

KTF11-AA

KTF11-AA DIAG
CJKDADO

AH-F137D-MC
FICHE 1 OF 1

AUG 1981
COPYRIGHT © 79-81
MADE IN USA



.REM @

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

IDENTIFICATION

PRODUCT CODE: AC-F138D-MC
PRODUCT NAME: CJKDADO KTF11-AA MEMORY MANAGEMENT DIAGNOSTIC
DATE: JANUARY, 1981
MAINTAINER: DIAGNOSTIC PROGRAMMING

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) 1979,1981 BY DIGITAL EQUIPMENT CORPORATION

42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70

PROGRAM HISTORY

DATE	REVISION	REASON FOR REVISION
JANUARY, 1979	A	FIRST RELEASE
JUNE, 1979	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.
NOVEMBER, 1979	C	CORRECTIONS WERE MADE TO THE MULTI-TESTER SUPPORT CODE. ALSO CODE WAS ADDED TO ALLOW PROGRAM OPERATION WHILE LINE CLOCK IS INTERRUPTING.
JANUARY, 1981	D	EXPANDED 'READ AND WRITE WHILE IN RELOCATE MODE' TEST TO ALLOW UP TO 1.92MW OF MEMORY TO BE TESTED. CORRECTED BUGS IN THE 'RELOCATION AND ADDER' TESTS TO ALLOW 22 BIT ADDRESSING TO BE TURNED ON AND TESTED. REMOVED RESET INSTRUCTIONS FROM TESTS WHERE THEY WERE NOT REQUIRED. PUT IN CODE TO SKIP TESTS AFTER FIRST PASS IN APT MODE THAT REQUIRED RESETS FOR TESTING.

71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104

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

105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160

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/MTPJ INSTRUCTIONS ARE DONE.

1.2 REQUIREMENTS

A KDF11 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

161
 162
 163
 164
 165
 166
 167
 168
 169
 170
 171
 172
 173
 174
 175
 176
 177
 178
 179
 180
 181
 182
 183
 184
 185
 186
 187
 188
 189
 190
 191
 192
 193
 194
 195
 196
 197
 198
 199
 200
 201
 202
 203
 204
 205
 206
 207
 208
 209
 210
 211
 212
 213
 214
 215
 216

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

217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272

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 ^U HAD BEEN TYPED.

NOTE: RECOGNITION OF A ^G MAY BE HAMPERED BY

273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328

----- 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

329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384

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

385
 386
 387
 388
 389
 390
 391
 392
 393
 394
 395
 396
 397
 398
 399
 400
 401
 402
 403
 404
 405
 406
 407
 408
 409
 410
 411
 412
 413
 414
 415
 416
 417
 418
 419
 420
 421
 422
 423
 424
 425
 426
 427
 428
 429
 430
 431
 432
 433
 434
 435
 436
 437
 438
 439
 440

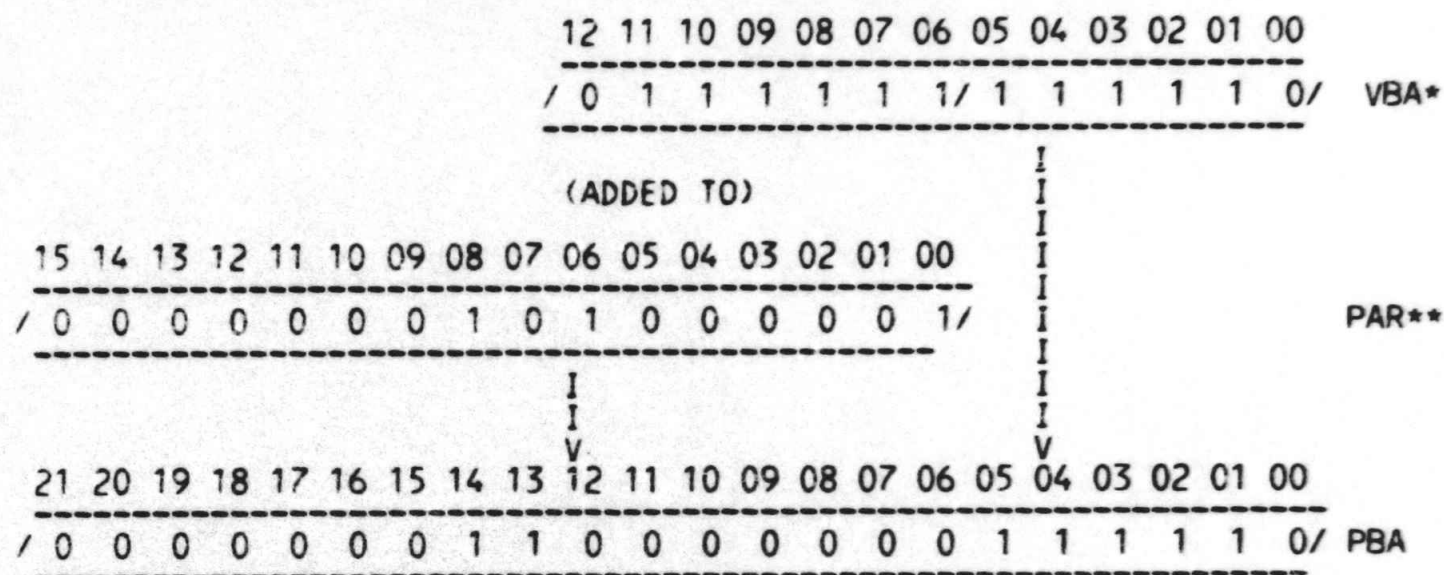
REGISTER (SEE SECTION 2.4).

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 60000 AND 67600 OF EACH 16K GROUP UP TO THE MAXIMUM MEMORY ON THE SYSTEM. THE FIRST LOCATION OF EACH BLOCK(32 WORDS) IS WRITTEN AND READ. SEE TEST #25 IN THE LISTING FOR MORE DETAILS.

<u>CONST.</u>	<u>MAX. MEM.</u>	<u>CONST.</u>	<u>MAX. MEM.</u>	<u>CONST.</u>	<u>MAX. MEM.</u>
0/600	16K	50600	656K	120600	1296K
1600	32K	51600	672K	121600	1312K

441	2600	48K	52600	688K	122600	1328K
442	3600	64K	53600	704K	123600	1344K
443	4600	80K	54600	720K	124600	1360K
444	5600	96K	55600	736K	125600	1376K
445	6600	112K	56600	752K	126600	1392K
446	7600	128K	57600	768K	127600	1408K
447	10600	144K	60600	784K	130600	1424K
448	11600	160K	61600	800K	131600	1440K
449	12600	176K	62600	816K	132600	1456K
450	13600	192K	63600	832K	133600	1472K
451	14600	208K	64600	848K	134600	1488K
452	15600	224K	65600	864K	135600	1504K
453	16600	240K	66600	880K	136600	1520K
454	17600	256K	67600	896K	137600	1536K
455	20600	272K	70600	912K	140600	1552K
456	21600	288K	71600	928K	141600	1568K
457	22600	304K	72600	944K	142600	1584K
458	23600	320K	73600	960K	143600	1600K
459	24600	336K	74600	976K	144600	1616K
460	25600	352K	75600	992K	145600	1632K
461	26600	368K	76600	1008K	146600	1648K
462	27600	384K	77600	1024K	147600	1664K
463	30600	400K	100600	1040K	150600	1680K
464	31600	416K	101600	1056K	151600	1696K
465	32600	432K	102600	1072K	152600	1712K
466	33600	448K	103600	1088K	153600	1728K
467	34600	464K	104600	1104K	154600	1744K
468	35600	480K	105600	1120K	155600	1760K
469	36600	496K	106600	1136K	156600	1776K
470	37600	512K	107600	1152K	157600	1792K
471	40600	528K	110600	1168K	160600	1808K
472	41600	544K	111600	1184K	161600	1824K
473	42600	560K	112600	1200K	162600	1840K
474	43600	576K	113600	1216K	163600	1856K
475	44600	592K	114600	1232K	164600	1872K
476	45600	608K	115600	1248K	165600	1888K
477	46600	624K	116600	1264K	166600	1904K
478	47600	640K	117600	1280K	167600	1920K

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

496

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

497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552

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 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'.

```
553 .TITLE CJKDADO KTF11-AA MMU DIAG
554 :*COPYRIGHT (C) AUGUST, 1980
555 :*DIGITAL EQUIPMENT CORP.
556 :*MAYNARD, MASS. 01754
557 :*
558 :*
559 :*
560 :*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
561 :*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
562 :*
563 .SBTTL OPERATIONAL SWITCH SETTINGS
564 :*
565 :*      SWITCH          USE
566 :*      -----
567 :*      15             HALT ON ERROR
568 :*      14             LOOP ON TEST
569 :*      13             INHIBIT ERROR TYPEOUTS
570 :*      12             INHIBIT TRACE TRAP
571 :*      11             INHIBIT ITERATIONS
572 :*      10             BELL ON ERROR
573 :*      9              LOOP ON ERROR
574 :*      8              LOOP ON TEST IN SWR<7:0>
575 .SBTTL BASIC DEFINITIONS
576 :*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
577      001100 STACK= 1100
578 :*EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
579 :*EQUIV IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
580
581 :*MISCELLANEOUS DEFINITIONS
582 :*
583      000011 HT= 11      ;;CODE FOR HORIZONTAL TAB
584      000012 LF= 12      ;;CODE FOR LINE FEED
585      000015 CR= 15      ;;CODE FOR CARRIAGE RETURN
586      000200 CRLF= 200   ;;CODE FOR CARRIAGE RETURN-LINE FEED
587      177776 PS= 177776  ;;PROCESSOR STATUS WORD
588 :*EQUIV PS,PSW
589      177774 STKLMT= 177774 ;;STACK LIMIT REGISTER
590      177772 PIRQ= 177772  ;;PROGRAM INTERRUPT REQUEST REGISTER
591      177570 DSWR= 177570  ;;HARDWARE SWITCH REGISTER
592      177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
593
594 :*GENERAL PURPOSE REGISTER DEFINITIONS
595 :*
596      000000 R0= %0      ;;GENERAL REGISTER
597      000001 R1= %1      ;;GENERAL REGISTER
598      000002 R2= %2      ;;GENERAL REGISTER
599      000003 R3= %3      ;;GENERAL REGISTER
600      000004 R4= %4      ;;GENERAL REGISTER
601      000005 R5= %5      ;;GENERAL REGISTER
602      000006 R6= %6      ;;GENERAL REGISTER
603      000007 R7= %7      ;;GENERAL REGISTER
604      000006 SP= %6      ;;STACK POINTER
605      000007 PC= %7      ;;PROGRAM COUNTER
606
607 :*PRIORITY LEVEL DEFINITIONS
608      000000 PR0= 0      ;;PRIORITY LEVEL 0
609      000040 PR1= 40     ;;PRIORITY LEVEL 1
```

609	000100	PR2=	100	::PRIORITY LEVEL 2
610	000140	PR3=	140	::PRIORITY LEVEL 3
611	000200	PR4=	200	::PRIORITY LEVEL 4
612	000240	PR5=	240	::PRIORITY LEVEL 5
613	000300	PR6=	300	::PRIORITY LEVEL 6
614	000340	PR7=	340	::PRIORITY LEVEL 7

615
616 :*'SWITCH REGISTER' SWITCH DEFINITIONS

617	100000	SW15=	100000
618	040000	SW14=	40000
619	020000	SW13=	20000
620	010000	SW12=	10000
621	004000	SW11=	4000
622	002000	SW10=	2000
623	001000	SW09=	1000
624	000400	SW08=	400
625	000200	SW07=	200
626	000100	SW06=	100
627	000040	SW05=	40
628	000020	SW04=	20
629	000010	SW03=	10
630	000004	SW02=	4
631	000002	SW01=	2
632	000001	SW00=	1
633		.EQUIV	SW09,SW9
634		.EQUIV	SW08,SW8
635		.EQUIV	SW07,SW7
636		.EQUIV	SW06,SW6
637		.EQUIV	SW05,SW5
638		.EQUIV	SW04,SW4
639		.EQUIV	SW03,SW3
640		.EQUIV	SW02,SW2
641		.EQUIV	SW01,SW1
642		.EQUIV	SW00,SW0

643
644 :*DATA BIT DEFINITIONS (BIT00 TO BIT15)

645	100000	BIT15=	100000
646	040000	BIT14=	40000
647	020000	BIT13=	20000
648	010000	BIT12=	10000
649	004000	BIT11=	4000
650	002000	BIT10=	2000
651	001000	BIT09=	1000
652	000400	BIT08=	400
653	000200	BIT07=	200
654	000100	BIT06=	100
655	000040	BIT05=	40
656	000020	BIT04=	20
657	000010	BIT03=	10
658	000004	BIT02=	4
659	000002	BIT01=	2
660	000001	BIT00=	1
661		.EQUIV	BIT09,BIT9
662		.EQUIV	BIT08,BIT8
663		.EQUIV	BIT07,BIT7
664		.EQUIV	BIT06,BIT6

```
665 .EQUIV BIT05,BIT5
666 .EQUIV BIT04,BIT4
667 .EQUIV BIT03,BIT3
668 .EQUIV BIT02,BIT2
669 .EQUIV BIT01,BIT1
670 .EQUIV BIT00,BIT0
671
672 ;*BASIC 'CPU' TRAP VECTOR ADDRESSES
673 000004 ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
674 000010 RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
675 000014 TBITVEC=14 ;: 'T' BIT
676 000014 TRTVEC= 14 ;:TRACE TRAP
677 000014 BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)
678 000020 IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
679 000024 PWRVEC= 24 ;:POWER FAIL
680 000030 EMTVEC= 30 ;:EMULATOR TRAP (EMT) **ERROR**
681 000034 TRAPVEC=34 ;:'TRAP' TRAP
682 000060 TKVEC= 60 ;:TTY KEYBOARD VECTOR
683 000064 TPVEC= 64 ;:TTY PRINTER VECTOR
684 000240 PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR
685 .SBTTL MEMORY MANAGEMENT DEFINITIONS
686
687 ;*KT11 VECTOR ADDRESS
688
689 000250 MMVEC= 250
690
691 ;*KT11 STATUS REGISTER ADDRESSES
692
693 177572 SR0= 177572
694 177574 SR1= 177574
695 177576 SR2= 177576
696 172516 SR3= 172516
697
698 ;*USER 'I' PAGE DESCRIPTOR REGISTERS
699
700 177600 UIPDR0= 177600
701 177602 UIPDR1= 177602
702 177604 UIPDR2= 177604
703 177606 UIPDR3= 177606
704 177610 UIPDR4= 177610
705 177612 UIPDR5= 177612
706 177614 UIPDR6= 177614
707 177616 UIPDR7= 177616
708
709 ;*USER 'I' PAGE ADDRESS REGISTERS
710
711 177640 UIPAR0= 177640
712 177642 UIPAR1= 177642
713 177644 UIPAR2= 177644
714 177646 UIPAR3= 177646
715 177650 UIPAR4= 177650
716 177652 UIPAR5= 177652
717 177654 UIPAR6= 177654
718 177656 UIPAR7= 177656
719
720 ;*KERNEL 'I' PAGE DESCRIPTOR REGISTERS
```

```
721
722      172300      KIPDR0= 172300
723      172302      KIPDR1= 172302
724      172304      KIPDR2= 172304
725      172306      KIPDR3= 172306
726      172310      KIPDR4= 172310
727      172312      KIPDR5= 172312
728      172314      KIPDR6= 172314
729      172316      KIPDR7= 172316
730
731      ;*KERNEL 'I' PAGE ADDRESS REGISTERS
732
733      172340      KIPAR0= 172340
734      172342      KIPAR1= 172342
735      172344      KIPAR2= 172344
736      172346      KIPAR3= 172346
737      172350      KIPAR4= 172350
738      172352      KIPAR5= 172352
739      172354      KIPAR6= 172354
740      172356      KIPAR7= 172356
741
742      .EQUIV SP,KSP
743      .EQUIV SP,USP
744      .EQUIV BIT4,TBIT
745      .EQUIV BIT6,WBIT
746      001100      KERSTK= STACK
747      000700      USESTK= STACK-200
748
749      ;*ADDITIONAL DEFINITIONS
750      ;*
751
752      .SBTTL TRAP CATCHER
753
754      .=0
755      000000
756      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
757      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
758      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
759      .=174
760      000174      000000      DISPREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
761      000176      000000      SWREG: .WORD 0      ;;SOFTWARE SWITCH REGISTER
762
763      000200      000137      020000      .SBTTL STARTING ADDRESS(ES)
764      .SBTTL JMP @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
765      .SBTTL ACT11 HOOKS
766
767      ;:*****
768      ;:HOOKS REQUIRED BY ACT11
769      .SSVPC=.      ;:SAVE PC
770      000046      034166      .=46      SENDAD      ;;1)SET LOC.46 TO ADDRESS OF SENDAD IN .SEOP
771      000052      000052      .=52
772      000052      000000      .WORD 0      ;;2)SET LOC.52 TO ZERO
773      000204      000204      .=$SVPC      ;; RESTORE PC
774      .SBTTL APT PARAMETER BLOCK
775
776      ;:*****
```



```
777 ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
778 ;*****
779      000204      .SX=      ;;SAVE CURRENT LOCATION
780      000024      =24      ;;SET POWER FAIL TO POINT TO START OF PROGRAM
781 000024      000200      200      ;;FOR APT START UP
782      000044      =44      ;;POINT TO APT INDIRECT ADDRESS PNTR.
783 000044      000204      $APTHDR ;;POINT TO APT HEADER BLOCK
784      000204      =.SX      ;;RESET LOCATION COUNTER
785 ;*****
786 ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
787 ;INTERFACE SPEC.
788
789 000204      $APTHD:
790 000204      000000      $HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
791 000206      001226      $MBADR: .WORD $MAIL ;;ADDRESS OF APT MAILBOX (BITS 0-15)
792 000210      000014      $STMT: .WORD 14      ;;RUN TIM OF LONGEST TEST
793 000212      000020      $PASTM: .WORD 20      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
794 000214      000005      $UNITM: .WORD 5      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
795 000216      000016      .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

796
797
798
799
800
801
802 001100
803 001100
804 001100 000000
805 001102 000
806 001103 000
807 001104 000000
808 001106 000000
809 001110 000000
810 001112 000000
811 001114 000
812 001115 001
813 001116 000000
814 001120 000000
815 001122 000000
816 001124 000000
817 001126 000000
818 001130 000000
819 001132 000000
820 001134 000
821 001135 000
822 001136 000000
823 001140 177570
824 001142 177570
825 001144 177560
826 001146 177562
827 001150 177564
828 001152 177566
829 001154 000
830 001155 002
831 001156 012
832 001157 000
833 001160 000000
834
835 001162 000000
836 001164 000000
837 001166 000000
838 001170 000000
839 001172 000000
840 001174 000000
841 001176 000000
842 001200 000000
843 001202 000000
844 001204 000000
845 001206 000000
846 001210 000000
847 001212 000000
848 001214 000000
849 001216 177607 000377
850 001222 077
851 001223 015

.SBTTL COMMON TAGS

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

 .=1100
\$CMTAG: :::START OF COMMON TAGS
 .WORD 0
\$TSTNM: .BYTE 0 :::CONTAINS THE TEST NUMBER
\$ERFLG: .BYTE 0 :::CONTAINS ERROR FLAG
\$ICNT: .WORD 0 :::CONTAINS SUBTEST ITERATION COUNT
\$LPADR: .WORD 0 :::CONTAINS SCOPE LOOP ADDRESS
\$LPERR: .WORD 0 :::CONTAINS SCOPE RETURN FOR ERRORS
\$ERTTL: .WORD 0 :::CONTAINS TOTAL ERRORS DETECTED
\$ITEMB: .BYTE 0 :::CONTAINS ITEM CONTROL BYTE
\$ERMAX: .BYTE 1 :::CONTAINS MAX. ERRORS PER TEST
\$ERRPC: .WORD 0 :::CONTAINS PC OF LAST ERROR INSTRUCTION
\$GDADR: .WORD 0 :::CONTAINS ADDRESS OF 'GOOD' DATA
\$BDADR: .WORD 0 :::CONTAINS ADDRESS OF 'BAD' DATA
\$GDDAT: .WORD 0 :::CONTAINS 'GOOD' DATA
\$BDDAT: .WORD 0 :::CONTAINS 'BAD' DATA
 .WORD 0 :::RESERVED--NOT TO BE USED
\$AUTOB: .BYTE 0 :::AUTOMATIC MODE INDICATOR
\$INTAG: .BYTE 0 :::INTERRUPT MODE INDICATOR
 .WORD 0
SWR: .WORD DSWR :::ADDRESS OF SWITCH REGISTER
DISPLAY: .WORD DDISP :::ADDRESS OF DISPLAY REGISTER
\$TKS: 177560 :::TTY KBD STATUS
\$TKB: 177562 :::TTY KBD BUFFER
\$TPS: 177564 :::TTY PRINTER STATUS REG. ADDRESS
\$TPB: 177566 :::TTY PRINTER BUFFER REG. ADDRESS
\$NULL: .BYTE 0 :::CONTAINS NULL CHARACTER FOR FILLS
\$FILLS: .BYTE 2 :::CONTAINS # OF FILLER CHARACTERS REQUIRED
\$FILLC: .BYTE 12 :::INSERT FILL CHARS. AFTER A 'LINE FEED'
\$TPFLG: .BYTE 0 ::: "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
\$REGAD: .WORD 0 :::CONTAINS THE ADDRESS FROM
 .WORD 0 :::WHICH (\$REGO) WAS OBTAINED
\$REG0: .WORD 0 :::CONTAINS ((\$REGAD)+0)
\$REG1: .WORD 0 :::CONTAINS ((\$REGAD)+2)
\$REG2: .WORD 0 :::CONTAINS ((\$REGAD)+4)
\$REG3: .WORD 0 :::CONTAINS ((\$REGAD)+6)
\$REG4: .WORD 0 :::CONTAINS ((\$REGAD)+10)
\$REG5: .WORD 0 :::CONTAINS ((\$REGAD)+12)
\$TMP0: .WORD 0 :::USER DEFINED
\$TMP1: .WORD 0 :::USER DEFINED
\$TMP2: .WORD 0 :::USER DEFINED
\$TMP3: .WORD 0 :::USER DEFINED
\$TMP4: .WORD 0 :::USER DEFINED
\$TMP5: .WORD 0 :::USER DEFINED
\$TIMES: 0 :::MAX. NUMBER OF ITERATIONS
\$ESCAPE: 0 :::ESCAPE ON ERROR ADDRESS
\$BELL: .ASCII <207><377><377> :::CODE FOR BELL
\$QUES: .ASCII /?/ :::QUESTION MARK
\$CRLF: .ASCII <15> :::CARRIAGE RETURN

```
852 001224 000012 $LF: .ASCIZ <12> ;;LINE FEED
853 ::*****
854 .SBTTL APT MAILBOX-ETABLE
855 ::*****
856 .EVEN
857 $MAIL: ;;APT MAILBOX
858 001226 $MSGTY: .WORD AMSGTY ;;MESSAGE TYPE CODE
859 001226 000000 $FATAL: .WORD AFATAL ;;FATAL ERROR NUMBER
860 001230 000000 $TESTN: .WORD ATESTN ;;TEST NUMBER
861 001232 000000 $PASS: .WORD APASS ;;PASS COUNT
862 001234 000000 $DEVCT: .WORD ADEVCT ;;DEVICE COUNT
863 001236 000000 $UNIT: .WORD AUNIT ;;I/O UNIT NUMBER
864 001240 000000 $MSGAD: .WORD AMSGAD ;;MESSAGE ADDRESS
865 001242 000000 $MSGLG: .WORD AMSGLG ;;MESSAGE LENGTH
866 001244 000000 $ETABLE: ;;APT ENVIRONMENT TABLE
867 001246 $ENV: .BYTE AENV ;;ENVIRONMENT BYTE
868 001246 000 $ENVM: .BYTE AENVM ;;ENVIRONMENT MODE BITS
869 001247 000 $SWREG: .WORD ASWREG ;;APT SWITCH REGISTER
870 001250 000000 $USWR: .WORD AUSWR ;;USER SWITCHES
871 001252 000000 $CPUOP: .WORD ACPUOP ;;CPU TYPE,OPTIONS
872 001254 000000 *
873 * BITS 15-11=CPU TYPE
874 * 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
875 * 11/70=06,PDQ=07,Q=10
876 * BIT 10=REAL TIME CLOCK
877 * BIT 9=FLOATING POINT PROCESSOR
878 * BIT 8=MEMORY MANAGEMENT
879 001256 000 $MAMS1: .BYTE AMAMS1 ;;HIGH ADDRESS,M.S. BYTE
880 001257 000 $MTYP1: .BYTE AMTYP1 ;;MEM. TYPE,BLK#1
881 * MEM.TYPE BYTE -- (HIGH BYTE)
882 * 900 NSEC CORE=001
883 * 300 NSEC BIPOLAR=002
884 * 500 NSEC MOS=003
885 001260 000000 $MADR1: .WORD AMADR1 ;;HIGH ADDRESS,BLK#1
886 * MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
887 001262 $ETEND:
888 .MEXIT
889
890 001262 000000 TESTNO: .WORD 0 ;:HOLDS TEST NUMBER FOR TYPEOUTS
891 001264 000000 WASR6: .WORD 0 ;:USED TO STORE THE STACK POINTER AFTER A TRAP
892 001266 000000 TRAPPC: .WORD 0 ;:USED TO STORE THE PC OF A TRAP OR ABORT
893 001270 000000 TRAPPS: .WORD 0 ;:USED TO STORE THE PS OF A TRAP OR ABORT
894 001272 000000 WASSR0: .WORD 0 ;:USED TO STORE CONTENTS OF SRO
895 001274 000000 WASSR2: .WORD 0 ;:USED TO STORE CONTENTS OR SR2
896 001276 000000 TBITPS: .WORD 0 ;:SAVES THE PSW THAT MAY HAVE ITS T-BIT ON
897 001300 000000 ANDADR: .WORD 0 ;:HOLDS RESULT OF ADDRESSES BEING AND-ED
898 001302 000000 ORADR: .WORD 0 ;:HOLDS RESULT OF ADDRESSES BEING OR-ED
899 001304 000000 TONUM: .WORD 0 ;:HOLDS NUMBER OF TIME-OUTS
900 001306 000000 VIRT1: .WORD 0 ;:HOLDS VIRTUAL ADDRESS TO BE CONVERTED
901 001310 000000 VIRT2: .WORD 0 ;:
902 001312 000000 PBALO: .WORD 0 ;:HOLDS BITS <15:00> OF PHYSICAL ADDRESS
903 001314 000000 PBAHI: .WORD 0 ;:HOLDS BITS <17:16> OF PHYSICAL ADDRESS
```

```

904      .SBTTL  ERROR POINTER TABLE
905
906      ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
907      ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
908      ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
909      ;*NOTE1:      IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
910      ;*NOTE2:      EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
911
912      ;*      EM      ;;POINTS TO THE ERROR MESSAGE
913      ;*      DH      ;;POINTS TO THE DATA HEADER
914      ;*      DT      ;;POINTS TO THE DATA
915      ;*      DF      ;;POINTS TO THE DATA FORMAT
916
917
918      001316      $ERRTB:
919
920      ;*ITEM 1
921      001316      041162      EM1      ;UNEXPECTED CPU TRAP TO LOC. 004
922      001320      044253      DH1      ;OLD PC OLD PSW R6 WAS TESTNO ERRORPC
923      001322      047460      DT1      ;TRAPPC, TRAPPS, WASR6, TESTNO, $ERPPC, 0
924      001324      050270      DF1      ;0,0,0,0,0
925
926      ;*ITEM 2
927      001326      041222      EM2      ;UNEXPECTED MEM. MGMT. TRAP TO LOC. 250
928      001330      044323      DH2      ;OLD PC OLD PSW R6 WAS SRO SR2 TESTNO ERRORPC
929      001332      047474      DT2      ;TRAPPC, TRAPPS, WASR6, WASSR0, WASSR2, TESTNO, $ERRPC,
930      001334      050275      DF2      ;0,0,0,0,0,0,0
931
932      ;*ITEM 3
933      001336      041271      EM3      ;PRIORITY BITS SET WRONG IN PSW
934      001340      044413      DH3      ;WROTE READ TESTNO ERRORPC
935      001342      047514      DT3      ;$REG0,$REG1,TESTNO,$ERRPC,0
936      001344      050304      DF3      ;0,0,0,0
937
938      ;*ITEM 4
939      001346      041330      EM4      ;MODE BITS SET WRONG IN PSW
940      001350      044413      DH3      ;WROTE READ TESTNO ERRORPC
941      001352      047514      DT3      ;$REG0,$REG1,TESTNO,$ERRPC,0
942      001354      050304      DF3      ;0,0,0,0
943
944      ;*ITEM 5
945      001356      041363      EM5      ;DUAL ADDRESSING BETWEEN HI&LO BYTES OF PSW
946      001360      044413      DH3      ;WROTE READ TESTNO ERRORPC
947      001362      047514      DT3      ;$REG0,$REG1,TESTNO,$ERRPC,0
948      001364      050304      DF3      ;0,0,0,0
949
950      ;*ITEM 6
951      001366      041436      EM6      ;KERNEL R6 CHANGED BY WRITING USER R6
952      001370      044413      DH3      ;WROTE READ TESTNO ERRORPC
953      001372      047514      DT3      ;$REG0,$REG1,TESTNO,$ERRPC,0
954      001374      050304      DF3      ;0,0,0,0
955
956      ;*ITEM 7
957      001376      041503      EM7      ;A MEMORY MGMT. REG. TIMED OUT
958      001400      044453      DH7      ;ADDRESS TESTNO ERRORPC
959      001402      047526      DT7      ;$REG0,TESTNO,$ERRPC,0
  
```

960	001404	050310	DF7	:0,0,0
961				
962			:*ITEM 10	
963	001406	041541	EM10	:SUMMARY OF MEM. MGMT. REG. TIMEOUTS
964	001410	044503	DH10	:REGISTER-ADDRS NUM. OF
965				:AND-ED OR-ED TIMOUTS TESTNO ERRORPC
966	001412	047536	DT10	:ANDADR,ORADR,TONUM,TESTNO,\$ERRPC,0
967	001414	050313	DF10	:0,0,1,0,0
968				
969			:*ITEM 11	
970	001416	041605	EM11	:MEM. MGMT. REG. WOULD NOT CLEAR
971	001420	044603	DH11	:REGISTR READ READ-(BINARY)
972				:ADDRESS (OCTAL) 5432109876543210 TESTNO ERRORPC
973	001422	047552	DT11	:\$REG0,\$REG1,\$REG1,TESTNO,\$ERRPC,0
974	001424	050320	DF11	:0,0,2,0,0
975				
976			:*ITEM 12	
977	001426	041645	EM12	:MEM. MGMT. REG. BITS NOT SET CORRECTLY
978	001430	044723	DH12	:REGISTR WROTE READ READ
979				:ADDRESS (OCTAL) (OCTAL) (BINARY) TESTNO ERRORPC
980	001432	047566	DT12	:\$REG0,\$REG1,\$REG2,\$REG2,TESTNO,\$ERRPC,0
981	001434	050325	DF12	:0,0,0,2,0,0
982				
983			:*ITEM 13	
984	001436	041714	EM13	:SR0 EFFECTED BY WRITE TO PSW
985	001440	045063	DH13	:READ TESTNO ERRORPC
986	001442	047604	DT13	:\$REG0,TESTNO,\$ERRPC,0
987	001444	050333	DF13	:0,0,0
988				
989			:*ITEM 14	
990	001446	041751	EM14	:SR1 DID NOT READ ALL ZEROS
991	001450	045063	DH13	:READ TESTNO ERRORPC
992	001452	047604	DT13	:\$REG0,TESTNO,\$ERRPC,0
993	001454	050333	DF13	:0,0,0
994				
995			:*ITEM 15	
996	001456	042004	EM15	:DUAL ADDRESSING BETWEEN BYTES OF PAR OR PDR
997	001460	044723	DH12	:REGISTER WROTE READ READ
998				:ADDRESS (OCTAL) (OCTAL) (BINARY) TESTNO ERRORPC
999	001462	047566	DT12	:\$REG0,\$REG1,\$REG2,\$REG2,TESTNO,\$ERRPC,0
1000	001464	050325	DF12	:0,0,0,2,0,0
1001				
1002			:*ITEM 16	
1003	001466	042060	EM16	:DUAL ADDRESSING BETWEEN PAR-PDR'S
1004	001470	045113	DH16	:PAR-PDR PAR-PDR
1005				:CLEARED EFFECTD EXPECTD RECEIVD TESTNO ERRORPC
1006	001472	047614	DT16	:\$REG0,\$REG1,\$REG5,\$REG2,TESTNO,\$ERRPC,0
1007	001474	050336	DF16	:0,0,0,0,0,0
1008				
1009			:*ITEM 17	
1010	001476	042122	EM17	:PHYS. ADDR. FORMED READ WRONG
1011	001500	045213	DH17	:PHYSICAL VIRTUAL
1012				:ADDRESS ADDRESS KIPAR4 TESTNO ERRORPC
1013	001502	047632	DT17	:\$BAL0,VIRT1,\$REG4,TESTNO,\$ERRPC,0
1014	001504	050344	DF17	:3,0,0,0,0
1015				

Line	Code	Address	Item	Code	Description
1016			*ITEM 20		
1017	001506	042160		EM20	:PHYS. ADDR. FORMED READ WRONG IN RELOCATE MODE
1018	001510	045303		DH20	:PHYSICL PAR 4 PAR 5
1019					:ADDRESS VBA VBA PAR 4 PAR 5 PSW TESTNO
1020	001512	047646		DT20	:PBA0,VIRT1,VIRT2,\$REG4,\$REG5,\$TMP0,TESTNO,\$ERRPC,0
1021	001514	050351		DF20	:3,0,0,0,0,0,0,0
1022					
1023			*ITEM 21		
1024	001516	042232		EM21	:W-BIT DID NOT GET SET IN PDR
1025	001520	045431		DH21	:PDR VIRTUAL
1026					:TESTED ADDRESS TESTNO ERRORPC
1027	001522	047670		DT21	:\$REG5,\$REG3,TESTNO,\$ERRPC,0
1028	001524	050361		DF21	:0,0,0,0
1029					
1030			*ITEM 22		
1031	001526	042267		EM22	:W-BIT SET IN MORE THAN ONE PDR
1032	001530	045511		DH22	:PDR IN PDR VIRTUAL
1033					:ERROR TESTED ADDRESS TESTNO ERRORPC
1034	001532	047702		DT22	:\$REG0,\$REG5,\$REG3,TESTNO,\$ERRPC,0
1035	001534	050365		DF22	:0,0,0,0,0
1036					
1037			*ITEM 23		
1038	001536	042326		EM23	:W-BIT NOT CLEARED BY WRITING TO PDR
1039	001540	045610		DH23	:PDR TESTNO ERRORPC
1040	001542	047716		DT23	:\$REG5,TESTNO,\$ERRPC,0
1041	001544	050372		DF23	:0,0,0
1042					
1043			*ITEM 24		
1044	001546	042372		EM24	:WRITING SRO SET W-BIT IN KIPDR7
1045	001550	045640		DH24	:PDR WAS EXPECTD TESTNO ERRORPC
1046	001552	047726		DT24	:\$REG2,\$REG1,TESTNO,\$ERRPC,0
1047	001554	050375		DF24	:0,0,0,0
1048					
1049			*ITEM 25		
1050	001556	042432		EM25	:W-BIT GOT SET DURING TIMEOUT ABORT
1051	001560	045640		DH24	:PDR WAS EXPECTD TESTNO ERRORPC
1052	001562	047726		DT24	:\$REG2,\$REG1,TESTNO,\$ERRPC,0
1053	001564	050375		DF24	:0,0,0,0
1054					
1055			*ITEM 26		
1056	001566	042475		EM26	:MEMORY MGMT. ACCESS ABORT DID NOT OCCUR
1057	001570	045700		DH26	:PDR 4 PSW TESTNO ERRORPC
1058	001572	047740		DT26	:\$REG2,\$TMP0,TESTNO,\$ERRPC,0
1059	001574	050375		DF24	:0,0,0,0
1060					
1061			*ITEM 27		
1062	001576	042545		EM27	:ACCESS ERROR DID NOT ABORT INSTRUCTION
1063	001600	045700		DH26	:PDR 4 PSW TESTNO ERRORPC
1064	001602	047740		DT26	:\$REG2,\$TMP0,TESTNO,\$ERRPC,0
1065	001604	050375		DF24	:0,0,0,0
1066					
1067			*ITEM 30		
1068	001606	042614		EM30	:SRO DID NOT REPORT ACCESS ERROR CORRECTLY
1069	001610	045740		DH30	:SRO WAS EXPECTD PDR 4 PSW TESTNO ERRORPC
1070	001612	047752		DT30	:WASSRO,\$REG3,\$REG2,\$TMP0,TESTNO,\$ERRPC,0
1071	001614	050401		DF30	:0,0,0,0,0,0

1072					
1073					
1074	001616	042666	;*ITEM 31	EM31	:SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR.
1075	001620	046020		DH31	:SR2 WAS EXPECTD PDR 4 PSW TESTNO ERRORPC
1076	001622	047770		DT31	:WASSR2,\$REG4,\$REG2,\$TMP0,TESTNO,\$ERRPC,0
1077	001624	050401		DF30	:0,0,0,0,0,0
1078					
1079			;*ITEM 32	EM32	:PAGE LGTH. ABORT OCCURRED WHEN IT SHOULDN'T HAVE
1080	001626	042733		DH32	:V.B.A. KIPDR4 SR0 WAS SR2 WAS TESTNO ERRORPC
1081	001630	046100		DT32	:\$REG0,\$REG4,WASSR0,WASSR2,TESTNO,\$ERRPC,0
1082	001632	050006		DF30	:0,0,0,0,0,0
1083	001634	050401			
1084					
1085			;*ITEM 33	EM33	:PAGE LGTH. ABORT DID NOT OCCUR WHEN IT SHOULD HAVE
1086	001636	043014		DH33	:V.B.A. KIPDR4 TESTNO ERRORPC
1087	001640	046160		DT33	:\$REG0,\$REG4,TESTNO,\$ERRPC,0
1088	001642	050024		DF24	:0,0,0,0
1089	001644	050375			
1090					
1091			;*ITEM 34	EM34	:SR0 DID NOT REPORT PAGE LGTH. ABORT CORRECTLY
1092	001646	043077		DH34	:V.B.A. KIPDR4 SR0 WAS EXPECTD TESTNO ERRORPC
1093	001650	046220		DT34	:\$REG0,\$REG4,WASSR0,\$REG2,TESTNO,\$ERRPC,0
1094	001652	050036		DF30	:0,0,0,0,0,0
1095	001654	050401			
1096			;*ITEM 35	EM31	:SR2 DID NOT LOCKUP CORRECT VIRUAL ADDR.
1097	001656	042666		DH35	:V.B.A. KIPDR4 SR2 WAS EXPECTD TESTNO ERRORPC
1098	001660	046300		DT35	:\$REG0,\$REG4,WASSR2,\$REG3,TESTNO,\$ERRPC,0
1099	001662	050054		DF30	:0,0,0,0,0,0
1100	001664	050401			
1101					
1102			;*ITEM 36	EM31	:SR2 DID NOT LOCKUP CORRECT VIRUAL ADDR.
1103	001666	042666		DH36	:SR2 WAS EXPECTD TESTNO ERRORPC
1104	001670	046360		DT36	:WASSR2,\$REG1,TESTNO,\$ERRPC,0
1105	001672	050072		DF24	:0,0,0,0
1106	001674	050375			
1107					
1108			;*ITEM 37	EM37	:SR0 OR SR2 CHANGED BY A SECOND ABORT
1109	001676	043155		DH37	:FIRST ABORT SECOND ABORT
1110	001700	046420			:SR0 WAS SR2 WAS SR0 WAS SR2 WAS TESTNO ERRORPC
1111				DT37	:\$TMP0,\$TMP2,WASSR0,WASSR2,TESTNO,\$ERRPC,0
1112	001702	050104		DF30	:0,0,0,0,0,0
1113	001704	050401			
1114					
1115			;*ITEM 40	EM40	:SR0 OR SR2 WAS NOT 'RESET' BY A RESET
1116	001706	043222		DH40	:SR0 WAS SR2 WAS TESTNO ERRORPC
1117	001710	046535		DT40	:WASSR0,WASSR2,TESTNO,\$ERRPC,0
1118	001712	050122		DF24	:0,0,0,0
1119	001714	050375			
1120					
1121			;*ITEM 41	EM41	:SR2 NOT TRACKING CORRECTLY
1122	001716	043271		DH36	:SR2 WAS EXPECTD TESTNO ERROPC
1123	001720	046360		DT36	:WASSR2,\$REG1,TESTNO,\$ERRPC,0
1124	001722	050072		DF24	:0,0,0,0
1125	001724	050375			
1126					
1127			;*ITEM 42		

1128	001726	043324	EM42	:DID NOT TRAP THRU KERNEL SPACE
1129	001730	046575	DH42	:PSW WAS R6 WAS TESTNO ERRORPC
1130	001732	050134	DT42	:\$REG1,\$REG2,TESTNO,\$ERRPC,0
1131	001734	050375	DF24	:0,0,0,0
1132				
1133			:*ITEM 43	
1134	001736	043363	EM43	:KT ERROR NOT SERVICED ON TIMEOUT ERROR
1135	001740	045610	DH23	:PDR TESTNO ERRORPC
1136	001742	047716	DT23	:\$REG5,TESTNO,\$ERRPC,0
1137	001744	050372	DF23	:0,0,0
1138				
1139			:*ITEM 44	
1140	001746	043432	EM44	:SRO OR SR2 CHANGED BY TIMEOUT ERROR
1141	001750	046635	DH44	:EXPECTED RECEIVED
1142				:SRO SR2 SRO WAS SR2 WAS TESTNO ERRORPC
1143	001752	050146	DT44	:\$REG0,\$REG1,WASSRO,WASSR2,TESTNO,\$ERRPC,0
1144	001754	050401	DF30	:0,0,0,0,0,0
1145				
1146			:*ITEM 45	
1147	001756	043476	EM45	:ERROR DURING 'DOUBLE ERROR' (KT & ODD ADDR.)
1148	001760	046747	DH45	:EXPECTED:
1149				:PSW PC SRO SR2
1150				:170017 (3\$+4) 020147 (3\$)
1151				:RECEIVED
1152				:PSW PC SRO SR2 TESTNO ERRORPC
1153	001762	050164	DT45	:\$REG1,\$REG3,WASSRO,WASSR2,TESTNO,\$ERRPC,0
1154	001764	050401	DF30	:0,0,0,0,0,0
1155				
1156			:*ITEM 46	
1157	001766	043551	EM46	:MFPI INSTRUCTION PUSHED WRONG DATA
1158	001770	047144	DH46	:DATA DATA
1159				:EXPECTD RECEIVD TESTNO ERRORPC
1160	001772	050202	DT46	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
1161	001774	050407	DF46	:0,0,0,0
1162				
1163			:*ITEM 47	
1164	001776	043614	EM47	:MTPI INSTRUCTION LOADED WRONG DATA
1165	002000	047144	DH46	:DATA DATA
1166				:EXPECTD RECEIVD TESTNO ERRORPC
1167	002002	050202	DT46	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
1168	002004	050407	DF46	:0,0,0,0
1169				
1170			:*ITEM 50	
1171	002006	043657	EM50	:STACK NOT PUSHED BY MFPI-MTPI
1172	002010	047221	DH50	:TESTNO ERRORPC
1173	002012	050214	DT50	:TESTNO,\$ERRPC,0
1174	002014	050413	DF50	:0,0
1175				
1176			:*ITEM 51	
1177	002016	043715	EM51	:KERNEL PAGE ACCESSED INSTEAD OF USER: MFPI-MTPI
1178	002020	047241	DH51	:SRO WAS SR2 WAS TESTNO ERRORPC
1179	002022	050222	DT51	:WASSRO,WASSR2,TESTNO,\$ERRPC,0
1180	002024	050415	DF51	:0,0,0,0
1181				
1182			:*ITEM 52	
1183	002026	043773	EM52	:WRONG PDR'S REFERENCED WHILE IN RELOCATE MODE

Line	Code	Address	Pointer	Description
1184	002030	047301	DH52	:PHYSICL PAR 4
1185				:ADDRESS V.B.A. PAR 4 SRO WAS SR2 WAS PSW TESTNO
1186	002032	050234	DT52	:PBALO,VIRT1,\$REG4,WASSR0,WASSR2,\$TMPO,TESTNO,\$ERRPC,0
1187	002034	050421	DF52	:3,0,0,0,0,0,0,0
1188				
1189	002036	044051	EM53	:MFPD INSTRUCTION PUSHED WRONG DATA
1190	002040	047144	DH46	:DATA DATA
1191				:EXPECTD RECEIVD TESTNO ERRORPC
1192	002042	050202	DT46	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
1193	002044	050407	DF46	:0,0,0,0
1194				
1195				
1196	002046	044114	EM54	:STACK NOT PUSHED BY MFPD-MTPD
1197	002050	047221	DH50	:TESTNO ERRORPC
1198	002052	050214	DT50	:TESTNO,\$ERRPC,0
1199	002054	050413	DF50	:0,0
1200				
1201				
1202	002056	044152	EM55	:PAR OR PDR WAS CHANGED BY A RESET
1203	002060	044603	DH11	:REGISTR READ READ-(BINARY)
1204				:ADDRESS (OCTAL) 5432109876543210 TESTNO ERRORPC

1205 002062 047552
1206 002064 050320
1207
1208
1209 002066 044210
1210 002070 047417
1211 002072 050256
1212 002074 050431
1213
1214

DT11
DF11
:*ITEM 56
EM56
DH56
DT56
DF56

;\$REG0,\$REG1,\$REG1,TESTNO,\$ERRPC,0
:0,0,2,0,0

;PSW CHANGED BY AN RTI IN USER MODE
;PSW WAS EXPECTD TESTNO ERRORPC
;\$REG1,\$REG2,TESTNO,\$ERRPC,0
:0,0,0,0

1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226 002076 005227
1227 002100 177777
1228 002102 001403
1229 002104 005237 001226
1230 002110 000000
1231
1232
1233
1234
1235 002112 012637 001266
1236 002116 012637 001270
1237 002122 010637 001264
1238 002126 104001
1239 002130 012737 177777 002100
1240 002136 013746 001270
1241 002142 013746 001266
1242 002146 000006
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254 002150 005227
1255 002152 177777
1256 002154 001403
1257 002156 005237 001226
1258 002162 000000
1259
1260
1261
1262
1263 002164 012637 001266
1264 002170 012637 001270
1265 002174 010637 001264
1266 002200 013737 177572 001272
1267 002206 013737 177576 001274
1268 002214 042737 160000 177572
1269 002222 104002
1270 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
```

CJKDADO KTF11-AA MMU DIAG
CJKDAD.P11 19-DEC-80 11:05

MACY11 30A(1052) 14-JAN-81 11:36 ^{B 3} PAGE 27
MEMORY MANAGEMENT TRAP HANDLER ROUTINE

SEQ 0027

1271 002232 013746 001270
1272 002236 013746 001266
1273 002242 000006
1274

MOV TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
MOV TRAPPC,-(KSP)
RTI ;RETURN FROM INTERRUPT OR ABORT.

```
1275 .SBTTL
1276 .SBTTL ***** STARTING POINT OF TEST *****
1277 .SBTTL ***** STARTING ADDRESS OF 200 *****
1278 .S=20000
1279
1280 020000
1281
1282 START:
1283 .SBTTL INITIALIZE THE COMMON TAGS
1284 ::CLEAR THE COMMON TAGS ($CMTAG) AREA
1285 MOV #CMTAG,R6 ::FIRST LOCATION TO BE CLEARED
1286 CLR (R6)+ ::CLEAR MEMORY LOCATION
1287 CMP #SWR,R6 ::DONE?
1288 BNE -6 ::LOOP BACK IF NO
1289 MOV #STACK,SP ::SETUP THE STACK POINTER
1290 ::INITIALIZE A FEW VECTORS
1291 MOV #SCOPE,@#IOTVEC ::IOT VECTOR FOR SCOPE ROUTINE
1292 MOV #340,@#IOTVEC+2 ::LEVEL 7
1293 MOV #ERROR,@#EMTVEC ::EMT VECTOR FOR ERROR ROUTINE
1294 MOV #340,@#EMTVEC+2 ::LEVEL 7
1295 MOV #TRAP,@#TRAPVEC ::TRAP VECTOR FOR TRAP CALLS
1296 MOV #340,@#TRAPVEC+2 ::LEVEL 7
1297 MOV #PWRDN,@#PWRVEC ::POWER FAILURE VECTOR
1298 MOV #340,@#PWRVEC+2 ::LEVEL 7
1299 MOV $ENDCT,$EOPCT ::SETUP END-OF-PROGRAM COUNTER
1300 CLR $TIMES ::INITIALIZE NUMBER OF ITERATIONS
1301 CLR $ESCAPE ::CLEAR THE ESCAPE ON ERROR ADDRESS
1302 MOV #1,$ERMAX ::ALLOW ONE ERROR PER TEST
1303 ::INITIALIZE THE 'T-BIT' TRAP VECTOR. THEN LOAD LOCATION '$RTRN', IN
1304 ::THE 'END-OF-PASS' ($EOP) ROUTINE, WITH A 'RTI' OR 'RTT'.
1305 MOV #RTRN,@#TBITVEC ::SET 'T' BIT VECTOR TO $RTRN
1306 MOV #340,@#TBITVEC+2 ::LEVEL 7
1307 MOV #RTI,$RTRN ::SET $RTRN TO A RTI
1308 MOV #65$,@#RESVEC ::TRY TO DO A RTT
1309 CLR -(SP) ::DUMMY PS
1310 MOV #64$,-(SP) ::AND PC
1311 RTT ::TRY THE RTT
1312 MOV #RTT,$RTRN ::RTT IS LEGAL--SET $RTRN TO A RTT
1313 BR 66$
1314 ADD #10,SP ::RTT ILLEGAL--CLEAN OFF THE STACK
1315 MOV #RESVEC+2,@#RESVEC ::RESTORE TRAP CATCHER
1316 CLR $TBIT ::CLEAR 'T' BIT SWITCH
1317 MOV #,$LPADR ::INITIALIZE THE LOOP ADDRESS FOR SCOPE
1318 MOV #,$LPERR ::SETUP THE ERROR LOOP ADDRESS
1319 ::SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1320 ::EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
1321 MOV @#ERRVEC,-(SP) ::SAVE ERROR VECTOR
1322 MOV #67$,@#ERRVEC ::SET UP ERROR VECTOR
1323 MOV #DSWR,SWR ::SETUP FOR A HARDWARE SWITCH REGISTER
1324 MOV #DDISP,DISPLAY ::AND A HARDWARE DISPLAY REGISTER
1325 CMP #-1,@SWR ::TRY TO REFERENCE HARDWARE SWR
1326 BNE 69$ ::BRANCH IF NO TIMEOUT TRAP OCCURRED
1327 ::AND THE HARDWARE SWR IS NOT = -1
1328 BR 68$ ::BRANCH IF NO TIMEOUT
1329 MOV #68$,(SP) ::SET UP FOR TRAP RETURN
1330 RTI
1331 MOV #SWREG,SWR ::POINT TO SOFTWARE SWR
1332 MOV #DISPREG,DISPLAY
```

```

1331 020310 012637 000004      69$:  MOV      (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
1332
1333 020314 005037 001234      CLR      $PASS                ;;CLEAR PASS COUNT
1334 020320 132737 000200 001247  BITB     #APTSIZE,$ENVM      ;;TEST USER SIZE UNDER APT
1335 020326 001403                BEQ      70$                  ;;YES,USE NON-APT SWITCH
1336 020330 012737 001250 001140  MOV      #$$SWREG,$SWR      ;;NO,USE APT SWITCH REGISTER
1337 020336
1338                                70$:
1339 020336 005037 001112      ;; INITIALIZE THE ERROR COUNTER FOR EOP REPORT($ERTTL).
                                CLR      $ERTTL                ;CLEAR ERROR COUNTER
1340                                .SBTTL  TYPE PROGRAM NAME
1341                                ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
1342 020342 005227 177777      INC      #-1                  ;;FIRST TIME?
1343 020346 001046                BNE     71$                   ;;BRANCH IF NO
1344 020350 022737 034166 000042  CMP      #$$ENDAD,@#42      ;;ACT-11?
1345 020356 001442                BEQ     71$                   ;;BRANCH IF YES
1346 020360 104401 020426      TYPE     ,72$                ;;TYPE ASCIZ STRING
1347                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
1348 020364 005737 000042      TST     @#42                 ;;ARE WE RUNNING UNDER XXDP/ACT?
1349 020370 001012                BNE     73$                   ;;BRANCH IF YES
1350 020372 123727 001246 000001  CMPB    $ENV,#1              ;;ARE WE RUNNING UNDER APT?
1351 020400 001406                BEQ     73$                   ;;BRANCH IF YES
1352 020402 023727 001140 000176  CMP     $SWR,#$SWREG        ;;SOFTWARE SWITCH REG SELECTED?
1353 020410 001005                BNE     74$                   ;;BRANCH IF NO
1354 020412 104407                GT$SWR                        ;;GET SOFT-SWR SETTINGS
1355 020414 000403                BR      74$
1356 020416 112737 000001 001134  73$:  MOVVB   #1,$AUTOB          ;;SET AUTO-MODE INDICATOR
1357 020424      74$:
1358 020424 000417                BR      71$                   ;;GET OVER THE ASCIZ
1359                                ;;72$:  .ASCIZ <CRLF>#CJKDADO KTF11-AA MMU DIAG.#<CRLF>
1360 020464      71$:
1361
1362 020464                                RESTRT:
1363 020464 012706 001100      LOOP:  MOV     #STACK,KSP      ;INITIALIZE THE STACK POINTER
1364 020470 012737 002076 000004  MOV     #TIMERR,ERRVEC     ;LOAD CPU SERVICE ROUTINE INTO TRAP VECTOR
1365 020476 012737 000340 000006  MOV     #340,ERRVEC+2     ;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
1366 020504 012737 002150 000250  MOV     #MGMERR,MMVEC     ;LOAD MEMORY MANAGENT ROUTINE INTO VECTOR
1367 020512 012737 000340 000252  MOV     #340,MMVEC+2     ;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
1368 020520 012700 177777      MOV     #-1,R0            ;PUT -1 INTO R0 TO INITIALIZE FLAGS
1369 020524 010037 002100      MOV     R0,TIMFLG        ;INITIALIZE CPU ERROR FLAG
1370 020530 010037 002152      MOV     R0,MGMFLG       ;INITIALIZE MEMORY MANAGEMENT ERROR FLAG
1371 020534 012737 000340 001276  MOV     #340,TBITPS     ;INITIALIZE LOG THAT HOLDS T-BIT PSW
1372 020542 005037 177572      CLR     SRO              ;BE SURE MEM. MGMT IS OFF TO START WITH
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
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429

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

```
1430
1431
1432 020704 000004
1433 020706 012737 020714 001110
1434 020714 005037 177776
1435 020720 012700 000360
1436 020724 110037 177777
1437 020730 013701 177776
1438 020734 042701 007437
1439 020740 000300
1440 020742 020001
1441 020744 001403
1442 020746 005037 177776
1443 020752 104005
1444
1445
1446
1447 020754 012737 020762 001110
1448 020762 005037 177776
1449 020766 012700 000340
1450 020772 110037 177776
1451 020776 013701 177776
1452 021002 042701 007437
1453 021006 020001
1454 021010 001403
1455 021012 005037 177776
1456 021016 104005
1457
1458
1459
1460 021020 012737 020706 001110
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471 021026 000004
1472 021030 005037 177776
1473 021034 012706 001100
1474 021040 012737 140000 177776
1475 021046 012706 000700
1476 021052 005037 177776
1477 021056 022706 001100
1478 021062 001404
1479 021064 012700 001100
1480 021070 010601
1481 021072 104006
1482
1483
1484
1485

:*****
: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
```


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
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541

021074 000004
021076 012737 021140 001110
021104 012737 021202 000004
021112 012700 177572
021116 012701 000003
021122 012737 177777 001300
021130 005037 001302
021134 005037 001304
021140 005710
021142 062700 000002
021146 077104
021150 005737 172516
021154 012737 021076 001110
021162 005737 001304
021166 001401
021170 104010
021172 012737 002076 000004
021200 000414
021202 062706 000004
021206 104007
021210 010002
021212 050237 001302
021216 005102
021220 040237 001300
021224 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. TIMEOUTS
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
```

```
1542 021230 000744 BR 3$ ;BRANCH BACK TO TEST THE NEXT ADDR.
1543
1544 :*****
1545 :*TEST 6 KERNEL PAR'S TIMEOUT TEST
1546 :*
1547 :* THIS TEST ADDRESSES THE EIGHT (8) KERNEL PAGE ADDRESS
1548 :* REGISTERS (KIPAR0-KIPAR7) AND CHECKS THAT SOMETHING
1549 :* RESPONDS TO THEIR ADDRESSES.
1550 :*
1551 :*****
1552 021232 000004 TST6: SCOPE
1553
1554 021234 012737 021276 001110 1$: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
1555 021242 012737 021334 000004 MOV #5$, @#4 ;SET TIMEOUT VECTOR TO 5$
1556 021250 012700 172340 MOV #KIPAR0, R0 ;LOAD R0 WITH ADDRESS OF FIRST REG.
1557 021254 012701 000010 MOV #10, R1 ;LOAD R1 WITH LOOP COUNT (8)
1558 021260 012737 177777 001300 MOV #-1, ANDADR ;INITIALIZE 'AND' OF ADDR. LOC
1559 021266 005037 001302 CLR ORADR ;INITIALIZE 'OR' OF ADDR. LOC.
1560 021272 005037 001304 CLR TONUM ;INITIALIZE 'TIMEOUTS' COUNTER
1561 021276 005710 2$: TST (..) ;TRY ADDRESSING A KIPAR
1562 ;IF IT TIMES OUT, WILL GO TO 5$
1563 021300 062700 000002 3$: ADD #2, R0 ;PUT NEXT KIPAR ADDRESS IN R0
1564 021304 077104 SOB R1, 2$ ;LOOP BACK TO 2$ UNTIL ALL TESTED
1565 021306 012737 021234 001110 MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
1566 021314 005737 001304 TST TONUM ;DID ANY OF THE KIPARS TIME OUT?
1567 021320 001401 BEQ 4$ ;BRANCH IF NO
1568 021322 104010 ERROR 10 ;SUMMARY OF KIPAR TIMEOUTS
1569 021324 012737 002076 000004 4$: MOV #TIMERR, @#4 ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1570 021332 000414 BR TST7 ;GO TO NEXT TEST
1571
1572 021334 062706 000004 5$: ADD #4, KSP ;CLEAN UP THE STACK
1573 021340 104007 ERROR 7 ;ONE OF THE KIPARS TIMED OUT
1574 ;FOR TIGHTER SCOPE LOOP
1575 ;REPLACE ERROR CALL WITH
1576 ;'BR 2$' = 000756
1577 021342 010002 MOV R0, R2 ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
1578 021344 050237 001302 BIS R2, ORADR ;'OR' IT WITH OTHER ADDRS. THAT TIMED OUT
1579 021350 005102 COM R2 ;'AND' IT WITH OTHER ADDRS. THAT TIMED OUT
1580 021352 040237 001300 BIC R2, ANDADR
1581 021356 005237 001304 INC TONUM ;INCREMENT THE TIMEOUT COUNTER
1582 021362 000746 BR 3$ ;BRANCH BACK TO TEST THE NEXT KIPAR
1583
1584 :*****
1585 :*TEST 7 KERNEL PDR'S TIMEOUT TEST
1586 :*
1587 :* THIS TEST ADDRESSES THE EIGHT (8) KERNEL PAGE DESCRIPTOR
1588 :* REGISTERS (KIPDR0-KIPDR7) AND CHECKS THAT SOMETHING
1589 :* RESPONDS TO THEIR ADDRESSES.
1590 :*
1591 :*****
1592 021364 000004 TST7: SCOPE
1593
1594 021366 012737 021430 001110 1$: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
1595 021374 012737 021466 000004 MOV #5$, @#4 ;SET TIMEOUT VECTOR TO 5$
1596 021402 012700 172300 MOV #KIPDR0, R0 ;LOAD R0 WITH ADDRESS OF FIRST REG.
1597 021406 012701 000010 MOV #10, R1 ;LOAD R1 WITH LOOP COUNT (8)
```

```
1598 021412 012737 177777 001300      MOV    #-1,ANDADR      ;INITIALIZE 'AND' OF ADDR. LOC
1599 021420 005037 001302      CLR    ORADR           ;INITIALIZE 'OR' OF ADDR. LOC.
1600 021424 005037 001304      CLR    TONUM           ;INITIALIZE 'TIMEOUTS' COUNTER
1601 021430 005710                2$:   TST    (R0)         ;TRY ADDRESSING A KIPDR
1602                                ;IF IT TIMES OUT, WILL GO TO 5$
1603 021432 062700 000002      3$:   ADD    #2,R0        ;PUT NEXT KIPDR ADDRESS IN R0
1604 021436 077104                SOB    R1,2$          ;LOOP BACK TO 2$ UNTIL ALL TESTED
1605 021440 012737 021366 001110      MOV    #1$,$LPERR     ;RESET LOOP ON ERROR POINTER TO 1$
1606 021446 005737 001304      TST    TONUM          ;DID ANY OF THE KIPDRS TIME OUT?
1607 021452 001401                BEQ    4$             ;BRANCH IF NO
1608 021454 104010                ERROR  10            ;SUMMARY OF KIPDR TIMEOUTS
1609 021456 012737 002076 000004 4$:   MOV    #TIMERR,@#4    ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1610 021464 000414                BR     TST10         ;;GO TO NEXT TEST
1611
1612 021466 062706 000004      5$:   ADD    #4,KSP        ;CLEAN UP THE STACK
1613 021472 104007                ERROR  7             ;ONE OF THE KIPDRS TIMED OUT
1614                                ;FOR TIGHTER SCOPE LOOP
1615                                ;REPLACE ERROR CALL WITH
1616                                ;'BR 2$' = 000756
1617 021474 010002                MOV    R0,R2         ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
1618 021476 050237 001302      BIS    R2,ORADR       ;'OR' IT WITH OTHER ADDRS. THAT TIMED OUT
1619 021502 005102                COM    R2            ;'AND' IT WITH OTHER ADDRS. THAT TIMED OUT
1620 021504 040237 001300      BIC    R2,ANDADR      ;INCREMENT THE TIMEOUT COUNTER
1621 021510 005237 001304      INC    TONUM          ;BRANCH BACK TO TEST THE NEXT KIPDR
1622 021514 000746                BR     3$
```

```
1623
1624 *****
1625 :*TEST 10      USER PAR'S TIMEOUT TEST
1626 :*
1627 :*      THIS TEST ADDRESSES THE EIGHT (8) USER PAGE ADDRESS
1628 :*      REGISTERS (UIPAR0-UIPAR7) AND CHECKS THAT SOMETHING
1629 :*      RESPONDS TO THEIR ADDRESSES.
1630 :*
```

```
1631 *****
1632 021516 000004      TST10: SCOPE
1633
1634 021520 012737 021562 001110 1$:   MOV    #2$,$LPERR     ;SET LOOP ON ERROR POINTER TO 2$
1635 021526 012737 021620 000004      MOV    #5$,@#4       ;SET TIMEOUT VECTOR TO 5$
1636 021534 012700 177640                MOV    #UIPAR0,R0    ;LOAD R0 WITH ADDRESS OF FIRST REG.
1637 021540 012701 000010                MOV    #10,R1        ;LOAD R1 WITH LOOP COUNT (8)
1638 021544 012737 177777 001300      MOV    #-1,ANDADR     ;INITIALIZE 'AND' OF ADDR. LOC
1639 021552 005037 001302      CLR    ORADR          ;INITIALIZE 'OR' OF ADDR. LOC.
1640 021556 005037 001304      CLR    TONUM          ;INITIALIZE 'TIMEOUTS' COUNTER
1641 021562 005710                2$:   TST    (R0)         ;TRY ADDRESSING A UIPAR
1642                                ;IF IT TIMES OUT, WILL GO TO 5$
1643 021564 062700 000002      3$:   ADD    #2,R0        ;PUT NEXT UIPAR ADDRESS IN R0
1644 021570 077104                SOB    R1,2$          ;LOOP BACK TO 2$ UNTIL ALL TESTED
1645 021572 012737 021520 001110      MOV    #1$,$LPERR     ;RESET LOOP ON ERROR POINTER TO 1$
1646 021600 005737 001304      TST    TONUM          ;DID ANY OF THE UIPARS TIME OUT?
1647 021604 001401                BEQ    4$             ;BRANCH IF NO
1648 021606 104010                ERROR  10            ;SUMMARY OF UIPAR TIMEOUTS
1649 021610 012737 002076 000004 4$:   MOV    #TIMERR,@#4    ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1650 021616 000414                BR     TST11         ;;GO TO NEXT TEST
1651
1652 021620 062706 000004      5$:   ADD    #4,KSP        ;CLEAN UP THE STACK
1653 021624 104007                ERROR  7             ;ONE OF THE UIPARS TIMED OUT
```

```
1654  
1655  
1656  
1657 021626 010002          MOV    R0,R2          ;FOR TIGHTER SCOPE LOOP  
1658 021630 050237 001302  BIS    R2,ORADR      ;REPLACE ERROR CALL WITH  
1659 021634 005102          COM    R2             ;'BR 2$' = 000756  
1660 021636 040237 001300  BIC    R2,ANDADR     ;LOAD THE ADDRESS THAT TIMED OUT INTO R2  
1661 021642 005237 001304  INC    TONUM         ;'OR' IT WITH OTHER ADDRS. THAT TIMED OUT  
1662 021646 000746          BR     3$            ;'AND' IT WITH OTHER ADDRS. THAT TIMED OUT  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672 021650 000004          ;INCREMENT THE TIMEOUT COUNTER  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697 021760 010002          ;BRANCH BACK TO TEST THE NEXT UIPAR  
1698 021762 050237 001302  MOV    R0,R2  
1699 021766 005102          BIS    R2,ORADR  
1700 021770 040237 001300  COM    R2  
1701 021774 005237 001304  BIC    R2,ANDADR  
1702 022000 000746          INC    TONUM  
1703  
1704  
1705  
1706  
1707  
1708  
1709
```

```
::*****  
:*TEST 11          USER PDR'S TIMEOUT TEST
```

```
::*  
:*          THIS TEST ADDRESSES THE EIGHT (8) USER PAGE DESCRIPTOR  
:*          REGISTERS (UIPDR0-UIPDR7) AND CHECKS THAT SOMETHING  
:*          RESPONDS TO THEIR ADDRESSES.  
:*
```

```
::*****  
TST11: SCOPE
```

```
1674 021652 012737 021714 001110 1$:  MOV    #2$, $LPERR      ;SET LOOP ON ERROR POINTER TO 2$  
1675 021660 012737 021752 000004  MOV    #5$, @#4         ;SET TIMEOUT VECTOR TO 5$  
1676 021666 012700 177600          MOV    #UIPDR0,R0      ;LOAD R0 WITH ADDRESS OF FIRST REG.  
1677 021672 012701 000010          MOV    #10,R1         ;LOAD R1 WITH LOOP COUNT (8)  
1678 021676 012737 177777 001300  MOV    #-1,ANDADR     ;INITIALIZE 'AND' OF ADDR. LOC  
1679 021704 005037 001302          CLR    ORADR          ;INITIALIZE 'OR' OF ADDR. LOC.  
1680 021710 005037 001304          CLR    TONUM         ;INITIALIZE 'TIMEOUTS' COUNTER  
1681 021714 005710          2$:  TST    (R0)         ;TRY ADDRESSING A UIPDR  
1682  
1683 021716 062700 000002          3$:  ADD    #2,R0         ;IF IT TIMES OUT, WILL GO TO 5$  
1684 021722 077104          SOB    R1,2$         ;PUT NEXT UIPDR ADDRESS IN R0  
1685 021724 012737 021652 001110  MOV    #1$, $LPERR     ;LOOP BACK TO 2$ UNTIL ALL TESTED  
1686 021732 005737 001304          TST    TONUM         ;RESET LOOP ON ERROR POINTER TO 1$  
1687 021736 001401          BEQ    4$            ;DID ANY OF THE UIPDRS TIME OUT?  
1688 021740 104010          ERROR 10            ;BRANCH IF NO  
1689 021742 012737 002076 000004 4$:  MOV    #TIMERR,@#4    ;SUMMARY OF UIPDR TIMEOUTS  
1690 021750 000414          BR     TST12         ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS  
1691  
1692 021752 062706 000004          5$:  ADD    #4,KSP        ;GO TO NEXT TEST  
1693 021756 104007          ERROR 7  
1694  
1695  
1696  
1697 021760 010002          ;CLEAN UP THE STACK  
1698 021762 050237 001302  MOV    R0,R2          ;ONE OF THE UIPDRS TIMED OUT  
1699 021766 005102          BIS    R2,ORADR      ;FOR TIGHTER SCOPE LOOP  
1700 021770 040237 001300  COM    R2             ;REPLACE ERROR CALL WITH  
1701 021774 005237 001304  BIC    R2,ANDADR     ;'BR 2$' = 000756  
1702 022000 000746          INC    TONUM         ;LOAD THE ADDRESS THAT TIMED OUT INTO R2  
1703  
1704  
1705  
1706  
1707  
1708  
1709
```

```
::*****  
:*TEST 12          SR0(15:13) BIT TEST & SR2 TEST
```

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

```
1710      :*      THE REST OF BITS IN SR0 WILL BE CHECKED LATER.  
1711      :*      ALSO CHECK THAT SR2 IS TRACKING WITH MEM. MGMT.  
1712      :*      OFF BUT LOCKS UP WHEN ANY OF SR0 ERROR BITS SET.  
1713      :*  
1714      :*  
1715      :*****  
1715 022002 000004      TST12: SCOPE  
1716  
1717 022004 012700 177572      1$:  MOV    #SR0,R0      ;LOAD ADDRESS OF SR0 INTO R0  
1718 022010 012710 160000      MOV    #160000,(R0)    ;SET BITS <15:13> IN SR0 (ERROR BITS)  
1719 022014 000005      RESET      ;ISSUE AND "INIT" SIGNAL  
1720 022016 011001      MOV    (R0),R1        ;READ SR0 INTO R1 TO SEE IF CLEAR  
1721 022020 001401      BEQ    2$             ;BRANCH IF SR0<15:13> CLEARED BY "INIT"  
1722 022022 104011      ERROR   11          ;SR0<15:13> NOT CLEARED BY A "RESET"  
1723      ;FOR TIGHTER SCOPE LOOP  
1724      ;REPLACE ERROR CALL WITH  
1725      ;"BR 1$" = 000770  
1726 022024 012737 022024 001110 2$:  MOV    #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$  
1727 022032 013737 177576 001274 8$:  MOV    SR2,WASSR2      ;READ CONTENTS OF SR2  
1728 022040 012701 022032      MOV    #8$,R1         ;LOAD EXPECTED CONTENTS INTO R1  
1729 022044 020137 0C1274      CMP    R1,WASSR2      ;IS SR2 TRACKING?  
1730 022050 001401      BEQ    3$             ;BRANCH IF YES  
1731 022052 104041      ERROR   4!          ;SR2 NOT "TRACKING" VIRTUAL ADDRESSES  
1732      ;FOR TIGHTER SCOPE LOOP  
1733      ;REPLACE ERROR CALL WITH  
1734      ;"BR 2$" = 000767  
1735 022054 012737 022072 001110 3$:  MOV    #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$  
1736 022062 012701 100000      MOV    #BIT15,R1      ;PUT DATA TO BE WRITTEN IN R1  
1737 022066 012703 000003      MOV    #3,R3          ;SETUP R3 AS A LOOP COUNTER  
1738 022072 005010      4$:  CLR    (R0)          ;CLEAR SR0  
1739 022074 050110      5$:  BIS    R1,(R0)        ;SET ONE OF THE ERROR BITS IN SR0  
1740 022076 011002      MOV    (R0),R2        ;READ SR0 INTO R2  
1741 022100 020102      CMP    R1,R2          ;DID RIGHT ERROR BIT GET SET?  
1742 022102 001401      BEQ    6$             ;BRANCH IF YES  
1743 022104 104012      ERROR   12          ;BITS WERE SET WRONG IN SR0  
1744      ;FOR TIGHTER SCOPE LOOP  
1745      ;REPLACE ERROR CALL WITH  
1746      ;"BR 4$" = 000772  
1747 022106 012704 022074      6$:  MOV    #5$,R4         ;LOAD EXPECTED CONTENTS OF SR2 IN R4  
1748 022112 013737 177576 001274      MOV    SR2,WASSR2      ;READ SR2  
1749 022120 020437 001274      CMP    R4,WASSR2      ;DID SR2 LOCK UP WHEN ERROR  
1750      ;BIT SET IN SR1?  
1751 022124 001401      BEQ    7$             ;BRANCH IF YES  
1752 022126 104064      ERROR   64          ;SR2 DID NOT LOCK UP  
1753      ;FOR TIGHTER SCOPE LOOP  
1754      ;REPLACE ERROR CALL WITH  
1755      ;"BR 4$" = 000761  
1756 022130 006001      7$:  ROR    R1            ;CHANGE DATA TO CHECK NEXT ERROR BIT  
1757 022132 077321      SOB    R3,4$         ;LOOP BACK UNTIL <15:13> ALL TESTED  
1758 022134 005010      CLR    (R0)          ;CLEAR SR0 BEFORE LEAVING  
1759 022136 012737 022004 001110      MOV    #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$  
1760  
1761      :*****  
1762      ;*TEST 13      SR0 & PSW DUAL ADDRESSING TEST  
1763      :*  
1764      :*      THIS TEST CHECKS MORE OF THE ADDRESS DETECTION LOGIC BY  
1765      :*      VERIFYING THAT STATUS REGISTER 0 IS NOT EFFECTED BY WRITING
```

```
1766          : *      TO THE PSW AND THAT THE LOW BYTE OF STATUS REGISTER 0
1767          : *      IS NOT EFFECTED BY WRITING TO ITS HIGH BYTE. THIS IS TO
1768          : *      SEE IF ADJACENT OUTPUTS ARE SHORTED ON THE ADDRESS DET. LOGIC.
1769          : *
1770          : *****
1771 022144 000004 TST13: SCOPE
1772
1773 022146 005037 177776 1$: CLR PSW ;CLEAR THE PSW
1774 022152 005037 177572 CLR SR0 ;CLEAR STATUS REGISTER 0
1775 022156 106427 000340 MTPS #340 ;SET PRIORITY 7 IN LOW BYTE OF PSW
1776 022162 013700 177572 MOV SR0,R0 ;READ STATUS REGISTER 0
1777 022166 001401 BEQ 2$ ;BRANCH IF IT WAS STILL 0
1778 022170 104013 ERROR 13 ;SR0 EFFECTED BY A WRITE TO THE PSW
1779 ;FOR TIGHTER SCOPE LOOP
1780 ;REPLACE ERROR CALL WITH
1781 ;'BR 1$' = 000767
1782 022172 005037 177572 2$: CLR SR0 ;BE SURE SR0 IS 0 BEFORE LEAVING
1783 022176 005037 177776 CLR PSW ;BE SURE PSW IS 0 BEFORE LEAVING
1784
1785          : *****
1786          : *TEST 14 TEST THAT SR1 READS ALL ZEROS
1787          : *
1788          : *      THIS TESTS CHECKS THAT STATUS REGISTER 1
1789          : *      RESPONDS WITH ALL ZEROS, AND THAT ONLY BITS<5:4>
1790          : *      OF STATUS REGISTER 3 ARE WRITEABLE.
1791          : *
1792          : *****
1793 022202 000004 TST14: SCOPE
1794 022204 012700 177777 1$: MOV #-1,R0 ;FILL R0 WITH ALL ONES
1795 022210 013700 177574 MOV SR1,R0 ;READ SR1 INTO R0
1796 022214 001401 BEQ 2$ ;BRANCH IF SR1 READS ALL ZEROS
1797 022216 104014 ERROR 14 ;SR1 DID NOT READ ALL ZEROS
1798 ;FOR TIGHTER SCOPE LOOP
1799 ;REPLACE ERROR CALL WITH
1800 ;000772
1801 022220 012737 177777 172516 2$: MOV #-1,SR3 ;TRY TO WRITE ONES TO SR3
1802 022226 022737 000060 172516 CMP #60,SR3 ;ONLY BITS <5:4> SHOULD BE ONES
1803 022234 001401 BEQ 3$
1804 022236 104012 ERROR 12 ;DIDN'T READ BACK A '60'
1805 022240 105737 001246 3$: TST @#ENV ;TEST APT ENVIRONMENT BIT
1806 022244 001406 BEQ 6$ ;IF CLEAR, NOT ON APT SO DO TEST
1807 022246 005737 001234 TST @#PASS ;IF ON APT TEST PASS COUNTER
1808 022252 001403 BEQ 6$ ;IF CLEAR, FIRST PASS SO DO TEST
1809 022254 005037 172516 CLR SR3 ;IF ON APT AND NOT FIRST PASS FUDGE TEST
1810 022260 000401 BR 7$
1811 022262 000005 6$: RESET ;CLEARS SR3
1812 022264 005737 172516 7$: TST SR3 ;VERIFY THAT IT WAS CLEARED
1813 022270 4$:
1814 022270 001401 BEQ TST15 ;BRANCH IF SR3 READ ALL ZEROS
1815 022272 104012 ERROR 12 ;SR3 DIDN'T READ ALL ZEROS
1816
1817
1818
1819          : *****
1820          : *TEST 15 BIT TEST OF KERNEL & USER PAR'S
1821          : *
```

```
1822      : * THE FOLLOWING TEST CHECKS THE BITS <15:00> OF BOTH THE KERNEL
1823      : * AND USER PAGE ADDRESS REGISTERS. A '0' IS ROTATED THRU
1824      : * THE REGISTERS FROM LEFT TO RIGHT.
1825      : *
1826      : *****
1827 022274 000004      TST15: SCOPE
1828
1829 022276 012700 172340 1$: MOV #KIPAR0,R0 ;LOAD ADDRESS OF FIRST PAR IN R0
1830 022302 012703 000010 2$: MOV #10,R3 ;SETUP R3 TO COUNT 8 PAR'S
1831 022306 012737 022314 001110 3$: MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
1832 022314 005010 3$: CLR (R0) ;CLEAR THE PAR
1833 022316 011001 3$: MOV (R0),R1 ;READ THE PAR INTO R1
1834 022320 001401 4$: BEQ 4$ ;BRANCH IF PAR CLEARED OK
1835 022322 104011 4$: ERROR 11 ;PAR WOULD NOT CLEAR
1836 ;FOR TIGHTER SCOPE LOOP
1837 ;REPLACE ERROR CALL WITH
1838 ;'BR 3$' = 000774
1839 022324 012704 077777 4$: MOV #077777,R4 ;LOAD 'WALKING 0' TEST PATTERN IN R4
1840 022330 012737 022336 001110 5$: MOV #5$, $LPERR ;SET LOOP ON ERROR POINTER TO 5$
1841 022336 005010 5$: CLR (R0) ;CLEAR THE PAR BEFORE LOADING DATA
1842 022340 050410 5$: BIS R4,(R0) ;BIT SET THE TEST PATTERN INTO THE PAR
1843 022342 011002 5$: MOV (R0),R2 ;READ THE PAR INTO R2
1844 022344 020402 5$: CMP R4,R2 ;DOES DATA WRITTEN=DATA READ?
1845 022346 001402 5$: BEQ 6$ ;BRANCH IF YES
1846 022350 010401 5$: MOV R4,R1 ;SETUP FOR ERROR REPORTING
1847 022352 104012 5$: ERROR 12 ;PAR BITS DID NOT SET CORRECTLY
1848 ;FOR TIGHTER SCOPE LOOP
1849 ;REPLACE ERROR CALL WITH
1850 ;'BR 5$' = 000767
1851 022354 000261 6$: SEC ;SET THE C-BIT FOR THE ROTATE INST.
1852 022356 006004 6$: ROR R4 ;ROTATE THE TEST PATTERN IN R4
1853 022360 103766 6$: BCS 5$ ;BRANCH BACK IF MORE BITS TO TEST
1854 022362 062700 000002 6$: ADD #2,R0 ;GET NEXT PAR ADDRESS IN R0
1855 022366 077326 6$: SOB R3,3$ ;BRANCH BACK UNTIL ALL PAR'S TESTED
1856 022370 022700 177660 6$: CMP #UIPAR7+2,R0 ;HAVE USER PAR'S BEEN TESTED
1857 022374 103003 6$: BHIS 7$ ;BRANCH IF YES
1858 022376 012700 177640 6$: MOV #UIPAR0,R0 ;LOAD FIRST USER PAR ADDR. IN R0
1859 022402 000737 6$: BR 2$ ;BRANCH BACK TO TEST USER PAR'S
1860 022404 012737 022276 001110 7$: MOV #1$, $LPERR ;RESET LOOP OR ERROR POINTER TO 1$
1861 ;LEAVE TEST WITH BITS <11:1>=? IN ALL PAR'S
1862 : *****
1863 : *TEST 16 BIT TEST OF KERNEL & USER PDR'S
1864 : *
1865 : * THE FOLLOWING TEST CHECKS THE BITS <14:8> AND <3:1> OF BOTH THE
1866 : * KERNEL AND USER PAGE DESCRIPTOR REGISTERS. A '0' IS ROTATED
1867 : * THRU THE REGISTERS FROM LEFT TO RIGHT. SOME TEST PATTERNS WILL
1868 : * BE LOADED MORE THAN ONCE DUE TO THE UNUSED BITS IN THE PDR'S.
1869 : *
1870 : *****
1871 022412 000004      TST16: SCOPE
1872
1873 022414 012700 172300 1$: MOV #KIPDR0,R0 ;LOAD ADDRESS OF FIRST PDR IN R0
1874 022420 012703 000010 2$: MOV #10,R3 ;SETUP R3 TO COUNT 8 PDR'S
1875 022424 012737 022432 001110 3$: MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
1876 022432 005010 3$: CLR (R0) ;CLEAR THE PDR
1877 022434 011001 3$: MOV (R0),R1 ;READ THE PDR INTO R1
```

1878	022436	001401				BEQ	4\$:BRANCH IF PDR CLEARED OK
1879	022440	104011				ERROR	11		:PDR WOULD NOT CLEAR
1880									:FOR TIGHTER SCOPE LOOP
1881									:REPLACE ERROR CALL WITH
1882									: 'BR 3\$' = 000774
1883	022442	012704	077777		4\$:	MOV	#077777,R4		:LOAD 'WALKING '0' TEST PATTERN IN R4
1884	022446	012737	022454	001110		MOV	#5\$,\$LPERR		:SET LOOP ON ERROR POINTER TO 5\$
1885	022454	005010			5\$:	CLR	(R0)		:CLEAR THE PDR BEFORE LOADING DATA
1886	022456	010401				MOV	R4,R1		:LOAD DATA INTO R1
1887	022460	042701	100361			BIC	#100361,R1		:MASK UNUSED BITS OUT OF THE DATA
1888	022464	050110				BIS	R1,(R0)		:BIT SET THE TEST PATTERN INTO THE PDR
1889	022466	011002				MOV	(R0),R2		:READ THE PDR INTO R2
1890	022470	020102				CMP	R1,R2		:DOES DATA WRITTEN=DATA READ?
1891	022472	001401				BEQ	6\$:BRANCH IF YES
1892	022474	104012				ERROR	12		:PDR BITS DID NOT SET CORRECTLY
1893									:FOR TIGHTER SCOPE LOOP
1894									:REPLACE ERROR CALL WITH
1895									: 'BR 5\$' = 000767
1896	022476	000261			6\$:	SEC			:SET THE C-BIT FOR THE ROTATE INST.
1897	022500	006004				ROR	R4		:ROTATE THE TEST PATTERN IN R4
1898	022502	103764				BCS	5\$:BRANCH BACK IF MORE BITS TO TEST
1899	022504	062700	000002			ADD	#2,R0		:GET NEXT PDR ADDRESS IN R0
1900	022510	077330				SOB	R3,3\$:BRANCH BACK UNTIL ALL PDR'S TESTED
1901	022512	022700	177620			CMP	#UIPDR7+2,R0		:HAVE USER PDR'S BEEN TESTED?
1902	022516	103003				BHIS	7\$:BRANCH IF YES
1903	022520	012700	177600			MOV	#UIPDRO,R0		:LOAD FIRST USER PDR ADDR. IN R0
1904	022524	000735				BR	2\$:BRANCH BACK TO TEST USER PDR'S
1905	022526	012737	022414	001110	7\$:	MOV	#1\$,\$LPERR		:RESET LOOP ON ERROR POINTER TO 1\$
1906									:LEAVE TEST WITH ALL WRITEABLE BITS IN
1907									: ALL PDR'S = 1

1908
1909
1910
1911
1912
1913
1914
1915
1916
1917

```
*****  
*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
```

022534 000004


```
1918
1919 022536 012700 172340 001110 1$: MOV #KIPAR0,R0 ;LOAD ADDRESS OF FIRST PAR INTO R0
1920 022542 012737 022554 2$: MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
1921 022550 012703 000010 MOV #10,R3 ;LOAD LOOP COUNTER TO DO 8 PAR'S
1922 022554 012701 177777 3$: MOV #-1,R1 ;LOAD TEST PATTERN INTO R1
1923 022560 005010 CLR (R0) ;CLEAR THE PAR
1924 022562 110110 MOVB R1,(R0) ;WRITE 1'S TO THE LOW BYTE OF THE PAR
1925 022564 011002 MOV (R0),R2 ;READ THE ENTIRE PAR INTO R2
1926 022566 042701 177400 BIC #177400,R1 ;MASK HIGH BYTE & UNUSED BITS OUT OF THE DATA
1927 022572 020102 CMP R1,R2 ;WAS ONLY THE LOW BYTE WRITTEN TO
1928 022574 001401 BEQ 4$ ;BRANCH IF YES
1929 022576 104015 ERROR 15 ;HIGH BYTE EFFECTED BY WRITING LOW BYTE IN PAR
1930
1931 ;FOR TIGHTER SCOPE LOOP
1932 ;REPLACE ERROR CALL WITH
1933 022600 012737 022606 001110 4$: MOV #5$, $LPERR ;SET LOOP ON ERROR POINTER TO 5$
1934 022606 005010 5$: CLR (R0) ;CLEAR THE PAR
1935 022610 012701 177777 MOV #-1,R1 ;LOAD TEST, PATTERN INTO R1
1936 022614 110160 000001 MOVB R1,1(R0) ;WRITE 1'S TO THE HIGH BYTE OF THE PAR
1937 022620 011002 MOV (R0),R2 ;READ THE ENTIRE PAR INTO R2
1938 022622 042701 000377 BIC #000377,R1 ;MASK LOW BYTE
1939 022626 020102 CMP R1,R2 ;WAS ONLY THE HIGH BYTE WRITTEN TO?
1940 022630 001401 BEQ 6$ ;BRANCH IF YES
1941 022632 104015 ERROR 15 ;LOW BYTE EFFECTED BY WRITING HIGH BYTE IN PAR
1942
1943 ;FOR TIGHTER SCOPE LOOP
1944 ;REPLACE ERROR CALL WITH
1945 022634 062700 000002 6$: ADD #2,R0 ;PUT ADDRESS OF NEXT PAR IN R0
1946 022640 077333 SOB R3,3$ ;BRANCH BACK UNTIL 8 PAR'S TESTED
1947 022642 022700 177660 CMP #UIPAR7+2,R0 ;HAVE USER PAR'S BEEN TESTED
1948 022646 103003 BHIS 7$ ;BRANCH IF YES
1949 022650 012700 177640 MOV #UIPAR0,R0 ;LOAD ADDRESS OF FIRST USER PAR IN R0
1950 022654 000732 BR 2$ ;BRANCH BACK TO TEST USER PAR'S
1951 022656 012737 022536 001110 7$: MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
1952
1953 ;*****
1954 ;*TEST 20 TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PDR'S
1955 ;*
1956 ;* THE FOLLOWING TEST WRITES TO BOTH BYTES OF THE KERNEL & USER
1957 ;* PDR'S SEPERATELY TO SEE THAT WRITING TO ONE DOES NOT EFFECT
1958 ;* THE OTHER.
1959 ;*
1960 ;*****
1961 022664 000004 TST20: SCOPE
1962
1963 022666 012700 172300 001110 1$: MOV #KIPDR0,R0 ;LOAD ADDRESS OF FIRST PDR INTO R0
1964 022672 012737 022704 2$: MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
1965 022700 012703 000010 MOV #10,R3 ;LOAD LOOP COUNTER TO DO 8 PDR'S
1966 022704 012701 177777 3$: MOV #-1,R1 ;LOAD TEST PATTERN INTO R1
1967 022710 005010 CLR (R0) ;CLEAR THE PDR
1968 022712 110110 MOVB R1,(R0) ;WRITE 1'S TO THE LOW BYTE OF THE PDR
1969 022714 011002 MOV (R0),R2 ;READ THE ENTIRE PDR INTO R2
1970 022716 042701 177761 BIC #177761,R1 ;MASK HIGH BYTE & UNUSED BITS OUT OF DATA
1971 022722 020102 CMP R1,R2 ;WAS ONLY THE LOW BYTE WRITTEN TO?
1972 022724 001401 BEQ 4$ ;BRANCH IF YES
1973 022726 104015 ERROR 15 ;HIGH BYTE EFFECTED BY WRITING LOW BYTE IN PDR
```

```
1974                                     :FOR TIGHTER SCOPE LOOP
1975                                     :REPLACE ERROR CALL WITH
1976                                     :'BR 3$' = 000766
1977 022730 012737 022736 001110 4$:   MOV      #5$, $LPERR      :SET LOOP ON ERROR POINTER TO 5$
1978 022736 005010                    5$:   CLR      (R0)          :CLEAR THE PDR
1979 022740 012701 177777              MOV      #-1, R1        :LOAD TEST PATTERN INTO R1
1980 022744 110160 000001              MOVVB   R1, 1(R0)      :WRITE 1'S TO THE HIGH BYTE OF THE PDR
1981 022750 011002                    MOV      (R0), R2     :READ THE ENTIRE PDR INTO R2
1982 022752 042701 100377              BIC     #100377, R1   :MASK LOW BYTE & UNUSED BITS OUT OF DATA
1983 022756 020102                    CMP      R1, R2       :WAS ONLY THE HIGH BYTE WRITTEN TO?
1984 022760 001401                    BEQ     6$            :BRANCH IF YES
1985 022762 104015                    ERROR   15           :LOW BYTE EFFECTED BY WRITING HIGH BYTE IN PDR
1986                                     :FOR TIGHTER SCOPE LOOP
1987                                     :REPLACE ERROR CALL WITH
1988                                     :'BR 5$' = 000765
1989 022764 062700 000002              6$:   ADD     #2, R0       :PUT ADDRESS OF NEXT PDR IN R0
1990 022770 077333                    SOB     R3, 3$       :BRANCH BACK UNTIL 8 PDR'S TESTED
1991 022772 022700 177620              CMP     #UIPDR7+2, R0 :HAVE USER PDR'S BEEN TESTED?
1992 022776 103003                    BHIS   7$            :BRANCH IF YES
1993 023000 012700 177600              MOV     #UIPDRO, R0  :LOAD ADDRESS OF FIRST USER PDR IN R0
1994 023004 000732                    BR      2$           :BRANCH BACK TO TEST USER PDR'S
1995 023006 012737 022666 001110 7$:   MOV     #1$, $LPERR  :RESET LOOP ON ERROR POINTER TO 1$
1996
1997                                     :*****
1998                                     :*TEST 21      PAR-PDR DUAL ADDRESSING TEST
1999                                     :*
2000                                     :*   THE FOLLOWING TEST SETS ALL OF THE WRITEABLE BITS TO 1
2001                                     :*   IN THE SIXTEEN (16) PAR'S AND PDR'S USING THE "SETREG"
2002                                     :*   SUBROUTINE AND THEN CLEARS JUST ONE OF THEM.  THE "CMPREG"
2003                                     :*   SUBROUTINE IS USED TO READ ALL OF THE PAR'S AND PDR'S TO SEE
2004                                     :*   THAT ONLY ONE REGISTER WAS CLEARED IN RESPONSE TO THAT ONE
2005                                     :*   PAR OR PDR ADDRESS.  THE "CMPREG" SUBROUTINE REPORTS THE
2006                                     :*   ADDRESS OF ANY REGISTER WHOSE BITS DID NOT REMAIN SET WHEN
2007                                     :*   ANOTHER REGISTER WAS CLEARED.
2008                                     :*
2009                                     :*****
2009 023014 000004                      TST21: SCOPE
2010
2011 023016 012737 023040 001110 1$:   MOV     #2$, $LPERR  :SET LOOP ON ERROR POINTER 2$
2012 023024 012703 000010              MOV     #10, R3     :LOAD LOOP COUNTER WITH AN 8
2013 023030 012700 172300              MOV     #KIPDRO, R0 :LOAD ADDRESS OF FIRST KERNEL PDR AND R0
2014 023034 004737 035334              JSR     PC, SETREG   :SET ALL BITS IN ALL PAR'S IN PDR'S
2015 023040 012706 001100              2$:   MOV     #KERSTK, KSP :SETUP STACK POINTER
2016 023044 005010                    CLR     (R0)         :CLEAR ONE OF THE KERNEL PDR'S
2017 023046 004737 035426              JSR     PC, CMPREG   :SEE IF OTHER PAR/PDR'S WERE EFFECTED
2018 023052 012720 177777              MOV     #-1, (R0)+  :RESTORE ALL ONES, AND SETUP FOR NEXT PDR
2019 023056 077310                    SOB     R3, 2$       :LOOP TO 2$ UNTIL ALL KERNEL PDR'S CHECKED
2020 023060 012737 023076 001110      MOV     #3$, $LPERR  :SET LOOP ON ERROR POINTER TO 3$
2021 023066 012703 000010              MOV     #10, R3     :LOAD LOOP COUNTER WITH AN 8
2022 023072 012700 172340              MOV     #KIPARO, R0 :LOAD ADDRESS OF FIRST KERNEL PAR IN R0
2023 023076 012706 001100              3$:   MOV     #KERSTK, KSP :SETUP STACK POINTER
2024 023102 005010                    CLR     (R0)         :CLEAR ONE OF THE KERNEL PAR'S
2025 023104 004737 035426              JSR     PC, CMPREG   :SEE IF OTHER PAR/PDR'S WERE EFFECTED
2026 023110 012720 177777              MOV     #-1, (R0)+  :RESTORE ALL ONES, AND SETUP FOR NEXT PAR
2027 023114 077310                    SOB     R3, 3$       :LOOP TO 3$ UNTIL ALL KERNEL PAR'S CHECKED
2028 023116 012737 023134 001110      MOV     #4$, $LPERR  :SET LOOP ON ERROR POINTER TO 4$
2029 023124 012703 000010              MOV     #10, R3     :LOAD LOOP COUNTER WITH AN 8
```

```

2030 023130 012700 177600      MOV      #UIPDR0,R0      ;LOAD ADDRESS OF FIRST USER PDR IN R0
2031 023134 012706 001100      4$:     MOV      #KERSTK,KSP      ;SETUP STACK POINTER
2032 023140 005010              CLR      (R0)            ;CLEAR ONE OF THE USER PDR'S
2033 023142 004737 035426      JSR      PC,CMPREG      ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
2034 023146 012720 177777      MOV      #-1,(R0)+      ;RESTORE ALL ONES, AND SETUP FOR NEXT UPDR
2035 023152 077310              SOB      R3,4$          ;LOOP TO 4$ UNTIL ALL USER PDR'S CHECKED
2036 023154 012737 023172 001110  MOV      #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
2037 023162 012703 000010      MOV      #10,R3         ;LOAD LOOP COUNTER WITH AN 8
2038 023166 012700 177640      MOV      #UIPAR0,R0     ;LOAD ADDRESS OF FIRST USER PAR IN R0
2039 023172 012706 001100      5$:     MOV      #KERSTK,KSP      ;SETUP STACK POINTER
2040 023176 005010              CLR      (R0)            ;CLEAR ONE OF THE USER PAR'S
2041 023200 004737 035426      JSR      PC,CMPREG      ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
2042 023204 012720 177777      MOV      #-1,(R0)+      ;RESTORE ALL ONES, AND SETUP FOR NEXT UPAR
2043 023210 077310              SOB      R3,5$          ;LOOP TO 5$ UNTIL ALL USER PAR'S CHECKED
2044 023212 012737 023016 001110  MOV      #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
2045
2046
2047      ;*****
2048      ;*TEST 22      TEST THAT PAR-PDR'S NOT AFFECTED BY RESET
2049      ;*
2050      ;*      THIS TEST CHECKS TO SEE THAT THE KERNEL OR USER PAR/PDR'S ARE
2051      ;*      NOT AFFECTED BY THE EXECUTION OF A 'RESET' INSTRUCTION.  THE
2052      ;*      'SETREG' SUBROUTINE IS USED TO SET ALL WRITEABLE BITS TO A '1' IN
2053      ;*      THE PAR/PDR'S.  THEN THEY ARE READ TO SEE THAT THEY REMAINED
2054      ;*      UNCHANGED
2055      ;*****
2056 023220 000004      TST22: SCOPE
2057
2058
2059 023222 105737 001246      TSTB    @#$ENV          ;TEST APT ENVIRONMENT BIT
2060 023226 001403          BEQ     1$              ;IF CLEAR, NOT ON APT SO DO TEST
2061 023230 005737 001234      TST     @#$PASS         ;IF ON APT TEST PASS COUNTER
2062 023234 001063          BNE     10$            ;IF NOT FIRST PASS SKIP TEST
2063 023236 004737 035334      1$:     JSR      PC,SETREG      ;SET ALL BITS IN ALL PAR'S AND PDR'S
2064 023242 000005          RESET                    ;ISSUE AN 'INIT' BY EXECUTING A RESET
2065 023244 012700 172300      MOV      #KIPDR0,R0     ;LOAD ADDRESS OF FIRST KERNEL PDR IN R0
2066 023250 012704 000010      MOV      #10,R4         ;LOAD LOOP COUNTER WITH AN 8
2067 023254 011001 077416      2$:     MOV      (R0),R1        ;READ A KERNEL PDR INTO R1
2068 023256 022701          CMP     #77416,R1       ;ARE ALL THE BITS STILL SET?
2069 023262 001401          BEQ     3$              ;BRANCH IF YES
2070 023264 104055          ERROR   55             ;KERNEL PDR AFFECTED BY A RESET
2071
2072
2073
2074 023266 062700 000002      3$:     ADD     #2,R0           ;FORM ADDRESS OF NEXT KERNEL PDR
2075 023272 077410          SOB    R4,2$          ;LOOP TO 2$ UNTIL ALL KERNEL PDR'S CHECKED
2076 023274 012700 172340      MOV      #KIPAR0,R0     ;LOAD ADDRESS OF FIRST KERNEL PAR IN R0
2077 023300 012704 000010      MOV      #10,R4         ;LOAD LOOP COUNTER WITH AN 8
2078 023304 011001 177777      4$:     MOV      (R0),R1        ;READ A KERNEL PAR INTO R1
2079 023306 022701          CMP     #177777,R1      ;ARE ALL THE BITS STILL SET?
2080 023312 001401          BEQ     5$              ;BRANCH IF YES
2081 023314 104055          ERROR   55             ;KERNEL PAR AFFECTED BY A RESET
2082
2083
2084
2085 023316 062700 000002      5$:     ADD     #2,R0           ;FORM ADDRESS OF NEXT KERNEL PAR

```

```
2086 023322 077410          SOB      R4,4$          :LOOP TO 4$ UNTIL ALL KERNEL PAR'S CHECKED
2087 023324 012700 177600    MOV      #UIPDR0,R0    :LOAD ADDRESS OF FIRST USER PDR IN R0
2088 023330 012704 000010    MOV      #10,R4        :LOAD LOOP COUNTER WITH AN 8
2089 023334 011001          6$:    MOV      (R0),R1      :READ A USER PDR INTO R1
2090 023336 022701 077416    CMP      #77416,R1     :ARE ALL THE BITS STILL SET?
2091 023342 001401          BEQ      7$            :BRANCH IF YES
2092 023344 104055          ERROR    55           :USER PDR AFFECTED BY A RESET
2093                                :FOR TIGHTER SCOPE LOOP
2094                                :REPLACE ERROR CALL WITH
2095                                :'BR 6$' = 000773
2096 023346 062700 000002    7$:    ADD      #2,R0        :FORM ADDRESS OF NEXT USER PDR
2097 023352 077410          SOB      R4,6$        :LOOP TO 6$ UNTIL ALL USER PDR'S CHECKED
2098
2099 023354 012700 177640    MOV      #UIPAR0,R0    :LOAD ADDRESS OF FIRST USER PAR IN R0
2100 023360 012704 000010    MOV      #10,R4        :LOAD LOOP COUNTER WITH AN 8
2101 023364 011001          8$:    MOV      (R0),R1      :READ A USER PAR INTO R1
2102 023366 022701 177777    CMP      #177777,R1    :ARE ALL THE BITS STILL SET?
2103 023372 001401          BEQ      9$            :BRANCH IF YES
2104 023374 104055          ERROR    55           :USER PAR AFFECTED BY A RESET
2105                                :FOR TIGHTER SCOPE LOOP
2106                                :REPLACE ERROR CALL WITH
2107                                :'BR 8$' = 000773
2108 023376 062700 000002    9$:    ADD      #2,R0        :FORM ADDRESS OF NEXT USER PAR
2109 023402 077410          SOB      R4,8$        :LOOP TO 8$ UNTIL ALL USER PAR'S CHECKED
2110 023404
2111                                10$:
2112                                :*****
2113                                :*TEST 23          RELOCATION & ADDER TEST (NO CARRIES)
2114                                :*
2115                                :* THE FOLLOWING TEST SETS UP THE KERNEL PAR'S AND PDR'S
2116                                :* FOR THE REST OF THE PROGRAM. IT THEN USES DIFFERENT
2117                                :* VIRTUAL ADDRESSES AND DIFFERENT VALUES FOR KERNEL PAR 4
2118                                :* TO PUT DIFFERENT PATTERNS AT THE INPUTS OF THE
2119                                :* MEMORY MANAGEMENT ADDER. THE VALUES ARE SUCH
2120                                :* THAT NO CARRIES ARE GENERATED OUT OF THE ADDER.
2121                                :*
2122                                :* THE METHOD USED TO SEE THAT THE RIGHT PHYSICAL BUS ADDRESS
2123                                :* IS FORMED BY THE ADDER IS TO WRITE A PATTERN TO VIRTUAL
2124                                :* LOCATION WITH MEMORY MGMT., AND
2125                                :* THEN READ THAT LOCATION USING THE PHYSICAL ADDRESS THAT SHOULD
2126                                :* HAVE BEEN FORMED TO SEE IF THE TEST PATTERN GOT THERE.
2127                                :* 22-BIT AND 18-BIT ADDRESSING ARE USED.
2128                                :*****
2128 023404 000004    TST23:  SCOPE
2129
2130 023406 012700 172340    1$:    MOV      #KIPAR0,R0    :LOAD ADDRESS OF FIRST KERNEL PAR IN R0
2131 023412 005001          CLR      R1            :CLEAR R1
2132 023414 012702 000007    MOV      #7,R2         :LOAD LOOP COUNTER WITH A 7
2133 023420 010120          2$:    MOV      R1,(R0)+      :MAP KERNEL PAR'S TO PAGES 0-6 (4K EACH)
2134 023422 062701 000200    ADD      #200,R1
2135 023426 077204          SOB      R2,2$        :LOOP UNTIL KIPAR0 - KIPAR6 ARE LOADED
2136 023430 012710 177600    MOV      #177600,(R0)  :MAP KIPAR7 TO THE I/O PAGE
2137 023434 012700 172300    MOV      #KIPDR0,R0    :LOAD ADDRESS OF FIRST KERNEL PDR IN R0
2138 023440 012701 077406    MOV      #77406,R1     :LOAD PDR DATA INTO R1
2139 023444 012702 000010    MOV      #10,R2        :LOAD LOOP COUNTER WITH AN 8
2140 023450 010120          3$:    MOV      R1,(R0)+      :MAP ALL 8 PAGES 128 BLOCKS, UPWARD
2141 023452 077202          SOB      R2,3$        : EXPANDABLE, READ/WRITE
```

```

2142
2143 023454 012737 023454 001110 4$: MOV #4$, $LPERR ;SET LOOP ON ERROR POINTER TO 4$
2144 023462 012700 067776 MOV #67776, R0 ;LOAD PHYSICAL ADDR. PBA INTO R0
2145 023466 012701 107776 MOV #107776, R1 ;LOAD VIRTUAL ADDR. VBA INTO R1
2146 023472 012702 125250 MOV #125250, R2 ;LOAD TEST PATTERN INTO R2
2147 023476 012704 000600 MOV #600, R4 ;LOAD R4 WITH PAR VALUE
2148 023502 010437 172350 MOV R4, KIPAR4 ;LOAD KERNEL PAR 4 BITS <15:00>
2149 023506 011037 001176 MOV (R0), $TMP0 ;SAVE CONTENTS AT TEST LOCATION
2150 023512 005037 172516 CLR SR3 ;SET UP FOR 18 BIT ADDRESSING
2151 023516 052737 000001 177572 BIS #BIT0, SR0 ;TURN ON MEM. MGMT.
2152 023524 010211 MOV R2, (R1) ;LOAD 125250 USING ADDER (PAR4 + VIRT ADDR.)
2153 023526 005037 177572 CLR SR0 ;TURN OFF MEMORY MGMT.
2154 023532 011003 MOV (R0), R3 ;READ 125250 BACK WITHOUT USING MEM. MGMT.
2155 023534 013710 001176 MOV $TMP0, (R0) ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2156 023540 020203 CMP R2, R3 ;WAS SAME PATTERN READ BACK THAT WAS
2157 ;WRITTEN USING MEMORY MANAGEMENT?
2158 023542 001405 BEQ 5$ ;BRANCH IF YES
2159 023544 010137 001306 MOV R1, VIRT1 ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2160 023550 004737 035620 JSR PC, FORMPA ;GO FORM PHYSICAL ADDRESS FOR TYPING
2161 023554 104017 ERROR 17 ;TEST LOCATION DID NOT HAVE PATTERN
2162 ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
2163 ;APPARENTLY PHYSICAL ADDR. WAS
2164 ;FORMED WRONG BY ADDERS USING
2165 ;THE VIRTUAL ADDR. AND KIPAR4
2166 ;FOR TIGHTER SCOPE LOOP
2167 ;REPLACE ERROR CALL WITH
2168 ;'BR 4$' = 000742
2169 023556 5$:
2170 023556 012737 023556 001110 6$: MOV #6$, $LPERR ;SET LOOP ON ERROR POINTER TO 6$
2171 023564 012700 067776 MOV #67776, R0 ;LOAD PHYSICAL ADDR. PBA INTO R0
2172 023570 012701 102576 MOV #102576, R1 ;LOAD VIRTUAL ADDR. VBA INTO R1
2173 023574 012702 125251 MOV #125251, R2 ;LOAD TEST PATTERN INTO R2
2174 023600 012704 000652 MOV #652, R4 ;LOAD R4 WITH PAR VALUE
2175 023604 010437 172350 MOV R4, KIPAR4 ;LOAD KERNEL PAR 4 BITS <15:00>
2176 023610 011037 001176 MOV (R0), $TMP0 ;SAVE CONTENTS AT TEST LOCATION
2177 023614 005037 172516 CLR SR3 ;SET UP FOR 18 BIT ADDRESSING
2178 023620 052737 000001 177572 BIS #BIT0, SR0 ;TURN ON MEM. MGMT.
2179 023626 010211 MOV R2, (R1) ;LOAD 125251 USING ADDER (PAR4 + VIRT ADDR.)
2180 023630 005037 177572 CLR SR0 ;TURN OFF MEMORY MGMT.
2181 023634 011003 MOV (R0), R3 ;READ 125251 BACK WITHOUT USING MEM. MGMT.
2182 023636 013710 001176 MOV $TMP0, (R0) ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2183 023642 020203 CMP R2, R3 ;WAS SAME PATTERN READ BACK THAT WAS
2184 ;WRITTEN USING MEMORY MANAGEMENT?
2185 023644 001405 BEQ 7$ ;BRANCH IF YES
2186 023646 010137 001306 MOV R1, VIRT1 ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2187 023652 004737 035620 JSR PC, FORMPA ;GO FORM PHYSICAL ADDRESS FOR TYPING
2188 023656 104017 ERROR 17 ;TEST LOCATION DID NOT HAVE PATTERN
2189 ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
2190 ;APPARENTLY PHYSICAL ADDR. WAS
2191 ;FORMED WRONG BY ADDERS USING
2192 ;THE VIRTUAL ADDR. AND KIPAR4
2193 ;FOR TIGHTER SCOPE LOOP
2194 ;REPLACE ERROR CALL WITH
2195 ;'BR 6$' = 000742
2196 023660 7$:
2197 023660 012737 023660 001110 8$: MOV #8$, $LPERR ;SET LOOP ON ERROR POINTER TO 8$
  
```

2198	023666	012700	067776			MOV	#67776,R0	:LOAD PHYSICAL ADDR. PBA INTO R0
2199	023672	012701	105276			MOV	#105276,R1	:LOAD VIRTUAL ADDR. VBA INTO R1
2200	023676	012702	125252			MOV	#125252,R2	:LOAD TEST PATTERN INTO R2
2201	023702	012704	000625			MOV	#625,R4	:LOAD R4 WITH PAR VALUE
2202	023706	010437	172350			MOV	R4,KIPAR4	:LOAD KERNEL PAR 4 BITS <15:00>
2203	023712	011037	001176			MOV	(R0), \$TMP0	:SAVE CONTENTS AT TEST LOCATION
2204	023716	052737	000020	172516		BIS	#BIT4,SR3	:SET UP FOR 22 BIT ADDRESSING
2205	023724	052737	000001	177572		BIS	#BIT0,SR0	:TURN ON MEM. MGMT.
2206	023732	010211				MOV	R2,(R1)	:LOAD 125252 USING ADDER (PAR4 + VIRT ADDR.)
2207	023734	005037	177572			CLR	SR0	:TURN OFF MEMORY MGMT.
2208	023740	011003				MOV	(R0),R3	:READ 125252 BACK WITHOUT USING MEM. MGMT.
2209	023742	013710	001176			MOV	\$TMP0,(R0)	:RESTORE ORIGINAL CONTENTS TO TEST LOC.
2210	023746	020203				CMP	R2,R3	:WAS SAME PATTERN READ BACK THAT WAS
2211								:WRITTEN USING MEMORY MANAGEMENT?
2212	023750	001405				BEQ	9\$:BRANCH IF YES
2213	023752	010137	001306			MOV	R1,VIRT1	:SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2214	023756	004737	035620			JSR	PC,FORMPA	:GO FORM PHYSICAL ADDRESS FOR TYPING
2215	023762	104017				ERROR	17	:TEST LOCATION DID NOT HAVE PATTERN
2216								:THAT SHOULD HAVE BEEN WRITTEN TO IT.
2217								:APPARENTLY PHYSICAL ADDR. WAS
2218								:FORMED WRONG BY ADDERS USING
2219								:THE VIRTUAL ADDR. AND KIPAR4
2220								:FOR TIGHTER SCOPE LOOP
2221								:REPLACE ERROR CALL WITH
2222								: 'BR 8\$' = 000742
2223	023764				9\$:			
2224								
2225	023764	012737	023764	001110	10\$:	MOV	#10\$,\$LPERR	:SET LOOP ON ERROR POINTER TO 10\$
2226	023772	012700	177776			MOV	#PSW,R0	:LOAD PHYS. ADDR. OF PSW INTO R0
2227	023776	012701	100076			MOV	#100076,R1	:LOAD VIRTUAL ADDR. FOR PSW INTO R1
2228	024002	012702	030340			MOV	#030340,R2	:LOAD DATA FOR PSW IN R2
2229	024006	012704	007777			MOV	#7777,R4	:LOAD R4 WITH PAR VALUE
2230	024012	010437	172350			MOV	R4,KIPAR4	:LOAD KERNEL PAR 4 BITS <11:00>
2231	024016	005010				CLR	(R0)	:CLEAR THE PSW
2232	024020	005037	172516			CLR	SR3	:SET UP FOR 18 BIT ADDRESSING
2233	024024	052737	000001	177572		BIS	#BIT0,SR0	:TURN ON MEMORY MANAGEMENT
2234	024032	010211				MOV	R2,(R1)	:LOAD PSW USING ADDER (PAR4 + VIRT ADDR.)
2235	024034	005037	177572			CLR	SR0	:TURN OFF MEM. MGMT (SR0=0)
2236	024040	011003				MOV	(R0),R3	:READ PSW BACK WITHOUT USING MEM. MGMT.
2237	024042	005010				CLR	(R0)	:CLEAR THE PSW
2238	024044	042703	000037			BIC	#37,R3	:MASK T-BIT & CC BITS OUT OF DATA READ
2239	024050	020203				CMP	R2,R3	:WAS PSW WRITTEN?
2240	024052	001405				BEQ	11\$:BRANCH IF YES
2241	024054	010137	001306			MOV	R1,VIRT1	:SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2242	024060	004737	035620			JSR	PC,FORMPA	:GO FORM PHYSICAL ADDR. FOR TYPING
2243	024064	104017				ERROR	17	:PSW DID NOT HAVE DATA THAT IT SHOULD HAVE.
2244								:APPARENTLY PHYS. ADDR. OF PSW WAS
2245								:NOT FORMED BY ADDERS USING THE
2246								:VIRTUAL ADDR. AND KIPAR4
2247								:FOR TIGHTER SCOPE LOOP
2248								:REPLACE ERROR CALL WITH
2249								: 'BR 10\$' = 000742
2250	024066	012737	024066	001110	11\$:	MOV	#11\$,\$LPERR	:SET LOOP ON ERROR POINTER TO 11\$
2251	024074	012700	177776			MOV	#PSW,R0	:LOAD PHYS. ADDR. OF PSW INTO R0
2252	024100	012701	117776			MOV	#117776,R1	:LOAD VIRTUAL ADDR. FOR PSW INTO R1
2253	024104	012702	030240			MOV	#030240,R2	:LOAD DATA FOR PSW IN R2

```
2254 024110 012704 177600      MOV      #177600,R4      ;LOAD R4 WITH PAR VALUE
2255 024114 010437 172350      MOV      R4,KIPAR4      ;LOAD KERNEL PAR 4 BITS <15:00>
2256 024120 052737 000020 172516  BIS      #BIT4,SR3      ;SET UP FOR 22 BIT ADDRESSING
2257 024126 052737 000001 177572  BIS      #BIT0,SR0      ;TURN ON MEMORY MANAGEMENT
2258 024134 010211              MOV      R2,(R1)        ;LOAD PSW USING ADDER (PAR4 + VIRT. ADDR.)
2259 024136 005037 177572      CLR      SR0            ;TURN OFF MEM. MGMT (SR0=0)
2260 024142 011003              MOV      (R0),R3        ;READ PSW BACK WITHOUT USING MEM. MGMT.
2261 024144 005010              CLR      (R0)           ;CLEAR THE PSW
2262 024146 042703 000037      BIC      #37,R3         ;MASK T-BIT & CC BITS OUT OF DATA READ
2263 024152 020203              CMP      R2,R3          ;WAS PSW WRITTEN?
2264 024154 001405              BEQ      12$            ;BRANCH IF YES
2265 024156 010137 001306      MOV      R1,VIRT1       ;SAVE VIRTUAL ADDR. TO FORM PHYSICAL ADDR.
2266 024162 004737 035620      JSR      PC,FORMPA      ;GO FORM PHYSICAL ADDR. FOR TYPING
2267 024166 104017              ERROR   17              ;PSW DID NOT HAVE DATA THAT IT SHOULD
2268                                ;HAVE, APPARENTLY PHYS. ADDR. OF PSW WAS
2269                                ;NOT FORMED BY ADDERS USING THE
2270                                ;VIRTUAL ADDR. AND KIPAR4
2271                                ;FOR TIGHTER SCOPE LOOP
2272                                ;REPLACE ERROR CALL WITH
2273                                ;'BR 11$' = 000743
2274 024170 012737 023406 001110 12$:  MOV      #1$,$LPERR     ;RESET LOOP ON ERROR POINTER TO 1$
2275
2276                                ;*****
2277                                ;*TEST 24      RELOCATION & ADDER TEST (WITH CARRIES)
2278                                ;*
2279                                ;*      THE FOLLOWING TEST USES THE SAME METHOD AS THE PREVIOUS
2280                                ;*      TEST TO VERIFY MEMORY MANagements ABILITY TO CONSTRUCT
2281                                ;*      PHYSICAL BUS ADDRESSES USING A VIRTUAL BUS ADDRESS AND THE
2282                                ;*      CONTENTS OF A PAGE ADDRESS REGISTER.  HOWEVER, THE VALUES
2283                                ;*      AND PATTERNS USED IN THIS TEST WILL GENERATE CARRIES
2284                                ;*      AND CHECK 'WRAPAROUND' TO ADDRESS 000000 BY
2285                                ;*      USING VIRTUAL ADDR. 111400 AND KIPAR4 = 177664.
2286                                ;*      22-BIT ADDRESSING IS USED.
2287                                ;*****
2288 024176 000004      TST24:  SCOPE
2289
2290 024200              1$:      ;KERNEL PAR'S AND PDR'S HAVE BEEN
2291                                ;SET UP BY THE PREVIOUS TEST
2292 024200 012737 024200 001110 2$:  MOV      #2$,$LPERR     ;SET LOOP ON ERROR POINTER TO 2$
2293 024206 012700 066476      MOV      #66476,R0      ;LOAD PHYSICAL ADDR. PBA INTO R0
2294 024212 012701 114376      MOV      #114376,R1     ;LOAD VIRTUAL ADDR. VBA INTO R1
2295 024216 012702 125253      MOV      #125253,R2     ;LOAD TEST PATTERN INTO R2
2296 024222 012704 000521      MOV      #521,R4        ;LOAD R4 WITH PAR VALUE
2297 024226 010437 172350      MOV      R4,KIPAR4      ;LOAD KERNEL PAR 4 BITS <15:00>
2298 024232 011037 001176      MOV      (R0),$TMP0     ;SAVE CONTENTS AT TEST LOCATION
2299 024236 052737 000020 172516  BIS      #BIT4,SR3      ;SET UP FOR 22 BIT ADDRESSING
2300 024244 052737 000001 177572  BIS      #BIT0,SR0      ;TURN ON MEM. MGMT.
2301 024252 010211              MOV      R2,(R1)        ;LOAD 125253 USING ADDER (PAR4 + VIRT ADDR.)
2302 024254 005037 177572      CLR      SR0            ;TURN OFF MEMORY MGMT.
2303 024260 011003              MOV      (R0),R3        ;READ 125253 BACK WITHOUT USING MEM. MGMT.
2304 024262 013710 001176      MOV      $TMP0,(R0)     ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2305 024266 020203              CMP      R2,R3          ;WAS SAME PATTERN READ BACK THAT WAS
2306                                ;WRITTEN USING MEMORY MANAGEMENT?
2307                                ;BRANCH IF YES
2308 024272 010137 001306      MOV      R1,VIRT1       ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2309 024276 004737 035620      JSR      PC,FORMPA      ;GO FORM PHYSICAL ADDRESS FOR TYPING
```



```

2366 ;APPARENTLY PHYSICAL ADDR. WAS
2367 ;FORMED WRONG BY ADDERS USING
2368 ;THE VIRTUAL ADDR. AND KIPAR4
2369 ;FOR TIGHTER SCOPE LOOP
2370 ;REPLACE ERROR CALL WITH
2371 ;'BR 6$' = 000742
2372 024514 7$:
2373 024514 012737 024514 001110 8$: MOV #8$, $LPERR ;SET LOOP ON ERROR POINTER TO 8$
2374 024522 012700 000000 MOV #0000, R0 ;LOAD PHYSICAL ADDR. PBA INTO R0
2375 024526 012701 111400 MOV #111400, R1 ;LOAD VIRTUAL ADDR. VBA INTO R1
2376 024532 012702 125256 MOV #125256, R2 ;LOAD TEST PATTERN INTO R2
2377 024536 012704 177664 MOV #177664, R4 ;LOAD R4 WITH PAR VALUE
2378 024542 010437 172350 MOV R4, KIPAR4 ;LOAD KERNEL PAR 4 BITS <15:00>
2379 024546 011037 001176 MOV (R0), $TMP0 ;SAVE CONTENTS AT TEST LOCATION
2380 024552 052737 000020 172516 BIS #BIT4, SR3 ;SET UP FOR 22 BIT ADDRESSING
2381 024560 052737 000001 177572 BIS #BIT0, SR0 ;TURN ON MEM. MGMT.
2382 024566 010211 MOV R2, (R1) ;LOAD 125256 USING ADDER (PAR4 + VIRT ADDR.)
2383 024570 005037 177572 CLR SR0 ;TURN OFF MEMORY MGMT.
2384 024574 011003 MOV (R0), R3 ;READ 125256 BACK WITHOUT USING MEM. MGMT.
2385 024576 013710 001176 MOV $TMP0, (R0) ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2386 024602 020203 CMP R2, R3 ;WAS SAME PATTERN READ BACK THAT WAS
2387 ;WRITTEN USING MEMORY MANAGEMENT?
2388 BEQ 9$ ;BRANCH IF YES
2389 024606 010137 001306 MOV R1, VIRT1 ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2390 024612 004737 035620 JSR PC, FORMPA ;GO FORM PHYSICAL ADDRESS FOR TYPING
2391 024616 104017 ERROR 17 ;TEST LOCATION DID NOT HAVE PATTERN
2392 ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
2393 ;APPARENTLY PHYSICAL ADDR. WAS
2394 ;FORMED WRONG BY ADDERS USING
2395 ;THE VIRTUAL ADDR. AND KIPAR4
2396 ;FOR TIGHTER SCOPE LOOP
2397 ;REPLACE ERROR CALL WITH
2398 ;'BR 8$' = 000742
2399 024620 9$:
2400 024620 012737 024200 001110 MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$

```

```

2401
2402 *****
2403 *TEST 25 READ AND WRITE WHILE IN RELOCATE MODE
2404 *
2405 * THE FOLLOWING TEST TURNS ON MEMORY MANAGEMENT AND THEN
2406 * READS AND WRITES LOCATIONS BETWEEN PHYSICAL ADDRESSES
2407 * 060000-067600. ONE LOCATION IN EVERY BLOCK (32. WORDS)
2408 * IS WRITTEN USING PAR4 AND READ USING PAR5. THIS IS
2409 * DONE IN BOTH USER AND KERNEL MODES. THE USER PAR/PDR'S
2410 * ARE SET UP AT THE BEGINNING OF THE TEST AND ONCE MEMORY
2411 * MANAGEMENT IS TURNED ON IT IS LEFT ON FOR THE REST OF THE
2412 * OF THE PROGRAM. THE 'MODE' INPUT TO THE PAR/PDR ADDRESS MUX
2413 * IS CHECKED BY READING AND WRITING IN USER MODE. REMEMBER
2414 * ALSO, THAT SINCE MEMORY MANAGEMENT IS ON (IN RELOCATE
2415 * MODE) THE PROGRAM ITSELF IS USING ITS VIRTUAL ADDRESSES AND
2416 * PAR/PDR'S 0-3 TO EXECUTE. PAR7/PDR7 ARE USED TO ACCESS THE
2417 * I/O PAGE.
2418 *
2419 * WHILE TESTING IN KERNEL MODE, USER PAGES 4 & 5 ARE MAPPED
2420 * NON-RESIDENT WITH DIFFERENT PAR VALUES THAN THE KERNEL
2421 * PAR'S TO BE SURE THAT THE KERNEL PAR'S AND PDR'S ARE BEING

```

```

2422      : *      USED WHEN IN KERNEL MODE (AND VICE VERSA WHILE TESTING IN
2423      : *      USER MODE).  IF A MEM. MGMT. TRAP OCCURS, THE PROGRAM GOES
2424      : *      TO 9$ WHERE THE TRAP IS REPORTED.
2425      : *
2426      : *      BY SETTING THE LOCATION $MADR1 IN THE E-TABLE TO A CONSTANT,
2427      : *      AS DESCRIBED IN THE DOCUMENTATION, THIS TEST WILL CONTINUE
2428      : *      ACCESSING LOCATIONS THROUGHOUT MEMORY BY INCREASING THE VALUE
2429      : *      OF PAR4 AND PAR5.
2430      : *****
2431 024626 000004  TST25: SCOPE
2432
2433 024630 005037 177776 1$: CLR PSW ;START IN KERNEL MODE
2434 024634 012704 000577 MOV #577,R4 ;LOAD R4 WITH VALUE FOR PAR4
2435 024640 012705 000600 MOV #600,R5 ;LOAD R5 WITH VALUE FOR PAR5
2436 024644 010437 172350 MOV R4,KIPAR4 ;LOAD KERNEL PAR4
2437 024650 010537 172352 MOV R5,KIPAR5 ;LOAD KERNEL PAR5
2438 024654 012700 177640 MOV #UIPAR0,R0 ;LOAD ADDRESS OF FIRST USER PAR IN R0
2439 024660 005001 CLR R1 ;CLEAR R1
2440 024662 012702 000007 MOV #7,R2 ;LOAD LOOP COUNTER WITH A 7
2441 024666 010120 2$: MOV R1,(R0)+ ;MAP USER PAR'S TO PAGES 0-6 (4K EACH)
2442 024670 062701 000200 ADD #200,R1
2443 024674 077204 SOB R2,2$ ;LOOP UNTIL UIPAR0-UIPAR6 ARE LOADED
2444 024676 012710 177600 MOV #177600,(R0) ;MAP USER PAR7 TO THE I/O PAGE
2445 024702 012700 177600 MOV #UIPDR0,R0 ;LOAD ADDRESS OF FIRST USER PDR IN R0
2446 024706 012701 077406 MOV #77406,R1 ;LOAD PDR DATA INTO R1
2447 024712 012702 000010 MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8
2448 024716 010120 3$: MOV R1,(R0)+ ;MAP ALL 8 PAGES 128 BLOCKS, UPWARD
2449 024720 077202 SOB R2,3$ ; EXPANDABLE, READ/WRITE
2450 024722 012737 025010 001110 MOV #5$,$LPERR ;SET LOOP ON ERROR POINTER TO 5$
2451 024730 012737 025274 000250 MOV #9$,MMVEC ;SET M. M. TRAP VECTOR TO 9$
2452 024736 052737 000020 172516 BIS #BIT4,SR3 ;SET UP FOR 22 BIT ADDRESSING
2453 024744 012737 000001 177572 MOV #BIT0,SR0 ;TURN ON MEMORY MANAGEMENT
2454 024752 105037 177610 10$: CLRB UIPDR4 ;MAP USER SPACE NON-RESIDENT WHILE
2455 024756 105037 177612 CLRB UIPDR5 ; TESTING KERNEL SPACE
2456 024762 010537 177650 MOV R5,UIPAR4 ;MAP USER PAR'S OPPOSITE OF KIPAR'S
2457 024766 010437 177652 MOV R4,UIPAR5
2458 024772 013737 177776 001176 4$: MOV PSW,$TMP0 ;SAVE PSW IN CASE OF ERROR
2459 025000 012700 100100 MOV #100100,R0 ;PUT VIRTUAL ADDR. THAT USES PAR4 IN R0
2460 025004 012701 120000 MOV #120000,R1 ;PUT VIRTUAL ADDR. THAT USES PAR5 IN R1
2461 025010 010010 5$: MOV R0,(R0) ;WRITE TO TEST LOC. USING PAR4
2462 025012 011102 MOV (R1),R2 ;READ THE SAME LOC., BUT USING PAR5
2463 025014 020002 CMP R0,R2 ;DID WE READ WHAT WE WROTE?
2464 025016 001411 BEQ 6$ ;BRANCH IF YES
2465 025020 010137 001310 MOV R1,VIRT2 ;SAVE VIRTUAL ADDR. THAT SELECTED PAR5
2466 025024 010037 001306 MOV R0,VIRT1 ;SAVE VIRTUAL ADDR. THAT SELECTED PAR4
2467 025030 004737 035620 JSR PC,FORMPA ;GO FORM PHYSICAL ADDRESS BEING USED
2468 025034 104020 ERROR 20 ;READING LOC. USING PAR5 AND A VIRT.
2469 ;ADDR. DID NOT FIND DATA WRITTEN WHEN USING
2470 ;PAR4 AND VIRT. ADDRESS.
2471 ;FOR TIGHTER SCOPE LOOP
2472 ;REPLACE ERROR CALL WITH
2473 ;'BR 5$' = 000765
2474 025036 013700 001306 MOV VIRT1,R0 ;RESTORE VBA IN R0
2475 025042 062700 000100 6$: ADD #100,R0 ;CHANGE VIRTUAL ADDRS. TO POINT TO NEXT BLOCK
2476 025046 062701 000100 ADD #100,R1
2477 025052 020127 127700 CMP R1,#127700 ;WERE BLOCKS FROM 60000-67600 ALL TRIED?
  
```

```

2478 025056 001354          BNE      5$          ;BRANCH IF NO
2479 025060 032737 140000 177776 BIT      #140000,PSW ;HAVE WE DONE TEST IN USER MODE YET?
2480 025066 001026          BNE      7$          ;BRANCH IF YES
2481 025070 010437 177650  MOV      R4,UIPAR4  ;LOAD USER PAR4
2482 025074 010537 177652  MOV      R5,UIPAR5  ;LOAD USER PAR5
2483 025100 112737 000006 177610 MOV      #6,UIPDR4  ;MAP USER SPACE R/W TO TEST IT
2484 025106 112737 000006 177612 MOV      #6,UIPDR5
2485 025114 105037 172310  CLRB    KIPDR4      ;MAP KERNEL SPACE NON-RESIDENT WHILE
2486 025120 105037 172312  CLRB    KIPDR5      ;      TESTING USER SPACE
2487 025124 010537 172350  MOV      R5,KIPAR4  ;MAP KERNEL PAR'S OPPOSITE UIPAR'S
2488 025130 010437 172352  MOV      R4,KIPAR5
2489 025134 012737 140000 177776 MOV      #140000,PSW ;GO TO USER MODE
2490 025142 000713          BR       4$          ;GO BACK AND READ/WRITE IN USER MODE
2491 025144 005037 177776 7$: CLR     PSW        ;GO BACK TO KERNEL MODE BEFORE LEAVING
2492
2493 025150 020537 001260  CMP     R5,@#$MADR1 ;HAVE WE CHECKED ALL MEMORY?
2494 025154 002020          BGE     8$          ;BRANCH IF YES
2495 025156 062704 001000  ADD     #1000,R4    ;LOAD WITH NEXT VALUE FOR PAR4
2496 025162 062705 001000  ADD     #1000,R5    ;LOAD WITH NEXT VALUE FOR PAR5
2497 025166 010437 172350  MOV     R4,KIPAR4   ;LOAD KERNAL PAR4
2498 025172 010537 172352  MOV     R5,KIPAR5   ;LOAD KERNAL PAR5
2499 025176 112737 000006 172310 MOV      #6,KIPDR4  ;MAP KERNAL SPACE R/W WHILE TESTING
2500 025204 112737 000006 172312 MOV      #6,KIPDR5
2501 025212 000137 024752  JMP     10$         ;CONTINUE TEST
2502
2503 025216 012737 077406 172310 8$: MOV     #77406,KIPDR4 ;REMAP KERNEL PAGES READ/WRITE
2504 025224 012737 077406 172312 MOV     #77406,KIPDR5
2505 025232 012705 000600  MOV     #600,R5
2506 025236 010537 172350  MOV     R5,KIPAR4   ;MAP KERNEL AND USER PAR'S 4 & 5
2507 025242 010537 172352  MOV     R5,KIPAR5   ; BACK TO 12-16K
2508 025246 010537 177650  MOV     R5,UIPAR4
2509 025252 010537 177652  MOV     R5,UIPAR5
2510 025256 012737 002150 000250 MOV     #MGMERR,MMVEC ;RESTORE ADDR. OF NORMAL M.M. TRAP ROUTINE
2511 025264 012737 024630 001110 MOV     #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
2512 025272 000427          BR       TST26      ;:GO TO NEXT TEST
2513
2514 025274 012637 001266 9$: MOV     (KSP)+,TRAPPC ;SAVE PC & PS OF TRAP
2515 025300 012637 001270  MOV     (KSP)+,TRAPPS
2516
2517
2518
2519 025304 010037 001306  MOV     R0,VIRT1    ;PROGRAM WILL TRAP TO HERE IF TRY
2520 025310 004737 035620  JSR     PC,FORMPA   ;TO USE USER PDR'S WHEN IN KERNEL MODE
2521 025314 013737 177572 001272 MOV     SR0,WASSR0  ;OR KERNEL PDR'S WHEN IN USER MODE
2522 025322 013737 177576 001274 MOV     SR2,WASSR2  ;SAVE SR0 & SR2 FOR ERROR REPORT
2523 025330 042737 160000 177572 BIC     #160000,SR0 ;CLEAR ERROR BITS IN SR0
2524 025336 104052          ERROR    52        ;M.M. TRAP WHILE IN RELOCATE MODE -
2525
2526
2527
2528
2529 025340 013746 001270  MOV     TRAPPS,-(KSP) ;REFERENCED WRONG SET OF PDR'S
2530 025344 013746 001266  MOV     TRAPPC,-(KSP) ;FOR TIGHTER SCOPE LOOP
2531 025350 000002          RTI              ;REPLACE ERROR CALL WITH
2532
2533

```

```
2534 .....
2535 :*TEST 26 W-BIT LOGIC TEST, KERNEL PDR'S
2536 :*
2537 :* THIS TEST WRITES TO EIGHT (8) DIFFERENT VIRTUAL ADDRESSES
2538 :* (VBA'S = 17776,37776,57776,77776,117776,137776,157776, & 177776
2539 :* & PBA'S CONSTRUCTED = 17776,37776,57776,77776,77776,
2540 :* 77776,77776, & 777776 RESPECTIVELY).
2541 :* WHICH SHOULD CAUSE THE 'W-BIT' TO SET IN EACH OF THE
2542 :* EIGHT (8) KERNEL PAGE DESCRIPTOR REGISTERS. THE PDR'S
2543 :* ARE CHECKED TO SEE THAT IT'S W-BIT DOES SET WHEN THE
2544 :* PAGE IT IS MAPPED TO IS WRITTEN TO AND THAT THE W-BIT
2545 :* DOES NOT SET IN ANY OF THE OTHER PDR'S. KERNEL PDR'S 3,4,5,6
2546 :* ARE MAPPED TO 12-16K FOR THIS TEST. ALSO THE W-BIT
2547 :* SHOULD BE CLEARED WHEN THE PDR IS WRITTEN TO. THE
2548 :* W-BIT PORTION OF THE PDR'S IS BEING CHECKED.
2549 .....
2550 TST26: SCOPE
2551 025352 000004 177776 1$: MOV #340,PSW ;LOCK OUT ALL POSSIBLE INTERRUPTS
2552 025354 012737 000340 JSR PC,TOFF ;TURN T-BIT TRAPPING OFF FOR THIS TEST
2553 025362 004737 035246 MOV #4,R2 ;SET LOOP COUNTER TO 4
2554 025366 012702 000004 MOV #KIPAR3,R0 ;LOAD ADDRESS OF PAR3 INTO R0
2555 025372 012700 172346 MOV #600,R1 ;LOAD '12-16K' PAR VALUE INTO R1
2556 025402 010120 000600 2$: MOV R1,(R0)+ ;MAP PARS 3-6 TO 12-16K
2557 025404 077202 SOB R2,2$ ;LOOP TIL ALL 4 OF THEM LOADED
2558 025406 012705 172300 MOV #KIPDR0,R5 ;LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
2559 025412 012704 000010 MOV #10,R4 ;SET LOOP COUNTER TO 8
2560 025416 012703 017776 MOV #17776,R3 ;INITIALIZE VIRTUAL ADDRESS TO BE IN R3
2561 025422 012737 025430 001110 MOV #3$,$LPERR ;SET LOOP ON ERROR POINTER TO 3$
2562 025430 012700 172300 3$: MOV #KIPDR0,R0 ;LOAD ADDR. OF FIRST PDR TO BE SETUP IN R0
2563 025434 012702 000010 MOV #10,R2 ;SET LOOP COUNTER TO 8
2564 025440 012701 077406 MOV #77406,R1 ;PUT 'W-BIT OFF DATA' INTO R1
2565 025444 010120 4$: MOV R1,(R0)+ ;CLEAR ALL W-BITS BY WRITING TO ALL PDRS
2566 025446 077202 SOB R2,4$ ;LOOP UNTIL ALL OF THEM SETUP
2567 025450 011313 MOV (R3),(R3) ;DO 'DATO' TO VIRTUAL ADDR.-SETTING A W-BIT
2568 025452 031527 000100 BIT (R5),#WBIT ;DID THAT CAUSE W-BIT TO BE SET?
2569 025456 001002 BNE 5$ ;BRANCH IF YES
2570 025460 104021 ERROR 21 ;W-BIT DID NOT GET SET IN PDR
2571 ;FOR TIGHTER SCOPE LOOP
2572 ;REPLACE ERROR CALL WITH
2573 ;'BR 3$' = 000763
2574 025462 000422 BR 8$ ;SKIP CHECKING OTHER PDR'S-ERROR WILL SET W-BITS
2575 025464 012702 000010 5$: MOV #10,R2 ;SET LOOP COUNTER TO 8
2576 025470 012700 172300 MOV #KIPDR0,R0 ;LOAD ADDR. OF FIRST PDR TO BE CHECKED IN R0
2577 025474 031027 000100 6$: BIT (R0),#WBIT ;DID W-BIT IN OTHER PDRS REMAIN CLEAR?
2578 025500 001403 BEQ 7$ ;BRANCH IF YES
2579 025502 020500 CMP R5,R0 ;IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
2580 025504 001401 BEQ 7$ ;BRANCH IF YES
2581 025506 104022 ERROR 22 ;W-BIT GOT SET IN MORE THAN ONE PDR
2582 ;FOR TIGHTER SCOPE LOOP
2583 ;REPLACE ERROR CALL WITH
2584 ;'BR 3$' = 000750
2585 025510 062700 000002 7$: ADD #2,R0 ;POINT R0 TO NEXT PDR TO BE CHECKED
2586 025514 077211 SOB R2,6$ ;LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
2587 025516 010115 MOV R1,(R5) ;WRITE TO THE PDR TESTED TO CLEAR W-BIT
2588 025520 031527 000100 BIT (R5),#WBIT ;DID WRITING PDR CLEAR THE W-BIT?
2589 025524 001401 BEQ 8$ ;BRANCH IF YES
```

```
2590 025526 104023          ERROR 23          ;W-BIT DID NOT CLEAR BY WRITING THE PDR
2591                                     ;FOR TIGHTER SCOPE LOOP
2592                                     ;REPLACE ERROR CALL WITH
2593                                     ;'BR 3$' = 000740
2594 025530 062705 000002      8$:  ADD      #2,R5          ;POINT R5 TO THE NEXT PDR TO BE TESTED
2595 025534 062703 020000      ADD      #20000,R3       ;CHANGE VIRT. ADDR TO REF. NEXT PDR
2596 025540 077445              SOB      R4,3$          ;LOOP BACK TO 3$ UNTIL ALL 8 PDR'S TESTED
2597 025542 012737 025354 001110  MOV      #1$,$LPERR     ;RESET LOOP ON ERROR POINTER TO 1$
2598 025550 004737 035302      JSR      PC,TON         ;TURN T-BIT BACK ON FOR NEXT TEST
2599
2600                                     ;*****
2601 *TEST 27          W-BIT LOGIC TEST, USER PDR'S
2602 *
2603 * THIS TEST WRITES TO EIGHT (8) DIFFERENT VIRTUAL ADDRESSES
2604 * (VBA'S = 17776,37776,57776,77776,117776,137776,157776, & 177776
2605 * & PBA'S CONSTRUCTED = 17776,37776,57776,77776,77776,
2606 * 77776,77776, & 777776 RESPECTIVELY).
2607 * WHICH SHOULD CAUSE THE 'W-BIT' TO SET IN EACH OF THE
2608 * EIGHT (8) USER PAGE DESCRIPTOR REGISTERS. THE PDR'S
2609 * ARE CHECKED TO SEE THAT IT'S W-BIT DOES SET WHEN THE
2610 * PAGE IT IS MAPPED TO IS WRITTEN TO AND THAT THE W-BIT
2611 * DOES NOT SET IN ANY OF THE OTHER PDR'S. USER PDR'S 3,4,5,6
2612 * ARE MAPPED TO 12-16K FOR THIS TEST. ALSO THE W-BIT
2613 * SHOULD BE CLEARED WHEN THE PDR IS WRITTEN TO. THE
2614 * W-BIT PORTION OF THE PDR'S IS BEING CHECKED.
2615 *
2616 *
2616 025554 000004          TST27: SCOPE
2617 025556 012737 140000 177776 1$:  MOV      #140000,PSW     ;GO TO USER MODE FOR THIS TEST
2618 025564 004737 035246      JSR      PC,TOFF        ;TURN T-BIT TRAPPING OFF FOR THIS TEST
2619 025570 012702 000004      MOV      #4,R2          ;SET LOOP COUNTER TO 4
2620 025574 012700 177646      MOV      #UIPAR3,R0     ;LOAD ADDRESS OF PAR3 INTO R0
2621 025600 012701 000600      MOV      #600,R1        ;LOAD '12-16K' PAR VALUE INTO R1
2622 025604 010120          2$:  MOV      R1,(R0)+       ;MAP PARS 3-6 TO 12-16K
2623 025606 077202          SOB      R2,2$          ;LOOP TIL ALL 4 OF THEM LOADED
2624 025610 012705 177600      MOV      #UIPDR0,R5     ;LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
2625 025614 012704 000010      MOV      #10,R4         ;SET LOOP COUNTER TO 8
2626 025620 012703 017776      MOV      #17776,R3      ;INITIALIZE VIRTUAL ADDRESS TO BE IN R3
2627 025624 012737 025632 001110  MOV      #3$,$LPERR     ;SET LOOP ON ERROR POINTER TO 3$
2628 025632 012700 177600      3$:  MOV      #UIPDR0,R0     ;LOAD ADDR. OF FIRST PDR TO BE SETUP IN R0
2629 025636 012702 000010      MOV      #10,R2         ;SET LOOP COUNTER TO 8
2630 025642 012701 077406      MOV      #77406,R1      ;PUT 'W-BIT OFF DATA' INTO R1
2631 025646 010120          4$:  MOV      R1,(R0)+       ;CLEAR ALL W-BITS BY WRITING TO ALL PDRS
2632 025650 077202          SOB      R2,4$          ;LOOP UNTIL ALL OF THEM SETUP
2633 025652 011313          MOV      (R3),(R3)      ;DO 'DATO' TO VIRTUAL ADDR.-SETTING A W-BIT
2634 025654 031527 000100      BIT      (R5),#WBIT     ;DID THAT CAUSE W-BIT TO BE SET?
2635 025660 001002          BNE      5$             ;BRANCH IF YES
2636 025662 104021          ERROR 21              ;W-BIT DID NOT GET SET IN PDR
2637                                     ;FOR TIGHTER SCOPE LOOP
2638                                     ;REPLACE ERROR CALL WITH
2639                                     ;'BR 3$' = 000763
2640 025664 000422          BR      8$              ;SKIP CHECKING OTHER PDR'S-ERROR WILL SET W-BITS
2641 025666 012702 000010      5$:  MOV      #10,R2         ;SET LOOP COUNTER TO 8
2642 025672 012700 177600      MOV      #UIPDR0,R0     ;LOAD ADDR. OF FIRST PDR TO BE CHECKED IN R0
2643 025676 031027 000100      6$:  BIT      (R0),#WBIT     ;DID W-BIT IN OTHER PDRS REMAIN CLEAR?
2644 025702 001403          BEQ      7$             ;BRANCH IF YES
2645 025704 020500          CMP      R5,R0         ;IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
```

```
2646 025706 001401 BEQ 7$ :BRANCH IF YES
2647 025710 104022 ERROR 22 :W-BIT GOT SET IN MORE THAN ONE PDR
2648 :FOR TIGHTER SCOPE LOOP
2649 :REPLACE ERROR CALL WITH
2650 :'BR 3$' = 000750
2651 025712 062700 000002 7$: ADD #2,R0 :POINT R0 TO NEXT PDR TO BE CHECKED
2652 025716 077211 SOB R2,6$ :LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
2653 025720 010115 MOV R1,(R5) :WRITE TO THE PDR TESTED TO CLEAR W-BIT
2654 025722 031527 000100 BIT (R5),#WBIT :DID WRITING PDR CLEAR THE W-BIT?
2655 025726 001401 BEQ 8$ :BRANCH IF YES
2656 025730 104023 ERROR 23 :W-BIT DID NOT CLEAR BY WRITING THE PDR
2657 :FOR TIGHTER SCOPE LOOP
2658 :REPLACE ERROR CALL WITH
2659 :'BR 3$' = 000740
2660 025732 062705 000002 8$: ADD #2,R5 :POINT R5 TO THE NEXT PDR TO BE TESTED
2661 025736 062703 020000 ADD #20000,R3 :CHANGE VIRT. ADDR TO REF. NEXT PDR
2662 025742 077445 SOB R4,3$ :LOOP BACK TO 3$ UNTIL ALL 8 PDR'S TESTED
2663 025744 012737 025556 001110 MOV #1$,$LPERR :RESET LOOP ON ERROR POINTER TO 1$
2664 025752 004737 035302 JSR PC,TON :TURN T-BIT BACK ON FOR NEXT TEST
2665 025756 005037 177776 CLR PSW :BACK TO KERNEL MODE BEFORE LEAVING
2666
2667 :*****
2668 :*TEST 30 TEST 'W-BIT' SPECIAL CASES
2669 :*
2670 :* THIS TEST CHECKS TWO SPECIAL CASES OF THE W-BIT. FIRST CASE IS
2671 :* THAT THE W-BIT SHOULD NOT SET IN PDR 7 WHEN WRITING TO
2672 :* STATUS REG SRO (KERNEL PDR 7 IS USED). SECOND CASE IS THAT
2673 :* THE W-BIT IS STILL SET IF THE 'DATO' IS ABORTED DUE TO A
2674 :* TIMEOUT ERROR (KERNEL PDR6 & VIRTUAL ADDR 140000 ARE USED).
2675 :*
2676 :*****
2677 025762 000004 TST30: SCOPE
2678
2679 025764 004737 035246 1$: JSR PC,TOFF :TURN OFF T-BIT TRAPPING FOR THIS TEST
2680 025770 012701 077406 MOV #77406,R1 :PUT 'W-BIT OFF' VALUE FOR PDR IN R1
2681 025774 012737 026002 001110 MOV #2$,$LPERR :SET LOOP ON ERROR POINTER TO 2$
2682 026002 010137 172316 2$: MOV R1,KIPDR7 :LOAD KERNEL PDR 7 TO CLEAR W-BIT
2683 026006 013700 177572 MOV SRO,R0 :READ PRESENT CONTENTS OF STATUS REG. 0
2684 026012 010037 177572 MOV R0,SRO :WRITE PRESENT CONTENTS OF SRO BACK TO ITSELF
2685 026016 013702 172316 MOV KIPDR7,R2 :READ CONTENTS OF KIPDR7 INTO R2
2686 026022 020102 CMP R1,R2 :WAS W-BIT LEFT CLEARED?
2687 026024 001401 BEQ 3$ :BRANCH IF YES
2688 026026 104024 ERROR 24 :W-BIT IN KIPDR7 SET WHEN SRO WAS WRITTEN TO
2689 :FOR TIGHTER SCOPE LOOP
2690 :REPLACE ERROR CALL WITH
2691 :'BR 2$' = 000765
2692 026030 012737 026030 001110 3$: MOV #3$,$LPERR :SET LOOP ON ERROR POINTER TO 3$
2693 026036 010137 172314 MOV R1,KIPDR6 :LOAD KERNEL PDR6 WITH 77406 TO CLEAR W-BIT
2694 026042 012737 026054 000004 MOV #4$,ERRVEC :SET UP LOC. 4 TO 4$ FOR ODD ADDR. ABORT
2695 026050 005037 140000 CLR @#140000 :CAUSE TIMEOUT ABORT THRU LOC. 4
2696 026054 012706 001100 4$: MOV #KERSTK,KSP :RESTORE THE STACK POINTER
2697 026060 013702 172314 MOV KIPDR6,R2 :READ KIPDR6 INTO R2
2698 026064 052701 000100 BIS #100,R1 :R1-77506
2699 026070 020102 CMP R1,R2 :WAS W-BIT SET?
2700 026072 001401 BEQ 5$ :BRANCH IF YES
2701 026074 104025 ERROR 25 :W-BIT WAS NOT SET DURING A TIMEOUT ABORT
```

```
2702                                     :FOR TIGHTER SCOPE LOOP
2703                                     :REPLACE ERROR CALL WITH
2704                                     :'BR 3$' = 000757
2705 026076 010137 172314 5$: MOV R1,KIPDR6 :RESTORE KIPDR6 TO 77406
2706 026102 012737 001400 172354 MOV #1400,KIPAR6 :RESTORE KIPAR6 TO 1400
2707 026110 012737 002076 000004 MOV #TIMERR,ERRVEC :RESTORE NORMAL CPU TRAP ROUTINE TO LOC.4
2708 026116 012737 025764 001110 MOV #1$, $LPERR :RESET LOOP ON ERROR POINTER TO 1$
2709 026124 004737 035302 JSR PC,TON :TURN T-BIT TRAPPING BACK ON
2710
2711 :*****
2712 :*
2713 :* THE NEXT THREE (3) TESTS CAUSE MEMORY MANAGEMENT ERRORS
2714 :* TO CHECK THE ABILITY OF STATUS REGISTER 0 TO RECORD KT
2715 :* ERRORS AND THE ABILITY OF STATUS REGISTER 2 TO LOCK UP THE
2716 :* VIRTUAL ADDR. OF THE INSTRUCTION THAT CAUSED THE ERROR.
2717 :* THE BITS OF SR2 ARE CHECKED AND BITS <15:13>, <6:5>, AND <3:0>
2718 :* ARE CHECKED IN SRO. SO THE SRO AND SR2 LOGIC AND THE
2719 :* KT ERROR LOGIC ARE CHECKED.
2720 :*
2721 :*****
2722 :*****
2723 :*TEST 31 NON-RESIDENT ABORT TEST (ACF=0&4)
2724 :*
2725 :* THIS TEST CHECKS THE ACCESS CONTROL FIELD (ACF) COMPARATOR
2726 :* LOGIC BY CAUSING NON-RESIDENT ABORTS IN BOTH KERNEL AND
2727 :* USER MODES. PDR 4 IS LOADED WITH ACF'S = 0&4 AND
2728 :* THEN PHYSICAL ADDR. 60000 IS ACCESSED TO CAUSE THE ABORT.
2729 :*
2730 :*****
2731 :*****
2732 026130 000004 TST31: SCOPE
2733
2734 026132 012700 000600 1$: MOV #600,R0 :LOAD DATA FOR PAR'S INTO R0
2735 026136 010037 172346 MOV R0,KIPAR3 :MAP KERNEL PAR'S 3&4 TO 12-16K
2736 026142 010037 172350 MOV R0,KIPAR4
2737 026146 010037 177646 MOV R0,UIPAR3 :MAP USER PAR'S 3&4 TO 12-16K
2738 026152 010037 177650 MOV R0,UIPAR4
2739 026156 012737 077406 172306 MOV #77406,KIPDR3 :MAP KERNEL PDR 3 128 BLKS, READ-WRITE
2740 026164 012737 077406 177606 MOV #77406,UIPDR3 :MAP USER PDR 3 128 BLKS, READ-WRITE
2741 026172 012700 060000 MOV #60000,R0 :LOAD VIRTUAL ADDR. TO REFERENCE PDR3 INTO R0
2742 026176 012701 100000 MOV #100000,R1 :LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
2743 026202 012703 100011 MOV #100011,R3 :LOAD R3 WITH WHAT SRO SHOULD READ - N.R., KERNEL, PG.4
2744 026206 012702 077400 MOV #77400,R2 :LOAD ACF=0 (NON-RESIDENT) PDR VALUE IN R2
2745 026212 012737 026254 000250 2$: MOV #5$,MMVEC :POINT MEM. MGMT. TRAP VECTOR TO 5$ BELOW
2746 026220 010237 172310 MOV R2,KIPDR4 :LOAD ACF TEST VALUE INTO KIPDR4
2747 026224 010237 177610 MOV R2,UIPDR4 :LOAD ACF TEST VALUE INTO UIPDR4
2748 026230 012737 026236 001110 MOV #3$, $LPERR :SET LOOP ON ERROR POINTER TO 3$
2749 026236 005010 3$: CLR (R0) :CLEAR PHYS. LOC. 60000 USING PDR3
2750 026240 013737 177776 001176 MOV PSW,$TMPO :SAVE PSW IN CASE OF ERROR
2751 026246 005211 4$: INC (R1) :TRY TO REF. IT USING PDR4 - SHOULD TRAP TO 5$
2752 026250 104026 ERROR 26 :MEM. MGMT. ABORT DID NOT OCCUR
2753 :FOR TIGHTER SCOPE LOOP
2754 :REPLACE ERROR CALL WITH
2755 :'BR 3$' = 000772
2756 026252 000425 5$: BR 8$ :BRANCH AROUND STATUS REG. CHECKS IF NO ABORT
2757 026254 062706 000004 ADD #4,SP :RESTORE STACK POINTER
```

2758	026260	005710				TST	(R0)		:DID INSTRUCTION GET ABORTED & NOT EXECUTE
2759	026262	001401				BEQ	6\$:BRANCH IF YES
2760	026264	104027				ERROR	27		:INSTRUCTION WAS NOT ABORTED, LOC. GOT CHANGED
2761									:FOR TIGHTER SCOPE LOOP
2762									:REPLACE ERROR CALL WITH
2763									: 'BR 3\$' = 000764
2764	026266	013737	177572	001272	6\$:	MOV	SRO,WASSRO		:READ STATUS REGISTER 0
2765	026274	013737	177576	001274		MOV	SR2,WASSR2		:READ STATUS REGISTER 2
2766	026302	020337	001272			CMP	R3,WASSRO		:DID SRO REPORT NON-RESIDENT ERROR CORRECTLY?
2767	026306	001401				BEQ	7\$:BRANCH IF YES
2768	026310	104030				ERROR	30		:SRO DID NOT REPORT NON-RES. ERROR CORRECTLY
2769									:FOR TIGHTER SCOPE LOOP
2770									:REPLACE ERROR CALL WITH
2771									: 'BR 3\$' = 000752
2772	026312	012704	026246		7\$:	MOV	#4\$,R4		:LOAD R4 WITH WHAT SR2 SHOULD READ
2773	026316	020437	001274			CMP	R4,WASSR2		:DID SR2 LOCKUP RIGHT VIRTUAL ADDR. (=4\$)?
2774	026322	001401				BEQ	8\$:BRANCH IF YES
2775	026324	104031				ERROR	31		:SR2 DID NOT LOCK VIRTUAL ADDR. OF NON-RES. ERROR
2776									:FOR TIGHTER SCOPE LOOP
2777									:REPLACE ERROR CALL WITH
2778									: 'BR 3\$' = 000744
2779	026326	042737	160000	177572	8\$:	BIC	#160000,SRO		:CLEAR THE ERROR BITS IN SRO
2780	026334	032737	140000	001176		BIT	#140000,\$TMP0		:HAS ACF=084 BEEN TESTED IN USER YET
2781	026342	001006				BNE	9\$:BRANCH IF YES
2782	026344	012703	100151			MOV	#100151,R3		:LOAD R3 WITH WHAT SRO SHOULD READ - N.R., USER, PG.4
2783	026350	012737	140000	177776		MOV	#140000,PSW		:GO TO USER MODE
2784	026356	000715				BR	2\$:REPEAT TEST IN USER MODE
2785	026360	022702	077404		9\$:	CMP	#77404,R2		:HAS ACF=4 BEEN TESTED YET?
2786	026364	001407				BEQ	10\$:BRANCH IF YES
2787	026366	012702	077404			MOV	#77404,R2		:THEN LOAD ACF=4 (NON-RES) PDR VALUE IN R2
2788	026372	012703	100011			MOV	#100011,R3		:LOAD R3 WITH WHAT SRO SHOULD READ-N.R.,KERNEL,PG. 4
2789	026376	005037	177776			CLR	PSW		:GO BACK TO KERNEL MODE
2790	026402	000703				BR	2\$:GO BACK & TEST ACF=4 IN SAME MODE
2791	026404	005037	177776		10\$:	CLR	PSW		:GO BACK TO KERNEL MODE BEFORE LEAVING
2792	026410	012737	026132	001110		MOV	#1\$, \$LPERR		:RESET LOOP ON ERROR POINTER TO 1\$
2793	026416	012737	002150	000250		MOV	#MGMERR,MMVEC		:RESTORE ADDRESS OF NORMAL MEMORY
2794									:MANAGEMENT ERROR ROUTINE TO MMVEC
2795									
2796									
2797									
2798									
2799									
2800									
2801									
2802									
2803									
2804									
2805	026424	000004				TST32:	SCOPE		
2806	026426					1\$:			:KERNEL & USER PAR'S 3 & 4 AND PDR 3
2807									:ARE SETUP FROM LAST TEST
2808	026426	012700	060000			MOV	#60000,R0		:LOAD VIRTUAL ADDR. TO REFERENCE PDR3 INTO R0
2809	026432	012701	100000			MOV	#100000,R1		:LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
2810	026436	012703	020011			MOV	#20011,R3		:LOAD R3 WITH WHAT SRO SHOULD READ - R/O, KERNEL, PG.4
2811	026442	012702	077402			MOV	#77402,R2		:LOAD ACF=2 (READ-ONLY) PDR VALUE IN R2
2812	026446	012737	026510	000250	2\$:	MOV	#5\$,MMVEC		:POINT MEM. MGMT. TRAP VECTOR TO 5\$ BELOW
2813	026454	010237	172310			MOV	R2,KIPDR4		:LOAD ACF=2 INTO KIPDR4

2814	026460	010237	177610			MOV	R2,UIPDR4	:LOAD ACF=2 INTO UIPDR4
2815	026464	012737	026472	001110		MOV	#3\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 3\$
2816	026472	005010			3\$:	CLR	(R0)	:CLEAR PHYS. LOC. 60000 USING PDR3
2817	026474	013737	177776	001176		MOV	PSW,\$TMP0	:SAVE PSW IN CASE OF ERROR
2818	026502	005211			4\$:	INC	(R1)	:TRY TO WRITE USING PDR4 - SHOULD TRAP TO 5\$
2819	026504	104026				ERROR	26	:MEM. MGMT. ABORT DID NOT OCCUR
2820								:FOR TIGHTER SCOPE LOOP
2821								:REPLACE ERROR CALL WITH
2822								: 'BR 3\$' = 000772
2823	026506	000425				BR	8\$:BRANCH AROUND STATUS REG. CHECKS IF NO ABORT
2824	026510	062706	000004		5\$:	ADD	#4,\$SP	:RESTORE STACK POINTER
2825	026514	005710				TST	(R0)	:DID INSTRUCTION GET ABORTED & NOT EXECUTE
2826	026516	001401				BEQ	6\$:BRANCH IF YES
2827	026520	104027				ERROR	27	:INSTRUCTION WAS NOT ABORTED, LOC. GOT CHANGED
2828								:FOR TIGHTER SCOPE LOOP
2829								:REPLACE ERROR CALL WITH
2830								: 'BR 3\$' = 000764
2831	026522	013737	177572	001272	6\$:	MOV	SRO,WASSRO	:READ STATUS REG. 0
2832	026530	013737	177576	001274		MOV	SR2,WASSR2	:READ STATUS REG. 2
2833	026536	020337	001272			CMP	R3,WASSRO	:DID SRO REPORT READ-ONLY ERROR CORRECTLY?
2834	026542	001401				BEQ	7\$:BRANCH IF YES
2835	026544	104030				ERROR	30	:SRO DID NOT REPORT R/O ERROR CORRECTLY
2836								:FOR TIGHTER SCOPE LOOP
2837								:REPLACE ERROR CALL WITH
2838								: 'BR 3\$' = 000752
2839	026546	012704	026502		7\$:	MOV	#4\$,R4	:LOAD R4 WITH WHAT SR2 SHOULD READ
2840	026552	020437	001274			CMP	R4,WASSR2	:DID SR2 LOCKUP RIGHT VIRTUAL ADDR. (=4\$)?
2841	026556	001401				BEQ	8\$:BRANCH IF YES
2842	026560	104031				ERROR	31	:SR2 DID NOT LOCKUP VIRTUAL ADDR. OF R/O ERROR
2843								:FOR TIGHTER SCOPE LOOP
2844								:REPLACE ERROR CALL WITH
2845								: 'BR 3\$' = 000744
2846	026562	042737	160000	177572	8\$:	BIC	#160000,SRO	:CLEAR THE ERROR BITS IN SRO
2847	026570	032737	140000	001176		BIT	#140000,\$TMP0	:HAS ACF=2 BEEN TESTED IN USER MODE?
2848	026576	001006				BNE	9\$:BRANCH IF YES
2849	026600	012703	020151			MOV	#20151,R3	:LOAD R3 WITH WHAT SRO SHOULD READ-R/O, USER, PG.4
2850	026604	012737	140000	177776		MOV	#140000,PSW	:GO TO USER MODE
2851	026612	000715				BR	2\$:REPEAT TEST IN USER MODE
2852	026614	005037	177776		9\$:	CLR	PSW	:GO BACK TO KERNEL MODE BEFORE LEAVING
2853	026620	012737	026426	001110		MOV	#1\$, \$LPERR	:RESET LOOP ON ERROR POINTER TO 1\$
2854	026626	012737	002150	000250		MOV	#MGMERR,MMVEC	:RESTORE ADDRESS OF NORMAL MEMORY
2855								:MANAGEMENT ERROR ROUTINE TO MMVEC.
2856								
2857								
2858								
2859								
2860								
2861								
2862								
2863								
2864								
2865								
2866								
2867								
2868								
2869								

*
* THE NEXT TWO (2) TESTS WILL BE CHECKING THE PAGE LENGTH
*

2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925

COMPARATORS AND SOME MORE OF THE KT ERROR DETECTION AND STATUS LOGIC. THE PAGE LENGTH FIELD (PLF) IN KERNEL PDR 4 IS VARIED AND FOR EVERY PLF, THREE (3) VIRTUAL ADDRESSES ARE READ. WHILE USING BOTH UPWARD & DOWNWARD PAGE EXPANSION, ONE OF THOSE THREE VIRTUAL ADDRESSES WILL CAUSE A 'PAGE LENGTH ABORT' WHILE THE OTHER TWO WON'T.

STATUS REGISTER 0 & 2 ARE CHECKED WHEN THE PAGE LENGTH ABORT DOES OCCUR TO SEE THAT THE ABORT IS REPORTED AND THAT THE VIRTUAL ADDRESS OF THE INSTRUCTION THAT CAUSED THE ABORT IS LOCKED UP.

:TEST 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).

```
TST33: 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$: MOV (R4)+,KIPDR4 :LOAD KIPDR4 WITH PAGE LENGTH VALUE
    TST @ (R0)+ :ACCESS VIRTUAL ADDR. (VBA < OR = PLF)
    SOB R1,2$ :NO ABORT SHOULD OCCUR!!!
    :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 33 :EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
    :FOR TIGHTER SCOPE LOOP
    :REPLACE ERROR CALL WITH
```

```

2926                                     : 'BR 5$' = 000776
2927 026756 000424                       BR      8$      : BRANCH AROUND ABORT CHECKS
2928 026760 012706 001100                 6$: MOV    #KERSTK,KSP : RESTORE STACK POINTER FOLLOWING ABORT
2929 026764 013737 177572 001272         MOV    SR0,WASSR0 : READ M.M. STATUS REG. 0
2930 026772 013737 177576 001274         MOV    SR2,WASSR2 : READ M.M. STATUS REG. 2
2931 027000 012702 040011                 MOV    #40011,R2  : PUT EXPECTED SR0 CONTENTS IN R2
2932 027004 020237 001272                 CMP    R2,WASSR0  : DID SR0 REPORT PG. LENGTH ABORT, PAGE 4, KERNEL?
2933 027010 001401                         BEQ    7$      : BRANCH IF YES
2934 027012 104034                         ERROR   34      : SR0 DID NOT REPORT PG. LENGTH ABORT CORRECTLY
2935                                     : FOR TIGHTER SCOPE LOOP
2936                                     : REPLACE ERROR CALL WITH
2937                                     : 'BR 5$' = 000757
2938 027014 012703 026752                 7$: MOV    #5$,R3   : PUT EXPECTED SR2 CONTENTS IN R3
2939 027020 020337 001274                 CMP    R3,WASSR2  : DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?
2940 027024 001401                         BEQ    8$      : BRANCH IF YES
2941 027026 104035                         ERROR   35      : SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY
2942                                     : FOR TIGHTER SCOPE LOOP
2943                                     : REPLACE ERROR CALL WITH
2944                                     : 'BR 5$' = 000751
2945 027030 042737 160000 177572         8$: BIC    #160000,SR0 : CLEAR ERROR BITS IN SR0
2946 027036 077135                         SOB    R1,4$     : DONE?...NO - GET NEXT DAL & PDR PAIR
2947 027040 000137 027112                 JMP    10$      : YES...
2948 027044 012637 001266                 9$: MOV    (KSP)+,TRAPPC : SAVE PC & PS OF TRAP
2949 027050 012637 001270                 MOV    (KSP)+,TRAPPS
2950 027054 013737 177572 001272         MOV    SR0,WASSR0 : SAVE CONTENTS OF SR0 FOR ERROR
2951 027062 013737 177576 001274         MOV    SR2,WASSR2 : SAVE CONTENTS OF SR2 FOR ERROR
2952 027070 042737 160000 177572         BIC    #160000,SR0 : CLEAR ERROR BITS IN SR0
2953 027076 104032                         ERROR   32      : GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
2954                                     : FOR TIGHTER SCOPE LOOP
2955                                     : REPLACE ERROR CALL WITH
2956                                     : A 'NOP' = 000240
2957 027100 013746 001270                 MOV    TRAPPS,-(KSP) : PUT PC & PS OF TRAP ON STACK
2958 027104 013746 001266                 MOV    TRAPPC,-(KSP)
2959 027110 000002                         RTI          : RETURN FROM UNEXPECTED ABORT
2960
2961 027112 012737 026636 001110         10$: MOV   #1$,$LPERR  : RESET LOOP ON ERROR POINTER TO 1$
2962 027120 012737 002150 000250         MOV   #MGMERR,MMVEC : RESTORE NORMAL M.M. TRAP HANDLER
2963                                     : ADDRESS TO M.M. TRAP VECTOR
2964 027126 000137 027216                 JMP    TST34
2965
2966                                     ;DAL TABLE FOR UPWARD EXPANSION (NON-ABORT CASES)
2967 027132 100000                         DALTB1: 100000
2968 027134 106100                         106100
2969 027136 102300                         102300
2970 027140 102500                         102500
2971 027142 113700                         113700
2972 027144 104600                         104600
2973 027146 117700                         117700
2974
2975                                     ;PDR TABLE FOR KPDR4 (NON-ABORT CASES)
2976 027150 000006                         PDRTB1: 000006
2977 027152 052006                         052006
2978 027154 045006                         045006
2979 027156 052006                         052006
2980 027160 074406                         074406
2981 027162 025006                         025006

```

2982 027164 077406
2983
2984
2985 027166 100100
2986 027170 110100
2987 027172 116600
2988 027174 112700
2989 027176 117000
2990 027200 117700
2991
2992
2993 027202 000006
2994 027204 030406
2995 027206 046406
2996 027210 042006
2997 027212 073406
2998 027214 077006
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013 027216 000004
3014 027220 012700 027500
3015 027224 012704 027516
3016 027230 012701 000006
3017 027234 012737 027412 000250
3018 027242 012737 027254 001110
3019 027250 012706 001100
3020
3021
3022 027254 012437 172310
3023 027260 005730
3024
3025 027262 077104
3026
3027
3028 027264 012701 000005
3029 027270 012700 027534
3030 027274 012704 027550
3031 027300 012737 027314 001110
3032 027306 012737 027326 000250
3033
3034 027314 012437 172310
3035 027320 005730
3036 027322 104033
3037

077406
:DAL TABLE (ABORT CASES)
DALTB2: 100100
110100
116600
112700
117000
117700

:PDR TABLE (ABORT CASES)
PDRTB2: 000006
030406
046406
042006
073406
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 SRO & SR2.
THE PAGE EXPANSION DIRECTION IN THIS TEST IS DOWNWARD, (THE ED BIT
(BIT 3) OF PDR4=1).

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

: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 #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
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

```

3038                                     :REPLACE ERROR CALL WITH
3039                                     :'BR 5$' = 000776
3040 027324 000424                       BR      8$                                     :BRANCH AROUND ABORT CHECKS
3041 027326 012706 001100                 6$:  MOV    #KERSTK,KSP                            :RESTORE STACK POINTER FOLLOWING ABORT
3042 027332 013737 177572 001272         MOV    SR0,WASSR0                             :READ M.M. STATUS REG. 0
3043 027340 013737 177576 001274         MOV    SR2,WASSR2                             :READ M.M. STATUS REG. 2
3044 027346 012702 040011                 MOV    #40011,R2                             :PUT EXPECTED SR0 CONTENTS IN R2
3045 027352 020237 001272                 CMP    R2,WASSR0                             :DID SR0 REPORT PG. LENGTH ABORT, PAGE 4, KERNEL?
3046 027356 001401                         BEQ    7$                                     :BRANCH IF YES
3047 027360 104034                         ERROR  34                                     :SR0 DID NOT REPORT PG. LENGTH ABORT CORRECTLY
3048                                     :FOR TIGHTER SCOPE LOOP
3049                                     :REPLACE ERROR CALL WITH
3050                                     :'BR 5$' = 000757
3051 027362 012703 027320                 7$:  MOV    #5$,R3                             :PUT EXPECTED SR2 CONTENTS IN R3
3052 027366 020337 001274                 CMP    R3,WASSR2                             :DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?
3053 027372 001401                         BEQ    8$                                     :BRANCH IF YES
3054 027374 104035                         ERROR  35                                     :SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY
3055                                     :FOR TIGHTER SCOPE LOOP
3056                                     :REPLACE ERROR CALL WITH
3057                                     :'BR 5$' = 000751
3058 027376 042737 160000 177572         8$:  BIC    #160000,SR0                         :CLEAR ERROR BITS IN SR0
3059 027404 077135                         SOB    R1,4$                                :DONE?...NO - GET NEXT DAL & PDR PAIR
3060 027406 000137 027460                 JMP    10$                                  :YES...
3061 027412 012637 001266                 9$:  MOV    (KSP)+,TRAPPC                       :SAVE PC & PS OF TRAP
3062 027416 012637 001270                 MOV    (KSP)+,TRAPPS
3063 027422 013737 177572 001272         MOV    SR0,WASSR0                             :SAVE CONTENTS OF SR0 FOR ERROR
3064 027430 013737 177576 001274         MOV    SR2,WASSR2                             :SAVE CONTENTS OF SR2 FOR ERROR
3065 027436 042737 160000 177572         BIC    #160000,SR0                         :CLEAR ERROR BITS IN SR0
3066 027444 104032                         ERROR  32                                     :GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
3067                                     :FOR TIGHTER SCOPE LOOP
3068                                     :REPLACE ERROR CALL WITH
3069                                     :A 'NOP' = 000240
3070 027446 013746 001270                 MOV    TRAPPS,-(KSP)                         :PUT PC & PS OF TRAP ON STACK
3071 027452 013746 001266                 MOV    TRAPPC,-(KSP)
3072 027456 000002                         RTI                                           :RETURN FROM UNEXPECTED ABORT
3073
3074 027460 012737 027220 001110         10$: MOV    #1$, $LPERR                          :RESET LOOP ON ERROR POINTER TO 1$
3075 027466 012737 002150 000250         MOV    #MGMERR,MMVEC                       :RESTORE NORMAL M.M. TRAP HANDLER
3076                                     :ADDRESS TO M.M. TRAP VECTOR
3077 027474 000137 027564                 JMP    TST35
3078
3079                                     ;DAL TABLE FOR DOWNWARD EXPANSION (NON-ABORT CASES)
3080 027500 117700                         DALTB3: 117700
3081 027502 111600                         111600
3082 027504 115400                         115400
3083 027506 115200                         115200
3084 027510 104000                         104000
3085 027512 113100                         113100
3086 027514 100000                         100000
3087
3088                                     ;PDR TABLE (NON-ABORT CASES)
3089 027516 077416                         PDRTB3: 77416
3090 027520 025416                         25416
3091 027522 032416                         32416
3092 027524 025416                         25416
3093 027526 003016                         03016

```

3094 027530 052416
 3095 027532 000016
 3096
 3097
 3098 027534 117600
 3099 027536 107600
 3100 027540 101100
 3101 027542 105000
 3102 027544 100700
 3103 027546 100000
 3104
 3105
 3106 027550 077416
 3107 027552 047016
 3108 027554 031016
 3109 027556 035416
 3110 027560 004016
 3111 027562 000416
 3112
 3113
 3114
 3115
 3116
 3117
 3118
 3119
 3120
 3121
 3122
 3123
 3124
 3125
 3126
 3127
 3128

52416
00016

:DAL TABLE (ABORT CASES)

DALTB4: 117600
 107600
 101100
 105000
 100700
 100000

:PDR TABLE (ABORT CASES)

PDRTB4: 77416
 47016
 31016
 35416
 04016
 00416

*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.

3129 027564 000004
 3130 027566 012737 000600 172346
 3131 027574 012737 000600 172350
 3132 027602 012737 077406 172306
 3133 027610 012737 077402 172310
 3134 027616 012700 060002
 3135 027622 012701 100002
 3136 027626 012737 027662 000250
 3137 027634 012737 027642 001110
 3138 027642 012720 010727 2\$:
 3139 027646 005020
 3140 027650 012720 000137
 3141 027654 012710 027662
 3142 027660 010107
 3143 027662 012706 001100 3\$:
 3144 027666 013737 177576 001274
 3145 027674 020137 001274
 3146 027700 001401
 3147 027702 104036
 3148
 3149

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
 ; TRANSFER PROGRAM EXECUTION TO 'PAGE 4 INSTRUCTIONS'
 3\$: 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

```
3150  
3151 027704 042737 160000 177572 4$: BIC #160000,SR0 ;'BR 2$' = 000757  
3152 027712 060101 ADD R1,R1 ;CLEAR THE ERROR BITS IN SR0  
3153 027714 010100 MOV R1,R0 ;SETUP TO FORM NEXT VIRTUAL ADDRESS  
3154 027716 052701 100000 BIS #100000,R1 ;SETUP R0 TO FORM NEXT VIRT. ADDR. TO LOAD  
3155 027722 052700 060000 BIS #60000,R0 ;FORM VIRTUAL ADDR. THAT SHOULD BE LOCKED UP NEXT  
3156 027726 020127 110000 CMP R1,#110000 ;POINT R0 TO NEXT VIRT. ADDR. TO LOAD  
3157 027732 101743 BLOS 2$ ;HAVE ALL VBA'S 100000-110000 BEEN TESTED?  
3158 ;BRANCH IF NO  
3159 027734 012737 027566 001110 5$: MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$  
3160 027742 012737 077406 172310 MOV #77406,KIPDR4 ;RESTORE PDR4 TO R/W ACCESS  
3161 027750 012737 002150 000250 MOV #MGMERR,MMVEC ;RESTORE ADDRESS OF NORMAL M.M.  
3162 ;TRAP HANDLER TO M.M. VECTOR  
3163  
3164  
3165
```

```
::*****  
: *TEST 36 MORE CHECKS OF SR0 & SR2
```

```
: *  
: * THIS TEST PERFORMS SOME ADDITIONAL CHECKS OF THE SR0 & SR2 LOGIC.  
: * FIRST IT CHECKS THAT SR2 "TRACKS" ALONG ACTING AS A VIRTUAL ADDRESS  
: * PROGRAM COUNTER. ALSO SR0 & SR2 ARE LOCKED UP BY A PAGE LENGTH  
: * ABORT, THEN WITHOUT CLEARING SR0'S ERROR BITS, A R/O ABORT IS CAUSED.  
: * SR0 & SR2 SHOULD NOT BE CHANGED BY THE SECOND ABORT AND THE  
: * INFORMATION ABOUT THE PAGE LENGTH ABORT SHOULD STILL BE LOCKED UP.  
: * IN ADDITION A "RESET" IS EXECUTED TO VERIFY THAT SR0 IS CLEARED  
: * AND SR2 IS UNLOCKED BY A RESET. AFTER MEMORY MANAGEMENT IS TURNED BACK ON,  
: * SR2 IS CHECKED TO SEE THAT IT IS TRACKING AGAIN.  
: *
```

```
::*****
```

```
3178  
3179 027756 000004 TST36: SCOPE  
3180 027760 012737 000600 172352 1$: MOV #600,KIPAR5 ;MAP KERNEL PAGE 5 TO 12-16K  
3181 027766 012737 000406 172310 MOV #406,KIPDR4 ;SETUP PDR4 FOR PAGE LENGTH ABORT  
3182 027774 012737 077402 172312 MOV #77402,KIPDR5 ;SETUP PDR5 FOR R/O ABORT  
3183 030002 012737 030010 001110 MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$  
3184 030010 013737 177576 001274 2$: MOV SR2,WASSR2 ;READ SR2 TO SEE IF ITS TRACKING  
3185 030016 012701 030010 MOV #2$,R1 ;PUT EXPECTED VIRTUAL PC IN R1  
3186 030022 020137 001274 CMP R1,WASSR2 ;DID SR2 CONTAIN VIRTUAL PC AT 2$?  
3187 030026 001401 BEQ 3$ ;BRANCH IF YES  
3188 030030 104041 ERROR 41 ;SR2 NOT TRACKING CORRECTLY  
3189 ;FOR TIGHTER SCOPE LOOP  
3190 ;REPLACE ERROR CALL WITH  
3191 ;'BR 2$' = 000767  
3192 030032 012737 030040 001110 3$: MOV #4$, $LPERR ;SET LOOP ON ERROR POINTER TO 4$  
3193 030040 013737 177576 001274 4$: MOV SR2,WASSR2 ;READ SR2 TO SEE IF ITS TRACKING  
3194 030046 012701 030040 MOV #4$,R1 ;PUT EXPECTED VIRTUAL PC IN R1  
3195 030052 020137 001274 CMP R1,WASSR2 ;DID SR2 CONTAIN VIRTUAL PC AT 4$  
3196 030056 001401 BEQ 5$ ;BRANCH IF YES  
3197 030060 104041 ERROR 41 ;SR2 NOT TRACKING CORRECTLY  
3198 ;FOR TIGHTER SCOPE LOOP  
3199 ;REPLACE ERROR CALL WITH  
3200 ;'BR 4$' = 000767  
3201 030062 012737 030070 001110 5$: MOV #6$, $LPERR ;SET LOOP ON ERROR POINTER TO 6$  
3202 030070 012737 030106 000250 6$: MOV #7$,MMVEC ;PUT ADDRESS OF 7$ IN M.M. TRAP VECTOR  
3203 030076 005037 001200 CLR $TMP1 ;CLEAR ERROR INDICATOR  
3204 030102 005237 100500 INC @#100500 ;CAUSE PAGE LENGTH ABORT - TRAP TO 7$  
3205 030106 012706 001100 7$: MOV #KERSTK,KSP ;RESTORE STACK POINTER AFTER ABORT
```

3206	030112	013737	177572	001176		MOV	SRO,\$TMP0	:SAVE SRO'S INFORMATION ON PG. LGTH. ABORT
3207	030120	013737	177576	001202		MOV	SR2,\$TMP2	:SAVE SR2'S INFORMATION ON PG. LGTH. ABORT
3208	030126	012737	030140	000250		MOV	#8\$,MMVEC	:PUT ADDRESS OF 8\$ IN M.M. TRAP VECTOR
3209	030134	005237	120000			INC	@#120000	:CAUSE R/O ABORT - TRAP TO 8\$
3210	030140	012706	001100		8\$:	MOV	#KERSTK,KSP	:RESTORE STACK POINTER AFTER ABORT
3211	030144	013737	177572	001272		MOV	SRO,WASSRO	:READ SRO FOLLOWING SECOND KT ABORT
3212	030152	013737	177576	001274		MOV	SR2,WASSR2	:READ SR2 FOLLOWING SECOND KT ABORT
3213	030160	023737	001176	001272		CMP	\$TMP0,WASSRO	:IS SRO STILL HOLDING INFO ON FIRST ABORT?
3214	030166	001402				BEQ	9\$:BRANCH IF YES
3215	030170	005237	001200			INC	\$TMP1	:SET ERROR INDICATOR
3216	030174	023737	001202	001274	9\$:	CMP	\$TMP2,WASSR2	:DOES SR2 STILL HOLD PC OF FIRST ABORT?
3217	030202	001402				BEQ	10\$:BRANCH IF YES
3218	030204	005237	001200			INC	\$TMP1	:SET ERROR INDICATOR
3219	030210	005737	001200		10\$:	TST	\$TMP1	:WERE SRO OR SR2 CHANGED BY A SECOND ABORT?
3220	030214	001401				BEQ	11\$:BRANCH IF NO
3221	030216	104037				ERROR	37	:ONE OF STATUS REGS. CHANGED BY SECOND ABORT
3222								:FOR TIGHTER SCOPE LOOP
3223								:REPLACE ERROR CALL WITH
3224								: 'BR 6\$' = 000726
3225	030220	005037	001200		11\$:	CLR	\$TMP1	:CLEAR ERROR INDICATOR
3226	030224	000005				RESET		:EXECUTE A RESET, APPLYING AN "INIT"
3227	030226	013737	177572	001272		MOV	SRO,WASSRO	:READ SRO
3228	030234	005737	001272			TST	WASSRO	:WAS SRO CLEARED BY THE RESET?
3229	030240	001402				BEQ	12\$:BRANCH IF YES


```
3230 030242 005237 001200          INC      $TMP1          ;SRO NOT CLEARED BY A RESET
3231 030246 013737 177576 001274 12$: MOV      SR2,WASSR2      ;READ SR2
3232 030254 022737 030246 001274  CMP      #12$,WASSR2    ;WAS SR2 UNLOCKED BY A RESET?
3233 030262 001402          BEQ      13$           ;BRANCH IF YES
3234 030264 005237 001200          INC      $TMP1          ;SR2 NOT UNLOCKED BY A RESET
3235 030270 005737 001200 13$: TST      $TMP1          ;WERE SRO & SR2 BOTH 'RESET' BY A RESET?
3236 030274 001401          BEQ      14$           ;BRANCH IF YES
3237 030276 104040          ERROR    40           ;SRO OR SR2 NOT 'RESET' BY A RESET
3238                                     ;FOR TIGHTER SCOPE LOOP
3239                                     ;REPLACE ERROR CALL WITH
3240                                     ;'BR 6$' = 000676
3241 030300 012737 000001 177572 14$: MOV      #1,SRO          ;TURN MEMORY MANAGEMENT BACK ON
3242 030306 013737 177576 001274 15$: MOV      SR2,WASSR2      ;READ SR2 TO SEE IF ITS TRACKING AGAIN
3243 030314 012701 030306          MOV      #15$,R1        ;PUT EXPECTED VIRTUAL PC IN R1
3244 030320 020137 001274          CMP      R1,WASSR2      ;DID SR2 CONTAIN VIRTUAL PC AT 15$
3245 030324 001401          BEQ      16$           ;BRANCH IF YES
3246 030326 104041          ERROR    41           ;SR2 NOT TRACKING CORRECTLY
3247                                     ;FOR TIGHTER SCOPE LOOP
3248                                     ;REPLACE ERROR CALL WITH
3249                                     ;'BR 6$' = 000663
3250 030330 012737 027760 001110 16$: MOV      #1$, $LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
3251 030336 012737 077406 172310  MOV      #77406,KIPDR4  ;RESET PDR4 TO 128 BLKS, R/W
3252 030344 012737 077406 172312  MOV      #77406,KIPDR5  ;RESET PDR5 TO 128 BLKS, R/W
3253 030352 012737 002150 000250  MOV      #MGMERR,MMVEC  ;RESTORE ADDRESS OF NORMAL MEMORY
3254                                     ;MANAGEMENT TRAP ROUTINE TO M.M. VECTOR
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269 030360 000004          TST37: SCOPE
3270 030362 004737 035246 001110 1$: JSR      PC,TOFF      ;TURN OFF T-BIT TRAPPING FOR THIS TEST
3271 030366 012737 030374          MOV      #2$, $LPERR    ;SET LOOP ON ERROR POINTER TO 2$
3272 030374 005037 177776          CLR      PSW           ;GO TO KERNEL MODE
3273 030400 012706 001100          MOV      #KERSTK,KSP    ;SETUP KERNEL STACK PTR.
3274 030404 012737 000600 177640  MOV      #600,UIPARO    ;MAP USER PAGE 0 TO 12K
3275 030412 012737 030474 000004  MOV      #4$, @#4       ;LOAD KERNEL VECTOR 4 (LOC.4) WITH 4$
3276 030420 012737 000340 000006  MOV      #340, @#6      ;LOAD VECTOR+2 WITH NEW PSW
3277 030426 012737 140000 177776  MOV      #140000,PSW    ;GO TO USER MODE
3278 030434 012706 000700          MOV      #USESTK,USP    ;SETUP USER STACK PTR.
3279 030440 012737 030460 000004  MOV      #3$, @#4       ;LOAD USER VECTOR 4 (LOC. 60004) WITH 3$
3280 030446 012737 000340 000006  MOV      #340, @#6      ;LOAD VECTOR+2 WITH NEW PSW
3281 030454 005737 160000          TST      160000        ;CAUSE TIMEOUT ERROR TRAP TO '4'
3282                                     ;SHOULD PICK UP NEW PC=4$ FROM KERNEL
3283                                     ;LOC. 4, NOT PC=3$ FROM USER LOC. 4 (=60004)
3284 030460 013701 177776 3$: MOV      PSW,R1        ;SAVE PSW FOR ERROR
3285 030464 010602          MOV      SP,R2         ;SAVE VALUE OF STACK POINTER FOR ERROR
```

```
3286 030466 005037 177776 CLR PSW ;BE SURE BACK IN KERNEL MODE
3287 030472 104042 ERROR 42 ;DID NOT TRAP THRU KERNEL SPACE
3288 ;FOR TIGHTER SCOPE LOOP
3289 ;REPLACE ERROR CALL WITH
3290 ;'BR 2$' = 000740
3291 030474 005037 177776 4$: CLR PSW ;BE SURE BACK IN KERNEL MODE
3292 030500 012706 001100 MOV #KERSTK,KSP ;RESTORE KERNEL S.P. IN CASE IT CHANGED
3293 030504 005037 177640 CLR UIPARO ;REMAP USER PAGE 0 TO 0-4K
3294 030510 012737 140000 177776 MOV #140000,PSW ;GO TO USER MODE
3295 030516 012706 000700 MOV #USESTK,USP ;RESTORE USER STACK POINTER
3296 030522 005037 177776 CLR PSW ;GO BACK TO KERNEL MODE
3297 030526 012737 002076 000004 MOV #TIMERR,@#4 ;RESTORE ADDR. OF NORMAL CPU TRAP HANDLER TO 4
3298 030534 012737 030362 001110 MOV #1$,$LPERR ;RESET LOOP ON ERROR POINTER TO 1$
3299 030542 004737 035302 JSR PC,TON ;TURN T-BIT TRAPPING BACK ON
3300
3301 ;:*****
3302 ;*TEST 40 RTI IN USER MODE DOES NOT CHANGE PSW
3303 ;*
3304 ;* THIS TEST CHECKS TO SEE THAT WHEN AN RTI IS EXECUTED IN USER
3305 ;* MODE, THE MODE OR PRIORITY BITS OF THE PSW ARE NOT CHANGED.
3306 ;*
3307 ;:*****
3308 030546 000004 TST40: SCOPE
3309
3310 030550 012737 030562 001110 1$: MOV #2$,$LPERR ;SET LOOP ON ERROR POINTER TO 2$
3311 030556 012702 170000 MOV #170000,R2 ;LOAD 'PRESENT & EXPECTED' PSW VALUE INTO R2
3312 030562 010237 177776 2$: MOV R2,PSW ;GO TO USER MODE-PRIORITY 0
3313 030566 012746 000340 MOV #340,-(SP) ;PUT A NEW PSW (PRIORITY=7) ON STACK
3314 030572 012746 030600 MOV #3$,-(SP) ;PUT NEW PC ON THE STACK
3315 030576 000002 RTI ;DO AN RTI FROM USER MODE
3316 030600 013701 177776 3$: MOV PSW,R1 ;READ NEW PSW INTO R1
3317 030604 042701 007437 BIC #7437,R1 ;MASK OFF COND. CODE, T-BIT, AND UNUSED BITS
3318 030610 005037 177776 CLR PSW ;GO BACK TO KERNEL MODE
3319 030614 020201 CMP R2,R1 ;DID PSW STAY IN USER, PRIORITY=0?
3320 030616 001401 BEQ 4$ ;BRANCH IF YES
3321 030620 104060 ERROR 60 ;PSW CHANGED BY AN RTI FROM USER
3322 ;FOR A TIGHTER SCOPE LOOP
3323 ;REPLACE ERROR CALL WITH
3324 ;'BR=2$' = 000760
3325 030622 012737 030550 001110 4$: MOV #1$,$LPERR ;RESET LOOP ON ERROR POINTER TO 1$
3326
3327
```

3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383

```
*****  
*TEST 41          KT ERROR SERVICED BEFORE TIMEOUT ERROR  
*  
* THIS TEST CHECKS TO SEE THAT IF A CERTAIN VIRTUAL ADDRESS THAT  
* WOULD CAUSE A MEMORY MANAGEMENT ERROR CAUSES A TIMEOUT  
* ERROR FIRST, THE TIMEOUT ERROR IS SERVICED BUT THE MEMORY  
* MANAGEMENT ERROR ISN'T. THIS MEANS THAT SRO AND SR2  
* SHOULD NOT REPORT THE ERROR OR LOCK UP ITS VIRTUAL ADDRESS.  
* A READ-ONLY VIOLATION IS USED AS THE POTENTIAL MEMORY MANAGEMENT  
* ERROR  
*  
*****  
TST41:  SCOPE  
1$:  MOV      #77006,R5          ;LOAD PDR7 DATA INTO R5  
      MOV      R5,KIPDR7       ;MAP PAGE 7 R/W PLF=176  
      MOV      #3$,@#4         ;SET CPU TRAP VECTOR TO ADDRESS OF 3$  
      MOV      #4$,@#250       ;SET M.M. TRAP VECTOR TO ADDRESS OF 4$  
      MOV      #2$, $LPERR     ;SET LOOP ON ERROR POINTER TO 2$  
2$:  INC      @#177700         ;CAUSE PLF ABORT AND POTENTIAL TIMEOUT  
3$:  ERROR    43              ;TRAPPED THRU CPU TRAP VECTOR BUT SHOULDN'T HAVE  
      ;FOR TIGHTER SCOPE LOOP  
      ;REPLACE ERROR CALL WITH  
      ;'BR 2$' = 000776  
4$:  MOV      #KERSTK,KSP      ;RESTORE STACK POINTER AFTER TRAPPING  
      MOV      SRO,WASSRO      ;READ STATUS REG. 0  
5$:  MOV      SR2,WASSR2      ;READ STATUS REG. 2  
      MOV      #40017,R0       ;LOAD EXPECTED SRO CONTENTS INTO R0  
      CMP      R0,WASSRO      ;SRO PLF ERROR BIT SET?  
      BEQ      6$              ;BRANCH IF YES  
      ERROR    44              ;SRO DIDN'T REPORT PLF ERROR  
      ;FOR TIGHTER SCOPE LOOP  
      ;REPLACE ERROR CALL WITH  
      ;'BR 2$' = 000741  
6$:  MOV      #2$,R1           ;LOAD EXPECTED SR2 CONTENTS INTO R1  
      CMP      R1,WASSR2      ;WAS SR2 LOCKED BY PLF ABORT?  
      BEQ      7$              ;BRANCH IF YES  
      ERROR    44              ;SR2 DIDN'T LOCK UP VIRTUAL ADDRESS  
      ;FOR TIGHTER SCOPE LOOP  
      ;REPLACE ERROR CALL WITH  
      ;'BR 2$' = 000741  
7$:  BIC      #160000,SRO      ;CLEAR ERROR BITS THAT WERE SET IN SRO  
      MOV      #TIMERR,@#4     ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER  
      MOV      #MGMERR,@#250  ;RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER  
      MOV      #77406,KIPDR7  ;REMAP PAGE 7 TO READ/WRITE PLF=177  
      MOV      #1$, $LPERR     ;RESET LOOP ON ERROR POINTER TO 1$
```

```
*****  
*TEST 42          PC & PSW SAVED FOR KT ERROR DURING SERVICE OF TIMEOUT ERROR  
*  
* THIS TEST CHECKS THE PC AND PROCESSOR STATUS WORD SAVED WHEN  
* A KT ERROR OCCURS DURING THE SECOND PUSH ON THE STACK DURING  
* SERVICING OF A TIMEOUT ERROR. DURING A 'DOUBLE ERROR'  
* SEQUENCE SUCH AS THIS, THE PSW SAVED WILL BE THE ONE PICKED UP  
* FROM VECTOR+2 (LOC. 6 IN THIS CASE) AFTER THE FIRST TRAP,  
* NOT THE PSW PRESENT BEFORE THE FIRST TRAP. SRO AND SR2
```

```
3384 :* SHOULD RECORD THE KT ERROR (A R/O VIOLATION BY THE USER STACK PTR.)
3385 :*
3386 :* NOTE THAT THE PREVIOUS MODE BITS <13:12> OF THE PSW
3387 :* WILL BE SET IN THE PSW THAT IS SAVED.
3388 :*
3389 :*****
3390 031000 000004 TST42: SCOPE
3391 031002 004737 035246 1$: JSR PC,TOFF ;TURN T-BIT TRAPPING OFF FOR THIS TEST
3392 031006 012737 000600 177646 MOV #600,UIPAR3 ;MAP USER PAGE 3 TO 12-16K
3393 031014 012737 000600 177650 MOV #600,UIPAR4 ;MAP USER PAGE 4 TO 12-16K
3394 031022 012737 077402 177606 MOV #77402,UIPDR3 ;MAP USER PAGE 3 READ-ONLY
3395 031030 012737 077406 177610 MOV #77406,UIPDR4 ;MAP USER PAGE 4 READ/WRITE
3396 031036 012737 031112 000004 MOV #4$,@#4 ;LOAD ADDRESS OF 4$ IN CPU (TIMEOUT) VECTOR
3397 031044 012737 140017 000006 MOV #140017,@#6 ;LOAD PSW THAT SHOULD BE PUT ON STACK IN VECTOR+2
3398 031052 012737 031112 000250 MOV #4$,@#250 ;LOAD ADDRESS OF 4$ IN M.M. TRAP VECTOR
3399 031060 012737 000340 000252 MOV #340,@#252 ;LOAD A KERNEL PSW IN MMVEC+2
3400 031066 012737 031074 001110 MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
3401 031074 012737 140000 177776 2$: MOV #140000,PSW ;GO TO USER MODE
3402 031102 012706 100002 MOV #100002,USP ;SET USER STACK PTR. SO SECOND PUSH IS IN PG. 3
3403 031106 005737 177700 3$: TST @#177700 ;CAUSE TIMEOUT ERROR THAT WILL CAUSE
3404 ;R/O ERROR WHEN TRY TO SAVE OLD PC
3405 031112 016601 000002 4$: MOV 2(KSP),R1 ;PUT PSW SAVED ON KERNEL STACK INTO R1
3406 031116 011603 MOV (KSP),R3 ;PUT PC SAVED ON KERNEL STACK INTO R3
3407 031120 013737 177572 001272 MOV SRO,WASSRO ;READ THE CONTENTS OF M.M. STATUS REG. 0
3408 031126 013737 177576 001274 MOV SR2,WASSR2 ;READ THE CONTENTS OF M.M. STATUS REG. 2
3409 031134 042737 160000 177572 BIC #160000,SRO ;CLEAR THE ERROR BITS IN SRO
3410 031142 005037 177776 CLR PSW ;BE SURE IN KERNEL MODE
3411 031146 012706 001100 MOV #KERSTK,KSP ;RESTORE KERNEL STACK POINTER
3412 031152 012737 140000 177776 MOV #140000,PSW ;GO TO USER MODE
3413 031160 012706 000700 MOV #USESTK,USP ;RESTORE USER STACK POINTER
3414 031164 005037 177776 CLR PSW ;GO BACK TO KERNEL MODE
3415 031170 005037 001176 CLR $TMP0 ;CLEAR ERROR INDICATOR
3416 031174 020127 170017 CMP R1,#170017 ;WAS THE PSW SAVED THE ONE PICKED UP BY THE
3417 ;TIMEOUT TRAP FROM ERRVEC+2?
3418 ;VALUE 170017 = PSW FROM LOC. 6 WITH
3419 ;PREVIOUS MODE BITS = USER
3420 031200 001402 BEQ 5$ ;BRANCH IF YES
3421 031202 005237 001176 INC $TMP0 ;WRONG PSW SAVED DURING 'DOUBLE ERROR' SEQUENCE
3422 031206 020327 031112 5$: CMP R3,#3$+4 ;WAS THE PC AT THE TIME OF THE TIMEOUT ERROR
3423 ;SAVED ON THE STACK?
3424 031212 001402 BEQ 6$ ;BRANCH IF YES
3425 031214 005237 001176 INC $TMP0 ;WRONG PC SAVED DURING TRAP SEQUENCE
3426 031220 023727 001272 020147 6$: CMP WASSRO,#20147 ;DID SRO REPORT - USER, PAGE 3, R/O ABORT?
3427 031226 001402 BEQ 7$ ;BRANCH IF YES
3428 031230 005237 001176 INC $TMP0 ;SRO DID NOT REPORT R/O ABORT
3429 031234 023727 001274 031106 7$: CMP WASSR2,#3$ ;DID SR2 LOCK UP VIRTUAL ADDR. OF LAST
3430 ;INSTRUCTION SUCCESSFULLY FETCHED?
3431 031242 001402 BEQ 8$ ;BRANCH IF YES
3432 031244 005237 001176 INC $TMP0 ;SR2 DID NOT LOCK UP ADDR. OF TIMEOUT INST.
3433 031250 005737 001176 8$: TST $TMP0 ;ANY 'ERRORS' DURING TRAP SEQUENCE?
3434 031254 001401 BEQ 9$ ;BRANCH IF NO
3435 031256 104045 ERROR 45 ;THE WRONG PC OR PSW WERE SAVED
3436 ;OR SRO OR SR2 DID NOT REPORT R/O
3437 ;ERROR DURING TIMEOUT - KT TRAP
3438 ;SEQUENCE
3439 ;FOR TIGHTER SCOPE LOOP
```

```
3440                                     :REPLACE ERROR CALL WITH
3441                                     :'BR 2$' = 000710
3442 031260 012737 002076 000004 9$:  MOV    #TIMERR,@#4      :RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
3443 031266 012737 000340 000006      MOV    #340,@#6        :RELOAD ERRVEC+2 WITH KERNEL PSW
3444 031274 012737 002150 000250      MOV    #MGMERR,@#250   :RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER
3445 031302 012737 077406 177606      MOV    #77406,UIPDR3   :REMAP USER PAGE 3 READ/WRITE
3446 031310 012737 031002 001110      MOV    #1$,$LPERR     :RESET LOOP ON ERROR POINTER TO 1$
3447 031316 004737 035302              JSR    PC,TON          :TURN T-BIT TRAPPING BACK ON
3448                                     :*****
3449                                     :*
3450                                     :*   THIS GROUP OF TESTS WILL TEST ALL THE LOGIC ASSOCIATED WITH
3451                                     :*   THE 'MOVE FROM PREVIOUS' AND MOVE TO PREVIOUS' INSTRUCTIONS.
3452                                     :*
3453                                     :*****
3454
3455
3456                                     :*****
3457 :*TEST 43      MOVE FROM PREVIOUS (USER) I-SPACE
3458                                     :*
3459                                     :*   THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT THE
3460                                     :*   PREVIOUS MODE IS CLOKED CORRECTLY
3461                                     :*   THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
3462                                     :*
3463                                     :*
3464                                     :*   IF THE CORRECT MODE (USER) IS NOT ENABLED A NON-RESIDENT ABORT
3465                                     :*   WILL OCCUR AND TRAP TO 23$, WHERE THE ERRORS ARE REPORTED.
3466                                     :*
3467                                     :*****
3468 031322 000004      TST43: SCOPE
3469 031324 005037 172340 1$:  CLR    KIPAR0          :MAP KERNEL PAGE 0 TO 0-4K
3470 031330 012737 000200 172342      MOV    #200,KIPAR1    :MAP KERNEL PAGE 1 TO 4-8K
3471 031336 012737 000400 172344      MOV    #400,KIPAR2    :MAP KERNEL PAGE 2 TO 8-12K
3472 031344 012737 000600 172346      MOV    #600,KIPAR3    :MAP KERNEL PAGE 3 TO 12-16K
3473 031352 012737 000600 172350      MOV    #600,KIPAR4    :MAP KERNEL PAGE 4 TO 12-16K
3474 031360 012737 007600 172356      MOV    #7600,KIPAR7   :MAP KERNEL PAGE 7 TO THE I/O PAGE
3475 031366 012700 077406              MOV    #77406,R0      :MAKE ALL KERNEL I-SPACE PAGES RESIDENT
3476                                     :READ/WRITE, LENGTH 200 BLOCKS
3477 031372 012702 000010              MOV    #10,R2         :SET LOOP COUNTER TO 8
3478 031376 012701 172300              MOV    #KIPDR0,R1     :PUT ADDRESS OF FIRST PDR IN R1
3479 031402 010021      2$:  MOV    R0,(R1)+        :LOAD PDR WITH 77406
3480 031404 077202      SOB    R2,2$          :LOOP TO 2$ UNTIL ALL PDRS LOADED
3481 031406 012702 000010              MOV    #10,R2         :SET LOOP COUNTER TO 8
3482 031412 012701 177600              MOV    #UIPDR0,R1     :PUT ADDRESS OF FIRST PDR IN R1
3483 031416 010021      3$:  MOV    R0,(R1)+        :LOAD PDR WITH 77406
3484 031420 077202      SOB    R2,3$          :LOOP TO 3$ UNTIL ALL PDRS LOADED
3485 031422 012737 000000 177640      MOV    #000,UIPAR0    :MAP USER I PAGE 0 TO 0-4K
3486 031430 012737 000200 177642      MOV    #200,UIPAR1    :MAP USER I PAGE 1 TO 4-8K
3487 031436 012737 000400 177644      MOV    #400,UIPAR2    :MAP USER I PAGE 2 TO 8-12K
3488 031444 012737 000600 177646      MOV    #600,UIPAR3    :MAP USER I PAGE 3 TO 12-16K
3489 031452 012737 007600 177656      MOV    #7600,UIPAR7   :MAP USER I PAGE 7 TO THE I/O PAGE
3490 031460 012737 031466 001110      MOV    #4$,$LPERR     :SET LOOP ON ERROR TO 4$
3491 031466
3492 031466 012737 077406 172310      MOV    #77406,KIPDR4  :KERNEL I-SPACE PAGE 4 READ/WRITE
3493 031474 012737 000600 172350      MOV    #600,KIPAR4    :MAP KERNEL I PAGE 4 TO 12K
3494 031502 012737 000600 177650      MOV    #600,UIPAR4    :MAP USER I PAGE 4 TO 12K
3495 031510 012700 036514              MOV    #36514,R0      :LOAD DATA PATTERN INTO R0
```

3496	031514	010037	100000			MOV	R0,@#100000	:LOAD DATA PATTERN INTO PHY 60000
3497	031520	012737	032122	000250		MOV	#23\$,MMVEC	:SET M.M. VECTOR TO 23\$
3498	031526	105037	172310			CLRB	KIPDR4	:MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
3499								:THE FOLLOWING WILL TEST DSTM=0 MFPI
3500								
3501	031532	012737	031540	001110		MOV	#5\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 5\$
3502	031540	012737	030340	177776	5\$:	MOV	#030340,PSW	:MAKE PREVIOUS MODE USER
3503	031546	006506			6\$:	MFPI	USP	:PUT USER STACK POINTER ON KERNEL
3504								:STACK
3505	031550	022706	001100			CMP	#KERSTK,KSP	:WAS SOMETHING PUSHED ON STACK AT 6\$
3506	031554	001407				BEQ	7\$:BRANCH IF NOTHING WAS PUSHED
3507	031556	012600				MOV	(KSP)+,R0	:POP KERNEL STACK INTO R0
3508	031560	012701	000700			MOV	#USESTK,R1	:EXPECTING TO GET 700 AS USP
3509	031564	020001				CMP	R0,R1	:DID YOU GET THE RIGHT POINTER?
3510	031566	001403				BEQ	8\$:BRANCH IF YOU DID
3511	031570	104046				ERROR	46	:WRONG THING WAS PUSHED ON STACK
3512								:FOR TIGHTER SCOPE LOOP
3513								:REPLACE ERROR CALL WITH
3514								: 'BR 5\$' = 000763
3515	031572	000401				BR	8\$:BRANCH TO NEXT TRY
3516	031574	104050			7\$:	ERROR	50	:NOTHING PUSHED ON STACK
3517								:FOR TIGHTER SCOPE LOOP
3518								:REPLACE ERROR CALL WITH
3519								: 'BR 5\$' = 000761
3520	031576				8\$:			:THE FOLLOWING WILL TEST DSTM=1 MFPI.
3521	031576	012737	031610	001110		MOV	#9\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 9\$
3522	031604	012700	036514			MOV	#36514,R0	:RELOAD DATA PATTERN IN R0
3523	031610	012737	030340	177776	9\$:	MOV	#030340,PSW	:MAKE PREVIOUS MODE USER
3524	031616	012702	100000			MOV	#100000,R2	:LOAD VIRTUAL ADDRESS INTO R2
3525	031622	006512				MFPI	(R2)	:READ FROM PHYSICAL 60000
3526	031624	012601				MOV	(KSP)+,R1	:POP KERNEL STACK INTO R1
3527	031626	020001				CMP	R0,R1	:WAS DATA FETCHED SAME AS STORED
3528	031630	001401				BEQ	10\$:BRANCH IF CORRECT DATA WAS FETCHED
3529	031632	104046				ERROR	46	:WRONG DATA WAS FETCHED
3530								:FOR TIGHTER SCOPE LOOP
3531								:REPLACE ERROR CALL WITH
3532								: 'BR 9\$' = 000766
3533	031634				10\$:			:THE FOLLOWING WILL TEST DSTM=2 MFPI.
3534	031634	012737	031642	001110		MOV	#11\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 11\$
3535	031642	012737	030340	177776	11\$:	MOV	#030340,PSW	:MAKE PREVIOUS MODE USER
3536	031650	012702	100000			MOV	#100000,R2	:LOAD VIRTUAL ADDRESS INTO R2
3537	031654	006522				MFPI	(R2)+	:READ FROM PHYSICAL 60000
3538	031656	012601				MOV	(KSP)+,R1	:POP KERNEL STACK INTO R1
3539	031660	020001				CMP	R0,R1	:WAS DATA FETCHED SAME AS STORED
3540	031662	001401				BEQ	12\$:BRANCH IF CORRECT DATA WAS FETCHED
3541	031664	104046				ERROR	46	:WRONG DATA WAS FETCHED
3542								:FOR TIGHTER SCOPE LOOP
3543								:REPLACE ERROR CALL WITH
3544								: 'BR 11\$' = 000766
3545	031666				12\$:			:THE FOLLOWING WILL TEST DSTM=3 MFPI.
3546	031666	012737	031674	001110		MOV	#13\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 13\$
3547	031674	012737	030340	177776	13\$:	MOV	#030340,PSW	:MAKE PREVIOUS MODE USER
3548	031702	006537	100000			MFPI	@#100000	:READ FROM PHYSICAL 60000
3549	031706	012601				MOV	(KSP)+,R1	:POP KERNEL STACK INTO R1
3550	031710	020001				CMP	R0,R1	:WAS DATA FETCHED SAME AS STORED
3551	031712	001401				BEQ	14\$:BRANCH IF CORRECT DATA WAS FETCHED

```

3552 031714 104046          ERROR 46          :WRONG DATA WAS FETCHED
3553                                     :FOR TIGHTER SCOPE LOOP
3554                                     :REPLACE ERROR CALL WITH
3555                                     :'BR 13$' = 000767
3556 031716          14$:      ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
3557 031716 012737 031724 001110  MOV #15$, $LPERR      ;SET LOOP ON ERROR POINTER TO 15$
3558 031724 012737 030340 177776 15$:  MOV #030340,PSW      ;MAKE PREVIOUS MODE USER
3559 031732 012702 100002          MOV #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3560 031736 006542          MFPI -(R2)          ;READ FROM PHYSICAL 60000
3561 031740 012601          MOV (KSP)+,R1        ;POP KERNEL STACK INTO R1
3562 031742 020001          CMP R0,R1           ;WAS DATA FETCHED SAME AS STORED
3563 031744 001401          BEQ 16$              ;BRANCH IF CORRECT DATA WAS FETCHED
3564 031746 104046          ERROR 46          :WRONG DATA WAS FETCHED
3565                                     :FOR TIGHTER SCOPE LOOP
3566                                     :REPLACE ERROR CALL WITH
3567                                     :'BR 15$' = 000766
3568 031750          16$:      ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
3569                                     :
3570                                     :
3571 031750 012737 031756 001110  MOV #17$, $LPERR      ;SET LOOP ON ERROR POINTER TO 17$
3572 031756 012737 030340 177776 17$:  MOV #030340,PSW      ;MAKE PREVIOUS MODE USER
3573 031764 012737 100000 001202  MOV #100000,$TMP2    ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
3574 031772 012702 001204          MOV #<$TMP2+2>,R2    ;LOAD ADDR. OF $TMP2+2 INTO R2
3575 031776 006552          MFPI @-(R2)          ;READ FROM PHYSICAL 60000
3576 032000 012601          MOV (KSP)+,R1        ;POP KERNEL STACK INTO R1
3577 032002 020001          CMP R0,R1           ;WAS DATA FETCHED SAME AS STORED
3578 032004 001401          BEQ 18$              ;BRANCH IF CORRECT DATA WAS FETCHED
3579 032006 104046          ERROR 46          :WRONG DATA WAS FETCHED
3580                                     :FOR TIGHTER SCOPE LOOP
3581                                     :REPLACE ERROR CALL WITH
3582                                     :'BR 17$' = 000763
3583 032010          18$:      ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
3584                                     :
3585 032010 012737 032016 001110  MOV #19$, $LPERR      ;SET LOOP ON ERROR POINTER TO 19$
3586 032016 012737 030340 177776 19$:  MOV #030340,PSW      ;MAKE PREVIOUS MODE USER
3587 032024 005002          CLR R2              ;MAKE REGISTER 2 A ZERO
3588 032026 006562 100000          MFPI 100000(R2)      ;READ FROM PHYSICAL 60000
3589 032032 012601          MOV (KSP)+,R1        ;POP KERNEL STACK INTO R1
3590 032034 020001          CMP R0,R1           ;WAS DATA FETCHED SAME AS STORED
3591 032036 001401          BEQ 20$              ;BRANCH IF CORRECT DATA WAS FETCHED
3592 032040 104046          ERROR 46          :WRONG DATA WAS FETCHED
3593                                     :FOR TIGHTER SCOPE LOOP
3594                                     :REPLACE ERROR CALL WITH
3595                                     :'BR 19$' = 000766
3596 032042          20$:      ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
3597                                     :
3598 032042 012737 032050 001110  MOV #21$, $LPERR      ;SET LOOP ON ERROR POINTER TO 21$
3599 032050 012737 030340 177776 21$:  MOV #030340,PSW      ;MAKE PREVIOUS MODE USER
3600 032056 012737 100000 001202  MOV #100000,$TMP2    ;LOAD TEST LOC. V.A. INTO $TMP2
3601 032064 012702 001202          MOV #$TMP2,R2        ;LOAD ADDRESS OF $TMP2 INTO R2
3602 032070 006572 000000          MFPI @0(R2)          ;USE $TMP2 TO FETCH VIRTUAL
3603                                     :ADDRESS OF 60000
3604 032074 012601          MOV (KSP)+,R1        ;POP KERNEL STACK INTO R1
3605 032076 020001          CMP R0,R1           ;WAS DATA FETCHED SAME AS STORED
3606 032100 001401          BEQ 22$              ;BRANCH IF CORRECT DATA WAS FETCHED
3607 032102 104046          ERROR 46          :WRONG DATA WAS FETCHED

```

```
3608                                     ;FOR TIGHTER SCOPE LOOP
3609                                     ;REPLACE ERROR CALL WITH
3610                                     ;'BR 21$' = 000762
3611 032104 012737 002150 000250 22$:  MOV    #MGMERR,MMVEC ;SET M.M. VECTOR TO NORMAL ROUTINE
3612 032112 012737 031324 001110    MOV    #1$, $LPERR ;SET LOOP POINTER TO START OF TEST
3613 032120 000423                    BR     TST44      ;:BRANCH TO NEXT TEST
3614
3615
3616 032122 012637 001266                23$:  MOV    (KSP)+,TRAPPC ;SAVE PC & PS OF TRAP
3617 032126 012637 001270                MOV    (KSP)+,TRAPPS
3618 032132 013737 177572 001272        MOV    SR0,WASSR0 ;SAVE SR0 FOR ERROR TYPEOUT
3619 032140 013737 177576 001274        MOV    SR2,WASSR2 ;SAVE SR2 FOR ERROR TYPEOUT
3620 032146 042737 160000 177572        BIC    #160000,SR0 ;CLEAR ERROR BITS IN SR0 AND LEAVE
3621 032154 104051                    ERROR  51        ;TRIED TO READ NON-RESIDENT PAGE
3622                                     ;FOR TIGHTER SCOPE LOOP
3623                                     ;REPLACE ERROR CALL WITH
3624                                     ;A 'NOP' = 000240
3625 032156 013746 001270                MOV    TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
3626 032162 013746 001266                MOV    TRAPPC,-(KSP)
3627 032166 000002                    RTI
3628
3629
3630                                     :*****
3631                                     :*TEST 44      MOVE TO PREVIOUS (USER) I-SPACE
3632                                     :*
3633                                     :*   THIS TEST USES THE 'MTPI' INSTRUCTION TO ENSURE THAT THE
3634                                     :*   PREVIOUS MODE IS CLOKED CORRECTLY
3635                                     :*   THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.
3636                                     :*
3637                                     :*
3638                                     :*   IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
3639                                     :*   WILL OCCUR AND TRAP TO 20$, WHERE THE ERRORS ARE REPORTED.
3640                                     :*
3641                                     :*****
3642 032170 000004                    TST44:  SCOPE
3643 032172 012737 077406 172310 1$:  MOV    #77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE
3644 032200 012737 077406 177610        MOV    #77406,UIPDR4 ;USER I-SPACE PAGE 4 READ/WRITE
3645 032206 012737 000600 172350        MOV    #600,KIPAR4  ;MAP KERNEL I PAGE 4 TO 12K
3646 032214 012737 000600 177650        MOV    #600,UIPAR4  ;MAP USER I PAGE 4 TO 12K
3647 032222 012737 033002 000250        MOV    #20$,MMVEC   ;SET M.M. VECTOR TO 20$
3648                                     ;THE FOLLOWING WILL TEST DSTM=0 MTPI
3649                                     ;
3650 032230 012737 030340 177776 2$:  MOV    #030340,PSW  ;MAKE PREVIOUS MODE USER
3651 032236 012746 007777                MOV    #7777,-(KSP) ;PUSH DATA ON KERNEL STACK
3652 032242 006606                    MTPI   USP         ;LOAD USER STACK POINTER
3653 032244 006506                    MFPI   USP         ;READ USER STACK POINTER
3654 032246 012601                    MOV    (KSP)+,R1    ;POP KERNEL STACK INTO R1
3655 032250 022701 007777                CMP    #7777,R1    ;WAS USER STACK POINTER CHANGED
3656 032254 001401                    BEQ    3$          ;BRANCH IF IT WAS
3657 032256 104050                    ERROR  50          ;USER STACK POINTER NOT CHANGED
3658                                     ;FOR TIGHTER SCOPE LOOP
3659                                     ;REPLACE ERROR CALL WITH
3660                                     ;'BR 2$' = 000764
3661 032260 012737 030340 177776 3$:  MOV    #030340,PSW  ;MAKE PREVIOUS MODE USER
3662 032266 012746 000700                MOV    #USESTK,-(KSP) ;GET READY TO RESTORE USER S. POINT
3663 032272 006606                    MTPI   USP         ;RESTORE USER STACK POINTER
```



```
3664 032274          4$:      ;THIS WILL TEST DSTM = 1 MTPI.
3665 032274 012737 032312 001110  MOV    #5$, $LPERR      ;SET LOOP ON ERROR POINTER TO 5$
3666 032302 012702 100000          MOV    #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3667 032306 012700 125252          MOV    #125252,R0      ;LOAD TEST DATA INTO R0
3668 032312 010046          5$:      MOV    R0,-(KSP)        ;PUSH TEST DATA ON KERNEL STACK
3669 032314 105037 172310          CLRB   KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
3670 032320 006612          MTPI   (R2)            ;LOAD TEST DATA INTO PHYSICAL 60000
3671 032322 112737 000006 172310  MOV    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
3672 032330 011201          MOV    (R2),R1        ;READ FROM ADDRESS 60000
3673 032332 020001          CMP    R0,R1          ;SEE IF DATA WAS STORED AT CORRECT PLACE
3674 032334 001401          BEQ    6$              ;BRANCH IF STORE WAS CORRECT
3675 032336 104047          ERROR  47             ;INCORRECT STORE
3676
3677
3678
3679 032340          6$:      ;THE FOLLOWING WILL TEST DSTM=2 MTPI.
3680
3681 032340 012737 032364 001110  MOV    #8$, $LPERR      ;SET LOOP ON ERROR POINTER TO 8$
3682 032346 012737 030340 177776  MOV    #030340,PSW     ;MAKE PREVIOUS MODE USER
3683 032354 012700 125252          MOV    #125252,R0      ;LOAD TEST DATA INTO R0
3684 032360 012702 100000          MOV    #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3685 032364 010046          8$:      MOV    R0,-(KSP)        ;PUSH TEST DATA ON KERNEL STACK
3686 032366 105037 172310          CLRB   KIPDR4          ;MAKE KERNEL PAGE 4 NON-RESIDENT
3687 032372 006612          MTPI   (R2)            ;LOAD TEST DATA INTO PHYSICAL 60000
3688 032374 112737 000006 172310  MOV    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
3689 032402 013701 100000          MOV    @#100000,R1     ;READ FROM ADDRESS 60000
3690 032406 020001          CMP    R0,R1          ;SEE IF DATA WAS STORED CORRECTLY
3691 032410 001401          BEQ    9$              ;BRANCH IF STORE WAS CORRECT
3692 032412 104047          ERROR  47             ;INCORRECT STORE
3693
3694
3695
3696 032414          9$:      ;THIS WILL TEST DSTM = 3 MTPI.
3697 032414 012737 032434 001110  MOV    #10$, $LPERR     ;SET LOOP ON ERROR POINTER TO 10$
3698 032422 012737 030340 177776  MOV    #030340,PSW     ;MAKE PREVIOUS MODE USER
3699 032430 012700 052525          MOV    #52525,R0       ;LOAD TEST DATA INTO R0
3700 032434 010046          10$:     MOV    R0,-(KSP)        ;PUSH TEST DATA ON KERNEL STACK
3701 032436 105037 172310          CLRB   KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
3702 032442 006637 100000          MTPI   @#100000        ;LOAD TEST DATA INTO PHYSICAL 60000
3703 032446 112737 000006 172310  MOV    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
3704 032454 013701 100000          MOV    @#100000,R1     ;READ FROM ADDRESS 60000
3705 032460 020001          CMP    R0,R1          ;SEE IF DATA WAS STORED CORRECTLY
3706 032462 001401          BEQ    11$             ;BRANCH IF STORE WAS CORRECT
3707 032464 104047          ERROR  47             ;INCORRECT STORE
3708
3709
3710
3711 032466          11$:     ;THIS WILL TEST DSTM = 4 MTPI.
3712 032466 012737 032506 001110  MOV    #12$, $LPERR     ;SET LOOP ON ERROR POINTER TO 12$
3713 032474 012737 030340 177776  MOV    #030340,PSW     ;MAKE PREVIOUS MODE USER
3714 032502 012700 125252          MOV    #125252,R0      ;LOAD TEST DATA INTO R0
3715 032506 010046          12$:     MOV    R0,-(KSP)        ;PUSH TEST DATA ON KERNEL STACK
3716 032510 012702 100002          MOV    #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3717 032514 105037 172310          CLRB   KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
3718 032520 006642          MTPI   -(R2)           ;LOAD TEST DATA INTO PHYSICAL 60000
3719 032522 112737 000006 172310  MOV    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
```

3720	032530	013701	100000		MOV	@#100000,R1	:READ FROM ADDRESS 60000
3721	032534	020001			CMP	R0,R1	:SEE IF DATA WAS STORED CORRECTLY
3722	032536	001401			BEQ	13\$:BRANCH IF STORE WAS CORRECT
3723	032540	104047			ERROR	47	:INCORRECT STORE
3724							:FOR TIGHTER SCOPE LOOP
3725							:REPLACE ERROR CALL WITH
3726							: 'BR 12\$' = 000762
3727	032542			13\$:			:THE FOLLOWING WILL TEST DSTM=5 MTPI.
3728							:
3729	032542	012737	032574	001110	MOV	#14\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 14\$
3730	032550	012737	030340	177776	MOV	#030340,PSW	:MAKE PREVIOUS MODE USER
3731	032556	012700	052525		MOV	#52525,R0	:LOAD TEST DATA INTO R0
3732	032562	012702	001204		MOV	#<\$TMP2+2>,R2	:LOAD ADDR. OF LOC. \$TMP2+2 INTO R2
3733	032566	012737	100000	001202	MOV	#100000,\$TMP2	:LOAD VIRT. ADDR. OF TEST LOC. INTO \$TMP2
3734	032574	010046			MOV	R0,-(KSP)	:PUSH TEST DATA ON KERNEL STACK
3735	032576	105037	172310		CLRB	KIPDR4	:MAKE KERNEL PAGE 4 NON-RESIDENT
3736	032602	006652			MTPI	@-(R2)	:LOAD TEST DATA INTO PHYSICAL 60000
3737	032604	112737	000006	172310	MOVB	#006,KIPDR4	:MAKE KERNEL PAGE 4 RESIDENT
3738	032612	013701	100000		MOV	@#100000,R1	:READ FROM ADDRESS 60000
3739	032616	020001			CMP	R0,R1	:SEE IF DATA WAS STORED CORRECTLY
3740	032620	001401			BEQ	15\$:BRANCH IF STORE WAS CORRECT
3741	032622	104047			ERROR	47	:INCORRECT STORE
3742							:FOR TIGHTER SCOPE LOOP
3743							:REPLACE ERROR CALL WITH
3744							: 'BR 14\$' = 000764
3745	032624						:THIS WILL TEST DSTM = 6 MTPI.
3746							:
3747	032624	012737	032646	001110	MOV	#16\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 16\$
3748	032632	012737	030340	177776	MOV	#030340,PSW	:MAKE PREVIOUS MODE USER
3749	032640	012700	052525		MOV	#52525,R0	:LOAD TEST DATA INTO R0
3750	032644	005002			CLR	R2	:MAKE REGISTER 2 ZERO
3751	032646	010046			MOV	R0,-(KSP)	:PUSH TEST DATA ON KERNEL STACK
3752	032650	105037	172310		CLRB	KIPDR4	:MAKE KERNEL I PAGE 4 NON-RESIDENT
3753	032654	006662	100000		MTPI	100000(R2)	:LOAD TEST DATA INTO PHYSICAL 60000
3754	032660	112737	000006	172310	MOVB	#006,KIPDR4	:MAKE KERNEL PAGE 4 RESIDENT
3755	032666	013701	100000		MOV	@#100000,R1	:READ FROM ADDRESS 60000
3756	032672	020001			CMP	R0,R1	:SEE IF DATA WAS STORED CORRECTLY
3757	032674	001401			BEQ	17\$:BRANCH IF STORE WAS CORRECT
3758	032676	104047			ERROR	47	:INCORRECT STORE
3759							:FOR TIGHTER SCOPE LOOP
3760							:REPLACE ERROR CALL WITH
3761							: 'BR 16\$' = 000763
3762	032700						:THE FOLLOWING WILL TEST DSTM=7 MTPI.
3763							:
3764	032700	012737	032732	001110	MOV	#18\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 18\$
3765	032706	012737	030340	177776	MOV	#030340,PSW	:MAKE PREVIOUS MODE USER
3766	032714	012700	125252		MOV	#125252,R0	:LOAD TEST DATA INTO R0
3767	032720	012737	100000	001202	MOV	#100000,\$TMP2	:LOAD VIRT. ADDR. OF TEST LOCATION
3768							: INTO LOCATION \$TMP2
3769	032726	012702	001202		MOV	#\$TMP2,R2	:LOAD ADDRESS OF \$TMP2 INTO R2
3770	032732	010046			MOV	R0,-(KSP)	:PUSH TEST DATA ON KERNEL STACK
3771	032734	105037	172310		CLRB	KIPDR4	:MAKE KERNEL PAGE 4 NON-RESIDENT
3772	032740	006672	000000		MTPI	@0(R2)	:LOAD TEST DATA INTO PHYSICAL 60000
3773	032744	112737	000006	172310	MOVB	#006,KIPDR4	:MAKE KERNEL PAGE 4 RESIDENT
3774	032752	013701	100000		MOV	@#100000,R1	:READ FROM ADDRESS 60000
3775	032756	020001			CMP	R0,R1	:SEE IF DATA WAS STORED CORRECTLY

```
3776 032760 001401 BEQ 19$ ;BRANCH IF STORE WAS CORRECT
3777 032762 104047 ERROR 47 ;INCORRECT STORE
3778 ;FOR TIGHTER SCOPE LOOP
3779 ;REPLACE ERROR CALL WITH
3780 ;'BR 18$' = 000763
3781 032764 012737 032172 001110 19$: MOV #1$, $LPERR ;SET LOOP POINTER TO START OF TEST
3782 032772 012737 002150 000250 MOV #MGMERR, MMVEC ;RESTORE M.M. VECTOR TO NORMAL ROUTINE
3783 033000 000423 BR TST45 ;:BRANCH TO NEXT TEST
3784
3785
3786 033002 012637 001266 20$: MOV (KSP)+, TRAPPC ;SAVE PC & PS OF TRAP
3787 033006 012637 001270 MOV (KSP)+, TRAPPS
3788 033012 013737 177572 001272 MOV SR0, WASSRO ;SAVE SR0 FOR ERROR TYPEOUT
3789 033020 013737 177576 001274 MOV SR2, WASSR2 ;SAVE SR2 FOR ERROR TYPEOUT
3790 033026 042737 160000 177572 BIC #160000, SR0 ;CLEAR ERROR BITS IN SR0
3791 033034 104051 ERROR 51 ;TRIED TO LOAD A N.R. PAGE 4
3792 ;FOR TIGHTER SCOPE LOOP
3793 ;REPLACE ERROR CALL WITH
3794 ;A 'NOP' = 000240
3795 033036 013746 001270 MOV TRAPPS, -(KSP) ;PUT PC & PS OF TRAP ON STACK
3796 033042 013746 001266 MOV TRAPPC, -(KSP)
3797 033046 000002 RTI ;RETURN TO TEST
3798
3799
3800
3801
```

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

```
3812 TST45: SCOPE
3813 033050 000004 1$: MOV #77406, R0 ;MAKE ALL USER I-SPACE PAGES RESIDENT
3814 033052 012700 077406 ;READ/WRITE, LENGTH 200 BLOCKS
3815 ;SET LOOP COUNTER TO 8
3816 033056 012702 000010 MOV #10, R2 ;LOAD ADDRESS OF FIRST PDR IN R1
3817 033062 012701 177600 MOV #UIPDRO, R1 ;LOAD PDR WITH 77406
3818 033066 010021 2$: MOV R0, (R1)+ ;LOOP UNTIL 8 USER PDRS LOADED
3819 033070 077202 SOB R2, 2$ ;SET LOOP ON ERROR TO 3$
3820 033072 012737 03310C 001110 MOV #3$, $LPERR ;GO TO USER MODE FOR THIS TEST
3821 033100 012737 140340 177776 3$: MOV #140340, PSW ;KERNEL I-SPACE PAGE 4 READ/WRITE
3822 033106 012737 077406 172310 MOV #77406, KIPDR4 ;MAP KERNEL I PAGE 4 TO 12K
3823 033114 012737 000600 172350 MOV #600, KIPAR4 ;MAP USER I PAGE 4 TO 12K
3824 033122 012737 000600 177650 MOV #600, UIPAR4 ;LOAD DATA PATTERN INTO R0
3825 033130 012700 036514 MOV #36514, R0 ;LOAD DATA PATTERN INTO PHY 60000
3826 033134 010037 100000 MOV R0, @#100000 ;LOAD VIRTUAL ADDRESS INTO R2
3827 033140 012702 100000 MOV #100000, R2
3828 ;THE FOLLOWING WILL TEST DSTM=0 MFPI
3829
3830 033144 012737 033546 000250 MOV #21$, MMVEC ;SET M.M. VECTOR TO 21$
3831 033152 105037 177610 CLR B UIPDR4 ;MAKE USER I-SPACE PAGE 4 NON-RESIDENT
```

3832	033156	012737	140340	177776		MOV	#140340,PSW	:MAKE PREVIOUS MODE KERNEL PRESENT USER
3833	033164	006506			4\$:	MFPI	KSP	:PUT KERNEL STACK POINTER ON USER STACK
3834	033166	022706	000700			CMP	#USESTK,USP	:WAS SOMETHING PUSHED ON STACK AT 1\$
3835	033172	001407				BEQ	5\$:BRANCH IF NOTHING WAS PUSHED
3836	033174	012600				MOV	(USP)+,R0	:POP USER STACK INTO R0
3837	033176	012701	001100			MOV	#KERSTK,R1	:EXPECTING 1100 AS KSP
3838	033202	020001				CMP	R0,R1	:DID YOU GET THE RIGHT POINTER?
3839	033204	0C1403				BEQ	6\$:BRANCH IF YOU DID
3840	033206	104046				ERROR	46	:WRONG THING WAS PUSHED ON STACK
3841								:FOR TIGHTER SCOPE LOOP
3842								:REPLACE ERROR CALL WITH
3843								: 'BR 4\$' = 000766
3844	033210	000401				BR	6\$:BRANCH TO NEXT TRY
3845	033212	104050			5\$:	ERROR	50	:NOTHING PUSHED ON STACK
3846								:FOR TIGHTER SCOPE LOOP
3847								:REPLACE ERROR CALL WITH
3848								: 'BR 4\$' = 000764
3849	033214				6\$:		:THE FOLLOWING WILL TEST	DSTM=1 MFPI.
3850	033214	012737	033222	001110		MOV	#7\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 7\$
3851	033222	012737	140340	177776	7\$:	MOV	#140340,PSW	:MAKE PREVIOUS MODE KERNEL PRESENT USER
3852	033230	012700	036514			MOV	#36514,R0	:LOAD DATA EXPECTED INTO R0
3853	033234	012702	100000			MOV	#100000,R2	:LOAD VIRTUAL ADDRESS INTO R2
3854	033240	006512				MFPI	(R2)	:READ FROM PHYSICAL 60000
3855	033242	012601				MOV	(USP)+,R1	:POP USER STACK INTO R1
3856	033244	020001				CMP	R0,R1	:WAS DATA FETCHED SAME AS STORED
3857	033246	001401				BEQ	8\$:BRANCH IF CORRECT DATA WAS FETCHED
3858	033250	104046				ERROR	46	:WRONG DATA WAS FETCHED
3859								:FOR TIGHTER SCOPE LOOP
3860								:REPLACE ERROR CALL WITH
3861								: 'BR 7\$' = 000764
3862	033252				8\$:		:THE FOLLOWING WILL TEST	DSM=2 MFPI.
3863	033252	012737	033260	001110		MOV	#9\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 9\$
3864	033260	012737	140340	177776	9\$:	MOV	#140340,PSW	:MAKE PREVIOUS MODE KERNEL PRESENT USER
3865	033266	012702	100000			MOV	#100000,R2	:LOAD VIRTUAL ADDRESS INTO R2
3866	033272	006522				MFPI	(R2)+	:READ FROM PHYSICAL 60000
3867	033274	012601				MOV	(USP)+,R1	:POP USER STACK INTO R1
3868	033276	020001				CMP	R0,R1	:WAS DATA FETCHED SAME AS STORED
3869	033300	001401				BEQ	10\$:BRANCH IF CORRECT DATA WAS FETCHED
3870	033302	104046				ERROR	46	:WRONG DATA WAS FETCHED
3871								:FOR TIGHTER SCOPE LOOP
3872								:REPLACE ERROR CALL WITH
3873								: 'BR 9\$' = 000766
3874	033304				10\$:		:THE FOLLOWING WILL TEST	DSTM=3 MFPI.
3875	033304	012737	033312	001110		MOV	#11\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 11\$
3876	033312	012737	140340	177776	11\$:	MOV	#140340,PSW	:MAKE PREVIOUS MODE KERNEL PRESENT USER
3877	033320	006537	100000			MFPI	@#100000	:READ FROM PHYSICAL 60000
3878	033324	012601				MOV	(USP)+,R1	:POP USER STACK INTO R1
3879	033326	020001				CMP	R0,R1	:WAS DATA FETCHED SAME AS STORED
3880	033330	001401				BEQ	12\$:BRANCH IF CORRECT DATA WAS FETCHED
3881	033332	104046				ERROR	46	:WRONG DATA WAS FETCHED
3882								:FOR TIGHTER SCOPE LOOP
3883								:REPLACE ERROR CALL WITH
3884								: 'BR 11\$' = 000767
3885	033334				12\$:		:THE FOLLOWING WILL TEST	DSTM=4 MFPI.
3886	033334	012737	033342	001110		MOV	#13\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 13\$
3887	033342	012737	140340	177776	13\$:	MOV	#140340,PSW	:MAKE PREVIOUS MODE DERNEL PRESENT USER

```
3888 033350 012702 100002      MOV      #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3889 033354 006542      MFPI     -(R2)          ;READ FROM PHYSICAL 60000
3890 033356 012601      MOV     (USP)+,R1      ;POP USER STACK INTO R1
3891 033360 020001      CMP     R0,R1          ;WAS DATA FETCHED SAME AS STORED
3892 033362 001401      BEQ    14$             ;BRANCH IF CORRECT DATA WAS FETCHED
3893 033364 104046      ERROR  46             ;WRONG DATA WAS FETCHED
3894                                     ;FOR TIGHTER SCOPE LOOP
3895                                     ;REPLACE ERROR CALL WITH
3896                                     ;'BR 13$' = 000766
3897 033366                                     14$: ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
3898                                     ;
3899 033366 012737 033374 001110      MOV     #15$, $LPERR    ;SET LOOP ON ERROR POINTER TO 15$
3900 033374 012737 140340 177776      MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3901 033402 012737 100000 001202      MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
3902 033410 012702 001204                                     MOV     #<$TMP2+2>,R2   ;LOAD ADDRESS OF $TMP2+2 INTO R2
3903 033414 006552      MFPI     @-(R2)        ;READ FROM PHYSICAL 60000
3904 033416 012601      MOV     (USP)+,R1      ;POP USER STACK INTO R1
3905 033420 020001      CMP     R0,R1          ;WAS DATA FETCHED SAME AS STORED
3906 033422 001401      BEQ    16$             ;BRANCH IF CORRECT DATA FETCHED
3907 033424 104046      ERROR  46             ;WRONG DATA WAS FETCHED
3908                                     ;FOR TIGHTER SCOPE LOOP
3909                                     ;REPLACE ERROR CALL WITH
3910                                     ;'BR 15$' = 000763
3911 033426                                     16$: ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
3912                                     ;
3913 033426 012737 033434 001110      MOV     #17$, $LPERR    ;SET LOOP ON ERROR POINTER TO 17$.
3914 033434 012737 140340 177776      MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3915 033442 005002      CLR     R2              ;MAKE REGISTER 2 A ZERO
3916 033444 006562 100000      MFPI     100000(R2)    ;READ FROM PHYSICAL 60000
3917 033450 012601      MOV     (USP)+,R1      ;POP USER STACK INTO R1
3918 033452 020001      CMP     R0,R1          ;WAS DATA FETCHED SAME AS STORED
3919 033454 001401      BEQ    18$             ;BRANCH IF CORRECT DATA FETCHED
3920 033456 104046      ERROR  46             ;WRONG DATA WAS FETCHED
3921                                     ;FOR TIGHTER SCOPE LOOP
3922                                     ;REPLACE ERROR CALL WITH
3923                                     ;'BR 17$' = 000766
3924 033460                                     18$: ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
3925                                     ;
3926 033460 012737 033466 001110      MOV     #19$, $LPERR    ;SET LOOP ON ERROR POINTER TO 19$
3927 033466 012737 140340 177776      MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3928 033474 012737 100000 001202      MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR. INTO $TMP2
3929 033502 012702 001202                                     MOV     #$TMP2,R2       ;LOAD ADDRESS OF $TMP2 INTO R2
3930 033506 006572 000000      MFPI     @0(R2)        ;READ FROM PHYSICAL 60000
3931 033512 012601      MOV     (USP)+,R1      ;POP USER STACK INTO R1
3932 033514 020001      CMP     R0,R1          ;WAS DATA FETCHED SAME AS STORED
3933 033516 001401      BEQ    20$             ;BRANCH IF CORRECT DATA FETCHED
3934 033520 104046      ERROR  46             ;WRONG DATA WAS FETCHED
3935                                     ;FOR TIGHTER SCOPE LOOP
3936                                     ;REPLACE ERROR CALL WITH
3937                                     ;'BR 19$' = 000762
3938 033522 012737 002150 000250      MOV     #MGMERR,MMVEC   ;SET M.M. VECTOR TO NORMAL ROUTINE
3939 033530 012737 000340 177776      MOV     #00340,PSW     ;GO BACK TO KERNEL MODE, PREVIOUS KERNEL
3940 033536 012737 033052 001110      MOV     #1$, $LPERR     ;SET LOOP POINTER TO START OF TEST
3941 033544 000423      BR      TST46          ;;BRANCH TO NEXT TEXT
3942
3943
```

```
3944 033546 012637 001266 21$: MOV (KSP)+,TRAPPC ;SAVE PC & PS OF TRAP
3945 033552 012637 001270 MOV (KSP)+,TRAPPS
3946 033556 013737 177572 001272 MOV SR0,WASSRO ;SAVE SRO FOR ERROR TYPEOUT
3947 033564 013737 177576 001274 MOV SR2,WASSR2 ;SAVE SR2 FOR ERROR TYPEOUT
3948 033572 042737 160000 177572 BIC #160000,SRO ;CLEAR ERROR BITS IN SRO
3949 033600 104051 ERROR 51 ;TRIED TO READ NON-RESIDENT PAGE
3950 ;FOR TIGHTER SCOPE LOOP
3951 ;REPLACE ERROR CALL WITH
3952 ;A 'NOP' = 000240
3953 033602 013746 001270 MOV TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
3954 033606 013746 001266 MOV TRAPPC,-(KSP)
3955 033612 000002 RTI ;RETURN TO TEST
3956
3957 ;*****
3958 ;*TEST 46 MOVE FROM/TO D-SPACE = MOVE FROM/TO I-SPACE
3959 ;*
3960 ;* THIS TEST CHECKS THAT SINCE THERE IS NO DISTINCTION
3961 ;* BETWEEN INSTRUCTION AND DATA SPACE IN THE FONZ-11
3962 ;* MFPD & MTPD SHOULD BE DECODED THE SAME AS MFPI & MTP1.
3963 ;*
3964 ;*****
3965 033614 000004 TST46: SCOPE
3966 033616 012737 030340 177776 1$: MOV #030340,PSW ;MAKE PREVIOUS MODE=USER,CURRENT=KERNEL
3967 033624 106506 MFPD USP ;MFPD SHOULD ACT LIKE MFPI PUTTING
3968 ;USER STACK POINTER ON THE KERNEL STACK
3969 033626 022706 001100 CMP #KERSTK,KSP ;WAS SOMETHING PUSHED ON KERNEL STACK?
3970 033632 001407 BEQ 2$ ;BRANCH IF NO
3971 033634 012600 MOV (KSP)+,R0 ;POP KERNEL STACK INTO R0
3972 033636 012701 00070C MOV #USESTK,R1 ;EXPECTING TO GET 700 AS USP
3973 033642 020001 CMP R0,R1 ;DID GET RIGHT POINTER VALUE?
3974 033644 001403 BEQ 3$ ;BRANCH IF YES
3975 033646 104053 ERROR 53 ;WRONG THING WAS PUSHED ON STACK
3976 ;FOR TIGHTER SCOPE LOOP
3977 ;REPLACE ERROR CALL WITH
3978 ;'BR 1$' = 000763
3979 033650 000401 BR 3$ ;BRANCH TO NEXT TRY
3980 033652 104054 2$: ERROR 54 ;NOTHING PUSHED ON STACK
3981 ;FOR TIGHTER SCOPE LOOP
3982 ;REPLACE ERROR CALL WITH
3983 ;'BR 1$' = 000761
3984 033654 012737 033662 001110 3$: MOV #4$,$LPERR ;SET LOOP ON ERROR POINTER TO 4$
3985 033662 012746 007777 4$: MOV #7777,-(KSP) ;PUSH DATA ON KERNEL STACK
3986 033666 106606 MTPD USP ;LOAD THE USER STACK POINTER
3987 033670 106506 MFPD USP ;READ USER STACK POINTER
3988 033672 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
3989 033674 022701 007777 CMP #7777,R1 ;WAS USER STACK POINTER CHANGED?
3990 033700 001401 BEQ 5$ ;BRANCH IF YES
3991 033702 104054 ERROR 54 ;USER STACK POINTER NOT CHANGED
3992 ;FOR TIGHTER SCOPE LOOP
3993 ;REPLACE ERROR CALL WITH
3994 ;'BR 4$' = 000767
3995 033704 012746 000700 5$: MOV #USESTK,-(KSP) ;GET READY TO RESTORE USER STK. PTR.
3996 033710 106606 MTPD USP ;RESTORE USER STACK POINTER
3997 033712 012737 033616 001110 MOV #1$,$LPERR ;SET LOOP POINTER TO START OF TEST
3998
3999 ;*****
```

4000
4001
4002
4003
4004
4005
4006
4007
4008 033720 000004
4009 033722 005037 177776
4010 033726 012700 001100
4011 033732 010006
4012 033734 006506
4013
4014 033736 011601
4015 033740 020001
4016
4017 033742 001401
4018 033744 104046
4019
4020
4021
4022 033746 005740
4023 033750 020600
4024 033752 001401
4025 033754 104050
4026
4027
4028
4029 033756 012706 001100

```
;*TEST 47 MOVE FROM PREVIOUS I=SPACE (PREVIOUS=CURRENT=KERNEL)
;*
;* THIS TEST CHECKS THAT IF BOTH PREVIOUS AND CURRENT MODES
;* ARE KERNEL, AND THE SOURCE MODE IS 0, THE DESTINATION
;* STACK IS NOT DECREMENTED BEFORE ACCESS.
;* 'MFPI KSP' SHOULD PUSH THE NON-DECREMENTED VALUE
;* OF KSP (1100) ONTO THE STACK (AT LOC. 1076).
;*****
TST47: SCOPE
1$: CLR @#PSW ;SET PREVIOUS = CURRENT = KERNEL
MOV #STACK,R0 ;SETUP VALUE FOR STACK POINTER
MOV R0,KSP ;LOAD STACK POINTER
MFPI KSP ;THE VALUE 'STACK' SHOULD BE PUSHED
;BEFORE BEING DECREMENTED
MOV (KSP),R1 ;READ DATA WHICH WAS PUSHED
CMP R0,R1 ;WAS THE ORIGINAL VALUE OF THE
;STACK POINTER PUSHED?
BEQ 2$ ;BRANCH IF YES
ERROR 46 ;MFPI FETCHED WRONG DATA
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 1$' = 000766
2$: TST -(R0) ;SETUP EXPECTED STACK POINTER VALUE
CMP KSP,R0 ;WAS THE STACK POINTER DECREMENTED?
BEQ 3$ ;BRANCH IF YES
ERROR 50 ;STACK NOT PUSHED BY THE MFPI
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 1$' = 000762
3$: MOV #STACK,KSP ;RESTORE STACK POINTER
```

4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048
4049
4050
4051
4052
4053
4054
4055
4056
4057
4058
4059
4060
4061
4062
4063
4064
4065
4066
4067
4068
4069
4070
4071
4072
4073
4074
4075
4076
4077
4078
4079
4080
4081
4082
4083
4084
4085

033762
033762 000004
033764 005037 001102
033770 005037 001212
033774 005237 001234
034000 042737 100000 001234
034006 005327
034010 000001
034012 003110
034014 012737
034016 000001
034020 034010
034022 104401 034030
034026 000407
034030 006412 047105 020104
034036 040520 051523 021440
034044 000
034046
034046 013746 001234
034052 104405
034054 104401 034062
034060 000425
034062 035411 047524 040524
034070 020114 051105 047522
034076 051522 051440 047111
034104 042503 046040 051501
034112 020124 052123 051101
034120 020124 052101 031040
034126 030060 020040 000
034134
034134 013746 001112
034140 104405
034142 104401 001223
034146 013700 000042
034152 001411
034154 005046
034156 012746 034164
034162 000442
034164 000005
034166 004710

.SBTTL END OF PASS ROUTINE

*INCREMENT THE PASS NUMBER (\$PASS)
*TYPE 'END OF PASS #XXXX'
*TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYY''
*WHERE XXXX AND YYYY ARE DECIMAL NUMBERS
*IF SW12=1 INHIBIT TRACE TRAP
*IF THERES A MONITOR GO TO IT
*IF THERE ISN'T JUMP TO LOOP

\$EOP:

SCOPE
CLR \$TSTNM ;;ZERO THE TEST NUMBER
CLR \$TIMES ;;ZERO THE NUMBER OF ITERATIONS
INC \$PASS ;;INCREMENT THE PASS NUMBER
BIC #100000,\$PASS ;;DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;;LOOP?

\$EOPCT:

.WORD 1
BGT \$DOAGN ;;YES
MOV (PC)+,@(PC)+ ;;RESTORE COUNTER

\$ENDCT:

.WORD 1
\$EOPCT
TYPE ,65\$;;TYPE ASCIZ STRING
BR ,64\$;;GET OVER THE ASCIZ
65\$: .ASCIZ <12><15>/END PASS #/

.EVEN

64\$:

MOV \$PASS,-(SP) ;;SAVE \$PASS FOR TYPEOUT
;;TYPE PASS NUMBER
TYPDS ;;GO TYPE--DECIMAL ASCIZ WITH SIGN
TYPE ,67\$;;TYPE ASCII STRING
BR ,66\$;;GET OVER THE ASCIZ
67\$: .ASCIZ / ;TOTAL ERRORS SINCE LAST START AT 200 /

.EVEN

66\$:

MOV \$ERTTL,-(SP) ;;SAVE \$ERTTL FOR TYPEOUT
;;TOTAL NUMBER OF ERRORS
TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
TYPE , \$CRLF ;;TYPE CARRIAGE RETURN, LINE FEED
\$GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS
BEQ DOAGIN ;;BRANCH IF NO MONITOR
CLR -(SP) ;;INSURE THE 'T' BIT IS CLEAR
MOV #\$CLR.T,-(SP) ;;SETUP FOR AN RTI OR RTT
BR \$RTRN ;;GO DO AN RTI OR RTT TO LOAD THE PSW
;;WITH A CLEARED 'T' BIT

\$CLR.T: RESET

\$ENDAD: JSR PC,(R0)

;;CLEAR THE WORLD
;;GO TO MONITOR


```
4086 034170 000240      NOP      ;;SAVE ROOM
4087 034172 000240      NOP      ;;FOR
4088 034174 000240      NOP      ;;ACT11
4089
4090 034176 013737 000004 001176 DOAGIN: MOV @#4,@#$TMP0  ;;SAVE CONTENTS OF LOCATION 4
4091 034204 012737 034222 000004      MOV #1$,@#4  ;;SET UP VECTOR IN CASE OF TRAP
4092 034212 012737 000001 164000      MOV #1,@#164000  ;;NOTIFY MULTI-TESTER OF PASS COMPLETE
4093 034220 000402      BR 2$  ;;IF NO TRAP DON'T TOUCH STACK
4094 034222 062706 000004      1$: ADD #4,SP  ;;RESET STACK IN CASE OF TRAP
4095 034226 013737 001176 000004      2$: MOV @#$TMP0,@#4  ;;RESTORE CONTENTS OF LOACTION 4
4096 034234
4097 034234 104400      TRAP      ;;PUSH OLD PSW AND PC ON STACK
4098 034236 042716 000020      BIC #20,(SP)  ;;CLEAR THE 'T' BIT
4099 034242 032777 010000 144670      BIT #BIT12,@SWR  ;;RUN WITH TRACE TRAP?
4100 034250 001005      BNE 1$  ;;BR IF NO
4101 034252 005137 034276      COM $TBIT  ;;IS IT TIME FOR TRACE TRAP
4102 034256 100402      BMI 1$  ;;BR IF NO
4103 034260 052716 000020      BIS #20,(SP)  ;;SET TRACE TRAP
4104 034264 012746 034272      1$: MOV # $LOOP,-(SP)  ;;JUMP TO START OF TEST
4105 034270 000002      $RTRN: RTI  ;;RETURN--THIS IS CHANGED TO
4106                                     ;;AN 'RTT' IF 'RTT' IS A LEGAL
4107                                     ;;INSTRUCTION
4108                                     $LOOP:
4109 034272 000137      JMP @ (PC)+  ;;RETURN
4110 034274 020464      $RTNAD: .WORD RESTRT
4111 034276 000000      $TBIT: .WORD 0  ;;'T' BIT STATE INDICATOR
4112 034300 377 377 000 $ENULL: .BYTE -1,-1,0  ;;NULL CHARACTER STRING
4113                                     .EVEN
4114                                     .SBTTL SCOPE HANDLER ROUTINE
4115
4116                                     ;;*****
4117                                     ;;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
4118                                     ;;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
4119                                     ;;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
4120                                     ;;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4121                                     ;;*SW14=1 LOOP ON TEST
4122                                     ;;*SW11=1 INHIBIT ITERATIONS
4123                                     ;;*SW09=1 LOOP ON ERROR
4124                                     ;;*SW08=1 LOOP ON TEST IN SWR<7:0>
4125                                     ;;*CALL
4126                                     ;;* SCOPE ;;SCOPE=IOT
4127
4128 034304      $SCOPE:
4129 034304 104410      CKSWR
4130 034306 032777 040000 144624 1$: BIT #BIT14,@SWR  ;;TEST FOR CHANGE IN SOFT-SWR
4131 034314 001114      BNE $OVER  ;;LOOP ON PRESENT TEST?
4132                                     ;;YES IF SW14=1
4133 034316 000416      ;#####START OF CODE FOR THE XOR TESTER#####
4134      $XTSTR: BR 6$  ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
4135                                     ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
4136 034320 013746 000004      MOV @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
4137 034324 012737 034344 000004      MOV #5,@#ERRVEC  ;;SET FOR TIMEOUT
4138 034332 005737 177060      TST @#177060  ;;TIME OUT ON XOR?
4139 034336 012637 000004      MOV (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
4140 034342 000463      BR $SVLAD  ;;GO TO THE NEXT TEST
4141 034344 022626 000004      5$: CMP (SP)+,(SP)+  ;;CLEAR THE STACK AFTER A TIME OUT
4142 034346 012637 000004      MOV (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
```

```
4142 034352 000423          BR      7$          ;;LOOP ON THE PRESENT TEST
4143 034354          6$:;#####END OF CODE FOR THE XOR TESTER#####
4144 034354 032777 000400 144556  BIT      #BIT08,@SWR  ;;LOOP ON SPEC. TEST?
4145 034362 001404          BEQ      2$          ;;BR IF NO
4146 034364 127737 144550 001102  CMPB    @SWR,$STNM   ;;ON THE RIGHT TEST? SWR<7:0>
4147 034372 001465          BEQ      $OVER      ;;BR IF YES
4148 034374 105737 001103  2$:    TSTB    $ERFLG   ;;HAS AN ERROR OCCURRED?
4149 034400 001421          BEQ      3$          ;;BR IF NO
4150 034402 123737 001115 001103  CMPB    $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
4151 034410 101015          BHI      3$          ;;BR IF NO
4152 034412 032777 001000 144520  BIT      #BIT09,@SWR  ;;LOOP ON ERROR?
4153 034420 001404          BEQ      4$          ;;BR IF NO
4154 034422 013737 001110 001106  7$:    MOV     $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
4155 034430 000446          BR      $OVER
4156 034432 105037 001103  4$:    CLRB    $ERFLG   ;;ZERO THE ERROR FLAG
4157 034436 005037 001212          CLR     $TIMES      ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
4158 034442 000415          BR      1$          ;;ESCAPE TO THE NEXT TEST
4159 034444 032777 004000 144466  3$:    BIT      #BIT11,@SWR ;;INHIBIT ITERATIONS?
4160 034452 001011          BNE     1$          ;;BR IF YES
4161 034454 005737 001234          TST    $PASS        ;;IF FIRST PASS OF PROGRAM
4162 034460 001406          BEQ     1$          ;;INHIBIT ITERATIONS
4163 034462 005237 001104          INC    $ICNT        ;;INCREMENT ITERATION COUNT
4164 034466 023737 001212 001104  CMP     $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
4165 034474 002024          BGE     $OVER       ;;BR IF MORE ITERATION REQUIRED
4166 034476 012737 000001 001104  1$:    MOV     #1,$ICNT  ;;REINITIALIZE THE ITERATION COUNTER
4167 034504 013737 034562 001212  MOV     $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
4168 034512 105237 001102  2$:    INCB    $STNM     ;;COUNT TEST NUMBERS
4169 034516 113737 001102 001232  MOVB   $STNM,$TESTN  ;;SET TEST NUMBER IN APT MAILBOX
4170 034524 011637 001106          MOV    (SP),$LPADR  ;;SAVE SCOPE LOOP ADDRESS
4171 034530 011637 001110          MOV    (SP),$LPERR  ;;SAVE ERROR LOOP ADDRESS
4172 034534 005037 001214          CLR   $ESCAPE      ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
4173 034540 112737 000001 001115  MOVB   #1,$ERMAX    ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
4174 034546 013777 001102 144366  $OVER: MOV    $STNM,@DISPLAY ;;DISPLAY TEST NUMBER
4175 034554 013716 001106          MOV    $LPADR,(SP)  ;;FUDGE RETURN ADDRESS
4176 034560 000002          RTI
4177 034562 000200          $MXCNT: 200        ;;FIXES PS
4178          .SBTTL ERROR HANDLER ROUTINE
4179
4180          ;;*****
4181          ;;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
4182          ;;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
4183          ;;*AND GO TO ERRYP ON ERROR
4184          ;;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4185          ;;*SW15=1 HALT ON ERROR
4186          ;;*SW13=1 INHIBIT ERROR TYPEOUTS
4187          ;;*SW10=1 BELL ON ERROR
4188          ;;*SW09=1 LOOP ON ERROR
4189          ;;*CALL
4190          ;;* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER
4191
4192          $ERROR:
4193          CKSWR
4194          MOV    R0,$REG0 ;;TEST FOR CHANGE IN SOFT-SWR
4195          MOV    R1,$REG1 ;;SAVE THE CONTENTS OF R0
4196          MOV    R2,$REG2 ;;SAVE THE CONTENTS OF R1
4197          MOV    R3,$REG3 ;;SAVE THE CONTENTS OF R2
4198          MOV    R3,$REG3 ;;SAVE THE CONTENTS OF R3
```

```

4198 034606 010437 001172      MOV      R4,$REG4      ;SAVE THE CONTENTS OF R4
4199 034612 010537 001174      MOV      R5,$REG5      ;SAVE THE CONTENTS OF R5
4200 034616 113737 001102 001262  MOVB     $STNM,TESTNO  ;SAVE THE TEST NUMBER
4201 034624 105237 001103      INCB     $ERFLG        ;SET THE ERROR FLAG
4202 034630 001775                BEQ      7$            ;DON'T LET THE FLAG GO TO ZERO
4203 034632 013777 001102 144302  MOV     $STNM,@DISPLAY ;DISPLAY TEST NUMBER AND ERROR FLAG
4204 034640 032777 002000 144272  BIT     #BIT10,@SWR    ;BELL ON ERROR?
4205 034646 001402                BEQ      1$            ;NO - SKIP
4206 034650 104401 001216      TYPE    ,SBELL        ;RING BELL
4207 034654 005237 001112      1$:     INC     $ERTTL    ;COUNT THE NUMBER OF ERRORS
4208 034660 011637 001116      MOV     (SP),$ERRPC    ;GET ADDRESS OF ERROR INSTRUCTION
4209 034664 162737 000002 001116  SUB     #2,$ERRPC
4210 034672 117737 144220 001114  MOVB   @ERRPC,$ITEMB  ;STRIP AND SAVE THE ERROR ITEM CODE
4211 034700 032777 020000 144232  BIT     #BIT13,@SWR    ;SKIP TYPEOUT IF SET
4212 034706 001004                BNE     20$           ;SKIP TYPEOUTS
4213 034710 004737 035022      JSR     PC,ERRTYP     ;GO TO USER ERROR ROUTINE
4214 034714 104401 001223      TYPE    ,SCRLF
4215 034720                20$:
4216 034720 122737 000001 001246  CMPB   #APTENV,$ENV   ;RUNNING IN APT MODE
4217 034726 001007                BNE     2$            ;NO,SKIP APT ERROR REPORT
4218 034730 113737 001114 034742  MOVB   $ITEMB,21$    ;SET ITEM NUMBER AS ERROR NUMBER
4219 034736 004737 037454      JSR     PC,$ATY4     ;REPORT FATAL ERROR TO APT
4220 034742      000                21$:     .BYTE   0
4221 034743      000                .BYTE   0
4222 034744 000777                22$:     BR      22$           ;APT ERROR LOOP
4223 034746 005777 144166      2$:     TST     @SWR        ;HALT ON ERROR
4224 034752 100002                BPL     3$            ;SKIP IF CONTINUE
4225 034754 000000                HALT    ;HALT ON ERROR!
4226 034756 104410                CKSWR   ;TEST FOR CHANGE IN SOFT-SWR
4227 034760 032777 001000 144152  3$:     BIT     #BIT09,@SWR ;LOOP ON ERROR SWITCH SET?
4228 034766 001402                BEQ     4$            ;BR IF NO
4229 034770 013716 001110      MOV     $LPERR,(SP)   ;FUDGE RETURN FOR LOOPING
4230 034774 005737 001214      4$:     TST     $ESCAPE    ;CHECK FOR AN ESCAPE ADDRESS
4231 035000 001402                BEQ     5$            ;BR IF NONE
4232 035002 013716 001214      MOV     $ESCAPE,(SP) ;FUDGE RETURN ADDRESS FOR ESCAPE
4233 035006                5$:
4234 035006 022737 034166 000042  CMP     #SENDAD,@#42  ;ACT-11 AUTO-ACCEPT?
4235 035014 001001                BNE     6$            ;BRANCH IF NO
4236 035016 000000                HALT    ;YES
4237 035020                6$:
4238 035020 000002      RTI      ;RETURN
4239                .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
4240
4241                ;*****
4242                ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
4243                ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERTTB),
4244                ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
4245                ;*
4246                ;*NOTES:
4247                ;*1) THIS ROUTINE PROVIDES AN AUTOMATIC "CARRIAGE RETURN-LINE FEED"
4248                ;*   FOR "EM", "DH", AND "DT".
4249                ;*2) TWO SPACES ARE TYPED AFTER EACH NUMBER FOR "DT"
4250                ;*3) FOR $ITEMB=0, JUST THE ERROR PC IS TYPED
4251                ;*4) THE AVAILABLE FORMATS FOR TYPING DATA ARE:
4252                ;*   DF      FORMAT
4253                ;*   0      TYPE A 6 DIGIT OCTAL NUMBER (FROM 16-BIT BINARY)

```

```

4254          : *      1      TYPE A DECIMAL NUMBER WITHOUT LEADING ZEROS
4255          : *      2      TYPE A 16 DIGIT BINARY NUMBER
4256          : *      3      TYPE A 6 DIGIT OCTAL NUMBER (FROM 18-BIT BINARY)
4257          : *
4258
4259 035022    ERRTYP:
4260 035022    104401 001223      TYPE      , $CRLF      : 'CARRIAGE RETURN' & 'LINE FEED'
4261 035026    010046      MOV      RO, -(SP)      : SAVE RO
4262 035030    005000      CLR      RO              : PICKUP THE ITEM INDEX
4263 035032    153700 001114      BISB     @#$ITEMB, RO
4264 035036    001004      BNE     1$              : IF ITEM NUMBER IS ZERO, JUST
4265                                     : TYPE THE PC OF THE ERROR
4266 035040    013746 001116      MOV      $ERRPC, -(SP)   : SAVE $ERRPC FOR TYPEOUT
4267                                     : ERROR ADDRESS
4268 035044    104402      TYPOC                                     : GO TYPE--OCTAL ASCII(ALL DIGITS)
4269 035046    000471      BR      13$            : GET OUT
4270 035050    005300      1$: DEC      RO              : ADJUST THE INDEX SO THAT IT WILL
4271 035052    006300      ASL     RO              : WORK FOR THE ERROR TABLE
4272 035054    006300      ASL     RO
4273 035056    006300      ASL     RO
4274 035060    062700 001316      ADD     #$ERRTB, RO     : FORM TABLE POINTER
4275 035064    012037 035074      MOV     (RO)+, 2$      : PICKUP 'ERROR MESSAGE' POINTER
4276 035070    001404      BEQ     3$              : SKIP TYPEOUT IF NO POINTER
4277 035072    104401      TYPE                                     : TYPE THE 'ERROR MESSAGE'
4278 035074    000000      2$: .WORD 0           : 'ERROR MESSAGE' POINTER GOES HERE
4279 035076    104401 001223      TYPE      , $CRLF      : 'CARRIAGE RETURN' & 'LINE FEED'
4280 035102    012037 035112      3$: MOV     (RO)+, 4$      : PICKUP 'DATA HEADER' POINTER
4281 035106    001404      BEQ     5$              : SKIP TYPEOUT IF 0
4282 035110    104401      TYPE                                     : TYPE THE 'DATA HEADER'
4283 035112    000000      4$: .WORD 0           : 'DATA HEADER' POINTER GOES HERE
4284 035114    104401 001223      TYPE      , $CRLF      : 'CARRIAGE RETURN' & 'LINE FEED'
4285 035120    010146      5$: MOV     R1, -(SP)    : SAVE R1
4286 035122    012001      MOV     (R0)+, R1      : PICKUP 'DATA TABLE' POINTER
4287 035124    001441      BEQ     12$             : BR IF NO DATA TO BE TYPED
4288 035126    012000      MOV     (R0)+, R0      : PICKUP 'DATA FORMAT' POINTER
4289 035130    105710      6$: TSTB   (R0)          : IS IT FORMAT 0?
4290 035132    001003      BNE     7$              : BR IF NO
4291
4292          : *THIS CODE IS FOR OCTAL (16-BIT) FORMAT (DF=0)
4293 035134    013146      MOV     @ (R1)+, -(SP)  : SAVE @ (R1)+ FOR TYPEOUT
4294 035136    104402      TYPOC                                     : GO TYPE--OCTAL ASCII(ALL DIGITS)
4295 035140    000425      BR      11$
4296
4297          : *THIS CODE IS FOR DECIMAL FORMAT (DF=1)
4298 035142    121027 000001      7$: CMPB   (R0), #1       : IS IT FORMAT 1?
4299 035146    001003      BNE     8$              : BRANCH IF NO
4300 035150    013146      MOV     @ (R1)+, -(SP)  : SAVE @ (R1)+ FOR TYPEOUT
4301 035152    104405      TYPDS                                     : GO TYPE--DECIMAL ASCII WITH SIGN
4302 035154    000417      BR      11$
4303
4304          : *THIS CODE IS FOR BINARY FORMAT (DF=2)
4305 035156    121027 000002      8$: CMPB   (R0), #2       : IS IT FORMAT 2?
4306 035162    001003      BNE     9$              : BRANCH IF NO
4307 035164    013146      MOV     @ (R1)+, -(SP)  : SAVE @ (R1)+ FOR TYPEOUT
4308 035166    104406      TYPBN                                     : GO TYPE--BINARY ASCII
4309 035170    000411      BR      11$

```

```
4310
4311
4312 035172 012146
4313 035174 004737 040526
4314 035200 062716 000005
4315 035204 012637 035212
4316 035210 104401
4317 035212 000000
4318
4319 035214 005711
4320 035216 001404
4321 035220 104401 035242
4322 035224 105720
4323 035226 000740
4324
4325 035230 012601
4326 035232 012600
4327 035234 104401 001223
4328 035240 000207
4329 035242 020040 000
4330 035246
4331
```

;*THIS CODE IS FOR OCTAL (18-BIT) FORMAT (DF=3)

```
9$:  MOV      (R1)+,-(SP)      ;PUT ADDRESS OF FIRST LOC. ON STACK
      JSR      PC,$DB20      ;CONVERT TWO LOCS. TO AN ASCII STRING
      ADD      #5,(SP)        ;ONLY NEED 6 CHARACTERS NOT 11
      MOV      (SP)+,10$      ;PUT ADDRESS OF ASCII CHARS. AT 10$
      TYPE     (SP)+,10$      ;TYPE OCTAL VALUE OF 18-BIT BINARY NO.

10$:  .WORD    0

11$:  TST      (R1)           ;IS THERE ANOTHER NUMBER?
      BEQ      12$           ;BR IF NO
      TYPE     ,14$          ;TYPE TWO(2) SPACES
      TSTB     (R0)+         ;POINT TO NEW 'DATA FORMAT'
      BR       6$           ;LOOP

12$:  MOV      (SP)+,R1       ;RESTORE R1
13$:  MOV      (SP)+,R0       ;RESTORE R0
      TYPE     ,$CRLF        ;'CARRIAGE RETURN' & 'LINE FEED'
      RTS      PC           ;RETURN

14$:  .ASCIZ   / /           ;TWO(2) SPACES
      .EVEN
```

```
4332 .SBTTL ***** SUBROUTINES USED BY THIS PROGRAM *****
4333
4334 .SBTTL TURN OFF T-BIT AND SAVE CURRENT PSW
4335 :*****
4336 :*
4337 :* THIS SUBROUTINE IS USED TO TURN OFF THE TRACE TRAP BIT IN THE PSW
4338 :* IF IT IS ON. THE PROCESSOR STATUS IS SAVED IN 'TBITPS' SO THAT
4339 :* THE PSW CAN BE RESTORED TO ITS PREVIOUS CONDITION WHEN CONDITIONS
4340 :* WARRANT T-BIT TRAPPING.
4341 :*
4342 :*****
4343 035246 033727 177776 000020 TOFF: BIT PSW,#TBIT ;IS THE T-BIT SET IN THE PSW?
4344 035254 001411 BEQ 1$ ;EXIT IF NO
4345 035256 013746 177776 MOV PSW,-(SP) ;PUSH PRESENT PSW ON THE STACK
4346 035262 011637 001276 MOV (SP),TBITPS ;ALSO SAVE IT IN 'TBITPS' FOR
4347 ;RESTORING LATER
4348 035266 042716 000020 BIC #TBIT,(SP) ;CLEAR THE T-BIT (BIT 4) IN THE PSW
4349 035272 012746 035300 MOV #1$,-(SP) ;PUSH PC OF 'RTS' ON STACK
4350 035276 000006 RTT ;'RETURN' TO 1$ WITH T-BIT OFF
4351 035300 000207 1$: RTS PC ;RETURN TO PROGRAM
4352
4353 .SBTTL TURN ON T-BIT AND RESTORE PREVIOUS PSW
4354 :*****
4355 :*
4356 :* THIS SUBROUTINE IS USED TO RESTORE THE PROCESSOR STATUS TO ITS
4357 :* PREVIOUS CONDITION BY RESTORING THE 'T-BIT PSW' SAVED BY THE
4358 :* 'TOFF' SUBROUTINE IN THE 'TBITPS' LOCATION.
4359 :*
4360 :*****
4361 035302 033727 001276 000020 TON: BIT TBITPS,#TBIT ;WAS T-BIT ON IN THE PREVIOUS PSW?
4362 035310 001410 BEQ 1$ ;EXIT IF NO
4363 035312 013746 001276 MOV TBITPS,-(SP) ;PUSH PREVIOUS PSW ON THE STACK
4364 035316 012737 000340 001276 MOV #340,TBITPS ;RESET THE 'TBITPS' LOCATION
4365 035324 012746 035332 MOV #1$,-(SP) ;PUSH PC OF 'RTS' ON STACK
4366 035330 000006 RTT ;'RETURN' TO 1$ WITH T-BIT RESTORED
4367 035332 000207 1$: RTS PC ;RETURN TO PROGRAM
4368
4369 .SBTTL SET ALL WRITEABLE BITS IN ALL PAR/PDR'S
4370 :*****
4371 :*
4372 :* THIS SUBROUTINE IS USED BY THE PAR/PDR DUAL ADDRESSING TEST
4373 :* TO SET ALL WRITEABLE BITS IN ALL KERNEL AND USE PAR'S AND
4374 :* PDR'S TO A 1. THE 'INITIAL STATE' OF HAVING ALL BITS=1 IS
4375 :* USED TO SEE THAT ONLY ONE REGISTER IS CLEARED IN RESPONSE TO
4376 :* A SINGLE PAR OR PDR ADDRESS.
4377 :*
4378 :*****
4379
4380 035334 012702 000010 SETREG: MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8
4381 035340 012701 172300 MOV #KIPDRO,R1 ;LOAD ADDRESS OF FIRST PDR INTO R1
4382 035344 012721 177777 1$: MOV #-1,(R1)+ ;SET BITS IN KERNEL PDR TO 1
4383 035350 077203 SOB R2,1$ ;LOOP TO 1$ UNTIL ALL KERNEL PDR'S LOADED
4384 035352 012702 000010 MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8
4385 035356 012701 172340 MOV #KIPARO,R1 ;LOAD ADDRESS OF FIRST PAR INTO R1
4386 035362 012721 177777 2$: MOV #-1,(R1)+ ;SET BITS IN A KERNEL PAR TO 1
4387 035366 077203 SOB R2,2$ ;LOOP TO 2$ UNTIL ALL KERNEL PAR'S LOADED
```

4388 035370 012702 000010
 4389 035374 012701 177600
 4390 035400 012721 177777
 4391 035404 077203
 4392 035406 012702 000010
 4393 035412 012701 177640
 4394 035416 012721 177777
 4395 035422 077203
 4396 035424 000207
 4397
 4398
 4399
 4400
 4401
 4402
 4403
 4404
 4405
 4406
 4407

```

MOV #10,R2          ;LOAD LOOP COUNTER WITH AN 8
MOV #UIPDR0,R1     ;LOAD ADDRESS OF FIRST PDR INTO R1
3$: MOV #-1,(R1)+   ;SET BITS IN A USER PDR TO 1
SOB R2,3$          ;LOOP TO 3$ UNTIL ALL USER PDR'S LOADED
MOV #10,R2          ;LOAD LOOP COUNTER WITH AN 8
MOV #UIPAR0,R1     ;LOAD ADDRESS OF FIRST PAR INTO R1
4$: MOV #-1,(R1)+   ;SET BITS IN A USER PAR TO 1
SOB R2,4$          ;LOOP TO 4$ UNTIL ALL USER PAR'S LOADED
RTS PC              ;RETURN TO TEST
  
```

.SBTTL READ & COMPARE KERNEL & USER PAR/PDR'S

THIS SUBROUTINE IS USED BY PAR/PDR DUAL ADDRESSING TEST TO
 READ ALL THE PAR'S AND PDR'S TO SEE THAT ONLY ONE REGISTER
 WAS CLEARED IN RESPONSE TO A SINGLE PAR OR PDR ADDRESS.
 ANY FAILURES FOUND BY THE PAR/PDR DUAL ADDRESSING TEST WILL
 BE REPORTED BY THIS SUBROUTINE.

4408 035426
 4409 035426 012701 172300
 4410 035432 012704 000010
 4411 035436 012705 077416
 4412 035442 021105
 4413 035444 001404
 4414 035446 020100
 4415 035450 001402
 4416 035452 011102
 4417 035454 104016
 4418
 4419
 4420
 4421 035456 062701 000002
 4422 035462 077411
 4423 035464 012701 172340
 4424 035470 012704 000010
 4425 035474 012705 177777
 4426 035500 021105
 4427 035502 001404
 4428 035504 020100
 4429 035506 001402
 4430 035510 011102
 4431 035512 104016
 4432
 4433
 4434
 4435 035514 062701 000002
 4436 035520 077411
 4437 035522 012701 177600
 4438 035526 012704 000010
 4439 035532 012705 077416
 4440 035536 021105
 4441 035540 001404
 4442 035542 020100
 4443 035544 001402

```

CMPREG:
MOV #KIPDR0,R1     ;LOAD ADDRESS OF FIRST KERNEL PDR IN R1
MOV #10,R4          ;LOAD LOOP COUNTER WITH AN 8
MOV #77416,R5      ;PUT EXPECTED PDR CONTENTS IN R5
1$: CMP (R1),R5    ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
BEQ 2$             ;BRANCH IF YES
CMP R1,R0           ;WAS IT THE REG. THAT WAS CLEARED?
BEQ 2$             ;BRANCH IF YES
MOV (R1),R2        ;SAVE CONTENTS OF IMPROPERLY CLEARED REGISTER
ERROR 16           ;A PDR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
                    ;FOR TIGHTER SCOPE LOOP
                    ;REPLACE ERROR CALL WITH
                    ;AN 'RTS PC' = 000207
2$: ADD #2,R1       ;FORM NEXT ADDRESS
SOB R4,1$          ;LOOP TO 1$ UNTIL ALL KERNEL PDR'S CHECKED
MOV #KIPAR0,R1     ;LOAD ADDRESS OF FIRST KERNEL PAR IN R1
MOV #10,R4          ;LOAD LOOP COUNTER WITH AN 8
MOV #177777,R5     ;PUT EXPECTED PAR CONTENTS IN R5
3$: CMP (R1),R5    ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
BEQ 4$             ;BRANCH IF YES
CMP R1,R0           ;WAS IT THE REG. THAT WAS CLEARED?
BEQ 4$             ;BRANCH IF YES
MOV (R1),R2        ;SAVE CONTENTS OF IMPROPERLY CLEARED REGISTER
ERROR 16           ;A PAR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
                    ;FOR TIGHTER SCOPE LOOP
                    ;REPLACE ERROR CALL WITH
                    ;AN 'RTS PC' = 000207
4$: ADD #2,R1       ;FORM NEXT ADDRESS
SOB R4,3$          ;LOOP TO 3$ UNTIL ALL KERNEL PAR'S CHECKED
MOV #UIPDR0,R1     ;LOAD ADDRESS OF FIRST USER PDR IN R1
MOV #10,R4          ;LOAD LOOP COUNTER WITH AN 8
MOV #77416,R5      ;PUT EXPECTED PDR CONTENTS IN R5
5$: CMP (R1),R5    ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
BEQ 6$             ;BRANCH IF YES
CMP R1,R0           ;WAS IT THE REG. THAT WAS CLEARED?
BEQ 6$             ;BRANCH IF YES
  
```

```
4444 035546 011102          MOV      (R1),R2          ;SAVE CONTENTS OF IMPROPERLY CLEARED REGISTER
4445 035550 104016          ERROR    16              ;A PDR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
4446                                     ;FOR TIGHTER SCOPE LOOP
4447                                     ;REPLACE ERROR CALL WITH
4448                                     ;AN 'RTS PC' = 000207
4449 035552 062701 000002    6$:      ADD      #2,R1          ;FORM NEXT ADDRESS
4450 035556 077411          SOB      R4,5$           ;LOOP TO 5$ UNTIL ALL USER PDR'S CHECKED
4451 035560 012701 177640    MOV      #UIPAF0,R1      ;LOAD ADDRESS OF FIRST USER PAR IN R1
4452 035564 012704 000010    MOV      #10,R4          ;LOAD LOOP COUNTER WITH AN 8
4453 035570 012705 177777    MOV      #177777,R5      ;PUT EXPECTED PAR CONTENTS IN R5
4454 035574 021105          7$:      CMP      (R1),R5        ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
4455 035576 001404          BEQ      8$              ;BRANCH IF YES
4456 035600 020100          CMP      R1,R0           ;WAS IT THE REG. THAT WAS CLEARED?
4457 035602 001402          BEQ      8$              ;BRANCH IF YES
4458 035604 011102          MOV      (R1),R2          ;SAVE CONTENTS OF IMPROPERLY CLEARED REGISTER
4459 035606 104016          ERROR    16              ;A PAR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
4460                                     ;FOR TIGHTER SCOPE LOOP
4461                                     ;REPLACE ERROR CALL WITH
4462                                     ;AN 'RTS PC' = 000207
4463 035610 062701 000002    8$:      ADD      #2,R1          ;FORM NEXT ADDRESS
4464 035614 077411          SOB      R4,7$           ;LOOP TO 7$ UNTIL ALL USER PAR'S CHECKED
4465 035616 000207          RTS      PC              ;RETURN TO TEST
```

```
4466                                     .SBTTL  CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS
4467                                     ;*****
4468                                     ;
4469                                     ;
4470                                     ; THIS SUBROUTINE IS USED TO FORM AN 18-BIT OR 22-BIT PHYSICAL ADDRESS
4471                                     ; (PBA) FROM THE 16-BIT VIRTUAL ADDRESS (VBA) AND THE APPROPRIATE
4472                                     ; PAGE ADDRESS REGISTER (PAR). THE SAME METHOD USED BY THE MEMORY
4473                                     ; MANAGEMENT LOGIC IS USED. VBA <15:13> SELECTS WHICH PAR/PDR
4474                                     ; IS TO BE USED, VBA <5:0>+PBA <5:0>, AND VBA <12:6> IS ADDED
4475                                     ; TO PAR <15:00> TO GIVE PBA <21:6>. BITS <21:16> OF THE
4476                                     ; PHYSICAL ADDRESS ARE LEFT IN LOC. 'PBAHI' AND BITS <15:00>
4477                                     ; ARE LEFT IN LOC. 'PBALO'. THE PSW'S 'CURRENT MODE' BITS
4478                                     ; ARE USED TO SELECT THE KERNEL OR USER PAR/PDR'S. THE ROUTINE
```



```

4479      :*      IS ENTERED WITH LOC. 'VIRT1' CONTAINING THE 16-BIT VIRTUAL
4480      :*      ADDRESS.  IF 18-BIT ADDRESSING IS SELECTED IN SR3, THEN
4481      :*      PBA<21:18> ARE CLEARED IN 'PBAHI'.
4482      :*
4483      :*****
4484
4485      035620      FORMPA:
4486      035620      010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
4487      035622      010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
4488      035624      012702      172340      MOV      #KIPAR0,R2      ;LOAD ADDRESS OF FIRST KERNEL PAR IN R2
4489      035630      032737      140000      177776      BIT      #140000,PSW      ;IN USER MODE?
4490      035636      001402      BEQ      1$      ;BRANCH IF NO
4491      035640      012702      177640      MOV      #UIPAR0,R2      ;LOAD ADDRESS OF FIRST USER PAR IN R2
4492      035644      013700      001306      1$:      MOV      VIRT1,R0      ;LOAD VIRTUAL ADDR. (VBA) INTO R0
4493      035650      072027      177764      ASH      #-14,R0      ;GET BITS <15:13> DOWN TO BITS <3:1>
4494      035654      042700      177761      BIC      #177761,R0      ;MASK OF ALL BITS BUT BITS <3:1>
4495      035660      060002      ADD      R0,R2      ;ADD OFFSET TO BASE PAR ADDRESS
4496      035662      011200      MOV      (R2),R0      ;GET BITS <15:00> FROM APPROPRIATE PAR
4497      035664      010002      MOV      R0,R2      ;COPY PAR BITS <15:00> INTO R2
4498      035666      013737      001306      001312      MOV      VIRT1,PBALO      ;PUT VIRTUAL ADDR. IN LOC. 'PBALO'
4499      035674      042737      160000      001312      BIC      #160000,PBALO      ;CLEAR OFF BITS <15:13> OF ORIGINAL VBA
4500      035702      072227      177766      ASH      #-12,R2      ;GET PAR <15:10> DOWN TO BITS <5:0> OF R2
4501      035706      042702      177700      BIC      #177700,R2      ;CLEAR OFF ALL BITS BUT BITS <5:0>
4502      035712      072027      000006      ASH      #6,R0      ;SHIFT PAR<9:0> TO <15:6> OF R0
4503      035716      042700      000077      BIC      #77,R0      ;CLEAR BITS <5:0> OF R0
4504      035722      060037      001312      ADD      R0,PBALO      ;IN EFFECT, ADD VBA<12:0> TO PAR<9:0>
4505      ADC      R2      ;(PAR<9:0> IN BITS <15:6> OF R0)
4506      035726      005502      ADD      ANY CARRY TO R2
4507      035730      010237      001314      MOV      R2,PBAHI      ;PUT BITS <21:16> OF PHYSICAL ADDR. IN PBAHI
4508      035734      032737      000020      172516      BIT      #BIT4,SR3      ;SEE IF 18 OR 22-BIT ADDRESSING
4509      035742      001003      BNE      2$      ;BRANCH IF 22-BIT ADDRESSING
4510      035744      042737      000074      001314      BIC      #74,PBAHI      ;CLEAR BITS <5:2>
4511      035752      2$:
4512      035752      012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
4513      035754      012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
4514      035756      000207      RTS      PC      ;RETURN TO PROGRAM
4515
4516

```

```
4517 .SBTTL TTY INPUT ROUTINE
4518
4519 ::*****
4520 .ENABL LSB
4521
4522 ::*****
4523 ::*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
4524 ::*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
4525 ::*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
4526 ::*WHEN OPERATING IN TTY FLAG MODE.
4527 035760 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ::IS THE SOFT-SWR SELECTED?
4528 035766 001114 BNE 15$ ::BRANCH IF NO
4529 035770 105777 143150 TSTB @STKS ::CHAR THERE?
4530 035774 100111 BPL 15$ ::IF NO, DON'T WAIT AROUND
4531 035776 117746 143144 MOVB @STKB,-(SP) ::SAVE THE CHAR
4532 036002 042716 177600 BIC #^C177,(SP) ::STRIP-OFF THE ASCII
4533 036006 022726 000007 CMP #7,(SP)+ ::IS IT A CONTROL G?
4534 036012 001102 BNE 15$ ::NO, RETURN TO USER
4535 036014 123727 001134 000001 CMPB $AUTOB,#1 ::ARE WE RUNNING IN AUTO-MODE?
4536 036022 001476 BEQ 15$ ::BRANCH IF YES
4537
4538 036024 104401 036722 $GTSWR: TYPE , $CNTLG ::ECHO THE CONTROL-G (^G)
4539 036030 104401 036727 TYPE , $MSWR ::TYPE CURRENT CONTENTS
4540 036034 013746 000176 MOV SWREG,-(SP) ::SAVE SWREG FOR TYPEOUT
4541 036040 104402 TYPOC ::GO TYPE--OCTAL ASCII(ALL DIGITS)
4542 036042 104401 036740 TYPE , $MNEW ::PROMPT FOR NEW SWR
4543 036046 005046 19$: CLR -(SP) ::CLEAR COUNTER
4544 036050 005046 CLR -(SP) ::THE NEW SWR
4545 036052 105777 143066 7$: TSTB @STKS ::CHAR THERE?
4546 036056 100375 BPL 7$ ::IF NOT TRY AGAIN
4547
4548 036060 117746 143062 MOVB @STKB,-(SP) ::PICK UP CHAR
4549 036064 042716 177600 BIC #^C177,(SP) ::MAKE IT 7-BIT ASCII
4550
4551 036070 021627 000003 CMP (SP),#3 ::IS IT A CONTROL-C?
4552 036074 001015 BNE 9$ ::BRANCH IF NOT
4553 036076 104401 036710 TYPE , $CNTLC ::YES, ECHO CONTROL-C (^C)
4554 036102 062706 000006 ADD #6,SP ::CLEAN UP STACK
4555 036106 123727 001135 000001 CMPB $INTAG,#1 ::REENABLE TTY KEYBOARD INTERRUPTS?
4556 036114 001003 BNE 8$ ::BRANCH IF NO
4557 036116 012777 000100 143020 MOV #100,@STKS ::ALLOW TTY KEYBOARD INTERRUPTS
4558 036124 000137 036752 8$: JMP CNTRLC ::CONTROL-C RESTART
4559
4560
4561 036130 021627 000025 9$: CMP (SP),#25 ::IS IT A CONTROL-U?
4562 036134 001005 BNE 10$ ::BRANCH IF NOT
4563 036136 104401 036715 TYPE , $CNTLU ::YES, ECHO CONTROL-U (^U)
4564 036142 062706 000006 20$: ADD #6,SP ::IGNORE PREVIOUS INPUT
4565 036146 000757 BR 19$ ::LET'S TRY IT AGAIN
4566
4567
4568 036150 021627 000015 10$: CMP (SP),#15 ::IS IT A <CR>?
4569 036154 001022 BNE 16$ ::BRANCH IF NO
4570 036156 005766 000004 TST 4(SP) ::YES, IS IT THE FIRST CHAR?
4571 036162 001403 BEQ 11$ ::BRANCH IF YES
4572 036164 016677 000002 142746 MOV 2(SP),@SWR ::SAVE NEW SWR
```

```
4573 036172 062706 000006 11$: ADD #6,SP ;:CLEAR UP STACK
4574 036176 104401 001223 14$: TYPE $CRLF ;:ECHO <CR> AND <LF>
4575 036202 123727 001135 000001 CMPB $INTAG,#1 ;:RE-ENABLE TTY KBD INTERRUPTS?
4576 036210 001003 BNE 15$ ;:BRANCH IF NOT
4577 036212 012777 000100 142724 MOV #100,@$TKS ;:RE-ENABLE TTY KBD INTERRUPTS
4578 036220 000002 15$: RTI ;:RETURN
4579 036222 004737 037314 16$: JSR PC,$TYPEC ;:ECHO CHAR
4580 036226 021627 000060 CMP (SP),#60 ;:CHAR < 0?
4581 036232 002420 BLT 18$ ;:BRANCH IF YES
4582 036234 021627 000067 CMP (SP),#67 ;:CHAR > 7?
4583 036240 003015 BGT 18$ ;:BRANCH IF YES
4584 036242 042726 000060 BIC #60,(SP)+ ;:STRIP-OFF ASCII
4585 036246 005766 000002 TST 2(SP) ;:IS THIS THE FIRST CHAR
4586 036252 001403 BEQ 17$ ;:BRANCH IF YES
4587 036254 006316 ASL (SP) ;:NO, SHIFT PRESENT
4588 036256 006316 ASL (SP) ;: CHAR OVER TO MAKE
4589 036260 006316 ASL (SP) ;: ROOM FOR NEW ONE.
4590 036262 005266 000002 17$: INC 2(SP) ;:KEEP COUNT OF CHAR
4591 036266 056616 177776 BIS --2(SP),(SP) ;:SET IN NEW CHAR
4592 036272 000667 BR 7$ ;:GET THE NEXT ONE
4593 036274 104401 001222 18$: TYPE $QUES ;:TYPE ?<CR><LF>
4594 036300 000720 BR 20$ ;:SIMULATE CONTROL-U
4595 .DSABL LSB
```

```
4596
4597
4598 ;:*****
4599 ;:*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
4600 ;:*CALL:
4601 ;:* RDCHR ;:INPUT A SINGLE CHARACTER FROM THE TTY
4602 ;:* RETURN HERE ;:CHARACTER IS ON THE STACK
4603 ;:* ;:WITH PARITY BIT STRIPPED OFF
4604 ;:
4605
```

```
4606 036302 011646 $RDCHR: MOV (SP),-(SP) ;:PUSH DOWN THE PC
4607 036304 016666 000004 000002 MOV 4(SP),2(SP) ;:SAVE THE PS
4608 036312 105777 142626 1$: TSTB @$TKS ;:WAIT FOR
4609 036316 100375 BPL 1$ ;:A CHARACTER
4610 036320 117766 142622 000004 MOVB @$TKB,4(SP) ;:READ THE TTY
4611 036326 042766 177600 000004 BIC #^C<177>,4(SP) ;:GET RID OF JUNK IF ANY
4612 036334 026627 000004 000023 CMP 4(SP),#23 ;:IS IT A CONTROL-S?
4613 036342 001013 BNE 3$ ;:BRANCH IF NO
4614 036344 105777 142574 2$: TSTB @$TKS ;:WAIT FOR A CHARACTER
4615 036350 100375 BPL 2$ ;:LOOP UNTIL ITS THERE
4616 036352 117746 142570 MOVB @$TKB,-(SP) ;:GET CHARACTER
4617 036356 042716 177600 BIC #^C177,(SP) ;:MAKE IT 7-BIT ASCII
4618 036362 022627 000021 CMP (SP)+,#21 ;:IS IT A CONTROL-Q?
4619 036366 001366 BNE 2$ ;:IF NOT DISCARD IT
4620 036370 000750 BR 1$ ;:YES, RESUME
4621 036372 026627 000004 000140 3$: CMP 4(SP),#140 ;:IS IT UPPER CASE?
4622 036400 002407 BLT 4$ ;:BRANCH IF YES
4623 036402 026627 000004 000175 CMP 4(SP),#175 ;:IS IT A SPECIAL CHAR?
4624 036410 003003 BGT 4$ ;:BRANCH IF YES
4625 036412 042766 000040 000004 BIC #40,4(SP) ;:MAKE IT UPPER CASE
4626 036420 000002 4$: RTI ;:GO BACK TO USER
```

```
4627 ;:*****
4628 ;:*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
```

```

4629          ;*CALL:
4630          ;*      RDLIN          ;;INPUT A STRING FROM THE TTY
4631          ;*      RETURN HERE    ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
4632          ;*                      ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
4633
4634 036422 010346 $RDLIN: MOV      R3,-(SP)      ;;SAVE R3
4635 036424 005046 CLR      -(SP)          ;;CLEAR THE RUBOUT KEY
4636 036426 012703 036700 1$:  MOV      #$TTYIN,R3      ;;GET ADDRESS
4637 036432 022703 036710 2$:  CMP      #$TTYIN+8.,R3    ;;BUFFER FULL?
4638 036436 101467 BLOS     4$              ;;BR IF YES
4639 036440 104411 RDCHR   RDCHR          ;;GO READ ONE CHARACTER FROM THE TTY
4640 036442 112613 MOVVB   (SP)+,(R3)      ;;GET CHARACTER
4641 036444 122713 000003 CMPB    #3,(R3)        ;;IS IT A CONTROL-C?
4642 036450 001006 BNE     10$            ;;BRANCH IF NO
4643 036452 104401 036710 TYPE    ,$CNTLC        ;;TYPE A CONTROL-C (^C)
4644 036456 005726 TST     (SP)+          ;;CLEAN RUBOUT KEY OFF OF THE STACK
4645 036460 012603 MOV      (SP)+,R3      ;;RESTORE R3
4646 036462 000137 036752 JMP     CNTRLC         ;;GOTO CONTROL-C RESTART
4647 036466 122713 000177 10$:  CMPB    #177,(R3)     ;;IS IT A RUBOUT
4648 036472 001022 BNE     5$            ;;BR IF NO
4649 036474 005716 TST     (SP)          ;;IS THIS THE FIRST RUBOUT?
4650 036476 001007 BNE     6$            ;;BR IF NO
4651 036500 112737 000134 036676 MOVVB   #' \,9$        ;;TYPE A BACK SLASH
4652 036506 104401 036676 TYPE    ,9$
4653 036512 012716 177777 MOV      #-1,(SP)     ;;SET THE RUBOUT KEY
4654 036516 005303 6$:  DEC     R3          ;;BACKUP BY ONE
4655 036520 020327 036700 CMP      R3,$$TTYIN  ;;STACK EMPTY?
4656 036524 103434 BLO     4$            ;;BR IF YES
4657 036526 111337 036676 MOVVB   (R3),9$        ;;SETUP TO TYPEOUT THE DELETED CHAR.
4658 036532 104401 036676 TYPE    ,9$
4659 036536 000735 BR      2$            ;;GO TYPE
4660 036540 005716 5$:  TST     (SP)          ;;GO READ ANOTHER CHAR.
4661 036542 001406 BEQ     7$            ;;RUBOUT KEY SET?
4662 036544 112737 000134 036676 MOVVB   #' \,9$        ;;BR IF NO
4663 036552 104401 036676 TYPE    ,9$
4664 036556 005016 CLR      (SP)          ;;TYPE A BACK SLASH
4665 036560 122713 000025 7$:  CMPB    #25,(R3)     ;;CLEAR THE RUBOUT KEY
4666 036564 001003 BNE     8$            ;;IS CHARACTER A CTRL U?
4667 036566 104401 036715 TYPE    ,$CNTLU        ;;BR IF NO
4668 036572 000715 BR      1$            ;;TYPE A CONTROL 'U'
4669 036574 122713 000022 8$:  CMPB    #22,(R3)     ;;GO START OVER
4670 036600 001011 BNE     3$            ;;IS CHARACTER A '^R'?
4671 036602 105013 CLRB   (R3)          ;;BRANCH IF NO
4672 036604 104401 001223 TYPE    ,$CRLF        ;;CLEAR THE CHARACTER
4673 036610 104401 036700 TYPE    ,$TTYIN       ;;TYPE A 'CR' & 'LF'
4674 036614 000706 BR      2$            ;;TYPE THE INPUT STRING
4675 036616 104401 001222 4$:  TYPE    ,$QUES        ;;GO PICKUP ANOTHER CHACTER
4676 036622 000701 BR      1$            ;;TYPE A '?'
4677 036624 111337 036676 3$:  MOVVB   (R3),9$        ;;CLEAR THE BUFFER AND LOOP
4678 036630 104401 036676 TYPE    ,9$
4679 036634 122723 000015 CMPB    #15,(R3)+    ;;ECHO THE CHARACTER
4680 036640 001274 BNE     2$            ;;CHECK FOR RETURN
4681 036642 105063 177777 CLRB   -1(R3)        ;;LOOP IF NOT RETURN
4682 036646 104401 001224 TYPE    ,$LF          ;;CLEAR RETURN (THE 15)
4683 036652 005726 TST     (SP)+          ;;TYPE A LINE FEED
4684 036654 012603 MOV      (SP)+,R3    ;;CLEAN RUBOUT KEY FROM THE STACK
                      ;;RESTORE R3

```

```
4685 036656 011646          MOV      (SP),-(SP)      ::ADJUST THE STACK AND PUT ADDRESS OF THE
4686 036660 016666 000004 000002  MOV      4(SP),2(SP)      ::      FIRST ASCII CHARACTER ON IT
4687 036666 012766 036700 000004  MOV      #$TTYIN,4(SP)
4688 036674 000002          RTI                      ::RETURN
4689 036676      000          9$: .BYTE 0              ::STORAGE FOR ASCII CHAR. TO TYPE
4690 036677      000          .BYTE 0              ::TERMINATOR
4691 036700 000010          $TTYIN: .BLKB 8.         ::RESERVE 8 BYTES FOR TTY INPUT
4692 036710 041536 005015      000  $CNTLC: .ASCIZ /^C/<15><12>  ::CONTROL 'C'
4693 036715      136 006525 000012  $CNTLU: .ASCIZ /^U/<15><12>  ::CONTROL 'U'
4694 036722 043536 005015      000  $CNTLG: .ASCIZ /^G/<15><12>  ::CONTROL 'G'
4695 036727      015 051412 051127  $MSWR: .ASCIZ <15><12>/SWR = /
4696 036734 036440 000040          $MNEW: .ASCIZ / NEW = /
4697 036740 020040 042516 020127
4698 036746 020075      000
4699          036752          .EVEN
4700
4701          .SBTTL CONTROL-C SERVICING ROUTINE
4702
4703          :* THE FOLLOWING CODE IS EXECUTED WHEN A CONTROL-C HAS
4704          :* BEEN TYPED INSTEAD OF A NEW SWITCH REG. VALUE.
```

```
4705      :*      (IN OTHER WORDS, AFTER A CONTROL-G WAS TYPED).
4706      :*      A NEW SWITCH REG. VALUE WILL BE ASKED FOR,
4707      :*      THE TEST NUMBER AND PASS NUMBER WILL BE TYPED,
4708      :*      AND THEN THE PROGRAM WILL GO TO 'END-OF-PASS' AND CONTINUE
4709
4710 036752 013737 001234 001210 CNTRLC: MOV      $PASS,$TMP5      ;GET THE VALUE OF '$PASS'
4711 036760 005237 001210      INC      $TMP5      ;FORM CURRENT PASS NO.
4712 036764 104401 037031      TYPE     ,CM5G      ;TYPE THE TEST STOPS MESSAGE
4713 036770 113737 001102 037024 MOVVB   $TSTNM,1$    ;SAVE THE TEST NUMBER
4714 036776 013746 037024      MOV      1$,-(SP)   ;SAVE 1$ FOR TYPEOUT
4715 037002 104402      TYPOC      ;GO TYPE--OCTAL ASCII(ALL DIGITS)
4716 037004 104401 037026      TYPE     ,2$      ;TYPE 2 SPACES
4717 037010 013746 001210      MOV      $TMP5,-(SP) ;SAVE $TMP5 FOR TYPEOUT
4718 037014 104405      TYPDS      ;GO TYPE--DECIMAL ASCII WITH SIGN
4719 037016 104407      GTSWR      ;ASK FOR NEW SWR VALUE
4720 037020 000137 033764      JMP      $EOP+2     ;CONTINUE AT $EOP+2
4721 037024 000000      1$:      .WORD      0      ;BUFFER FOR TEST NUMBER
4722 037026 020040      2$:      .ASCIZ    / /      ;TWO SPACES AND THE STOP MESSAGE
4723 037031      112 046525 044520  CMSG:    .ASCII   /JUMPING TO END-OF-PASS/<15><12>
4724 037036 043516 052040 020117
4725 037044 047105 026504 043117
4726 037052 050055 051501 006523
4727 037060      012
4728 037061      124 051505 047124      .ASCIZ  /TESTNO PASSNO/<15><12>
4729 037066 020117 050040 051501
4730 037074 047123 006517 000012
4731
4732      .EVEN
4733      .SBTTL  TYPE ROUTINE
4734      ;*****
4735      ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
4736      ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
4737      ;*NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
4738      ;*NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
4739      ;*NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
4740      ;*
4741      ;*CALL:
4742      ;*1) USING A TRAP INSTRUCTION
4743      ;*      TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
4744      ;*OR
4745      ;*      TYPE
4746      ;*      MESADR
4747      ;*
4748
4749 037102 105737 001157 $TYPE:  TSTB   $TPFLG      ;;IS THERE A TERMINAL?
4750 037106 100002      BPL      1$      ;;BR IF YES
4751 037110 000000      HALT      ;;HALT HERE IF NO TERMINAL
4752 037112 000430      BR      3$      ;;LEAVE
4753 037114 010046      1$:      MOV      RO,-(SP)   ;;SAVE RO
4754 037116 017600 000002      MOV      @2(SP),RO   ;;GET ADDRESS OF ASCIZ STRING
4755 037122 122737 000001 001246      CMPB   #APTENV,$ENV  ;;RUNNING IN APT MODE
4756 037130 001011      BNE     62$      ;;NO,GO CHECK FOR APT CONSOLE
4757 037132 132737 000100 001247      BITB   #APTPOOL,$ENVM ;SPOOL MESSAGE TO APT
4758 037140 001405      BEQ     62$      ;;NO,GO CHECK FOR CONSOLE
4759 037142 010037 037152      MOV     RO,61$     ;;SETUP MESSAGE ADDRESS FOR APT
4760 037146 004737 037444      JSR     PC,$ATY3   ;;SPOOL MESSAGE TO APT
```

```

4761 037152 000000 61$: .WORD 0 ::MESSAGE ADDRESS
4762 037154 132737 000040 001247 62$: BITB #APTCSUP,$ENVM ::APT CONSOLE SUPPRESSED
4763 037162 001003 BNE 60$ ::YES,SKIP TYPE OUT
4764 037164 112046 2$: MOVB (R0)+,-(SP) ::PUSH CHARACTER TO BE TYPED ONTO STACK
4765 037166 001005 BNE 4$ ::BR IF IT ISN'T THE TERMINATOR
4766 037170 005726 TST (SP)+ ::IF TERMINATOR POP IT OFF THE STACK
4767 037172 012600 60$: MOV (SP)+,R0 ::RESTORE R0
4768 037174 062716 000002 3$: ADD #2,(SP) ::ADJUST RETURN PC
4769 037200 000002 RTI ::RETURN
4770 037202 122716 000011 4$: CMPB #HT,(SP) ::BRANCH IF <HT>
4771 037206 001430 BEQ 8$
4772 037210 122716 000200 CMPB #CRLF,(SP) ::BRANCH IF NOT <CRLF>
4773 037214 001006 BNE 5$
4774 037216 005726 TST (SP)+ ::POP <CR><LF> EQUIV
4775 037220 104401 TYPE ::TYPE A CR AND LF
4776 037222 001223 $CRLF
4777 037224 105037 037432 CLRB $CHARCNT ::CLEAR CHARACTER COUNT
4778 037230 000755 BR 2$ ::GET NEXT CHARACTER
4779 037232 004737 037314 5$: JSR PC,$TYPEC ::GO TYPE THIS CHARACTER
4780 037236 123726 001156 6$: CMPB $FILLC,(SP)+ ::IS IT TIME FOR FILLER CHARS.?
4781 037242 001350 BNE 2$ ::IF NO GO GET NEXT CHAR.
4782 037244 013746 001154 MOV $NULL,-(SP) ::GET # OF FILLER CHARS. NEEDED
4783 ::AND THE NULL CHAR.
4784 037250 105366 000001 7$: DECB 1(SP) ::DOES A NULL NEED TO BE TYPED?
4785 037254 002770 BLT 6$ ::BR IF NO--GO POP THE NULL OFF OF STACK
4786 037256 004737 037314 JSR PC,$TYPEC ::GO TYPE A NULL
4787 037262 105357 037432 DECB $CHARCNT ::DO NOT COUNT AS A COUNT
4788 037266 000770 BR 7$ ::LOOP
4789
4790 ;HORIZONTAL TAB PROCESSOR
4791
4792 037270 112716 000040 8$: MOVB #' ,(SP) ::REPLACE TAB WITH SPACE
4793 037274 004737 037314 9$: JSR PC,$TYPEC ::TYPE A SPACE
4794 037300 132737 000007 037432 BITB #7,$CHARCNT ::BRANCH IF NOT AT
4795 037306 001372 BNE 9$ ::TAB STOP
4796 037310 005726 TST (SP)+ ::POP SPACE OFF STACK
4797 037312 000724 BR 2$ ::GET NEXT CHARACTER
4798 037314 $TYPEC:
4799 037314 105777 141624 TSTB @$TKS ::CHAR IN KYBD BUFFER? :MJD001
4800 037320 100022 BPL 10$ ::BR IF NOT :MJD001
4801 037322 017746 141620 MOV @$TKB,-(SP) ::GET CHAR :MJD001
4802 037326 042716 177600 BIC #177600,(SP) ::STRIP EXTRANEIOUS BITS :MJD001
4803 037332 122716 000023 CMPB #$XOFF,(SP) ::WAS CHAR XOFF :MJD001
4804 037336 001012 BNE 102$ ::BR IF NOT :MJD001
4805 037340 101$:
4806 037340 105777 141600 TSTB @$TKS ::WAIT FOR CHAR :MJD001
4807 037344 100375 BPL 101$ :MJD001
4808 037346 117716 141574 MOVB @$TKB,(SP) ::GET CHAR :MJD001
4809 037352 042716 177600 BIC #177600,(SP) ::STRIP IT :MJD001
4810 037356 122716 000021 CMPB #$XON,(SP) ::WAS IT XON? :MJD001
4811 037362 001366 BNE 101$ ::BR IF NOT :MJD001
4812 037364 102$:
4813 037364 005726 TST (SP)+ ::FIX STACK :MJD001
4814 037366 10$:
4815 037366 105777 141556 TSTB @$TPS ::WAIT UNTIL PRINTER IS READY :MJD001
4816 037372 100375 BPL 10$ :MJD001

```

```

4817 037374 116677 000002 141550      MOVB    2(SP),@STPB      ;;LOAD CHAR TO BE TYPED INTO DATA REG.
4818 037402 122766 000015 000002      CMPB    #CR,2(SP)      ;;IS CHARACTER A CARRIAGE RETURN?
4819 037410 001003          BNE     1$             ;;BRANCH IF NO
4820 037412 105037 037432      CLRB    $CHARCNT      ;;YES--CLEAR CHARACTER COUNT
4821 037416 000406          BR     $TYPEX         ;;EXIT
4822 037420 122766 000012 000002 1$:      CMPB    #LF,2(SP)      ;;IS CHARACTER A LINE FEED?
4823 037426 001402          BEQ    $TYPEX         ;;BRANCH IF YES
4824 037430 105227          INCB   (PC)+          ;;COUNT THE CHARACTER
4825 037432 000000      $CHARCNT: .WORD    0      ;;CHARACTER COUNT STORAGE
4826 037434 000207      $TYPEX: RTS         PC
4827
4828          .SBTTL  APT COMMUNICATIONS ROUTINE
4829
4830          ;*****
4831 037436 112737 000001 037702  $ATY1: MOVB    #1,$FFLG      ;;TO REPORT FATAL ERROR
4832 037444 112737 000001 037700  $ATY3: MOVB    #1,$MFLG      ;;TO TYPE A MESSAGE
4833 037452 000403          BR     $ATYC
4834 037454 112737 000001 037702  $ATY4: MOVB    #1,$FFLG      ;;TO ONLY REPORT FATAL ERROR
4835 037462          $ATYC:
4836 037462 010046          MOV    R0,-(SP)        ;;PUSH R0 ON STACK
4837 037464 010146          MOV    R1,-(SP)        ;;PUSH R1 ON STACK
4838 037466 105737 037700      TSTB   $MFLG          ;;SHOULD TYPE A MESSAGE?
4839 037472 001450          BEQ    5$             ;;IF NOT: BR
4840 037474 122737 000001 001246      CMPB   #APTENV,$ENV    ;;OPERATING UNDER APT?
4841 037502 001031          BNE   3$             ;;IF NOT: BR
4842 037504 132737 000100 001247      BITB   #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
4843 037512 001425          BEQ    3$             ;;IF NOT: BR
4844 037514 017600 000004          MOV    @4(SP),R0      ;;GET MESSAGE ADDR.
4845 037520 062766 000002 000004          ADD    #2,4(SP)        ;;BUMP RETURN ADDR.
4846 037526 005737 001226 1$:      TST    $MSGTYPE      ;;SEE IF DONE W/ LAST XMISSION?
4847 037532 001375          BNE   1$             ;;IF NOT: WAIT
4848 037534 010037 001242          MOV    R0,$MSGAD      ;;PUT ADDR IN MAILBOX
4849 037540 105720          2$:      TSTB   (R0)+          ;;FIND END OF MESSAGE
4850 037542 001376          BNE   2$
4851 037544 163700 001242          SUB    $MSGAD,R0      ;;SUB START OF MESSAGE
4852 037550 006200          ASR    R0             ;;GET MESSAGE LNGTH IN WORDS
4853 037552 010037 001244          MOV    R0,$MSGGLT     ;;PUT LENGTH IN MAILBOX
4854 037556 012737 000004 001226      MOV    #4,$MSGTYPE    ;;TELL APT TO TAKE MSG.
4855 037564 000413          BR     5$
4856 037566 017637 000004 037612 3$:      MOV    @4(SP),4$      ;;PUT MSG ADDR IN JSR LINKAGE
4857 037574 062766 000002 000004          ADD    #2,4(SP)        ;;BUMP RETURN ADDRESS
4858 037602 013746 177776          MOV    177776,-(SP)   ;;PUSH 177776 ON STACK
4859 037606 004737 037102          JSR   PC,$TYPE        ;;CALL TYPE MACRO
4860 037612 000000      4$:      .WORD    0
4861 037614          5$:
4862 037614 105737 037702      10$:     TSTB   $FFLG          ;;SHOULD REPORT FATAL ERROR?
4863 037620 001416          BEQ   12$            ;;IF NOT: BR
4864 037622 005737 001246          TST   $ENV           ;;RUNNING UNDER APT?
4865 037626 001413          BEQ   12$            ;;IF NOT: BR
4866 037630 005737 001226      11$:     TST    $MSGTYPE      ;;FINISHED LAST MESSAGE?
4867 037634 001375          BNE   11$            ;;IF NOT: WAIT
4868 037636 017637 000004 001230      MOV    @4(SP),$FATAL  ;;GET ERROR #
4869 037644 062766 000002 000004          ADD    #2,4(SP)        ;;BUMP RETURN ADDR.
4870 037652 005237 001226          INC    $MSGTYPE      ;;TELL APT TO TAKE ERROR
4871 037656 105037 037702      12$:     CLRB   $FFLG          ;;CLEAR FATAL FLAG
4872 037662 105037 037701          CLRB   $LFLG         ;;CLEAR LOG FLAG

```



```
4873 037666 105037 037700          CLRB   $MFLG      ;;CLEAR MESSAGE FLAG
4874 037672 012601          MOV    (SP)+,R1   ;;POP STACK INTO R1
4875 037674 012600          MOV    (SP)+,R0   ;;POP STACK INTO R0
4876 037676 000207          RTS     PC        ;;RETURN
4877 037700      000          $MFLG: .BYTE 0    ;;MESSG. FLAG
4878 037701      000          $LFLG: .BYTE 0    ;;LOG FLAG
4879 037702      000          $FFLG: .BYTE 0    ;;FATAL FLAG
4880          037704          .EVEN
4881          000200          APTSIZE=200
4882          000001          APTENV=001
4883          000100          APTSPool=100
4884          000040          APTCSUP=040
4885          .SBTTL  BINARY TO ASCII AND TYPE ROUTINE
4886
4887          ;;*****
4888          ;;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
4889          ;;BINARY-ASCII NUMBER AND TYPE IT.
4890          ;;CALL:
4891          ;;      MOV     NUMBER,-(SP)      ;;NUMBER TO BE TYPED
4892          ;;      TYPBN          ;;TYPE IT
4893
4894 037704 010146          $TYPBN: MOV    R1,-(SP)      ;;SAVE R1 ON THE STACK
4895 037706 016601 000006          MOV    6(SP),R1      ;;GET THE INPUT NUMBER
4896 037712 000261          SEC          ;;SET 'C' SO CAN KEEP TRACK OF THE NUMBER OF BITS
4897 037714 112737 000060 037756 1$:  MOVB   #'0,$BIN      ;;SET CHARACTER TO AN ASCII '0'.
4898 037722 006101          ROL    R1          ;;GET THIS BIT
4899 037724 001406          BEQ    2$          ;;DONE?
4900 037726 105537 037756          ADCB   $BIN        ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
4901 037732 104401 037756          TYPE   , $BIN      ;;GO TYPE THIS BIT
4902 037736 000241          CLC          ;;CLEAR 'C' SO CAN KEEP TRACK OF BITS
4903 037740 000765          BR    1$          ;;GO DO THE NEXT BIT
4904 037742 012601          2$:  MOV    (SP)+,R1      ;;POP THE STACK INTO R1
4905 037744 016666 000002 000004          MOV    2(SP),4(SP)  ;;ADJUST THE STACK
4906 037752 012616          MOV    (SP)+,(SP)
4907 037754 000002          RTI          ;;RETURN TO USER
4908 037756      000      000          $BIN:  .BYTE 0,0    ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
4909          .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
4910
4911          ;;*****
4912          ;;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
4913          ;;OCTAL (ASCII) NUMBER AND TYPE IT.
4914          ;;$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
4915          ;;CALL:
4916          ;;      MOV     NUM,-(SP)        ;;NUMBER TO BE TYPED
4917          ;;      TYPOS          ;;CALL FOR TYPEOUT
4918          ;;      .BYTE  N          ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
4919          ;;      .BYTE  M          ;;M=1 OR 0
4920          ;;          ;;1=TYPE LEADING ZEROS
4921          ;;          ;;0=SUPPRESS LEADING ZEROS
4922
4923          ;;$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
4924          ;;$TYPOS OR $TYPOC
4925          ;;CALL:
4926          ;;      MOV     NUM,-(SP)        ;;NUMBER TO BE TYPED
4927          ;;      TYPON          ;;CALL FOR TYPEOUT
4928          ;;
```


4985 040204 000000
4986
4987
4988
4989
4990
4991
4992
4993
4994
4995
4996
4997
4998 040206
4999 040206 010046
5000 040210 010146
5001 040212 010246
5002 040214 010346
5003 040216 010546
5004 040220 012746 020200
5005 040224 016605 000020
5006 040230 100004
5007 040232 005405
5008 040234 112766 000055 000001
5009 040242 005000
5010 040244 012703 040422
5011 040250 112723 000040
5012 040254 005002
5013 040256 016001 040412
5014 040262 160105
5015 040264 002402
5016 040266 005202
5017 040270 000774
5018 040272 060105
5019 040274 005702
5020 040276 001002
5021 040300 105716
5022 040302 100407
5023 040304 106316
5024 040306 103003
5025 040310 116663 000001 177777
5026 040316 052702 000060
5027 040322 052702 000040
5028 040326 110223
5029 040330 005720
5030 040332 020027 000010
5031 040336 002746
5032 040340 003002
5033 040342 010502
5034 040344 000764
5035 040346 105726
5036 040350 100003
5037 040352 116663 177777 177776
5038 040360 105013
5039 040362 012605
5040 040364 012603

```

$OMODE: .WORD 0 ;:NUMBER OF DIGITS TO TYPE
.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

;:*****
;:*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
;:*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
;:*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
;:*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
;:*REPLACED WITH SPACES.
;:*CALL:
;:* MOV NUM,-(SP) ;:PUT THE BINARY NUMBER ON THE STACK
;:* TYPDS ;:GO TO THE ROUTINE

$TYPDS:
MOV R0,-(SP) ;:PUSH R0 ON STACK
MOV R1,-(SP) ;:PUSH R1 ON STACK
MOV R2,-(SP) ;:PUSH R2 ON STACK
MOV R3,-(SP) ;:PUSH R3 ON STACK
MOV R5,-(SP) ;:PUSH R5 ON STACK
MOV #20200,-(SP) ;:SET BLANK SWITCH AND SIGN
MOV 20(SP),R5 ;:GET THE INPUT NUMBER
BPL 1$ ;:BR IF INPUT IS POS.
NEG R5 ;:MAKE THE BINARY NUMBER POS.
MOVB #'-,1(SP) ;:MAKE THE ASCII NUMBER NEG.
1$: CLR R0 ;:ZERO THE CONSTANTS INDEX
MOV #SDBLK,R3 ;:SETUP THE OUTPUT POINTER
MOVB #' ,(R3)+ ;:SET THE FIRST CHARACTER TO A BLANK
2$: CLR R2 ;:CLEAR THE BCD NUMBER
MOV $DTBL(R0),R1 ;:GET THE CONSTANT
3$: SUB R1,R5 ;:FORM THIS BCD DIGIT
BLT 4$ ;:BR IF DONE
INC R2 ;:INCREASE THE BCD DIGIT BY 1
BR 3$
4$: ADD R1,R5 ;:ADD BACK THE CONSTANT
TST R2 ;:CHECK IF BCD DIGIT=0
BNE 5$ ;:FALL THROUGH IF 0
TSTB (SP) ;:STILL DOING LEADING 0'S?
BMI 7$ ;:BR IF YES
5$: ASLB (SP) ;:MSD?
BCC 6$ ;:BR IF NO
MOVB 1(SP),-1(R3) ;:YES--SET THE SIGN
6$: BIS #'0,R2 ;:MAKE THE BCD DIGIT ASCII
7$: BIS #' ,R2 ;:MAKE IT A SPACE IF NOT ALREADY A DIGIT
MOVB R2,(R3)+ ;:PUT THIS CHARACTER IN THE OUTPUT BUFFER
TST (R0)+ ;:JUST INCREMENTING
CMP R0,#10 ;:CHECK THE TABLE INDEX
BLT 2$ ;:GO DO THE NEXT DIGIT
BGT 8$ ;:GO TO EXIT
MOV R5,R2 ;:GET THE LSD
BR 6$ ;:GO CHANGE TO ASCII
8$: TSTB (SP)+ ;:WAS THE LSD THE FIRST NON-ZERO?
BPL 9$ ;:BR IF NO
MOVB -1(SP),-2(R3) ;:YES--SET THE SIGN FOR TYPING
9$: CLRB (R3) ;:SET THE TERMINATOR
MOV (SP)+,R5 ;:POP STACK INTO R5
MOV (SP)+,R3 ;:POP STACK INTO R3

```

```
5041 040366 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
5042 040370 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
5043 040372 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
5044 040374 104401 040422  TYPE      $DBLK      ;;NOW TYPE THE NUMBER
5045 040400 016666 000002 000004  MOV      2(SP),4(SP)   ;;ADJUST THE STACK
5046 040406 012616      MOV      (SP)+,(SP)
5047 040410 000002      RTI                          ;;RETURN TO USER
5048 040412 023420      $DTBL: 10000.
5049 040414 001750      1000.
5050 040416 000144      100.
5051 040420 000012      10.
5052 040422 000004      $DBLK: .BLKW 4
5053                      .SBTTL SAVE AND RESTORE R0-R5 ROUTINES
5054
5055                      ;:*****
5056                      ;*SAVE R0-R5
5057                      ;*CALL:
5058                      ;* SAVREG
5059                      ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
5060                      ;*
5061                      ;*TOP---(+16)
5062                      ;* +2---(+18)
5063                      ;* +4---R5
5064                      ;* +6---R4
5065                      ;* +8---R3
5066                      ;*+10---R2
5067                      ;*+12---R1
5068                      ;*+14---R0
5069
5070                      $SAVREG:
5071 040432 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
5072 040434 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
5073 040436 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
5074 040440 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
5075 040442 010446      MOV      R4,-(SP)      ;;PUSH R4 ON STACK
5076 040444 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
5077 040446 016646 000022  MOV      22(SP),-(SP)   ;;SAVE PS OF MAIN FLOW
5078 040452 016646 000022  MOV      22(SP),-(SP)   ;;SAVE PC OF MAIN FLOW
5079 040456 016646 000022  MOV      22(SP),-(SP)   ;;SAVE PS OF CALL
5080 040462 016646 000022  MOV      22(SP),-(SP)   ;;SAVE PC OF CALL
5081 040466 000002      RTI
5082
5083                      ;*RESTORE R0-R5
5084                      ;*CALL:
5085                      ;* RESREG
5086                      $RESREG:
5087 040470 012666 000022  MOV      (SP)+,22(SP)   ;;RESTORE PC OF CALL
5088 040474 012666 000022  MOV      (SP)+,22(SP)   ;;RESTORE PS OF CALL
5089 040500 012666 000022  MOV      (SP)+,22(SP)   ;;RESTORE PC OF MAIN FLOW
5090 040504 012666 000022  MOV      (SP)+,22(SP)   ;;RESTORE PS OF MAIN FLOW
5091 040510 012605      MOV      (SP)+,R5      ;;POP STACK INTO R5
5092 040512 012604      MOV      (SP)+,R4      ;;POP STACK INTO R4
5093 040514 012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
5094 040516 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
5095 040520 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
5096 040522 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
```

```

5097 040524 000002          RTI
5098                      .SBTTL  DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
5099
5100                      ::*****
5101                      ::*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
5102                      ::*UNSIGNED OCTAL ASCII NUMBER.
5103                      ::*CALL
5104                      ::*      MOV      #PNTR,-(SP)      ::*POINTER TO LOW WORD OF BINARY NUMBER
5105                      ::*      JSR      PC,@#$DB20      ::*CALL THE ROUTINE
5106                      ::*      RETURN      ::*THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK
5107
5108
5109 040526 104413          $DB20: SAVREG      ::SAVE ALL REGISTERS
5110 040530 016601 000002  MOV      2(SP),R1      ::PICKUP THE POINTER TO LOW WORD
5111 040534 012705 040645  MOV      #$OCTVL+13.,R5  ::POINTER TO DATA TABLE
5112 040540 012704 000014  MOV      #12.,R4      ::DO ELEVEN CHARACTERS
5113 040544 012703 177770  MOV      #^C7,R3      ::MASK
5114 040550 012100          MOV      (R1)+,R0      ::LOWER WORD
5115 040552 012101          MOV      (R1)+,R1      ::HIGH WORD
5116 040554 005002          CLR      R2      ::TERMINATOR
5117 040556 110245          1$:  MOVVB  R2,-(R5)      ::PUT CHARACTER IN DATA TABLE
5118 040560 010002          MOV      R0,R2      ::GET THIS DIGIT
5119 040562 005304          DEC      R4      ::COUNT THIS CHARACTER
5120 040564 003007          BGT      3$      ::BR IF NOT THE LAST DIGIT
5121 040566 001405          BEQ      2$      ::BR IF IT IS THE LAST DIGIT
5122 040570 005205          INC      R5      ::ALL DIGITS DONE-ADJUST POINTER FOR FIRST
5123 040572 010566 000002  MOV      R5,2(SP)      ::ASCII CHAR. & PUT IT ON THE STACK
5124 040576 104414          RESREG      ::RESTORE ALL REGISTERS
5125 040600 000207          RTS      PC      ::RETURN TO USER
5126 040602 006203          2$:  ASR      R3      ::POSITION THE MASK FOR THE LAST DIGIT
5127 040604 006001          3$:  ROR      R1      ::POSITION THE BINARY NUMBER FOR
5128 040606 006000          ROR      R0      ::      THE NEXT OCTAL DIGIT
5129 040610 006001          ROR      R1
5130 040612 006000          ROR      R0
5131 040614 006001          ROR      R1
5132 040616 006000          ROR      R0
5133 040620 040302          BIC      R3,R2      ::MASK OUT ALL JUNK
5134 040622 062702 000060  ADD      #'0,R2      ::MAKE THIS CHAR. ASCII
5135 040626 000753          BR       1$      ::GO PUT IT IN THE DATA TABLE
5136 040630 000016          $OCTVL: .BLKB  14.      ::RESERVE DATA TABLE
  
```

5137
5138
5139
5140
5141
5142
5143
5144
5145
5146
5147
5148
5149
5150
5151
5152
5153
5154
5155
5156
5157
5158
5159
5160
5161
5162
5163
5164
5165
5166
5167
5168
5169
5170
5171
5172
5173
5174
5175
5176
5177
5178
5179
5180
5181
5182
5183
5184
5185
5186
5187
5188
5189
5190
5191
5192

040646 010046
040650 016600 000002
040654 005740
040656 111000
040660 006300
040662 016000 040702
040666 000200

000004 000002

.SBTTL TRAP DECODER

*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
*GO TO THAT ROUTINE.

```
$TRAP: MOV R0,-(SP)      ;;SAVE R0
        MOV 2(SP),R0    ;;GET TRAP ADDRESS
        TST -(R0)      ;;BACKUP BY 2
        MOVB (R0),R0   ;;GET RIGHT BYTE OF TRAP
        ASL R0         ;;POSITION FOR INDEXING
        MOV $TRPAD(R0),R0 ;;INDEX TO TABLE
        RTS R0         ;;GO TO ROUTINE
```

;;THIS IS USE TO HANDLE THE 'GETPRI' MACRO

```
$TRAP2: MOV (SP),-(SP) ;;MOVE THE PC DOWN
        MOV 4(SP),2(SP) ;;MOVE THE PSW DOWN
        RTI           ;;RESTORE THE PSW
```

.SBTTL TRAP TABLE

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

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

.SBTTL POWER DOWN AND UP ROUTINES

POWER DOWN ROUTINE

```
$PWRDN: MOV $ILLUP,@#PWRVEC ;;SET FOR FAST UP
        MOV #340,@#PWRVEC+2 ;;PRIO:7
        MOV R0,-(SP)      ;;PUSH R0 ON STACK
        MOV R1,-(SP)      ;;PUSH R1 ON STACK
        MOV R2,-(SP)      ;;PUSH R2 ON STACK
        MOV R3,-(SP)      ;;PUSH R3 ON STACK
        MOV R4,-(SP)      ;;PUSH R4 ON STACK
```

```
5193 040762 010546          MOV    R5,-(SP)          ;;PUSH R5 ON STACK
5194 040764 017746 140150    MOV    @SWR,-(SP)       ;;PUSH @SWR ON STACK
5195 040770 010637 041116    MOV    SP,$SAVR6        ;;SAVE SP
5196 040774 012737 041006 000024  MOV    #PWRUP,@PWRVEC   ;;SET UP VECTOR
5197 041002 000000          HALT
5198 041004 000776          BR     .-2              ;;HANG UP
5199
5200          ;:*****
5201          ;:POWER UP ROUTINE
5202 041006 012737 041112 000024 $PWRUP: MOV    #SILLUP,@PWRVEC ;;SET FOR FAST DOWN
5203 041014 013706 041116    MOV    $SAVR6,SP        ;;GET SP
5204 041020 005037 041116    CLR    $SAVR6           ;;WAIT LOOP FOR THE TTY
5205 041024 005237 041116    1$:   INC    $SAVR6       ;;WAIT FOR THE INC
5206 041030 001375          BNE    1$               ;;OF WORD
5207 041032 012677 140102    MOV    (SP)+,@SWR       ;;POP STACK INTO @SWR
5208 041036 012605          MOV    (SP)+,R5        ;;POP STACK INTO R5
5209 041040 012604          MOV    (SP)+,R4        ;;POP STACK INTO R4
5210 041042 012603          MOV    (SP)+,R3        ;;POP STACK INTO R3
5211 041044 012602          MOV    (SP)+,R2        ;;POP STACK INTO R2
5212 041046 012601          MOV    (SP)+,R1        ;;POP STACK INTO R1
5213 041050 012600          MOV    (SP)+,R0        ;;POP STACK INTO R0
5214 041052 012737 040734 000024  MOV    #PWRDN,@PWRVEC   ;;SET UP THE POWER DOWN VECTOR
5215 041060 012737 000340 000026  MOV    #340,@PWRVEC+2   ;;PRIO:7
5216 041066 104401          TYPE                    ;;REPORT THE POWER FAILURE
5217 041070 041120    $PWRMG: .WORD    PWRMSG    ;;POWER FAIL MESSAGE POINTER
5218 041072 012716          MOV    (PC)+,(SP)      ;;RESTART AT RESTRT
5219 041074 020464    $PWRAD: .WORD    RESTRT    ;;RESTART ADDRESS
5220 041076 042766 000020 000002  BIC    #20,2(SP)        ;;CLEAR 'T' BIT
5221 041104 005037 034276    CLR    $TBIT           ;;CLEAR THE 'T' BIT FLAG
5222 041110 000002          RTI
5223 041112 000000    $SILLUP: HALT          ;;THE POWER UP SEQUENCE WAS STARTED
5224 041114 000776          BR     .-2              ;; BEFORE THE POWER DOWN WAS COMPLETE
5225 041116 000000    $SAVR6: 0              ;;PUT THE SP HERE
5226 041120 006412 050040 053517  PWRMSG: .ASCIZ <12><15>? POWER FAILURE - RESTARTING ?<12><15>
5227 041126 051105 043040 044501
5228 041134 052514 042522 026440
5229 041142 051040 051505 040524
5230 041150 052122 047111 020107
5231 041156 006412      000
5232          041162          .EVEN
5233
5234
```

Address	Hex 1	Hex 2	Hex 3	Hex 4	Label	Description
5235					.SBTTL	ERROR MESSAGES, DATA HEADERS-TABLES & FORMATS
5236	041162	047125	054105	042520	EM1:	.ASCIZ /UNEXPECTED CPU TRAP TO LOC. 004/
5237	041170	052103	042105	041440		
5238	041176	052520	052040	040522		
5239	041204	020120	047524	046040		
5240	041212	041577	020056	030060		
5241	041220	000064				
5242	041222	047125	054105	042520	EM2:	.ASCIZ /UNEXPECTED MEM. MGMT. TRAP TO LOC. 250/
5243	041230	052103	042105	046440		
5244	041236	046505	020056	043515		
5245	041244	052115	020056	051124		
5246	041252	050101	052040	020117		
5247	041260	047514	027103	031040		
5248	041266	030065	000			
5249	041271	120	044522	051117	EM3:	.ASCIZ /PRIORITY BITS SET WRONG IN PSW/
5250	041276	052111	020131	044502		
5251	041304	051524	051440	052105		
5252	041312	053440	047522	043516		
5253	041320	044440	020116	051520		
5254	041326	000127				
5255	041330	047515	042504	041040	EM4:	.ASCIZ /MODE BITS SET WRONG IN PSW/
5256	041336	052111	020123	042523		
5257	041344	020124	051127	047117		
5258	041352	020107	047111	050040		
5259	041360	053523	000			
5260	041363	104	040525	020114	EM5:	.ASCIZ /DUAL ADDRESSING BETWEEN HI&LO BYTES OF PSW/
5261	041370	042101	051104	051505		
5262	041376	044523	043516	041040		
5263	041404	052105	042527	047105		
5264	041412	044040	023111	047514		
5265	041420	041040	052131	051505		
5266	041426	047440	020106	051520		
5267	041434	000127				
5268	041436	042513	047122	046105	EM6:	.ASCIZ /KERNEL R6 CHANGED BY WRITING USER R6/
5269	041444	051040	020066	044103		
5270	041452	047101	042507	020104		
5271	041460	054502	053440	044522		
5272	041466	044524	043516	052440		
5273	041474	042523	020122	033122		
5274	041502	000				
5275	041503	101	046440	046505	EM7:	.ASCIZ /A MEMORY MGMT. REG. TIMED OUT/
5276	041510	051117	020131	043515		
5277	041516	052115	020056	042522		
5278	041524	027107	052040	046511		
5279	041532	042105	047440	052125		
5280	041540	000				
5281	041541	123	046525	040515	EM10:	.ASCIZ /SUMMARY OF MEM. MGMT. REG. TIMEOUTS/
5282	041546	054522	047440	020106		
5283	041554	042515	027115	046440		
5284	041562	046507	027124	051040		
5285	041570	043505	020056	044524		
5286	041576	042515	052517	051524		
5287	041604	000				
5288	041605	115	046505	020056	EM11:	.ASCIZ /MEM. MGMT. REG. WOULD NOT CLEAR/
5289	041612	043515	052115	020056		
5290	041620	042522	027107	053440		

5291	041626	052517	042114	047040	
5292	041634	052117	041440	042514	
5293	041642	051101	000		
5294	041645	115	046505	020056	EM12: .ASCIZ /MEM. MGMT. REG. BITS NOT SET CORRECTLY/
5295	041652	043515	052115	020056	
5296	041660	042522	027107	041040	
5297	041666	052111	020123	047516	
5298	041674	020124	042523	020124	
5299	041702	047503	051122	041505	
5300	041710	046124	000131		
5301	041714	051123	020060	043105	EM13: .ASCIZ /SRO EFFECTED BY WRITE TO PSW/
5302	041722	042506	052103	042105	
5303	041730	041040	020131	051127	
5304	041736	052111	020105	047524	
5305	041744	050040	053523	000	
5306	041751	123	030522	042040	EM14: .ASCIZ /SR1 DID NOT READ ALL ZEROS/
5307	041756	042111	047040	052117	
5308	041764	051040	040505	020104	
5309	041772	046101	020114	042532	
5310	042000	047522	000123		
5311	042004	052504	046101	040440	EM15: .ASCIZ /DUAL ADDRESSING BETWEEN BYTES OF PAR OR PDR/
5312	042012	042104	042522	051523	
5313	042020	047111	020107	042502	
5314	042026	053524	042505	020116	
5315	042034	054502	042524	020123	
5316	042042	043117	050040	051101	
5317	042050	047440	020122	042120	
5318	042056	000122			
5319	042060	052504	046101	040440	EM16: .ASCIZ /DUAL ADDRESSING BETWEEN PAR-PDR'S/
5320	042066	042104	042522	051523	
5321	042074	047111	020107	042502	
5322	042102	053524	042505	020116	
5323	042110	040520	026522	042120	
5324	042116	023522	000123		
5325	042122	044120	051531	041511	EM17: .ASCIZ /PHYSICAL ADDRESS FORMED WRONG/
5326	042130	046101	040440	042104	
5327	042136	042522	051523	043040	
5328	042144	051117	042515	020104	
5329	042152	051127	047117	000107	
5330	042160	044120	051531	020056	EM20: .ASCIZ /PHYS. ADDR. FORMED WRONG IN RELOCATE MODE/
5331	042166	042101	051104	020056	
5332	042174	047506	046522	042105	
5333	042202	053440	047522	043516	
5334	042210	044440	020116	042522	
5335	042216	047514	040503	042524	
5336	042224	046440	042117	000105	
5337	042232	026527	044502	020124	EM21: .ASCIZ /W-BIT DID NOT GET SET IN PDR/
5338	042240	044504	020104	047516	
5339	042246	020124	042507	020124	
5340	042254	042523	020124	047111	
5341	042262	050040	051104	000	
5342	042267	127	041055	052111	EM22: .ASCIZ /W-BIT SET IN MORE THAN ONE PDR/
5343	042274	051440	052105	044440	
5344	042302	020116	047515	042522	
5345	042310	052040	040510	020116	
5346	042316	047117	020105	042120	

5347	042324	000122				
5348	042326	026527	044502	020124	EM23:	.ASCIZ /W-BIT NOT CLEARED BY WRITING TO PDR/
5349	042334	047516	020124	046103		
5350	042342	040505	042522	020104		
5351	042350	054502	053440	044522		
5352	042356	044524	043516	052040		
5353	042364	020117	042120	000122		
5354	042372	051127	052111	047111	EM24:	.ASCIZ /WRITING SRO SET W-BIT IN KIPDR7/
5355	042400	020107	051123	020060		
5356	042406	042523	020124	026527		
5357	042414	044502	020124	047111		
5358	042422	045440	050111	051104		
5359	042430	000067				
5360	042432	026527	044502	020124	EM25:	.ASCIZ /W-BIT GOT SET DURING TIMEOUT ABORT/
5361	042440	047507	020124	042523		
5362	042446	020124	052504	044522		
5363	042454	043516	052040	046511		
5364	042462	047505	052125	040440		
5365	042470	047502	052122	000		
5366	042475	115	046505	051117	EM26:	.ASCIZ /MEMORY MGMT. ACCESS ABORT DID NOT OCCUR/
5367	042502	020131	043515	052115		
5368	042510	020056	041501	042503		
5369	042516	051523	040440	047502		
5370	042524	052122	042040	042111		
5371	042532	047040	052117	047440		
5372	042540	041503	051125	000		
5373	042545	101	041503	051505	EM27:	.ASCIZ /ACCESS ERROR DID NOT ABORT INSTRUCTION/
5374	042552	020123	051105	047522		
5375	042560	020122	044504	020104		
5376	042566	047516	020124	041101		
5377	042574	051117	020124	047111		
5378	042602	052123	052522	052103		
5379	042610	047511	000116			
5380	042614	051123	020060	044504	EM30:	.ASCIZ /SRO DID NOT REPORT ACCESS ERROR CORRECTLY/
5381	042622	020104	047516	020124		
5382	042630	042522	047520	052122		
5383	042636	040440	041503	051505		
5384	042644	020123	051105	047522		
5385	042652	020122	047503	051122		
5386	042660	041505	046124	000131		
5387	042666	044504	020104	047516	EM31:	.ASCIZ /DID NOT LOCKUP CORRECT VIRTUAL ADDR./
5388	042674	020124	047514	045503		
5389	042702	050125	041440	051117		
5390	042710	042522	052103	053040		
5391	042716	051111	052524	046101		
5392	042724	040440	042104	027122		
5393	042732	000				
5394	042733	120	043501	020105	EM32:	.ASCIZ /PAGE LGTH. ABORT OCCURRED WHEN IT SHOULDN'T HAVE/
5395	042740	043514	044124	020056		
5396	042746	041101	051117	020124		
5397	042754	041517	052503	051122		
5398	042762	042105	053440	042510		
5399	042770	020116	052111	051440		
5400	042776	047510	046125	047104		
5401	043004	052047	044040	053101		
5402	043012	000105				

5403	043014	040520	042507	046040	EM33: .ASCIZ /PAGE LGTH. ABORT DID NOT OCCUR WHEN IT SHOULD HAVE/
5404	043022	052107	027110	040440	
5405	043030	047502	052122	042040	
5406	043036	042111	047040	052117	
5407	043044	047440	041503	051125	
5408	043052	053440	042510	020116	
5409	043060	052111	051440	047510	
5410	043066	046125	020104	040510	
5411	043074	042526	000		
5412	043077	123	030122	042040	EM34: .ASCIZ /SR0 DID NOT REPORT PAGE LGTH. ABORT CORRECTLY/
5413	043104	042111	047040	052117	
5414	043112	051040	050105	051117	
5415	043120	020124	040520	042507	
5416	043126	046040	052107	027110	
5417	043134	040440	047502	052122	
5418	043142	041440	051117	042522	
5419	043150	052103	054514	000	
5420	043155	123	030122	047440	EM37: .ASCIZ /SR0 OR SR2 CHANGED BY A SECCND ABORT/
5421	043162	020122	051123	020062	
5422	043170	044103	047101	042507	
5423	043176	020104	054502	040440	
5424	043204	051440	041505	047117	
5425	043212	020104	041101	051117	
5426	043220	000124			
5427	043222	051123	020060	051117	EM40: .ASCIZ /SR0 OR SR2 WERE NOT 'RESET' BY A RESET/
5428	043230	051440	031122	053440	
5429	043236	051105	020105	047516	
5430	043244	020124	051042	051505	
5431	043252	052105	020042	054502	
5432	043260	040440	051040	051505	
5433	043266	052105	000		
5434	043271	123	031122	047040	EM41: .ASCIZ /SR2 NOT TRACKING CORRECTLY/
5435	043276	052117	052040	040522	
5436	043304	045503	047111	020107	
5437	043312	047503	051122	041505	
5438	043320	046124	000131		
5439	043324	044504	020104	047516	EM42: .ASCIZ /DID NOT TRAP THRU KERNEL SPACE/
5440	043332	020124	051124	050101	
5441	043340	052040	051110	020125	
5442	043346	042513	047122	046105	
5443	043354	051440	040520	042503	
5444	043362	000			
5445	043363	113	020124	051105	EM43: .ASCIZ /KT ERROR NOT SERVICED ON TIMEOUT ERROR/
5446	043370	047522	020122	047516	
5447	043376	020124	042523	053122	
5448	043404	041511	042105	047440	
5449	043412	020116	044524	042515	
5450	043420	052517	020124	051105	
5451	043426	047522	000122		
5452	043432	051123	020060	051117	EM44: .ASCIZ /SR0 OR SR2 CHANGED BY TIMEOUT ERROR/
5453	043440	051440	031122	041440	
5454	043446	040510	043516	042105	
5455	043454	041040	020131	044524	
5456	043462	042515	052517	020124	
5457	043470	051105	047522	000122	
5458	043476	051105	047522	020122	EM45: .ASCIZ /ERROR DURING 'DOUBLE ERROR' (KT & TIMEOUT)/

5459	043504	052504	044522	043516	
5460	043512	021040	047504	041125	
5461	043520	042514	042440	051122	
5462	043526	051117	020042	045450	
5463	043534	020124	020046	044524	
5464	043542	042515	052517	024524	
5465	043550	000			
5466	043551	115	050106	020111	EM46: .ASCIZ /MFPI INSTRUCTION PUSHED WRONG DATA/
5467	043556	047111	052123	052522	
5468	043564	052103	047511	020116	
5469	043572	052520	044123	042105	
5470	043600	053440	047522	043516	
5471	043606	042040	052101	000101	
5472	043614	052115	044520	044440	EM47: .ASCIZ /MTPI INSTRUCTION LOADED WRONG DATA/
5473	043622	051516	051124	041525	
5474	043630	044524	047117	046040	
5475	043636	040517	042504	020104	
5476	043644	051127	047117	020107	
5477	043652	040504	040524	000	
5478	043657	123	040524	045503	EM50: .ASCIZ /STACK NOT PUSHED BY MFPI-MTPI/
5479	043664	047040	052117	050040	
5480	043672	051525	042510	020104	
5481	043700	054502	046440	050106	
5482	043706	026511	052115	044520	
5483	043714	000			
5484	043715	113	051105	042516	EM51: .ASCIZ /KERNEL PAGE ACCESS INSTEAD OF USER: MFPI-MTPI/
5485	043722	020114	040520	042507	
5486	043730	040440	041503	051505	
5487	043736	020123	047111	052123	
5488	043744	040505	020104	043117	
5489	043752	052440	042523	035122	
5490	043760	046440	050106	026511	
5491	043766	052115	044520	000	
5492	043773	127	047522	043516	EM52: .ASCIZ /WRONG PDR'S REFERENCED WHILE IN RELOCATE MODE/
5493	044000	050040	051104	051447	
5494	044006	051040	043105	051105	
5495	044014	047105	042503	020104	
5496	044022	044127	046111	020105	
5497	044030	047111	051040	046105	
5498	044036	041517	052101	020105	
5499	044044	047515	042504	000	
5500	044051	115	050106	020104	EM53: .ASCIZ /MFPD INSTRUCTION PUSHED WRONG DATA/
5501	044056	047111	052123	052522	
5502	044064	052103	047511	020116	
5503	044072	052520	044123	042105	
5504	044100	053440	047522	043516	
5505	044106	042040	052101	000101	
5506	044114	052123	041501	020113	EM54: .ASCIZ /STACK NOT PUSHED BY MFPD-MTPD/
5507	044122	047516	020124	052520	
5508	044130	044123	042105	041040	
5509	044136	020131	043115	042120	
5510	044144	046455	050124	000104	
5511	044152	040520	020122	051117	EM55: .ASCIZ /PAR OR PDR CHANGED BY A RESET/
5512	044160	050040	051104	041440	
5513	044166	040510	043516	042105	
5514	044174	041040	020131	020101	

5515	044202	042522	042523	000124	
5516	044210	051520	020127	044103	EM56: .ASCIZ /PSW CHANGED BY AN RTI IN USER MODE/
5517	044216	047101	042507	020104	
5518	044224	054502	040440	020116	
5519	044232	052122	020111	047111	
5520	044240	052440	042523	020122	
5521	044246	047515	042504	000	
5522					
5523	044253	117	042114	050040	DH1: .ASCIZ /OLD PC OLD PSW R6 WAS TESTNO ERRORPC/
5524	044260	020103	047440	042114	
5525	044266	050040	053523	051040	
5526	044274	020066	040527	020123	
5527	044302	052040	051505	047124	
5528	044310	020117	042440	051122	
5529	044316	051117	041520	000	
5530	044323	117	042114	050040	DH2: .ASCIZ /OLD PC OLD PSW R6 WAS SR0 SR2 TESTNO ERRORPC/
5531	044330	020103	047440	042114	
5532	044336	050040	053523	051040	
5533	044344	020066	040527	020123	
5534	044352	051440	030122	020040	
5535	044360	020040	051440	031122	
5536	044366	020040	020040	052040	
5537	044374	051505	047124	020117	
5538	044402	042440	051122	051117	
5539	044410	041520	000		
5540	044413	127	047522	042524	DH3: .ASCIZ /WROTE READ TESTNO ERRORPC/
5541	044420	020040	051040	040505	
5542	044426	020104	020040	052040	
5543	044434	051505	047124	020117	
5544	044442	042440	051122	051117	
5545	044450	041520	000		
5546	044453	101	042104	042522	DH7: .ASCIZ /ADDRESS TESTNO ERRORPC/
5547	044460	051523	052040	051505	
5548	044466	047124	020117	042440	
5549	044474	051122	051117	041520	
5550	044502	000			
5551	044503	122	043505	051511	DH10: .ASCII /REGISTER-ADDRS NUM OF/<CRLF>
5552	044510	042524	026522	042101	
5553	044516	051104	020123	047040	
5554	044524	046525	020040	043117	
5555	044532	200			
5556	044533	101	042116	042455	.ASCIZ /AND-ED OR-ED TIMOUTS TESTNO ERRORPC/
5557	044540	020104	047440	026522	
5558	044546	042105	020040	052040	
5559	044554	046511	052517	051524	
5560	044562	052040	051505	047124	
5561	044570	020117	042440	051122	
5562	044576	051117	041520	000	
5563	044603	122	043505	051511	DH11: .ASCII /REGISTR READ READ-(BINARY)/<CRLF>
5564	044610	051124	051040	040505	
5565	044616	020104	020040	051040	
5566	044624	040505	026504	041050	
5567	044632	047111	051101	024531	
5568	044640	200			
5569	044641	101	042104	042522	.ASCIZ /ADDRESS (OCTAL) 5432109876543210 TESTNO ERRORPC/
5570	044646	051523	024040	041517	

5627	045336	051523	053040	040502					
5628	045344	020040	020040	053040					
5629	045352	040502	020040	020040					
5630	045360	050040	051101	032040					
5631	045366	020040	050040	051101					
5632	045374	032440	020040	050040					
5633	045402	053523	020040	020040					
5634	045410	052040	051505	047124					
5635	045416	020117	042440	051122					
5636	045424	051117	041520	000					
5637	045431	120	051104	020040	DH21:	.ASCII	/PDR	VIRTUAL/<CRLF>	
5638	045436	020040	053040	051111					
5639	045444	052524	046101	200					
5640	045451	124	051505	042524		.ASCIZ	/TESTED	ADDRESS TESTNO ERRORPC/	
5641	045456	020104	040440	042104					
5642	045464	042522	051523	052040					
5643	045472	051505	047124	020117					
5644	045500	042440	051122	051117					
5645	045506	041520	000						
5646	045511	120	051104	044440	DH22:	.ASCII	/PDR IN PDR	VIRTUAL/	
5647	045516	020116	050040	051104					
5648	045524	020040	020040	053040					
5649	045532	051111	052524	046101					
5650	045540	051105	047522	020122		.ASCIZ	/ERROR	TESTED ADDRESS TESTNO ERRORPC/	
5651	045546	020040	042524	052123					
5652	045554	042105	020040	042101					
5653	045562	051104	051505	020123					
5654	045570	042524	052123	047516					
5655	045576	020040	051105	047522					
5656	045604	050122	000103						
5657	045610	042120	020122	020040	DH23:	.ASCIZ	/PDR	TESTNO ERRORPC/	
5658	045616	020040	042524	052123					
5659	045624	047516	020040	051105					
5660	045632	047522	050122	000103					
5661	045640	042120	020122	040527	DH24:	.ASCIZ	/PDR WAS EXPECTD	TESTNO ERRORPC/	
5662	045646	020123	054105	042520					
5663	045654	052103	020104	042524					
5664	045662	052123	047516	020040					
5665	045670	051105	047522	050122					
5666	045676	000103							
5667	045700	042120	020122	020064	DH26:	.ASCIZ	/PDR 4 PSW	TESTNO ERRORPC/	
5668	045706	020040	051520	020127					
5669	045714	020040	020040	042524					
5670	045722	052123	047516	020040					
5671	045730	051105	047522	050122					
5672	045736	000103							
5673	045740	051123	020060	040527	DH30:	.ASCIZ	/SR0 WAS EXPECTD	PDR 4 PSW TESTNO ERRORPC/	
5674	045746	020123	054105	042520					
5675	045754	052103	020104	042120					
5676	045762	020122	020064	020040					
5677	045770	051520	020127	020040					
5678	045776	020040	042524	052123					
5679	046004	047516	020040	051105					
5680	046012	047522	050122	000103					
5681	046020	051123	020062	040527	DH31:	.ASCIZ	/SR2 WAS EXPECTD	PDR 4 PSW TESTNO ERRORPC/	
5682	046026	020123	054105	042520					

5683	046034	052103	020104	042120	
5684	046042	020122	020064	020040	
5685	046050	051520	020127	020040	
5686	046056	020040	042524	052123	
5687	046064	047516	020040	051105	
5688	046072	047522	050122	000103	
5689	046100	027126	027102	027101	DH32: .ASCIZ /V.B.A. KIPDR4 SR0 WAS SR2 WAS TESTNO ERRORPC/
5690	046106	020040	044513	042120	
5691	046114	032122	020040	051123	
5692	046122	020060	040527	020123	
5693	046130	051123	020062	040527	
5694	046136	020123	042524	052123	
5695	046144	047516	020040	051105	
5696	046152	047522	050122	000103	
5697	046160	027126	027102	027101	DH33: .ASCIZ /V.B.A. KIPDR4 TESTNO ERRORPC/
5698	046166	020040	044513	042120	
5699	046174	032122	020040	042524	
5700	046202	052123	047516	020040	
5701	046210	051105	047522	050122	
5702	046216	000103			
5703	046220	027126	027102	027101	DH34: .ASCIZ /V.B.A. KIPDR4 SR0 WAS EXPECTD TESTNO ERRORPC/
5704	046226	020040	044513	042120	
5705	046234	032122	020040	051123	
5706	046242	020060	040527	020123	
5707	046250	054105	042520	052103	
5708	046256	020104	042524	052123	
5709	046264	047516	020040	051105	
5710	046272	047522	050122	000103	
5711	046300	027126	027102	027101	DH35: .ASCIZ /V.B.A. KIPDR4 SR2 WAS EXPECTD TESTNO ERRORPC/
5712	046306	020040	044513	042120	
5713	046314	032122	020040	051123	
5714	046322	020062	040527	020123	
5715	046330	054105	042520	052103	
5716	046336	020104	042524	052123	
5717	046344	047516	020040	051105	
5718	046352	047522	050122	000103	
5719	046360	051123	020062	040527	DH36: .ASCIZ /SR2 WAS EXPECTD TESTNO ERRORPC/
5720	046366	020123	054105	042520	
5721	046374	052103	020104	042524	
5722	046402	052123	047516	020040	
5723	046410	051105	047522	050122	
5724	046416	000103			
5725	046420	044506	051522	020124	DH37: .ASCII /FIRST ABORT SECOND ABORT/<CRLF>
5726	046426	041101	051117	020124	
5727	046434	020040	020040	042523	
5728	046442	047503	042116	040440	
5729	046450	047502	052122	200	
5730	046455	123	030122	053440	.ASCIZ /SR0 WAS SR2 WAS SR0 WAS SR2 WAS TESTNO ERRORPC/
5731	046462	051501	051440	031122	
5732	046470	053440	051501	051440	
5733	046476	030122	053440	051501	
5734	046504	051440	031122	053440	
5735	046512	051501	052040	051505	
5736	046520	047124	020117	042440	
5737	046526	051122	051117	041520	
5738	046534	000			

5739	046535	123	030122	053440	DH40:	.ASCIZ	/SRO WAS SR2 WAS TESTNO ERRORPC/
5740	046542	051501	051440	031122			
5741	046550	053440	051501	052040			
5742	046556	051505	047124	020117			
5743	046564	042440	051122	051117			
5744	046572	041520	000				
5745	046575	120	053523	053440	DH42:	.ASCIZ	/PSW WAS R6 WAS TESTNO ERRORPC/
5746	046602	051501	051040	020066			
5747	046610	040527	020123	052040			
5748	046616	051505	047124	020117			
5749	046624	042440	051122	051117			
5750	046632	041520	000				
5751	046635	105	050130	041505	DH44:	.ASCII	/EXPECTED RECEIVED/<CRLF>
5752	046642	042524	020104	020040			
5753	046650	020040	020040	020040			
5754	046656	042522	042503	053111			
5755	046664	042105	200				
5756	046667	123	030122	020040		.ASCIZ	/SRO SR2 SRO WAS SR2 WAS TESTNO ERRORPC/
5757	046674	020040	051440	031122			
5758	046702	020040	020040	051440			
5759	046710	030122	053440	051501			
5760	046716	051440	031122	053440			
5761	046724	051501	052040	051505			
5762	046732	047124	020117	042440			
5763	046740	051122	051117	041520			
5764	046746	000					
5765	046747	105	050130	041505	DH45:	.ASCII	/EXPECTED:<CRLF>
5766	046754	042524	035104	200			
5767	046761	120	053523	020040		.ASCII	/PSW PC SRO SR2/<CRLF>
5768	046766	020040	050040	020103			
5769	046774	020040	020040	051440			
5770	047002	030122	020040	020040			
5771	047010	051440	031122	200			
5772	047015	061	030067	030460		.ASCII	/170017 (3\$+4) 020147 (3\$)/<CRLF>
5773	047022	020067	024040	022063			
5774	047030	032053	020051	030040			
5775	047036	030062	032061	020067			
5776	047044	024040	022063	100051			
5777	047052	042522	042503	053111		.ASCII	/RECEIVED:<CRLF>
5778	047060	042105	100072				
5779	047064	051520	020127	020040		.ASCIZ	/PSW PC SRO SR2 TESTNO ERRORPC/
5780	047072	020040	041520	020040			
5781	047100	020040	020040	051123			
5782	047106	020060	020040	020040			
5783	047114	051123	020062	020040			
5784	047122	020040	042524	052123			
5785	047130	047516	020040	051105			
5786	047136	047522	050122	000103			
5787	047144	040504	040524	020040	DH46:	.ASCII	/DATA DATA/<CRLF>
5788	047152	020040	040504	040524			
5789	047160	200					
5790	047161	105	050130	041505		.ASCIZ	/EXPECTD RECEIVD TESTNO ERRORPC/
5791	047166	042124	051040	041505			
5792	047174	044505	042126	052040			
5793	047202	051505	047124	020117			
5794	047210	042440	051122	051117			

J 9

5795	047216	041520	000						
5796	047221	124	051505	047124	DH50:	.ASCIZ	/TESTNO	ERRORPC/	
5797	047226	020117	042440	051122					
5798	047234	051117	041520	000					
5799	047241	123	030122	053440	DH51:	.ASCIZ	/SRO WAS SR2 WAS TESTNO	ERRORPC/	
5800	047246	051501	051440	031122					
5801	047254	053440	051501	052040					
5802	047262	051505	047124	020117					
5803	047270	042440	051122	051117					
5804	047276	041520	000						
5805	047301	120	054510	044523	DH52:	.ASCII	/PHYSICL PAR 4/<CRLF>		
5806	047306	046103	050040	051101					
5807	047314	032040	200						
5808	047317	101	042104	042522		.ASCIZ	/ADDRESS V.B.A. PAR 4	SRO WAS SR2 WAS PSW	TESTNO ERRORPC/
5809	047324	051523	053040	041056					
5810	047332	040456	020056	050040					
5811	047340	051101	032040	020040					
5812	047346	051440	030122	053440					
5813	047354	051501	051440	031122					
5814	047362	053440	051501	050040					
5815	047370	053523	020040	020040					
5816	047376	052040	051505	047124					
5817	047404	020117	042440	051122					
5818	047412	051117	041520	000					
5819	047417	120	053523	053440	DH56:	.ASCIZ	/PSW WAS EXPECTD TESTNO	ERRORPC/	
5820	047424	051501	042440	050130					
5821	047432	041505	042124	052040					
5822	047440	051505	047124	020117					
5823	047446	042440	051122	051117					
5824	047454	041520	000						
5825									
5826		047460				.EVEN			
5827									
5828	047460	001266	001270	001264	DT1:	.WORD	TRAPPC, TRAPPS, WASR6, TESTNO, \$ERRPC, 0		
5829	047466	001262	001116	000000					
5830	047474	001266	001270	001264	DT2:	.WORD	TRAPPC, TRAPPS, WASR6, WASSR0, WASSR2, TESTNO, \$ERRPC, 0		
5831	047502	001272	001274	001262					
5832	047510	001116	000000						
5833	047514	001162	001164	001262	DT3:	.WORD	\$REG0, \$REG1, TESTNO, \$ERRPC, 0		
5834	047522	001116	000000						
5835	047526	001162	001262	001116	DT7:	.WORD	\$REG0, TESTNO, \$ERRPC, 0		
5836	047534	000000							
5837	047536	001300	001302	001304	DT10:	.WORD	ANDADR, ORADR, TONUM, TESTNO, \$ERRPC, 0		
5838	047544	001262	001116	000000					
5839	047552	001162	001164	001164	DT11:	.WORD	\$REG0, \$REG1, \$REG1, TESTNO, \$ERRPC, 0		
5840	047560	001262	001116	000000					
5841	047566	001162	001164	001166	DT12:	.WORD	\$REG0, \$REG1, \$REG2, \$REG2, TESTNO, \$ERRPC, 0		
5842	047574	001166	001262	001116					
5843	047602	000000							
5844	047604	001162	001262	001116	DT13:	.WORD	\$REG0, TESTNO, \$ERRPC, 0		
5845	047612	000000							
5846	047614	001162	001164	001174	DT16:	.WORD	\$REG0, \$REG1, \$REG5, \$REG2, TESTNO, \$ERRPC, 0		
5847	047622	001166	001262	001116					
5848	047630	000000							
5849	047632	001312	001306	001172	DT17:	.WORD	PBALO, VIRT1, \$REG4, TESTNO, \$ERRPC, 0		
5850	047640	001262	001116	000000					

5851	047646	001312	001306	001310	DT20:	.WORD	PBALO,VIRT1,VIRT2,\$REG4,\$REG5,\$TMP0,TESTNO,\$ERRPC,0
5852	047654	001172	001174	001176			
5853	047662	001262	001116	000000			
5854	047670	001174	001170	001262	DT21:	.WORD	\$REG5,\$REG3,TESTNO,\$ERRPC,0
5855	047676	001116	000000				
5856	047702	001162	001174	001170	DT22:	.WORD	\$REG0,\$REG5,\$REG3,TESTNO,\$ERRPC,0
5857	047710	001262	001116	000000			
5858	047716	001174	001262	001116	DT23:	.WORD	\$REG5,TESTNO,\$ERRPC,0
5859	047724	000000					
5860	047726	001166	001164	001262	DT24:	.WORD	\$REG2,\$REG1,TESTNO,\$ERRPC,0
5861	047734	001116	000000				
5862	047740	001166	001176	001262	DT26:	.WORD	\$REG2,\$TMP0,TESTNO,\$ERRPC,0
5863	047746	001116	000000				
5864	047752	001272	001170	001166	DT30:	.WORD	WASSR0,\$REG3,\$REG2,\$TMP0,TESTNO,\$ERRPC,0
5865	047760	001176	001262	001116			
5866	047766	000000					
5867	047770	001274	001172	001166	DT31:	.WORD	WASSR2,\$REG4,\$REG2,\$TMP0,TESTNO,\$ERRPC,0
5868	047776	001176	001262	001116			
5869	050004	000000					
5870	050006	001162	001172	001272	DT32:	.WORD	\$REG0,\$REG4,WASSR0,WASSR2,TESTNO,\$ERRPC,0
5871	050014	001274	001262	001116			
5872	050022	000000					
5873	050024	001162	001172	001262	DT33:	.WORD	\$REG0,\$REG4,TESTNO,\$ERRPC,0
5874	050032	001116	000000				
5875	050036	001162	001172	001272	DT34:	.WORD	\$REG0,\$REG4,WASSR0,\$REG2,TESTNO,\$ERRPC,0
5876	050044	001166	001262	001116			
5877	050052	000000					
5878	050054	001162	001172	001274	DT35:	.WORD	\$REG0,\$REG4,WASSR2,\$REG3,TESTNO,\$ERRPC,0
5879	050062	001170	001262	001116			
5880	050070	000000					
5881	050072	001274	001164	001262	DT36:	.WORD	WASSR2,\$REG1,TESTNO,\$ERRPC,0
5882	050100	001116	000000				
5883	050104	001176	001202	001272	DT37:	.WORD	\$TMP0,\$TMP2,WASSR0,WASSR2,TESTNO,\$ERRPC,0
5884	050112	001274	001262	001116			
5885	050120	000000					
5886	050122	001272	001274	001262	DT40:	.WORD	WASSR0,WASSR2,TESTNO,\$ERRPC,0
5887	050130	001116	000000				
5888	050134	001164	001166	001262	DT42:	.WORD	\$REG1,\$REG2,TESTNO,\$ERRPC,0
5889	050142	001116	000000				
5890	050146	001162	001164	001272	DT44:	.WORD	\$REG0,\$REG1,WASSR0,WASSR2,TESTNO,\$ERRPC,0
5891	050154	001274	001262	001116			
5892	050162	000000					
5893	050164	001164	001170	001272	DT45:	.WORD	\$REG1,\$REG3,WASSR0,WASSR2,TESTNO,\$ERRPC,0
5894	050172	001274	001262	001116			
5895	050200	000000					
5896	050202	001162	001164	001262	DT46:	.WORD	\$REG0,\$REG1,TESTNO,\$ERRPC,0
5897	050210	001116	000000				
5898	050214	001262	001116	000000	DT50:	.WORD	TESTNO,\$ERRPC,0
5899	050222	001272	001274	001262	DT51:	.WORD	WASSR0,WASSR2,TESTNO,\$ERRPC,0
5900	050230	001116	000000				
5901	050234	001312	001306	001172	DT52:	.WORD	PBALO,VIRT1,\$REG4,WASSR0,WASSR2,\$TMP0,TESTNO,\$ERRPC,0
5902	050242	001272	001274	001176			
5903	050250	001262	001116	000000			
5904	050256	001164	001166	001262	DT56:	.WORD	\$REG1,\$REG2,TESTNO,\$ERRPC,0
5905	050264	001116	000000				
5906							

5907	050270	000	000	000	DF1:	.BYTE	0,0,0,0,0
5908	050273	000	000				
5909	050275	000	000	000	DF2:	.BYTE	0,0,0,0,0,0,0
5910	050300	000	000	000			
5911	050303	000					
5912	050304	000	000	000	DF3:	.BYTE	0,0,0,0
5913	050307	000					
5914	050310	000	000	000	DF7:	.BYTE	0,0,0
5915	050313	000	000	001	DF10:	.BYTE	0,0,1,0,0
5916	050316	000	000				
5917	050320	000	000	002	DF11:	.BYTE	0,0,2,0,0
5918	050323	000	000				
5919	050325	000	000	000	DF12:	.BYTE	0,0,0,2,0,0
5920	050330	002	000	000			
5921	050333	000	000	000	DF13:	.BYTE	0,0,0
5922	050336	000	000	000	DF16:	.BYTE	0,0,0,0,0,0
5923	050341	000	000	000			
5924	050344	003	000	000	DF17:	.BYTE	3,0,0,0,0
5925	050347	000	000				
5926	050351	003	000	000	DF20:	.BYTE	3,0,0,0,0,0,0,0
5927	050354	000	000	000			
5928	050357	000	000				
5929	050361	000	000	000	DF21:	.BYTE	0,0,0,0
5930	050364	000					
5931	050365	000	000	000	DF22:	.BYTE	0,0,0,0,0
5932	050370	000	000				
5933	050372	000	000	000	DF23:	.BYTE	0,0,0
5934	050375	000	000	000	DF24:	.BYTE	0,0,0,0
5935	050400	000					
5936	050401	000	000	000	DF30:	.BYTE	0,0,0,0,0,0
5937	050404	000	000	000			
5938	050407	000	000	000	DF46:	.BYTE	0,0,0,0
5939	050412	000					
5940	050413	000	000		DF50:	.BYTE	0,0
5941	050415	000	000	000	DF51:	.BYTE	0,0,0,0
5942	050420	000					
5943	050421	003	000	000	DF52:	.BYTE	3,0,0,0,0,0,0,0
5944	050424	000	000	000			
5945	050427	000	000				
5946	050431	000	000	000	DF56:	.BYTE	0,0,0,0
5947	050434	000					
5948							
5949		000001				.END	

DF7	050310	960	5914#			
DH1	044253	922	5523#			
DH10	044503	964	5551#			
DH11	044603	971	1203	5563#		
DH12	044723	978	997	5578#		
DH13	045063	985	991	5595#		
DH16	045113	1004	5600#			
DH17	045213	1011	5612#			
DH2	044323	928	5530#			
DH20	045303	1018	5622#			
DH21	045431	1025	5637#			
DH22	045511	1032	5646#			
DH23	045610	1039	1135	5657#		
DH24	045640	1045	1051	5661#		
DH26	045700	1057	1063	5667#		
DH3	044413	934	940	946	952	5540#
DH30	045740	1069	5673#			
DH31	046020	1075	5681#			
DH32	046100	1081	5689#			
DH33	046160	1087	5697#			
DH34	046220	1093	5703#			
DH35	046300	1098	5711#			
DH36	046360	1104	1123	5719#		
DH37	046420	1110	5725#			
DH40	046535	1117	5739#			
DH42	046575	1129	5745#			
DH44	046635	1141	5751#			
DH45	046747	1148	5765#			
DH46	047144	1158	1165	1190	5787#	
DH50	047221	1172	1197	5796#		
DH51	047241	1178	5799#			
DH52	047301	1184	5805#			
DH56	047417	1210	5819#			
DH7	044453	958	5546#			
DISPLA	001142	824#	1322*	1330*	4174*	4203*
DISPRE	000174	760#	1330			
DOAGIN	034176	4079	4090#			
DSWR =	177570	591#	823	1321		
DT1	047460	923	5828#			
DT10	047536	966	5837#			
DT11	047552	973	1205	5839#		
DT12	047566	980	999	5841#		
DT13	047604	986	992	5844#		
DT16	047614	1006	5846#			
DT17	047632	1013	5849#			
DT2	047474	929	5830#			
DT20	047646	1020	5851#			
DT21	047670	1027	5854#			
DT22	047702	1034	5856#			
DT23	047716	1040	1136	5858#		
DT24	047726	1046	1052	5860#		
DT26	047740	1058	1064	5862#		
DT3	047514	935	941	947	953	5833#
DT30	047752	1070	5864#			
DT31	047770	1076	5867#			
DT32	050006	1082	5870#			

CJKDADO KTF11-AA MMU DIAG
CJKDAD.P11 19-DEC-80 11:05

MACY11 30A(1052) 14-JAN-81 11:36 PAGE 120
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0119

DT33	050024	1088	5873#		
DT34	050036	1094	5875#		
DT35	050054	1099	5878#		
DT36	050072	1105	1124	5881#	
DT37	050104	1112	5883#		
DT40	050122	1118	5886#		
DT42	050134	1130	5888#		
DT44	050146	1143	5890#		
DT45	050164	1153	5893#		
DT46	050202	1160	1167	1192	5896#
DT50	050214	1173	1198	5898#	
DT51	050222	1179	5899#		
DT52	050234	1186	5901#		
DT56	050256	1211	5904#		
DT7	047526	959	5835#		
EMTVEC=	000030	680#	1291*	1292*	
EM1	041162	921	5236#		
EM10	041541	963	5281#		
EM11	041605	970	5288#		
EM12	041645	977	5294#		
EM13	041714	984	5301#		
EM14	041751	990	5306#		
EM15	042004	996	5311#		
EM16	042060	1003	5319#		
EM17	042122	1010	5325#		
EM2	041222	927	5242#		
EM20	042160	1017	5330#		
EM21	042232	1024	5337#		
EM22	042267	1031	5342#		
EM23	042326	1038	5348#		
EM24	042372	1044	5354#		
EM25	042432	1050	5360#		
EM26	042475	1056	5366#		
EM27	042545	1062	5373#		
EM3	041271	933	5249#		
EM30	042614	1068	5380#		
EM31	042666	1074	1097	1103	5387#
EM32	042733	1080	5394#		
EM33	043014	1086	5403#		
EM34	043077	1092	5412#		
EM37	043155	1109	5420#		
EM4	041330	939	5255#		
EM40	043222	1116	5427#		
EM41	043271	1122	5434#		
EM42	043324	1128	5439#		
EM43	043363	1134	5445#		
EM44	043432	1140	5452#		
EM45	043476	1147	5458#		
EM46	043551	1157	5466#		
EM47	043614	1164	5472#		
EM5	041363	945	5260#		
EM50	043657	1171	5478#		
EM51	043715	1177	5484#		
EM52	043773	1183	5492#		
EM53	044051	1189	5500#		
EM54	044114	1196	5506#		

\$CHARC	037432	4777*	4787*	4794	4820*	4825#												
\$CKSWR	035760	4527#	5177															
\$CLR.T	034164	4081	4084#															
\$CMTAG	001100	803#	1282	1283	1291	1297	1298	1299										
\$CM1 =	000006	835#	836#	837#	838#	839#	840#	841#										
\$CM2 =	000014	835#	836#	837#	838#	839#	840#	841#										
\$CM3 =	000006	833#	835															
\$CM4 =	000006	841#	842#	843#	844#	845#	846#	847#										
\$CNTLC	036710	4553	4643	4692#														
\$CNTLG	036722	4538	4694#															
\$CNTLU	036715	4563	4667	4693#														
\$CPUOP	001254	872#																
\$CRLF	001223	851#	4077	4214	4239	4260	4279	4284	4327	4574	4672	4692	4776	4828				
\$DBLK	040422	5010	5044	5052#														
\$DB20	040526	4313	5109#															
\$DEVCT	001236	863#																
\$DOAGN	034234	4049	4096#															
\$DTBL	040412	5013	5048#															
\$ENDAD	034166	770	1344	4085#	4234													
\$ENDCT	C34016	1297	4051#															
\$ENULL	034300	4112#																
\$ENV	001246	868#	1350	1805	2059	4216	4755	4840	4864									
\$ENVM	001247	869#	1334	4757	4762	4842												
\$EOP	033762	4041#	4720															
\$EOPCT	034010	1297*	4048#	4052														
\$ERFLG	001103	806#	4119	4148	4150	4156*	4178	4201*	4239									
\$ERMAX	001115	812#	1300*	4150	4173*	4178												
\$ERROR	034564	1291	4192#															
\$ERRPC	001116	813#	4208*	4209*	4210	4239	4266	5828	5830	5833	5835	5837	5839	5841				
		5844	5846	5849	5851	5854	5856	5858	5860	5862	5864	5867	5870	5873				
		5875	5878	5881	5883	5886	5888	5890	5893	5896	5898	5899	5901	5904				
		918#	4274															
\$ERRTB	001316	810#	1339*	4074	4207*	4239												
\$ERTTL	001112	848#	1299*	4172*	4230	4232	4239											
\$ESCAP	001214	867#																
\$ETABL	001246	795	887#															
\$ETEND	001262	860#	4868*															
\$FATAL	001230	4831*	4834*	4862	4871*	4879#												
\$FFLG	037702	831#	4780	4828														
\$FILLC	001156	830#	4828															
\$FILLS	001155	814#																
\$GDADR	001120	816#																
\$GDDAT	001124	4078#																
\$GET42	034146	4539#	5175															
\$GTSWR	036030	563																
\$HD =	000000	790#																
\$HIBTS	000204	807#	4163*	4164	4166*	4177												
\$ICNT	001104	5186	5202	5223#														
\$ILLUP	041112	821#	4555	4575	4699													
\$INTAG	001135	811#	4210*	4218	4239	4263												
\$ITEMB	001114	852#	4239	4682	4692	4828												
\$LF	001224	4872*	4878#															
\$LFLG	037701	4104	4108#															
\$LOOP	034272	808#	1315*	4154*	4170*	4175	4177											
\$LPADR	001106	809#	1316*	1384*	1399*	1407*	1422*	1433*	1447*	1460*	1512*	1525*	1554*	1565*				
\$LPERR	001110	1594*	1605*	1634*	1645*	1674*	1685*	1726*	1735*	1759*	1831*	1840*	1860*	1875*				

	752#	2143	2170	2197	2292	2319	2346	2373								
ADDTST	685#															
COMMEN	685#															
ENDCOM	579#	1238	1269	1392	1416	1443	1456	1481	1528	1533	1568	1573	1608	1613	1648	
ERROR	1653	1688	1693	1722	1731	1743	1752	1778	1797	1804	1815	1835	1847	1879	1892	
	1929	1941	1973	1985	2070	2081	2092	2104	2161	2188	2215	2243	2267	2310	2337	
	2364	2391	2468	2524	2570	2581	2590	2636	2647	2656	2688	2701	2752	2760	2768	
	2775	2819	2827	2835	2842	2923	2934	2941	2953	3036	3047	3054	3066	3147	3188	
	3197	3221	3237	3246	3287	3321	3347	3357	3364	3435	3511	3516	3529	3541	3552	
	3564	3579	3592	3607	3621	3657	3675	3692	3707	3723	3741	3758	3777	3791	3840	
	3845	3858	3870	3881	3893	3907	3920	3934	3949	3975	3980	3991	4018	4025	4417	
	4431	4445	4459													
ESCAPE	685#															
GETPRI	685#															
GETSWR	685#	1347#														
MSG1	1376#	1378														
MSG10	1624#	1626														
MSG11	1664#	1666														
MSG12	1704#	1706														
MSG13	1761#	1763														
MSG14	1785#	1787														
MSG15	1817#	1821														
MSG16	1862#	1864														
MSG17	1909#	1911														
MSG2	1401#	1403														
MSG20	1953#	1955														
MSG21	1997#	1999														
MSG21A	2046#	2048														
MSG22	2111#	2113														
MSG23	2276#	2278														
MSG24	2402#	2404														
MSG25	2534#	2536														
MSG26	2600#	2602														
MSG27	2667#	2669														
MSG3	1425#	1427														
MSG30	2723#	2725														
MSG31	2796#	2798														
MSG32	2885#	2888														
MSG33	3000#	3003														
MSG34	3114#	3117														
MSG35	3164#	3167														
MSG36	3256#	3259														
MSG36A	3301#	3303														
MSG37	3327#	3330														
MSG4	1462#	1464														
MSG40	3374#	3377														
MSG41	3456#	3458														
MSG42	3630#	3632														
MSG43	3801#	3803														
MSG44	3957#	3959														
MSG45	3999#	4001														
MSG5	1499#	1502														
MSG6	1544#	1546														
MSG7	1584#	1586														
MULT	685#															
NEWTST	685#	1376	1401	1425	1462	1500	1544	1584	1624	1664	1704	1761	1785	1819	1862	

.\$APTH	553#	774
.\$APTY	553#	4828
.\$CATC	553#	753
.\$CMTA	553#	796
.\$DB20	553#	5098
.\$EOP	553#	
.\$ERRO	553#	4178
.\$ERRT	553#	
.\$POWE	553#	5182
.\$READ	553#	4517
.\$SAVE	553#	5053
.\$SCOP	553#	4114
.\$TRAP	553#	5137
.\$TYPB	553#	4885
.\$TYPD	553#	4986
.\$TYPE	553#	4732
.\$TYPO	553#	4909

. ABS. C50435 000

ERRORS DETECTED: 0

DSKZ:CJKDAD,DSKZ:CJKDAD/CRF/SOL/NL:TOC=CJKDAD.P11

RUN-TIME: 95 56 4 SECONDS

RUN-TIME RATIO: 239/156=1.5

CORE USED: 32K (63 PAGES)