

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52

.REM e

IDENTIFICATION

PRODUCT CODE: AC-8054B-MC
PRODUCT NAME: CFKTHBO PDP 11/34 MEM MGMT
DATE: JUNE 22, 1978
MAINTAINER: DIAGNOSTIC PROGRAMMING
AUTHOR: DIAGNOSTIC ENGINEERING

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

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

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1977, 1978 BY DIGITAL EQUIPMENT CORPORATION

53
54
55
56
57
58
59
60
61
62
63
64
65
66
67

PROGRAM HISTORY

DATE	REVISION	REASON FOR REVISION
31-JAN-77	A	FIRST RELEASE
28-JUN-78	B	HARDWARE FAULT NOT DETECTED

68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103

TABLE OF CONTENTS

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

104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159

1.0 PROGRAM INFORMATION

1.1 ABSTRACT

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

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

1.2 REQUIREMENTS

A PDP 11/34 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 DIAGNOSTICS SHOULD BE RUN:

MD-11-DFKAA	PDP 11/34 BASIC CPU TESTS
MD-11-DFKAB	PDP 11/34 TRAPS 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

160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215

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 AND STARTING. IF THE PROCESSOR HAS THE OPTIONAL PROGRAMMER'S CONSOLE, THE SWITCH REGISTER SHOULD BE SET ACCORDING TO SECTION 2.3 BEFORE THE PROGRAM IS STARTED. IF 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)

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

2.3 CONTROL SWITCH SETTINGS

SWITCH	OCTAL VALUE	USE
SW15	100000	HALT ON ERROR THIS SWITCH WHEN SET WILL HALT THE PROCESSOR WHEN AN ERROR IS DETECTED AFTER THE ERROR MESSAGE HAS BEEN TYPED. PRESSING CONTINUE WILL RESUME TESTING (SEE SECTION 3.1 ABOUT LOADING THE SWITCH REG BEFORE CONTINUING).
SW14	040000	LOOP ON TEST THIS SWITCH WHEN SET WILL CAUSE THE PROGRAM TO LOOP ON THE CURRENT SUBTEST.
SW13	020000	INHIBIT ERROR TYPEOUTS THIS SWITCH WHEN SET WILL INHIBIT THE TYPING OF ERROR MESSAGES.
SW12	010000	INHIBIT TRACE TRAP THIS SWITCH WHEN SET WILL INHIBIT T-BIT TRAPPING WHICH

216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271

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 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

THE HARDWARE SWITCH REGISTER PROVIDED WHEN THE OPTIONAL
PROGRAMMER'S CONSOLE IS PRESENT IS LOADED DIRECTLY FROM
THE CONSOLE KEYPAD BY DEPRESSING THE "LSR" KEY. THE
VALUE OF THE HARDWARE SWITCH REG. CAN BE CHANGED ANY
TIME WHETHER THE PROGRAM IS RUNNING OR NOT.

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

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

SWR = XXXXXX NEW =

WHERE "XXXXXX" IS THE CURRENT OCTAL CONTENTS OF LOC. 176.

THE OPERATOR MAY THEN TYPE ANY ONE OF THE FOLLOWING:
XXXXXX<CR> ONE TO SIX OCTAL DIGITS FOLLOWED BY A

CARRIAGE RETURN WHICH WILL BE LOADED
AS THE NEW VALUE FOR THE SWITCH REG.

<CR> JUST A <CR>, LEAVES THE SWITCH REG.
AS IT IS.

XXX^U A CONTROL-U (^U) WILL CAUSE ALL OF THE

272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327

^C DIGITS TYPED SO FAR TO BE IGNORED.
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
----- 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 3 1/4 MINUTES.

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

328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383

TO THE PROGRAM LISTING, THE OPERATOR CAN THEN READ THE COMMENTS ASSOCIATED WITH THAT PARTICULAR ERROR AND SUBTEST. A DESCRIPTION OF THE TEST FOUND IN THE PROGRAM LISTING WILL ALSO PROVIDE THE OPERATOR WITH INFORMATION ON THE LOGIC AND FUNCTIONS BEING TESTED.

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

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

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 (SR0). IN THE LISTING, THE TEST DESCRIPTION SAYS THAT THE ERROR BITS (BITS <15:13>) OF SR0 WERE BEING SET AND CLEARED INDIVIDUALLY. THE ERROR REPORT SAYS WE TRIED TO SET BIT 14 BY WRITING "040000" TO SR0 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

384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439

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

4.3 T-BIT TRAPPING

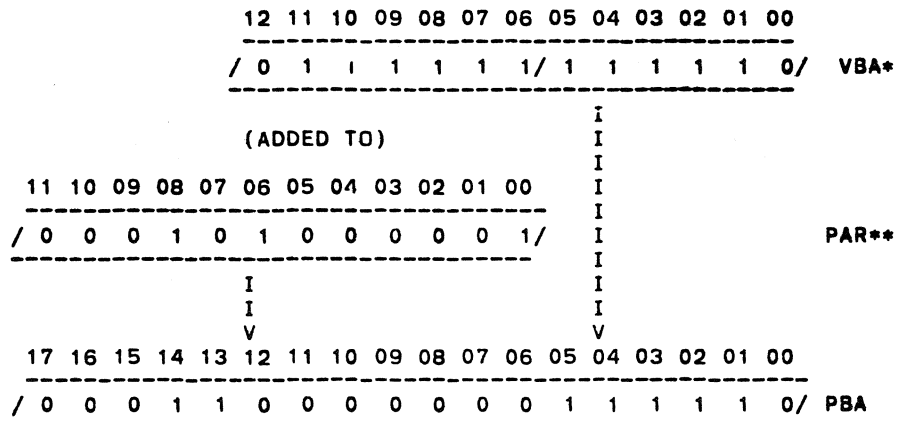
THE "T-BIT" (BIT 4) IN THE PROCESSOR STATUS WORD IS SET BY AN "RTI" IN THE END-OF-PASS ROUTINE FOR EVERY OTHER PASS BEGINNING WITH THE THIRD PASS (PASSES 3,5,7,9...). T-BIT TRAPPING CAN BE INHIBITED BY SETTING BIT 12 = 1 IN THE SWITCH REGISTER (SEE SECTION 2.4).

4.4 POWER FAILURE HANDLING

IF A POWER FAIL OCCURS (FOLLOWED BY A POWER UP), THE MESSAGE "POWER FAILURE-RESTARTING" IS TYPED OUT AND THE PROGRAM WILL RESTART EXECUTION AT "START:" (THE VERY BEGINNING OF THE PROGRAM. IF THE SOFTWARE SWITCH REGISTER IS BEING USED, ITS CONTENTS WILL BE RESTORED. IF THERE IS A HARDWARE SWITCH REGISTER, THERE IS NO WAY TO RESTORE THE VALUE OF THE SWITCH REGISTER SO THE OPERATOR MUST RELOAD IT FROM THE CONSOLE.

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

440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495

KERNEL (=0) SET OF PAR'S/PDR'S

5.0 PROGRAM DESCRIPTION

5.1 SUBROUTINES USED BY THIS PROGRAM

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

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

5.2 PROGRAM LISTING

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

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

5.3 USING THE PROGRAM TO DIAGNOSE A FAULT

WHEN AN ERROR OCCURS, ONE OF THE THINGS THAT'S IMPORTANT TO NOTE IS WHAT PASS THE ERROR OCCURRED ON. IF THE PASS NUMBER IS ODD AND IS THREE OR GREATER, THE ERROR MIGHT BE T-BIT SENSITIVE. TRY RUNNING THE PROGRAM AGAIN WITH BIT 12 OF THE SWITCH REG. EQUAL TO "1" TO INHIBIT T-BIT TRAPPING. IF THE PASS NUMBER IS GREATER THAN ONE, THE ERROR MAY BE ITERATION SENSITIVE. TRY RUNNING THE PROGRAM AGAIN WITH BIT 11 OF THE SWITCH REG. EQUAL TO "1" TO INHIBIT ITERATIONS. THESE HINTS SHOULD HELP YOU DETERMINE WHAT MAKES THE MACHINE FAIL AND WHEN.

IF YOU HAVE BEEN RUNNING WITH BIT 15 OF THE SWITCH 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.

496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514

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".

```

515 .TITLE CFKTHBO PDP 11/34 MEM MGNT DIAG
516 ;*
517 ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
518 ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
519 ;*
520 .SBTTL OPERATIONAL SWITCH SETTINGS
521 ;*
522 ;* SWITCH USE
523 ;* -----
524 ;* 15 HALT ON ERROR
525 ;* 14 LOOP ON TEST
526 ;* 13 INHIBIT ERROR TYPEDUTS
527 ;* 12 INHIBIT TRACE TRAP
528 ;* 11 INHIBIT ITERATIONS
529 ;* 10 BELL ON ERROR
530 ;* 9 LOOP ON ERROR
531 ;* 8 LOOP ON TEST IN SWR<7:0>
532 .SBTTL BASIC DEFINITIONS
533 ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
534 STACK= 1100
535 001100 .EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
536 .EQUIV IDT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL
537
538 ;*MISCELLANEOUS DEFINITIONS
539
540 000011 HT= 11 ;;CODE FOR HORIZONTAL TAB
541 000012 LF= 12 ;;CODE FOR LINE FEED
542 000015 CR= 15 ;;CODE FOR CARRIAGE RETURN
543 000200 CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
544 177776 PS= 177776 ;;PROCESSOR STATUS WORD
545 .EQUIV PS,PSW
546 177774 STKLM= 177774 ;;STACK LIMIT REGISTER
547 177772 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
548 177570 DSWR= 177570 ;;HARDWARE SWITCH REGISTER
549 177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
550
551 ;*GENERAL PURPOSE REGISTER DEFINITIONS
552 000000 R0= %0 ;;GENERAL REGISTER
553 000001 R1= %1 ;;GENERAL REGISTER
554 000002 R2= %2 ;;GENERAL REGISTER
555 000003 R3= %3 ;;GENERAL REGISTER
556 000004 R4= %4 ;;GENERAL REGISTER
557 000005 R5= %5 ;;GENERAL REGISTER
558 000006 R6= %6 ;;GENERAL REGISTER
559 000007 R7= %7 ;;GENERAL REGISTER
560 000005 SP= %6 ;;STACK POINTER
561 000007 PC= %7 ;;PROGRAM COUNTER
562
563 ;*PRIORITY LEVEL DEFINITIONS
564 000000 PR0= 0 ;;PRIORITY LEVEL 0
565 000040 PR1= 40 ;;PRIORITY LEVEL 1
566 000100 PR2= 100 ;;PRIORITY LEVEL 2
567 000140 PR3= 140 ;;PRIORITY LEVEL 3
568 000200 PR4= 200 ;;PRIORITY LEVEL 4
569 000240 PR5= 240 ;;PRIORITY LEVEL 5
570

```

```

571 000300 PR6= 300 ;;PRIORITY LEVEL 6
572 000340 PR7= 340 ;;PRIORITY LEVEL 7
573
574 ;*"SWITCH REGISTER" SWITCH DEFINITIONS
575 100000 SW15= 100000
576 040000 SW14= 40000
577 020000 SW13= 20000
578 010000 SW12= 10000
579 004000 SW11= 4000
580 002000 SW10= 2000
581 001000 SW09= 1000
582 000400 SW08= 400
583 000200 SW07= 200
584 000100 SW06= 100
585 000040 SW05= 40
586 000020 SW04= 20
587 000010 SW03= 10
588 000004 SW02= 4
589 000002 SW01= 2
590 000001 SW00= 1
591 .EQUIV SW09,SW9
592 .EQUIV SW08,SW8
593 .EQUIV SW07,SW7
594 .EQUIV SW06,SW6
595 .EQUIV SW05,SW5
596 .EQUIV SW04,SW4
597 .EQUIV SW03,SW3
598 .EQUIV SW02,SW2
599 .EQUIV SW01,SW1
600 .EQUIV SW00,SW0
601
602 ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
603 100000 BIT15= 100000
604 040000 BIT14= 40000
605 020000 BIT13= 20000
606 010000 BIT12= 10000
607 004000 BIT11= 4000
608 002000 BIT10= 2000
609 001000 BIT09= 1000
610 000400 BIT08= 400
611 000200 BIT07= 200
612 000100 BIT06= 100
613 000040 BIT05= 40
614 000020 BIT04= 20
615 000010 BIT03= 10
616 000004 BIT02= 4
617 000002 BIT01= 2
618 000001 BIT00= 1
619 .EQUIV BIT09,BIT9
620 .EQUIV BIT08,BIT8
621 .EQUIV BIT07,BIT7
622 .EQUIV BIT06,BIT6
623 .EQUIV BIT05,BIT5
624 .EQUIV BIT04,BIT4
625 .EQUIV BIT03,BIT3
626 .EQUIV BIT02,BIT2

```

```

627 .EQUIV BIT01,BIT1
628 .EQUIV BIT00,BIT0
629
630 ;*BASIC "CPU" TRAP VECTOR ADDRESSES
631 000004 ERRVEC= 4 ;;TIME OUT AND OTHER ERRORS
632 000010 RESVEC= 10 ;;RESERVED AND ILLEGAL INSTRUCTIONS
633 000014 TBITVEC=14 ;; "T" BIT
634 000014 TRTVEC= 14 ;;TRACE TRAP
635 000014 BPTVEC= 14 ;;BREAKPOINT TRAP (BPT)
636 000020 IOTVEC= 20 ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
637 000024 PWRVEC= 24 ;;POWER FAIL
638 000030 EMTVEC= 30 ;;EMULATOR TRAP (EMT) **ERROR**
639 000034 TRAPVEC=34 ;; "TRAP" TRAP
640 000060 TKVEC= 60 ;;TTY KEYBOARD VECTOR
641 000064 TPVEC= 64 ;;TTY PRINTER VECTOR
642 000240 PIRQVEC=240 ;;PROGRAM INTERRUPT REQUEST VECTOR
643 .SBTTL MEMORY MANAGEMENT DEFINITIONS
644
645 ;*KT11 VECTOR ADDRESS
646
647 000250 MMVEC= 250
648
649 ;*KT11 STATUS REGISTER ADDRESSES
650
651 177572 SR0= 177572
652 177574 SR1= 177574
653 177576 SR2= 177576
654 172516 SR3= 172516
655
656 ;*USER "I" PAGE DESCRIPTOR REGISTERS
657
658 177600 UIPDR0= 177600
659 177602 UIPDR1= 177602
660 177604 UIPDR2= 177604
661 177606 UIPDR3= 177606
662 177610 UIPDR4= 177610
663 177612 UIPDR5= 177612
664 177614 UIPDR6= 177614
665 177616 UIPDR7= 177616
666
667 ;*USER "I" PAGE ADDRESS REGISTERS
668
669 177640 UIPAR0= 177640
670 177642 UIPAR1= 177642
671 177644 UIPAR2= 177644
672 177646 UIPAR3= 177646
673 177650 UIPAR4= 177650
674 177652 UIPAR5= 177652
675 177654 UIPAR6= 177654
676 177656 UIPAR7= 177656
677
678 ;*KERNEL "I" PAGE DESCRIPTOR REGISTERS
679
680 172300 KIPDR0= 172300
681 172302 KIPDR1= 172302
682 172304 KIPDR2= 172304
    
```

```

683 172306 KIPDR3= 172306
684 172310 KIPDR4= 172310
685 172312 KIPDR5= 172312
686 172314 KIPDR6= 172314
687 172316 KIPDR7= 172316
688
689 ;*KERNEL "I" PAGE ADDRESS REGISTERS
690
691 172340 KIPAR0= 172340
692 172342 KIPAR1= 172342
693 172344 KIPAR2= 172344
694 172346 KIPAR3= 172346
695 172350 KIPAR4= 172350
696 172352 KIPAR5= 172352
697 172354 KIPAR6= 172354
698 172356 KIPAR7= 172356
699
700 .EQUIV SP,KSP
701 .EQUIV SP,USP
702 .EQUIV BIT4,TBIT
703 .EQUIV BIT6,WBIT
704 001100 KERSTK= STACK
705 000700 USESTK= STACK-200
706
707 ;*ADDITIONAL DEFINITIONS
708 ;*
709
710 .SBTTL TRAP CATCHER
711
712 .=0
713 000000 .=0
714 ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
715 ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
716 ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
717 .=174
718 000174 000000 DISPREG: .WORD 0 ;;SOFTWARE DISPLAY REGISTER
719 000176 000000 SWREG: .WORD 0 ;;SOFTWARE SWITCH REGISTER
720 .SBTTL STARTING ADDRESS(ES)
721 000200 000137 020000 JMP @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
722 .SBTTL ACT11 HOOKS
723
724 ;*****
725 ;HOOKS REQUIRED BY ACT11
726 000204 $SVP=.; ;SAVE PC
727 000046 .=46 ;
728 000046 034760 $ENDAD ;(1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
729 000052 .=52 ;
730 000052 000000 .WORD 0 ;(2)SET LOC.52 TO ZERO
731 000204 .=$SVPC ;; RESTORE PC
732 .SBTTL APT PARAMETER BLOCK
733
734 ;*****
735 ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
736 ;*****
737 000204 .X=.; ;SAVE CURRENT LOCATION
738 000024 .=24 ;;SET POWER FAIL TO POINT TO START OF PROGRAM
    
```

```

739 000024 000200          200      ;;FOR APT START UP
740          000044          .-44      ;;POINT TO APT INDIRECT ADDRESS PNTR.
741 000044 000204          $APTHDR ;;POINT TO APT HEADER BLOCK
742          000204          .-$X      ;;RESET LOCATION COUNTER
743          ;;*****
744          ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
745          ;INTERFACE SPEC.
746
747 000204          $APTHD:
748 000204 000000          $HIBITS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
749 000206 001226          $MBADR: .WORD $MAIL  ;;ADDRESS OF APT MAILBOX (BITS 0-15)
750 000210 000010          $STMT: .WORD 10     ;;RUN TIM OF LONGEST TEST
751 000212 000020          $PASTM: .WORD 20     ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
752 000214 000005          $UNITM: .WORD 5      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
753 000216 000052          .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
    
```

```

754          .SBTTL COMMON TAGS
755          ;;*****
756          ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
757          ;*USED IN THE PROGRAM.
758
759          .-1100
760          $CMTAG:          ;;START OF COMMON TAGS
761 001100          .WORD 0
762 001100 000000          $TSTNM: .BYTE 0      ;;CONTAINS THE TEST NUMBER
763 001102 000      $ERFLG: .BYTE 0      ;;CONTAINS ERROR FLAG
764 001103 000      $ICNT: .WORD 0      ;;CONTAINS SUBTEST ITERATION COUNT
765 001104 000000          $LPADR: .WORD 0      ;;CONTAINS SCOPE LOOP ADDRESS
766 001106 000000          $LPERR: .WORD 0      ;;CONTAINS SCOPE RETURN FOR ERRORS
767 001110 000.30          $ERTTL: .WORD 0      ;;CONTAINS TOTAL ERRORS DETECTED
768 001112 000000          $ITEMB: .BYTE 0      ;;CONTAINS ITEM CONTROL BYTE
769 001114 000      $ERRMAX: .BYTE 1      ;;CONTAINS MAX. ERRORS PER TEST
770 001115 001      $ERRPC: .WORD 0      ;;CONTAINS PC OF LAST ERROR INSTRUCTION
771 001116 000000          $GDADR: .WORD 0      ;;CONTAINS ADDRESS OF 'GOOD' DATA
772 001120 000000          $BDADR: .WORD 0      ;;CONTAINS ADDRESS OF 'BAD' DATA
773 001122 000000          $GGDAT: .WORD 0      ;;CONTAINS 'GOOD' DATA
774 001124 000000          $BDDAT: .WORD 0      ;;CONTAINS 'BAD' DATA
775 001126 000000          .WORD 0      ;;RESERVED--NOT TO BE USED
776 001130 000000          .WORD 0
777 001132 000000          $AUTOB: .BYTE 0      ;;AUTOMATIC MODE INDICATOR
778 001134 000      $INTAG: .BYTE 0      ;;INTER:PT MODE INDICATOR
779 001135 000          .WORD 0
780 001136 000000          $SWR: .WORD $DSWR  ;;ADDRESS OF SWITCH REGISTER
781 001140 177570          $DISPLAY: .WORD $DISP  ;;ADDRESS OF DISPLAY REGISTER
782 001142 177570          $ST: 3: 177560  ;;TTY KBD STATUS
783 001144 177560          $TKB: 177562  ;;TTY KBD BUFFER
784 001146 177562          $TPS: 177564  ;;TTY PRINTER STATUS REG. ADDRESS
785 001150 177564          $TPB: 177566  ;;TTY PRINTER BUFFER REG. ADDRESS
786 001152 177566          $NULL: .BYTE 0      ;;CONTAINS NULL CHARACTER FOR FILLS
787 001154 000          $FILLS: .BYTE 2      ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
788 001155 002          $FILLC: .BYTE 12     ;;INSERT FILL CHARS. AFTER A "LINE FEED"
789 001156 012          $TPFLG: .BYTE 0      ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
790 001157 000          $REGAD: .WORD 0      ;;CONTAINS THE ADDRESS FROM
791 001160 000000          .WHICH ($REGO) WAS OBTAINED
792
793 001162 000000          $REG0: .WORD 0      ;;CONTAINS (($REGAD)+0)
794 001164 000000          $REG1: .WORD 0      ;;CONTAINS (($REGAD)+2)
795 001166 000000          $REG2: .WORD 0      ;;CONTAINS (($REGAD)+4)
796 001170 000000          $REG3: .WORD 0      ;;CONTAINS (($REGAD)+6)
797 001172 000000          $REG4: .WORD 0      ;;CONTAINS (($REGAD)+10)
798 001174 000000          $REG5: .WORD 0      ;;CONTAINS (($REGAD)+12)
799 001176 000000          $TMP0: .WORD 0      ;;USER DEFINED
800 001200 000000          $TMP1: .WORD 0      ;;USER DEFINED
801 001202 000000          $TMP2: .WORD 0      ;;USER DEFINED
802 001204 000000          $TMP3: .WORD 0      ;;USER DEFINED
803 001206 000000          $TMP4: .WORD 0      ;;USER DEFINED
804 001210 000000          $TMP5: .WORD 0      ;;USER DEFINED
805 001212 000000          $TIMES: 0      ;;MAX. NUMBER OF ITERATIONS
806 001214 000000          $ESCAPE: 0      ;;ESCAPE ON ERROR ADDRESS
807 001216 177607 000377          $BELL: .ASCIZ <207><377><377>  ;;CODE FOR BELL
808 001222 077          $QUES: .ASCII ?/      ;;QUESTION MARK
809 001223 015          $CRLF: .ASCII <15>      ;;CARRIAGE RETURN
    
```

```
810 001224 000012 $LF: .ASCIZ <12> ;;LINE FEED
811 ;:*****
812 .SBTTL APT MAILBOX-ETABLE
813 ;:*****
814 .EVEN
815 $MAIL: .WORD AMMSGTY ;;APT MAILBOX
816 001226 000000 $MSGTY: .WORD AMMSGTY ;;MESSAGE TYPE CODE
817 001226 000000 $FATAL: .WORD AFATAL ;;FATAL ERROR NUMBER
818 001230 000000 $TESTN: .WORD ATESTN ;;TEST NUMBER
819 001232 000000 $PASS: .WORD APASS ;;PASS COUNT
820 001234 000000 $DEVCT: .WORD ADEVCT ;;DEVICE COUNT
821 001236 000000 $UNIT: .WORD AUNIT ;;I/O UNIT NUMBER
822 001240 000000 $MSGAD: .WORD AMSGAD ;;MESSAGE ADDRESS
823 001242 000000 $MSGLG: .WORD AMSGLG ;;MESSAGE LENGTH
824 001244 000000 $ETABLE: .WORD AETABLE ;;APT ENVIRONMENT TABLE
825 001246 000 $ENV: .BYTE AENV ;;ENVIRONMENT BYTE
826 001246 000 $ENVM: .BYTE AENVM ;;ENVIRONMENT MODE BITS
827 001247 000 $SWREG: .WORD ASWREG ;;APT SWITCH REGISTER
828 001250 000000 $USWR: .WORD AUSWR ;;USER SWITCHES
829 001252 000000 $CPUOP: .WORD ACPUOP ;;CPU TYPE,OPTIONS
830 001254 000000 ;*
831 ;* BITS 15-11-CPU TYPE
832 ;* 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
833 ;* 11/70=06,PDO=07,Q=10
834 ;* BIT 10=REAL TIME CLOCK
835 ;* BIT 9=FLOATING POINT PROCESSOR
836 ;* BIT 8=MEMORY MANAGEMENT
837 001256 000 $MAMS1: .BYTE AMAMS1 ;;HIGH ADDRESS,M.S. BYTE
838 001257 000 $MTYP1: .BYTE AMTYP1 ;;MEM. TYPE,BLK#1
839 ;* MEM.TYPE BYTE -- (HIGH BYTE)
840 ;* 900 NSEC CORE=001
841 ;* 300 NSEC BIPOLAR=002
842 ;* 500 NSEC MOS=003
843 001260 000000 $MADR1: .WORD AMADR1 ;;HIGH ADDRESS,BLK#1
844 ;* MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
845 001262 000 $MAMS2: .BYTE AMAMS2 ;;HIGH ADDRESS,M.S. BYTE
846 001263 000 $MTYP2: .BYTE AMTYP2 ;;MEM. TYPE,BLK#2
847 001264 000000 $MADR2: .WORD AMADR2 ;;MEM.LAST ADDRESS,BLK#2
848 001266 000 $MAMS3: .BYTE AMAMS3 ;;HIGH ADDRESS,M.S.BYTE
849 001267 000 $MTYP3: .BYTE AMTYP3 ;;MEM. TYPE,BLK#3
850 001270 000000 $MADR3: .WORD AMADR3 ;;MEM.LAST ADDRESS,BLK#3
851 001272 000 $MAMS4: .BYTE AMAMS4 ;;HIGH ADDRESS,M.S.BYTE
852 001273 000 $MTYP4: .BYTE AMTYP4 ;;MEM. TYPE,BLK#4
853 001274 000000 $MADR4: .WORD AMADR4 ;;MEM.LAST ADDRESS,BLK#4
854 001276 000000 $VECT1: .WORD AVECT1 ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
855 001300 000000 $VECT2: .WORD AVECT2 ;;INTERRUPT VECTOR#2BUS PRIORITY#2
856 001302 000000 $BASE: .WORD ABASE ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
857 001304 000000 $DEV: .WORD ADEV ;;DEVICE MAP
858 001306 000000 $CDW1: .WORD ACDW1 ;;CONTROLLER DESCRIPTION WORD#1
859 001310 000000 $CDW2: .WORD ACDW2 ;;CONTROLLER DESCRIPTION WORD#2
860 001312 000000 $DDW0: .WORD ADDW0 ;;DEVICE DESCRIPTOR WORD#0
861 001314 000000 $DDW1: .WORD ADDW1 ;;DEVICE DESCRIPTOR WORD#1
862 001316 000000 $DDW2: .WORD ADDW2 ;;DEVICE DESCRIPTOR WORD#2
863 001320 000000 $DDW3: .WORD ADDW3 ;;DEVICE DESCRIPTOR WORD#3
864 001322 000000 $DDW4: .WORD ADDW4 ;;DEVICE DESCRIPTOR WORD#4
865 001324 000000 $DDW5: .WORD ADDW5 ;;DEVICE DESCRIPTOR WORD#5
```

```
866 001326 000000 $DDW6: .WORD ADDW6 ;;DEVICE DESCRIPTOR WORD#6
867 001330 000000 $DDW7: .WORD ADDW7 ;;DEVICE DESCRIPTOR WORD#7
868 001332 000000 $DDW8: .WORD ADDW8 ;;DEVICE DESCRIPTOR WORD#8
869 001334 000000 $DDW9: .WORD ADDW9 ;;DEVICE DESCRIPTOR WORD#9
870 001336 000000 $DDW10: .WORD ADDW10 ;;DEVICE DESCRIPTOR WORD#10
871 001340 000000 $DDW11: .WORD ADDW11 ;;DEVICE DESCRIPTOR WORD#11
872 001342 000000 $DDW12: .WORD ADDW12 ;;DEVICE DESCRIPTOR WORD#12
873 001344 000000 $DDW13: .WORD ADDW13 ;;DEVICE DESCRIPTOR WORD#13
874 001346 000000 $DDW14: .WORD ADDW14 ;;DEVICE DESCRIPTOR WORD#14
875 001350 000000 $DDW15: .WORD ADDW15 ;;DEVICE DESCRIPTOR WORD#15
876
877
878 001352 $ETEND:
879
880
881 001352 000000 TESTNO: .WORD 0 ;HOLDS TEST NUMBER FOR TYPEOUTS
882 001354 000000 WASR6: .WORD 0 ;USED TO STORE THE STACK POINTER AFTER A TRAP
883 001356 000000 TRAPP: .WORD 0 ;USED TO STORE THE PC OF A TRAP OR ABORT
884 001360 000000 TRAPP: .WORD 0 ;USED TO STORE THE PS OF A TRAP OR ABORT
885 001362 000000 ;+USED TO STORE THE CORRECT SR0
886 001364 000000 CDORSR0: .WORD 0 ;+USED TO STORE THE CORRECT SR2
887 001366 000000 WASSR0: .WORD 0 ;USED TO STORE CONTENTS OF SR0
888 001370 000000 WASSR2: .WORD 0 ;USED TO STORE CONTENTS OF SR2
889 001372 000000 TBITPS: .WORD 0 ;SAVES THE PSW THAT MAY HAVE ITS T-BIT ON
890 001374 000000 ANDADR: .WORD 0 ;HOLDS RESULT OF ADDRESSES BEING AND-ED
891 001376 000000 ORADR: .WORD 0 ;HOLDS RESULT OF ADDRESSES BEING OR-ED
892 001400 000000 TONUM: .WORD 0 ;HOLDS NUMBER OF TIME-OUTS
893 001402 000000 VIRT1: .WORD 0 ;HOLDS VIRTUAL ADDRESS TO BE CONVERTED
894 001404 000000 VIRT2: .WORD 0 ; " "
895 001406 000000 PBALO: .WORD 0 ;HOLDS BITS <15:00> OF PHYSICAL ADDRESS
896 001410 000000 PBAHI: .WORD 0 ;HOLDS BITS <17:16> OF PHYSICAL ADDRESS
```

```

.SBTTL ERROR POINTER TABLE

;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

;* EM ;;POINTS TO THE ERROR MESSAGE
;* DH ;;POINTS TO THE DATA HEADER
;* DT ;;POINTS TO THE DATA
;* DF ;;POINTS TO THE DATA FORMAT

$ERRTB:

;*ITEM 1
      EM1 ;UNEXPECTED CPU TRAP TO LOC. 004
      DH1 ;OLD PC OLD PSW R6 WAS TESTNO ERRORPC
      DT1 ;TRAPPC, TRAPPS, WASR6, TESTNO, $ERRPC, 0
      DF1 ;0,0,0,0

;*ITEM 2
      EM2 ;UNEXPECTED MEM. MGMT. TRAP TO LOC. 250
      DH2 ;OLD PC OLD PSW R6 WAS SRO SR2 TESTNO ERRORPC
      DT2 ;TRAPPC, TRAPPS, WASR6, WASSR0, WASSR2, TESTNO, $ERRPC,
      DF2 ;0,0,0,0,0,0

;*ITEM 3
      EM3 ;PRIORITY BITS SET WRONG IN PSW
      DH3 ;WROTE READ TESTNO ERRORPC
      DT3 ;$REG0,$REG1,TESTNO,$ERRPC,0
      DF3 ;0,0,0,0

;*ITEM 4
      EM4 ;MODE BITS SET WRONG IN PSW
      DH3 ;WROTE READ TESTNO ERRORPC
      DT3 ;$REG0,$REG1,TESTNO,$ERRPC,0
      DF3 ;0,0,0,0

;*ITEM 5
      EM5 ;DUAL ADDRESSING BETWEEN HI&LO BYTES OF PSW
      DH3 ;WROTE READ TESTNO ERRORPC
      DT3 ;$REG0,$REG1,TESTNO,$ERRPC,0
      DF3 ;0,0,0,0

;*ITEM 6
      EM6 ;KERNEL R6 CHANGED BY WRITING USER R6
      DH3 ;WROTE READ TESTNO ERRORPC
      DT3 ;$REG0,$REG1,TESTNO,$ERRPC,0
      DF3 ;0,0,0,0

;*ITEM 7
      EM7 ;A MEMORY MGMT. REG. TIMED OUT
      DH7 ;ADDRESS TESTNO ERRORPC
      DT7 ;$REG0,TESTNO,$ERRPC,0
    
```

```

      DF7 ;0,0,0

;*ITEM 10
      EM10 ;SUMMARY OF MEM. MGMT. REG. TIMEOUTS
      DH10 ;REGISTER-ADDRS NUM. OF
      DT10 ;AND-ED OR-ED TIMOUTS TESTNO ERRORPC
      DF10 ;ANDADR,DRADR,TONUM,TESTNO,$ERRPC,0
      ;0,0,1,0,0

;*ITEM 11
      EM11 ;MEM. MGMT. REG. WOULD NOT CLEAR
      DH11 ;REGISTER READ READ-(BINARY)
      DT11 ;ADDRESS (OCTAL) 5432109876543210 TESTNO ERRORPC
      DF11 ;$REG0,$REG1,$REG1,TESTNO,$ERRPC,0
      ;0,0,2,0,0

;*ITEM 12
      EM12 ;MEM. MGMT. REG. BITS NOT SET CORRECTLY
      DH12 ;REGISTER WROTE READ READ
      DT12 ;ADDRESS (OCTAL) (OCTAL) (BINARY) TESTNO ERRORPC
      DF12 ;$REG0,$REG1,$REG2,$REG2,TESTNO,$ERRPC,0
      ;0,0,0,2,0,0

;*ITEM 13
      EM13 ;SRO EFFECTED BY WRITE TO PSW
      DH13 ;READ TESTNO ERRORPC
      DT13 ;$REG0,TESTNO,$ERRPC,0
      DF13 ;0,0,0

;*ITEM 14
      EM14 ;SR1 DID NOT READ ALL ZEROS
      DH13 ;READ TESTNO ERRORPC
      DT13 ;$REG0,TESTNO,$ERRPC,0
      DF13 ;0,0,0

;*ITEM 15
      EM15 ;DUAL ADDRESSING BETWEEN BYTES OF PAR OR PDR
      DH12 ;REGISTER WROTE READ READ
      DT12 ;ADDRESS (OCTAL) (OCTAL) (BINARY) TESTNO ERRORPC
      DF12 ;$REG0,$REG1,$REG2,$REG2,TESTNO,$ERRPC,0
      ;0,0,0,2,0,0

;*ITEM 16
      EM16 ;DUAL ADDRESSING BETWEEN PAR-PDR'S
      DH16 ;PAR-PDR PAR-PDR
      DT16 ;CLEARED EFFECTD EXPECTD RECEIVD TESTNO ERRORPC
      DF16 ;$REG0,$REG1,$REG5,$REG2,TESTNO,$ERRPC,0
      ;0,0,0,0,0,0

;*ITEM 17
      EM17 ;PHYS. ADDR. FORMED READ WRONG IN MAINT. MODE
      DH17 ;PHYSICAL VIRTUAL
      DT17 ;ADDRESS ADDRESS KIPAR4 TESTNO ERRORPC
      DF17 ;PBA0,VIRT1,$REG4,TESTNO,$ERRPC,0
      ;3,0,0,0,0
    
```



```

1009                ;*ITEM 20
1010 001602 042626          EM20          ;PHYS. ADDR. FORMED READ WRONG IN RELOCATE MODE
1011 001604 046550          DH20          ;PHYSICL PAR 4 PAR 5
1012                ;ADDRESS VBA VBA PAR 4 PAR 5 PSW TESTNO
1013 001606 051406          DT20          ;PBALD,VIRT1,VIRT2,$REG4,$REG5,$TMP0,TESTNO,$ERRPC,0
1014 001610 052173          DF20          ;3,0,0,0,0,0,0,0
1015
1016                ;*ITEM 21
1017 001612 042700          EM21          ;W-BIT DID NOT GET SET IN PDR
1018 001614 046776          DH21          ;PDR VIRTUAL
1019                ;TESTED ADDRESS TESTNO ERRORPC
1020 001616 051430          DT21          ;$REG5,$REG3,TESTNO,$ERRPC,0
1021 001620 052203          DF21          ;0,0,0,0
1022
1023                ;*ITEM 22
1024 001622 042735          EM22          ;W-BIT SET IN MORE THAN ONE PDR
1025 001624 047056          DH22          ;PDR IN PDR VIRTUAL
1026                ;ERROR TESTED ADDRESS TESTNO ERRORPC
1027 001626 051442          DT22          ;$REG0,$REG5,$REG3,TESTNO,$ERRPC,0
1028 001630 052207          DF22          ;0,0,0,0,0
1029
1030                ;*ITEM 23
1031 001632 042774          EM23          ;W-BIT NOT CLEARED BY WRITING TO PDR
1032 001634 047155          DH23          ;PDR TESTNO ERRORPC
1033 001636 051456          DT23          ;$REG5,TESTNO,$ERRPC,0
1034 001640 052214          DF23          ;0,0,0
1035
1036                ;*ITEM 24
1037 001642 043040          EM24          ;WRITING SRO SET W-BIT IN KIPDR7
1038 001644 047205          DH24          ;PDR WAS EXPECTD TESTNO ERRORPC
1039 001646 051466          DT24          ;$REG2,$REG1,TESTNO,$ERRPC,0
1040 001650 052217          DF24          ;0,0,0,0
1041
1042                ;*ITEM 25
1043 001652 043100          EM25          ;W-BIT GOT SET DURING ODD ADDR. ABORT
1044 001654 047205          DH24          ;PDR WAS EXPECTD TESTNO ERRORPC
1045 001656 051466          DT24          ;$REG2,$REG1,TESTNO,$ERRPC,0
1046 001660 052217          DF24          ;0,0,0,0
1047
1048                ;*ITEM 26
1049 001662 043145          EM26          ;MEMORY MGMT. ACCESS ABORT DID NOT OCCUR
1050 001664 047245          DH26          ;PDR 4 PSW TESTNO ERRORPC
1051 001666 051500          DT26          ;$REG2,$TMP0,TESTNO,$ERRPC,0
1052 001670 052217          DF24          ;0,0,0,0
1053
1054                ;*ITEM 27
1055 001672 043215          EM27          ;ACCESS ERROR DID NOT ABORT INSTRUCTION
1056 001674 047245          DH26          ;PDR 4 PSW TESTNO ERRORPC
1057 001676 051500          DT26          ;$REG2,$TMP0,TESTNO,$ERRPC,0
1058 001700 052217          DF24          ;0,0,0,0
1059
1060                ;*ITEM 30
1061 001702 043264          EM30          ;SRO DID NOT REPORT ACCESS ERROR CORRECTLY
1062 001704 047305          DH30          ;SRO WAS EXPECTD PDR 4 PSW TESTNO ERRORPC
1063 001706 051512          DT30          ;WASSR0,$REG3,$REG2,$TMP0,TESTNO,$ERRPC,0
1064 001710 052223          DF30          ;0,0,0,0,0,0
    
```

```

1065
1066                ;*ITEM 31
1067 001712 043336          EM31          ;SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR.
1068 001714 047365          DH31          ;SR2 WAS EXPECTD PDR 4 PSW TESTNO ERRORPC
1069 001716 051530          DT31          ;WASSR2,$REG4,$REG2,$TMP0,TESTNO,$ERRPC,0
1070 001720 052223          DF30          ;0,0,0,0,0,0
1071
1072                ;*ITEM 32
1073 001722 043403          EM32          ;PAGE LGTH. ABORT OCCURRED WHEN IT SHOULDN'T HAVE
1074 001724 047445          DH32          ;V.B.A. KIPDR4 SRO WAS SR2 WAS TESTNO ERRORPC
1075 001726 051546          DT32          ;$REG0,$REG4,WASSR0,WASSR2,TESTNO,$ERRPC,0
1076 001730 052223          DF30          ;0,0,0,0,0,0
1077
1078                ;*ITEM 33
1079 001732 043464          EM33          ;PAGE LGTH. ABORT DID NOT OCCUR WHEN IT SHOULD HAVE
1080 001734 047525          DH33          ;V.B.A. KIPDR4 TESTNO ERRORPC
1081 001736 051564          DT33          ;$REG0,$REG4,TESTNO,$ERRPC,0
1082 001740 052217          DF24          ;0,0,0,0
1083
1084                ;*ITEM 34
1085 001742 043547          EM34          ;SRO DID NOT REPORT PAGE LGTH. ABORT CORRECTLY
1086 001744 047565          DH34          ;V.B.A. KIPDR4 SRO WAS EXPECTD TESTNO ERRORPC
1087 001746 051576          DT34          ;$REG0,$REG4,WASSR0,$REG2,TESTNO,$ERRPC,0
1088 001750 052223          DF30          ;0,0,0,0,0,0
1089
1090                ;*ITEM 35
1091 001752 043336          EM31          ;SR2 DID NOT LOCKUP CORRECT VIRUAL ADDR.
1092 001754 047645          DH35          ;V.B.A. KIPDR4 SR2 WAS EXPECTD TESTNO ERRORPC
1093 001756 051614          DT35          ;$REG0,$REG4,WASSR2,$REG3,TESTNO,$ERRPC,0
1094 001760 052223          DF30          ;0,0,0,0,0,0
1095
1096                ;*ITEM 36
1097 001762 043336          EM31          ;SR2 DID NOT LOCKUP CORRECT VIRUAL ADDR.
1098 001764 047725          DH36          ;SR2 WAS EXPECTD TESTNO ERRORPC
1099 001766 051632          DT36          ;WASSR2,$REG1,TESTNO,$ERRPC,0
1100 001770 052217          DF24          ;0,0,0,0
1101
1102                ;*ITEM 37
1103 001772 043625          EM37          ;SRO OR SR2 CHANGED BY A SECOND ABORT
1104 001774 047765          DH37          ;FIRST ABORT SECOND ABORT
1105                ;SRO WAS SR2 WAS SRO WAS SR2 WAS TESTNO ERRORPC
1106 001776 051644          DT37          ;$TMP0,$TMP2,WASSR0,WASSR2,TESTNO,$ERRPC,0
1107 002000 052223          DF30          ;0,0,0,0,0,0
1108
1109                ;*ITEM 40
1110 002002 043672          EM40          ;SRO OR SR2 WAS NOT "RESET" BY A RESET
1111 002004 050102          DH40          ;SRO WAS SR2 WAS TESTNO ERRORPC
1112 002006 051662          DT40          ;WASSR0,WASSR2,TESTNO,$ERRPC,0
1113 002010 052217          DF24          ;0,0,0,0
1114
1115                ;*ITEM 41
1116 002012 043741          EM41          ;SR2 NOT TRACKING CORRECTLY
1117 002014 047725          DH36          ;SR2 WAS EXPECTD TESTNO ERRORPC
1118 002016 051632          DT36          ;WASSR2,$REG1,TESTNO,$ERRPC,0
1119 002020 052217          DF24          ;0,0,0,0
1120
1121                ;*ITEM 42
    
```

```

1121 002022 043774          EM42          ;DID NOT TRAP THRU KERNEL SPACE
1122 002024 050142          DH42          ;PSW WAS R6 WAS TESTNO ERRORPC
1123 002026 051674          DT42          ;$REG1,$REG2,TESTNO,$ERRPC,0
1124 002030 052217          DF24          ;0,0,0,0
1125
1126          ;*ITEM 43
1127 002032 044033          EM43          ;KT ERROR SERVICED ON ODD ADDR. ERROR
1128 002034 047155          DH23          ;PDR TESTNO ERRORPC
1129 002036 051456          DT23          ;$REG5,TESTNO,$ERRPC,0
1130 002040 052214          DF23          ;0,0,0
1131
1132          ;*ITEM 44
1133 002042 044100          EM44          ;SRO OR SR2 CHANGED BY ODD ADDR. ERROR
1134 002044 050202          DH44          ;EXPECTED RECEIVED
1135
1136 002046 051706          DT44          ;SRO SR2 SRO WAS SR2 WAS TESTNO ERRORPC
1137 002050 052223          DF30          ;$REG0,$REG1,WASSRO,WASSR2,TESTNO,$ERRPC,0
1138
1139          ;*ITEM 45
1140 002052 044146          EM45          ;ERROR DURING "DOUBLE ERROR" (KT & ODD ADDR.)
1141 002054 050314          DH45          ;EXPECTED:
1142
1143          ;PSW PC SRO SR2
1144          ;170017 (3$+4) 020147 (3$)
1145          ;RECEIVED
1146 002056 051724          DT45          ;PSW PC SRO SR2 TESTNO ERRORPC
1147 002060 052223          DF30          ;$REG1,$REG3,WASSRO,WASSR2,TESTNO,$ERRPC,0
1148
1149          ;*ITEM 46
1150 002062 044223          EM46          ;MFPI INSTRUCTION PUSHED WRONG DATA
1151 002064 050511          DH46          ;DATA DATA
1152          ;EXPECTD RECEIVD TESTNO ERRORPC
1153 002066 051742          DT46          ;$REG0,$REG1,TESTNO,$ERRPC,0
1154 002070 052231          DF46          ;0,0,0,0
1155
1156          ;*ITEM 47
1157 002072 044266          EM47          ;MTPI INSTRUCTION LOADED WRONG DATA
1158 002074 050511          DH46          ;DATA DATA
1159          ;EXPECTD RECEIVD TESTNO ERRORPC
1160 002076 051742          DT46          ;$REG0,$REG1,TESTNO,$ERRPC,0
1161 002100 052231          DF46          ;0,0,0,0
1162
1163          ;*ITEM 50
1164 002102 044331          EM50          ;STACK NOT PUSHED BY MFPI-MTPI
1165 002104 050566          DH50          ;TESTNO ERRORPC
1166 002106 051754          DT50          ;TESTNO,$ERRPC,0
1167 002110 052235          DF50          ;0,0
1168
1169          ;*ITEM 51
1170 002112 044367          EM51          ;KERNEL PAGE ACCESSED INSTEAD OF USER: MFPI-MTPI
1171 002114 050606          DH51          ;SRO WAS SR2 WAS TESTNO ERRORPC
1172 002116 051762          DT51          ;WASSRO,WASSR2,TESTNO,$ERRPC,0
1173 002120 052737          DF51          ;0,0,0,0
1174
1175          ;*ITEM 52
1176 002122 044445          EM52          ;WRONG PDR'S REFERENCED WHILE IN RELOCATE MODE
    
```

```

1177 002124 050346          DH52          ;PHYSICL PAR 4
1178
1179 002126 051774          DT52          ;ADDRESS V.B.A. PAR 4 SRO WAS SR2 WAS PSW TESTNO
1180 002130 052243          DF52          ;PBALO,VIRT1,$REG4,WASSRO,WASSR2,$TMP0,TESTNO,$ERRPC,0
1181
1182          ;*ITEM 53
1182 002132 044523          EM53          ;MFPD INSTRUCTION PUSHED WRONG DATA
1183 002134 050511          DH46          ;DATA DATA
1184          ;EXPECTD RECEIVD TESTNO ERRORPC
1185 002136 051742          DT46          ;$REG0,$REG1,TESTNO,$ERRPC,0
1186 002140 052231          DF46          ;0,0,0,0
1187
1188          ;*ITEM 54
1189 002142 044566          EM54          ;STACK NOT PUSHED BY MFDP-MTPD
1190 002144 050566          DH50          ;TESTNO ERRORPC
1191 002146 051754          DT50          ;TESTNO,$ERRPC,0
1192 002150 052235          DF50          ;0,0
1193
1194          ;*ITEM 55
1195 002152 044624          EM55          ;PAR OR PDR WAS CHANGED BY A RESET
1196 002154 046150          DH11          ;REGISTR READ READ-(BINARY)
1197          ;ADDRESS (OCTAL) 5432109876543210 TESTNO ERRORPC
1198 002156 051312          DT11          ;$REG0,$REG1,$REG1,TESTNO,$ERRPC,0
1199 002160 052142          DF11          ;0,0,2,0,0
1200
1201          ;*ITEM 56
1202 002162 044662          EM56          ;ILLEGAL MODE 01 NOT ABORTED
1203 002164 050566          DH50          ;TESTNO ERRORPC
1204 002166 051754          DT50          ;TESTNO,$ERRPC,0
1205 002170 052235          DF50          ;0,0
1206
1207          ;*ITEM 57
1208 002172 044716          EM57          ;SRO DID NOT REPORT ILLEGAL MODE 01 CORRECTLY
1209 002174 050764          DH57          ;SRO WAS EXPECTD TESTNO ERRORPC
1210 002176 052016          DT57          ;WASSRO,$REG1,TESTNO,$ERRPC,0
1211 002200 052253          DF57          ;0,0,0,0
1212
1213          ;*ITEM 60
1214 002202 044773          EM60          ;PSW CHANGED BY AN RTI IN USER MODE
1215 002204 051024          DH60          ;PSW WAS EXPECTD TESTNO ERRORPC
1216 002206 052030          DT60          ;$REG1,$REG2,TESTNO,$ERRPC,0
1217 002210 052257          DF60          ;0,0,0,0
1218
1219          ;*ITEM 61
1220 002212 045036          EM61          ;MAINT MODE (SRO<8>) NOT DISABLED BY A RESET
1221 002214 050566          JH60          ;TESTNO ERRORPC
1222 002216 051754          DT50          ;TESTNO,$ERRPC,0
1223 002220 052235          DF50          ;0,0
1224
1225          ;*ITEM 62
1226 002222 045114          EM62          ;DATA INCORRECT AFTER A MAINT. MODE WRITE
1227 002224 045760          DH3          ;WROTE READ TESTNO ERRORPC
1228 002226 051254          DT3          ;$REG0,$REG1,TESTNO,$ERRPC,0
1229 002230 052126          DF3          ;0,0,0,0
1230
1231          ;*ITEM 63
1232 002232 045165          EM63          ;SOURCE RELOCATED IN MAINT. MODE
    
```

```

1233 002234 045670          DH2          ;OLD PC OLD PSW R6 WAS SRO SR2 TESTNO ERRORPC
1234 002236 051234          DT2          ;TRAPPC,TRAPPS,WASR6,WASSR0,WASSR2,TESTNO,$ERRPC,0
1235 002240 052117          DF2          ;0,0,0,0,0,0,0
1236
1237          ;*ITEM 64
1238 002242 043336          EM31          ;SR2 DIDNOT LOCKUP CORRECT VIRTUAL ADDR.
1239 002244 047725          DH36          ;SR2 WAS EXPECTD TESTNO ERRORPC
1240 002246 052042          DT64          ;WASSR0,$REG4,TESTNO,$ERRPC,0
1241 002250 052217          DF24          ;0,0,0,0
1242
1243          ;*ITEM 65
1244 002252 045225          EM64          ;+NON RESIDENT ABORT DID NOT OCCUR
1245 002254 051064          DH61          ;+SRO SR2 TESTNO ERRORPC
1246 002256 052354          DT65          ;+WASSR0,WASSR2,TESTNO,$ERRPC
1247 002260 052263          DF61          ;+0,0,0,0
1248
1249          ;*ITEM 66
1250 002262 045266          EM65          ;+ERROR FLAG FOR KR ABORT DID NOT SET
1251 002264 051064          DH61          ;+SRO SR2 TESTNO ERRORPC
1252 002266 052054          DT65          ;+WASSR0,WASSR2,TESTNO,$ERRPC
1253 002270 052263          DF61          ;+0,0,0,0
1254
1255          ;*ITEM67
1256
1257 002272 045351          EM66          ;+SR2 DID NOT FREEZE THE VIRTUAL ADRS OF ABORT
1258 002274 051064          DH61          ;+SRO SR2 TESTNO ERRORPC
1259 002276 052054          DT65          ;+WASSR0,WASSR2,TESTNO,$ERRPC
1260 002300 052263          DF61          ;+0,0,0,0
1261
1262          ;*ITEM 70
1263
1264 002302 045441          EM67          ;+2ND NON RESIDENT ABORT DID NOT OCCUR
1265 002304 051064          DH61          ;+SRO SR2 TESTNO ERRORPC
1266 002306 052054          DT65          ;+WASSR0,WASSR2,TESTNO,$ERRPC
1267 002310 052263          DF61          ;+0,0,0,0
1268
1269          ;*ITEM 71
1270
1271 002312 045506          EM70          ;+2ND NR ABORT CHANGED SR2
1272 002314 051122          DH62          ;+SR2 EXPT SR2 RECD TESTNO ERRORPC
1273 002316 052066          DT66          ;+CORSR0,WASSR2,TESTNO,$ERRPC
1274 002320 052263          DF61          ;+0,0,0,0
1275
1276          ;*ITEM 72
1277 002322 045562          EM71          ;+SRO WAS NOT CLEARED BY INIT
1278 002324 051161          DH63          ;+SRO EXPT SRO RECD TESTNO ERRORPC
1279 002326 052100          DT67          ;+CORSR0,WASSR0,TESTNO,ERRORPC
1280 002330 052263          DF61          ;+0,0,0,0
    
```

```

1281          .SBTTL ***** TRAP HANDLING ROUTINES *****
1282
1283          .SBTTL CPU TRAP HANDLER ROUTINE
1284          ;*****
1285          ;*
1286          ;* THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS THRU
1287          ;* "ERRVEC" (LOC. 004). IF THIS SUBROUTINE IS ENTERED BY A
1288          ;* SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A HALT IS
1289          ;* EXECUTED.
1290          ;*
1291          ;*****
1292 002332 005227          TIMERR: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME THRU
1293 002334 177777          TIMFLG: .WORD -1 ;NEGATIVE ONE FOR "HAVE ENTERED" FLAG
1294 002336 001401          BEQ 1$ ;BRANCH IF FIRST TIME IN
1295 002340 000000          HALT ;STOP! - I'VE ENTERED THIS ROUTINE
1296          ;A SECOND TIME BEFORE I FINISHED
1297          ;REPORTING THE FIRST ERROR. THE
1298          ;SECOND ENTRY ADDRESS SHOULD BE ON
1299          ;THE KERNEL STACK.
1300 002342 012637 001356          1$: MOV (KSP)+,TRAPPC ;SAVE PC+2 AT TIME OF ABORT
1301 002346 012637 001360          MOV (KSP)+,TRAPPS ;SAVE PS AT TIME OF ABORT
1302 002352 010637 001354          MOV KSP,WASR6 ;SAVE STACK POINTER VALUE
1303 002356 104001          ERROR 1 ;UNEXPECTED TRAP OR ABORT TO LOC. 4
1304 002360 012737 177777 002334          MOV #-1,TIMFLG ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
1305 002366 013746 001360          MOV TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
1306 002372 013746 001356          MOV TRAPPC,-(KSP)
1307 002376 000006          RTT ;RETURN FROM INTERRUPT OR ABORT
1308
1309          .SBTTL MEMORY MANAGEMENT TRAP HANDLER ROUTINE
1310          ;*****
1311          ;*
1312          ;* THIS SUBROUTINE WILL HANDLE ALL UNEXPECTED MEMORY MANAGEMENT
1313          ;* TRAPS AND ABORTS THRU "MMVEC" (LOC. 250). IF THIS SUBROUTINE IS
1314          ;* ENTERED BY A SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A
1315          ;* HALT IS EXECUTED.
1316          ;*
1317          ;*****
1318          ;*****
1319 002400 005227          MGMERR: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME THRU
1320 002402 177777          MGMFLG: .WORD -1 ;NEGATIVE ONE FOR "HAVE ENTERED" FLAG
1321 002404 001401          BEQ 1$ ;BRANCH IF FIRST TIME IN
1322 002406 000000          HALT ;STOP! - I'VE ENTERED THIS ROUTINE
1323          ;A SECOND TIME BEFORE I FINISHED
1324          ;REPORTING THE FIRST ERROR. THE
1325          ;SECOND ENTRY ADDRESS SHOULD BE ON
1326          ;THE KERNEL STACK.
1327 002410 012637 001356          1$: MOV (KSP)+,TRAPPC ;SAVE PC+2 AT TIME OF ABORT
1328 002414 012637 001360          MOV (KSP)+,TRAPPS ;SAVE PS AT TIME OF ABORT
1329 002420 010637 001354          MOV KSP,WASR6 ;SAVE STACK POINTER VALUE
1330 002424 013737 177572 001366          MOV SRO,WASSR0 ;SAVE CONTENTS OF KT STATUS REG. 0
1331 002432 013737 177576 001370          MOV SR2,WASSR2 ;SAVE CONTENTS OF KT STATUS REG. 2
1332 002440 042737 160000 177572          BIC #160000,SRO ;CLEAR ERROR BITS IN STATUS REG 0
1333 002446 104002          ERROR 2 ;UNEXPECTED TRAP OR ABORT TO LOC. 250
1334 002450 012737 177777 002402          MOV #-1,MGMFLG ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
1335 002456 013746 001360          MOV TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
1336 002462 013746 001356          MOV TRAPPC,-(KSP)
    
```

1337 002466 000006 RTT ;RETURN FROM INTERRUPT OR ABORT.
 1338

```

1339 .SBTTL
1340 .SBTTL ***** STARTING POINT OF TEST *****
1341 .SBTTL ***** STARTING ADDRESS OF 200 *****
1342 .:=20000
1343
1344 020000
1345
1346
1347 020000 012706 001100
1348 020004 005026
1349 020006 022706 001140
1350 020012 001374
1351 020014 012706 001100
1352
1353 020020 012737 035040 000020
1354 020026 012737 000340 000022
1355 020034 012737 035320 000030
1356 020042 012737 000340 000032
1357 020050 012737 041302 000034
1358 020056 012737 000340 000036
1359 020064 012737 041370 000024
1360 020072 012737 000340 000026
1361 020100 013737 034606 034600
1362 020106 005037 001212
1363 020112 005037 001214
1364 020116 112737 000001 001115
1365
1366
1367 020124 012737 035024 000014
1368 020132 012737 000340 000016
1369 020140 012737 000002 035024
1370 020146 012737 020174 000010
1371 020154 005046
1372 020156 012746 020164
1373 020162 000006
1374 020164 012737 000006 035024 64$:
1375 020172 004002
1376 020174 062706 000010 65$:
1377 020200 012737 000012 000010 66$:
1378 020206 005037 035032
1379 020212 012737 020212 001106
1380 020220 012737 020220 001110
1381
1382
1383 020226 013746 000004
1384 020232 012737 020266 000004
1385 020240 012737 177570 001140
1386 020246 012737 177570 001142
1387 020254 022777 177777 160658
1388 020262 001012
1389
1390 020264 000403
1391 020266 012716 020274 67$:
1392 020272 000002
1393 020274 012737 000176 001140 68$:
1394 020302 012737 000174 001142
    
```

```

1395 020310 012637 000004 69$: MOV (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
1396
1397 020314 005037 001234 CLR $PASS ;;CLEAR PASS COUNT
1398 020320 132737 000200 001247 BITB #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
1399 020326 001403 BEQ 70$ ;;YES,USE NON-APT SWITCH
1400 020330 012737 001250 001140 MOV #5$WREG,$SWR ;;NO,USE APT SWITCH REGISTER
1401 020336
1402
1403 .SBTTL TYPE PROGRAM NAME
;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
1404 020336 005227 177777 INC #-1 ;;FIRST TIME?
1405 020342 001051 BNE 71$ ;;BRANCH IF NO
1406 020344 022737 034760 000042 CMP #5$ENDAD,@#42 ;;ACT-11?
1407 020352 001445 BEQ 71$ ;;BRANCH IF YES
1408 020354 104401 020422 TYPE ,72$ ;;TYPE ASCIZ STRING
1409 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
1410 020360 005737 000042 TST @#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
1411 020364 001012 BNE 73$ ;;BRANCH IF YES
1412 020366 123727 001246 000001 CMPB $ENV,#1 ;;ARE WE RUNNING UNDER APT?
1413 020374 001406 BEQ 73$ ;;BRANCH IF YES
1414 020376 023727 001140 000176 CMP $SWR,#5$WREG ;;SOFTWARE SWITCH REG SELECTED?
1415 020404 001005 BNE 74$ ;;BRANCH IF NO
1416 020406 104407 GTSWR 74$ ;;GET SOFT-SWR SETTINGS
1417 020410 000403 BR 74$
1418 020412 112737 000001 001134 73$: MOVB #1,$AUTOB ;;SET AUTO-MODE INDICATOR
1419 020420 74$: BR 71$ ;;GET OVER THE ASCIZ
1420 020420 000422 ;;72$: .ASCIZ <CRLF>#CFKTHB0 11/34 MEMORY MGNT. DIAG.#<CRLF>
1421 71$:
1422 020466
1423
1424 020466 LOOP: MOV #STACK,KSP ;INITIALIZE THE STACK POINTER
1425 020466 012706 001100 MOV #TIMERR,ERRVEC ;LOAD CPU SERVICE ROUTINE INTO TRAP VECTOR
1426 020472 012737 002332 MOV #340,ERRVEC+2 ;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
1427 020500 012737 000340 000006 MOV #MGMERR,MMVEC ;LOAD MEMORY MANAGEMENT ROUTINE INTO VECTOR
1428 020506 012737 002400 000250 MOV #340,MMVEC+2 ;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
1429 020514 012737 000340 000252 MOV #-1,R0 ;PUT -1 INTO R0 TO INITIALIZE FLAGS
1430 020522 012700 177777 MOV R0,TIMFLG ;INITIALIZE CPU ERROR FLAG
1431 020526 010037 002334 MOV R0,MGMFLG ;INITIALIZE MEMORY MANAGEMENT ERROR FLAG
1432 020532 010037 002402 MOV #340,TBITPS ;INITIALIZE LOG THAT HOLDS T-BIT PSW
1433 020536 012737 000340 001372 CLR SRO ;BE SURE MEM. MGNT IS OFF TO START WITH
1434 020544 005037 177572
1435

```

```

1436
1437
1438
1439 *****
; *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
1445 020550 000004 1$: MOV #2,$LPERR ;SET LOOP ON ERROR POINTER TO 2$
1446 020552 012737 020562 001110 CLR R0 ;INITIALIZE R0 WITH PRIORITY=0 DATA
1447 020560 005000 2$: CLR R1 ;PREPARE R1 TO ACCEPT DATA READ
1448 020562 005001 MTPS R0 ;WRITE PRIORITY BITS IN THE PSW
1449 020564 106400 MFPS R1 ;READ BACK THE LOW BYTE OF PSW
1450 020566 106701 BIC #177437,R1 ;MASK OFF EVERYTHING EXCEPT PRIORITY BITS
1451 020570 042701 177437 CMP R0,R1 ;WAS CORRECT PRIORITY SET IN THE PSW?
1452 020574 020001 BEQ 3$ ;BRANCH IF YES
1453 020576 001401 ERROR 3 ;PRIORITY BITS SET WRONG IN PSW
1454 020600 104003 ;FOR TIGHTER SCOPE LOOP
1455 ;REPLACE ERROR CALL WITH
1456 ;"BR 2$" = 000770
1457 3$: ADD #40,R0 ;CHANGE DATA TO NEXT PRIORITY
1458 020602 062700 000040 CMP #400,R0 ;HAVE PRIORITIES 0-7 ALL BEEN CHECKED?
1459 020606 022700 000400 BNE 2$ ;BRANCH IF NO
1460 020612 001363 MOV #1,$LPERR ;RESET LOOP ON ERROR POINTER TO 1$
1461 020614 012737 020552 001110
1462
; * *****
; *TEST 2 PSW MODE BIT TEST
; *
; * THIS TEST READS AND WRITES THE PROCESSOR STATUS WORD <15:12> "MODE BITS"
; * TO FURTHER CHECK THE BASIC CPU DATA PATHS
; *
; * *****
TST2: SCOPE
1469 020622 000004 1$: MOV #2,$LPERR ;SET LOOP ON ERROR POINTER TO 2$
1470 020624 012737 020634 001110 CLR R0 ;INITIALIZE R0 WITH MODE BITS = 0000
1471 020632 005000 2$: CLR PSW ;INITIALIZE PSW
1472 020634 005037 177776 BIS R0,PSW ;BIT SET THE PSW MODE BITS WITH R0
1473 020640 050037 177776 MOV PSW,R1 ;READ BACK THE CONTENTS OF THE PSW
1474 020644 013701 177776 BIC #007777,R1 ;MASK OFF EVERYTHING EXCEPT THE MODE BITS
1475 020650 042701 007777 CMP R0,R1 ;WERE THE MODE BITS SET CORRECTLY?
1476 020654 020001 BEQ 3$ ;BRANCH IF YES
1477 020656 001403 CLR PSW ;CLEAR PSW FOR ERROR REPORT
1478 020660 005037 177776 ERROR -1 ;MODE BITS SET WRONG IN PSW
1479 020664 104004 ;FOR TIGHTER SCOPE LOOP
1480 ;REPLACE ERROR CALL WITH
1481 ;"BR 2$" = 000763
1482 3$: ADD #10000,R0 ;CHANGE MODE BIT DATA
1483 020666 062700 010000 BNE 2$ ;BRANCH IF STILL MORE COMBINATIONS
1484 020672 001360 MOV #1,$LPERR ;RESET LOOP ON ERROR POINTER TO 1$
1485 020674 012737 020624 001110 CLR PSW ;RESET PSW BEFORE LEAVING
1486 020702 005037 177776
1487
; * *****
; *TEST 3 BYTE ADDRESSING TEST FOR PSW
; *
; * THIS TEST WRITES THE HIGH AND LOW BYTES OF THE PROCESSOR STATUS WORD
1488
1489
1490
1491

```

```

1492 ;* AND READS THEM BACK TO BE SURE THEY CAN BE WRITTEN INDEPENDENTLY.
1493 ;* THIS CHECKS THE PSW PORTION OF THE ADDRESS DETECTION LOGIC.
1494 ;*
1495 ;*****
1496 020706 000004 TST3: SCOPE
1497 020710 012737 020716 001110 1$: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
1498 020716 005037 177776 2$: CLR PSW ;CLEAR THE PSW
1499 020722 012700 000360 MOV #360, R0 ;PUT THE HIGH BYTE DATA INTO R0
1500 020726 110037 177777 RO, PSW+1 ;WRITE THE HIGH BYTE OF THE PSW
1501 020732 013701 177776 MOV PSW, R1 ;READ BACK THE ENTIRE PSW
1502 020736 042701 007437 BIC #007437, R1 ;MASK OFF THE T & CC BITS
1503 020742 000300 SWAB R0 ;GET DATA WRITTEN IN HIGH BYTE OF R0
1504 020744 020001 CMP RO, R1 ;WAS THE PSW WRITTEN TO CORRECTLY
1505 020746 001:03 BEQ 3$ ;BRANCH IF YES
1506 020750 005037 177776 CLR PSW ;CLEAR PSW FOR ERROR REPORT
1507 020754 104005 ERROR 5 ;LOW BYTE EFFECTED BY WRITE TO HIGH BYTE OF PSW
1508 ;*
1509 ;* FOR TIGHTER SCOPE LOOP
1510 ;* REPLACE ERROR CALL WITH
1511 ;* "BR 2$" = 000760
1511 020756 012737 020764 001110 3$: MOV #4$, $LPERR ;SET LOOP ON ERROR POINTER TO 4$
1512 020764 005037 177776 4$: CLR PSW ;CLEAR THE PSW
1513 020770 012700 000340 MOV #340, R0 ;PUT THE LOW BYTE DATA INTO R0
1514 020774 110037 177776 RO, PSW ;WRITE THE LOW BYTE OF THE PSW
1515 021000 013701 177776 MOV PSW, R1 ;READ BACK THE ENTIRE PSW
1516 021004 042701 007437 BIC #007437, R1 ;MASK OFF THE T&CC BITS
1517 021010 020001 CMP RO, R1 ;WAS PSW WRITTEN TO CORRECTLY
1518 021012 001403 BEQ 5$ ;BRANCH IF YES
1519 021014 005037 177776 CLR PSW ;CLEAR PSW FOR ERROR REPORT
1520 021020 104005 ERROR 5 ;HIGH BYTE EFFECTED BY WRITE TO LOW BYTE OF PSW
1521 ;*
1522 ;* FOR TIGHTER SCOPE LOOP
1523 ;* REPLACE ERROR CALL WITH
1524 021022 012737 020710 001110 5$: MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
1525 ;*
1526 ;*****
1527 ;* TEST 4 TEST AND SETUP OF STACK POINTERS
1528 ;*
1529 ;* THIS TEST SETS THE USER AND KERNEL STACK POINTERS FOR THE
1530 ;* REST OF THE PROGRAM AND MAKES SURE THEY ARE INDEPENDENT OF
1531 ;* EACH OTHER. KERNEL R6 IS SET TO 1100, USER R6 IS SET TO 700, THEN
1532 ;* KERNEL R6 IS READ TO BE SURE ITS STILL 1100.
1533 ;*
1534 ;*****
1535 021030 000004 TST4: SCOPE
1536 021032 005037 177776 CLR PSW ;GO TO KERNEL MODE
1537 021036 012706 001100 MOV #KERSTK, KSP ;SET KERNEL STACK POINTER TO 1100
1538 021042 012737 140000 177776 MOV #140000, PSW ;GO TO USER MODE
1539 021050 012706 000700 MOV #USESTK, USP ;SET USER STACK POINTER TO 700
1540 021054 005037 177776 CLR PSW ;BACK TO KERNEL MODE
1541 021060 022706 001100 CMP #KERSTK, KSP ;IS KERNEL R6 STILL 1100?
1542 021064 001404 BEQ TST5 ;BRANCH IF KERNEL R6 IS OKAY
1543 021066 012700 001100 MOV #KERSTK, R0 ;SAVE DATA WRITTEN FOR ERROR REPORT
1544 021072 010601 MOV KSP, R1 ;SAVE DATA READ AFTER USER R6 WAS WRITTEN
1545 021074 104006 ERROR 6 ;KERNEL R6 CHANGED BY WRITING USER R6
1546 ;*
1547 ;* FOR TIGHTER SCOPE LOOP
1548 ;* REPLACE ERROR CALL WITH
    
```

```

1548 ;*****
1549 ;*
1550 ;* THE NEXT FIVE (5) TESTS WILL TRY TO ADDRESS ALL OF THE
1551 ;* MEMORY MANAGEMENT REGISTERS (SRO, SR1, SR2, KERNEL & USER PAR/PDR'S).
1552 ;* EVERY TIME A REGISTER TIMES OUT ITS ADDRESS WILL BE REPORTED.
1553 ;* AT THE END OF EACH TEST A SUMMARY OF THE ADDRESSES THAT TIMED
1554 ;* OUT DURING THAT TEST IS GIVEN. THE RESULTS OF "AND-ING" AND "OR-ING"
1555 ;* THEIR ADDRESSES IS GIVEN TO SHOW WHICH ADDRESS LINES MAY BE
1556 ;* STUCK AT 0 OR 1. THE PAR/PDR ADDRESS AND KT MUX'S ARE THE
1557 ;* THINGS BEING CHECKED.
1558 ;*
1559 ;*****
1560 ;*****
1561 ;*
1562 ;*****
1563 ;*****
1564 ;*****
1565 ;* TEST 5 SRO, SR1, SR2 TIMEOUT TEST
1566 ;*
1567 ;* THIS TEST ADDRESSES THE MEMORY MANAGEMENT STATUS REGISTERS
1568 ;* 0, 1, AND 2. STATUS REG. 1 IS NOT USED BUT SHOULD STILL
1569 ;* RESPOND TO ITS UNIBUS ADDRESS. DATA WILL BE WRITTEN OR READ
1570 ;* FROM THESE REGISTERS IN LATER TESTS, THIS TEST JUST CHECK
1571 ;* FOR A RESPONSE.
1572 ;*
1573 ;*****
1574 021076 000004 TST5: SCOPE
1575 ;*
1576 021100 012737 021142 001110 1$: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
1577 021106 012737 021200 000004 MOV #5$, 0#4 ;SET TIMEOUT VECTOR TO 5$
1578 021114 012700 177572 MOV #SRO, R0 ;LOAD R0 WITH ADDRESS OF FIRST REG.
1579 021120 012701 000003 MOV #3, R1 ;LOAD R1 WITH THE LOOP COUNT
1580 021124 012737 177777 001374 MOV #-1, ANDADR ;INITIALIZE "AND" OF ADDRS. LOC.
1581 021132 005037 001376 CLR ORADR ;INITIALIZE "OR" OF ADDRS. LOC.
1582 021136 005037 001400 CLR TONUM ;INITIALIZE "TIMEOUTS" COUNTER
1583 021142 005710 2$: TST (R0) ;TRY ADDRESSING A STATUS REGISTER
1584 ;*
1585 021144 062700 000002 3$: ADD #2, R0 ;PUT NEXT ADDRESS IN R0
1586 021150 077104 SOB R1, 2$ ;LOOP BACK TO 2$ UNTIL ALL TESTED
1587 021152 012737 021100 001110 MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
1588 021160 005737 001400 TST TONUM ;DID ANY OF THE STATUS REG.S TIMEOUT?
1589 021164 001401 BEQ 4$ ;BRANCH IF NO
1590 021166 104010 ERROR 10 ;SUMMARY OF STATUS REG. TIMEOUTS
1591 021170 012737 002332 000004 4$: MOV #TIMERR, 0#4 ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1592 021176 000414 BR T5B ;GO TO NEXT TEST
1593 ;*
1594 021200 062706 000004 5$: ADD #4, KSP ;CLEAN UP THE STACK
1595 021204 104007 ERROR 7 ;ONE OF THE STATUS REGS. TIMED OUT
1596 ;*
1597 ;* FOR TIGHTER SCOPE LOOP
1598 ;* REPLACE ERROR CALL WITH
1599 ;* "BR 2$" = 000756
1599 021206 010002 MOV R0, R2 ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
1600 021210 050737 001376 BIS R2, ORADR ;"OR" IT WITH OTHER ADDRS. THAT TIMED OUT
1601 021214 005102 COM R2 ;"AND" IT WITH OTHER ADDRS. THAT TIMED OUT
1602 021216 040237 001374 BIC R2, ANDADR
1603 021222 005237 001400 INC TONUM ;INCREMENT THE TIMEOUT COUNTER
    
```

```

1604 021226 000746 BR 3$ ;BRANCH BACK TO TEST THE NEXT ADDR.
1605
1606 ;*****
1607 ;*TEST 6 KERNEL PAR'S TIMEOUT TEST
1608 ;*
1609 ;* THIS TEST ADDRESSES THE EIGHT (8) KERNEL PAGE ADDRESS
1610 ;* REGISTERS (KIPAR0-KIPAR7) AND CHECKS THAT SOMETHING
1611 ;* RESPONDS TO THEIR ADDRESSES.
1612 ;*
1613 ;*****
1614 021230 000004 TST6: SCOPE
1615
1616 021232 012737 021274 001110 1$: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
1617 021240 012737 021332 000004 MOV #5$, @#4 ;SET TIMEOUT VECTOR TO 5$
1618 021246 012700 172340 MOV #KIPAR0, R0 ;LOAD R0 WITH ADDRESS OF FIRST REG.
1619 021252 012701 000010 MOV #10, R1 ;LOAD R1 WITH LOOP COUNT (8)
1620 021256 012737 177777 001374 MOV #-1, ANDADR ;INITIALIZE "AND" OF ADDR. LOC
1621 021264 005037 001376 CLR DRADR ;INITIALIZE "OR" OF ADDR. LOC
1622 021270 005037 001400 CLR TONUM ;INITIALIZE "TIMEOUTS" COUNTER
1623 021274 005710 2$: TST (R0) ;TRY ADDRESSING A KIPAR
1624 ;IF IT TIMES OUT, WILL GO TO 5$
1625 021276 062700 000002 3$: ADD #2, R0 ;PUT NEXT KIPAR ADDRESS IN R0
1626 021302 077104 SOB R1, 2$ ;LOOP BACK TO 2$ UNTIL ALL TESTED
1627 021304 012737 021232 001110 MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
1628 021312 005737 001400 TST TONUM ;DID ANY OF THE KIPARS TIME OUT?
1629 021316 001401 BEQ 4$ ;BRANCH IF NO
1630 021320 104010 ERROR 10 ;SUMMARY OF KIPAR TIMEOUTS
1631 021322 012737 002332 000004 4$: MOV #TIMERR, @#4 ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1632 021330 000414 BR TST7 ;GO TO NEXT TEST
1633
1634 021332 062706 000004 5$: ADD #4, KSP ;CLEAN UP THE STACK
1635 021336 104007 ERROR 7 ;ONE OF THE KIPARS TIMED OUT
1636 ;FOR TIGHTER SCOPE LOOP
1637 ;REPLACE ERROR CALL WITH
1638 ;"BR 2$" = 000756
1639 021340 010002 MOV R0, R2 ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
1640 021342 050237 001376 BIS R2, DRADR ;"OR" IT WITH OTHER ADDRS. THAT TIMED OUT
1641 021346 005102 COM R2 ;"AND" IT WITH OTHER ADDRS. THAT TIMED OUT
1642 021350 040237 001374 BIC R2, ANDADR
1643 021354 005237 001400 INC TONUM ;INCREMENT THE TIMEOUT COUNTER
1644 021360 000746 BR 3$ ;BRANCH BACK TO TEST THE NEXT KIPAR
1645
1646 ;*****
1647 ;*TEST 7 KERNEL PDR'S TIMEOUT TEST
1648 ;*
1649 ;* THIS TEST ADDRESSES THE EIGHT (8) KERNEL PAGE DESCRIPTOR
1650 ;* REGISTERS (KIPDR0-KIPDR7) AND CHECKS THAT SOMETHING
1651 ;* RESPONDS TO THEIR ADDRESSES.
1652 ;*
1653 ;*****
1654 021362 000004 TST7: SCOPE
1655
1656 021364 012737 021426 001110 1$: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
1657 021372 012737 021464 000004 MOV #5$, @#4 ;SET TIMEOUT VECTOR TO 5$
1658 021400 012700 172300 MOV #KIPDR0, R0 ;LOAD R0 WITH ADDRESS OF FIRST REG.
1659 021404 012701 000010 MOV #10, R1 ;LOAD R1 WITH LOOP COUNT (8)
    
```

```

1660 021410 012737 177777 001374 MOV #-1, ANDADR ;INITIALIZE "AND" OF ADDR. LOC
1661 021416 005037 001376 CLR DRADR ;INITIALIZE "OR" OF ADDR. LOC
1662 021422 005037 001400 CLR TONUM ;INITIALIZE "TIMEOUTS" COUNTER
1663 021426 005710 2$: TST (R0) ;TRY ADDRESSING A KIPDR
1664 ;IF IT TIMES OUT, WILL GO TO 5$
1665 021430 062700 000002 3$: ADD #2, R0 ;PUT NEXT KIPDR ADDRESS IN R0
1666 021434 077104 SOB R1, 2$ ;LOOP BACK TO 2$ UNTIL ALL TESTED
1667 021436 012737 021364 001110 MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
1668 021444 005737 001400 TST TONUM ;DID ANY OF THE KIPDRS TIME OUT?
1669 021450 001401 BEQ 4$ ;BRANCH IF NO
1670 021452 104010 ERROR 10 ;SUMMARY OF KIPDR TIMEOUTS
1671 021454 012737 002332 000004 4$: MOV #TIMERR, @#4 ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1672 021462 000414 BR TST10 ;GO TO NEXT TEST
1673
1674 021464 062706 000004 5$: ADD #4, KSP ;CLEAN UP THE STACK
1675 021470 104007 ERROR 7 ;ONE OF THE KIPDRS TIMED OUT
1676 ;FOR TIGHTER SCOPE LOOP
1677 ;REPLACE ERROR CALL WITH
1678 ;"BR 2$" = 000756
1679 021472 010002 MOV R0, R2 ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
1680 021474 050237 001376 BIS R2, DRADR ;"OR" IT WITH OTHER ADDRS. THAT TIMED OUT
1681 021500 005102 COM R2 ;"AND" IT WITH OTHER ADDRS. THAT TIMED OUT
1682 021502 040237 001374 BIC R2, ANDADR
1683 021506 005237 001400 INC TONUM ;INCREMENT THE TIMEOUT COUNTER
1684 021512 000746 BR 3$ ;BRANCH BACK TO TEST THE NEXT KIPDR
1685
1686 ;*****
1687 ;*TEST 10 USER PAR'S TIMEOUT TEST
1688 ;*
1689 ;* THIS TEST ADDRESSES THE EIGHT (8) USER PAGE ADDRESS
1690 ;* REGISTERS (UIPAR0-UIPAR7) AND CHECKS THAT SOMETHING
1691 ;* RESPONDS TO THEIR ADDRESSES.
1692 ;*
1693 ;*****
1694 021514 000004 TST10: SCOPE
1695
1696 021516 012737 021560 001110 1$: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
1697 021524 012737 021616 000004 MOV #5$, @#4 ;SET TIMEOUT VECTOR TO 5$
1698 021532 012700 177640 MOV #UIPAR0, R0 ;LOAD R0 WITH ADDRESS OF FIRST REG.
1699 021536 012701 000010 MOV #10, R1 ;LOAD R1 WITH LOOP COUNT (8)
1700 021542 012737 177777 001374 MOV #-1, ANDADR ;INITIALIZE "AND" OF ADDR. LOC
1701 021550 005037 001376 CLR DRADR ;INITIALIZE "OR" OF ADDR. LOC
1702 021554 005037 001400 CLR TONUM ;INITIALIZE "TIMEOUTS" COUNTER
1703 021560 005710 2$: TST (R0) ;TRY ADDRESSING A UIPAR
1704 ;IF IT TIMES OUT, WILL GO TO 5$
1705 021562 062700 000002 3$: ADD #2, R0 ;PUT NEXT UIPAR ADDRESS IN R0
1706 021566 077104 SOB R1, 2$ ;LOOP BACK TO 2$ UNTIL ALL TESTED
1707 021570 012737 021516 001110 MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
1708 021576 005737 001400 TST TONUM ;DID ANY OF THE UIPARS TIME OUT?
1709 021602 001401 BEQ 4$ ;BRANCH IF NO
1710 021604 104010 ERROR 10 ;SUMMARY OF UIPAR TIMEOUTS
1711 021606 012737 002332 000004 4$: MOV #TIMERR, @#4 ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1712 021614 000414 BR TST11 ;GO TO NEXT TEST
1713
1714 021616 062706 000004 5$: ADD #4, KSP ;CLEAN UP THE STACK
1715 021622 104007 ERROR 7 ;ONE OF THE UIPARS TIMED OUT
    
```

```

1716 ;FOR TIGHTER SCOPE LOOP
1717 ;REPLACE ERROR CALL WITH
1718 ;"BR 2$" = 000756
1719 021624 010002 MOV R0,R2 ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
1720 021626 050237 001376 BIS R2,ORADR ;"OR" IT WITH OTHER ADDR. THAT TIMED OUT
1721 021632 005102 COM R2 ;"AND" IT WITH OTHER ADDR. THAT TIMED OUT
1722 021634 040237 001374 BIC R2,ANDADR
1723 021640 005237 001400 INC TONUM ;INCREMENT THE TIMEOUT COUNTER
1724 021644 000746 BR 3$ ;BRANCH BACK TO TEST THE NEXT UIPAR
1725
1726 ;*****
1727 ;*TEST 11 USER PDR'S TIMEOUT TEST
1728 ;*
1729 ;* THIS TEST ADDRESSES THE EIGHT (8) USER PAGE DESCRIPTOR
1730 ;* REGISTERS (UIPDR0-UIPDR7) AND CHECKS THAT SOMETHING
1731 ;* RESPONDS TO THEIR ADDRESSES.
1732 ;*
1733 ;*****
1734 021646 000004 TST11: SCOPE
1735
1736 021650 012737 021712 001110 1$: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
1737 021656 012737 021750 000004 MOV #5$, #4 ;SET TIMEOUT VECTOR TO 5$
1738 021664 012700 177600 MOV #UIPDR0, R0 ;LOAD R0 WITH ADDRESS OF FIRST REG.
1739 021670 012701 000010 MOV #10, R1 ;LOAD R1 WITH LOOP COUNT (8)
1740 021674 012737 177777 001374 MOV #-1, ANDADR ;INITIALIZE "AND" OF ADDR. LOC
1741 021702 005037 001376 CLR ORADR ;INITIALIZE "OR" OF ADDR. LOC.
1742 021706 005037 001400 CLR TONUM ;INITIALIZE "TIMEOUTS" COUNTER
1743 021712 005710 2$: TST (R0) ;TRY ADDRESSING A UIPDR
1744 ;IF IT TIMES OUT, WILL GO TO 5$
1745 021714 062700 000002 3$: ADD #2, R0 ;PUT NEXT UIPDR ADDRESS IN R0
1746 021720 077104 SOB R1, 2$ ;LOOP BACK TO 2$ UNTIL ALL TESTED
1747 021722 012737 021650 001110 MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
1748 021730 005737 001400 TST TONUM ;DID ANY OF THE UIPDRS TIME OUT?
1749 021734 001401 BEQ 4$ ;BRANCH IF NO
1750 021736 104010 ERROR 10 ;SUMMARY OF UIPDR TIMEOUTS
1751 021740 012737 002332 000004 4$: MOV #TIMERR, #4 ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1752 021746 000414 BR TST12 ;GO TO NEXT TEST
1753
1754 021750 062706 000004 5$: ADD #4, KSP ;CLEAN UP THE STACK
1755 021754 104007 ERROR 7 ;ONE OF THE UIPDRS TIMED OUT
1756 ;FOR TIGHTER SCOPE LOOP
1757 ;REPLACE ERROR CALL WITH
1758 ;"BR 2$" = 000756
1759 021756 010002 MOV R0,R2 ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
1760 021760 050237 001376 BIS R2,ORADR ;"OR" IT WITH OTHER ADDR. THAT TIMED OUT
1761 021764 005102 COM R2 ;"AND" IT WITH OTHER ADDR. THAT TIMED OUT
1762 021766 040237 001374 BIC R2,ANDADR
1763 021772 005237 001400 INC TONUM ;INCREMENT THE TIMEOUT COUNTER
1764 021776 000746 BR 3$ ;BRANCH BACK TO TEST THE NEXT UIPDR
1765
1766 ;*****
1767 ;*TEST 12 SRO(15:13) BIT TEST & SR2 TEST
1768 ;*
1769 ;* THIS TEST CHECKS BITS <15:13> OF STATUS REGISTER 0 TO SEE
1770 ;* THAT EACH CAN BE SET AND CLEARED AND THAT A "RESET" WILL
1771 ;* CLEAR ALL OF THEM. A TEST OF THESE THREE ERROR BITS CHECKS
    
```

```

1772 ;* PART OF SRO, THE SRO MUX AND THE KTMUX. THE REST OF THE
1773 ;* BITS IN SRO WILL BE CHECKED LATER.
1774 ;* ALSO CHECK THAT SR2 IS TRACKING WITH MEM. MGMT.
1775 ;* OFF BUT LOCKS UP WHEN ANY OF SRO ERROR BITS SET.
1776 ;*
1777 ;*****
1778 022000 000004 TST12: SCOPE
1779
1780 022002 012700 177572 1$: MOV #SRO, R0 ;LOAD ADDRESS OF SRO INTO R0
1781 022006 012710 160000 MOV #160000, (R0) ;SET BITS <15:13> IN SRO (ERROR BITS)
1782 022012 000005 RESET ;ISSUE AND "INIT" SIGNAL
1783 022014 011001 MOV (R0), R1 ;READ SRO INTO R1 TO SEE IF CLEAR
1784 022016 001404 BEQ 2$ ;BRANCH IF SRO<15:13> CLEARED BY "INIT"
1785 022020 104011 ERROR 11 ;SRO<15:13> NOT CLEARED BY A "RESET"
1786 ;FOR TIGHTER SCOPE LOOP
1787 ;REPLACE ERROR CALL WITH
1788 ;"BR 1$" = 000770
1789 022022 012737 022030 001110 MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
1790 022030 013737 177576 001370 2$: MOV SR2, WASSR2 ;READ CONTENTS OF SR2
1791 022036 012701 022030 MOV #2$, R1 ;LOAD EXPECTED CONTENTS INTO R1
1792 022042 020137 001370 CMP R1, WASSR2 ;IS SR2 TRACKING?
1793 022046 001401 BEQ 3$ ;BRANCH IF YES
1794 022050 104041 ERROR 41 ;SR2 NOT "TRACKING" VIRTUAL ADDRESSES
1795 ;FOR TIGHTER SCOPE LOOP
1796 ;REPLACE ERROR CALL WITH
1797 ;"BR 2$" = 000767
1798 022052 012737 022070 001110 3$: MOV #4$, $LPERR ;SET LOOP ON ERROR POINTER TO 4$
1799 022060 012701 100000 MOV #BIT15, R1 ;PUT DATA TO BE WRITTEN IN R1
1800 022064 012703 000003 MOV #3, R3 ;SETUP R3 AS A LOOP COUNTER
1801 022070 005010 4$: CLR (R0) ;CLEAR SRO
1802 022072 050110 5$: BIS R1, (R0) ;SET ONE OF THE ERROR BITS IN SRO
1803 022074 011002 MOV (R0), R2 ;READ SRO INTO R2
1804 022076 020102 CMP R1, R2 ;DID RIGHT ERROR BIT GET SET?
1805 022100 001401 BEQ 6$ ;BRANCH IF YES
1806 022102 104012 ERROR 12 ;BITS WERE SET WRONG IN SRO
1807 ;FOR TIGHTER SCOPE LOOP
1808 ;REPLACE ERROR CALL WITH
1809 ;"BR 4$" = 000772
1810 022104 012704 022072 6$: MOV #5$, R4 ;LOAD EXPECTED CONTENTS OF SR2 IN R4
1811 022110 013737 177576 001370 MOV SR2, WASSR2 ;READ SR2
1812 022116 020437 001370 CMP R4, WASSR2 ;DID SR2 LOCK UP WHEN ERROR
1813 ;BIT SET IN SR1?
1814 022122 001401 BEQ 7$ ;BRANCH IF YES
1815 022124 104064 ERROR J4 ;SR2 DID NOT LOCK UP
1816 ;FOR TIGHTER SCOPE LOOP
1817 ;REPLACE ERROR CALL WITH
1818 ;"BR 4$" = 000761
1819 022126 006001 7$: ROR R1 ;CHANGE DATA TO CHECK NEXT ERROR BIT
1820 022130 077321 SOB R3, 4$ ;LOOP BACK UNTIL <15:13> ALL TESTED
1821 022132 005010 CLR (R0) ;CLEAR SRO BEFORE LEAVING
1822 022134 012737 022002 001110 MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
1823
1824 ;*****
1825 ;*TEST 13 SRO & PSW DUAL ADDRESSING TEST
1826 ;*
1827 ;* THIS TEST CHECKS MORE OF THE ADDRESS DETECTION LOGIC BY
    
```



```

1828 ;* VERIFYING THAT STATUS REGISTER 0 IS NOT EFFECTED BY WRITING
1829 ;* TO THE PSW AND THAT THE LOW BYTE OF STATUS REGISTER 0
1830 ;* IS NOT EFFECTED BY WRITING TO ITS HIGH BYTE. THIS IS TO
1831 ;* SEE IF ADJACENT OUTPUTS ARE SHORTED ON THE ADDRESS DET. LOGIC.
1832 ;*
1833 ;*
1834 022142 000004 TST13: SCOPE
1835 ;*
1836 022144 005037 177776 1$: CLR PSW ;CLEAR THE PSW
1837 022150 005037 177572 CLR SRO ;CLEAR STATUS REGISTER 0
1838 022154 106427 000340 MTPS #340 ;SET PRIORITY 7 IN LOW BYTE OF PSW
1839 022160 013700 177572 MOV SRO,R0 ;READ STATUS REGISTER 0
1840 022164 001401 BEQ 2$ ;BRANCH IF IT WAS STILL 0
1841 022166 104713 ERROR 13 ;SRO EFFECTED BY A WRITE TO THE PSW
1842 ;* ;FOR TIGHTER SCOPE LOOP
1843 ;* ;REPLACE ERROR CALL WITH
1844 ;* ;"BR 1$" = 000767
1845 022170 005037 177572 2$: CLR SRO ;BE SURE SRO IS 0 BEFORE LEAVING
1846 022174 005037 177776 CLR PSW ;BE SURE PSW IS 0 BEFORE LEAVING
1847 ;*
1848 ;*
1849 ;*TEST 14 TEST THAT SR1 READS ALL ZEROS
1850 ;*
1851 ;* THIS TEST'S CHECKS THAT EVEN THOUGH STATUS REGISTER 1
1852 ;* IS NON-EXISTENT, ITS ADDRESS SHOULD RESPOND WITH ALL ZEROS,
1853 ;* THEREBY CHECK ANOTHER PORTION OF THE ADDRESS DETECTION LOGIC.
1854 ;*
1855 ;*
1856 022200 000004 TST14: SCOPE
1857 022202 012700 177777 MOV #-1,R0 ;FILL R0 WITH ALL ONES
1858 022206 013700 177574 MOV SR1,R0 ;READ SR1 INTO R0
1859 022212 001401 BEQ TST15 ;BRANCH IF SR1 READS ALL ZEROS
1860 022214 104014 ERROR 14 ;SR1 DID NOT READ ALL ZEROS
1861 ;* ;FOR TIGHTER SCOPE LOOP
1862 ;* ;REPLACE ERROR CALL WITH
1863 ;* ;000772
1864 ;*
1865 ;*
1866 ;*TEST 15 BIT TEST OF KERNEL & USER PAR'S
1867 ;*
1868 ;* THE FOLLOWING TEST CHECKS THE BITS <11:60> OF BOTH THE KERNEL
1869 ;* AND USER PAGE ADDRESS REGISTERS. A "0" IS ROTATED THRU
1870 ;* THE REGISTERS FROM LEFT TO RIGHT. THIS CHECKS THE OPERATION
1871 ;* OF THE PAR/PDR ADDRESS MUX, THE KT MUX, AND THE PAR
1872 ;* OUTPUT DATA LINES.
1873 ;*
1874 ;*
1875 022216 000004 TST15: SCOPE
1876 ;*
1877 022220 012700 172340 1$: MOV #KIPAR0,R0 ;LOAD ADDRESS OF FIRST PAR IN R0
1878 022224 012703 000010 2$: MOV #10,R3 ;SETUP R3 TO COUNT 8 PAR'S
1879 022230 012737 022236 001110 3$: MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
1880 022236 005010 CLR (R0) ;CLEAR THE PAR
1881 022240 011001 MOV (R0),R1 ;READ THE PAR INTO R1
1882 022242 001401 BEQ 4$ ;BRANCH IF PAR CLEARED OK
1883 022244 104011 ERROR 11 ;PAR WOULD NOT CLEAR
    
```

```

1884 ;* ;FOR TIGHTER SCOPE LOOP
1885 ;* ;REPLACE ERROR CALL WITH
1886 ;* ;"BR 3$" = 000774
1887 022246 012704 167777 4$: MOV #167777,R4 ;LOAD "WALKING 0" TEST PATTERN IN R4
1888 022252 012737 022260 001110 5$: MOV #5$, $LPERR ;SET LOOP ON ERROR POINTER TO 5$
1889 022260 005010 CLR (R0) ;CLEAR THE PAR BEFORE LOADING DATA
1890 022262 010401 MOV R4,R1 ;LOAD DATA INTO R1
1891 022264 042701 170000 BIC #170000,R1 ;MASK UNUSED BITS OUT OF THE DATA
1892 022270 050110 BIS R1,(R0) ;BIT SET THE TEST PATTERN INTO THE PAR
1893 022272 011002 MOV (R0),R2 ;READ THE PAR INTO R2
1894 022274 020102 CMP R1,R2 ;DOES DATA WRITTEN=DATA READ?
1895 022276 001401 BEQ 6$ ;BRANCH IF YES
1896 022300 104012 ERROR 12 ;PAR BITS DID NOT SET CORRECTLY
1897 ;* ;FOR TIGHTER SCOPE LOOP
1898 ;* ;REPLACE ERROR CALL WITH
1899 ;* ;"BR 5$" = 000767
1900 022302 000261 6$: SEC ;SET THE C-BIT FOR THE ROTATE INST.
1901 022304 006004 ROR R4 ;ROTATE THE TEST PATTERN IN R4
1902 022306 103764 BCS 5$ ;BRANCH BACK IF MORE BITS TO TEST
1903 022310 062700 000002 ADD #2,R0 ;GET NEXT PAR ADDRESS IN R0
1904 022314 077330 SOB R3,$ ;BRANCH BACK UNTIL ALL PAR'S TESTED
1905 022316 022700 177660 CMP #UIPAR7+2,R0 ;HAVE USER PAR'S BEEN TESTED
1906 022322 103003 BHIS 7$ ;BRANCH IF YES
1907 022324 012700 177640 MOV #UIPAR0,R0 ;LOAD FIRST USER PAR ADDR. IN R0
1908 022330 000735 BR 2$ ;BRANCH BACK TO TEST USER PAR'S
1909 022332 012737 022220 001110 7$: MOV #1$, $LPERR ;RESET LOOP OR ERROR POINTER TO 1$
1910 ;* ;LEAVE TEST WITH BITS <11:1>=1 IN ALL PAR'S
    
```

```

1911 ;*****
1912 ;*TEST 16 BIT TEST OF KERNEL & USER PDR'S
1913 ;*
1914 ;* THE FOLLOWING TEST CHECKS THE BITS <14:8> AND <3:1> OF BOTH THE
1915 ;* KERNEL AND USER PAGE DESCRIPTOR REGISTERS. A "0" IS ROTATED
1916 ;* THRU THE REGISTERS FROM LEFT TO RIGHT. SOME TEST PATTERNS WILL
1917 ;* BE LOADED MORE THAN ONCE DUE TO THE UNUSED BITS IN THE PDR'S.
1918 ;* THE PAR/PDR ADDRESS MUX, KTMUX, AND PDR OUTPUT DATA LINES
1919 ;* ARE BEING CHECKED.
1920 ;*
1921 ;*****
1922 022340 000004 TST16: SCOPE
1923
1924 022342 012700 172300 1$: MOV #KIPDR0,R0 ;LOAD ADDRESS OF FIRST PDR IN R0
1925 022346 012703 000010 2$: MOV #10,R3 ;SETUP R3 TO COUNT 8 PDR'S
1926 022352 012737 022360 001110 3$: MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
1927 022360 005010 3$: CLR (R0) ;CLEAR THE PDR
1928 022362 011001 MOV (R0),R1 ;READ THE PDR INTO R1
1929 022364 001401 BEQ 4$ ;BRANCH IF PDR CLEARED OK
1930 022366 104011 ERROR 11 ;PDR WOULD NOT CLEAR
1931 ;FOR TIGHTER SCOPE LOOP
1932 ;REPLACE ERROR CALL WITH
1933 ;"BR 3$" = 000774
1934 022370 012704 077777 4$: MOV #077777,R4 ;LOAD "WALKING 0" TEST PATTERN IN R4
1935 022374 012737 022402 001110 5$: MOV #5$, $LPERR ;SET LOOP ON ERROR POINTER TO 5$
1936 022402 005010 CLR (R0) ;CLEAR THE PDR BEFORE LOADING DATA
1937 022404 010401 MOV R4,R1 ;LOAD DATA INTO R1
1938 022406 042701 100361 BIC #100361,R1 ;MASK UNUSED BITS OUT OF THE DATA
1939 022412 050110 BIS R1,(R0) ;BIT SET THE TEST PATTERN INTO THE PDR
1940 022414 011002 MOV (R0),R2 ;READ THE PDR INTO R2
1941 022416 020102 CMP R1,R2 ;DOES DATA WRITTEN=DATA READ?
1942 022420 001401 BEQ 6$ ;BRANCH IF YES
1943 022422 104012 ERROR 12 ;PDR BITS DID NOT SET CORRECTLY
1944 ;FOR TIGHTER SCOPE LOOP
1945 ;REPLACE ERROR CALL WITH
1946 ;"BR 5$" = 000767
1947 022424 000261 6$: SEC ;SET THE C-BIT FOR THE ROTATE INST.
1948 022426 006004 ROR R4 ;ROTATE THE TEST PATTERN IN R4
1949 022430 103764 BCS 5$ ;BRANCH BACK IF MORE BITS TO TEST
1950 022432 062700 000002 ADD #2,R0 ;GET NEXT PDR ADDRESS IN R0
1951 022436 077330 SOB R3,3$ ;BRANCH BACK UNTIL ALL PDR'S TESTED
1952 022440 022700 177620 CMP #UIPDR7+2,R0 ;HAVE USER PDR'S BEEN TESTED?
1953 022444 103003 BHIS 7$ ;BRANCH IF YES
1954 022446 012700 177600 MOV #UIPDR0,R0 ;LOAD FIRST USER PDR ADDR. IN R0
1955 022452 000735 BR 2$ ;BRANCH BACK TO TEST USER PDR'S
1956 022454 012737 022342 001110 7$: MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
1957 ;LEAVE TEST WITH ALL WRITEABLE BITS IN
1958 ;ALL PDR'S = 1
1959
1960 ;*****
1961 ;*TEST 17 TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PAR'S
1962 ;*
1963 ;* THE FOLLOWING TEST WRITES TO BOTH BYTES OF THE KERNEL & USER
1964 ;* PAR'S SEPERATELY TO SEE THAT WRITING TO ONE DOES NOT EFFECT
1965 ;* THE OTHER. THIS FURTHER VERIFIES THE OPERATION OF THE PAR/PDR
1966 ;* ADDR. MUX AND THE ADDR. DETECTION LOGIC.
    
```

```

1967 ;*
1968 ;*****
1969 022462 000004 TST17: SCOPE
    
```

```

1970
1971 022464 012700 172340      1$:  MOV    #KIPAR0,R0      ;LOAD ADDRESS OF FIRST PAR INTO R0
1972 022470 012737 022502    001110 2$:  MOV    #3$, $LPERR      ;SET LOOP ON ERROR POINTER TO 3$
1973 022476 012703 000010      3$:  MOV    #10,R3          ;LOAD LOOP COUNTER TO DO 8 PAR'S
1974 022502 012701 177777      3$:  MOV    #-1,R1          ;LOAD TEST PATTERN INTO R1
1975 022506 005010              CLR    (R0)             ;CLEAR THE PAR
    
```

```

1976 022510 110110              MOVB   R1,(R0)          ;WRITE 1'S TO THE LOW BYTE OF THE PAR
1977 022512 011002              MOV    (R0),R2         ;READ THE ENTIRE PAR INTO R2
1978 022514 042701 177400      BIC    #177400,R1      ;MASK HIGH BYTE & UNUSED BITS OUT OF THE DATA
1979 022520 020102              CMP    R1,R2           ;WAS ONLY THE LOW BYTE WRITTEN TO
1980 022522 001401              BEQ    4$              ;BRANCH IF YES
1981 022524 104015              ERROR  15             ;HIGH BYTE EFFECTED BY WRITING LOW BYTE IN PAR
1982
1983
1984
1985 022526 012737 022534    001110 4$:  MOV    #5$, $LPERR      ;SET LOOP ON ERROR POINTER TO 5$
1986 022534 005010              CLR    (R0)            ;CLEAR THE PAR
1987 022536 012701 177777      5$:  MOV    #-1,R1          ;LOAD TEST, PATTERN INTO R1
1988 022542 110160 000001      MOVB   R1,1(R0)        ;WRITE 1'S TO THE HIGH BYTE OF THE PAR
1989 022546 011002              MOV    (R0),R2         ;READ THE ENTIRE PAR INTO R2
1990 022550 042701 170377      BIC    #170377,R1      ;MASK LOW BYTE & UNUSED BITS OUT OF DATA
1991 022554 020102              CMP    R1,R2           ;WAS ONLY THE HIGH BYTE WRITTEN TO?
1992 022556 001401              BEQ    6$              ;BRANCH IF YES
1993 022560 104015              ERROR  15             ;LOW BYTE EFFECTED BY WRITING HIGH BYTE IN PAR
1994
1995
1996
1997 022562 062700 000002      6$:  ADD    #2,R0            ;PUT ADDRESS OF NEXT PAR IN R0
1998 022566 077333              SOB    R3,3$           ;BRANCH BACK UNTIL 8 PAR'S TESTED
1999 022570 022700 177660      CMP    #UIPAR7+2,R0    ;HAVE USER PAR'S BEEN TESTED
2000 022574 103003              BHIS   7$              ;BRANCH IF YES
2001 022576 012700 177640      MOV    #UIPAR0,R0      ;LOAD ADDRESS OF FIRST USER PAR IN R0
2002 022602 000732              BR     2$              ;BRANCH BACK TO TEST USER PAR'S
2003 022604 012737 022464    001110 7$:  MOV    #1$, $LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
2004
    
```

```

2005 ;:*****
2006 ;*TEST 20 TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PDR'S
2007 ;*
2008 ;* THE FOLLOWING TEST WRITES TO BOTH BYTES OF THE KERNEL & USER
2009 ;*
2010 ;*
2011 ;*
2012 ;* PDR'S SEPERATELY TO SEE THAT WRITING TO ONE DOES NOT EFFECT
2013 ;* THE OTHER. THIS FURTHER VERIFIES THE OPERATION OF THE PAR/PDR
2014 ;* ADDR. MUX AND THE ADDR. DETECTION LOGIC.
2015 ;*
2016 ;:*****
2017 022612 000004 TST20: SCOPE
2018 ;*
2019 022614 012700 172300 15: MOV #KIPDR0,R0 ;LOAD ADDRESS OF FIRST PDR INTO R0
2020 022620 012737 022632 001110 25: MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
2021 022626 012703 000010 MOV #10,R3 ;LOAD ADDRESS OF FIRST KERNEL PDR AND R0
2022 022632 012701 177777 35: MOV #-1,R1 ;SETUP STACK POINTER
2023 022636 005010 CLR (R0) ;SET ALL BITS IN ALL PAR'S IN PDR'S
2024 022640 110110 MOVB R1,(R0) ;CLEAR ONE OF THE KERNEL PDR'S
2025 022642 011002 MOV (R0),R2 ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
2026 022644 042701 177761 BIC #177761,R1 ;FORM ADDRESS OF NEXT KERNEL PDR TO CLEAR
2027 022650 020102 CMP R1,R2 ;LOOP TO 2$ UNTIL ALL KERNEL PDR'S CHECKED
2028 022652 001401 BEQ -$ ;SET LOOP ON ERROR POINTER TO 3$
2029 022654 104015 ERROR 15 ;SET LOOP ON ERROR POINTER TO 4$
2030 ;*
2031 ;* HIGH BYTE EFFECTED BY WRITING LOW BYTE IN PDR
2032 ;* FOR TIGHTER SCOPE LOOP
2033 ;* REPLACE ERROR CALL WITH
2034 ;* "BR 3$" = 000766
2035 022656 012737 022664 001110 45: MOV #5$, $LPERR ;SET LOOP ON ERROR POINTER TO 5$
2036 022664 005010 CLR (R0) ;CLEAR THE PDR
2037 022666 012701 177777 MOV #-1,R1 ;LOAD TEST PATTERN INTO R1
2038 022672 110160 000001 MOVB R1,1(R0) ;WRITE 1'S TO THE HIGH BYTE OF THE PDR
2039 022676 011002 MOV (R0),R2 ;READ THE ENTIRE PDR INTO R2
2040 022700 042701 100377 BIC #100377,R1 ;MASK LOW BYTE & UNUSED BITS OUT OF DATA
2041 022704 020102 CMP R1,R2 ;WAS ONLY THE HIGH BYTE WRITTEN TO?
2042 022706 001401 BEQ 6$ ;BRANCH IF YES
2043 022710 104015 ERROR 15 ;LOW BYTE EFFECTED BY WRITING HIGH BYTE IN PDR
2044 ;*
2045 ;* FOR TIGHTER SCOPE LOOP
2046 ;* REPLACE ERROR CALL WITH
2047 ;* "BR 5$" = 000765
2048 022712 062700 000002 65: ADD #2,R0 ;PUT ADDRESS OF NEXT PDR IN R0
2049 022716 077333 SOB R3,3$ ;BRANCH BACK UNTIL 8 PDR'S TESTED
2050 022720 022700 177620 CMP #UIPDR7+2,R0 ;HAVE USER PDR'S BEEN TESTED?
2051 022724 103003 BHIS 7$ ;BRANCH IF YES
2052 022726 012700 177600 MOV #UIPDR0,R0 ;LOAD ADDRESS OF FIRST USER PDR IN R0
2053 022732 000732 BR 2$ ;BRANCH BACK TO TEST USER PDR'S
2054 022734 012737 022614 001110 75: MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
    
```

```

2053 ;:*****
2054 ;*TEST 2: PAR-PDR DUAL ADDRESSING TEST
2055 ;*
2056 ;* THE FOLLOWING TEST SETS ALL OF THE WRITEABLE BITS TO 1
2057 ;* IN THE SIXTEEN (16) PAR'S AND PDR'S USING THE "SETREG"
2058 ;* SUBROUTINE AND THEN CLEARS JUST ONE OF THEM. THE "CMPREG"
2059 ;* SUBROUTINE IS USED TO READ ALL OF THE PAR'S AND PDR'S TO SEE
2060 ;* THAT ONLY ONE REGISTER WAS CLEARED IN RESPONSE TO THAT ONE
2061 ;* PAR OR PDR ADDRESS. THE "CMPREG" SUBROUTINE REPORTS THE
2062 ;* ADDRESS OF ANY REGISTER WHOSE BITS DID NOT REMAIN SET WHEN
2063 ;* ANOTHER REGISTER WAS CLEARED.
2064 ;*
2065 ;*
2066 ;* THE PAR AND PDR CHIPS, PAR/PDR ADDR. MUX, AND ADDR. DETECTION
2067 ;* LOGIC ARE CHECKED.
2068 ;*
2069 ;*
2070 022742 000004 TST21: SCOPE
2071 ;*
2072 022744 012737 022762 001110 15: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER 2$
2073 022752 012703 000010 MOV #10,R3 ;LOAD LOOP COUNTER WITH AN 8
2074 022756 012700 172300 MOV #KIPDR0,R0 ;LOAD ADDRESS OF FIRST KERNEL PDR AND R0
2075 022762 012706 001100 25: MOV #KERSTK,KSP ;SETUP STACK POINTER
2076 022766 004737 036070 JSR PC,SETREG ;SET ALL BITS IN ALL PAR'S IN PDR'S
2077 022772 005010 CLR (R0) ;CLEAR ONE OF THE KERNEL PDR'S
2078 022774 004737 036162 JSR PC,CMPREG ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
2079 023000 062700 000002 ADD #2,R0 ;FORM ADDRESS OF NEXT KERNEL PDR TO CLEAR
2080 023004 077312 SOB R3,2$ ;LOOP TO 2$ UNTIL ALL KERNEL PDR'S CHECKED
2081 023006 012737 023024 001110 MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
2082 023014 012703 000010 MOV #10,R3 ;LOAD LOOP COUNTER WITH AN 8
2083 023020 012700 172340 MOV #KIPAR0,R0 ;LOAD ADDRESS OF FIRST KERNEL PAR IN R0
2084 023024 012706 001100 35: MOV #KERSTK,KSP ;SETUP STACK POINTER
2085 023030 004737 036070 JSR PC,SETREG ;SET ALL BITS IN ALL PAR'S AND PDR'S
2086 023034 005010 CLR (R0) ;CLEAR ONE OF THE KERNEL PAR'S
2087 023036 004737 036162 JSR PC,CMPREG ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
2088 023042 062700 000002 ADD #2,R0 ;FORM ADDRESS OF NEXT KERNEL PAR TO CLEAR
2089 023046 077312 SOB R3,3$ ;LOOP TO 3$ UNTIL ALL KERNEL PAR'S CHECKED
2090 023050 012737 023066 001110 MOV #4$, $LPERR ;SET LOOP ON ERROR POINTER TO 4$
2091 023056 012703 000010 MOV #10,R3 ;LOAD LOOP COUNTER WITH AN 8
2092 023062 012700 177600 MOV #UIPDR0,R0 ;LOAD ADDRESS OF FIRST USER PDR IN R0
2093 023066 012706 001100 45: MOV #KERSTK,KSP ;SETUP STACK POINTER
2094 023072 004737 036070 JSR PC,SETREG ;SET ALL BITS IN ALL PAR'S AND PDR'S
2095 023076 005010 CLR (R0) ;CLEAR ONE OF THE USER PDR'S
2096 023100 004737 036162 JSR PC,CMPREG ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
2097 023104 062700 000002 ADD #2,R0 ;FORM ADDRESS OF NEXT USER PDR TO CLEAR
2098 023110 077312 SOB R3,4$ ;LOOP TO 4$ UNTIL ALL USER PDR'S CHECKED
2099 023112 012737 023130 001110 MOV #5$, $LPERR ;SET LOOP ON ERROR POINTER TO 5$
2100 023120 012703 000010 MOV #10,R3 ;LOAD LOOP COUNTER WITH AN 8
2101 023124 012700 177640 MOV #UIPAR0,R0 ;LOAD ADDRESS OF FIRST USER PAR IN R0
2102 023130 012706 001100 55: MOV #KERSTK,KSP ;SETUP STACK POINTER
2103 023134 004737 036070 JSR PC,SETREG ;SET ALL BITS IN ALL PAR'S AND PDR'S
2104 023140 005010 CLR (R0) ;CLEAR ONE OF THE USER PAR'S
2105 023142 004737 036162 JSR PC,CMPREG ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
    
```

```

2106 023146 062700 000002 ADD #2,R0 ;FORM ADDRESS OF NEXT USER PAR TO CLEAR
2107 023152 077312 SOB R3,5$ ;LOOP TO 5$ UNTIL ALL USER PAR'S CHECKED
2108 023154 012737 022744 001110 MOV #1$,$LPERR ;SET LOOP ON ERROR POINTER TO 1$
2109
2110 ;*****
2111 ;*TEST 22 TEST THAT PAR-PDR'S NOT AFFECTED BY RESET
2112 ;*
2113 ;* THIS TEST CHECKS TO SEE THAT THE KERNEL OR USER PAR/PDR'S ARE
2114 ;* NOT AFFECTED BY THE EXECUTION OF A "RESET" INSTRUCTION. THE
2115 ;* "SETREG" SUBROUTINE IS USED TO SET ALL WRITEABLE BITS TO A "1" IN
2116 ;* THE PAR/PDR'S. THEN THEY ARE READ TO SEE THAT THEY REMAINED
2117 ;* UNCHANGED
2118 ;*
2119 ;*****
2120 023162 000004 TST22: SCOPE
2121
2122
2123 023164 004737 036070 1$: JSR PC,SETREG ;SET ALL BITS IN ALL PAR'S AND PDR'S
2124 023170 000005 RESET ;ISSUE AN "INIT" BY EXECUTING A RESET
2125 023172 012700 172300 MOV #KIPDR0,R0 ;LOAD ADDRESS OF FIRST KERNEL PDR IN R0
2126 023176 012704 000010 MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
2127 023202 011001 2$: MOV (R0),R1 ;LOAD LOOP COUNTER WITH AN 8
2128 023204 022701 077416 CMP #77416,R1 ;READ A KERNEL PDR INTO R1
2129 023210 001401 BEQ 3$ ;ARE ALL THE BITS STILL SET?
2130 023212 104055 ERROR 5$ ;BRANCH IF YES
2131 ;*
2132 ;* KERNEL PDR AFFECTED BY A RESET
2133 ;* FOR TIGHTER SCOPE LOOP
2134 ;* REPLACE ERROR CALL WITH
2135 ;* "BR 2$" = 000773
2136 023214 062700 000002 3$: ADD #2,R0 ;FORM ADDRESS OF NEXT KERNEL PDR
2137 023220 077410 SOB R4,2$ ;LOOP TO 2$ UNTIL ALL KERNEL PDR'S CHECKED
2138 023222 012700 172340 MOV #KIPAR0,R0 ;LOAD ADDRESS OF FIRST KERNEL PAR IN R0
2139 023226 012704 000010 MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
2140 023232 011001 4$: MOV (R0),R1 ;LOAD LOOP COUNTER WITH AN 8
2141 023234 022701 007777 CMP #7777,R1 ;READ A KERNEL PAR INTO R1
2142 023240 001401 BEQ 5$ ;ARE ALL THE BITS STILL SET?
2143 023242 104055 ERROR 5$ ;BRANCH IF YES
2144 ;*
2145 ;* KERNEL PAR AFFECTED BY A RESET
2146 ;* FOR TIGHTER SCOPE LOOP
2147 ;* REPLACE ERROR CALL WITH
2148 ;* "BR 4$" = 000773
2149 023244 062700 000002 5$: ADD #2,R0 ;FORM ADDRESS OF NEXT KERNEL PAR
2150 023250 077410 SOB R4,4$ ;LOOP TO 4$ UNTIL ALL KERNEL PAR'S CHECKED
2151 023252 012700 177600 MOV #UIPDR0,R0 ;LOAD ADDRESS OF FIRST USER PDR IN R0
2152 023256 012704 000010 MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
2153 023262 011001 6$: MOV (R0),R1 ;LOAD LOOP COUNTER WITH AN 8
2154 023264 022701 077416 CMP #77416,R1 ;READ A USER PDR INTO R1
2155 023270 001401 BEQ 7$ ;ARE ALL THE BITS STILL SET?
2156 023272 104055 ERROR 5$ ;BRANCH IF YES
2157 ;*
2158 ;* USER PDR AFFECTED BY A RESET
2159 ;* FOR TIGHTER SCOPE LOOP
2160 ;* REPLACE ERROR CALL WITH
2161 ;* "BR 6$" = 000773
2162 023274 062700 000002 7$: ADD #2,R0 ;FORM ADDRESS OF NEXT USER PDR
2163 023300 077410 SOB R4,6$ ;LOOP TO 6$ UNTIL ALL USER PDR'S CHECKED
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195 023332 000004 TST23: SCOPE
2196 023334 004737 036002 1$: JSR PC,TOFF ;TURN T-BIT TRAPPING OFF FOR THIS TEST
2197 023340 012700 172300 MOV #KIPDR0,R0 ;LOAD ADDRESS OF FIRST KERNEL PDR INTO R0
2198 023344 012704 000010 MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
2199 023350 005020 2$: CLR (R0)+ ;CLEAR PDR - MAPPING PAGE NON-RES. 0 BLKS.
2200 023352 077402 SOB R4,2$ ;LOOP TO 2$ UNTIL ALL KERNEL PDR'S CLEARED
2201 023354 012737 000252 000250 MOV #MMVEC+2,MMVEC ;LOAD TRAP CATCHER INTO MEM MGM. VECTOR
2202 023362 005037 000252 CLR MMVEC+2 ;
2203 ;*
2204 ;* A HALT WILL OCCUR IF RESET FAILS
2205 ;* TO DISABLE DEST.-ONLY RELOCATION
2206 023366 012737 001006 172300 MOV #1006,KIPDR0 ;MAP KERNEL PG 0 R/W, 3 BLOCKS LONG.
2207 023374 012737 023402 001110 MOV #3$,$LPERR ;SET LOOP ON ERROR POINTER TO 3$
2208 023402 012737 000400 177572 3$: MOV #BIT8,SRO ;TURN ON DEST-ONLY-RELOCATION
2209 023410 000005 RESET ;SHOULD CLEAR MAINT. BIT - WILL ABORT IF RELOCATED
2210 023412 032737 000400 177572 BIT #BIT8,SRO ;WAS MAINT. BIT (BIT 8) OF SRO CLEARED?
2211 023420 001403 BEQ 4$ ;BRANCH IF YES
2212 023422 005037 177572 CLR SRO ;CLEAR SRO SO ERROR CAN BE REPORTED
2213 023426 104061 ERROR 61 ;MAINT. MODE NOT DISABLED BY A RESET
2214 ;*
2215 ;* FOR A TIGHTER SCOPE LOOP
2216 ;* REPLACE ERROR CALL WITH
2217 ;* "BR 3$" = 000765
2218 023430 012706 001100 4$: MOV #KERSTK,KSP ;RESTORE STACK POINTER
2219 023434 005037 177572 CLR SRO ;BE SURE SRO IS CLEAR

```

```

2162 023314 022701 007777 CMP #7777,R1 ;ARE ALL THE BITS STILL SET?
2163 023320 001401 BEQ 9$ ;BRANCH IF YES
2164 023322 104055 ERROR 5$ ;USER PAR AFFECTED BY A RESET
2165 ;*
2166 ;* FOR TIGHTER SCOPE LOOP
2167 ;* REPLACE ERROR CALL WITH
2168 ;* "BR 8$" = 000773
2169 023324 062700 000002 9$: ADD #2,R0 ;FORM ADDRESS OF NEXT USER PAR
2170 023330 077410 SOB R4,8$ ;LOOP TO 8$ UNTIL ALL USER PAR'S CHECKED
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182 ;*****
2183 ;*TEST 23 INSTRUCTION FETCH NOT RELOCATED IN MAINT. MODE
2184 ;*
2185 ;* THIS TEST CHECKS TO SEE THAT WHEN MEMORY MANAGEMENT IS IN
2186 ;* MAINTENANCE MODE (DESTINATION-ONLY-RELOCATION), AN INSTRUCTION
2187 ;* FETCH IS NOT RELOCATED AND A RESET CLEARS THE MAINTENANCE BIT
2188 ;* (BIT 08) IN SRO. IF THE "FETCH" IS RELOCATED, A PG.LENGTH ABORT
2189 ;* SHOULD OCCUR, CAUSING A HALT SINCE TRAP CATCHER IS PLACED IN VECTOR 250
2190 ;*
2191 ;*
2192 ;* NOTE: A HALT MAY OCCUR IF MAINT. MODE NOT DISABLED BY RESET
2193 ;*
2194 ;*****
2195 TST23: SCOPE
2196 1$: JSR PC,TOFF ;TURN T-BIT TRAPPING OFF FOR THIS TEST
2197 MOV #KIPDR0,R0 ;LOAD ADDRESS OF FIRST KERNEL PDR INTO R0
2198 MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
2199 CLR (R0)+ ;CLEAR PDR - MAPPING PAGE NON-RES. 0 BLKS.
2200 SOB R4,2$ ;LOOP TO 2$ UNTIL ALL KERNEL PDR'S CLEARED
2201 MOV #MMVEC+2,MMVEC ;LOAD TRAP CATCHER INTO MEM MGM. VECTOR
2202 CLR MMVEC+2 ;
2203 ;*
2204 ;* A HALT WILL OCCUR IF RESET FAILS
2205 ;* TO DISABLE DEST.-ONLY RELOCATION
2206 023366 012737 001006 172300 MOV #1006,KIPDR0 ;MAP KERNEL PG 0 R/W, 3 BLOCKS LONG.
2207 023374 012737 023402 001110 MOV #3$,$LPERR ;SET LOOP ON ERROR POINTER TO 3$
2208 023402 012737 000400 177572 3$: MOV #BIT8,SRO ;TURN ON DEST-ONLY-RELOCATION
2209 023410 000005 RESET ;SHOULD CLEAR MAINT. BIT - WILL ABORT IF RELOCATED
2210 023412 032737 000400 177572 BIT #BIT8,SRO ;WAS MAINT. BIT (BIT 8) OF SRO CLEARED?
2211 023420 001403 BEQ 4$ ;BRANCH IF YES
2212 023422 005037 177572 CLR SRO ;CLEAR SRO SO ERROR CAN BE REPORTED
2213 023426 104061 ERROR 61 ;MAINT. MODE NOT DISABLED BY A RESET
2214 ;*
2215 ;* FOR A TIGHTER SCOPE LOOP
2216 ;* REPLACE ERROR CALL WITH
2217 ;* "BR 3$" = 000765
2218 023430 012706 001100 4$: MOV #KERSTK,KSP ;RESTORE STACK POINTER
2219 023434 005037 177572 CLR SRO ;BE SURE SRO IS CLEAR

```

```

2218 023440 012737 002400 000250      MOV      #MGMERR,MMVEC      ;RESTORE MEM. MGMT. TRAP VECTOR
2219 023446 012737 000340 000252      MOV      #340,MMVEC+2      ;RESTORE MEM. MGMT VECTOR+2
2220 023454 012737 023334 001110      MOV      #1$,SLPERR        ;RESET LOOP ON ERROR POINTER TO 1$
2221 023462 004737 036036      JSR      PC,TON            ;TURN T-BIT TRAPPING BACK ON
    
```

2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235

```

;*****
;TEST 24      TEST THAT SOURCE NOT RELOCATED IN MAINTENANCE MODE
;
; THIS TEST CHECKS TO SEE THAT WHEN MEMORY MANAGEMENT IS IN
; MAINTENANCE MODE, THE SOURCE IS NOT RELOCATED. ONLY THE
; DESTINATION IS RELOCATED. KERNEL PAR'S 3 & 4 ARE MAPPED TO
; PHYSICAL ADDRESS 60000-77776. PDR4 IS SET TO ALLOW FULL READ/WRITE
; BUT PDR3 IS CLEAR ALLOWING TO ACCESS. VIRTUAL ADDRESSES REFERENCING
; PAR/PDR'S 4 & 3 ARE USED IN AS DESTINATION AND SOURCE RESPECTIVELY.
; IF THE SOURCE IS RELOCATED IN MAINTENANCE A MEM. MGMT. TRAP WILL
; OCCUR AND THE ERROR WILL BE REPORTED. KERNEL PG. 7 IS MAPPED R/W.
;*****
    
```

2248 023466 000004
2249 023470 004737 036002
2250 023474 012737 023564 001110
2251 023502 012737 000600 172346
2252 023510 012737 000600 172350
2253 023516 012737 007600 172356
2254 023524 005037 172306
2255 023530 012737 077406 172310
2256 023536 012737 077406 172316
2257 023544 012737 023644 000250
2258 023552 012737 000377 060000
2259 023560 012700 177777
2260 023564 052737 000400 177572 2\$:
2261 023572 113737 060000 100001
2262 023600 000005
2263 023602 013701 060000
2264 023606 020001
2265 023610 001401
2266 023612 104062
2267
2268
2269
2270
2271 023614 012737 002400 000250 3\$:
2272 023622 012737 077406 172306
2273 023630 012737 023470 001110

```

TST24: SCOPE
1$: JSR      PC,TOFF        ;TURN OFF T-BIT TRAPPING FOR THIS TEST
    MOV      #2$,SLPERR    ;SET LOOP ON ERROR POINTER TO 2$
    MOV      #600,KIPAR3   ;MAP KERNAL PAGE 3 TO 12-16K
    MOV      #600,KIPAR4   ;MAP KERNAL PAGE 4 TO 12-16K
    MOV      #7600,KIPAR7  ;MAP KERNAL PAGE 7 TO THE I/O PAGE
    CLR      KIPDR3        ;MAP KERNAL PAGE 3 "NO ACCESS"
    MOV      #77406,KIPDR4 ;MAP KERNAL PAGE 4 R/W, 128 BLKS
    MOV      #77406,KIPDR7 ;MAP KERNAL PAGE 7 R/W, 128 BLKS
    MOV      #4$,MMVEC     ;SET M.M. VECTOR TO 4$ IN CASE OF ABORT
    MOV      #377,060000   ;LOAD ALL 1'S IN LOW BYTE OF TEST LOC.
    MOV      #-1,R0        ;LOAD EXPECTED CONTENTS OF TEST LOC. IN R0
    BIC      #BIT8,SRO     ;TURN ON DEST.-ONLY-RELOCATION
    MOVNB   @#60000,@#100001 ;LOAD HI BYTE OF TEST LOC.-ABORT IF SOURCE RELOCATED
    RESET
    MOV      @#60000,R1    ;TURN OFF DEST.-ONLY-RELOCATION
    CMP      R0,R1        ;READ CONTENTS OF TEST LOC.
    BEQ     3$            ;WAS TEST LOCATION LOADED PROPERLY?
    BR      2$           ;BRANCH IF YES
    ERROR   62           ;TEST LOC. NOT LOADED - INST. WAS ABORTED
    ;WHEN SOURCE WAS RELOCATED
    ;FOR TIGHTER SCOPE LOOP REPLACE
    ;ERROR CALL WITH
    ;"BR 2$" = 000764
    MOV      #MGMERR,MMVEC ;RESTORE MEM. MGMT. VECTOR
    MOV      #77406,KIPDR3 ;MAP KERNAL PAGE 3 R/W, 128 BLKS
    MOV      #1$,SLPERR    ;RESET LOOP ON ERROR POINTER TO 1$
    
```

2274 023636 004737 036036
2275 023642 000430
2276
2277
2278 023644 042737 000400 177572 4\$:
2279 023652 013737 177572 001366
2280 023660 013737 177576 001370
2281 023666 010637 001354
2282 023672 012637 001356
2283 023676 012537 001360
2284 023702 042737 160000 177572
2285 023710 104063
2286
2287
2288
2289 023712 013746 001360
2290 023716 013746 001356
2291 023722 000002
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305

```

JSR      PC,TON            ;TURN T-BIT TRAPPING BACK ON
BR       TST25            ;SKIP TO NEXT TEST
;* IF THE PROGRAM TRIES TO RELOCATE THE SOURCE, IT SHOULD TRAP TO 4$
;*****
;TEST 25      RELOCATION & ADDER TEST (NO CARRIES)
;
; THE FOLLOWING TEST SETS UP THE KERNEL PAR'S AND PDR'S
; FOR THE REST OF THE PROGRAM. IT THEN USES DIFFERENT
; VIRTUAL ADDRESSES AND DIFFERENT VALUES FOR KERNEL PAR 4
; TO PUT DIFFERENT PATTERNS AT THE INPUTS OF THE THREE
; MEMORY MANAGEMENT ADDER CHIPS. THE VALUES ARE SUCH
; THAT NO CARRIES ARE GENERATED OUT OF ANY OF THE ADDER CHIPS.
;
; THE METHOD USED TO SEE THAT THE RIGHT PHYSICAL BUS ADDRESS
; IS FORMED BY THE ADDERS IS TO WRITE A PATTERN TO VIRTUAL
; LOCATION WITH MEMORY MGMT. IN THE MAINTENANCE MODE, AND
; THEN READ THAT LOCATION USING THE PHYSICAL ADDRESS THAT SHOULD
; HAVE BEEN FORMED TO SEE IF THE TEST PATTERN GOT THEIR.
;*****
TST25: SCOPE
1$: MOV      #KIPAR0,R0     ;LOAD ADDRESS OF FIRST KERNEL PAR IN R0
    CLR      R1            ;CLEAR R1
    MOV      #7,R2         ;LOAD LOOP COUNTER WITH A 7
    MOV      R1,(R0)+      ;MAP KERNAL PAR'S TO PAGES 0-6 (4K EACH)
    ADD     #200,R1
    SOB     R2,2$         ;LOOP UNTIL KIPAR0 - KIPAR6 ARE LOADED
    
```

2322 023724 000004
2323
2324 023726 012700 172340
2325 023732 005001
2326 023734 012702 000007
2327 023740 010120
2328 023742 062701 000200
2329 023746 077204

```

;*****
;TEST 25      RELOCATION & ADDER TEST (NO CARRIES)
;
; THE FOLLOWING TEST SETS UP THE KERNEL PAR'S AND PDR'S
; FOR THE REST OF THE PROGRAM. IT THEN USES DIFFERENT
; VIRTUAL ADDRESSES AND DIFFERENT VALUES FOR KERNEL PAR 4
; TO PUT DIFFERENT PATTERNS AT THE INPUTS OF THE THREE
; MEMORY MANAGEMENT ADDER CHIPS. THE VALUES ARE SUCH
; THAT NO CARRIES ARE GENERATED OUT OF ANY OF THE ADDER CHIPS.
;
; THE METHOD USED TO SEE THAT THE RIGHT PHYSICAL BUS ADDRESS
; IS FORMED BY THE ADDERS IS TO WRITE A PATTERN TO VIRTUAL
; LOCATION WITH MEMORY MGMT. IN THE MAINTENANCE MODE, AND
; THEN READ THAT LOCATION USING THE PHYSICAL ADDRESS THAT SHOULD
; HAVE BEEN FORMED TO SEE IF THE TEST PATTERN GOT THEIR.
;*****
TST25: SCOPE
1$: MOV      #KIPAR0,R0     ;LOAD ADDRESS OF FIRST KERNEL PAR IN R0
    CLR      R1            ;CLEAR R1
    MOV      #7,R2         ;LOAD LOOP COUNTER WITH A 7
    MOV      R1,(R0)+      ;MAP KERNAL PAR'S TO PAGES 0-6 (4K EACH)
    ADD     #200,R1
    SOB     R2,2$         ;LOOP UNTIL KIPAR0 - KIPAR6 ARE LOADED
    
```

```

2330 023750 012710 007600      MOV      #7600,(R0)      ;MAP KIPAR7 TO THE I/O PAGE
2331 023754 012700 172300      MOV      #KIPDR0,R0    ;LOAD ADDRESS OF FIRST KERNEL PDR IN R0
2332 023760 012701 077406      MOV      #77406,R1     ;LOAD PDR DATA INTO R1
2333 023764 012702 000010      MOV      #10,R2        ;LOAD LOOP COUNTER WITH AN 8
2334 023770 010120                3$:     MOV      R1,(R0)+       ;MAP ALL 8 PAGES 12B BLOCKS, UPWARD
2335 023772 077202                SOB      R2,$$         ;
2336                                ; EXPANDABLE, READ/WRITE
2337 023774 012737 023774 001110 4$:     MOV      #4$, $LPERR    ;SET LOOP ON ERROR POINTER TO 4$
2338 024002 012700 067776      MOV      #67776,R0     ;LOAD PHYSICAL ADDR. PBA INTO R0
2339 024006 012701 107776      MOV      #107776,R1    ;LOAD VIRTUAL ADDR. VBA INTO R1
2340 024012 012702 125250      MOV      #125250,R2    ;LOAD TEST PATTERN INTO R2
2341 024016 012704 000600      MOV      #600,R4 ;LOAD R4 WITH PAR VALUE
2342 024022 010437 172350      MOV      R4,KIPAR4     ;LOAD KERNEL PAR 4 BITS <11:00>
2343 024026 01137 001176      MOV      (R0), $TMP0   ;SAVE CONTENTS AT TEST LOCATION
2344 024032 052737 000400 177572     BIS      #BIT8,SRO     ;TURN ON "DESTINATION-ONLY-RELOCATION"
2345 024040 010211                MOV      R2,(R1)       ;LOAD 125250 USING ADDER CHIPS (PAR4 + VIRT ADDR.)
2346 024042 011003                MOV      (R0),R3       ;READ 125250 BACK WITHOUT USING MEM. MGMT.
2347 024044 000005                RESET     ;TURN OFF MEMORY MGMT. MAINT. MODE
2348 024046 013710 001176      MOV      $TMP0,(R0)    ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2349 024052 020203                CMP      R2,R3         ;WAS SAME PATTERN READ BACK THAT WAS
2350                                ; WRITTEN USING "DEST-ONLY-RELOC.?"
2351 024054 001405                BEQ      5$           ;BRANCH IF YES
2352 024056 010137 001402      MOV      R1,VIRT1     ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2353 024062 004737 036354      JSR      PC,FORMPA    ;GO FORM PHYSICAL ADDRESS FOR TYPING
2354 024066 104017                ERROR   17            ;TEST LOCATION DID NOT HAVE PATTERN
2355                                ; THAT SHOULD HAVE BEEN WRITTEN TO IT.
2356                                ; APPARENTLY PHYSICAL ADDR. WAS
2357                                ; FORMED WRONG BY ADDERS USING
2358                                ; THE VIRTUAL ADDR. AND KIPAR4
2359                                ; FOR TIGHTER SCOPE LOOP
2360                                ; REPLACE ERROR CALL WITH
2361                                ; "BR 4$" = 000742
2362 024070                5$:
2363 024070 012737 024070 001110 6$:     MOV      #6$, $LPERR    ;SET LOOP ON ERROR POINTER TO 6$
2364 024076 012700 067776      MOV      #67776,R0     ;LOAD PHYSICAL ADDR. PBA INTO R0
2365 024102 012701 125276      MOV      #102576,R1    ;LOAD VIRTUAL ADDR. VBA INTO R1
2366 024106 012702 125251      MOV      #125251,R2    ;LOAD TEST PATTERN INTO R2
2367 024112 012704 000652      MOV      #652,R4 ;LOAD R4 WITH PAR VALUE
2368 024116 010437 172350      MOV      R4,KIPAR4     ;LOAD KERNEL PAR 4 BITS <11:00>
2369 024122 011037 001176      MOV      (R0), $TMP0   ;SAVE CONTENTS AT TEST LOCATION
2370 024126 052737 000400 177572     BIS      #BIT8,SRO     ;TURN ON "DESTINATION-ONLY-RELOCATION"
2371 024134 010211                MOV      R2,(R1)       ;LOAD 125251 USING ADDER CHIPS (PAR4 + VIRT ADDR.)
2372 024136 011003                MOV      (R0),R3       ;READ 125251 BACK WITHOUT USING MEM. MGMT.
2373 024140 000005                RESET     ;TURN OFF MEMORY MGMT. MAINT. MODE
2374 024142 013710 001176      MOV      $TMP0,(R0)    ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2375 024146 020203                CMP      R2,R3         ;WAS SAME PATTERN READ BACK THAT WAS
2376                                ; WRITTEN USING "DEST-ONLY-RELOC.?"
2377                                ; BRANCH IF YES
2378 024150 001405                BEQ      7$           ;BRANCH IF YES
2379 024152 010137 001402      MOV      R1,VIRT1     ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2380 024156 004737 036354      JSR      PC,FORMPA    ;GO FORM PHYSICAL ADDRESS FOR TYPING
2381 024162 104017                ERROR   17            ;TEST LOCATION DID NOT HAVE PATTERN
2382                                ; THAT SHOULD HAVE BEEN WRITTEN TO IT.
2383                                ; APPARENTLY PHYSICAL ADDR. WAS
2384                                ; FORMED WRONG BY ADDERS USING
2385                                ; THE VIRTUAL ADDR. AND KIPAR4
2386                                ; FOR TIGHTER SCOPE LOOP
    
```

```

2386                                ;REPLACE ERROR CALL WITH
2387                                ; "BR 6$" = 000742
2388 024164                7$:
2389 024164 012737 024164 001110 8$:     MOV      #8$, $LPERR    ;SET LOOP ON ERROR POINTER TO 8$
2390 024172 012700 067776      MOV      #67776,R0     ;LOAD PHYSICAL ADDR. PBA INTO R0
2391 024176 012701 105276      MOV      #105276,R1    ;LOAD VIRTUAL ADDR. VBA INTO R1
2392 024202 012702 125252      MOV      #125252,R2    ;LOAD TEST PATTERN INTO R2
2393 024206 012704 000625      MOV      #625,R4 ;LOAD R4 WITH PAR VALUE
2394 024212 010437 172350      MOV      R4,KIPAR4     ;LOAD KERNEL PAR 4 BITS <11:00>
2395 024216 011037 001176      MOV      (R0), $TMP0   ;SAVE CONTENTS AT TEST LOCATION
2396 024222 052737 000400 177572     BIS      #BIT8,SRO     ;TURN ON "DESTINATION-ONLY-RELOCATION"
2397 024230 010211                MOV      R2,(R1)       ;LOAD 125252 USING ADDER CHIPS (PAR4 + VIRT ADDR.)
2398 024232 011003                MOV      (R0),R3       ;READ 125252 BACK WITHOUT USING MEM. MGMT.
2399 024234 000005                RESET     ;TURN OFF MEMORY MGMT. MAINT. MODE
2400 024236 013710 001176      MOV      $TMP0,(R0)    ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2401 024242 020203                CMP      R2,R3         ;WAS SAME PATTERN READ BACK THAT WAS
2402                                ; WRITTEN USING "DEST-ONLY-RELOC.?"
2403                                ; BRANCH IF YES
2404 024244 001405                BEQ      9$           ;BRANCH IF YES
2405 024246 010137 001402      MOV      R1,VIRT1     ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2406 024252 004737 036354      JSR      PC,FORMPA    ;GO FORM PHYSICAL ADDRESS FOR TYPING
2407 024256 104017                ERROR   17            ;TEST LOCATION DID NOT HAVE PATTERN
2408                                ; THAT SHOULD HAVE BEEN WRITTEN TO IT.
2409                                ; APPARENTLY PHYSICAL ADDR. WAS
2410                                ; FORMED WRONG BY ADDERS USING
2411                                ; THE VIRTUAL ADDR. AND KIPAR4
2412                                ; FOR TIGHTER SCOPE LOOP
2413                                ; REPLACE ERROR CALL WITH
2414                                ; "BR 8$" = 000742
2415                                9$:
2416 024260                9$:
2417 024260 012737 024260 001110 10$:    MOV      #10$, $LPERR   ;SET LOOP ON ERROR POINTER TO 10$
2418 024266 012700 177776      MOV      #PSW,R0       ;LOAD PHYS. ADDR. OF PSW INTO R0
2419 024272 012701 100076      MOV      #100076,R1    ;LOAD VIRTUAL ADDR. FOR PSW INTO R1
2420 024276 012702 030340      MOV      #030340,R2    ;LOAD DATA FOR PSW IN R2
2421 024302 012704 007777      MOV      #77777,R4     ;LOAD R4 WITH PAR VALUE
2422 024306 010437 172350      MOV      R4,KIPAR4     ;LOAD KERNEL PAR 4 BITS <11:00>
2423 024312 005010                CLR      (R0)          ;CLEAR THE PSW
2424 024314 052737 000400 177572     BIS      #BIT8,SRO     ;TURN ON "DESTINATION-ONLY-RELOCATION"
2425 024322 010211                MOV      R2,(R1)       ;LOAD PSW USING ADDER CHIPS (PAR4 + VIRT ADDR.)
2426 024324 011003                MOV      (R0),R3       ;READ PSW BACK WITHOUT USING MEM. MGMT.
2427 024326 000005                RESET     ;TURN OFF MEM. MGMT MAINT. MODE
2428 024330 005010                CLR      (R0)          ;CLEAR THE PSW
2429 024332 042703 000037      BIC      #37,R3        ;MASK T-BIT & CC BITS OUT OF DATA READ
2430 024336 020203                CMP      R2,R3         ;WAS PSW WRITTEN WHILE IN MAINT. MODE?
2431 024340 001405                BEQ      11$          ;BRANCH IF YES
2432 024342 010137 001402      MOV      R1,VIRT1     ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2433 024346 004737 036354      JSR      PC,FORMPA    ;GO FORM PHYSICAL ADDR. FOR TYPING
2434 024352 104017                ERROR   17            ;PSW DID NOT HAVE DATA THAT IT SHOULD HAVE,
2435                                ; APPARENTLY PHYS. ADDR. OF PSW WAS
2436                                ; NOT FORMED BY ADDERS USING THE
2437                                ; VIRTUAL ADDR. AND KIPAR4
2438                                ; FOR TIGHTER SCOPE LOOP
2439                                ; REPLACE ERROR CALL WITH
2440                                ; "BR 10$" = 000742
2441 024354 012737 024354 001110 11$:    MOV      #11$, $LPERR   ;SET LOOP ON ERROR POINTER TO 11$
2442 024362 012700 177776      MOV      #PSW,R0       ;LOAD PHYS. ADDR. OF PSW INTO R0
    
```



```

2554                                     ;THE VIRTUAL ADDR. AND KIPAR4
2555                                     ;FOR TIGHTER SCOPE LOOP
2556                                     ;REPLACE ERROR CALL WITH
2557                                     ;"BR 6$" = 000742
2558 024742                                7$:
2559 024742 012737 024742 001110 8$: MOV #8$, $LPERR ;SET LOO ON ERROR POINTER TO 8$
2560 024750 012700 060076 MOV #60076, R0 ;LOAD PHYSICAL ADDR. PBA INTO R0
2561 024754 012701 107776 MOV #107776, R1 ;LOAD VIRTUAL ADDR. VBA INTO R1
2562 024760 012702 125256 MOV #125256, R2 ;LOAD TEST PATTERN INTO R2
2563 024764 012704 000501 MOV #501, R4 ;LOAD R4 WITH PAR VALUE
2564 024770 010437 172350 MOV R4, KIPAR4 ;LOAD KERNEL PAR 4 BITS <11:00>
2565 024774 011037 001176 MOV (R0), $TMPD ;SAVE CONTENTS AT TEST LOCATION
2566 025000 052737 000400 177572 BIS #BIT8, SR0 ;TURN ON "DESTINATION-ONLY-RELOCATION"
2567 025006 010211 MOV R2, (R1) ;LOAD 125256 USING ADDER CHIPS (PAR4 + VIRT ADDR.)
2568 025010 011003 MOV (R0), R3 ;READ 125256 BACK WITHOUT USING MEM. MGMT.
2569 025012 000005 RESET ;TURN OFF MEMORY MGMT. MAINT. MODE
2570 025014 013710 001176 MOV $TMPD, (R0) ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2571 025020 020203 CMP R2, R3 ;WAS SAME PATTERN READ BACK THAT WAS
2572                                     ;WRITTEN USING "DEST-ONLY-RELOC. "?
2573 025022 001405 BEQ 9$ ;BRANCH IF YES
2574 025024 010137 001402 MOV R1, VIRT1 ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2575 025030 004737 036354 JSR PC, FORMPA ;GO FORM PHYSICAL ADDRESS FOR TYPING
2576 025034 104017 ERROR 17 ;TEST LOCATION DID NOT HAVE PATTERN
2577                                     ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
2578                                     ;APPARENTLY PHYSICAL ADDR. WAS
2579                                     ;FORMED WRONG BY ADDERS USING
2580                                     ;THE VIRTUAL ADDR. AND KIPAR4
2581                                     ;FOR TIGHTER SCOPE LOOP
2582                                     ;REPLACE ERROR CALL WITH
2583                                     ;"BR 8$" = 000742
2584 025036                                9$:
2585 025036 012737 025036 001110 10$: MOV #10$, $LPERR ;SET LOOP ON ERROR POINTER TO 10$
2586 025044 012700 000000 MOV #000000, R0 ;LOAD PHYSICAL ADDR. PBA INTO R0
2587 025050 012701 100100 MOV #100100, R1 ;LOAD VIRTUAL ADDR. VBA INTO R1
2588 025054 012702 125257 MOV #125257, R2 ;LOAD TEST PATTERN INTO R2
2589 025060 012704 007777 MOV #7777, R4 ;LOAD R4 WITH PAR VALUE
2590 025064 010437 172350 MOV R4, KIPAR4 ;LOAD KERNEL PAR 4 BITS <11:00>
2591 025070 011037 001176 MOV (R0), $TMPD ;SAVE CONTENTS AT TEST LOCATION
2592 025074 052737 000400 177572 BIS #BIT8, SR0 ;TURN ON "DESTINATION-ONLY-RELOCATION"
2593 025102 010211 MOV R2, (R1) ;LOAD 125257 USING ADDER CHIPS (PAR4 + VIRT ADDR.)
2594 025104 011003 MOV (R0), R3 ;READ 125257 BACK WITHOUT USING MEM. MGMT.
2595 025106 000005 RESET ;TURN OFF MEMORY MGMT. MAINT. MODE
2596 025110 013710 001176 MOV $TMPD, (R0) ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
2597 025114 020203 CMP R2, R3 ;WAS SAME PATTERN READ BACK THAT WAS
2598                                     ;WRITTEN USING "DEST-ONLY-RELOC. "?
2599 025116 001405 BEQ 11$ ;BRANCH IF YES
2600 025120 010137 001402 MOV R1, VIRT1 ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2601 025124 004737 036354 JSR PC, FORMPA ;GO FORM PHYSICAL ADDRESS FOR TYPING
2602 025130 104017 ERROR 17 ;TEST LOCATION DID NOT HAVE PATTERN
2603                                     ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
2604                                     ;APPARENTLY PHYSICAL ADDR. WAS
2605                                     ;FORMED WRONG BY ADDERS USING
2606                                     ;THE VIRTUAL ADDR. AND KIPAR4
2607                                     ;FOR TIGHTER SCOPE LOOP
2608                                     ;REPLACE ERROR CALL WITH
2609                                     ;"BR 10$" = 000742
    
```

```

2610 025132                                11$:
2611 025132 012737 024456 001110 MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
2612
2613 ;*****
2614 ;*TEST 27 READ AND WRITE WHILE IN RELOCATE MODE
2615 ;*
2616 ;* THE FOLLOWING TEST TURNS ON MEMORY MANAGEMENT AND THEN
2617 ;* READS AND WRITES LOCATIONS BETWEEN PHYSICAL ADDRESSES
2618 ;* 060000-067600. ONE LOCATION IN EVERY BLOCK (32. WORDS)
2619 ;* IS WRITTEN USING PAR4 AND READ USING PAR5. THIS IS
2620 ;* DONE IN BOTH USER AND KERNEL MODES. THE USER PAR/PDR'S
2621 ;* ARE SETUP AT THE BEGINNING OF THE TEST AND ONCE MEMORY
2622 ;* MANAGEMENT IS TURNED ON IT IS LEFT ON FOR THE REST OF THE
2623 ;* OF THE PROGRAM. THE "MODE" INPUT TO THE PAR/PDR ADDRESS MUX
2624 ;* IS CHECKED BY READING AND WRITING IN USER MODE. REMEMBER
2625 ;* ALSO, THAT SINCE MEMORY MANAGEMENT IS ON (IN RELOCATE
2626 ;* MODE) THE PROGRAM ITSELF IS USING ITS VIRTUAL ADDRESSES AND
2627 ;* THE PAR/PDR'S TO EXECUTE.
2628 ;*
2629 ;* WHILE TESTING IN KERNEL MODE, USER PAGFS 4 & 5 ARE MAPPED
2630 ;* NON-RESIDENT WITH DIFFERENT PAR VALUES THAN THE KERNEL
2631 ;* PAR'S TO BE SURE THAT THE KERNEL PAR'S AND PDR'S ARE BEING
2632 ;* USED WHEN IN KERNEL MODE (AND VICE VERSA WHILE TESTING IN
2633 ;* USER MODE). IF A MEM. MGMT. TRAP OCCURS, THE PROGRAM GOES
2634 ;* TO 8$ WHERE THE TRAP IS REPORTED.
2635 ;*
2636 ;*****
2637 025140 000004 TST27: SCOPE
2638
2639 025142 005037 177776 1$: CLR PSW ;START IN KERNEL MODE
2640 025146 012704 000577 MOV #577, R4 ;LOAD R4 WITH VALUE FOR PAR4
2641 025152 012705 000600 MOV #600, R5 ;LOAD R5 WITH VALUE FOR PAR5
2642 025156 010437 172350 MOV R4, KIPAR4 ;LOAD KERNEL PAR4
2643 025162 010537 172352 MOV R5, KIPAR5 ;LOAD KERNEL PAR5
2644 025166 012700 177640 MOV #UIPAR0, R0 ;LOAD ADDRESS OF FIRST USER PAR IN R0
2645 025172 005001 CLR R1 ;CLEAR R1
2646 025174 012702 000007 MOV #7, R2 ;LOAD LOOP COUNTER WITH A 7
2647 025200 010120 2$: MOV R1, (R0)+ ;MAP USER PAR'S TO PAGES 0-6 (4K EACH)
2648 025202 062701 000200 ADD #200, R1
2649 025206 077204 SOB R2, 2$ ;LOOP UNTIL UIPAR0-UIPAR6 ARE LOADED
2650 025210 012710 007600 MOV #7600, (R0) ;MAP USER PAR7 TO THE I/O PAGE
2651 025214 012700 177600 MOV #UIPDR0, R0 ;LOAD ADDRESS OF FIRST USER PDR IN R0
2652 025220 012701 077406 MOV #77406, R1 ;LOAD PDR DATA INTO R1
2653 025224 012702 000010 MOV #10, R2 ;LOAD LOOP COUNTER WITH AN 8
2654 025230 010120 3$: MOV R1, (R0)+ ;MAP ALL 8 PAGES 128 BLOCKS. UPWARD
2655 025232 077202 SOB R2, 3$ ;EXPANDABLE, READ/WRITE
2656 025234 105037 177610 CLR UIPDR4 ;MAP USER SPACE NON-RESIDENT WHILE
2657 025240 105037 177612 CLR UIPDR5 ; TESTING KERNEL SPACE
2658 025244 010537 177650 MOV R5, UIPAR4 ;MAP USER PAR'S OPPOSITE OF KIPAR'S
2659 025250 010437 177652 MOV R4, UIPAR5
2660 025254 012737 000001 177572 MOV #1, SR0 ;TURN ON MEMORY MANAGEMENT (RELOCATE MODE)
2661 025262 012737 025314 001110 MOV #5$, $LPERR ;SET LOOP ON ERROR POINTER TO 5$
2662 025270 012737 025526 000250 MOV #8$, MVEC ;SET M. TRAP VECTOR TO 8$
2663 025276 013737 177776 001176 4$: MOV PSW, $TMPD ;SAVE PSW IN CASE OF ERROR
2664 025304 012700 100100 MOV #100100, R0 ;PUT VIRTUAL ADDR. THAT USER PAR4 IN R0
2665 025310 012701 120000 MOV #120000, R1 ;PUT VIRTUAL ADDR. THAT USES PAR5 IN R1
    
```

```

2666 025314 010010 5S: MOV R0,(R0) ;WRITE TO TEST LOC. USING PAR4
2667 025316 011102 MOV (R1),R2 ;READ THE SAME LOC., BUT USING PAR5
2668 025320 020002 CMP R0,R2 ;DID WE READ WHAT WE WROTE?
2669 025322 001411 BEQ 6$ ;BRANCH IF YES
2670 025324 010137 001404 MOV R1,VIRT2 ;SAVE VIRTUAL ADDR. THAT SELECTED PAR5
2671 025330 010037 001402 MOV R0,VIRT1 ;SAVE VIRTUAL ADDR. THAT SELECTED PAR4
2672 025334 004737 036354 JSR PC,FORMPA ;GO FORM PHYSICAL ADDRESS BEING USED
2673 025340 104020 ERROR 20 ;READING LOC. USING PARS AND A VIRT.
2674 ;ADDR. DID NOT FIND DATA WRITTEN WHEN USING
2675 ;PAR4 AND VIRT. ADDRESS.
2676 ;FOR TIGHTER SCOPE LOOP
2677 ;REPLACE ERROR CALL WITH
2678 ;"BR 5$" = 000765
2679 025342 013700 001402 MOV VIRT1,R0 ;RESTORE VBA IN R0
2680 025346 062700 000100 ADD #100,R0 ;CHANGE VIRTUAL ADDR. TO POINT TO NEXT BLOCK
2681 025352 062701 000100 ADD #100,R1
2682 025356 020127 127700 CMP R1,#127700 ;WERE BLOCKS FROM 60000-676000 ALL TRIED?
2683 025362 001354 BNE 5$ ;BRANCH IF NO
2684 025364 032737 140000 177776 BIT #140000,PSW ;HAVE WE DONE TEST IN USER MODE YET?
2685 025372 001026 BNE 7$ ;BRANCH IF YES
2686 025374 010437 177650 MOV R4,UIPAR4 ;LOAD USER PAR4
2687 025400 010537 177652 MOV R5,UIPAR5 ;LOAD USER PARS
2688 025404 112737 000006 177810 MOV# #6,UIPDR4 ;MAP USER SPACE R/W TO TEST IT
2689 025412 112737 000006 177812 MOV# #6,UIPDR5
2690 025420 105037 172310 CLRB KIPDR4 ;MAP KERNEL SPACE NON-RESIDENT WHILE
2691 025424 105037 172312 CLRB KIPDR5 ; TESTING USER SPACE
2692 025430 010537 172350 MOV R5,KIPAR4 ;MAP KERNEL PAR'S OPPOSITE UIPAR'S
2693 025434 010437 172352 MOV R4,KIPAR5
2694 025440 012737 140000 177776 MOV #140000,PSW ;GO TO USER MODE
2695 025446 000713 BR 4$ ;GO BACK AND READ/WRITE IN USER MODE
2696 025450 005037 177776 CLR PSW ;GO BACK TO KERNEL MODE BEFORE LEAVING
2697 025454 012737 077406 172310 MOV #77406,KIPDR4 ;REMAP KERNEL PAGES READ/WRITE
2698 025462 012737 077406 172312 MOV #77406,KIPDR5
2699 025470 010537 172350 MOV R5,KIPAR4 ;MAP KERNEL AND USER PAR'S 4 & 5
2700 025474 010537 172352 MOV R5,KIPAR5 ; BACK TO 12-16K
2701 025500 010537 177650 MOV R5,UIPAR4
2702 025504 010537 177652 MOV R5,UIPAR5
2703 025510 012737 002400 000250 MOV #MGERR,MMVEC ;RESTORE ADDR. OF NORMAL M.M. TRAP ROUTINE
2704 025516 012737 025142 001110 MOV #1$,SLPERR ;RESET LOOP ON ERROR POINTER TO 1$
2705 025524 000427 BR TST30 ;GO TO NEXT TEST
2706
2707 025526 012637 001356 8S: MOV (KSP)+,TRAPPC ;SAVE PC & PS OF TRAP
2708 025532 012637 001360 MOV (KSP)+,TRAPPS
2709 ;PROGRAM WILL TRAP TO HERE IF TRY
2710 ;TO USE USER PDR'S WHEN IN KERNEL MODE
2711 ;OR KERNEL PDR'S WHEN IN USER MODE
2712 025536 010037 001402 MOV R0,VIRT1 ;SAVE VIRTUAL ADDRESS FOR ERROR REPORT
2713 025542 004737 036354 JSR PC,FORMPA ;GO FORM THE PHYSICAL ADDRESS BEING USED
2714 025546 013737 177572 001366 MOV SRO,WASSRO ;SAVE SRO & SR2 FOR ERROR REPORT
2715 025554 013737 177576 001370 MOV SR2,WASSR2
2716 025562 042737 160000 177572 BIC #160000,SRO ;CLEAR ERROR BITS IN SRO
2717 025570 104052 ERROR 52 ;M.M. TRAP WHILE IN RELOCATE MODE -
2718 ;REFERENCED WRONG SET OF PDR'S
2719 ;FOR TIGHTER SCOPE LOOP
2720 ;REPLACE ERROR CALL WITH
2721 ;A "NOP" = 000240
    
```

```

2722 025572 013746 001360 MOV TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
2723 025576 013746 001356 MOV TRAPPC,-(KSP)
2724 025602 000002 RTI ;RETURN TO TEST
2725
2726
2727 ;*****
2728 ;*TEST 30 W-BIT LOGIC TEST, KERNEL PDR'S
2729 ;*
2730 ;* THIS TEST WRITES TO EIGHT (8) DIFFERENT VIRTUAL ADDRESSES
2731 ;* (VBA'S = 17776,37776,57776,77776,117776,137776,157776, & 177776
2732 ;* & PBA'S CONSTRUCTED = 17776,37776,57776,77776,77776,
2733 ;* 77776,77776, & 77776 RESPECTIVELY).
2734 ;* WHICH SHOULD CAUSE THE "W-BIT" TO SET IN EACH OF THE
2735 ;* EIGHT (8) KERNEL PAGE DESCRIPTOR REGISTERS. THE PDR'S
2736 ;* ARE CHECKED TO SEE THAT IT'S W-BIT DOES SET WHEN THE
2737 ;* PAGE IT IS MAPPED TO IS WRITTEN TO AND THAT THE W-BIT
2738 ;* DOES NOT SET IN ANY OF THE OTHER PDR'S. KERNEL PDR'S 3,4,5,6
2739 ;* ARE MAPPED TO 12-16K FOR THIS TEST. ALSO THE W-BIT
2740 ;* SHOULD BE CLEARED WHEN THE PDR IS WRITTEN TO. THE
2741 ;* W-BIT PORTION OF THE PDR'S AND THE PAR/PDR ADRS MUX
2742 ;* ARE BEING CHECKED.
2743 ;*****
2744 TST30: SCOPE
2745 1$:
2746 JSR PC,TOFF ;TURN T-BIT TRAPPING OFF FOR THIS TEST
2747 MOV #4,R2 ;SET LOOP COUNTER TO 4
2748 MOV #KIPAR3,R0 ;LOAD ADDRESS OF PAR3 INTO R0
2749 MOV #600,R1 ;LOAD "12-16K" PAR VALUE INTO R1
2750 2$: MOV R1,(R0)+ ;MAP PARS 3-6 TO 12-16K
2751 SOB R2,2$ ;LOOP TIL ALL 4 OF THEM LOADED
2752 MOV #KIPDR0,R5 ;LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
2753 MOV #10,R4 ;SET LOOP COUNTER TO 8
2754 3$: MOV #17776,R3 ;INITIALIZE VIRTUAL ADDRESS TO BE IN R3
2755 MOV #3$,SLPERR ;SET LOOP ON ERROR POINTER TO 3$
2756 MOV #KIPDR0,R0 ;LOAD ADDR. OF FIRST PDR TO BE SETUP IN R0
2757 MOV #10,R2 ;SET LOOP COUNTER TO 8
2758 MOV #77406,R1 ;PUT "W-BIT OFF DATA" INTO R1
2759 4$: MOV R1,(R0)+ ;CLEAR ALL W-BITS BY WRITING TO ALL PDRS
2760 SOB R2,4$ ;LOOP UNTIL ALL OF THEM SETUP
2761 MOV (R3),(R3) ;DO "DATA" TO VIRTUAL ADDR.-SETTING A W-BIT
2762 BIT (R5),#WBIT ;DID THAT CAUSE W-BIT TO BE SET?
2763 BNE 5$ ;BRANCH IF YES
2764 ERROR 21 ;W-BIT DID NOT GET SET IN PDR
2765 ;FOR TIGHTER SCOPE LOOP
2766 ;REPLACE ERROR CALL WITH
2767 ;"BR 3$" = 000763
2768 BR 8$ ;SKIP CHECKING OTHER PDR'S-ERROR WILL SET W-BITS
2769 5$: MOV #10,R2 ;SET LOOP COUNTER TO 8
2770 MOV #KIPDR0,R0 ;LOAD ADDR. OF FIRST PDR TO BE CHECKED IN R0
2771 6$: BIT (R0),#WBIT ;DID W-BIT IN OTHER PDRS REMAIN CLEAR?
2772 BEQ 7$ ;BRANCH IF YES
2773 CMP R5,R0 ;IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
2774 BEQ 7$ ;BRANCH IF YES
2775 ERROR 22 ;W-BIT GOT SET IN MORE THAN ONE PDR
2776 ;FOR TIGHTER SCOPE LOOP
2777 ;REPLACE ERROR CALL WITH
    
```

```
2778 ;"BR 3$" = 000750
2779 025734 062700 000002 7$: ADD #2,R0 ;POINT R0 TO NEXT PDR TO BE CHECKED
2780 025740 077211 SOB R2,6$ ;LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
2781 025742 010115 MOV R1,(R5) ;WRITE TO THE PDR TESTED TO CLEAR W-BIT
2782 025744 031527 000100 BIT (R5),#WBIT ;DID WRITING PDR CLEAR THE W-BIT?
2783 025750 001401 BEQ 8$ ;BRANCH IF YES
2784 025752 104023 ERROR 23 ;W-BIT DID NOT CLEAR BY WRITING THE PDR
2785 ;FOR TIGHTER SCOPE LOOP
2786 ;REPLACE ERROR CALL WITH
2787 ;"BR 3$" = 000740
2788 025754 062705 000002 8$: ADD #2,R5 ;POINT R5 TO THE NEXT PDR TO BE TESTED
2789 025760 062703 020000 ADD #20000,R3 ;CHANGE VIRT. ADDR TO REF. NEXT PDR
2790 025764 077445 SOB R4,3$ ;LOOP BACK TO 3$ UNTIL ALL 8 PDR'S TESTED
2791 025766 012737 025606 001110 MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
2792 025774 004737 036036 JSR PC,TON ;TURN T-BIT BACK ON FOR NEXT TEST
2793
2794 ;*****
2795 ;*TEST 31 W-BIT LOGIC TEST, USER PDR'S
2796 ;*
2797 ;* THIS TEST WRITES TO EIGHT (8) DIFFERENT VIRTUAL ADDRESSES
2798 ;* (VBA'S = 17776,37776,57776,77776,117776,137776,157776, & 177776
2799 ;* & PBA'S CONSTRUCTED = 17776,37776,57776,77776,77776,
2800 ;* 77776,77776, & 777776 RESPECTIVELY).
2801 ;* WHICH SHOULD CAUSE THE "W-BIT" TO SET IN EACH OF THE
2802 ;* EIGHT (8) USER PAGE DESCRIPTOR REGISTERS. THE PDR'S
2803 ;* ARE CHECKED TO SEE THAT IT'S W-BIT DOES SET WHEN THE
2804 ;* PAGE IT IS MAPPED TO IS WRITTEN TO AND THAT THE W-BIT
2805 ;* DOES NOT SET IN ANY OF THE OTHER PDR'S. USER PDR'S 3,4,5,6
2806 ;* ARE MAPPED TO 12-16K FOR THIS TEST. ALSO THE W-BIT
2807 ;* SHOULD BE CLEARED WHEN THE PDR IS WRITTEN TO. THE
2808 ;* W-BIT PORTION OF THE PDR'S AND THE PAR/PDR ADRS MUX
2809 ;* ARE BEING CHECKED.
2810 ;*****
2811 026000 000004 TST31: SCOPE
2812 026002 012737 140000 177776 1$: MOV #140000,PSW ;GO TO USER MODE FOR THIS TEST
2813 026010 004737 036002 JSR PC,TOFF ;TURN T-BIT TRAPPING OFF FOR THIS TEST
2814 026014 012702 000004 MOV #4,R2 ;SET LOOP COUNTER TO 4
2815 026020 012700 177646 MOV #UIPAR3,R0 ;LOAD ADDRESS OF PAR3 INTO R0
2816 026024 012701 000600 MOV #600,R1 ;LOAD "12-16K" PAR VALUE INTO R1
2817 026030 010120 2$: MOV R1,(R0)+ ;MAP PARS 3-6 TO 12-16K
2818 026032 077202 SOB R2,2$ ;LOOP TIL ALL 4 OF THEM LOADED
2819 026034 012705 177600 MOV #UIPDR0,R5 ;LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
2820 026040 012704 000010 MOV #10,R4 ;SET LOOP COUNTER TO 8
2821 026044 012703 017776 MOV #17776,R3 ;INITIALIZE VIRTUAL ADDRESS TO BE IN R3
2822 026050 012737 026056 001110 MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
2823 026056 012700 177600 3$: MOV #UIPDR0,R0 ;LOAD ADDR. OF FIRST PDR TO BE SETUP IN R0
2824 026062 012702 000010 MOV #10,R2 ;SET LOOP COUNTER TO 8
2825 026066 012701 077406 MOV #77406,R1 ;PUT "W-BIT OFF DATA" INTO R1
2826 026072 010120 4$: MOV R1,(R0)+ ;CLEAR ALL W-BITS BY WRITING TO ALL PDRS
2827 026074 077202 SOB R2,4$ ;LOOP UNTIL ALL OF THEM SETUP
2828 026076 011313 MOV (R3),(R3) ;DO "DATO" TO VIRTUAL ADDR.-SETTING A W-BIT
2829 026100 031527 000100 BIT (R5),#WBIT ;DID THAT CAUSE W-BIT TO BE SET?
2830 026104 001002 BNE 5$ ;BRANCH IF YES
2831 026106 104021 ERROR 21 ;W-BIT DID NOT GET SET IN PDR
2832 ;FOR TIGHTER SCOPE LOOP
2833 ;REPLACE ERROR CALL WITH
```

```
2834 ;"BR 3$" = 000763
2835 026110 000422 BR 8$ ;SKIP CHECKING OTHER PDR'S-ERROR WILL SET W-BITS
2836 026112 012702 000010 5$: MOV #10,R2 ;SET LOOP COUNTER TO 8
2837 026116 012700 177600 MOV #UIPDR0,R0 ;LOAD ADDR. OF FIRST PDR TO BE CHECKED IN R0
2838 026122 031027 000100 6$: BIT (R0),#WBIT ;DID W-BIT IN OTHER PDRS REMAIN CLEAR?
2839 026126 001403 BEQ 7$ ;BRANCH IF YES
2840 026130 020500 CMP R5,R0 ;IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
2841 026132 001401 BEQ 7$ ;BRANCH IF YES
2842 026134 104022 ERROR 22 ;W-BIT GOT SET IN MORE THAN ONE PDR
2843 ;FOR TIGHTER SCOPE LOOP
2844 ;REPLACE ERROR CALL WITH
2845 ;"BR 3$" = 000750
2846 026136 062700 000002 7$: ADD #2,R0 ;POINT R0 TO NEXT PDR TO BE CHECKED
2847 026142 077.11 SOB R2,6$ ;LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
2848 026144 010115 MOV R1,(R5) ;WRITE TO THE PDR TESTED TO CLEAR W-BIT
2849 026146 031527 000100 BIT (R5),#WBIT ;DID WRITING PDR CLEAR THE W-BIT?
2850 026152 001401 BEQ 8$ ;BRANCH IF YES
2851 026154 104023 ERROR 23 ;W-BIT DID NOT CLEAR BY WRITING THE PDR
2852 ;FOR TIGHTER SCOPE LOOP
2853 ;REPLACE ERROR CALL WITH
2854 ;"BR 3$" = 000740
2855 026156 062705 000002 8$: ADD #2,R5 ;POINT R5 TO THE NEXT PDR TO BE TESTED
2856 026162 062703 020000 ADD #20000,R3 ;CHANGE VIRT. ADDR TO REF. NEXT PDR
2857 026166 077445 SOB R4,3$ ;LOOP BACK TO 3$ UNTIL ALL 8 PDR'S TESTED
2858 026170 012737 026002 001110 MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
2859 026176 004737 036036 JSR PC,TON ;TURN T-BIT BACK ON FOR NEXT TEST
2860 026202 005037 177776 CLR PSW ;BACK TO KERNEL MODE BEFORE LEAVING
2861
2862 ;*****
2863 ;*TEST 32 TEST "W-BIT NOT SET" CASES
2864 ;*
2865 ;* THIS TEST CHECKS TWO SPECIAL CASES WHERE THE W-BIT DOES
2866 ;* NOT GET SET ON A WRITE. FIRST CASE IS THAT THE W-BIT
2867 ;* SHOULD NOT SET IN PAGE DESCRIPTOR REG. 7 WHEN WRITING TO
2868 ;* STATUS REG SRO (KERNEL PDR 7 IS USED). SECOND CASE IS THAT
2869 ;* THE W-BIT IS NOT SET IF THE "DATO" IS ABORTED DUE TO AN
2870 ;* ODD ADDRESS ERROR (KERNEL PDR3 & VIRTUAL ADDR 60001 ARE USED).
2871 ;*
2872 ;*****
2873 026206 000004 TST32: SCOPE
2874
2875 026210 004737 036002 1$: JSR PC,TOFF ;TURN OFF T-BIT TRAPPING FOR THIS TEST
2876 026214 012701 077406 MOV #77406,R1 ;PUT "W-BIT OFF" VALUE FOR PDR IN R1
2877 026220 012737 026226 001110 MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
2878 026226 010137 172316 2$: MOV R1,KIPDR7 ;LOAD KERNEL PDR 7 TO CLEAR W-BIT
2879 026232 013700 177572 MOV SRO,R0 ;READ PRESENT CONTENTS OF STATUS REG. 0
2880 026236 010037 177572 MOV R0,SRO ;WRITE PRESENT CONTENTS OF SRO BACK TO ITSELF
2881 026242 013702 172316 MOV KIPDR7,R2 ;READ CONTENTS OF KIPDR7 INTO R2
2882 026246 020102 CMP R1,R2 ;WAS W-BIT LEFT CLEARED?
2883 026250 001401 BEQ 3$ ;BRANCH IF YES
2884 026252 104024 ERROR 24 ;W-BIT IN KIPDR7 SET WHEN SRO WAS WRITTEN TO
2885 ;FOR TIGHTER SCOPE LOOP
2886 ;REPLACE ERROR CALL WITH
2887 ;"BR 2$" = 000765
2888 026254 012737 026254 001110 3$: MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
2889 026262 010137 172306 MOV R1,KIPDR3 ;LOAD KERNEL PDR3 WITH 77406 TO CLEAR W-BIT
```

```

2690 026266 012737 026300 000004      MOV    #4$,ERRVEC    ;SET UP LOC. 4 TO 4$ FOR ODD ADDR. ABORT
2691 026274 005237 060001      INC    60001        ;CAUSE ODD ADDRESS ABORT THRU LOC. 4
2692 026300 012706 001100      4$:   MOV    #KERSTK,KSP ;RESTORE THE STACK POINTER
2693 026304 013702 172306      MOV    #IPDR3,R2    ;READ KIPDR3 INTO R2
2694 026310 020102      CMP    R1,R2        ;WAS W-BIT LEFT CLEARED?
2695 026312 001401      BEQ    5$           ;BRANCH IF YES
2696 026314 104025      ERROR  25           ;W-BIT GOT SET DURING AN ODD ADDR. ABORT
2697                                ;FOR TIGHTER SCOPE LOOP
2698                                ;REPLACE ERROR CALL WITH
2699                                ;"BR 3$" = 000757
2900 026316 012737 002332 000004 5$:   MOV    #TIMERR,ERRVEC ;RESTORE NORMAL CPU TRAP ROUTINE TO LOC.4
2901 026324 012737 026210 001110      MOV    #1$, $LPERR  ;RESET LOOP ON ERROR POINTER TO 1$
2902 026332 004737 036036      JSR    PC,TON       ;TURN T-BIT TRAPPING BACK ON
2903
2904                                ;*****
2905                                ;*
2906                                ;*   THE NEXT THREE (3) TESTS CAUSE MEMORY MANAGEMENT ERRORS
2907                                ;*   TO CHECK THE ABILITY OF STATUS REGISTER 0 TO RECORD KT
2908                                ;*   ERRORS AND THE ABILITY OF STATUS REGISTER 2 TO LOCK UP THE
2909                                ;*   VIRTUAL ADDR. OF THE INSTRUCTION THAT CAUSED THE ERROR.
2910                                ;*   THE BITS OF SR2 ARE CHECKED AND BITS <15:13>, <6:5>, AND <3:0>
2911                                ;*   ARE CHECKED IN SRO. SO THE SRO AND SR2 LOGIC AND THE
2912                                ;*   KT ERROR LOGIC ARE CHECKED.
2913                                ;*
2914                                ;*****
2915
2916                                ;*****
2917                                ;* TEST 33   NON-RESIDENT ABORT TEST (ACF=0&4)
2918                                ;*
2919                                ;*   THIS TEST CHECKS THE ACCESS CONTROL FIELD (ACF) COMPARATOR
2920                                ;*   LOGIC BY CAUSING NON-RESIDENT ABORTS IN BOTH KERNEL AND
2921                                ;*   USER MODES. PDR 4 IS LOADED WITH ACF'S = 0&4 AND
2922                                ;*   THEN PHYSICAL ADDR. 60000 IS ACCESSED TO CAUSE THE ABORT.
2923                                ;*
2924                                ;*****
2925                                ;* TST33: SCOPE
2926
2927 026340 012700 000600      1$:   MOV    #600,R0    ;LOAD DATA FOR PAR'S INTO R0
2928 026344 010037 172346      MOV    R0,KIPAR3    ;MAP KERNEL PAR'S 3&4 TO 12-16K
2929 026350 010037 172350      MOV    R0,KIPAR4
2930 026354 010037 177646      MOV    R0,UIPAR3    ;MAP USER PAR'S 3&4 TO 12-16K
2931 026360 010037 177650      MOV    R0,UIPAR4
2932 026364 012737 077406 172306      MOV    #77406,KIPDR3 ;MAP KERNEL PDR 3 128 BLKS, READ-WRITE
2933 026372 012737 077406 177606      MOV    #77406,UIPDR3 ;MAP USER PDR 3 128 BLKS, READ-WRITE
2934 026400 012700 060000      MOV    #60000,R0    ;LOAD VIRTUAL ADDR. TO REFERENCE PDR3 INTO R0
2935 026404 012701 100000      MOV    #100000,R1   ;LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
2936 026410 012703 100011      MOV    #100011,R3   ;LOAD R3 WITH WHAT SRO SHOULD READ - N.R., KERNEL, PG.4
2937 026414 012702 077400      MOV    #77400,R2    ;LOAD ACF=0 (NON-RESIDENT) PDR VALUE IN R2
2938 026420 012737 026462 000250 25$:  MOV    #5$,MMVEC    ;POINT MEM. MGMT. TRAP VECTOR TO 5$ BELOW
2939 026426 010237 172310      MOV    R2,KIPDR4   ;LOAD ACF TEST VALUE INTO KIPDR4
2940 026432 010237 177610      MOV    R2,UIPDR4   ;LOAD ACF TEST VALUE INTO UIPDR4
2941 026436 012737 026444 001110      MOV    #3$, $LPERR  ;SET LOOP ON ERROR POINTER TO 3$
2942 026444 005010      CLR    (R0)        ;CLEAR PHYS. LOC. 60000 USING PDR3
2943 026446 013737 177776 001176      MOV    PSW,$TMP0   ;SAVE PSW IN CASE OF ERROR
2944 026454 005211      4$:   INC    (R1)        ;TRY TO REF. IT USING PDR4 - SHOULD TRAP TO 5$
2945 026456 104026      ERROR  26           ;MEM. MGMT. ABORT DID NOT OCCUR
    
```

```

2946                                ;FOR TIGHTER SCOPE LOOP
2947                                ;REPLACE ERROR CALL WITH
2948                                ;"BR 3$" = 000772
2949 026460 000425      BR    8$           ;BRANCH AROUND STATUS REG. CHECKS IF NO ABORT
2950 026462 062706 000004 5$:   ADD    #4, SP      ;RESTORE STACK POINTER
2951 026466 005710      TST   (R0)        ;DID INSTRUCTION GET ABORTED & NOT EXECUTE
2952 026470 001401      BEQ    6$           ;BRANCH IF YES
2953 026472 104027      ERROR  27         ;INSTRUCTION WAS NOT ABORTED, LOC. GOT CHANGED
2954                                ;FOR TIGHTER SCOPE LOOP
2955                                ;REPLACE ERROR CALL WITH
2956                                ;"BR 3$" = 000764
2957 026474 013737 177572 001366 6$:   MOV    SRO,WASSRO  ;READ STATUS REGISTER 0
2958 026502 013737 177576 001370      MOV    SR2,WASSR2  ;READ STATUS REGISTER 2
2959 026510 020337 001366      CMP    R3,WASSR0   ;DID SRO REPORT NON-RESIDENT ERROR CORRECTLY?
2960 026514 001401      BEQ    7$           ;BRANCH IF YES
2961 026516 104030      ERROR  30         ;SRO DID NOT REPORT NON-RES. ERROR CORRECTLY
2962                                ;FOR TIGHTER SCOPE LOOP
2963                                ;REPLACE ERROR CALL WITH
2964                                ;"BR 3$" = 000752
2965 026520 012704 026454 7$:   MOV    #4$,R4      ;LOAD P4 WITH WHAT SR2 SHOULD READ
2966 026524 020437 001370      CMP    R4,WASSR2  ;DID SR2 LOCKUP RIGHT VIRTUAL ADDR. (=4$)?
2967 026530 001401      BEQ    8$           ;BRANCH IF YES
2968 026532 104031      ERROR  31         ;SR2 DID NOT LOCK VIRTUAL ADDR. OF NON-RES. ERROR
2969                                ;FOR TIGHTER SCOPE LOOP
2970                                ;REPLACE ERROR CALL WITH
2971                                ;"BR 3$" = 000744
2972 026534 042737 160000 177572 8$:   BIC    #160000,SRO ;CLEAR THE ERROR BITS IN SRO
2973 026542 032737 140000 001176      BIT    #140000,$TMP0 ;HAS ACF=0&4 BEEN TESTED IN USER YET
2974 026550 001006      BNE    9$           ;BRANCH IF YES
2975 026552 012703 100151      MOV    #100151,R3  ;LOAD R3 WITH WHAT SRO SHOULD READ - N.R., USER, PG.4
2976 026556 012737 140000 177776      MOV    #140000,PSW ;GO TO USER MODE
2977 026564 000715      BR    2$           ;REPEAT TEST IN USER MODE
2978 026566 022702 077404 9$:   CMP    #77404,R2  ;HAS ACF=4 BEEN TESTED YET?
2979 026572 001407      BEQ    10$          ;BRANCH IF YES
2980 026574 012702 077404      MOV    #77404,R2  ;THEN LOAD ACF=4 (NON-RES) PDR VALUE IN R2
2981 026600 012703 100011      MOV    #100011,R3 ;LOAD R3 WITH WHAT SRO SHOULD READ-N.R.,KERNEL,PG. 4
2982 026604 005037 177776      CLR    PSW        ;GO BACK TO KERNEL MODE
2983 026610 000703      BR    2$           ;GO BACK & TEST ACF=4 IN SAME MODE
2984 026612 005037 177776 10$:  CLR    PSW        ;GO BACK TO KERNEL MODE BEFORE LEAVING
2985 026616 012737 026340 001110      MOV    #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
2986 026624 012737 002400 000250      MOV    #MGMERR,MMVEC ;RESTORE ADDRESS OF NORMAL MEMORY
2987                                ;MANAGEMENT ERROR ROUTINE TO MMVEC
2988
2989                                ;*****
2990                                ;* TEST 34   READ-ONLY ABORT TEST (ACF=2)
2991                                ;*
2992                                ;*   THIS TEST CHECKS THE ACCESS CONTROL FIELD (ACF) COMPARATOR
2993                                ;*   LOGIC BY CAUSING READ-ONLY ABORTS IN BOTH KERNEL AND
2994                                ;*   USER MODES. PDR 4 IS LOAD WITH ACF=2 AND THEN
2995                                ;*   PHYSICAL ADDR. 60000 IS WRITTEN TO CAUSE THE ABORT.
2996                                ;*
2997                                ;*****
2998                                ;* TST34: SCOPE
2999                                ;*
3000                                ;*   KERNEL & USER PAR'S 3 & 4 AND PDR 3
3001 026634 012700 060000      1$:   MOV    #60000,R0  ;ARE SETUP FROM LAST TEST
    
```

```

3002 026640 012701 100000      MOV    #100000,R1      ;LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
3003 026644 012703 020011      MOV    #20011,R3      ;LOAD R3 WITH WHAT SRO SHOULD READ - R/O, KERNEL, PG.4
3004 026650 012702 077402      MOV    #77402,R2      ;LOAD ACF=2 (READ-ONLY) PDR VALUE IN R2
3005 026654 012737 026716 000250 2$:  MOV    #5$,MMVEC      ;POINT MEM. MGMT. TRAP VECTOR TO 5$ BELOW
3006 026662 010237 172310      MOV    R2,KIPDR4      ;LOAD ACF=2 INTO KIPDR4
3007 026666 010237 177610      MOV    R2,UIPDR4      ;LOAD ACF=2 INTO UIPDR4
3008 026672 012737 026700 001110      MOV    #3$,LPERR      ;SET LOOP ON ERROR POINTER TO 3$
3009 026700 005010      CLR    (R0)           ;CLEAR PHYS. LOC. 60000 USING PDR3
3010 026702 013737 177776 001176      MOV    PSW,$TMP0      ;SAVE PSW IN CASE OF ERROR
3011 026710 005211      INC    (R1)           ;TRY TO WRITE USING PDR4 - SHOULD TRAP TO 5$
3012 026712 104026      ERROR  26            ;MEM. MGMT. ABORT DID NOT OCCUR
3013
3014
3015
3016 026714 000425      BR     8$             ;FOR TIGHTER SCOPE LOOP
3017 026716 062706 000004      ADD    #4,SP          ;REPLACE ERROR CALL WITH
3018 026722 005710      TST   (R0)           ;"BR 3$" = 000772
3019 026724 001401      BEQ   6$             ;BRANCH AROUND STATUS REG. CHECKS IF NO ABORT
3020 026726 104027      ERROR  27            ;RESTORE STACK POINTER
3021
3022
3023
3024 026730 013737 177572 001366 6$:  MOV    SRO,WASSR0     ;DID INSTRUCTION GET ABORTED & NOT EXECUTE
3025 026736 013737 177576 001370      MOV    SR2,WASSR2     ;BRANCH IF YES
3026 026744 020337 001366      CMP    R3,WASSR0     ;INSTRUCTION WAS NOT ABORTED, LOC. GOT CHANGED
3027 026750 001401      BEQ   7$             ;FOR TIGHTER SCOPE LOOP
3028 026752 104030      ERROR  30            ;REPLACE ERROR CALL WITH
3029
3030
3031
3032 026754 012704 026710 7$:  MOV    #4$,R4         ;"BR 3$" = 000764
3033 026760 020437 001370      CMP    R4,WASSR2     ;READ STATUS REG. 0
3034 026764 001401      BEQ   8$             ;READ STATUS REG. 2
3035 026766 104031      ERROR  31            ;DID SRO REPORT READ-ONLY ERROR CORRECTLY?
3036
3037
3038
3039 026770 042737 160000 177572 8$:  BIC    #160000,SRO    ;BRANCH IF YES
3040 026776 032737 140000 001176      BIT    #140000,$TMP0  ;SR2 DID NOT LOCKUP VIRTUAL ADDR. OF R/O ERROR
3041 027004 001006      BNE   9$             ;FOR TIGHTER SCOPE LOOP
3042 027006 012703 020151      MOV    #20151,R3     ;REPLACE ERROR CALL WITH
3043 027012 012737 140000 177776      MOV    #140000,PSW    ;"BR 3$" = 000744
3044 027020 000715      BR     2$            ;CLEAR THE ERROR BITS IN SRO
3045 027022 005037 177776 9$:  CLR    PSW           ;HAS ACF=2 BEEN TESTED IN USER MODE?
3046 027026 012737 026634 001110      MOV    #1$,LPERR     ;BRANCH IF YES
3047 027034 012737 002400 000250      MOV    #MGMERR,MMVEC ;LOAD R3 WITH WHAT SRO SHOULD READ-R/O, USER, PG.4
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057

```

```

3058
3059
3060 027042 000004      TST35: SCOPE
3061 027044 012737 027060 001110 1$:  MOV    #2$,LPERR      ;SET LOOP ON ERROR POINTER TO 2$
3062 027052 012737 027074 000250      MOV    #3$,MMVEC     ;LOAD MEM. MGMT. TRAP VECTOR WITH 3$
3063 027060 012737 040000 177776 2$:  MOV    #40000,PSW    ;SET 01 :N PSW CURRENT MODE BITS
3064 027066 000240      NOP
3065 027070 104056      ERROR  56            ;FETCH OF THIS INSTRUCTION SHOULD CAUSE ABORT
3066
3067
3068
3069 027072 000424      BR     5$             ;ILLEGAL MODE 01 NOT ABORTED
3070 027074 012706 001100 3$:  MOV    #KERSTK,SP     ;FOR A TIGHTER SCOPE LOOP
3071 027100 012701 100043      MOV    #100043,R1    ;REPLACE ERROR CALL WITH
3072 027104 013737 177572 001366      MOV    SRO,WASSR0    ;"BR 2$" = 000773
3073
3074
3075
3076 027112 020137 001366      CMP    R1,WASSR0     ;BRANCH AROUND SRO & SR2 CHECKS
3077 027116 001401      BEQ   4$             ;RESTORE STACK POINTER
3078 027120 104057      ERROR  57            ;LOAD EXPECTED CONTENTS OR SRO INTO R1
3079
3080
3081
3082 027122 012701 027060 4$:  MOV    #2$,R1         ;READ CONTENTS OF SRO
3083 027126 013737 177576 001370      MOV    SR2,WASSR2    ;DID SRO REPORT ILLEGAL MODE CORRECTLY?
3084 027134 020137 001370      CMP    R1,WASSR2     ;BRANCH IF YES
3085 027140 001401      BEQ   5$             ;SRO DID NOT REPORT NR ABORT, PG=1, MODE=01
3086 027142 104036      ERROR  36            ;FOR TIGHTER SCOPE LOOP
3087
3088
3089
3090 027144 042737 160000 177572 5$:  BIC    #160000,SRO    ;REPLACE ERROR CALL WITH
3091 027152 012737 002400 000250      MOV    #MGMERR,MMVEC ;"BR 2$" = 000757
3092 027160 012737 027044 001110      MOV    #1$,LPERR     ;LOAD EXPECTED CONTENTS OF SR2 INTO R1
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113

```

```

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

```

```

3114 ;* ABORT DOES OCCUR TO SEE THAT THE ABORT IS REPORTED AND THAT
3115 ;* THE VIRTUAL ADDRESS OF THE INSTRUCTION THAT CAUSED THE ABORT
3116 ;* IS LOCKED UP.
3117 ;*
3118 ;*****
3119
3120
3121
3122 ;*****
3123 ;*TEST 36 PAGE LENGTH FAULTS-UPWARD EXPANSION
3124 ;*
3125 ;* THIS TEST VARIES THE PAGE LENGTH FIELD (PLF) IN KERNEL PDR 4
3126 ;* FROM 1 TO 177 AND FOR EACH PLF, THREE VIRTUAL ADDRESSES (VBA'S)
3127 ;* ARE ACCESSED. WHEN VBA <12:6> IS LESS THAN OR EQUAL TO PDR <14:8>
3128 ;* NO ABORT SHOULD OCCUR. WHEN VBA <12:6> IS GREATER THAN PDR <14:8>,
3129 ;* A PAGE LENGTH ABORT SHOULD OCCUR AND BE REPORTED BY SRO & SR2.
3130 ;* THE PAGE EXPANSION DIRECTION IN THIS TEST IS UPWARD, (THE ED BIT
3131 ;* (BIT 3) OF PDR 4 = 0).
3132 ;*
3133 ;*****
3134 TST36: SCOPE
3135 027166 000004 1$: MOV #77406,KIPDR3 ;MAKE SURE PDR3 IS DESCRIBED AS R/W
3136 027176 012737 077406 172306 172312 MOV #77406,KIPDR5 ;MAKE SURE PDR5 IS DESCRIBED AS R/W
3137 027204 012700 100000 MOV #100000,R0 ;LOAD VIRTUAL ADDR. TO SELECT PDR4 INTO R0
3138 027210 012704 000406 MOV #406,R4 ;LOAD FIRST PDR VALUE IN R4 (PLF=1,ACF=6)
3139 027214 012737 027406 000250 2$: MOV #9$,MMVEC ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS
3140 027222 010437 172310 MOV R4,KIPDR4 ;LOAD KIPDR4 WITH PAGE LENGTH VALUE
3141 027226 012737 027234 001110 MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
3142 027234 012706 001100 MOV #KERSTK,KSP ;MAKE SURE STACK POINTER IS ALL SET UP
3143 027240 011001 MOV (R0),R1 ;ACCESS VIRTUAL ADDR. (VBA < PLF - NO ABORT)
3144 027242 062700 000100 ADD #100,R0 ;FORM NEXT VIRTUAL ADDRESS IN R0
3145 027246 012737 027254 001110 MOV #4$, $LPERR ;SET LOOP ON ERROR POINTER TO 4$
3146 027254 012706 001100 MOV #KERSTK,KSP ;MAKE SURE STACK POINTER IS ALL SET UP
3147 027260 011001 MOV (R0),R1 ;ACCESS VIRTUAL ADDR. (VBA=PLF - NO ABORT)
3148 027262 062700 000100 ADD #100,R0 ;FORM NEXT VIRTUAL ADDR IN R0
3149 027266 020027 117700 CMP R0,#117700 ;HAVE ALL PLF'S BEEN TESTED YET?
3150 027272 001470 BEQ 10$ ;BRANCH IF ALL VBA'S & PLF'S HAVE BEEN USED
3151 027274 012737 027310 001110 MOV #5$, $LPERR ;SET LOOP ON ERROR POINTER TO 5$
3152 027302 012737 027316 000250 MOV #6$,MMVEC ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT
3153 027310 011001 MOV (R0),R1 ;ACCESS VIRTUAL ADDR. (VBA > PLF = ABORT TO 6$)
3154 027312 104033 ERROR 33 ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
3155 ;*
3156 ;* FOR TIGHTER SCOPE LOOP
3157 ;* REPLACE ERROR CALL WITH
3158 ;* "BR 5$" = 000776
3159 027314 000424 BR 8$ ;BRANCH AROUND ABORT CHECKS
3160 027316 012706 001100 6$: MOV #KERSTK,KSP ;RESTORE STACK POINTER FOLLOWING ABORT
3161 027322 013737 177572 001366 MOV SRO,WASSR0 ;READ M.M. STATUS REG. 0
3162 027330 013737 177576 001370 MOV SR2,WASSR2 ;READ M.M. STATUS REG. 2
3163 027336 012702 040011 MOV #40011,R2 ;PUT EXPECTED SRO CONTENTS IN R2
3164 027342 020237 001366 MOV R2,WASSR0 ;DID SRO REPORT PG. LENGTH ABORT, PAGE 4, KERNEL?
3165 027346 001401 BEQ 7$ ;BRANCH IF YES
3166 027350 104034 ERROR 34 ;SRO DID NOT REPORT PG. LENGTH ABORT CORRECTLY
3167 ;*
3168 ;* FOR TIGHTER SCOPE LOOP
3169 027352 012703 027310 7$: MOV #5$,R3 ;REPLACE ERROR CALL WITH
; "BR 5$" = 000757
; PUT EXPECTED SR2 CONTENTS IN R3

```

```

3170 027356 020337 001370 CMP R3,WASSR2 ;DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?
3171 027362 001401 BEQ 8$ ;BRANCH IF YES
3172 027364 104035 ERROR 35 ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY
3173 ;*
3174 ;* FOR TIGHTER SCOPE LOOP
3175 ;* REPLACE ERROR CALL WITH
3176 ;* "BR 5$" = 000751
3177 027366 042737 160000 177572 8$: BIC #160000,SRO ;CLEAR ERROR BITS IN SRO
3178 027374 062704 000400 ADD #400,R4 ;FORM NEXT PLF VALUE FOR KIPDR4
3179 027400 162700 000100 SUB #100,R0 ;FORM FIRST VIRT. ADDR FOR THAT PLF VALUE
3180 027404 000703 BR 2$ ;BRANCH BACK AND ACCESS 3 VBA'S FOR
; THE PLF VALUE JUST FORMED
3181 027406 012637 001356 9$: MOV (KSP)+,TRAPPC ;SAVE PC & PS OF TRAP
3182 027412 012637 001360 MOV (KSP)+,TRAPPS
3183 027416 013737 177572 001366 MOV SRO,WASSR0 ;SAVE CONTENTS OF SRO FOR ERROR
3184 027424 013737 177576 001370 MOV SR2,WASSR2 ;SAVE CONTENTS OF SR2 FOR ERROR
3185 027432 042737 160000 177572 BIC #160000,SRO ;CLEAR ERROR BITS IN SRO
3186 027440 104032 ERROR 32 ;GET PG. LENGTH ABORT BEFORE IT WAS EXPECTED
3187 ;*
3188 ;* FOR TIGHTER SCOPE LOOP
3189 ;* REPLACE ERROR CALL WITH
3190 ;* "A NOP" = 000240
3191 027442 013746 001360 MOV TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
3192 027446 013746 001356 MOV TRAPPC,-(KSP)
3193 027452 000002 RTI ;RETURN FROM UNEXPECTED ABORT
3194 027454 012737 027170 001110 10$: MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
3195 027462 012737 002400 000250 MOV #MGMERR,MMVEC ;RESTORE NORMAL M.M. TRAP HANDLER
3196 ;*
3197 ;* ADDRESS TO M.M. TRAP VECTOR
3198
3199 ;*****
3200 ;*TEST 37 PAGE LENGTH FAULTS-DOWNWARD EXPANSION
3201 ;*
3202 ;* THIS TEST VARIES THE PAGE LENGTH FIELD (PLF) IN KERNEL PDR4
3203 ;* FROM 176 TO 0 AND FOR EACH PLF, THREE VIRTUAL ADDRESSES (VBA'S)
3204 ;* ARE ACCESSED. WHEN VBA <12:6> IS GREATER THAN OR EQUAL TO PDR <14:8>
3205 ;* NO PAGE ABORT SHOULD OCCUR. WHEN VBA <12:6> IS LESS THAN PDR <14:8>
3206 ;* A PAGE LENGTH ABORT SHOULD OCCUR AND BE REPORTED BY SRO & SR2.
3207 ;* THE PAGE EXPANSION DIRECTION IN THIS TEST IS DOWNWARD, (THE ED BIT
3208 ;* (BIT 3) OF PDR4=1).
3209 ;*
3210 ;*****
3211 TST37: SCOPE
3212 027470 000004 1$: MOV #117700,R0 ;LOAD VIRTUAL ADDR. TO SELECT PDR4 INTO R0
3213 027472 012700 117700 MOV #77016,R4 ;LOAD FIRST PDR VALUE IN R4 (PLF=176,ACF=6)
3214 027502 012737 027674 000250 2$: MOV #9$,MMVEC ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS
3215 027510 010437 172310 MOV R4,KIPDR4 ;LOAD KIPDR4 WITH PAGE LENGTH VALUE
3216 027514 012737 027522 001110 MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
3217 027522 012706 001100 MOV #KERSTK,KSP ;MAKE SURE STACK POINTER IS ALL SET UP
3218 027526 011001 MOV (R0),R1 ;ACCESS VIRTUAL ADDR. (VBA > PLF - NO ABORT)
3219 027530 162700 000100 SUB #100,R0 ;FORM NEXT VIRTUAL ADDRESS IN R0
3220 027534 012737 027542 001110 MOV #4$, $LPERR ;SET LOOP ON ERROR POINTER TO 4$
3221 027542 012706 001100 MOV #KERSTK,KSP ;MAKE SURE STACK POINTER IS ALL SET UP
3222 027546 011001 MOV (R0),R1 ;ACCESS VIRTUAL ADDR. (VBA=PLF - NO ABORT)
3223 027550 162700 000100 SUB #100,R0 ;FORM NEXT VIRTUAL ADDR. IN R0
3224 027554 020027 100000 CMP R0,#100000 ;HAVE ALL PLF'S BEEN TESTED YET?
3225 027560 001470 BEQ 10$ ;BRANCH IF ALL VBA'S & PLF'S HAVE BEEN USED

```

```

3226 027562 012737 027576 001110 MOV #5$, $LPERR ;SET LOOP ON ERROR POINTER TO 5$
3227 027570 012737 027604 000250 MOV #6$, MMVEC ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT
3228 027576 011001 MOV (R0), R1 ;ACCESS VIRTUAL ADDR. (VBA < PLF - ABORT TO 6$)
3229 027600 104033 ERROR J3 ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
3230 ;FOR TIGHTER SCOPE LOOP
3231 ;REPLACE ERROR CALL WITH
3232 ;"BR 5$" = 000776
3233 027602 000424 BR 8$ ;BRANCH AROUND ABORT CHECKS
3234 027604 012706 001100 MOV #KERSTK, KSP ;RESTORE STACK POINTER FOLLOWING ABORT
3235 027610 013737 177572 001366 MOV SRO, WASSR0 ;READ M.M. STATUS REG. 0
3236 027616 013737 177576 001370 MOV SR2, WASSR2 ;READ M.M. STATUS REG. 2
3237 027624 012702 040011 MOV #40011, R2 ;PUT EXPECTED SRO CONTENTS IN R2
3238 027630 020237 001366 CMP R2, WASSR0 ;DID SRO REPORT PG. LENGTH ABORT, PG. 4, KERNEL?
3239 027634 001401 BEQ 7$ ;BRANCH IF YES
3240 027636 104034 ERROR 34 ;SRO DID NOT REPORT PG. LENGTH ABORT CORRECTLY
3241 ;FOR TIGHTER SCOPE LOOP
3242 ;REPLACE ERROR CALL WITH
3243 ;"BR 5$" = 000757
3244 027640 012703 027576 7$: MOV #5$, R3 ;PUT EXPECTED SR2 CONTENTS IN R3
3245 027644 020337 001370 CMP R3, WASSR2 ;DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?
3246 027650 001401 BEQ 8$ ;BRANCH IF YES
3247 027652 104035 ERROR 35 ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY
3248 ;FOR TIGHTER SCOPE LOOP
3249 ;REPLACE ERROR CALL WITH
3250 ;"BR 5$" = 000751
3251 027654 042737 160000 177572 8$: BIC #160000, SRO ;CLEAR ERROR BITS IN SRO
3252 027662 162704 000400 SUB #400, R4 ;FORM NEXT PLF VALUE FOR KIPDR4
3253 027666 062700 000100 ADD #100, R0 ;FORM FIRST VIRT. ADDR. FOR THAT PLF VALUE
3254 027672 000703 BR 2$ ;BRANCH BACK AND ACCESS 3 VBA'S FOR
3255 ;THE PLF VALUE JUST FORMED
3256 027674 012637 001356 9$: MOV (KSP)+, TRAPPC ;SAVE PC & PS OF TRAP
3257 027700 012637 001360 MOV (KSP)+, TRAPPS
3258 027704 013737 177572 001366 MOV SRO, WASSR0 ;SAVE CONTENTS OF SRO FOR ERROR
3259 027712 013737 177576 001370 MOV SR2, WASSR2 ;SAVE CONTENTS OF SR2 FOR ERROR
3260 027720 042737 160000 177572 BIC #160000, SRO ;CLEAR ERROR BITS IN SRO
3261 027726 104032 ERROR 32 ;GET PG. LENGTH ABORT BEFORE IT WAS EXPECTED
3262 ;FOR TIGHTER SCOPE LOOP
3263 ;REPLACE ERROR CALL WITH
3264 ;"NOP" = 000240
3265 027730 013746 001360 MOV TRAPPS, -(KSP) ;PUT PC & PS OF TRAP ON STACK
3266 027734 013746 001356 MOV TRAPPC, -(KSP)
3267 027740 000002 RTI ;RETURN FROM UNEXPECTED ABORT
3268
3269 027742 012737 027472 001110 10$: MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
3270 027750 012737 002400 000250 MOV #MGMERR, MMVEC ;RESTORE NORMAL M.M. TRAP HANDLER
3271 ;ADDRESS TO M.M. TRAP VECTOR
3272
3273
3274 ;*****
3275 ;*TEST 40 SR2 BIT TEST
3276 ;*
3277 ;* THIS TEST CHECKS THE BITS IN MEMORY MANAGEMENT REGISTER 2 BY
3278 ;* CAUSING "READ-ONLY ABORTS" AT VIRTUAL ADDRESSES BETWEEN 100000
3279 ;* TO 111000 (PHYSICAL ADDRESSES 060000-071000). KIPDR4 IS USED TO EXECUTE
3280 ;* THE FOLLOWING FOUR WORDS OF CODE WHICH ARE MOVED THRU MEMORY:
3281 ;* 010727 MOV PC, (PC)+ ;THIS INSTRUCTION SHOULD CAUSE A R/O ABORT
    
```

```

3282 ;* 000000 ;ITS VIRTUAL ADDR. SHOULD BE LOCKED UP IN SR2
3283 ;* 000137 JMP @#3$ ;THIS INSTRUCTION IS ALSO MOVED THRU MEMORY
3284 ;* (ADDR. OF 3$) ;IN CASE A R/O ABORT DOES NOT OCCUR,
3285 ;* ;IN WHICH CASE SR2 WILL NOT CONTAIN CORRECT ADDR.
3286 ;*
3287 ;*****
3288 027756 000004 TST40: SCOPE
3289 027760 012737 000600 172346 1$: MOV #600, KIPAR3 ;BE SURE PAR3 IS MAPPED TO 12-16K
3290 027766 012737 000600 172350 MOV #600, KIPAR4 ;BE SURE PAR4 IS MAPPED TO 12-16K
3291 027774 012737 077406 172306 MOV #77406, KIPDR3 ;MAP PAGE 3 128 BLOCKS, R/W
3292 030002 012737 077402 172310 MOV #77402, KIPDR4 ;MAP PAGE 4 128 BLOCKS, READ-ONLY
3293 030010 012700 060000 MOV #60000, R0 ;LOAD R0 WITH VIRTUAL ADDR. WHICH USES PDR3
3294 030014 012701 100000 MOV #100000, R1 ;LOAD R1 WITH VIRTUAL ADDR. WHICH USES PDR4
3295 030020 012737 030054 000250 MOV #3$, MMVEC ;SET M.M. TRAP VECTOR TO 3$
3296 030026 012737 030034 001110 MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
3297 030034 012720 010727 2$: MOV #010727, (R0)+ ;LOAD "MOV PC, (PC)+" INSTRUCTION AT ADDR.
3298 030040 005020 CLR (R0)+ ; REACHED THRU PDR/PAR 4.
3299 030042 012720 000137 MOV #000137, (R0)+ ; LOAD "JMP @#3$" INSTRUCTION AT VIRT. ADDR.
3300 030046 012710 030054 MOV #3$, (R0) ; IN CASE R/O VIOL. DOES NOT ABORT
3301 030052 010107 MOV R1, PC ;TRANSFER PROGRAM EXECUTION TO "PAGE 4 INSTRUCTIONS"
3302 030054 012706 001100 MOV #KERSTK, KSP ;RESTORE STACK POINTER
3303 030060 013737 177576 001370 MOV SR2, WASSR2 ;READ CONTENTS OF STATUS REG 2
3304 030066 020137 001370 CMP R1, WASSR2 ;WAS ADDR. OF "RELOCATED - R/O ABORT" LOCKED UP?
3305 030072 001401 BEQ 4$ ;BRANCH IF YES
3306 030074 104036 ERROR 36 ;SR2 DID NOT LOCK UP VIRTUAL ADDR. OF R/O VIOL.
3307 ;FOR TIGHTER SCOPE LOOP
3308 ;REPLACE ERROR CALL WITH
3309 ;"BR 2$" = 000757
3310 030076 042737 160000 177572 4$: BIC #160000, SRO ;CLEAR THE ERROR BITS IN SRO
3311 030104 162700 000004 SUB #4, R0 ;RESET R0 TO POINT TO NEXT VIRT. ADDR. TO LOAD
3312 030110 062701 000002 ADD #2, R1 ;FORM VIRTUAL ADDR. THAT SHOULD BE LOCKED UP NEXT
3313 030114 020127 111002 CMP R1, #111002 ;HAVE ALL VBA'S 100000-111000 BEEN TESTED?
3314 030120 103745 BLD 2$ ;BRANCH IF NO
3315
3316 030122 012737 027760 001110 5$: MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
3317 030130 012737 077406 172310 MOV #77406, KIPDR4 ;RESTORE PDR4 TO R/W ACCESS
3318 030136 012737 002400 000250 MOV #MGMERR, MMVEC ;RESTORE ADDRESS OF NORMAL M.M.
3319 ;TRAP HANDLER TO M.M. VECTOR
3320
3321
    
```

```

3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338 030144 000004
3339 030146 012737 000600 172352 1$: MOV #600,KIPAR5 ;MAP KERNEL PAGE 5 TO 12-16K
3340 030154 012737 000406 172310 MOV #406,KIPDR4 ;SETUP PDR4 FOR PAGE LENGTH ABORT
3341 030162 012737 077402 172312 MOV #77402,KIPDR5 ;SETUP PDR5 FOR R/O ABORT
3342 030170 012737 030176 001110 MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
3343 030176 013737 177576 001370 2$: MOV SR2,WASSR2 ;READ SR2 TO SEE IF ITS TRACKING
3344 030204 012701 030176 MOV #2$,R1 ;PUT EXPECTED VIRTUAL PC IN R1
3345 030210 020137 001370 CMP #1,WASSR2 ;DID SR2 CONTAIN VIRTUAL PC AT 25?
3346 030214 001401 BEQ 3$ ;BRANCH IF YES
3347 030216 104041 ERROR 41 ;SR2 NOT TRACKING CORRECTLY
3348 ;FOR TIGHTER SCOPE LOOP
3349 ;REPLACE ERROR CALL WITH
3350 ;"BR 2$" = 000767
3351 030220 012737 030226 001110 3$: MOV #4$, $LPERR ;SET LOOP ON ERROR POINTER TO 4$
3352 030226 013737 177576 001370 4$: MOV SR2,WASSR2 ;READ SR2 TO SEE IF ITS TRACKING
3353 030234 012701 030226 MOV #4$,R1 ;PUT EXPECTED VIRTUAL PC IN R1
3354 030240 020137 001370 CMP R1,WASSR2 ;DID SR2 CONTAIN VIRTUAL PC AT 4$
3355 030244 001401 BEQ 5$ ;BRANCH IF YES
3356 030246 104041 ERROR 41 ;SR2 NOT TRACKING CORRECTLY
3357 ;FOR TIGHTER SCOPE LOOP
3358 ;REPLACE ERROR CALL WITH
3359 ;"BR 4$" = 000767
3360 030250 012737 030256 001110 5$: MOV #6$, $LPERR ;SET LOOP ON ERROR POINTER TO 6$
3361 030256 012737 030274 000250 6$: MOV #7$,MMVEC ;PUT ADDRESS OF 7$ IN M.M. TRAP VECTOR
3362 030264 050537 001200 CLR $TMP1 ;CLEAR ERROR INDICATOR
3363 030270 05237 100500 INC @#100500 ;CAUSE PAGE LENGTH ABORT - TRAP TO 7$
3364 030274 012706 001100 7$: MOV #KERSTK,KSP ;RESTORE STACK POINTER AFTER ABORT
3365 030300 013737 177572 001176 MOV SRO,$TMP0 ;SAVE SRO'S INFORMATION ON PG. LGTH. ABORT
3366 030306 013737 177576 001202 MOV SR2,$TMP2 ;SAVE SR2'S INFORMATION ON PG. LGTH. ABORT
3367 030314 012737 030326 000250 MOV #8$,MMVEC ;PUT ADDRESS OF 8$ IN M.M. TRAP VECTOR
3368 030322 05237 120000 INC @#120000 ;CAUSE R/O ABORT - TRAP TO 8$
3369 030326 012706 001100 8$: MOV #KERSTK,KSP ;RESTORE STACK POINTER AFTER ABORT
3370 030332 013737 177572 001366 MOV SRO,WASSR0 ;READ SRO FOLLOWING SECOND KT ABORT
3371 030340 013737 177576 001370 MOV SR2,WASSR2 ;READ SR2 FOLLOWING SECOND KT ABORT
3372 030346 022737 001176 001366 CMP $TMP0,WASSR0 ;IS SRO STILL HOLDING INFO ON FIRST ABORT?
3373 030354 001402 BEQ 9$ ;BRANCH IF YES
3374 030356 05237 001200 INC $TMP1 ;SET ERROR INDICATOR
3375 030362 023737 001202 001370 9$: CMP $TMP2,WASSR2 ;DOES SR2 STILL HOLD PC OF FIRST ABORT?
3376 030370 001402 BEQ 10$ ;BRANCH IF YES
3377 030372 05237 001200 INC $TMP1 ;SET ERROR INDICATOR
    
```

```

3378 030376 005737 001200 10$: TST $TMP1 ;WERE SRO OR SR2 CHANGED BY A SECOND ABORT?
3379 030402 001401 BEQ 11$ ;BRANCH IF NO
3380 030404 104037 ERROR 37 ;ONE OF STATUS REGS. CHANGED BY SECOND ABORT
3381 ;FOR TIGHTER SCOPE LOOP
3382 ;REPLACE ERROR CALL WITH
3383 ;"BR 6$" = 000726
3384 030406 005037 001200 11$: CLR $TMP1 ;CLEAR ERROR INDICATOR
3385 030412 000005 RESET ;EXECUTE A RESET, APPLYING AN "INIT"
3386 030414 013737 177572 001366 MOV SRO,WASSR0 ;READ SRO
3387 030422 005737 001366 TST WASSR0 ;WAS SRO CLEARED BY THE RESET?
3388 030426 001402 BEQ 12$ ;BRANCH IF YES
3389 030430 005237 001200 INC $TMP1 ;SRO NOT CLEARED BY A RESET
3390 030434 013737 177576 001370 12$: MOV SR2,WASSR2 ;READ SR2
3391 030442 022737 030434 001370 CMP #12$,WASSR2 ;WAS SR2 UNLOCKED BY A RESET?
3392 030450 001402 BEQ 13$ ;BRANCH IF YES
3393 030452 005237 001200 INC $TMP1 ;SR2 NOT UNLOCKED BY A RESET
3394 030456 005737 001200 13$: TST $TMP1 ;WERE SRO & SR2 BOTH "RESET" BY A RESET?
3395 030462 001401 BEQ 14$ ;BRANCH IF YES
3396 030464 104040 ERROR 40 ;SRO OR SR2 NOT "RESET" BY A RESET
3397 ;FOR TIGHTER SCOPE LOOP
3398 ;REPLACE ERROR CALL WITH
3399 ;"BR 6$" = 000676
3400 030466 005237 177572 14$: INC SRO ;TURN MEMORY MANAGEMENT BACK ON
3401 030472 013737 177576 001370 15$: MOV SR2,WASSR2 ;READ SR2 TO SEE IF ITS TRACKING AGAIN
3402 030500 012701 030472 MOV #15$,R1 ;PUT EXPECTED VIRTUAL PC IN R1
3403 030504 020137 001370 CMP R1,WASSR2 ;DID SR2 CONTAIN VIRTUAL PC AT 15$
3404 030510 001401 BEQ 16$ ;BRANCH IF YES
3405 030512 104041 ERROR 41 ;SR2 NOT TRACKING CORRECTLY
3406 ;FOR TIGHTER SCOPE LOOP
3407 ;REPLACE ERROR CALL WITH
3408 ;"BR 6$" = 000663
3409
3410 ;*****
3411 ; ALL CODE WITH A '+' DESIGNATION IN THE COMMENT INDICATES THAT IT HAS BEEN
3412 ; ADDED TO THE ORIGINAL REV OF THIS DIAGNOSTIC.THESE ENHANCEMENTS OR CORRECTIONS
3413 ; ARE MADE BY THE PRODUCT ENHANCEMENT GROUP (DIAGNOSTICS).
3414 ;*****
3415
3416
3417
3418
3419 030514 012737 030574 000250 16$: MOV #21$,MMVEC ;+SET UP KIVEC FOR NON-RESIDENT ABORT
3420 030522 012737 077400 172312 MOV #77400,KIPDR5 ;+SET UP KIPDR5 FOR NR ABORT
3421 030530 012737 030536 001110 MOV #17$, $LPERR ;+SET LOOP ON ERROR POINTER
3422 030536 005037 001200 17$: CLR $TMP1 ;+CLEAR ERROR INDICATOR
3423 030542 005237 130000 20$: INC @#130000 ;+CAUSE A NON-RESIDENT ABORT
3424 030546 005237 001200 INC $TMP1 ;+SET ERROR INDICATOR
3425 030552 013737 177572 001366 MOV SRO,WASSR0 ;+READ SRO
3426 030560 013737 177576 001370 MOV SR2,WASSR2 ;+READ SR2
3427 030566 104065 ERROR 65 ;+NON-RESIDENT ABORT DID NOT OCCUR
3428
3429
3430 030570 005737 001200 CLR $TMP1 ;+CLEAR ERROR INDICATOR
3431 030574 012706 001100 21$: MOV #KERSTK,KSP ;+RESTORE KERNEL STACK POINTER
3432 030600 005737 177572 22$: TST SRO ;+TEST FOR THE NR ABORT IN SRO (BIT 15)
3433 030604 100413 BMI 23$ ;+IF SET BRANCH
    
```



```

3434 030606 005237 001200          INC    $TMP1          ;+SET ERROR INDICATOR
3435 030612 013737 177572 001366  MOV    SRO,WASSR0    ;+READ SRO
3436 030620 013737 177576 001370  MOV    SR2,WASSR2    ;+READ SR2
3437 030626 104066          ERROR   66           ;+NR ABORT ERROR DID NOT SET IN SRO (BIT 15)
3438
3439
3440 030630 005037 001200          CLR    $TMP1          ;+CLEAR ERROR INDICATOR
3441 030634 023727 177576 030542 23$:  CMP    SR2,#20$      ;+TEST THAT SR2 FROZE TO THE VIRTUAL
3442          ;+ADDRESS OF THE NR ABORT INSTR.
3443 030642 001413          BEQ    24$           ;+IF EQUAL BRANCH
3444 030644 005237 001200          INC    $TMP1          ;+ELSE ERROR
3445 030650 013737 177572 001366  MOV    SRO,WASSR0    ;+READ SRO
3446 030656 013737 177576 001370  MOV    SR2,WASSR2    ;+READ SR2
3447 030664 104066          ERROR   67           ;+SR2 DID NOT CONTAIN THE CORRECT ADDRESS
3448
3449
3450 030666 005037 001200          CLR    $TMP1          ;+CLEAR ERROR INDICATOR
3451 030672 012737 030732 000250 24$:  MOV    #25$,MMVEC    ;+SET KVEC FOR 2ND NR ABORT
3452 030700 005237 130000          INC    #130000       ;+CAUSE ABORT
3453 030704 005237 001200          INC    $TMP1          ;+SET ERROR INDICATOR
3454 030710 013737 177572 001366  MOV    SRO,WASSR0    ;+READ SRO
3455 030716 013737 177576 001370  MOV    SR2,WASSR2    ;+READ SR2
3456 030724 104070          ERROR   70           ;+2ND NR ABORT DID NOT OCCUR
3457
3458
3459 030726 005037 001200          CLR    $TMP1          ;+CLEAR ERROR INDICATOR
3460 030732 012706 001100          MOV    #KERSTK,KSP   ;+RESTORE KERNEL STACK
3461 030736 013737 177576 001370  MOV    SR2,WASSR2    ;+READ SR2
3462 030744 023727 001370 030542  CMP    WASSR2,#20$   ;+TEST SR2 FOR VIRTUAL ADDRESS OF
3463          ;+OF THE FIRST NON-RESIDENT ABORT
3464 030752 001413          BEQ    26$           ;+IF EQUAL BRANCH
3465          ;+ELSE ERROR
3466 030754 005237 001200          INC    $TMP1          ;+SET ERROR INDICATOR
3467 030760 013737 177572 001366  MOV    SRO,WASSR0    ;+READ SRO
3468 030766 012737 030542 001364  MOV    #20$,CORSR2   ;+READ CORRECT SR2 VALUE
3469 030774 104071          ERROR   71           ;+SR2 DID NOT CONTAIN THE VALUE OF
3470          ;+THE FIRST NR ABORT
3471
3472
3473 030776 005037 001200          CLR    $TMP1          ;+CLEAR ERROR INDICATOR
3474 031002 000005          RESET          ;+ISSUE INIT. CONDITIONS
3475 031004 005737 177572          TST    SRO           ;+SRO SHOULD BE CLEAR
3476 031010 001412          BEQ    27$           ;+IF TRUE BRANCH
3477          ;+ELSE ERROR
3478 031012 005237 001200          INC    $TMP1          ;+SET ERROR INDICATOR
3479 031016 013737 177572 001366  MOV    SRO,WASSR0    ;+READ SRO
3480 031024 005037 001362          CLR    CORSR0        ;+SET CORRECT SRO VALUE FOR ERROR MSG
3481 031030 104072          ERROR   72           ;+SRO DID NOT CLEAR AFTER INIT.
3482
3483
3484
3485 031032 005037 001200          CLR    $TMP1          ;+CLEAR ERROR INDICATOR
3486 031036 005237 177572          INC    SRO           ;+TURN MEM MANG ON
3487 031042 013737 177576 001370 30$:  MOV    SR2,WASSR2    ;+READ SR2
3488 031050 023727 001370 031042  CMP    WASSR2,#30$   ;+IS SR2 TRACKING AGAIN
3489 031056 001405          BEQ    31$           ;+YES, BRANCH
    
```

```

3490
3491 031060 005237 001200          INC    $TMP1          ;+SET ERROR INDICATOR
3492 031064 012701 031042          MOV    #30$,R1       ;+SET CORRECT SR2 VALUE FOR ERROR MSG
3493 031070 104041          ERROR   41           ;+ERROR,SR2 IS NOT TRACKING CORRECTLY
3494
3495
3496 031072 012737 030146 001110 31$:  MOV    #1$, $LPERR    ;RESET LOOP ON ERROR POINTER TO 1$
3497 031100 012737 077406 172310  MOV    #77406,KIPDR4 ;RESET PDR4 TO 128 BLKS. R/W
3498 031106 012737 077406 172312  MOV    #77406,KIPDR5 ;RESET PDR5 TO 128 BLKS. R/W
3499 031114 012737 002400 000250  MOV    #MGMERR,MMVEC ;RESTORE ADDRESS OF NORMAL MEMORY
3500          ;MANAGEMENT TRAP ROUTINE TO M.M. VECTOR
3501
3502
3503          ;*****
3504          ;*TEST 42 USER ABORT PICKS UP KERNEL SPACE VECTOR
3505          ;*
3506          ;* THIS TEST CHECKS TO BE SURE THAT WHEN AN ABORT OCCURS WHILE IN
3507          ;* USER MODE, THE TRAP VECTOR INFORMATION FETCHED IS TAKEN FROM
3508          ;* KERNEL SPACE. USER PAGE 0 IS MAPPED TO 12K (60000-77776) SO
3509          ;* THAT IF USER SPACE IS USED INSTEAD OF KERNEL, THE NEW PC THAT
3510          ;* WAS LOADED AT LOC. 060004 IS USED INSTEAD OF THE NEW PC THAT
3511          ;* SHOULD BE PICKED UP FROM LOC. 000004. AN ODD ADDRESS ERROR IS USED
3512          ;* TO CAUSE A TRAP TO "4".
3513          ;*
3514          ;*****
3515 031122 000004          TST42: SCOPE
3516 031124 004737 036002 001110 1$:  JSR    PC,TOFF        ;TURN OFF T-BIT TRAPPING FOR THIS TEST
3517 031130 012737 031136          MOV    #2$, $LPERR    ;SET LOOP ON ERROR POINTER TO 2$
3518 031136 005037 177776          CLR    PSW           ;GO TO KERNEL MODE
3519 031142 012706 001100          MOV    #KERSTK,KSP   ;SETUP KERNEL STACK PTR.
3520 031146 012737 000600 177640  MOV    #600,UIPAR0    ;MAP USER PAGE 0 TO 12K
3521 031154 012737 031236 000004  MOV    #4$,#4         ;LOAD KERNEL VECTOR 4 (LOC.4) WITH 4$
3522 031162 012737 000340 000006  MOV    #340,#6        ;LOAD VECTOR+2 WITH NEW PSW
3523 031170 012737 140000 177776  MOV    #140000,PSW    ;GO TO USER MODE
3524 031176 012706 000700          MOV    #USESTK,USP   ;SETUP USER STACK PTR.
3525 031202 012737 031222 000004  MOV    #3$,#4         ;LOAD USER VECTOR 4 (LOC. 60004) WITH 3$
3526 031210 012737 000340 000006  MOV    #340,#6        ;LOAD VECTOR+2 WITH NEW PSW
3527 031216 005737 031223          TST    3$+1         ;CAUSE ODD ADDR. ERROR TRAP TO "4"
3528          ;SHOULD PICK UP NEW PC=4$ FROM KERNEL
3529          ;LOC. 4, NOT PC=3$ FROM USER LOC. 4 (+60004)
3530 031222 013701 177776          3$:  MOV    PSW,R1         ;SAVE PSW FOR ERROR
3531 031226 010602          MOV    SP,R2         ;SAVE VALUE OF STACK POINTER FOR ERROR
3532 031230 005037 177776          CLR    PSW           ;BE SURE BACK IN KERNEL MODE
3533 031234 104042          ERROR   42          ;DID NOT TRAP THRU KERNEL SPACE
3534          ;FOR TIGHTER SCOPE LOOP
    
```

```

3535                                     ;REPLACE ERROR CALL WITH
3536                                     ;"BR 2$" = 000740
3537 031236 005037 177776 4$: CLR PSW ;BE SURE BACK IN KERNEL MODE
3538 031242 012706 001100 MOV #KERSTK,KSP ;RESTORE KERNEL S.P. IN CASE IT CHANGED
3539 031246 005037 177640 CLR UIPARO ;REMAP USER PAGE 0 TO 0-4K
3540 031252 012737 140000 177776 MOV #140000,PSW ;GO TO USER MODE
3541 031260 012706 000700 MOV #USESTK,USP ;RESTORE USER STACK POINTER
3542 031264 005037 177776 CLR PSW ;GO BACK TO KERNEL MODE
3543 031270 012737 002332 000004 MOV #TIMERR,@#4 ;RESTORE ADDR. OF NORMAL CPU TRAP HANDLER TO 4
3544 031276 012737 031124 001110 MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
3545 031304 004737 036036 JSR PC,TON ;TURN T-BIT TRAPPING BACK ON
3546
3547 ;*****
3548 ;*TEST 43 RTI IN USER MODE DOES NOT CHANGE PSW
3549 ;*
3550 ;* THIS TEST CHECKS TO SEE THAT WHEN AN RTI IS EXECUTED IN USER
3551 ;* MODE, THE MODE OR PRIORITY BITS OF THE PSW ARE NOT CHANGED.
3552 ;*
3553 ;*****
3554 031310 000004 TST43: SCOPE
3555
3556 031312 012737 031324 001110 1$: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
3557 031320 012702 170000 MOV #170000,R2 ;LOAD "PRESENT & EXPECTED" PSW VALUE INTO R2
3558 031324 010237 177776 2$: MOV R2,PSW ;GO TO USER MODE-PRIORITY 0
3559 031330 012746 000340 MOV #340,-(SP) ;PUT A NEW PSW (PRIORITY=7) ON STACK
3560 031334 012746 031342 MOV #3$,-(SP) ;PUT NEW PC ON THE STACK
3561 031340 000002 RTI ;DO AN RTI FROM USER MODE
3562 031342 013701 177776 3$: MOV PSW,R1 ;READ NEW PSW INTO R1
3563 031346 042701 007437 BIC #7437,R1 ;MASK OFF COND. CODE, T-BIT, AND UNUSED BITS
3564 031352 005037 177776 CLR PSW ;GO BACK TO KERNEL MODE
3565 031356 020201 CMP R2,R1 ;DID PSW STAY IN USER, PRIORITY=0?
3566 031360 001401 BEQ 4$ ;BRANCH IF YES
3567 031362 104060 ERROR 60 ;PSW CHANGED BY AN RTI FROM USER
3568 ;FOR A TIGHTER SCOPE LOOP
3569 ;REPLACE ERROR CALL WITH
3570 ;"BR=2$" = 000760
3571 031364 012737 031312 001110 4$: MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
3572
3573
    
```

```

3574 ;*****
3575 ;*TEST 44 KT ERROR NOT SERVICED IF ODD ADDR. ERROR
3576 ;*
3577 ;* THIS TEST CHECKS TO SEE THAT IF A CERTAIN VIRTUAL ADDRESS THAT
3578 ;* WOULD CAUSE A MEMORY MANAGEMENT ERROR CAUSES AN ODD ADDRESS
3579 ;* ERROR FIRST, THE ODD ADDRESS ERROR IS SERVICED BUT THE MEMORY
3580 ;* MANAGEMENT ERROR ISN'T. THIS MEANS THAT SRO AND SR2
3581 ;* SHOULD NOT REPORT THE ERROR OR LOCK UP ITS VIRTUAL ADDRESS.
3582 ;* A READ-ONLY VIOLATION IS USED AS THE POTENTIAL MEMORY MANAGEMENT
3583 ;* ERROR
3584 ;*
3585 ;*****
3586 031372 000004 TST44: SCOPE
3587 031374 012737 000600 172350 1$: MOV #600,KIPAR4 ;MAP KERNEL PAGE 4 TO 12-16K
3588 031402 012705 077402 MOV #77402,R5 ;LOAD PDR4 DATA INTO R5
3589 031406 010537 172310 MOV R5,KIPDR4 ;MAP PAGE 4 READ-ONLY
3590 031412 012737 031442 000004 MOV #4$,@#4 ;SET CPU TRAP VECTOR TO ADDRESS OF 4$
3591 031420 012737 031440 000250 MOV #3$,@#250 ;SET M.M. TRAP VECTOR TO ADDRESS OF 3$
3592 031426 012737 031434 001110 MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
3593 031434 005237 060001 2$: INC 60001 ;CAUSE ODD ADDR. ERROR & POTENTIAL R/O ABORT
3594 031440 104043 3$: ERROR 43 ;TRAPPED THRU M.M. VECTOR BUT SHOULDN'T HAVE
3595 ;FOR TIGHTER SCOPE LOOP
3596 ;REPLACE ERROR CALL WITH
3597 ;"BR 2$" = 000776
3598 031442 012706 001100 4$: MOV #KERSTK,KSP ;RESTORE STACK POINTER AFTER TRAPPING
3599 031446 005037 001200 CLR $TMP1 ;CLEAR ERROR INDICATOR
3600 031452 013737 177572 001366 MOV SRO,WASSR0 ;READ STATUS REG. 0
3601 031460 013737 177576 001370 5$: MOV SR2,WASSR2 ;READ STATUS REG. 2
3602 031466 012700 000017 MOV #17,R0 ;LOAD EXPECTED SRO CONTENTS INTO R0
3603 031472 020037 001366 CMP R0,WASSR0 ;SRO ERROR BITS LEFT CLEAR BY TRAPPING?
3604 031476 001402 BEQ 6$ ;BRANCH IF YES
3605 031500 005237 001200 INC $TMP1 ;SRO ERROR BITS SET WHEN ODD ADDR. SERVICED
3606 031504 012701 031460 6$: MOV #5$,R1 ;LOAD EXPECTED SR2 CONTENTS INTO R1
3607 031510 020137 001370 CMP R1,WASSR2 ;WAS SR2 LEFT UNLOCKED BY TRAPPING?
3608 031514 001402 BEQ 7$ ;BRANCH IF YES
3609 031516 005237 001200 INC $TMP1 ;SR2 LOCKED UP BY ODD ADDR. ERROR
3610 031522 005737 001200 7$: TST $TMP1 ;WHERE SRO OR SR2 EFFECTED?
3611 031526 001404 BEQ 8$ ;BRANCH IF NO
3612 031530 104044 ERROR 44 ;SRO OR SR2 CHANGED BY ODD ADDR. ERROR
3613 ;FOR TIGHTER SCOPE LOOP
3614 ;REPLACE ERROR CALL WITH
3615 ;"BR 2$" = 000741
3616 031532 042737 160000 177572 BIC #160000,SRO ;CLEAR ERROR BITS THAT MAY BE SET IN SRO
3617 031540 012737 002332 000004 8$: MOV #TIMERR,@#4 ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
3618 031546 012737 002400 000250 MOV #MGMERR,@#250 ;RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER
3619 031554 012737 077406 172310 MOV #77406,KIPDR4 ;REMAP PAGE 4 TO READ/WRITE
3620 031562 012737 031374 001110 MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
3621
3622
3623 ;*****
3624 ;*TEST 45 PC & PSW SAVED FOR KT ERROR DURING SERVICE OF ODD ADDR. ERROR
3625 ;*
3626 ;* THIS TEST CHECKS THE PC AND PROCESSOR STATUS WORD SAVED WHEN
3627 ;* A KT ERROR OCCURS DURING THE SECOND PUSH ON THE STACK DURING
3628 ;* SERVICING OF AN ODD ADDR. ERROR. DURING A "DOUBLE ERROR"
3629 ;* SEQUENCE SUCH AS THIS, THE PSW SAVED WILL BE THE ONE PICKED UP
    
```

```

3630          : * FROM VECTOR+2 (LOC. 6 IN THIS CASE) AFTER THE FIRST TRAP,
3631          : * NOT THE PSW PRESENT BEFORE THE FIRST TRAP. SRO AND SR2
3632          : * SHOULD RECORD THE KT ERROR (A R/O VIOLATION BY THE USER STACK PTR.)
3633          : *
3634          : * NOTE THAT THE PREVIOUS MODE BITS <13:12> OF THE PSW
3635          : * WILL BE SET IN THE PSW THAT IS SAVED.
3636          : *
3637          : *
3638          : *****
3638 031570 000004 TST45: SCOPE
3639 031572 004737 036002 1$: JSR PC,TOFF ;TURN T-BIT TRAPPING OFF FOR THIS TEST
3640 031576 012737 000600 177646 MOV #600,UIPAR3 ;MAP USER PAGE 3 TO 12-16K
3641 031604 012737 000600 177650 MOV #600,UIPAR4 ;MAP USER PAGE 4 TO 12-16K
3642 031612 012737 077402 177606 MOV #77402,UIPDR3 ;MAP USER PAGE 3 READ-ONLY
3643 031620 012737 077406 177610 MOV #77406,UIPDR4 ;MAP USER PAGE 4 READ/WRITE
3644 031626 012737 031702 000004 MOV #4$,#4 ;LOAD ADDRESS OF 4$ IN CPU (ODD ADDR.) VECTOR
3645 031634 012737 140017 000006 MOV #140017,#6 ;LOAD PSW THAT SHOULD BE PUT ON STACK IN VECTOR+2
3646 031642 012737 031702 000250 MOV #4$,#250 ;LOAD ADDRESS OF 4$ IN M.M. TRAP VECTOR
3647 031650 012737 000340 000252 MOV #340,#252 ;LOAD A KERNEL PSW IN MMVEC+2
3648 031656 012737 031664 001110 MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
3649 031664 012737 140000 177778 2$: MOV #140000,PSW ;GO TO USER MODE
3650 031672 012706 100002 MOV #100002,USP ;SET USER STACK PTR. SO SECOND PUSH IS IN PG. 3
3651 031676 005237 100005 3$: INC 100005 ;CAUSE ODD ADDRESS ERROR THAT WILL CAUSE
3652          : ;R/O ERROR WHEN TRY TO SAVE OLD PC
3653 031702 016601 000002 4$: MOV 3(KSP),R1 ;PUT PSW SAVED ON KERNEL STACK INTO R1
3654 031706 011603 MOV (KSP),R3 ;PUT PC SAVED ON KERNEL STACK INTO R3
3655 031710 013737 177572 001366 MOV SRO,WASSR0 ;READ THE CONTENTS OF M.M. STATUS REG. 0
3656 031716 013737 177576 001370 MOV SR2,WASSR2 ;READ THE CONTENTS OF M.M. STATUS REG. 2
3657 031724 042737 160000 177572 BIC #160000,SRO ;CLEAR THE ERROR BITS IN SRO
3658 031732 005037 177776 CLR PSW ;BE SURE IN KERNEL MODE
3659 031736 012706 001100 MOV #KERSTK,KSP ;RESTORE KERNEL STACK POINTER
3660 031742 012737 140000 177778 MOV #140000,PSW ;GO TO USER MODE
3661 031750 012706 000700 MOV #USESTK,USP ;RESTORE USER STACK POINTER
3662 031754 005037 177776 CLR PSW ;GO BACK TO KERNEL MODE
3663 031760 005037 001176 CLR $TMP0 ;CLEAR ERROR INDICATOR
3664 031764 020127 170017 CMP R1,#170017 ;WAS THE PSW SAVED THE ONE PICKED UP BY THE
3665          : ;ODD ADDR. TRAP FROM ERRVEC+2?
3666          : ;VALUE 170017 = PSW FROM LOC. 6 WITH
3667          : ;PREVIOUS MODE BITS = USER
3668 031770 001402 BEQ 5$ ;BRANCH IF YES
3669 031772 005237 001176 INC $TMP0 ;WRONG PSW SAVED DURING "DOUBLE ERROR" SEQUENCE
    
```

```

3670 031776 020327 031702 5$: CMP R3,#3$+4 ;WAS THE PC AT THE TIME OF THE ODD ADDR. ERROR
3671          : ;SAVED ON THE STACK?
3672 032002 001402 BEQ 6$ ;BRANCH IF YES
3673 032004 005237 001176 INC $TMP0 ;WRONG PC SAVED DURING TRAP SEQUENCE
3674 032010 023727 001366 020147 6$: CMP WASSR0,#20147 ;DID SRO REPORT - USER, PAGE 3, R/O ABORT?
3675 032016 001402 BEQ 7$ ;BRANCH IF YES
3676 032020 005237 001176 INC $TMP0 ;SRO DID NOT REPORT R/O ABORT
3677 032024 023727 001370 031678 7$: CMP WASSR2,#3$ ;DID SR2 LOCK UP VIRTUAL ADDR. OF LAST
3678          : ;INSTRUCTION SUCCESSFULLY FETCHED?
3679 032032 001402 BEQ 8$ ;BRANCH IF YES
3680 032034 005237 001176 INC $TMP0 ;SR2 DID NOT LOCK UP ADDR. OF ODD ADDR. INST.
3681 032040 005737 001176 8$: TST $TMP0 ;ANY "ERRORS" DURING TRAP SEQUENCE?
3682 032044 001401 BEQ 9$ ;BRANCH IF NO
3683 032046 104045 ERROR 4$ ;THE WRONG PC OR PSW WERE SAVED
3684          : ;OR SRO OR SR2 DID NOT REPORT R/O
3685          : ;ERROR DURING ODD ADDR. - KT TRAP
3686          : ;SEQUENCE
3687          : ;FOR TIGHTER SCOPE LOOP
3688          : ;REPLACE ERROR CALL WITH
3689          : ;"BR 2$" = C00710
3690 032050 012737 002332 000004 9$: MOV #TIMERR,#4 ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
3691 032056 012737 000340 000006 MOV #340,#6 ;RELOAD ERRVEC+2 WITH KERNEL PSW
3692 032064 012737 002400 000250 MOV #MGMERR,#250 ;RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER
3693 032072 012737 077406 177606 MOV #77406,UIPDR3 ;REMAP USER PAGE 3 READ/WRITE
3694 032100 012737 031572 001110 MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
3695 032106 004737 036038 JSR PC,TON ;TURN T-BIT TRAPPING BACK ON
3696          : ;*****
3697          : ;*
3698          : ;* THIS GROUP OF TESTS WILL TEST ALL THE LOGIC ASSOCIATED WITH
3699          : ;* THE "MOVE FROM PREVIOUS" AND MOVE TO PREVIOUS" INSTRUCTIONS.
3700          : ;*
3701          : ;*****
3702          : ;*****
3703          : ;*****
3704          : ;*****
3705          : ;*TEST 46 MOVE FROM PREVIOUS (USER) I-SPACE
3706          : ;*
3707          : ;* THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT THE
3708          : ;* PREVIOUS MODE IS CLOCKED CORRECTLY
3709          : ;* THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.
3710          : ;*
3711          : ;*
3712          : ;* IF THE CORRECT MODE (USER) IS NOT ENABLED A NON-RESIDENT ABORT
3713          : ;* WILL OCCUR AND TRAP TO 23$, WHERE THE ERRORS ARE REPORTED.
3714          : ;*
3715          : ;*****
3716 032112 000004 TST46: SCOPE
3717 032114 005037 172340 1$: CLR KIPAR0 ;MAP KERNEL PAGE 0 TO 0-4K
3718 032120 012737 000200 172342 MOV #200,KIPAR1 ;MAP KERNEL PAGE 1 TO 4-8K
3719 032126 012737 000400 172344 MOV #400,KIPAR2 ;MAP KERNEL PAGE 2 TO 8-12K
3720 032134 012737 000600 172346 MOV #600,KIPAR3 ;MAP KERNEL PAGE 3 TO 12-16K
3721 032142 012737 000600 172350 MOV #600,KIPAR4 ;MAP KERNEL PAGE 4 TO 12-16K
3722 032150 012737 007600 172353 MOV #7600,KIPAR7 ;MAP KERNEL PAGE 7 TO THE I/O PAGE
3723 032156 012700 077406 MOV #77406,R0 ;MAKE ALL KERNEL I-SPACE PAGES RESIDENT
3724          : ;READ/WRITE. LENGTH 200 BLOCKS
3725 032162 012702 000010 MOV #10,R2 ;SET LOOP COUNTER TO 8
    
```

3726	032166	012701	172300		MOV	#KIPDR0,R1	;PUT ADDRESS OF FIRST PDR IN R1
3727	032172	010021		2\$:	MOV	R0,(R1)+	;LOAD PDR WITH 77406
3728	032174	077202			SOB	R2,2\$;LOOP TO 2\$ UNTIL ALL PDRS LOADED
3729	032176	012702	000010		MOV	#10,R2	;SET LOOP COUNTER TO 8
3730	032202	012701	177600		MOV	#UIPDR0,R1	;PUT ADDRESS OF FIRST PDR IN R1
3731	032206	010021		3\$:	MOV	R0,(R1)+	;LOAD PDR WITH 77406
3732	032210	077202			SOB	R2,3\$;LOOP TO 3\$ UNTIL ALL PDRS LOADED
3733	032212	012737	000000	177640	MOV	#000,UIPAR0	;MAP USER I PAGE 0 TO 0-4K
3734	032220	012737	000200	177642	MOV	#200,UIPAR1	;MAP USER I PAGE 1 TO 4-8K
3735	032226	012737	000400	177644	MOV	#400,UIPAR2	;MAP USER I PAGE 2 TO 8-12K
3736	032234	012737	000600	177646	MOV	#600,UIPAR3	;MAP USER I PAGE 3 TO 12-16K
3737	032242	012737	007600	177650	MOV	#7600,UIPAR7	;MAP USER I PAGE 7 TO THE I/D PAGE
3738	032250	012737	032256	001110	MOV	#4\$, \$LPERR	;SET LOOP ON ERROR TO 4\$
3739	032256						
3740	032256	012737	077406	172310	4\$:	MOV	#77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE
3741	032264	012737	000600	172350	MOV	#600,KIPAR4 ;MAP KERNEL I PAGE 4 TO 12K	
3742	032272	012737	000600	177650	MOV	#600,UIPAR4 ;MAP USER I PAGE 4 TO 12K	
3743	032300	012700	036514		MOV	#36514,R0 ;LOAD DATA PATTERN INTO R0	
3744	032304	010037	100000		MOV	R0,#100000 ;LOAD DATA PATTERN INTO PHY 60000	
3745	032310	012737	032712	000250	MOV	#23\$,MMVEC ;SET M.M. VECTOR TO 23\$	
3746	032316	105037	172310		CLRB	KIPDR4 ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT	
3747							;THE FOLLOWING WILL TEST DSTM=0 MFPI
3748							
3749	032322	012737	032330	001110	MOV	#5\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 5\$	
3750	032330	012737	030340	177778	5\$:	MOV	#030340,PSW ;MAKE PREVIOUS MODE USER
3751	032336	006506			6\$:	MFPI	USP ;PUT USER STACK POINTER ON KERNEL
3752							;STACK
3753	032340	022706	001100		CMP	#KERSTK,KSP ;WAS SOMETHING PUSHED ON STACK AT 6\$	

3754	032344	001407			BEQ	7\$;BRANCH IF NOTHING WAS PUSHED
3755	032346	012600			MOV	(KSP)+,R0 ;POP KERNEL STACK INTO R0	
3756	032350	012701	000700		MOV	#USESTK,R1 ;EXPECTING TO GET 700 AS USP	


```

3803                                     ;"BR 13$" = 000767
3804 032506                             ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
3805 032506 012737 032514 001110 14$: MOV #15$, $LPERR ;SET LOOP ON ERROR POINTER TO 15$
3806 032514 012737 030340 177776 15$: MOV #030340, PSW ;MAKE PREVIOUS MODE USER
3807 032522 012702 100002             MOV #100000, R2 ;LOAD TEST LOC. VIRT. ADDR INTO R2
3808 032526 066542                     MFPI -(R2) ;READ FROM PHYSICAL 60000
3809 032530 012601                     MOV (KSP)+, R1 ;POP KERNEL STACK INTO R1
3810 032532 020001                     CMP R0, R1 ;WAS DATA FETCHED SAME AS STORED
3811 032534 001401                     BEQ 16$ ;BRANCH IF CORRECT DATA WAS FETCHED
3812 032536 104046                     ERROR 46 ;WRONG DATA WAS FETCHED
3813                                     ;FOR TIGHTER SCOPE LOOP
3814                                     ;REPLACE ERROR CALL WITH
3815                                     ;"BR 15$" = 000766
3816 032540                             16$: ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
3817                                     ;
3818                                     ;
3819 032540 012737 032546 001110       MOV #17$, $LPERR ;SET LOOP ON ERROR POINTER TO 17$
3820 032546 012737 030340 177776 17$: MOV #030340, PSW ;MAKE PREVIOUS MODE USER
3821 032554 012737 100000 001202       MOV #100000, $TMP2 ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
3822 032562 012702 001204             MOV #<$TMP2+2>, R2 ;LOAD ADDR. OF $TMP2+2 INTO R2
3823 032566 066552                     MFPI @-(R2) ;READ FROM PHYSICAL 60000
3824 032570 012601                     MOV (KSP)+, R1 ;POP KERNEL STACK INTO R1
3825 032572 020001                     CMP R0, R1 ;WAS DATA FETCHED SAME AS STORED
3826 032574 001401                     BEQ '8$ ;BRANCH IF CORRECT DATA WAS FETCHED
3827 032576 104046                     ERROR 46 ;WRONG DATA WAS FETCHED
3828                                     ;FOR TIGHTER SCOPE LOOP
3829                                     ;REPLACE ERROR CALL WITH
3830                                     ;"BR 17$" = 000763
3831 032600                             18$: ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
3832                                     ;
3833 032600 012737 032606 001110       MOV #19$, $LPERR ;SET LOOP ON ERROR POINTER TO 19$
3834 032606 012737 030340 177776 19$: MOV #030340, PSW ;MAKE PREVIOUS MODE USER
3835 032614 005002                     CLR R2 ;MAKE REGISTER 2 A ZERO
3836 032616 066562 100000             MFPI 100000(R2) ;READ FROM PHYSICAL 60000
3837 032622 012601                     MOV (KSP)+, R1 ;POP KERNEL STACK INTO R1
3838 032624 020001                     CMP R0, R1 ;WAS DATA FETCHED SAME AS STORED
3839 032626 001401                     BEQ 20$ ;BRANCH IF CORRECT DATA WAS FETCHED
3840 032630 104046                     ERROR 46 ;WRONG DATA WAS FETCHED
3841                                     ;FOR TIGHTER SCOPE LOOP
3842                                     ;REPLACE ERROR CALL WITH
3843                                     ;"BR 19$" = 000766
3844 032632                             20$: ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
3845                                     ;
3846 032632 012737 032640 001110       MOV #21$, $LPERR ;SET LOOP ON ERROR POINTER TO 21$
3847 032640 012737 030340 177776 21$: MOV #030340, PSW ;MAKE PREVIOUS MODE USER
3848 032646 012737 100000 001202       MOV #100000, $TMP2 ;LOAD TEST LOC. V.A. INTO $TMP2
3849 032654 012702 001202             MOV # $TMP2, R2 ;LOAD ADDRESS OF $TMP2 INTO R2
3850 032660 066572 000000             MFPI @0(R2) ;USE $TMP2 TO FETCH VIRTUAL
3851                                     ;ADDRESS OF 60000
3852 032664 012601                     MOV (KSP)+, R1 ;POP KERNEL STACK INTO R1
3853 032666 020001                     CMP R0, R1 ;WAS DATA FETCHED SAME AS STORED
3854 032670 001401                     BEQ 22$ ;BRANCH IF CORRECT DATA WAS FETCHED
3855 032672 104046                     ERROR 46 ;WRONG DATA WAS FETCHED
3856                                     ;FOR TIGHTER SCOPE LOOP
3857                                     ;REPLACE ERROR CALL WITH
3858                                     ;"BR 21$" = 000762
    
```

```

3859 032674 012737 002400 000250 22$: MOV #MGMERR, MMVEC ;SET M.M. VECTOR TO NORMAL ROUTINE
3860 032702 012737 032114 001110     MOV #1$, $LPERR ;SET LOOP POINTER TO START OF TEST
3861 032710 000423                     BR TST47 ;BRANCH TO NEXT TEST
3862                                     ;
3863                                     ;
3864 032712 012637 001356             23$: MOV (KSP)+, TRAPPC ;SAVE PC & PS OF TRAP
3865 032716 012637 001360             MOV (KSP)+, TRAPPS ;
3866 032722 013737 177572 001366     MOV SRO, WASSR0 ;SAVE SRO FOR ERROR TYPEOUT
3867 032730 013737 177576 001370     MOV SR2, WASSR2 ;SAVE SR2 FOR ERROR TYPEOUT
3868 032736 042737 160000 177572     BIC #160000, SRO ;CLEAR ERROR BITS IN SRO AND LEAVE
3869 032744 104051                     ERROR 51 ;TRIED TO READ NON-RESIDENT PAGE
3870                                     ;FOR TIGHTER SCOPE LOOP
3871                                     ;REPLACE ERROR CALL WITH
3872                                     ;"A NOP" = 000240
3873 032746 013746 001360             MOV TRAPPS, -(KSP) ;PUT PC & PS OF TRAP ON STACK
3874 032752 013746 001356             MOV TRAPPC, -(KSP) ;
3875 032756 000002                     RTI ;
3876                                     ;
3877                                     ;
3878                                     ;*****
3879 *TEST 47 MOVE TO PREVIOUS (USER) I-SPACE
3880 *
3881 * THIS TEST USES THE 'MTPI' INSTRUCTION TO ENSURE THAT THE
3882 * PREVIOUS MODE IS CLOKED CORRECTLY
3883 * THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.
3884 *
3885 *
3886 * IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
3887 * WILL OCCUR AND TRAP TO 20$, WHERE THE ERRORS ARE REPORTED.
3888 *
3889 * ;*****
3890 032760 000004                       TST47: SCOPE
3891 032762 012737 077406 172310 1$: MOV #77406, KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE
3892 032770 012737 077406 177610     MOV #77406, UIPDR4 ;USER I-SPACE PAGE 4 READ/WRITE
3893 032776 012737 000600 172350     MOV #600, KIPAR4 ;MAP KERNEL I PAGE 4 TO 12K
3894 033004 012737 000600 177650     MOV #600, UIPAR4 ;MAP USER I PAGE 4 TO 12K
3895 033012 012737 033572 000250     MOV #20$, MMVEC ;SET M.M. VECTOR TO 20$
3896                                     ;
3897                                     ;THE FOLLOWING WILL TEST DSTM=0 MTPI
3898 033020 012737 030340 177776 2$: MOV #030340, PSW ;MAKE PREVIOUS MODE USER
3899 033026 012746 007777             MOV #7777, -(KSP) ;PUSH DATA ON KERNEL STACK
3900 033032 066606                     MTPI USP ;LOAD USER STACK POINTER
3901 033034 066506                     MFPI USP ;READ USER STACK POINTER
3902 033036 012601                     MOV (KSP)+, R1 ;POP KERNEL STACK INTO R1
3903 033040 022701 007777             CMP #7777, R1 ;WAS USER STACK POINTER CHANGED
3904 033044 001401                     BEQ 35 ;BRANCH IF IT WAS
3905 033046 104050                     ERROR 50 ;USER STACK POINTER NOT CHANGED
3906                                     ;FOR TIGHTER SCOPE LOOP
3907                                     ;REPLACE ERROR CALL WITH
3908                                     ;"BR 2$" = 000764
3909 033050 012737 030340 177776 3$: MOV #030340, PSW ;MAKE PREVIOUS MODE USER
3910 033056 012746 000700             MOV #USESTK, -(KSP) ;GET READY TO RESTORE USER S. POINT
3911 033062 066506                     MTPI USP ;RESTORE USER STACK POINTER
3912 033064                                     ;THIS WILL TEST DSTM = 1 MTPI.
3913 033064 012737 033102 001110 4$: MOV #5$, $LPERR ;SET LOOP ON ERROR POINTER TO 5$
3914 033072 012702 100000             MOV #100000, R2 ;LOAD VIRTUAL ADDRESS INTO R2
    
```

```

3915 033076 012700 125252          MOV    #125252,R0      ;LOAD TEST DATA INTO R0
3916 033102 010046          5$:   MOV    R0,-(KSP)    ;PUSH TEST DATA ON KERNEL STACK
3917 033104 105037 172310          CLRB  KIPDR4         ;MAKE KERNEL I PAGE 4 NON-RESIDENT
3918 033110 006612          MTPI  (R2)           ;LOAD TEST DATA INTO PHYSICAL 60000
3919 033112 112737 000006 172310  MOVB  #006,KIPDR4    ;MAKE KERNEL PAGE 4 RESIDENT
3920 033120 011201          MOV   (R2),R1        ;READ FROM ADDRESS 60000
3921 033122 020001          CMP   R0,R1          ;SEE IF DATA WAS STORED AT CORRECT PLACE
3922 033124 001401          BEQ   6$             ;BRANCH IF STORE WAS CORRECT
3923 033126 104047          ERROR 47             ;INCORRECT STORE
3924          ;FOR TIGHTER SCOPE LOOP
3925          ;REPLACE ERROR CALL WITH
3926          ;"BR 5$" = 000765
3927 033130          6$:   ;THE FOLLOWING WILL TEST DSTM=2 MTPI.
3928          ;
3929 033130 012737 033154 001110  MOV   #8$,SLPERR     ;SET LOOP ON ERROR POINTER TO 8$
3930 033136 012737 030340 177776  MOV   #030340,PSW    ;MAKE PREVIOUS MODE USER
3931 033144 012700 125252          MOV   #125252,R0     ;LOAD TEST DATA INTO R0
3932 033150 012702 100000          MOV   #100000,R2     ;LOAD VIRTUAL ADDRESS INTO R2
3933 033154 010046          8$:   CLRB  KIPDR4         ;PUSH TEST DATA ON KERNEL STACK
3934 033156 105037 172310          MTPI  (R2)           ;MAKE KERNEL PAGE 4 NON-RESIDENT
3935 033162 006612          MOVB  #006,KIPDR4    ;LOAD TEST DATA INTO PHYSICAL 60000
3936 033164 112737 000006 172310  MOVB  #006,KIPDR4    ;MAKE KERNEL PAGE 4 RESIDENT
3937 033172 013701 100000          MOV   #100000,R1     ;READ FROM ADDRESS 60000
3938 033176 020001          CMP   R0,R1          ;SEE IF DATA WAS STORED CORRECTLY
3939 033200 001401          BEQ   9$             ;BRANCH IF STORE WAS CORRECT
3940 033202 104047          ERROR 47             ;INCORRECT STORE
3941          ;FOR TIGHTER SCOPE LOOP
3942          ;REPLACE ERROR CALL WITH
3943          ;"BR 8$" = 000764
3944 033204          9$:   ;THIS WILL TEST DSTM = 3 MTPI.
3945 033204 012737 033224 001110  MOV   #10$,SLPERR    ;SET LOOP ON ERROR POINTER TO 10$
3946 033212 012737 030340 177776  MOV   #030340,PSW    ;MAKE PREVIOUS MODE USER
3947 033220 012700 052525          MOV   #52525,R0     ;LOAD TEST DATA INTO R0
3948 033224 010046          10$:  MOV   R0,-(KSP)     ;PUSH TEST DATA ON KERNEL STACK
3949 033226 105037 172310          CLRB  KIPDR4         ;MAKE KERNEL I PAGE 4 NON-RESIDENT
3950 033232 006637 100000          MTPI  #100000        ;LOAD TEST DATA INTO PHYSICAL 60000
3951 033236 112737 000006 172310  MOVB  #006,KIPDR4    ;MAKE KERNEL PAGE 4 RESIDENT
3952 033244 013701 100000          MOV   #100000,R1     ;READ FROM ADDRESS 60000
3953 033250 020001          CMP   R0,R1          ;SEE IF DATA WAS STORED CORRECTLY
    
```

```

3954 033252 001401          BEQ   11$            ;BRANCH IF STORE WAS CORRECT
3955 033254 104047          ERROR 47             ;INCORRECT STORE
3956          ;FOR TIGHTER SCOPE LOOP
3957          ;REPLACE ERROR CALL WITH
3958          ;"BR 10$" = 000763
3959 033256          11$: ;THIS WILL TEST DSTM = 4 MTPI.
3960 033256 012737 033276 001110  MOV   #12$,SLPERR    ;SET LOOP ON ERROR POINTER TO 12$
3961 033264 012737 030340 177776  MOV   #030340,PSW    ;MAKE PREVIOUS MODE USER
3962 033272 012700 125252          MOV   #125252,R0     ;LOAD TEST DATA INTO R0
3963 033276 010046          12$:  MOV   R0,-(KSP)     ;PUSH TEST DATA ON KERNEL STACK
3964 033300 012702 100002          MOV   #100002,R2     ;LOAD VIRTUAL ADDRESS INTO R2
    
```

```

3965 033304 105037 172310 CLR B KIPDR4 ;MAKE KERNEL I PAGE 4 NON-RESIDENT
3966 033310 006642 MTPI -(R2) ;LOAD TEST DATA INTO PHYSICAL 60000
3967 033312 112737 000008 172310 MOVB #006,KIPDR4 ;MAKE KERNEL PAGE 4 RESIDENT
3968 033320 013701 100000 MOV #100000,R1 ;READ FROM ADDRESS 60000
3969 033324 020001 CMP R0,R1 ;SEE IF DATA WAS STORED CORRECTLY
3970 033326 001401 BEQ 13$ ;BRANCH IF STORE WAS CORRECT
3971 033330 104047 ERROR 47 ;INCORRECT STORE
3972 ;FOR TIGHTER SCOPE LOOP
3973 ;REPLACE ERROR CALL WITH
3974 ;"BR 12$" = 000762
3975 033332 13$: ;THE FOLLOWING WILL TEST DSTM=5 MTPI.
3976 ;
3977 033332 012737 033364 001110 MOV #14$,$LPERR ;SET LOOP ON ERROR POINTER TO 14$
3978 033340 012737 030340 177776 MOV #030340,PSW ;MAKE PREVIOUS MODE USER
3979 033346 012700 052525 MOV #52525,R0 ;LOAD TEST DATA INTO R0
3980 033352 012702 001204 MOV #<STMP2+2>,R2 ;LOAD ADDR. OF LOC. $TMP2+2 INTO R2
3981 033356 012737 100000 001202 MOV #100000,$TMP2 ;LOAD VIRT. ADDR. OF TEST LOC. INTO $TMP2
3982 033364 010046 14$: MOV R0,-(KSP) ;PUSH TEST DATA ON KERNEL STACK
3983 033366 105037 172310 CLR B KIPDR4 ;MAKE KERNEL PAGE 4 NON-RESIDENT
3984 033372 006652 MTPI @-(R2) ;LOAD TEST DATA INTO PHYSICAL 60000
3985 033374 112737 000006 172310 MOVB #006,KIPDR4 ;MAKE KERNEL PAGE 4 RESIDENT
3986 033402 013701 100000 MOV #100000,R1 ;READ FROM ADDRESS 60000
3987 033406 020001 CMP R0,R1 ;SEE IF DATA WAS STORED CORRECTLY
3988 033410 001401 BEQ 15$ ;BRANCH IF STORE WAS CORRECT
3989 033412 104047 ERROR 47 ;INCORRECT STORE
3990 ;FOR TIGHTER SCOPE LOOP
3991 ;REPLACE ERROR CALL WITH
3992 ;"BR 14$" = 000764
3993 033414 15$: ;THIS WILL TEST DSTM = 6 MTPI.
3994 ;
3995 033414 012737 033436 001110 MOV #16$,$LPERR ;SET LOOP ON ERROR POINTER TO 16$
3996 033422 012737 030340 177776 MOV #030340,PSW ;MAKE PREVIOUS MODE USER
3997 033430 012700 052525 MOV #52525,R0 ;LOAD TEST DATA INTO R0
3998 033434 005002 CLR R2 ;MAKE REGISTER 2 ZERO
3999 033436 010046 16$: MOV R0,-(KSP) ;PUSH TEST DATA ON KERNEL STACK
4000 033440 105037 172310 CLR B KIPDR4 ;MAKE KERNEL I PAGE 4 NON-RESIDENT
4001 033444 006662 MTPI 100000(R2) ;LOAD TEST DATA INTO PHYSICAL 60000
4002 033450 112737 000006 172310 MOVB #006,KIPDR4 ;MAKE KERNEL PAGE 4 RESIDENT
4003 033456 013701 100000 MOV #100000,R1 ;READ FROM ADDRESS 60000
4004 033462 020001 CMP R0,R1 ;SEE IF DATA WAS STORED CORRECTLY
4005 033464 001401 BEQ 17$ ;BRANCH IF STORE WAS CORRECT
4006 033466 104047 ERROR 47 ;INCORRECT STORE
4007 ;FOR TIGHTER SCOPE LOOP
4008 ;REPLACE ERROR CALL WITH
4009 ;"BR 16$" = 000763
4010 033470 17$: ;THE FOLLOWING WILL TEST DSTM=7 MTPI.
4011 ;
4012 033470 012737 033522 001110 MOV #18$,$LPERR ;SET LOOP ON ERROR POINTER TO 18$
4013 033476 012737 030340 177776 MOV #030340,PSW ;MAKE PREVIOUS MODE USER
4014 033504 012700 125252 MOV #125252,R0 ;LOAD TEST DATA INTO R0
4015 033510 012737 100000 001202 MOV #100000,$TMP2 ;LOAD VIRT. ADDR. OF TEST LOCATION
4016 INTO LOCATION $TMP2
4017 033516 012702 001202 18$: MOV #<TMP2,R2 ;LOAD ADDRESS OF $TMP2 INTO R2
4018 033522 010046 MOV R0,-(KSP) ;PUSH TEST DATA ON KERNEL STACK
4019 033524 105037 172310 CLR B KIPDR4 ;MAKE KERNEL PAGE 4 NON-RESIDENT
4020 033530 006672 000000 MTPI @0(R2) ;LOAD TEST DATA INTO PHYSICAL 60000
    
```

```

4021 033534 112737 000008 172310 MOVB #006,KIPDR4 ;MAKE KERNEL PAGE 4 RESIDENT
4022 033542 013701 100000 MOV #100000,R1 ;READ FROM ADDRESS 60000
4023 033546 020001 CMP R0,R1 ;SEE IF DATA WAS STORED CORRECTLY
4024 033550 001401 BEQ 19$ ;BRANCH IF STORE WAS CORRECT
4025 033552 104047 ERROR 47 ;INCORRECT STORE
4026 ;FOR TIGHTER SCOPE LOOP
4027 ;REPLACE ERROR CALL WITH
4028 ;"BR 18$" = 000763
4029 033554 012737 032762 001110 19$: MOV #1$,$LPERR ;SET LOOP POINTER TO START OF TEST
4030 033562 012737 002400 000250 MOV #MGMERR,MMVEC ;RESTORE M.M. VECTOR TO NORMAL ROUTINE
4031 033570 000423 BR TST50 ;BRANCH TO NEXT TEST
4032
4033
4034 033572 012637 001356 20$: MOV (KSP)+,TRAPPC ;SAVE PC & PS OF TRAP
4035 033576 012637 001360 MOV (KSP)+,TRAPPS
4036 033602 013737 177572 001366 MOV SRO,WASSR0 ;SAVE SRO FOR ERROR TYPEDUT
4037 033610 013737 177576 001370 MOV SR2,WASSR2 ;SAVE SR2 FOR ERROR TYPEDUT
4038 033616 042737 160000 177572 BIC #160000,SRO ;CLEAR ERROR BITS IN SRO
4039 033624 104051 ERROR 51 ;TRIED TO LOAD A N.R. PAGE 4
4040 ;FOR TIGHTER SCOPE LOOP
4041 ;REPLACE ERROR CALL WITH
4042 ;A "NOP" = 000240
4043 033626 013746 001360 MOV TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
4044 033632 013746 001356 MOV TRAPPC,-(KSP)
4045 033636 000002 RTI ;RETURN TO TEST
4046
4047
4048
4049
4050 ;*****
4051 ;*TEST 50 MOVE FROM PREVIOUS (KERNEL) I-SPACE TO USER MODE
4052 ;*
4053 ;* THIS TEST CHECKS THAT IF THE PREVIOUS MODE IS KERNEL THE
4054 ;* FETCH IS FROM KERNEL SPACE.
4055 ;* THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.
4056 ;*
4057 ;* IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
4058 ;* WILL OCCUR AND TRAP TO 21$, WHERE THE ERRORS ARE REPORTED.
4059 ;*
4060 ;*****
4061 033640 000004 TST50: SCOPE
4062 033642 012700 077406 1$: MOV #77406,R0 ;MAKE ALL USER I-SPACE PAGES RESIDENT
4063 ;READ/WRITE, LENGTH 200 BLOCKS
4064 033646 012702 000010 MOV #10,R2 ;SET LOOP COUNTER TO 8
4065 033652 012701 177600 MOV #UIPDRO,R1 ;LOAD ADDRESS OF FIRST PDR IN R1
4066 033656 010021 2$: MOV R0,(R1)+ ;LOAD PC-1 WITH 77406
4067 033660 077202 SOB R2,2$ ;LOOP UNTIL 8 USER PDRS LOADED
4068 033662 012737 033670 001110 MOV #3$,$LPERR ;SET LOOP ON ERROR TO 3$
4069 033670 012737 140340 177776 MOV #140340,PSW ;GO TO USER MODE FOR THIS TEST
4070 033676 012737 077406 172310 3$: MOV #77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE
4071 033704 012737 000600 172350 MOV #600,KIPPAR4 ;MAP KERNEL I PAGE 4 TO 12K
4072 033712 012737 000600 177650 MOV #600,UIPPAR4 ;MAP USER I PAGE 4 TO 12K
4073 033720 012700 036514 MOV #36514,R0 ;LOAD DATA PATTERN INTO R0
4074 033724 010037 100000 MOV R0,#100000 ;LOAD DATA PATTERN INTO PHY 60000
4075 033730 012702 100000 MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
4076 ;THE FOLLOWING WILL TEST DSTM=0 MFPI
    
```



```

4077 ;
4078 033734 012737 034336 000250 MOV #21$,MMVEC ;SET M.M. VECTOR TO 21$
4079 033742 105037 177610 CLR B UIPDR4 ;MAKE USER I-SPACE PAGE 4 NON-RESIDENT
4080 033746 012737 140340 177776 MOV #140340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT USER
4081 033754 006506 4$: MFPI KSP ;PUT KERNEL STACK POINTER ON USER STACK
4082 033756 022706 000700 CMP #USESTK,USP ;WAS SOMETHING PUSHED ON STACK AT 1$
4083 033762 001407 BEQ 5$ ;BRANCH IF NOTHING WAS PUSHED
4084 033764 012600 MOV (USP)+,R0 ;POP USER STACK INTO R0
4085 033766 012701 001100 MOV #KERSTK,R1 ;EXPECTING 1100 AS KSP
4086 033772 020001 CMP R0,R1 ;DID YOU GET THE RIGHT POINTER?
4087 033774 001403 BEQ 6$ ;BRANCH IF YOU DID
4088 033776 104046 ERROR 46 ;WRONG THING WAS PUSHED ON STACK
4089 ;FOR TIGHTER SCOPE LOOP
4090 ;REPLACE ERROR CALL WITH
4091 ;"BR 4$" = 000766
4092 034000 000401 BR 6$ ;BRANCH TO NEXT TRY
4093 034002 104050 5$: ERROR 50 ;NOTHING PUSHED ON STACK
4094 ;FOR TIGHTER SCOPE LOOP
4095 ;REPLACE ERROR CALL WITH
4096 ;"BR 4$" = 000764
4097 034004 6$: ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
4098 034004 012737 034012 001110 MOV #7$, $LPERR ;SET LOOP ON ERROR POINTER TO 7$
4099 034012 012737 140340 177776 7$: MOV #140340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT USER
4100 034020 012700 036514 MOV #36514,R0 ;LOAD DATA EXPECTED INTO R0
4101 034024 012702 100000 MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
4102 034030 006512 MFPI (R2) ;READ FROM PHYSICAL 60000
4103 034032 012601 MOV (USP)+,R1 ;POP USER STACK INTO R1
4104 034034 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
4105 034036 001401 BEQ 8$ ;BRANCH IF CORRECT DATA WAS FETCHED
4106 034040 104046 ERROR 46 ;WRONG DATA WAS FETCHED
4107 ;FOR TIGHTER SCOPE LOOP
4108 ;REPLACE ERROR CALL WITH
4109 ;"BR 7$" = 000764
4110 034042 8$: ;THE FOLLOWING WILL TEST DSTM=2 MFPI.
4111 034042 012737 034050 001110 MOV #9$, $LPERR ;SET LOOP ON ERROR POINTER TO 9$
4112 034050 012737 140340 177776 9$: MOV #140340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT USER
4113 034056 012702 100000 MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
4114 034062 006522 MFPI (R2)+ ;READ FROM PHYSICAL 60000
4115 034064 012601 MOV (USP)+,R1 ;POP USER STACK INTO R1
4116 034066 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
4117 034070 001401 BEQ 10$ ;BRANCH IF CORRECT DATA WAS FETCHED
4118 034072 104046 ERROR 46 ;WRONG DATA WAS FETCHED
4119 ;FOR TIGHTER SCOPE LOOP
4120 ;REPLACE ERROR CALL WITH
4121 ;"BR 9$" = 000766
4122 034074 10$: ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
4123 034074 012737 034102 001110 MOV #11$, $LPERR ;SET LOOP ON ERROR POINTER TO 11$
4124 034102 012737 140340 177776 11$: MOV #140340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT USER
4125 034110 006537 100000 MFPI @#100000 ;READ FROM PHYSICAL 60000
4126 034114 012601 MOV (USP)+,R1 ;POP USER STACK INTO R1
4127 034116 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
4128 034120 001401 BEQ 12$ ;BRANCH IF CORRECT DATA WAS FETCHED
4129 034122 104046 ERROR 46 ;WRONG DATA WAS FETCHED
4130 ;FOR TIGHTER SCOPE LOOP
4131 ;REPLACE ERROR CALL WITH
4132 ;"BR 11$" = 000767
    
```

```

4133 034124 12$: ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
4134 034124 012737 034132 001110 MOV #13$, $LPERR ;SET LOOP ON ERROR POINTER TO 13$
4135 034132 012737 140340 177776 13$: MOV #140340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT USER
4136 034140 012702 100002 MOV #100002,R2 ;LOAD VIRTUAL ADDRESS INTO R2
4137 034144 006542 MFPI -(R2) ;READ FROM PHYSICAL 60000
4138 034146 012601 MOV (USP)+,R1 ;POP USER STACK INTO R1
4139 034150 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
4140 034152 001401 BEQ 14$ ;BRANCH IF CORRECT DATA WAS FETCHED
4141 034154 104046 ERROR 46 ;WRONG DATA WAS FETCHED
4142 ;FOR TIGHTER SCOPE LOOP
4143 ;REPLACE ERROR CALL WITH
4144 ;"BR 13$" = 000766
4145 034156 14$: ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
4146 ;
4147 034156 012737 034164 001110 MOV #15$, $LPERR ;SET LOOP ON ERROR POINTER TO 15$
4148 034164 012737 140340 177776 15$: MOV #140340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT USER
4149 034172 012737 100000 001202 MOV #100000,$TMP2 ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
4150 034200 012702 001204 MOV #<TMP2+2>,R2 ;LOAD ADDRESS OF $TMP2+2 INTO R2
4151 034204 006552 MFPI @-(R2) ;READ FROM PHYSICAL 60000
4152 034206 012601 MOV (USP)+,R1 ;POP USER STACK INTO R1
4153 034210 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
4154 034212 001401 BEQ 15$ ;BRANCH IF CORRECT DATA WAS FETCHED
4155 034214 104046 ERROR 46 ;WRONG DATA WAS FETCHED
4156 ;FOR TIGHTER SCOPE LOOP
4157 ;REPLACE ERROR CALL WITH
4158 ;"BR 15$" = 000763
4159 034216 16$: ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
4160 ;
4161 034216 012737 034224 001110 MOV #17$, $LPERR ;SET LOOP ON ERROR POINTER TO 17$.
4162 034224 012737 140340 177776 17$: MOV #140340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT USER
4163 034232 005002 CLR R2 ;MAKE REGISTER 2 A ZERO
4164 034234 006562 100000 MFPI 100000(R2) ;READ FROM PHYSICAL 60000
4165 034240 012601 MOV (USP)+,R1 ;POP USER STACK INTO R1
4166 034242 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
4167 034244 001401 BEQ 18$ ;BRANCH IF CORRECT DATA WAS FETCHED
4168 034246 104046 ERROR 46 ;WRONG DATA WAS FETCHED
4169 ;FOR TIGHTER SCOPE LOOP
4170 ;REPLACE ERROR CALL WITH
4171 ;"BR 17$" = 000766
4172 034250 18$: ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
4173 ;
4174 034250 012737 034256 001110 MOV #19$, $LPERR ;SET LOOP ON ERROR POINTER TO 19$
4175 034256 012737 140340 177776 19$: MOV #140340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT USER
4176 034264 012737 100000 001202 MOV #100000,$TMP2 ;LOAD TEST LOC. VIRT. ADDR. INTO $TMP2
4177 034272 012702 001202 MOV #<TMP2,R2 ;LOAD ADDRESS OF $TMP2 INTO R2
4178 034276 006572 000000 MFPI @0(R2) ;READ FROM PHYSICAL 60000
4179 034302 012601 MOV (USP)+,R1 ;POP USER STACK INTO R1
4180 034304 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
4181 034306 001401 BEQ 20$ ;BRANCH IF CORRECT DATA WAS FETCHED
4182 034310 104046 ERROR 46 ;WRONG DATA WAS FETCHED
4183 ;FOR TIGHTER SCOPE LOOP
4184 ;REPLACE ERROR CALL WITH
4185 ;"BR 19$" = 000762
4186 034312 012737 002400 000250 20$: MOV #MGMERR,MMVEC ;SET M.M. VECTOR TO NORMAL ROUTINE
4187 034320 012737 000340 177776 MOV #00340,PSW ;GO BACK TO KERNEL MODE, PREVIOUS KERNEL
4188 034326 012737 033642 001110 MOV #1$, $LPERR ;SET LOOP POINTER TO START OF TEST
    
```



```

.SBTTL END OF PASS ROUTINE
4285
4286
4287 ;*****
4288 ;*INCREMENT THE PASS NUMBER ($PASS)
4289 ;*TYPE "END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYY"
4290 ;*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
4291 ;*IF SW12=1 INHIBIT TRACE TRAP
4292 ;*IF THERES A MONITOR GO TO IT
4293 ;*IF THERE ISN'T JUMP TO LOOP
4294
4295 $EOP:
4296 SCOPE
4297 CLR $STNM ;;ZERO THE TEST NUMBER
4298 CLR $TIMES ;;ZERO THE NUMBER OF ITERATIONS
4299 INC $PASS ;;INCREMENT THE PASS NUMBER
4300 BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
4301 DEC (PC)+ ;;LOOP?
4302 $EOPCT: .WORD 1
4303 BGT $DOAGN ;;YES
4304 MOV (PC)+,(PC)+ ;;RESTORE COUNTER
4305 $SENDCT: .WORD 1
4306 $EOPCT
4307 TYPE ,65$ ;;TYPE ASCIZ STRING
4308 BR 64$ ;;GET OVER THE ASCIZ
4309 ;:65$ .ASCIZ <12><15>/END PASS #/
4310 64$:
4311 MOV $PASS,-(SP) ;;SAVE $PASS FOR TYPEDOUT
4312 ;;TYPE PASS NUMBER
4313 TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
4314 TYPE ,67$ ;;TYPE ASCIZ STRING
4315 BR 66$ ;;GET OVER THE ASCIZ
4316 ;:67$ .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
4317 66$:
4318 MOV $ERTTL,-(SP) ;;SAVE $ERTTL FOR TYPEDOUT
4319 ;;TOTAL NUMBER OF ERRORS
4320 TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
4321 TYPE ,$SCLRF ;;TYPE CARRIAGE RETURN, LINE FEED
4322 CLR $ERTTL ;;CLEAR ERROR TOTAL
4323 $GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS
4324 BEQ $DOAGN ;;BRANCH IF NO MONITOR
4325 CLR -(SP) ;;INSURE THE "T" BIT IS CLEAR
4326 MOV #$CLR.T,-(SP) ;;SETUP FOR AN RTI OR RTT
4327 BR $RTRN ;;GO DO AN RTI OR RTT TO LOAD THE PSW
4328 ;;WITH A CLEARED "T" BIT
4329 $SCLR.T:
4330 MOV @#42,R0 ;;INSURE R0 CONTAINS THE MONITORS
4331 BEQ $DOAGN ;;RETURN ADDRESS
4332 RESET ;;CLEAR THE WORLD
4333 $SENDAD: JSR PC,(R0) ;;GO TO MONITOR
4334 NOP ;;SAVE ROOM
4335 NOP ;;FOR
4336 NOP ;;ACT11
4337 $DOAGN:
4338 TRAP ;;PUSH OLD PSW AND PC ON STACK
4339 BIC #20,(SP) ;;CLEAR THE "T" BIT
4340 BIT #BIT12,@SWR ;;RUN WITH TRACE TRAP?

```

```

4341 035004 001305 BNE 1$ ;;BR IF NO
4342 035006 005137 035032 COM $TBIT ;;IS IT TIME FOR TRACE TRAP
4343 035012 100402 BMI 1$ ;;BR IF NO
4344 035014 052716 000020 BIS #20,(SP) ;;SET TRACE TRAP
4345 035020 012746 035026 1$: MOV #$LOOP,-(SP) ;;JUMP TO START OF TEST
4346 035024 000002 $RTRN: RTI ;;RETURN--THIS IS CHANGED TO
4347 ;;AN "RTT" IF "RTT" IS A LEGAL
4348 ;;INSTRUCTION
4349 035026 $LOOP:
4350 035026 000137 JMP @(PC)+ ;;RETURN
4351 035030 020466 $RTNAD: .WORD LOOP
4352 035032 000000 $TBIT: .WORD 0 ;;"T" BIT STATE INDICATOR
4353 035034 377 377 000 $ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
4354 035040 .EVEN
4355 .SBTTL SCOPE HANDLER ROUTINE
4356
4357 ;*****
4358 ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
4359 ;*AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
4360 ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
4361 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4362 ;*SW14=1 LOOP ON TEST
4363 ;*SW11=1 INHIBIT ITERATIONS
4364 ;*SW09=1 LOOP ON ERROR
4365 ;*SW08=1 LOOP ON TEST IN SWR<7:0>
4366 ;*CALL
4367 ;* SCOPE ;;SCOPE=IOT
4368
4369 035040 $SCOPE:
4370 035040 104410 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
4371 035042 032777 040000 144070 1$: BIT #BIT14,@SWR ;;LOOP ON PRESENT TEST?
4372 035050 001114 BNE $OVER ;;YES IF SW14=1
4373 ;####START OF CODE FOR THE XOR TESTER####
4374 035052 000 16 $XTSTR: BR 6$ ;;IF RUNNING ON THE "XOR" TESTER CHANGE
4375 ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
4376 035054 013746 000004 MOV @#ERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
4377 035060 012737 035100 000004 MOV #5$,@#ERRVEC ;;SET FOR TIMEDOUT
4378 035066 005737 177080 TST @#177060 ;;TIME OUT ON XOR?
4379 035072 012637 000004 MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
4380 035076 000463 BR $SVLAD ;;GO TO THE NEXT TEST
4381 035100 022626 5$: CMP (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
4382 035102 012637 000004 MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
4383 035106 000423 BR 7$ ;;LOOP ON THE PRESENT TEST
4384 035110 6$:####END OF CODE FOR THE XOR TESTER####
4385 035110 032777 000400 144022 BIT #BIT08,@SWR ;;LOOP ON SPEC. TEST?
4386 035116 001404 BEQ 2$ ;;BR IF NO
4387 035120 127737 144014 001102 CMPB @SWR,$STNM ;;ON THE RIGHT TEST? SWR<7:0>
4388 035126 001465 BEQ $OVER ;;BR IF YES
4389 035130 105737 001103 2$: TSTB $ERFLG ;;HAS AN ERROR OCCURRED?
4390 035134 001421 BEQ 3$ ;;BR IF NO
4391 035136 123737 001115 001103 CMPB $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
4392 035144 101015 BHI 3$ ;;BR IF NO
4393 035146 032777 001000 143764 BIT #BIT09,@SWR ;;LOOP ON ERROR?
4394 035154 001404 BEQ 4$ ;;BR IF NO
4395 035156 013737 001110 001106 7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
4396 035164 000446 BR $OVER

```

```

4397 035166 10537 001103 4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
4398 035172 00537 001212 CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
4399 035176 000415 BR 1$ ;;ESCAPE TO THE NEXT TEST
4400 035200 032777 004000 143732 3$: BIT #BIT11,@SWR ;;INHIBIT ITERATIONS?
4401 035206 001011 BNE 1$ ;;BR IF YES
4402 035210 005737 001234 TST $PASS ;;IF FIRST PASS OF PROGRAM
4403 035214 001406 BEQ 1$ ;; INHIBIT ITERATIONS
4404 035216 005237 001104 INC $ICNT ;;INCREMENT ITERATION COUNT
4405 035222 023737 001212 001104 CMP $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
4406 035230 002024 BGE $OVER ;;BR IF MORE ITERATION REQUIRED
4407 035232 012737 000001 001104 1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
4408 035240 013737 035316 001212 MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
4409 035246 105237 001102 $SVLAD: INCB $TSTNM ;;COUNT TEST NUMBERS
4410 035252 113737 001102 001232 MOV $TSTNM,$TSTN ;;SET TEST NUMBER IN APT MAILBOX
4411 035260 011637 001106 MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
4412 035264 011637 001110 MOV (SP),$LPERR ;;SAVE ERROR LOOP ADDRESS
4413 035270 005037 001214 CLR $ESCAPE ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
4414 035274 112737 000001 001115 MOV #1,$ERRMAX ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
4415 035302 013777 001102 143632 $OVER: MOV $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
4416 035310 013716 001106 MOV $LPADR,(SP) ;;FUDGE RETURN ADDRESS
4417 035314 000002 RTI ;;FIXES PS
4418 035316 000200 $MXCNT: 200 ;;MAX. NUMBER OF ITERATIONS
4419 .SBTTL ERROR HANDLER ROUTINE
4420
4421 ;;*****
4422 *THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
4423 *SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
4424 *AND GO TO ERRTP ON ERROR
4425 *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4426 *C115=1 HALT ON ERROR
4427 *SW13=1 INHIBIT ERROR TYPEOUTS
4428 *SW10=1 BELL ON ERROR
4429 *SW09=1 LOOP ON ERROR
4430 *CALL
4431 *;
4432 *; ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER
4433
4434 $ERROR: CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
4435 MOV R0,$REG0 ;;SAVE THE CONTENTS OF R0
4436 MOV R1,$REG1 ;;SAVE THE CONTENTS OF R1
4437 MOV R2,$REG2 ;;SAVE THE CONTENTS OF R2
4438 MOV R3,$REG3 ;;SAVE THE CONTENTS OF R3
4439 MOV R4,$REG4 ;;SAVE THE CONTENTS OF R4
4440 MOV R5,$REG5 ;;SAVE THE CONTENTS OF R5
4441 MOV $TSTNM,TESTNO ;;SAVE THE TEST NUMBER
4442 7$: INCB $ERFLG ;;SET THE ERROR FLAG
4443 BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO
4444 MOV $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
4445 BIT #BIT10,@SWR ;;BELL ON ERROR?
4446 BEQ 1$ ;;NO - SKIP
4447 TYPE $BELL ;;RING BELL
4448 1$: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
4449 MOV (SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
4450 SUB #2,$ERRPC
4451 MOV $ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
4452 BIT #BIT13,@SWR ;;SKIP TYPEOUT IF SET
    
```

```

4453 035442 001004 BNE 20$ ;;SKIP TYPEOUTS
4454 035444 004737 035556 JSR PC,ERRTP ;;GO TO USER ERROR ROUTINE
4455 035450 104401 001223 TYPE $CRLF
4456 035454
4457 035454 122737 000001 001246 CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
4458 035462 001007 BNE 2$ ;;NO,SKI- APT ERROR REPORT
4459 035464 113737 001114 035476 MOV $ITEMB,21$ ;;SET ITEM NUMBER AS ERROR NUMBER
4460 035472 004737 040110 JSR PC,$ATY4 ;;REPORT FATAL ERROR TO APT
4461 035476 000 21$: .BYTE 0
4462 035477 000 .BYTE 0
4463 035500 000777 BR 22$ ;;APT ERROR LOOP
4464 035502 005777 143432 2$: TST @SWR ;;HALT ON ERROR
4465 035506 100002 BPL 3$ ;;SKIP IF CONTINUE
4466 035510 000000 HALT ;;HALT ON ERROR!
4467 035512 104410 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
4468 035514 032777 001000 143416 3$: BIT #BIT09,@SWR ;;LOOP ON ERROR SWITCH SET?
4469 035522 001402 BEQ 4$ ;;BR IF NO
4470 035524 013716 001110 MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
4471 035530 005737 001214 4$: TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
4472 035534 001402 BEQ 5$ ;;BR IF NONE
4473 035536 013716 001214 MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
4474 035542
4475 035542 022737 034760 000042 5$: CMP #ENDAD,@#42 ;;ACT-11 AUTO-ACCEPT?
4476 035550 001001 BNE 6$ ;;BRANCH IF NO
4477 035552 000000 HALT ;;YES
4478 035554
4479 035554 000002 6$: RTI ;;RETURN
4480 .SBTTL ERROR MESSAGE TYPEOUT ROUTINE
4481
4482 ;;*****
4483 *THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
4484 *ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
4485 *AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
4486 *;
4487 *NOTES:
4488 *1) THIS ROUTINE PROVIDES AN AUTOMATIC "CARRIAGE RETURN-LINE FEED"
4489 * FOR "EM", "DH", AND "DT".
4490 *2) TWO SPACES ARE TYPED AFTER EACH NUMBER FOR "DT"
4491 *3) FOR $ITEMB=0, JUST THE ERROR PC IS TYPED
4492 *4) THE AVAILABLE FORMATS FOR TYPING DATA ARE:
4493 * DF FORMAT
4494 * 0 TYPE A 6 DIGIT OCTAL NUMBER (FROM 16-BIT BINARY)
4495 * 1 TYPE A DECIMAL NUMBER WITHOUT LEADING ZEROS
4496 * 2 TYPE A 16 DIGIT BINARY NUMBER
4497 * 3 TYPE A 6 DIGIT OCTAL NUMBER (FROM 18-BIT BINARY)
4498 *;
4499
4500 $ERRTP: TYPE $CRLF ;;"CARRIAGE RETURN" & "LINE FEED"
4501 035556 104401 001223 MOV R0,-(SP) ;;SAVE R0
4502 035562 010046 CLR R0 ;;PICKUP THE ITEM INDEX
4503 035564 005000 BISB @#$ITEMB,R0
4504 035566 153700 001114 BNE 1$ ;;IF ITEM NUMBER IS ZERO, JUST
4505 035572 001004 1$ TYPE PC OF THE ERROR
4506 MOV $ERRPC,-(SP) ;;SAVE $ERRPC FOR TYPEOUT
4507 035574 013746 001116 ;;ERROR ADDRESS
4508
    
```

```

4509 035600 104402          TYPCC                ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
4510 035602 000471          BR 13$              ;:GET OUT
4511 035604 005300          1$: DEC R0          ;:ADJUST THE INDEX SO THAT IT WILL
4512 035606 006300          ASL R0              ;:
4513 035610 006300          ASL R0              ;: WORK FOR THE ERROR TABLE
4514 035612 006300          ASL R0              ;:
4515 035614 062700          ADD #ERRTB,R0      ;:FORM TABLE POINTER
4516 035620 012037          MOV (R0)+,2$      ;:PICKUP "ERROR MESSAGE" POINTER
4517 035624 001404          BEQ 3$             ;:SKIP TYPEOUT IF NO POINTER
4518 035626 104401          TYPE               ;:TYPE THE "ERROR MESSAGE"
4519 035630 000000          2$: .WORD 0        ;:"ERROR MESSAGE" POINTER GOES HERE
4520 035632 104401          TYPE ,%$RLF       ;:"CARRIAGE RETURN" & "LINE FEED"
4521 035636 012037          3$: MOV (R0)+,4$   ;:PICKUP "DATA HEADER" POINTER
4522 035642 001404          BEQ 5$            ;:SKIP TYPEOUT IF 0
4523 035644 104401          TYPE               ;:TYPE THE "DATA HEADER"
4524 035646 000000          4$: .WORD 0        ;:"DATA HEADER" POINTER GOES HERE
4525 035650 104401          TYPE ,%$RLF       ;:"CARRIAGE RETURN" & "LINE FEED"
4526 035654 010146          5$: MOV R1,-(SP)   ;:SAVE R1
4527 035656 012001          MOV (R0)+,R1      ;:PICKUP "DATA TABLE" POINTER
4528 035660 001441          BEQ 12$           ;:BR IF NO DATA TO BE TYPED
4529 035662 012000          MOV (R0)+,R0      ;:PICKUP "DATA FORMAT" POINTER
4530 035664 105710          6$: TSTB (R0)      ;:IS IT FORMAT 0?
4531 035666 001003          BNE 7$            ;:BR IF NO
4532
4533          ;*THIS CODE IS FOR OCTAL (16-BIT) FORMAT (DF=0)
4534 035670 013146          MOV @{R1)+,-(SP) ;:SAVE @{R1)+ FOR TYPEOUT
4535 035672 104402          TYPCC              ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
4536 035674 000425          BR 11$
4537
4538          ;*THIS CODE IS FOR DECIMAL FORMAT (DF=1)
4539 035676 121027          7$: CMPB (R0),#1   ;:IS IT FORMAT 1?
4540 035702 001003          BNE 8$            ;:BRANCH IF NO
4541 035704 013146          MOV @{R1)+,-(SP) ;:SAVE @{R1)+ FOR TYPEOUT
4542 035706 104.05          TYPDS              ;:GO TYPE--DECIMAL ASCII WITH SIGN
4543 035710 000417          BR 11$
4544
4545          ;*THIS CODE IS FOR BINARY FORMAT (DF=2)
4546 035712 121027          8$: CMPB (R0),#2   ;:IS IT FORMAT 2?
4547 035716 001003          BNE 9$            ;:BRANCH IF NO
4548 035720 013146          MOV @{R1)+,-(SP) ;:SAVE @{R1)+ FOR TYPEOUT
4549 035722 104406          TYPBN              ;:GO TYPE--BINARY ASCII
4550 035724 000411          BR 11$
4551
4552          ;*THIS CODE IS FOR OCTAL (18-BIT) FORMAT (DF=3)
4553 035726 012146          9$: MOV (R1)+,-(SP) ;:PUT ADDRESS OF FIRST LOC. ON STACK
4554 035730 004737          JSR PC,$DB20      ;:CONVERT TWO LOCS. TO AN ASCII STRING
4555 035734 062716          ADD #5,(SP)        ;:ONLY NEED 6 CHARACTERS NOT 11
4556 035740 012637          MOV (SP)+,10$     ;:PUT ADDRESS OF ASCII CHARS. AT 10$
4557 035744 104401          TYPE               ;:TYPE OCTAL VALUE OF 18-BIT BINARY NO.
4558 035746 000000          10$: .WORD 0
4559
4560 035750 005711          11$: TST (R1)      ;:IS THERE ANOTHER NUMBER?
4561 035752 001404          BEQ 12$           ;:BR IF NO
4562 035754 104401          TYPE ,14$         ;:TYPE TWO(2) SPACES
4563 035760 105720          TSTB (R0)+       ;:POINT TO NEW "DATA FORMAT"
4564 035762 000740          BR 6$            ;:LOOP
    
```

```

4565
4566 035764 012601          12$: MOV (SP)+,R1  ;:RESTORE R1
4567 035766 012600          13$: MOV (SP)+,R0  ;:RESTORE R0
4568 035770 104401          TYPE ,%$RLF       ;:"CARRIAGE RETURN" & "LINE FEED"
4569 035774 000207          RTS PC            ;:RETURN
4570 035776 020040          14$: .ASCIZ / /    ;:TWO(2) SPACES
4571 036002
4572
    
```

```
4573 .SBTTL ***** SUBROUTINES USED BY THIS PROGRAM *****
4574
4575 .SBTTL TURN OFF T-BIT AND SAVE CURRENT PSW
4576 ;*****
4577 ;*
4578 ;* THIS SUBROUTINE IS USED TO TURN OFF THE TRACE TRAP BIT IN THE PSW
4579 ;* IF IT IS ON. THE PROCESSOR STATUS IS SAVED IN "TBITPS" SO THAT
4580 ;* THE PSW CAN BE RESTORED TO ITS PREVIOUS CONDITION WHEN CONDITIONS
4581 ;* WARRANT T-BIT TRAPPING.
4582 ;*
4583 ;*****
4584 036002 033727 177776 000020 TOFF: BIT PSW,#TBIT ;IS THE T-BIT SET IN THE PSW?
4585 036010 001411 BEQ 1$ ;EXIT IF NO
4586 036012 013746 177776 MOV PSW,-(SP) ;PUSH PRESENT PSW ON THE STACK
4587 036016 011637 001372 MOV (SP),TBITPS ;ALSO SAVE IT IN "TBITPS" FOR
4588 ;RESTORING LATER
4589 036022 042716 000020 BIC #TBIT,(SP) ;CLEAR THE T-BIT (BIT 4) IN THE PSW
4590 036026 012746 036034 MOV #1$,-(SP) ;PUSH PC OF "RTS" ON STACK
4591 036032 000006 RTT ;"RETURN" TO 1$ WITH T-BIT OFF
4592 036034 000207 RTS PC ;RETURN TO PROGRAM
4593
4594 .SBTTL TURN ON T-BIT AND RESTORE PREVIOUS PSW
4595 ;*****
4596 ;*
4597 ;* THIS SUBROUTINE IS USED TO RESTORE THE PROCESSOR STATUS TO ITS
4598 ;* PREVIOUS CONDITION BY RESTORING THE "T-BIT PSW" SAVED BY THE
4599 ;* "TOFF" SUBROUTINE IN THE "TBITPS" LOCATION.
4600 ;*
4601 ;*****
4602 036036 033727 001372 000020 TON: BIT TBITPS,#TBIT ;WAS T-BIT ON IN THE PREVIOUS PSW?
4603 036044 001410 BEQ 1$ ;EXIT IF NO
4604 036046 013746 001372 MOV TBITPS,-(SP) ;PUSH PREVIOUS PSW ON THE STACK
4605 036052 012737 000340 001372 MOV #340,TBITPS ;RESET THE "TBITPS" LOCATION
4606 036060 012746 036066 MOV #1$,-(SP) ;PUSH PC OF "RTS" ON STACK
4607 036064 000006 RTT ;"RETURN" TO 1$ WITH T-BIT RESTORED
4608 036066 000207 RTS PC ;RETURN TO PROGRAM
4609
4610 .SBTTL SET ALL WRITEABLE BITS IN ALL PAR/PDR'S
4611 ;*****
4612 ;*
4613 ;* THIS SUBROUTINE IS USED BY THE PAR/PDR DUAL ADDRESSING TEST
4614 ;* TO SET ALL WRITEABLE BITS IN ALL KERNEL AND USER PAR'S AND
4615 ;* PDR'S TO A 1. THE "INITIAL STATE" OF HAVING ALL BITS=1 IS
4616 ;* USED TO SEE THAT ONLY ONE REGISTER IS CLEARED IN RESPONSE TO
4617 ;* A SINGLE PAR OR PDR ADDRESS.
4618 ;*
4619 ;*****
4620
4621 036070 012702 000010 SETREG: MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8
4622 036074 012701 172300 MOV #KIPDR0,R1 ;LOAD ADDRESS OF FIRST PDR INTO R1
4623 036100 012721 177777 1$: MOV #-1,(R1)+ ;SET BITS IN KERNEL PDR TO 1
4624 036104 077203 SOB R2,1$ ;LOOP TO 1$ UNTIL ALL KERNEL PDR'S LOADED
4625 036106 012702 000010 MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8
4626 036112 012701 172340 MOV #KIPAR0,R1 ;LOAD ADDRESS OF FIRST PAR INTO R1
4627 036116 012721 177777 2$: MOV #-1,(R1)+ ;SET BITS IN A KERNEL PAR TO 1
4628 036122 077203 SOB R2,2$ ;LOOP TO 2$ UNTIL ALL KERNEL PAR'S LOADED
```

```
4629 036124 012702 000010 MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8
4630 036130 012701 176000 MOV #UIPDR0,R1 ;LOAD ADDRESS OF FIRST PDR INTO R1
4631 036134 012721 177777 3$: MOV #-1,(R1)+ ;SET BITS IN A USER PDR TO 1
4632 036140 077203 SOB R2,3$ ;LOOP TO 3$ UNTIL ALL USER PDR'S LOADED
4633 036142 012702 000010 MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8
4634 036146 012701 176400 MOV #UIPAR0,R1 ;LOAD ADDRESS OF FIRST PAR INTO R1
4635 036152 012721 177777 4$: MOV #-1,(R1)+ ;SET BITS IN A USER PAR TO 1
4636 036156 077203 SOB R2,4$ ;LOOP TO 4$ UNTIL ALL USER PAR'S LOADED
4637 036160 000207 RTS PC ;RETURN TO TEST
4638
4639 .SBTTL READ & COMPARE KERNEL & USER PAR/PDR'S
4640 ;*****
4641 ;*
4642 ;* THIS SUBROUTINE IS USED BY PAR/PDR DUAL ADDRESSING TEST TO
4643 ;* READ ALL THE PAR'S AND PDR'S TO SEE THAT ONLY ONE REGISTER
4644 ;* WAS CLEARED IN RESPONSE TO A SINGLE PAR OR PDR ADDRESS.
4645 ;* ANY FAILURES FOUND BY THE PAR/PDR DUAL ADDRESSING TEST WILL
4646 ;* BE REPORTED BY THIS SUBROUTINE.
4647 ;*
4648 ;*****
4649 CMPREG:
4650 036162 MOV #KIPDR0,R1 ;LOAD ADDRESS OF FIRST KERNEL PDR IN R1
4651 036166 012701 172300 MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
4652 036172 012705 077416 MOV #77416,R5 ;PUT EXPECTED PDR CONTENTS IN R5
4653 036176 011102 1$: MOV (R1),R2 ;READ A KERNEL PDR INTO R2
4654 036200 020205 CMP R2,R5 ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
4655 036202 001403 BEQ 2$ ;BRANCH IF YES
4656 036204 020100 CMP R1,R0 ;WAS IT THE REG. THAT WAS CLEARED?
4657 036206 001401 BEQ 2$ ;BRANCH IF YES
4658 036210 104016 ERROR 16 ;A PDR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
4659 ;FOR TIGHTER SCOPE LOOP
4660 ;REPLACE ERROR CALL WITH
4661 ;AN "RTS PC" = 000207
4662 036212 062101 000002 2$: ADD #2,R1 ;FORM NEXT KERNEL PDR ADDRESS
4663 036216 077411 SOB R4,1$ ;LOOP TO 1$ UNTIL ALL KERNEL PDR'S CHECKED
4664 036220 012701 172340 MOV #KIPAR0,R1 ;LOAD ADDRESS OF FIRST KERNEL PAR IN R1
4665 036224 012704 000010 MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
4666 036230 012705 007777 3$: MOV #7777,R5 ;PUT EXPECTED PAR CONTENTS IN R5
4667 036234 011102 MOV (R1),R2 ;READ A KERNEL PAR INTO R2
4668 036236 020205 CMP R2,R5 ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
4669 036240 001403 BEQ 4$ ;BRANCH IF YES
4670 036242 020100 CMP R1,R0 ;WAS IT THE REG. THAT WAS CLEARED?
4671 036244 001401 BEQ 4$ ;BRANCH IF YES
4672 036246 104016 ERROR 16 ;A PAR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
4673 ;FOR TIGHTER SCOPE LOOP
4674 ;REPLACE ERROR CALL WITH
4675 ;AN "RTS PC" = 000207
4676 036250 062701 000002 4$: ADD #2,R1 ;FORM NEXT KERNEL PAR ADDRESS
4677 036254 077411 SOB R4,3$ ;LOOP TO 3$ UNTIL ALL KERNEL PAR'S CHECKED
4678 036256 012701 176000 MOV #UIPDR0,R1 ;LOAD ADDRESS OF FIRST USER PDR IN R1
4679 036262 012704 000010 MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
4680 036266 012705 077416 5$: MOV #77416,R5 ;PUT EXPECTED PDR CONTENTS IN R5
4681 036272 011102 MOV (R1),R2 ;READ A USER PDR INTO R2
4682 036274 020205 CMP R2,R5 ;ARE ALL WRITABLE BITS SET AS EXPECTED?
4683 036276 001403 BEQ 6$ ;BRANCH IF YES
4684 036300 020100 CMP R1,R0 ;WAS IT THE REG. THAT WAS CLEARED?
```

```

4685 036302 001401      BEQ      6$          ;BRANCH IF YES
4686 036304 104016      ERROR    16          ;A PDR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
4687                                ;FOR TIGHTER SCOPE LOOP
4688                                ;REPLACE ERROR CALL WITH
4689                                ;AN "RTS PC" = 000207
4690 036306 062701 000002 6$:      ADD      #2,R1    ;FORM NEXT USER PDR ADDRESS
4691 036312 077411      SOB      R4,$$       ;LOOP TO 5$ UNTIL ALL USER PDR'S CHECKED
4692 036314 012701 177640      MOV      #UIPAR0,R1  ;LOAD ADDRESS OF FIRST USER PAR IN R1
4693 036320 012704 000010      MOV      #10,R4      ;LOAD LOOP COUNTER WITH AN 8
4694 036324 012705 007777      MOV      #7777,R5    ;PUT EXPECTED PAR CONTENTS IN R5
4695 036330 011102      7$:      MOV      (R1),R2     ;READ A USER PAR INTO R2
4696 036332 020205      CMP      R2,R5       ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
4697 036334 001403      BEQ      6$          ;BRANCH IF YES
4698 036336 020100      CMP      R1,R0       ;WAS IT THE REG. THAT WAS CLEARED?
4699 036340 001401      BEQ      6$          ;BRANCH IF YES
4700 036342 104016      ERROR    16          ;A PAR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
4701                                ;FOR TIGHTER SCOPE LOOP
4702                                ;REPLACE ERROR CALL WITH
4703                                ;AN "RTS PC" = 000207
4704 036344 062701 000002 6$:      ADD      #2,R1    ;FORM NEXT USER PAR ADDRESS
4705 036350 077411      SOB      R4,7$       ;LOOP TO 7$ UNTIL ALL USER PAR'S CHECKED
4706 036352 000207      RTS      PC          ;RETURN TO TEST
    
```

.SBTTL CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS

```

;*****
;*
;* THIS SUBROUTINE IS USED TO FORM AN 18-BIT PHYSICAL ADDRESS
;* (PBA) FROM THE 16-BIT VIRTUAL ADDRESS (VBA) AND THE APPROPRIATE
;* PAGE ADDRESS REGISTER (PAR). THE SAME METHOD USED BY THE MEMORY
;* MANAGEMENT LOGIC IS USED. VBA <15:13> SELECTS WHICH PAR/PDR
;* IS TO BE USED, VBA <5:0>+PBA <5:0>, AND VBA <12:6> IS ADDED
;* TO PAR <11:00> TO GIVE PBA <17:6>. BITS <17:16> OF THE
;* PHYSICAL ADDRESS ARE LEFT IN LOC. "PBAHI" AND BITS <15:00>
;* ARE LEFT IN LOC. "PBALO". THE PSW'S "CURRENT MODE" BITS
;* ARE USED TO SELECT THE KERNEL OR USER PAR/PDR'S. THE ROUTINE
;* IS ENTERED WITH LOC. "VIRT1" CONTAINING THE 16-BIT VIRTUAL
;* ADDRESS.
;*****
4725 036354 012702 172340      FORMPA: MOV      #KIPAR0,R2    ;LOAD ADDRESS OF FIRST KERNEL PAR IN R2
4726 036360 032737 140000 177776 BIT      #140000,PSW        ;IN USER MODE?
4727 036366 001402      BEQ      1$          ;BRANCH IF NO
4728 036370 012702 177640      MOV      #UIPAR0,R2    ;LOAD ADDRESS OF FIRST USER PAR IN R2
4729 036374 013700 001402      1$:      MOV      VIRT1,R0       ;LOAD VIRTUAL ADDR. (VBA) INTO R0
4730 036400 072027 177754      ASH      #-14,R0      ;GET BITS <15:13> DOWN TO BITS <3:1>
4731 036404 042700 177761      BIC      #177761,R0    ;MASK OF ALL BITS BUT BITS <3:1>
4732 036410 050002      ADD      (R2),R0      ;ADD OFFSET TO BASE PAR ADDRESS
4733 036412 011200      MOV      (R2),R0      ;GET BITS <11:00> FROM APPROPRIATE PAR
4734 036414 010002      MOV      R0,R2        ;COPY PAR BITS <11:00> INTO R2
4735 036416 013737 001402 001406 MOV      VIRT1,PBALO    ;PUT VIRTUAL ADDR. IN LOC. "PBALO"
4736 036424 042737 160000 001406 BIC      #160000,PBALO  ;CLEAR OFF BITS <15:13> OF ORIGINAL VBA
4737 036432 072227 177766      ASH      #-12,R2     ;GET PAR <11:00> DOWN TO BITS <1:0> OF R2
4738 036436 042702 177774      BIC      #177774,R2   ;CLEAR OFF ALL BITS BUT BITS <1:0>
4739 036442 072027 000006      ASH      #6,R0        ;SHIFT PAR<9:0> TO <15:6> OF R0
4740 036446 042700 000077      BIC      #77,R0       ;CLEAR BITS <5:0> OF R0
    
```

```

4741 036452 060037 001406      ADD      R0,PBALO    ;IN EFFECT, ADD VBA<12:0> TO PAR<9:0>
4742                                ;{PAR<9:0> IN BITS <15:6> OF R0}
4743 036456 005502      ADC      R2          ;ADD ANY CARRY TO R2
4744 036460 010237 001410      MOV      #2,PBAHI    ;PUT BITS <17:16> OF PHYSICAL ADDR. IN PBAHI
4745 036464 000207      RTS      PC          ;RETURN TO PROGRAM
4746
    
```

```

4747 .SBTTL TTY INPUT ROUTINE
4748
4749 ;*****
4750 .ENABL LSB
4751
4752 ;*****
4753 ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
4754 ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
4755 ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
4756 ;*WHEN OPERATING IN TTY FLAG MODE.
4757 036466 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ;;IS THE SOFT-SWR SELECTED?
4758 036474 001114 BNE 15$ ;;BRANCH IF NO
4759 036476 105777 142442 TSTB @STKS ;;CHAR THERE?
4760 036502 100111 BPL 15$ ;;IF NO, DON'T WAIT AROUND
4761 036504 117746 142436 MOVB @STKB,-(SP) ;;SAVE THE CHAR
4762 036510 042716 177600 BIC #'C177,(SP) ;;STRIP-OFF THE ASCII
4763 036514 022726 000007 CMP #7,(SP)+ ;;IS IT A CONTROL G?
4764 036520 001102 BNE 15$ ;;NO, RETURN TO USER
4765 036522 123727 001134 000001 CMPB $AUTOB,#1 ;;ARE WE RUNNING IN AUTO-MODE?
4766 036530 001476 BEQ 15$ ;;BRANCH IF YES
4767
4768 036532 104401 037430 $GTSWR: TYPE ,SCNTLG ;;ECHO THE CONTROL-G (^G)
4769 036536 104401 037435 ,SMSWR ;;TYPE CURRENT CONTENTS
4770 036542 013746 000176 MOV SWREG,-(SP) ;;SAVE SWREG FOR TYPEDUT
4771 036546 104402 TYPDC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
4772 036550 104401 037446 TYPE ,SMNEW ;;PROMPT FOR NEW SWR
4773 036554 005046 19$: CLR -(SP) ;;CLEAR COUNTER
4774 036556 005046 CLR -(SP) ;;THE NEW SWR
4775 036560 105777 142360 7$: TSTB @STKS ;;CHAR THERE?
4776 036564 100375 BPL 7$ ;;IF NOT TRY AGAIN
4777
4778 036566 117746 142354 MOVB @STKB,-(SP) ;;PICK UP CHAR
4779 036572 042716 177600 BIC #'C177,(SP) ;;MAKE IT 7-BIT ASCII
4780
4781 036576 021627 000003 CMP (SP),#3 ;;IS IT A CONTROL-C?
4782 036602 001015 BNE 9$ ;;BRANCH IF NOT
4783 036604 104401 037416 TYPE ,SCNTLC ;;YES, ECHO CONTROL-C (^C)
4784 036610 062706 000006 ADD #6,SP ;;CLEAN UP STACK
4785 036614 123727 001135 000001 CMPB $INTAG,#1 ;;REENABLE TTY KEYBOARD INTERRUPTS?
4786 036622 001003 BNE 8$ ;;BRANCH IF NO
4787 036624 012777 000100 142312 MOV #100,@STKS ;;ALLOW TTY KEYBOARD INTERRUPTS
4788 036632 000137 037460 8$: JMP CNTRLC ;;CONTROL-C RESTART
4789
4790
4791 036636 021627 000025 9$: CMP (SP),#25 ;;IS IT A CONTROL-U?
4792 036642 001005 BNE 10$ ;;BRANCH IF NOT
4793 036644 104401 037423 TYPE ,SCNTLU ;;YES, ECHO CONTROL-U (^U)
4794 036650 062706 000006 20$: ADD #6,SP ;;IGNORE PREVIOUS INPUT
4795 036654 000737 BR 19$ ;;LET'S TRY IT AGAIN
4796
4797
4798 036656 021627 000015 10$: CMP (SP),#15 ;;IS IT A <CR>?
4799 036662 001022 BNE 16$ ;;BRANCH IF NO
4800 036664 005766 000004 TST 4(SP) ;;YES, IS IT THE FIRST CHAR?
4801 036670 001403 BEQ 11$ ;;BRANCH IF YES
4802 036672 016677 000002 142240 MOV 2(SP),@SWR ;;SAVE NEW SWR
    
```

```

4803 036700 062706 000006 11$: ADD #6,SP ;;CLEAR UP STACK
4804 036704 104401 001223 14$: TYPE ,SCRLF ;;ECHO <CR> AND <LF>
4805 036710 123727 001135 000001 CMPB $INTAG,#1 ;;RE-ENABLE TTY KBD INTERRUPTS?
4806 036716 001003 BNE 15$ ;;BRANCH IF NOT
4807 036720 012777 000100 142216 15$: MOV #100,@STKS ;;RE-ENABLE TTY KBD INTERRUPTS
4808 036726 000302 RTI ;;RETURN
4809 036730 004737 040022 16$: JSR PC,$TYPEC ;;ECHO CHAR
4810 036734 021627 000060 CMP (SP),#60 ;;CHAR < 0?
4811 036740 002420 BLT 18$ ;;BRANCH IF YES
4812 036742 021627 000067 CMP (SP),#67 ;;CHAR > ??
4813 036746 003015 BGT 18$ ;;BRANCH IF YES
4814 036750 042726 000060 BIC #60,(SP)+ ;;STRIP-OFF ASCII
4815 036754 005766 000002 TST 2(SP) ;;IS THIS THE FIRST CHAR
4816 036760 001133 BEQ 17$ ;;BRANCH IF YES
4817 036762 006316 ASL (SP) ;;NO, SHIFT PRESENT
4818 036764 006316 ASL (SP) ;; CHAR OVER TO MAKE
4819 036766 006316 ASL (SP) ;; ROOM FOR NEW ONE.
4820 036770 005266 000002 17$: INC 2(SP) ;;KEEP COUNT OF CHAR
4821 036774 056616 177776 BIS -2(SP),(SP) ;;SET IN NEW CHAR
4822 037000 000667 BR 7$ ;;GET THE NEXT ONE
4823 037002 104401 001222 18$: TYPE ,SQUES ;;TYPE ?<CR><LF>
4824 037006 000720 BR 20$ ;;SIMULATE CONTROL-U
4825
4826 .DSABL LSB
4827
4828 ;*****
4829 ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
4830 ;*CALL:
4831 ;* RDCHR ;;INPUT A SINGLE CHARACTER FROM THE TTY
4832 ;* RETURN HERE ;;CHARACTER IS ON THE STACK
4833 ;* ;;WITH PARITY BIT STRIPPED OFF
4834 ;*
4835
4836 037010 011646 $RDCHR: MOV (SP),-(SP) ;;PUSH DOWN THE PC
4837 037012 016666 MOV 4(SP),2(SP) ;;SAVE THE PS
4838 037020 105777 142120 1$: TSTB @STKS ;;WAIT FOR
4839 037024 100375 BPL 1$ ;;A CHARACTER
4840 037026 117766 142114 000004 MOVB @STKB,4(SP) ;;READ THE TTY
4841 037034 042766 177600 000004 BIC #'C<177>,4(SP) ;;GET RID OF JUNK IF ANY
4842 037042 026627 000004 000023 CMP 4(SP),#23 ;;IS IT A CONTROL-S?
4843 037050 001013 BNE 3$ ;;BRANCH IF NO
4844 037052 105777 142066 2$: TSTB @STKS ;;WAIT FOR A CHARACTER
4845 037056 100375 BPL 2$ ;;LOOP UNTIL ITS THERE
4846 037060 117746 142062 MOVB @STKB,-(SP) ;;GET CHARACTER
4847 037064 042716 177600 BIC #'C177,(SP) ;;MAKE IT 7-BIT ASCII
4848 037070 026627 000021 CMP (SP)+,#21 ;;IS IT A CONTROL-Q?
4849 037074 001366 BNE 2$ ;;IF NOT DISCARD IT
4850 037076 000750 BR 1$ ;;YES, RESUME
4851 037100 026627 000004 000140 3$: CMP 4(SP),#140 ;;IS IT UPPER CASE?
4852 037106 002407 BLT 4$ ;;BRANCH IF YES
4853 037110 026627 000004 000175 CMP 4(SP),#175 ;;IS IT A SPECIAL CHAR?
4854 037116 003003 BGT 4$ ;;BRANCH IF YES
4855 037120 042766 000004 000004 BIC #40,4(SP) ;;MAKE IT UPPER CASE
4856 037128 000002 4$: RTI ;;GO BACK TO USER
4857 ;*****
4858 ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
    
```



```

4859 ;*CALL:
4860 ;* RDLIN ;:INPUT A STRING FROM THE TTY
4861 ;* RETURN HERE ;:ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
4862 ;* ;:TERMINATOR WILL BE A BYTE OF ALL 0'S
4863
4864 037130 010346 SRDLIN: MOV R3,-(SP) ;:SAVE R3
4865 037132 005046 CLR -(SP) ;:CLEAR THE RUBOUT KEY
4866 037134 012703 037406 1$: MOV #TTYIN,R3 ;:GET ADDRESS
4867 037140 022703 037416 2$: CMP #TTYIN+8.,R3 ;:BUFFER FULL?
4868 037144 101467 BLOS 4$ ;:BR IF YES
4869 037146 104411 RDCHR ;:GO READ ONE CHARACTER FROM THE TTY
4870 037150 112613 MOVB (SP)+,(R3) ;:GET CHARACTER
4871 037152 122713 000003 3$: CNPB #3,(R3) ;:IS IT A CONTROL-C?
4872 037156 001006 BNE 10$ ;:BRANCH IF NO
4873 037160 104401 037416 TYPE ,SCNTLC ;:TYPE A CONTROL-C (^C)
4874 037164 005726 TST (SP)+ ;:CLEAN RUBOUT KEY OFF OF THE STACK
4875 037166 012603 MOV (SP)+,R3 ;:RESTORE R3
4876 037170 000137 037460 JMP CNTRLC ;:GOTO CONTROL-C RESTART
4877 037174 122713 000177 10$: CNPB #177,(R3) ;:IS IT A RUBOUT
4878 037200 001022 BNE 5$ ;:BR IF NO
4879 037202 005716 TST (SP) ;:IS THIS THE FIRST RUBOUT?
4880 037204 001007 BNE 6$ ;:BR IF NO
4881 037206 112737 000134 037404 MOVB #' \,9$ ;:TYPE A BACK SLASH
4882 037214 104401 037404 TYPE ,9$
4883 037220 012716 177777 MOV #-1,(SP) ;:SET THE RUBOUT KEY
4884 037224 005303 6$: DEC R3 ;:BACKUP BY ONE
4885 037226 020327 037406 CMP R3,#TTYIN ;:STACK EMPTY?
4886 037232 103434 BLO 4$ ;:BR IF YES
4887 037234 111337 037404 MOVB (R3),9$ ;:SETUP TO TYPEOUT THE DELETED CHAR.
4888 037240 104401 037404 TYPE ,9$ ;:GO TYPE
4889 037244 000735 BR 2$ ;:GO READ ANOTHER CHAR.
4890 037246 005716 5$: TST (SP) ;:RUBOUT KEY SET?
4891 037250 001406 BEQ 7$ ;:BR IF NO
4892 037252 112737 000134 037404 MOVB #' \,9$ ;:TYPE A BACK SLASH
4893 037260 104401 037404 TYPE ,9$
4894 037264 005016 CLR (SP) ;:CLEAR THE RUBOUT KEY
4895 037266 122713 000025 7$: CNPB #25,(R3) ;:IS CHARACTER A CTRL U?
4896 037272 001003 BNE 8$ ;:BR IF NO
4897 037274 104401 037423 TYPE ,SCNTLU ;:TYPE A CONTROL "U"
4898 037300 000715 BR 1$ ;:GO START OVER
4899 037302 122713 000022 8$: CNPB #22,(R3) ;:IS CHARACTER A "R"?
4900 037306 001011 BNE 3$ ;:BRANCH IF NO
4901 037310 105013 CLRB (R3) ;:CLEAR THE CHARACTER
4902 037312 104401 001223 TYPE ,$CRLF ;:TYPE A "CR" & "LF"
4903 037316 104401 037406 TYPE ,TTYIN ;:TYPE THE INPUT STRING
4904 037322 000706 BR 2$ ;:GO PICKUP ANOTHER CHACTER
4905 037324 104401 001222 4$: TYPE ,SQUES ;:TYPE A '?'
4906 037330 000701 BR 1$ ;:CLEAR THE BUFFER AND LOOP
4907 037332 111337 037404 3$: MOVB (R3),9$ ;:ECHO THE CHARACTER
4908 037336 104401 037404 TYPE ,9$
4909 037342 122723 000015 CNPB #15,(R3)+ ;:CHECK FOR RETURN
4910 037346 001274 BNE 2$ ;:LOOP IF NOT RETURN
4911 037350 105963 177777 CLRB -(R3) ;:CLEAR RETURN (THE 15)
4912 037354 104401 001224 TYPE ,LF ;:TYPE A LINE FEED
4913 037360 005726 TST (SP)+ ;:CLEAN RUBOUT KEY FROM THE STACK
4914 037362 012603 MOV (SP)+,R3 ;:RESTORE R3
    
```

```

4915 037364 011646 MOV (SP),-(SP) ;:ADJUST THE STACK AND PUT ADDRESS OF THE
4916 037366 016366 000004 000002 MOV 4(SP),2(SP) ;: FIRST ASCII CHARACTER ON IT
4917 037374 012766 037406 000004 MOV #TTYIN,4(SP)
4918 037402 000002 RTI ;:RETURN
4919 037404 000 9$: .BYTE 0 ;:STORAGE FOR ASCII CHAR. TO TYPE
4920 037405 000 .BYTE 0 ;:TERMINATOR
4921 037406 000010 .BLKB 8. ;:RESERVE 8 BYTES FOR TTY INPUT
4922 037416 041536 005015 000 SCNTLC: .ASCIZ /~C/<15><12> ;:CONTROL "C"
4923 037423 136 006525 000012 SCNTLU: .ASCIZ /~U/<15><12> ;:CONTROL "U"
4924 037430 043536 005015 000 SCNTLG: .ASCIZ /~G/<15><12> ;:CONTROL "G"
4925 037435 015 051412 051127 SMSWR: .ASCIZ <15><12>/SWR = /
4926 037442 036440 000040
4927 037446 020040 042516 020127 $MNEW: .ASCIZ / NEW = /
4928 037454 020075 000
4929 037460 .EVEN
4930
4931 .SBTTL CONTROL-C SERVICING ROUTINE
4932
4933 ;* THE FOLLOWING CODE IS EXECUTED WHEN A CONTROL-C HAS
4934 ;* BEEN TYPED INSTEAD OF A NEW SWITCH REG. VALUE.
    
```

```

4935 ;* (IN OTHER WORDS, AFTER A CONTROL-G WAS TYPED).
4936 ;* A NEW SWITCH REG. VALUE WILL BE ASKED FOR.
4937 ;* THE TEST NUMBER AND PASS NUMBER WILL BE TYPED.
4938 ;* AND THEN THE PROGRAM WILL GO TO "END-OF-PASS" AND CONTINUE
4939
4940 037460 013737 001234 001210 CNTRLC: MOV $PASS,$TMP5 ;GET THE VALUE OF "$PASS"
4941 037466 005237 001210 INC $TMP5 ;FORM CURRENT PASS NO.
4942 037472 104401 037537 TYPE ,CMSG ;TYPE THE TEST STOPS MESSAGE
4943 037476 113737 001102 037532 MOV $STNM,1$ ;SAVE THE TEST NUMBER
4944 037504 013746 037532 MOV 1$,-(SP) ;SAVE 1$ FOR TYPEOUT
4945 037510 104402 TYPDC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
4946 037512 104401 037534 TYPE ,2$ ;TYPE 2 SPACES
4947 037516 013746 001210 MOV $TMP5,-(SP) ;SAVE $TMP5 FOR TYPEOUT
4948 037522 104:05 TYPDS ;GO TYPE--DECIMAL ASCII WITH SIGN
4949 037524 104407 GTSWR ;ASK FOR NEW SWR VALUE
4950 037526 000137 034554 JMP $EOP+2 ;CONTINUE AT $EOP+2
4951 037532 000000 1$: .WORD 0 ;BUFFER FOR TEST NUMBER
4952 037534 020040 000 2$: .ASCIZ / / ;TWO SPACES AND THE STOP MESSAGE
4953 037537 112 046525 044520 CMSG: .ASCII /JUMPING TO END-OF-PASS/<15><12>
4954 037544 043516 052040 020117
4955 037552 047105 026504 043117
4956 037560 050055 051501 006523
4957 037566 012
4958 037567 124 051505 047124 .ASCIZ 'TESTNO PASSNO/<15><12>'
4959 037574 020117 050040 051501
4960 037602 047123 006517 000012
4961 .EVEN
4962 .SBTTL TYPE ROUTINE
4963
4964 *****
4965 ;ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
4966 ;THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
4967 ;NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
4968 ;NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
4969 ;NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
4970 ;
4971 ;CALL:
4972 ;*1) USING A TRAP INSTRUCTION
4973 ;* TYPE ,MESADR ;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
4974 ;OR
4975 ;* TYPE
4976 ;* MESADR
4977 ;*
4978
4979 037610 105737 001157 $TYPE: TSTB $TPFLG ;IS THERE A TERMINAL?
4980 037614 100002 BPL 1$ ;BR IF YES
4981 037616 000000 HALT ;HALT HERE IF NO TERMINAL
4982 037620 000430 BR 3$ ;LEAVE
4983 037622 010046 1$: MOV RO,-(SP) ;SAVE RO
4984 037624 017600 000002 MOV @2(SP),RO ;GET ADDRESS OF ASCIZ STRING
4985 037630 122737 000001 001246 CMPB #APTENV,$ENV ;RUNNING IN APT MODE
4986 037636 001011 BNE 62$ ;NO,GO CHECK FOR APT CONSOLE
4987 037640 132737 000100 001247 BITB #APTSPOOL,$ENVM ;SPOOL MESSAGE TO APT
4988 037646 001405 BEQ 62$ ;NO,GO CHECK FOR CONSOLE
4989 037650 010037 037650 MOV RO,61$ ;SETUP MESSAGE ADDRESS FOR APT
4990 037654 004737 040100 JSR PC,$ATY3 ;SPOOL MESSAGE TO APT
    
```

```

4991 037660 000000 61$: .WORD 0 ;MESSAGE ADDRESS
4992 037662 132737 000040 001247 62$: BITB #APTCUP,$ENVM ;APT CONSOLE SUPPRESSED
4993 037670 001003 BNE 60$ ;YES,SKIP TYPE OUT
4994 037672 112046 2$: MOV (RO)+,-(SP) ;PUSH CHARACTER TO BE TYPED ONTO STACK
4995 037674 001005 BNE 4$ ;BR IF IT ISN'T THE TERMINATOR
4996 037676 005726 TST (SP)+ ;IF TERMINATOR POP IT OFF THE STACK
4997 037700 012600 60$: MOV (SP)+,RO ;RESTORE RO
4998 037702 062716 000002 3$: ADD #2,(SP) ;ADJUST RETURN PC
4999 037706 000002 RTI ;RETURN
5000 037710 122716 000011 4$: CMPB #HT,(SP) ;BRANCH IF <HT>
5001 037714 001430 BEQ 8$
5002 037716 122716 000200 CMPB #CRLF,(SP) ;BRANCH IF NOT <CRLF>
5003 037722 001006 BNE 5$
5004 037724 005726 TST (SP)+ ;POP <CR><LF> EQUIV
5005 037726 104401 TYPE ;TYPE A CR AND LF
5006 037730 001223 $CRLF
5007 037732 105037 040066 CLR B $CHARCNT ;CLEAR CHARACTER COUNT
5008 037736 000755 BR 2$ ;GET NEXT CHARACTER
5009 037740 004737 040022 5$: JSR PC,$TYPEC ;GO TYPE THIS CHARACTER
5010 037744 132726 001156 6$: CMPB $FILLC,(SP)+ ;IS IT TIME FOR FILLER CHARS.?
5011 037750 001350 BNE 2$ ;IF NO GO GET NEXT CHAR.
5012 037752 013746 001154 MOV $NULL,-(SP) ;GET # OF FILLER CHARS. NEEDED
5013 ;AND THE NULL CHAR.
5014 037756 105366 000001 7$: DECB 1(SP) ;DOES A NULL NEED TO BE TYPED?
5015 037762 002770 BLT 6$ ;BR IF NO--GO POP THE NULL OFF OF STACK
5016 037764 004737 040022 JSR PC,$TYPEC ;GO TYPE A NULL
5017 037770 105337 040066 DECB $CHARCNT ;DD NOT COUNT AS A COUNT
5018 037774 000770 BR 7$ ;LOOP
5019
5020 ;HORIZONTAL TAB PROCESSOR
5021
5022 037776 112716 000040 8$: MOV B #1,(SP) ;REPLACE TAB WITH SPACE
5023 040002 004737 040022 9$: JSR PC,$TYPEC ;TYPE A SPACE
5024 040006 132737 000007 040066 BITB #7,$CHARCNT ;BRANCH IF NOT AT
5025 040014 001372 BNE 9$ ;TAB STOP
5026 040016 005726 TST (SP)+ ;POP SPACE OFF STACK
5027 040020 000724 BR 2$ ;GET NEXT CHARACTER
5028 040022 105777 141122 $TYPEC: TSTB @STPS ;WAIT UNTIL PRINTER IS READY
5029 040026 100375 BPL $TYPEC
5030 040030 116677 000002 141114 MOV B 2(SP),@STPB ;LOAD CHAR TO BE TYPED INTO DATA REG.
5031 040036 122766 000015 000002 CMPB #CR,2(SP) ;IS CHARACTER A CARRIAGE RETURN?
5032 040044 001003 BNE 1$ ;BRANCH IF NO
5033 040046 105037 040066 CLR B $CHARCNT ;YES--CLEAR CHARACTER COUNT
5034 040052 000406 BR $TYPEC ;EXIT
5035 040054 122766 000012 000002 1$: CMPB #LF,2(SP) ;IS CHARACTER A LINE FEED?
5036 040062 001402 BEQ $TYPEC ;BRANCH IF YES
5037 040064 105227 INCB (PC)+ ;COUNT THE CHARACTER
5038 040066 000000 $CHARCNT: .WORD 0 ;CHARACTER COUNT STORAGE
5039 040070 000207 $TYPEC: RTS PC
5040
5041 .SBTTL APT COMMUNICATIONS ROUTINE
5042
5043 *****
5044 040072 112737 000001 040336 $ATY1: MOV B #1,$FLG ;TO REPORT FATAL ERROR
5045 040100 112737 000001 040334 $ATY3: MOV B #1,$MFLG ;TO TYPE A MESSAGE
5046 040106 000403 BR $ATYC
    
```

```

5047 040110 112737 000001 040336 $ATY4: MOV  #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
5048 040116 $ATYC:
5049 040116 010046 MOV  R0,-(SP) ;;PUSH R0 ON STACK
5050 040120 010146 MOV  R1,-(SP) ;;PUSH R1 ON STACK
5051 040122 105737 040334 TSTB $MFLG ;;SHOULD TYPE A MESSAGE?
5052 040126 001450 BEQ  5$ ;;IF NOT: BR
5053 040130 122737 000001 001246 CMPB  #APTENV,$ENV ;;OPERATING UNDER APT?
5054 040136 001031 BNE  3$ ;;IF NOT: BR
5055 040140 132737 000100 001247 BITB  #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
5056 040146 001425 BEQ  3$ ;;IF NOT: BR
5057 040150 017600 000004 MOV  @4(SP),R0 ;;GET MESSAGE ADDR.
5058 040154 062766 000002 000004 ADD  #2,4(SP) ;;BUMP RETURN ADDR.
5059 040162 005737 001226 1$: TST  $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
5060 040166 001375 BNE  1$ ;;IF NOT: WAIT
5061 040170 010037 001242 MOV  R0,$MSGAD ;;PUT ADDR IN MAILBOX
5062 040174 105720 2$: TSTB (R0)+ ;;FIND END OF MESSAGE
5063 040176 001376 BNE  2$
5064 040200 163700 001242 SUB  $MSGAD,R0 ;;SUB START OF MESSAGE
5065 040204 006200 ASR  R0 ;;GET MESSAGE LNTH IN WORDS
5066 040206 010037 001244 MOV  R0,$MSGGLT ;;PUT LENGTH IN MAILBOX
5067 040212 012737 000004 001226 MOV  #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
5068 040220 000413 BR   5$
5069 040222 017637 000004 040246 3$: MOV  @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE
5070 040230 062766 000002 000004 ADD  #2,4(SP) ;;BUMP RETURN ADDRESS
5071 040236 013746 177776 MOV  177776,-(SP) ;;PUSH 177776 ON STACK
5072 040242 004737 037610 JSR  PC,$TYPE ;;CALL TYPE MACRO
5073 040246 000000 4$: .WORD 0
5074 040250 5$:
5075 040250 105737 040336 10$: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?
5076 040254 001416 BEQ  12$ ;;IF NOT: BR
5077 040256 005737 001246 TST  $ENV ;;RUNNING UNDER APT?
5078 040262 001413 BEQ  12$ ;;IF NOT: BR
5079 040264 005737 001226 11$: TST  $MSGTYPE ;;FINISHED LAST MESSAGE?
5080 040270 00175 BNE  11$ ;;IF NOT: WAIT
5081 040272 017637 000004 001230 MOV  @4(SP),$FATAL ;;GET ERROR #
5082 040300 062766 000002 000004 ADD  #2,4(SP) ;;BUMP RETURN ADDR.
5083 040306 005237 001226 INC  $MSGTYPE ;;TELL APT TO TAKE ERROR
5084 040312 105037 040336 12$: CLRB $FFLG ;;CLEAR FATAL FLAG
5085 040316 105037 040335 CLRB $LFLG ;;CLEAR LOG FLAG
5086 040322 105037 040334 CLRB $MFLG ;;CLEAR MESSAGE FLAG
5087 040326 012601 MOV  (SP)+,R1 ;;POP STACK INTO R1
5088 040330 012600 MOV  (SP)+,R0 ;;POP STACK INTO R0
5089 040332 000207 RTS  PC ;;RETURN
5090 040334 000 $MFLG: .BYTE 0 ;;MESSG. FLAG
5091 040335 000 $LFLG: .BYTE 0 ;;LOG FLAG
5092 040336 000 $FFLG: .BYTE 0 ;;FATAL FLAG
5093 040340 .EVEN
5094 000200 APTSIZE=200
5095 000001 APTENV=001
5096 000100 AP.SPOOL=100
5097 000040 APTCSUP=040
5098 .SBTTL BINARY TO ASCII AND TYPE ROUTINE
5099
5100 *****
5101 *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
5102 *BINARY-ASCII NUMBER AND TYPE IT.
    
```

```

5103 *CALL:
5104 * MOV  NUMBER,-(SP) ;;NUMBER TO BE TYPED
5105 * TYPBN ;;TYPE IT
5106
5107 040340 010146 $TYPBN: MOV  R1,-(SP) ;;SAVE R1 ON THE STACK
5108 040342 016601 000006 MOV  6(SP),R1 ;;GET THE INPUT NUMBER
5109 040346 000261 SEC ;;SET "C" SO CAN KEEP TRACK OF THE NUMBER OF BITS
5110 040350 112737 000060 040412 1$: MOV  #0,$BIN ;;SET CHARACTER TO AN ASCII "0".
5111 040356 006101 ROL  R1 ;;GET THIS BIT
5112 040360 001406 BEQ  2$ ;;DONE?
5113 040362 105537 040412 ADCB  $BIN ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
5114 040366 104401 040412 TYPE  $BIN ;;GO TYPE THIS BIT
5115 040372 000241 CLC ;;CLEAR "C" SO CAN KEEP TRACK OF BITS
5116 040374 000755 BR   1$ ;;GO DO THE NEXT BIT
5117 040376 012601 2$: MOV  (SP)+,R1 ;;POP THE STACK INTO R1
5118 040400 016666 000002 000004 MOV  2(SP),4(SP) ;;ADJUST THE STACK
5119 040406 012616 MOV  (SP)+,(SP)
5120 040410 000002 RTI ;;RETURN TO USER
5121 040412 000 000 $BIN: .BYTE 0,0 ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
5122 .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
5123
5124 *****
5125 *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
5126 *OCTAL (ASCII) NUMBER AND TYPE IT.
5127 *$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
5128 *CALL:
5129 * MOV  NUM,-(SP) ;;NUMBER TO BE TYPED
5130 * TYPOS ;;CALL FOR TYPEOUT
5131 * .BYTE  N ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
5132 * .BYTE  M ;;M=1 OR 0
5133 * ;;1=TYPE LEADING ZEROS
5134 * ;;0=SUPPRESS LEADING ZEROS
5135 *
5136 *$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
5137 *$TYPOS OR $TYPOC
5138 *CALL:
5139 * MOV  NUM,-(SP) ;;NUMBER TO BE TYPED
5140 * TYPON ;;CALL FOR TYPEOUT
5141 *
5142 *$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
5143 *CALL:
5144 * MOV  NUM,-(SP) ;;NUMBER TO BE TYPED
5145 * TYPOC ;;CALL FOR TYPEOUT
5146 *
5147 040414 017646 000000 $TYPOS: MOV  @($SP),-(SP) ;;PICKUP THE MODE
5148 040420 116637 000001 040637 MOV  1(SP),$OFILL ;;LOAD ZERO FILL SWITCH
5149 040426 112637 040641 MOV  (SP)+,$SOMODE+1 ;;NUMBER OF DIGITS TO TYPE
5150 040432 062716 000002 ADD  #2,(SP) ;;ADJUST RETURN ADDRESS
5151 040436 000406 BR   $TYPON
5152 040440 112737 000001 040637 $TYPOC: MOV  #1,$OFILL ;;SET THE ZERO FILL SWITCH
5153 040446 112737 000006 040641 MOV  #6,$SOMODE+1 ;;SET FOR SIX(6) DIGITS
5154 040454 112737 000005 040637 $TYPON: MOV  #5,$SOCNT ;;SET THE ITERATION COUNT
5155 040462 010346 MOV  R3,-(SP) ;;SAVE R3
5156 040464 010446 MOV  R4,-(SP) ;;SAVE R4
5157 040466 010546 MOV  R5,-(SP) ;;SAVE R5
5158 040470 113704 040641 MOV  $SOMODE+1,R4 ;;GET THE NUMBER OF DIGITS TO TYPE
    
```

```

5159 040474 005404          NEG      R4          ;;SUBTRACT IT FOR MAX. ALLOWED
5160 040476 062704 000006  ADD      #6,R4      ;;SAVE IT FOR USE
5161 040502 110437 040640  MOVB    R4,$OMODE   ;;GET THE ZERO FILL SWITCH
5162 040506 113704 040637  MOVB    30FILL,R4   ;;PICKUP THE INPUT NUMBER
5163 040512 016505 000012  MOV     12(SP),R5    ;;CLEAR THE OUTPUT WORD
5164 040516 005003          CLR      R3          ;;ROTATE MSB INTO "C"
5165 040520 006105          1$:    ROL     R5          ;;GO DO MSB
5166 040522 000404          BR      3$          ;;FORM THIS DIGIT
5167 040524 006105          2$:    ROL     R5
5168 040526 006105          ROL     R5
5169 040530 006105          ROL     R5
5170 040532 010503          MOV     R5,R3
5171 040534 006103          3$:    ROL     R3          ;;GET LSB OF THIS DIGIT
5172 040536 105337 040640  DECB    $OMODE      ;;TYPE THIS DIGIT?
5173 040542 100016          BPL     7$          ;;BR IF NO
5174 040544 042703 177770  BIC     #177770,R3  ;;GET RID OF JUNK
5175 040550 001002          BNE     4$          ;;TEST FOR 0
5176 040552 005704          TST    R4          ;;SUPPRESS THIS 0?
5177 040554 001403          BEQ     5$          ;;BR IF YES
5178 040556 005204          INC    R4          ;;DON'T SUPPRESS ANYMORE 0'S
5179 040560 052703 000060  BIS     #'0,R3      ;;MAKE THIS DIGIT ASCII
5180 040564 052703 000040  BIS     #'1,R3      ;;MAKE ASCII IF NOT ALREADY
5181 040570 110337 040634  MOVB    R3,$$       ;;SAVE FOR TYPING
5182 040574 104401 040634  TYPE    ,$$         ;;GO TYPE THIS DIGIT
5183 040600 105337 040636  7$:    DECB    $OCNT  ;;COUNT BY 1
5184 040604 003347          BGT     2$          ;;BR IF MORE TO DO
5185 040606 002402          BLT     6$          ;;BR IF DONE
5186 040610 005204          INC    R4          ;;INSURE LAST DIGIT ISN'T A BLANK
5187 040612 000744          BR      2$          ;;GO DO THE LAST DIGIT
5188 040614 012605          6$:    MOV     (SP)+,R5  ;;RESTORE R5
5189 040616 012604          MOV     (SP)+,R4  ;;RESTORE R4
5190 040620 012603          MOV     (SP)+,R3  ;;RESTORE R3
5191 040622 016666 000002 000004  MOV     2(SP),4(SP) ;;SET THE STACK FOR RETURNING
5192 040630 012616          MOV     (SP)+,(SP)
5193 040632 000002          RTI                    ;;RETURN
5194 040634 000          .BYTE  0             ;;STORAGE FOR ASCII DIGIT
5195 040635 000          .BYTE  0             ;;TERMINATOR FOR TYPE ROUTINE
5196 040636 000          SOCNT: .BYTE 0        ;;OCTAL DIGIT COUNTER
5197 040637 000          $OFILL: .BYTE 0     ;;ZERO FILL SWITCH
5198 040640 000000          $OMODE: .WORD 0     ;;NUMBER OF DIGITS TO TYPE
5199          .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
5200
5201          ;;*****
5202          ;;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
5203          ;;SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
5204          ;;NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
5205          ;;BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
5206          ;;REPLACED WITH SPACES.
5207          ;;CALL:
5208          ;;*   MOV     NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
5209          ;;*   TYPDS          ;;GO TO THE ROUTINE
5210
5211          STYPDS:
5212          MOV     R0,-(SP)      ;;PUSH R0 ON STACK
5213          MOV     R1,-(SP)      ;;PUSH R1 ON STACK
5214          MOV     R2,-(SP)      ;;PUSH R2 ON STACK
    
```

```

5215 040650 010346          MOV     R3,-(SP)     ;;PUSH R3 ON STACK
5216 040652 010546          MOV     R5,-(SP)     ;;PUSH R5 ON STACK
5217 040654 012746 020200  MOV     #20200,-(SP) ;;SET BLANK SWITCH AND SIGN
5218 040660 016605 000020  MOV     20(SP),R5    ;;GET THE INPUT NUMBER
5219 040664 100004          BPL     1$          ;;BR IF INPUT IS POS.
5220 040666 005405          NEG     R5          ;;MAKE THE BINARY NUMBER POS.
5221 040670 112766 000055 000001  MOVB    #'-,1(SP)    ;;MAKE THE ASCII NUMBER NEG.
5222 040676 005000          1$:    CLR     R0          ;;ZERO THE CONSTANTS INDEX
5223 040700 012703 041056  MOV     #SDBLK,R3    ;;SETUP THE OUTPUT POINTER
5224 040704 112723 000040  MOVB    #' ,(R3)+    ;;SET THE FIRST CHARACTER TO A BLANK
5225 040710 005002          2$:    CLR     R2          ;;CLEAR THE BCD NUMBER
5226 040712 016001 041048  MOV     $DTBL(R0),R1 ;;GET THE CONSTANT
5227 040716 160105          3$:    SUB     R1,R5     ;;FORM THIS BCD DIGIT
5228 040720 002402          BLT     4$          ;;BR IF DONE
5229 040722 005202          INC    R2          ;;INCREASE THE BCD DIGIT BY 1
5230 040724 000774          BR      3$
5231 040726 060105          4$:    ADD     R1,R5     ;;ADD BACK THE CONSTANT
5232 040730 005702          TST    R2          ;;CHECK IF BCD DIGIT=0
5233 040732 001002          BNE     5$          ;;FALL THROUGH IF 0
5234 040734 195716          TSTB   (SP)         ;;STILL DOING LEADING 0'S?
5235 040736 100407          BMI     7$          ;;BR IF YES
5236 040740 106316          5$:    ASLB   (SP)         ;;MSD?
5237 040742 103003          BCC     6$          ;;BR IF NO
5238 040744 116663 000001 177777  MOVB    1(SP),-1(R3) ;;YES--SET THE SIGN
5239 040752 052702 000060  6$:    BIS     #'0,R2     ;;MAKE THE BCD DIGIT ASCII
5240 040756 052702 000040  7$:    BIS     #'1,R2     ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
5241 040762 110223          MOVB    R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
5242 040764 005720          TST    (R0)+        ;;JUST INCREMENTING
5243 040766 020027 000010  CMP     R0,#10      ;;CHECK THE TABLE INDEX
5244 040772 002746          BLT     2$          ;;GO DO THE NEXT DIGIT
5245 040774 003002          BGT     8$          ;;GO TO EXIT
5246 040776 010502          MOV     R5,R2       ;;GET THE LSD
5247 041000 000754          BR      6$          ;;GO CHANGE TO ASCII
5248 041002 105.26          8$:    TSTB   (SP)+    ;;WAS THE LSD THE FIRST NON-ZERO?
5249 041004 100003          BPL     9$          ;;BR IF NO
5250 041006 116663 177777 177776  MOVB    -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
5251 041014 105013          CLRB   (R3)         ;;SET THE TERMINATOR
5252 041016 012605          MOV     (SP)+,R5    ;;POP STACK INTO R5
5253 041020 012603          MOV     (SP)+,R3    ;;POP STACK INTO R3
5254 041022 012602          MOV     (SP)+,R2    ;;POP STACK INTO R2
5255 041024 012601          MOV     (SP)+,R1    ;;POP STACK INTO R1
5256 041026 012600          MOV     (SP)+,R0    ;;POP STACK INTO R0
5257 041030 104401 041056  TYPE    ,SDBLK      ;;NOW TYPE THE NUMBER
5258 041034 016666 000002 000004  MOV     2(SP),4(SP) ;;ADJUST THE STACK
5259 041042 012616          MOV     (SP)+,(SP)
5260 041044 000002          RTI                    ;;RETURN TO USER
5261 041046 023420          $DTBL: 10000.
5262 041050 001750          1000.
5263 041052 000144          100.
5264 041054 000012          10.
5265 041056 000004          $DBLK: .BLKW 4
5266          .SBTTL SAVE AND RESTORE R0-R5 ROUTINES
5267
5268          ;;*****
5269          ;;SAVE R0-R5
5270          ;;CALL:
    
```

```

5271          ;* SAVREG
5272          ;*UPON RETURN FROM $$SAVREG THE STACK WILL LOOK LIKE:
5273          ;*
5274          ;*TOP---(+16)
5275          ;* +2---(+18)
5276          ;* +4---R5
5277          ;* +6---R4
5278          ;* +8---R3
5279          ;*+10---R2
5280          ;*+12---R1
5281          ;*+14---R0
5282
5283          $$SAVREG:
5284          041066      MOV     R0,-(SP)      ;;PUSH R0 ON STACK
5285          041070      MOV     R1,-(SP)      ;;PUSH R1 ON STACK
5286          041072      MOV     R2,-(SP)      ;;PUSH R2 ON STACK
5287          041074      MOV     R3,-(SP)      ;;PUSH R3 ON STACK
5288          041076      MOV     R4,-(SP)      ;;PUSH R4 ON STACK
5289          041100      MOV     R5,-(SP)      ;;PUSH R5 ON STACK
5290          041102      MOV     22(SP),-(SP)  ;;SAVE PS OF MAIN FLOW
5291          041106      MOV     22(SP),-(SP)  ;;SAVE PC OF MAIN FLOW
5292          041112      MOV     22(SP),-(SP)  ;;SAVE PS OF CALL
5293          041116      MOV     22(SP),-(SP)  ;;SAVE PC OF CALL
5294          041122      000022      RTI
5295
5296          ;*RESTORE R0-R5
5297          ;*CALL:
5298          ;* RESREG
5299          $$RESREG:
5300          041124      MOV     (SP)+,22(SP)  ;;RESTORE PC OF CALL
5301          041130      MOV     (SP)+,22(SP)  ;;RESTORE PS OF CALL
5302          041134      MOV     (SP)+,22(SP)  ;;RESTORE PC OF MAIN FLOW
5303          041140      MOV     (SP)+,22(SP)  ;;RESTORE PS OF MAIN FLOW
5304          041144      MOV     (SP)+,R5      ;;POP STACK INTO R5
5305          041146      MOV     (SP)+,R4      ;;POP STACK INTO R4
5306          041150      MOV     (SP)+,R3      ;;POP STACK INTO R3
5307          041152      MOV     (SP)+,R2      ;;POP STACK INTO R2
5308          041154      MOV     (SP)+,R1      ;;POP STACK INTO R1
5309          041156      MOV     (SP)+,R0      ;;POP STACK INTO R0
5310          041160      000022      RTI
5311          .SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
5312
5313          ;*****
5314          ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
5315          ;*UNSIGNED OCTAL ASCII NUMBER.
5316          ;*CALL
5317          ;* MOV     #PNTR,-(SP)      ;;POINTER TO LOW WORD OF BINARY NUMBER
5318          ;* JSR     PC,#$SDB20      ;;CALL THE ROUTINE
5319          ;* RETURN      ;;THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK
5320
5321          $SDB20: SAVREG      ;;SAVE ALL REGISTERS
5322          041162      MOV     2(SP),R1      ;;PICKUP THE POINTER TO LOW WORD
5323          041164      MOV     #$OCTVL+13.,R5 ;;POINTER TO DATA TABLE
5324          041170      MOV     #12.,R4      ;;DO ELEVEN CHARACTERS
5325          041174      MOV     #*C7,R3      ;;MASK
5326          041200      012703      177770
    
```

```

5327          MOV     (R1)+,R0      ;;LOWER WORD
5328          041206      MOV     (R1)+,R1      ;;HIGH WORD
5329          041210      CLR     R2          ;;TERMINATOR
5330          041212      110245      1$: MOVB  W2,-(R5)      ;;PUT CHARACTER IN DATA TABLE
5331          041214      010002      MOV     R0,R2      ;;GET THIS DIGIT
5332          041216      005304      DEC     R4          ;;COUNT THIS CHARACTER
5333          041220      003007      BGT     3$        ;;BR IF NOT THE LAST DIGIT
5334          041222      001405      BEQ     2$        ;;BR IF IT IS THE LAST DIGIT
5335          041224      005205      INC     R5          ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
5336          041226      010566      000002      MOV     R5,2(SP)  ;;ASCIZ CHAR. & PUT IT ON THE STACK
5337          041232      104414      RESREG      ;;RESTORE ALL REGISTERS
5338          041234      000207      RTS     PC        ;;RETURN TO USER
5339          041236      006203      2$: ASR     R3      ;;POSITION THE MASK FOR THE LAST DIGIT
5340          041240      006001      3$: ROR     R1      ;;POSITION THE BINARY NUMBER FOR
5341          041242      006000      ROR     R0        ;; THE NEXT OCTAL DIGIT
5342          041244      006001      ROR     R1
5343          041246      006000      ROR     R0
5344          041250      006001      ROR     R1
5345          041252      006000      ROR     R0
5346          041254      040302      BIC     R3,R2      ;;MASK OUT ALL JUNK
5347          041256      062702      000060      ADD     #'0,R2     ;;MAKE THIS CHAR. ASCII
5348          041262      000753      BR     1$        ;;GO PUT IT IN THE DATA TABLE
5349          041264      000016      $OCTVL: .BLKB 14. ;;RESERVE DATA TABLE
    
```

```

5350 .SBTTL TRAP DECODER
5351
5352
5353 ;*****
5354 ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
5355 ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
5356 ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
5357 ;*GO TO THAT ROUTINE.
5358 041302 010046 $TRAP: MOV RO,-(SP) ;;SAVE RO
5359 041304 016600 000002 MOV 2(SP),RO ;;GET TRAP ADDRESS
5360 041310 005740 TST -(RO) ;;BACKUP BY 2
5361 041312 111000 MOVB (RO),RO ;;GET RIGHT BYTE OF TRAP
5362 041314 006300 ASL RO ;;POSITION FOR INDEXING
5363 041316 016000 041336 MOV $TRPAD(RO),RO ;;INDEX TO TABLE
5364 041322 000200 RTS RO ;;GO TO ROUTINE
5365
5366
5367 ;THIS IS USED TO HANDLE THE "GETPRI" MACRO
5368
5369 041324 011646 $TRAP2: MOV (SP),-(SP) ;;MOVE THE PC DOWN
5370 041326 016666 000004 000002 MOV 4(SP),2(SP) ;;MOVE THE PSW DOWN
5371 041334 000002 RTI ;;RESTORE THE PSW
5372
5373 .SBTTL TRAP TABLE
5374
5375 ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
5376 ;*BY THE "TRAP" INSTRUCTION.
5377
5378 ; ROUTINE
5379 ;-----
5380 041336 041324 $TRPAD: .WORD $TRAP2 TRAP+1(104401) TTY TYPEOUT ROUTINE
5381 041340 037610 $TYPE ;;CALL=TYPE TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
5382 041342 040440 $TYPOC ;;CALL=TYPOC TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
5383 041344 040414 $TYPOS ;;CALL=TYPOS TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
5384 041346 040454 $TYPON ;;CALL=TYPON TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
5385 041350 040642 $TYPDS ;;CALL=TYPDS TRAP+6(104406) TYPE BINARY (ASCII) NUMBER
5386 041352 040340 $TYPBN ;;CALL=TYPBN TRAP+7(104407) GET SOFT-SWR SETTING
5387
5388 041354 036536 $GTSWR ;;CALL=GTSWR TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
5389
5390 041356 036466 $CKSWR ;;CALL=CKSWR TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
5391 041360 037010 $RDCHR ;;CALL=RDCHR TRAP+12(104412) TTY TYPEIN STRING ROUTINE
5392 041362 037130 $RDLIN ;;CALL=RDLIN TRAP+13(104413) SAVE R0-R5 ROUTINE
5393 041364 041066 $SAVREG ;;CALL=SAVREG TRAP+14(104414) RESTORE R0-R5 ROUTINE
5394 041366 041124 $RESREG ;;CALL=RESREG
5395
5396 .SBTTL POWER DOWN AND UP ROUTINES
5397
5398 ;*****
5399 .POWER DOWN ROUTINE
5400 041370 012737 041546 000024 $PWRDN: MOV #ILLUP,@#PWRVEC ;;SET FOR FAST UP
5401 041376 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;PRIO:7
5402 041404 010046 MOV RO,-(SP) ;;PUSH R0 ON STACK
5403 041406 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
5404 041410 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
5405 041412 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
5406 041414 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK
    
```

```

5406 041416 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
5407 041420 017746 137514 MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
5408 041424 010637 041552 MOV SP,$SAVR6 ;;SAVE SP
5409 041430 012737 041442 000024 MOV #PWRUP,@#PWRVEC ;;SET UP VECTOR
5410 041436 000000 HALT
5411 041440 000776 BR -2 ;;HANG UP
5412
5413 ;*****
5414 .POWER UP ROUTINE
5415 041442 012737 041546 000024 $PWRUP: MOV #ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
5416 041450 013706 041552 MOV $SAVR6,SP ;;GET SP
5417 041454 005037 041552 CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
5418 041460 005237 041552 1$: INC $SAVR6 ;;WAIT FOR THE INC
5419 041464 001775 BNE 1$ ;;OF WORD
5420 041466 012677 137446 MOV (SP)+,@SWR ;;POP STACK INTO @SWR
5421 041472 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
5422 041474 012604 MOV (SP)+,R4 ;;POP STACK INTO R4
5423 041476 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
5424 041500 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
5425 041502 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
5426 041504 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
5427 041506 012737 041370 000024 MOV #PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
5428 041514 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;PRIO:7
5429 041522 104401 TYPE ;;REPORT THE POWER FAILURE
5430 041524 041554 $PWRMG: .WORD PWRMSG ;;POWER FAIL MESSAGE POINTER
5431 041526 012716 MOV (PC)+,(SP) ;;RESTART AT START
5432 041530 020000 $PWRAD: .WORD START ;;RESTART ADDRESS
5433 041532 042766 000020 000002 BIC #20,2(SP) ;;CLEAR "T" BIT
5434 041540 005037 035032 CLR $TBIT ;;CLEAR THE "T" BIT FLAG
5435
5436 041544 000002 RTI
5437 041546 000000 $ILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
5438 041550 000776 BR -2 ;;BEFORE THE POWER DOWN WAS COMPLETE
5439 041552 000000 $SAVR6: 0 ;;PUT THE SP HERE
5440 041554 006412 050040 053517 PWRMSG: .ASCIZ <12><15>? POWER FAILURE - RESTARTING ?<12><15>
5441 041562 051105 043040 044501
5442 041570 052514 042522 026440
5443 041576 051040 051505 040524
5444 041604 052122 047111 020107
5445 041612 006412 000
5446 041616 .EVEN
5447
    
```

```
.SBTTL ERROR MESSAGES, DATA HEADERS-TABLES & FORMATS
EM1: .ASCIZ /UNEXPECTED CPU TRAP TO LOC. 004/
5448
5449 041616 047125 054105 042520
5450 041624 052103 042105 041440
5451 041632 052520 052040 040522
5452 041640 020120 047524 046040
5453 041646 041517 020056 030060
5454 041654 000064
5455 041656 047125 054105 042520 EM2: .ASCIZ /UNEXPECTED MEM. MGMT. TRAP TO LOC. 250/
5456 041664 052103 042105 046440
5457 041672 046505 020056 043515
5458 041700 052115 020056 051124
5459 041706 050101 052040 020117
5460 041714 047514 027103 031040
5461 041722 030065 000
5462 041725 120 044522 051117 EM3: .ASCIZ /PRIORITY BITS SET WRONG IN PSW/
5463 041732 052111 020131 044502
5464 041740 051524 051440 052105
5465 041746 053440 047522 043516
5466 041754 044440 020116 051520
5467 041762 000127
5468 041764 047515 042504 041040 EM4: .ASCIZ /MODE BITS SET WRONG IN PSW/
5469 041772 052111 020123 042523
5470 042000 020124 051127 047117
5471 042006 020107 047111 050040
5472 042014 053523 000
5473 042017 104 040525 020114 EM5: .ASCIZ /DUAL ADDRESSING BETWEEN HI&LO BYTES OF PSW/
5474 042024 042101 051104 051505
5475 042032 044523 043516 041040
5476 042040 052105 042527 047105
5477 042046 044040 023111 047514
5478 042054 041040 052131 051505
5479 042062 047440 020106 051520
5480 042070 000127
5481 042072 042513 047122 046105 EM6: .ASCIZ /KERNEL R0 CHANGED BY WRITING USER R0/
5482 042100 051040 020066 044103
5483 042106 047101 042507 020104
5484 042114 054502 053440 044522
5485 042122 044524 043516 052440
5486 042130 042523 020122 033122
5487 042136 000
5488 042137 101 046440 046505 EM7: .ASCIZ /A MEMORY MGMT. REG. TIMED OUT/
5489 042144 051117 020131 043515
5490 042152 052115 020056 042522
5491 042160 027107 052040 046511
5492 042166 042105 047440 052125
5493 042174 000
5494 042175 123 046525 040515 EM10: .ASCIZ /SUMMARY OF MEM. MGMT. REG. TIMEOUTS/
5495 042202 054522 047440 020106
5496 042210 042515 027115 046440
5497 042216 046507 027124 051040
5498 042224 043505 020056 044524
5499 042232 042515 052517 051524
5500 042240 000
5501 042241 115 046505 020056 EM11: .ASCIZ /MEM. MGMT. REG. WOULD NOT CLEAR/
5502 042246 043515 052115 020056
5503 042254 042522 027107 053440
```

```
5504 042262 052317 042114 047040
5505 042270 052117 041440 042514
5506 042276 051101 000
5507 042301 115 046505 020056 EM12: .ASCIZ /MEM. MGMT. REG. BITS NOT SET CORRECTLY/
5508 042306 043515 052115 020056
5509 042314 042522 027107 041040
5510 042322 052111 020123 047516
5511 042330 020124 042523 020124
5512 042336 047503 051122 041505
5513 042344 046124 000131
5514 042350 051123 020060 043105 EM13: .ASCIZ /SRO EFFECTED BY WRITE TO PSW/
5515 042356 042506 052103 042105
5516 042364 041040 020131 051127
5517 042372 052111 020105 047524
5518 042400 050040 053523 000
5519 042405 123 030522 042040 EM14: .ASCIZ /SR1 DID NOT READ ALL ZEROS/
5520 042412 042111 047040 052117
5521 042420 051040 040505 020104
5522 042426 046101 020114 042532
5523 042434 047522 000123
5524 042440 052504 046101 040440 EM15: .ASCIZ /DUAL ADDRESSING BETWEEN BYTES OF PAR OR PDR/
5525 042446 042104 042522 051523
5526 042454 047111 020107 042502
5527 042462 053524 042505 020116
5528 042470 054502 042524 020123
5529 042476 043117 050040 051101
5530 042504 047440 020122 042120
5531 042512 000122
5532 042514 052504 046101 040440 EM16: .ASCIZ /DUAL ADDRESSING BETWEEN PAR-PDR'S/
5533 042522 042104 042522 051523
5534 042530 047111 020107 042502
5535 042536 053524 042505 020116
5536 042544 040520 026522 042120
5537 042552 023222 000123
5538 042556 044120 051531 020056 EM17: .ASCIZ /PHYS. ADDR. FORMED WRONG IN MAINT. MODE/
5539 042564 042101 051104 020056
5540 042572 047506 048522 042105
5541 042600 053440 047522 043516
5542 042606 044440 020116 040515
5543 042614 047111 027124 046440
5544 042622 042117 000105
5545 042626 044120 051531 020056 EM20: .ASCIZ /PHYS. ADDR. FORMED WRONG IN RELOCATE MODE/
5546 042634 042101 051104 020056
5547 042642 047506 046522 042105
5548 042650 053440 047522 043516
5549 042656 044440 020116 042522
5550 042664 047514 040503 042524
5551 042672 046440 042117 000105
5552 042700 026527 044502 020124 EM21: .ASCIZ /W-BIT DID NOT GET SET IN PDR/
5553 042706 044504 020104 047516
5554 042714 020124 042507 020124
5555 042722 042523 020124 047111
5556 042730 050040 051104 000
5557 042735 127 041055 052111 EM22: .ASCIZ /W-BIT SET IN MORE THAN ONE PDR/
5558 042742 051440 052105 044440
5559 042750 020116 047515 042522
```

5560	042756	052040	040510	020116	
5561	042764	047117	020105	042120	
5562	042772	000122			
5563	042774	026527	044502	020124	EM23: .ASCIZ /W-BIT NOT CLEARED BY WRITING TO PDR/
5564	043002	047516	020124	046103	
5565	043010	040505	042522	020104	
5566	043016	054502	053440	044522	
5567	043024	044524	043516	052040	
5568	043032	020117	042120	000122	
5569	043040	051127	052111	047111	EM24: .ASCIZ /WRITING SRO SET W-BIT IN KIPDR7/
5570	043046	020107	051123	020060	
5571	043054	042523	020124	026527	
5572	043062	044502	020124	047111	
5573	043070	045140	050111	051104	
5574	043076	000067			
5575	043100	026527	044502	020124	EM25: .ASCIZ /W-BIT GOT SET DURING ODD ADDR. ABORT/
5576	043106	047507	020124	042523	
5577	043114	020124	052504	044522	
5578	043122	043516	047440	042104	
5579	043130	040440	042104	027122	
5580	043136	040440	047502	052122	
5581	043144	000			
5582	043145	115	046505	051117	EM26: .ASCIZ /MEMORY MGMT. ACCESS ABORT DID NOT OCCUR/
5583	043152	020131	043515	052115	
5584	043160	020056	041501	042503	
5585	043166	051523	040440	047502	
5586	043174	052122	042040	042111	
5587	043202	047040	052117	047440	
5588	043210	041503	051125	000	
5589	043215	101	041503	051505	EM27: .ASCIZ /ACCESS ERROR DID NOT ABORT INSTRUCTION/
5590	043222	020123	051105	047522	
5591	043230	020122	044504	020104	
5592	043236	047516	020124	041101	
5593	043244	051117	020124	047111	
5594	043252	052123	052522	052103	
5595	043260	047511	000116		
5596	043264	051123	020060	044504	EM30: .ASCIZ /SRO DID NOT REPORT ACCESS ERROR CORRECTLY/
5597	043272	020104	047516	020124	
5598	043300	042522	047520	052122	
5599	043306	040440	041503	051505	
5600	043314	020123	051105	047522	
5601	043322	020122	047503	051122	
5602	043330	041505	046124	000131	
5603	043336	044504	020104	047516	EM31: .ASCIZ /DID NOT LOCKUP CORRECT VIRTUAL ADDR./
5604	043344	020124	047514	045503	
5605	043352	050125	041440	051117	
5606	043360	042522	052103	053040	
5607	043366	051111	052524	046101	
5608	043374	040440	042104	027122	
5609	043402	000			
5610	043403	120	043501	020105	EM32: .ASCIZ /PAGE LGTH. ABORT OCCURRED WHEN IT SHOULDN'T HAVE/
5611	043410	043514	044124	020057	
5612	043416	041101	051117	020124	
5613	043424	041517	052503	051122	
5614	043432	042105	053440	042510	
5615	043440	020116	052111	051440	

5616	043446	047510	046125	047104	
5617	043454	052047	040400	053101	
5618	043462	000105			
5619	043462	040520	042507	046040	EM33: .ASCIZ /PAGE LGTH. ABORT DID NOT OCCUR WHEN IT SHOULD HAVE/
5620	043472	052107	027110	040440	
5621	043500	047502	052122	042040	
5622	043506	042111	047040	052117	
5623	043514	047440	041503	051125	
5624	043522	053440	042510	020116	
5625	043530	052111	051440	047510	
5626	043536	046125	020104	040510	
5627	043544	042526	000		
5628	043547	123	030122	042040	EM34: .ASCIZ /SRO DID NOT REPORT PAGE LGTH. ABORT CORRECTLY/
5629	043554	042111	047040	052117	
5630	043562	051040	050105	051117	
5631	043570	020124	040520	042507	
5632	043576	046040	052107	027110	
5633	043604	040440	047502	052122	
5634	043612	041440	051117	042522	
5635	043620	052103	054514	000	
5636	043625	123	030122	047440	EM37: .ASCIZ /SRO OR SR2 CHANGED BY A SECOND ABORT/
5637	043632	020122	051123	020062	
5638	043640	044103	047101	042507	
5639	043646	020104	054502	040440	
5640	043654	051440	041505	047117	
5641	043662	020104	041101	051117	
5642	043670	000124			
5643	043672	051123	020060	051117	EM40: .ASCIZ /SRO OR SR2 WERE NOT "RESET" BY A RESET/
5644	043700	051440	031122	053440	
5645	043706	051105	020105	047516	
5646	043714	020124	051042	051505	
5647	043722	052105	020042	054501	
5648	043730	040440	051040	051505	
5649	043736	052105	000		
5650	043741	123	031122	047040	EM41: .ASCIZ /SR2 NOT TRACKING CORRECTLY/
5651	043746	052117	052040	040522	
5652	043754	045503	047111	020107	
5653	043762	047503	051122	041505	
5654	043770	046124	000131		
5655	043774	044504	020104	047516	EM42: .ASCIZ /DID NOT TRAP THRU KERNEL SPACE/
5656	044002	020124	051124	050101	
5657	044010	052040	051110	020125	
5658	044016	042513	047122	046105	
5659	044024	051440	040520	042503	
5660	044032	000			
5661	044033	113	020124	051105	EM43: .ASCIZ /KT ERROR SERVICED ON ODD ADDR. ERROR/
5662	044040	047522	020122	042523	
5663	044046	053122	041511	042105	
5664	044054	047440	020116	042117	
5665	044062	020104	042101	051104	
5666	044070	020056	051105	047522	
5667	044076	000122			
5668	044100	051123	020060	051117	EM44: .ASCIZ /SRO OR SR2 CHANGED BY ODD ADDR. ERROR/
5669	044106	051440	031122	041440	
5670	044114	040510	043516	042105	
5671	044122	041040	020131	042117	

5672	044130	020104	042101	051104		
5673	044136	020956	051105	047522		
5674	044144	000122				
5675	044146	051105	047522	020122	EM45:	.ASCIZ /ERROR DURING "DOUBLE ERROR" (KT & ODD ADDR.)/
5676	044154	052504	044522	043516		
5677	044162	021040	047504	041125		
5678	044170	042514	042440	051122		
5679	044176	051117	020042	045450		
5680	044204	020124	020046	042117		
5681	044212	020104	042101	051104		
5682	044220	024456	000			
5683	044223	115	050106	020111	EM46:	.ASCIZ /MFPI INSTRUCTION PUSHED WRONG DATA/
5684	044230	047111	052123	052522		
5685	044236	052103	047511	020116		
5686	044244	052520	044123	042105		
5687	044252	053440	047522	043516		
5688	044260	042040	052101	000101		
5689	044266	052115	044520	044440	EM47:	.ASCIZ /MTPI INSTRUCTION LOADED WRONG DATA/
5690	044274	051516	051124	041525		
5691	044302	044524	047117	046040		
5692	044310	040517	042504	020104		
5693	044316	051127	047117	020107		
5694	044324	040504	040524	000		
5695	044331	123	040524	045503	EM50:	.ASCIZ /STACK NOT PUSHED BY MFPI-MTPI/
5696	044336	047040	052117	050040		
5697	044344	051525	042510	020104		
5698	044352	054502	046440	050106		
5699	044360	026511	052115	044520		
5700	044366	000				
5701	044367	113	051105	042516	EM51:	.ASCIZ /KERNEL PAGE ACCESS INSTEAD OF USER: MFPI-MTPI/
5702	044374	020114	040520	042507		
5703	044402	040440	041503	051505		
5704	044410	020123	047111	052123		
5705	044416	040505	020104	043117		
5706	044424	052440	042523	035122		
5707	044432	046440	050106	026511		
5708	044440	052115	044520	000		
5709	044445	127	047522	043516	EM52:	.ASCIZ /WRONG PDR'S REFERENCED WHILE IN RELOCATE MODE/
5710	044452	050040	051104	051447		
5711	044460	051040	043105	051105		
5712	044466	047105	042503	020104		
5713	044474	044127	046111	020105		
5714	044502	047111	051040	046105		
5715	044510	041517	052101	020105		
5716	044516	047515	042504	000		
5717	044523	115	050106	020104	EM53:	.ASCIZ /MFPD INSTRUCTION PUSHED WRONG DATA/
5718	044530	047111	052123	052522		
5719	044536	052103	047511	020116		
5720	044544	052520	044123	042105		
5721	044552	053440	047522	043516		
5722	044560	042040	052101	000101		
5723	044566	052123	041501	020113	EM54:	.ASCIZ /STACK NOT PUSHED BY MFPD-MTPD/
5724	044574	047516	020124	052520		
5725	044602	044123	042105	041040		
5726	044610	020131	043115	042120		
5727	044616	046455	050124	000104		

5728	044624	040520	020122	051117	EM55:	.ASCIZ /PAR OR PDR CHANGED BY A RESET/
5729	044632	050040	051104	041440		
5730	044640	040510	043516	042105		
5731	044646	041040	020131	020101		
5732	044654	042522	042523	000124		
5733	044662	046111	042514	040507	EM56:	.ASCIZ /ILLEGAL MODE 01 NOT ABORTED/
5734	044670	020114	047515	042504		
5735	044676	030040	020061	047516		
5736	044704	020124	041101	051117		
5737	044712	042524	000104			
5738	044716	051123	020060	044504	EM57:	.ASCIZ /SRO DID NOT REPORT ILLEGAL MODE 01 CORRECTLY/
5739	044724	020104	047516	020124		
5740	044732	042522	047520	052122		
5741	044740	044130	046114	043505		
5742	044746	046101	046440	042117		
5743	044754	020105	030460	041440		
5744	044762	051117	042522	052103		
5745	044770	054514	000			
5746	044773	120	053523	041440	EM60:	.ASCIZ /PSW CHANGED BY AN RTI IN USER MODE/
5747	045000	040510	043516	042105		
5748	045006	041040	020131	047101		
5749	045014	051040	044524	044440		
5750	045022	020116	051525	051105		
5751	045030	046440	042117	000105		
5752	045036	040515	047111	027124	EM61:	.ASCIZ /MAINT. MODE (SRO) NOT DISABLED BY A RESET/
5753	045044	046440	042117	020105		
5754	045052	051450	030122	036040		
5755	045060	037070	020051	047516		
5756	045066	020124	044504	040523		
5757	045074	046102	042105	041040		
5758	045102	020131	020101	042522		
5759	045110	042523	000124			
5760	045114	040504	040524	044440	EM62:	.ASCIZ /DATA INCORRECT AFTER A MAINT. MODE WRITE/
5761	045122	041516	051117	042522		
5762	045130	052103	040440	052106		
5763	045136	051105	040440	046440		
5764	045144	044501	052116	020056		
5765	045152	047515	042504	053440		
5766	045160	044522	042524	000		
5767	045165	123	052517	041522	EM63:	.ASCIZ /SOURCE RELOCATED IN MAINT. MODE/
5768	045172	020105	042522	047514		
5769	045200	040503	042524	020104		
5770	045206	047111	046440	044501		
5771	045214	052116	020056	047515		
5772	045222	042504	000			
5773	045225	116	047117	051040	EM64:	.ASCIZ /NON RESIDENT ABORT DID NOT OCCUR/
5774	045232	051505	042111	047105		
5775	045240	020124	041101	051117		
5776	045246	020124	044504	020104		
5777	045254	047516	020124	041517		
5778	045262	052503	000122			
5779	045266	051105	047522	020124	EM65:	.ASCIZ /ERROR FLAG FOR NR ABORT (BIT15) IN SRO DID NOT SET/
5780	045274	046106	043501	043040		
5781	045302	051117	047040	020122		
5782	045310	041101	051117	020124		
5783	045316	041050	052111	032461		

5784	045324	020051	047111	051440	
5785	045332	030122	042040	042111	
5786	045340	047040	052117	051440	
5787	045346	052105	000		
5788	045351	123	031122	042040	EM66: .ASCIZ /SR2 DID NOT FREEZE THE VIRTUAL ADDR OF THE ABORTD INTR/
5789	045356	042111	047040	052117	
5790	045364	043040	042522	055105	
5791	045372	020105	044124	020105	
5792	045400	044526	052122	040525	
5793	045406	020114	042101	051104	
5794	045414	020123	043117	052040	
5795	045422	042510	040400	047502	
5796	045430	052122	020104	047111	
5797	045436	051124	000		
5798	045441	062	042116	047040	EM67: .ASCIZ /2ND NON RESIDENT ABORT DID NOT OCCUR/
5799	045446	047117	051040	051505	
5800	045454	042111	047105	020124	
5801	045462	041101	051117	020124	
5802	045470	044504	020104	047516	
5803	045476	020124	041517	052503	
5804	045504	000122			
5805	045506	047062	020104	047516	EM70: .ASCIZ /2ND NON RESIDENT ABORT CAUSED SR2 TO CHANGE/
5806	045514	020116	042522	044523	
5807	045522	042504	052116	040440	
5808	045530	047502	052122	041440	
5809	045536	052501	042523	020104	
5810	045544	051123	020062	047524	
5811	045552	041440	040510	043516	
5812	045560	000105			
5813	045562	051123	020060	040527	EM71: .ASCIZ /SRO WAS NOT CLEARED BY INIT. /
5814	045570	020123	047516	020124	
5815	045576	046103	040505	042524	
5816	045604	020104	054502	044440	
5817	045612	044516	027124	000040	
5818					
5819	045620	046117	020104	041520	DH1: .ASCIZ /OLD PC OLD PSW R6 WAS TESTNO ERRORPC/
5820	045626	020040	046117	020104	
5821	045634	051520	020127	033122	
5822	045642	053440	051501	020040	
5823	045650	042524	052123	047516	
5824	045656	020040	051105	047522	
5825	045664	050122	000103		
5826	045670	046117	020104	041520	DH2: .ASCIZ /OLD PC OLD PSW R6 WAS SRO SR2 TESTNO ERRORPC/
5827	045676	020040	046117	020104	
5828	045704	051520	020127	033122	
5829	045712	053440	051501	020040	
5830	045720	051123	020060	020040	
5831	045726	020040	051123	020062	
5832	045734	020040	020040	042524	
5833	045742	052123	047516	020040	
5834	045750	051105	047522	050122	
5835	045756	000103			
5836	045760	051127	052117	0201C3	DH3: .ASCIZ /WROTE READ TESTNO ERRORPC/
5837	045766	020040	042522	042101	
5838	045774	020040	020040	042524	
5839	046002	052123	047516	020040	

5840	046010	051105	047522	050122	
5841	046016	000103			
5842	046020	042101	051104	051505	DH7: .ASCIZ /ADDRESS TESTNO ERRORPC/
5843	046026	020123	042524	052123	
5844	046034	047516	020040	051105	
5845	046042	047522	050122	000103	DH10: .ASCIZ /REGISTER-ADDRS NUM OF/<CRLF>
5846	046050	042522	044507	052123	
5847	046056	051105	040455	042104	
5848	046064	051522	020040	052516	
5849	046072	020115	047440	100106	
5850	046100	047101	026504	042105	.ASCIZ /AND-ED OR-ED TIMOUTS TESTNO ERRORPC/
5851	046106	020040	051117	04245E	
5852	046114	020104	020040	044524	
5853	046122	047515	052125	020123	
5854	046130	042524	052123	047516	
5855	046136	020040	051105	047522	
5856	046144	050122	000103		
5857	046150	042522	044507	052123	DH11: .ASCIZ /REGISTR READ READ-(BINARY)/<CRLF>
5858	046156	020122	042522	042101	
5859	046164	020040	020040	042522	
5860	046172	042101	024055	044502	
5861	046200	040516	054522	100051	
5862	046206	042101	051104	051505	.ASCIZ /ADDRESS (OCTAL) 5432109876543210 TESTNO ERRORPC/
5863	046214	020123	047450	052103	
5864	046222	046101	020051	032065	
5865	046230	031063	030061	034071	
5866	046236	033067	032065	031063	
5867	046244	030061	020040	042524	
5868	046252	052123	047516	020040	
5869	046260	051105	047522	050122	
5870	046266	000103			
5871	046270	042522	044507	052123	DH12: .ASCIZ /REGISTR WROTE READ READ-(BINARY)/<CRLF>
5872	046276	020122	051127	052117	
5873	046304	020105	020040	042522	
5874	046312	042101	020040	020040	
5875	046320	042522	042101	024055	
5876	046326	044502	040516	054522	
5877	046334	100051			
5878	046336	042101	051104	051505	.ASCIZ /ADDRESS (OCTAL) (OCTAL) 5432109876543210 TESTNO ERRORPC/
5879	046344	020123	047450	052103	
5880	046352	046101	020051	047450	
5881	046360	052103	046101	020051	
5882	046366	032065	031063	030061	
5883	046374	034071	033067	032065	
5884	046402	031063	030061	020040	
5885	046410	042524	052123	047516	
5886	046416	020040	051105	047522	
5887	046424	050122	000103		
5888	046430	042522	042101	020040	DH13: .ASCIZ /READ TESTNO ERRORPC/
5889	046436	020040	042524	052123	
5890	046444	047516	020040	051105	
5891	046452	047522	050122	000103	
5892	046460	040520	026522	042120	DH16: .ASCIZ /PAR-PDR PAR-PDR/<CRLF>
5893	046466	020122	040520	026522	
5894	046474	042120	100122		
5895	046500	046103	040505	042522	.ASCIZ /CLEARED EFFECTD EXPECTD RECEIVD TESTNO ERRORPC/

```

5896 046506 020104 043105 042506
5897 046514 052103 020104 054105
5898 046522 042520 052103 020104
5899 046530 042522 042503 053111
5900 046536 020104 042524 052123
5901 046544 047516 020040 051105
5902 046552 047522 050122 000103
5903 046560 044120 051531 041511 DH17: .ASCII /PHYSICL VIRTUAL/<CRLF>
5904 046566 020114 044526 052122
5905 046574 040525 100114
5906 046600 042101 051104 051505 .ASCIZ /ADDRESS ADDRESS KIPAR4 TESTNO ERRORPC/
5907 046606 020123 042101 051104
5908 046614 051505 020123 044513
5909 046622 040:20 032122 020040
5910 046630 042524 052123 047516
5911 046636 020040 051105 047522
5912 046644 050122 000103
5913 046650 044120 051531 041511 DH20: .ASCII /PHYSICL PAR 4 PAR 5/<CRLF>
5914 046656 020114 040520 020122
5915 046664 020064 020040 040520
5916 046672 020122 100065
5917 046676 042101 051104 051505 .ASCIZ /ADDRESS VBA VBA PAR 4 PAR 5 PSW TESTNO ERRORPC/
5918 046704 020123 041126 020101
5919 046712 020040 020040 041126
5920 046720 020101 020040 020040
5921 046726 040520 020122 020064
5922 046734 020040 040520 020122
5923 046742 020065 020040 051520
5924 046750 020127 020040 020040
5925 046756 042524 052123 047516
5926 046764 020040 051105 047522
5927 046772 050122 000103
5928 046776 042120 020122 020040 DH21: .ASCII /PDR VIRTUAL/<CRLF>
5929 047004 020040 044526 052122
5930 047012 040525 100114
5931 047016 042524 052123 042105 .ASCIZ /TESTED ADDRESS TESTNO ERRORPC/
5932 047024 020040 042101 051104
5933 047032 051505 020123 042524
5934 047040 052123 047516 020040
5935 047046 051105 047522 050122
5936 047054 000103
5937 047056 042120 020122 047111 DH22: .ASCII /PDR IN PDR VIRTUAL/
5938 047064 020040 042120 020122
5939 047072 020040 020040 044526
5940 047100 052122 040525 114
5941 047105 105 051122 051117 .ASCIZ /ERROR TESTED ADDRESS TESTNO ERRORPC/
5942 047112 020040 052040 051505
5943 047120 042524 020104 040440
5944 047126 042104 042522 051523
5945 047134 052040 051505 047124
5946 047142 020117 042440 051122
5947 047150 051117 041520 007
5948 047155 120 051104 020040 DH23: .ASCIZ /PDR TESTNO ERRORPC/
5949 047162 020040 052040 051505
5950 047170 047124 020117 042440
5951 047176 051122 051117 041520
    
```

```

5952 047204 000
5953 047205 120 051104 053440 DH24: .ASCIZ /PDR WAS EXPECTD TESTNO ERRORPC/
5954 047212 051501 042440 050130
5955 047220 041505 042124 052040
5956 047226 051505 047124 020117
5957 047234 042440 051122 051117
5958 047242 041520 000
5959 047245 120 051104 032040 DH26: .ASCIZ /PDR 4 PSW TESTNO ERRORPC/
5960 047252 020040 050040 053523
5961 047260 020040 020040 052040
5962 047266 051505 047124 020117
5963 047274 042440 051122 051117
5964 047302 041520 000
5965 047305 123 030122 053440 DH30: .ASCIZ /SRO WAS EXPECTD PDR 4 PSW TESTNO ERRORPC/
5966 047312 051501 042440 050130
5967 047320 041505 042124 050040
5968 047326 051104 032040 020040
5969 047334 050040 053523 020040
5970 047342 020040 052040 051505
5971 047350 047124 020117 042440
5972 047356 051122 051117 041520
5973 047364 000
5974 047365 123 031122 053440 DH31: .ASCIZ /SR2 WAS EXPECTD PDR 4 PSW TESTNO ERRORPC/
5975 047372 051501 042440 050130
5976 047400 041505 042124 050040
5977 047406 051104 032040 020040
5978 047414 050040 053523 020040
5979 047422 020040 052040 051505
5980 047430 047124 020117 042440
5981 047436 051122 051117 041520
5982 047444 000
5983 047445 126 041056 040456 DH32: .ASCIZ /V.B.A. KIPDR4 SRO WAS SR2 WAS TESTNO ERRORPC/
5984 047452 020056 045440 050111
5985 047460 051104 020064 051440
5986 047466 030122 053440 051501
5987 047474 051440 031122 053440
5988 047502 051501 052040 051505
5989 047510 047124 020117 042440
5990 047516 051122 051117 041520
5991 047524 000
5992 047525 126 041056 040456 DH33: .ASCIZ /V.B.A. KIPDR4 TESTNO ERRORPC/
5993 047532 020056 045440 050111
5994 047540 051104 020064 052040
5995 047546 051505 047124 020117
5996 047554 042440 051122 051117
5997 047562 041520 000
5998 047565 126 041056 040456 DH34: .ASCIZ /V.B.A. KIPDR4 SRO WAS EXPECTD TESTNO ERRORPC/
5999 047572 020056 045440 050111
6000 047600 051104 020064 051440
6001 047606 030122 053440 051501
6002 047614 042440 050130 041505
6003 047622 042124 052040 051505
6004 047630 047*24 020117 042440
6005 047636 051122 051117 041520
6006 047644 000
6007 047645 126 041056 040456 DH35: .ASCIZ /V.B.A. KIPDR4 SR2 WAS EXPECTD TESTNO ERRORPC/
    
```

6008	047652	020366	045440	050111					
6009	047660	051104	020064	051440					
6010	047666	031122	053440	051501					
6011	047674	042440	050130	041505					
6012	047702	042124	052040	051505					
6013	047710	047124	020117	042440					
6014	047716	051122	051117	041520					
6015	047724	000							
6016	047725	123	031122	053440	DH36:	.ASCIZ	/SR2 WAS EXPECTD	TESTNO	ERRORPC/
6017	047732	051501	042440	050130					
6018	047740	041505	042124	052040					
6019	047746	051505	047124	020117					
6020	047754	042440	051122	051117					
6021	047762	041520	000						
6022	047765	106	051111	052123	DH37:	.ASCII	/FIRST ABORT	SECOND ABORT/<CRLF>	
6023	047772	040440	047502	052122					
6024	050000	020040	020040	051440					
6025	050006	041505	047117	020104					
6026	050014	041101	051117	100124					
6027	050022	051123	020060	040527		.ASCIZ	/SRO WAS SR2 WAS SRO WAS SR2 WAS	TESTNO	ERRORPC/
6028	050030	020123	051123	020062					
6029	050036	040527	020123	051123					
6030	050044	020060	040527	020123					
6031	050052	051123	020062	040527					
6032	050060	020123	042524	052123					
6033	050066	047516	020040	051105					
6034	050074	047522	050122	000103					
6035	050102	051123	020060	040527	DH40:	.ASCIZ	/SRO WAS SR2 WAS	TESTNO	ERRORPC/
6036	050110	020123	051123	020062					
6037	050116	040527	020123	042524					
6038	050124	052123	047516	020040					
6039	050132	051105	047522	050122					
6040	050140	000103							
6041	050142	05120	020127	040527	DH42:	.ASCIZ	/PSW WAS R6 WAS	TESTNO	ERRORPC/
6042	050150	020123	033122	053440					
6043	050156	051501	020040	042524					
6044	050164	052123	047516	020040					
6045	050172	051105	047522	050122					
6046	050200	000103							
6047	050202	054105	042520	052103	DH44:	.ASCII	/EXPECTED	RECEIVED/<CRLF>	
6048	050210	042105	020040	020040					
6049	050216	020040	020040	051040					
6050	050224	041505	044505	042526					
6051	050232	100104							
6052	050234	051123	020060	020040		.ASCIZ	/SRO	SR2	SRO WAS SR2 WAS
6053	050242	020040	051123	020062					TESTNO
6054	050250	020040	020040	051123					ERRORPC/
6055	050256	020060	040527	020123					
6056	050264	051123	020062	040527					
6057	050272	020123	042524	052123					
6058	050300	047516	020040	051105					
6059	050306	047522	050122	000103					
6060	050314	054105	042520	052103	DH45:	.ASCII	/EXPECTED:/<CRLF>		
6061	050322	042105	100072						
6062	050326	051520	020127	020040		.ASCII	/PSW	PC	SRO
6063	050334	020040	041520	020040					SR2/<CRLF>

6064	050342	020040	020040	051123					
6065	050350	020060	020040	020040					
6066	050356	051123	100062						
6067	050362	033461	030060	033461		.ASCII	/170017 (3\$+4)	020147 (3\$)/<CRLF>	
6068	050370	020040	031450	025444					
6069	050376	024464	020040	031060					
6070	050404	030460	033464	020040					
6071	050412	031450	024444	200		.ASCII	/RECEIVED:/<CRLF>		
6072	050417	122	041505	044505					
6073	050424	042526	035104	200					
6074	050431	120	053523	020040		.ASCIZ	/PSW	PC	SRO
6075	050436	020040	050040	020103					SR2
6076	050444	020040	020040	051440					TESTNO
6077	050452	03022	020040	020040					ERRORPC/
6078	050460	051440	031122	020040					
6079	050466	020040	052040	051505					
6080	050474	047124	020117	042440					
6081	050502	051122	051117	041520					
6082	050510	000							
6083	050511	104	052101	020101	DH46:	.ASCII	/DATA	DATA/<CRLF>	
6084	050516	020040	042040	052101					
6085	050524	100101							
6086	050526	054105	042520	052103		.ASCIZ	/EXPECTD RECEIVD	TESTNO	ERRORPC/
6087	050534	020104	042522	042503					
6088	050542	053111	020104	042524					
6089	050550	052123	047516	020040					
6090	050556	051105	047522	050122					
6091	050564	000103							
6092	050566	042524	052123	047516	DH50:	.ASCIZ	/TESTNO	ERRORPC/	
6093	050574	020040	051105	047522					
6094	050602	050122	000103						
6095	050606	051123	020060	040527	DH51:	.ASCIZ	/SRO WAS SR2 WAS	TESTNO	ERRORPC/
6096	050614	020123	051123	020062					
6097	050622	040527	020123	042524					
6098	050630	052123	047516	020040					
6099	050636	051105	047522	050122					
6100	050644	000103							
6101	050646	044120	051531	041511	DH52:	.ASCII	/PHYSICL PAR 4/<CRLF>		
6102	050654	020114	040520	020122					
6103	050662	100064							
6104	050664	042101	051104	051505		.ASCIZ	/ADDRESS V.B.A.	PAR 4	SRO WAS SR2 WAS
6105	050672	020123	027126	027102					PSW
6106	050700	027101	020040	040520					TESTNO
6107	050706	020122	020064	020040					ERRORPC/
6108	050714	051123	020060	040527					
6109	050722	020123	051123	020062					
6110	050730	040527	020123	051520					
6111	050736	020127	020040	020040					
6112	050744	042524	052123	047516					
6113	050752	020040	051105	047522					
6114	050760	050122	000103						
6115	050764	051123	020060	040527	DH57:	.ASCIZ	/SRO WAS EXPECTD	TESTNO	ERRORPC/
6116	050772	020123	054105	042520					
6117	051000	052103	020104	042524					
6118	051006	052123	047516	020040					
6119	051014	051105	047522	050122					

6120	051022	000103							
6121	051024	051520	020127	040527	DH60:	.ASCIZ	/PSW WAS EXPECTD	TESTNO	ERRORPC/
6122	051032	020123	054105	042520					
6123	051040	052103	020104	042524					
6124	051046	052123	047516	020040					
6125	051054	051105	047522	050122					
6126	051062	000103							
6127	051064	051123	020060	020040	DH61:	.ASCIZ	/SRO SR2	TESTNO	ERRORPC/
6128	051072	020040	051123	020062					
6129	051100	020040	042524	052123					
6130	051106	047516	020040	051105					
6131	051114	047522	050122	000103					
6132	051122	051123	020062	054105	DH62:	.ASCIZ	/SR2 EXP SR2 REC	TESTNO	ERRORPC/
6133	051130	020120	051123	020062					
6134	051136	042522	020103	042524					
6135	051144	052123	047516	042440					
6136	051152	051122	051117	041520					
6137	051160	000							
6138	051161	123	030122	042440	DH63:	.ASCIZ	/SRO EXP SRO REC	TESTNO	ERRORPC/
6139	051166	050130	051440	030122					
6140	051174	051040	041505	052040					
6141	051202	051505	047124	020117					
6142	051210	051105	047522	050122					
6143	051216	000103							
6144									
6145						.EVEN			
6146									
6147	051220	001356	001360	001354	DT1:	.WORD	TRAPPC, TRAPPS, WASR6,	TESTNO,	\$ERRPC, 0
6148	051226	001352	001116	000000					
6149	051234	001356	001360	001354	DT2:	.WORD	TRAPPC, TRAPPS, WASR6, WASSR0, WASSR2, TESTNO, \$ERRPC, 0		
6150	051242	001366	001370	001352					
6151	051250	001116	000000						
6152	051254	001162	001164	001352	DT3:	.WORD	\$REG0, \$REG1, TESTNO, \$ERRPC, 0		
6153	051262	001116	000000						
6154	051266	001162	001352	001116	DT7:	.WORD	\$REG0, TESTNO, \$ERRPC, 0		
6155	051274	000000							
6156	051276	001374	001376	001400	DT10:	.WORD	ANDADR, ORADR, TONUM, TESTNO, \$ERRPC, 0		
6157	051304	001352	001116	000000					
6158	051312	001162	001164	001164	DT11:	.WORD	\$REG0, \$REG1, \$REG1, TESTNO, \$ERRPC, 0		
6159	051320	001352	001116	000000					
6160	051326	001162	001164	001166	DT12:	.WORD	\$REG0, \$REG1, \$REG2, \$REG2, TESTNO, \$ERRPC, 0		
6161	051334	001166	001352	001116					
6162	051342	000000							
6163	051344	001162	001352	001116	DT13:	.WORD	\$REG0, TESTNO, \$ERRPC, 0		
6164	051352	000000							
6165	051354	001162	001164	001174	DT16:	.WORD	\$REG0, \$REG1, \$REG5, \$REG2, TESTNO, \$ERRPC, 0		
6166	051362	001166	001352	001116					
6167	051370	000000							
6168	051372	001406	001402	001172	DT17:	.WORD	PBALO, VIRT1, \$REG4, TESTNO, \$ERRPC, 0		
6169	051400	001352	001116	000000					
6170	051406	001406	001402	001404	DT20:	.WORD	PBALO, VIRT1, VIRT2, \$REG4, \$REG5, \$TMP0, TESTNO, \$ERRPC, 0		
6171	051414	001172	001174	001176					
6172	051422	001352	001116	000000					
6173	051430	001174	001176	001352	DT21:	.WORD	\$REG5, \$REG3, TESTNO, \$ERRPC, 0		
6174	051436	001116	000000						
6175	051442	001162	001174	001170	DT22:	.WORD	\$REG0, \$REG5, \$REG3, TESTNO, \$ERRPC, 0		

6176	051450	001352	001116	000000					
6177	051456	001174	001352	001116	DT23:	.WORD	\$REG5, TESTNO, \$ERRPC, 0		
6178	051464	000000							
6179	051466	001166	001164	001352	DT24:	.WORD	\$REG2, \$REG1, TESTNO, \$ERRPC, 0		
6180	051474	001116	000000						
6181	051500	001166	001176	001352	DT26:	.WORD	\$REG2, \$TMP0, TESTNO, \$ERRPC, 0		
6182	051506	001116	000000						
6183	051512	001366	001170	001166	DT30:	.WORD	WASSR0, \$REG3, \$REG2, \$TMP0, TESTNO, \$ERRPC, 0		
6184	051520	001176	001352	001116					
6185	051526	000000							
6186	051530	001370	001172	001166	DT31:	.WORD	WASSR2, \$REG4, \$REG2, \$TMP0, TESTNO, \$ERRPC, 0		
6187	051536	001176	001352	001116					
6188	051544	000000							
6189	051546	001162	001172	001366	DT32:	.WORD	\$REG0, \$REG4, WASSR0, WASSR2, TESTNO, \$ERRPC, 0		
6190	051554	001370	001352	001116					
6191	051562	000000							
6192	051564	001162	001172	001352	DT33:	.WORD	\$REG0, \$REG4, TESTNO, \$ERRPC, 0		
6193	051572	001116	000000						
6194	051576	001162	001172	001366	DT34:	.WORD	\$REG0, \$REG4, WASSR0, \$REG2, TESTNO, \$ERRPC, 0		
6195	051604	001166	001352	001116					
6196	051612	000000							
6197	051614	001162	001172	001370	DT35:	.WORD	\$REG0, \$REG4, WASSR2, \$REG3, TESTNO, \$ERRPC, 0		
6198	051622	001170	001352	001116					
6199	051630	000000							
6200	051632	001370	001164	001352	DT36:	.WORD	WASSR2, \$REG1, TESTNO, \$ERRPC, 0		
6201	051640	001116	000000						
6202	051644	001176	001202	001366	DT37:	.WORD	\$TMP0, \$TMP2, WASSR0, WASSR2, TESTNO, \$ERRPC, 0		
6203	051652	001370	001352	001116					
6204	051660	000000							
6205	051662	001366	001370	001352	DT40:	.WORD	WASSR0, WASSR2, TESTNO, \$ERRPC, 0		
6206	051670	001116	000000						
6207	051674	001164	001166	001352	DT42:	.WORD	\$REG1, \$REG2, TESTNO, \$ERRPC, 0		
6208	051702	001116	000000						
6209	051706	001:62	001164	001366	DT44:	.WORD	\$REG0, \$REG1, WASSR0, WASSR2, TESTNO, \$ERRPC, 0		
6210	051714	001370	001352	001116					
6211	051722	000000							
6212	051724	001164	001170	001366	DT45:	.WORD	\$REG1, \$REG3, WASSR0, WASSR2, TESTNO, \$ERRPC, 0		
6213	051732	001370	001352	001116					
6214	051740	000000							
6215	051742	001162	001164	001352	DT46:	.WORD	\$REG0, \$REG1, TESTNO, \$ERRPC, 0		
6216	051750	001116	000000						
6217	051754	001352	001116	000000	DT50:	.WORD	TESTNO, \$ERRPC, 0		
6218	051762	001366	001370	001352	DT51:	.WORD	WASSR0, WASSR2, TESTNO, \$ERRPC, 0		
6219	051770	001116	000000						
6220	051774	001406	001402	001172	DT52:	.WORD	PBALO, VIRT1, \$REG4, WASSR0, WASSR2, \$TMP0, TESTNO, \$ERRPC, 0		
6221	052002	001366	001370	001176					
6222	052010	001352	001116	000000					
6223	052016	001366	001164	001352	DT57:	.WORD	WASSR0, \$REG1, TESTNO, \$ERRPC, 0		
6224	052024	001116	000000						
6225	052030	001164	001166	001352	DT59:	.WORD	\$REG1, \$REG2, TESTNO, \$ERRPC, 0		
6226	052036	001116	000000						
6227	052042	001370	001172	001352	DT64:	.WORD	WASSR2, \$REG4, TESTNO, \$ERRPC, 0		
6228	052050	001116	000000						
6229	052054	001366	001370	001352	DT65:	.WORD	WASSR0, WASSR2, TESTNO, \$ERRPC, 0		
6230	052062	001116	000000						
6231	052066	001364	001370	001352	DT66:	.WORD	CORSR2, WASSR2, TESTNO, \$ERRPC, 0		

```

6232 052074 001116 000000
6233 052100 001362 001366 001352 DT67: .WORD CORSRO,WASSRO,TESTNO,SERRPC.0
6234 052106 001116 000000
6235
6236 052112 000 000 000 DF1: .BYTE 0,0,0,0,0
6237 052115 000 000 000 DF2: .BYTE 0,0,0,0,0,0,0
6238 052117 000 000 000
6239 052122 000 000 000
6240 052125 000
6241 052126 000 000 000 DF3: .BYTE 0,0,0,0
6242 052131 000
6243 052132 000 000 000 DF7: .BYTE 0,0,0
6244 052135 000 000 001 DF10: .BYTE 0,0,1,0,0
6245 052140 000 000
6246 052142 000 000 002 DF11: .BYTE 0,0,2,0,0
6247 052145 000 000
6248 052147 000 000 000 DF12: .BYTE 0,0,0,2,0,0
6249 052152 002 000 000
6250 052155 000 000 000 DF13: .BYTE 0,0,0
6251 052160 000 000 000 DF16: .BYTE 0,0,0,0,0,0
6252 052163 000 000 000
6253 052166 003 000 000 DF17: .BYTE 3,0,0,0,0
6254 052171 000 000 000
6255 052173 003 000 000 DF20: .BYTE 3,0,0,0,0,0,0,0
6256 052176 000 000 000
6257 052201 000 000
6258 052203 000 000 000 DF21: .BYTE 0,0,0,0
6259 052206 000
6260 052207 000 000 000 DF22: .BYTE 0,0,0,0,0
6261 052212 000 000
6262 052214 000 000 000 DF23: .BYTE 0,0,0
6263 052217 000 000 000 DF24: .BYTE 0,0,0,0
6264 052222 000
6265 052223 000 000 000 DF30: .BYTE 0,0,0,0,0,0
6266 052226 000 000 000
6267 052231 000 000 000 DF46: .BYTE 0,0,0,0
6268 052234 000
6269 052235 000 000 000 DF50: .BYTE 0,0
6270 052237 000 000 000 DF51: .BYTE 0,0,0,0
6271 052242 000
6272 052243 003 000 000 DF52: .BYTE 3,0,0,0,0,0,0,0
6273 052246 000 000 000
6274 052251 000 000
6275 052253 000 000 000 DF57: .BYTE 0,0,0,0
6276 052256 000
6277 052257 000 000 000 DF60: .BYTE 0,0,0,0
6278 052262 000
6279 052263 000 000 000 DF61: .BYTE 0,0,0,0
6280 052266 000
6281 000001 .END
6282
    
```

```

ABASE = 000000      815      856
ACDW1 = 000000      815      858
ACDW2 = 000000      815      859
ACPUOP= 000000      815      830
ADDW0 = 000000      815      860
ADDW1 = 000000      815      861
ADDW10= 000000      815      870
ADDW11= 000000      815      871
ADDW12= 000000      815      872
ADDW13= 000000      815      873
ADDW14= 000000      815      874
ADDW15= 000000      815      875
ADDW2 = 000000      815      862
ADDW3 = 000000      815      863
ADDW4 = 000000      815      864
ADDW5 = 000000      815      865
ADDW6 = 000000      815      866
ADDW7 = 000000      815      867
ADDW8 = 000000      815      868
ADDW9 = 000000      815      869
ADEVCT= 000000      815      821
ADEVM = 000000      815      857
AENV = 000000      815      826
AENVM = 000000      815      827
AFATAL= 000000      815      818
AMADR1= 000000      815      843
AMADR2= 000000      815      847
AMADR3= 000000      815      850
AMADR4= 000000      815      853
AMAMS1= 000000      815      837
AMAMS2= 000000      815      845
AMAMS3= 000000      815      848
AMAMS4= 000000      815      851
AMSAD= 000000      815      823
AMSGLG= 000000      815      824
AMSGTY= 000000      815      817
AMTYP1= 000000      815      838
AMTYP2= 000000      815      846
AMTYP3= 000000      815      849
AMTYP4= 000000      815      852
ANDADR 001374      890#    1580*  1602*  1620*  1642*  1680*  1682*  1700*  1722*  1740*  1762*  8158
APASS = 000000      815      820
APRIOR= 000000      815
APTCSU= 000040      4992    5097#
APTENV= 000001      4457    4985    5053    5095#
APTSIZ= 000200      1398    5094#
APTSPO= 000100      4987    5055    5098#
ASWREG= 000000      815      828
ATESTN= 000000      815      819
AUNIT = 000000      815      822
AUSWR = 000000      815      829
AVECT1= 000000      815      854
AVECT2= 000000      815      855
BIT0 = 000001      628#
BIT00 = 000001      618#    628
BIT01 = 000002      617#    627
    
```


.SWRLD	533#		
.\$ACT1	1#	515#	722
.\$APT8	1#	812#	
.\$APTH	1#	515#	732
.\$APTY	1#	515#	5041
.\$ASTA	1#		
.\$CATC	1#	515#	711
.\$CMTA	1#	515#	754
.\$DB2D	1#		
.\$DB2D	1#	515#	5311
.\$DIV	1#		
.\$EOP	1#	515#	4285
.\$ERRD	1#	515#	4419
.\$ERRT	1#	515#	
.\$MULT	1#		
.\$POWE	1#	515#	5395
.\$RAND	1#		
.\$RDE	1#		
.\$RDOC	1#		
.\$RFAD	1#	515#	4747
.\$R2AZ	1#		
.\$SAVE	1#	515#	5266
.\$SB2D	1#		
.\$SU2D	1#		
.\$SCDP	1#	515#	4355
.\$SIZE	1#		
.\$SUPR	1#		
.\$TRAP	1#	515#	5350
.\$TYPB	1#	515#	5098
.\$TYPD	1#	515#	5199
.\$TYPE	1#	515#	4962
.\$TYPD	1#	515#	5122
.\$40CA	1#		
.1170	1#		

. ABS. 052267 000

ERRORS DETECTED: 0

CFKTHB.CFKTHB.LST/CRF/SOL=CFKTHB.SML,CFKTHB.P11
RUN-TIME: 22 30 2 SECONDS
RUN-TIME RATIO: 266/55=4.8
CORE USED: 39K (77 PAGES)