

# PDP11/23

KTF11-AA DIAG  
CJKDAB0

AH-F137B-MC  
COPYRIGHT 1979  
FICHE 1 OF 1

SEP 1979  
**digital**  
MADE IN USA

The microfiche card contains a grid of frames. Each frame likely contains a small portion of a larger technical document, such as a schematic diagram, a table of data, or a text-based instruction. The frames are arranged in a regular grid pattern, typical of microfiche format. The overall appearance is that of a dense technical reference tool.



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-F138B-MC  
PRODUCT NAME: CJKDAB0 KTF11-AA MEMORY MANAGEMENT DIAGNOSTIC  
DATE: JUNE-1979  
MAINTAINER: DIAGNOSTIC PROGRAMMING  
AUTHOR: DIAGNOSTIC ENGINEERING

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

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED 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

43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58

PROGRAM HISTORY

<u>DATE</u>	<u>REVISION</u>	<u>REASON FOR REVISION</u>
12-JAN-79	A	FIRST RELEASE
JUNE-79	B	SUBROUTINE FORMPA MODIFIED ERROR INFORMATION STORED IN R0,R2. THIS REVISION SAVES THE REGISTERS ON ENTRY TO THE ROUTINE AND RESTORES THEM ON EXIT.

59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92

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
4.6	RELOCATION THROUGHOUT MEMORY
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  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148

1.0 PROGRAM INFORMATION  
-----

1.1 ABSTRACT  
-----

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

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

1.2 REQUIREMENTS  
-----

A KDF11-A PROCESSOR WITH A MINIMUM OF 16K OF MEMORY AND A CONSOLE TERMINAL ARE REQUIRED TO RUN THE PROGRAM UNLESS THE PROGRAM IS RUNNING UNDER APT OR ACT IN WHICH CASE THE CONSOLE TERMINAL IS NOT NECESSARY.

1.3 RELATED DOCUMENTS AND STANDARDS  
-----

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

1.4 PRELIMINARY PROGRAMS  
-----

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

CJKDB DCF11-AA CPU TESTS

ALSO, ONE OF THE MAIN MEMORY DIAGNOSTICS SHOULD BE RUN TO SCAN AT LEAST THE FIRST 16K TO SEE THAT A PROGRAM CAN BE EXECUTED.

2.0 OPERATING INSTRUCTIONS  
-----

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

2.1 LOADING PROCEDURES

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. SINCE THERE IS NO HARDWARE SWITCH REGISTER, 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)

2.3 CONTROL SWITCH SETTINGS

<u>SWITCH</u>	<u>OCTAL VALUE</u>	<u>USE</u>
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.
SW11	004000	INHIBIT SUBTEST ITERATIONS THIS SWITCH WHEN SET INHIBITS ITERATIONS OF EACH SUBTEST AFTER

205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260

THE FIRST PASS. IF THIS SWITCH IS NOT SET, EACH SUBTEST IS RUN 200. TIMES.

SW10	002000	BELL ON ERROR
		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

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 ^J HAD BEEN TYPED.

NOTE: RECOGNITION OF A ^G MAY BE HAMPERED BY

261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316

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

2.5 EXECUTION TIMES  
-----

THE RUN TIME FOR A SINGLE PASS WITH NO ITERATIONS  
OR TRACE TRAPPING IS APPROXIMATELY 5 SECONDS.

THE RUN TIME FOR A SINGLE PASS WITH ITERATIONS  
AND TRACE TRAPPING ENABLED IS APPROXIMATELY 30 SECONDS.

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



317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372

' " : LOGIC WAS BEING TESTED, THE OPERATOR CAN THEN REFER TO THE ENGINEERING DRAWINGS TO ISOLATE THE PROBABLE CAUSE FOR THE FAILURE.

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

### 4.0 MISCELLANEOUS INFORMATION

-----

#### 4.1 ACT/APT/XXDP COMPATABILITY

-----

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

#### 4.2 END-OF-PASS MESSAGE

-----

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

END OF PASS #2 TOTAL ERRORS SINCE LAST REPORT 0

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

#### 4.3 T-BIT TRAPPING

-----

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

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

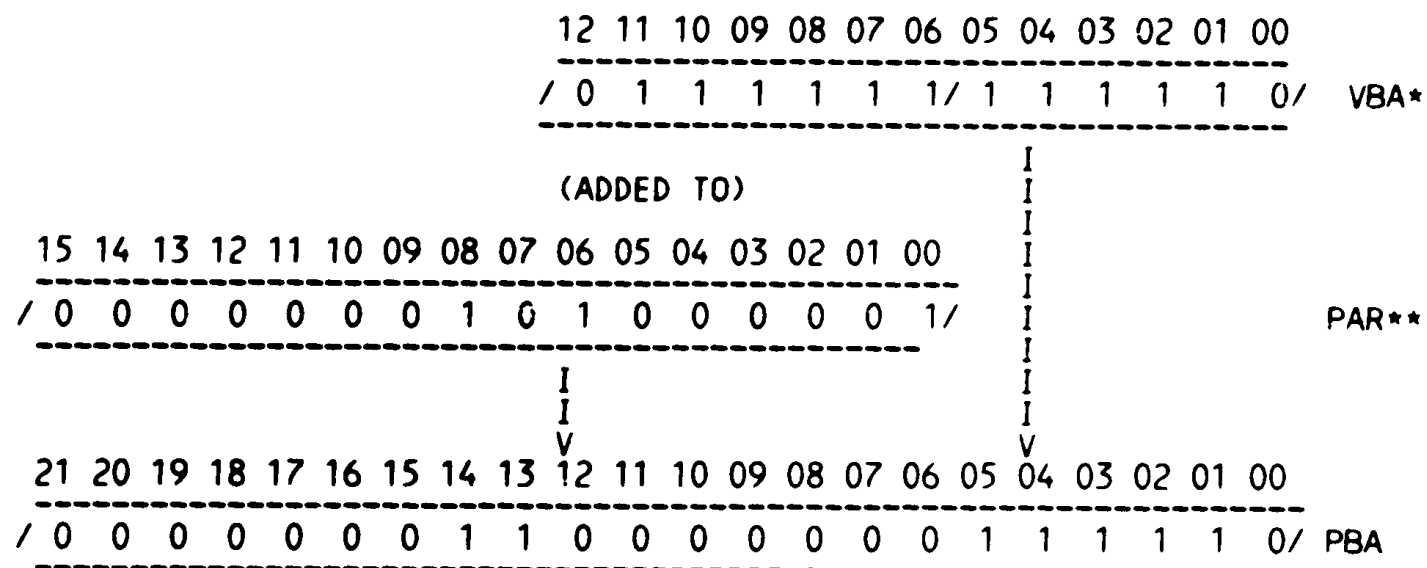
REGISTER (SEE SECTION 2.4).

4.4 POWER FAILURE HANDLING

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 'RESTRT:'

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 BIT 01 (BIT 15) SELECTS THE USER (=1) OR KERNEL (=0) SET OF PAR'S/PDR'S

4.6 RELOCATION THROUGHOUT MEMORY

A FEATURE WAS ADDED TO ALLOW THE CONSTRUCTION OF PHYSICAL BUS ADDRESSES ABOVE THE NORMAL 16K LIMIT. THE SETTING OF THE LOCATION \$MADR1 IN THE E-TABLE WITH ONE OF THE FOLLOWING CONSTANTS WILL ACCESS LOCATIONS BETWEEN 0 AND 7600 OF EACH 4K GROUP UP TO THE MAXIMUM MEMORY ON THE SYSTEM. THE FIRST LOCATION OF EACH BLOCK(32 WORDS) IS WRITTEN AND READ. SEE TEST #24 IN THE LISTING FOR MORE DETAILS.

CONSTANT	MAX. MEM.	CONSTANT	MAX. MEM.	CONSTANT	MAX. MEM.
0 OR 600	16K	3200	56K	5600	96K
1000	20K	3400	60K	6000	100K

429	1200	24K	3600	64K	6200	104K
430	1400	28K	4000	68K	6400	108K
431	1600	32K	4200	72K	6600	112K
432	2000	36K	4400	76K	7000	116K
433	2200	40K	4600	80K	7200	120K
434	2400	44K	5000	84K	7400	124K
435	2600	48K	5200	88K	7600	128K
436	3000	52K	5400	92K		

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. READ AND COMPARE KERNEL AND USER PAR/PDR'S
5. CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS

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. IF THE PASS NUMBER IS GREATER THAN ONE, THE ERROR MAY BE ITERATION SENSITIVE. TRY RUNNING THE PROGRAM AGAIN WITH BIT 11 OF THE SWITCH REG. EQUAL TO '1' TO INHIBIT ITERATIONS. THESE HINTS SHOULD HELP YOU DETERMINE WHAT MAKES THE MACHINE FAIL AND WHEN.

IF YOU HAVE BEEN RUNNING WITH BIT 15 OF THE SWITCH

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  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511

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'.

@

!

```

512
513 .TITLE MD-11-CJKDA-B KTF11-AA MMU DIAG
514 :*COPYRIGHT (C) JUNE-79
515 :*DIGITAL EQUIPMENT CORP.
516 :*MAYNARD, MASS. 01754
517 :*
518 :*
519 :*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
520 :*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
521 :*
522 .SBTTL OPERATIONAL SWITCH SETTINGS
523 :*
524 :*      SWITCH      USE
525 :*      -----      -----
526 :*      15          HALT ON ERROR
527 :*      14          LOOP ON TEST
528 :*      13          INHIBIT ERROR TYPEOUTS
529 :*      12          INHIBIT TRACE TRAP
530 :*      11          INHIBIT ITERATIONS
531 :*      10          BELL ON ERROR
532 :*      9           LOOP ON ERROR
533 :*      8           LOOP ON TEST IN SWR<7:0>
534 .SBTTL BASIC DEFINITIONS
535 :*
536 :*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
537 001100 STACK= 1100
538 .EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
539 .EQUIV IOT,SCOPE     ;;BASIC DEFINITION O. SCOPE CALL
540 :*
541 :*MISCELLANEOUS DEFINITIONS
542 000011 HT= 11      ;;CODE FOR HORIZONTAL TAB
543 000012 LF= 12      ;;CODE FOR LINE FEED
544 000015 CR= 15      ;;CODE FOR CARRIAGE RETURN
545 000200 CRLF= 200   ;;CODE FOR CARRIAGE RETURN-LINE FEED
546 177776 PS= 177776 ;;PROCESSOR STATUS WORD
547 .EQUIV PS,PSW
548 177774 STKLMT= 177774 ;;STACK LIMIT REGISTER
549 177772 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
550 177570 DSWR= 177570 ;;HARDWARE SWITCH REGISTER
551 177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
552 :*
553 :*GENERAL PURPOSE REGISTER DEFINITIONS
554 000000 R0= %0      ;;GENERAL REGISTER
555 000001 R1= %1      ;;GENERAL REGISTER
556 000002 R2= %2      ;;GENERAL REGISTER
557 000003 R3= %3      ;;GENERAL REGISTER
558 000004 R4= %4      ;;GENERAL REGISTER
559 000005 R5= %5      ;;GENERAL REGISTER
560 000006 R6= %6      ;;GENERAL REGISTER
561 000007 R7= %7      ;;GENERAL REGISTER
562 000006 SP= %6      ;;STACK POINTER
563 000007 PC= %7      ;;PROGRAM COUNTER
564 :*
565 :*PRIORITY LEVEL DEFINITIONS
566 000000 PRG= 0      ;;PRIORITY LEVEL 0
567 000040 PR1= 40     ;;PRIORITY LEVEL 1

```



568	000100	PR2=	100	::PRIORITY LEVEL 2
569	000140	PR3=	140	::PRIORITY LEVEL 3
570	000200	PR4=	200	::PRIORITY LEVEL 4
571	000240	PR5=	240	::PRIORITY LEVEL 5
572	000300	PR6=	300	::PRIORITY LEVEL 6
573	000340	PR7=	340	::PRIORITY LEVEL 7

574  
575 :\*'SWITCH REGISTER' SWITCH DEFINITIONS

576	100000	SW15=	100000
577	040000	SW14=	40000
578	020000	SW13=	20000
579	010000	SW12=	10000
580	004000	SW11=	4000
581	002000	SW10=	2000
582	001000	SW09=	1000
583	000400	SW08=	400
584	000200	SW07=	200
585	000100	SW06=	100
586	000040	SW05=	40
587	000020	SW04=	20
588	000010	SW03=	10
589	000004	SW02=	4
590	000002	SW01=	2
591	000001	SW00=	1
592		.EQUIV	SW09,SW9
593		.EQUIV	SW08,SW8
594		.EQUIV	SW07,SW7
595		.EQUIV	SW06,SW6
596		.EQUIV	SW05,SW5
597		.EQUIV	SW04,SW4
598		.EQUIV	SW03,SW3
599		.EQUIV	SW02,SW2
600		.EQUIV	SW01,SW1
601		.EQUIV	SW00,SW0

602  
603 :\*DATA BIT DEFINITIONS (BIT00 TO BIT15)

604	100000	BIT15=	100000
605	040000	BIT14=	40000
606	020000	BIT13=	20000
607	010000	BIT12=	10000
608	004000	BIT11=	4000
609	002000	BIT10=	2000
610	001000	BIT09=	1000
611	000400	BIT08=	400
612	000200	BIT07=	200
613	000100	BIT06=	100
614	000040	BIT05=	40
615	000020	BIT04=	20
616	000010	BIT03=	10
617	000004	BIT02=	4
618	000002	BIT01=	2
619	000001	BIT00=	1
620		.EQUIV	BIT09,BIT9
621		.EQUIV	BIT08,BIT8
622		.EQUIV	BIT07,BIT7
623		.EQUIV	BIT06,BIT6

```
624 .EQUIV BIT05,BIT5
625 .EQUIV BIT04,BIT4
626 .EQUIV BIT03,BIT3
627 .EQUIV BIT02,BIT2
628 .EQUIV BIT01,BIT1
629 .EQUIV BIT00,BIT0
630
631 ;*BASIC 'CPU' TRAP VECTOR ADDRESSES
632 000004 ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
633 000010 RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
634 000014 TBITVEC=14 ;: 'T' BIT
635 000014 TRTVEC= 14 ;:TRACE TRAP
636 000014 BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)
637 000020 IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
638 000024 PWRVEC= 24 ;:POWER FAIL
639 000030 EMTVEC= 30 ;:EMULATOR TRAP (EMT) **ERROR**
640 000034 TRAPVEC=34 ;: 'TRAP' TRAP
641 000060 TKVEC= 60 ;:TTY KEYBOARD VECTOR
642 000064 TPVEC= 64 ;:TTY PRINTER VECTOR
643 000240 PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR
644 .SBTTL MEMORY MANAGEMENT DEFINITIONS
645
646 ;*KT11 VECTOR ADDRESS
647
648 000250 MMVEC= 250
649
650 ;*KT11 STATUS REGISTER ADDRESSES
651
652 177572 SR0= 177572
653 177574 SR1= 177574
654 177576 SR2= 177576
655 172516 SR3= 172516
656
657 ;*USER 'I' PAGE DESCRIPTOR REGISTERS
658
659 177600 UIPDR0= 177600
660 177602 UIPDR1= 177602
661 177604 UIPDR2= 177604
662 177606 UIPDR3= 177606
663 177610 UIPDR4= 177610
664 177612 UIPDR5= 177612
665 177614 UIPDR6= 177614
666 177616 UIPDR7= 177616
667
668 ;*USER 'I' PAGE ADDRESS REGISTERS
669
670 177640 UIPAR0= 177640
671 177642 UIPAR1= 177642
672 177644 UIPAR2= 177644
673 177646 UIPAR3= 177646
674 177650 UIPAR4= 177650
675 177652 UIPAR5= 177652
676 177654 UIPAR6= 177654
677 177656 UIPAR7= 177656
678
679 ;*KERNEL 'I' PAGE DESCRIPTOR REGISTERS
```

```

680
681      172300      KIPDR0= 172300
682      172302      KIPDR1= 172302
683      172304      KIPDR2= 172304
684      172306      KIPDR3= 172306
685      172310      KIPDR4= 172310
686      172312      KIPDR5= 172312
687      172314      KIPDR6= 172314
688      172316      KIPDR7= 172316
689
690      ;*KERNEL 'I' PAGE ADDRESS REGISTERS
691
692      172340      KIPAR0= 172340
693      172342      KIPAR1= 172342
694      172344      KIPAR2= 172344
695      172346      KIPAR3= 172346
696      172350      KIPAR4= 172350
697      172352      KIPAR5= 172352
698      172354      KIPAR6= 172354
699      172356      KIPAR7= 172356
700
701      .EQUIV SP,KSP
702      .EQUIV SP,USP
703      .EQUIV BIT4,TBIT
704      .EQUIV BIT6,WBIT
705      001100      KERSTK= STACK
706      000700      USESTK= STACK-200
707
708      ;*ADDITIONAL DEFINITIONS
709      ;*
710
711
712      .SBTTL TRAP CATCHER
713
714      000000      .=0
715      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A '+2,HALT'
716      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
717      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
718      000174      .=174
719      000174      000000      DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
720      000176      000000      SWREG: .WORD 0          ;;SOFTWARE SWITCH REGISTER
721
722      000200      000137      020000      .SBTTL STARTING ADDRESS(ES)
723      .SBTTL      JMP @*START ;;JUMP TO STARTING ADDRESS OF PROGRAM
724
725      .SBTTL ACT11 HOOKS
726
727      000204      $SVPC=.          ;;HOOKS REQUIRED BY ACT11          ;SAVE PC
728      000046      .=46
729      000046      034072      $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
730      000052      .=52
731      000052      000000      .WORD 0          ;;2)SET LOC.52 TO ZERO
732      000204      .= $SVPC          ;; RESTORE PC
733      .SBTTL APT PARAMETER BLOCK
734
735      ;:*****
    
```

```
736 ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
737 ;*****
738 .SX=. ;:SAVE CURRENT LOCATION
739 =24 ;:SET POWER FAIL TO POINT TO START OF PROGRAM
740 000024 000200 200 ;:FOR APT START UP
741 000044 000044 -.44 ;:POINT TO APT INDIRECT ADDRESS PNTR.
742 000044 000204 $APTHDR ;:POINT TO APT HEADER BLOCK
743 000204 .=$X ;:RESET LOCATION COUNTER
744 ;*****
745 ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
746 ;INTERFACE SPEC.
747
748 000204 $APTHD:
749 000204 000000 $HIBTS: .WORD 0 ;:TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
750 000206 001226 $MBADR: .WORD $MAIL ;:ADDRESS OF APT MAILBOX (BITS 0-15)
751 000210 000010 $STMT: .WORD 10 ;:RUN TIM OF LONGEST TEST
752 000212 000020 $PASTM: .WORD 20 ;:RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
753 000214 000005 $UNITM: .WORD 5 ;:ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
754 000216 000016 .WORD $ETEND-$MAIL/2 ;:LENGTH MAILBOX-ETABLE(WORDS)
```

```

755          .SBTTL COMMON TAGS
756
757          ::*****
758          ::*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
759          ::*USED IN THE PROGRAM.
760
761          001100          .=1100
762          001100          $CMTAG:          ;;START OF COMMON TAGS
763          001100          000000          .WORD          0          ;;CONTAINS THE TEST NUMBER
764          001102          000          $TSTNM: .BYTE          0          ;;CONTAINS ERROR FLAG
765          001103          000          $ERFLG: .BYTE          0          ;;CONTAINS SUBTEST ITERATION COUNT
766          001104          000000          $ICNT:  .WORD          0          ;;CONTAINS SCOPE LOOP ADDRESS
767          001106          000000          $LPADR: .WORD          0          ;;CONTAINS SCOPE RETURN FOR ERRORS
768          001110          000000          $LPERR: .WORD          0          ;;CONTAINS TOTAL ERRORS DETECTED
769          001112          000000          $ERTTL: .WORD          0          ;;CONTAINS ITEM CONTROL BYTE
770          001114          000          $ITEMB: .BYTE          0          ;;CONTAINS MAX. ERRORS PER TEST
771          001115          001          $ERMAX: .BYTE          1          ;;CONTAINS PC OF LAST ERROR INSTRUCTION
772          001116          000000          $ERRPC: .WORD          0          ;;CONTAINS ADDRESS OF 'GOOD' DATA
773          001120          000000          $GDADR: .WORD          0          ;;CONTAINS ADDRESS OF 'BAD' DATA
774          001122          000000          $BDADR: .WORD          0          ;;CONTAINS 'GOOD' DATA
775          001124          000000          $GDDAT: .WORD          0          ;;CONTAINS 'BAD' DATA
776          001126          000000          $BDDAT: .WORD          0          ;;RESERVED--NOT TO BE USED
777          001130          000000          .WORD          0
778          001132          000000          .WORD          0
779          001134          000          $AUTOB: .BYTE          0          ;;AUTOMATIC MODE INDICATOR
780          001135          000          $INTAG: .BYTE          0          ;;INTERRUPT MODE INDICATOR
781          001136          000000          .WORD          0
782          001140          177570          SWR:      .WORD          DSWR          ;;ADDRESS OF SWITCH REGISTER
783          001142          177570          DISPLAY: .WORD          DDISP          ;;ADDRESS OF DISPLAY REGISTER
784          001144          177560          $TKS:      177560          ;;TTY KBD STATUS
785          001146          177562          $TKB:      177562          ;;TTY KBD BUFFER
786          001150          177564          $TPS:      177564          ;;TTY PRINTER STATUS REG. ADDRESS
787          001152          177566          $TPB:      177566          ;;TTY PRINTER BUFFER REG. ADDRESS
788          001154          000          $NULL:     .BYTE          0          ;;CONTAINS NULL CHARACTER FOR FILLS
789          001155          002          $FILLS:    .BYTE          2          ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
790          001156          012          $FILLC:    .BYTE          12         ;;INSERT FILL CHARS. AFTER A 'LINE FEED'
791          001157          000          $TPFLG:    .BYTE          0          ;;'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
792          001160          000000          $REGAD:    .WORD          0          ;;CONTAINS THE ADDRESS FROM
793          001162          000000          $REG0:     .WORD          0          ;;WHICH ($REG0) WAS OBTAINED
794          001164          000000          $REG1:     .WORD          0          ;;CONTAINS (($REGAD)+0)
795          001166          000000          $REG2:     .WORD          0          ;;CONTAINS (($REGAD)+2)
796          001170          000000          $REG3:     .WORD          0          ;;CONTAINS (($REGAD)+4)
797          001172          000000          $REG4:     .WORD          0          ;;CONTAINS (($REGAD)+6)
798          001174          000000          $REG5:     .WORD          0          ;;CONTAINS (($REGAD)+10)
799          001176          000000          $REG6:     .WORD          0          ;;CONTAINS (($REGAD)+12)
800          001176          000000          $TMP0:     .WORD          0          ;;USER DEFINED
801          001200          000000          $TMP1:     .WORD          0          ;;USER DEFINED
802          001202          000000          $TMP2:     .WORD          0          ;;USER DEFINED
803          001204          000000          $TMP3:     .WORD          0          ;;USER DEFINED
804          001206          000000          $TMP4:     .WORD          0          ;;USER DEFINED
805          001210          000000          $TMP5:     .WORD          0          ;;USER DEFINED
806          001212          000000          $TIMES:    0          ;;MAX. NUMBER OF ITERATIONS
807          001214          000000          $ESCAPE:   0          ;;ESCAPE ON ERROR ADDRESS
808          001216          177607          000377          $BELL:     .ASCII          <207><377><377> ;;CODE FOR BELL
809          001222          077          $QUES:     .ASCII          /?/          ;;QUESTION MARK
810          001223          015          $CRLF:     .ASCII          <15>          ;;CARRIAGE RETJRN
    
```



```

811 001224 000012 $LF: .ASCIZ <12> ;:LINE FEED
812 ;:*****
813 .SBTTL APT MAILBOX-ETABLE
814 ;:*****
815 ;:*****
816 .EVEN
817 001226 $MAIL: ;:APT MAILBOX
818 001226 000000 $MSGTY: .WORD AMSGTY ;:MESSAGE TYPE CODE
819 001230 000000 $FATAL: .WORD AFATAL ;:FATAL ERROR NUMBER
820 001232 000000 $TESTN: .WORD ATESTN ;:TEST NUMBER
821 001234 000000 $PASS: .WORD APASS ;:PASS COUNT
822 001236 000000 $DEVCT: .WORD ADEVCT ;:DEVICE COUNT
823 001240 000000 $UNIT: .WORD AUNIT ;:I/O UNIT NUMBER
824 001242 000000 $MSGAD: .WORD AMSGAD ;:MESSAGE ADDRESS
825 001244 000000 $MSGLG: .WORD AMSGLG ;:MESSAGE LENGTH
826 001246 $ETABLE: ;:APT ENVIRONMENT TABLE
827 001246 000 $ENV: .BYTE AENV ;:ENVIRONMENT BYTE
828 001247 000 $ENVM: .BYTE AENVM ;:ENVIRONMENT MODE BITS
829 001250 000000 $SWREG: .WORD ASWREG ;:APT SWITCH REGISTER
830 001252 000000 $USWR: .WORD AUSWR ;:USER SWITCHES
831 001254 000000 $CPUOP: .WORD ACPUOP ;:CPU TYPE,OPTIONS
832 ;* BITS 15-11=CPU TYPE
833 ;* 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
834 ;* 11/70=06,PDQ=07,Q=10
835 ;* BIT 10=REAL TIME CLOCK
836 ;* BIT 9=FLOATING POINT PROCESSOR
837 ;* BIT 8=MEMORY MANAGEMENT
838 001256 000 $MAMS1: .BYTE AMAMS1 ;:HIGH ADDRESS,M.S. BYTE
839 001257 000 $MTYP1: .BYTE AMTYP1 ;:MEM. TYPE,BLK#1
840 ;* MEM.TYPE BYTE -- (HIGH BYTE)
841 ;* 900 NSEC CORE=001
842 ;* 300 NSEC BIPOLAR=002
843 ;* 500 NSEC MOS=003
844 001260 000000 $MADR1: .WORD AMADR1 ;:HIGH ADDRESS,BLK#1
845 ;* MEM.LAST ADDR.-3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE
846 001262 $ETEND:
847 .MEXIT
848
849 001262 000000 TESTNO: .WORD 0 ;:HOLDS TEST NUMBER FOR TYPEOUTS
850 001264 000000 WASR6: .WORD 0 ;:USED TO STORE THE STACK POINTER AFTER A TRAP
851 001266 000000 TRAPPC: .WORD 0 ;:USED TO STORE THE PC OF A TRAP OR ABORT
852 001270 000000 TRAPPS: .WORD 0 ;:USED TO STORE THE PS OF A TRAP OR ABORT
853 001272 000000 WASSR0: .WORD 0 ;:USED TO STORE CONTENTS OF SR0
854 001274 000000 WASSR2: .WORD 0 ;:USED TO STORE CONTENTS OF SR2
855 001276 000000 TBITPS: .WORD 0 ;:SAVES THE PSW THAT MAY HAVE ITS T-BIT ON
856 001300 000000 ANDADR: .WORD 0 ;:HOLDS RESULT OF ADDRESSES BEING AND-ED
857 001302 000000 ORADR: .WORD 0 ;:HOLDS RESULT OF ADDRESSES BEING OR-ED
858 001304 000000 TONUM: .WORD 0 ;:HOLDS NUMBER OF TIME-OUTS
859 001306 000000 VIRT1: .WORD 0 ;:HOLDS VIRTUAL ADDRESS TO BE CONVERTED
860 001310 000000 VIRT2: .WORD 0 ;:
861 001312 000000 PBALO: .WORD 0 ;:HOLDS BITS <15:00> OF PHYSICAL ADDRESS
862 001314 000000 PBAHI: .WORD 0 ;:HOLDS BITS <17:16> OF PHYSICAL ADDRESS
    
```

```

863      .SBTTL  ERROR POINTER TABLE
864
865      ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
866      ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
867      ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
868      ;*NOTE1:      IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
869      ;*NOTE2:      EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
870
871      ;*      EM      ;;POINTS TO THE ERROR MESSAGE
872      ;*      DH      ;;POINTS TO THE DATA HEADER
873      ;*      DT      ;;POINTS TO THE DATA
874      ;*      DF      ;;POINTS TO THE DATA FORMAT
875
876
877      001316      $ERRTB:
878
879      ;*ITEM 1
880      001316      040740      EM1      ;UNEXPECTED CPU TRAP TO LOC. 004
881      001320      044031      DH1      ;OLD PC OLD PSW R6 WAS TESTNO ERRORPC
882      001322      047236      DT1      ;TRAPPC, TRAPPS, WASR6, TESTNO, $ERRPC, 0
883      001324      050046      DF1      ;0,0,0,0,0
884
885      ;*ITEM 2
886      001326      041000      EM2      ;UNEXPECTED MEM. MGMT. TRAP TO LOC. 250
887      001330      044101      DH2      ;OLD PC OLD PSW R6 WAS SR0 SR2 TESTNO ERRORPC
888      001332      047252      DT2      ;TRAPPC, TRAPPS, WASR6, WASSR0, WASSR2, TESTNO, $ERRPC,
889      001334      050053      DF2      ;0,0,0,0,0,0,0
890
891      ;*ITEM 3
892      001336      041047      EM3      ;PRIORITY BITS SET WRONG IN PSW
893      001340      044171      DH3      ;WROTE READ TESTNO ERRORPC
894      001342      047272      DT3      ;$REG0,$REG1,TESTNO,$ERRPC,0
895      001344      050062      DF3      ;0,0,0,0
896
897      ;*ITEM 4
898      001346      041106      EM4      ;MODE BITS SET WRONG IN PSW
899      001350      044171      DH3      ;WROTE READ TESTNO ERRORPC
900      001352      047272      DT3      ;$REG0,$REG1,TESTNO,$ERRPC,0
901      001354      050062      DF3      ;0,0,0,0
902
903      ;*ITEM 5
904      001356      041141      EM5      ;DUAL ADDRESSING BETWEEN HI&LO BYTES OF PSW
905      001360      044171      DH3      ;WROTE READ TESTNO ERRORPC
906      001362      047272      DT3      ;$REG0,$REG1,TESTNO,$ERRPC,0
907      001364      050062      DF3      ;0,0,0,0
908
909      ;*ITEM 6
910      001366      041214      EM6      ;KERNEI R6 CHANGED BY WRITING USER R6
911      001370      044171      DH3      ;WROTE READ TESTNO ERRORPC
912      001372      047272      DT3      ;$REG0,$REG1,TESTNO,$ERRPC,0
913      001374      050062      DF3      ;0,0,0,0
914
915      ;*ITEM 7
916      001376      041261      EM7      ;A MEMORY MGMT. REG. TIMED OUT
917      001400      044231      DH7      ;ADDRESS TESTNO ERRORPC
918      001402      047304      DT7      ;$REG0,TESTNO,$ERRPC,0
    
```

919	001404	050066	DF7	:0,0,0
920				
921			:*ITEM 10	
922	001406	041317	EM10	:SUMMARY OF MEM. MGMT. REG. TIMEOUTS
923	001410	044261	DH10	:REGISTER-ADDRS NUM. OF
924				:AND-ED OR-ED TIMOUTS TESTNO ERRORPC
925	001412	047314	DT10	:ANDADR,ORADR,TONUM,TESTNO,\$ERRPC,0
926	001414	050071	DF10	:0,0,1,0,0
927				
928			:*ITEM 11	
929	001416	041363	FM11	:MEM. MGMT. REG. WOULD NOT CLEAR
930	001420	044361	DH11	:REGISTR READ READ-(BINARY)
931				:ADDRESS (OCTAL) 5432109876543210 TESTNO ERRORPC
932	001422	047330	DT11	:\$REG0,\$REG1,\$REG1,TESTNO,\$ERRPC,0
933	001424	050076	DF11	:0,0,2,0,0
934				
935			:*ITEM 12	
936	001426	041423	EM12	:MEM. MGMT. REG. BITS NOT SET CORRECTLY
937	001430	044501	DH12	:REGISTR WROTE READ READ
938				:ADDRESS (OCTAL) (OCTAL) (BINARY) TESTNO ERRORPC
939	001432	047344	DT12	:\$REG0,\$REG1,\$REG2,\$REG2,TESTNO,\$ERRPC,0
940	001434	050103	DF12	:0,0,0,2,0,0
941				
942			:*ITEM 13	
943	001436	041472	EM13	:SR0 EFFECTED BY WRITE TO PSW
944	001440	044641	DH13	:READ TESTNO ERRORPC
945	001442	047362	DT13	:\$REG0,TESTNO,\$ERRPC,0
946	001444	050111	DF13	:0,0,0
947				
948			:*ITEM 14	
949	001446	041527	EM14	:SR1 DID NOT READ ALL ZEROS
950	001450	044641	DH13	:READ TESTNO ERRORPC
951	001452	047362	DT13	:\$REG0,TESTNO,\$ERRPC,0
952	001454	050111	DF13	:0,0,0
953				
954			:*ITEM 15	
955	001456	041562	EM15	:DUAL ADDRESSING BETWEEN BYTES OF PAR OR PDR
956	001460	044501	DH12	:REGISTER WROTE READ READ
957				:ADDRESS (OCTAL) (OCTAL) (BINARY) TESTNO ERRORPC
958	001462	047364	DT12	:\$REG0,\$REG1,\$REG2,\$REG2,TESTNO,\$ERRPC,0
959	001464	050103	DF12	:0,0,0,2,0,0
960				
961			:*ITEM 16	
962	001466	041636	EM16	:DUAL ADDRESSING BETWEEN PAR-PDR'S
963	001470	044671	DH16	:PAR-PDR PAR-PDR
964				:CLEARED EFFECTD EXPECTD RECEIVD TESTNO ERRORPC
965	001472	047372	DT16	:\$REG0,\$REG1,\$REG5,\$REG2,TESTNO,\$ERRPC,0
966	001474	050114	DF16	:0,0,0,0,0,0
967				
968			:*ITEM 17	
969	001476	041700	EM17	:PHYS. ADDR. FORMED READ WRONG
970	001500	044771	DH17	:PHYSICAL VIRTUAL
971				:ADDRESS ADDRESS KIPAR4 TESTNO ERRORPC
972	001502	047410	DT17	:PBALO,VIRT1,\$REG4,TESTNO,\$ERRPC,0
973	001504	050122	DF17	:3,0,0,0,0
974				

Line	Code	Address	Item	Register	Description
975			*ITEM 20		
976	001506	041736		EM20	:PHYS. ADDR. FORMED READ WPONG IN RELOCATE MODE
977	001510	045061		DH20	:PHYSICL PAR 4 PAR 5
978					:ADDRESS VBA VBA PAR 4 PAR 5 PSW TESTNO
979	001512	047424		DT20	:PBALO,VIRT1,VIRT2,\$REG4,\$REG5,\$TMP0,TESTNO,\$ERRPC,0
980	001514	050127		DF20	:3,0,0,0,0,0,0,0
981					
982			*ITEM 21		
983	001516	042010		EM21	:W-BIT DID NOT GET SET IN PDR
984	001520	045207		DH21	:PDR VIRTUAL
985					:TESTED ADDRESS TESTNO ERRORPC
986	001522	047446		DT21	:\$REG5,\$REG3,TESTNO,\$ERRPC,0
987	001524	050137		DF21	:0,0,0,0
988					
989			*ITEM 22		
990	001526	042045		EM22	:W-BIT SET IN MORE THAN ONE PDR
991	001530	045267		DH22	:PDR IN PDR VIRTUAL
992					:ERROR TESTED ADDRESS TESTNO ERRORPC
993	001532	047460		DT22	:\$REG0,\$REG5,\$REG3,TESTNO,\$ERRPC,0
994	001534	050143		DF22	:0,0,0,0,0
995					
996			*ITEM 23		
997	001536	042104		EM23	:W-BIT NOT CLEARED BY WRITING TO PDR
998	001540	045366		DH23	:PDR TESTNO ERRORPC
999	001542	047474		DT23	:\$REG5,TESTNO,\$ERRPC,0
1000	001544	050150		DF23	:0,0,0
1001					
1002			*ITEM 24		
1003	001546	042150		EM24	:WRITING SRO SET W-BIT IN KIPDR7
1004	001550	045416		DH24	:PDR WAS EXPECTD TESTNO ERRORPC
1005	001552	047504		DT24	:\$REG2,\$REG1,TESTNO,\$ERRPC,0
1006	001554	050153		DF24	:0,0,0,0
1007					
1008			*ITEM 25		
1009	001556	042210		EM25	:W-BIT GOT SET DURING TIMEOUT ABORT
1010	001560	045416		DH24	:PDR WAS EXPECTD TESTNO ERRORPC
1011	001562	047504		DT24	:\$REG2,\$REG1,TESTNO,\$ERRPC,0
1012	001564	050153		DF24	:0,0,0,0
1013					
1014			*ITEM 26		
1015	001566	042253		EM26	:MEMORY MGMT. ACCESS ABORT DID NOT OCCUR
1016	001570	045456		DH26	:PDR 4 PSW TESTNO ERRORPC
1017	001572	047516		DT26	:\$REG2,\$TMP0,TESTNO,\$ERRPC,0
1018	001574	050153		DF24	:0,C,0,0
1019					
1020			*ITEM 27		
1021	001576	042323		EM27	:ACCESS ERROR DID NOT ABORT INSTRUCTION
1022	001600	045456		DH26	:PDR 4 PSW TESTNO ERRORPC
1023	001602	047516		DT26	:\$REG2,\$TMP0,TESTNO,\$ERRPC,0
1024	001604	050153		DF24	:0,0,0,0
1025					
1026			*ITEM 30		
1027	001606	042372		EM30	:SRO DID NOT REPORT ACCESS ERROR CORRECTLY
1028	001610	045516		DH30	:SRO WAS EXPECTD PDR 4 PSW TESTNO ERRORPC
1029	001612	047530		DT30	:WASSRO,\$REG3,\$REG2,\$TMP0,TESTNO,\$ERRPC,0
1030	001614	050157		DF30	:0,0,0,0,0,0

1031					
1032			:*ITEM 31		
1033	001616	042444	EM31	:SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR.	
1034	001620	045576	DH31	:SR2 WAS EXPECTD PDR 4 PSW TESTNO ERRORPC	
1035	001622	047546	DT31	:WASSR2,\$REG4,\$REG2,\$TMP0,TESTNO,\$ERRPC,0	
1036	001624	050157	DF30	:0,0,0,0,0,0	
1037					
1038			:*ITEM 32		
1039	001626	042511	EM32	:PAGE LGTH. ABORT OCCURRED WHEN IT SHOULDN'T HAVE	
1040	001630	045656	DH32	:V.B.A. KIPDR4 SRO WAS SR2 WAS TESTNO ERRORPC	
1041	001632	047564	DT32	:\$REG0,\$REG4,WASSRO,WASSR2,TESTNO,\$ERRPC,0	
1042	001634	050157	DF30	:0,0,0,0,0,0	
1043					
1044			:*ITEM 33		
1045	001636	042572	EM33	:PAGE LGTH. ABORT DID NOT OCCUR WHEN IT SHOULD HAVE	
1046	001640	045736	DH33	:V.B.A. KIPDR4 TESTNO ERRORPC	
1047	001642	047602	DT33	:\$REG0,\$REG4,TESTNO,\$ERRPC,0	
1048	001644	050153	DF24	:0,0,0,0	
1049					
1050			:*ITEM 34		
1051	001646	042655	EM34	:SRO DID NOT REPORT PAGE LGTH. ABORT CORRECTLY	
1052	001650	045776	DH34	:V.B.A. KIPDR4 SRO WAS EXPECTD TESTNO ERRORPC	
1053	001652	047614	DT34	:\$REG0,\$REG4,WASSRO,\$REG2,TESTNO,\$ERRPC,0	
1054	001654	050157	DF30	:0,0,0,0,0,0	
1055			:*ITEM 35		
1056	001656	042444	EM31	:SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR.	
1057	001660	046056	DH35	:V.B.A. KIPDR4 SR2 WAS EXPECTD TESTNO ERRORPC	
1058	001662	047632	DT35	:\$REG0,\$REG4,WASSR2,\$REG3,TESTNO,\$ERRPC,0	
1059	001664	050157	DF30	:0,0,0,0,0,0	
1060					
1061			:*ITEM 36		
1062	001666	042444	EM31	:SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR.	
1063	001670	046136	DH36	:SR2 WAS EXPECTD TESTNO ERRORPC	
1064	001672	047650	DT36	:WASSR2,\$REG1,TESTNO,\$ERRPC,0	
1065	001674	050153	DF24	:0,0,0,0	
1066					
1067			:*ITEM 37		
1068	001676	042733	EM37	:SRO OR SR2 CHANGED BY A SECOND ABORT	
1069	001700	046176	DH37	:FIRST ABORT SECOND ABORT	
1070				:SRO WAS SR2 WAS SRO WAS SR2 WAS TESTNO ERRORPC	
1071	001702	047662	DT37	:\$TMP0,\$TMP2,WASSRO,WASSR2,TESTNO,\$ERRPC,0	
1072	001704	050157	DF30	:0,0,0,0,0,0	
1073					
1074			:*ITEM 40		
1075	001706	043000	EM40	:SRO OR SR2 WAS NOT 'RESET' BY A RESET	
1076	001710	046313	DH40	:SRO WAS SR2 WAS TESTNO ERRORPC	
1077	001712	047700	DT40	:WASSRO,WASSR2,TESTNO,\$ERRPC,0	
1078	001714	050153	DF24	:0,0,0,0	
1079					
1080			:*ITEM 41		
1081	001716	043047	EM41	:SR2 NOT TRACKING CORRECTLY	
1082	001720	046136	DH36	:SR2 WAS EXPECTD TESTNO ERRORPC	
1083	001722	047650	DT36	:WASSR2,\$REG1,TESTNO,\$ERRPC,0	
1084	001724	050153	DF24	:0,0,0,0	
1085					
1086			:*ITEM 42		



1087	001726	043102	EM42	:DID NOT TRAP THRU KERNEL SPALE
1088	001730	046353	DH42	:PSW WAS R6 WAS TESTNO ERRORPC
1089	001732	047712	DT42	:\$REG1,\$REG2,TESTNO,\$ERRPC,0
1090	001734	050153	DF24	:0,0,0,0
1091				
1092			:*ITEM 43	
1093	001736	043141	EM43	:KT ERROR NOT SERVICED ON TIMEOUT ERROR
1094	001740	045366	DH23	:PDR TESTNO ERRORPC
1095	001742	047474	DT23	:\$REG5,TESTNO,\$ERRPC,0
1096	001744	050150	DF23	:0,0,0
1097				
1098			:*ITEM 44	
1099	001746	043210	EM44	:SRO OR SR2 CHANGED BY TIMEOUT ERROR
1100	001750	046413	DH44	:EXPECTED RECEIVED
1101				:SRO SR2 SRO WAS SR2 WAS TESTNO ERRORPC
1102	001752	047724	DT44	:\$REG0,\$REG1,WASSRO,WASSR2,TESTNO,\$ERRPC,0
1103	001754	050157	DF30	:0,0,0,0,0,0
1104				
1105			:*ITEM 45	
1106	001756	043254	EM45	:ERROR DURING 'DOUBLE ERROR' (KT & ODD ADDR.)
1107	001760	046525	DH45	:EXPECTED:
1108				:PSW PC SRO SR2
1109				:170017 (3\$+4) 020147 (3\$)
1110				:RECEIVED
1111				:PSW PC SRO SR2 TESTNO ERRORPC
1112	001762	047742	DT45	:\$REG1,\$REG3,WASSRO,WASSR2,TESTNO,\$ERRPC,0
1113	001764	050157	DF30	:0,0,0,0,0,0
1114				
1115			:*ITEM 46	
1116	001766	043327	EM46	:MFPI INSTRUCTION PUSHED WRONG DATA
1117	001770	046722	DH46	:DATA DATA
1118				:EXPECTD RECEIVD TESTNO ERRORPC
1119	001772	047760	DT46	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
1120	001774	050165	DF46	:0,0,0,0
1121				
1122			:*ITEM 47	
1123	001776	043372	EM47	:MTPI INSTRUCTION LOADED WRONG DATA
1124	002000	046722	DH46	:DATA DATA
1125				:EXPECTD RECEIVD TESTNO ERRORPC
1126	002002	047760	DT46	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
1127	002004	050165	DF46	:0,0,0,0
1128				
1129			:*ITEM 50	
1130	002006	043435	EM50	:STACK NOT PUSHED BY MFPI-MTPI
1131	002010	046777	DH50	:TESTNO ERRORPC
1132	002012	047772	DT50	:TESTNO,\$ERRPC,0
1133	002014	050171	DF50	:0,0
1134				
1135			:*ITEM 51	
1136	002016	043473	EM51	:KERNEL PAGE ACCESSED INSTEAD OF USER: MFPI-MTPI
1137	002020	047017	DH51	:SRO WAS SR2 WAS TESTNO ERRORPC
1138	002022	050000	DT51	:WASSRO,WASSR2,TESTNO,\$ERRPC,0
1139	002024	050173	DF51	:0,0,0,0
1140				
1141			:*ITEM 52	
1142	002026	043551	EM52	:WRONG PDR'S REFERENCED WHILE IN RELOCATE MODE

1143	002030	047057	DH52	:PHYSICL PAR 4
1144				:ADDRESS V.B.A. PAR 4 SRO WAS SR2 WAS PSW TESTNO
1145	002032	050012	DT52	:PBALO,VIRT1,\$REG4,WASSRO,WASSR2,\$TMPO,TESTNO,\$ERRPC,0
1146	002034	050177	DF52	:3,0,0,0,0,0,0,0
1147				
1148	002036	043627	:*ITEM 53	
1149	002040	046722	EM53	:MFPD INSTRUCTION PUSHED WRONG DATA
1150			DH46	:DATA DATA
1151	002042	047760	DT46	:EXPECTD RECEIVD TESTNO ERRORPC
1152	002044	050165	DF46	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
1153				:0,0,0,0
1154				
1155	002046	043672	:*ITEM 54	
1156	002050	046777	EM54	:STACK NOT PUSHED BY MFPD-MTPD
1157	002052	047772	DH50	:TESTNO ERRORPC
1158	002054	050171	DT50	:TESTNO,\$ERRPC,0
1159			DF50	:0,0
1160				
1161	002056	043730	:*ITEM 55	
1162	002060	044361	EM55	:PAR OR PDR WAS CHANGED BY A RESET
1163			DH11	:REGISTR READ READ-(BINARY)
1164	002062	047330	DT11	:ADDRESS (OCTAL) 5432109876543210 TESTNO ERRORPC
1165	002064	050076	DF11	:\$REG0,\$REG1,\$REG1,TESTNO,\$ERRPC,0
1166				:0,0,2,0,0
1167				
1168	002066	043766	:*ITEM 56	
1169	002070	047175	EM56	:PSW CHANGED BY AN RTI IN USER MODE
1170	002072	050034	DH56	:PSW WAS EXPECTD TESTNO ERRORPC
1171	002074	050207	DT56	:\$REG1,\$REG2,TESTNO,\$ERRPC,0
1172			DF56	:0,0,0,0
1173				

1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229

002076 005227  
002100 177777  
002102 001403  
002104 005237 001226  
002110 000000  
  
002112 012637 001266  
002116 012637 001270  
002122 010637 001264  
002126 104001  
002130 012737 177777 002100  
002136 013746 001270  
002142 013746 001266  
002146 000006  
  
  
  
  
  
  
  
  
  
  
  
002150 005227  
002152 177777  
002154 001403  
002156 005237 001226  
002162 000000  
  
  
002164 012637 001266  
002170 012637 001270  
002174 010637 001264  
002200 013737 177572 001272  
002206 013737 177576 001274  
002214 042737 160000 177572  
002222 104002  
002224 012737 177777 002152

```

.SBTTL ***** TRAP HANDLING ROUTINES *****
.SBTTL CPU TRAP HANDLER ROUTINE
:*****
:
: THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS THRU
: 'ERRVEC' (LOC. 004). IF THIS SUBROUTINE IS ENTERED BY A
: SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A HALT IS
: EXECUTED.
:*****
TIMERR: INC      (PC)+          ;MAKE FLAG ZERO IF FIRST TIME THRU
TIMFLG: .WORD   -1            ;NEGATIVE ONE FOR 'HAVE ENTERED' FLAG
        BEQ     1$            ;BRANCH IF FIRST TIME IN
        INC     $MSGTYPE      ;TELL APT THERE WAS AN ERROR
        HALT                    ;STOP! - I'VE ENTERED THIS ROUTINE
:                               ;A SECOND TIME BEFORE I FINISHED
:                               ;REPORTING THE FIRST ERROR. THE
:                               ;SECOND ENTRY ADDRESS SHOULD BE ON
:                               ;THE KERNEL STACK
1$:     MOV     (KSP)+,TRAPPC   ;SAVE PC+2 AT TIME OF ABORT
        MOV     (KSP)+,TRAPPS  ;SAVE PS AT TIME OF ABORT
        MOV     KSP,WASR6      ;SAVE STACK POINTER VALUE
        ERROR   1             ;UNEXPECTED TRAP OR ABORT TO LOC. 4
        MOV     #-1,TIMFLG     ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
        MOV     TRAPPS,-(KSP)  ;PUT PC & PS OF TRAP ON STACK
        MOV     TRAPPC,-(KSP)
        RTT                    ;RETURN FROM INTERRUPT OR ABORT

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

1230	002232	013746	001270	MOV	TRAPPS,-(KSP)	;PUT PC & PS OF TRAP ON STACK
1231	002236	013746	001266	MOV	TRAPPC,-(KSP)	
1232	002242	000006		RTT		;RETURN FROM INTERRUPT OR ABORT.
1233						

```

1234          .SBTTL
1235          .SBTTL ***** STARTING POINT OF TEST *****
1236          .SBTTL ***** STARTING ADDRESS OF 200 *****
1237          . 20000
1238
1239 020000    START:
1240          .SBTTL INITIALIZE THE COMMON TAGS
1241          ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
1242 020000    012706 001100    MOV    # $CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
1243 020004    005026          CLR    (R6)+           ;;CLEAR MEMORY LOCATION
1244 020006    022706 001140    CMP    #SWR,R6      ;;DONE?
1245 020012    001374          BNE   -6             ;;LOOP BACK IF NO
1246 020014    012706 001100    MOV    #STACK,SP     ;;SETUP THE STACK POINTER
1247          ;;INITIALIZE A FEW VECTORS
1248 020020    012737 034152 000020    MOV    # $SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
1249 020026    012737 000340 000022    MOV    #340,@#IOTVEC+2 ;;LEVEL 7
1250 020034    012737 034432 000030    MOV    # $ERROR,@#EMTVEC ;;EMT VECTOR FOR ERRCR ROUTINE
1251 020042    012737 000340 000032    MOV    #340,@#EMTVEC+2 ;;LEVEL 7
1252 020050    012737 040424 000034    MOV    # $TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1253 020056    012737 000340 000036    MOV    #340,@#TRAPVEC+2;LEVEL 7
1254 020064    012737 040512 000024    MOV    # $PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
1255 020072    012737 000340 000026    MOV    #340,@#PWRVEC+2 ;;LEVEL 7
1256 020100    013737 033714 033706    MOV    $ENDCT,$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
1257 020106    005037 001212          CLR    $TIMES        ;;INITIALIZE NUMBER OF ITERATIONS
1258 020112    005037 001214          CLR    $ESCAPE      ;;CLEAR THE ESCAPE ON ERROR ADDRESS
1259 020116    112737 000001 001115    MOV    #1,$ERMAX     ;;ALLOW ONE ERROR PER TEST
1260          ;;INITIALIZE THE 'T-BIT' TRAP VECTOR. THEN LOAD LOCATION '$RTRN', IN
1261          ;;THE 'END-OF-PASS' ($EOP) ROUTINE, WITH A 'RTI' OR 'RTT'.
1262 020124    012737 034136 000014    MOV    # $RTRN,@#TBITVEC ;;SET 'T' BIT VECTOR TO $RTRN
1263 020132    012737 000340 000016    MOV    #340,@#TBITVEC+2 ;;LEVEL 7
1264 020140    012737 000002 034136    MOV    #RTI,$RTRN    ;;SET $RTRN TO A RTI
1265 020146    012737 020174 000010    MOV    #65$,@#RESVEC ;;TRY TO DO A RTI
1266 020154    005046          CLR    -(SP)         ;;DUMMY PS
1267 020156    012746 020164          MOV    #64$,-(SP)   ;;AND PC
1268 020162    000006          RTI    ;            ;;TRY THE RTI
1269 020164    012737 000006 034136 64$: MOV    #RTT,$RTRN    ;;RTI IS LEGAL--SET $RTRN TO A RTT
1270 020172    000402          BR    66$           ;;
1271 020174    062706 000010 65$: ADD    #10,SP      ;;RTI ILLEGAL--CLEAN OFF THE STACK
1272 020200    012737 000012 000010 66$: MOV    #RESVEC+2,@#RESVEC ;;RESTORE TRAP CATCHER
1273 020206    005037 034144          CLR    $TBIT        ;;CLEAR 'T' BIT SWITCH
1274 020212    012737 020212 001106    MOV    #.,$LPADR     ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1275 020220    012737 020220 001110    MOV    #.,$LPERR     ;;SETUP THE ERROR LOOP ADDRESS
1276          ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1277          ;;EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
1278 020226    013746 000004          MOV    @#ERRVEC,-(SP) ;;SAVE ERROR VECTOR
1279 020232    012737 020266 000004    MOV    #67$,@#ERRVEC ;;SET UP ERROR VECTOR
1280 020240    012737 177570 001140    MOV    #DSWR,SWR     ;;SETUP FOR A HARDWARE SWICH REGISTER
1281 020246    012737 177570 001142    MOV    #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
1282 020254    022777 177777 160656    CMP    #-1,@SWR     ;;TRY TO REFERENCE HARDWARE SWR
1283 020262    001012          BNE   69$           ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1284          ;;AND THE HARDWARE SWR IS NOT -1
1285 020264    000403          BR    68$           ;;BRANCH IF NO TIMEOUT
1286 020266    012716 020274 67$: MOV    #68$,(SP)     ;;SET UP FOR TRAP RETURN
1287 020272    000002          RTI    ;            ;;
1288 020274    012737 000176 001140 68$: MOV    #SWREG,SWR    ;;POINT TO SOFTWARE SWR
1289 020302    012737 000174 001142    MOV    #DISPREG,DISPLAY
    
```



```

1290 020310 012637 000004      69$:  MOV      (SP)+,@ERRVEC  ;;RESTORE ERROR VECTOR
1291
1292 020314 005037 001234      CLR      $PASS              ;;CLEAR PASS COUNT
1293 020320 132737 000200 001247  BITB     #APTSIZE,$ENVM     ;;TEST USER SIZE UNDER APT
1294 020326 001403              BEQ      70$                ;;YES,USE NON-APT SWITCH
1295 020330 012737 001250 001140  MOV      #$$SWREG,$SWR     ;;NO,USE APT SWITCH REGISTER
1296 020336
1297                                70$:
1298 020336 005037 001112      ;; INITIALIZE THE ERROR COUNTER FOR EOP REPORT($ERTTL).
1299                                CLR      $ERTTL              ;;CLEAR ERROR COUNTER
1300                                .SBTTL  TYPE PROGRAM NAME
1301                                ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
1302                                INC      #-1                      ;;FIRST TIME?
1303 020342 005227 177777      BNE      71$                ;;BRANCH IF NO
1304 020344 001051              CMP      #SENDAD,@#42     ;;ACT-11?
1305 020350 022737 034072 000042  BEQ      71$                ;;BRANCH IF YES
1306 020356 001445              TYPE     ,72$              ;;TYPE ASCIZ STRING
1307 020360 104401 020426      .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
1308 020364 005737 000042      TST     @#42                ;;ARE WE RUNNING UNDER XNDP/ACT?
1309 020370 001012              BNE      73$                ;;BRANCH IF YES
1310 020372 123727 001246 000001  CMPB    $ENV,#1            ;;ARE WE RUNNING UNDER APT?
1311 020400 001406              BEQ      73$                ;;BRANCH IF YES
1312 020402 023727 001140 000176  CMP     $SWR,#$SWREG      ;;SOFTWARE SWITCH REG SELECTED?
1313 020410 001005              BNE      74$                ;;BRANCH IF NO
1314 020412 104407              GTSWR   74$                ;;GET SOFT-SWR SETTINGS
1315 020414 000403              BR      74$
1316 020416 112737 000001 001134  73$:  MOVB    #1,$AUTOB         ;;SET AUTO-MODE INDICATOR
1317 020424 000422              74$:
1318                                BR      71$                ;;GET OVER THE ASCIZ
1319 020472      ;;72$: .ASCIZ <CRLF>#MD-11-CJKDA-B KTF11-AA MMU DIAG.#<CRLF>
1320      71$:
1321                                RESTRT:
1322
1323                                ;;*****
1324                                ;;THE FOLLOWING TURNS ON A TIMER ON THE MULTI-OPTION TESTER
1325                                ;;IN MANUFACTURING.
1326
1327 020472 012737 020514 000004  TIMEON: MOV    #LOOP,@#4          ;;SETUP RETURN ADDRESS IN CASE OF TIMEOUT
1328 020500 012737 000340 000006      MOV    #340,@#6           ;;SETUP PSW
1329 020506 012737 000002 164000  MOV    #2,@#164000        ;;SET START BIT IN MULTI-TESTER
1330
1331                                ;;*****
1332 020514 012706 001100      LOOP:  MOV    #STACK,$KSP    ;;INITIALIZE THE STACK POINTER
1333 020520 012737 002076 000004      MOV    #TIMERR,$ERRVEC     ;;LOAD CPU SERVICE ROUTINE INTO TRAP VECTOR
1334 020526 012737 000340 000006      MOV    #340,$ERRVEC+2     ;;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
1335 020534 012737 002150 000250      MOV    #MGMERR,$MMVEC     ;;LOAD MEMORY MANAGENT ROUTINE INTO VECTOR
1336 020542 012737 000340 000252      MOV    #340,$MMVEC+2     ;;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
1337 020550 012700 177777      MOV    #-1,$RO            ;;PUT -1 INTO RO TO INITIALIZE FLAGS
1338 020554 010037 002100      MOV    $RO,$TIMFLG        ;;INITIALIZE CPU ERROR FLAG
1339 020560 010037 002152      MOV    $RO,$MGMFLG        ;;INITIALIZE MEMORY MANAGEMENT ERROR FLAG
1340 020564 012737 000340 001276      MOV    #340,$TBITPS       ;;INITIALIZE LOG THAT HOLDS T-BIT PSW
1341 020572 005037 177572      CLR    $SRO                ;;BE SURE MEM. MGMT IS OFF TO START WITH
1342
    
```

1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398

020576 000004  
020600 012737 020610 001110  
020606 005000  
020610 005001  
020612 106400  
020614 106701  
020616 042701 177437  
020622 020001  
020624 001401  
020626 104003  
  
020630 062700 000040  
020634 022700 000400  
020640 001363  
020642 012737 020600 001110  
  
020650 000004  
020652 012737 020662 001110  
020660 005000  
020662 005037 177776  
020666 050037 177776  
020672 013701 177776  
020676 042701 007777  
020702 020001  
020704 001403  
020706 005037 177776  
020712 104004  
  
020714 062700 010000  
020720 001360  
020722 012737 020652 001110  
020730 005037 177776

```
*****  
*TEST 1 PSW PRIORITY BIT TEST  
*  
* THIS TEST READS AND WRITES THE PROCESSOR STATUS WORD <7:5> 'PRIORITY BITS'  
* TO SEE THAT SOME OF THE BASIC 'DATA PATH' LOGIC IS WORKING.  
*****  
TST1: SCOPE  
1$: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$  
CLR R0 ;INITIALIZE R0 WITH PRIORITY=0 DATA  
2$: CLR R1 ;PREPARE R1 TO ACCEPT DATA READ  
MTPS R0 ;WRITE PRIORITY BITS IN THE PSW  
MFPS R1 ;READ BACK THE LOW BYTE OF PSW  
BIC #177437,R1 ;MASK OFF EVERYTHING EXCEPT PRIORITY BITS  
CMP R0,R1 ;WAS CORRECT PRIORITY SET IN THE PSW?  
BEQ 3$ ;BRANCH IF YES  
ERROR 3 ;PRIORITY BITS SET WRONG IN PSW  
;FOR TIGHTER SCOPE LOOP  
;REPLACE ERROR CALL WITH  
;'BR 2$' = 000770  
3$: ADD #40,R0 ;CHANGE DATA TO NEXT PRIORITY  
CMP #400,R0 ;HAVE PRIORITIES 0-7 ALL BEEN CHECKED?  
BNE 2$ ;BRANCH IF NO  
MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$  
  
*****  
*TEST 2 PSW MODE BIT TEST  
* THIS TEST READS AND WRITES THE PROCESSOR STATUS WORD <15:12> 'MODE BITS'  
*****  
TST2: SCOPE  
1$: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$  
CLR R0 ;INITIALIZE R0 WITH MODE BITS = 0000  
2$: CLR PSW ;INITIALIZE PSW  
BIS R0,PSW ;BIT SET THE PSW MODE BITS WITH R0  
MOV PSW,R1 ;READ BACK THE CONTENTS OF THE PSW  
BIC #007777,R1 ;MASK OFF EVERYTHING EXCEPT THE MODE BITS  
CMP R0,R1 ;WERE THE MODE BITS SET CORRECTLY?  
BEQ 3$ ;BRANCH IF YES  
CLR PSW ;CLEAR PSW FOR ERROR REPORT  
ERROR 4 ;MODE BITS SET WRONG IN PSW  
;FOR TIGHTER SCOPE LOOP  
;REPLACE ERROR CALL WITH  
;'BR 2$' = 000763  
3$: ADD #10000,R0 ;CHANGE MODE BIT DATA  
BNE 2$ ;BRANCH IF STILL MORE COMBINATIONS  
MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$  
CLR PSW ;RESET PSW BEFORE LEAVING  
  
*****  
*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.
```

```

1399
1400
1401 020734 000004
1402 020736 012737 020744 001110
1403 020744 005037 177776
1404 020750 012700 000360
1405 020754 110037 177777
1406 020760 013701 177776
1407 020764 042701 007437
1408 020770 000300
1409 020772 020001
1410 020774 001403
1411 020776 005037 177776
1412 021002 104005
1413
1414
1415
1416 021004 012737 021012 001110
1417 021012 005037 177776
1418 021016 012700 000340
1419 021022 110037 177776
1420 021026 013701 177776
1421 021032 042701 007437
1422 021036 020001
1423 021040 001403
1424 021042 005037 177776
1425 021046 104005
1426
1427
1428
1429 021050 012737 020736 001110
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440 021056 000004
1441 021060 005037 177776
1442 021064 012706 001100
1443 021070 012737 140000 177776
1444 021076 012706 000700
1445 021102 005037 177776
1446 021106 022706 001100
1447 021112 001404
1448 021114 012700 001100
1449 021120 010601
1450 021122 104006
1451
1452
1453
1454

:*****
:*****
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$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$

:*****
:*****
: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, USER R6 IS SET TO 700, THEN
: KERNEL R6 IS READ TO BE SURE ITS STILL 1100.
:*****
:*****
TST4: SCOPE
CLR PSW ;GO TO KERNEL MODE
MOV #KERSTK, KSP ;SET KERNEL STACK POINTER TO 1100
MOV #140000, PSW ;GO TO USER MODE
MOV #USESTK, USP ;SET USER STACK POINTER TO 700
CLR PSW ;BACK TO KERNEL MODE
CMP #KERSTK, KSP ;IS KERNEL R6 STILL 1100?
BEQ TST5 ;BRANCH IF KERNEL R6 IS OKAY
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

```

1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510

021124 000004  
021126 012737 021170 001110  
021134 012737 021232 000004  
021142 012700 177572  
021146 0127C1 000003  
021152 012737 177777 001300  
021160 005037 001302  
021164 005037 001304  
021170 005710  
021172 062700 000002  
021176 077104  
021200 005737 172516  
021204 012737 021126 001110  
021212 005737 001304  
021216 001401  
021220 104010  
021222 012737 002076 000004  
021230 000414  
021232 062706 000004  
021236 104007  
010002  
050237 001302  
005102  
040237 001300  
005237 001304

```

*****
*
* THE NEXT FIVE (5) TESTS WILL TRY TO ADDRESS ALL OF THE
* MEMORY MANAGEMENT REGISTERS (SR0,SR1,SR2,SR3,KERNEL & 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. STATUS REG. 1 IS NOT USED BUT SHOULD STILL
* RESPOND TO ITS UNIBUS ADDRESS. DATA WILL BE WRITTEN OR READ
* FROM THESE REGISTERS IN LATER TESTS, THIS TEST JUST CHECK
* FOR A RESPONSE.
*
    
```

```

*****
TST5:  SCOPE
    
```

```

1$:  MOV    #2$, $LPERR      ;SET LOOP ON ERROR POINTER TO 2$
      MOV    #5$, @#4        ;SET TIMEOUT VECTOR TO 5$
      MOV    #SR0,R0         ;LOAD R0 WITH ADDRESS OF FIRST REG.
      MOV    #3,R1           ;LOAD R1 WITH THE LOOP COUNT
      MOV    #-1,ANDADR      ;INITIALIZE 'AND' OF ADDRS. LOC.
      CLR    ORADR           ;INITIALIZE 'OR' OF ADDRS. LOC.
      CLR    TONUM           ;INITIALIZE 'TIMEOUTS' COUNTER
2$:  TST    (R0)             ;TRY ADDRESSING A STATUS REGISTER
      ;IF IT TIMES OUT GO TO 5$
3$:  ADD    #2,R0            ;PUT NEXT ADDRESS IN R0
      SOB   R1,2$           ;LOOP BACK TO 2$ UNTIL ALL TESTED
      TST   @#172516        ;CHECK SR3 FOR RESPONSE
      ;IF IT TIMES OUT GO TO 5$
      MOV    #1$, $LPERR     ;RESET LOOP ON ERROR POINTER TO 1$
      TST   TONUM           ;DID ANY OF THE STATUS REG.S TIMEOUT?
      BEQ   4$              ;BRANCH IF NO
      ERROR 10              ;SUMMARY OF STATUS REG. TIMEJUTS
4$:  MOV    #TIMERR,@#4      ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
      BR    TST6            ;GO TO NEXT TEST
5$:  ADD    #4,KSP           ;CLEAN UP THE STACK
      ERROR 7               ;ONE OF THE STATUS REGS. TIMED OUT
      ;FOR TIGHTER SCOPE LOOP
      ;REPLACE ERROR CALL WITH
      ;'BR 2$' = 000756
      MOV    R0,R2           ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
      BIS   R2,ORADR        ;'OR' IT WITH OTHER ADDRS. THAT TIMED OUT
      COM   R2              ;'AND' IT WITH OTHER ADDRS. THAT TIMED OUT
      BIC   R2,ANDADR       ;
      INC   TONUM           ;INCREMENT THE TIMEOUT COUNTER
    
```

```

1511 021260 000744          BR      3$          ;BRANCH BACK TO TEST THE NEXT ADDR.
1512
1513
1514          ;*****
1515          ;*TEST 6          KERNEL PAR'S TIMEOUT TEST
1516          ;*
1517          ;*          THIS TEST ADDRESSES THE EIGHT (8) KERNEL PAGE ADDRESS
1518          ;*          REGISTERS (KIPAR0-KIPAR7) AND CHECKS THAT SOMETHING
1519          ;*          RESPONDS TO THEIR ADDRESSES.
1520          ;*****
1521 021262 000004          TST6:  SCOPE
1522
1523 021264 012737 021326 001110 1$:  MOV      #2$, $LPERR          ;SET LOOP ON ERROR POINTER TO 2$
1524 021272 012737 021364 000004      MOV      #5$, @#4          ;SET TIMEOUT VECTOR TO 5$
1525 021300 012700 172340      MOV      #KIPAR0, R0      ;LOAD R0 WITH ADDRESS OF FIRST REG.
1526 021304 012701 000010      MOV      #10, R1         ;LOAD R1 WITH LOOP COUNT (8)
1527 021310 012737 177777 001300      MOV      #-1, ANDADR      ;INITIALIZE 'AND' OF ADDR. LOC
1528 021316 005037 001302      CLR      ORADR           ;INITIALIZE 'OR' OF ADDR. LOC.
1529 021322 005037 001304      CLR      TONUM          ;INITIALIZE 'TIMEOUTS' COUNTER
1530 021326 005710          2$:  TST      (R0)          ;TRY ADDRESSING A KIPAR
1531          ;IF IT TIMES OUT, WILL GO TO 5$
1532 021330 062700 000002          3$:  ADD      #2, R0          ;PUT NEXT KIPAR ADDRESS IN R0
1533 021334 077104          SOB      R1, 2$          ;LOOP BACK TO 2$ UNTIL ALL TESTED
1534 021336 012737 021264 001110      MOV      #1$, $LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
1535 021344 005737 001304          TST      TONUM          ;DID ANY OF THE KIPARS TIME OUT?
1536 021350 001401          BEQ      4$             ;BRANCH IF NO
1537 021352 104010          ERROR   10            ;SUMMARY OF KIPAR TIMEOUTS
1538 021354 012737 002076 000004 4$:  MOV      #TIMERR, @#4      ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1539 021362 000414          BR      TST7          ;GO TO NEXT TEST
1540
1541 021364 062706 000004          5$:  ADD      #4, KSP        ;CLEAN UP THE STACK
1542 021370 104007          ERROR   7             ;ONE OF THE KIPARS TIMED OUT
1543          ;FOR TIGHTER SCOPE LOOP
1544          ;REPLACE ERROR CALL WITH
1545          ;'BR 2$' = 000756
1546 021372 010002          MOV      R0, R2         ;LOAD THE ADDRESS THAT TIMED OUT INTO R?
1547 021374 050237 001302          BIS      R2, ORADR      ;'OR' IT WITH OTHER ADDRS. THAT TIMED OUT
1548 021400 005102          COM      R2            ;'AND' IT WITH OTHER ADDRS. THAT TIMED OUT
1549 021402 040237 001300          BIC      R2, ANDADR
1550 021406 005237 001304          INC      TONUM          ;INCREMENT THE TIMEOUT COUNTER
1551 021412 000746          BR      3$            ;BRANCH BACK TO TEST THE NEXT KIPAR
1552
1553          ;*****
1554          ;*TEST 7          KERNEL PDR'S TIMEOUT TEST
1555          ;*
1556          ;*          THIS TEST ADDRESSES THE EIGHT (8) KERNEL PAGE DESCRIPTOR
1557          ;*          REGISTERS (KIPDR0-KIPDR7) AND CHECKS THAT SOMETHING
1558          ;*          RESPONDS TO THEIR ADDRESSES.
1559          ;*
1560          ;*****
1561 021414 000004          TST7:  SCOPE
1562
1563 021416 012737 021460 001110 1$:  MOV      #2$, $LPERR          ;SET LOOP ON ERROR POINTER TO 2$
1564 021424 012737 021516 000004      MOV      #5$, @#4          ;SET TIMEOUT VECTOR TO 5$
1565 021432 012700 172300      MOV      #KIPDR0, R0      ;LOAD R0 WITH ADDRESS OF FIRST REG.
1566 021436 012701 000010      MOV      #10, R1         ;LOAD R1 WITH LOOP COUNT (8)
    
```

```

1567 021442 012737 177777 001300      MOV    #-1,ANDADR      ;INITIALIZE 'AND' OF ADDR. LOC
1568 021450 005037 001302              CLR    ORADR           ;INITIALIZE 'OR' OF ADDR. LOC.
1569 021454 005037 001304              CLR    TONUM          ;INITIALIZE 'TIMEOUTS' COUNTER
1570 021460 005710                2$:   TST    (R0)        ;TRY ADDRESSING A KIPDR
1571                                ;IF IT TIMES OUT, WILL GO TO 5$
1572 021462 062700 000002                3$:   ADD    #2,R0        ;PUT NEXT KIPDR ADDRESS IN R0
1573 021466 077104                SOB    R1,2$          ;LOOP BACK TO 2$ UNTIL ALL TESTED
1574 021470 012737 021416 001110      MOV    #1$,$LPERR     ;RESET LOOP ON ERROR POINTER TO 1$
1575 021476 005737 001304              TST    TONUM          ;DID ANY OF THE KIPDRS TIME OUT?
1576 021502 001401              BEQ    4$             ;BRANCH IF NO
1577 021504 104010              ERROR  10            ;SUMMARY OF KIPDR TIMEOUTS
1578 021506 012737 002076 000004      4$:   MOV    #TIMERR,@#4 ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1579 021514 000414              BR     TST10         ;;GO TO NEXT TEST
1580
1581 021516 062706 000004                5$:   ADD    #4,KSP       ;CLEAN UP THE STACK
1582 021522 104007              ERROR  7             ;ONE OF THE KIPDRS TIMED OUT
1583                                ;FOR TIGHTER SCOPE LOOP
1584                                ;REPLACE ERROR CALL WITH
1585                                ;'BR 2$' = 000756
1586 021524 010002              MOV    R0,R2         ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
1587 021526 050237 001302              BIS    R2,ORADR      ;'OR' IT WITH OTHER ADDRS. THAT TIMED OUT
1588 021532 005102              COM    R2            ;'AND' IT WITH OTHER ADDRS. THAT TIMED OUT
1589 021534 040237 001300              BIC    R2,ANDADR     ;
1590 021540 005237 001304              INC    TONUM         ;INCREMENT THE TIMEOUT COUNTER
1591 021544 000746              BR     3$            ;BRANCH BACK TO TEST THE NEXT KIPDR
1592
1593
1594
1595
1596
1597
1598
1599
1600

```

```

*****
;TEST 10      USER PAR'S TIMEOUT TEST
;
;      THIS TEST ADDRESSES THE EIGHT (8) USER PAGE ADDRESS
;      REGISTERS (UIPAR0-UIPAR7) AND CHECKS THAT SOMETHING
;      RESPONDS TO THEIR ADDRESSES.
*****

```

```

1601 021546 000004      TST10: SCOPE
1602
1603 021550 012737 021612 001110      1$:   MOV    #2$,$LPERR   ;SET LOOP ON ERROR POINTER TO 2$
1604 021556 012737 021650 000004      MOV    #5$,@#4       ;SET TIMEOUT VECTOR TO 5$
1605 021564 012700 177640              MOV    #UIPAR0,R0    ;LOAD R0 WITH ADDRESS OF FIRST REG.
1606 021570 012701 000010              MOV    #10,R1        ;LOAD R1 WITH LOOP COUNT (8)
1607 021574 012737 177777 001300      MOV    #-1,ANDADR    ;INITIALIZE 'AND' OF ADDR. LOC
1608 021602 005037 001302              CLR    ORADR         ;INITIALIZE 'OR' OF ADDR. LOC.
1609 021606 005037 001304              CLR    TONUM        ;INITIALIZE 'TIMEOUTS' COUNTER
1610 021612 005710                2$:   TST    (R0)        ;TRY ADDRESSING A UIPAR
1611                                ;IF IT TIMES OUT, WILL GO TO 5$
1612 021614 062700 000002                3$:   ADD    #2,R0        ;PUT NEXT UIPAR ADDRESS IN R0
1613 021620 077104                SOB    R1,2$          ;LOOP BACK TO 2$ UNTIL ALL TESTED
1614 021622 012737 021550 001110      MOV    #1$,$LPERR     ;RESET LOOP ON ERROR POINTER TO 1$
1615 021630 005737 001304              TST    TONUM        ;DID ANY OF THE UIPARS TIME OUT?
1616 021634 001401              BEQ    4$             ;BRANCH IF NO
1617 021636 104010              ERROR  10            ;SUMMARY OF UIPAR TIMEOUTS
1618 021640 012737 002076 000004      4$:   MOV    #TIMERR,@#4 ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1619 021646 000414              BR     TST11         ;;GO TO NEXT TEST
1620
1621 021650 062706 000004                5$:   ADD    #4,KSP       ;CLEAN UP THE STACK
1622 021654 104007              ERROR  7             ;ONE OF THE UIPARS TIMED OUT

```

```
1623                                     ;FOR TIGHTER SCOPE LOOP
1624                                     ;REPLACE ERROR CALL WITH
1625                                     ;'BR 2$' = 000756
1626 021656 010002                       MOV    R0,R2                       ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
1627 021660 050237 001302                BIS    R2,ORADR                    ;'OR' IT WITH OTHER ADDRS. THAT TIMED OUT
1628 021664 005102                       COM    R2                          ;'AND' IT WITH OTHER ADDRS. THAT TIMED OUT
1629 021666 040237 001300                BIC    R2,ANDADR
1630 021672 005237 001304                INC    TONUM                       ;INCREMENT THE TIMEOUT COUNTER
1631 021676 000746                       BR     3$                          ;BRANCH BACK TO TEST THE NEXT UIPAR
```

```
1632
1633                                     ;*****
1634                                     ;*TEST 11      USER PDR'S TIMEOUT TEST
1635                                     ;*
```

```
1636                                     ;*
1637                                     ;*      THIS TEST ADDRESSES THE EIGHT (8) USER PAGE DESCRIPTOR
1638                                     ;*      REGISTERS (UIPDR0-UIPDR7) AND CHECKS THAT SOMETHING
1639                                     ;*      RESPONDS TO THEIR ADDRESSES.
1640                                     ;*
```

```
1641                                     ;*****
1642 TST11:  SCOPE
```

```
1643 021702 012737 021744 001110 1$:  MOV    #2$,SLPERR                ;SET LOOP ON ERROR POINTER TO 2$
1644 021710 012737 022002 000004        MOV    #5$,@#4                    ;SET TIMEOUT VECTOR TO 5$
1645 021716 012700 177600                MOV    #UIPDR0,R0                 ;LOAD R0 WITH ADDRESS OF FIRST REG.
1646 021722 012701 000010                MOV    #10,R1                     ;LOAD R1 WITH LOOP COUNT (8)
1647 021726 012737 177777 001300        MOV    #-1,ANDADR                 ;INITIALIZE 'AND' OF ADDR. LOC
1648 021734 005037 001302                CLR    ORADR                       ;INITIALIZE 'OR' OF ADDR. LOC.
1649 021740 005037 001304                CLR    TONUM                       ;INITIALIZE 'TIMEOUTS' COUNTER
1650 021744 005710                2$:  TST    (R0)                    ;TRY ADDRESSING A UIPDR
1651                                     ;IF IT TIMES OUT, WILL GO TO 5$
1652 021746 062700 000002                3$:  ADD    #2,R0                    ;PUT NEXT UIPDR ADDRESS IN R0
1653 021752 077104                SOB    R1,2$                       ;LOOP BACK TO 2$ UNTIL ALL TESTED
1654 021754 012737 021702 001110        MOV    #1$,SLPERR                 ;RESET LOOP ON ERROR POINTER TO 1$
1655 021762 005737 001304                TST    TONUM                       ;DID ANY OF THE UIPDRS TIME OUT?
1656 021766 001401                BEQ    4$                          ;BRANCH IF NO
1657 021770 104010                ERROR 10                            ;SUMMARY OF UIPDR TIMEOUTS
1658 021772 012737 002076 000004 4$:  MOV    #TIMERR,@#4                ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1659 022000 000414                BR     TST12                       ;GO TO NEXT TEST
```

```
1660
1661 022002 062706 000004                5$:  ADD    #4,KSP                    ;CLEAN UP THE STACK
1662 022006 104007                ERROR 7                            ;ONE OF THE UIPDRS TIMED OUT
1663                                     ;FOR TIGHTER SCOPE LOOP
1664                                     ;REPLACE ERROR CALL WITH
1665                                     ;'BR 2$' = 000756
1666 022010 010002                       MOV    R0,R2                       ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
1667 022012 050237 001302                BIS    R2,ORADR                    ;'OR' IT WITH OTHER ADDRS. THAT TIMED OUT
1668 022016 005102                       COM    R2                          ;'AND' IT WITH OTHER ADDRS. THAT TIMED OUT
1669 022020 040237 001300                BIC    R2,ANDADR
1670 022024 005237 001304                INC    TONUM                       ;INCREMENT THE TIMEOUT COUNTER
1671 022030 000746                       BR     3$                          ;BRANCH BACK TO TEST THE NEXT UIPDR
```

```
1672
1673                                     ;*****
1674                                     ;*TEST 12      SR0(15:13) BIT TEST & SR2 TEST
1675                                     ;*
```

```
1676                                     ;*
1677                                     ;*      THIS TEST CHECKS BITS <15:13> OF STATUS REGISTER 0 TO SEE
1678                                     ;*      THAT EACH CAN BE SET AND CLEARED AND THAT A 'RESET' WILL
1679                                     ;*      CLEAR ALL OF THEM.
1680                                     ;*
```



```

1679 ;* THE REST OF BITS IN SR0 WILL BE CHECKED LATER.
1680 ;* ALSO CHECK THAT SR2 IS TRACKING WITH MEM. MGMT.
1681 ;* OFF BUT LOCKS UP WHEN ANY OF SR0 ERROR BITS SET.
1682 ;*
1683 ;*
1684 022032 000004 ;* *****
1685 TST12: SCOPE
1686 022034 012700 177572 1$: MOV #SR0,R0 ;LOAD ADDRESS OF SR0 INTO R0
1687 022040 012710 160000 MOV #160000,(R0) ;SET BITS <15:13> IN SR0 (ERROR BITS)
1688 022044 000005 RESET ;ISSUE AND 'INIT' SIGNAL
1689 022046 011001 MOV (R0),R1 ;READ SR0 INTO R1 TO SEE IF CLEAR
1690 022050 001404 BEQ 2$ ;BRANCH IF SR0<15:13> CLEARED BY 'INIT'
1691 022052 104011 ERROR 11 ;SR0<15:13> NOT CLEARED BY A 'RESET'
1692 ;FOR TIGHTER SCOPE LOOP
1693 ;REPLACE ERROR CALL WITH
1694 ;'BR 1$' = 000770
1695 022054 012737 022062 001110 MOV #2$,$LPERR ;SET LOOP ON ERROR POINTER TO 2$
1696 022062 013737 177576 001274 2$: MOV SR2,WASSR2 ;READ CONTENTS OF SR2
1697 022070 012701 022062 MOV #2$,R1 ;LOAD EXPECTED CONTENTS INTO R1
1698 022074 020137 001274 CMP R1,WASSR2 ;IS SR2 TRACKING?
1699 022100 001401 BEQ 3$ ;BRANCH IF YES
1700 022102 104041 ERROR 41 ;SR2 NOT 'TRACKING' VIRTUAL ADDRESSES
1701 ;FOR TIGHTER SCOPE LOOP
1702 ;REPLACE ERROR CALL WITH
1703 ;'BR 2$' = 000767
1704 022104 012737 022122 001110 3$: MOV #4$,$LPERR ;SET LOOP ON ERROR POINTER TO 4$
1705 022112 012701 100000 MOV #BIT15,R1 ;PUT DATA TO BE WRITTEN IN R1
1706 022116 012703 000003 MOV #3,R3 ;SETUP R3 AS A LOOP COUNTER
1707 022122 005010 4$: CLR (R0) ;CLEAR SR0
1708 022124 050110 5$: BIS R1,(R0) ;SET ONE OF THE ERROR BITS IN SR0
1709 022126 011002 MOV (R0),R2 ;READ SR0 INTO R2
1710 022130 020102 CMP R1,R2 ;DID RIGHT ERROR BIT GET SET?
1711 022132 001401 BEQ 6$ ;BRANCH IF YES
1712 022134 104012 ERROR 12 ;BITS WERE SET WRONG IN SR0
1713 ;FOR TIGHTER SCOPE LOOP
1714 ;REPLACE ERROR CALL WITH
1715 ;'BR 4$' = 000772
1716 022136 012704 022124 6$: MOV #5$,R4 ;LOAD EXPECTED CONTENTS OF SR2 IN R4
1717 022142 013737 177576 001274 MOV SR2,WASSR2 ;READ SR2
1718 022150 020437 001274 CMP R4,WASSR2 ;DID SR2 LOCK UP WHEN ERROR
1719 ;BIT SET IN SR1?
1720 022154 001401 BEQ 7$ ;BRANCH IF YES
1721 022156 104064 ERROR 64 ;SR2 DID NOT LOCK UP
1722 ;FOR TIGHTER SCOPE LOOP
1723 ;REPLACE ERROR CALL WITH
1724 ;'BR 4$' = 000761
1725 022160 006001 7$: ROR R1 ;CHANGE DATA TO CHECK NEXT ERROR BIT
1726 022162 077321 SOB R3,4$ ;LOOP BACK UNTIL <15:13> ALL TESTED
1727 022164 005010 CLR (R0) ;CLEAR SR0 BEFORE LEAVING
1728 022166 012737 022034 001110 MOV #1$,$LPERR ;RESET LOOP ON ERROR POINTER TO 1$
1729
1730 ;* *****
1731 ;*TEST 13 SR0 & PSW DUAL ADDRESSING TEST
1732 ;*
1733 ;* THIS TEST CHECKS MORE OF THE ADDRESS DETECTION LOGIC BY
1734 ;* VERIFYING THAT STATUS REGISTER 0 IS NOT EFFECTED BY WRITING
    
```

```
1735 ;* TO THE PSW AND THAT THE LOW BYTE OF STATUS REGISTER 0
1736 ;* IS NOT EFFECTED BY WRITING TO ITS HIGH BYTE. THIS IS TO
1737 ;* SEE IF ADJACENT OUTPUTS ARE SHORTED ON THE ADDRESS DET. LOGIC.
1738 ;*
1739 ;*****
1740 022174 000004 TST13: SCOPE
1741
1742 022176 005037 177776 1$: CLR PSW ;CLEAR THE PSW
1743 022202 005037 177572 CLR SRO ;CLEAR STATUS REGISTER 0
1744 022206 106427 000340 MTPS #340 ;SET PRIORITY 7 IN LOW BYTE OF PSW
1745 022212 013700 177572 MOV SRO,R0 ;READ STATUS REGISTER 0
1746 022216 001401 BEQ 2$ ;BRANCH IF IT WAS STILL 0
1747 022220 104013 ERROR 13 ;SRO EFFECTED BY A WRITE TO THE PSW
1748 ;FOR TIGHTER SCOPE LOOP
1749 ;REPLACE ERROR CALL WITH
1750 ;'BR 1$' = 000767
1751 022222 005037 177572 2$: CLR SRO ;BE SURE SRO IS 0 BEFORE LEAVING
1752 022226 005037 177776 CLR PSW ;BE SURE PSW IS 0 BEFORE LEAVING
1753
1754 ;*****
1755 ;*TEST 14 TEST THAT SR1 READS ALL ZEROS
1756 ;*
1757 ;* THIS TESTS CHECKS THAT STATUS REGISTER 1
1758 ;* RESPONDS WITH ALL ZEROS, AND THAT ONLY BITS<5:4>
1759 ;* OF STATUS REGISTER 3 ARE WRITEABLE.
1760 ;*
1761 ;*****
1762 022232 000004 TST14: SCOPE
1763 022234 012700 177777 1$: MOV #-1,R0 ;FILL R0 WITH ALL ONES
1764 022240 013700 177574 MOV SR1,R0 ;READ SR1 INTO R0
1765 022244 001401 BEQ 2$ ;BRANCH IF SR1 READS ALL ZEROS
1766 022246 104014 ERROR 14 ;SR1 DID NOT READ ALL ZEROS
1767 ;FOR TIGHTER SCOPE LOOP
1768 ;REPLACE ERROR CALL WITH
1769 ;000772
1770 022250 012737 177777 172516 2$: MOV #-1,SR3 ;TRY TO WRITE ONES TO SR3
1771 022256 022737 000060 172516 CMP #60,SR3 ;ONLY BITS <5:4> SHOULD BE ONES
1772 022264 001401 BEQ 3$
1773 022266 104012 ERROR 12 ;DIDN'T READ BACK A '60'
1774 022270 000005 3$: RESET ;CLEARS SR3
1775 022272 005737 172516 TST SR3 ;VERIFY THAT IT WAS CLEARED
1776 022276 4$:
1777 022276 001401 BEQ TST15 ;:BRANCH IF SR3 READ ALL ZEROS
1778 022300 104012 ERROR 12 ;SR3 DIDN'T READ ALL ZEROS
1779
1780
1781
1782 ;*****
1783 ;*TEST 15 BIT TEST OF KERNEL & USER PAR'S
1784 ;*
1785 ;* THE FOLLOWING TEST CHECKS THE BITS <15:00> OF BOTH THE KERNEL
1786 ;* AND USER PAGE ADDRESS REGISTERS. A '0' IS ROTATED THRU
1787 ;* THE REGISTERS FROM LEFT TO RIGHT.
1788 ;*
1789 ;*****
1790 022302 000004 TST15: SCOPE
```

1791										
1792	022304	012700	172340		1\$:	MOV	#KIPAR0,R0			:LOAD ADDRESS OF FIRST PAR IN R0
1793	022310	012703	000010		2\$:	MOV	#10,R3			:SETUP R3 TO COUNT 8 PAR'S
1794	022314	012737	022322	001110		MOV	#3\$,\$LPERR			:SET LOOP ON ERROR POINTER TO 3\$
1795	022322	005010			3\$:	CLR	(R0)			:CLEAR THE PAR
1796	022324	011001				MOV	(R0),R1			:READ THE PAR INTO R1
1797	022326	001401				BEQ	4\$			:BRANCH IF PAR CLEARED OK
1798	022330	104011				ERROR	11			:PAR WOULD NOT CLEAR
1799										:FOR TIGHTER SCOPE LOOP
1800										:REPLACE ERROR CALL WITH
1801										: 'BR 3\$' = 000774
1802	022332	012704	077777		4\$:	MOV	#077777,R4			:LOAD 'WALKING 0' TEST PATTERN IN R4
1803	022336	012737	022344	001110		MOV	#5\$,\$LPERR			:SET LOOP ON ERROR POINTER TO 5\$
1804	022344	005010			5\$:	CLR	(R0)			:CLEAR THE PAR BEFORE LOADING DATA
1805	022346	050410				BIS	R4,(R0)			:BIT SET THE TEST PATTERN INTO THE PAR
1806	022350	011002				MOV	(R0),R2			:READ THE PAR INTO R2
1807	022352	020402				CMP	R4,R2			:DOES DATA WRITTEN=DATA READ?
1808	022354	001402				BEQ	6\$			:BRANCH IF YES
1809	022356	010401				MOV	R4,R1			:SETUP FOR ERROR REPORTING
1810	022360	104012				ERROR	12			:PAR BITS DID NOT SET CORRECTLY
1811										:FOR TIGHTER SCOPE LOOP
1812										:REPLACE ERROR CALL WITH
1813										: 'BR 5\$' = 000767
1814	022362	000261			6\$:	SEC				:SET THE C-BIT FOR THE ROTATE INST.
1815	022364	006004				ROR	R4			:ROTATE THE TEST PATTERN IN R4
1816	022366	103766				BLS	5\$			:BRANCH BACK IF MORE BITS TO TEST
1817	022370	062700	000002			ADD	#2,R0			:GET NEXT PAR ADDRESS IN R0
1818	022374	077326				SOB	R3,3\$			:BRANCH BACK UNTIL ALL PAR'S TESTED
1819	022376	022700	177660			CMP	#UIPAR7+2,R0			:HAVE USER PAR'S BEEN TESTED
1820	022402	103003				BHIS	7\$			:BRANCH IF YES
1821	022404	012700	177640			MOV	#UIPAR0,R0			:LOAD FIRST USER PAR ADDR. IN R0
1822	022410	000737				BR	2\$			:BRANCH BACK TO TEST USER PAR'S
1823	022412	012737	022304	001110	7\$:	MOV	#1\$,\$LPERR			:RESET LOOP OR ERROR POINTER TO 1\$
1824										:LEAVE TEST WITH BITS <11:1>=1 IN ALL PAR'S

```

1825
1826
1827
1828
1829
1830
1831
1832
1833
1834 022420 000004
1835
1836 022422 012700 172300 1$: MOV #KIPDR0,R0 ;LOAD ADDRESS OF FIRST PDR IN R0
1837 022426 012703 000010 2$: MOV #10,R3 ;SETUP R3 TO COUNT 8 PDR'S
1838 022432 012737 02244C 001110 3$: MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
1839 022440 005010 3$: CLR (R0) ;CLEAR THE PDR
1840 022442 011001 ;READ THE PDR INTO R1
1841 022444 001401 ;BEQ 4$ ;BRANCH IF PDR CLEARED OK
1842 022446 104011 ;ERROR 11 ;PDR WOULD NOT CLEAR
1843 ;FOR TIGHTER SCOPE LOOP
1844 ;REPLACE ERROR CALL WITH
1845 ;'BR 3$' = 000774
1846 022450 012704 077777 4$: MOV #077777,R4 ;LOAD 'WALKING '0' TEST PATTERN IN R4
1847 022454 012737 022462 001110 5$: MOV #5$, $LPERR ;SET LOOP ON ERROR POINTER TO 5$
1848 022462 005010 5$: CLR (R0) ;CLEAR THE PDR BEFORE LOADING DATA
1849 022464 010401 ;MOV R4,R1 ;LOAD DATA INTO R1
1850 022466 042701 100361 ;BIC #100361,R1 ;MASK UNUSED BITS OUT OF THE DATA
1851 022472 050110 ;BIS R1,(R0) ;BIT SET THE TEST PATTERN INTO THE PDR
1852 022474 011002 ;MOV (R0),R2 ;READ THE PDR INTO R2
1853 022476 020102 ;CMP R1,R2 ;DOES DATA WRITTEN=DATA READ?
1854 022500 001401 ;BEQ 6$ ;BRANCH IF YES
1855 022502 104012 ;ERROR 12 ;PDR BITS DID NOT SET CORRECTLY
1856 ;FOR TIGHTER SCOPE LOOP
1857 ;REPLACE ERROR CALL WITH
1858 ;'BR 5$' = 000767
1859 022504 000261 6$: SEC ;SET THE C-BIT FOR THE ROTATE INST.
1860 022506 006004 ;ROR R4 ;ROTATE THE TEST PATTERN IN R4
1861 022510 103764 ;BCS 5$ ;BRANCH BACK IF MORE BITS TO TEST
1862 022512 062700 000002 ;ADD #2,R0 ;GET NEXT PDR ADDRESS IN R0
1863 022516 077330 ;SOB R3,3$ ;BRANCH BACK UNTIL ALL PDR'S TESTED
1864 022520 022700 177620 ;CMP #UIPDR7+2,R0 ;HAVE USER PDR'S BEEN TESTED?
1865 022524 103003 ;BHS 7$ ;BRANCH IF YES
1866 022526 012700 177600 ;MOV #UIPDR0,R0 ;LOAD FIRST USER PDR ADDR. IN R0
1867 022532 000735 ;BR 2$ ;BRANCH BACK TO TEST USER PDR'S
1868 022534 012737 022422 001110 7$: MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
1869 ;LEAVE TEST WITH ALL WRITEABLE BITS IN
1870 ;ALL PDR'S = 1
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880 022542 000004
    
```

```

*****
*TEST 17 TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PAR'S
*
* THE FOLLOWING TEST WRITES TO BOTH BYTES OF THE KERNEL & USER
* PAR'S SEPERATELY TO SEE THAT WRITING TO ONE DOES NOT EFFECT
* THE OTHER.
*****
TST17: SCOPE
    
```

1881									
1882	022544	012700	172340		1\$:	MOV	#KIPAR0,RC		;LOAD ADDRESS OF FIRST PAR INTO R0
1883	022550	012737	022562	001110	2\$:	MOV	#3\$,\$LPERR		;SET LOOP ON ERROR POINTER TO 3\$
1884	022556	012703	000010			MOV	#10,R3		;LOAD LOOP COUNTER TO DO 8 PAR'S
1885	022562	012701	177777		3\$:	MOV	#-1,R1		;LOAD TEST PATTERN INTO R1
1886	022566	005010				CLR	(R0)		;CLEAR THE PAR

1887	022570	110110				MOVB	R1,(R0)	:WRITE 1'S TO THE LOW BYTE OF THE PAR
1888	022572	011002				MOV	(R0),R2	:READ THE ENTIRE PAR INTO R2
1889	022574	042701	177400			BIC	#177400,R1	:MASK HIGH BYTE & UNUSED BITS OUT OF THE DATA
1890	022600	020102				CMP	R1,R2	:WAS ONLY THE LOW BYTE WRITTEN TO
1891	022602	001401				BEQ	4\$	:BRANCH IF YES
1892	022604	104015				ERROR	15	:HIGH BYTE EFFECTED BY WRITING LOW BYTE IN PAR
1893								:FOR TIGHTER SCOPE LOOP
1894								:REPLACE ERROR CALL WITH
1895								: 'BR 3\$' = 000766
1896	022606	012737	022614	001110	4\$:	MOV	#5\$,\$LPERR	:SET LOOP ON ERROR POINTER TO 5\$
1897	022614	005010			5\$:	CLR	(R0)	:CLEAR THE PAR
1898	022616	012701	177777			MOV	#-1,R1	:LOAD TEST, PATTERN INTO R1
1899	022622	110160	000001			MOVB	R1,1(R0)	:WRITE 1'S TO THE HIGH BYTE OF THE PAR
1900	022626	011002				MOV	(R0),R2	:READ THE ENTIRE PAR INTO R2
1901	022630	042701	000377			BIC	#000377,R1	:MASK LOW BYTE
1902	022634	020102				CMP	R1,R2	:WAS ONLY THE HIGH BYTE WRITTEN TO?
1903	022636	001401				BEQ	6\$	:BRANCH IF YES
1904	022640	104015				ERROR	15	:LOW BYTE EFFECTED BY WRITING HIGH BYTE IN PAR
1905								:FOR TIGHTER SCOPE LOOP
1906								:REPLACE ERROR CALL WITH
1907								: 'BR 5\$' = 000765
1908	022642	062700	000002		6\$:	ADD	#2,R0	:PUT ADDRESS OF NEXT PAR IN R0
1909	022646	077333				SOB	R3,3\$	:BRANCH BACK UNTIL 8 PAR'S TESTED
1910	022650	022700	177660			CMP	#UIPAR7+2,R0	:HAVE USER PAR'S BEEN TESTED
1911	022654	103003				BHIS	7\$	:BRANCH IF YES
1912	022656	012700	177640			MOV	#UIPAR0,R0	:LOAD ADDRESS OF FIRST USER PAR IN R0
1913	022662	000732				BR	2\$	:BRANCH BACK TO TEST USER PAR'S
1914	022664	012737	022544	001110	7\$:	MOV	#1\$,\$LPERR	:RESET LOOP ON ERROR POINTER TO 1\$
1915								

1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962

022672 000004  
  
022674 012700 172300  
022700 012737 022712 001110  
022706 012703 000010  
022712 012701 177777  
022716 005010  
022720 110110  
022722 011002  
022724 042701 177761  
022730 020102  
022732 001401  
022734 104015  
  
022736 012737 022744 001110  
022744 005010  
022746 012701 177777  
022752 110160 000001  
022756 011002  
022760 042701 100377  
022764 020102  
022766 001401  
022770 104015  
  
022772 062700 000002  
022776 077333  
023000 022700 177620  
023004 103003  
023006 012700 177600  
023012 000732  
023014 012737 022674 001110

```

:*****
:TEST 20 TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PDR'S
:
:* THE FOLLOWING TEST WRITES TO BOTH BYTES OF THE KERNEL & USER
:
:* PDR'S SEPERATELY TO SEE THAT WRITING TO ONE DOES NOT EFFECT
:* THE OTHER.
:*****
TST20: SCOPE
1$: MOV #KIPDR0,R0 ;LOAD ADDRESS OF FIRST PDR INTO R0
2$: MOV #3$,$LPERR ;SET LOOP ON ERROR POINTER TO 3$
MOV #10,R3 ;LOAD LOOP COUNTER TO DO 8 PDR'S
3$: MOV #-1,R1 ;LOAD TEST PATTERN INTO R1
CLR (R0) ;CLEAR THE PDR
MOVB R1,(R0) ;WRITE 1'S TO THE LOW BYTE OF THE PDR
MOV (R0),R2 ;READ THE ENTIRE PDR INTO R2
BIC #177761,R1 ;MASK HIGH BYTE & UNUSED BITS OUT OF DATA
CMP R1,R2 ;WAS ONLY THE LOW BYTE WRITTEN TO?
BEQ 4$ ;BRANCH IF YES
ERROR 15 ;HIGH BYTE EFFECTED BY WRITING LOW BYTE IN PDR
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 3$' = 000766
4$: MOV #5$,$LPERR ;SET LOOP ON ERROR POINTER TO 5$
5$: CLR (R0) ;CLEAR THE PDR
MOV #-1,R1 ;LOAD TEST PATTERN INTO R1
MOVB R1,1(R0) ;WRITE 1'S TO THE HIGH BYTE OF THE PDR
MOV (R0),R2 ;READ THE ENTIRE PDR INTO R2
BIC #100377,R1 ;MASK LOW BYTE & UNUSED BITS OUT OF DATA
CMP R1,R2 ;WAS ONLY THE HIGH BYTE WRITTEN TO?
BEQ 6$ ;BRANCH IF YES
ERROR 15 ;LOW BYTE EFFECTED BY WRITING HIGH BYTE IN PDR
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 5$' = 000765
6$: ADD #2,R0 ;PUT ADDRESS OF NEXT PDR IN R0
SOB R3,3$ ;BRANCH BACK UNTIL 8 PDR'S TESTED
CMP #UIPDR7+2,R0 ;HAVE USER PDR'S BEEN TESTED?
BHS 7$ ;BRANCH IF YES
MOV #UIPDR0,R0 ;LOAD ADDRESS OF FIRST USER PDR IN R0
BR 2$ ;BRANCH BACK TO TEST USER PDR'S
7$: MOV #1$,$LPERR ;RESET LOOP ON ERROR POINTER TO 1$

```

```

1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976 023022 000004
1977
1978 023024 012737 023046 001110 1$:  MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER 2$
1979 023032 012703 000010      MOV #10,R3 ;LOAD LOOP COUNTER WITH AN 8
1980 023036 012700 172300      MOV #KIPDRO,R0 ;LOAD ADDRESS OF FIRST KERNEL PDR AND RO
1981 023042 004737 035202      JSR PC,SETREG ;SET ALL BITS IN ALL PAR'S IN PDR'S
1982 023046 012706 001100      2$:  MOV #KERSTK,KSP ;SETUP STACK POINTER
1983 023052 005010      CLR (R0) ;CLEAR ONE OF THE KERNEL PDR'S
1984 023054 004737 035274      JSR PC,CMPREG ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
1985 023060 012720 177777      MOV #-1,(R0)+ ;RESTORE ALL ONES, AND SETUP FOR NEXT PDR
1986 023064 077310      SOB R3,2$ ;LOOP TO 2$ UNTIL ALL KERNEL PDR'S CHECKED
1987 023066 012737 023104 001110      MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
1988 023074 012703 000010      MOV #10,R3 ;LOAD LOOP COUNTER WITH AN 8
1989 023100 012700 172340      MOV #KIPARO,R0 ;LOAD ADDRESS OF FIRST KERNEL PAR IN RO
1990 023104 012706 001100      3$:  MOV #KERSTK,KSP ;SETUP STACK POINTER
1991 023110 005010      CLR (R0) ;CLEAR ONE OF THE KERNEL PAR'S
1992 023112 004737 035274      JSR PC,CMPREG ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
1993 023116 012720 177777      MOV #-1,(R0)+ ;RESTORE ALL ONES, AND SETUP FOR NEXT PAR
1994 023122 077310      SOB R3,3$ ;LOOP TO 3$ UNTIL ALL KERNEL PAR'S CHECKED
1995 023124 012737 023142 001110      MOV #4$, $LPERR ;SET LOOP ON ERROR POINTER TO 4$
1996 023132 012703 000010      MOV #10,R3 ;LOAD LOOP COUNTER WITH AN 8
1997 023136 012700 177600      MOV #UIPDRO,R0 ;LOAD ADDRESS OF FIRST USER PDR IN RO
1998 023142 012706 001100      4$:  MOV #KERSTK,KSP ;SETUP STACK POINTER
1999 023146 005010      CLR (R0) ;CLEAR ONE OF THE USER PDR'S
2000 023150 004737 035274      JSR PC,CMPREG ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
2001 023154 012720 177777      MOV #-1,(R0)+ ;RESTORE ALL ONES, AND SETUP FOR NEXT UPDR
2002 023160 077310      SOB R3,4$ ;LOOP TO 4$ UNTIL ALL USER PDR'S CHECKED
2003 023162 012737 023200 001110      MOV #5$, $LPERR ;SET LOOP ON ERROR POINTER TO 5$
2004 023170 012703 000010      MOV #10,R3 ;LOAD LOOP COUNTER WITH AN 8
2005 023174 012700 177640      MOV #UIPARO,R0 ;LOAD ADDRESS OF FIRST USER PAR IN RO
2006 023200 012706 001100      5$:  MOV #KERSTK,KSP ;SETUP STACK POINTER
2007 023204 005010      CLR (R0) ;CLEAR ONE OF THE USER PAR'S
2008 023206 004737 035274      JSR PC,CMPREG ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
    
```



```

2009 023212 012720 177777      MOV    #-1,(R0)+      ;RESTORE ALL ONES, AND SETUP FOR NEXT UPAR
2010 023216 077310              SOB    R3,5$         ;LOOP TO 5$ UNTIL ALL USER PAR'S CHECKED
2011 023220 012737 023024 001110  MOV    #1$,$LPERR    ;SET LOOP ON ERROR POINTER TO 1$
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023 023226 000004      ;*****
                ;*TEST 22      TEST THAT PAR-PDR'S NOT AFFECTED BY RESET
                ;*
                ;*      THIS TEST CHECKS TO SEE THAT THE KERNEL OR USER PAR/PDR'S ARE
                ;*      NOT AFFECTED BY THE EXECUTION OF A 'RESET' INSTRUCTION.  THE
                ;*      'SETREG' SUBROUTINE IS USED TO SET ALL WRITEABLE BITS TO A '1' IN
                ;*      THE PAR/PDR'S.  THEN THEY ARE READ TO SEE THAT THEY REMAINED
                ;*      UNCHANGED
                ;*
                ;*****
2024
2025
2026 023230 004737 035202      1$:    JSR    PC,SETREG    ;SET ALL BITS IN ALL PAR'S AND PDR'S
2027 023234 000005              RESET                ;ISSUE AN 'INIT' BY EXECUTING A RESET
2028 023236 012700 172300      MOV    #KIPDR0,R0    ;LOAD ADDRESS OF FIRST KERNEL PDR IN R0
2029 023242 012704 000010      MOV    #10,R4        ;LOAD LOOP COUNTER WITH AN 8
2030 023246 011001              MOV    (R0),R1       ;READ A KERNEL PDR INTO R1
2031 023250 022701 077416      2$:    CMP    #77416,R1    ;ARE ALL THE BITS STILL SET?
2032 023254 001401              BEQ    3$            ;BRANCH IF YES
2033 023256 104055              ERROR   55          ;KERNEL PDR AFFECTED BY A RESET
2034
2035
2036
2037 023260 062700 000002      3$:    ADD    #2,R0      ;FORM ADDRESS OF NEXT KERNEL PDR
2038 023264 077410              SOB    R4,2$        ;LOOP TO 2$ UNTIL ALL KERNEL PDR'S CHECKED
2039 023266 012700 172340      MOV    #KIPAR0,R0    ;LOAD ADDRESS OF FIRST KERNEL PAR IN R0
2040 023272 012704 000010      MOV    #10,R4        ;LOAD LOOP COUNTER WITH AN 8
2041 023276 011001              MOV    (R0),R1       ;READ A KERNEL PAR INTO R1
2042 023300 022701 177777      4$:    CMP    #177777,R1 ;ARE ALL THE BITS STILL SET?
2043 023304 001401              BEQ    5$            ;BRANCH IF YES
2044 023306 104055              ERROR   55          ;KERNEL PAR AFFECTED BY A RESET
2045
2046
2047
2048 023310 062700 000002      5$:    ADD    #2,R0      ;FORM ADDRESS OF NEXT KERNEL PAR
2049 023314 077410              SOB    R4,4$        ;LOOP TO 4$ UNTIL ALL KERNEL PAR'S CHECKED
2050 023316 012700 177600      MOV    #UIPDR0,R0    ;LOAD ADDRESS OF FIRST USER PDR IN R0
2051 023322 012704 000010      MOV    #10,R4        ;LOAD LOOP COUNTER WITH AN 8
2052 023326 011001              MOV    (R0),R1       ;READ A USER PDR INTO R1
2053 023330 022701 077416      6$:    CMP    #77416,R1  ;ARE ALL THE BITS STILL SET?
2054 023334 001401              BEQ    7$            ;BRANCH IF YES
2055 023336 104055              ERROR   55          ;USER PDR AFFECTED BY A RESET
2056
2057
2058
2059 023340 062700 000002      7$:    ADD    #2,R0      ;FORM ADDRESS OF NEXT USER PDR
2060 023344 077410              SOB    R4,6$        ;LOOP TO 6$ UNTIL ALL USER PDR'S CHECKED
2061
2062 023346 012700 177640      MOV    #UIPAR0,R0    ;LOAD ADDRESS OF FIRST USER PAR IN R0
2063 023352 012704 000010      MOV    #10,R4        ;LOAD LOOP COUNTER WITH AN 8
2064 023356 011001              8$:    MOV    (R0),R1       ;READ A USER PAR INTO R1
    
```

```

2065 023360 022701 177777          CMP    #177777,R1      ;ARE ALL THE BITS STILL SET?
2066 023364 001401          BEQ    9$              ;BRANCH IF YES
2067 023366 104055          ERROR  55              ;USER PAR AFFECTED BY A RESET
2068                                ;FOR TIGHTER SCOPE LOOP
2069                                ;REPLACE ERROR CALL WITH
2070                                ;'BR 8$' = 000773
2071 023370 062700 000002          9$:   ADD    #2,R0      ;FORM ADDRESS OF NEXT USER PAR
2072 023374 077410          SOB    R4,8$          ;LOOP TO 8$ UNTIL ALL USER PAR'S CHECKED
    
```

2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096

```

:*****
:TEST 23      RELOCATION & ADDER TEST (NO CARRIES)
:
:   THE FOLLOWING TEST SETS UP THE KERNEL PAR'S AND PDR'S
:   FOR THE REST OF THE PROGRAM.  IT THEN USES DIFFERENT
:   VIRTUAL ADDRESSES AND DIFFERENT VALUES FOR KERNEL PAR 4
:   TO PUT DIFFERENT PATTERNS AT THE INPUTS OF THE
:   MEMORY MANAGEMENT ADDER.  THE VALUES ARE SUCH
:   THAT NO CARRIES ARE GENERATED OUT OF THE ADDER.
:
:   THE METHOD USED TO SEE THAT THE RIGHT PHYSICAL BUS ADDRESS
:   IS FORMED BY THE ADDER IS TO WRITE A PATTERN TO VIRTUAL
:   LOCATION WITH MEMORY MGMT., AND
:   THEN READ THAT LOCATION USING THE PHYSICAL ADDRESS THAT SHOULD
:   HAVE BEEN FORMED TO SEE IF THE TEST PATTERN GOT THEIR.
:   22-BIT AND 18-BIT ADDRESSING ARE USED.
:*****
    
```

2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105  
2106  
2107  
2108  
2109  
2110  
2111  
2112

```

2113 023376 000004          TST23: SCOPE
2114
2115 023400 012700 172340          1$:   MOV    #KIPAR0,R0  ;LOAD ADDRESS OF FIRST KERNEL PAR IN R0
2116 023404 005001          CLR    R1              ;CLEAR R1
2117 023406 012702 000007          MOV    #7,R2           ;LOAD LOOP COUNTER WITH A 7
2118 023412 010120          2$:   MOV    R1,(R0)+     ;MAP KERNEL PAR'S TO PAGES 0-6 (4K EACH)
2119 023414 062701 000200          ADD    #200,R1
2120 023420 077204          SOB    R2,2$          ;LOOP UNTIL KIPAR0 - KIPAR6 ARE LOADED
    
```



```

2177                                     :REPLACE ERROR CALL WITH
2178                                     :'BR 6$' = 000742
2179 023636                                7$:
2180 023636 012737 023636 001110 8$:    MOV    #8$, $LPERR    ;SET LOOP ON ERROR POINTER TO 8$
2181 023644 012700 067776                MOV    #67776,R0     ;LOAD PHYSICAL ADDR. PBA INTO R0
2182 023650 012701 105276                MOV    #105276,R1    ;LOAD VIRTUAL ADDR. VBA INTO R1
2183 023654 012702 125252                MOV    #125252,R2    ;LOAD TEST PATTERN INTO R2
2184 023660 012704 000625                MOV    #625,R4 ;LOAD R4 WITH PAR VALUE
2185 023664 010437 172350                MOV    R4,KIPAR4    ;LOAD KERNEL PAR 4 BITS <15:00>
2186 023670 011037 001176                MOV    (R0), $TMP0  ;SAVE CONTENTS AT TEST LOCATION
2187 023674 052737 000021 177572        BIS    #21,SRO      ;TURN ON MEM. MGMT. 22-BIT ADDRESSING
2188 023702 010211                        MOV    R2,(R1)      ;LOAD 125252 USING ADDER (PAR4 + VIRT ADDR.)
2189 023704 000005                        RESET             ;TURN OFF MEMORY MGMT.
2190 023706 011003                        MOV    (R0),R3      ;READ 125252 BACK WITHOUT USING MEM. MGMT.
2191 023710 013710 001176                MOV    $TMP0,(R0)   ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2192 023714 020203                        CMP     R2,R3        ;WAS SAME PATTERN READ BACK THAT WAS
2193                                     ;WRITTEN USING MEMORY MANAGEMENT?
2194 023716 001405                        BEQ     9$           ;BRANCH IF YES
2195 023720 010137 001306                MOV    R1,VIRT1     ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2196 023724 004737 035466                JSR    PC,FORMPA    ;GO FORM PHYSICAL ADDRESS FOR TYPING
2197 023730 104017                        ERROR   17          ;TEST LOCATION DID NOT HAVE PATTERN
2198                                     ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
2199                                     ;APPARENTLY PHYSICAL ADDR. WAS
2200                                     ;FORMED WRONG BY ADDERS USING
2201                                     ;THE VIRTUAL ADDR. AND KIPAR4
2202                                     ;FOR TIGHTER SCOPE LOOP
2203                                     ;REPLACE ERROR CALL WITH
2204                                     ;'BR 8$' = 000742
2205 023732                                9$:
2206
2207 023732 012737 023732 001110 10$:    MOV    #10$, $LPERR ;SET LOOP ON ERROR POINTER TO 10$
2208 023740 012700 177776                MOV    #PSW,R0      ;LOAD PHYS. ADDR. OF PSW INTO R0
2209 023744 012701 100076                MOV    #100076,R1   ;LOAD VIRTUAL ADDR. FOR PSW INTO R1
2210 023750 012702 030340                MOV    #030340,R2   ;LOAD DATA FOR PSW IN R2
2211 023754 012704 007777                MOV    #7777,R4     ;LOAD R4 WITH PAR VALUE
2212 023760 010437 172350                MOV    R4,KIPAR4    ;LOAD KERNEL PAR 4 BITS <11:00>
2213 023764 005010                        CLR    (R0)         ;CLEAR THE PSW
2214 023766 052737 000001 177572        BIS    #BIT0,SRO    ;TURN ON 'MEMORY MANAGEMENT' (18 BIT ADDRESSING)
2215 023774 010211                        MOV    R2,(R1)      ;LOAD PSW USING ADDER (PAR4 + VIRT ADDR.)
2216 023776 000005                        RESET             ;TURN OFF MEM. MGMT (SRO=0)
2217 024000 011003                        MOV    (R0),R3      ;READ PSW BACK WITHOUT USING MEM. MGMT.
2218 024002 005010                        CLR    (R0)         ;CLEAR THE PSW
2219 024004 042703 000037                BIC    #37,R3       ;MASK T-BIT & CC BITS OUT OF DATA READ
2220 024010 020203                        CMP     R2,R3        ;WAS PSW WRITTEN?
2221 024012 001405                        BEQ     11$         ;BRANCH IF YES
2222 024014 010137 001306                MOV    R1,VIRT1     ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2223 024020 004737 035466                JSR    PC,FORMPA    ;GO FORM PHYSICAL ADDR. FOR TYPING
2224 024024 104017                        ERROR   17          ;PSW DID NOT HAVE DATA THAT IT SHOULD HAVE,
2225                                     ;APPARENTLY PHYS. ADDR. OF PSW WAS
2226                                     ;NOT FORMED BY ADDERS USING THE
2227                                     ;VIRTUAL ADDR. AND KIPAR4
2228                                     ;FOR TIGHTER SCOPE LOOP
2229                                     ;REPLACE ERROR CALL WITH
2230                                     ;'BR 10$' = 000742
2231 024026 012737 024026 001110 11$:    MOV    #11$, $LPERR ;SET LOOP ON ERROR POINTER TO 11$
2232 024034 012700 177776                MOV    #PSW,R0      ;LOAD PHYS. ADDR. OF PSW INTO R0
    
```

```

2233 024040 012701 117776      MOV      #117776,R1      ;LOAD VIRTUAL ADDR. FOR PSW INTO R1
2234 024044 012702 030240      MOV      #030240,R2      ;LOAD DATA FOR PSW IN R2
2235 024050 012704 177600      MOV      #177600,R4      ;LOAD R4 WITH PAR VALUE
2236 024054 010437 172350      MOV      R4,KIPAR4      ;LOAD KERNEL PAR 4 BITS <15:00>
2237 024060 052737 000021 177572  BIS      #21,SRO        ;TURN ON 'MEMORY MANAGEMENT'(22 BIT ADDRESSING)
2238 024066 010211              MOV      R2,(R1)        ;LOAD PSW USING ADDER (PAR4 + VIRT. ADDR.)
2239 024070 000005              RESET                     ;TURN OFF MEM. MGMT (SRO=0)
2240 024072 011003              MOV      (R0),R3        ;READ PSW BACK WITHOUT USING MEM. MGMT.
2241 024074 005010              CLR      (R0)           ;CLEAR THE PSW
2242 024076 042703 000037      BIC      #37,R3         ;MASK T-BIT & CC BITS OUT OF DATA READ
2243 024102 020203              CMP      R2,R3         ;WAS PSW WRITTEN?
2244 024104 001405              BEQ     12$            ;BRANCH IF YES
2245 024106 010137 001306      MOV      R1,VIRT1       ;SAVE VIRTUAL ADDR. TO FORM PHYSICAL ADDR.
2246 024112 004737 035466      JSR     PC,FORMPA      ;GO FORM PHYSICAL ADDR. FOR TYPING
2247 024116 104017              ERROR    17            ;PSW DID NOT HAVE DATA THAT IT SHOULD
2248                                ;HAVE, APPARENTLY PHYS. ADDR. OF PSW WAS
2249                                ;NOT FORMED BY ADDERS USING THE
2250                                ;VIRTUAL ADDR. AND KIPAR4
2251                                ;FOR TIGHTER SCOPE LOOP
2252                                ;REPLACE ERROR CALL WITH
2253                                ;'BR 11$' = 000743
2254 024120 012737 023400 001110 12$:  MOV      #1$,$LPERR    ;RESET LOOP ON ERROR POINTER TO 1$
2255
2256                                ;*****
2257                                ;*TEST 24      RELOCATION & ADDER TEST (WITH CARRIES)
2258                                ;*
2259                                ;* THE FOLLOWING TEST USES THE SAME METHOD AS THE PREVIOUS
2260                                ;* TEST TO VERIFY MEMORY MANagements ABILITY TO CONSTRUCT
2261                                ;* PHYSICAL BUS ADDRESSES USING A VIRTUAL BUS ADDRESS AND THE
2262                                ;* CONTENTS OF A PAGE ADDRESS REGISTER. HOWEVER, THE VALUES
2263                                ;* AND PATTERNS USED IN THIS TEST WILL GENERATE CARRIES
2264                                ;* AND CHECK 'WRAPAROUND' TO ADDRESS 000000 BY
2265                                ;* USING VIRTUAL ADDR. 111400 AND KIPAR4 = 177664.
2266                                ;* 22-BIT ADDRESSING IS USED.
2267                                ;*****
2268 024126 000004      TST24: SCOPE
2269
2270 024130              1$:
2271                                ;KERNEL PAR'S AND PDR'S HAVE BEEN
2272                                ;SETUP BY THE PREVIOUS TEST
2272 024130 012737 024130 001110 2$:  MOV      #2$,$LPERR    ;SET LOOP ON ERROR POINTER TO 2$
2273 024136 012700 066476      MOV      #66476,R0      ;LOAD PHYSICAL ADDR. PBA INTO R0
2274 024142 012701 114376      MOV      #114376,R1     ;LOAD VIRTUAL ADDR. VBA INTO R1
2275 024146 012702 125253      MOV      #125253,R2     ;LOAD TEST PATTERN INTO R2
2276 024152 012704 000521      MOV      #521,R4        ;LOAD R4 WITH PAR VALUE
2277 024156 010437 172350      MOV      R4,KIPAR4      ;LOAD KERNEL PAP 4 BITS <15:00>
2278 024162 011037 001176      MOV      (R0),$TMP0     ;SAVE CONTENTS AT TEST LOCATION
2279 024166 052737 000021 177572  BIS      #21,SRO        ;TURN ON MEM. MGMT. 22-BIT ADDRESSING
2280 024174 010211              MOV      R2,(R1)        ;LOAD 125253 USING ADDER (PAR4 + VIRT ADDR.)
2281 024176 000005              RESET                     ;TURN OFF MEMORY MGMT.
2282 024200 011003              MOV      (R0),R3        ;READ 125253 BACK WITHOUT USING MEM. MGMT.
2283 024202 013710 001176      MOV      $TMP0,(R0)     ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2284 024206 020203              CMP      R2,R3         ;WAS SAME PATTERN READ BACK THAT WAS
2285                                ;WRITTEN USING MEMORY MANAGEMENT?
2286 024210 001405              BEQ     3$              ;BRANCH IF YES
2287 024212 010137 001306      MOV      R1,VIRT1       ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2288 024216 004737 035466      JSR     PC,FORMPA      ;GO FORM PHYSICAL ADDRESS FOR TYPING
    
```



```

2345                                     ;THE VIRTUAL ADDR. AND KIPAR4
2346                                     ;FOR TIGHTER SCOPE LOOP
2347                                     ;REPLACE ERROR CALL WITH
2348                                     ;'BR 6$' = 000742
2349 024414                                7$:
2350 024414 012737 024414 001110 8$:    MOV    #3$, $LPERR    ;SET LOOP ON ERROR POINTER TO 8$
2351 024422 012700 000000                MOV    #00000, R0    ;LOAD PHYSICAL ADDR. FBA INTO R0
2352 024426 012701 111400                MOV    #111400, R1   ;LOAD VIRTUAL ADDR. VBA INTO R1
2353 024432 012702 125256                MOV    #125256, R2   ;LOAD TEST PATTERN INTO R2
2354 024436 012704 177664                MOV    #177664, R4   ;LOAD R4 WITH PAR VALUE
2355 024442 010437 172350                MOV    R4, KIPAR4    ;LOAD KERNEL PAR 4 BITS <15:00>
2356 024446 011037 001176                MOV    (R0), $TMP0   ;SAVE CONTENTS AT TEST LOCATION
2357 024452 052737 000021 177572        BIS    #21, SR0      ;TURN ON MEM. MGMT. 22-BIT ADDRESSING
2358 024460 010211                        MOV    R2, (R1)      ;LOAD 125256 USING ADDER (PAR4 + VIRT ADDR.)
2359 024462 000005                        RESET                       ;TURN OFF MEMORY MGMT.
2360 024464 011003                        MOV    (R0), R3      ;READ 125256 BACK WITHOUT USING MEM. MGMT.
2361 024466 013710 001176                MOV    $TMP0, (R0)   ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2362 024472 020203                        CMP    R2, R3        ;WAS SAME PATTERN READ BACK THAT WAS
2363                                     ;WRITTEN USING MEMORY MANAGEMENT?
2364 024474 001405                        BEQ    9$            ;BRANCH IF YES
2365 024476 010137 001306                MOV    R1, VIRT1     ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2366 024502 004737 035466                JSR    PC, FORMPA    ;GO FORM PHYSICAL ADDRESS FOR TYPING
2367 024506 104017                        ERROR   17           ;TEST LOCATION DID NOT HAVE PATTERN
2368                                     ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
2369                                     ;APPARENTLY PHYSICAL ADDR. WAS
2370                                     ;FORMED WRONG BY ADDERS USING
2371                                     ;THE VIRTUAL ADDR. AND KIPAR4
2372                                     ;FOR TIGHTER SCOPE LOOP
2373                                     ;REPLACE ERROR CALL WITH
2374                                     ;'BR 8$' = 000742
2375 024510                                9$:
2376 024510 012737 024130 001110        MOV    #1$, $LPERR    ;RESET LOOP ON ERROR POINTER TO 1$
    
```

```

2377                                     ;*****
2378                                     ;*TEST 25      READ AND WRITE WHILE IN RELOCATE MODE
2379                                     ;*
2380                                     ;*
2381                                     ;*   THE FOLLOWING TEST TURNS ON MEMORY MANAGEMENT AND THEN
2382                                     ;*   READS AND WRITES LOCATIONS BETWEEN PHYSICAL ADDRESSES
2383                                     ;*   060000-067600.  ONE LOCATION IN EVERY BLOCK (32. WORDS)
2384                                     ;*   IS WRITTEN USING PAR4 AND READ USING PAR5.  THIS IS
2385                                     ;*   DONE IN BOTH USER AND KERNEL MODES.  THE USER PAR/PDR'S
2386                                     ;*   ARE SETUP AT THE BEGINNING OF THE TEST AND ONCE MEMORY
2387                                     ;*   MANAGEMENT IS TURNED ON IT IS LEFT ON FOR THE REST OF THE
2388                                     ;*   OF THE PROGRAM.  THE 'MODE' INPUT TO THE PAR/PDR ADDRESS MUX
2389                                     ;*   IS CHECKED BY READING AND WRITING IN USER MODE.  REMEMBER
2390                                     ;*   ALSO, THAT SINCE MEMORY MANAGEMENT IS ON (IN RELOCATE
2391                                     ;*   MCDE) THE PROGRAM ITSELF IS USING ITS VIRTUAL ADDRESSES AND
2392                                     ;*   THE PAR/PDR'S TO EXECUTE.
2393                                     ;*
2394                                     ;*   WHILE TESTING IN KERNEL MODE, USER PAGES 4 & 5 ARE MAPPED
2395                                     ;*   NON-RESIDENT WITH DIFFERENT PAR VALUES THAN THE KERNEL
2396                                     ;*   PAR'S TO BE SURE THAT THE KERNEL PAR'S AND PDR'S ARE BEING
2397                                     ;*   USED WHEN IN KERNEL MODE (AND VICE VERSA WHILE TESTING IN
2398                                     ;*   USER MODE).  IF A MEM. MGMT. TRAP OCCURS, THE PROGRAM GOES
2399                                     ;*   TO 9$ WHERE THE TRAP IS REPORTED.
2400                                     ;*
    
```



```

2401      ;*      BY SETTING THE LOCATION $MADR1 IN THE E-TABLE TO A CONSTANT,
2402      ;*      AS DESCRIBED IN THE DOCUMENTATION, THIS TEST WILL CONTINUE
2403      ;*      ACCESSING LOCATIONS THROUGHOUT MEMORY BY INCREASING THE VALUE
2404      ;*      OF PAR4 AND PAR5.
2405      ;*****
2406 024516 000004      TST25: SCOPE
2407
2408 024520 005037 177776      1$: CLR      PSW      ;START IN KERNEL MODE
2409 024524 012704 000577      MOV      #577,R4    ;LOAD R4 WITH VALUE FOR PAR4
2410 024530 012705 000600      MOV      #600,R5    ;LOAD R5 WITH VALUE FOR PAR5
2411 024534 010437 172350      MOV      R4,KIPAR4  ;LOAD KERNEL PAR4
2412 024540 010537 172352      MOV      R5,KIPAR5  ;LOAD KERNEL PAR5
2413 024544 012700 177640      MOV      #UIPAR0,R0 ;LOAD ADDRESS OF FIRST USER PAR IN R0
2414 024550 005001      CLR      R1      ;CLEAR R1
2415 024552 012702 000007      MOV      #7,R2      ;LOAD LOOP COUNTER WITH A 7
2416 024556 010120      2$: MOV      R1,(R0)+  ;MAP USER PAR'S TO PAGES 0-6 (4K EACH)
2417 024560 062701 000200      ADD      #200,R1
2418 024564 077204      SOB      R2,2$      ;LOOP UNTIL UIPAR0-UIPAR6 ARE LOADED
2419 024566 012710 177600      MOV      #177600,(R0) ;MAP USER PAR7 TO THE I/O PAGE
2420 024572 012700 177600      MOV      #UIPDR0,R0 ;LOAD ADDRESS OF FIRST USER PDR IN R0
2421 024576 012701 077406      MOV      #77406,R1  ;LOAD PDR DATA INTO R1
2422 024602 012702 000010      MOV      #10,R2     ;LOAD LOOP COUNTER WITH AN 8
2423 024606 010120      3$: MOV      R1,(R0)+  ;MAP ALL 8 PAGES 128 BLOCKS, UPWARD
2424 024610 077202      SOB      R2,3$      ;
2425 024612 012737 024672 001110      MOV      #5$,$LPERR ;SET LOOP ON ERROR POINTER TO 5$
2426 024620 012737 025156 000250      MOV      #9$,MMVEC  ;SET M. M. TRAP VECTOR TO 9$
2427 024626 012737 000021 177572      MOV      #21,SRO   ;TURN ON MEMORY MANAGEMENT(22-BIT ADDRESSING)
2428 024634 105037 177610      10$: CLRB   UIPDR4  ;MAP USER SPACE NON-RESIDENT WHILE
2429 024640 105037 177612      CLRB   UIPDR5   ;
2430 024644 010537 177650      MOV      R5,UIPAR4 ;TESTING KERNEL SPACE
2431 024650 010437 177652      MOV      R4,UIPAR5 ;MAP USER PAR'S OPPOSITE OF KIPAR'S
2432 024654 013737 177776 001176      4$: MOV      PSW,$TMP0 ;SAVE PSW IN CASE OF ERROR
2433 024662 012700 100100      MOV      #100100,R0 ;PUT VIRTUAL ADDR. THAT USER PAR4 IN R0
2434 024666 012701 120000      MOV      #120000,R1 ;PUT VIRTUAL ADDR. THAT USES PAR5 IN R1
2435 024672 010010      5$: MOV      R0,(R0)  ;WRITE TO TEST LOC. USING PAR4
2436 024674 011102      MOV      (R1),R2   ;READ THE SAME LOC., BUT USING PAR5
2437 024676 020002      CMP      R0,R2     ;DID WE READ WHAT WE WROTE?
2438 024700 001411      BEQ      6$        ;BRANCH IF YES
2439 024702 010137 001310      MOV      R1,VIRT2  ;SAVE VIRTUAL ADDR. THAT SELECTED PAR5
2440 024706 010037 001306      MOV      R0,VIRT1  ;SAVE VIRTUAL ADDR. THAT SELECTED PAR4
2441 024712 004737 035466      JSR      PC,FORMPA ;GO FORM PHYSICAL ADDRESS BEING USED
2442 024716 104020      ERROR   20        ;READING LOC. USING PAR5 AND A VIRT.
2443      ;ADDR. DID NOT FIND DATA WRITTEN WHEN USING
2444      ;PAR4 AND VIRT. ADDRESS.
2445      ;FOR TIGHTER SCOPE LOOP
2446      ;REPLACE ERROR CALL WITH
2447      ;'BR 5$' = 000765
2448 024720 013700 001306      6$: MOV      VIRT1,R0 ;RESTORE VBA IN R0
2449 024724 062700 000100      ADD      #100,R0   ;CHANGE VIRTUAL ADDRS. TO POINT TO NEXT BLOCK
2450 024730 062701 000100      ADD      #100,R1
2451 024734 020127 127700      CMP      R1,#127700 ;WERE BLOCKS FROM 60000-676000 ALL TRIED?
2452 024740 001354      BNE      5$        ;BRANCH IF NO
2453 024742 032737 140000 177776      BIT      #140000,PSW ;HAVE WE DONE TEST IN USER MODE YET?
2454 024750 001026      BNF      7$        ;BRANCH IF YES
2455 024752 010437 177650      MOV      R4,UIPAR4 ;LOAD USER PAR4
2456 024756 010537 177652      MOV      R5,UIPAR5 ;LOAD USER PAR5
    
```



```

2457 024762 112737 000006 177610      MOVB    #6,UIPDR4      ;MAP USER SPACE R/W TO TEST IT
2458 024770 112737 000006 177612      MOVB    #6,UIPDR5
2459 024776 105037 172310      CLR     KIPDR4        ;MAP KERNEL SPACE NON-RESIDENT WHILE
2460 025002 105037 172312      CLR     KIPDR5        ;      TESTING USER SPACE
2461 025006 010537 172350      MOV     R5,KIPAR4     ;MAP KERNEL PAR'S OPPOSITE UIPAR'S
2462 025012 010437 172352      MOV     R4,KIPAR5
2463 025016 012737 140000 177776      MOV     #140000,PSW   ;GO TO USER MODE
2464 025024 000713          BR      4$            ;GO BACK AND READ/WRITE IN USER MODE
2465 025026 005037 177776      7$:    CLR     PSW      ;GO BACK TO KERNEL MODE BEFORE LEAVING
2466
2467 025032 020537 001260      CMP     R5,#$MADR1    ;HAVE WE CHECKED ALL MEMORY?
2468 025036 002020          BGE     8$            ;BRANCH IF YES
2469 025040 062704 000200      ADD     #200,R4       ;LOAD WITH NEXT VALUE FOR PAR4
2470 025044 062705 000200      ADD     #200,R5       ;LOAD WITH NEXT VALUE FOR PAR5
2471 025050 010437 172350      MOV     R4,KIPAR4     ;LOAD KERNAL PAR4
2472 025054 010537 172352      MOV     R5,KIPAR5     ;LOAD KERNAL PAR5
2473 025060 112737 000006 172310      MOVB    #6,KIPDR4     ;MAP KERNAL SPACE R/W WHILE TESTING
2474 025066 112737 000006 172312      MOVB    #6,KIPDR5
2475 025074 000137 024634      JMP     10$           ;CONTINUE TEST
2476
2477 025100 012737 077406 172310      8$:    MOV     #77406,KIPDR4 ;REMAP KERNEL PAGES READ/WRITE
2478 025106 012737 077406 172312      MOV     #77406,KIPDR5
2479 025114 012705 000600      MOV     #600,R5
2480 025120 010537 172350      MOV     R5,KIPAR4     ;MAP KERNEL AND USER PAR'S 4 & 5
2481 025124 010537 172352      MOV     R5,KIPAR5     ; BACK TO 12-16K
2482 025130 010537 177650      MOV     R5,UIPAR4
2483 025134 010537 177652      MOV     R5,UIPAR5
2484 025140 012737 002150 000250      MOV     #MGMERR,MMVEC ;RESTORE ADDR. OF NORMAL M.M. TRAP ROUTINE
2485 025146 012737 024520 001110      MOV     #1$, $LPERR   ;RESET LOOP ON ERROR POINTER TO 1$
2486 025154 000427          BR      TST26        ;:GO TO NEXT TEST
2487
2488 025156 012637 001266      9$:    MOV     (KSP)+,TRAPPC ;SAVE PC & PS OF TRAP
2489 025162 012637 001270      MOV     (KSP)+,TRAPPS
2490
2491          ;PROGRAM WILL TRAP TO HERE IF TRY
2492          ;TO USE USER PDR'S WHEN IN KERNEL MODE
2493          ;OR KERNEL PDR'S WHEN IN USER MODE
2493 025166 010037 001306      MOV     R0,VIRT1      ;SAVE VIRTUAL ADDRESS FOR ERROR REPORT
2494 025172 004737 035466      JSR     PC,FORMPA     ;GO FORM THE PHYSICAL ADDRESS BEING USED
2495 025176 013737 177572 001272      MOV     SR0,WASSRO    ;SAVE SR0 & SR2 FOR ERROR REPORT
2496 025204 013737 177576 001274      MOV     SR2,WASSR2
2497 025212 042737 160000 177572      BIC     #160000,SR0   ;CLEAR ERROR BITS IN SR0
2498 025220 104052          ERROR  52            ;M.M. TRAP WHILE IN RELOCATE MODE -
2499          ;REFERENCED WRONG SET OF PDR'S
2500          ;FOR TIGHTER SCOPE LOOP
2501          ;REPLACE ERROR CALL WITH
2502          ;A 'NOP' = 000240
2503 025222 013746 001270      MOV     TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
2504 025226 013746 001266      MOV     TRAPPC,-(KSP)
2505 025232 000002          RTI                    ;RETURN TO TEST
2506
2507
2508          ;*****
2509          ;*TEST 26      W-BIT LOGIC TEST, KERNEL PDR'S
2510          ;*
2511          ;*      THIS TEST WRITES TO EIGHT (8) DIFFERENT VIRTUAL ADDRESSES
2512          ;*      (VBA'S = 17776,37776,57776,77776,117776,137776,157776, & 177776
    
```

```

2513      ;*      & PBA'S CONSTRUCTED = 17776,37776,57776,77776,77776,
2514      ;*      77776,77776, & 777776 RESPECTIVELY).
2515      ;*      WHICH SHOULD CAUSE THE 'W-BIT' TO SET IN EACH OF THE
2516      ;*      EIGHT (8) KERNEL PAGE DESCRIPTOR REGISTERS. THE PDR'S
2517      ;*      ARE CHECKED TO SEE THAT IT'S W-BIT DOES SET WHEN THE
2518      ;*      PAGE IT IS MAPPED TO IS WRITTEN TO AND THAT THE W-BIT
2519      ;*      DOES NOT SET IN ANY OF THE OTHER PDR'S. KERNEL PDR'S 3,4,5,6
2520      ;*      ARE MAPPED TO 12-16K FOR THIS TEST. ALSO THE W-BIT
2521      ;*      SHOULD BE CLEARED WHEN THE PDR IS WRITTEN TO. THE
2522      ;*      W-BIT PORTION OF THE PDR'S IS BEING CHECKED.
2523      ;*****
2524 025234 000004      TST26: SCOPE
2525 025236      1$:
2526 025236 004737 035114      JSR      PC,TOFF      ;TURN T-BIT TRAPPING OFF FOR THIS TEST
2527 025242 012702 000004      MOV      #4,R2      ;SET LOOP COUNTER TO 4
2528 025246 012700 172346      MOV      #KIPAR3,R0 ;LOAD ADDRESS OF PAR3 INTO R0
2529 025252 012701 000600      MOV      #600,R1    ;LOAD '12-16K' PAR VALUE INTO R1
2530 025256 010120      2$:      MOV      R1,(R0)+    ;MAP PARS 3-6 TO 12-16K
2531 025260 077202      SGB      R2,2$      ;LOOP TIL ALL 4 OF THEM LOADED
2532 025262 012705 172300      MOV      #KIPDR0,R5 ;LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
2533 025266 012704 000010      MOV      #10,R4     ;SET LOOP COUNTER TO 8
2534 025272 012703 017776      MOV      #17776,R3  ;INITIALIZE VIRTUAL ADDRESS TO BE IN R3
2535 025276 012737 025304 001110      MOV      #3$,SLPERR ;SET LOOP ON ERROR POINTER TO 3$
2536 025304 012700 172300      3$:      MOV      #KIPDR0,R0 ;LOAD ADDR. OF FIRST PDR TO BE SETUP IN R0
2537 025310 012702 000010      MOV      #10,R2     ;SET LOOP COUNTER TO 8
2538 025314 012701 077406      MOV      #77406,R1  ;PUT 'W-BIT OFF DATA' INTO R1
2539 025320 010120      4$:      MOV      R1,(R0)+    ;CLEAR ALL W-BITS BY WRITING TO ALL PDRS
2540 025322 077202      SOB      R2,4$      ;LOOP UNTIL ALL OF THEM SETUP
2541 025324 011313      MOV      (R3),(R3)  ;DO 'DATA' TO VIRTUAL ADDR.-SETTING A W-BIT
2542 025326 031527 000100      BIT      (R5),#WBIT ;DID THAT CAUSE W-BIT TO BE SET?
2543 025332 001002      BNE      5$        ;BRANCH IF YES
2544 025334 104021      ERROR    21        ;W-BIT DID NOT GET SET IN PDR
2545      ;FOR TIGHTER SCOPE LOOP
2546      ;REPLACE ERROR CALL WITH
2547      ;'BR 3$' = 00C763
2548 025336 000422      BR      8$        ;SKIP CHECKING OTHER PDR'S-ERROR WILL SET W-BITS
2549 025340 012702 000010      5$:      MOV      #10,R2     ;SET LOOP COUNTER TO 8
2550 025344 012700 172300      MOV      #KIPDR0,R0 ;LOAD ADDR. OF FIRST PDR TO BE CHECKED IN R0
2551 025350 031027 000100      6$:      BIT      (R0),#WBIT ;DID W-BIT IN OTHER PDRS REMAIN CLEAR?
2552 025354 001403      BEQ      7$        ;BRANCH IF YES
2553 025356 020500      CMP      R5,R0     ;IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
2554 025360 001401      BEQ      7$        ;BRANCH IF YES
2555 025362 104022      ERROR    22        ;W-BIT GOT SET IN MORE THAN ONE PDR
2556      ;FOR TIGHTER SCOPE LOOP
2557      ;REPLACE ERROR CALL WITH
2558      ;'BR 3$' = 000750
2559 025364 062700 000002      7$:      ADD      #2,R0     ;POINT R0 TO NEXT PDR TO BE CHECKED
2560 025370 077211      SOB      R2,6$      ;LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
2561 025372 010115      MOV      R1,(R5)    ;WRITE TO THE PDR TESTED TO CLEAR W-BIT
2562 025374 031527 000100      BIT      (R5),#WBIT ;DID WRITING PDR CLEAR THE W-BIT?
2563 025400 001401      BEQ      8$        ;BRANCH IF YES
2564 025402 104023      ERROR    23        ;W-BIT DID NOT CLEAR BY WRITING THE PDR
2565      ;FOR TIGHTER SCOPE LOOP
2566      ;REPLACE ERROR CALL WITH
2567      ;'BR 3$' = 000740
2568 025404 062705 000002      8$:      ADD      #2,R5     ;POINT R5 TO THE NEXT PDR TO BE TESTED
    
```

```

2569 025410 062703 020000      ADD    #20000,R3      ;CHANGE VIRT. ADDR TO REF. NEXT PDR
2570 025414 077445              SOB    R4,3$         ;LOOP BACK TO 3$ UNTIL ALL 8 PDR'S TESTED
2571 025416 012737 025236 001110  MOV    #1$, $LPERR   ;RESET LOOP ON ERROR POINTER TO 1$
2572 025424 004737 035150      JSR    PC,TON        ;TURN T-BIT BACK ON FOR NEXT TEST
2573
2574
2575 :*****
2576 :*TEST 27      W-BIT LOGIC TEST, USER PDR'S
2577 :*
2578 :*      THIS TEST WRITES TO EIGHT (8) DIFFERENT VIRTUAL ADDRESSES
2579 :*      (VBA'S = 17776,37776,57776,77776,117776,137776,157776, & 177776
2580 :*      & PBA'S CONSTRUCTED = 17776,37776,57776,77776,77776,
2581 :*      77776,77776, & 77776 RESPECTIVELY).
2582 :*      WHICH SHOULD CAUSE THE 'W-BIT' TO SET IN EACH OF THE
2583 :*      EIGHT (8) USER PAGE DESCRIPTOR REGISTERS.  THE PDR'S
2584 :*      ARE CHECKED TO SEE THAT IT'S W-BIT DOES SET WHEN THE
2585 :*      PAGE IT IS MAPPED TO IS WRITTEN TO AND THAT THE W-BIT
2586 :*      DOES NOT SET IN ANY OF THE OTHER PDR'S.  USER PDR'S 3,4,5,6
2587 :*      ARE MAPPED TO 12-16K FOR THIS TEST.  ALSO THE W-BIT
2588 :*      SHOULD BE CLEARED WHEN THE PDR IS WRITTEN TO.  THE
2589 :*      W-BIT PORTION OF THE PDR'S IS BEING CHECKED.
2590 :*****
2590 025430 000004      TST27: SCOPE
2591 025432 012737 140000 177776 1$:  MOV    #140000,PSW   ;GO TO USER MODE FOR THIS TEST
2592 025440 004737 035114              JSR    PC,TOFF      ;TURN T-BIT TRAPPING OFF FOR THIS TEST
2593 025444 012702 000004              MOV    #4,R2        ;SET LOOP COUNTER TO 4
2594 025450 012700 177646              MOV    #UIPAR3,R0   ;LOAD ADDRESS OF PAR3 INTO R0
2595 025454 012701 000600              MOV    #600,R1      ;LOAD '12-16K' PAR VALUE INTO R1
2596 025460 010120      2$:  MOV    R1,(R0)+      ;MAP PARS 3-6 TO 12-16K
2597 025462 077202              SOB    R2,2$        ;LOOP TIL ALL 4 OF THEM LOADED
2598 025464 012705 177600              MOV    #UIPDR0,R5   ;LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
2599 025470 012704 000010              MOV    #10,R4       ;SET LOOP COUNTER TO 8
2600 025474 012703 017776              MOV    #17776,R3    ;INITIALIZE VIRTUAL ADDRESS TO BE IN R3
2601 025500 012737 025506 001110      3$:  MOV    #3$, $LPERR   ;SET LOOP ON ERROR POINTER TO 3$
2602 025506 012700 177600              MOV    #UIPDR0,R0   ;LOAD ADDR. OF FIRST PDR TO BE SETUP IN R0
2603 025512 012702 000010              MOV    #10,R2       ;SET LOOP COUNTER TO 8
2604 025516 012701 077406              MOV    #77406,R1    ;PUT 'W-BIT OFF DATA' INTO R1
2605 025522 010120      4$:  MOV    R1,(R0)+      ;CLEAR ALL W-BITS BY WRITING TO ALL PDRS
2606 025524 077202              SOB    R2,4$        ;LOOP UNTIL ALL OF THEM SETUP
2607 025526 011313              MOV    (R3),(R3)    ;DO 'DATO' TO VIRTUAL ADDR.-SETTING A W-BIT
2608 025530 031527 000100              BIT    (R5),#WBIT   ;DID THAT CAUSE W-BIT TO BE SET?
2609 025534 001002              BNE    5$           ;BRANCH IF YES
2610 025536 104021              ERROR  21          ;W-BIT DID NOT GET SET IN PDR
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2614 025540 000422      5$:  BR     8$           ;FOR TIGHTER SCOPE LOOP
2615 025542 012702 000010              MOV    #10,R2        ;REPLACE ERROR CALL WITH
2616 025546 012700 177600              MOV    #UIPDR0,R0   ;'BR 3$' = 000763
2617 025552 031027 000100      6$:  BIT    (R0),#WBIT   ;SKIP CHECKING OTHER PDR'S-ERROR WILL SET W-BITS
2618 025556 001403              BEQ    7$           ;SET LOOP COUNTER TO 8
2619 025560 020500              CMP    R5,R0        ;LOAD ADDR. OF FIRST PDR TO BE CHECKED IN R0
2620 025562 001401              BEQ    7$           ;DID W-BIT IN OTHER PDRS REMAIN CLEAR?
2621 025564 104022              ERROR  22          ;BRANCH IF YES
2622
2623
2624
2621 025564 104022              ERROR  22          ;BRANCH IF YES
2622
2623
2624
2621 025564 104022              ERROR  22          ;IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
2622
2623
2624
2621 025564 104022              ERROR  22          ;BRANCH IF YES
2622
2623
2624
2621 025564 104022              ERROR  22          ;W-BIT GOT SET IN MORE THAN ONE PDR
2622
2623
2624
2621 025564 104022              ERROR  22          ;FOR TIGHTER SCOPE LOOP
2622
2623
2624
2621 025564 104022              ERROR  22          ;REPLACE ERROR CALL WITH
2622
2623
2624
2621 025564 104022              ERROR  22          ;'BR 3$' = 000750
    
```

```

2625 025566 062700 000002      7$:  ADD      #2,R0      ;POINT R0 TO NEXT PDR TO BE CHECKED
2626 025572 077211             SOB      R2,6$      ;LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
2627 025574 010115             MOV      R1,(R5)    ;WRITE TO THE PDR TESTED TO CLEAR W-BIT
2628 025576 031527 000100     BIT      (R5),#WBIT ;DID WRITING PDR CLEAR THE W-BIT?
2629 025602 001401             BEQ      8$         ;BRANCH IF YES
2630 025604 104023             ERROR    23         ;W-BIT DID NOT CLEAR BY WRITING THE PDR
2631                                     ;FOR TIGHTER SCOPE LOOP
2632                                     ;REPLACE ERROR CALL WITH
2633                                     ;'BR 3$' = 000740
2634 025606 062705 000002      8$:  ADD      #2,R5      ;POINT R5 TO THE NEXT PDR TO BE TESTED
2635 025612 062703 020000     ADD      #2000,R3   ;CHANGE VIRT. ADDR TO REF. NEXT PDR
2636 025616 077445             SOB      R4,3$      ;LOOP BACK TO 3$ UNTIL ALL 8 PDR'S TESTED
2637 025620 012737 025432 001110  MOV      #1$,$LPERR ;RESET LOOP ON ERROR POINTER TO 1$
2638 025626 004737 035150     JSR      PC,TON     ;TURN T-BIT BACK ON FOR NEXT TEST
2639 025632 005037 177776     CLR      PSW        ;BACK TO KERNEL MODE BEFORE LEAVING
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651 025636 000004      TST30: SCOPE
2652
2653 025640 004737 035114      1$:  JSR      PC,TOFF   ;TURN OFF T-BIT TRAPPING FOR THIS TEST
2654 025644 012701 077406     MOV      #77406,R1  ;PUT 'W-BIT OFF' VALUE FOR PDR IN R1
2655 025650 012737 025656 001110  MOV      #2$,$LPERR ;SET LOOP ON ERROR POINTER TO 2$
2656 025656 010137 172316     MOV      R1,KIPDR7  ;LOAD KERNEL PDR 7 TO CLEAR W-BIT
2657 025662 013700 177572     MOV      SR0,R0     ;READ PRESENT CONTENTS OF STATUS REG. 0
2658 025666 010037 177572     MOV      R0,SR0     ;WRITE PRESENT CONTENTS OF SR0 BACK TO ITSELF
2659 025672 013702 172316     MOV      KIPDR7,R2  ;READ CONTENTS OF KIPDR7 INTO R2
2660 025676 020102             CMP      R1,R2      ;WAS W-BIT LEFT CLEARED?
2661 025700 001401             BEQ      3$         ;BRANCH IF YES
2662 025702 104024             ERROR    24         ;W-BIT IN KIPDR7 SET WHEN SR0 WAS WRITTEN TO
2663                                     ;FOR TIGHTER SCOPE LOOP
2664                                     ;REPLACE ERROR CALL WITH
2665                                     ;'BR 2$' = 000765
2666 025704 012737 025704 001110  3$:  MOV      #3$,$LPERR ;SET LOOP ON ERROR POINTER TO 3$
2667 025712 010137 172314     MOV      R1,KIPDR6  ;LOAD KERNEL PDR6 WITH 77406 TO CLEAR W-BIT
2668 025716 012737 025730 000004  MOV      #4$,ERRVEC ;SET UP LOC. 4 TO 4$ FOR ODD ADDR. ABORT
2669 025724 005037 140000     CLR      @#140000   ;CAUSE TIMEOUT ABORT THRU LOC. 4
2670 025730 012706 001100     4$:  MOV      #KERSTK,KSP ;RESTORE THE STACK POINTER
2671 025734 013702 172314     MOV      KIPDR6,R2  ;READ KIPDR6 INTO R2
2672 025740 052701 000100     BIS      #100,R1     ;R1-77506
2673 025744 020102             CMP      R1,R2      ;WAS W-BIT SET?
2674 025746 001401             BEQ      5$         ;BRANCH IF YES
2675 025750 104025             ERROR    25         ;W-BIT WAS NOT SET DURING A TIMEOUT ABORT
2676                                     ;FOR TIGHTER SCOPE LOOP
2677                                     ;REPLACE ERROR CALL WITH
2678                                     ;'BR 3$' = 000757
2679 025752 010137 172314     5$:  MOV      R1,KIPDR6  ;RESTORE KIPDR6 TO 77406
2680 025756 012737 001400 172354  MOV      #1400,KIPAR6 ;RESTORE KIPAR6 TO 1400
    
```

```

2681 025764 012737 002076 000004      MOV      #TIMERR,ERRVEC ;RESTORE NORMAL CPU TRAP ROUTINE TO LOC.4
2682 025772 012737 025640 001110      MOV      #1$, $LPERR   ;RESET LOOP ON ERROR POINTER TO 1$
2683 026000 004737 035150      JSR      PC,TON        ;TURN T-BIT TRAPPING BACK ON
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705

```

```

:*****
:
: THE NEXT THREE (3) TESTS CAUSE MEMORY MANAGEMENT ERRORS
: TO CHECK THE ABILITY OF STATUS REGISTER 0 TO RECORD KT
: ERRORS AND THE ABILITY OF STATUS REGISTER 2 TO LOCK UP THE
: VIRTUAL ADDR. OF THE INSTRUCTION THAT CAUSED THE ERROR.
: THE BITS OF SR2 ARE CHECKED AND BITS <15:13>, <6:5>, AND <3:0>
: ARE CHECKED IN SRO. SO THE SRO AND SR2 LOGIC AND THE
: KT ERROR LOGIC ARE CHECKED.
:*****

```

```

2706 026004 000004
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736

```

```

:*****
:TEST 31          NON-RESIDENT ABORT TEST (ACF=084)
:
: THIS TEST CHECKS THE ACCESS CONTROL FIELD (ACF) COMPARATOR
: LOGIC BY CAUSING NON-RESIDENT ABORTS IN BOTH KERNEL AND
: USER MODES. PDR 4 IS LOADED WITH ACF'S = 084 AND
: THEN PHYSICAL ADDR. 60000 IS ACCESSED TO CAUSE THE ABORT.
:*****

```

```

TST31: SCOPE
1$:  MOV      #600,R0          ;LOAD DATA FOR PAR'S INTO R0
      MOV      R0,KIPAR3      ;MAP KERNEL PAR'S 3&4 TO 12-16K
      MOV      R0,KIPAR4
      MOV      R0,UIPAR3      ;MAP USER PAR'S 3&4 TO 12-16K
      MOV      R0,UIPAR4
      MOV      #77406,KIPDR3  ;MAP KERNEL PDR 3 128 BLKS, READ-WRITE
      MOV      #77406,UIPDR3  ;MAP USER PDR 3 128 BLKS, READ-WRITE
      MOV      #60000,R0      ;LOAD VIRTUAL ADDR. TO REFERENCE PDR3 INTO R0
      MOV      #100000,R1     ;LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
      MOV      #100011,R3     ;LOAD R3 WITH WHAT SRO SHOULD READ - N.R., KERNEL, PG.4
      MOV      #77400,R2     ;LOAD ACF=0 (NON-RESIDENT) PDR VALUE IN R2
2$:  MOV      #5$,MMVEC       ;POINT MEM. MGMT. TRAP VECTOR TO 5$ BELOW
      MOV      R2,KIPDR4      ;LOAD ACF TEST VALUE INTO KIPDR4
      MOV      R2,UIPDR4      ;LOAD ACF TEST VALUE INTO UIPDR4
      MOV      #3$, $LPERR    ;SET LOOP ON ERROR POINTER TO 3$
3$:  CLR      (R0)            ;CLEAR PHYS. LOC. 60000 USING PDR3
      MOV      PSW,$TMP0      ;SAVE PSW IN CASE OF ERROR
4$:  INC      (R1)            ;TRY TO REF. IT USING PDR4 - SHOULD TRAP TO 5$
      ERROR 26                ;MEM. MGMT. ABORT DID NOT OCCUR
      ;FOR TIGHTER SCOPE LOOP
      ;REPLACE ERROR CALL WITH
      ;'BR 3$' = 000772
5$:  BR      8$                ;BRANCH AROUND STATUS REG. CHECKS IF NO ABORT
      ADD      #4,$SP         ;RESTORE STACK POINTER
      TST      (R0)           ;DID INSTRUCTION GET ABORTED & NOT EXECUTE
      BEQ      6$             ;BRANCH IF YES
      ERROR 27                ;INSTRUCTION WAS NOT ABORTED, LOC. GOT CHANGED
      ;FOR TIGHTER SCOPE LOOP
      ;REPLACE ERROR CALL WITH

```

```

2737
2738 026142 013737 177572 001272 6$: MOV SRO,WASSRO ;'BR 3$' = 000764
2739 026150 013737 177576 001274 MOV SR2,WASSR2 ;READ STATUS REGISTER 0
2740 026156 020337 001272 CMP R3,WASSRO ;READ STATUS REGISTER 2
2741 026162 001401 BEQ 7$ ;DID SRO REPORT NON-RESIDENT ERROR CORRECTLY?
2742 026164 104030 ERROR 30 ;BRANCH IF YES
2743 ;SRO DID NOT REPORT NON-RES. ERROR CORRECTLY
2744 ;FOR TIGHTER SCOPE LOOP
2745 ;REPLACE ERROR CALL WITH
2746 026166 012704 026122 7$: MOV #4$,R4 ;'BR 3$' = 000752
2747 026172 020437 001274 CMP R4,WASSR2 ;LOAD R4 WITH WHAT SR2 SHOULD READ
2748 026176 001401 BEQ 8$ ;DID SR2 LOCKUP RIGHT VIRTUAL ADDR. (=4$)?
2749 026200 104031 ERROR 31 ;BRANCH IF YES
2750 ;SR2 DID NOT LOCK VIRTUAL ADDR. OF NON-RES. ERROR
2751 ;FOR TIGHTER SCOPE LOOP
2752 ;REPLACE ERROR CALL WITH
2753 026202 042737 160000 177572 8$: BIC #160000,SRO ;'BR 3$' = 000744
2754 026210 032737 140000 001176 BIT #140000,$TMP0 ;CLEAR THE ERROR BITS IN SRO
2755 026216 001006 BNE 9$ ;HAS ACF=0&4 BEEN TESTED IN USER YET
2756 026220 012703 100151 MOV #100151,R3 ;BRANCH IF YES
2757 026224 012737 140000 177776 MOV #140000,PSW ;LOAD R3 WITH WHAT SRO SHOULD READ - N.R., USER, PG.4
2758 026232 000715 BR 2$ ;GO TO USER MODE
2759 026234 022702 077404 9$: CMP #77404,R2 ;REPEAT TEST IN USER MODE
2760 026240 001407 BEQ 10$ ;HAS ACF=4 BEEN TESTED YET?
2761 026242 012702 077404 MOV #77404,R2 ;BRANCH IF YES
2762 026246 012703 100011 MOV #100011,R3 ;THEN LOAD ACF=4 (NON-RES) PDR VALUE IN R2
2763 026252 005037 177776 CLR PSW ;LOAD R3 WITH WHAT SRO SHOULD READ-N.R.,KERNEL,PG. 4
2764 026256 000703 BR 2$ ;GO BACK TO KERNEL MODE
2765 026260 005037 177776 10$: CLR PSW ;GO BACK & TEST ACF=4 IN SAME MODE
2766 026264 012737 026006 001110 MOV #1$, $LPERR ;GO BACK TO KERNEL MODE BEFORE LEAVING
2767 026272 012737 002150 000250 MOV #MGMERR,MMVEC ;RESET LOOP ON ERROR POINTER TO 1$
2768 ;RESTORE ADDRESS OF NORMAL MEMORY
2769 ;MANAGEMENT ERROR ROUTINE TO MMVEC
2770
2771 ;*****
2772 ;*TEST 32 READ-ONLY ABORT TEST (ACF=2)
2773 ;*
2774 ;* THIS TEST CHECKS THE ACCESS CONTROL FIELD (ACF) COMPARATOR
2775 ;* LOGIC BY CAUSING READ-ONLY ABORTS IN BOTH KERNEL AND
2776 ;* USER MODES. PDR 4 IS LOAD WITH ACF=2 AND THEN
2777 ;* PHYSICAL ADDR. 60000 IS WRITTEN TO CAUSE THE ABORT.
2778 ;*
2779 ;*****
2780 026300 000004 TST32: SCOPE
2781 026302 1$: ;KERNEL & USER PAR'S 3 & 4 AND PDR 3
2782 026302 012700 060000 MOV #60000,R0 ;ARE SETUP FROM LAST TEST
2783 026306 012701 100000 MOV #100000,R1 ;LOAD VIRTUAL ADDR. TO REFERENCE PDR3 INTO R0
2784 026312 012703 020011 MOV #20011,R3 ;LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
2785 026316 012702 077402 MOV #77402,R2 ;LOAD R3 WITH WHAT SRO SHOULD READ - R/O, KERNEL, PG.4
2786 026322 012737 026364 000250 2$: MOV #5$,MMVEC ;LOAD ACF=2 (READ-ONLY) PDR VALUE IN R2
2787 026330 010237 172310 MOV R2,KIPDR4 ;POINT MEM. MGMT. TRAP VECTOR TO 5$ BELOW
2788 026334 010237 177610 MOV R2,UIPDR4 ;LOAD ACF=2 INTO KIPDR4
2789 026340 012737 026346 001110 MOV #3$, $LPERR ;LOAD ACF=2 INTO UIPDR4
2790 026346 005010 3$: CLR (R0) ;SET LOOP ON ERROR POINTER TO 3$
2791 026350 013737 177776 001176 MOV PSW,$TMP0 ;CLEAR PHYS. LOC. 60000 USING PDR3
2792 026356 005211 4$: INC (R1) ;SAVE PSW IN CASE OF ERROR
;TRY TO WRITE USING PDR4 - SHOULD TRAP TO 5$

```

```

2793 026360 104026          ERROR 26          ;MEM. MGMT. ABORT DID NOT OCCUR
2794                                     ;FOR TIGHTER SCOPE LOOP
2795                                     ;REPLACE ERROR CALL WITH
2796                                     ;'BR 3$' = 000772
2797 026362 000425          BR          8$          ;BRANCH AROUND STATUS REG. CHECKS IF NO ABORT
2798 026364 062706 000004    5$:  ADD      #4,SP      ;RESTORE STACK POINTER
2799 026370 005710          TST      (R0)         ;DID INSTRUCTION GET ABORTED & NOT EXECUTE
2800 026372 001401          BEQ      6$          ;BRANCH IF YES
2801 026374 104027          ERROR 27          ;INSTRUCTION WAS NOT ABORTED, LOC. GOT CHANGED
2802                                     ;FOR TIGHTER SCOPE LOOP
2803                                     ;REPLACE ERROR CALL WITH
2804                                     ;'BR 3$' = 000764
2805 026376 013737 177572 001272 6$:  MOV      SRO,WASSRO   ;READ STATUS REG. 0
2806 026404 013737 177576 001274    MOV      SR2,WASSR2   ;READ STATUS REG. 2
2807 026412 020337 001272    CMP      R3,WASSR0   ;DID SRO REPORT READ-ONLY ERROR CORRECTLY?
2808 026416 001401          BEQ      7$          ;BRANCH IF YES
2809 026420 104030          ERROR 30          ;SRO DID NOT REPORT R/O ERROR CORRECTLY
2810                                     ;FOR TIGHTER SCOPE LOOP
2811                                     ;REPLACE ERROR CALL WITH
2812                                     ;'BR 3$' = 000752
2813 026422 012704 026356 7$:  MOV      #4$,R4       ;LOAD R4 WITH WHAT SR2 SHOULD READ
2814 026426 020437 001274    CMP      R4,WASSR2   ;DID SR2 LOCKUP RIGHT VIRTUAL ADDR. (=4$)?
2815 026432 001401          BEQ      8$          ;BRANCH IF YES
2816 026434 104031          ERROR 31          ;SR2 DID NOT LOCKUP VIRTUAL ADDR. OF R/O ERROR
2817                                     ;FOR TIGHTER SCOPE LOOP
2818                                     ;REPLACE ERROR CALL WITH
2819                                     ;'BR 3$' = 000744
2820 026436 042737 160000 177572 8$:  BIC      #160000,SRO  ;CLEAR THE ERROR BITS IN SRO
2821 026444 032737 140000 001176    BIT      #140000,$TMP0 ;HAS ACF=2 BEEN TESTED IN USER MODE?
2822 026452 001006          BNE      9$          ;BRANCH IF YES
2823 026454 012703 020151    MOV      #20151,R3   ;LOAD R3 WITH WHAT SRO SHOULD READ-R/O, USER, PG.4
2824 026460 012737 140000 177776    MOV      #140000,PSW  ;GO TO USER MODE
2825 026466 000715          BR        2$          ;REPEAT TEST IN USER MODE
2826 026470 005037 177776 9$:  CLR      PSW         ;GO BACK TO KERNEL MODE BEFORE LEAVING
2827 026474 012737 026302 001110    MOV      #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
2828 026502 012737 002150 000250    MOV      #MGMERR,MMVEC ;RESTORE ADDRESS OF NORMAL MEMORY
2829                                     ;MANAGEMENT ERROR ROUTINE TO MMVEC.
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
    
```

```

:*****
:*
:* THE NEXT TWO (2) TESTS WILL BE CHECKING THE PAGE LENGTH
:* COMPARATORS AND SOME MORE OF THE KT ERROR DETECTION
:* AND STATUS LOGIC. THE PAGE LENGTH FIELD (PLF) IN KERNEL
:* PDR 4 IS VARIED AND FOR EVERY PLF, THREE (3) VIRTUAL
:* ADDRESSES ARE READ. WHILE USING BOTH UPWARD & DOWNWARD PAGE
:* EXPANSION, ONE OF THOSE THREE VIRTUAL ADDRESSES WILL CAUSE A
    
```



```

2849      : *      'PAGE LENGTH ABORT' WHILE THE OTHER TWO WON'T.
2850      : *
2851      : *      STATUS REGISTER 0 & 2 ARE CHECKED WHEN THE PAGE LENGTH
2852      : *      ABORT DOES OCCUR TO SEE THAT THE ABORT IS REPORTED AND THAT
2853      : *      THE VIRTUAL ADDRESS OF THE INSTRUCTION THAT CAUSED THE ABORT
2854      : *      IS LOCKED UP.
2855      : *
2856      : *****
2857
2858
2859
2860      : *****
2861      : *TEST 33      PAGE LENGTH FAULTS-UPWARD EXPANSION
2862      : *
2863      : *      THIS TEST VARIES THE PAGE LENGTH FIELD (PLF) IN KERNEL PDR 4
2864      : *      FROM 1 TO 177 AND FOR EACH PLF, THREE VIRTUAL ADDRESSES (VBA'S)
2865      : *      ARE ACCESSED. WHEN VBA <12:6> IS LESS THAN OR EQUAL TO PDR <14:8>
2866      : *      NO ABORT SHOULD OCCUR. WHEN VBA <12:6> IS GREATER THAN PDR <14:8>,
2867      : *      A PAGE LENGTH ABORT SHOULD OCCUR AND BE REPORTED BY SR0 & SR2.
2868      : *      THE PAGE EXPANSION DIRECTION IN THIS TEST IS UPWARD, (THE ED BIT
2869      : *      (BIT 3) OF PDR 4 = 0).
2870      : *
2871      : *****
2872      026510 000004
2873      026512 012737 077406 172306
2874      026520 012737 077406 172312
2875      026526 012700 027006
2876      026532 012704 027024
2877      026536 012701 000006
2878      026542 012737 026720 000250
2879      026550 012737 026562 001110
2880      026556 012706 001100
2881
2882      :TEST NON-ABORT CASES (VBA < OR = PLF)
2883      026562 012437 172310
2884      026566 005730
2885
2886      026570 077104
2887
2888      :TEST ABORT CASES (VBA > PLF)
2889      026572 012701 000005
2890      026576 012700 027042
2891      026602 012704 027056
2892      026606 012737 026622 001110
2893      026614 012737 026634 000250
2894
2895      026622 012437 172310
2896      026626 005730
2897      026630 104033
2898
2899
2900
2901      026632 000424
2902      026634 012706 001100
2903      026640 013737 177572 001272
2904      026646 013737 177576 001274
    
```

```

: *      'PAGE LENGTH ABORT' WHILE THE OTHER TWO WON'T.
: *
: *      STATUS REGISTER 0 & 2 ARE CHECKED WHEN THE PAGE LENGTH
: *      ABORT DOES OCCUR TO SEE THAT THE ABORT IS REPORTED AND THAT
: *      THE VIRTUAL ADDRESS OF THE INSTRUCTION THAT CAUSED THE ABORT
: *      IS LOCKED UP.
: *
: *****
: *****
: *TEST 33      PAGE LENGTH FAULTS-UPWARD EXPANSION
: *
: *      THIS TEST VARIES THE PAGE LENGTH FIELD (PLF) IN KERNEL PDR 4
: *      FROM 1 TO 177 AND FOR EACH PLF, THREE VIRTUAL ADDRESSES (VBA'S)
: *      ARE ACCESSED. WHEN VBA <12:6> IS LESS THAN OR EQUAL TO PDR <14:8>
: *      NO ABORT SHOULD OCCUR. WHEN VBA <12:6> IS GREATER THAN PDR <14:8>,
: *      A PAGE LENGTH ABORT SHOULD OCCUR AND BE REPORTED BY SR0 & SR2.
: *      THE PAGE EXPANSION DIRECTION IN THIS TEST IS UPWARD, (THE ED BIT
: *      (BIT 3) OF PDR 4 = 0).
: *****
TSI33: SCOPE
1$:  MOV      #77406,KIPDR3      ;MAKE SURE PDR3 IS DESCRIBED AS R/W
    MOV      #77406,KIPDR5      ;MAKE SURE PDR5 IS DESCRIBED AS R/W
    MOV      #DALTB1,R0         ;DAL TABLE FOR VIRTUAL ADDR'S. TO SELECT PDR4.
    MOV      #PDRTB1,R4        ;PDR TABLE FOR PDR4 (COINCIDES WITH DAL TABLE).
    MOV      #6,R1              ;SET UP LOOP COUNTER.
    MOV      #9$,MMVEC         ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS
    MOV      #2$, $LPERR        ;SET LOOP ON ERROR POINTER TO 2$
    MOV      #KERSTK,KSP       ;MAKE SURE STACK POINTER IS ALL SET UP

:TEST NON-ABORT CASES (VBA < OR = PLF)
2$:  MOV      (R4)+,KIPDR4      ;LOAD KIPDR4 WITH PAGE LENGTH VALUE
    TST      @ (R0)+           ;ACCESS VIRTUAL ADDR. (VBA < OR = PLF)
                                ;NO ABORT SHOULD OCCUR!!!
    SOB      R1,2$            ;DONE?...NO- TEST NEXT COMBINATION OF DAL & PDR.

:TEST ABORT CASES (VBA > PLF)
3$:  MOV      #5,R1              ;SET UP LOOP COUNTER.
    MOV      #DALTB2,R0         ;DAL TABLE
    MOV      #PDRTB2,R4        ;PDR TABLE
    MOV      #4$, $LPERR        ;SET LOOP ON ERROR POINTER TO 4$
    MOV      #6$,MMVEC         ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT

4$:  MOV      (R4)+,KIPDR4      ;LOAD KIPDR4 WITH PAGE LENGTH VALUE
5$:  TST      @ (R0)+           ;ACCESS VIRTUAL ADDR. (VBA > PLF - ABORT TO 6$)
    ERROR    3,                ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
                                ;FOR TIGHTER SCOPE LOOP
                                ;REPLACE ERROR CALL WITH
                                ;'BR 5$' = 000776
6$:  BR      8$
    MOV      #KERSTK,KSP       ;RESTORE STACK POINTER FOLLOWING ABORT
    MOV      SR0,WASSR0        ;READ M.M. STATUS REG. 0
    MOV      SR2,WASSR2        ;READ M.M. STATUS REG. 2
    
```



```

2905 026654 012702 040011      MOV    #40011,R2      ;PUT EXPECTED SRO CONTENTS IN R2
2906 026660 020237 001272      CMP    R2,WASSR0     ;DID SRO REPORT PG. LENGTH ABORT, PAGE 4, KERNEL?
2907 026664 001401              BEQ    7$            ;BRANCH IF YES
2908 026666 104034              ERROR  34           ;SRO DID NOT REPORT PG. LENGTH ABORT CORRECTLY
2909                                ;FOR TIGHTER SCOPE LOOP
2910                                ;REPLACE ERROR CALL WITH
2911                                ;'BR 5$' = 000757
2912 026670 012703 026626      7$:  MOV    #5$,R3      ;PUT EXPECTED SR2 CONTENTS IN R3
2913 026674 020337 001274      CMP    R3,WASSR2     ;DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?
2914 026700 001401              BEQ    8$            ;BRANCH IF YES
2915 026702 104035              ERROR  35           ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY
2916                                ;FOR TIGHTER SCOPE LOOP
2917                                ;REPLACE ERROR CALL WITH
2918                                ;'BR 5$' = 000751
2919 026704 042737 160000 177572 8$:  BIC    #160000,SRO   ;CLEAR ERROR BITS IN SRO
2920 026712 077135              SOB    R1,4$         ;DONE?..NO - GET NEXT DAL & PDR PAIR
2921 026714 000137 026766              JMP    10$          ;YES...
2922 026720 012637 001266      9$:  MOV    (KSP)+,TRAPPC ;SAVE PC & PS OF TRAP
2923 026724 012637 001270              MOV    (KSP)+,TRAPPS
2924 026730 013737 177572 001272      MOV    SRO,WASSR0    ;SAVE CONTENTS OF SRO FOR ERROR
2925 026736 013737 177576 001274      MOV    SR2,WASSR2    ;SAVE CONTENTS OF SR2 FOR ERROR
2926 026744 042737 160000 177572      BIC    #160000,SRO   ;CLEAR ERROR BITS IN SRO
2927 026752 104032              ERROR  32           ;GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
2928                                ;FOR TIGHTER SCOPE LOOP
2929                                ;REPLACE ERROR CALL WITH
2930                                ;A 'NOP' = 000240
2931 026754 013746 001270              MOV    TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
2932 026760 013746 001266              MOV    TRAPPC,-(KSP)
2933 026764 000002              RTI                    ;RETURN FROM UNEXPECTED ABORT
2934
2935 026766 012737 026512 001110 10$: MOV    #1$,PERR      ;RESET LOOP ON ERROR POINTER TO 1$
2936 026774 012737 002150 000250      MOV    #MGMERR,MMVEC ;RESTORE NORMAL M.M. TRAP HANDLER
2937                                ;ADDRESS TO M.M. TRAP VECTOR
2938 027002 000137 027072              JMP    TST34
2939
2940                                ;DAL TABLE FOR UPWARD EXPANSION (NON-ABORT CASES)
2941 027006 100000      DALTB1: 100000
2942 027010 106100      106100
2943 027012 102300      102300
2944 027014 102500      102500
2945 027016 113700      113700
2946 027020 104600      104600
2947 027022 117700      117700
2948
2949                                ;PDR TABLE FOR KPDR4 (NON-ABORT CASES)
2950 027024 000006      PDRTB1: 000006
2951 027026 052006      052006
2952 027030 045006      045006
2953 027032 052006      052006
2954 027034 074406      074406
2955 027036 025006      025006
2956 027040 077406      077406
2957
2958                                ;DAL TABLE (ABORT CASES)
2959 027042 100100      DALTB2: 100100
2960 027044 110100      110100

```

2961 027046 116600 116600  
 2962 027050 112700 112700  
 2963 027052 117000 117000  
 2964 027054 117700 117700

:PDR TABLE (ABORT CASES)

2966 PDRTB2: 000006  
 2967 027056 000006 030406  
 2968 027060 030406 030406  
 2969 027062 046406 046406  
 2970 027064 042006 042006  
 2971 027066 073406 073406  
 2972 027070 077006 077006

\*\*\*\*\*  
 :TEST 34 PAGE LENGTH FAULTS-DOWNWARD EXPANSION  
 \*\*\*\*\*

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

2987 027072 000004  
 2988 027074 012700 027354  
 2989 027100 012704 027372  
 2990 027104 012701 000006  
 2991 027110 012737 027266 000250  
 2992 027116 012737 027130 001110  
 2993 027124 012706 001100

\*\*\*\*\*  
 TST34: SCOPE  
 1\$: MOV #DALTB3,R0 ;DAL TABLE FOR VIRTUAL ADDR'S. TO SELECT PDR4.  
 MOV #PDRTB3,R4 ;PDR TABLE FOR PDR4 (COINCIDES WITH DAL TABLE).  
 MOV #6,R1 ;SET UP LOOP COUNTER.  
 MOV #9\$,MMVEC ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS  
 MOV #2\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 2\$  
 MOV #KERSTK,KSP ;MAKE SURE STACK POINTER IS ALL SET UP

2995  
 2996 027130 012437 172310  
 2997 027134 005730  
 2998  
 2999 027136 077104  
 3000

:TEST NON-ABORT CASES (VBA > OR = PLF)  
 2\$: MOV (R4)+,KIPDR4 ;LOAD KIPDR4 WITH PAGE LENGTH VALUE  
 TST @ (R0)+ ;ACCESS VIRTUAL ADDR. (VBA > OR = PLF)  
 ;NO ABORT SHOULD OCCUR!!!  
 SOB R1,2\$ ;DONE?...NO- TEST NEXT COMBINATION OF DAL & PDR.

3001  
 3002 027140 012701 000005  
 3003 027144 012700 027410  
 3004 027150 012704 027424  
 3005 027154 012737 027170 001110  
 3006 027162 012737 027202 000250  
 3007

:TEST ABORT CASES (VBA < PLF)  
 3\$: MOV #5,R1 ;SET UP LOOP COUNTER.  
 MOV #DALTB4,R0 ;DAL TABLE  
 MOV #PDRTB4,R4 ;PDR TABLE  
 MOV #4\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 4\$  
 MOV #6\$,MMVEC ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT

3008 027170 012437 172310  
 3009 027174 005730  
 3010 027176 104033  
 3011  
 3012  
 3013

4\$: MOV (R4)+,KIPDR4 ;LOAD KIPDR4 WITH PAGE LENGTH VALUE  
 5\$: TST @ (R0)+ ;ACCESS VIRTUAL ADDR. (VBA < PLF - ABORT TO 6\$)  
 ERROR 33 ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR  
 ;FOR TIGHTER SCOPE LOOP  
 ;REPLACE ERROR CALL WITH  
 ;'BR 5\$' = 000776

3014 027200 000424  
 3015 027202 012706 001100  
 3016 027206 013737 177572 001272

6\$: BR 8\$ ;BRANCH AROUND ABORT CHECKS  
 MOV #KERSTK,KSP ;RESTORE STACK POINTER FOLLOWING ABORT  
 MOV SR0,WASSR0 ;READ M.M. STATUS REG. 0

```

3017 027214 013737 177576 001274      MOV      SR2,WASSR2      ;READ M.M. STATUS REG. 2
3018 027222 012702 040011              MOV      #40011,R2      ;PUT EXPECTED SRO CONTENTS IN R2
3019 027226 020237 001272              CMP      R2,WASSR0      ;DID SRO REPORT PG. LENGTH ABORT, PAGE 4, KERNEL?
3020 027232 001401              BEQ      7$              ;BRANCH IF YES
3021 027234 104034              ERROR    34              ;SRO DID NOT REPORT PG. LENGTH ABORT CORRECTLY
3022                                ;FOR TIGHTER SCOPE LOOP
3023                                ;REPLACE ERROR CALL WITH
3024                                ;'BR 5$' = 000757
3025 027236 012703 027174      7$:      MOV      #5$,R3          ;PUT EXPECTED SR2 CONTENTS IN R3
3026 027242 020337 001274              CMP      R3,WASSR2      ;DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?
3027 027246 001401              BEQ      8$              ;BRANCH IF YES
3028 027250 104035              ERROR    35              ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY
3029                                ;FOR TIGHTER SCOPE LOOP
3030                                ;REPLACE ERROR CALL WITH
3031                                ;'BR 5$' = 000751
3032 027252 042737 160000 177572      8$:      BIC      #160000,SRO     ;CLEAR ERROR BITS IN SRO
3033 027260 077135              SOB      R1,4$          ;DONE?...NO - GET NEXT DAL & PDR PAIR
3034 027262 000137 027334              JMP      10$            ;YES...
3035 027266 012637 001266              9$:      MOV      (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
3036 027272 012637 001270              MOV      (KSP)+,TRAPPS
3037 027276 013737 177572 001272      MOV      SRO,WASSR0      ;SAVE CONTENTS OF SRO FOR ERROR
3038 027304 013737 177576 001274      MOV      SR2,WASSR2      ;SAVE CONTENTS OF SR2 FOR ERROR
3039 027312 042737 160000 177572      BIC      #160000,SRO     ;CLEAR ERROR BITS IN SRO
3040 027320 104032              ERROR    32              ;GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
3041                                ;FOR TIGHTER SCOPE LOOP
3042                                ;REPLACE ERROR CALL WITH
3043                                ;A 'NOP' = 000240
3044 027322 013746 001270              MOV      TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
3045 027326 013746 001266              MOV      TRAPPC,-(KSP)
3046 027332 000002              RTI                          ;RETURN FROM UNEXPECTED ABORT
3047
3048 027334 012737 027074 001110      10$:     MOV      #1$, $LPERR     ;RESET LOOP ON ERROR POINTER TO 1$
3049 027342 012737 002150 000250      MOV      #MGMERR,MMVEC   ;RESTORE NORMAL M.M. TRAP HANDLER
3050                                ;ADDRESS TO M.M. TRAP VECTOR
3051 027350 000137 027440              JMP      TST35
3052
3053                                ;DAL TABLE FOR DOWNWARD EXPANSION (NON-ABORT CASES)
3054 027354 117700      DALTB3: 117700
3055 027356 111600              111600
3056 027360 115400              115400
3057 027362 115200              115200
3058 027364 104000              104000
3059 027366 113100              113100
3060 027370 100000              100000
3061
3062                                ;PDR TABLE (NON-ABORT CASES)
3063 027372 077416      PDRTB3: 77416
3064 027374 025416              25416
3065 027376 032416              32416
3066 027400 025416              25416
3067 027402 003016              03016
3068 027404 052416              52416
3069 027406 000016              00016
3070
3071                                ;DAL TABLE (ABORT CASES)
3072 027410 117600      DALTB4: 117600
    
```

```

3073 027412 107600 107600
3074 027414 101100 101100
3075 027416 105000 105000
3076 027420 100700 100700
3077 027422 100000 100000
3078
3079

```

```

:PDR TABLE (ABORT CASES)
PDRTB4: 77416
        47016
        31016
        35416
        04016
        00416

```

```

3080 027424 077416
3081 027426 047016
3082 027430 031016
3083 027432 035416
3084 027434 004016
3085 027436 000416
3086
3087
3088
3089

```

\*\*\*\*\*

\*TEST 35 SR2 BIT TEST

```

*
* THIS TEST CHECKS THE BITS IN MEMORY MANAGEMENT REGISTER 2 BY
* CAUSING 'READ-ONLY ABORTS' AT VIRTUAL ADDRESSES BETWEEN 100000
* TO 110000 (PHYSICAL ADDRESSES 060000-070000). KIPDR4 IS USED TO EXECUTE
* THE FOLLOWING FOUR WORDS OF CODE WHICH ARE MOVED THRU MEMORY:
* 010727 MOV PC,(PC)+ ;THIS INSTRUCTION SHOULD CAUSE A R/O ABORT
* 000000 ;ITS VIRTUAL ADDR. SHOULD BE LOCKED UP IN SR2
* 000137 JMP @#3$ ;THIS INSTRUCTION IS ALSO MOVED THRU MEMORY
* (ADDR. OF 3$) ;IN CASE A R/O ABORT DOES NOT OCCUR,
* ;IN WHICH CASE SR2 WILL NOT CONTAIN CORRECT ADDR.

```

\*\*\*\*\*

```

3103 027440 000004
3104 027442 012737 000600 172346
3105 027450 012737 000600 172350
3106 027456 012737 077406 172306
3107 027464 012737 077402 172310
3108 027472 012700 060002
3109 027476 012701 100002
3110 027502 012737 027536 000250
3111 027510 012737 027516 001110
3112 027516 012720 010727
3113 027522 005020
3114 027524 012720 000137
3115 027530 012710 027536
3116 027534 010107
3117 027536 012706 001100
3118 027542 013737 177576 001274
3119 027550 020137 001274
3120 027554 001401
3121 027556 104036
3122
3123
3124
3125 027560 042737 160000 177572
3126 027566 060101
3127 027570 010100
3128 027572 052701 100000

```

```

TST35: SCOPE
1$: MOV #600,KIPAR3 ;BE SURE PAR3 IS MAPPED TO 12-16K
    MOV #600,KIPAR4 ;BE SURE PAR4 IS MAPPED TO 12-16K
    MOV #77406,KIPDR3 ;MAP PAGE 3 128 BLOCKS, R/W
    MOV #77402,KIPDR4 ;MAP PAGE 4 128 BLOCKS, READ-ONLY
    MOV #60002,R0 ;LOAD R0 WITH VIRTUAL ADDR. WHICH USES PDR3
    MOV #100002,R1 ;LOAD R1 WITH VIRTUAL ADDR. WHICH USES PDR4
    MOV #3$,MMVEC ;SET M.M. TRAP VECTOR TO 3$
    MOV #2$,SLPERR ;SET LOOP ON ERROR POINTER TO 2$
2$: MOV #010727,(R0)+ ;LOAD 'MOV PC,(PC)+' INSTRUCTION AT ADDR.
    CLR (R0)+ ; REACHED THRU PDR/PAR 4.
    MOV #000137,(R0)+ ;LOAD 'JMP @#3$' INSTRUCTION AT VIRT. ADDR.
    MOV #3$,(R0) ; IN CASE R/O VIOL. DOES NOT ABORT
3$: MOV R1,PC ;TRANSFER PROGRAM EXECUTION TO 'PAGE 4 INSTRUCTIONS'
    MOV #KERSTK,KSP ;RESTORE STACK POINTER
    MOV SR2,WASSR2 ;READ CONTENTS OF STATUS REG 2
    CMP R1,WASSR2 ;WAS ADDR. OF 'RELOCATED - R/O ABORT' LOCKED UP?
    BEQ 4$ ;BRANCH IF YES
    ERROR 36 ;SR2 DID NOT LOCK UP VIRTUAL ADDR. OF R/O VIOL.
    ;FOR TIGHTER SCOPE LOOP
    ;REPLACE ERROR CALL WITH
    ;'BR 2$' = 000757
4$: BIC #160000,SRO ;CLEAR THE ERROR BITS IN SRO
    ADD R1,R1 ;SETUP TO FORM NEXT VIRTUAL ADDRESS
    MOV R1,R0 ;SETUP R0 TO FORM NEXT VIRT. ADDR. TO LOAD
    BIS #100000,R1 ;FORM VIRTUAL ADDR. THAT SHOULD BE LOCKED UP NEXT

```

```

3129 027576 052700 060000      BIS      #60000,R0      ;POINT R0 TO NEXT VIRT. ADDR. TO LOAD
3130 027602 020127 110000      CMP      R1,#110000   ;HAVE ALL VBA'S 100000-110000 BEEN TESTED?
3131 027606 101743                BLOS    2$           ;BRANCH IF NO
3132
3133 027610 012737 027442 001110 5$:  MOV     #1$,$LPERR   ;RESET LOOP ON ERROR POINTER TO 1$
3134 027616 012737 077406 172310      MOV     #77406,KIPDR4 ;RESTORE PDR4 TO R/W ACCESS
3135 027624 012737 002150 000250      MOV     #MGMERR,MMVEC ;RESTORE ADDRESS OF NORMAL M.M.
3136                                     ;TRAP HANDLER TO M.M. VECTOR
3137
3138
3139

```

\*\*\*\*\*

```

3140 :*TEST 36      MORE CHECKS OF SRO & SR2
3141 :*
3142 :*      THIS TEST PERFORMS SOME ADDITIONAL CHECKS OF THE SRO & SR2 LOGIC.
3143 :*      FIRST IT CHECKS THAT SR2 'TRACKS' ALONG ACTING AS A VIRTUAL ADDRESS
3144 :*      PROGRAM COUNTER.  ALSO SRO & SR2 ARE LOCKED UP BY A PAGE LENGTH
3145 :*      ABORT, THEN WITHOUT CLEARING SRO'S ERROR BITS, A R/O ABORT IS CAUSED.
3146 :*      SRO & SR2 SHOULD NOT BE CHANGED BY THE SECOND ABORT AND THE
3147 :*      INFORMATION ABOUT THE PAGE LENGTH ABORT SOULD STILL BE LOCKED UP.
3148 :*      IN ADDITION A 'RESET' IS EXECUTED TO VERIFY THAT SRO IS CLEARED
3149 :*      AND SR2 IS UNLOCKED BY A RESET.  AFTER MEMORY MANAGEMENT IS TURNED BACK ON,
3150 :*      SR2 IS CHECKED TO SEE THAT IT IS TRACKING AGAIN.
3151 :*
3152 :*

```

\*\*\*\*\*

```

3153 027632 000004      TST36: SCOPE
3154 027634 012737 000600 172352 1$:  MOV     #600,KIPAR5   ;MAP KERNEL PAGE 5 TO 12-16K
3155 027642 012737 000406 172310      MOV     #406,KIPDR4  ;SETUP PDR4 FOR PAGE LENGTH ABORT
3156 027650 012737 077402 172312      MOV     #77402,KIPDR5 ;SETUP PDR5 FOR R/O ABORT
3157 027656 012737 027664 001110      MOV     #2$,$LPERR   ;SET LOOP ON ERROR POINTER TO 2$
3158 027664 013737 177576 001274 2$:  MOV     SR2,WASSR2   ;READ SR2 TO SEE IF ITS TRACKING
3159 027672 012701 027664      MOV     #2$,R1       ;PUT EXPECTED VIRTUAL PC IN R1
3160 027676 020137 001274      CMP     R1,WASSR2   ;DID SR2 CONTAIN VIRTUAL PC AT 2$?
3161 027702 001401      BEQ    3$           ;BRANCH IF YES
3162 027704 104041      ERROR   41         ;SR2 NOT TRACKING CORRECTLY
3163                                     ;FOR TIGHTER SCOPE LOOP
3164                                     ;REPLACE ERROR CALL WITH
3165                                     ;'BR 2$' = 000767
3166 027706 012737 027714 001110 3$:  MOV     #4$,$LPERR   ;SET LOOP ON ERROR POINTER TO 4$
3167 027714 013737 177576 001274 4$:  MOV     SR2,WASSR2   ;READ SR2 TO SEE IF ITS TRACKING
3168 027722 012701 027714      MOV     #4$,R1       ;PUT EXPECTED VIRTUAL PC IN R1
3169 027726 020137 001274      CMP     R1,WASSR2   ;DID SR2 CONTAIN VIRTUAL PC AT 4$
3170 027732 001401      BEQ    5$           ;BRANCH IF YES
3171 027734 104041      ERROR   41         ;SR2 NOT TRACKING CORRECTLY
3172                                     ;FOR TIGHTER SCOPE LOOP
3173                                     ;REPLACE ERROR CALL WITH
3174                                     ;'BR 4$' = 000767
3175 027736 012737 027744 001110 5$:  MOV     #6$,$LPERR   ;SET LOOP ON ERROR POINTER TO 6$
3176 027744 012737 027762 000250 6$:  MOV     #7$,MMVEC    ;PUT ADDRESS OF 7$ IN M.M. TRAP VECTOR
3177 027752 005037 001200      CLR    $TMP1        ;CLEAR ERROR INDICATOR
3178 027756 005237 100500      INC    @#100500     ;CAUSE PAGE LENGTH ABORT - TRAP TO 7$
3179 027762 012706 001100      MOV    #KERSTK,KSP  ;RESTORE STACK POINTER AFTER ABORT
3180 027766 013737 177572 001176      MOV    SRO,$TMP0    ;SAVE SRO'S INFORMATION ON PG. LGTH. ABORT
3181 027774 013737 177576 001202      MOV    SR2,$TMP2    ;SAVE SR2'S INFORMATION ON PG. LGTH. ABORT
3182 030002 012737 030014 000250      MOV    #8$,MMVEC    ;PUT ADDRESS OF 8$ IN M.M. TRAP VECTOR
3183 030010 005237 120000      INC    @#120000     ;CAUSE R/O ABORT - TRAP TO 8$
3184 030014 012706 001100      MOV    #KERSTK,KSP  ;RESTORE STACK POINTER AFTER ABORT

```

```

3185 030020 013737 177572 001272      MOV      SRO,WASSRO      ;READ SRO FOLLOWOING SECOND KT ABORT
3186 030026 013737 177576 001274      MOV      SR2,WASSR2     ;READ SR2 FOLLOWING SECOND KT ABORT
3187 030034 023737 001176 001272      CMP      $TMP0,WASSRO   ;IS SRO STILL HOLDING INFO ON FIRST ABORT?
3188 030042 001402                BLQ      9$             ;BRANCH IF YES
3189 030044 005237 001200                INC      $TMP1          ;SET ERROR INDICATOR
3190 030050 023737 001202 001274  9$:    CMP      $TMP2,WASSR2   ;DOES SR2 STILL HOLD PC OF FIRST ABORT?
3191 030056 001402                BEQ      10$           ;BRANCH IF YES
3192 030060 005237 001200                INC      $TMP1          ;SET ERROR INDICATOR
3193 030064 005737 001200                TST      $TMP1          ;WERE SRO OR SR2 CHANGED BY A SECOND ABORT?
3194 030070 001401                BEQ      11$           ;BRANCH IF NO
3195 030072 104037                ERROR    37            ;ONE OF STATUS REGS. C. ANGED BY SECOND ABORT
3196                                ;FOR TIGHTER SCOPE LOOP
3197                                ;REPLACE ERROR CALL WITH
3198                                ;'BR 6$' = 000726
3199 030074 005037 001200                11$:    CLR      $TMP1          ;CLEAR ERROR INDICATOR
3200 030100 000005                RESET                    ;EXECUTE A RESET, APPLYING AN 'INIT'
3201 030102 013737 177572 001272      MOV      SRO,WASSRO   ;READ SRO
3202 030110 005737 001272                TST      WASSRO         ;WAS SRO CLEARED BY THE RESET?
3203 030114 001402                BEQ      12$           ;BRANCH IF YES
3204 030116 005237 001200                INC      $TMP1          ;SRO NOT CLEARED BY A RESET
3205 030122 013737 177576 001274  12$:    MOV      SR2,WASSR2     ;READ SR2
3206 030130 022737 030122 001274      CMP      #12$,WASSR2   ;WAS SR2 UNLOCKED BY A RESET?
3207 030136 001402                BEQ      13$           ;BRANCH IF YES
3208 030140 005237 001200                INC      $TMP1          ;SR2 NOT UNLOCKED BY A RESET
3209 030144 005737 001200                13$:    TST      $TMP1          ;WERE SRO & SR2 BOTH 'RESET' BY A RESET?
3210 030150 001401                BEQ      14$           ;BRANCH IF YES
3211 030152 104040                ERROR    40            ;SRO OR SR2 NOT 'RESET' BY A RESET
3212                                ;FOR TIGHTER SCOPE LOOP
3213                                ;REPLACE ERROR CALL WITH
3214                                ;'BR 6$' = 000676
3215 030154 012737 000001 177572  14$:    MOV      #1,SRO        ;TURN MEMORY MANAGEMENT BACK ON
3216 030162 013737 177576 001274  15$:    MOV      SR2,WASSR2     ;READ SR2 TO SEE IF ITS TRACKING AGAIN
3217 030170 012701 030162                MOV      #15$,R1       ;PUT EXPECTED VIRTUAL PC IN R1
3218 030174 020137 001274                CMP      R1,WASSR2     ;DID SR2 CONTAIN VIRTUAL PC AT 15$
3219 030200 001401                BEQ      16$           ;BRANCH IF YES
3220 030202 104041                ERROR    41            ;SR2 NOT TRACKING CORRECTLY
3221                                ;FOR TIGHTER SCOPE LOOP
3222                                ;REPLACE ERROR CALL WITH
3223                                ;'BR 6$' = 000663
3224 030204 012737 027634 001110  16$:    MOV      #1$, $LPERR    ;RESET LOOP ON ERROR POINTER TO 1$
3225 030212 012737 077406 172310      MOV      #77406,KIPDR4 ;RESET PDR4 TO 128 BLKS, R/W
3226 030220 012737 077406 172312      MOV      #77406,KIPDR5 ;RESET PDR5 TO 128 BLKS, R/W
3227 030226 012737 002150 000250      MOV      #MGMERR,MMVEC ;RESTORE ADDRESS OF NORMAL MEMORY
3228                                ;MANAGEMENT TRAP ROUTINE TO M.M. VECTOR
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240

```

```

:*****
:TEST 37      USER ABORT PICKS UP KERNEL SPACE VECTOR
:
:
:   THIS TEST CHECKS TO BE SURE THAT WHEN AN ABORT OCCURS WHILE IN
:   USER MODE, THE TRAP VECTOR INFORMATION FETCHED IS TAKEN FROM
:   KERNEL SPACE.  USER PAGE 0 IS MAPPED TO 12K (60000-77776) SO
:   THAT IF USER SPACE IS USED INSTEAD OF KERNEL, THE NEW PC THAT
:   WAS LOADED AT LOC. 060004 IS USED INSTEAD OF THE NEW PC THAT
:   SHOULD BE PICKED UP FROM LOC. 000004.  A TIMEOUT ERROR IS USED
:   TO CAUSE A TRAP TO '4'.
:

```

```

3241
3242
3243 030234 000004
3244 030236 004737 035114
3245 030242 012737 030250 001110
3246 030250 005037 177776
3247 030254 012706 001100
3248 030260 012737 000600 177640
3249 030266 012737 030350 000004
3250 030274 012737 000340 000006
3251 030302 012737 140000 177776
3252 030310 012706 000700
3253 030314 012737 030334 000004
3254 030322 012737 000340 000006
3255 030330 005737 160000
3256
3257
3258 030334 013701 177776
3259 030340 010602
3260 030342 005037 177776
3261 030346 104042
3262

: *
: *****
TST37: SCOPE
1$: JSR PC,TOFF ;TURN OFF T-BIT TRAPPING FOR THIS TEST
MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
2$: CLR PSW ;GO TO KERNEL MODE
MOV #KERSTK,KSP ;SETUP KERNEL STACK PTR.
MOV #600,UIPAR0 ;MAP USER PAGE 0 TO 12K
MOV #4$, @#4 ;LOAD KERNEL VECTOR 4 (LOC.4) WITH 4$
MOV #340, @#6 ;LOAD VECTOR+2 WITH NEW PSW
MOV #140000,PSW ;GO TO USER MODE
MOV #USESTK,USP ;SETUP USER STACK PTR.
MOV #3$, @#4 ;LOAD USER VECTOR 4 (LOC. 60004) WITH 3$
MOV #340, @#6 ;LOAD VECTOR+2 WITH NEW PSW
TST 160000 ;CAUSE TIMEOUT ERROR TRAP TO '4'
;SHOULD PICK UP NEW PC=4$ FROM KERNEL
;LOC. 4, NOT PC=3$ FROM USER LOC. 4 (=60004)
3$: MOV PSW,R1 ;SAVE PSW FOR ERROR
MOV SP,R2 ;SAVE VALUE OF STACK POINTER FOR ERROR
CLR PSW ;BE SURE BACK IN KERNEL MODE
ERROR 42 ;DID NOT TRAP THRU KERNEL SPACE
;FOR TIGHTER SCOPE LOOP
    
```

```

3263                                     ;REPLACE ERROR CALL WITH
3264                                     ;'BR 2$' = 000740
3265 030350 005037 177776                4$: CLR     PSW                ;BE SURE BACK IN KERNEL MODE
3266 030354 012706 001100                MOV     #KERSTK,KSP        ;RESTORE KERNEL S.P. IN CASE IT CHANGED
3267 030360 005037 177640                CLR     UIPARO             ;REMAP USER PAGE 0 TO 0-4K
3268 030364 012737 140000 177776        MOV     #140000,PSW        ;GO TO USER MODE
3269 030372 012706 000700                MOV     #USESTK,USP        ;RESTORE USER STACK POINTER
3270 030376 005037 177776                CLR     PSW                ;GO BACK TO KERNEL MODE
3271 030402 012737 002076 000004        MOV     #TIMERR,@#4        ;RESTORE ADDR. OF NORMAL CPU TRAP HANDLER TO 4
3272 030410 012737 030236 001110        MOV     #1$,$LPERR        ;RESET LOOP ON ERROR POINTER TO 1$
3273 030416 004737 035150                JSR     PC,TON             ;TURN T-BIT TRAPPING BACK ON
3274
3275                                     ;*****
3276                                     ;*TEST 40          RTI IN USER MODE DOES NOT CHANGE PSW
3277                                     ;*
3278                                     ;*          THIS TEST CHECKS TO SEE THAT WHEN AN RTI IS EXECUTED IN USER
3279                                     ;*          MODE, THE MODE OR PRIORITY BITS OF THE PSW ARE NOT CHANGED.
3280                                     ;*
3281                                     ;*****
3282 030422 000004                TST40: SCOPE
3283
3284 030424 012737 030436 001110 1$: MOV     #2$,$LPERR        ;SET LOOP ON ERROR POINTER TO 2$
3285 030432 012702 170000                MOV     #170000,R2        ;LOAD 'PRESENT & EXPECTED' PSW VALUE INTO R2
3286 030436 010237 177776                2$: MOV     R2,PSW        ;GO TO USER MODE-PRIORITY 0
3287 030442 012746 000340                MOV     #340,-(SP)        ;PUT A NEW PSW (PRIORITY=7) ON STACK
3288 030446 012746 030454                MOV     #3$,-(SP)        ;PUT NEW PC ON THE STACK
3289 030452 000002                RTI                    ;DO AN RTI FROM USER MODE
3290 030454 013701 177776                3$: MOV     PSW,R1        ;READ NEW PSW INTO R1
3291 030460 042701 007437                BIC     #7437,R1         ;MASK OFF COND. CODE, T-BIT, AND UNUSED BITS
3292 030464 005037 177776                CLR     PSW                ;GO BACK TO KERNEL MODE
3293 030470 020201                CMP     R2,R1             ;DID PSW STAY IN USER, PRIORITY=0?
3294 030472 001401                BEQ     4$                ;BRANCH IF YES
3295 030474 104060                ERROR  60                ;PSW CHANGED BY AN RTI FROM USER
3296                                     ;FOR A TIGHTER SCOPE LOOP
3297                                     ;REPLACE ERROR CALL WITH
3298                                     ;'BR=2$' = 000760
3299 030476 012737 030424 001110 4$: MOV     #1$,$LPERR        ;RESET LOOP ON ERROR POINTER TO 1$
3300
3301
    
```



```

3302 :*****
3303 :*TEST 41      KT ERROR SERVICED BEFORE TIMEOUT ERROR
3304 :*
3305 :*      THIS TEST CHECKS TO SEE THAT IF A CERTAIN VIRTUAL ADDRESS THAT
3306 :*      WOULD CAUSE A MEMORY MANAGEMENT ERROR CAUSES A TIMEOUT
3307 :*      ERROR FIRST, THE TIMEOUT ERROR IS SERVICED BUT THE MEMORY
3308 :*      MANAGEMENT ERROR ISN'T.  THIS MEANS THAT SRO AND SR2
3309 :*      SHOULD NOT REPORT THE ERROR OR LOCK UP ITS VIRTUAL ADDRESS.
3310 :*      A READ-ONLY VIOLATION IS USED AS THE POTENTIAL MEMORY MANAGEMENT
3311 :*      ERROR
3312 :*
3313 :*****
3314 030504 000004 TST41: SCOPE
3315 030506 012705 077006 1$:  MOV      #77006,R5      ;LOAD PDR7 DATA INTO R5
3316 030512 010537 172316      MOV      R5,KIPDR7     ;MAP PAGE 7 R/W PLF=176
3317 030516 012737 030544 000004      MOV      #3$,@#4      ;SET CPU TRAP VECTOR TO ADDRESS OF 3$
3318 030524 012737 030546 000250      MOV      #4$,@#250    ;SET M.M. TRAP VECTOR TO ADDRESS OF 4$
3319 030532 012737 030540 001110      MOV      #2$, $LPERR  ;SET LOOP ON ERROR POINTER TO 2$
3320 030540 005237 177700      2$:  INC      @#177700   ;CAUSE PLF ABORT AND POTENTIAL TIMEOUT
3321 030544 104043      3$:  ERROR   43         ;TRAPPED THRU CPU TRAP VECTOR BUT SHOULDN'T HAVE
3322 :*
3323 :*      FOR TIGHTER SCOPE LOOP
3324 :*      REPLACE ERROR CALL WITH
3325 :*      'BR 2$' = 000776
3326 030546 012706 001100      4$:  MOV      #KERSTK,KSP ;RESTORE STACK POINTER AFTER TRAPPING
3327 030552 013737 177572 001272      MOV      SRO,WASSRO   ;READ STATUS REG.0
3328 030560 013737 177576 001274      5$:  MOV      SR2,WASSR2  ;READ STATUS REG. 2
3329 030566 012700 040017      MOV      #40017,R0    ;LOAD EXPECTED SRO CONTENTS INTO R0
3330 030572 020037 001272      CMP      R0,WASSR0    ;SRO PLF ERROR BIT SET?
3331 030576 001401      BEQ      6$          ;BRANCH IF YES
3332 030600 104044      ERROR   44         ;SRO DIDN'T REPORT PLF ERROR
3333 :*
3334 :*      FOR TIGHTER SCOPE LOOP
3335 :*      REPLACE ERROR CALL WITH
3336 :*      'BR 2$' = 000741
3337 030602 012701 030540      6$:  MOV      #2$,R1      ;LOAD EXPECTED SR2 CONTENTS INTO R1
3338 030606 020137 001274      CMP      R1,WASSR2   ;WAS SR2 LOCKED BY PLF ABORT?
3339 030612 001401      BEQ      7$          ;BRANCH IF YES
3340 030614 104044      ERROR   44         ;SR2 DIDN'T LOCK UP VIRTUAL ADDRESS
3341 :*
3342 :*      FOR TIGHTER SCOPE LOOP
3343 :*      REPLACE ERROR CALL WITH
3344 :*      'BR 2$' = 000741
3345 030616 042737 160000 177572      7$:  BIC      #160000,SRO ;CLEAR ERROR BITS THAT WERE SET IN SRO
3346 030624 012737 002076 000004      MOV      #TIMERR,@#4  ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
3347 030632 012737 002150 000250      MOV      #IMGERR,@#250 ;RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER
3348 030640 012737 077406 172316      MOV      #77406,KIPDR7 ;REMAP PAGE 7 TO READ/WRITE PLF=177
3349 030646 012737 030506 001110      MOV      #1$, $LPERR  ;RESET LOOP ON ERROR POINTER TO 1$
3350 :*****
3351 :*TEST 42      PC & PSW SAVED FOR KT ERROR DURING SERVICE OF TIMEOUT ERROR
3352 :*
3353 :*      THIS TEST CHECKS THE PC AND PROCESSOR STATUS WORD SAVED WHEN
3354 :*      A KT ERROR OCCURS DURING THE SECOND PUSH ON THE STACK DURING
3355 :*      SERVICING OF A TIMEOUT ERROR.  DURING A 'DOUBLE ERROR'
3356 :*      SEQUENCE SUCH AS THIS, THE PSW SAVED WILL BE THE ONE PICKED UP
3357 :*      FROM VECTOR+2 (LOC. 6 IN THIS CASE) AFTER THE FIRST TRAP,
3358 :*      NOT THE PSW PRESENT BEFORE THE FIRST TRAP.  SRO AND SR2
    
```

```

3358      :*      SHOULD RECORD THE KT ERROR (A R/O VIOLATION BY THE USER STACK PTR.)
3359      :*
3360      :*      NOTE THAT THE PREVIOUS MODE BITS <13:12> OF THE PSW
3361      :*      WILL BE SET IN THE PSW THAT IS SAVED.
3362      :*
3363      :*****
3364 030654 000004 TST42: SCOPE
3365 030656 004737 035114 1$: JSR PC,TOFF ;TURN T-BIT TRAPPING OFF FOR THIS TEST
3366 030662 012737 000600 177646 MOV #600,UIPAR3 ;MAP USER PAGE 3 TO 12-16K
3367 030670 012737 000600 177650 MOV #600,UIPAR4 ;MAP USER PAGE 4 TO 12-16K
3368 030676 012737 077402 177606 MOV #77402,UIPDR3 ;MAP USER PAGE 3 READ-ONLY
3369 030704 012737 077406 177610 MOV #77406,UIPDR4 ;MAP USER PAGE 4 READ/WRITE
3370 030712 012737 030766 000004 MOV #4$,@#4 ;LOAD ADDRESS OF 4$ IN CPU (TIMEOUT) VECTOR
3371 030720 012737 140017 000006 MOV #140017,@#6 ;LOAD PSW THAT SHOULD BE PUT ON STACK IN VECTOR+2
3372 030726 012737 030766 000250 MOV #4$,@#250 ;LOAD ADDRESS OF 4$ IN M.M. TRAP VECTOR
3373 030734 012737 000340 000252 MOV #340,@#252 ;LOAD A KERNEL PSW IN MMVEC+2
3374 030742 012737 030750 001110 MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
3375 030750 012737 140000 177776 2$: MOV #140000,PSW ;GO TO USER MODE
3376 030756 012706 100002 MOV #100002,USP ;SET USER STACK PTR. SO SECOND PUSH IS IN PG. 3
3377 030762 005737 177700 3$: TST @#177700 ;CAUSE TIMEOUT ERROR THAT WILL CAUSE
3378 ;R/O ERROR WHEN TRY TO SAVE OLD PC
3379 030766 016601 000002 4$: MOV 2(KSP),R1 ;PUT PSW SAVED ON KERNEL STACK INTO R1
3380 030772 011603 MOV (KSP),R3 ;PUT PC SAVED ON KERNEL STACK INTO R3
3381 030774 013737 177572 001272 MOV SR0,WASSRO ;READ THE CONTENTS OF M.M. STATUS REG. 0
3382 031002 013737 177576 001274 MOV SR2,WASSR2 ;READ THE CONTENTS OF M.M. STATUS REG. 2
3383 031010 042737 160000 177572 BIC #160000,SR0 ;CLEAR THE ERROR BITS IN SR0
3384 031016 005037 177776 CLR PSW ;BE SURE IN KERNEL MODE
3385 031022 012706 001100 MOV #KERSTK,KSP ;RESTORE KERNEL STACK POINTER
3386 031026 012737 140000 177776 MOV #140000,PSW ;GO TO USER MODE
3387 031034 012706 000700 MOV #USESTK,USP ;RESTORE USER STACK POINTER
3388 031040 005037 177776 CLR PSW ;GO BACK TO KERNEL MODE
3389 031044 005037 001176 CLR $TMP0 ;CLEAR ERROR INDICATOR
3390 031050 020127 170017 CMP R1,#170017 ;WAS THE PSW SAVED THE ONE PICKED UP BY THE
3391 ;TIMEOUT TRAP FROM ERRVEC+2?
3392 ;VALUE 170017 = PSW FROM LOC. 6 WITH
3393 ;PREVIOUS MODE BITS = USER
3394 031054 001402 BEQ 5$ ;BRANCH IF YES
3395 031056 005237 001176 INC $TMP0 ;WRONG PSW SAVED DURING 'DOUBLE ERROR' SEQUENCE
3396 031062 020327 030766 5$: CMP R3,#3$+4 ;WAS THE PC AT THE TIME OF THE TIMEOUT ERROR
3397 ;SAVED ON THE STACK?
3398 031066 001402 BEQ 6$ ;BRANCH IF YES
3399 031070 005237 001176 INC $TMP0 ;WRONG PC SAVED DURING TRAP SEQUENCE
3400 031074 023727 001272 020147 6$: CMP WASSRO,#20147 ;DID SR0 REPORT - USER, PAGE 3, R/O ABORT?
3401 031102 001402 BEQ 7$ ;BRANCH IF YES
3402 031104 005237 001176 INC $TMP0 ;SR0 DID NOT REPORT R/O ABORT
3403 031110 023727 001274 030762 7$: CMP WASSR2,#3$ ;DID SR2 LOCK UP VIRTUAL ADDR. OF LAST
3404 ;INSTRUCTION SUCCESSFULLY FETCHED?
3405 031116 001402 BEQ 8$ ;BRANCH IF YES
3406 031120 005237 001176 INC $TMP0 ;SR2 DID NOT LOCK UP ADDR. OF TIMEOUT INST.
3407 031124 005737 001176 8$: TST $TMP0 ;ANY 'ERRORS' DURING TRAP SEQUENCE?
3408 031130 001401 BEQ 9$ ;BRANCH IF NO
3409 031132 104045 ERROR 45 ;THE WRONG PC OR PSW WERE SAVED
3410 ;OR SR0 OR SR2 DID NOT REPORT R/O
3411 ;ERROR DURING TIMEOUT - KT TRAP
3412 ;SEQUENCE
3413 ;FOR TIGHTER SCOPE LOOP
    
```

```

3414                                     ;REPLACE ERROR CALL WITH
3415                                     ;'BR 2$' = 000710
3416 031134 012737 002076 000004 9$: MOV #TIMERR,@#4 ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
3417 031142 012737 000340 000006 MOV #340,@#6 ;RELOAD ERRVEC+2 WITH KERNEL PSW
3418 031150 012737 002150 000250 MOV #MGMERR,@#250 ;RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER
3419 031156 012737 077406 177606 MOV #77406,UIPDR3 ;REMAP USER PAGE 3 READ/WRITE
3420 031164 012737 030656 001110 MOV #1$,$LPERR ;RESET LOOP ON ERROR POINTER TO 1$
3421 031172 004737 035150 JSR PC,TON ;TURN T-BIT TRAPPING BACK ON

```

\*\*\*\*\*

\* THIS GROUP OF TESTS WILL TEST ALL THE LOGIC ASSOCIATED WITH  
\* THE 'MOVE FROM PREVIOUS' AND MOVE TO PREVIOUS' INSTRUCTIONS.  
\*

\*\*\*\*\*

3422  
3423  
3424  
3425  
3426  
3427  
3428  
3429  
3430  
3431  
3432  
3433  
3434  
3435  
3436  
3437  
3438  
3439  
3440  
3441

\*\*\*\*\*  
\*TEST 43 MOVE FROM PREVIOUS (USER) I-SPACE  
\*

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

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

\*\*\*\*\*

```

3442 031176 000004 TST43: SCOPE
3443 031200 005037 172340 1$: CLR KIPAR0 ;MAP KERNEL PAGE 0 TO 0-4K
3444 031204 012737 000200 172342 MOV #200,KIPAR1 ;MAP KERNEL PAGE 1 TO 4-8K
3445 031212 012737 000400 172344 MOV #400,KIPAR2 ;MAP KERNEL PAGE 2 TO 8-12K
3446 031220 012737 000600 172346 MOV #600,KIPAR3 ;MAP KERNEL PAGE 3 TO 12-16K
3447 031226 012737 000600 172350 MOV #600,KIPAR4 ;MAP KERNEL PAGE 4 TO 12-16K
3448 031234 012737 007600 172356 MOV #7600,KIPAR7 ;MAP KERNEL PAGE 7 TO THE I/O PAGE
3449 031242 012700 077406 MOV #77406,R0 ;MAKE ALL KERNEL I-SPACE PAGES RESIDENT
3450                                     ;READ/WRITE, LENGTH 200 BLOCKS
3451 031246 012702 000010 MOV #10,R2 ;SET LOOP COUNTER TO 8
3452 031252 012701 172300 MOV #KIPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
3453 031256 010021 2$: MOV R0,(R1)+ ;LOAD PDR WITH 77406
3454 031260 077202 SOB R2,2$ ;LOOP TO 2$ UNTIL ALL PDRS LOADED
3455 031262 012702 000010 MOV #10,R2 ;SET LOOP COUNTER TO 8
3456 031266 012701 177600 MOV #UIPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
3457 031272 010021 3$: MOV R0,(R1)+ ;LOAD PDR WITH 77406
3458 031274 077202 SOB R2,3$ ;LOOP TO 3$ UNTIL ALL PDRS LOADED
3459 031276 012737 000000 177640 MOV #000,UIPAR0 ;MAP USER I PAGE 0 TO 0-4K
3460 031304 012737 000200 177642 MOV #200,UIPAR1 ;MAP USER I PAGE 1 TO 4-8K
3461 031312 012737 000400 177644 MOV #400,UIPAR2 ;MAP USER I PAGE 2 TO 8-12K
3462 031320 012737 000600 177646 MOV #600,UIPAR3 ;MAP USER I PAGE 3 TO 12-16K
3463 031326 012737 007600 177656 MOV #7600,UIPAR7 ;MAP USER I PAGE 7 TO THE I/O PAGE
3464 031334 012737 031342 001110 MOV #4$,$LPERR ;SET LOOP ON ERROR TO 4$
3465 031342 4$:
3466 031342 012737 077406 172310 MOV #77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE
3467 031350 012737 000600 172350 MOV #600,KIPAR4 ;MAP KERNEL I PAGE 4 TO 12K
3468 031356 012737 000600 177650 MOV #600,UIPAR4 ;MAP USER I PAGE 4 TO 12K
3469 031364 012700 036514 MOV #36514,R0 ;LOAD DATA PATTERN INTO R0

```

```

3470 031370 010037 100000      MOV      R0,@#100000      ;LOAD DATA PATTERN INTO PHY 60000
3471 031374 012737 031776 000250  MOV      #23$,MMVEC      ;SET M.M. VECTOR TO 23$
3472 031402 105037 172310      CLR      KIPDR4          ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
3473                                     ;THE FOLLOWING WILL TEST DSTM=C MFPI
3474                                     ;
3475 031406 012737 031414 001110  MOV      #5$,SLPERR      ;SET LOOP ON ERROR POINTER TO 5$
3476 031414 012737 030340 177776 5$:  MOV      #030340,PSW     ;MAKE PREVIOUS MODE USER
3477 031422 006506 6$:  MFFI     USP            ;PUT USER STACK POINTER ON KERNEL
3478                                     ;STACK
3479 031424 022706 001100      CMP      #KERSTK,KSP     ;WAS SOMETHING PUSHED ON STACK AT 6$
3480 031430 001407 7$:  BEQ      7$            ;BRANCH IF NOTHING WAS PUSHED
3481 031432 012600  MOV      (KSP)+,R0       ;POP KERNEL STACK INTO R0
3482 031434 012701 000700  MOV      #USESTK,R1      ;EXPECTING TO GET 700 AS USP
3483 031440 020001  CMP      R0,R1           ;DID YOU GET THE RIGHT POINTER?
3484 031442 001403 8$:  BEQ      8$            ;BRANCH IF YOU DID
3485 031444 104046  ERROR    46        ;WRONG THING WAS PUSHED ON STACK
3486                                     ;FOR TIGHTER SCOPE LOOP
3487                                     ;REPLACE ERROR CALL WITH
3488                                     ;'BR 5$' = 000763
3489 031446 000401 7$:  BR       8$            ;BRANCH TO NEXT TRY
3490 031450 104050  ERROR    50        ;NOTHING PUSHED ON STACK
3491                                     ;FOR TIGHTER SCOPE LOOP
3492                                     ;REPLACE ERROR CALL WITH
3493                                     ;'BR 5$' = 000761
3494 031452 8$:  ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
3495 031452 012737 031464 001110  MOV      #9$,SLPERR      ;SET LOOP ON ERROR POINTER TO 9$
3496 031460 012700 036514  MOV      #36514,R0       ;RELOAD DATA PATTERN IN R0
3497 031464 012737 030340 177776 9$:  MOV      #030340,PSW     ;MAKE PREVIOUS MODE USER
3498 031472 012702 100000  MOV      #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3499 031476 006512  MFPI     (R2)           ;READ FROM PHYSICAL 60000
3500 031500 012601  MOV      (KSP)+,R1       ;POP KERNEL STACK INTO R1
3501 031502 020001  CMP      R0,R1           ;WAS DATA FETCHED SAME AS STORED
3502 031504 001401 10$: BEQ      10$          ;BRANCH IF CORRECT DATA WAS FETCHED
3503 031506 104046  ERROR    46        ;WRONG DATA WAS FETCHED
3504                                     ;FOR TIGHTER SCOPE LOOP
3505                                     ;REPLACE ERROR CALL WITH
3506                                     ;'BR 9$' = 000766
3507 031510 10$: ;THE FOLLOWING WILL TEST DSTM=2 MFPI.
3508 031510 012737 031516 001110  MOV      #11$,SLPERR     ;SET LOOP ON ERROR POINTER TO 11$
3509 031516 012737 030340 177776 11$: MOV      #030340,PSW     ;MAKE PREVIOUS MODE USER
3510 031524 012702 100000  MOV      #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3511 031530 006522  MFPI     (R2)+          ;READ FROM PHYSICAL 60000
3512 031532 012601  MOV      (KSP)+,R1       ;POP KERNEL STACK INTO R1
3513 031534 020001  CMP      R0,R1           ;WAS DATA FETCHED SAME AS STORED
3514 031536 001401 12$: BEQ      12$          ;BRANCH IF CORRECT DATA WAS FETCHED
3515 031540 104046  ERROR    46        ;WRONG DATA WAS FETCHED
3516                                     ;FOR TIGHTER SCOPE LOOP
3517                                     ;REPLACE ERROR CALL WITH
3518                                     ;'BR 11$' = 000766
3519 031542 12$: ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
3520 031542 012737 031550 001110  MOV      #13$,SLPERR     ;SET LOOP ON ERROR POINTER TO 13$
3521 031550 012737 030340 177776 13$: MOV      #030340,PSW     ;MAKE PREVIOUS MODE USER
3522 031556 006537 100000  MFPI     @#100000       ;READ FROM PHYSICAL 60000
3523 031562 012601  MOV      (KSP)+,R1       ;POP KERNEL STACK INTO R1
3524 031564 020001  CMP      R0,R1           ;WAS DATA FETCHED SAME AS STORED
3525 031566 001401 14$: BEQ      14$          ;BRANCH IF CORRECT DATA WAS FETCHED

```

```

3526 031570 104046          ERROR 46          ;WRONG DATA WAS FETCHED
3527                                     ;FOR TIGHTER SCOPE LOOP
3528                                     ;REPLACE ERROR CALL WITH
3529                                     ;'BR 13$' = 000767
3530 031572                                     ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
3531 031572 012737 031600 001110 14$:      MOV #15$, $LPERR      ;SET LOOP ON ERROR POINTER TO 15$
3532 031600 012737 030340 177776 15$:      MOV #030340,PSW      ;MAKE PREVIOUS MODE USER
3533 031606 012702 100002          MOV #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3534 031612 006542          MFPI -(R2)          ;READ FROM PHYSICAL 60000
3535 031614 012601          MOV (KSP)+,R1      ;POP KERNEL STACK INTO R1
3536 031616 020001          CMP R0,R1          ;WAS DATA FETCHED SAME AS STORED
3537 031620 001401          BEQ 16$           ;BRANCH IF CORRECT DATA WAS FETCHED
3538 031622 104046          ERROR 46          ;WRONG DATA WAS FETCHED
3539                                     ;FOR TIGHTER SCOPE LOOP
3540                                     ;REPLACE ERROR CALL WITH
3541                                     ;'BR 15$' = 000766
3542 031624                                     ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
3543                                     ;
3544                                     ;
3545 031624 012737 031632 001110          MOV #17$, $LPERR      ;SET LOOP ON ERROR POINTER TO 17$
3546 031632 012737 030340 177776 17$:      MOV #030340,PSW      ;MAKE PREVIOUS MODE USER
3547 031640 012737 100000 001202          MOV #100000,$TMP2    ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
3548 031646 012702 001204          MOV #<$TMP2+2>,R2    ;LOAD ADDR. OF $TMP2+2 INTO R2
3549 031652 006552          MFPI @-(R2)         ;READ FROM PHYSICAL 60000
3550 031654 012601          MOV (KSP)+,R1      ;POP KERNEL STACK INTO R1
3551 031656 020001          CMP R0,R1          ;WAS DATA FETCHED SAME AS STORED
3552 031660 001401          BEQ 18$           ;BRANCH IF CORRECT DATA WAS FETCHED
3553 031662 104046          ERROR 46          ;WRONG DATA WAS FETCHED
3554                                     ;FOR TIGHTER SCOPE LOOP
3555                                     ;REPLACE ERROR CALL WITH
3556                                     ;'BR 17$' = 000763
3557 031664                                     ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
3558                                     ;
3559 031664 012737 031672 001110          MOV #19$, $LPERR      ;SET LOOP ON ERROR POINTER TO 19$
3560 031672 012737 030340 177776 19$:      MOV #030340,PSW      ;MAKE PREVIOUS MODE USER
3561 031700 005002          CLR R2             ;MAKE REGISTER 2 A ZERO
3562 031702 006562 100000          MFPI 100000(R2)     ;READ FROM PHYSICAL 60000
3563 031706 012601          MOV (KSP)+,R1      ;POP KERNEL STACK INTO R1
3564 031710 020001          CMP R0,R1          ;WAS DATA FETCHED SAME AS STORED
3565 031712 001401          BEQ 20$           ;BRANCH IF CORRECT DATA WAS FETCHED
3566 031714 104046          ERROR 46          ;WRONG DATA WAS FETCHED
3567                                     ;FOR TIGHTER SCOPE LOOP
3568                                     ;REPLACE ERROR CALL WITH
3569                                     ;'BR 19$' = 000766
3570 031716                                     ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
3571                                     ;
3572 031716 012737 031724 001110          MOV #21$, $LPERR      ;SET LOOP ON ERROR POINTER TO 21$
3573 031724 012737 030340 177776 21$:      MOV #030340,PSW      ;MAKE PREVIOUS MODE USER
3574 031732 012737 100000 001202          MOV #100000,$TMP2    ;LOAD TEST LOC. V.A. INTO $TMP2
3575 031740 012702 001202          MOV # $TMP2,R2      ;LOAD ADDRESS OF $TMP2 INTO R2
3576 031744 006572 000000          MFPI @0(R2)        ;USE $TMP2 TO FETCH VIRTUAL
3577                                     ;ADDRESS OF 60000
3578 031750 012601          MOV (KSP)+,R1      ;POP KERNEL STACK INTO R1
3579 031752 020001          CMP R0,R1          ;WAS DATA FETCHED SAME AS STORED
3580 031754 001401          BEQ 22$           ;BRANCH IF CORRECT DATA WAS FETCHED
3581 031756 104046          ERROR 46          ;WRONG DATA WAS FETCHED

```

```

3582                                     ;FOR TIGHTER SCOPE LOOP
3583                                     ;REPLACE ERROR CALL WITH
3584                                     ;'BR 21$' = 000762
3585 031760 012737 002150 000250 22$:  MOV  #MGMERR,MMVEC ;SET M.M. VECTOR TO NORMAL ROUTINE
3586 031766 012737 031200 001110      MOV  #1$, $LPERR ;SET LOOP POINTER TO START OF TEST
3587 031774 000423                       BR   TST44      ;:BRANCH TO NEXT TEST
3588
3589
3590 031776 012637 001266                 23$:  MOV  (KSP)+,TRAPPC ;SAVE PC & PS OF TRAP
3591 032002 012637 001270                MOV  (KSP)+,TRAPPS
3592 032006 013737 177572 001272        MOV  SR0,WASSR0 ;SAVE SR0 FOR ERROR TYPEOUT
3593 032014 013737 177576 001274        MOV  SR2,WASSR2 ;SAVE SR2 FOR ERROR TYPEOUT
3594 032022 042737 160000 177572        BIC  #160000,SR0 ;CLEAR ERROR BITS IN SR0 AND LEAVE
3595 032030 104051                       FRROR 51        ;TRIED TO READ NON-RESIDENT PAGE
3596                                     ;FOR TIGHTER SCOPE LOOP
3597                                     ;REPLACE ERROR CALL WITH
3598                                     ;A 'NOP' = 000240
3599 032032 013746 001270                MOV  TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
3600 032036 013746 001266                MOV  TRAPPC,-(KSP)
3601 032042 000002                       RTI
3602
3603
3604                                     ;*****
3605                                     ;*TEST 44      MOVE TO PREVIOUS (USER) I-SPACE
3606                                     ;*
3607                                     ;*      THIS TEST USES THE 'MTP1' INSTRUCTION TO ENSURE THAT THE
3608                                     ;*      PREVIOUS MODE IS CLOKED CORRECTLY
3609                                     ;*      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.
3610                                     ;*
3611                                     ;*
3612                                     ;*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
3613                                     ;*      WILL OCCUR AND TRAP TO 20$, WHERE THE ERRORS ARE REPORTED.
3614                                     ;*
3615                                     ;*****
3616 032044 000004                         TST44: SCOPE
3617 032046 012737 077406 172310 1$:  MOV  #77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE
3618 032054 012737 077406 177610        MOV  #77406,UIPDR4 ;USER I-SPACE PAGE 4 READ/WRITE
3619 032062 012737 000500 172350        MOV  #600,KIPAR4  ;MAP KERNEL I PAGE 4 TO 12K
3620 032070 012737 000600 177650        MOV  #600,UIPAR4  ;MAP USER I PAGE 4 TO 12K
3621 032076 012737 032656 000250        MOV  #20$,MMVEC  ;SET M.M. VECTOR TO 20$
3622                                     ;THE FOLLOWING WILL TEST DSTM=0 MTP1
3623                                     ;
3624 032104 012737 030340 177776 2$:  MOV  #030340,PSW ;MAKE PREVIOUS MODE USER
3625 032112 012746 007777                MOV  #7777,-(KSP) ;PUSH DATA ON KERNEL STACK
3626 032116 006606                       MTP1  USP        ;LOAD USER STACK POINTER
3627 032120 006506                       MFPI  USP        ;READ USER STACK POINTER
3628 032122 012601                       MOV  (KSP)+,R1   ;POP KERNEL STACK INTO R1
3629 032124 022701 007777                CMP  #7777,R1   ;WAS USER STACK POINTER CHANGED
3630 032130 001401                       BEQ  3$         ;BRANCH IF IT WAS
3631 032132 104050                       ERROR 50        ;USER STACK POINTER NOT CHANGED
3632                                     ;FOR TIGHTER SCOPE LOOP
3633                                     ;REPLACE ERROR CALL WITH
3634                                     ;'BR 2$' = 000764
3635 032134 012737 030340 177776 3$:  MOV  #030340,PSW ;MAKE PREVIOUS MODE USER
3636 032142 012746 000700                MOV  #USESTK,-(KSP) ;GET READY TO RESTORE USER S. POINT
3637 032146 006606                       MTP1  USP        ;RESTORE USER STACK POINTER
    
```

```

3638 032150          4$: ;THIS WILL TEST DSTM = 1 MTPi.
3639 032150 012737 032166 001110 MOV #5$, $LPERR ;SET LOOP ON ERROR POINTER TO 5$
3640 032156 012702 100000 MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
3641 032162 012700 125252 MOV #125252,R0 ;LOAD TEST DATA INTO R0
3642 032166 010046          5$: MOV R0,-(KSP) ;PUSH TEST DATA ON KERNEL STACK
3643 032170 105037 172310 CLR B KIPDR4 ;MAKE KERNEL I PAGE 4 NON-RESIDENT
3644 032174 006612 MTPi (R2) ;LOAD TEST DATA INTO PHYSICAL 60000
3645 032176 112737 000006 172310 MOV B #006,KIPDR4 ;MAKE KERNEL PAGE 4 RESIDENT
3646 032204 011201 MOV (R2),R1 ;READ FROM ADDRESS 60000
3647 032206 020001 CMP R0,R1 ;SEE IF DATA WAS STORED AT CORRECT PLACE
3648 032210 001401 BEQ 6$ ;BRANCH IF STORE WAS CORRECT
3649 032212 104047 ERROR 47 ;INCORRECT STORE
3650 ;FOR TIGHTER SCOPE LOOP
3651 ;REPLACE ERROR CALL WITH
3652 ;'BR 5$' = 000765
3653 032214          6$: ;THE FOLLOWING WILL TEST DSTM=2 MTPi.
3654
3655 032214 012737 032240 001110 MOV #8$, $LPERR ;SET LOOP ON ERROR POINTER TO 8$
3656 032222 012737 030340 177776 MOV #030340,PSW ;MAKE PREVIOUS MODE USER
3657 032230 012700 125252 MOV #125252,R0 ;LOAD TEST DATA INTO R0
3658 032234 012702 100000 MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
3659 032240 010046          8$: MOV R0,-(KSP) ;PUSH TEST DATA ON KERNEL STACK
3660 032242 105037 172310 CLR B KIPDR4 ;MAKE KERNEL PAGE 4 NON-RESIDENT
3661 032246 006612 MTPi (R2) ;LOAD TEST DATA INTO PHYSICAL 60000
3662 032250 112737 000006 172310 MOV B #006,KIPDR4 ;MAKE KERNEL PAGE 4 RESIDENT
3663 032256 013701 100000 MOV @#100000,R1 ;READ FROM ADDRESS 60000
3664 032262 020001 CMP R0,R1 ;SEE IF DATA WAS STORED CORRECTLY
3665 032264 001401 BEQ 9$ ;BRANCH IF STORE WAS CORRECT
3666 032266 104047 ERROR 47 ;INCORRECT STORE
3667 ;FOR TIGHTER SCOPE LOOP
3668 ;REPLACE ERROR CALL WITH
3669 ;'BR 8$' = 000764
3670 032270          9$: ;THIS WILL TEST DSTM = 3 MTPi.
3671 032270 012737 032310 001110 MOV #10$, $LPERR ;SET LOOP ON ERROR POINTER TO 10$
3672 032276 012737 030340 177776 MOV #030340,PSW ;MAKE PREVIOUS MODE USER
3673 032304 012700 052525 MOV #52525,R0 ;LOAD TEST DATA INTO R0
3674 032310 010046          10$: MOV R0,-(KSP) ;PUSH TEST DATA ON KERNEL STACK
3675 032312 105037 172310 CLR B KIPDR4 ;MAKE KERNEL I PAGE 4 NON-RESIDENT
3676 032316 006637 100000 MTPi @#100000 ;LOAD TEST DATA INTO PHYSICAL 60000
3677 032322 112737 000006 172310 MOV B #006,KIPDR4 ;MAKE KERNEL PAGE 4 RESIDENT
3678 032330 013701 100000 MOV @#100000,R1 ;READ FROM ADDRESS 60000
3679 032334 020001 CMP R0,R1 ;SEE IF DATA WAS STORED CORRECTLY
3680 032336 001401 BEQ 11$ ;BRANCH IF STORE WAS CORRECT
3681 032340 104047 ERROR 47 ;INCORRECT STORE
3682 ;FOR TIGHTER SCOPE LOOP
3683 ;REPLACE ERROR CALL WITH
3684 ;'BR 10$' = 000763
3685 032342          11$: ;THIS WILL TEST DSTM = 4 MTPi.
3686 032342 012737 032362 001110 MOV #12$, $LPERR ;SET LOOP ON ERROR POINTER TO 12$
3687 032350 012737 030340 177776 MOV #030340,PSW ;MAKE PREVIOUS MODE USER
3688 032356 012700 125252 MOV #125252,R0 ;LOAD TEST DATA INTO R0
3689 032362 010046          12$: MOV R0,-(KSP) ;PUSH TEST DATA ON KERNEL STACK
3690 032364 012702 100002 MOV #100002,R2 ;LOAD VIRTUAL ADDRESS INTO R2
3691 032370 105037 172310 CLR B KIPDR4 ;MAKE KERNEL I PAGE 4 NON-RESIDENT
3692 032374 006642 MTPi -(R2) ;LOAD TEST DATA INTO PHYSICAL 60000
3693 032376 112737 000006 172310 MOV B #006,KIPDR4 ;MAKE KERNEL PAGE 4 RESIDENT

```



```

3694 032404 013701 100000      MOV      @#100000,R1      ;READ FROM ADDRESS 60000
3695 032410 020001              CMP      R0,R1          ;SEE IF DATA WAS STORED CORRECTLY
3696 032412 001401              BEQ      13$            ;BRANCH IF STORE WAS CORRECT
3697 032414 104047              ERROR    47             ;INCORRECT STORE
3698                                ;FOR TIGHTER SCOPE LOOP
3699                                ;REPLACE ERROR CALL WITH
3700                                ;'BR 12$' = 000762
3701 032416                    13$:      ;THE FOLLOWING WILL TEST DSTM=5 MTP1.
3702                                ;
3703 032416 012737 032450 001110      MOV      #14$, $LPERR    ;SET LOOP ON ERROR POINTER TO 14$
3704 032424 012737 030340 177776      MOV      #030340,PSW     ;MAKE PREVIOUS MODE USER
3705 032432 012700 052525              MOV      #52525,R0       ;LOAD TEST DATA INTO R0
3706 032436 012702 001204              MOV      #<$TMP2+2>,R2   ;LOAD ADDR. OF LOC. $TMP2+2 INTO R2
3707 032442 012737 100000 001202      MOV      #100000,$TMP2   ;LOAD VIRT. ADDR. OF TEST LOC. INTO $TMP2
3708 032450 010046                    14$:      MOV      R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
3709 032452 105037 172310              CLRB    KIPDR4          ;MAKE KERNEL PAGE 4 NON-RESIDENT
3710 032456 006652                    MTP1    @-(R2)          ;LOAD TEST DATA INTO PHYSICAL 60000
3711 032460 112737 000006 172310      MOV      #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
3712 032466 013701 100000              MOV      @#100000,R1     ;READ FROM ADDRESS 60000
3713 032472 020001              CMP      R0,R1          ;SEE IF DATA WAS STORED CORRECTLY
3714 032474 001401              BEQ      15$            ;BRANCH IF STORE WAS CORRECT
3715 032476 104047              ERROR    47             ;INCORRECT STORE
3716                                ;FOR TIGHTER SCOPE LOOP
3717                                ;REPLACE ERROR CALL WITH
3718                                ;'BR 14$' = 000764
3719 032500                    15$:      ;THIS WILL TEST DSTM = 6 MTP1.
3720                                ;
3721 032500 012737 032522 001110      MOV      #16$, $LPERR    ;SET LOOP ON ERROR POINTER TO 16$
3722 032506 012737 030340 177776      MOV      #030340,PSW     ;MAKE PREVIOUS MODE USER
3723 032514 012700 052525              MOV      #52525,R0       ;LOAD TEST DATA INTO R0
3724 032520 005002                    CLR     R2              ;MAKE REGISTER 2 ZERO
3725 032522 010046                    16$:      MOV      R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
3726 032524 105037 172310              CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
3727 032530 006662 100000              MTP1    100000(R2)      ;LOAD TEST DATA INTO PHYSICAL 60000
3728 032534 112737 000006 172310      MOV      #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
3729 032542 013701 100000              MOV      @#100000,R1     ;READ FROM ADDRESS 60000
3730 032546 020001              CMP      R0,R1          ;SEE IF DATA WAS STORED CORRECTLY
3731 032550 001401              BEQ      17$            ;BRANCH IF STORE WAS CORRECT
3732 032552 104047              ERROR    47             ;INCORRECT STORE
3733                                ;FOR TIGHTER SCOPE LOOP
3734                                ;REPLACE ERROR CALL WITH
3735                                ;'BR 16$' = 000763
3736 032554                    17$:      ;THE FOLLOWING WILL TEST DSTM=7 MTP1.
3737                                ;
3738 032554 012737 032606 001110      MOV      #18$, $LPERR    ;SET LOOP ON ERROR POINTER TO 18$
3739 032562 012737 030340 177776      MOV      #030340,PSW     ;MAKE PREVIOUS MODE USER
3740 032570 012700 125252              MOV      #125252,R0     ;LOAD TEST DATA INTO R0
3741 032574 012737 100000 001202      MOV      #100000,$TMP2   ;LOAD VIRT. ADDR. OF TEST LOCATION
3742                                ;INTO LOCATION $TMP2
3743 032602 012702 001202                    MOV      #$TMP2,R2      ;LOAD ADDRESS OF $TMP2 INTO R2
3744 032606 010046                    18$:      MOV      R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
3745 032610 105037 172310              CLRB    KIPDR4          ;MAKE KERNEL PAGE 4 NON-RESIDENT
3746 032614 006672 000000              MTP1    @0(R2)          ;LOAD TEST DATA INTO PHYSICAL 60000
3747 032620 112737 000006 172310      MOV      #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
3748 032626 013701 100000              MOV      @#100000,R1     ;READ FROM ADDRESS 60000
3749 032632 020001              CMP      R0,R1          ;SEE IF DATA WAS STORED CORRECTLY
    
```



```

3750 032634 001401          BEQ      19$          ;BRANCH IF STORE WAS CORRECT
3751 032636 104047          ERROR    47          ;INCORRECT STORE
3752                                     ;FOR TIGHTER SCOPE LOOP
3753                                     ;REPLACE ERROR CALL WITH
3754                                     ;'BR 18$' = 000763
3755 032640 012737 032046 001110 19$:  MOV      #1$, $LPERR   ;SET LOOP POINTER TO START OF TEST
3756 032646 012737 002150 000250      MOV      #MGMERR,MMVEC ;RESTORE M.M. VECTOR TO NORMAL ROUTINE
3757 032654 000423          BR        TST45       ;;BRANCH TO NEXT TEST
3758
3759
3760 032656 012637 001266          20$:  MOV      (KSP)+,TRAPPC ;SAVE PC & PS OF TRAP
3761 032662 012637 001270          MOV      (KSP)+,TRAPPS
3762 032666 013737 177572 001272      MOV      SR0,WASSRO   ;SAVE SR0 FOR ERROR TYPEOUT
3763 032674 013737 177576 001274      MOV      SR2,WASSR2   ;SAVE SR2 FOR ERROR TYPEOUT
3764 032702 042737 160000 177572      BIC      #160000,SR0  ;CLEAR ERROR BITS IN SR0
3765 032710 104051          ERROR    51          ;TRIED TO LOAD A N.R. PAGE 4
3766                                     ;FOR TIGHTER SCOPE LOOP
3767                                     ;REPLACE ERROR CALL WITH
3768                                     ;A 'NOP' = 000240
3769 032712 013746 001270          MOV      TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
3770 032716 013746 001266          MOV      TRAPPC,-(KSP)
3771 032722 000002          RTI                    ;RETURN TO TEST
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787 032724 000004          TST45: SCOPE
3788 032726 012700 077406          1$:   MOV      #77406,R0   ;MAKE ALL USER I-SPACE PAGES RESIDENT
3789                                     ;READ/WRITE, LENGTH 200 BLOCKS
3790 032732 012702 000010          MOV      #10,R2       ;SET LOOP COUNTER TO 8
3791 032736 012701 177600          MOV      #UIPDR0,R1   ;LOAD ADDRESS OF FIRST PDR IN R1
3792 032742 010021          2$:   MOV      R0,(R1)+     ;LOAD PDR WITH 77406
3793 032744 077202          SOB      R2,2$        ;LOOP UNTIL 8 USER PDRS LOADED
3794 032746 012737 032754 001110      MOV      #3$, $LPERR  ;SET LOOP ON ERROR TO 3$
3795 032754 012737 140340 177776      3$:   MOV      #140340,PSW  ;GO TO USER MODE FOR THIS TEST
3796 032762 012737 077406 172310      MOV      #77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE
3797 032770 012737 000600 172350      MOV      #600,KIPAR4  ;MAP KERNEL I PAGE 4 TO 12K
3798 032776 012737 000600 177650      MOV      #600,UIPAR4  ;MAP USER I PAGE 4 TO 12K
3799 033004 012700 036514          MOV      #36514,R0    ;LOAD DATA PATTERN INTO R0
3800 033010 010037 100000          MOV      R0,#100000   ;LOAD DATA PATTERN INTO PHY 60000
3801 033014 012702 100000          MOV      #100000,R2   ;LOAD VIRTUAL ADDRESS INTO R2
3802                                     ;THE FOLLOWING WILL TEST DSTM=0 MFPI
3803
3804 033020 012737 033422 000250      MOV      #21$,MMVEC   ;SET M.M. VECTOR TO 21$
3805 033026 105037 177610          CLRB     UIPDR4       ;MAKE USER I-SPACE PAGE 4 NON-RESIDENT
    
```

```

:*****
:*TEST 45      MOVE FROM PREVIOUS (KERNEL) I-SPACE TO USER MODE
:
:*          THIS TEST CHECKS THAT IF THE PREVIOUS MODE IS KERNEL THE
:*          FETCH IS FROM KERNEL SPACE.
:*          THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.
:
:*          IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
:*          WILL OCCUR AND TRAP TO 21$, WHERE THE ERRORS ARE REPORTED.
:*****
    
```

L 6

MD-11-CJKDA-B KTF11-AA MMU DIAG MACY11 30A(1052) 11-JUN-79 13:21 PAGE 76  
 CJKDAB.P11 11-JUN-79 13:10 T45 MOVE FROM PREVIOUS (KERNEL) I-SPACE TO USER MODE SEQ 0076

```

3806 033032 012737 140340 177776      MOV      #140340,PSW      ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3807 033040 006506                MFPI     KSP              ;PUT KERNEL STACK POINTER ON USER STACK
3808 033042 022706 000700          CMP      #USESTK,USP     ;WAS SOMETHING PUSHED ON STACK AT 1$
3809 033046 001407                BEQ      5$              ;BRANCH IF NOTHING WAS PUSHED
3810 033050 012600                MOV      (USP)+,R0       ;POP USER STACK INTO R0
3811 033052 012701 001100          MOV      #KERSTK,R1     ;EXPECTING 1100 AS KSP
3812 033056 020001                CMP      R0,R1          ;DID YOU GET THE RIGHT POINTER?
3813 033060 001403                BEQ      6$              ;BRANCH IF YOU DID
3814 033062 104046                ERROR   46              ;WRONG THING WAS PUSHED ON STACK
3815                                ;FOR TIGHTER SCOPE LOOP
3816                                ;REPLACE ERROR CALL WITH
3817                                ;'BR 4$' = 000766
3818 033064 000401                BR       6$              ;BRANCH TO NEXT TRY
3819 033066 104050          5$:      ERROR   50              ;NOTHING PUSHED ON STACK
3820                                ;FOR TIGHTER SCOPE LOOP
3821                                ;REPLACE ERROR CALL WITH
3822                                ;'BR 4$' = 000764
3823 033070                6$:      ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
3824 033070 012737 033076 001110          MOV      #7$, $LPERR    ;SET LOOP ON ERROR POINTER TO 7$
3825 033076 012737 140340 177776          7$:      MOV      #140340,PSW    ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3826 033104 012700 036514                MOV      #36514,R0      ;LOAD DATA EXPECTED INTO R0
3827 033110 012702 100000                MOV      #100000,R2     ;LOAD VIRTUAL ADDRESS INTO R2
3828 033114 006512                MFPI     (R2)           ;READ FROM PHYSICAL 60000
3829 033116 012601                MOV      (USP)+,R1     ;POP USER STACK INTO R1
3830 033120 020001                CMP      R0,R1          ;WAS DATA FETCHED SAME AS STORED
3831 033122 001401                BEQ      8$              ;BRANCH IF CORRECT DATA WAS FETCHED
3832 033124 104046                ERROR   46              ;WRONG DATA WAS FETCHED
3833                                ;FOR TIGHTER SCOPE LOOP
3834                                ;REPLACE ERROR CALL WITH
3835                                ;'BR 7$' = 000764
3836 033126                8$:      ;THE FOLLOWING WILL TEST DSM=2 MFPI.
3837 033126 012737 033134 001110          MOV      #9$, $LPERR    ;SET LOOP ON ERROR POINTER TO 9$
3838 033134 012737 140340 177776          9$:      MOV      #140340,PSW    ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3839 033142 012702 100000                MOV      #100000,R2     ;LOAD VIRTUAL ADDRESS INTO R2
3840 033146 006522                MFPI     (R2)+          ;READ FROM PHYSICAL 60000
3841 033150 012601                MOV      (USP)+,R1     ;POP USER STACK INTO R1
3842 033152 020001                CMP      R0,R1          ;WAS DATA FETCHED SAME AS STORED
3843 033154 001401                BEQ      10$             ;BRANCH IF CORRECT DATA WAS FETCHED
3844 033156 104046                ERROR   46              ;WRONG DATA WAS FETCHED
3845                                ;FOR TIGHTER SCOPE LOOP
3846                                ;REPLACE ERROR CALL WITH
3847                                ;'BR 9$' = 000766
3848 033160                10$:     ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
3849 033160 012737 033166 001110          MOV      #11$, $LPERR   ;SET LOOP ON ERROR POINTER TO 11$
3850 033166 012737 140340 177776          11$:     MOV      #140340,PSW    ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3851 033174 006537 100000                MFPI     @#100000       ;READ FROM PHYSICAL 60000
3852 033200 012601                MOV      (USP)+,R1     ;POP USER STACK INTO R1
3853 033202 020001                CMP      R0,R1          ;WAS DATA FETCHED SAME AS STORED
3854 033204 001401                BEQ      12$             ;BRANCH IF CORRECT DATA WAS FETCHED
3855 033206 104046                ERROR   46              ;WRONG DATA WAS FETCHED
3856                                ;FOR TIGHTER SCOPE LOOP
3857                                ;REPLACE ERROR CALL WITH
3858                                ;'BR 11$' = 000767
3859 033210                12$:     ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
3860 033210 012737 033216 001110          MOV      #13$, $LPERR   ;SET LOOP ON ERROR POINTER TO 13$
3861 033216 012737 140340 177776          13$:     MOV      #140340,PSW    ;MAKE PREVIOUS MODE DERNEL PRESENT USER

```

```

3862 033224 012702 100002      MOV      #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3863 033230 006542      MFPI    -(R2)          ;READ FROM PHYSICAL 60000
3864 033232 012601      MOV     (USP)+,R1      ;POP USER STACK INTO R1
3865 033234 020001      CMP     R0,R1          ;WAS DATA FETCHED SAME AS STORED
3866 033236 001401      BEQ    14$             ;BRANCH IF CORRECT DATA WAS FETCHED
3867 033240 104046      ERROR  46             ;WRONG DATA WAS FETCHED
3868                                ;FOR TIGHTER SCOPE LOOP
3869                                ;REPLACE ERROR CALL WITH
3870                                ;'BR 13$' = 000766
3871 033242                14$: ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
3872                                ;
3873 033242 012737 033250 001110      MOV     #15$,SLPERR    ;SET LOOP ON ERROR POINTER TO 15$
3874 033250 012737 140340 177776      MOV     #140340,PSW   ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3875 033256 012737 100000 001202      MOV     #100000,$TMP2 ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
3876 033264 012702 001204                MOV     #<$TMP2+2>,R2 ;LOAD ADDRESS OF $TMP2+2 INTO R2
3877 033270 006552      MFPI    @-(R2)        ;READ FROM PHYSICAL 60000
3878 033272 012601      MOV     (USP)+,R1      ;POP USER STACK INTO R1
3879 033274 020001      CMP     R0,R1          ;WAS DATA FETCHED SAME AS STORED
3880 033276 001401      BEQ    16$             ;BRANCH IF CORRECT DATA FETCHED
3881 033300 104046      ERROR  46             ;WRONG DATA WAS FETCHED
3882                                ;FOR TIGHTER SCOPE LOOP
3883                                ;REPLACE ERROR CALL WITH
3884                                ;'BR 15$' = 000763
3885 033302                16$: ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
3886                                ;
3887 033302 012737 033310 001110      MOV     #17$,SLPERR    ;SET LOOP ON ERROR POINTER TO 17$.
3888 033310 012737 140340 177776      MOV     #140340,PSW   ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3889 033316 005002      CLR     R2             ;MAKE REGISTER 2 A ZERO
3890 033320 006562 100000      MFPI    100000(R2)    ;READ FROM PHYSICAL 60000
3891 033324 012601      MOV     (USP)+,R1      ;POP USER STACK INTO R1
3892 033326 020001      CMP     R0,R1          ;WAS DATA FETCHED SAME AS STORED
3893 033330 001401      BEQ    18$             ;BRANCH IF CORRECT DATA FETCHED
3894 033332 104046      ERROR  46             ;WRONG DATA WAS FETCHED
3895                                ;FOR TIGHTER SCOPE LOOP
3896                                ;REPLACE ERROR CALL WITH
3897                                ;'BR 17$' = 000766
3898 033334                18$: ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
3899                                ;
3900 033334 012737 033342 001110      MOV     #19$,SLPERR    ;SET LOOP ON ERROR POINTER TO 19$
3901 033342 012737 140340 177776      MOV     #140340,PSW   ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3902 033350 012737 100000 001202      MOV     #100000,$TMP2 ;LOAD TEST LOC. VIRT. ADDR INTO $TMP2
3903 033356 012702 001202                MOV     #$TMP2,R2     ;LOAD ADDRESS OF $TMP2 INTO R2
3904 033362 006572 000000      MFPI    @0(R2)        ;READ FROM PHYSICAL 60000
3905 033366 012601      MOV     (USP)+,R1      ;POP USER STACK INTO R1
3906 033370 020001      CMP     R0,R1          ;WAS DATA FETCHED SAME AS STORED
3907 033372 001401      BEQ    20$             ;BRANCH IF CORRECT DATA FETCHED
3908 033374 104046      ERROR  46             ;WRONG DATA WAS FETCHED
3909                                ;FOR TIGHTER SCOPE LOOP
3910                                ;REPLACE ERROR CALL WITH
3911                                ;'BR 19$' = 000762
3912 033376 012737 002150 000250      MOV     #MGMERR,MMVEC ;SET M.M. VECTOR TO NORMAL ROUTINE
3913 033404 012737 000340 177776      MOV     #00340,PSW    ;GO BACK TO KERNEL MODE, PREVIOUS KERNEL
3914 033412 012737 032726 001110      MOV     #1$,SLPERR    ;SET LOOP POINTER TO START OF TEST
3915 033420 000423      BR     TST46          ;BRANCH TO NEXT TEXT
3916
3917

```

```

3918 033422 012637 001266      21$:  MOV      (KSP)+,TRAPPC      ;SAVE PC & PS OF TRAP
3919 033426 012637 001270      MOV      (KSP)+,TRAPPS
3920 033432 013737 177572 001272  MOV      SR0,WASSR0      ;SAVE SR0 FOR ERROR TYPEOUT
3921 033440 013737 177576 001274  MOV      SR2,WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
3922 033446 042737 160000 177572  BIC      #160000,SR0      ;CLEAR ERROR BITS IN SR0
3923 033454 104051                                ERROR 51      ;TRIED TO READ NON-RESIDENT PAGE
3924                                ;FOR TIGHTER SCOPE LOOP
3925                                ;REPLACE ERROR CALL WITH
3926                                ;A 'NOP' = 000240
3927 033456 013746 001270      MOV      TRAPPS,-(KSP)    ;PUT PC & PS OF TRAP ON STACK
3928 033462 013746 001266      MOV      TRAPPC,-(KSP)
3929 033466 000002      RTI                                ;RETURN TO TEST
3930
3931                                ;*****
3932                                ;*TEST 46      MOVE FROM/TO D-SPACE = MOVE FROM/TO I-SPACE
3933                                ;*
3934                                ;*      THIS TEST CHECKS THAT SINCE THERE IS NO DISTINCTION
3935                                ;*      BETWEEN INSTRUCTION AND DATA SPACE IN THE FONZ-11
3936                                ;*      MFPD & MTPD SHOULD BE DECODED THE SAME AS MFPI & MTPi.
3937                                ;*
3938                                ;*****
3939 033470 000004      TST46:  SCOPE
3940 033472 012737 030340 177776  1$:  MOV      #030340,PSW      ;MAKE PREVIOUS MODE=USER,CURRENT=KERNEL
3941 033500 106506      MFPD      USP            ;MFPD SHOULD ACT LIKE MFPI PUTTING
3942                                ;USER STACK POINTER ON THE KERNEL STACK
3943 033502 022706 001100      CMP      #KERSTK,KSP     ;WAS SOMETHING PUSHED ON KERNEL STACK?
3944 033506 001407      BEQ      2$              ;BRANCH IF NO
3945 033510 012600      MOV      (KSP)+,R0       ;POP KERNEL STACK INTO R0
3946 033512 012701 000700      MOV      #USESTK,R1      ;EXPECTING TO GET 700 AS USP
3947 033516 020001      CMP      R0,R1           ;DID GET RIGHT POINTER VALUE?
3948 033520 001403      BEQ      3$              ;BRANCH IF YES
3949 033522 104053      ERROR 53                ;WRONG THING WAS PUSHED ON STACK
3950                                ;FOR TIGHTER SCOPE LOOP
3951                                ;REPLACE ERROR CALL WITH
3952                                ;'BR 1$' = 000763
3953 033524 000401      BR       3$              ;BRANCH TO NEXT TRY
3954 033526 104054      2$:  ERROR 54                ;NOTHING PUSHED ON STACK
3955                                ;FOR TIGHTER SCOPE LOOP
3956                                ;REPLACE ERROR CALL WITH
3957                                ;'BR 1$' = 000761
3958 033530 012737 033536 001110  3$:  MOV      #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
3959 033536 012746 007777  4$:  MOV      #7777,-(KSP)    ;PUSH DATA ON KERNEL STACK
3960 033542 106606      MTPD      USP            ;LOAD THE USER STACK POINTER
3961 033544 106506      MFPD      USP            ;READ USER STACK POINTER
3962 033546 012601      MOV      (KSP)+,R1       ;POP KERNEL STACK INTO R1
3963 033550 022701 007777      CMP      #7777,R1        ;WAS USER STACK POINTER CHANGED?
3964 033554 001401      BEQ      5$              ;BRANCH IF YES
3965 033556 104054      ERROR 54                ;USER STACK POINTER NOT CHANGED
3966                                ;FOR TIGHTER SCOPE LOOP
3967                                ;REPLACE ERROR CALL WITH
3968                                ;'BR 4$' = 000767
3969 033560 012746 000700  5$:  MOV      #USESTK,-(KSP)  ;GET READY TO RESTORE USER STK. PTR.
3970 033564 106606      MTPD      USP            ;RESTORE USER STACK POINTER
3971 033566 012737 033472 001110  MOV      #1$,$LPERR      ;SET LOOP POINTER TO START OF TEST
3972
3973                                ;*****
    
```

```
3974 ;*TEST 47 MOVE FROM PREVIOUS I=SPACE (PREVIOUS=CURRENT=KERNEL)
3975 ;*
3976 ;* THIS TEST CHECKS THAT IF BOTH PREVIOUS AND CURRENT MODES
3977 ;* ARE KERNEL, AND THE SOURCE MODE IS 0, THE DESTINATION
3978 ;* STACK IS NOT DECREMENTED BEFORE ACCESS.
3979 ;* 'MFPI KSP' SHOULD PUSH THE NON-DECREMENTED VALUE
3980 ;* OF KSP (1100) ONTO THE STACK (AT LOC. 1076).
3981 ;*****
3982 033574 000004 TST47: SCOPE
3983 033576 005037 177776 1$: CLR @#PSW ;SET PREVIOUS = CURRENT = KERNEL
3984 033602 012700 001100 MOV #STACK,R0 ;SETUP VALUE FOR STACK POINTER
3985 033606 010006 MOV R0,KSP ;LOAD STACK POINTER
3986 033610 006506 MFPI KSP ;THE VALUE 'STACK' SHOULD BE PUSHED
3987 ;BEFORE BEING DECREMENTED
3988 033612 011601 MOV (KSP),R1 ;READ DATA WHICH WAS PUSHED
3989 033614 020001 CMP R0,R1 ;WAS THE ORIGINAL VALUE OF THE
3990 ;STACK POINTER PUSHED?
3991 033616 001401 BEQ 2$ ;BRANCH IF YES
3992 033620 104046 ERROR 46 ;MFPI FETCHED WRONG DATA
3993 ;FOR TIGHTER SCOPE LOOP
3994 ;REPLACE ERROR CALL WITH
3995 ;'BR 1$' = 000766
3996 033622 005740 2$: TST -(R0) ;SETUP EXPECTED STACK POINTER VALUE
3997 033624 020600 CMP KSP,R0 ;WAS THE STACK POINTER DECREMENTED?
3998 033626 001401 BEQ 3$ ;BRANCH IF YES
3999 033630 104050 ERROR 50 ;STACK NOT PUSHED BY THE MFPI
4000 ;FOR TIGHTER SCOPE LOOP
4001 ;REPLACE ERROR CALL WITH
4002 ;'BR 1$' = 000762
4003 033632 012706 001100 3$: MOV #STACK,KSP ;RESTORE STACK POINTER
4004
4005
4006
4007
4008
4009 033636 013746 000004 TIMOFF: MOV @#ERRVEC,-(SP) ;SAVE CONTENTS OF LOC 4
4010 033642 012737 033654 000004 MOV #1$,@#ERRVEC ;SETUP IN CASE OF TIMEOUT
4011 033650 005337 164000 DEC @#164000 ;TURN OFF TIMER ON MULTI-TESTER
4012 033654 012637 000004 1$: MOV (SP)+,@#ERRVEC ;RESTORE CONTENTS OF LOC 4
4013
4014 .SBTTL END OF PASS ROUTINE
4015
4016 ;*****
4017 ;*INCREMENT THE PASS NUMBER ($PASS)
4018 ;*TYPE 'END OF PASS #XXXX ;TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYY''
4019 ;*WHERE XXXX AND YYYY ARE DECIMAL NUMBERS
4020 ;*IF SW12=1 INHIBIT TRACE TRAP
4021 ;*IF THERES A MONITOR GO TO IT
4022 ;*IF THERE ISN'T JUMP TO LOOP
4023
4024 033660 $EOP:
4025 033660 000004 SCOPE
4026 033662 005037 001102 CLR $STNM ;;ZERO THE TEST NUMBER
4027 033666 005037 001212 CLR $TIMES ;;ZERO THE NUMBER OF ITERATIONS
4028 033672 005237 001234 INC $PASS ;;INCREMENT THE PASS NUMBER
4029 033676 042737 100000 001234 BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
```

```

4030 033704 005327          DEC      (PC)+          ;;LOOP?
4031 033706 000001          $EOPCT: .WORD      1
4032 033710 003074          BGT      $DOAGN
4033 033712 012737          MOV      (PC)+,@(PC)+  ;;YES
4034 033714 000001          $ENDCT: .WORD      1          ;;RESTORE COUNTER
4035 033716 033706          $EOPCT
4036 033720 104401 033726          TYPE    ,65$          ;;TYPE ASCIZ STRING
4037 033724 000407          BR      64$          ;;GET OVER THE ASCIZ
4038 033726 006412 047105 020104 65$: .ASCIZ <12><15>/END PASS #/
4039 033734 040520 051523 021440
4040 033742          000
4041          033744          .EVEN
4042 033744          64$:
4043 033744 013746 001234          MOV      $PASS,-(SP)    ;;SAVE $PASS FOR TYPEOUT
4044          ;;TYPE PASS NUMBER
4045 033750 104405          TYPDS    ;;GO TYPE--DECIMAL ASCIZ WITH SIGN
4046 033752 104401 033760          TYPE    ,67$          ;;TYPE ASCII STRING
4047 033756 000425          BR      66$          ;;GET OVER THE ASCIZ
4048 033760 035411 047524 040524 67$: .ASCIZ /          ;TOTAL ERRORS SINCE LAST START AT 200 /
4049 033766 020114 051105 047522
4050 033774 051522 051440 047111
4051 034002 042503 046040 051501
4052 034010 020124 052123 051101
4053 034016 020124 052101 031040
4054 034024 030060 020040          000
4055          034032          .EVEN
4056 034032          66$:
4057 034032 013746 001112          MOV      $ERTTL,-(SP)  ;;SAVE $ERTTL FOR TYPEOUT
4058          ;;TOTAL NUMBER OF ERRORS
4059 034036 104405          TYPDS    ;;GO TYPE--DECIMAL ASCII WITH SIGN
4060 034040 104401 001223          TYPE    , $CRLF
4061 034044 013700 000042          $GET42: MOV      @#42,R0  ;;TYPE CARRIAGE RETURN, LINE FEED
4062 034050 001414          BEQ      $DOAGN        ;;GET MONITOR ADDRESS
4063 034052 005046          CLR      -(SP)         ;;BRANCH IF NO MONITOR
4064 034054 012746 034062          MOV      #$CLR.T,-(SP) ;;INSURE THE 'T' BIT IS CLEAR
4065 034060 000426          BR      $RTRN          ;;SETUP FOR AN RTI OR RTT
4066          ;;GO DO AN RTI OR RTT TO LOAD THE PSW
4067          ;;WITH A CLEARED 'T' BIT
4068 034062 013700 000042          $CLR.T: MOV      @#42,R0    ;;INSURE R0 CONTAINS THE MONITORS
4069 034066 001405          BEQ      $DOAGN        ;;RETURN ADDRESS
4070 034070 000005          RESET   ;;CLEAR THE WORLD
4071 034072 004710          $ENDAD: JSR      PC,(R0) ;;GO TO MONITOR
4072 034074 000240          NOP     ;;SAVE ROOM
4073 034076 000240          NOP     ;;FOR
4074 034100 000240          NOP     ;;ACT11
4075
4076 034102          $DOAGN:
4077 034102 104400          TRAP
4078 034104 042716 000020          BIC      #20,(SP)      ;;PUSH OLD PSW AND PC ON STACK
4079 034110 032777 010000 145022          BIT      #BIT12,@SWR  ;;CLEAR THE 'T' BIT
4080 034116 001005          BNE     1$            ;;RUN WITH TRACE TRAP?
4081 034120 005137 034144          COM     $TBIT         ;;BR IF NO
4082 034124 100402          BMI     1$            ;;IS IT TIME FOR TRACE TRAP
4083 034126 052716 000020          BIS     #20,(SP)      ;;BR IF NO
4084 034132 012746 034140          1$: MOV      #$LOOP,-(SP) ;;SET TRACE TRAP
4085 034136 000002          $RTRN: RTI          ;;JUMP TO START OF TEST
          ;;RETURN--THIS IS CHANGED TO
    
```

```

4086                                     ;;AN 'RTT' IF 'RTT' IS A LEGAL
4087                                     ;;INSTRUCTION
4088 034140                               $LOOP:
4089 034140 000137                       JMP @ (PC)+ ;;RETURN
4090 034142 020472                       $RTNAD: .WORD RESTR
4091 034144 C00000                       $TBIT: .WORD 0 ;;'T' BIT STATE INDICATOR
4092 034146 377 377 000                 $ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
4093                                     .EVEN
4094                                     .SBTTL SCOPE HANDLER ROUTINE
4095
4096                                     ;:*****
4097                                     ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
4098                                     ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
4099                                     ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
4100                                     ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4101                                     ;*SW14=1 LOOP ON TEST
4102                                     ;*SW11=1 INHIBIT ITERATIONS
4103                                     ;*SW09=1 LOOP ON ERROR
4104                                     ;*SW08=1 LOOP ON TEST IN SWR<7:0>
4105                                     ;*CALL
4106                                     ;* SCOPE ;;SCOPE=IOT
4107
4108 034152                               $SCOPE:
4109 034152 104410                           CKSWR
4110 034154 032777 040000 144756          1$: BIT #BIT14,@SWR ;;TEST FOR CHANGE IN SOFT-SWR
4111 034162 001114                           BNE $OVER ;;LOOP ON PRESENT TEST?
4112                                     ;;YES IF SW14=1
4113 034164 000416                           ;:#####START OF CODE FOR THE XOR TESTER#####
4114                                     $XTSTR: BR 6$ ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
4115                                     MOV @#ERRVEC,-(SP) ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
4116 034172 012737 034212 000004           MOV #5$,@#ERRVEC ;;SAVE THE CONTENTS OF THE ERROR VECTOR
4117 034200 005737 177060                 TST @#177060 ;;SET FOR TIMEOUT
4118 034204 012637 000004                 MOV (SP)+,@#ERRVEC ;;TIME OUT ON XOR?
4119 034210 000463                           BR $SVLAD ;;RESTORE THE ERROR VECTOR
4120 034212 022626                           5$: CMP (SP)+,(SP)+ ;;GO TO THE NEXT TEST
4121 034214 012637 000004                 MOV (SP)+,@#ERRVEC ;;CLEAR THE STACK AFTER A TIME OUT
4122 034220 000423                           BR 7$ ;;RESTORE THE ERROR VECTOR
4123 034222                                     6$:;#####END OF CODE FOR THE XOR TESTER#####
4124 034222 032777 000400 144710           BIT #BIT08,@SWR ;;LOOP ON SPEC. TEST?
4125 034230 001404                           BEQ 2$ ;;BR IF NO
4126 034232 127737 144702 001102           CMPB @SWR,$TSTNM ;;ON THE RIGHT TEST? SWR<7:0>
4127 034240 001465                           BEQ $OVER ;;BR IF YES
4128 034242 105737 001103                 2$: TSTB $ERFLG ;;HAS AN ERROR OCCURRED?
4129 034246 001421                           BEQ 3$ ;;BR IF NO
4130 034250 123737 001115 001103           CMPB $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
4131 034256 101015                           BHI 3$ ;;BR IF NO
4132 034260 032777 001000 144652           BIT #BIT09,@SWR ;;LOOP ON ERROR?
4133 034266 001404                           BEQ 4$ ;;BR IF NO
4134 034270 013737 001110 001106          7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
4135 034276 000446                           BR $OVER
4136 034300 105037 001103                 4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
4137 034304 005037 001212                 CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
4138 034310 000415                           BR 1$ ;;ESCAPE TO THE NEXT TEST
4139 034312 032777 004000 144620          3$: BIT #BIT11,@SWR ;;INHIBIT ITERATIONS?
4140 034320 001011                           BNE 1$ ;;BR IF YES
4141 034322 005737 001234                 TST $PASS ;;IF FIRST PASS OF PROGRAM
    
```



```

4142 034326 001406          BEQ      1$          ;; INHIBIT ITERATIONS
4143 034330 005237 001104    INC      $ICNT       ;; INCREMENT ITERATION COUNT
4144 034334 023737 001212 001104    CMP      $TIMES,$ICNT ;; CHECK THE NUMBER OF ITERATIONS MADE
4145 034342 002024          BGE      $OVER      ;; BR IF MORE ITERATION REQUIRED
4146 034344 012737 000001 001104 1$:    MOV      #1,$ICNT   ;; REINITIALIZE THE ITERATION COUNTER
4147 034352 013737 034430 001212    MOV      $MXCNT,$TIMES ;; SET NUMBER OF ITERATIONS TO DO
4148 034360 105237 001102          $SVLAD: INCB     $TSTNM  ;; COUNT TEST NUMBERS
4149 034364 113737 001102 001232    MOV      $TSTNM,$TSTNM ;; SET TEST NUMBER IN APT MAILBOX
4150 034372 011637 001106          MOV      (SP),$LPADR ;; SAVE SCOPE LOOP ADDRESS
4151 034376 011637 001110          MOV      (SP),$LPERR ;; SAVE ERROR LOOP ADDRESS
4152 034402 005037 001214          CLR      $ESCAPE    ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
4153 034406 112737 000001 001115    MOV      #1,$ERMAX  ;; ONLY ALLOW ONE(1) ERROR ON NEXT TES'
4154 034414 013777 001102 144520 $OVER: MOV      $TSTNM,@DISPLAY ;; DISPLAY TEST NUMBER
4155 034422 013716 001106          MOV      $LPADR,(SP) ;; FUDGE RETURN ADDRESS
4156 034426 000002          RTI              ;; FIXES PS
4157 034430 000200          $MXCNT: 200       ;; MAX. NUMBER OF ITERATIONS
4158          .SBTTL  ERROR HANDLER ROUTINE
4159
4160          ;;*****
4161          ;;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
4162          ;;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
4163          ;;*AND GO TO ERRTP ON ERROR
4164          ;;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4165          ;;*SW15=1      HALT ON ERROR
4166          ;;*SW13=1      INHIBIT ERROR TYPEOUTS
4167          ;;*SW10=1     BELL ON ERROR
4168          ;;*SW09=1    LOOP ON ERROR
4169          ;;*CALL
4170          ;;*      ERROR      N      ;;:ERROR=EMT AND N=ERROR ITEM NUMBER
4171
4172          $ERROR:
4173          CKSWR
4174          MOV      R0,$REG0  ;; TEST FOR CHANGE IN SOFT-SWR
4175          MOV      R1,$REG1  ;; SAVE THE CONTENTS OF R0
4176          MOV      R2,$REG2  ;; SAVE THE CONTENTS OF R1
4177          MOV      R3,$REG3  ;; SAVE THE CONTENTS OF R2
4178          MOV      R4,$REG4  ;; SAVE THE CONTENTS OF R3
4179          MOV      R5,$REG5  ;; SAVE THE CONTENTS OF R4
4180          MOV      $TSTNM,TESTNO ;; SAVE THE CONTENTS OF R5
4181          MOV      $ERFLG,TESTNO ;; SAVE THE TEST NUMBER
4182          INCB     $ERFLG    ;; SET THE ERROR FLAG
4183          BEQ      7$          ;; DON'T LET THE FLAG GO TO ZERO
4184          MOV      $TSTNM,@DISPLAY ;; DISPLAY TEST NUMBER AND ERROR FLAG
4185          BIT      #BIT10,@SWR ;; BELL ON ERROR?
4186          BEQ      1$          ;; NO - SKIP
4187          TYPE    ,SBELL     ;; RING BELL
4188          INC      $ERTTL    ;; COUNT THE NUMBER OF ERRORS
4189          MOV      (SP),$ERRPC ;; GET ADDRESS OF ERROR INSTRUCTION
4190          SUB      #2,$ERRPC
4191          MOV      @ $ERRPC,$ITEMB ;; STRIP AND SAVE THE ERROR ITEM CODE
4192          BIT      #BIT13,@SWR ;; SKIP TYPEOUT IF SET
4193          BNE     20$         ;; SKIP TYPEOUTS
4194          JSR     PC,ERRTP    ;; GO TO USER ERROR ROUTINE
4195          TYPE    ,SCRLF
4196          CMPB    #APTENV,$ENV ;; RUNNING IN APT MODE
4197          BNE     2$          ;; NO,SKIP APT ERROR REPORT
    
```



```

4198 034576 113737 001114 034610      MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
4199 034604 004737 037232              JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
4200 034610      000                21$:   .BYTE    0
4201 034611      000                .BYTE    0
4202 034612 000777              22$:   BR      22$          ;;APT ERROR LOOP
4203 034614 005777 144320        2$:   TST     @SWR          ;;HALT ON ERROR
4204 034620 100002              BPL     3$              ;;SKIP IF CONTINUE
4205 034622 000000              HALT                    ;;HALT ON ERROR!
4206 034624 104410              CKSWR                    ;;TEST FOR CHANGE IN SOFT-SWR
4207 034626 032777 001000 144304  3$:   BIT     #BIT09,@SWR    ;;LOOP ON ERROR SWITCH SET?
4208 034634 001402              BEQ     4$              ;;BR IF NO
4209 034636 013716 001110        MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
4210 034642 005737 001214        4$:   TST     $ESCAPE      ;;CHECK FOR AN ESCAPE ADDRESS
4211 034646 001402              BEQ     5$              ;;BR IF NONE
4212 034650 013716 001214        MOV     $ESCAPE,(SP)   ;;FUDGE RETURN ADDRESS FOR ESCAPE
4213 034654
4214 034654 022737 034072 000042      CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
4215 034662 001001              BNE     6$              ;;BRANCH IF NO
4216 034664 000000              HALT                    ;;YES
4217 034666
4218 034666 000002              6$:   RTI                    ;;RETURN
    
```

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

```

4221 *****
4222 *THIS ROUTINE USES THE 'ITEM CONTROL BYTE' ($ITEMB) TO DETERMINE WHICH
4223 *ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE 'ERROR TABLE' ($ERRTB),
4224 *AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
4225 *
    
```

```

4226 *NOTES:
4227 *1) THIS ROUTINE PROVIDES AN AUTOMATIC 'CARRIAGE RETURN-LINE FEED'
4228 *   FOR 'EM', 'DH', AND 'DT'.
4229 *2) TWO SPACES ARE TYPED AFTER EACH NUMBER FOR 'DT'
4230 *3) FOR $ITEMB=0, JUST THE ERROR PC IS TYPED
4231 *4) THE AVAILABLE FORMATS FOR TYPING DATA ARE:
4232 *
4233 *   DF      FORMAT
4234 *   0      TYPE A 6 DIGIT OCTAL NUMBER (FROM 16-BIT BINARY)
4235 *   1      TYPE A DECIMAL NUMBER WITHOUT LEADING ZEROS
4236 *   2      TYPE A 16 DIGIT BINARY NUMBER
4237 *   3      TYPE A 6 DIGIT OCTAL NUMBER (FROM 18-BIT BINARY)
4238 *
    
```

```

4239 034670      ERRTYP:
4240 034670 104401 001223      TYPE    ,$CRLF          ;'CARRIAGE RETURN' & 'LINE FEED'
4241 034674 010046              MOV     RO,-(SP)        ;SAVE RO
4242 034676 005000              CLR     RO              ;PICKUP THE ITEM INDEX
4243 034700 153700 001114      BISB   @#$ITEMB,RO
4244 034704 001004              BNE     1$              ;IF ITEM NUMBER IS ZERO, JUST
4245 034706 013746 001116      MOV     $ERRPC,-(SP)   ;TYPE THE PC OF THE ERROR
4246 034712 104402              ;SAVE $ERRPC FOR TYPEOUT
4247 034714 000471              ;ERROR ADDRESS
4248 034716 005300              TYPOC                    ;GO TYPE--OCTAL ASCII(ALL DIGITS)
4249 034720 006300              1$:   BR      13$          ;GET OUT
4250 034722 006300              DEC     RO              ;ADJUST THE INDEX SO THAT IT WILL
4251 034724 006300              ASL     RO              ;
4252 034724 006300              ASL     RO              ;
4253 034724 006300              ASL     RO              ;
    
```

```

4254 034726 062700 001316      ADD    #SERRTB,RO      ;FORM TABLE POINTER
4255 034732 012037 034742      MOV    (R0)+,2$      ;PICKUP 'ERROR MESSAGE' POINTER
4256 034736 001404              BEQ    3$             ;SKIP TYPEOUT IF NO POINTER
4257 034740 104401              TYPE   ;TYPE THE 'ERROR MESSAGE'
4258 034742 000000      2$: .WORD 0           ;'ERROR MESSAGE' POINTER GOES HERE
4259 034744 104401 001223      TYPE   ,SCLF         ;'CARRIAGE RETURN' & 'LINE FEED'
4260 034750 012037 034760      3$: MOV    (R0)+,4$      ;PICKUP 'DATA HEADER' POINTER
4261 034754 001404              BEQ    5$             ;SKIP TYPEOUT IF 0
4262 034756 104401              TYPE   ;TYPE THE 'DATA HEADER'
4263 034760 000000      4$: .WORD 0           ;'DATA HEADER' POINTER GOES HERE
4264 034762 104401 001223      TYPE   ,SCLF         ;'CARRIAGE RETURN' & 'LINE FEED'
4265 034766 010146      5$: MOV    R1,-(SP)      ;SAVE R1
4266 034770 012001      MOV    (R0)+,R1      ;PICKUP 'DATA TABLE' POINTER
4267 034772 001441      BEQ    12$            ;BR IF NO DATA TO BE TYPED
4268 034774 012000      MOV    (R0)+,R0      ;PICKUP 'DATA FORMAT' POINTER
4269 034776 105710      6$: TSTB   (R0)         ;IS IT FORMAT 0?
4270 035000 001003      BNE    7$             ;BR IF NO
4271
4272      ;*THIS CODE IS FOR OCTAL (16-BIT) FORMAT (DF=0)
4273 035002 013146      MOV    @ (R1)+,-(SP)  ;;SAVE @ (R1)+ FOR TYPEOUT
4274 035004 104402      TYPDC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
4275 035006 000425      BR     11$
4276
4277      ;*THIS CODE IS FOR DECIMAL FORMAT (DF=1)
4278 035010 121027 000001      7$: CMPB   (R0),#1      ;IS IT FORMAT 1?
4279 035014 001003      BNE    8$             ;BRANCH IF NO
4280 035016 013146      MOV    @ (R1)+,-(SP)  ;;SAVE @ (R1)+ FOR TYPEOUT
4281 035020 104405      TYPDS ;GO TYPE--DECIMAL ASCII WITH SIGN
4282 035022 000417      BR     11$
4283
4284      ;*THIS CODE IS FOR BINARY FORMAT (DF=2)
4285 035024 121027 000002      8$: CMPB   (R0),#2      ;IS IT FORMAT 2?
4286 035030 001003      BNE    9$             ;BRANCH IF NO
4287 035032 013146      MOV    @ (R1)+,-(SP)  ;;SAVE @ (R1)+ FOR TYPEOUT
4288 035034 104406      TYPBN ;GO TYPE--BINARY ASCII
4289 035036 000411      BR     11$
4290
4291      ;*THIS CODE IS FOR OCTAL (18-BIT) FORMAT (DF=3)
4292 035040 012146      9$: MOV    (R1)+,-(SP)  ;PUT ADDRESS OF FIRST LOC. ON STACK
4293 035042 004737 040304      JSR    PC,$DB20      ;CONVERT TWO LOCS. TO AN ASCII STRING
4294 035046 062716 000005      ADD    #5,(SP)        ;ONLY NEED 5 CHARACTERS NOT 11
4295 035052 012637 035060      MOV    (SP)+,10$     ;PUT ADDRESS OF ASCII CHARS. AT 10$
4296 035056 104401      TYPE   ;TYPE OCTAL VALUE OF 18-BIT BINARY NO.
4297 035060 000000      10$: .WORD 0
4298
4299 035062 005711      11$: TST    (R1)         ;IS THERE ANOTHER NUMBER?
4300 035064 001404      BEQ    12$            ;BR IF NO
4301 035066 104401 035110      TYPE   ,14$          ;TYPE TWO(2) SPACES
4302 035072 105720      TSTB   (R0)+        ;POINT TO NEW 'DATA FORMAT'
4303 035074 000740      BR     6$             ;LOOP
4304
4305 035076 012601      12$: MOV    (SP)+,R1      ;RESTORE R1
4306 035100 012600      13$: MOV    (SP)+,R0      ;RESTORE R0
4307 035102 104401 001223      TYPE   ,SCLF         ;'CARRIAGE RETURN' & 'LINE FEED'
4308 035106 000207      RTS    PC            ;RETURN
4309 035110 020040      14$: .ASCIZ / /      ;TWO(2) SPACES
    
```

MD-11-CJKDA-B KTF11-AA MMU DIAG MACY11 30A(1052) 11-JUN-79 13:21 H 7 PAGE 85  
CJKDAB.P11 11-JUN-79 13:10 ERROR MESSAGE TYPEOUT ROUTINE

SEQ 0085

4310  
4311

035114

.EVEN

\*\*\*\*\* SUBROUTINES USED BY THIS PROGRAM \*\*\*\*\*

4312  
4313  
4314  
4315  
4316  
4317  
4318  
4319  
4320  
4321  
4322  
4323 035114 033727 177776 000020  
4324 035122 001411  
4325 035124 013746 177776  
4326 035130 011637 001276  
4327  
4328 035134 042716 000020  
4329 035140 012746 035146  
4330 035144 000006  
4331 035146 000207  
4332  
4333  
4334  
4335  
4336  
4337  
4338  
4339  
4340  
4341 035150 033727 001276 000020  
4342 035156 001410  
4343 035160 013746 001276  
4344 035164 012737 000340 001276  
4345 035172 012746 035200  
4346 035176 000006  
4347 035200 000207  
4348  
4349  
4350  
4351  
4352  
4353  
4354  
4355  
4356  
4357  
4358  
4359  
4360 035202 012702 000010  
4361 035206 012701 172300  
4362 035212 012721 177777  
4363 035216 077203  
4364 035220 012702 000010  
4365 035224 012701 172340  
4366 035230 012721 177777  
4367 035234 077203

```
.SBTTL ***** SUBROUTINES USED BY THIS PROGRAM *****  
:*****  
.SBTTL TURN OFF T-BIT AND SAVE CURRENT PSW  
:*****  
: THIS SUBROUTINE IS USED TO TURN OFF THE TRACE TRAP BIT IN THE PSW  
: IF IT IS ON. THE PROCESSOR STATUS IS SAVED IN 'TBITPS' SO THAT  
: THE PSW CAN BE RESTORED TO ITS PREVIOUS CONDITION WHEN CONDITIONS  
: WARRANT T-BIT TRAPPING.  
:*****  
TOFF: BIT PSW,#TBIT ;IS THE T-BIT SET IN THE PSW?  
BEQ 1$ ;EXIT IF NO  
MOV PSW,-(SP) ;PUSH PRESENT PSW ON THE STACK  
MOV (SP),TBITPS ;ALSO SAVE IT IN 'TBITPS' FOR  
;RESTORING LATER  
BIC #TBIT,(SP) ;CLEAR THE T-BIT (BIT 4) IN THE PSW  
MOV #1$,-(SP) ;PUSH PC OF 'RTS' ON STACK  
RTT ;'RETURN' TO 1$ WITH T-BIT OFF  
1$: RTS PC ;RETURN TO PROGRAM  
:*****  
.SBTTL TURN ON T-BIT AND RESTORE PREVIOUS PSW  
:*****  
: THIS SUBROUTINE IS USED TO RESTORE THE PROCESSOR STATUS TO ITS  
: PREVIOUS CONDITION BY RESTORING THE 'T-BIT PSW' SAVED BY THE  
: 'TOFF' SUBROUTINE IN THE 'TBITPS' LOCATION.  
:*****  
TON: BIT TBITPS,#TBIT ;WAS T-BIT ON IN THE PREVIOUS PSW?  
BEQ 1$ ;EXIT IF NO  
MOV TBITPS,-(SP) ;PUSH PREVIOUS PSW ON THE STACK  
MOV #340,TBITPS ;RESET THE 'TBITPS' LOCATION  
MOV #1$,-(SP) ;PUSH PC OF 'RTS' ON STACK  
RTT ;'RETURN' TO 1$ WITH T-BIT RESTORED  
1$: RTS PC ;RETURN TO PROGRAM  
:*****  
.SBTTL SET ALL WRITEABLE BITS IN ALL PAR/PDR'S  
:*****  
: THIS SUBROUTINE IS USED BY THE PAR/PDR DUAL ADDRESSING TEST  
: TO SET ALL WRITEABLE BITS IN ALL KERNEL AND USE PAR'S AND  
: PDR'S TO A 1. THE 'INITIAL STATE' OF HAVING ALL BITS=1 IS  
: USED TO SEE THAT ONLY ONE REGISTER IS CLEARED IN RESPONSE TO  
: A SINGLE PAR OR PDR ADDRESS.  
:*****  
SETREG: MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8  
MOV #KIPDR0,R1 ;LOAD ADDRESS OF FIRST PDR INTO R1  
1$: MOV #-1,(R1)+ ;SET BITS IN KERNEL PDR TO 1  
SOB R2,1$ ;LOOP TO 1$ UNTIL ALL KERNEL PDR'S LOADED  
MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8  
MOV #KIPAR0,R1 ;LOAD ADDRESS OF FIRST PAR INTO R1  
2$: MOV #-1,(R1)+ ;SET BITS IN A KERNEL PAR TO 1  
SOB R2,2$ ;LOOP TO 2$ UNTIL ALL KERNEL PAR'S LOADED
```

SET ALL WRITABLE BITS IN ALL PAR/PDR'S

```

4368 035236 012702 000010      MOV      #10,R2      ;LOAD LOOP COUNTER WITH AN 8
4369 035242 012701 177600      MOV      #UIPDR0,R1  ;LOAD ADDRESS OF FIRST PDR INTO R1
4370 035246 012721 177777      3$:     MOV      #-1,(R1)+ ;SET BITS IN A USER PDR TO 1
4371 035252 077203              SOB      R2,3$      ;LOOP TO 3$ UNTIL ALL USER PDR'S LOADED
4372 035254 012702 000010      MOV      #10,R2      ;LOAD LOOP COUNTER WITH AN 8
4373 035260 012701 177640      MOV      #UIPAR0,R1  ;LOAD ADDRESS OF FIRST PAR IN'O R1
4374 035264 012721 177777      4$:     MOV      #-1,(R1)+ ;SET BITS IN A USER PAR TO 1
4375 035270 077203              SOB      R2,4$      ;LOOP TO 4$ UNTIL ALL USER PAR'S LOADED
4376 035272 000207              RTS          PC      ;RETURN TO TEST
4377
4378
4379
4380
4381
4382
4383
4384
4385
4386
4387
4388 035274
4389 035274 012701 172300      .SBTTL  READ & COMPARE KERNEL & USER PAR/PDR'S
4390 035300 012704 000010      :*****
4391 035304 012705 077416      :*
4392 035310 021105              :*   THIS SUBROUTINE IS USED BY PAR/PDR DUAL ADDRESSING TEST TO
4393 035312 001404              :*   READ ALL THE PAR'S AND PDR'S TO SEE THAT ONLY ONE REGISTER
4394 035314 020100              :*   WAS CLEARED IN RESPONSE TO A SINGLE PAR OR PDR ADDRESS.
4395 035316 001402              :*   ANY FAILURES FOUND BY THE PAR/PDR DUAL ADDRESSING TEST WILL
4396 035320 011102              :*   BE REPORTED BY THIS SUBROUTINE.
4397 035322 104016              :*
4398
4399
4400
4401 035324 062701 000002      :*****
4402 035330 077411              :*   CMPREG:
4403 035332 012701 172340      MOV      #KIPDR0,R1  ;LOAD ADDRESS OF FIRST KERNEL PDR IN R1
4404 035336 012704 000010      MOV      #10,R4      ;LOAD LOOP COUNTER WITH AN 8
4405 035342 012705 177777      MOV      #77416,R5   ;PUT EXPECTED PDR CONTENTS IN R5
4406 035346 021105              1$:     CMP      (R1),R5     ;ARE ALL WRITABLE BITS SET AS EXPECTED?
4407 035350 001404              BEQ      2$          ;BRANCH IF YES
4408 035352 020100              CMP      R1,R0       ;WAS IT THE REG. THAT WAS CLEARED?
4409 035354 001402              BEQ      2$          ;BRANCH IF YES
4410 035356 011102              MOV      (R1),R2     ;SAVE CONTENTS OF IMPROPERLY CLEARED REGISTER
4411 035360 104016              ERROR   16          ;A PDR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
4412
4413
4414
4415 035362 062701 000002      ;FOR TIGHTER SCOPE LOOP
4416 035366 077411              ;REPLACE ERROR CALL WITH
4417 035370 012701 177600      ;AN 'RTS PC' = 000207
4418 035374 012704 000010      2$:     ADD      #2,R1       ;FORM NEXT ADDRESS
4419 035400 012705 077416      SOB      R4,1$      ;LOOP TO 1$ UNTIL ALL KERNEL PDR'S CHECKED
4420 035404 021105              MOV      #KIPAR0,R1  ;LOAD ADDRESS OF FIRST KERNEL PAR IN R1
4421 035406 001404              MOV      #10,R4      ;LOAD LOOP COUNTER WITH AN 8
4422 035410 020100              MOV      #177777,R5  ;PUT EXPECTED PAR CONTENTS IN R5
4423 035412 001402              3$:     CMP      (R1),R5     ;ARE ALL WRITABLE BITS SET AS EXPECTED?
4424
4425
4426
4427
4428
4429
4430
4431
4432
4433
4434
4435 035362 062701 000002      4$:     ADD      #2,R1       ;FORM NEXT ADDRESS
4436 035366 077411              SOB      R4,3$      ;LOOP TO 3$ UNTIL ALL KERNEL PAR'S CHECKED
4437 035370 012701 177600      MOV      #UIPDR0,R1  ;LOAD ADDRESS OF FIRST USER PDR IN R1
4438 035374 012704 000010      MOV      #10,R4      ;LOAD LOOP COUNTER WITH AN 8
4439 035400 012705 077416      MOV      #77416,R5   ;PUT EXPECTED PDR CONTENTS IN R5
4440 035404 021105              5$:     CMP      (R1),R5     ;ARE ALL WRITABLE BITS SET AS EXPECTED?
4441 035406 001404              BEQ      6$          ;BRANCH IF YES
4442 035410 020100              CMP      R1,R0       ;WAS IT THE REG. THAT WAS CLEARED?
4443 035412 001402              BEQ      6$          ;BRANCH IF YES
    
```

```

4424 035414 011102          MOV    (R1),R2          ;SAVE CONTENTS OF IMPROPERLY CLEARED REGISTER
4425 035416 104016          ERROR  16              ;A PDR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
4426                                     ;FOR TIGHTER SCOPE LOOP
4427                                     ;REPLACE ERROR CALL WITH
4428                                     ;AN 'RTS PC' = 000207
4429 035420 062701 00000?    6$:   ADD    #2,R1          ;FORM NEXT ADDRESS
4430 035424 077411          SOB    R4,5$           ;LOOP TO 5$ UNTIL ALL USER PDR'S CHECKED
4431 035426 012701 177640    MOV    #UIPAR0,R1      ;LOAD ADDRESS OF FIRST USER PAR IN R1
4432 035432 012704 000010    MOV    #10,R4          ;LOAD LOOP COUNTER WITH AN 8
4433 035436 012705 17777?    MOV    #177777,R5      ;PUT EXPECTED PAR CONTENTS IN R5
4434 035442 021105          7$:   CMP    (R1),R5        ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
4435 035444 001404          BEQ    8$              ;BRANCH IF YES
4436 035446 020100          CMP    R1,R0           ;WAS IT THE REG. THAT WAS CLEARED?
4437 035450 001402          BEQ    8$              ;BRANCH IF YES
4438 035452 011102          MOV    (R1),R2          ;SAVE CONTENTS OF IMPROPERLY CLEARED REGISTER
4439 035454 104016          ERROR  16              ;A PAR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
4440                                     ;FOR TIGHTER SCOPE LOOP
4441                                     ;REPLACE ERROR CALL WITH
4442                                     ;AN 'RTS PC' = 000207
4443 035456 062701 00000?    8$:   ADD    #2,P1          ;FORM NEXT ADDRESS
4444 035462 077411          SOB    R4,7$           ;LOOP TO 7$ UNTIL ALL USER PAR'S CHECKED
4445 035464 000207          RTS    PC              ;RETURN TO TEST
    
```

SBTTL CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS

\*\*\*\*\*

THIS SUBROUTINE IS USED TO FORM AN 18-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 <11:00> TO GIVE PBA <17:6>. BITS <17:16> 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 OR USER PAR/PDR'S. THE ROUTINE IS ENTERED WITH LOC. 'VIRT1' CONTAINING THE 16-BIT VIRTUAL ADDRESS.

\*\*\*\*\*

FORMPA:

```

4464 035466 010046          MOV    R0,-(SP)        ;;PUSH R0 ON STACK
4465 035466 010246          MOV    R2,-(SP)        ;;PUSH R2 ON STACK
4466 035470 010246          MOV    #KIPAR0,R2      ;LOAD ADDRESS OF FIRST KERNEL PAR IN R2
4467 035472 012702 172340    BIT    #140000,PSW     ;IN USER MODE?
4468 035476 032737 140000 177776  BEQ    1$              ;BRANCH IF NO
4469 035504 001402          MOV    #UIPAR0,R2      ;LOAD ADDRESS OF FIRST USER PAR IN R2
4470 035506 012702 177640    1$:   MOV    VIRT1,R0        ;LOAD VIRTUAL ADDR. (VBA) INTO R0
4471 035512 013700 001306    ASH    #-14,R0         ;GET BITS <15:13> DOWN TO BITS <3:1>
4472 035516 072027 177764    BIC    #177761,R0      ;MASK OF ALL BITS BUT BITS <3:1>
4473 035522 042700 177761    ADD    R0,R2           ;ADD OFSET TO BASE PAR ADDRESS
4474 035526 060002          MOV    (R2),R0         ;GET BITS <11:00> FROM APPROPRIATE PAR
4475 035530 011200          MOV    R0,R2           ;COPY PAR BITS <11:00> INTO R2
4476 035532 010002          MOV    VIRT1,PBALO     ;PUT VIRTUAL ADDR. IN LOC. 'PBALO'
4477 035534 013737 001306 001312  BIC    #160000,PBALU   ;CLEAR OFF BITS <15:13> OF ORIGINAL VBA
4478 035542 042737 160000 001312  ASH    #-12,R2         ;GET PAR <11:00> DOWN TO BITS <1:0> OF R2
4479 035550 072227 177766
    
```

4480	035554	042702	177774	BIC	#177774,R2	;CLEAR OFF ALL BITS BUT BITS <1:0>
4481	035560	072027	000006	ASH	#6,R0	;SHIFT PAR<9:0> TC <15:6> OF R0
4482	035564	042700	000077	BIC	#77,R0	;CLEAR BITS <5:0> OF R0
4483	035570	060037	001312	ADD	R0,PBALO	;IN EFFECT, ADD VBA<12:0> TO PAR<9:0>
4484						;(PAR<9:0> IN BITS <15:6> OF R0)
4485	035574	005502		ADC	R2	;ADD ANY CARRY TO R2
4486	035576	010237	001314	MOV	R2,PBAHI	;PUT BITS <17:16> OF PHYSICAL ADDR. IN PBAHI
4487	035602	012602		MOV	(SP)+,R2	;;POP STACK INTO R2
4488	035604	012600		MOV	(SP)+,R0	;;POP STACK INTO R0
4489	035606	000207		RTS	PC	;RETURN TO PROGRAM
4490						
4491						

```

4492      .SBTTL  TTY INPUT ROUTINE
4493
4494      ::*****
4495      .ENABL  LSB
4496
4497      ::*****
4498      ::*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
4499      ::*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
4500      ::*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
4501      ::*WHEN OPERATING IN TTY FLAG MODE.
4502 035610 022737 000176 001140 $CKSWR: CMP      #SWREG,SWR      :: IS THE SOFT-SWR SELECTED?
4503 035616 001114          BNE      15$              :: BRANCH IF NO
4504 035620 105777 143320          TSTB   @STKS              :: CHAR THERE?
4505 035624 100111          BPL      15$              :: IF NO, DON'T WAIT AROUND
4506 035626 117746 143314          MOVB   @STKB,-(SP)         :: SAVE THE CHAR
4507 035632 042716 177600          BIC    #^C177,(SP)        :: STRIP-OFF THE ASCII
4508 035636 022726 000007          CMP    #7,(SP)+          :: IS IT A CONTROL G?
4509 035642 001102          BNE      15$              :: NO, RETURN TO USER
4510 035644 123727 001134 000001          CMPB   $AUTOB,#1         :: ARE WE RUNNING IN AUTO-MODE?
4511 035652 001476          BEQ     15$              :: BRANCH IF YES
4512
4513 035654 104401 036552          $GTSWR: TYPE   , $CNTLG      :: ECHO THE CONTROL-G (^G)
4514 035660 104401 036557          TYPE   , $MSWR          :: TYPE CURRENT CONTENTS
4515 035664 013746 000176          MOV     SWREG,-(SP)      :: SAVE SWREG FOR TYPEOUT
4516 035670 104402          TYPOC          :: GO TYPE--OCTAL ASCII(ALL DIGITS)
4517 035672 104401 036570          TYPE   , $MNEW          :: PROMPT FOR NEW SWR
4518 035676 005046          19$:  CLR     -(SP)        :: CLEAR COUNTER
4519 035700 005046          CLR     -(SP)        :: THE NEW SWR
4520 035702 105777 143236          7$:  TSTB   @STKS          :: CHAR THERE?
4521 035706 100375          BPL     7$            :: IF NOT TRY AGAIN
4522
4523 035710 117746 143232          MOVB   @STKB,-(SP)      :: PICK UP CHAR
4524 035714 042716 177600          BIC    #^C177,(SP)      :: MAKE IT 7-BIT ASCII
4525
4526 035720 021627 000003          CMP    (SP),#3          :: IS IT A CONTROL-C?
4527 035724 001015          BNE     9$              :: BRANCH IF NOT
4528 035726 104401 036540          TYPE   , $CNTLC         :: YES, ECHO CONTROL-C (^C)
4529 035732 062706 000006          ADD    #6,SP            :: CLEAN UP STACK
4530 035736 123727 001135 000001          CMPB   $INTAG,#1        :: REENABLE TTY KEYBOARD INTERRUPTS?
4531 035744 001003          BNE     8$              :: BRANCH IF NO
4532 035746 012777 000100 143170          MOV    #100,@STKS       :: ALLOW TTY KEYBOARD INTERRUPTS
4533 035754 000137 036602          8$:  JMP     CNTRLC        :: CONTROL-C RESTART
4534
4535
4536 035760 021627 000025          9$:  CMP    (SP),#25        :: IS IT A CONTROL-U?
4537 035764 001005          BNE     10$             :: BRANCH IF NOT
4538 035766 104401 036545          TYPE   , $CNTLU         :: YES, ECHO CONTROL-U (^U)
4539 035772 062706 000006          20$: ADD    #6,SP            :: IGNORE PREVIOUS INPUT
4540 035776 000737          BR     19$             :: LET'S TRY IT AGAIN
4541
4542
4543 036000 021627 000015          10$: CMP    (SP),#15         :: IS IT A <CR>?
4544 036004 001022          BNE     16$             :: BRANCH IF NO
4545 036006 005766 000004          TST    4(SP)            :: YES, IS IT THE FIRST CHAR?
4546 036012 001403          BEQ     11$             :: BRANCH IF YES
4547 036014 016677 000002 143116          MOV    2(SP),@SWR       :: SAVE NEW SWR
    
```



```
4548 036022 062706 000006 11$: ADD #6,SP ;;CLEAR UP STACK
4549 036026 104401 001223 14$: TYPE , $CRLF ;;ECHO <CR> AND <LF>
4550 036032 123727 001135 000001 CMPB $INTAG,#1 ;;RE-ENABLE TTY KBD INTERRUPTS?
4551 036040 001003 BNE 15$ ;;BRANCH IF NOT
4552 036042 012777 000100 143074 MOV #100,@$TKS ;;RE-ENABLE TTY KBD INTERRUPTS
4553 036050 000002 15$: RTI ;;RETURN
4554 036052 004737 037144 16$: JSR PC,$TYPEC ;;ECHO CHAR
4555 036056 021627 000060 CMP (SP),#60 ;;CHAR < 0?
4556 036062 002420 BLT 18$ ;;BRANCH IF YES
4557 036064 021627 000067 CMP (SP),#67 ;;CHAR > 7?
4558 036070 003015 BGT 18$ ;;BRANCH IF YES
4559 036072 042726 000060 BIC #60,(SP)+ ;;STRIP-OFF ASCII
4560 036076 005766 000002 TST 2(SP) ;;IS THIS THE FIRST CHAR
4561 036102 001403 BEQ 17$ ;;BRANCH IF YES
4562 036104 006316 ASL (SP) ;;NO, SHIFT PRESENT
4563 036106 006316 ASL (SP) ;; CHAR OVER TO MAKE
4564 036110 006316 ASL (SP) ;; ROOM FOR NEW ONE.
4565 036112 005266 000002 17$: INC 2(SP) ;;KEEP COUNT OF CHAR
4566 036116 056616 177776 BIS -2(SP),(SP) ;;SET IN NEW CHAR
4567 036122 000667 BR 7$ ;;GET THE NEXT ONE
4568 036124 104401 001222 18$: TYPE , $QUES ;;TYPE ?<CR><LF>
4569 036130 000720 BR 20$ ;;SIMULATE CONTROL-U
4570 .DSABL LSB
4571
4572
4573 ;;*****
4574 ;;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
4575 ;;*CALL:
4576 ;;* RDCHR ;;INPUT A SINGLE CHARACTER FROM THE TTY
4577 ;;* RETURN HERE ;;CHARACTER IS ON THE STACK
4578 ;;* ;;WITH PARITY BIT STRIPPED OFF
4579 ;;
4580
4581 036132 011646 $RDCHR: MOV (SP),-(SP) ;;PUSH DOWN THE PC
4582 036134 016666 000004 000002 MOV 4(SP),2(SP) ;;SAVE THE PS
4583 036142 105777 142776 1$: TSTB @$TKS ;;WAIT FOR
4584 036146 100375 BPL 1$ ;;A CHARACTER
4585 036150 117766 142772 000004 MOVB @$TKB,4(SP) ;;READ THE TTY
4586 036156 042766 177600 000004 BIC #^C<177>,4(SP) ;;GET RID OF JUNK IF ANY
4587 036164 026627 000004 000023 CMP 4(SP),#23 ;;IS IT A CONTROL-S?
4588 036172 001013 BNE 3$ ;;BRANCH IF NO
4589 036174 105777 142744 2$: TSTB @$TKS ;;WAIT FOR A CHARACTER
4590 036200 100375 BPL 2$ ;;LOOP UNTIL ITS THERE
4591 036202 117746 142740 MOVB @$TKB,-(SP) ;;GET CHARACTER
4592 036206 042716 177600 BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII
4593 036212 022627 000021 CMP (SP)+,#21 ;;IS IT A CONTROL-Q?
4594 036216 001366 BNE 2$ ;;IF NOT DISCARD IT
4595 036220 000750 BR 1$ ;;YES, RESUME
4596 036222 026627 000004 000140 3$: CMP 4(SP),#140 ;;IS IT UPPER CASE?
4597 036230 002407 BLT 4$ ;;BRANCH IF YES
4598 036232 026627 000004 000175 CMP 4(SP),#175 ;;IS IT A SPECIAL CHAR?
4599 036240 003003 BGT 4$ ;;BRANCH IF YES
4600 036242 042766 000040 000004 BIC #40,4(SP) ;;MAKE IT UPPER CASE
4601 036250 000002 4$: RTI ;;GO BACK TO USER
4602
4603 ;;*****
;;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
```

```

4604          : *CALL :
4605          : *      RDLIN          : INPUT A STRING FROM THE TTY
4606          : *      RETURN HERE   : ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
4607          : *                               : TERMINATOR WILL BE A BYTE OF ALL 0'S
4608
4609 036252 010346 $RDLIN: MOV      R3,-(SP)          : SAVE R3
4610 036254 005046      CLR      -(SP)          : CLEAR THE RUBOUT KEY
4611 036256 012703 036530 1$:      MOV      #$TTYIN,R3      : GET ADDRESS
4612 036262 022703 036540 2$:      CMP      #$TTYIN+8.,R3  : BUFFER FULL?
4613 036266 101467      BLOS     4$              : BR IF YES
4614 036270 104411      RDCHR   : GO READ ONE CHARACTER FROM THE TTY
4615 036272 112613      MOVB    (SP)+,(R3)      : GET CHARACTER
4616 036274 122713 000003      CMPB    #3,(R3)      : IS IT A CONTROL-C?
4617 036300 001006      BNE     10$             : BRANCH IF NO
4618 036302 104401 036540      TYPE    , $CNTLC      : TYPE A CONTROL-C (^C)
4619 036306 005726      TST     (SP)+          : CLEAN RUBOUT KEY OFF OF THE STACK
4620 036310 012603      MOV      (SP)+,R3      : RESTORE R3
4621 036312 000137 036602      JMP     CNTRLC      : GOTO CONTROL-C RESTART
4622 036316 122713 000177 10$:    CMPB    #177,(R3)     : IS IT A RUBOUT
4623 036322 001022      BNE     5$              : BR IF NO
4624 036324 005716      TST     (SP)          : IS THIS THE FIRST RUBOUT?
4625 036326 001007      BNE     6$              : BR IF NO
4626 036330 112737 000134 036526      MOVB    #' \ ,9$      : TYPE A BACK SLASH
4627 036336 104401 036526      TYPE    ,9$
4628 036342 012716 177777      MOV      #-1,(SP)      : SET THE RUBOUT KEY
4629 036346 005303 6$:      DEC     R3              : BACKUP BY ONE
4630 036350 020327 036530      CMP     R3,$TTYIN     : STACK EMPTY?
4631 036354 103434      BLO     4$              : BR IF YES
4632 036356 111337 036526      MOVB    (R3),9$      : SETUP TO TYPEOUT THE DELETED CHAR.
4633 036362 104401 036526      TYPE    ,9$
4634 036366 000735      BR     2$              : GO TYPE
4635 036370 005716 5$:      TST     (SP)          : GO READ ANOTHER CHAR.
4636 036372 001406      BEQ    7$              : RUBOUT KEY SET?
4637 036374 112737 000134 036526      MOVB    #' \ ,9$      : TYPE A BACK SLASH
4638 036402 104401 036526      TYPE    ,9$
4639 036406 005016      CLR     (SP)          : CLEAR THE RUBOUT KEY
4640 036410 122713 000025 7$:    CMPB    #25,(R3)     : IS CHARACTER A CTRL U?
4641 036414 001003      BNE     8$              : BR IF NO
4642 036416 104401 036545      TYPE    , $CNTLU      : TYPE A CONTROL 'U'
4643 036422 000715      BR     1$              : GO START OVER
4644 036424 122713 000022 8$:    CMPB    #22,(R3)     : IS CHARACTER A '^R'?
4645 036430 001011      BNE     3$              : BRANCH IF NO
4646 036432 105013      CLRB   (R3)          : CLEAR THE CHARACTER
4647 036434 104401 001223      TYPE    , $CRLF      : TYPE A 'CR' & 'LF'
4648 036440 104401 036530      TYPE    , $TTYIN     : TYPE THE INPUT STRING
4649 036444 000706      BR     2$              : GO PICKUP ANOTHER CHACTER
4650 036446 104401 001222 4$:    TYPE    , $QUES      : TYPE A '?'
4651 036452 000701      BR     1$              : CLEAR THE BUFFER AND LOOP
4652 036454 111337 036526 3$:    MOVB    (R3),9$      : ECHO THE CHARACTER
4653 036460 104401 036526      TYPE    ,9$
4654 036464 122723 000015      CMPB    #15,(R3)+    : CHECK FOR RETURN
4655 036470 001274      BNE     2$              : LOOP IF NOT RETURN
4656 036472 105063 177777      CLRB   -1(R3)        : CLEAR RETURN (THE 15)
4657 036476 104401 001224      TYPE    , $LF        : TYPE A LINE FEED
4658 036502 005726      TST     (SP)+          : CLEAN RUBOUT KEY FROM THE STACK
4659 036504 012603      MOV      (SP)+,R3    : RESTORE R3
    
```

```

4660 036506 011646          MOV      (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
4661 036510 016666 000004 000002  MOV      4(SP),2(SP)    ;;      FIRST ASCII CHARACTER ON IT
4662 036516 012766 036530 000004  MOV      #$TTYIN,4(SP)
4663 036524 000002          RTI                      ;;RETURN
4664 036526      000          9$: .BYTE      0          ;;STORAGE FOR ASCII CHAR. TO TYPE
4665 036527      000          .BYTE      0          ;;TERMINATOR
4666 036530 000010          $TTYIN: .BLKB     8.      ;;RESERVE 8 BYTES FOR TTY INPUT
4667 036540 041536 005015      000  $CNTLC: .ASCIZ  /^C/<15><12>  ;;CONTROL 'C'
4668 036545      136 006525 000012  $CNTLU: .ASCIZ  /^U/<15><12>  ;;CONTROL 'U'
4669 036552 043536 005015      000  $CNTLG: .ASCIZ  /^G/<15><12>  ;;CONTROL 'G'
4670 036557      015 051412 051127  $MSWR: .ASCIZ  <15><12>/SWR = /
4671 036564 036440 000040          $MNEW: .ASCIZ  / NEW = /
4672 036570 020040 042516 020127
4673 036576 020075      000
4674      036602          .EVEN
4675
4676          .SBTTL CONTROL-C SERVICING ROUTINE
4677
4678          ;*      THE FOLLOWING CODE IS EXECUTED WHEN A CONTROL-C HAS
4679          ;*      BEEN TYPED INSTEAD OF A NEW SWITCH REG. VALUE.
    
```

```

4680      ;*      (IN OTHER WORDS, AFTER A CONTROL-G WAS TYPED).
4681      ;*      A NEW SWITCH REG. VALUE WILL BE ASKED FOR,
4682      ;*      THE TEST NUMBER AND PASS NUMBER WILL BE TYPED,
4683      ;*      AND THEN THE PROGRAM WILL GO TO 'END-OF-PASS' AND CONTINUE
4684
4685 036602 013737 001234 001210 CNTRLC: MOV      $PASS,$TMP5      ;GET THE VALUE OF '$PASS'
4686 036610 005237 001210          !INC      $TMP5          ;FORM CURRENT PASS NO.
4687 036614 104401 036661          TYPE      ,CMMSG        ;TYPE THE TEST STOPS MESSAGE
4688 036620 113737 001102 036654 MOVVB    $TSTNM,1$      ;SAVE THE TEST NUMBER
4689 036626 013746 036654          MOV      1$,-(SP)      ;SAVE 1$ FOR TYPEOUT
4690 036632 104402          TYPOC         ;GO TYPE--OCTAL ASCII(ALL DIGITS)
4691 036634 104401 036656          TYPE      ,2$          ;TYPE 2 SPACES
4692 036640 013746 001210          MOV      $TMP5,-(SP)    ;SAVE $TMP5 FOR TYPEOUT
4693 036644 104405          TYPDS         ;GO TYPE--DECIMAL ASCII WITH SIGN
4694 036646 104407          GTSWR         ;ASK FOR NEW SWR VALUE
4695 036650 000137 033662          JMP      $EOP+2        ;CONTINUE AT $EOP+2
4696 036654 000000          1$:      .WORD      0          ;BUFFER FOR TEST NUMBER
4697 036656 020040          2$:      .ASCIZ    / /          ;TWO SPACES AND THE STOP MESSAGE
4698 036661          112 046525 044520  CMSG:     .ASCIZ    /JUMPING TO END-OF-PASS/<15><12>
4699 036666 043516 052040 020117
4700 036674 047105 026504 043117
4701 036702 050055 051501 006523
4702 036710          012
4703 036711          124 051505 047124      .ASCIZ /TESTNO PASSNO/<15><12>
4704 036716 020117 050040 051501
4705 036724 047123 006517 000012
4706
4707      .EVEN
4708      .SBTTL TYPE ROUTINE
4709
4710      ;*****
4711      ;ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
4712      ;THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
4713      ;*NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
4714      ;*NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
4715      ;*NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
4716      ;*
4717      ;*CALL:
4718      ;*1) USING A TRAP INSTRUCTION
4719      ;*      TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
4720      ;*OR
4721      ;*      TYPE
4722      ;*      MESADR
4723      ;*
4724 036732 105737 001157      $TYPE:  TSTB    $TPFLG      ;;IS THERE A TERMINAL?
4725 036736 100002          BPL      1$              ;;BR IF YES
4726 036740 000000          HALT         ;;HALT HERE IF NO TERMINAL
4727 036742 000430          BR      3$              ;;LEAVE
4728 036744 010046          1$:      MOV      RO,-(SP)    ;SAVE RO
4729 036746 017600 000002          MOV      @2(SP),RO      ;GET ADDRESS OF ASCIZ STRING
4730 036752 122737 000001 001246          CMPB    #APTENV,$ENV    ;RUNNING IN APT MODE
4731 036760 001011          BNE      62$            ;NO,GO CHECK FOR APT CONSOLE
4732 036762 132737 000100 001247          BITB    #APTPOOL,$ENVM ;SPOOL MESSAGE TO APT
4733 036770 001405          BEQ      62$            ;NO,GO CHECK FOR CONSOLE
4734 036772 010037 037002          MOV      RO,61$         ;SETUP MESSAGE ADDRESS FOR APT
4735 036776 004737 037222          JSR      PC,$ATY3      ;SPOOL MESSAGE TO APT
    
```

```

4736 037002 000000 61$: .WORD 0 ;;MESSAGE ADDRESS
4737 037004 132737 000040 001247 62$: BITB #APTC SUP,$ENV M ;;APT CONSOLE SUPPRESSED
4738 037012 001003 BNE 60$ ;;YES,SKIP TYPE OUT
4739 037014 112046 2$: MOV B (R0)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
4740 037016 001005 BNE 4$ ;;BR IF IT ISN'T THE TERMINATOR
4741 037020 005726 TST (SP)+ ;;IF TERMINATOR POP IT OFF THE STACK
4742 037022 012600 60$: MOV (SP)+,R0 ;;RESTORE R0
4743 037024 062716 000002 3$: ADD #2,(SP) ;;ADJUST RETURN PC
4744 037030 000002 RTI ;;RETURN
4745 037032 122716 000011 4$: CMPB #HT,(SP) ;;BRANCH IF <HT>
4746 037036 001430 BEQ 8$
4747 037040 122716 000200 CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
4748 037044 001006 BNE 5$
4749 037046 005726 TST (SP)+ ;;POP <CR><LF> EQUIV
4750 037050 104401 TYPE ;;TYPE A CR AND LF
4751 037052 001223 $CRLF
4752 037054 105037 037210 CLRB $CHARCNT ;;CLEAR CHARACTER COUNT
4753 037060 000755 BR 2$ ;;GET NEXT CHARACTER
4754 037062 004737 037144 5$: JSR PC,$TYPE C ;;GO TYPE THIS CHARACTER
4755 037066 123726 001156 6$: CMPB $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
4756 037072 001350 BNE 2$ ;;IF NO GO GET NEXT CHAR.
4757 037074 013746 001154 MOV $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
4758 ;;AND THE NULL CHAR.
4759 037100 105366 000001 7$: DECB 1(SP) ;;DOES A NULL NEED TO BE TYPED?
4760 037104 002770 BLT 6$ ;;BR IF NO--GO POP THE NULL OFF OF STACK
4761 037106 004737 037144 JSR PC,$TYPE C ;;GO TYPE A NULL
4762 037112 105337 037210 DECB $CHARCNT ;;DO NOT COUNT AS A COUNT
4763 037116 000770 BR 7$ ;;LOOP

```

;HORIZONTAL TAB PROCESSOR

```

4764
4765
4766
4767 037120 112716 000040 8$: MOV B #' ,(SP) ;;REPLACE TAB WITH SPACE
4768 037124 004737 037144 9$: JSR PC,$TYPE C ;;TYPE A SPACE
4769 037130 132737 000007 037210 BITB #7,$CHARCNT ;;BRANCH IF NOT AT
4770 037136 001372 BNE 9$ ;;TAB STOP
4771 037140 005726 TST (SP)+ ;;POP SPACE OFF STACK
4772 037142 000724 BR 2$ ;;GET NEXT CHARACTER
4773 037144 105777 142000 $TYPE C: TSTB @ $TPS ;;WAIT UNTIL PRINTER IS READY
4774 037150 100375 BPL $TYPE C
4775 037152 116677 000002 141772 MOV B 2(SP),@$TPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
4776 037160 122766 000015 000002 CMPB #CR,2(SP) ;;IS CHARACTER A CARRIAGE RETURN?
4777 037166 001003 BNE 1$ ;;BRANCH IF NO
4778 037170 105037 037210 CLRB $CHARCNT ;;YES--CLEAR CHARACTER COUNT
4779 037174 000406 BR $TYPE X ;;EXIT
4780 037176 122766 000012 000002 1$: CMPB #LF,2(SP) ;;IS CHARACTER A LINE FEED?
4781 037204 001402 BEQ $TYPE X ;;BRANCH IF YES
4782 037206 105227 INCB (PC)+ ;;COUNT THE CHARACTER
4783 037210 000000 $CHARCNT: .WORD 0 ;;CHARACTER COUNT STORAGE
4784 037212 000207 $TYPE X: RTS PC
4785
4786
4787
4788

```

.SBTTL APT COMMUNICATIONS ROUTINE

```

4789 037214 112737 000001 037460 ;;*****
4790 037222 112737 000001 037456 $ATY1: MOV B #1,$FFLG ;;TO REPORT FATAL ERROR
4791 037230 000403 $ATY3: MOV B #1,$MFLG ;;TO TYPE A MESSAGE
BR $ATYC

```

```

4792 037232 112737 000001 037460 $ATY4: MOVB #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
4793 037240 $ATYC:
4794 037240 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
4795 037242 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
4796 037244 105737 037456 TSTB $MFLG ;;SHOULD TYPE A MESSAGE?
4797 037250 001450 BEQ 5$ ;;IF NOT: BR
4798 037252 122737 000001 001246 CMPB #AFTENV,$ENV ;;OPERATING UNDER APT?
4799 037260 001031 BNE 3$ ;;IF NOT: BR
4800 037262 132737 000100 001247 BITB #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
4801 037270 001425 BEQ 3$ ;;IF NOT: BR
4802 037272 017600 000004 MOV @4(SP),R0 ;;GET MESSAGE ADDR.
4803 037276 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
4804 037304 005737 001226 1$: TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
4805 037310 001375 BNE 1$ ;;IF NOT: WAIT
4806 037312 010037 001242 MOV R0,$MSGAD ;;PUT ADDR IN MAILBOX
4807 037316 105720 2$: TSTB (R0)+ ;;FIND END OF MESSAGE
4808 037320 001376 BNE 2$
4809 037322 163700 001242 SUB $MSGAD,R0 ;;SUB START OF MESSAGE
4810 037326 006200 ASR R0 ;;GET MESSAGE LNTH IN WORDS
4811 037330 010037 001244 MOV R0,$MSGLGT ;;PUT LENGTH IN MAILBOX
4812 037334 012737 000004 001226 MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
4813 037342 000413 BR 5$
4814 037344 017637 000004 037370 3$: MOV @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE
4815 037352 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS
4816 037360 013746 177776 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
4817 037364 004737 036732 JSR PC,$TYPE ;;CALL TYPE MACRO
4818 037370 000000 4$: .WORD 0
4819 037372 5$:
4820 037372 105737 037460 10$: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?
4821 037376 001416 BEQ 12$ ;;IF NOT: BR
4822 037400 005737 001246 TST $ENV ;;RUNNING UNDER APT?
4823 037404 001413 BEQ 12$ ;;IF NOT: BR
4824 037406 005737 001226 11$: TST $MSGTYPE ;;FINISHED LAST MESSAGE?
4825 037412 001375 BNE 11$ ;;IF NOT: WAIT
4826 037414 017637 000004 001230 MOV @4(SP),$FATAL ;;GET ERROR #
4827 037422 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
4828 037430 005237 001226 INC $MSGTYPE ;;TELL APT TO TAKE ERROR
4829 037434 105037 037460 12$: CLRB $FFLG ;;CLEAR FATAL FLAG
4830 037440 105037 037457 CLRB $LFLG ;;CLEAR LOG FLAG
4831 037444 105037 037456 CLRB $MFLG ;;CLEAR MESSAGE FLAG
4832 037450 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
4833 037452 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
4834 037454 000207 RTS PC ;;RETURN
4835 037456 000 $MFLG: .BYTE 0 ;;MESSG. FLAG
4836 037457 000 $LFLG: .BYTE 0 ;;LOG FLAG
4837 037460 000 $FFLG: .BYTE 0 ;;FATAL FLAG
4838 037462 .EVEN
4839 000200 AFTSIZE=200
4840 000001 APTENV=001
4841 000100 APTSPOOL=100
4842 000040 APTCSUP=040
4843 .SBTTL BINARY TO ASCII AND TYPE ROUTINE
4844
4845 ;;*****
4846 ;;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
4847 ;;*BINARY-ASCII NUMBER AND TYPE IT.
    
```

```

4848      ;*CALL:
4849      ;*      MOV      NUMBER,-(SP)      ;;NUMBER TO BE TYPED
4850      ;*      TYPBN      ;;TYPE IT
4851
4852      037462 010146      $TYPBN: MOV      R1,-(SP)      ;;SAVE R1 ON THE STACK
4853      037464 016607 000006      MOV      6(SP),R1      ;;GET THE INPUT NUMBER
4854      037470 000261      SEC      ;;SET 'C' SO CAN KEEP TRACK OF THE NUMBER OF BITS
4855      037472 112737 000060 037534 1$: MOVVB   #'0,$BIN      ;;SET CHARACTER TO AN ASCII '0'.
4856      037500 006101      ROL      R1      ;;GET THIS BIT
4857      037502 001406      BEQ      2$      ;;DONE?
4858      037504 105537 037534      ADCB    $BIN      ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
4859      037510 104401 037534      TYPE    , $BIN      ;;GO TYPE THIS BIT
4860      037514 000241      CLC      ;;CLEAR 'C' SO CAN KEEP TRACK OF BITS
4861      037516 000765      BR      1$      ;;GO DO THE NEXT BIT
4862      037520 012601      2$: MOV      (SP)+,R1      ;;POP THE STACK INTO R1
4863      037522 016666 000002 000004      MOV      2(SP),4(SP)      ;;ADJUST THE STACK
4864      037530 012616      MOV      (SP)+,(SP)
4865      037532 000002      RTI      ;;RETURN TO USER
4866      037534      000      000      $BIN: .BYTE 0,0      ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
4867      ;SBITL BINARY TO OCTAL (ASCII) AND TYPE
4868
4869      ;*****
4870      ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
4871      ;*OCTAL (ASCII) NUMBER AND TYPE IT.
4872      ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
4873      ;*CALL:
4874      ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
4875      ;*      TYPOS      ;;CALL FOR TYPEOUT
4876      ;*      .BYTE    N      ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
4877      ;*      .BYTE    M      ;;M=1 OR 0
4878      ;*      ;;1=TYPE LEADING ZEROS
4879      ;*      ;;0=SUPPRESS LEADING ZEROS
4880
4881      ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
4882      ;*$TYPOS OR $TYPOC
4883      ;*CALL:
4884      ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
4885      ;*      TYPON      ;;CALL FOR TYPEOUT
4886
4887      ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
4888      ;*CALL:
4889      ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
4890      ;*      TYPOC      ;;CALL FOR TYPEOUT
4891
4892      037536 017646 000000      $TYPOS: MOV      @ (SP),-(SP)      ;;PICKUP THE MODE
4893      037542 116637 000001 037761      MOVVB   1(SP),$OFILL      ;;LOAD ZERO FILL SWITCH
4894      037550 112637 037763      MOVVB   (SP)+,$OMODE+1      ;;NUMBER OF DIGITS TO TYPE
4895      037554 062716 000002      ADD     #2,(SP)      ;;ADJUST RETURN ADDRESS
4896      037560 000406      BR      $TYPON
4897      037562 112737 000001 037761      $TYPOC: MOVVB   #1,$OFILL      ;;SET THE ZERO FILL SWITCH
4898      037570 112737 000006 037763      MOVVB   #6,$OMODE+1      ;;SET FOR SIX(6) DIGITS
4899      037576 112737 000005 037760      $TYPON: MOVVB   #5,$OCNT      ;;SET THE ITERATION COUNT
4900      037604 010346      MOV     R3,-(SP)      ;;SAVE R3
4901      037606 010446      MOV     R4,-(SP)      ;;SAVE R4
4902      037610 010546      MOV     R5,-(SP)      ;;SAVE R5
4903      037612 113704 037763      MOVVB   $OMODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE

```

```

4904 037616 005404          NEG      R4
4905 037620 062704 000006  ADD      #6,R4          ;;SUBTRACT IT FOR MAX. ALLOWED
4906 037624 110437 037762  MOVVB   R4,$OMODE     ;;SAVE IT FOR USE
4907 037630 113704 037761  MOVVB   $OFILL,R4     ;;GET THE ZERO FILL SWITCH
4908 037634 016605 000012  MOV     12(SP),R5     ;;PICKUP THE INPUT NUMBER
4909 037640 005003          CLR      R3          ;;CLEAR THE OUTPUT WORD
4910 037642 006105          1$:    ROL      R5     ;;ROTATE MSB INTO 'C'
4911 037644 000404          BR       3$          ;;GO DO MSB
4912 037646 006105          2$:    ROL      R5     ;;FORM THIS DIGIT
4913 037650 006105          ROL      R5
4914 037652 006105          ROL      R5
4915 037654 010503          MOV     R5,R3
4916 037656 006103          3$:    ROL      R3     ;;GET LSB OF THIS DIGIT
4917 037660 105337 037762  DECB   $OMODE         ;;TYPE THIS DIGIT?
4918 037664 100016          BPL     7$           ;;BR IF NO
4919 037666 042703 177770  BIC    #177770,R3     ;;GET RID OF JUNK
4920 037672 001002          BNE     4$           ;;TEST FOR 0
4921 037674 005704          TST     R4           ;;SUPPRESS THIS 0?
4922 037676 001403          BEQ     5$           ;;BR IF YES
4923 037700 005204          4$:    INC     R4     ;;DON'T SUPPRESS ANYMORE 0'S
4924 037702 052703 000060  BIS    #'0,R3         ;;MAKE THIS DIGIT ASCII
4925 037706 052703 000040  BIS    #' ,R3         ;;MAKE ASCII IF NOT ALREADY
4926 037712 110337 037756  MOVVB   R3,8$         ;;SAVE FOR TYPING
4927 037716 104401 037756  TYPE    ,8$          ;;GO TYPE THIS DIGIT
4928 037722 105337 037760  7$:    DECB   $OCNT     ;;COUNT BY 1
4929 037726 003347          BGT     2$           ;;BR IF MORE TO DO
4930 037730 002402          BLT     6$           ;;BR IF DONE
4931 037732 005204          INC     R4           ;;INSURE LAST DIGIT ISN'T A BLANK
4932 037734 000744          BR       2$          ;;GO DO THE LAST DIGIT
4933 037736 012605          6$:    MOV     (SP)+,R5 ;;RESTORE R5
4934 037740 012604          MOV     (SP)+,R4     ;;RESTORE R4
4935 037742 012603          MOV     (SP)+,R3     ;;RESTORE R3
4936 037744 016666 000002 000004  MOV     2(SP),4(SP)  ;;SET THE STACK FOR RETURNING
4937 037752 012616          MOV     (SP)+,(SP)
4938 037754 000002          RTI                    ;;RETURN
4939 037756 000          8$:    .BYTE  0          ;;STORAGE FOR ASCII DIGIT
4940 037757 000          .BYTE  0          ;;TERMINATOR FOR TYPE ROUTINE
4941 037760 000          $OCNT: .BYTE  0          ;;OCTAL DIGIT COUNTER
4942 037761 000          $OFILL: .BYTE  0        ;;ZERO FILL SWITCH
4943 037762 000000          $OMODE: .WORD  0        ;;NUMBER OF DIGITS TO TYPE
4944          .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
4945
4946          ;;*****
4947          ;;*THIS ROUTINE 'S USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
4948          ;;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
4949          ;;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
4950          ;;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
4951          ;;*REPLACED WITH SPACES.
4952          ;;*CALL:
4953          ;;*   MOV     NUM,-(SP)          ;;PUT THE BINARY NUMBER ON THE STACK
4954          ;;*   TYPDS          ;;GO TO THE ROUTINE
4955
4956          $1YPDS:
4957          037764 010046  MOV     R0,-(SP)     ;;PUSH R0 ON STACK
4958          037766 010146  MOV     R1,-(SP)     ;;PUSH R1 ON STACK
4959          037770 010246  MOV     R2,-(SP)     ;;PUSH R2 ON STACK
    
```



```

4960 037772 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
4961 037774 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
4962 037776 012746 020200  MOV      #2020,-(SP)   ;;SET BLANK SWITCH AND SIGN
4963 040002 016605 000020  MOV      20(SP),R5     ;;GET THE INPUT NUMBER
4964 040006 100004      BPL      1$           ;;BR IF INPUT IS POS.
4965 040010 005405      NEG      R5           ;;MAKE THE BINARY NUMBER POS.
4966 040012 112766 000055 000001  MOVB     #'-,1(SP)     ;;MAKE THE ASCII NUMBER NEG.
4967 040020 005000 1$:      CLR      R0           ;;ZERO THE CONSTANTS INDEX
4968 040022 012703 040200  MOV      $#DBLK,R3     ;;SETUP THE OUTPUT POINTER
4969 040026 112723 000040  MOVB     #' ,(R3)+     ;;SET THE FIRST CHARACTER TO A BLANK
4970 040032 005002 2$:      CLR      R2           ;;CLEAR THE BCD NUMBER
4971 040034 016001 040170  MOV      $DTBL(R0),R1  ;;GET THE CONSTANT
4972 040040 160105 3$:      SUB      R1,R5        ;;FORM THIS BCD DIGIT
4973 040042 002402      BLT     4$           ;;BR IF DONE
4974 040044 005202      INC     R2           ;;INCREASE THE BCD DIGIT BY 1
4975 040046 000774      BR      3$
4976 040050 060105 4$:      ADD     R1,R5        ;;ADD BACK THE CONSTANT
4977 040052 005702      TST     R2           ;;CHECK IF BCD DIGIT=0
4978 040054 001002      BNE     5$           ;;FALL THROUGH IF 0
4979 040056 105716      TSTB    (SP)         ;;STILL DOING LEADING 0'S?
4980 040060 100407      BMI     7$           ;;BR IF YES
4981 040062 106316 5$:      ASLB    (SP)         ;;MSD?
4982 040064 103003      BCC     6$           ;;BR IF NO
4983 040066 116663 000001 177777  MOVB     1(SP),-1(R3)  ;;YES--SET THE SIGN
4984 040074 052702 000060 6$:      BIS     #'0,R2       ;;MAKE THE BCD DIGIT ASCII
4985 040100 052702 000040 7$:      BIS     #' ,R2       ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
4986 040104 110223      MOVB    R2,(R3)+     ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
4987 040106 005720      TST     (R0)+        ;;JUST INCREMENTING
4988 040110 020027 000010  CMP     R0,#10       ;;CHECK THE TABLE INDEX
4989 040114 002746      BLT     2$           ;;GO DO THE NEXT DIGIT
4990 040116 003002      BGT     8$           ;;GO TO EXIT
4991 040120 010502      MOV     R5,R2        ;;GET THE LSD
4992 040122 000764      BR      6$           ;;GO CHANGE TO ASCII
4993 040124 105726 8$:      TSTB    (SP)+        ;;WAS THE LSD THE FIRST NON-ZERO?
4994 040126 100003      BPL     9$           ;;BR IF NO
4995 040130 116663 177777 177776  MOVB     -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
4996 040136 105013 9$:      (LRB   (R3)         ;;SET THE TERMINATOR
4997 040140 012605      MOV     (SP)+,R5     ;;POP STACK INTO R5
4998 040142 012603      MOV     (SP)+,R3     ;;POP STACK INTO R3
4999 040144 012602      MOV     (SP)+,R2     ;;POP STACK INTO R2
5000 040146 012601      MOV     (SP)+,R1     ;;POP STACK INTO R1
5001 040150 012600      MOV     (SP)+,R0     ;;POP STACK INTO R0
5002 040152 104401 040200  TYPE    $DBLK        ;;NOW TYPE THE NUMBER
5003 040156 016666 000002 000004  MOV     2(SP),4(SP)   ;;ADJUST THE STACK
5004 040164 012616      MOV     (SP)+,(SP)
5005 040166 000002      RTI
5006 040170 023420 $DTBL: 10000.        ;;RETURN TO USER
5007 040172 001750      1000.
5008 040174 000144      100.
5009 040176 000012      10.
5010 040200 000004 $DBLK: .BLKW 4
5011      .SBTTL  SAVF AND RESTORE R0-R5 ROUTINES
5012
5013      ;*****
5014      ;*SAVE R0-R5
5015      ;*CALL:

```

```

5016          ;*      SAVREG
5017          ;*UPCN RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
5018          ;*
5019          ;*TOP---(+16)
5020          ;* +2---(+18)
5021          ;* +4---R5
5022          ;* +6---R4
5023          ;* +8---R3
5024          ;*+10---R2
5025          ;*+12---R1
5026          ;*+14---R0
5027
5028          $SAVREG:
5029          040210 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
5030          040210 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
5031          040214 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
5032          040216 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
5033          040220 010446      MOV      R4,-(SP)      ;;PUSH R4 ON STACK
5034          040222 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
5035          040224 016646 000022  MOV      22(SP),-(SP)  ;;SAVE PS OF MAIN FLOW
5036          040230 016646 000022  MOV      22(SP),-(SP)  ;;SAVE PC OF MAIN FLOW
5037          040234 016646 000022  MOV      22(SP),-(SP)  ;;SAVE PS OF CALL
5038          040240 016646 000022  MOV      22(SP),-(SP)  ;;SAVE PC OF CALL
5039          040244 000002      RTI
5040
5041          ;*RESTORE R0-R5
5042          ;*CALL:
5043          ;*      RESREG
5044          $RESREG:
5045          040246 012666 000022  MOV      (SP)+,22(SP)  ;;RESTORE PC OF CALL
5046          040252 012666 000022  MOV      (SP)+,22(SP)  ;;RESTORE PS OF CALL
5047          040256 012666 000022  MOV      (SP)+,22(SP)  ;;RESTORE PC OF MAIN FLOW
5048          040262 012666 000022  MOV      (SP)+,22(SP)  ;;RESTORE PS OF MAIN FLOW
5049          040266 012605      MOV      (SP)+,R5      ;;POP STACK INTO R5
5050          040270 012604      MOV      (SP)+,R4      ;;POP STACK INTO R4
5051          040272 012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
5052          040274 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
5053          040276 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
5054          040300 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
5055          040302 000002      RTI
5056          .SBTTL  DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
5057
5058          ;*****
5059          ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
5060          ;*UNSIGNED OCTAL ASCII NUMBER.
5061          ;*CALL
5062          ;*      MOV      #PNTR,-(SP)      ;;POINTER TO LOW WORD OF BINARY NUMBER
5063          ;*      JSR      PC,@#$DB20      ;;CALL THE ROUTINE
5064          ;*      RETURN      ;;THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK
5065
5066          $DB20:  SAVREG
5067          040304 104413      MOV      2(SP),R1      ;;SAVE ALL REGISTERS
5068          040306 016601 000002  MOV      #12,R4        ;;PICKUP THE POINTER TO LOW WORD
5069          040312 012705 040423  MOV      #12,R4        ;;POINTER TO DATA TABLE
5070          040316 012704 000014  MOV      #^C7,R3      ;;DO ELEVEN CHARACTERS
5071          040322 012703 177770  MOV      #^C7,R3      ;;MASK
    
```

```

5072 040326 012100          MOV      (R1)+,R0          ;;LOWER WORD
5073 040330 012101          MOV      (R1)+,R1          ;;HIGH WORD
5074 040332 005002          CLR      R2              ;;TERMINATOR
5075 040334 110245          1$:     MOVVB   R2,-(R5)    ;;PUT CHARACTER IN DATA TABLE
5076 040336 010002          MOV      R0,R2          ;;GET THIS DIGIT
5077 040340 005304          DEC      R4              ;;COUNT THIS CHARACTER
5078 040342 003007          BGT     3$              ;;BR IF NOT THE LAST DIGIT
5079 040344 001405          BEQ     2$              ;;BR IF IT IS THE LAST DIGIT
5080 040346 005205          INC     R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
5081 040350 010566 000002    MOV      R5,2(SP)        ;;ASCIZ CHAR. & PUT IT ON THE STACK
5082 040354 104414          RESREG                    ;;RESTORE ALL REGISTERS
5083 040356 000207          RTS     PC              ;;RETURN TO USER
5084 040360 006203          2$:     ASR     R3              ;;POSITION THE MASK FOR THE LAST DIGIT
5085 040362 006001          3$:     ROR     R1              ;;POSITION THE BINARY NUMBER FOR
5086 040364 006000          ROR     R0              ;;
5087 040366 006001          ROR     R1              ;;
5088 040370 006000          ROR     R0              ;;
5089 040372 006001          ROR     R1              ;;
5090 040374 006000          ROR     R0              ;;
5091 040376 040302          BIC     R3,R2          ;;MASK OUT ALL JUNK
5092 040400 062702 000060    ADD     #'0,R2          ;;MAKE THIS CHAR. ASCII
5093 040404 000753          BR      1$              ;;GO PUT IT IN THE DATA TABLE
5094 040406 000016          $OCTVL: .BLKB 14.      ;;RESERVE DATA TABLE
    
```

```

5095          .SBTTL TRAP DECODER
5096
5097          ;:*****
5098          ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
5099          ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
5100          ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
5101          ;*GO TO THAT ROUTINE.
5102
5103 040424 010046          $TRAP: MOV     R0,-(SP)          ;;SAVE R0
5104 040426 016600 000002  MOV     2(SP),R0          ;;GET TRAP ADDRESS
5105 040432 005740          TST     -(R0)            ;;BACKUP BY 2
5106 040434 111000          MOVB   (R0),R0          ;;GET RIGHT BYTE OF TRAP
5107 040436 006300          ASL    R0              ;;POSITION FOR INDEXING
5108 040440 016000 040460  MOV     $TRPAD(R0),R0    ;;INDEX TO TABLE
5109 040444 000200          RTS     R0              ;;GO TO ROUTINE
5110
5111
5112          ;:THIS IS USE TO HANDLE THE "GETPRI" MACRO
5113
5114 040446 011646          $TRAP2: MOV    (SP),-(SP)  ;;MOVE THE PC DOWN
5115 040450 016666 000004 000002  MOV    4(SP),2(SP)      ;;MOVE THE PSW DOWN
5116 040456 000002          RTI                    ;;RESTORF THE PSW
5117
5118          .SBTTL TRAP TABLE
5119
5120          ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
5121          ;*BY THE "TRAP" INSTRUCTION.
5122
5123          :
5124          : ROUTINE
5125          :-----
5126 040460 040446          $TRPAD: .WORD   $TRAP2
5127 040462 036732          $TYPE   ;;CALL=TYPE     TRAP+1(104401) TTY TYPEOUT ROUTINE
5128 040464 037562          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
5129 040466 037536          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
5130 040470 037576          $TYPON  ;;CALL=TYPON    TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
5131 040472 037764          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
5132 040474 037462          $TYPBN  ;;CALL=TYPBN    TRAP+6(104406) TYPE BINARY (ASCII) NUMBER
5133 040476 035660          $GTSWR  ;;CALL=GTSWR    TRAP+7(104407) GET SOFT-SWR SETTING
5134
5135 040500 035610          $CKSWR  ;;CALL=CKSWR    TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
5136 040502 036132          $RDCHR  ;;CALL=RDCHR    TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
5137 040504 036252          $RDLIN  ;;CALL=RDLIN    TRAP+12(104412) TTY TYPEIN STRING ROUTINE
5138 040506 040210          $SAVREG ;;CALL=SAVREG    TRAP+13(104413) SAVE R0-R5 ROUTINE
5139 040510 040246          $RESREG ;;CALL=RESREG    TRAP+14(104414) RESTORE R0-R5 ROUTINE
5140          .SBTTL POWER DOWN AND UP ROUTINES
5141
5142          ;:*****
5143          ;:POWER DOWN ROUTINE
5144 040512 012737 040670 000024 $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
5145 040520 012737 000340 000026  MOV     #340,@#PWRVEC+2 ;;PRIO:7
5146 040526 010046          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
5147 040530 010146          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
5148 040532 010246          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
5149 040534 010346          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
5150 040536 010446          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
    
```

```

5151 040540 010546          MOV     R5,-(SP)          ;;PUSH R5 ON STACK
5152 040542 017746 140372   MOV     @SWR,-(SP)       ;;PUSH @SWR ON STACK
5153 040546 010637 040674   MOV     SP,$SAVR6       ;;SAVE SP
5154 040552 012737 040564 000024  MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
5155 040560 000000          HALT
5156 040562 000776          BR      -2              ;;HANG UP
5157
5158
5159
5160 040564 012737 040670 000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
5161 040572 013706 040674          MOV     $SAVR6,SP       ;;GET SP
5162 040576 005037 040674          CLR     $SAVR6         ;;WAIT LOOP FOR THE TTY
5163 040602 005237 040674 1$:    INC     $SAVR6         ;;WAIT FOR THE INC
5164 040606 001375          BNE    1$              ;;OF WORD
5165 040610 012677 140324   MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
5166 040614 012605          MOV     (SP)+,R5       ;;POP STACK INTO R5
5167 040616 012604          MOV     (SP)+,R4       ;;POP STACK INTO R4
5168 040620 012603          MOV     (SP)+,R3       ;;POP STACK INTO R3
5169 040622 012602          MOV     (SP)+,R2       ;;POP STACK INTO R2
5170 040624 012601          MOV     (SP)+,R1       ;;POP STACK INTO R1
5171 040626 012600          MOV     (SP)+,R0       ;;POP STACK INTO R0
5172 040630 012737 040512 000024  MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
5173 040636 012737 000340 000026  MOV     #340,@#PWRVEC+2 ;;PRIO:7
5174 040644 104401          TYPE
5175 040646 040676          $PWRMG: .WORD  PWRMSG    ;;REPORT THE POWER FAILURE
5176 040650 012716          MOV     (PC)+,(SP)     ;;POWER FAIL MESSAGE POINTER
5177 040652 020472          $PWRAD: .WORD  RESTRT   ;;RESTART AT RESTRT
5178 040654 042766 000020 000002  BIC    #20,2(SP)       ;;RESTART ADDRESS
5179 040662 005037 034144          CLR    $TBIT          ;;CLEAR 'T' BIT
5180 040666 000002          RTI
5181 040670 000000          $ILLUP: HALT          ;;THE POWER UP SEQUENCE WAS STARTED
5182 040672 000776          BR      -2              ;; BEFORE THE POWER DOWN WAS COMPLETE
5183 040674 000000          $SAVR6: 0              ;;PUT THE SP HERE
5184 040676 006412 050040 053517  PWRMSG: .ASCIIZ <12><15>? POWER FAILURE - RESTARTING ?<12><15>
5185 040704 051105 043040 044501
5186 040712 052514 042522 026440
5187 040720 051040 051505 040524
5188 040726 052122 047111 020107
5189 040734 006412 000
5190 040740
5191
5192
    .EVEN
    
```

5193					.SBTTL	ERROR MESSAGES, DATA HEADERS-TABLES & FORMATS
5194	040740	047125	054105	042520	EM1:	.ASCIZ /UNEXPECTED CPU TRAP TO LOC. 004/
5195	040746	052103	042105	041440		
5196	040754	052520	052040	040522		
5197	040762	020120	047524	046040		
5198	040770	041517	020056	030060		
5199	040776	000064				
5200	041000	047125	054105	042520	EM2:	.ASCIZ /UNEXPECTED MEM. MGMT. TRAP TO LOC. 250/
5201	041006	052103	042105	046440		
5202	041014	046505	020056	043515		
5203	041022	052115	020056	051124		
5204	041030	050101	052040	020117		
5205	041036	047514	027103	031040		
5206	041044	030065	000			
5207	041047	120	044522	051117	EM3:	.ASCIZ /PRIORITY BITS SET WRONG IN PSW/
5208	041054	052111	020131	044502		
5209	041062	051524	051440	052105		
5210	041070	053440	047522	043516		
5211	041076	044440	020116	051520		
5212	041104	000127				
5213	041106	047515	042504	041040	EM4:	.ASCIZ /MODE BITS SET WRONG IN PSW/
5214	041114	052111	020123	042523		
5215	041122	020124	051127	047117		
5216	041130	020107	047111	050040		
5217	041136	053523	000			
5218	041141	104	040525	020114	EM5:	.ASCIZ /DUAL ADDRESSING BETWEEN HI&LO BYTES OF PSW/
5219	041146	042101	051104	051505		
5220	041154	044523	043516	041040		
5221	041162	052105	042527	047105		
5222	041170	044040	023111	047514		
5223	041176	041040	052131	051505		
5224	041204	047440	020106	051520		
5225	041212	000127				
5226	041214	042513	047122	046105	EM6:	.ASCIZ /KERNEL R6 CHANGED BY WRITING USER R6/
5227	041222	051040	020066	044103		
5228	041230	047101	042507	020104		
5229	041236	054502	053440	044522		
5230	041244	044524	043516	052440		
5231	041252	042523	020122	033122		
5232	041260	000				
5233	041261	101	046440	046505	EM7:	.ASCIZ /A MEMORY MGMT. REG. TIMED OUT/
5234	041266	051117	020131	043515		
5235	041274	052115	020056	042522		
5236	041302	027107	052040	046511		
5237	041310	042105	047440	052125		
5238	041316	000				
5239	041317	123	046525	040515	EM10:	.ASCIZ /SUMMARY OF MEM. MGMT. REG. TIMEOUTS/
5240	041324	054522	047440	020106		
5241	041332	042515	027115	046440		
5242	041340	046507	027124	051040		
5243	041346	043505	020056	044524		
5244	041354	042515	052517	051524		
5245	041362	000				
5246	041363	115	046505	020056	EM11:	.ASCIZ /MEM. MGMT. REG. WOULD NOT CLEAR/
5247	041370	043515	052115	020056		
5248	041376	042522	027107	053440		

5249	041404	052517	042114	047040	
5250	041412	052117	041440	042514	
5251	041420	051101	000		
5252	041423	115	046505	020056	EM12: .ASCIZ /MEM. MGMT. REG. BITS NOT SET CORRECTLY/
5253	041430	043515	052115	020056	
5254	041436	042522	027107	041040	
5255	041444	052111	020123	047516	
5256	041452	020124	042523	020124	
5257	041460	047503	051122	041505	
5258	041466	046124	000131		
5259	041472	051123	020060	043105	EM13: .ASCIZ /SRO EFFECTED BY WRITE TO PSW/
5260	041500	042506	052103	042105	
5261	041506	041040	020131	051127	
5262	041514	052111	020105	047524	
5263	041522	050040	053523	000	
5264	041527	123	030522	042040	EM14: .ASCIZ /SR1 DID NOT READ ALL ZEROS/
5265	041534	042111	047040	052117	
5266	041542	051040	040505	020104	
5267	041550	046101	020114	042532	
5268	041556	047522	000123		
5269	041562	052504	046101	040440	EM15: .ASCIZ /DUAL ADDRESSING BETWEEN BYTES OF PAR OR PDR/
5270	041570	042104	042522	051523	
5271	041576	047111	020107	042502	
5272	041604	053524	042505	020116	
5273	041612	054502	042524	020123	
5274	041620	043117	050040	051101	
5275	041626	047440	020122	042120	
5276	041634	000122			
5277	041636	052504	046101	040440	EM16: .ASCIZ /DUAL ADDRESSING BETWEEN PAR-PDR'S/
5278	041644	042104	042522	051523	
5279	041652	047111	020107	042502	
5280	041660	053524	042505	020116	
5281	041666	040520	026522	042120	
5282	041674	023522	000123		
5283	041700	044120	051531	041511	EM17: .ASCIZ /PHYSICAL ADDRESS FORMED WRONG/
5284	041706	046101	040440	042104	
5285	041714	042522	051523	043040	
5286	041722	051117	042515	020104	
5287	041730	051127	047117	000107	
5288	041736	044120	051531	020056	EM20: .ASCIZ /PHYS. ADDR. FORMED WRONG IN RELOCATE MODE/
5289	041744	042101	051104	020056	
5290	041752	047506	046522	042105	
5291	041760	053440	047522	043516	
5292	041766	044440	020116	042522	
5293	041774	047514	040503	042524	
5294	042002	046440	042117	000105	
5295	042010	026527	044502	020124	EM21: .ASCIZ /W-BIT DID NOT GET SET IN PDR/
5296	042016	044504	020104	047516	
5297	042024	020124	042507	020124	
5298	042032	042523	020124	047111	
5299	042040	050040	051104	000	
5300	042045	127	041055	052111	EM22: .ASCIZ /W-BIT SET IN MORE THAN ONE PDR/
5301	042052	051440	052105	044440	
5302	042060	020116	047515	042522	
5303	042066	052040	040510	020116	
5304	042074	047117	020105	042120	

5305	042102	000122			
5306	042104	026527	044502	020124	EM23: .ASCIZ /W-BIT NOT CLEARED BY WRITING TO PDR/
5307	042112	047516	020124	046103	
5308	042120	040505	042522	020104	
5309	042126	054502	053440	044522	
5310	042134	044524	043516	052040	
5311	042142	020117	042120	000122	
5312	042150	051127	052111	047111	EM24: .ASCIZ /WRITING SRO SET W-BIT IN KIPDR7/
5313	042156	020107	051123	020060	
5314	042164	042523	020124	026527	
5315	042172	044502	020124	047111	
5316	042200	045440	050111	051104	
5317	042206	000067			
5318	042210	026527	044502	020124	EM25: .ASCIZ /W-BIT GOT SET DURING TIMEOUT ABORT/
5319	042216	047507	020124	042523	
5320	042224	020124	052504	044522	
5321	042232	043516	052040	046511	
5322	042240	047505	052125	040440	
5323	042246	047502	052122	000	
5324	042253	115	046505	051117	EM26: .ASCIZ /MEMORY MGMT. ACCESS ABORT DID NOT OCCUR/
5325	042260	020131	043515	052115	
5326	042266	020056	041501	042503	
5327	042274	051523	040440	047502	
5328	042302	052122	042040	042111	
5329	042310	047040	052117	047440	
5330	042316	041503	051125	000	
5331	042323	101	041503	051505	EM27: .ASCIZ /ACCESS ERROR DID NOT ABORT INSTRUCTION/
5332	042330	020123	051105	047522	
5333	042336	020122	044504	020104	
5334	042344	047516	020124	041101	
5335	042352	051117	020124	047111	
5336	042360	052123	052522	052103	
5337	042366	047511	000116		
5338	042372	051123	020060	044504	EM30: .ASCIZ /SRO DID NOT REPORT ACCESS ERROR CORRECTLY/
5339	042400	020104	047516	020124	
5340	042406	042522	047520	052122	
5341	042414	040440	041503	051505	
5342	042422	020123	051105	047522	
5343	042430	020122	047503	051122	
5344	042436	041505	046124	000131	
5345	042444	044504	020104	047516	EM31: .ASCIZ /DID NOT LOCKUP CORRECT VIRTUAL ADDR./
5346	042452	020124	047514	045503	
5347	042460	050125	041440	051117	
5348	042466	042522	052103	053040	
5349	042474	051111	052524	046101	
5350	042502	040440	042104	027122	
5351	042510	000			
5352	042511	120	043501	020105	EM32: .ASCIZ /PAGE LGTH. ABORT OCCURRED WHEN IT SHOULDN'T HAVE/
5353	042516	043514	044124	020056	
5354	042524	041101	051117	020124	
5355	042532	041517	052503	051122	
5356	042540	042105	053440	042510	
5357	042546	020116	052111	051440	
5358	042554	047510	046125	047104	
5359	042562	052047	044040	053101	
5360	042570	000105			



5361	042572	040520	042507	046040	EM33:	.ASCIZ /PAGE LGTH. ABORT DID NOT OCCUR WHEN IT SHOULD HAVE/
5362	042600	052107	027110	040440		
5363	042606	047502	052122	042040		
5364	042614	042111	047040	052117		
5365	042622	047440	041503	051125		
5366	042630	053440	042510	020116		
5367	042636	052111	051440	047510		
5368	042644	046125	020104	040510		
5369	042652	042526	000			
5370	042655	123	030122	042040	EM34:	.ASCIZ /SRC DID NOT REPORT PAGE LGTH. ABORT CORRECTLY/
5371	042662	042111	047040	052117		
5372	042670	051040	050105	051117		
5373	042676	020124	040520	042507		
5374	042704	046040	052107	027110		
5375	042712	040440	047502	052122		
5376	042720	041440	051117	042522		
5377	042726	052103	054514	000		
5378	042733	123	030122	047440	EM37:	.ASCIZ /SRO OR SR2 CHANGED BY A SECOND ABORT/
5379	042740	020122	051123	020062		
5380	042746	044103	047101	042507		
5381	042754	020104	054502	040440		
5382	042762	051440	041505	047117		
5383	042770	020104	041101	051117		
5384	042776	000124				
5385	043000	051123	020060	051117	EM40:	.ASCIZ /SRO OR SR2 WERE NOT 'RESET' BY A RESET/
5386	043006	051440	031122	053440		
5387	043014	051105	020105	047516		
5388	043022	020124	051042	051505		
5389	043030	052105	020042	054502		
5390	043036	040440	051040	051505		
5391	043044	052105	000			
5392	043047	123	031122	047040	EM41:	.ASCIZ /SR2 NOT TRACKING CORRECTLY/
5393	043054	052117	052040	040522		
5394	043062	045503	047111	020107		
5395	043070	047503	051122	041505		
5396	043076	046124	000131			
5397	043102	044504	020104	047516	EM42:	.ASCIZ /DID NOT TRAP THRU KERNEL SPACE/
5398	043110	020124	051124	050101		
5399	043116	052040	051110	020125		
5400	043124	042513	047122	046105		
5401	043132	051440	040520	042503		
5402	043140	000				
5403	043141	113	020124	051105	EM43:	.ASCIZ /KT ERROR NOT SERVICED ON TIMEOUT ERROR/
5404	043146	047522	020122	047516		
5405	043154	020124	042523	053122		
5406	043162	041511	042105	047440		
5407	043170	020116	044524	042515		
5408	043176	052517	020124	051105		
5409	043204	047522	000122			
5410	043210	051123	020060	051117	EM44:	.ASCIZ /SRO OR SR2 CHANGED BY TIMEOUT ERROR/
5411	043216	051440	031122	041440		
5412	043224	040510	043516	042105		
5413	043232	041040	020131	044524		
5414	043240	042515	052517	020124		
5415	043246	051105	047522	000122		
5416	043254	051105	047522	020122	EM45:	.ASCIZ /ERROR DURING 'DOUBLE ERROR' (KT & TIMEOUT)/

5417	043262	052504	044522	043516	
5418	043270	021040	047504	041125	
5419	043276	042514	042440	051122	
5420	043304	051117	020042	045450	
5421	043312	020124	020046	044524	
5422	043320	042515	052517	024524	
5423	043326	000			
5424	043327	115	050106	020111	EM46: .ASCIZ /MFPI INSTRUCTION PUSHED WRONG DATA/
5425	043334	047111	052123	052522	
5426	043342	052103	047511	020116	
5427	043350	052520	044123	042105	
5428	043356	053440	047522	043516	
5429	043364	042040	052101	000101	
5430	043372	052115	044520	044440	EM47: .ASCIZ /MTPI INSTRUCTION LOADED WRONG DATA/
5431	043400	051516	051124	041525	
5432	043406	044524	047117	046040	
5433	043414	040517	042504	020104	
5434	043422	051127	047117	020107	
5435	043430	040504	040524	000	
5436	043435	123	040524	045503	EM50: .ASCIZ /STACK NOT PUSHED BY MFPI-MTPI/
5437	043442	047040	052117	050040	
5438	043450	051525	042510	020104	
5439	043456	054502	046440	050106	
5440	043464	026511	052115	044520	
5441	043472	000			
5442	043473	113	051105	042516	EM51: .ASCIZ /KERNEL PAGE ACCESS INSTEAD OF USER: MFPI-MTPI/
5443	043500	020114	040520	042507	
5444	043506	040440	041503	051505	
5445	043514	020123	047111	052123	
5446	043522	040505	020104	043117	
5447	043530	052440	042523	035122	
5448	043536	046440	050106	026511	
5449	043544	052115	044520	000	
5450	043551	127	047522	043516	EM52: .ASCIZ /WRONG PDR'S REFERENCED WHILE IN RELOCATE MODE/
5451	043556	050040	051104	051447	
5452	043564	051040	043105	051105	
5453	043572	047105	042503	020104	
5454	043600	044127	046111	020105	
5455	043606	047111	051040	046105	
5456	043614	041517	052101	020105	
5457	043622	047515	042504	000	
5458	043627	115	050106	020104	EM53: .ASCIZ /MFPD INSTRUCTION PUSHED WRONG DATA/
5459	043634	047111	052123	052522	
5460	043642	052103	047511	020116	
5461	043650	052520	044123	042105	
5462	043656	053440	047522	043516	
5463	043664	042040	052101	000101	
5464	043672	052123	041501	020113	EM54: .ASCIZ /STACK NOT PUSHED BY MFPD-MTPD/
5465	043700	047516	020124	052520	
5466	043706	044123	042105	041040	
5467	043714	020131	043115	042120	
5468	043722	046455	050124	000104	
5469	043730	040520	020122	051117	EM55: .ASCIZ /PAR OR PDR CHANGED BY A RESET/
5470	043736	050040	051104	041440	
5471	043744	040510	043516	042105	
5472	043752	041040	020131	020101	

5473	043760	042522	042523	000124	
5474	043766	051520	020127	044103	FM56: .ASCIZ /PSW CHANGED BY AN RTI IN USER MODE/
5475	043774	047101	042507	020104	
5476	044002	054502	040440	020116	
5477	044010	052122	020111	047111	
5478	044016	052440	042523	020122	
5479	044024	047515	042504	000	
5480					
5481	044031	117	042114	050040	DH1: .ASCIZ /OLD PC OLD PSW R6 WAS TESTNO ERRORPC/
5482	044036	020103	047440	042114	
5483	044044	050040	053523	051040	
5484	044052	020066	040527	020123	
5485	044060	052040	051505	047124	
5486	044066	020117	042440	051122	
5487	044074	051117	041520	000	
5488	044101	117	042114	050040	DH2: .ASCIZ /OLD PC OLD PSW R6 WAS SR0 SR2 TESTNO ERRORPC/
5489	044106	020103	047440	042114	
5490	044114	050040	053523	051040	
5491	044122	020066	040527	020123	
5492	044130	051440	030122	020040	
5493	044136	020040	051440	031122	
5494	044144	020040	020040	052040	
5495	044152	051505	047124	020117	
5496	044160	042440	051122	051117	
5497	044166	041520	000		
5498	044171	127	047522	042524	DH3: .ASCIZ /WROTE READ TESTNO ERRORPC/
5499	044176	020040	051040	040505	
5500	044204	020104	020040	052040	
5501	044212	051505	047124	020117	
5502	044220	042440	051122	051117	
5503	044226	041520	000		
5504	044231	101	042104	042522	DH7: .ASCIZ /ADDRESS TESTNO ERRORPC/
5505	044236	051523	052040	051505	
5506	044244	047124	020117	042440	
5507	044252	051122	051117	041520	
5508	044260	000			
5509	044261	122	043505	051511	DH10: .ASCII /REGISTER-ADDRS NUM OF/<CRLF>
5510	044266	042524	026522	042101	
5511	044274	051104	020123	047040	
5512	044302	046525	020040	043117	
5513	044310	200			
5514	044311	101	042116	042455	.ASCIZ /AND-ED OR-ED TIMOUTS TESTNO ERRORPC/
5515	044316	020104	047440	026522	
5516	044324	042105	020040	052040	
5517	044332	046511	052517	051524	
5518	044340	052040	051505	047124	
5519	044346	020117	042440	051122	
5520	044354	051117	041520	000	
5521	044361	122	043505	051511	DH11: .ASCII /REG'ISTR READ READ-(BINARY)/<CRLF>
5522	044366	051124	051040	040505	
5523	044374	020104	020040	051040	
5524	044402	040505	026504	041050	
5525	044410	047111	051101	024531	
5526	044416	200			
5527	044417	101	042104	042522	.ASCIZ /ADDRESS (OCTAL) 5432109876543210 TESTNO ERRORPC/
5528	044424	051523	024040	041517	



5585	045114	051523	053040	040502					
5586	045122	020040	020040	053040					
5587	045130	040502	020040	020040					
5588	045136	050040	051101	032040					
5589	045144	020040	050040	051101					
5590	045152	032440	020040	050040					
5591	045160	053523	020040	020040					
5592	045166	052040	051505	047124					
5593	045174	020117	042440	051122					
5594	045202	051117	041520	000					
5595	045207	120	051104	020040	DH21:	.ASCII	/PDR	VIRTUAL/<CRLF>	
5596	045214	020040	053040	051111					
5597	045222	052524	046101	200					
5598	045227	124	051505	042524		.ASCIIZ	/TESTED	ADDRESS TESTNO ERRORPC/	
5599	045234	020104	040440	042104					
5600	045242	042522	051523	052040					
5601	045250	051505	047124	020117					
5602	045256	042440	051122	051117					
5603	045264	041520	000						
5604	045267	120	051104	044440	DH22:	.ASCII	/PDR IN PDR	VIRTUAL/	
5605	045274	020116	050040	051104					
5606	045302	020040	020040	053040					
5607	045310	051111	052524	046101					
5608	045316	051105	047522	020122		.ASCIIZ	/ERROR	TESTED ADDRESS TESTNO ERRORPC/	
5609	045324	020040	042524	052123					
5610	045332	042105	020040	042101					
5611	045340	051104	051505	020123					
5612	045346	042524	052123	047516					
5613	045354	020040	051105	047522					
5614	045362	050122	000103						
5615	045366	042120	020122	020040	DH23:	.ASCIIZ	/PDR	TESTNO ERRORPC/	
5616	045374	020040	042524	052123					
5617	045402	047516	020040	051105					
5618	045410	047522	050122	000103					
5619	045416	042120	020122	040527	DH24:	.ASCIIZ	/PDR WAS EXPECTD	TESTNO ERRORPC/	
5620	045424	020123	054105	042520					
5621	045432	052103	020104	042524					
5622	045440	052123	047516	020040					
5623	045446	051105	047522	050122					
5624	045454	000103							
5625	045456	042120	020122	020064	DH26:	.ASCIIZ	/PDR 4 PSW	TESTNO ERRORPC/	
5626	045464	020040	051520	020127					
5627	045472	020040	020040	042524					
5628	045500	052123	047516	020040					
5629	045506	051105	047522	050122					
5630	045514	000103							
5631	045516	051123	020060	040527	DH30:	.ASCIIZ	/SR0 WAS EXPECTD	PDR 4 PSW TESTNO ERRORPC/	
5632	045524	020123	054105	042520					
5633	045532	052103	020104	042120					
5634	045540	020122	020064	020040					
5635	045546	051520	020127	020040					
5636	045554	020040	042524	052123					
5637	045562	047516	020040	051105					
5638	045570	047522	050122	000103					
5639	045576	051123	020062	040527	DH31:	.ASCIIZ	/SR2 WAS EXPECTD	PDR 4 PSW TESTNO ERRORPC/	
5640	045604	020123	054105	042520					

5641	045612	052105	020104	042120	
5642	045620	020122	020064	020040	
5643	045626	051520	020127	020040	
5644	045634	020040	042524	052123	
5645	045642	047516	020040	051105	
5646	045650	047522	050122	000103	
5647	045656	027126	027102	027101	DH32: .ASCIZ /V.B.A. KIPDR4 SR0 WAS SR2 WAS TESTNO ERRORPC/
5648	045664	020040	044513	042120	
5649	045672	032122	020040	051123	
5650	045700	020050	040527	020123	
5651	045706	051123	020062	040527	
5652	045714	020123	042524	052123	
5653	045722	047516	020040	051105	
5654	045730	047522	050122	000103	
5655	045736	027126	027102	027101	DH33: .ASCIZ /V.B.A. KIPDR4 TESTNO ERRORPC/
5656	045744	020040	044513	042120	
5657	045752	032122	020040	042524	
5658	045760	052123	047516	020040	
5659	045766	051105	047522	050122	
5660	045774	000103			
5661	045776	027126	027102	027101	DH34: .ASCIZ /V.B.A. KIPDR4 SR0 WAS EXPECTD TESTNO ERRORPC/
5662	046004	020040	044513	042120	
5663	046012	032122	020040	051123	
5664	046020	020060	040527	020123	
5665	046026	054105	042520	052103	
5666	046034	020104	042524	052123	
5667	046042	047516	020040	051105	
5668	046050	047522	050122	000103	
5669	046056	027126	027102	027101	DH35: .ASCIZ /V.B.A. KIPDR4 SR2 WAS EXPECTD TESTNO ERRORPC/
5670	046064	020040	044513	042120	
5671	046072	032122	020040	051123	
5672	046100	020062	040527	020123	
5673	046106	054105	042520	052103	
5674	046114	020104	042524	052123	
5675	046122	047516	020040	051105	
5676	046130	047522	050122	000103	
5677	046136	051123	020062	040527	DH36: .ASCIZ /SR2 WAS EXPECTD TESTNO ERRORPC/
5678	046144	020123	054105	042520	
5679	046152	052103	020104	042524	
5680	046160	052123	047516	020040	
5681	046166	051105	047522	050122	
5682	046174	000103			
5683	046176	044506	051522	020124	DH37: .ASCII /FIRST ABORT SECOND ABORT/<CRLF>
5684	046204	041101	051117	020124	
5685	046212	020040	020040	042523	
5686	046220	047503	042116	040440	
5687	046226	047502	052122	200	
5688	046233	123	030122	053440	.ASCIZ /SR0 WAS SR2 WAS SR0 WAS SR2 WAS TESTNO ERRORPC/
5689	046240	051501	051440	031122	
5690	046246	053440	051501	051440	
5691	046254	030122	053440	051501	
5692	046262	051440	031122	053440	
5693	046270	051501	052040	051505	
5694	046276	047124	020117	042440	
5695	046304	051122	051117	041520	
5696	046312	000			



5753	046774	041520	000		
5754	046777	124	051505	047124	DH50: .ASCIZ /TESTNO ERRORPC/
5755	047004	020117	042440	051122	
5756	047012	051117	041520	000	
5757	047017	123	030122	053440	DH51: .ASCIZ /SRO WAS SR2 WAS TESTNO ERRORPC/
5758	047024	051501	051440	031122	
5759	047032	053440	051501	052040	
5760	047040	051505	047124	020117	
5761	047046	042440	051122	051117	
5762	047054	041520	000		
5763	047057	120	054510	044523	DH52: .ASCII /PHYSICL PAR 4/<CRLF>
5764	047064	046103	050040	051101	
5765	047072	032040	200		
5766	047075	101	042104	042522	.ASCIZ /ADDRESS V.B.A. PAR 4 SRO WAS SR2 WAS PSW TESTNO ERRORPC/
5767	047102	051523	053040	041056	
5768	047110	040456	020056	050040	
5769	047116	051101	032040	020040	
5770	047124	051440	030122	053440	
5771	047132	051501	051440	031122	
5772	047140	053440	051501	050040	
5773	047146	053523	020040	020040	
5774	047154	052040	051505	047124	
5775	047162	020117	042440	051122	
5776	047170	051117	041520	000	
5777	047175	120	053523	053440	DH56: .ASCIZ /PSW WAS EXPECTD TESTNO ERRORPC/
5778	047202	051501	042440	050130	
5779	047210	041505	042124	052040	
5780	047216	051505	047124	020117	
5781	047224	042440	051122	051117	
5782	047232	041520	000		
5783					
5784		047236			.EVEN
5785					
5786	047236	001266	001270	001264	DT1: .WORD TRAPPC,TRAPPS,WASR6,TESTNO,\$ERRPC,0
5787	047244	001262	001116	000000	
5788	047252	001266	001270	001264	DT2: .WORD TRAPPC,TRAPPS,WASR6,WASSRO,WASSR2,TESTNO,\$ERRPC,0
5789	047260	001272	001274	001262	
5790	047266	001116	000000		
5791	047272	001162	001164	001262	DT3: .WORD \$REG0,\$REG1,TESTNO,\$ERRPC,0
5792	047300	001116	000000		
5793	047304	001162	001262	001116	DT7: .WORD \$REG0,TESTNO,\$ERRPC,0
5794	047312	000000			
5795	047314	001300	001302	001304	DT10: .WORD ANDADR,ORADR,TONUM,TESTNO,\$ERRPC,0
5796	047322	001262	001116	000000	
5797	047330	001162	001164	001164	DT11: .WORD \$REG0,\$REG1,\$REG1,TESTNO,\$ERRPC,0
5798	047336	001262	001116	000000	
5799	047344	001162	001164	001166	DT12: .WORD \$REG0,\$REG1,\$REG2,\$REG2,TESTNO,\$ERRPC,0
5800	047352	001166	001262	001116	
5801	047360	000000			
5802	047362	001162	001262	001116	DT13: .WORD \$REG0,TESTNO,\$ERRPC,0
5803	047370	000000			
5804	047372	001162	001164	001174	DT16: .WORD \$REG0,\$REG1,\$REG5,\$REG2,TESTNO,\$ERRPC,0
5805	047400	001166	001262	001116	
5806	047406	000000			
5807	047410	001312	001306	001172	DT17: .WORD PBALO,VIRT1,\$RFG4,TESTNO,\$ERRPC,0
5808	047416	001262	001116	000000	



5809	047424	001312	001306	001310	DT20:	.WORD	PBALO,VIRT1,VIRT2,\$REG4,\$REG5,\$TMP0,TESTNO,\$ERRPC,0
5810	047432	001172	001174	001176			
5811	047440	001262	001116	000000			
5812	047446	001174	001170	001262	DT21:	.WORD	\$REG5,\$REG3,TESTNO,\$ERRPC,0
5813	047454	001116	000000				
5814	047460	001162	001174	001170	DT22:	.WORD	\$REG0,\$REG5,\$REG3,TESTNO,\$ERRPC,0
5815	047466	001262	001116	000000			
5816	047474	001174	001262	001116	DT23:	.WORD	\$REG5,TESTNO,\$ERRPC,0
5817	047502	000000					
5818	047504	001166	001164	001262	DT24:	.WORD	\$REG2,\$REG1,TESTNO,\$ERRPC,0
5819	047512	001116	000000				
5820	047516	001166	001176	001262	DT26:	.WORD	\$REG2,\$TMP0,TESTNO,\$ERRPC,0
5821	047524	001116	000000				
5822	047530	001272	001170	001166	DT30:	.WORD	WASSR0,\$REG3,\$REG2,\$TMP0,TESTNO,\$ERRPC,0
5823	047536	001176	001262	001116			
5824	047544	000000					
5825	047546	001274	001172	001166	DT31:	.WORD	WASSR2,\$REG4,\$REG2,\$TMP0,TESTNO,\$ERRPC,0
5826	047554	001176	001262	001116			
5827	047562	000000					
5828	047564	001162	001172	001272	DT32:	.WORD	\$REG0,\$REG4,WASSR0,WASSR2,TESTNO,\$ERRPC,0
5829	047572	001274	001262	001116			
5830	047600	000000					
5831	047602	001162	001172	001262	DT33:	.WORD	\$REG0,\$REG4,TESTNO,\$ERRPC,0
5832	047610	001116	000000				
5833	047614	001162	001172	001272	DT34:	.WORD	\$REG0,\$REG4,WASSR0,\$REG2,TESTNO,\$ERRPC,0
5834	047622	001166	001262	001116			
5835	047630	000000					
5836	047632	001162	001172	001274	DT35:	.WORD	\$REG0,\$REG4,WASSR2,\$REG3,TESTNO,\$ERRPC,0
5837	047640	001170	001262	001116			
5838	047646	000000					
5839	047650	001274	001164	001262	DT36:	.WORD	WASSR2,\$REG1,TESTNO,\$ERRPC,0
5840	047656	001116	000000				
5841	047662	001176	001202	001272	DT37:	.WORD	\$TMP0,\$TMP2,WASSR0,WASSR2,TESTNO,\$ERRPC,0
5842	047670	001274	001262	001116			
5843	047676	000000					
5844	047700	001272	001274	001262	DT40:	.WORD	WASSR0,WASSR2,TESTNO,\$ERRPC,0
5845	047706	001116	000000				
5846	047712	001164	001166	001262	DT42:	.WORD	\$REG1,\$REG2,TESTNO,\$ERRPC,0
5847	047720	001116	000000				
5848	047724	001162	001164	001272	DT44:	.WORD	\$REG0,\$REG1,WASSR0,WASSR2,TESTNO,\$ERRPC,0
5849	047732	001274	001262	001116			
5850	047740	000000					
5851	047742	001164	001170	001272	DT45:	.WORD	\$REG1,\$REG3,WASSR0,WASSR2,TESTNO,\$ERRPC,0
5852	047750	001274	001262	001116			
5853	047756	000000					
5854	047760	001162	001164	001262	DT46:	.WORD	\$REG0,\$REG1,TESTNO,\$ERRPC,0
5855	047766	001116	000000				
5856	047772	001262	001116	000000	DT50:	.WORD	TESTNO,\$ERRPC,0
5857	050000	001272	001274	001262	DT51:	.WORD	WASSR0,WASSR2,TESTNO,\$ERRPC,0
5858	050006	001116	000000				
5859	050012	001312	001306	001172	DT52:	.WORD	PBALO,VIRT1,\$REG4,WASSR0,WASSR2,\$TMP0,TESTNO,\$ERRPC,0
5860	050020	001272	001274	001176			
5861	050026	001262	001116	000000			
5862	050034	001164	001166	001262	DT56:	.WORD	\$REG1,\$REG2,TESTNO,\$ERRPC,0
5863	050042	001116	000000				
5864							

5865	050046	000	000	000	DF1:	.BYTE	0,0,0,0,0
5866	050051	000	000				
5867	050053	000	000	000	DF2:	.BYTE	0,0,0,0,0,0,0
5868	050056	000	000	000			
5869	050061	000					
5870	050062	000	000	000	DF3:	.BYTE	0,0,0,0
5871	050065	000					
5872	050066	000	000	000	DF7:	.BYTE	0,0,0
5873	050071	000	000	001	DF10:	.BYTE	0,0,1,0,0
5874	050074	000	000				
5875	050076	000	000	002	DF11:	.BYTE	0,0,2,0,0
5876	050101	000	000				
5877	050103	000	000	000	DF12:	.BYTE	0,0,0,2,0,0
5878	050106	002	000	000			
5879	050111	000	000	000	DF13:	.BYTE	0,0,0
5880	050114	000	000	000	DF16:	.BYTE	0,0,0,0,0,0
5881	050117	000	000	000			
5882	050122	003	000	000	DF17:	.BYTE	3,0,0,0,0
5883	050125	000	000				
5884	050127	003	000	000	DF20:	.BYTE	3,0,0,0,0,0,0,0
5885	050132	000	000	000			
5886	050135	000	000				
5887	050137	000	000	000	DF21:	.BYTE	0,0,0,0
5888	050142	000					
5889	050143	000	000	000	DF22:	.BYTE	0,0,0,0,0
5890	050146	000	000				
5891	050150	000	000	000	DF23:	.BYTE	0,0,0
5892	050153	000	000	000	DF24:	.BYTE	0,0,0,0
5893	050156	000					
5894	050157	000	000	000	DF30:	.BYTE	0,0,0,0,0,0
5895	050162	000	000	000			
5896	050165	000	000	000	DF46:	.BYTE	0,0,0,0
5897	050170	000					
5898	050171	000	000		DF50:	.BYTE	0,0
5899	050173	000	000	000	DF51:	.BYTE	0,0,0,0
5900	050176	000					
5901	050177	003	000	000	DF52:	.BYTE	3,0,0,0,0,0,0,0
5902	050202	000	000	000			
5903	050205	000	000				
5904	050207	000	000	000	DF56:	.BYTE	0,0,0,0
5905	050212	000					
5906							
5907		000001				.END	

ABASE = 000000	BIT0 = 000001	DF46 = 050165	DT23 = 047474	EM44 = 043210
ACDW1 = 000000	BIT00 = 000001	DF50 = 050171	DT24 = 047504	EM45 = 043254
ACDW2 = 000000	BIT01 = 000002	DF51 = 050173	DT26 = 047516	EM46 = 043327
ACPUOP= 000000	BIT02 = 000004	DF52 = 050177	DT3 = 047272	EM47 = 043372
ADDW0 = 000000	BIT03 = 000010	DF56 = 050207	DT30 = 047530	EM5 = 041141
ADDW1 = 000000	BIT04 = 000020	DF7 = 050066	DT31 = 047546	EM50 = 043435
ADDW10= 000000	BIT05 = 000040	DH1 = 044031	DT32 = 047564	EM51 = 043473
ADDW11= 000000	BIT06 = 000100	DH10 = 044261	DT33 = 047602	EM52 = 043551
ADDW12= 000000	BIT07 = 000200	DH11 = 044361	DT34 = 047614	EM53 = 043627
ADDW13= 000000	BIT08 = 000400	DH12 = 044501	DT35 = 047632	EM54 = 043672
ADDW14= 000000	BIT09 = 001000	DH13 = 044641	DT36 = 047650	EM55 = 043730
ADDW15= 000000	BIT1 = 000002	DH16 = 044671	DT37 = 047662	EM56 = 043766
ADDW2 = 000000	BIT10 = 002000	DH17 = 044771	DT40 = 047700	EM6 = 041214
ADDW3 = 000000	BIT11 = 004000	DH2 = 044101	DT42 = 047712	EM7 = 041261
ADDW4 = 000000	BIT12 = 010000	DH20 = 045061	DT44 = 047724	ERRTYP = 034670
ADDW5 = 000000	BIT13 = 020000	DH21 = 045207	DT45 = 047742	ERRVEC= 000004
ADDW6 = 000000	BIT14 = 040000	DH22 = 045267	DT46 = 047760	FORMPA = 035466
ADDW7 = 000000	BIT15 = 100000	DH23 = 045366	DT50 = 047772	GTSWR = 104407
ADDW8 = 000000	BIT2 = 000004	DH24 = 045416	DT51 = 050000	HT = 000011
ADDW9 = 000000	BIT3 = 000010	DH26 = 045456	DT52 = 050012	IOTVEC= 000020
ADEVCT= 000000	BIT4 = 000020	DH3 = 044171	DT56 = 050034	KERSTK= 001100
ADEVN = 000000	BIT5 = 000040	DH30 = 045516	DT7 = 047304	KIPAR0= 172340
AENV = 000000	BIT6 = 000100	DH31 = 045576	EMTVEC= 000030	KIPAR1= 172342
AENVN = 000000	BIT7 = 000200	DH32 = 045656	EM1 = 040740	KIPAR2= 172344
AFATAL= 000000	BIT8 = 000400	DH33 = 045736	EM10 = 041317	KIPAR3= 172346
AMADR1= 000000	BIT9 = 001000	DH34 = 045776	EM11 = 041363	KIPAR4= 172350
AMADR2= 000000	BPTVEC= 000014	DH35 = 046056	EM12 = 041423	KIPAR5= 172352
AMADR3= 000000	CKSWR = 104410	DH36 = 046136	EM13 = 041472	KIPAR6= 172354
AMADR4= 000000	CMPREG = 035274	DH37 = 046176	EM14 = 041527	KIPAR7= 172356
AMAMS1= 000000	CMSG = 036661	DH40 = 046313	EM15 = 041562	KIPDR0= 172300
AMAMS2= 000000	CNTRLC = 036602	DH42 = 046353	EM16 = 041636	KIPDR1= 172302
AMAMS3= 000000	CR = 000015	DH44 = 046413	EM17 = 041700	KIPDR2= 172304
AMAMS4= 000000	CRLF = 000200	DH45 = 046525	EM2 = 041000	KIPDR3= 172306
AMSGAD= 000000	DALTB1 = 027006	DH46 = 046722	EM20 = 041736	KIPDR4= 172310
AMSGLG= 000000	DALTB2 = 027042	DH50 = 046777	EM21 = 042010	KIPDR5= 172312
AMSGTY= 000000	DALTB3 = 027354	DH51 = 047017	EM22 = 042045	KIPDR6= 172314
AMTYP1= 000000	DALTB4 = 027410	DH52 = 047057	EM23 = 042104	KIPDR7= 172316
AMTYP2= 000000	DDISP = 177570	DH56 = 047175	EM24 = 042150	KSP = 000006
AMTYP3= 000000	DF1 = 050046	DH7 = 044231	EM25 = 042210	LF = 000012
AMTYP4= 000000	DF10 = 050071	DISPLA = 001142	EM26 = 042253	LOOP = 020514
ANDADR = 001300	DF11 = 050076	DISPRE = 000174	EM27 = 042323	MGMERR = 002150
APASS = 000000	DF12 = 050103	DSWR = 177570	EM3 = 041047	MGMFLG = 002152
APRIOR= 000000	DF13 = 050111	DT1 = 047236	EM30 = 042372	MMVEC = 000250
APTC SU= 000040	DF16 = 050114	DT10 = 047314	EM31 = 042444	ORADR = 001302
APTENV= 000001	DF17 = 050122	DT11 = 047330	EM32 = 042511	PBAHI = 001314
APTSIZ= 000200	DF2 = 050053	DT12 = 047344	EM33 = 042572	PBALO = 001312
APTSP0= 000100	DF20 = 050127	DT13 = 047362	EM34 = 042655	PDRTB1 = 027024
ASWREG= 000000	DF21 = 050137	DT16 = 047372	EM37 = 042733	PDRTB2 = 027056
ATESTN= 000000	DF22 = 050143	DT17 = 047410	EM4 = 041106	PDRTB3 = 027372
AUNIT = 000000	DF23 = 050150	DT2 = 047252	EM40 = 043000	PDRTB4 = 027424
AUSWR = 000000	DF24 = 050153	DT20 = 047424	EM41 = 043047	PIRQ = 177772
AVECT1= 000000	DF3 = 050062	DT21 = 047446	EM42 = 043102	PIRQVE= 000240
AVECT2= 000000	DF30 = 050157	DT22 = 047460	EM43 = 043141	PRO = 000000

PR1 = 000040	SW8 = 000400	TST46 = 033470	\$CNTLC = 036540	\$MSGTY = 001226
PR2 = 000100	SW9 = 001000	TST47 = 033574	\$CNTLG = 036552	\$MSWR = 036557
PR3 = 000140	TBIT = 000020	TST5 = 021124	\$CNTLU = 036545	\$MTYP1 = 001257
PR4 = 000200	TBITPS = 001276	TST6 = 021262	\$CPUOP = 001254	\$MXCNT = 034430
PR5 = 000240	TBITVE = 000014	TST7 = 021414	\$CRLF = 001223	\$NULL = 001154
PR6 = 000300	TESTNO = 001262	TYPBN = 104406	\$DBLK = 040200	\$NWTST = 000001
PR7 = 000340	TIMEON = 020472	TYPDS = 104405	\$DBZ0 = 040304	\$OCNT = 037760
PS = 177776	TIMERR = 002076	TYPE = 104401	\$DEVCT = 001236	\$OCTVL = 040406
PSW = 177776	TIMFLG = 002100	TYPOC = 104402	\$DOAGN = 034102	\$OMODE = 037762
PWRMSG = 040676	TIMOFF = 033636	TYPON = 104404	\$DTBL = 040170	\$OVER = 034414
PWRVEC = 000024	TKVEC = 000060	TYPOS = 104403	\$ENDAD = 034072	\$PASS = 001234
RDCHR = 104411	TOFF = 035114	UIPAR0 = 177640	\$ENDCT = 033714	\$PASTM = 000212
RDLIN = 104412	TON = 035150	UIPAR1 = 177642	\$ENULL = 034146	\$PWRAD = 040652
RESREG = 104414	TONUM = 001304	UIPAR2 = 177644	\$ENV = 001246	\$PWRDN = 040512
RESTR = 020472	TPVEC = 000064	UIPAR3 = 177646	\$ENVM = 001247	\$PWRMG = 040646
RESVEC = 000010	TRAPPC = 001266	UIPAR4 = 177650	\$EOP = 033660	\$PWRUP = 040564
R6 = %000006	TRAPPS = 001270	UIPAR5 = 177652	\$EOPCT = 033706	\$QUES = 001222
R7 = %000007	TRAPVE = 000034	UIPAR6 = 177654	\$ERFLG = 001103	\$RDCHR = 036132
SAVREG = 104413	TRTVEC = 000014	UIPAR7 = 177656	\$ERMAX = 001115	\$RDLIN = 036252
SETREG = 035202	TST1 = 020576	UIPDR0 = 177600	\$ERROR = 034432	\$RDSZ = 000010
SRO = 177572	TST10 = 021546	UIPDR1 = 177602	\$ERRPC = 001116	\$REGAD = 001160
SR1 = 177574	TST11 = 021700	UIPDR2 = 177604	\$ERRTB = 001316	\$REG0 = 001162
SR2 = 177576	TST12 = 022032	UIPDR3 = 177606	\$ERTTL = 001112	\$REG1 = 001164
SR3 = 172516	TST13 = 022174	UIPDR4 = 177610	\$ESCAP = 001214	\$REG2 = 001166
STACK = 001100	TST14 = 022232	UIPDR5 = 177612	\$ETABL = 001246	\$REG3 = 001170
START = 020000	TST15 = 022302	UIPDR6 = 177614	\$ETEND = 001262	\$REG4 = 001172
STKMT = 177774	TST16 = 022420	UIPDR7 = 177616	\$FATAL = 001230	\$REG5 = 001174
SWR = 001140	TST17 = 022542	USESTK = 000700	\$FFLG = 037460	\$RESRE = 040246
SWREG = 000176	TST2 = 020650	USP = %000006	\$FILLC = 001156	\$RTNAD = 034142
SW0 = 000001	TST20 = 022672	VIRT1 = 001306	\$FILLS = 001155	\$RTRN = 034136
SW00 = 000001	TST21 = 023022	VIRT2 = 001310	\$GDADR = 001120	\$SAVRE = 040210
SW01 = 000002	TST22 = 023226	WASR6 = 001264	\$GDDAT = 001124	\$SAVR6 = 040674
SW02 = 000004	TST23 = 023376	WASSR0 = 001272	\$GET42 = 034044	\$SCOF E = 034152
SW03 = 000010	TST24 = 024126	WASSR2 = 001274	\$GTSWR = 035660	\$SETUP = 000137
SW04 = 000020	TST25 = 024516	WBIT = 000100	\$HD = 000000	\$STUP = 177777
SW05 = 000040	TST26 = 025234	\$APTHD = 000204	\$HIBTS = 000204	\$SVLAD = 034360
SW06 = 000100	TST27 = 025430	\$ATYC = 037240	\$ICNT = 001104	\$SVPC = 000204
SW07 = 000200	TST3 = 020734	\$ATY1 = 037214	\$ILLUP = 040670	\$SWR = 177400
SW08 = 000400	TST30 = 025636	\$ATY3 = 037222	\$INTAG = 001135	\$SWREG = 001250
SW09 = 001000	TST31 = 026004	\$ATY4 = 037232	\$ITEMB = 001114	\$SWRMK = 000000
SW1 = 000002	TST32 = 026300	\$AUTOB = 001134	\$LF = 001224	\$TBIT = 034144
SW10 = 002000	TST33 = 026510	\$BDADR = 001122	\$LFLG = 037457	\$TESTN = 001232
SW11 = 004000	TST34 = 027072	\$BDDAT = 001126	\$LOOP = 034140	\$TIMES = 001212
SW12 = 010000	TST35 = 027440	\$BELL = 001216	\$LPADR = 001106	\$TKB = 001146
SW13 = 020000	TST36 = 027632	\$BIN = 037534	\$LPERR = 001110	\$TKS = 001144
SW14 = 040000	TST37 = 030234	\$CHARC = 037210	\$MADR1 = 001260	\$TMP0 = 001176
SW15 = 100000	TST4 = 021056	\$CKSWR = 035610	\$MAIL = 001226	\$TMP1 = 001200
SW2 = 000004	TST40 = 030422	\$CLR.T = 034062	\$MAMS1 = 001256	\$TMP2 = 001202
SW3 = 000010	TST41 = 030504	\$CMTAG = 001100	\$MBADR = 000206	\$TMP3 = 001204
SW4 = 000020	TST42 = 030654	\$CM1 = 000006	\$MFLG = 037456	\$TMP4 = 001206
SW5 = 000040	TST43 = 031176	\$CM2 = 000014	\$MNEW = 036570	\$TMP5 = 001210
SW6 = 000100	TST44 = 032044	\$CM3 = 000006	\$MSGAD = 001242	\$TN = 000050
SW7 = 000200	TST45 = 032724	\$CM4 = 000006	\$MSGLG = 001244	\$TPB = 001152

\$TFFLG 001157	\$TRPAD 040460	\$TYPDS 037764	\$TYPON 037576	\$XTSTR 034164
\$TPS 001150	\$TSTM 000210	\$TYPE 036732	\$TYPOS 037536	\$OFILL 037761
\$TRAP 040424	\$TSTNM 001102	\$TYPEC 037144	\$UNIT 001240	. 050213
\$TRAP2 040446	\$TTYIN 036530	\$TYPEX 037212	\$UNITM 000214	.\$X - 000204
\$TRP = 000015	\$TYPBN 037462	\$TYPOC 037562	\$USWR 001252	

. ABS. 050213 000

ERRORS DETECTED: 0

MULE:CJKDAB,MULE:CJKDAB.SEQ/SOL/NL:TOC-CJKDAB.P11  
 RUN-TIME: 102 58 1 SECONDS  
 RUN-TIME RATIO: 527/162=3.2  
 CORE USED: 32K (63 PAGES)