RESVEC= 000010	SW07 = 000200	TST35 025750	UIPDR6= 177614	SFATAL 001226
RMIREG= 177770	SW08 = 000400	TST36 026054	UIPDR7= 177616	SFFLG 040024
R6 =%000006	SW09 = 001000	TST37 026160	USESTK= 000600	SFILLC 001156
R7 =%000007	SW1 = 000002	TST4 021052	USP =%000006	SFILLS 001155
SAVREG= 104413	SW10 = 002000	TST40 026264	VIRT1 001276	SGDADR 001120
SCOPE = 000004	SW11 = 004000	TST41 026370	VIRT2 001300	SGDDAT 001124
SDPAR0= 172260	SW12 = 010000	TST42 026474	WASR6 001256	SGET42 035070
SDPAR1= 172262	SW13 = 020000	TST43 026570	WASSR0 001264	SGTSWR 036202
SDPAR2= 172264	SW14 = 040000	TST44 027116	WASSR1 001266	SHD = 000000
SDPAR3= 172266	SW15 = 100000	TST45 027252	WASSR2 001270	SHIBTS 000204
SDPAR4= 172270	SW2 = 000004	TST46 027510	WBIT = 000100	SICNT 001104
SDPAR5= 172272	SW3 = 000010	TST47 031452	SAPTHD 000204	SILLUP 041234
SDPAR6= 172274	SW4 = 000020	TST5 021212	SATYC 037604	SINTAG 001135
SDPAR7= 172276	SW5 = 000040	TST50 032312	SATY1 037560	SITEMB 001114
SDPDR0= 172220	SW6 = 000100	TST51 033174	SATY3 037566	SKTNEX 002670
SDPDR1= 172222	SW7 = 000200	TST52 034010	SATY4 037576	SKTOUT 002660
SDPDR2= 172224	SW8 = 000400	TST53 034204	SAUTOB 001134	SKT11 062512
SDPDR3= 172226	SW9 = 001000	TST54 034412	SBADR 001122	SLF 001222
SDPDR4= 172230	TBIT = 000020	TST55 034620	SBDDAT 001126	SLFLG 040023
SDPDR5= 172232	TBITPS 001272	TST6 021364	SBELL 001214	SLOOP 035164
SDPDR6= 172234	TBITVE= 000014	TST7 021516	SBIN 040100	SLPADR 001106
SDPDR7= 172236	TESTNO 001254	TYPBN = 104406	SCHARC 037554	SLPERR 001110
SIPAR0= 172240	TIMERR 003024	TYPDS = 104405	SCKSWR 036132	SLSTAD 003020
SIPAR1= 172242	TIMFLG 003026	TYPE = 104401	SCLR.T 035106	SLSTBK 003022
SIPAR2= 172244	TKVEC = 000060	TYPOC = 104402	SCMTAG 001100	SMAIL 001224
SIPAR3= 172246	TOFF 002172	TYPON = 104404	SCM1 = 000006	SMBADR 000206
SIPAR4= 172250	TON 002226	TYPOS = 104403	SCM2 = 000014	SMFLG 040022
SIPAR5= 172252	TONUM 001274	UDPAR0= 177660	SCM3 = 000006	SMNEW 037105
SIPAR6= 172254	TPVEC = 000064	UDPAR1= 177662	SCM4 = 000006	SMSGAD 001240
SIPAR7= 172256	TRAPPC 001260	UDPAR2= 177664	SCNTLC 001350	SMSGLG 001242
SIPDR0= 172200	TRAPPS 001262	UDPAR3= 177666	SCNTLG 037067	SMSGTY 001224
SIPDR1= 172202	TRAPVE= 000034	UDPAR4= 177670	SCNTLU 037062	SMSWR 037074
SIPDR2= 172204	TRTVEC= 000014	UDPAR5= 177672	SCORE 002676	SMXCNT 001344
SIPDR3= 172206	TST1 020566	UDPAR6= 177674	SCPUOP 001252	SNULL 001154
SIPDR4= 172210	TST10 021650	UDPAR7= 177676	SCRFL 001221	SNWTST= 000001
SIPDR5= 172212	TST11 022002	UDPDR0= 177620	SCROUT 002726	SOCNT 040324
SIPDR6= 172214	TST12 022134	UDPDR1= 177622	SDBLK 040544	SOCTL 040752

SOMODE 040326	\$REG1 001164	\$STUP = 177777	\$TMP4 001206	\$TYPDS 040330
SOVER 035352	\$REG2 001166	\$SVLAD 035316	\$TMP5 001210	\$TYPE 037226
SPASS 001232	\$REG3 001170	\$SVPC = 000204	\$TN = 000056	\$TYPEC 037440
SPASTM 000212	\$REG4 001172	\$SWR = 173400	\$TPB 001152	\$TYPEX 037556
SPWRAD 041216	\$REG5 001174	\$SWREG 001246	\$TPFLG 001157	\$TYPOC 040126
SPWRDN 041056	\$RESRE 040612	\$SWRMK= 000000	\$TPS 001150	\$TYPON 040142
SPWRMG 041212	\$RTNAD 035166	\$TBIT 001346	\$TRAP 040770	\$TYPOS 040102
SPWRUP 041130	\$RTRN 035162	\$TESTN 001230	\$TRAP2 041012	\$UNIT 001236
SQUES 001220	\$SAVRE 040554	\$TKB 001146	\$TRP = 000015	\$UNITM 000214
\$RDCHR 036454	\$SAVR6 041240	\$TKS 001144	\$TRPAD 041024	\$USWR 001250
\$RDLIN 036574	\$SCOPE 035174	\$TMP0 001176	\$TSTM 000210	\$XTSTR 035206
\$RDSZ = 000010	\$SETUP= 000137	\$TMP1 001200	\$TSTM 001102	\$GET4= 000001
\$REGAD 001160	\$SIZE 002454	\$TMP2 001202	\$TYIN 037052	\$OFILL 040325
\$REGO 001162	\$SIZEX 002732	\$TMP3 001204	\$TYPBN 040026	.\$X - 000204

. ABS. 050466 000  
000000 001

ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 59296 WORDS ( 232 PAGES)  
DYNAMIC MEMORY: 20434 WORDS ( 78 PAGES)  
ELAPSED TIME: 00:07:09  
CKKTA00.BIN,CKKTA00.SEQ/CRF-CKKTA00.MLB/ML,CKKTA00.P11

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES								
ABASE	=	000000	10-884								
ACDW1	=	000000	10-884								
ACDW2	=	000000	10-884								
ACPUOP	=	000000	10-884	10-884							
ADDROR		001316	#10-884	*12-1255	*12-1274	*12-1297	*18-1609	*18-1630	*18-1642	*18-1642	*18-1648
			*18-1648	*18-1654	*18-1654	*18-1660	*18-1660	*18-1666	*18-1666	*18-1672	*18-1672
			36-3054	36-3068	36-3069						
ADDW0	-	000000	10-884								
ADDW1	-	000000	10-884								
ADDW10	=	000000	10-884								
ADDW11	=	000000	10-884								
ADDW12	=	000000	10-884								
ADDW13	=	000000	10-884								
ADDW14	=	000000	10-884								
ADDW15	-	000000	10-884								
ADDW2	=	000000	10-884								
ADDW3	=	000000	10-884								
ADDW4	=	000000	10-884								
ADDW5	=	000000	10-884								
ADDW6	-	000000	10-884								
ADDW7	=	000000	10-884								
ADDW8	-	000000	10-884								
ADDW9	-	000000	10-884								
ADEVCT	-	000000	10-884	10-884							
ADEVM		000000	10-884								
ADRAND		001320	#10-884	*12-1259	12-1275	12-1298	*18-1608	*18-1632	*18-1642	*18-1642	*18-1648
			*18-1648	*18-1654	*18-1654	*18-1660	*18-1660	*18-1666	*18-1666	*18-1672	*18-1672
			36-3054	36-3068	36-3069						
AENV		000000	10-884	10-884							
AENVM	-	000000	10-884	10-884							
AFATAL	-	000000	10-884	10-884							
AMADR1	-	000000	10-884								
AMADR2	-	000000	10-884								
AMADR3	=	000000	10-884								
AMADR4	=	000000	10-884								
AMAMS1	=	000000	10-884								
AMAMS2	=	000000	10-884								
AMAMS3	=	000000	10-884								
AMAMS4	=	000000	10-884								
AMSGAD	-	000000	10-884	10-884							
AMSGLG	=	000000	10-884	10-884							
AMSGTY	=	000000	10-884	10-884							
AMTYP1	=	000000	10-884								
AMTYP2	=	000000	10-884								
AMTYP3	=	000000	10-884								
AMTYP4	=	000000	10-884								
APASS	=	000000	10-884	10-884							
APRIOR	=	000000	10-884								
APTCSU	=	000040	34-2939	#34-2940							
APTENV		000001	34-2834	34-2939	34-2940	#34-2940					
APTSIZ	-	000200	13-1424	#34-2940							
APTSPO	-	000100	34-2939	34-2940	#34-2940						





SYMBOL	CROSS REFERENCE	VALUE	REFERENCES							
DATA0D		001310	#10-884	*12-1258	12-1277	12-1302	36-3068	36-3069		
DATA0R		001306	#10-884	*12-1254	*12-1276	*12-1301	36-3068	36-3069		
DDISP	=	177570	#9-807	10-884	13-1424					
DF1		050360	11-890	11-972	11-1067	11-1169	11-1187	#36-3083		
DF10		050404	11-933	#36-3087						
DF11		050411	11-940	11-1005	#36-3088					
DF12		050416	11-947	#36-3089						
DF16		050424	#36-3090							
DF2		050365	11-896	11-978	11-1023	11-1092	11-1099	11-1129	11-1135	11-1142 11-1148
			11-1155	#36-3084						
DF20		050432	11-985	11-998	#36-3091					
DF24		050442	11-1011	11-1111	#36-3092					
DF3		050375	11-902	11-908	11-914	11-920	11-959	11-965	11-991	11-1017 11-1029
			11-1035	11-1042	11-1162	11-1181	#36-3085			
DF32		050444	11-1048	#36-3093						
DF33		050452	11-1054	#36-3094						
DF43		050462	11-1105	#36-3095						
DF7		050401	11-926	11-953	11-1079	11-1086	11-1117	11-1123	11-1175	#36-3086
DH1		044647	11-888	#36-3003						
DH10		045116	11-930	#36-3007						
DH11		045216	11-937	11-1002	#36-3009					
DH12		045336	11-944	#36-3011						
DH13		045476	11-951	#36-3013						
DH14		045526	11-957	#36-3014						
DH15		045566	11-963	#36-3015						
DH16		045626	11-969	#36-3016						
DH17		045727	11-976	#36-3018						
DH2		044727	11-894	11-1021	#36-3004					
DH20		046010	11-982	#36-3019						
DH21		046136	11-989	11-1027	#36-3021					
DH22		046176	11-995	#36-3022						
DH24		046314	11-1009	11-1109	#36-3024					
DH3		045026	11-900	11-906	11-912	11-918	11-1015	#36-3005		
DH30		046334	11-1033	#36-3025						
DH31		046374	11-1039	#36-3026						
DH32		046513	11-1046	#36-3028						
DH33		046613	11-1052	#36-3029						
DH34		046663	11-1058	#36-3030						
DH36		046703	11-1071	#36-3031						
DH40		046752	11-1083	#36-3032						
DH41		047010	11-1090	11-1133	#36-3034					
DH42		047051	11-1096	11-1139	11-1152	#36-3035				
DH43		047153	11-1103	11-1146	#36-3037					
DH45		047203	11-1115	#36-3038						
DH46		047233	11-1121	#36-3039						
DH47		047262	11-1127	#36-3040						
DH54		047326	11-1159	#36-3041						
DH55		047406	11-1166	#36-3043						
DH56		047506	11-1173	#36-3045						
DH57		047536	11-1179	#36-3046						
DH60		047576	11-1185	#36-3047						
DH7		045066	11-924	#36-3006						

SYMBOL	VALUE	REFERENCES					
DISPLA	001142	#10-884	*13-1424	*13-1424	34-2833	34-2834	
DISPRE	000174	#9-881	13-1424				
DSWR	= 177570	#9-807	10-884	13-1424			
DT1	047606	11-889	#36-3050				
DT10	047670	11-932	#36-3054				
DT11	047704	11-939	11-1004	#36-3055			
DT12	047720	11-946	#36-3056				
DT14	047736	11-958	#36-3057				
DT15	047750	11-964	11-1180	#36-3058			
DT16	047762	11-971	#36-3059				
DT17	047776	11-977	#36-3060				
DT2	047624	11-895	11-1022	#36-3051			
DT20	050014	11-984	#36-3061				
DT21	050036	11-990	#36-3062				
DT22	050050	11-997	#36-3063				
DT24	050072	11-1010	11-1110	#36-3064			
DT25	050100	11-1016	#36-3065				
DT27	050112	11-1028	#36-3066				
DT3	047646	11-901	11-907	11-913	11-919	#36-3052	
DT30	050124	11-1034	#36-3067				
DT31	050136	11-1041	#36-3068				
DT32	050154	11-1047	#36-3069				
DT33	050176	11-1053	#36-3070				
DT35	050212	11-1066	#36-3071				
DT37	050222	11-1078	#36-3072				
DT41	050236	11-1091	11-1134	#36-3073			
DT42	050250	11-1098	11-1141	11-1154	#36-3074		
DT43	050264	11-1104	11-1147	#36-3075			
DT45	050274	11-1116	#36-3076				
DT47	050304	11-1128	#36-3077				
DT54	050316	11-1161	#36-3078				
DT55	050330	11-1168	#36-3079				
DT56	050344	11-1174	#36-3080				
DT60	050354	11-1186	#36-3081				
DT7	047660	11-925	11-952	11-1085	11-1122	#36-3053	
DUALAD	002334	#12-1272	21-1913	21-1918	22-1924	23-1930	24-1936 25-1942
EMTVEC	= 000030	#9-807	13-1424	13-1424			
EM1	041304	11-887	#36-2956				
EM10	041701	11-929	#36-2963				
EM11	041745	11-936	#36-2964				
EM12	042005	11-943	#36-2965				
EM13	042054	11-950	#36-2966				
EM14	042111	11-956	#36-2967				
EM15	042145	11-962	#36-2968				
EM16	042204	11-968	#36-2969				
EM17	042253	11-975	#36-2970				
EM2	041344	11-893	#36-2957				
EM20	042331	11-981	#36-2971				
EM21	042403	11-988	#36-2972				
EM22	042436	11-994	#36-2973				
EM23	042514	11-1001	#36-2974				
EM24	042552	11-1008	#36-2975				



SYMBOL	CROSS REFERENCE VALUE	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES
FORMPA	003220	12-1310	12-1310	13-1424	13-1424	13-1424	*13-1429	*13-1430	*30-2344	*30-2374
GNS	= *****	*30-2425	*30-2517	*30-2563	*33-2816	*33-2826	34-2833	34-2833	34-2833	34-2833
		#12-1394	30-2700	31-2760						
		9-881	9-881	13-1425	34-2832	34-2832	35-2947	35-2947	35-2947	35-2947
		35-2947	35-2947	35-2947	35-2947	35-2947	35-2947	35-2947	35-2947	35-2947
		35-2947	35-2947	35-2947	35-2947	35-2947	35-2947	35-2947	35-2947	35-2947
		35-2947	35-2947	35-2947	35-2947	35-2947	35-2947	35-2947	35-2947	35-2947
GTSWR	= 104407	13-1425	34-2933	#35-2947						
HDWFLA	001340	#10-884	*30-2337	*30-2349	30-2369	30-2381	30-2422	30-2513	30-2523	
HOLFLG	001336	#10-884	*30-2373	30-2388	*30-2393	30-2409	30-2482	*30-2485	*30-2490	*30-2518
		30-2531	*30-2536	30-2554	30-2622	*30-2625	*30-2630			
HT	= 000011	#9-807	34-2939	34-2939						
IOTVEC	= 000020	#9-807	13-1424	13-1424						
KDPA0	= 172360	#9-808								
KDPA1	= 172362	#9-808								
KDPA2	= 172364	#9-808								
KDPA3	= 172366	#9-808								
KDPA4	= 172370	#9-808								
KDPA5	= 172372	#9-808								
KDPA6	= 172374	#9-808								
KDPA7	= 172376	#9-808	21-1913	21-1913	25-1960	25-1972				
KDPDR0	= 172320	#9-808								
KDPDR1	= 172322	#9-808								
KDPDR2	= 172324	#9-808								
KDPDR3	= 172326	#9-808								
KDPDR4	= 172330	#9-808								
KDPDR5	= 172332	#9-808								
KDPDR6	= 172334	#9-808								
KDPDR7	= 172336	#9-808	23-1930	23-1930	25-1978					
KERSTK	= 001100	#9-820	17-1553	17-1559	17-1561	29-2202	33-2818			
KIPAR0	= 172340	#9-808	12-1310	12-1394	18-1642	21-1913	21-1913	21-1913	21-1913	21-1913
		25-1960	25-1972	26-2005	*26-2005	26-2005	26-2005	26-2005	27-2078	28-2107
		30-2288	*30-2305							
KIPAR1	= 172342	#9-808	*30-2306							
KIPAR2	= 172344	#9-808	*30-2307							
KIPAR3	= 172346	#9-808	*29-2225	*30-2308	32-2782					
KIPAR4	= 172350	#9-808	*29-2226	30-2310	30-2311	30-2312	30-2313	30-2317	30-2318	30-2319
		30-2320	30-2321	30-2322	30-2323	30-2324	30-2325	30-2326	30-2327	30-2328
		*30-2375	*30-2404	*30-2427	*30-2445	*30-2456	*30-2457	30-2474	*30-2519	*30-2547
		30-2550	*30-2564	*30-2584	*30-2595	*30-2596	30-2614	*30-2661	*30-2720	*30-2741
		36-3060	36-3077							
KIPAR5	= 172352	#9-808	*30-2376	30-2383	30-2390	*30-2405	30-2407	30-2484	*30-2520	30-2525
		30-2533	*30-2548	30-2552	*30-2565	30-2624	*30-2662	*30-2721	*30-2742	
KIPAR6	= 172354	#9-808								
KIPAR7	= 172356	#9-808	12-1310	*29-2227	*30-2309					
KIPDR0	= 172300	#9-808	18-1648	23-1930	23-1930	23-1930	23-1930	23-1930	25-1978	26-2020
		*26-2020	26-2020	26-2020	26-2020	27-2077	27-2095	29-2183	*29-2191	30-2290
		30-2302	32-2782	32-2782	32-2782					
KIPDR1	= 172302	#9-808								
KIPDR2	= 172304	#9-808								
KIPDR3	= 172306	#9-808	*29-2228	*29-2246	*33-2815	33-2819				
KIPDR4	= 172310	#9-808	*29-2229	*30-2718	*30-2737					

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES
KIPDR5	= 172312	#9-808 *30-2719 *30-2738
KIPDR6	= 172314	#9-808
KIPDR7	= 172316	#9-808 *29-2230
KSP	= %000006	#9-815 *12-1239 *12-1243 *12-1273 *12-1296 *12-1331 *12-1332 12-1334 *12-1347
		*12-1343 *12-1364 *12-1365 12-1366 *12-1373 *12-1374 *13-1428 *17-1553 17-1559
		17-1562 *17-1569 *18-1624 *18-1642 *18-1648 *18-1654 *18-1660 *18-1666 *18-1672
		*29-2202 *30-2469 *30-2470 *30-2492 *30-2493 *30-2609 *30-2610 *30-2632 *30-2633
		*31-2754 *31-2755 *31-2769 *31-2770 *33-2818 *34-2837 *34-2861 *34-2886 34-2888
		*34-2889 *34-2893 *34-2894 *34-2900 34-2902 *34-2903 *34-2906 *34-2907 *34-2913
		*34-2914
LF	= 000012	#9-807 34-2939 34-2939
LOOP	020456	#13-1427 34-2832
MGMERR	003122	12-1310 #12-1356 13-1431 29-2204 29-2245 30-2747
MGMFLG	003124	#12-1357 *12-1372 *13-1435
MMRO	= 177572	#9-811 *14-1438 17-1551 19-1772 19-1785 19-1793 19-1795 19-1805 19-1816
		19-1831 19-1844 19-1854 30-2310 30-2311 30-2312 30-2313 30-2317 30-2318
		30-2319 30-2320 30-2321 30-2322 30-2323 30-2324 30-2325 30-2326 30-2327
		30-2328 30-2368
MMR1	= 177574	#9-812 19-1769
MMR2	= 177576	#9-813
MMR3	= 172516	#9-814 *14-1439 20-1865 *30-2512 *30-2562 *30-2604
MMVEC	= 000200	#9-808 12-1310 12-1310 *13-1431 *13-1432 29-2187 *29-2187 *29-2188 *29-2204
		*29-2205 *29-2231 *29-2245 *30-2690 *30-2747
OLDPC	001326	#10-884 *12-1273 12-1283 *12-1296 12-1308 *30-2469 *30-2476 30-2487 *30-2489
		30-2493 *30-2609 *30-2616 30-2627 *30-2629 30-2633
OLDPS	001330	#10-884 *30-2470 30-2492 *30-2610 30-2632
PARCOU	002400	#12-1295 25-1960 25-1966 25-1972 25-1978 26-1985 26-1991
PARTAB	001356	#10-884 27-2050 27-2056 27-2059
PATAND	001314	#10-884 *12-1260 12-1300 36-3069
PATTOR	001312	#10-884 *12-1256 *12-1299 36-3069
PBAHI	001304	#10-884 *12-1417
PBALO	001302	#10-884 *12-1408 *12-1409 *12-1414 36-3061 36-3063
PCPUER	001332	#10-884 *12-1333 12-1337 *30-2380 30-2386 *30-2430 30-2434 *30-2455 30-2459
		*30-2471 30-2472 30-2480 *30-2527 30-2529 *30-2594 30-2598 *30-2611 30-2612
		30-2620
PIRQ	= 177772	#9-807
PIRQVE	= 000240	#9-807
PRO	= 000000	#9-807
PR1	= 000040	#9-807
PR2	= 000100	#9-807
PR3	= 000140	#9-807
PR4	= 000200	#9-807
PR5	= 000240	#9-807
PR6	= 000300	#9-807
PR7	= 000340	#9-807
PS	177776	#9-807 9-807
PSW	177776	#9-807 12-1201 12-1203 12-1395 12-1399 16-1464 16-1465 *16-1486 *16-1487
		16-1488 *16-1492 *16-1500 *16-1511 *16-1513 16-1514 *16-1519 *16-1525 *16-1527
		16-1528 *16-1532 *17-1552 *17-1554 *17-1556 *17-1558 *17-1568 *17-1570 *17-1572
		*17-1574 *18-1603 *18-1743 18-1745 *18-1753 19-1829 19-1842 19-1853 *30-2658
		30-2691 30-2712 *30-2722 30-2724 *30-2734 *30-2736 *32-2788 *32-2790 *32-2796
		*32-2798

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES							
PWRMSG		041242	35-2948	#35-2949						
PWRVEC	=	000024	#9-807	13-1424	13-1424	35-2948	35-2948	35-2948	35-2948	35-2948
RDCHR	=	104411	34-2920	#35-2947						
RDLIN	=	104412	#35-2947							
READON		001342	#10-884	19-1806	19-1818					
RESREG	=	104414	34-2945	#35-2947						
RESVEC	=	000010	#9-807	13-1424	13-1424	13-1424				
RMIREG	=	177770	#9-824	30-2345	*30-2398	*30-2424	*30-2541			
R6	=	%000006	#9-807	*13-1424	*13-1424	13-1424				
R7	=	%000007	#9-807							
SAVREG	=	104413	34-2945	#35-2947						
SCOPE	=	000004	#9-807	16-1460	16-1483	16-1509	17-1550	18-1601	18-1641	18-1647
			18-1659	18-1665	18-1671	18-1686	18-1741	19-1768	20-1864	21-1912
			22-1923	23-1929	24-1935	25-1941	25-1959	25-1965	25-1971	25-1977
			26-1990	26-2004	26-2009	26-2014	26-2019	26-2024	26-2029	27-2045
			29-2181	29-2222	30-2284	30-2365	30-2509	30-2656	32-2781	32-2787
			33-2810	34-2832						18-1653
SDPAR0	=	172260	#9-808							
SDPAR1	=	172262	#9-808							
SDPAR2	=	172264	#9-808							
SDPAR3	=	172266	#9-808							
SDPAR4	=	172270	#9-808							
SDPAR5	=	172272	#9-808							
SDPAR6	=	172274	#9-808							
SDPAR7	=	172276	#9-808	21-1918	21-1918	25-1966				
SDPDR0	=	172220	#9-808							
SDPDR1	=	172222	#9-808							
SDPDR2	=	172224	#9-808							
SDPDR3	=	172226	#9-808							
SDPDR4	=	172230	#9-808							
SDPDR5	=	172232	#9-808							
SDPDR6	=	172234	#9-808							
SDPDR7	=	172236	#9-808	24-1936	24-1936	26-1985				
SIPARO	=	172240	#9-808	12-1401	18-1654	21-1918	21-1918	21-1918	21-1918	25-1966
			26-2010	*26-2010	26-2010	26-2010	26-2010	27-2084	28-2131	30-2292
SIPAR1	=	172242	#9-808							30-2664
SIPAR2	=	172244	#9-808							
SIPAR3	=	172246	#9-808	32-2789						
SIPAR4	=	172250	#9-808	*30-2686	*30-2726	*30-2745				
SIPAR5	=	172252	#9-808	*30-2687	*30-2727	*30-2746				
SIPAR6	=	172254	#9-808							
SIPAR7	=	172256	#9-808							
SIPDR0	=	172200	#9-808	18-1660	24-1936	24-1936	24-1936	24-1936	24-1936	26-1985
			*26-2025	26-2025	26-2025	26-2025	27-2083	28-2119	30-2294	30-2674
			32-2789	32-2789						26-2025
SIPDR1	=	172202	#9-808							
SIPDR2	=	172204	#9-808							
SIPDR3	=	172206	#9-808							
SIPDR4	=	172210	#9-808	*30-2682	*30-2728					
SIPDR5	=	172212	#9-808	*30-2683	*30-2729					
SIPDR6	=	172214	#9-808							
SIPDR7	=	172216	#9-808							

SYMBOL	CROSS REFERENCE	SYMBOL	VALUE	REFERENCES
SR0	= 177572	#9-808	9-811	12-1310
		*18-1744	18-1746	*18-1752
		29-2253	*29-2258	*30-2688
SR1	= 177574	#9-808	9-812	12-1310
SR2	= 177576	#9-808	9-813	12-1368
SR3	= 172516	#9-808	9-814	12-1369
SSP	= %000006	#9-816	*17-1555	*17-1571
STACK	= J01100	#9-807	9-820	9-821
START	= 020000	9-881	#13-1424	35-2948
STKMT	= 177774	#9-807		
SUPSTK	= 000700	#9-821	17-1555	
SWR	= 001140	#10-884	13-1424	*13-1424
		34-2833	34-2833	34-2834
		35-2948	35-2948	34-2834
SWREG	= 000176	#9-881	13-1424	13-1425
SW0	= 000001	#9-807		34-2920
SW00	= 000001	#9-807	9-807	
SW01	= 000002	#9-807	9-807	
SW02	= 000004	#9-807	9-807	
SW03	= 000010	#9-807	9-807	
SW04	= 000020	#9-807	9-807	
SW05	= 000040	#9-807	9-807	
SW06	= 000100	#9-807	9-807	
SW07	= 000200	#9-807	9-807	
SW08	= 000400	#9-807	9-807	
SW09	= 001000	#9-807	9-807	
SW1	= 000002	#9-807		
SW10	= 002000	#9-807		
SW11	= 004000	#9-807		
SW12	= 010000	#9-807		
SW13	= 020000	#9-807		
SW14	= 040000	#9-807		
SW15	= 100000	#9-807		
SW2	= 000004	#9-807		
SW3	= 000010	#9-807		
SW4	= 000020	#9-807		
SW5	= 000040	#9-807		
SW6	= 000100	#9-807		
SW7	= 000200	#9-807		
SW8	= 000400	#9-807		
SW9	= 001000	#9-807		
TBIT	= 000020	#9-818	12-1201	12-1206
TBITPS	= 001272	#10-884	*12-1204	12-1219
TBITVE	= 000014	#9-807	13-1424	13-1424
TESTNO	= 001254	#10-884	*34-2834	36-3050
		36-3057	36-3058	36-3059
		36-3066	36-3067	36-3068
		36-3075	36-3076	36-3077
TIMERR	= 003024	#12-1323	13-1429	18-1621
		30-2425	30-2563	33-2826
TIMFLG	= 003026	#12-1324	*12-1341	*13-1434
TKVEC	= 000060	#9-807		

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
TOFF		002172	#12-1201 29-2182 29-2223 32-2782 32-2789 32-2797 33-2812
TON		002226	#12-1219 29-2207 29-2248 32-2782 32-2789 32-2797 33-2828
TONUM		001274	#10-884 *18-1610 18-1618 *18-1633 *18-1642 18-1642 *18-1642 *18-1648 18-1648 *18-1648 *18-1654 18-1654 *18-1654 *18-1660 18-1660 *18-1660 *18-1666 18-1666 *18-1666 *18-1672 18-1672 *18-1672 36-3054
TPVEC	=	000064	#9-807
TRAPPC		001260	#10-884 *12-1331 12-1343 *12-1364 12-1374 *29-2256 29-2264 *31-2754 31-2770 36-3050 36-3051
TRAPPS		001262	#10-884 *12-1332 12-1342 *12-1365 12-1373 *29-2257 29-2263 *31-2755 31-2769 36-3050 36-3051
TRAPVE	=	000034	#9-807 13-1424 13-1424
TRIVEC	=	000014	#9-807
TST1		020566	#16-1460
TST10		021650	18-1648 #18-1653
TST11		022002	18-1654 #18-1659
TST12		022134	18-1660 #18-1665
TST13		022266	18-1666 #18-1671
TST14		022420	18-1672 #18-1686
TST15		022562	#18-1741
TST16		022622	#19-1768
TST17		023226	#20-1864
TST2		020644	#16-1483
TST20		023350	#21-1912
TST21		023546	21-1913 #21-1917
TST22		023744	21-1918 #22-1923
TST23		024142	22-1924 #23-1929
TST24		024340	23-1930 #24-1935
TST25		024536	24-1936 #25-1941
TST26		024734	25-1942 #25-1959
TST27		025046	25-1960 #25-1965
TST3		020730	#16-1509
TST30		025160	25-1966 #25-1971
TST31		025272	25-1972 #25-1977
TST32		025410	25-1978 #26-1984
TST33		025526	26-1985 #26-1990
TST34		025644	26-1991 #26-2004
TST35		025750	#26-2009
TST36		026054	#26-2014
TST37		026160	#26-2019
TST4		021052	#17-1550
TST40		026264	#26-2024
TST41		026370	#26-2029
TST42		026474	#27-2045
TST43		026570	#27-2073
TST44		027116	#29-2181
TST45		027252	#29-2222
TST46		027510	29-2249 #30-2284
TST47		031452	30-2340 30-2343 30-2348 #30-2365
TST5		021212	17-1576 #18-1601
TST50		032312	30-2465 #30-2509
TST51		033174	30-2605 #30-2656
TST52		034010	30-2749 #32-2781



SYMBOL	CROSS REFERENCE	VALUE	REFERENCES							
TST53		034204	#32-2787							
TST54		034412	#32-2795							
TST55		034620	#33-2810							
TST6		021364	18-1622	#18-1641						
TST7		021516	18-1642	#18-1647						
TYPBN	=	104406	34-2881	#35-2947						
TYPDS	=	104405	34-2832	34-2832	34-2875	34-2932	#35-2947			
TYPE		104401	13-1425	34-2832	34-2832	34-2832	34-2834	34-2834	34-2836	34-2853
			34-2858	34-2860	34-2890	34-2904	34-2910	34-2915	34-2920	34-2920
			34-2920	34-2920	34-2920	34-2920	34-2920	34-2920	34-2920	34-2920
			34-2920	34-2920	34-2920	34-2920	34-2920	34-2920	34-2920	34-2920
			34-2942	34-2943	#35-2947	35-2948				
			34-2844	34-2869	34-2920	34-2929	#35-2947			
TYPOC	=	104402								
TYPON	=	104404	#35-2947							
TYPOS	=	104403	#35-2947							
UDPAR0	=	177660	#9-808							
UDPAR1	=	177662	#9-808							
UDPAR2	=	177664	#9-808							
UDPAR3	=	177666	#9-808							
UDPAR4	=	177670	#9-808							
UDPAR5	=	177672	#9-808							
UDPAR6	=	177674	#9-808							
UDPAR7	=	177676	#9-808	22-1924	22-1924					
UDPDR0	=	177620	#9-808							
UDPDR1	=	177622	#9-808							
UDPDR2	=	177624	#9-808							
UDPDR3	=	177626	#9-808							
UDPDR4	=	177630	#9-808							
UDPDR5	=	177632	#9-808							
UDPDR6	=	177634	#9-808							
UDPDR7	=	177636	#9-808	25-1942	25-1942	26-1991				
UIPAR0	=	177640	#9-808	12-1397	18-1666	22-1924	22-1924	22-1924	22-1924	26-2015
			*26-2015	26-2015	26-2015	26-2015	27-2090	28-2155	30-2296	30-2663
UIPAR1	=	177642	#9-808							
UIPAR2	=	177644	#9-808							
UIPAR3	=	177646	#9-808	32-2797						
UIPAR4	=	177650	#9-808	*30-2684	*30-2714	*30-2732	*30-2743			
UIPAR5	=	177652	#9-808	*30-2685	*30-2715	*30-2733	*30-2744			
UIPAR6	=	177654	#9-808							
UIPAR7	=	177656	#9-808							
UIPDR0	=	177600	#9-808	18-1672	25-1942	25-1942	25-1942	25-1942	25-1942	26-1991
			*26-2030	26-2030	26-2030	26-2030	27-2089	28-2143	30-2298	30-2673
			32-2797	32-2797						26-2030
UIPDR1	=	177602	#9-808							32-2797
UIPDR2	=	177604	#9-808							
UIPDR3	=	177606	#9-808							
UIPDR4	=	177610	#9-808	*30-2680	*30-2716	*30-2730	*30-2739			
UIPDR5	=	177612	#9-808	*30-2681	*30-2717	*30-2731	*30-2740			
UIPDR6	=	177614	#9-808							
UIPDR7	=	177616	#9-808							
USESTK	=	000600	#9-822	17-1557						
USP	=	%000006	#9-817	*17-1557	*17-1573	19-1843	*19-1844	*19-1845	*19-1852	

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES	12-1402	12-1408	*30-2699	30-2707	*31-2759	36-3061	36-3063	
VIRT1		001276	#10-884	12-1402	12-1408	*30-2699	30-2707	*31-2759	36-3061	36-3063	
VIRT2		001300	#10-884	*30-2698	36-3061						
WASR6		001256	#10-884	*12-1334	*12-1366	*29-2255	36-3050	36-3051			
WASSRO		001264	#10-884	*12-1367	*29-2253	*31-2761	36-3051	36-3063			
WASSR1		001266	#10-884	*12-1368	36-3051						
WASSR2		001270	#10-884	*12-1369	*18-1698	18-1700	*18-1719	18-1720	*29-2254	*31-2762	36-3051
				36-3062	36-3063	36-3066					
WBIT	=	000100	#9-819	32-2782	32-2782	32-2782	32-2789	32-2789	32-2789	32-2797	32-2797
				32-2797							
SAPTHD		000204	9-883	#9-883							
SASTAT	-	*****	34-2940	34-2940							
SATYC		037604	34-2940	#34-2940							
SATY1		037560	#34-2940								
SATY3		037566	34-2939	#34-2940							
SATY4		037576	34-2834	#34-2940							
SAUTOB		001134	#10-884	*13-1425	34-2920	34-2920	34-2920				
SBADR		001122	#10-884								
SBDDAT		001126	#10-884								
SBELL		001214	#10-884	34-2834	34-2834	34-2834					
SBIN		040100	*34-2941	*34-2941	34-2941	#34-2941					
SCHARC		037554	*34-2939	*34-2939	34-2939	*34-2939	#34-2939				
SCKSWR		036132	#34-2920	35-2947	35-2947						
SCLR.T		035106	34-2832	#34-2832							
SCMTAG		001100	#10-884	13-1424	13-1424	13-1424	13-1424	13-1424			
SCM1		000006	#10-884	10-884	10-884	#10-884	10-884	10-884	#10-884	10-884	
			#10-884	10-884	10-884	#10-884	10-884	10-884	#10-884	10-884	
			#10-884								
SCM2	-	000014	#10-884	10-884	10-884	#10-884	10-884	10-884	#10-884	10-884	
			#10-884	10-884	10-884	#10-884	10-884	10-884	#10-884	10-884	
			#10-884								
SCM3		000006	#10-884	10-884	10-884						
SCM4	=	000006	#10-884	10-884	10-884	#10-884	10-884	10-884	#10-884	10-884	
			#10-884	10-884	10-884	#10-884	10-884	10-884	#10-884	10-884	
			#10-884								
SCNTLC		001350	#10-884	34-2920	34-2920	34-2920	34-2920				
SCNTLG		037067	34-2920	#34-2920							
SCNTLU		037062	34-2920	34-2920	#34-2920						
SCORE		002676	12-1310	#12-1310							
SCPUOP		001252	#10-884								
SCRLF		001221	#10-884	34-2832	34-2834	34-2834	34-2834	34-2836	34-2855	34-2860	34-2915
				34-2920	34-2920	34-2920	34-2939	34-2939	34-2939		
SCROUT		002726	12-1310	#12-1310							
SDBLK		040544	34-2943	34-2943	#34-2943						
SDB20		040650	34-2887	34-2901	#34-2945						
SDEVCT		001234	#10-884								
SDOAGN		035126	34-2832	34-2832	34-2832	#34-2832					
SDTBL		040554	34-2943	#34-2943							
SENDAD		035116	9-882	13-1425	#34-2832	34-2834					
SENDCT		034744	13-1424	#34-2832							
SENULL		035170	#34-2832								
SENV		001244	#10-884	13-1425	30-2338	34-2834	34-2939	34-2940	34-2940		
SENVN		001245	#10-884	13-1424	34-2939	34-2939	34-2940				

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
\$EOP		034714	#34-2832 34-2934
\$EOPCT		034736	*13-1424 #34-2832 34-2832
\$ERFLG		001103	#10-884 12-1278 12-1303 34-2833 34-2833 *34-2833 34-2833 34-2833 *34-2834
\$ERMAX		001115	#10-884 *13-1424 *34-2833 34-2833 34-2833
\$ERROR		035366	13-1424 #34-2834
\$ERRPC		001116	#10-884 *34-2834 *34-2834 34-2834 34-2834 34-2834 34-2834 34-2842 36-3050 36-3051 36-3052 36-3053 36-3054 36-3055 36-3056 36-3057 36-3058 36-3059 36-3060 36-3061 36-3062 36-3063 36-3064 36-3065 36-3066 36-3070 36-3073 36-3074 36-3075 36-3076 36-3077 36-3078 36-3079 36-3080 36-3081
\$ERRTB		001372	#11-884 34-2850
\$ERTTL		001112	#10-884 34-2832 *34-2832 *34-2834 34-2834 34-2834
\$ESCAP		001212	#10-884 *13-1424 *34-2833 34-2834 34-2834 34-2834
\$ETABL		001244	#10-884
\$ETEND		001254	9-883 #10-884
\$FATAL		001226	#10-884 *34-2940
\$FFLG		040024	*34-2940 *34-2940 34-2940 *34-2940 #34-2940
\$FILLC		001156	#10-884 34-2939 34-2939 34-2939
\$FILLS		001155	#10-884 34-2939 34-2939
\$GDADR		001120	#10-884
\$GDDAT		001124	#10-884
\$GET42		035070	#34-2832
\$GTSWR		036202	#34-2920 35-2947 35-2947
\$HD		000000	9-805 9-805 9-805
\$HIBTS		000204	#9-883
\$ICNT		001104	#10-884
\$ILLUP		041234	35-2948 35-2948 #35-2948
\$INTAG		001135	#10-884 34-2920 34-2920 34-2920 34-2920
\$ITEMB		001114	#10-884 *34-2834 34-2834 34-2834 34-2834 34-2839
\$KTNEX		002670	12-1310 #12-1310
\$KTOUT		002660	12-1310 #12-1310
\$KT11		002512	#12-1310 *12-1310 *12-1310 *15-1442
\$LF		001222	#10-884 34-2834 34-2834 34-2920 34-2920 34-2920 34-2939 34-2939
\$LFLG		040023	*34-2940 #34-2940
\$LOOP		035164	34-2832 #34-2832
\$LPADR		001106	#10-884 *13-1424 30-2489 30-2629 *34-2833 *34-2833 34-2833 34-2833 34-2833
\$LPERF		001110	#10-884 *13-1424 *16-1461 *16-1476 *16-1484 *16-1499 *16-1510 *16-1524 *16-1537 *17-1567 *18-1604 *18-1617 *18-1642 *18-1642 *18-1648 *18-1648 *18-1654 *18-1654 *18-1660 *18-1660 *18-1666 *18-1666 *18-1672 *18-1672 *18-1697 *18-1706 *18-1730 *19-1771 *19-1779 *19-1784 *19-1794 *19-1804 *19-1815 *19-1828 *19-1841 *19-1855 *20-1872 *20-1881 *20-1887 *20-1892 *21-1913 *21-1913 *21-1913 *21-1918 *21-1918 *21-1918 *22-1924 *22-1924 *22-1924 *22-1924 *23-1930 *23-1930 *23-1930 *24-1936 *24-1936 *24-1936 *25-1942 *25-1942 *25-1942 *25-1942 *25-1960 *25-1960 *25-1966 *25-1966 *25-1972 *25-1972 *25-1978 *25-1978 *25-1978 *26-1985 *26-1985 *26-1991 *26-1991 *26-2005 *26-2005 *26-2005 *26-2010 *26-2010 *26-2010 *26-2010 *26-2015 *26-2015 *26-2015 *26-2020 *26-2020 *26-2020 *26-2025 *26-2025 *26-2025 *26-2025 *26-2030 *26-2030 *26-2030 *27-2046 *27-2054 *27-2062 *29-2192 *29-2206 *29-2224 *29-2247 *30-2310 *30-2311 *30-2312 *30-2313 *30-2317 *30-2318 *30-2319 *30-2320 *30-2321 *30-2322 *30-2323 *30-2324 *30-2325 *30-2326 *30-2327 *30-2328 *30-2329 *30-2379 *30-2391 *30-2426 *30-2443 *30-2451 *30-2464 *30-2522 *30-2534 *30-2567 *30-2582 *30-2590 *30-2603 *30-2689 *30-2748 *32-2782 *32-2782 *32-2789 *32-2789 *32-2797 *32-2797 *33-2814 *33-2827 34-2833 *34-2833 34-2833 34-2833 34-2834

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES								
\$LSTAD	003020	*12-1310	#12-1310							
\$LSTBK	003022	*12-1310	#12-1310	30-2314	30-2371	30-2383	30-2452	30-2456	30-2474	30-2515
		30-2525	30-2550	30-2591	30-2595	30-2614	36-3077			
\$MAIL	001224	9-883	9-883	#10-884	13-1424	13-1425	34-2833	34-2834	34-2939	
\$MBADR	000206	#9-883								
\$MFLG	040022	*34-2940	34-2940	*34-2940	#34-2940					
\$MNEW	037105	34-2920	#34-2920							
\$MSGAD	001240	#10-884	*34-2940	34-2940						
\$MSGLG	001242	#10-884	*34-2940							
\$MSGTY	001224	#10-884	34-2940	*34-2940	34-2940	*34-2940				
\$MSWR	037074	34-2920	#34-2920							
\$MXCNT	001344	#10-884								
\$NULL	001154	#10-884	34-2939	34-2939	34-2939					
\$NWTST	- 000001	#16-1460	16-1460	#16-1460	16-1460	#16-1483	16-1483	#16-1483	16-1483	#16-1509
		16-1509	#16-1509	16-1509	#17-1550	17-1550	#17-1550	17-1550	#18-1601	18-1601
		#18-1601	18-1601	#18-1641	18-1641	#18-1641	18-1641	#18-1647	18-1647	#18-1647
		#18-1647	#18-1653	18-1653	#18-1653	18-1653	#18-1659	18-1659	#18-1659	18-1659
		#18-1665	18-1665	#18-1665	18-1665	#18-1671	18-1671	#18-1671	18-1671	#18-1686
		18-1686	#18-1686	18-1686	#18-1741	18-1741	#18-1741	18-1741	#19-1768	19-1768
		#19-1768	19-1768	#20-1864	20-1864	#20-1864	20-1864	#21-1912	21-1912	#21-1912
		21-1912	#21-1917	21-1917	#21-1917	21-1917	#22-1923	22-1923	#22-1923	22-1923
		#23-1929	23-1929	#23-1929	23-1929	#24-1935	24-1935	#24-1935	24-1935	#25-1941
		25-1941	#25-1941	25-1941	#25-1959	25-1959	#25-1959	25-1959	#25-1965	25-1965
		#25-1965	25-1965	#25-1971	25-1971	#25-1971	25-1971	#25-1977	25-1977	#25-1977
		25-1977	#26-1984	26-1984	#26-1984	26-1984	#26-1990	26-1990	#26-1990	26-1990
		#26-2004	26-2004	#26-2004	26-2004	#26-2009	26-2009	#26-2009	26-2009	#26-2014
		26-2014	#26-2014	26-2014	#26-2019	26-2019	#26-2019	26-2019	#26-2024	26-2024
		#26-2024	26-2024	#26-2029	26-2029	#26-2029	26-2029	#27-2045	27-2045	#27-2045
		27-2045	#27-2073	27-2073	#27-2073	27-2073	#29-2181	29-2181	#29-2181	29-2181
		#29-2222	29-2222	#29-2222	29-2222	#30-2284	30-2284	#30-2284	30-2284	#30-2365
		30-2365	#30-2365	30-2365	#30-2509	30-2509	#30-2509	30-2509	#30-2656	30-2656
		#30-2656	30-2656	#32-2781	32-2781	#32-2781	32-2781	#32-2787	32-2787	#32-2787
		32-2787	#32-2795	32-2795	#32-2795	32-2795	#33-2810	33-2810	#33-2810	33-2810
\$OCNT	040324	*34-2942	*34-2942	#34-2942						
\$OCTVL	040752	34-2945	#34-2945							
\$OMODE	040326	*34-2942	*34-2942	34-2942	*34-2942	*34-2942	#34-2942			
\$OVER	035352	34-2833	34-2833	34-2833	#34-2833					
\$PASS	001232	#10-884	*13-1424	*34-2832	*34-2832	34-2832	34-2832	34-2832	34-2924	
\$PASTM	000212	#9-883								
\$PWRAD	041216	#35-2948								
\$PWRDN	041056	13-1424	#35-2948	35-2948						
\$PWRMG	041212	#35-2948								
\$PWRUP	041130	35-2948	#35-2948							
\$QUES	001220	#10-884	34-2834	34-2834	34-2920	34-2920	34-2920	34-2920	34-2939	34-2939
\$RDCHR	036454	#34-2920	35-2947	35-2947						
\$RDDEC	= *****	35-2947								
\$RDLIN	036574	#34-2920	35-2947	35-2947						
\$RDOCT	*****	35-2947								
\$RDSZ	- 000010	#34-2920	34-2920							
\$REGAD	001160	#10-884								
\$REGO	001162	#10-884	*34-2834	36-3052	36-3053	36-3055	36-3056	36-3059	36-3060	36-3067
		36-3070	36-3071	36-3072	36-3074	36-3079				





SYMBOL CROSS REFERENCE

SYMBOL	VALUE
\$USWR	001250
\$XTSTR	035206
\$GET4	= 000001
\$FILL	040325
\$OCAT	= *****
.\$ASTA	= *****
.\$X	= 000204

REFERENCES

#10-884	30-2341
#34-2833	
#34-2832	#34-2832 34-2832
*34-2942	*34-2942 34-2942 #34-2942
34-2833	34-2834
34-2940	34-2940
#9-883	9-883





MACRO CROSS REFERENCE

MACRO NAME	REFERENCES									
MSG5	#18-1593	18-1601								
MSG6	#18-1638	#18-1641								
MSG7	#18-1644	#18-1647								
MSG7A	#18-1650	18-1653								
MSG7B	#18-1656	18-1659								
MULT	#9-807									
NEWTST	#9-807	16-1460	16-1483	16-1509	17-1550	18-1601	18-1641	18-1647	18-1653	18-1659
		18-1665	18-1671	18-1686	18-1741	19-1768	20-1864	21-1912	21-1917	22-1923
		24-1935	25-1941	25-1959	25-1965	25-1971	25-1977	26-1984	26-1990	26-2004
		26-2014	26-2019	26-2024	26-2029	27-2045	27-2073	29-2181	29-2222	30-2284
		30-2509	30-2656	32-2781	32-2787	32-2795	33-2810			30-2365
PARMSG	#25-1944	25-1959	25-1965	25-1971	25-1977	26-1984	26-1990			
POP	#9-807	34-2940	34-2940	34-2943	34-2944	35-2948	35-2948			
PUSH	#9-807	#34-2940	#34-2940	#34-2940	#34-2943	#34-2944	#35-2948	#35-2948		
REPORT	#9-807									
SAVR	#7-499	34-2834								
SETPRI	#9-807									
SETTRA	#35-2947	#35-2947	#35-2947	#35-2947	#35-2947	#35-2947	#35-2947	#35-2947	#35-2947	#35-2947
	#35-2947	#35-2947	#35-2947							
SETUP	#9-807	#13-1424								
SKIP	#9-807	17-1576	18-1622	18-1642	18-1648	18-1654	18-1660	18-1666	18-1672	21-1913
		21-1918	22-1924	23-1930	24-1936	25-1942	25-1960	25-1966	25-1972	25-1978
		26-1991	29-2249	30-2465	30-2605	30-2749				26-1985
SLASH	#9-807									
SPACE	#7-496	#9-807								
STARS	#9-807	#9-882	#9-883	#9-883	#9-883	#10-884	#10-884	#10-884	#12-1193	#12-1200
	#12-1212	#12-1218	#12-1228	#12-1237	#12-1247	#12-1251	#12-1264	#12-1270	#12-1286	#12-1294
	#12-1315	#12-1322	#12-1348	#12-1355	#12-1378	#12-1392	#16-1460	#16-1460	#16-1483	#16-1483
	#16-1509	#16-1509	#17-1550	#17-1550	#18-1579	#18-1591	#18-1601	#18-1601	#18-1641	#18-1641
	#18-1647	#18-1647	#18-1653	#18-1653	#18-1659	#18-1659	#18-1665	#18-1665	#18-1671	#18-1671
	#18-1686	#18-1686	#18-1741	#18-1741	#19-1768	#19-1768	#20-1864	#20-1864	#21-1912	#21-1912
	#21-1917	#21-1917	#22-1923	#22-1923	#23-1929	#23-1929	#24-1935	#24-1935	#25-1941	#25-1941
	#25-1959	#25-1959	#25-1965	#25-1965	#25-1971	#25-1971	#25-1977	#25-1977	#26-1984	#26-1984
	#26-1990	#26-1990	#26-2004	#26-2004	#26-2009	#26-2009	#26-2014	#26-2014	#26-2019	#26-2019
	#26-2024	#26-2024	#26-2029	#26-2029	#27-2045	#27-2045	#27-2073	#27-2073	#29-2181	#29-2181
	#29-2222	#29-2222	#30-2284	#30-2284	#30-2365	#30-2365	#30-2509	#30-2509	#30-2656	#30-2656
	#32-2781	#32-2781	#32-2787	#32-2787	#32-2795	#32-2795	#33-2810	#33-2810	#34-2832	#34-2833
	#34-2834	#34-2920	#34-2920	#34-2920	#34-2920	#34-2920	#34-2939	#34-2940	#34-2941	#34-2943
	#34-2944	#34-2945	#35-2947	#35-2948	#35-2948					
SWRSU	#9-807	#13-1424	13-1424							
TIMMSG	#7-591	18-1641	18-1647	18-1653	18-1659	18-1665	18-1671			
TIMTST	#7-596	#18-1642	#18-1648	#18-1654	#18-1660	#18-1666	#18-1672			
TRMTRP	#35-2947									
TYPBIN	#9-807									
TYPDEC	#9-807	#34-2832	#34-2832							
TYPNAM	#9-807	#13-1425								
TYPNUM	#9-807									
TYPOCS	#9-807									
TYPOCT	#9-807	#34-2920								
TYPTXT	#9-807	#34-2832	#34-2832							
USER	#9-833	#10-884								
WMSG	#8-709	32-2781	32-2787	32-2795						

