


KD11-E

11/34 MEMORY MANAGEMENT
MD-11-DFKTH-A

EP-DFKTH-A-DL-A
COPYRIGHT © 1977
FICHE 1 OF 1

APR 1977

MADE IN USA



This microfiche card contains 374 frames of technical data, organized into 11 columns and 34 rows. The frames contain various types of information, including:

- Tables with multiple columns and rows of data.
- Text blocks, possibly instructions or descriptions.
- Diagrams or flowcharts.
- Small charts or graphs.

The data is presented in a highly structured and repetitive format, typical of microfiche used for technical documentation.

B01

MD-11-DFKTH-A 08:51 PAGE 00010000
DFKTHA.P11 19-JAN-77 16:02

770325

PDP10 411

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG MACY11 27(100

.REM 2

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

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DFKTH-A-D
PRODUCT NAME: PDP 11/34 MEMORY MANAGEMENT DIAGNOSTIC
DATE: JANUARY 31, 1977
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 BY DIGITAL EQUIPMENT CORPORATION

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
1591.0 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 (SEERR). 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 "SLPERR". AFTER REPORTING
THE ERROR, "SLPERR" 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 WRITE  READ  READ-(BINARY)
ADDRESS (OCTAL) (OCTAL) 5432109876543210  TESTNO  ERRORPC
177572  040000  060000  0110000000000000  000012  022060
```

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

4.0 MISCELLANEOUS INFORMATION

4.1 ACT/APT/XXDP COMPATABILITY

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

4.2 END-OF-PASS MESSAGE

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

END OF PASS #2 TOTAL ERRORS SINCE LAST REPORT 0

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

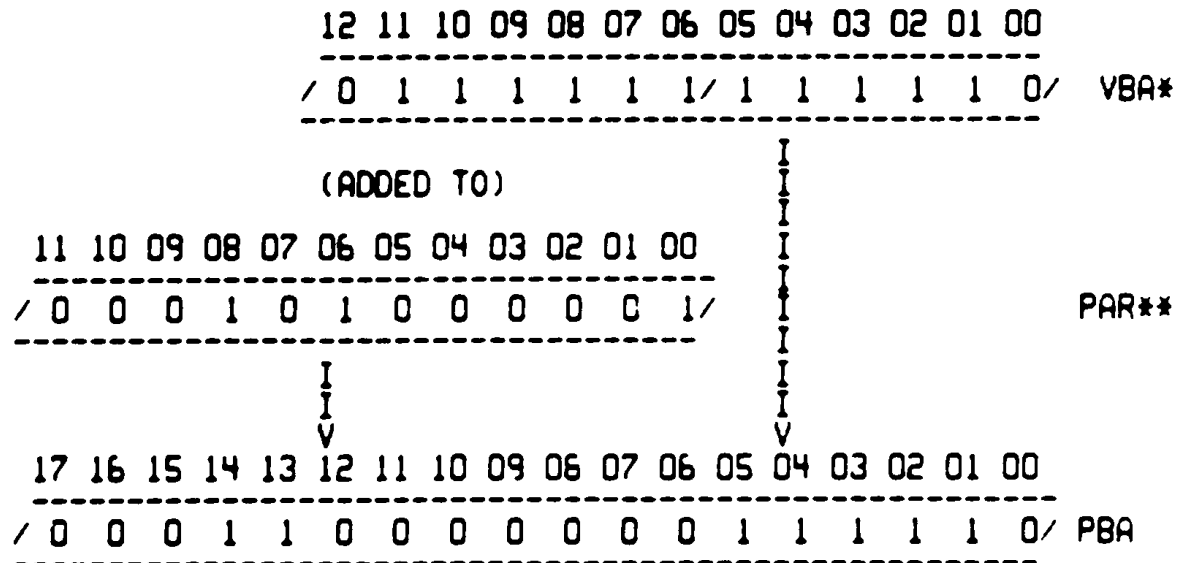
THE "T-BIT" (BIT 4) IN THE PROCESSOR STATUS WORD IS SET BY AN "RTI" IN THE END-OF-PASS ROUTINE FOR EVERY OTHER PASS BEGINNING WITH THE THIRD PASS (PASSES 3, 5, 7, 9...). T-BIT TRAPPING CAN BE INHIBITED BY SETTING BIT 12 = 1 IN THE SWITCH REGISTER (SEE SECTION 2.4).

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 PLR
**= PSW MODE BIT 01 (BIT 15) SELECTS THE USER (=1) OR

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

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
4957.0 PROGRAM DESCRIPTION
-----5.1 SUBROUTINES USED BY THIS PROGRAM

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

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

5.2 PROGRAM LISTING

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

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

5.3 USING THE PROGRAM TO DIAGNOSE A FAULT

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

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

2

```

515
516 .TITLE MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
517 *COPYRIGHT (C) JAN-77
518 *DIGITAL EQUIPMENT CORP.
519 *MAYNARD, MASS. 01754
520 *
521 *
522 *THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
523 *PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
524 *
525 .SBTTL OPERATIONAL SWITCH SETTINGS
526 *
527 *          SWITCH          USE
528 *          -----          -----
529 *          15          HALT ON ERROR
530 *          14          LOOP ON TEST
531 *          13          INHIBIT ERROR TYPEOUTS
532 *          12          INHIBIT TRACE TRAP
533 *          11          INHIBIT ITERATIONS
534 *          10          BELL ON ERROR
535 *          9           LOOP ON ERROR
536 *          8           LOOP ON TEST IN SWR<7:0>
537 .SBTTL BASIC DEFINITIONS
538 *
539 *INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
540 001100 STACK= 1100
541 .EQUIV EMT,ERROR          ;;BASIC DEFINITION OF ERROR CALL
542 .EQUIV IOT,SCOPE          ;;BASIC DEFINITION OF SCOPE CALL
543
544 *MISCELLANEOUS DEFINITIONS
545 000011 HT= 11          ;;CODE FOR HORIZONTAL TAB
546 000012 LF= 12          ;;CODE FOR LINE FEED
547 000015 CR= 15          ;;CODE FOR CARRIAGE RETURN
548 000200 CRLF= 200        ;;CODE FOR CARRIAGE RETURN-LINE FEED
549 177776 PS= 177776      ;;PROCESSOR STATUS WORD
550 .EQUIV PS,PSW
551 177774 STKLMT= 177774   ;;STACK LIMIT REGISTER
552 177772 PIRQ= 177772     ;;PROGRAM INTERRUPT REQUEST REGISTER
553 177570 DSWR= 177570    ;;HARDWARE SWITCH REGISTER
554 177570 DDISP= 177570   ;;HARDWARE DISPLAY REGISTER
555
556 *GENERAL PURPOSE REGISTER DEFINITIONS
557 000000 R0= %0          ;;GENERAL REGISTER
558 000001 R1= %1          ;;GENERAL REGISTER
559 000002 R2= %2          ;;GENERAL REGISTER
560 000003 R3= %3          ;;GENERAL REGISTER
561 000004 R4= %4          ;;GENERAL REGISTER
562 000005 R5= %5          ;;GENERAL REGISTER
563 000006 R6= %6          ;;GENERAL REGISTER
564 0J0007 R7= %7          ;;GENERAL REGISTER
565 0J0006 SP= %6          ;;STACK POINTER
566 000007 PC= %7          ;;PROGRAM COUNTER
567
568 *PRIORITY LEVEL DEFINITIONS
569 000000 PRO= 0          ;;PRIORITY LEVEL 0
570 000040 PRI= 40        ;;PRIORITY LEVEL 1

```

571 000100
572 000140
573 000200
574 000240
575 000300
576 000340
577
578

PR2= 100
PR3= 140
PR4= 200
PR5= 240
PR6= 300
PR7= 340
::: PRIORITY LEVEL 2
::: PRIORITY LEVEL 3
::: PRIORITY LEVEL 4
::: PRIORITY LEVEL 5
::: PRIORITY LEVEL 6
::: PRIORITY LEVEL 7

579 100000
580 040000
581 020000
582 010000
583 004000
584 002000
585 001000
586 000400
587 000200
588 000100
589 000040
590 000020
591 000010
592 000004
593 000002
594 000001
595
596
597
598
599

.*"SWITCH REGISTER" SWITCH DEFINITIONS

SW15= 100000
SW14= 40000
SW13= 20000
SW12= 10000
SW11= 4000
SW10= 2000
SW09= 1000
SW08= 400
SW07= 200
SW06= 100
SW05= 40
SW04= 20
SW03= 10
SW02= 4
SW01= 2
SW00= 1

600
601
602
603
604
605

.EQUIV SW09, SW9
.EQUIV SW08, SW8
.EQUIV SW07, SW7
.EQUIV SW06, SW6
.EQUIV SW05, SW5
.EQUIV SW04, SW4
.EQUIV SW03, SW3
.EQUIV SW02, SW2
.EQUIV SW01, SW1
.EQUIV SW00, SW0

606 100000
607 040000
608 020000
609 010000
610 004000
611 002000
612 001000
613 000400
614 000200
615 000100
616 000040
617 000020
618 000010
619 000004
620 000002
621 000001
622
623
624
625

.*DATA BIT DEFINITIONS (BIT00 TO BIT15)

BIT15= 100000
BIT14= 40000
BIT13= 20000
BIT12= 10000
BIT11= 4000
BIT10= 2000
BIT09= 1000
BIT08= 400
BIT07= 200
BIT06= 100
BIT05= 40
BIT04= 20
BIT03= 10
BIT02= 4
BIT01= 2
BIT00= 1

.EQUIV BIT09, BIT9
.EQUIV BIT08, BIT8
.EQUIV BIT07, BIT7
.EQUIV BIT06, BIT6

627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682

000004
000010
000014
000014
000014
000020
000024
000030
000034
000060
000064
000240

000250

177572
177574
177576
172516

177600
177602
177604
177606
177610
177612
177614
177616

177640
177642
177644
177646
177650
177652
177654
177656

```
.EQUIV BIT05,BIT5
.EQUIV BIT04,BIT4
.EQUIV BIT03,BIT3
.EQUIV BIT02,BIT2
.EQUIV BIT01,BIT1
.EQUIV BIT00,BIT0

;#BASIC "CPU" TRAP VECTOR ADDRESSES
ERRVEC= 4           ; TIME OUT AND OTHER ERRORS
RESVEC= 10          ; RESERVED AND ILLEGAL INSTRUCTIONS
TBITVEC=14          ; "T" BIT
TRTVEC= 14          ; TRACE TRAP
BPTVEC= 14          ; BREAKPOINT TRAP (BPT)
IOTVEC= 20          ; INPUT/OUTPUT TRAP (IOT) **SCOPE**
PWRVEC= 24          ; POWER FAIL
EMTVEC= 30          ; EMULATOR TRAP (EMT) **ERROR**
TRAPVEC=34          ; "TRAP" TRAP
TKVEC= 60           ; TTY KEYBOARD VECTOR
TPVEC= 64           ; TTY PRINTER VECTOR
PIRVEC=240          ; PROGRAM INTERRUPT REQUEST VECTOR
.SBTTL MEMORY MANAGEMENT DEFINITIONS

;#KT11 VECTOR ADDRESS
MMVEC= 250

;#KT11 STATUS REGISTER ADDRESSES
SR0= 177572
SR1= 177574
SR2= 177576
SR3= 172516

;#USER "I" PAGE DESCRIPTOR REGISTERS
UIPDR0= 177600
UIPDR1= 177602
UIPDR2= 177604
UIPDR3= 177606
UIPDR4= 177610
UIPDR5= 177612
UIPDR6= 177614
UIPDR7= 177616

;#USER "I" PAGE ADDRESS REGISTERS
UIPAR0= 177640
UIPAR1= 177642
UIPAR2= 177644
UIPAR3= 177646
UIPAR4= 177650
UIPAR5= 177652
UIPAR6= 177654
UIPAR7= 177656

;#KERNEL "I" PAGE DESCRIPTOR REGISTERS
```

683
684 172300
685 172302
686 172304
687 172306
688 172310
689 172312
690 172314
691 172316
692
693
694
695 172340
696 172342
697 172344
698 172346
699 172350
700 172352
701 172354
702 172356
703
704
705
706
707
708 001100
709 000700
710
711
712
713
714
715
716
717 000000
718
719
720
721 000174
722 000174 000000
723 000176 000000
724
725 000200 000137 020000
726
727
728
729
730 000204
731 000046 034410
732 000046 034410
733 000052 000052
734 000052 000000
735 000204
736
737
738

KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316

;*KERNEL "I" PAGE ADDRESS REGISTERS

KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356

.EQUIV SP,KSP
.EQUIV SP,USP
.EQUIV BIT4,TBIT
.EQUIV BIT6,WBIT
KERSTK= STACK
USESTK= STACK-200

;*ADDITIONAL DEFINITIONS
;*

.SBTTL TRAP CATCHER

.=0
;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS

.=174
DISPREG: .WORD 0 ;; SOFTWARE DISPLAY REGISTER
SWREG: .WORD 0 ;; SOFTWARE SWITCH REGISTER

.SBTTL STARTING ADDRESS(ES)
JMP @*START ;; JUMP TO STARTING ADDRESS OF PROGRAM
.SBTTL ACT11 HOOKS

;HOOKS REQUIRED BY ACT11

\$SVPC=.; ;SAVE PC
.=46
\$ENDAD ;;1)SET LOC.46 TO ADDRESS OF \$ENDAD IN .\$EOP
.=52
.WORD 0 ;;2)SET LOC.52 TO ZERO
.\$SVPC ;; RESTORE PC

.SBTTL APT PARAMETER BLOCK

739		
740		
741	000204	
742	000024	
743	000024	000200
744		000044
745	000044	000204
746		000204
747		
748		
749		
750		
751	000204	
752	000204	000000
753	000206	001226
754	000210	000010
755	000212	000020
756	000214	000005
757	000216	000052

```

;SET LOCATIONS 24 AND 44 AS REQUIRED FOR AP1
;*****
;$.X=.      ;;SAVE CURRENT LOCATION
;.=24      ;;SET POWER FAIL TO POINT TO START OF PROGRAM
;200       ;;FOR APT START UP
;.=44      ;;POINT TO APT INDIRECT ADDRESS PNTR.
;$APTHDR    ;;POINT TO APT HEADER BLOCK
;=.$X      ;;RESET LOCATION COUNTER
;*****
;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
;INTERFACE SPEC.

```

```

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

```

```

758
759
760
761
762
763
764      001100
765      001100 000000
766      001100 000
767      001102 000
768      001103 000
769      001104 000000
770      001106 000000
771      001110 000000
772      001112 000000
773      001114 000
774      001115 00i
775      001116 000000
776      001120 000000
777      001122 000000
778      001124 000000
779      001126 000000
780      001130 000000
781      001132 000000
782      001134 000
783      001135 000
784      001136 000000
785      001140 177570
786      001142 177570
787      001144 177560
788      001146 177562
789      001150 177564
790      001152 177566
791      001154 000
792      001155 002
793      001156 012
794      001157 000
795      001160 000000
796
797      001162 000000
798      001164 000000
799      001166 000000
800      001170 000000
801      001172 000000
802      001174 000000
803      001176 000000
804      001200 000000
805      001202 000000
806      001204 000000
807      001206 000000
808      001210 000000
809      001212 000000
810      001214 000000
811      001216 177607 000377
812      001222 077
813      001223 015
    
```

.SBTTL COMMON TAGS

```

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

.=1100

```

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

814 001224 000012
815
816
817
818
819
820 001226
821 001226 000000
822 001230 000000
823 001232 000000
824 001234 000000
825 001236 000000
826 001240 000000
827 001242 000000
828 001244 000000
829 001246
830 001246 000
831 001247 000
832 001250 000000
833 001252 000000
834 001254 000000
835
836
837
838
839
840
841 001256 000
842 001257 000
843
844
845
846
847 001260 000000
848
849 001262 000
850 001263 000
851 001264 000000
852 001266 000
853 001267 000
854 001270 000000
855 001272 000
856 001273 000
857 001274 000000
858 001276 000000
859 001300 000000
860 001302 000000
861 001304 000000
862 001306 000000
863 001310 000000
864 001312 000000
865 001314 000000
866 001316 000000
867 001320 000000
868 001322 000000
869 001324 000000

```

$LF: .ASCIZ <12> ;:LINE FEED
;:*****
;SBTTL APT MAILBOX-ETABLE
;:*****
.EVEN
$MAIL: ;: APT MAILBOX
$MSGTY: .WORD AMSGTY ;: MESSAGE TYPE CODE
$FATAL: .WORD AFATAL ;: FATAL ERROR NUMBER
$TESTN: .WORD ATESTN ;: TEST NUMBER
$PASS: .WORD APASS ;: PASS COUNT
$DEVCT: .WORD ADEVCT ;: DEVICE COUNT
$UNIT: .WORD AUNIT ;: I/O UNIT NUMBER
$MSGAD: .WORD AMSGAD ;: MESSAGE ADDRESS
$MSGLG: .WORD AMSGLG ;: MESSAGE LENGTH
$ETABLE: ;: APT ENVIRONMENT TABLE
$ENV: .BYTE AENV ;: ENVIRONMENT BYTE
$ENVM: .BYTE AENVM ;: ENVIRONMENT MODE BITS
$SWREG: .WORD ASWREG ;: APT SWITCH REGISTER
$USWR: .WORD AUSWR ;: USER SWITCHES
$CPUOP: .WORD ACPUOP ;: CPU TYPE, OPTIONS
;:
;: BITS 15-11=CPU TYPE
;: 11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05
;: 11/70=06, PDQ=07, Q=10
;:
;: BIT 10=REAL TIME CLOCK
;: BIT 9=FLOATING POINT PROCESSOR
;: BIT 8=MEMORY MANAGEMENT
$MAMS1: .BYTE AMAMS1 ;: HIGH ADDRESS, M.S. BYTE
$MTYP1: .BYTE AMTYP1 ;: MEM. TYPE, BLK#1
;: MEM. TYPE BYTE -- (HIGH BYTE)
;: 900 NSEC CORE=001
;: 300 NSEC BIPOLAR=002
;: 500 NSEC MOS=003
$MADR1: .WORD AMADR1 ;: HIGH ADDRESS, BLK#1
;: MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
$MAMS2: .BYTE AMAMS2 ;: HIGH ADDRESS, M.S. BYTE
$MTYP2: .BYTE AMTYP2 ;: MEM. TYPE, BLK#2
$MADR2: .WORD AMADR2 ;: MEM. LAST ADDRESS, BLK#2
$MAMS3: .BYTE AMAMS3 ;: HIGH ADDRESS, M.S. BYTE
$MTYP3: .BYTE AMTYP3 ;: MEM. TYPE, BLK#3
$MADR3: .WORD AMADR3 ;: MEM. LAST ADDRESS, BLK#3
$MAMS4: .BYTE AMAMS4 ;: HIGH ADDRESS, M.S. BYTE
$MTYP4: .BYTE AMTYP4 ;: MEM. TYPE, BLK#4
$MADR4: .WORD AMADR4 ;: MEM. LAST ADDRESS, BLK#4
$VECT1: .WORD AVECT1 ;: INTERRUPT VECTOR#1, BUS PRIORITY#1
$VECT2: .WORD AVECT2 ;: INTERRUPT VECTOR#2, BUS PRIORITY#2
$BASE: .WORD ABASE ;: BASE ADDRESS OF EQUIPMENT UNDER TEST
$DEVN: .WORD ADEVN ;: DEVICE MAP
$CDW1: .WORD ACDW1 ;: CONTROLLER DESCRIPTION WORD#1
$CDW2: .WORD ACDW2 ;: CONTROLLER DESCRIPTION WORD#2
$DDW0: .WORD ADDW0 ;: DEVICE DESCRIPTOR WORD#0
$DDW1: .WORD ADDW1 ;: DEVICE DESCRIPTOR WORD#1
$DDW2: .WORD ADDW2 ;: DEVICE DESCRIPTOR WORD#2
$DDW3: .WORD ADDW3 ;: DEVICE DESCRIPTOR WORD#3
$DDW4: .WORD ADDW4 ;: DEVICE DESCRIPTOR WORD#4
$DDW5: .WORD ADDW5 ;: DEVICE DESCRIPTOR WORD#5

```

870 001326 000000
871 001330 000000
872 001332 000000
873 001334 000000
874 001336 000000
875 001340 000000
876 001342 000000
877 001344 000000
878 001346 000000
879 001350 000000

\$DDW6: .WORD ADDW6 ::: DEVICE DESCRIPTOR WORD#6
\$DDW7: .WORD ADDW7 ::: DEVICE DESCRIPTOR WORD#7
\$DDW8: .WORD ADDW8 ::: DEVICE DESCRIPTOR WORD#8
\$DDW9: .WORD ADDW9 ::: DEVICE DESCRIPTOR WORD#9
\$DDW10: .WORD ADDW10 ::: DEVICE DESCRIPTOR WORD#10
\$DDW11: .WORD ADDW11 ::: DEVICE DESCRIPTOR WORD#11
\$DDW12: .WORD ADDW12 ::: DEVICE DESCRIPTOR WORD#12
\$DDW13: .WORD ADDW13 ::: DEVICE DESCRIPTOR WORD#13
\$DDW14: .WORD ADDW14 ::: DEVICE DESCRIPTOR WORD#14
\$DDW15: .WORD ADDW15 ::: DEVICE DESCRIPTOR WORD#15

880
881
882 001352

SETEND:

883
884
885 001352 000000
886 001354 000000
887 001356 000000
888 001360 000000
889 001362 000000
890 001364 000000
891 001366 000000
892 001370 000000
893 001372 000000
894 001374 000000
895 001376 000000
896 001400 000000
897 001402 000000
898 001404 000000

TESTNO: .WORD 0 ; HOLDS TEST NUMBER FOR TYPEOUTS
WASR6: .WORD 0 ; USED TO STORE THE STACK POINTER AFTER A TRAP
TRAPPC: .WORD 0 ; USED TO STORE THE PC OF A TRAP OR ABORT
TRAPPS: .WORD 0 ; USED TO STORE THE PS OF A TRAP OR ABORT
WASSR0: .WORD 0 ; USED TO STORE CONTENTS OF SR0
WASSR2: .WORD 0 ; USED TO STORE CONTENTS OF SR2
TBITPS: .WORD 0 ; SAVES THE PSM THAT MAY HAVE ITS T-BIT ON
ANDADR: .WORD 0 ; HOLDS RESULT OF ADDRESSES BEING AND-ED
ORADR: .WORD 0 ; HOLDS RESULT OF ADDRESSES BEING OR-ED
TONUM: .WORD 0 ; HOLDS NUMBER OF TIME-OUTS
VIRT1: .WORD 0 ; HOLDS VIRTUAL ADDRESS TO BE CONVERTED
VIRT2: .WORD 0 ;
PBALO: .WORD 0 ; HOLDS BITS <15:00> OF PHYSICAL ADDRESS
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:

899				
900				
901				
902				
903				
904				
905				
906				
907				
908				
909				
910				
911				
912				
913	001406			
914				
915				
916	001406	041246		
917	001410	044655		
918	001412	050122		
919	001414	050756		
920				
921				
922	001416	041306		
923	001420	044725		
924	001422	050136		
925	001424	050763		
926				
927				
928	001426	041355		
929	001430	045015		
930	001432	050156		
931	001434	050772		
932				
933				
934	001436	041414		
935	001440	045015		
936	001442	050156		
937	001444	050772		
938				
939				
940	001446	041447		
941	001450	045015		
942	001452	050156		
943	001454	050772		
944				
945				
946	001456	041522		
947	001460	045015		
948	001462	050156		
949	001464	050772		
950				
951				
952	001466	041567		
953	001470	045055		
954	001472	050170		

;*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 ;:0,0,0,0,0

;*ITEM 2
EM2 ;:UNEXPECTED MEM. MGMT. TRAP TO LOC. 250
DH2 ;:OLD PC OLD PSW R6 WAS SR0 SR2 TESTNO ERRORPC
DT2 ;:TRAPPC, TRAPPS, WASR6, WASSR0, WASSR2, TESTNO, \$ERRPC,
DF2 ;:0,0,0,0,0,0,0

;*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

955	001474	050776	DF7	;0,0,0
956				
957			;*ITEM 10	
958	001476	041625	EM10	;SUMMARY OF MEM. MGMT. REG. TIMEOUTS
959	001500	045105	DH10	;REGISTER-ADDRS NUM. OF
960				;AND-ED OR-ED TIMOUTS TESTNO ERRORPC
961	001502	050200	DT10	;ANDADR,ORADR,TONUM,TESTNO,\$ERRPC,0
962	001504	051001	DF10	;0,0,1,0,0
963				
964			;*ITEM 11	
965	001506	041671	EM11	;MEM. MGMT. REG. WOULD NOT CLEAR
966	001510	045205	DH11	;REGISTR READ READ-(BINARY)
967				;ADDRESS (OCTAL) 5432109876543210 TESTNO ERRORPC
968	001512	050214	DT11	;\$REG0,\$REG1,\$REG1,TESTNO,\$ERRPC,0
969	001514	051006	DF11	;0,0,2,0,0
970				
971			;*ITEM 12	
972	001516	041731	EM12	;MEM. MGMT. REG. BITS NOT SET CORRECTLY
973	001520	045325	DH12	;REGISTR WRITE READ READ
974				;ADDRESS (OCTAL) (OCTAL) (BINARY) TESTNO ERRORPC
975	001522	050230	DT12	;\$REG0,\$REG1,\$REG2,\$REG2,TESTNO,\$ERRPC,0
976	001524	051013	DF12	;0,0,0,2,0,0
977				
978			;*ITEM 13	
979	001526	042000	EM13	;SRO EFFECTED BY WRITE TO PSW
980	001530	045465	DH13	;READ TESTNO ERRORPC
981	001532	050246	DT13	;\$REG0,TESTNO,\$ERRPC,0
982	001534	051021	DF13	;0,0,0
983				
984			;*ITEM 14	
985	001536	042035	EM14	;SRI DID NOT READ ALL ZEROS
986	001540	045465	DH13	;READ TESTNO ERRORPC
987	001542	050246	DT13	;\$REG0,TESTNO,\$ERRPC,0
988	001544	051021	DF13	;0,0,0
989				
990			;*ITEM 15	
991	001546	042070	EM15	;DUAL ADDRESSING BETWEEN BYTES OF PAR OR PDR
992	001550	045325	DH12	;REGISTER WRITE READ READ
993				;ADDRESS (OCTAL) (OCTAL) (BINARY) TESTNO ERRORPC
994	001552	050230	DT12	;\$REG0,\$REG1,\$REG2,\$REG2,TESTNO,\$ERRPC,0
995	001554	051013	DF12	;0,0,0,2,0,0
996				
997			;*ITEM 16	
998	001556	042144	EM16	;DUAL ADDRESSING BETWEEN PAR-PDR'S
999	001560	045515	DH16	;PAR-PDR PAR-PDR
1000				;CLEARED EFFECTD EXPECTD RECEIVD TESTNO ERRORPC
1001	001562	050256	DT16	;\$REG0,\$REG1,\$REG5,\$REG2,TESTNO,\$ERRPC,0
1002	001564	051024	DF16	;0,0,0,0,0,0
1003				
1004			;*ITEM 17	
1005	001566	042206	EM17	;PHYS. ADDR. FORMED READ WRONG IN MAINT. MODE
1006	001570	045615	DH17	;PHYSICAL VIRTUAL
1007				;ADDRESS ADDRESS KIPAR4 TESTNO ERRORPC
1008	001572	050274	DT17	;\$REG0,VIRT1,\$REG4,TESTNO,\$ERRPC,0
1009	001574	051032	DF17	;3,0,0,0,0
1010				

1011			;	*ITEM 20		
1012	001576	042256		EM20		; PHYS. ADDR. FORMED READ WRONG IN RELOCATE MODE
1013	001600	045705		DH20		; PHYSICL PAR 4 PAR 5
1014						; ADDRESS VBA VBA PAR 4 PAR 5 PSW TESTNO
1015	001602	050310		DT20		; PBALO, VIRT1, VIRT2, \$REG4, \$REG5, \$TMPO, TESTNO, \$ERRPC, 0
1016	001604	051037		DF20		; 3, 0, 0, 0, 0, 0, 0, 0
1017						
1018				;	*ITEM 21	
1019	001606	042330		EM21		; W-BIT DID NOT GET SET IN PDR
1020	001610	046033		DH21		; PDR VIRTUAL
1021						; TESTED ADDRESS TESTNO ERRORPC
1022	001612	050332		DT21		; \$REG5, \$REG3, TESTNO, \$ERRPC, 0
1023	001614	051047		DF21		; 0, 0, 0, 0
1024						
1025				;	*ITEM 22	
1026	001616	042365		EM22		; W-BIT SET IN MORE THAN ONE PDR
1027	001620	046113		DH22		; PDR IN PDR VIRTUAL
1028						; ERROR TESTED ADDRESS TESTNO ERRORPC
1029	001622	050344		DT22		; \$REG0, \$REG5, \$REG3, TESTNO, \$ERRPC, 0
1030	001624	051053		DF22		; 0, 0, 0, 0, 0
1031						
1032				;	*ITEM 23	
1033	001626	042424		EM23		; W-BIT NOT CLEARED BY WRITING TO PDR
1034	001630	046212		DH23		; PDR TESTNO ERRORPC
1035	001632	050360		DT23		; \$REG5, TESTNO, \$ERRPC, 0
1036	001634	051060		DF23		; 0, 0, 0
1037						
1038				;	*ITEM 24	
1039	001636	042470		EM24		; WRITING SRO SET W-BIT IN KIPDR7
1040	001640	046242		DH24		; PDR WAS EXPECTD TESTNO ERRORPC
1041	001642	050370		DT24		; \$REG2, \$REG1, TESTNO, \$ERRPC, 0
1042	001644	051063		DF24		; 0, 0, 0, 0
1043						
1044				;	*ITEM 25	
1045	001646	042530		EM25		; W-BIT GOT SET DURING ODD ADDR. ABORT
1046	001650	046242		DH24		; PDR WAS EXPECTD TESTNO ERRORPC
1047	001652	050370		DT24		; \$REG2, \$REG1, TESTNO, \$ERRPC, 0
1048	001654	051063		DF24		; 0, 0, 0, 0
1049						
1050				;	*ITEM 26	
1051	001656	042575		EM26		; MEMORY MGMT. ACCESS ABORT DID NOT OCCUR
1052	001660	046302		DH26		; PDR 4 PSW TESTNO ERRORPC
1053	001662	050402		DT26		; \$REG2, \$TMPO, TESTNO, \$ERRPC, 0
1054	001664	051063		DF24		; 0, 0, 0, 0
1055						
1056				;	*ITEM 27	
1057	001666	042645		EM27		; ACCESS ERROR DID NOT ABORT INSTRUCTION
1058	001670	046302		DH26		; PDR 4 PSW TESTNO ERRORPC
1059	001672	050402		DT26		; \$REG2, \$TMPO, TESTNO, \$ERRPC, 0
1060	001674	051063		DF24		; 0, 0, 0, 0
1061						
1062				;	*ITEM 30	
1063	001676	042714		EM30		; SRO DID NOT REPORT ACCESS ERROR CORRECTLY
1064	001700	046342		DH30		; SRO WAS EXPECTD PDR 4 PSW TESTNO ERRORPC
1065	001702	050414		DT30		; WASSRO, \$REG3, \$REG2, \$TMPO, TESTNO, \$ERRPC, 0
1066	001704	051067		DF30		; 0, 0, 0, 0, 0, 0

1067					
1068					
1069	001706	042766			
1070	001710	046422			
1071	001712	050432			
1072	001714	051067			
1073					
1074					
1075	001716	043033			
1076	001720	046502			
1077	001722	050450			
1078	001724	051067			
1079					
1080					
1081	001726	043114			
1082	001730	046562			
1083	001732	050466			
1084	001734	051063			
1085					
1086					
1087	001736	043177			
1088	001740	046622			
1089	001742	050500			
1090	001744	051067			
1091					
1092	001746	042766			
1093	001750	046702			
1094	001752	050516			
1095	001754	051067			
1096					
1097					
1098	001756	042766			
1099	001760	046762			
1100	001762	050534			
1101	001764	051063			
1102					
1103					
1104	001766	043255			
1105	001770	047022			
1106					
1107	001772	050546			
1108	001774	051067			
1109					
1110					
1111	001776	043322			
1112	002000	047137			
1113	002002	050564			
1114	002004	051063			
1115					
1116					
1117	002006	043371			
1118	002010	046762			
1119	002012	050534			
1120	002014	051063			
1121					
1122					

;*ITEM 31

EM31
DH31
DT31
DF30;SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR.
;SR2 WAS EXPECTD PDR 4 PSW TESTNO ERRORPC
;WASSR2,\$REG4,\$REG2,\$TMP0,TESTNO,\$ERRPC,0
;0,0,0,0,0,0

;*ITEM 32

EM32
DH32
DT32
DF30;PAGE LGTH. ABORT OCCURRED WHEN IT SHOULDN'T HAVE
;V.B.A. KIPDR4 SR0 WAS SR2 WAS TESTNO ERRORPC
;\$REG0,\$REG4,WASSR0,WASSR2,TESTNO,\$ERRPC,0
;0,0,0,0,0,0

;*ITEM 33

EM33
DH33
DT33
DF24;PAGE LGTH. ABORT DID NOT OCCUR WHEN IT SHOULD HAVE
;V.B.A. KIPDR4 TESTNO ERRORPC
;\$REG0,\$REG4,TESTNO,\$ERRPC,0
;0,0,0,0

;*ITEM 34

EM34
DH34
DT34
DF30;SR0 DID NOT REPORT PAGE LGTH. ABORT CORRECTLY
;V.B.A. KIPDR4 SR0 WAS EXPECTD TESTNO ERRORPC
;\$REG0,\$REG4,WASSR0,\$REG2,TESTNO,\$ERRPC,0
;0,0,0,0,0,0

;*ITEM 35

EM31
DH35
DT35
DF30;SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR.
;V.B.A. KIPDR4 SR2 WAS EXPECTD TESTNO ERRORPC
;\$REG0,\$REG4,WASSR2,\$REG3,TESTNO,\$ERRPC,0
;0,0,0,0,0,0

;*ITEM 36

EM31
DH36
DT36
DF24;SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR.
;SR2 WAS EXPECTD TESTNO ERRORPC
;WASSR2,\$REG1,TESTNO,\$ERRPC,0
;0,0,0,0

;*ITEM 37

EM37
DH37
DT37
DF30;SR0 OR SR2 CHANGED BY A SECOND ABORT
;FIRST ABORT SECOND ABORT
;SR0 WAS SR2 WAS SR0 WAS SR2 WAS TESTNO ERRORPC
;\$TMP0,\$TMP2,WASSR0,WASSR2,TESTNO,\$ERRPC,0
;0,0,0,0,0,0

;*ITEM 40

EM40
DH40
DT40
DF24;SR0 OR SR2 WAS NOT "RESET" BY A RESET
;SR0 WAS SR2 WAS TESTNO ERRORPC
;WASSR0,WASSR2,TESTNO,\$ERRPC,0
;0,0,0,0

;*ITEM 41

EM41
DH36
DT36
DF24;SR2 NOT TRACKING CORRECTLY
;SR2 WAS EXPECTD TESTNO ERRORPC
;WASSR2,\$REG1,TESTNO,\$ERRPC,0
;0,0,0,0

;*ITEM 42

1123	002016	043424	EM42	;DID NOT TRAP THRU KERNEL SPACE
1124	002020	047177	DH42	;PSW WAS R6 WAS TESTNO ERRORPC
1125	002022	050576	DT42	;SREG1, SREG2, TESTNO, SERRPC, 0
1126	002024	051063	DF24	;0,0,0,0
1127				
1128			;*ITEM 43	
1129	002026	043463	EM43	;KT ERROR SERVICED ON ODD ADDR. ERROR
1130	002030	046212	DH23	;PDR TESTNO ERRORPC
1131	002032	050360	DT23	;SREG5, TESTNO, SERRPC, 0
1132	002034	051060	DF23	;0,0,0
1133				
1134			;*ITEM 44	
1135	002036	043530	EM44	;SR0 OR SR2 CHANGED BY ODD ADDR. ERROR
1136	002040	047237	DH44	;EXPECTED RECEIVED
1137				;SR0 SR2 SR0 WAS SR2 WAS TESTNO ERRORPC
1138	002042	050610	DT44	;SREG0, SREG1, WASSR0, WASSR2, TESTNO, SERRPC, 0
1139	002044	051067	DF30	;0,0,0,0,0,0
1140				
1141			;*ITEM 45	
1142	002046	043576	EM45	;ERROR DURING "DOUBLE ERROR" (KT & ODD ADDR.)
1143	002050	047351	DH45	;EXPECTED:
1144				;PSW PC SR0 SR2
1145				;170017 (3\$+4) 020147 (3\$)
1146				;RECEIVED
1147				;PSW PC SR0 SR2 TESTNO ERRORPC
1148	002052	050626	DT45	;SREG1, SREG3, WASSR0, WASSR2, TESTNO, SERRPC, 0
1149	002054	051067	DF30	;0,0,0,0,0,0
1150				
1151			;*ITEM 46	
1152	002056	043653	EM46	;MFPI INSTRUCTION PUSHED WRONG DATA
1153	002060	047546	DH46	;DATA DATA
1154				;EXPECTD RECEIVD TESTNO ERRORPC
1155	002062	050644	DT46	;SREG0, SREG1, TESTNO, SERRPC, 0
1156	002064	051075	DF46	;0,0,0,0
1157				
1158			;*ITEM 47	
1159	002066	043716	EM47	;MTPI INSTRUCTION LOADED WRONG DATA
1160	002070	047546	DH46	;DATA DATA
1161				;EXPECTD RECEIVD TESTNO ERRORPC
1162	002072	050644	DT46	;SREG0, SREG1, TESTNO, SERRPC, 0
1163	002074	051075	DF46	;0,0,0,0
1164				
1165			;*ITEM 50	
1166	002076	043761	EM50	;STACK NOT PUSHED BY MFPI-MTPI
1167	002100	047623	DH50	;TESTNO ERRORPC
1168	002102	050656	DT50	;TESTNO, SERRPC, 0
1169	002104	051101	DF50	;0,0
1170				
1171			;*ITEM 51	
1172	002106	044017	EM51	;KERNEL PAGE ACCESSED INSTEAD OF USER: MFPI-MTPI
1173	002110	047643	DH51	;SR0 WAS SR2 WAS TESTNO ERRORPC
1174	002112	050664	DT51	;WASSR0, WASSR2, TESTNO, SERRPC, 0
1175	002114	051103	DF51	;0,0,0,0
1176				
1177			;*ITEM 52	
1178	002116	044075	EM52	;WRONG PDR'S REFERENCED WHILE IN RELOCATE MODE

1179	002120	047703	DH52	: PHYSICL PAR 4
1180				: ADDRESS V.B.A. PAR 4 SRO WAS SR2 WAS PSW TESTNO
1181	002122	050676	DT52	: PBALO, VIRT1, \$REG4, WASSRO, WASSR2, \$TMPD, TESTNO, \$ERRPC, 0
1182	002124	051107	DF52	: 3, 0, 0, 0, 0, 0, 0, 0
1183			;*ITEM 53	
1184	002126	044153	EM53	: MFPD INSTRUCTION PUSHED WRONG DATA
1185	002130	047546	DH46	: DATA DATA
1186				: EXPECTD RECEIVD TESTNO ERRORPC
1187	002132	050644	DT46	: \$REGD, \$REG1, TESTNO, \$ERRPC, 0
1188	002134	051075	DF46	: 0, 0, 0, 0
1189				
1190			;*ITEM 54	
1191	002136	044216	EM54	: STACK NOT PUSHED BY MFPD-MTPD
1192	002140	047623	DH50	: TESTNO ERRORPC
1193	002142	050656	DT50	: TESTNO, \$ERRPC, 0
1194	002144	051101	DF50	: 0, 0
1195				
1196			;*ITEM 55	
1197	002146	044254	EM55	: PAR OR PDR WAS CHANGED BY A RESET
1198	002150	045205	DH11	: REGISTR READ READ-(BINARY)
1199				: ADDRESS (OCTAL) 5432109876543210 TESTNO ERRORPC
1200	002152	050214	DT11	: \$REGD, \$REG1, \$REG1, TESTNO, \$ERRPC, 0
1201	002154	051006	DF11	: 0, 0, 2, 0, 0
1202				
1203			;*ITEM 56	
1204	002156	044312	EM56	: ILLEGAL MODE 01 NOT ABORTED
1205	002160	047623	DH50	: TESTNO ERRORPC
1206	002162	050656	DT50	: TESTNO, \$ERRPC, 0
1207	002164	051101	DF50	: 0, 0
1208				
1209			;*ITEM 57	
1210	002166	044346	EM57	: SRO DID NOT REPORT ILLEGAL MODE 01 CORRECTLY
1211	002170	050021	DH57	: SRO WAS EXPECTD TESTNO ERRORPC
1212	002172	050720	DT57	: WASSRO, \$REG1, TESTNO, \$ERRPC, 0
1213	002174	051117	DF57	: 0, 0, 0, 0
1214				
1215			;*ITEM 60	
1216	002176	044423	EM60	: PSW CHANGED BY AN RTI IN USER MODE
1217	002200	050061	DH60	: PSW WAS EXPECTD TESTNO ERRORPC
1218	002202	050732	DT60	: \$REG1, \$REG2, TESTNO, \$ERRPC, 0
1219	002204	051123	DF60	: 0, 0, 0, 0
1220				
1221			;*ITEM 61	
1222	002206	044466	EM61	: MAINT MODE (SRO<8>) NOT DISABLED BY A RESET
1223	002210	047623	DH50	: TESTNO ERRORPC
1224	002212	050656	DT50	: TESTNO, \$ERRPC, 0
1225	002214	051101	DF50	: 0, 0
1226				
1227			;*ITEM 62	
1228	002216	044544	EM62	: DATA INCORRECT AFTER A MAINT. MODE WRITE
1229	002220	045015	DH3	: WROTE READ TESTNO ERRORPC
1230	002222	050156	DT3	: \$REGD, \$REG1, TESTNO, \$ERRPC, 0
1231	002224	050772	DF3	: 0, 0, 0, 0
1232				
1233			;*ITEM 63	
1234	002226	044615	EM63	: SOURCE RELOCATED IN MAINT. MODE

1235	002230	044725
1236	002232	050136
1237	002234	050763
1238		
1239		
1240	002236	042766
1241	002240	046762
1242	002242	050744
1243	002244	051063
1244		

	DH2
	DT2
	DF2
;*ITEM 64	
	EM31
	DH36
	DT64
	DF24

```

;OLD PC OLD PSW R6 WAS SR0 SR2 TESTNO ERRORPC
;TRAPPC, TRAPPS, WASR6, WASSR0, WASSR2, TESTNO, $ERRPC, 0
;0,0,0,0,0,0,0

```

```

;SR2 DIDNOT LOCKUP CORRECT VIRTUAL ADDR.
;SR2 WAS EXPECTD TESTNO ERRORPC
;WASSR2, $REG4, TESTNO, $ERRPC, 0
;0,0,0,0

```

1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300

002246 005227
002250 177777
002252 001401
002254 000000

002256 012637 001356
002262 012637 001360
002266 010637 001354
002272 104001
002274 012737 177777 002250
002302 013746 001360
002306 013746 001356
002312 000006

002314 005227
002316 177777
002320 001401
002322 000000

002324 012637 001356
002330 012637 001360
002334 010637 001354
002340 013737 177572 001362
002346 013737 177576 001364
002354 042737 160000 177572
002362 104002
002364 012737 177777 002316
002372 013746 001360
002376 013746 001356

.SBTTL ***** TRAP HANDLING ROUTINES *****

.SBTTL CPU TRAP HANDLER ROUTINE

* THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS THRU
* "ERRVEC" (LOC. 004). IF THIS SUBROUTINE IS ENTERED BY A
* SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A HALT IS
* EXECUTED.

TIMERR: INC (PC)+ ; MAKE FLAG ZERO IF FIRST TIME THRU
TIMFLG: .WORD -1 ; NEGATIVE ONE FOR "HAVE ENTERED" FLAG
BEQ 1\$; BRANCH IF FIRST TIME IN
HALT ; STOP! - I'VE ENTERED THIS ROUTINE
; A SECOND TIME BEFORE I FINISHED
; REPORTING THE FIRST ERROR. THE
; SECOND ENTRY ADDRESS SHOULD BE ON
; THE KERNEL STACK.
1\$: MOV (KSP)+, TRAPPC ; SAVE PC+2 AT TIME OF ABORT
MOV (KSP)+, TRAPPS ; SAVE PS AT TIME OF ABORT
MOV KSP, WASSR6 ; SAVE STACK POINTER VALUE
ERROR 1 ; UNEXPECTED TRAP OR ABORT TO LOC. 4
MOV #-1, TIMFLG ; MAKE FLAG NEGATIVE ONE FOR NEXT TIME
MOV TRAPPS, -(KSP) ; PUT PC & PS OF TRAP ON STACK
MOV TRAPPC, -(KSP)
RTT ; RETURN FROM INTERRUPT OR ABORT

.SBTTL MEMORY MANAGEMENT TRAP HANDLER ROUTINE

* THIS SUBROUTINE WILL HANDLE ALL UNEXPECTED MEMORY MANAGEMENT
* TRAPS AND ABORTS THRU "MMVEC" (LOC. 250). IF THIS SUBROUTINE IS
* ENTERED BY A SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A
* HALT IS EXECUTED.

MGMERR: INC (PC)+ ; MAKE FLAG ZERO IF FIRST TIME THRU
MGMFLG: .WORD -1 ; NEGATIVE ONE FOR "HAVE ENTERED" FLAG
BEQ 1\$; BRANCH IF FIRST TIME IN
HALT ; STOP! - I'VE ENTERED THIS ROUTINE
; A SECOND TIME BEFORE I FINISHED
; REPORTING THE FIRST ERROR. THE
; SECOND ENTRY ADDRESS SHOULD BE ON
; THE KERNEL STACK.
1\$: MOV (KSP)+, TRAPPC ; SAVE PC+2 AT TIME OF ABORT
MOV (KSP)+, TRAPPS ; SAVE PS AT TIME OF ABORT
MOV KSP, WASSR6 ; SAVE STACK POINTER VALUE
MOV SR0, WASSR0 ; SAVE CONTENTS OF KT STATUS REG. 0
MOV SR2, WASSR2 ; SAVE CONTENTS OF KT STATUS REG. 2
BIC #160000, SR0 ; CLEAR ERROR BITS IN STATUS REG 0
ERROR 2 ; UNEXPECTED TRAP OR ABORT TO LOC. 250
MOV #-1, MGMFLG ; MAKE FLAG NEGATIVE ONE FOR NEXT TIME
MOV TRAPPS, -(KSP) ; PUT PC & PS OF TRAP ON STACK
MOV TRAPPC, -(KSP)

C03

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 28
MEMORY MANAGEMENT TRAP HANDLER ROUTINE

1301 002402 000006
1302

RTT

;RETURN FROM INTERRUPT OR ABORT.

```

1303
1304
1305
1306      020000
1307
1308      020000
1309
1310
1311      020000      012706      001100
1312      020004      005026
1313      020006      022706      001140
1314      020012      001374
1315      020014      012706      001100
1316
1317      020020      012737      034470      000020
1318      020026      012737      000340      000022
1319      020034      012737      034750      000030
1320      020042      012737      000340      000032
1321      020050      012737      040732      000034
1322      020056      012737      000340      000036
1323      020064      012737      041020      000024
1324      020072      012737      000340      000026
1325      020100      013737      034236      034230
1326      020106      005037      001212
1327      020112      005037      001214
1328      020116      112737      000001      001115
1329
1330
1331      020124      012737      034454      000014
1332      020132      012737      000340      000016
1333      020140      012737      000002      034454
1334      020146      012737      020174      000010
1335      020154      005046
1336      020156      012746      020164
1337      020162      000006
1338      020164      012737      000006      034454      645:
1339      020172      000402
1340      020174      062706      000010      655:
1341      020200      012737      000012      000010      665:
1342      020206      005037      034462
1343      020212      012737      020212      001106
1344      020220      012737      020220      001110
1345
1346
1347      020226      013746      000004
1348      020232      012737      020266      000004
1349      020240      012737      177570      001140
1350      020246      012737      177570      001142
1351      020254      022777      177777      160656
1352      020262      001012
1353
1354      020264      000403
1355      020266      012716      020274      675:
1356      020272      000002
1357      020274      012737      000176      001140      685:
1358      020302      012737      000174      001142

```

```

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

START:
.SBTTL INITIALIZE THE COMMON TAGS
;;CLEAR THE COMMON TAGS ($CMTAG) AREA
MOV      #CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
CLR      (R6)+          ;;CLEAR MEMORY LOCATION
CMP      #SWR,R6      ;;DONE?
BNE      #-6           ;;LOOP BACK IF NO
MOV      #STACK,SP     ;;SETUP THE STACK POINTER
;;INITIALIZE A FEW VECTORS
MOV      #SCOPE,#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
MOV      #340,#IOTVEC+2 ;;LEVEL 7
MOV      #ERROR,#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
MOV      #340,#EMTVEC+2 ;;LEVEL 7
MOV      #TRAP,#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
MOV      #340,#TRAPVEC+2 ;;LEVEL 7
MOV      #SPWRON,#PWAVEC ;;POWER FAILURE VECTOR
MOV      #340,#PWAVEC+2 ;;LEVEL 7
MOV      #ENDCT,#SEOPCT ;;SETUP END-OF-PROGRAM COUNTER
CLR      $TIMES        ;;INITIALIZE NUMBER OF ITERATIONS
CLR      $ESCAPE       ;;CLEAR THE ESCAPE ON ERROR ADDRESS
MOV      #1,$SERMAX    ;;ALLOW ONE ERROR PER TEST
;;INITIALIZE THE "T-BIT" TRAP VECTOR. THEN LOAD LOCATION "$RTRN", IN
;;THE "END-OF-PASS" ($EOP) ROUTINE, WITH A "RTI" OR "RTT".
MOV      #RTRN,#TBITVEC ;;SET "T" BIT VECTOR TO $RTRN
MOV      #340,#TBITVEC+2 ;;LEVEL 7
MOV      #RTI,$RTRN     ;;SET $RTRN TO A RTI
MOV      #655,#RESVEC  ;;TRY TO DO A RTT
CLR      -(SP)         ;;DUMMY PS
MOV      #645,-(SP)    ;;AND PC
RTT      ;;TRY THE RTT
645:    MOV      #RTT,$RTRN ;;RTT IS LEGAL--SET $RTRN TO A RTT
BR      665
655:    ADD      #10,SP   ;;RTT ILLEGAL--CLEAN OFF THE STACK
665:    MOV      #RESVEC+2,#RESVEC ;;RESTORE TRAP CATCHER
CLR      $TBIT        ;;CLEAR "T" BIT SWITCH
MOV      #,$SLPADR    ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
MOV      #,$SLPERR    ;;SETUP THE ERROR LOOP ADDRESS
;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
;;EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
MOV      #ERRVEC,-(SP) ;;SAVE ERROR VECTOR
MOV      #675,#ERRVEC ;;SET UP ERROR VECTOR
MOV      #DSWR,$SWR   ;;SETUP FOR A HARDWARE SWICH REGISTER
MOV      #DDISP,$DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
CMP      #-1,$SWR     ;;TRY TO REFERENCE HARDWARE SWR
BNE      695         ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
;;AND THE HARDWARE SWR IS NOT = -1
BR      685         ;;BRANCH IF NO TIMEOUT
675:    MOV      #685,(SP) ;;SET UP FOR TRAP RETURN
RTI
685:    MOV      #SWREG,$SWR ;;POINT TO SOFTWARE SWR
MOV      #DISPREG,$DISPLAY

```

```

1359 020310 012637 000004      69$:  MOV      (SP)+, @#ERRVEC  ;;RESTORE ERROR VECTOR
1360
1361 020314 005037 001234      CLR      $PASS                ;;CLEAR PASS COUNT
1362 020320 132737 000200 001247    BITB    #APTSIZE, $ENVM       ;;TEST USER SIZE UNDER APT
1363 020326 001403                BEQ     70$                   ;;YES, USE NON-APT SWITCH
1364 020330 012737 001250 001140    MOV     #$$SWREG, $SWR       ;;NO, USE APT SWITCH REGISTER
1365 020336
1366
1367                                70$:
.SBTTL  TYPE PROGRAM NAME
;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
1368 020336 005227 177777      INC     #-1                   ;;FIRST TIME?
1369 020342 001054                BNE    71$                   ;;BRANCH IF NO
1370 020344 022737 034410 000042    CMP    #SENDAD, @#42        ;;ACT-11?
1371 020352 001450                BEQ    71$                   ;;BRANCH IF YES
1372 020354 104401 020422      TYPE    72$                 ;;TYPE ASCIZ STRING
1373                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
1374 020360 005737 000042      TST    @#42                 ;;ARE WE RUNNING UNDER XXDP/ACT?
1375 020364 001012                BNE    73$                   ;;BRANCH IF YES
1376 020366 123727 001246 000001    CMPB   $ENV, #1             ;;ARE WE RUNNING UNDER APT?
1377 020374 001406                BEQ    73$                   ;;BRANCH IF YES
1378 020376 023727 001140 000176    CMP    $SWR, #SWREG        ;;SOFTWARE SWITCH REG SELECTED?
1379 020404 001005                BNE    74$                   ;;BRANCH IF NO
1380 020406 104407                GTSWR                    ;;GET SOFT-SWR SETTINGS
1381 020410 000403                BR     74$
1382 020412 112737 000001 001134    73$:  MOVB    #1, $AUTOB          ;;SET AUTO-MODE INDICATOR
1383 020420
1384 020420 000425      74$:  BR     71$                 ;;GET OVER THE ASCIZ
1385                                .:72$: .ASCIZ <CRLF>#MD-11-DFKTH-A 11/34 MEMORY MGMT. DIAG.#<CRLF>
1386 020474
1387
1388                                71$:
LOOP:
1389 020474 012706 001100      MOV     #STACK, KSP         ;;INITIALIZE THE STACK POINTER
1390 020500 012737 002246 000004    MOV     #TIMER, ERRVEC     ;;LOAD CPU SERVICE ROUTINE INTO TRAP VECTOR
1391 020506 012737 000340 000006    MOV     #340, ERRVEC+2     ;;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
1392 020514 012737 002314 000250    MOV     #MGERR, MMVEC     ;;LOAD MEMORY MANAGENT ROUTINE INTO VECTOR
1393 020522 012737 000340 000252    MOV     #340, MMVEC+2     ;;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
1394 020530 012700 177777      MOV     #-1, RO            ;;PUT -1 INTO RO TO INITIALIZE FLAGS
1395 020534 010037 002250      MOV     RO, TIMFLG         ;;INITIALIZE CPU ERROR FLAG
1396 020540 010037 002316      MOV     RO, MGMFLG        ;;INITIALIZE MEMORY MANAGEMENT ERROR FLAG
1397 020544 012737 000340 001366    MOV     #340, TBITPS      ;;INITIALIZE LOG THAT HOLDS T-BIT PSW
1398 020552 005037 177572      CLR     SRO                ;;BE SURE MEM. MGMT IS OFF TO START WITH
1399

```

F03

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 31
GET VALUE FOR SOFTWARE SWITCH REGISTER

```

1400
1401
1402
1403
1404
1405
1406
1407
1408
1409 020556 000004
1410 020560 012737 020570 001110
1411 020566 005000
1412 020570 005001
1413 020572 106400
1414 020574 106701
1415 020576 042701 177437
1416 020602 020001
1417 020604 001401
1418 020606 104003
1419
1420
1421
1422 020610 062700 000040
1423 020614 022700 000400
1424 020620 001363
1425 020622 012737 020560 001110
1426
1427
1428
1429
1430
1431
1432
1433 020630 000004
1434 020632 012737 020642 001110
1435 020640 005000
1436 020642 005037 177776
1437 020646 050037 177776
1438 020652 013701 177776
1439 020656 042701 007777
1440 020662 020001
1441 020664 001403
1442 020666 005037 177776
1443 020672 104004
1444
1445
1446
1447 020674 062700 010000
1448 020700 001360
1449 020702 012737 020632 001110
1450 020710 005037 177776
1451
1452
1453
1454
1455

```

```

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

```


G03

MD-11-DFKTH-A POP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 32
T3 BYTE ADDRESSING TEST FOR PSW

```

1456      * AND READS THEM BACK TO BE SURE THEY CAN BE WRITTEN INDEPENDENTLY.
1457      * THIS CHECKS THE PSW PORTION OF THE ADDRESS DETECTION LOGIC.
1458      *
1459      * *****
1460 020714 000004 TST3: SCOPE
1461 020716 012737 020724 001110 15: MOV #25,$LPERR ;SET LOOP ON ERROR POINTER TO 25
1462 020724 005037 177776 25: CLR PSW ;CLEAR THE PSW
1463 020730 012700 000360 MOV #360,R0 ;PUT THE HIGH BYTE DATA INTO R0
1464 020734 110037 177777 MOVB R0,PSW+1 ;WRITE THE HIGH BYTE OF THE PSW
1465 020740 013701 177776 MOV PSW,R1 ;READ BACK THE ENTIRE PSW
1466 020744 042701 007437 BIC #007437,R1 ;MASK OFF THE T & CC BITS
1467 020750 000300 SWAB R0 ;GET DATA WRITTEN IN HIGH BYTE OF R0
1468 020752 020001 CMP R0,R1 ;WAS THE PSW WRITTEN TO CORRECTLY -
1469 020754 001403 BEQ 35 ;BRANCH IF YES
1470 020756 005037 177776 CLR PSW ;CLEAR PSW FOR ERROR REPORT
1471 020762 104005 ERROR 5 ;LOW BYTE EFFECTED BY WRITE TO HIGH BYTE OF PSW
1472 ;FOR TIGHTER SCOPE LOOP
1473 ;REPLACE ERROR CALL WITH
1474 ;"BR 25" = 000760
1475 020764 012737 020772 001110 35: MOV #45,$LPERR ;SET LOOP ON ERROR POINTER TO 45
1476 020772 005037 177776 45: CLR PSW ;CLEAR THE PSW
1477 020776 012700 000340 MOV #340,R0 ;PUT THE LOW BYTE DATA INTO R0
1478 021002 110037 177776 MOVB R0,PSW ;WRITE THE LOW BYTE OF THE PSW
1479 021006 013701 177776 MOV PSW,R1 ;READ BACK THE ENTIRE PSW
1480 021012 042701 007437 BIC #007437,R1 ;MASK OFF THE T&CC BITS
1481 021016 020001 CMP R0,R1 ;WAS PSW WRITTEN TO CORRECTLY
1482 021020 001403 BEQ 55 ;BRANCH IF YES
1483 021022 005037 177776 CLR PSW ;CLEAR PSW FOR ERROR REPORT
1484 021026 104005 ERROR 5 ;HIGH BYTE EFFECTED BY WRITE TO LOW BYTE OF PSW
1485 ;FOR TIGHTER SCOPE LOOP
1486 ;REPLACE ERROR CALL WITH
1487 ;"BR 25" = 000736
1488 021030 012737 020716 001110 55: MOV #15,$LPERR ;RESET LOOP ON ERROR POINTER TO 15
1489
1490 ; *****
1491 ; *TEST 4 TEST AND SETUP OF STACK POINTERS
1492 ; *
1493 ; * THIS TEST SETS THE USER AND KERNEL STACK POINTERS FOR THE
1494 ; * REST OF THE PROGRAM AND MAKES SURE THEY ARE INDEPENDENT OF
1495 ; * EACH OTHER. KERNEL R6 IS SET TO 1100, USER R6 IS SET TO 700, THEN
1496 ; * KERNEL R6 IS READ TO BE SURE ITS STILL 1100.
1497 ; *
1498 ; *****
1499 021036 000004 TST4: SCOPE
1500 021040 005037 177776 CLR PSW ;GO TO KERNEL MODE
1501 021044 012706 001100 MOV #KERSTK,KSP ;SET KERNEL STACK POINTER TO 1100
1502 021050 012737 140000 177776 MOV #140000,PSW ;GO TO USER MODE
1503 021056 012706 000700 MOV #USESTK,USP ;SET USER STACK POINTER TO 700
1504 021062 005037 177776 CLR PSW ;BACK TO KERNEL MODE
1505 021066 022706 001100 CMP #KERSTK,KSP ;IS KERNEL R6 STILL 1100?
1506 021072 001404 BEQ TSTS ;BRANCH IF KERNEL R6 IS OKAY
1507 021074 012700 001100 MOV #KERSTK,R0 ;SAVE DATA WRITTEN FOR ERROR REPORT
1508 021100 010601 MOV KSP,R1 ;SAVE DATA READ AFTER USER R6 WAS WRITTEN
1509 021102 104006 ERROR 6 ;KERNEL R6 CHANGED BY WRITING USER R6
1510 ;FOR TIGHTER SCOPE LOOP
1511 ;REPLACE ERROR CALL WITH

```

H03

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 33
T4 TEST AND SETUP OF STACK POINTERS

;000756

1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567

021104 000004
021106 012737 021150 001110
021114 012737 021206 000004
021122 012700 177572
021126 012701 000033
021132 012737 177777 001370
021140 005037 001372
021144 005037 001374
021150 005710
021152 062700 000002
021156 077104
021160 012737 021106 001110
021166 005737 001374
021172 001401
021174 104010
021176 012737 002246 000004
021204 000414
021206 062706 000004
021212 104007
021214 010002
021216 050237 001372
021222 005102
021224 040237 001370
021230 005237 001374

```
*****
*
* THE NEXT FIVE (5) TESTS WILL TRY TO ADDRESS ALL OF THE
* MEMORY MANAGEMENT REGISTERS (SR0, SR1, SR2, KERNEL & USER PAR/PDR'S).
* EVERY TIME A REGISTER TIMES OUT ITS ADDRESS WILL BE REPORTED.
* AT THE END OF EACH TEST A SUMMARY OF THE ADDRESSES THAT TIMED
* OUT DURING THAT TEST IS GIVEN. THE RESULTS OF "AND-ING" AND "OR-ING"
* THEIR ADDRESSES IS GIVEN TO SHOW WHICH ADDRESS LINES MAY BE
* STUCK AT 0 OR 1. THE PAR/PDR ADDRESS AND KT MUX'S ARE THE
* THINGS BEING CHECKED.
*
*****

*****
* TEST 5 SR0, SR1, SR2 TIMEOUT TEST
*
* THIS TEST ADDRESSES THE MEMORY MANAGEMENT STATUS REGISTERS
* 0, 1, AND 2. STATUS REG. 1 IS NOT USED BUT SHOULD STILL
* RESPOND TO ITS UNIBUS ADDRESS. DATA WILL BE WRITTEN OR READ
* FROM THESE REGISTERS IN LATER TESTS, THIS TEST JUST CHECK
* FOR A RESPONSE.
*
*****
TSTS: SCOPE
1$: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
MOV #5$, 2#4 ;SET TIMEOUT VECTOR TO 5$
MOV #SR0, R0 ;LOAD R0 WITH ADDRESS OF FIRST REG.
MOV #3, R1 ;LOAD R1 WITH THE LOOP COUNT
MOV #-1, ANDADR ;INITIALIZE "AND" OF ADDRS. LOC.
CLR ORADR ;INITIALIZE "OR" OF ADDRS. LOC.
CLR TONUM ;INITIALIZE "TIMEOUTS" COUNTER
2$: TST (R0) ;TRY ADDRESSING A STATUS REGISTER
;IF IT TIMES OUT GO TO 5$
3$: ADD #2, R0 ;PUT NEXT ADDRESS IN R0
SOB R1, 2$ ;LOOP BACK TO 2$ UNTIL ALL TESTED
MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
TST TONUM ;DID ANY OF THE STATUS REG.S TIMEOUT?
BEQ 4$ ;BRANCH IF NO
ERROR 10 ;SUMMARY OF STATUS REG. TIMEOUTS
4$: MOV #TIMERR, 2#4 ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
BR TST6 ;GO TO NEXT TEST

5$: ADD #4, KSP ;CLEAN UP THE STACK
ERROR 7 ;ONE OF THE STATUS REGS. TIMED OUT
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 2$" = 000756
MOV R0, R2 ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
BIS R2, ORADR ;"OR" IT WITH OTHER ADDRS. THAT TIMED OUT
COM R2 ;"AND" IT WITH OTHER ADDRS. THAT TIMED OUT
BIC R2, ANDADR
INC TONUM ;INCREMENT THE TIMEOUT COUNTER
```

```

1568 021234 000746          BR      3$          ;BRANCH BACK TO TEST THE NEXT ADDR.
1569
1570
1571 .....*****
1572 *TEST 6          KERNEL PAR'S TIMEOUT TEST
1573 *
1574 *          THIS TEST ADDRESSES THE EIGHT (8) KERNEL PAGE ADDRESS
1575 *          REGISTERS (KIPAR0-KIPAR7) AND CHECKS THAT SOMETHING
1576 *          RESPONDS TO THEIR ADDRESSES.
1577 .....*****
1578 021236 000004          †ST6:  SCOPE
1579
1580 021240 012737 021302 001110 1$:  MOV      #2$, $LPERR          ;SET LOOP ON ERROR POINTER TO 2$
1581 021246 012737 021340 000004          MOV      #5$, 2#4          ;SET TIMEOUT VECTOR TO 5$
1582 021254 012700 172340          MOV      #KIPAR0, R0      ;LOAD R0 WITH ADDRESS OF FIRST REG.
1583 021260 012701 000010          MOV      #10, R1         ;LOAD R1 WITH LOOP COUNT (8)
1584 021264 012737 177777 001370          MOV      #-1, ANDADR     ;INITIALIZE "AND" OF ADDR. LOC
1585 021272 005037 001372          CLR      ORADR          ;INITIALIZE "OR" OF ADDR. LOC.
1586 021276 005037 001374          CLR      TONUM          ;INITIALIZE "TIMEOUTS" COUNTER
1587 021302 005710          2$:  TST      (R0)        ;TRY ADDRESSING A KIPAR
1588                                     ;IF IT TIMES OUT, WILL GO TO 5$
1589 021304 062700 000002          3$:  ADD      #2, R0        ;PUT NEXT KIPAR ADDRESS IN R0
1590 021310 077104          SOB      R1, 2$         ;LOOP BACK TO 2$ UNTIL ALL TESTED
1591 021312 012737 021240 001110          MOV      #1$, $LPERR     ;RESET LOOP ON ERROR POINTER TO 1$
1592 021320 005737 001374          TST      TONUM          ;DID ANY OF THE KIPARS TIME OUT?
1593 021324 001401          BEQ      4$             ;BRANCH IF NO
1594 021326 104010          ERROR   10            ;SUMMARY OF KIPAR TIMEOUTS
1595 021330 012737 002246 000004 4$:  MOV      #TIMERR, 2#4   ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1596 021336 000414          BR      TST7          ;GO TO NEXT TEST
1597
1598 021340 062706 000004          5$:  ADD      #4, KSP      ;CLEAN UP THE STACK
1599 021344 104007          ERROR   7             ;ONE OF THE KIPARS TIMED OUT
1600                                     ;FOR TIGHTER SCOPE LOOP
1601                                     ;REPLACE ERROR CALL WITH
1602                                     ;"BR 2$" = 000756
1603 021346 010002          MOV      R0, R2         ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
1604 021350 050237 001372          BIS      R2, ORADR     ;"OR" IT WITH OTHER ADDRS. THAT TIMED OUT
1605 021354 005102          COM      R2            ;"AND" IT WITH OTHER ADDRS. THAT TIMED OUT
1606 021356 040237 001370          BIC      R2, ANDADR
1607 021362 005237 001374          INC      TONUM          ;INCREMENT THE TIMEOUT COUNTER
1608 021366 000746          BR      3$            ;BRANCH BACK TO TEST THE NEXT KIPAR
1609
1610 .....*****
1611 *TEST 7          KERNEL PDR'S TIMEOUT TEST
1612 *
1613 *          THIS TEST ADDRESSES THE EIGHT (8) KERNEL PAGE DESCRIPTOR
1614 *          REGISTERS (KIPDR0-KIPDR7) AND CHECKS THAT SOMETHING
1615 *          RESPONDS TO THEIR ADDRESSES.
1616 *
1617 .....*****
1618 021370 000004          †ST7:  SCOPE
1619
1620 021372 012737 021434 001110 1$:  MOV      #2$, $LPERR          ;SET LOOP ON ERROR POINTER TO 2$
1621 021400 012737 021472 000004          MOV      #5$, 2#4          ;SET TIMEOUT VECTOR TO 5$
1622 021406 012700 172300          MOV      #KIPDR0, R0     ;LOAD R0 WITH ADDRESS OF FIRST REG.
1623 021412 012701 000010          MOV      #10, R1         ;LOAD R1 WITH LOOP COUNT (8)

```

```

1624 021416 012737 177777 001370      MOV    #-1,ANDADR      ;INITIALIZE "AND" OF ADDR. LOC
1625 021424 005037 001372      CLR    ORADR          ;INITIALIZE "OR" OF ADDR. LOC.
1626 021430 005037 001374      CLR    TONUM         ;INITIALIZE "TIMEOUTS" COUNTER
1627 021434 005710          TST    (RO)          ;TRY ADDRESSING A KIPDR
1628                                ;IF IT TIMES OUT, WILL GO TO 5$
1629 021436 062700 000002      3$:   ADD    #2,RO      ;PUT NEXT KIPDR ADDRESS IN RO
1630 021442 077104          SOB    R1,2$        ;LOOP BACK TO 2$ UNTIL ALL TESTED
1631 021444 012737 021372 001110      MOV    #1$,SLPERR    ;RESET LOOP ON ERROR POINTER TO 1$
1632 021452 005737 001374          TST    TONUM        ;DID ANY OF THE KIPDRS TIME OUT?
1633 021456 001401          BEQ   4$           ;BRANCH IF NO
1634 021460 104010          ERROR  10          ;SUMMARY OF KIPDR TIMEOUTS
1635 021462 012737 002246 000004      4$:   MOV    #TIMERR,2#4 ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1636 021470 000414          BR    TST10        ;GO TO NEXT TEST
1637
1638 021472 062706 000004      5$:   ADD    #4,KSP      ;CLEAN UP THE STACK
1639 021476 104007          ERROR  7           ;ONE OF THE KIPDRS TIMED OUT
1640                                ;FOR TIGHTER SCOPE LOOP
1641                                ;REPLACE ERROR CALL WITH
1642                                ;"BR 2$" = 000756
1643 021500 010002          MOV    RO,R2        ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
1644 021502 050237 001372      BIS    R2,ORADR     ;"OR" IT WITH OTHER ADDRS. THAT TIMED OUT
1645 021506 005102          COM   R2           ;"AND" IT WITH OTHER ADDRS. THAT TIMED OUT
1646 021510 040237 001370      BIC   R2,ANDADR    ;
1647 021514 005237 001374      INC   TONUM        ;INCREMENT THE TIMEOUT COUNTER
1648 021520 000746          BR    3$          ;BRANCH BACK TO TEST THE NEXT KIPDR
1649
1650                                ;*****
1651                                ;*TEST 10      USER PAR'S TIMEOUT TEST
1652                                ;*
1653                                ;*   THIS TEST ADDRESSES THE EIGHT (8) USER PAGE ADDRESS
1654                                ;*   REGISTERS (UIPAR0-UIPAR7) AND CHECKS THAT SOMETHING
1655                                ;*   RESPONDS TO THEIR ADDRESSES.
1656                                ;*
1657                                ;*****
1658 021522 000004      TST10: SCOPE
1659
1660 021524 012737 021566 001110      1$:   MOV    #2$,SLPERR  ;SET LOOP ON ERROR POINTER TO 2$
1661 021532 012737 021624 000004      MOV    #5$,2#4      ;SET TIMEOUT VECTOR TO 5$
1662 021540 012700 177640          MOV    #UIPAR0,RO   ;LOAD RO WITH ADDRESS OF FIRST REG.
1663 021544 012701 000010          MOV    #10,R1      ;LOAD R1 WITH LOOP COUNT (8)
1664 021550 012737 177777 001370      MOV    #-1,ANDADR   ;INITIALIZE "AND" OF ADDR. LOC
1665 021556 005037 001372      CLR    ORADR        ;INITIALIZE "OR" OF ADDR. LOC.
1666 021562 005037 001374      CLR    TONUM       ;INITIALIZE "TIMEOUTS" COUNTER
1667 021566 005710          TST    (RO)        ;TRY ADDRESSING A UIPAR
1668                                ;IF IT TIMES OUT, WILL GO TO 5$
1669 021570 062700 000002      3$:   ADD    #2,RO      ;PUT NEXT UIPAR ADDRESS IN RO
1670 021574 077104          SOB    R1,2$        ;LOOP BACK TO 2$ UNTIL ALL TESTED
1671 021576 012737 021524 001110      MOV    #1$,SLPERR    ;RESET LOOP ON ERROR POINTER TO 1$
1672 021604 005737 001374          TST    TONUM        ;DID ANY OF THE UIPARS TIME OUT?
1673 021610 001401          BEQ   4$           ;BRANCH IF NO
1674 021612 104010          ERROR  10          ;SUMMARY OF UIPAR TIMEOUTS
1675 021614 012737 002246 000004      4$:   MOV    #TIMERR,2#4 ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1676 021622 000414          BR    TST11        ;GO TO NEXT TEST
1677
1678 021624 062706 000004      5$:   ADD    #4,KSP      ;CLEAN UP THE STACK
1679 021630 104007          ERROR  7           ;ONE OF THE UIPARS TIMED OUT

```

K03

```

1680                                     ;FOR TIGHTER SCOPE LOOP
1681                                     ;REPLACE ERROR CALL WITH
1682                                     ;"BR 2$" = 000756
1683 021632 010002                          MOV    R0,R2
1684 021634 050237 001372                   BIS    R2,ORADR
1685 021640 005102                          COM    R2
1686 021642 040237 001370                   BIC    R2,ANDADR
1687 021646 005237 001374                   INC    TONUM
1688 021652 000746                          BR     3$
1689                                     ;INCREMENT THE TIMEOUT COUNTER
1690                                     ;BRANCH BACK TO TEST THE NEXT UIPAR

```

*TEST 11 USER PDR'S TIMEOUT TEST

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

*ST11: SCOPE

```

1700 021656 012737 021720 001110 1$:   MOV    #2$, $LPERR    ;SET LOOP ON ERROR POINTER TO 2$
1701 021664 012737 021756 000004       MOV    #5$, 2#4      ;SET TIMEOUT VECTOR TO 5$
1702 021672 012700 177600               MOV    #UIPDR0,R0   ;LOAD R0 WITH ADDRESS OF FIRST REG.
1703 021676 012701 000010               MOV    #10,R1       ;LOAD R1 WITH LOOP COUNT (8)
1704 021702 012737 177777 001370       MOV    #-1,ANDADR   ;INITIALIZE "AND" OF ADDR. LOC
1705 021710 005037 001372               CLR    ORADR        ;INITIALIZE "OR" OF ADDR. LOC.
1706 021714 005037 001374               CLR    TONUM        ;INITIALIZE "TIMEOUTS" COUNTER
1707 021720 005710                       2$:   TST    (R0)       ;TRY ADDRESSING A UIPDR
1708                                     ;IF IT TIMES OUT, WILL GO TO 5$
1709 021722 062700 000002 3$:   ADD    #2,R0        ;PUT NEXT UIPDR ADDRESS IN R0
1710 021726 077104                       SOB    R1,2$        ;LOOP BACK TO 2$ UNTIL ALL TESTED
1711 021730 012737 021656 001110       MOV    #1$, $LPERR  ;RESET LOOP ON ERROR POINTER TO 1$
1712 021736 005737 001374               TST    TONUM        ;DID ANY OF THE UIPDRS TIME OUT?
1713 021742 001401                       BEQ    4$           ;BRANCH IF NO
1714 021744 104010                       ERROR  10           ;SUMMARY OF UIPDR TIMEOUTS
1715 021746 012737 002246 000004 4$:   MOV    #TIMERR,2#4  ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1716 021754 000414                       BR     TST1?       ;GO TO NEXT TEST

```

```

1718 021756 062706 000004 5$:   ADD    #4,KSP       ;CLEAN UP THE STACK
1719 021762 104007                       ERROR  7           ;ONE OF THE UIPDRS TIMED OUT
1720                                     ;FOR TIGHTER SCOPE LOOP
1721                                     ;REPLACE ERROR CALL WITH
1722                                     ;"BR 2$" = 000756
1723 021764 010002                          MOV    R0,R2
1724 021766 050237 001372                   BIS    R2,ORADR
1725 021772 005102                          COM    R2
1726 021774 040237 001370                   BIC    R2,ANDADR
1727 022000 005237 001374                   INC    TONUM
1728 022004 000746                          BR     3$
1729                                     ;INCREMENT THE TIMEOUT COUNTER
1730                                     ;BRANCH BACK TO TEST THE NEXT UIPDR

```

*TEST 12 SRO(15:13) BIT TEST & SR2 TEST

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

1731
1732
1733
1734
1735

L03

MD-11-DFKTH-A POP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 37
T12 SRO(15:13) BIT TEST & SR2 TEST

1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791

022006 000004
022010 012700 177572
022014 012710 160000
022020 000005
022022 011001
022024 001404
022026 104011

022030 012737 022036 001110
022036 013737 177576 001364
022044 012701 022036
022050 020137 001364
022054 001401
022056 104041

022060 012737 022076 001110
022066 012701 100000
022072 012703 000003
022076 005010
022100 050110
022102 011002
022104 020102
022106 001401
022110 104012

022112 012704 022100
022116 013737 177576 001364
022124 020437 001364

022130 001401
022132 104064

022134 006001
022136 077321
022140 005010
022142 012737 022010 001110

```

;* PART OF SRO, THE SRO MUX AND THE KTMUX. THE REST OF THE
;* BITS IN SRO WILL BE CHECKED LATER.
;* ALSO CHECK THAT SR2 IS TRACKING WITH MEM. MGMT.
;* OFF BUT LOCKS UP WHEN ANY OF SRO ERROR BITS SET.
;*****
TST12: SCOPE
15:  MOV    #SRO,R0      ;LOAD ADDRESS OF SRO INTO R0
      MOV    #160000,(R0) ;SET BITS <15:13> IN SRO (ERROR BITS)
      RESET  ;ISSUE AND "INIT" SIGNAL
      MOV    (R0),R1     ;READ SRO INTO R1 TO SEE IF CLEAR
      BEQ   25          ;BRANCH IF SRO<15:13> CLEARED BY "INIT"
      ERROR  11         ;SRO<15:13> NOT CLEARED BY A "RESET"
                          ;FOR TIGHTER SCOPE LOOP
                          ;REPLACE ERROR CALL WITH
                          ;"BR 15" = 000770
                          ;SET LOOP ON ERROR POINTER TO 25
25:  MOV    #25,$LPERR   ;READ CONTENTS OF SR2
      MOV    SR2,WASSR2 ;LOAD EXPECTED CONTENTS INTO R1
      MOV    #25,R1     ;IS SR2 TRACKING?
      CMP   R1,WASSR2  ;BRANCH IF YES
      BEQ   35          ;SR2 NOT "TRACKING" VIRTUAL ADDRESSES
      ERROR  41         ;FOR TIGHTER SCOPE LOOP
                          ;REPLACE ERROR CALL WITH
                          ;"BR 25" = 000767
                          ;SET LOOP ON ERROR POINTER TO 45
35:  MOV    #45,$LPERR   ;PUT DATA TO BE WRITTEN IN R1
      MOV    #BIT15,R1 ;SETUP R3 AS A LOOP COUNTER
      MOV    #3,R3     ;CLEAR SRO
      CLR   (R0)       ;SET ONE OF THE ERROR BITS IN SRO
      MOV   (R0),R2    ;READ SRO INTO R2
      BIS   R1,(R0)    ;DID RIGHT ERROR BIT GET SET?
      MOV   (R0),R2    ;BRANCH IF YES
      CMP   R1,R2     ;BITS WERE SET WRONG IN SRO
      BEQ   65          ;FOR TIGHTER SCOPE LOOP
      ERROR  12         ;REPLACE ERROR CALL WITH
                          ;"BR 45" = 000772
                          ;LOAD EXPECTED CONTENTS OF SR2 IN R4
65:  MOV    #55,R4       ;READ SR2
      MOV    SR2,WASSR2 ;DID SR2 LOCK UP WHEN ERROR
      CMP   R4,WASSR2  ;BIT SET IN SR1?
      BEQ   75          ;BRANCH IF YES
      ERROR  64         ;SR2 DID NOT LOCK UP
                          ;FOR TIGHTER SCOPE LOOP
                          ;REPLACE ERROR CALL WITH
                          ;"BR 45" = 000761
                          ;CHANGE DATA TO CHECK NEXT ERROR BIT
75:  ROR   R1           ;LOOP BACK UNTIL <15:13> ALL TESTED
      SOB  R3,45       ;CLEAR SRO BEFORE LEAVING
      CLR  (R0)        ;RESET LOOP ON ERROR POINTER TO 15
      MOV  #15,$LPERR
;*****

```

```

;*****
;TEST 13 SRO & PSW DUAL ADDRESSING TEST
;*****
;* THIS TEST CHECKS MORE OF THE ADDRESS DETECTION LOGIC BY

```

M03

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 38
T13 SRO & PSW DUAL ADDRESSING TEST

```

1792
1793
1794
1795
1796
1797
1798 022150 000004
1799
1800 022152 005037 177776
1801 022156 005037 177572
1802 022162 106427 000340
1803 022166 013700 177572
1804 022172 001401
1805 022174 104013
1806
1807
1808
1809 022176 005037 177572
1810 022202 005037 177776
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820 022206 000004
1821 022210 012700 177777
1822 022214 013700 177574
1823 022220 001401
1824 022222 104014
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839 022224 000004
1840
1841 022226 012700 172340
1842 022232 012703 000010
1843 022236 012737 022244 001110
1844 022244 005010
1845 022246 011001
1846 022250 001401
1847 022252 104011

```

```

;*
;* VERIFYING THAT STATUS REGISTER 0 IS NOT EFFECTED BY WRITING
;* TO THE PSW AND THAT THE LOW BYTE OF STATUS REGISTER 0
;* IS NOT EFFECTED BY WRITING TO ITS HIGH BYTE. THIS IS TO
;* SEE IF ADJACENT OUTPUTS ARE SHORTED ON THE ADDRESS DET. LOGIC.
;*

```

```

*****
TST13: SCOPE

```

```

15: CLR PSW ;CLEAR THE PSW
CLR SRO ;CLEAR STATUS REGISTER 0
MTPS #340 ;SET PRIORITY 7 IN LOW BYTE OF PSW
MOV SRO,RO ;READ STATUS REGISTER 0
BEQ 25 ;BRANCH IF IT WAS STILL 0
ERROR 13 ;SRO EFFECTED BY A WRITE TO THE PSW
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 15" = 000767
25: CLR SRO ;BE SURE SRO IS 0 BEFORE LEAVING
CLR PSW ;BE SURE PSW IS 0 BEFORE LEAVING

```

```

*****
TEST 14 TEST THAT SRI READS ALL ZEROS

```

```

;*
;* THIS TESTS CHECKS THAT EVEN THOUGH STATUS REGISTER 1
;* IS NON-EXISTENT, ITS ADDRESS SHOULD RESPOND WITH ALL ZEROS,
;* THEREBY CHECK ANOTHER PORTION OF THE ADDRESS DETECTION LOGIC.
;*

```

```

*****
TST14: SCOPE

```

```

MOV #-1,RO ;FILL RO WITH ALL ONES
MOV SRI,RO ;READ SRI INTO RO
BEQ TST15 ;BRANCH IF SRI READS ALL ZEROS
ERROR 14 ;SRI DID NOT READ ALL ZEROS
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;000772

```

```

*****
TEST 15 BIT TEST OF KERNEL & USER PAR'S

```

```

;*
;* THE FOLLOWING TEST CHECKS THE BITS <11:00> OF BOTH THE KERNEL
;* AND USER PAGE ADDRESS REGISTERS. A "0" IS ROTATED THRU
;* THE REGISTERS FROM LEFT TO RIGHT. THIS CHECKS THE OPERATION
;* OF THE PAR/PDR ADDRESS MUX, THE KT MUX, AND THE PAR
;* OUTPUT DATA LINES.

```

```

*****
TST15: SCOPE

```

```

15: MOV #KIPAR0,RO ;LOAD ADDRESS OF FIRST PAR IN RO
25: MOV #10,R3 ;SETUP R3 TO COUNT 8 PAR'S
35: MOV #35,SLPERR ;SET LOOP ON ERROR POINTER TO 35
CLR (RO) ;CLEAR THE PAR
MOV (RO),R1 ;READ THE PAR INTO R1
BEQ 45 ;BRANCH IF PAR CLEARED OK
ERROR 11 ;PAR WOULD NOT CLEAR

```

NO3

MD-11-DFKTH-A POP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 39
T15 BIT TEST OF KERNEL & USER PAR'S

1848								: FOR TIGHTER SCOPE LOOP
1849								: REPLACE ERROR CALL WITH
1850								: "BR 3\$" = 000774
1851	022254	012704	167777		4\$:	MOV	#167777, R4	: LOAD "WALKING 0" TEST PATTERN IN R4
1852	022260	012737	022266	001110		MOV	#5\$, SLPERR	: SET LOOP ON ERROR POINTER TO 5\$
1853	022266	005010			5\$:	CLR	(R0)	: CLEAR THE PAR BEFORE LOADING DATA
1854	022270	010401				MOV	R4, R1	: LOAD DATA INTO R1
1855	022272	042701	170000			BIC	#170000, R1	: MASK UNUSED BITS OUT OF THE DATA
1856	022276	050110				BIS	R1, (R0)	: BIT SET THE TEST PATTERN INTO THE PAR
1857	022300	011002				MOV	(R0), R2	: READ THE PAR INTO R2
1858	022302	020102				CMP	R1, R2	: DOES DATA WRITTEN=DATA READ?
1859	022304	001401				BEQ	6\$: BRANCH IF YES
1860	022306	104012				ERROR	12	: PAR BITS DID NOT SET CORRECTLY
1861								: FOR TIGHTER SCOPE LOOP
1862								: REPLACE ERROR CALL WITH
1863								: "BR 5\$" = 000767
1864	022310	000261			6\$:	SEC		: SET THE C-BIT FOR THE ROTATE INST.
1865	022312	006004				ROR	R4	: ROTATE THE TEST PATTERN IN R4
1866	022314	103764				BCS	5\$: BRANCH BACK IF MORE BITS TO TEST
1867	022316	062700	000002			ADD	#2, R0	: GET NEXT PAR ADDRESS IN R0
1868	022322	077330				SUB	R3, 3\$: BRANCH BACK UNTIL ALL PAR'S TESTED
1669	022324	022700	177660			CMP	#UIPAR7+2, R0	: HAVE USER PAR'S BEEN TESTED
1870	022330	103003				BHIS	7\$: BRANCH IF YES
1871	022332	012700	177640			MOV	#UIPAR0, R0	: LOAD FIRST USER PAR ADDR. IN R0
1872	022336	000735				BR	2\$: BRANCH BACK TO TEST USER PAR'S
1873	022340	012737	022226	001110	7\$:	MOV	#1\$, SLPERR	: RESET LOOP OR ERROR POINTER TO 1\$
1874								: LEAVE TEST WITH BITS <11:1>=1 IN ALL PAR'S

◆
↓
↓
◆


```

1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886 022346 000004
1887
1888 022350 012700 172300
1889 022354 012703 000010
1890 022360 012737 022366 001110
1891 022366 005010
1892 022370 011001
1893 022372 001401
1894 022374 104011
1895
1896
1897
1898 022376 012704 077777
1899 022402 012737 022410 001110
1900 022410 005010
1901 022412 010401
1902 022414 042701 100361
1903 022420 050110
1904 022422 011002
1905 022424 020102
1906 022426 001401
1907 022430 104012
1908
1909
1910
1911 022432 000261
1912 022434 006004
1913 022436 103764
1914 022440 062700 000002
1915 022444 077330
1916 022446 022700 177620
1917 022452 103003
1918 022454 012700 177600
1919 022460 000735
1920 022462 012737 022350 001110
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930

```

```

*****
*TEST 16 BIT TEST OF KERNEL & USER PDR'S
*
* THE FOLLOWING TEST CHECKS THE BITS <14:8> AND <3:1> OF BOTH THE
* KERNEL AND USER PAGE DESCRIPTOR REGISTERS. A "0" IS ROTATED
* THRU THE REGISTERS FROM LEFT TO RIGHT. SOME TEST PATTERNS WILL
* BE LOADED MORE THAN ONCE DUE TO THE UNUSED BITS IN THE PDR'S.
* THE PAR/PDR ADDRESS MUX, KTMUX, AND PDR OUTPUT DATA LINES
* ARE BEING CHECKED.
*****

```

```

*****
*ST16: SCOPE
*****
1$: MOV #KIPDR0,R0 ;LOAD ADDRESS OF FIRST PDR IN R0
2$: MOV #10,R3 ;SETUP R3 TO COUNT 8 PDR'S
3$: MOV #3$,SLPERR ;SET LOOP ON ERROR POINTER TO 3$
;CLR (R0) ;CLEAR THE PDR
;MOV (R0),R1 ;READ THE PDR INTO R1
;BEQ 4$ ;BRANCH IF PDR CLEARED OK
;ERROR 11 ;PDR WOULD NOT CLEAR
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 3$" = 000774
4$: MOV #077777,R4 ;LOAD "WALKING 0" TEST PATTERN IN R4
;MOV #5$,SLPERR ;SET LOOP ON ERROR POINTER TO 5$
5$: CLR (R0) ;CLEAR THE PDR BEFORE LOADING DATA
;MOV R4,R1 ;LOAD DATA INTO R1
;BIC #100361,R1 ;MASK UNUSED BITS OUT OF THE DATA
;BIS R1,(R0) ;BIT SET THE TEST PATTERN INTO THE PDR
;MOV (R0),R2 ;READ THE PDR INTO R2
;CMP R1,R2 ;DOES DATA WRITTEN=DATA READ?
;BEQ 6$ ;BRANCH IF YES
;ERROR 12 ;PDR BITS DID NOT SET CORRECTLY
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 5$" = 000767
6$: SEC ;SET THE C-BIT FOR THE ROTATE INST.
;ROR R4 ;ROTATE THE TEST PATTERN IN R4
;BCS 5$ ;BRANCH BACK IF MORE BITS TO TEST
;ADD #2,R0 ;GET NEXT PDR ADDRESS IN R0
;SOB R3,3$ ;BRANCH BACK UNTIL ALL PDR'S TESTED
;CMP #UIPDR7+2,R0 ;HAVE USER PDR'S BEEN TESTED?
;BHS 7$ ;BRANCH IF YES
;MOV #UIPDR0,R0 ;LOAD FIRST USER PDR ADDR. IN R0
;BR 2$ ;BRANCH BACK TO TEST USER PDR'S
7$: MOV #1$,SLPERR ;RESET LOOP ON ERROR POINTER TO 1$
;LEAVE TEST WITH ALL WRITEABLE BITS IN
;ALL PDR'S = 1

```

```

*****
*TEST 17 TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PAR'S
*
* THE FOLLOWING TEST WRITES TO BOTH BYTES OF THE KERNEL & USER
* PAR'S SEPERATELY TO SEE THAT WRITING TO ONE DOES NOT EFFECT
* THE OTHER. THIS FURTHER VERIFIES THE OPERATION OF THE PAR/PDR
* ADDR. MUX AND THE ADDR. DETECTION LOGIC.
*****

```

C04

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16.02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 41
T17 TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PAR'S

1931
1932
1933 022470 000004

;*
:*****
↑ST17: SCOPE

|
|
|
|
|
|

MO-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 42
T17 TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PAR'S

1934									
1935	022472	012700	172340		1\$:	MOV	#KIPAR0,R0		;LOAD ADDRESS OF FIRST PAR INTO R0
1936	022476	012737	022510	001110	2\$:	MOV	#3\$,SLPERR		;SET LOOP ON ERROR POINTER TO 3\$
1937	022504	012703	000010			MOV	#10,R3		;LOAD LOOP COUNTER TO DO 8 PAR'S
1938	022510	012701	177777		3\$:	MOV	#-1,R1		;LOAD TEST PATTERN INTO R1
1939	022514	005010				CLR	(R0)		;CLEAR THE PAR

E04

MD-11-DFKTH-A POP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 43
T17 TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PAR'S

1940	022516	110110				MOVB	R1,(R0)	:WRITE 1'S TO THE LOW BYTE OF THE PAR
1941	022520	0'1002				MOV	(R0),R2	:READ THE ENTIRE PAR INTO R2
1942	022522	042701	177400			BIC	#177400,R1	:MASK HIGH BYTE & UNUSED BITS OUT OF THE DATA
1943	022526	020102				CMP	R1,R2	:WAS ONLY THE LOW BYTE WRITTEN TO
1944	022530	001401				BEQ	4\$:BRANCH IF YES
1945	022532	104015				ERROR	15	:HIGH BYTE EFFECTED BY WRITING LOW BYTE IN PAR
1946								:FOR TIGHTER SCOPE LOOP
1947								:REPLACE ERROR CALL WITH
1948								"BR 3\$" = 000766
1949	022534	012737	022542	001110	4\$:	MOV	#5\$,SLPERR	:SET LOOP ON ERROR POINTER TO 5\$
1950	022542	005010			5\$:	CLR	(R0)	:CLEAR THE PAR
1951	022544	012701	177777			MOV	#-1,R1	:LOAD TEST, PATTERN INTO R1
1952	022550	110160	000001			MOVB	R1,1(R0)	:WRITE 1'S TO THE HIGH BYTE OF THE PAR
1953	022554	011002				MOV	(R0),R2	:READ THE ENTIRE PAR INTO R2
1954	022556	042701	170377			BIC	#170377,R1	:MASK LOW BYTE & UNUSED BITS OUT OF DATA
1955	022562	020102				CMP	R1,R2	:WAS ONLY THE HIGH BYTE WRITTEN TO?
1956	022564	001401				BEQ	6\$:BRANCH IF YES
1957	022566	104015				ERROR	15	:LOW BYTE EFFECTED BY WRITING HIGH BYTE IN PAR
1958								:FOR TIGHTER SCOPE LOOP
1959								:REPLACE ERROR CALL WITH
1960								"BR 5" = 000765
1961	022570	062700	000002		6\$:	ADD	#2,R0	:PUT ADDRESS OF NEXT PAR IN R0
1962	022574	077333				SOB	R3,3\$:BRANCH BACK UNTIL 8 PAR'S TESTED
1963	022576	022700	177660			CMP	#UIPAR7+2,R0	:HAVE USER PAR'S BEEN TESTED
1964	022602	103003				BHIS	7\$:BRANCH IF YES
1965	022604	012700	177640			MOV	#UIPAR0,R0	:LOAD ADDRESS OF FIRST USER PAR IN R0
1966	022610	000732				BR	2\$:BRANCH BACK TO TEST USER PAR'S
1967	022612	012737	022472	001110	7\$:	MOV	#1\$,SLPERR	:RESET LOOP ON ERROR POINTER TO 1\$
1968								

F04

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 44
T20 TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PDR'S

1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016

022620 000004
022622 012700 172300
022626 012737 022640 001110
022634 012703 000010
022640 012701 177777
022644 005010
022646 110110
022650 011002
022652 042701 177761
022656 020102
022660 001401
022662 104015

022664 012737 022672 001110
022672 005010
022674 012701 177777
022700 110160 000001
022704 011002
022706 042701 100377
022712 020102
022714 001401
022716 104015

022720 062700 000002
022724 077333
022726 022700 177620
022732 103003
022734 012700 177600
022740 000732
022742 012737 022622 001110

```

*****
*TEST 20      TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PDR'S
*
*      THE FOLLOWING TEST WRITES TO BOTH BYTES OF THE KERNEL & USER
*
*      PDR'S SEPERATELY TO SEE THAT WRITING TO ONE DOES NOT EFFECT
*      THE OTHER.  THIS FURTHER VERIFIES THE OPERATION OF THE PAR/PDR
*      ADDR. MUX AND THE ADDR. DETECTION LOGIC.
*****
↑ST20: SCOPE
1$:  MOV      #KIFDR, R0      ; LOAD ADDRESS OF FIRST PDR INTO R0
2$:  MOV      #3$, $LPERR    ; SET LOOP ON ERROR POINTER TO 3$
    MOV      #10, R3         ; LOAD LOOP COUNTER TO DO 8 PDR'S
3$:  MOV      #-1, R1        ; LOAD TEST PATTERN INTO R1
    CLR      (R0)           ; CLEAR THE PDR
    MOVB     R1, (R0)        ; WRITE 1'S TO THE LOW BYTE OF THE PDR
    MOV      (R0), R2       ; READ THE ENTIRE PDR INTO R2
    BIC      #177761, R1    ; MASK HIGH BYTE & UNUSED BITS OUT OF DATA
    CMP      R1, R2         ; WAS ONLY THE LOW BYTE WRITTEN TO?
    BEQ      4$             ; BRANCH IF YES
    ERROR    15            ; HIGH BYTE EFFECTED BY WRITING LOW BYTE IN PDR
                           ; FOR TIGHTER SCOPE LOOP
                           ; REPLACE ERROR CALL WITH
                           ; "BR 3$" = 000766
4$:  MOV      #5$, $LPERR    ; SET LOOP ON ERROR POINTER TO 5$
    CLR      (R0)           ; CLEAR THE PDR
    MOV      #-1, R1        ; LOAD TEST PATTERN INTO R1
    MOVB     R1, (R0)        ; WRITE 1'S TO THE HIGH BYTE OF THE PDR
    MOV      (R0), R2       ; READ THE ENTIRE PDR INTO R2
    BIC      #100377, R1    ; MASK LOW BYTE & UNUSED BITS OUT OF DATA
    CMP      R1, R2         ; WAS ONLY THE HIGH BYTE WRITTEN TO?
    BEQ      6$             ; BRANCH IF YES
    ERROR    15            ; LOW BYTE EFFECTED BY WRITING HIGH BYTE IN PDR
                           ; FOR TIGHTER SCOPE LOOP
                           ; REPLACE ERROR CALL WITH
                           ; "BR 5$" = 000765
6$:  ADD      #2, R0         ; PUT ADDRESS OF NEXT PDR IN R0
    SOB      R3, 3$         ; BRANCH BACK UNTIL 8 PDR'S TESTED
    CMP      #UIPDR7+2, R0  ; HAVE USER PDR'S BEEN TESTED?
    BHS     7$             ; BRANCH IF YES
    MOV      #UIPDR0, R0    ; LOAD ADDRESS OF FIRST USER PDR IN R0
    BR      2$             ; BRANCH BACK TO TEST USER PDR'S
7$:  MOV      #1$, $LPERR    ; RESET LOOP ON ERROR POINTER TO 1$

```

```

2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034 022750 000004
2035
2036 022752 012737 022770 001110 1$: MOV #2$, $LPERR ; SET LOOP ON ERROR POINTER 2$
2037 022760 012703 000010 MOV #10, R3 ; LOAD LOOP COUNTER WITH AN 8
2038 022764 012700 172300 MOV #KIPDR0, R0 ; LOAD ADDRESS OF FIRST KERNEL PDR AND R0
2039 022770 012706 001100 2$: MOV #KERSTK, KSP ; SETUP STACK POINTER
2040 022774 004737 035520 JSR PC, SETREG ; SET ALL BITS IN ALL PAR'S IN PDR'S
2041 023000 005010 CLR (R0) ; CLEAR ONE OF THE KERNEL PDR'S
2042 023002 004737 035612 JSR PC, CMPREG ; SEE IF OTHER PAR/PDR'S WERE EFFECTED
2043 023006 062700 000002 ADD #2, R0 ; FORM ADDRESS OF NEXT KERNEL PDR TO CLEAR
2044 023012 077312 SOB R3, 2$ ; LOOP TO 2$ UNTIL ALL KERNEL PDR'S CHECKED
2045 023014 012737 023032 001110 MOV #3$, $LPERR ; SET LOOP ON ERROR POINTER TO 3$
2046 023022 012703 000010 MOV #10, R3 ; LOAD LOOP COUNTER WITH AN 8
2047 023026 012700 172340 MOV #KIPAR0, R0 ; LOAD ADDRESS OF FIRST KERNEL PAR IN R0
2048 023032 012706 001100 3$: MOV #KERSTK, KSP ; SETUP STACK POINTER
2049 023036 004737 035520 JSR PC, SETREG ; SET ALL BITS IN ALL PAR'S AND PDR'S
2050 023042 005010 CLR (R0) ; CLEAR ONE OF THE KERNEL PAR'S
2051 023044 004737 035612 JSR PC, CMPREG ; SEE IF OTHER PAR/PDR'S WERE EFFECTED
2052 023050 062700 000002 ADD #2, R0 ; FORM ADDRESS OF NEXT KERNEL PAR TO CLEAR
2053 023054 077312 SOB R3, 3$ ; LOOP TO 3$ UNTIL ALL KERNEL PAR'S CHECKED
2054 023056 012737 023074 001110 MOV #4$, $LPERR ; SET LOOP ON ERROR POINTER TO 4$
2055 023064 012703 000010 MOV #10, R3 ; LOAD LOOP COUNTER WITH AN 8
2056 023070 012700 177600 MOV #UIPDR0, R0 ; LOAD ADDRESS OF FIRST USER PDR IN R0
2057 023074 012706 001100 4$: MOV #KERSTK, KSP ; SETUP STACK POINTER
2058 023100 004737 035520 JSR PC, SETREG ; SET ALL BITS IN ALL PAR'S AND PDR'S
2059 023104 005010 CLR (R0) ; CLEAR ONE OF THE USER PDR'S
2060 023106 004737 035612 JSR PC, CMPREG ; SEE IF OTHER PAR/PDR'S WERE EFFECTED
2061 023112 062700 000002 ADD #2, R0 ; FORM ADDRESS OF NEXT USER PDR TO CLEAR
2062 023116 077312 SOB R3, 4$ ; LOOP TO 4$ UNTIL ALL USER PDR'S CHECKED
2063 023120 012737 023136 001110 MOV #5$, $LPERR ; SET LOOP ON ERROR POINTER TO 5$
2064 023126 012703 000010 MOV #10, R3 ; LOAD LOOP COUNTER WITH AN 8
2065 023132 012700 177640 MOV #UIPAR0, R0 ; LOAD ADDRESS OF FIRST USER PAR IN R0
2066 023136 012706 001100 5$: MOV #KERSTK, KSP ; SETUP STACK POINTER
2067 023142 004737 035520 JSR PC, SETREG ; SET ALL BITS IN ALL PAR'S AND PDR'S
2068 023146 005010 CLR (R0) ; CLEAR ONE OF THE USER PAR'S
2069 023150 004737 035612 JSR PC, CMPREG ; SEE IF OTHER PAR/PDR'S WERE EFFECTED

```

*TEST 21 PAR-PDR DUAL ADDRESSING TEST

* THE FOLLOWING TEST SETS ALL OF THE WRITEABLE BITS TO 1
* IN THE SIXTEEN (16) PAR'S AND PDR'S USING THE "SETREG"
* SUBROUTINE AND THEN CLEARS JUST ONE OF THEM. THE "CMPREG"
* SUBROUTINE IS USED TO READ ALL OF THE PAR'S AND PDR'S TO SEE
* THAT ONLY ONE REGISTER WAS CLEARED IN RESPONSE TO THAT ONE
* PAR OR PDR ADDRESS. THE "CMPREG" SUBROUTINE REPORTS THE
* ADDRESS OF ANY REGISTER WHOSE BITS DID NOT REMAIN SET WHEN
* ANOTHER REGISTER WAS CLEARED.

* THE PAR AND PDR CHIPS, PAR/PDR ADDR. MUX, AND ADDR. DETECTION
* LOGIC ARE CHECKED.

*TST21: SCOPE

1\$: MOV #2\$, \$LPERR ; SET LOOP ON ERROR POINTER 2\$
MOV #10, R3 ; LOAD LOOP COUNTER WITH AN 8
MOV #KIPDR0, R0 ; LOAD ADDRESS OF FIRST KERNEL PDR AND R0
2\$: MOV #KERSTK, KSP ; SETUP STACK POINTER
JSR PC, SETREG ; SET ALL BITS IN ALL PAR'S IN PDR'S
CLR (R0) ; CLEAR ONE OF THE KERNEL PDR'S
JSR PC, CMPREG ; SEE IF OTHER PAR/PDR'S WERE EFFECTED
ADD #2, R0 ; FORM ADDRESS OF NEXT KERNEL PDR TO CLEAR
SOB R3, 2\$; LOOP TO 2\$ UNTIL ALL KERNEL PDR'S CHECKED
MOV #3\$, \$LPERR ; SET LOOP ON ERROR POINTER TO 3\$
MOV #10, R3 ; LOAD LOOP COUNTER WITH AN 8
MOV #KIPAR0, R0 ; LOAD ADDRESS OF FIRST KERNEL PAR IN R0
3\$: MOV #KERSTK, KSP ; SETUP STACK POINTER
JSR PC, SETREG ; SET ALL BITS IN ALL PAR'S AND PDR'S
CLR (R0) ; CLEAR ONE OF THE KERNEL PAR'S
JSR PC, CMPREG ; SEE IF OTHER PAR/PDR'S WERE EFFECTED
ADD #2, R0 ; FORM ADDRESS OF NEXT KERNEL PAR TO CLEAR
SOB R3, 3\$; LOOP TO 3\$ UNTIL ALL KERNEL PAR'S CHECKED
MOV #4\$, \$LPERR ; SET LOOP ON ERROR POINTER TO 4\$
MOV #10, R3 ; LOAD LOOP COUNTER WITH AN 8
MOV #UIPDR0, R0 ; LOAD ADDRESS OF FIRST USER PDR IN R0
4\$: MOV #KERSTK, KSP ; SETUP STACK POINTER
JSR PC, SETREG ; SET ALL BITS IN ALL PAR'S AND PDR'S
CLR (R0) ; CLEAR ONE OF THE USER PDR'S
JSR PC, CMPREG ; SEE IF OTHER PAR/PDR'S WERE EFFECTED
ADD #2, R0 ; FORM ADDRESS OF NEXT USER PDR TO CLEAR
SOB R3, 4\$; LOOP TO 4\$ UNTIL ALL USER PDR'S CHECKED
MOV #5\$, \$LPERR ; SET LOOP ON ERROR POINTER TO 5\$
MOV #10, R3 ; LOAD LOOP COUNTER WITH AN 8
MOV #UIPAR0, R0 ; LOAD ADDRESS OF FIRST USER PAR IN R0
5\$: MOV #KERSTK, KSP ; SETUP STACK POINTER
JSR PC, SETREG ; SET ALL BITS IN ALL PAR'S AND PDR'S
CLR (R0) ; CLEAR ONE OF THE USER PAR'S
JSR PC, CMPREG ; SEE IF OTHER PAR/PDR'S WERE EFFECTED

```

2070 023154 062700 000002
2071 023160 077312
2072 023162 012737 022752 001110
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084 023170 000004
2085
2086
2087 023172 004737 035520
2088 023176 000005
2089 023200 012700 172300
2090 023204 012704 000010
2091 023210 011001
2092 023212 022701 077416
2093 023216 001401
2094 023220 104055
2095
2096
2097
2098 023222 062700 000002
2099 023226 077410
2100 023230 012700 172340
2101 023234 012704 000010
2102 023240 011001
2103 023242 022701 007777
2104 023246 001401
2105 023250 104055
2106
2107
2108
2109 023252 062700 000002
2110 023256 077410
2111 023260 012700 177600
2112 023264 012704 000010
2113 023270 011001
2114 023272 022701 077416
2115 023276 001401
2116 023300 104055
2117
2118
2119
2120 023302 062700 000002
2121 023306 077410
2122
2123 023310 012700 177640
2124 023314 012704 000010
2125 023320 011001

```

```

ADD #2,R0 ;FORM ADDRESS OF NEXT USER PAR TO CLEAR
SOB R3,5$ ;LOOP TO 5$ UNTIL ALL USER PAR'S CHECKED
MOV #1$,SLPERR ;SET LOOP ON ERROR POINTER TO 1$

*****
*TEST 22 TEST THAT PAR-PDR'S NOT AFFECTED BY RESET
*
* THIS TEST CHECKS TO SEE THAT THE KERNEL OR USER PAR/PDR'S ARE
* NOT AFFECTED BY THE EXECUTION OF A "RESET" INSTRUCTION. THE
* "SETREG" SUBROUTINE IS USED TO SET ALL WRITEABLE BITS TO A "1" IN
* THE PAR/PDR'S. THEN THEY ARE READ TO SEE THAT THEY REMAINED
* UNCHANGED
*****
†ST22: SCOPE

1$: JSR PC,SETREG ;SET ALL BITS IN ALL PAR'S AND PDR'S
RESET ;ISSUE AN "INIT" BY EXECUTING A RESET
MOV #KIPDR0,R0 ;LOAD ADDRESS OF FIRST KERNEL PDR IN R0
MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
2$: MOV (R0),R1 ;READ A KERNEL PDR INTO R1
CMP #77416,R1 ;ARE ALL THE BITS STILL SET?
BEQ 3$ ;BRANCH IF YES
ERROR 5$ ;KERNEL PDR AFFECTED BY A RESET
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 2$" = 000773

3$: ADD #2,R0 ;FORM ADDRESS OF NEXT KERNEL PDR
SOB R4,2$ ;LOOP TO 2$ UNTIL ALL KERNEL PDR'S CHECKED
MOV #KIPAR0,R0 ;LOAD ADDRESS OF FIRST KERNEL PAR IN R0
MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
4$: MOV (R0),R1 ;READ A KERNEL PAR INTO R1
CMP #7777,R1 ;ARE ALL THE BITS STILL SET?
BEQ 5$ ;BRANCH IF YES
ERROR 5$ ;KERNEL PAR AFFECTED BY A RESET
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 4$" = 000773

5$: ADD #2,R0 ;FORM ADDRESS OF NEXT KERNEL PAR
SOB R4,4$ ;LOOP TO 4$ UNTIL ALL KERNEL PAR'S CHECKED
MOV #UIPDR0,R0 ;LOAD ADDRESS OF FIRST USER PDR IN R0
MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
6$: MOV (R0),R1 ;READ A USER PDR INTO R1
CMP #77416,R1 ;ARE ALL THE BITS STILL SET?
BEQ 7$ ;BRANCH IF YES
ERROR 5$ ;USER PDR AFFECTED BY A RESET
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 6$" = 000773

7$: ADD #2,R0 ;FORM ADDRESS OF NEXT USER PDR
SOB R4,6$ ;LOOP TO 6$ UNTIL ALL USER PDR'S CHECKED

MOV #UIPAR0,R0 ;LOAD ADDRESS OF FIRST USER PAR IN R0
MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
8$: MOV (R0),R1 ;READ A USER PAR INTO R1

```

```

2126 023322 022701 007777
2127 023326 001401
2128 023330 104055
2129
2130
2131
2132 023332 062700 000002
2133 023336 077410
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159 023340 000004
2160 023342 004737 035432
2161 023346 012700 172300
2162 023352 012704 000010
2163 023356 005020
2164 023360 077402
2165 023362 012737 000252 000250
2166 023370 005037 000252
2167
2168
2169 023374 012737 001006 172300
2170 023402 012737 023410 001110
2171 023410 012737 000400 177572
2172 023416 000005
2173 023420 032737 000400 177572
2174 023426 001403
2175 023430 005037 177572
2176 023434 104061
2177
2178
2179
2180 023436 012706 001100
2181 023442 005037 177572

```

```

CMP #7777,R1 ;ARE ALL THE BITS STILL SET?
BEQ 9$ ;BRANCH IF YES
ERROR 55 ;USER PAR AFFECTED BY A RESET
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 8$" = 000773
9$: ADD #2,R0 ;FORM ADDRESS OF NEXT USER PAR
SOB R4,8$ ;LOOP TO 8$ UNTIL ALL USER PAR'S CHECKED

```

```

*****
*TEST 23 INSTRUCTION FETCH NOT RELOCATED IN MAINT. MODE
*
* THIS TEST CHECKS TO SEE THAT WHEN MEMORY MANAGEMENT IS IN
* MAINTENANCE MODE (DESTINATION-ONLY-RELOCATION), AN INSTRUCTION
* FETCH IS NOT RELOCATED AND A RESET CLEARS THE MAINTENANCE BIT
* (BIT 08) IN SRO. IF THE "FETCH" IS RELOCATED, A PG.LENGTH ABORT
* SHOULD OCCUR, CAUSING A HALT SINCE TRAP CATCHER IS PLACED IN VECTOR 250

```

```

NOTE: A HALT MAY OCCUR IF MAINT. MODE NOT DISABLED BY RESET

```

```

*****
*TEST 23: SCOPE
1$: JSR PC,TOFF ;TURN T-BIT TRAPPING OFF FOR THIS TEST
MOV #KIPDR0,R0 ;LOAD ADDRESS OF FIRST KERNEL PDR INTO R0
MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
2$: CLR (R0)+ ;CLEAR PDR - MAPPING PAGE NON-RES, 0 BLKS.
SOB R4,2$ ;LOOP TO 2$ UNTIL ALL KERNEL PDR'S CLEARED
MOV #MMVEC+2,MMVEC ;LOAD TRAP CATCHER INTO MEM MGMT. VECTOR
CLR MMVEC+2
;A HALT WILL OCCUR IF RESET FAILS
;TO DISABLE DEST.-ONLY RELOCATION
;MAP KERNEL PG 0 R/W, 3 BLOCKS LONG.
3$: MOV #1006,KIPDR0 ;SET LOOP ON ERROR POINTER TO 3$
MOV #3$,SLPERR ;TURN ON DEST-ONLY-RELOCATION
MOV #BIT8,SRO ;SHOULD CLEAR MAINT. BIT - WILL ABORT IF RELOCATED
RESET ;WAS MAINT. BIT (BIT 8) OF SRO CLEARED?
BIT #BIT8,SRO ;BRANCH IF YES
BEQ 4$ ;CLEAR SRO SO ERROR CAN BE REPORTED
CLR SRO ;MAINT. MODE NOT DISABLED BY A RESET
ERROR 61 ;FOR A TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 3$" = 000765
4$: MOV #KERSTK,KSP ;RESTORE STACK POINTER
CLR SRO ;BE SURE SRO IS CLEAR

```


2182	023446	012737	002314	000250
2183	023454	012737	000340	000252
2184	023462	012737	023342	001110
2185	023470	004737	035466	

```

MOV    #MGMERR,MMVEC ;RESTORE MEM. MGMT. TRAP VECTOR
MOV    #340,MMVEC+2  ;RESTORE MEM. MGMT VECTOR+2
MOV    #15,$LPERR    ;RESET LOOP ON ERROR POINTER TO 15
JSR    PC,TON        ;TURN T-BIT TRAPPING BACK ON

```

2186				
2187				
2188				
2189				
2190				
2191				
2192				
2193				
2194				
2195				
2196				
2197				
2198				
2199				

```

*****
*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.

```

2212	023474	000004		
2213	023476	004737	035432	
2214	023502	012737	023572	001110
2215	023510	012737	000600	172346
2216	023516	012737	000600	172350
2217	023524	012737	007600	172356
2218	023532	005037	172306	
2219	023536	012737	077406	172310
2220	023544	012737	077406	172316
2221	023552	012737	023652	000250
2222	023560	012737	000377	060000
2223	023566	012700	177777	
2224	023572	052737	000400	177572
2225	023600	113737	060000	100001
2226	023606	000005		
2227	023610	013701	060000	
2228	023614	020001		
2229	023616	001401		
2230	023620	104062		

```

*****
*ST24: SCOPE
15: JSR    PC,TOFF ;TURN OFF T-BIT TRAPPING FOR THIS TEST
    MOV    #25,$LPERR ;SET LOOP ON ERROR POINTER TO 25
    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    #45,MMVEC ;SET M.M. VECTOR TO 45 IN CASE OF ABORT
    MOV    #377,2#60000 ;LOAD ALL 1'S IN LOW BYTE OF TEST LOC.
    MOV    #-1,R0 ;LOAD EXPECTED CONTENTS OF TEST LOC. IN R0
25: BIS    #BIT8,SRO ;TURN ON DEST.-ONLY-RELOCATION
    MOVB   2#60000,2#100001 ;LOAD HI BYTE OF TEST LOC.-ABORT IF SOURCE RELOCATED
    RESET ;TURN OFF DEST.-ONLY-RELOCATION
    MOV    2#60000,R1 ;READ CONTENTS OF TEST LOC.
    CMP    R0,R1 ;WAS TEST LOCATION LOADED PROPERLY?
    BEQ    35 ;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 25" = 000764

```

2235	023622	012737	002314	000250
2236	023630	012737	077406	172306
2237	023636	012737	023476	001110

```

35: MOV    #MGMERR,MMVEC ;RESTORE MEM. MGMT. VECTOR
    MOV    #77406,KIPDR3 ;MAP KERNAL PAGE 7 R/W, 128 BLKS
    MOV    #15,$LPERR ;RESET LOOP ON ERROR POINTER TO 15

```

K04

MD-11-DFKTH-A POP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 49
T24 TEST THAT SOURCE NOT RELOCATED IN MAINTENANCE MODE

2238 023644 004737 035466
 2239 023650 000430
 2240
 2241
 2242 023652 042737 000400 177572
 2243 023660 013737 177572 001362
 2244 023666 013737 177576 001364
 2245 023674 010637 001354
 2246 023700 012637 001356
 2247 023704 012637 001360
 2248 023710 042737 160000 177572
 2249 023716 104063
 2250
 2251
 2252
 2253 023720 013746 001360
 2254 023724 013746 001356
 2255 023730 000002
 2256
 2257
 2258
 2259
 2260
 2261
 2262
 2263
 2264
 2265
 2266
 2267
 2268
 2269
 2270
 2271
 2272
 2273
 2274
 2275
 2276
 2277
 2278
 2279
 2280
 2281
 2282
 2283
 2284
 2285
 2286 023732 000004
 2287
 2288 023734 012700 172340
 2289 023740 005001
 2290 023742 012702 000007
 2291 023746 010120
 2292 023750 062701 000200
 2293 023754 077204

```

JSR PC,T0N ;TURN T-BIT TRAPPING BACK ON
BR TST25 ;SKIP TO NEXT TEST
;* IF THE PROGRAM TRIES TO RELOCATE THE SOURCE, IT SHOULD TRAP TO 45
45: BIC #BIT8,SRO ;TURN OFF DEST.-ONLY-RELOCATION THRU PAR/PDR 7
MOV SRO,WASSRO ;SAVE REST OF SRO CONTENTS
MOV SR2,WASSR2 ;SAVE CONTENTS OF SR2
MOV SP,WASRG6 ;SAVE VALUE OF THE STACK POINTER
MOV (SP)+,TRAPPC ;SAVE PC OF TRAP
MOV (SP)+,TRAPPS ;SAVE PSW OF TRAP
BIC #160000,SRO ;CLEAR ERROR BITS IN SR1
ERROR 63 ;SOURCE APPARENTLY RELOCATED IN MAINT. MODE
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH A
;"NOP" = 000240
MOV TRAPPS,-(SP) ;PUT PSW OF TRAP BACK ON THE STACK
MOV TRAPPC,-(SP) ;PUT PC OF TRAP BACK ON THE STACK
RTI ;RETURN TO TEST

```

```

*****
;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
15: MOV #KIPAR0,R0 ;LOAD ADDRESS OF FIRST KERNEL PAR IN R0
CLR R1 ;CLEAR R1
MOV #7,R2 ;LOAD LOOP COUNTER WITH A 7
25: MOV R1,(R0)+ ;MAP KERNEL PAR'S TO PAGES 0-6 (4K EACH)
ADD #200,R1
SOB R2,25 ;LOOP UNTIL KIPAR0 - KIPAR6 ARE LOADED

```


2406	024374	012701	117776	MOV	#117776,R1	:LOAD VIRTUAL ADDR. FOR PSW INTO R1
2407	024400	012702	030240	MOV	#030240,R2	:LOAD DATA FOR PSW IN R2
2408	024404	012704	007600	MOV	#7600,R4	:LOAD R4 WITH PAR VALUE
2409	024410	010437	172350	MOV	R4,KIPAR4	:LOAD KERNEL PAR 4 BITS <11:00>
2410	024414	052737	000400	BIS	#BIT8,SRO	:TURN ON "DESTINATION-ONLY-RELOCATION"
2411	024422	010211		MOV	R2,(R1)	:LOAD PSW USING ADDER CHIPS (PAR4 + VIRT. ADDR.)
2412	024424	011003		MOV	(R0),R3	:READ PSW BACK WITHOUT USING MEM. MGMT.
2413	024426	000005		RESET		:TURN OFF MEM. MGMT. MAINT. MODE
2414	024430	005010		CLR	(R0)	:CLEAR THE PSW
2415	024432	042703	000037	BIC	#37,R3	:MASK T-BIT & CC BITS OUT OF DATA READ
2416	024436	020203		CMP	R2,R3	:WAS PSW WRITTEN WHILE IN MAINT. MODE?
2417	024440	001405		BEQ	12\$:BRANCH IF YES
2418	024442	010137	001376	MOV	R1,VIRT1	:SAVE VIRTUAL ADDR. TO FORM PHYSICAL ADDR.
2419	024446	004737	036004	JSR	PC,FORMPA	:GO FORM PHYSICAL ADDR. FOR TYPING
2420	024452	104017		ERROR	17	:PSW DID NOT HAVE DATA THAT IT SHOULD
2421						:HAVE, APPARENTLY PHYS. ADDR. OF PSW WAS
2422						:NOT FORMED BY ADDERS USING THE
2423						:VIRTUAL ADDR. AND KIPAR4
2424						:FOR TIGHTER SCOPE LOOP
2425						:REPLACE ERROR CALL WITH
2426						: "BR 11\$" = 000743
2427	024454	012737	023734	001110	12\$: MOV #1\$,SLPERR	:RESET LOOP ON ERROR POINTER TO 1\$
2428						
2429						
2430						*****
2431						:TEST 26 RELOCATION & ADDER TEST (WITH CARRIES)
2432						:*
2433						: THE FOLLOWING TEST USES THE SAME METHOD AS THE PREVIOUS
2434						: TEST TO VERIFY MEMORY MANagements ABILITY TO CONSTRUCT
2435						: PHYSICAL BUS ADDRESSES USING A VIRTUAL BUS ADDRESS AND THE
2436						: CONTENTS OF A PAGE ADDRESS REGISTER. HOWEVER, THE VALUES
2437						: AND PATTERNS USED IN THIS TEST WILL GENERATE CARRIES FROM
2438						: CHIP TO CHIP AND CHECK "WRAPAROUND" TO ADDRESS 000000 BY
2439						: USING VIRTUAL ADDR. 100100 AND KIPAR4 = 7777.
2440						:*
2441	024462	000004				*****
2442						:TEST 26: SCOPE
2443	024464				1\$:	: KERNEL PAR'S AND PDR'S HAVE BEEN
2444						: SETUP BY THE PREVIOUS TEST
2445	024464	012737	024464	001110	2\$: MOV #2\$,SLPERR	: SET LOOP ON ERROR POINTER TO 2\$
2446	024472	012700	064276		MOV #64276,R0	:LOAD PHYSICAL ADDR. PBA INTO R0
2447	024476	012701	102176		MOV #102176,R1	:LOAD VIRTUAL ADDR. VBA INTO R1
2448	024502	012702	125253		MOV #125253,R2	:LOAD TEST PATTERN INTO R2
2449	024506	012704	000621		MOV #621,R4	:LOAD R4 WITH PAR VALUE
2450	024512	010437	172350		MOV R4,KIPAR4	:LOAD KERNEL PAR 4 BITS <11:00>
2451	024516	011037	001176		MOV (R0),STMP0	:SAVE CONTENTS AT TEST LOCATION
2452	024522	052737	000400	177572	BIS #BIT8,SRO	:TURN ON "DESTINATION-ONLY-RELOCATION"
2453	024530	010211			MOV R2,(R1)	:LOAD 125253 USING ADDER CHIPS (PAR4 + VIRT ADDR.)
2454	024532	011003			MOV (R0),R3	:READ 125253 BACK WITHOUT USING MEM. MGMT.
2455	024534	000005			RESET	:TURN OFF MEMORY MGMT. MAINT. MODE
2456	024536	013710	001176		MOV STMP0,(R0)	:RESTORE ORIGINAL CONTENTS TO TEST LOC.
2457	024542	020203			CMP R2,R3	:WAS SAME PATTERN READ BACK THAT WAS
2458						: WRITTEN USING "DEST-ONLY-RELOC."?
2459	024544	001405			BEQ 3\$:BRANCH IF YES
2460	024546	010137	001376		MOV R1,VIRT1	:SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2461	024552	004737	036004		JSR PC,FORMPA	:GO FORM PHYSICAL ADDRESS FOR TYPING

2574	025140			
2575	025140	012737	024464	001110
2576				
2577				
2578				
2579				
2580				
2581				
2582				
2583				
2584				
2585				
2586				
2587				
2588				
2589				
2590				
2591				
2592				
2593				
2594				
2595				
2596				
2597				
2598				
2599				
2600				
2601	025146	000004		
2602				
2603	025150	005037	177776	
2604	025154	012704	000577	
2605	025160	012705	000600	
2606	025164	010437	172350	
2607	025170	010537	172352	
2608	025174	012700	177640	
2609	025200	005001		
2610	025202	012702	000007	
2611	025206	010120		
2612	025210	062701	000200	
2613	025214	077204		
2614	025216	012710	007600	
2615	025222	012700	177600	
2616	025226	012701	077406	
2617	025232	012702	000010	
2618	025236	010120		
2619	025240	077202		
2620	025242	105037	177610	
2621	025246	105037	177612	
2622	025252	010537	177650	
2623	025256	010437	177652	
2624	025262	012737	000001	177572
2625	025270	012737	025322	001110
2626	025276	012737	025534	000250
2627	025304	013737	177776	00117E
2628	025312	012700	100100	
2629	025316	012701	120000	

```

11$:  MOV    #15,$LPERR    ;RESET LOOP ON ERROR POINTER TO 15
:*****
*TEST 27  READ AND WRITE WHILE IN RELOCATE MODE
*
* THE FOLLOWING TEST TURNS ON MEMORY MANAGEMENT AND THEN
* READS AND WRITES LOCATIONS BETWEEN PHYSICAL ADDRESSES
* 060000-067600. ONE LOCATION IN EVERY BLOCK (32. WORDS)
* IS WRITTEN USING PAR4 AND READ USING PAR5. THIS IS
* DONE IN BOTH USER AND KERNEL MODES. THE USER PAR/PDR'S
* ARE SETUP AT THE BEGINNING OF THE TEST AND ONCE MEMORY
* MANAGEMENT IS TURNED ON IT IS LEFT ON FOR THE REST OF THE
* OF THE PROGRAM. THE "MODE" INPUT TO THE PAR/PDR ADDRESS MUX
* IS CHECKED BY READING AND WRITING IN USER MODE. REMEMBER
* ALSO, THAT SINCE MEMORY MANAGEMENT IS ON (IN RELOCATE
* MODE) THE PROGRAM ITSELF IS USING ITS VIRTUAL ADDRESSES AND
* THE PAR/PDR'S TO EXECUTE.
*
* WHILE TESTING IN KERNEL MODE, USER PAGES 4 & 5 ARE MAPPED
* NON-RESIDENT WITH DIFFERENT PAR VALUES THAN THE KERNEL
* PAR'S TO BE SURE THAT THE KERNEL PAR'S AND PDR'S ARE BEING
* USED WHEN IN KERNEL MODE (AND VICE VERSA WHILE TESTING IN
* USER MODE). IF A MEM. MGMT. TRAP OCCURS, THE PROGRAM GOES
* TO 8$ WHERE THE TRAP IS REPORTED.
*
:*****
†ST27:  SCOPE
1$:  CLR    PSW            ;START IN KERNEL MODE
      MOV    #577,R4      ;LOAD R4 WITH VALUE FOR PAR4
      MOV    #600,R5      ;LOAD R5 WITH VALUE FOR PAR5
      MOV    R4,KIPAR4    ;LOAD KERNEL PAR4
      MOV    R5,UIPAR5    ;LOAD KERNEL PAR5
      MOV    #UIPAR0,R0   ;LOAD ADDRESS OF FIRST USER PAR IN R0
      CLR    R1            ;CLEAR R1
      MOV    #7,R2        ;LOAD LOOP COUNTER WITH A 7
2$:  MOV    R1,(R0)+      ;MAP USER PAR'S TO PAGES 0-6 (4K EACH)
      ADD    #200,R1
      SOB   R2,2$        ;LOOP UNTIL UIPAR0-UIPAR6 ARE LOADED
      MOV    #7600,(R0)   ;MAP USER PAR7 TO THE I/O PAGE
      MOV    #UIPDR0,R0   ;LOAD ADDRESS OF FIRST USER PDR IN R0
      MOV    #77406,R1    ;LOAD PDR DATA INTO R1
      MOV    #10,R2       ;LOAD LOOP COUNTER WITH AN 8
3$:  MOV    R1,(R0)+      ;MAP ALL 8 PAGES 128 BLOCKS UPWARD
      SOB   R2,3$        ;EXPANDABLE, READ/WRITE
      CLRB  UIPDR4        ;MAP USER SPACE NON-RESIDENT WHILE
      CLRB  UIPDR5        ;TESTING KERNEL SPACE
      MOV    R5,UIPAR4    ;MAP USER PAR'S OPPOSITE OF KIPAR'S
      MOV    R4,UIPAR5
      MOV    #1,SRO       ;TURN ON MEMORY MANAGEMENT (RELOCATE MODE)
      MOV    #55,$LPERR   ;SET LOOP ON ERROR POINTER TO 55
      MOV    #85,$MVEC    ;SET M. M. TRAP VECTOR TO 85
4$:  MOV    PSW,$TMPD     ;SAVE PSW IN CASE OF ERROR
      MOV    #100100,R0   ;PUT VIRTUAL ADDR. THAT USER PAR4 IN R0
      MOV    #120000,R1   ;PUT VIRTUAL ADDR. THAT USES PAR5 IN R1

```


E05

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 2. JAN-77 08:51 PAGE 56
T27 READ AND WRITE WHILE IN RELOCATE MODE

2630	025322	010010		55:	MOV	RO,(RO)		;WRITE TO TEST LOC. USING PAR4
2631	025324	011102			MOV	(R1),R2		;READ THE SAME LOC. BUT USING PAR5
2632	025326	020002			CMP	RO,R2		;DID WE READ WHAT WE WROTE?
2633	025330	001411			BEQ	65		;BRANCH IF YES
2634	025332	010137	001400		MOV	R1,VIRT2		;SAVE VIRTUAL ADDR. THAT SELECTED PAR5
2635	025336	010037	001376		MOV	RO,VIRT1		;SAVE VIRTUAL ADDR. THAT SELECTED PAR4
2636	025342	004737	036004		JSR	PC,FORMPA		;GO FORM PHYSICAL ADDRESS BEING USED
2637	025346	104020			ERROR	20		;READING LOC. USING PAR5 AND A VIRT.
2638								;ADDR. DID NOT FIND DATA WRITTEN WHEN USING
2639								;PAR4 AND VIRT. ADDRESS.
2640								;FOR TIGHTER SCOPE LOOP
2641								;REPLACE ERROR CALL WITH
2642								;"BR 55" = 000765
2643	025350	013700	001376		MOV	VIRT1,RO		;RESTORE VBA IN RO
2644	025354	062700	000100	65:	ADD	#100,RO		;CHANGE VIRTUAL ADDRS. TO POINT TO NEXT BLOCK
2645	025360	062701	000100		ADD	#100,R1		
2646	025364	020127	127700		CMP	R1,#127700		;WERE BLOCKS FROM 60000-676000 ALL TRIED?
2647	025370	001354			BNE	55		;BRANCH IF NO
2648	025372	032737	140000 177776		BIT	#140000,PSW		;HAVE WE DONE TEST IN USER MODE YET?
2649	025400	001026			BNE	75		;BRANCH IF YES
2650	025402	010437	177650		MOV	R4,UIPAR4		;LOAD USER PAR4
2651	025406	010537	177652		MOV	R5,UIPAR5		;LOAD USER PAR5
2652	025412	112737	000006 177610		MOVSB	#6,UIPDR4		;MAP USER SPACE R/W TO TEST IT
2653	025420	112737	000006 177612		MOVSB	#6,UIPDR5		
2654	025426	105037	172310		CLRB	KIPDR4		;MAP KERNEL SPACE NON-RESIDENT WHILE
2655	025432	105037	172312		CLRB	KIPDR5		;TESTING USER SPACE
2656	025436	010537	172350		MOV	R5,KIPAR4		;MAP KERNEL PAR'S OPPOSITE UIPAR'S
2657	025442	010437	172352		MOV	R4,KIPAR5		
2658	025446	012737	140000 177776		MOV	#140000,PSW		;GO TO USER MODE
2659	025454	000713			BR	45		;GO BACK AND READ/WRITE IN USER MODE
2660	025456	005037	177776	75:	CLR	PSW		;GO BACK TO KERNEL MODE BEFORE LEAVING
2661	025462	012737	077406 172310		MOV	#77406,KIPDR4		;REMAP KERNEL PAGES READ/WRITE
2662	025470	012737	077406 172312		MOV	#77406,KIPDR5		
2663	025476	010537	172350		MOV	R5,KIPAR4		;MAP KERNEL AND USER PAR'S 4 & 5
2664	025502	010537	172352		MOV	R5,KIPAR5		;BACK TO 12-16K
2665	025506	010537	177650		MOV	R5,UIPAR4		
2666	025512	010537	177652		MOV	R5,UIPAR5		
2667	025516	012737	002314 000250		MOV	#MGMERR,MMVEC		;RESTORE ADDR. OF NORMAL M.M. TRAP ROUTINE
2668	025524	012737	025150 001110		MOV	#15,\$LPERR		;RESET LOOP ON ERROR POINTER TO 15
2669	025532	000427			BR	TST30		;GO TO NEXT TEST
2670								
2671	025534	012637	001356	85:	MOV	(KSP)+,TRAPPC		;SAVE PC & PS OF TRAP
2672	025540	012637	001360		MOV	(KSP)+,TRAPPS		
2673								;PROGRAM WILL TRAP TO HERE IF TRY
2674								;TO USE USER PDR'S WHEN IN KERNEL MODE
2675								;OR KERNEL PDR'S WHEN IN USER MODE
2676	025544	010037	001376		MOV	RO,VIRT1		;SAVE VIRTUAL ADDRESS FOR ERROR REPORT
2677	025550	004737	036004		JSR	PC,FORMPA		;GO FORM THE PHYSICAL ADDRESS BEING USED
2678	025554	013737	177572 001362		MOV	SRO,WASSRO		;SAVE SRO & SR2 FOR ERROR REPORT
2679	025562	013737	177576 001364		MOV	SR2,WASSR2		
2680	025570	042737	160000 177572		BIC	#160000,SRO		;CLEAR ERROR BITS IN SRO
2681	025576	104052			ERROR	52		;M.M. TRAP WHILE IN RELOCATE MODE -
2682								;REFERENCED WRONG SET OF PDR'S
2683								;FOR TIGHTER SCOPE LOOP
2684								;REPLACE ERROR CALL WITH
2685								;A "NOP" = 000240

F05

MD-11-DFKTH-A POP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 57
T27 READ AND WRITE WHILE IN RELOCATE MODE

2686 025600 013746 001360
2687 025604 013746 001356
2688 025610 000002

MOV TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
MOV TRAPPC,-(KSP)
RTI ;RETURN TO TEST

2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707

*TEST 30 W-BIT LOGIC TEST, KERNEL PDR'S

THIS TEST WRITES TO EIGHT (8) DIFFERENT VIRTUAL ADDRESSES
(VBA'S = 17776,37776,57776,77776,117776,137776,157776, & 177776
& PBA'S CONSTRUCTED = 17776,37776,57776,77776,77776,
77776,77776, & 777776 RESPECTIVELY).
WHICH SHOULD CAUSE THE "W-BIT" TO SET IN EACH OF THE
EIGHT (8) KERNEL PAGE DESCRIPTOR REGISTERS. THE PDR'S
ARE CHECKED TO SEE THAT IT'S W-BIT DOES SET WHEN THE
PAGE IT IS MAPPED TO IS WRITTEN TO AND THAT THE W-BIT
DOES NOT SET IN ANY OF THE OTHER PDR'S. KERNEL PDR'S 3,4,5,6
ARE MAPPED TO 12-16K FOR THIS TEST. ALSO THE W-BIT
SHOULD BE CLEARED WHEN THE PDR IS WRITTEN TO. THE
W-BIT PORTION OF THE PDR'S AND THE PAR/PDR ADRS MUX
ARE BEING CHECKED.

2708 025612 000004
2709 025614
2710 025614 004737 035432
2711 025620 012702 000004
2712 025624 012700 172346
2713 025630 012701 000600
2714 025634 010120
2715 025636 077202
2716 025640 012705 172300
2717 025644 012704 000010
2718 025650 012703 017776
2719 025654 012737 025662 001110
2720 025662 012700 172300
2721 025666 012702 000010
2722 025672 012701 077406
2723 025676 010120
2724 025700 077202
2725 025702 011313
2726 025704 03:527 000100
2727 025710 001002
2728 025712 104021

*ST30: SCOPE
1\$:

```

JSR PC,TOFF ;TURN T-BIT TRAPPING OFF FOR THIS TEST
MOV #4,R2 ;SET LOOP COUNTER TO 4
MOV #KIPAR3,RO ;LOAD ADDRESS OF PAR3 INTO RO
MOV #600,R1 ;LOAD "12-16K" PAR VALUE INTO R1
2$: MOV R1,(R0)+ ;MAP PARS 3-6 TO 12-16K
SOB R2,2$ ;LOOP TIL ALL 4 OF THEM LOADED
MOV #KIPDRO,R5 ;LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
MOV #10,R4 ;SET LOOP COUNTER TO 8
MOV #17776,R3 ;INITIALIZE VIRTUAL ADDRESS TO BE IN R3
MOV #3$,SLPERR ;SET LOOP ON ERROR POINTER TO 3$
3$: MOV #KIPDRO,RO ;LOAD ADDR. OF FIRST PDR TO BE SETUP IN RO
MOV #10,R2 ;SET LOOP COUNTER TO 8
MOV #77406,R1 ;PUT "W-BIT OFF DATA" INTO R1
4$: MOV R1,(R0)+ ;CLEAR ALL W-BITS BY WRITING TO ALL PDRS
SOB R2,4$ ;LOOP UNTIL ALL OF THEM SETUP
MOV (R3),(R3) ;DO "DATA" TO VIRTUAL ADDR.-SETTING A W-BIT
BIT (R5),#WBIT ;DID THAT CAUSE W-BIT TO BE SET?
BNE 5$ ;BRANCH IF YES
ERROR 21 ;W-BIT DID NOT GET SET IN PDR
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 3$" = 000763
5$: BR 8$ ;SKIP CHECKING OTHER PDR'S-ERROR WILL SET W-BITS
MOV #10,R2 ;SET LOOP COUNTER TO 8
MOV #KIPDRO,RO ;LOAD ADDR. OF FIRST PDR TO BE CHECKED IN RO
6$: BIT (RO),#WBIT ;DID W-BIT IN OTHER PDRS REMAIN CLEAR?
BEQ 7$ ;BRANCH IF YES
CMP R5,RO ;IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
BEQ 7$ ;BRANCH IF YES
ERROR 22 ;W-BIT GOT SET IN MORE THAN ONE PDR
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH

```

2729
2730
2731
2732 025714 000422
2733 025716 012702 000010
2734 025722 012700 172300
2735 025726 031027 000100
2736 025732 001403
2737 025734 020500
2738 025736 001401
2739 025740 104022
2740
2741

2742					
2743	025742	062700	000002		
2744	025746	077211			
2745	025750	010115			
2746	025752	031527	000100		
2747	025756	001401			
2748	025760	104023			
2749					
2750					
2751					
2752	025762	062705	000002		
2753	025766	062703	020000		
2754	025772	077445			
2755	025774	012737	025614	001110	
2756	026002	004737	035466		
2757					
2758					
2759					
2760					
2761					
2762					
2763					
2764					
2765					
2766					
2767					
2768					
2769					
2770					
2771					
2772					
2773					
2774					
2775	026006	000004			
2776	026010	012737	140000	177776	
2777	026016	004737	035432		
2778	026022	012702	000004		
2779	026026	012700	177646		
2780	026032	012701	000600		
2781	026036	010120			
2782	026040	077202			
2783	026042	012705	177600		
2784	026046	012704	000010		
2785	026052	012703	017776		
2786	026056	012737	026064	001110	
2787	026064	012700	177600		
2788	026070	012702	000010		
2789	026074	012701	077406		
2790	026100	010120			
2791	026102	077202			
2792	026104	011313			
2793	026106	031527	000100		
2794	026112	001002			
2795	026114	104021			
2796					
2797					

```

7$: ADD #2,R0 ;"BR 3$" = 000750
    SOB R2,6$ ;POINT R0 TO NEXT PDR TO BE CHECKED
    MOV R1,(R5) ;LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
    BIT (R5),#WBIT ;WRITE TO THE PDR TESTED TO CLEAR W-BIT
    BEQ 8$ ;DID WRITING PDR CLEAR THE W-BIT?
    ERROR 23 ;BRANCH IF YES
                    ;W-BIT DID NOT CLEAR BY WRITING THE PDR
                    ;FOR TIGHTER SCOPE LOOP
                    ;REPLACE ERROR CALL WITH
                    ;"BR 3$" = 000740
8$: ADD #2,R5 ;POINT R5 TO THE NEXT PDR TO BE TESTED
    ADD #20000,R3 ;CHANGE VIRT. ADDR TO REF. NEXT PDR
    SOB R4,3$ ;LOOP BACK TO 3$ UNTIL ALL 8 PDR'S TESTED
    MOV #1$,SLPERR ;RESET LOOP ON ERROR POINTER TO 1$
    JSR PC,TON ;TURN T-BIT BACK ON FOR NEXT TEST

```

```

*****
*TEST 31 W-BIT LOGIC TEST, USER PDR'S

```

```

* THIS TEST WRITES TO EIGHT (8) DIFFERENT VIRTUAL ADDRESSES
* (VBA'S = 17776, 37776, 57776, 77776, 117776, 137776, 157776, & 177776
* & PBA'S CONSTRUCTED = 17776, 37776, 57776, 77776, 77776,
* 77776, 77776, & 777776 RESPECTIVELY).
* WHICH SHOULD CAUSE THE "W-BIT" TO SET IN EACH OF THE
* EIGHT (8) USER PAGE DESCRIPTOR REGISTERS. THE PDR'S
* ARE CHECKED TO SEE THAT IT'S W-BIT DOES SET WHEN THE
* PAGE IT IS MAPPED TO IS WRITTEN TO AND THAT THE W-BIT
* DOES NOT SET IN ANY OF THE OTHER PDR'S. USER PDR'S 3,4,5,6
* ARE MAPPED TO 12-16K FOR THIS TEST. ALSO THE W-BIT
* SHOULD BE CLEARED WHEN THE PDR IS WRITTEN TO. THE
* W-BIT PORTION OF THE PDR'S AND THE PAR/PDR ADRS MUX
* ARE BEING CHECKED.

```

```

*****

```

```

TST31: SCOPE
1$: MOV #140000,PSW ;GO TO USER MODE FOR THIS TEST
    JSR PC,TOFF ;TURN T-BIT TRAPPING OFF FOR THIS TEST
    MOV #4,R2 ;SET LOOP COUNTER TO 4
    MOV #UIPAP3,R0 ;LOAD ADDRESS OF PAR3 INTO R0
    MOV #600,R1 ;LOAD "12-16K" PAR VALUE INTO R1
2$: MOV R1,(R0)+ ;MAP PARS 3-6 TO 12-16K
    SOB R2,2$ ;LOOP TIL ALL 4 OF THEM LOADED
    MOV #UIPDRO,R5 ;LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
    MOV #10,R4 ;SET LOOP COUNTER TO 8
    MOV #17776,R3 ;INITIALIZE VIRTUAL ADDRESS TO BE IN R3
    MOV #3$,SLPERR ;SET LOOP ON ERROR POINTER TO 3$
3$: MOV #UIPDRO,R0 ;LOAD ADDR. OF FIRST PDR TO BE SETUP IN R0
    MOV #10,R2 ;SET LOOP COUNTER TO 8
    MOV #77406,R1 ;PUT "W-BIT OFF DATA" INTO R1
4$: MOV R1,(R0)+ ;CLEAR ALL W-BITS BY WRITING TO ALL PDRS
    SOB R2,4$ ;LOOP UNTIL ALL OF THEM SETUP
    MOV (R3),(R3) ;DO "DATA" TO VIRTUAL ADDR.-SETTING A W-BIT
    BIT (R5),#WBIT ;DID THAT CAUSE W-BIT TO BE SET?
    BNE 5$ ;BRANCH IF YES
    ERROR 21 ;W-BIT DID NOT GET SET IN PDR
                    ;FOR TIGHTER SCOPE LOOP
                    ;REPLACE ERROR CALL WITH

```

H05

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 59
T31 W-BIT LOGIC TEST, USER PDR'S

```
2798  
2799 026116 000422  
2800 026120 012702 000010  
2801 026124 012700 177600  
2802 026130 031027 000100  
2803 026134 001403  
2804 026136 020500  
2805 026140 001401  
2806 026142 :04022  
2807  
2808  
2809  
2810 026144 062700 000002  
2811 026150 077211  
2812 026152 010115  
2813 026154 031527 000100  
2814 026160 001401  
2815 026162 104023  
2816  
2817  
2818  
2819 026164 062705 000002  
2820 026170 062703 020000  
2821 026174 077445  
2822 026176 012737 026010 001110  
2823 026204 004737 035466  
2824 026210 005037 177776  
2825  
2826  
2827  
2828  
2829  
2830  
2831  
2832  
2833  
2834  
2835  
2836  
2837 026214 000004  
2838  
2839 026216 004737 035432  
2840 026222 012701 077406  
2841 026226 012737 026234 001110  
2842 026234 010137 172316  
2843 026240 013700 177572  
2844 026244 010037 177572  
2845 026250 013702 172316  
2846 026254 020102  
2847 026256 001401  
2848 026260 104024  
2849  
2850  
2851  
2852 026262 012737 026262 001110  
2853 026270 010137 172306
```

```
5$: BR 8$  
MOV #10, R2  
MOV #UIPDR0, R0  
6$: BIT (R0), #WBIT  
BEQ 7$  
CMP R5, R0  
BEQ 7$  
ERROR 22  
  
7$: ADD #2, R0  
SOB R2, 6$  
MOV R1, (R5)  
BIT (R5), #WBIT  
BEQ 8$  
ERROR 23  
  
8$: ADD #2, R5  
ADD #20000, R3  
SOB R4, 3$  
MOV #1$, $LPERR  
JSR PC, ↑ON  
CLR PSW
```

:"BR 3\$" = 000763
:SKIP CHECKING OTHER PDR'S-ERROR WILL SET W-BITS
:SET LOOP COUNTER TO 8
:LOAD ADDR. OF FIRST PDR TO BE CHECKED IN R0
:DID W-BIT IN OTHER PDRS REMAIN CLEAR?
:BRANCH IF YES
:IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
:BRANCH IF YES
:W-BIT GOT SET IN MORE THAN ONE PDR
:FOR TIGHTER SCOPE LOOP
:REPLACE ERROR CALL WITH
:"BR 3\$" = 000750
:POINT R0 TO NEXT PDR TO BE CHECKED
:LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
:WRITE TO THE PDR TESTED TO CLEAR W-BIT
:DID WRITING PDR CLEAR THE W-BIT?
:BRANCH IF YES
:W-BIT DID NOT CLEAR BY WRITING THE PDR
:FOR TIGHTER SCOPE LOOP
:REPLACE ERROR CALL WITH
:"BR 3\$" = 000740
:POINT R5 TO THE NEXT PDR TO BE TESTED
:CHANGE VIRT. ADDR TO REF. NEXT PDR
:LOOP BACK TO 3\$ UNTIL ALL 8 PDR'S TESTED
:RESET LOOP ON ERROR POINTER TO 1\$
:TURN T-BIT BACK ON FOR NEXT TEST
:BACK TO KERNEL MODE BEFORE LEAVING

```
*****  
:TEST 32 TEST "W-BIT NOT SET" CASES  
:  
: THIS TEST CHECKS TWO SPECIAL CASES WHERE THE W-BIT DOES  
: NOT GET SET ON A WRITE. FIRST CASE IS THAT THE W-BIT  
: SHOULD NOT SET IN PAGE DESCRIPTOR REG. 7 WHEN WRITING TO  
: STATUS REG SRO (KERNEL PDR 7 IS USED). SECOND CASE IS THAT  
: THE W-BIT IS NOT SET IF THE "DATO" IS ABORTED DUE TO AN  
: ODD ADDRESS ERROR (KERNEL PDR3 & VIRTUAL ADDR 60001 ARE USED).  
:*****  
↑ST32: SCOPE  
1$: JSR PC, ↑OFF  
MOV #77406, R1  
MOV #2$, $LPERR  
2$: MOV R1, KIPDR7  
MOV SRO, R0  
MOV R0, SRO  
MOV KIPDR7, R2  
CMP R1, R2  
BEQ 3$  
ERROR 24  
:TURN OFF T-BIT TRAPPING FOR THIS TEST  
:PUT "W-BIT OFF" VALUE FOR PDR IN R1  
:SET LOOP ON ERROR POINTER TO 2$  
:LOAD KERNEL PDR 7 TO CLEAR W-BIT  
:READ PRESENT CONTENTS OF STATUS REG. 0  
:WRITE PRESENT CONTENTS OF SRO BACK TO ITSELF  
:READ CONTENTS OF KIPDR7 INTO R2  
:WAS W-BIT LEFT CLEARED?  
:BRANCH IF YES  
:W-BIT IN KIPDR7 SET WHEN SRO WAS WRITTEN TO  
:FOR TIGHTER SCOPE LOOP  
:REPLACE ERROR CALL WITH  
:"BR 2$" = 000765  
3$: MOV #3$, $LPERR  
MOV R1, KIPDR3  
:SET LOOP ON ERROR POINTER TO 3$  
:LOAD KERNEL PDR3 WITH 77406 TO CLEAR W-BIT
```

```

2854 026274 012737 026306 000004      MOV      #4$,ERRVEC      ;SET UP LOC. 4 TO 4$ FOR ODD ADDR. ABORT
2855 026302 005237 060001              INC      60001           ;CAUSE ODD ADDRESS ABORT THRU LOC. 4
2856 026306 012706 001100      4$:     MOV      #KERSTK,K$P    ;RESTORE THE STACK POINTER
2857 026312 013702 172306      MOV      KIPDR3,R2      ;READ KIPDR3 INTO R2
2858 026316 020102              CMP      R1,R2          ;WAS W-BIT LEFT CLEARED?
2859 026320 001401              BEQ      5$             ;BRANCH IF YES
2860 026322 104025              ERROR    25            ;W-BIT GOT SET DURING AN ODD ADDR. ABORT
2861                                ;FOR TIGHTER SCOPE LOOP
2862                                ;REPLACE ERROR CALL WITH
2863                                ;"BR 3$" = 000757
2864 026324 012737 002246 000004      5$:     MOV      #TIMERR,ERRVEC  ;RESTORE NORMAL CPU TRAP ROUTINE TO LOC.4
2865 026332 012737 026216 001110      MOV      #1$,SLPERR     ;RESET LOOP ON ERROR POINTER TO 1$
2866 026340 004737 035466      JSR      PC,TON        ;TURN T-BIT TRAPPING BACK ON
2867
2868 ;*****
2869 ;
2870 ; THE NEXT THREE (3) TESTS CAUSE MEMORY MANAGEMENT ERRORS
2871 ; TO CHECK THE ABILITY OF STATUS REGISTER 0 TO RECORD KT
2872 ; ERRORS AND THE ABILITY OF STATUS REGISTER 2 TO LOCK UP THE
2873 ; VIRTUAL ADDR. OF THE INSTRUCTION THAT CAUSED THE ERROR.
2874 ; THE BITS OF SR2 ARE CHECKED AND BITS <15:13>, <6:5>, AND <3:0>
2875 ; ARE CHECKED IN SR0. SO THE SR0 AND SR2 LOGIC AND THE
2876 ; KT ERROR LOGIC ARE CHECKED.
2877 ;
2878 ;*****
2879 ;
2880 ;*****
2881 ;TEST 33      NON-RESIDENT ABORT TEST (ACF=0&4)
2882 ;
2883 ; THIS TEST CHECKS THE ACCESS CONTROL FIELD (ACF) COMPARATOR
2884 ; LOGIC BY CAUSING NON-RESIDENT ABORTS IN BOTH KERNEL AND
2885 ; USER MODES. PDR 4 IS LOADED WITH ACF'S = 0&4 AND
2886 ; THEN PHYSICAL ADDR. 60000 IS ACCESSED TO CAUSE THE ABORT.
2887 ;
2888 ;*****
2889 026344 000004      †ST33: SCOPE
2890
2891 026346 012700 000600      1$:     MOV      #600,R0       ;LOAD DATA FOR PAR'S INTO R0
2892 026352 010037 172346      MOV      R0,KIPAR3      ;MAP KERNEL PAR'S 3&4 TO 12-16K
2893 026356 010037 172350      MOV      R0,KIPAR4
2894 026362 010037 177646      MOV      R0,UIPAR3      ;MAP USER PAR'S 3&4 TO 12-16K
2895 026366 010037 177650      MOV      R0,UIPAR4
2896 026372 012737 077406 172306      MOV      #77406,KIPDR3  ;MAP KERNEL PDR 3 128 BLKS, READ-WRITE
2897 026400 012737 077406 177606      MOV      #77406,UIPDR3  ;MAP USER PDR 3 128 BLKS, READ-WRITE
2898 026406 012700 060000      MOV      #60000,R0      ;LOAD VIRTUAL ADDR. TO REFERENCE PDR3 INTO R0
2899 026412 012701 100000      MOV      #100000,R1     ;LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
2900 026416 012703 100011      MOV      #100011,R3     ;LOAD R3 WITH WHAT SR0 SHOULD READ - N.R.  KERNEL, PG.4
2901 026422 012702 077400      MOV      #77400,R2     ;LOAD ACF=0 (NON-RESIDENT) PDR VALUE IN R2
2902 026426 012737 026470 000250      2$:     MOV      #5$,MMVEC      ;POINT MEM. MGMT. TRAP VECTOR TO 5$ BELOW
2903 026434 010237 172310      MOV      R2,KIPDR4     ;LOAD ACF TEST VALUE INTO KIPDR4
2904 026440 010237 177610      MOV      R2,UIPDR4     ;LOAD ACF TEST VALUE INTO UIPDR4
2905 026444 012737 026452 001110      MOV      #3$,SLPERR     ;SET LOOP ON ERROR POINTER TO 3$
2906 026452 005010      3$:     CLR      (R0)          ;CLEAR PHYS. LOC. 60000 USING PDR3
2907 026454 013737 177776 001176      MOV      PSW,$TMP0     ;SAVE PSW IN CASE OF ERROR
2908 026462 005211      4$:     INC      (R1)          ;TRY TO REF. IT USING PDR4 - SHOULD TRAP TO 5$
2909 026464 104026      ERROR    26            ;MEM. MGMT. ABORT DID NOT OCCUR

```

```

2910                                     ;FOR TIGHTER SCOPE LOOP
2911                                     ;REPLACE ERROR CALL WITH
2912                                     ;"BR 3$" = 000772
2913 026466 000425                                     BR      8$
2914 026470 062706 000004 5$: ADD      #4,SP
2915 026474 005710                                     TST     (R0)
2916 026476 001401                                     BEQ     6$
2917 026500 104027                                     ERROR   27
2918                                     ;FOR TIGHTER SCOPE LOOP
2919                                     ;REPLACE ERROR CALL WITH
2920                                     ;"BR 3$" = 000764
2921 026502 013737 177572 001362 6$: MOV     SRO,WASSRO
2922 026510 013737 177576 001364   MOV     SR2,WASSR2
2923 026516 020337 001362   CMP     R3,WASSRO
2924 026522 001401   BEQ     7$
2925 026524 104030   ERROR   30
2926                                     ;FOR TIGHTER SCOPE LOOP
2927                                     ;REPLACE ERROR CALL WITH
2928                                     ;"BR 3$" = 000752
2929 026526 012704 026462 7$: MOV     #4$,R4
2930 026532 020437 001364   CMP     R4,WASSR2
2931 026536 001401   BEQ     8$
2932 026540 104031   ERROR   31
2933                                     ;FOR TIGHTER SCOPE LOOP
2934                                     ;REPLACE ERROR CALL WITH
2935                                     ;"BR 3$" = 000744
2936 026542 042737 160000 177572 8$: BIC     #160000,SRO
2937 026550 032737 140000 001176   BIT     #140000,$TMP0
2938 026556 001006   BNE     9$
2939 026560 012703 100151   MOV     #100151,R3
2940 026564 012737 140000 177776   MOV     #140000,PSW
2941 026572 000715   BR      2$
2942 026574 022702 077404 9$: CMP     #77404,R2
2943 026600 001407   BEQ     10$
2944 026602 012702 077404   MOV     #77404,R2
2945 026606 012703 100011   MOV     #100011,R3
2946 026612 005037 177776   CLR     PSW
2947 026616 000703   BR      2$
2948 026620 005037 177776 10$: CLR     PSW
2949 026624 012737 026346 001110   MOV     #1$, $LPERR
2950 026632 012737 002314 000250   MOV     #MGMERR,MMVEC
2951
2952
2953                                     ;*****
2954                                     ;TEST 34 READ-ONLY ABORT TEST (ACF=2)
2955                                     ;
2956                                     ; THIS TEST CHECKS THE ACCESS CONTROL FIELD (ACF) COMPARATOR
2957                                     ; LOGIC BY CAUSING READ-ONLY ABORTS IN BOTH KERNEL AND
2958                                     ; USER MODES. PDR 4 IS LOAD WITH ACF=2 AND THEN
2959                                     ; PHYSICAL ADDR. 60000 IS WRITTEN TO CAUSE THE ABORT.
2960                                     ;
2961                                     ;*****
2962 026640 000004 1$ST34: SCOPE
2963 026642
2964
2965 026642 012700 060000   MOV     #60000,R0

```

```

;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 3$" = 000772
;BRANCH AROUND STATUS REG. CHECKS IF NO ABORT
;RESTORE STACK POINTER
;DID INSTRUCTION GET ABORTED & NOT EXECUTE
;BRANCH IF YES
;INSTRUCTION WAS NOT ABORTED, LOC. GOT CHANGED
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 3$" = 000764
;READ STATUS REGISTER 0
;READ STATUS REGISTER 2
;DID SRO REPORT NON-RESIDENT ERROR CORRECTLY?
;BRANCH IF YES
;SRO DID NOT REPORT NON-RES. ERROR CORRECTLY
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 3$" = 000752
;LOAD R4 WITH WHAT SR2 SHOULD READ
;DID SR2 LOCKUP RIGHT VIRTUAL ADDR. (=4$)?
;BRANCH IF YES
;SR2 DID NOT LOCK VIRTUAL ADDR. OF NON-RES. ERROR
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 3$" = 000744
;CLEAR THE ERROR BITS IN SRO
;HAS ACF=0&4 BEEN TESTED IN USER YET
;BRANCH IF YES
;LOAD R3 WITH WHAT SRO SHOULD READ - N.R., USER, PG.4
;GO TO USER MODE
;REPEAT TEST IN USER MODE
;HAS ACF=4 BEEN TESTED YET?
;BRANCH IF YES
;THEN LOAD ACF=4 (NON-RES) PDR VALUE IN R2
;LOAD R3 WITH WHAT SRO SHOULD READ-N.R., KERNEL, PG. 4
;GO BACK TO KERNEL MODE
;GO BACK & TEST ACF=4 IN SAME MODE
;GO BACK TO KERNEL MODE BEFORE LEAVING
;RESET LOOP ON ERROR POINTER TO 1$
;RESTORE ADDRESS OF NORMAL MEMORY
;MANAGEMENT ERROR ROUTINE TO MMVEC

```

```

;KERNEL & USER PAR'S 3 & 4 AND PDR 3
;ARE SETUP FROM LAST TEST
;LOAD VIRTUAL ADDR. TO REFERENCE PDR3 INTO R0

```


L05

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.F.1 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 63
T35 TEST ILLEGAL MODE "01"

```

3022 ;*
3023 ;*****
3024 027050 000004          15: SCOPE
3025 027052 012737 027066 001110 15: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
3026 027060 012737 027102 000250 15: MOV #3$, MMVEC ;LOAD MEM. MGMT. TRAP VECTOR WITH 3$
3027 027066 012737 040000 177776 25: MOV #40000, PSW ;SET 01 IN PSW CURRENT MODE BITS
3028 027074 000240          25: NOP ;FETCH OF THIS INSTRUCTION SHOULD CAUSE ABORT
3029 027076 104056          25: ERROR 56 ;ILLEGAL MODE 01 NOT ABORTED
3030 ;FOR A TIGHTER SCOPE LOOP
3031 ;REPLACE ERROR CALL WITH
3032 ;"BR 25" = 000773
3033 027100 000424          35: BR 5$ ;BRANCH AROUND SRO & SR2 CHECKS
3034 027102 012706 001100 35: MOV #KERSTK, SP ;RESTORE STACK POINTER
3035 027106 012701 100043 35: MOV #100043, R1 ;LOAD EXPECTED CONTENTS OR SRO INTO R1
3036 027112 013737 177572 001362 35: MOV SRO, WASSRO ;READ CONTENTS OF SRO
3037
3038 027120 020137 001362          45: CMP R1, WASSRO ;DID SRO REPORT ILLEGAL MODE CORRECTLY?
3039 027124 001401          45: BEQ 4$ ;BRANCH IF YES
3040 027126 104057          45: ERROR 57 ;SRO DID NOT REPORT NR ABORT, PG=1, MODE=01
3041 ;FOR TIGHTER SCOPE LOOP
3042 ;REPLACE ERROR CALL WITH
3043 ;"BR 25" = 000757
3044 027130 012701 027066 001364 45: MOV #2$, R1 ;LOAD EXPECTED CONTENTS OF SR2 INTO R1
3045 027134 013737 177576 001364 45: MOV SR2, WASSR2 ;READ CONTENTS OF SR2
3046 027142 020137 001364          45: CMP R1, WASSR2 ;DID SR2 LOCKUP VIRT. ADDR OF ABORTED INST.
3047 027146 001401          45: BEQ 5$ ;BRANCH IF YES
3048 027150 104036          45: ERROR 36 ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ILL. MODE INST.
3049 ;FOR TIGHTER SCOPE LOOP
3050 ;REPLACE ERROR CALL WITH
3051 ;"BR 25" = 000746
3052 027152 042737 160000 177572 55: BIC #160000, SRO ;CLEAR POSSIBLE ERROR BITS IN SRO
3053 027160 012737 002314 000250 55: MOV #MGMERR, MMVEC ;RESTORE MEM. MGMT. ABORT VECTOR
3054 027166 012737 027052 001110 55: MOV #15, $LPERR ;RESET LOOP ON ERROR POINTER TO 15
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068 ;*****
3069 ;
3070 ; THE NEXT TWO (2) TESTS WILL BE CHECKING THE PAGE LENGTH
3071 ; COMPARATORS AND SOME MORE OF THE KT ERROR DETECTION
3072 ; AND STATUS LOGIC. THE PAGE LENGTH FIELD (PLF) IN KERNEL
3073 ; PDR 4 IS VARIED AND FOR EVERY PLF, THREE (3) VIRTUAL
3074 ; ADDRESSES ARE READ. WHILE USING BOTH UPWARD & DOWNWARD PAGE
3075 ; EXPANSION, ONE OF THOSE THREE VIRTUAL ADDRESSES WILL CAUSE A
3076 ; "PAGE LENGTH ABORT" WHILE THE OTHER TWO WON'T.
3077 ;
3078 ; STATUS REGISTER 0 & 2 ARE CHECKED WHEN THE PAGE LENGTH

```


M05

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 64
T35 TEST ILLEGAL MODE "01"

3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133

027174	000004		
027176	012737	077406	172306
027204	012737	077406	172312
027212	012700	100000	
027216	012704	000406	
027222	012737	027414	000250
027230	010437	172310	
027234	012737	027242	001110
027242	012706	001100	
027246	011001		
027250	062700	000100	
027254	012737	027262	001110
027262	012706	001100	
027266	011001		
027270	062700	000100	
027274	020027	117700	
027300	001470		
027302	012737	027316	001110
027310	012737	027324	000250
027316	011001		
027320	104033		
027322	000424		
027324	012706	001100	
027330	013737	177572	001362
027336	013737	177576	001364
027344	012702	040011	
027350	020237	001362	
027354	001401		
027356	104034		
027360	012703	027316	

```

: * ABORT DOES OCCUR TO SEE THAT THE ABORT IS REPORTED AND THAT
: * THE VIRTUAL ADDRESS OF THE INSTRUCTION THAT CAUSED THE ABORT
: * IS LOCKED UP.
: *
: *****
: *****
: *TEST 36 PAGE LENGTH FAULTS-UPWARD EXPANSION
: *
: THIS TEST VARIES THE PAGE LENGTH FIELD (PLF) IN KERNEL PDR 4
: FROM 1 TO 177 AND FOR EACH PLF, THREE VIRTUAL ADDRESSES (VBA'S)
: ARE ACCESSED. WHEN VBA <12:6> IS LESS THAN OR EQUAL TO PDR <14:8>
: NO ABORT SHOULD OCCUR. WHEN VBA <12:6> IS GREATER THAN PDR <14:8>,
: A PAGE LENGTH ABORT SHOULD OCCUR AND BE REPORTED BY SRO & SR2.
: THE PAGE EXPANSION DIRECTION IN THIS TEST IS UPWARD, (THE ED BIT
: (BIT 3) OF PDR 4 = 0).
: *****
: *ST36: SCOPE
1$: MOV #77406,KIPDR3 ;MAKE SURE PDR3 IS DESCRIBED AS R/W
MOV #77406,KIPDR5 ;MAKE SURE PDR5 IS DESCRIBED AS R/W
MOV #100000,R0 ;LOAD VIRTUAL ADDR. TO SELECT PDR4 INTO R0
MOV #406,R4 ;LOAD FIRST PDR VALUE IN R4 (PLF=1, ACF=6)
2$: MOV #9$,MMVEC ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS
MOV R4,KIPDR4 ;LOAD KIPDR4 WITH PAGE LENGTH VALUE
MOV #3$,SLPERR ;SET LOOP ON ERROR POINTER 3 TO 3$
3$: MOV #KERSTK,KSP ;MAKE SURE STACK POINTER IS ALL SET UP
MOV (R0),R1 ;ACCESS VIRTUAL ADDR. (VBA < PLF - NO ABORT)
ADD #100,R0 ;FORM NEXT VIRTUAL ADDRESS IN R0
MOV #4$,SLPERR ;SET LOOP ON ERROR POINTER TO 4$
4$: MOV #KERSTK,KSP ;MAKE SURE STACK POINTER IS ALL SET UP
MOV (R0),R1 ;ACCESS VIRTUAL ADDR. (VBA=PLF - NO ABORT)
ADD #100,R0 ;FORM NEXT VIRTUAL ADDR IN R0
CMP R0,#117700 ;HAVE ALL PLF'S BEEN TESTED YET?
BEQ 10$ ;BRANCH IF ALL VBA'S & PLF'S HAVE BEEN USED
MOV #5$,SLPERR ;SET LOOP ON ERROR POINTER TO 5$
5$: MOV #6$,MMVEC ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT
MOV (R0),R1 ;ACCESS VIRTUAL ADDR. (VBA > PLF - ABORT TO 6$)
ERROR 33 ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 5$" = 000776
6$: BR 8$ ;BRANCH AROUND ABORT CHECKS
MOV #KERSTK,KSP ;RESTORE STACK POINTER FOLLOWING ABORT
MOV SRO,WASSRO ;READ M.M. STATUS REG. 0
MOV SR2,WASSR2 ;READ M.M. STATUS REG. 2
MOV #40011,R2 ;PUT EXPECTED SRO CONTENTS IN R2
CMP R2,WASSRO ;DID SRO REPORT PG. LENGTH ABORT, PAGE 4, KERNEL?
BEQ 7$ ;BRANCH IF YES
7$: ERROR 34 ;SRO DID NOT REPORT PG. LENGTH ABORT CORRECTLY
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 5$" = 000757
MOV #5$,R3 ;PUT EXPECTED SR2 CONTENTS IN R3

```

N05

MD-11-DFKTH-A POP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 65
T36 PAGE LENGTH FAULTS-UPWARD EXPANSION

3134	027364	020337	001364			CMP	R3, WASSR2	: DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?
3135	027370	001401				BEQ	8\$: BRANCH IF YES
3136	027372	104035				ERROR	35	: SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY
3137								: FOR TIGHTER SCOPE LOOP
3138								: REPLACE ERROR CALL WITH
3139								: "BR 5\$" = 000751
3140	027374	042737	160000	177572	8\$:	BIC	#160000, SRO	: CLEAR ERROR BITS IN SRO
3141	027402	062704	000400			ADD	#400, R4	: FORM NEXT PLF VALUE FOR KIPDR4
3142	027406	162700	000100			SUB	#100, R0	: FORM FIRST VIRT. ADDR FOR THAT PLF VALUE
3143	027412	000703				BR	2\$: BRANCH BACK AND ACCESS 3 VBA'S FOR
3144								: THE PLF VALUE JUST FORMED
3145	027414	012637	001356		9\$:	MOV	(KSP)+, TRAPPC	: SAVE PC & PS OF TRAP
3146	027420	012637	001360			MOV	(KSP)+, TRAPPS	
3147	027424	013737	177572	001362		MOV	SRO, WASSRO	: SAVE CONTENTS OF SRO FOR ERROR
3148	027432	013737	177576	001364		MOV	SR2, WASSR2	: SAVE CONTENTS OF SR2 FOR ERROR
3149	027440	042737	160000	177572		BIC	#160000, SRO	: CLEAR ERROR BITS IN SRO
3150	027446	104032				ERROR	32	: GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
3151								: FOR TIGHTER SCOPE LOOP
3152								: REPLACE ERROR CALL WITH
3153								: A "NOP" = 000240
3154	027450	013746	001360			MOV	TRAPPS, -(KSP)	: PUT PC & PS OF TRAP ON STACK
3155	027454	013746	001356			MOV	TRAPPC, -(KSP)	
3156	027460	000002				RTI		: RETURN FROM UNEXPECTED ABORT
3157								
3158	027462	012737	027176	001110	10\$:	MOV	#1\$, \$LPERR	: RESET LOOP ON ERROR POINTER TO 1\$
3159	027470	012737	002314	000250		MOV	#MGHERR, MMVEC	: RESTORE NORMAL M.M. TRAP HANDLER
3160								: ADDRESS TO M.M. TRAP VECTOR

```

*****
: TEST 37 PAGE LENGTH FAULTS-DOWNWARD EXPANSION
:
: THIS TEST VARIES THE PAGE LENGTH FIELD (PLF) IN KERNEL PDR4
: FROM 176 TO 0 AND FOR EACH PLF, THREE VIRTUAL ADDRESSES (VBA'S)
: ARE ACCESSED. WHEN VBA <12:6> IS GREATER THAN OR EQUAL TO PDR <14:8>
: NO PAGE ABORT SHOULD OCCUR. WHEN VBA <12:6> IS LESS THAN PDR <14:8>
: A PAGE LENGTH ABORT SHOULD OCCUR AND BE REPORTED BY SRO & SR2.
: THE PAGE EXPANSION DIRECTION IN THIS TEST IS DOWNWARD, (THE ED BIT
: (BIT 3) OF PDR4=1).
*****

```

3175	027476	000004				TEST37:	SCOPE	
3176	027500	012700	117700		1\$:	MOV	#117700, R0	: LOAD VIRTUAL ADDR. TO SELECT PDR4 INTO R0
3177	027504	012704	077016			MOV	#77016, R4	: LOAD FIRST PDR VALUE IN R4 (PLF=176, ACF=6)
3178	027510	012737	027702	000250	2\$:	MOV	#9\$, MMVEC	: SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS
3179	027516	010437	172310			MOV	R4, KIPDR4	: LOAD KIPDR4 WITH PAGE LENGTH VALUE
3180	027522	012737	027530	001110		MOV	#3\$, \$LPERR	: SET LOOP ON ERROR POINTER TO 3\$
3181	027530	012706	001100		3\$:	MOV	#KERSTK, KSP	: MAKE SURE STACK POINTER IS ALL SET UP
3182	027534	011001				MOV	(R0), R1	: ACCESS VIRTUAL ADDR. (VBA > PLF - NO ABORT)
3183	027536	162700	000100			SUB	#100, R0	: FORM NEXT VIRTUAL ADDRESS IN R0
3184	027542	012737	027550	001110		MOV	#4\$, \$LPERR	: SET LOOP ON ERROR POINTER TO 4\$
3185	027550	012706	001100		4\$:	MOV	#KERSTK, KSP	: MAKE SURE STACK POINTER IS ALL SET UP
3186	027554	011001				MOV	(R0), R1	: ACCESS VIRTUAL ADDR. (VBA=PLF - NO ABORT)
3187	027556	162700	000100			SUB	#100, R0	: FORM NEXT VIRTUAL ADDR. IN R0
3188	027562	020027	100000			CMP	R0, #100000	: HAVE ALL PLF'S BEEN TESTED YET?
3189	027566	001470				BEQ	10\$: BRANCH IF ALL VBA'S & PLF'S HAVE BEEN USED

3190	027570	012737	027604	001110		MOV	#5\$,SLPERR	;SET LOOP ON ERROR POINTER TO 5\$
3191	027576	012737	027612	000250		MOV	#6\$,MMVEC	;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT
3192	027604	011001			5\$:	MOV	(R0),R1	;ACCESS VIRTUAL ADDR. (VBA < PLF - ABORT TO 6\$)
3193	027606	104033				ERROR	33	;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
3194								;FOR TIGHTER SCOPE LOOP
3195								;REPLACE ERROR CALL WITH
3196								"BR 5\$" = 000776
3197	027610	000424				BR	8\$;BRANCH AROUND ABORT CHECKS
3198	027612	012706	001100		6\$:	MOV	#KERSTK,KSP	;RESTORE STACK POINTER FOLLOWING ABORT
3199	027616	013737	177572	001362		MOV	SRO,WASSRO	;READ M.M. STATUS REG. 0
3200	027624	013737	177576	001364		MOV	SR2,WASSR2	;READ M.M. STATUS REG. 2
3201	027632	012702	040011			MOV	#40011,R2	;PUT EXPECTED SRO CONTENTS IN R2
3202	027636	020237	001362			CMP	R2,WASSRO	;DID SRO REPORT PG. LENGTH ABORT, PG. 4, KERNEL?
3203	027642	001401				BEQ	7\$;BRANCH IF YES
3204	027644	104034				ERROR	34	;SRO DID NOT REPORT PG. LENGTH ABORT CORRECTLY
3205								;FOR TIGHTER SCOPE LOOP
3206								;REPLACE ERROR CALL WITH
3207								"BR 5\$" = 000757
3208	027646	012703	027604		7\$:	MOV	#5\$,R3	;PUT EXPECTED SR2 CONTENTS IN R3
3209	027652	020337	001364			CMP	R3,WASSR2	;DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?
3210	027656	001401				BEQ	8\$;BRANCH IF YES
3211	027660	104035				ERROR	35	;SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY
3212								;FOR TIGHTER SCOPE LOOP
3213								;REPLACE ERROR CALL WITH
3214								"BR 5\$" = 000751
3215	027662	042737	160000	177572	8\$:	BIC	#160000,SRO	;CLEAR ERROR BITS IN SRO
3216	027670	162704	000400			SUB	#400,R4	;FORM NEXT PLF VALUE FOR KIPDR4
3217	027674	062700	000100			ADD	#100,R0	;FORM FIRST VIRT. ADDR. FOR THAT PLF VALUE
3218	027700	000703				BR	2\$;BRANCH BACK AND ACCESS 3 VBA'S FOR
3219								;THE PLF VALUE JUST FORMED
3220	027702	012637	001356		9\$:	MOV	(KSP)+,TRAPPC	;SAVE PC & PS OF TRAP
3221	027706	012637	001360			MOV	(KSP)+,TRAPPS	
3222	027712	013737	177572	001362		MOV	SRO,WASSRO	;SAVE CONTENTS OF SRO FOR ERROR
3223	027720	013737	177576	001364		MOV	SR2,WASSR2	;SAVE CONTENTS OF SR2 FOR ERROR
3224	027726	042737	160000	177572		BIC	#160000,SRO	;CLEAR ERROR BITS IN SRO
3225	027734	104032				ERROR	32	;GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
3226								;FOR TIGHTER SCOPE LOOP
3227								;REPLACE ERROR CALL WITH
3228								A "NOP" = 000240
3229	027736	013746	001360			MOV	TRAPPS,-(KSP)	;PUT PC & PS OF TRAP ON STACK
3230	027742	013746	001356			MOV	TRAPPC,-(KSP)	
3231	027746	000002				RTI		;RETURN FROM UNEXPECTED ABORT
3232								
3233	027750	012737	027500	001110	10\$:	MOV	#1\$,SLPERR	;RESET LOOP ON ERROR POINTER TO 1\$
3234	027756	012737	002314	000250		MOV	#MGERR,MMVEC	;RESTORE NORMAL M.M. TRAP HANDLER
3235								;ADDRESS TO M.M. TRAP VECTOR
3236								
3237								
3238								
3239								
3240								
3241								
3242								
3243								
3244								
3245								

```

*****
;TEST 40      SR2 BIT TEST
;
; THIS TEST CHECKS THE BITS IN MEMORY MANAGEMENT REGISTER 2 BY
; CAUSING "READ-ONLY ABORTS" AT VIRTUAL ADDRESSES BETWEEN 10000
; TO 111000 (PHYSICAL ADDRESSES 060000-071000). KIPDR4 IS USED TO EXECUTE
; THE FOLLOWING FOUR WORDS OF CODE WHICH ARE MOVED THRU MEMORY:
;
; 010727 MOV      PC,(PC)+ ;THIS INSTRUCTION SHOULD CAUSE A R/O ABORT

```

```

3246
3247
3248
3249
3250
3251
3252 027764 000004
3253 027766 012737 000600 172346
3254 027774 012737 000600 172350
3255 030002 012737 077406 172306
3256 030010 012737 077402 172310
3257 030016 012700 060000
3258 030022 012701 100000
3259 030026 012737 030062 000250
3260 030034 012737 030042 001110
3261 030042 012720 010727
3262 030046 005020
3263 030050 012720 000137
3264 030054 012710 030062
3265 030060 010107
3266 030062 012706 001100
3267 030066 013737 177576 001364
3268 030074 020137 001364
3269 030100 001401
3270 030102 104036
3271
3272
3273
3274 030104 042737 160000 177572
3275 030112 162700 000004
3276 030116 062701 000002
3277 030122 020127 111002
3278 030126 103745
3279
3280 030130 012737 027766 001110
3281 030136 012737 077406 172310
3282 030144 012737 002314 000250
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300 030152 000004
3301 030154 012737 000600 172352

```

```

* 000000 ;ITS VIRTUAL ADDR. SHOULD BE LOCKED UP IN SR2
* 070137 JMP 2#3$ ;THIS INSTRUCTION IS ALSO MOVED THRU MEMORY
* (ADDR. OF 3$) ;IN CASE A R/O ABORT DOES NOT OCCUR,
* ;IN WHICH CASE SR2 WILL NOT CONTAIN CORRECT ADDR.
*****
TST40: SCOPE
1$: MOV #600,KIPAR3 ;BE SURE PAR3 IS MAPPED TO 12-16K
MOV #600,KIPAR4 ;BE SURE PAR4 IS MAPPED TO 12-16K
MOV #77406,KIPDR3 ;MAP PAGE 3 128 BLOCKS, R/W
MOV #77402,KIPDR4 ;MAP PAGE 4 128 BLOCKS, READ-ONLY
MOV #60000,R0 ;LOAD R0 WITH VIRTUAL ADDR. WHICH USES PDR3
MOV #100000,R1 ;LOAD R1 WITH VIRTUAL ADDR. WHICH USES PDR4
MOV #3$,MMVEC ;SET M.M. TRAP VECTOR TO 3$
MOV #2$,SLPERR ;SET LOOP ON ERROR POINTER TO 2$
2$: MOV #010727,(R0)+ ;LOAD "MOV PC,(PC)+" INSTRUCTION AT ADDR.
CLR (R0)+ ;REACHED THRU PDR/PAR 4.
MOV #000137,(R0)+ ;LOAD "JMP 2#3$" INSTRUCTION AT VIRT. ADDR.
MOV #3$,(R0) ;IN CASE R/O VIOL. DOES NOT ABORT
MOV R1,PC ;TRANSFER PROGRAM EXECUTION TO "PAGE 4 INSTRUCTIONS"
3$: MOV #KERSTK,KSP ;RESTORE STACK POINTER
MOV SR2,WASSR2 ;READ CONTENTS OF STATUS REG 2
CMP R1,WASSR2 ;WAS ADDR. OF "RELOCATED - R/O ABORT" LOCKED UP?
BEQ 4$ ;BRANCH IF YES
ERROR 36 ;SR2 DID NOT LOCK UP VIRTUAL ADDR. OF R/O VIOL.
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 2$" = 000757
4$: BIC #160000,SR0 ;CLEAR THE ERROR BITS IN SR0
SUB #4,R0 ;RESET R0 TO POINT TO NEXT VIRT. ADDR. TO LOAD
ADD #2,R1 ;FORM VIRTUAL ADDR. THAT SHOULD BE LOCKED UP NEXT
CMP R1,#111002 ;HAVE ALL VBA'S 100000-111000 BEEN TESTED?
BLO 2$ ;BRANCH IF NO
5$: MOV #1$,SLPERR ;RESET LOOP ON ERROR POINTER TO 1$
MOV #77406,KIPDR4 ;RESTORE PDR4 TO R/W ACCESS
MOV #MGMERR,MMVEC ;RESTORE ADDRESS OF NORMAL M.M.
;TRAP HANDLER TO M.M. VECTOR
*****
*TEST 41 MORE CHECKS OF SRO & SR2
*
* THIS TEST PERFORMS SOME ADDITIONAL CHECKS OF THE SRO & SR2 LOGIC.
* FIRST IT CHECKS THAT SR2 "TRACKS" ALONG ACTING AS A VIRTUAL ADDRESS
* PROGRAM COUNTER. ALSO SRO & SR2 ARE LOCKED UP BY A PAGE LENGTH
* ABORT, THEN WITHOUT CLEARING SRO'S ERROR BITS, A R/O ABORT IS CAUSED.
* SRO & SR2 SHOULD NOT BE CHANGED BY THE SECOND ABORT AND THE
* INFORMATION ABOUT THE PAGE LENGTH ABORT SHOULD STILL BE LOCKED UP.
* IN ADDITION A "RESET" IS EXECUTED TO VERIFY THAT SRO IS CLEARED
* AND SR2 IS UNLOCKED BY A RESET. AFTER MEMORY MANAGEMENT IS TURNED BACK ON,
* SR2 IS CHECKED TO SEE THAT IT IS TRACKING AGAIN.
*****
TST41: SCOPE
1$: MOV #600,KIPARS ;MAP KERNEL PAGE 5 TO 12-16K

```

3302	030162	012737	000406	172310		MOV	#406,KIPDR4	:SETUP PDR4 FOR PAGE LENGTH ABORT
3303	030170	012737	077402	172312		MOV	#77402,KIPDR5	:SETUP PDR5 FOR R/O ABORT
3304	030176	012737	030204	001110		MOV	#2\$,SLPERR	:SET LOOP ON ERROR POINTER TO 2\$
3305	030204	013737	177576	001364	2\$:	MOV	SR2,WASSR2	:READ SR2 TO SEE IF ITS TRACKING
3306	030212	012701	030204			MOV	#2\$,R1	:PUT EXPECTED VIRTUAL PC IN R1
3307	030216	020137	001364			CMP	R1,WASSR2	:DID SR2 CONTAIN VIRTUAL PC AT 2\$?
3308	030222	001401				BEQ	3\$:BRANCH IF YES
3309	030224	104041				ERROR	41	:SR2 NOT TRACKING CORRECTLY
3310								:FOR TIGHTER SCOPE LOOP
3311								:REPLACE ERROR CALL WITH
3312								"BR 2\$" = 000767
3313	030226	012737	030234	001110	3\$:	MOV	#4\$,SLPERR	:SET LOOP ON ERROR POINTER TO 4\$
3314	030234	013737	177576	001364	4\$:	MOV	SR2,WASSR2	:READ SR2 TO SEE IF ITS TRACKING
3315	030242	012701	030234			MOV	#4\$,R1	:PUT EXPECTED VIRTUAL PC IN R1
3316	030246	020137	001364			CMP	R1,WASSR2	:DID SR2 CONTAIN VIRTUAL PC AT 4\$
3317	030252	001401				BEQ	5\$:BRANCH IF YES
3318	030254	104041				ERROR	41	:SR2 NOT TRACKING CORRECTLY
3319								:FOR TIGHTER SCOPE LOOP
3320								:REPLACE ERROR CALL WITH
3321								"BR 4\$" = 000767
3322	030256	012737	030264	001110	5\$:	MOV	#6\$,SLPERR	:SET LOOP ON ERROR POINTER TO 6\$
3323	030264	012737	030302	000250	6\$:	MOV	#7\$,MMVEC	:PUT ADDRESS OF 7\$ IN M.M. TRAP VECTOR
3324	030272	005037	001200			CLR	STMP1	:CLEAR ERROR INDICATOR
3325	030276	005237	100500			INC	#100500	:CAUSE PAGE LENGTH ABORT - TRAP TO 7\$
3326	030302	012706	001100		7\$:	MOV	#KERSTK,KSP	:RESTORE STACK POINTER AFTER ABORT
3327	030306	013737	177572	001176		MOV	SR0,STMP0	:SAVE SR0'S INFORMATION ON PG. LGTH. ABORT
3328	030314	013737	177576	001202		MOV	SR2,STMP2	:SAVE SR2'S INFORMATION ON PG. LGTH. ABORT
3329	030322	012737	030334	000250		MOV	#8\$,MMVEC	:PUT ADDRESS OF 8\$ IN M.M. TRAP VECTOR
3330	030330	005237	120000			INC	#120000	:CAUSE R/O ABORT - TRAP TO 8\$
3331	030334	012706	001100		8\$:	MOV	#KERSTK,KSP	:RESTORE STACK POINTER AFTER ABORT
3332	030340	013737	177572	001362		MOV	SR0,WASSR0	:READ SR0 FOLLOWING SECOND KT ABORT
3333	030346	013737	177576	001364		MOV	SR2,WASSR2	:READ SR2 FOLLOWING SECOND KT ABORT
3334	030354	023737	001176	001362		CMP	STMP0,WASSR0	:IS SR0 STILL HOLDING INFO ON FIRST ABORT?
3335	030362	001402				BEQ	9\$:BRANCH IF YES
3336	030364	005237	001200			INC	STMP1	:SET ERROR INDICATOR
3337	030370	023737	001202	001364	9\$:	CMP	STMP2,WASSR2	:DOES SR2 STILL HOLD PC OF FIRST ABORT?
3338	030376	001402				BEQ	10\$:BRANCH IF YES
3339	030400	005237	001200			INC	STMP1	:SET ERROR INDICATOR
3340	030404	005737	001200		10\$:	TST	STMP1	:WERE SR0 OR SR2 CHANGED BY A SECOND ABORT?
3341	030410	001401				BEQ	11\$:BRANCH IF NO
3342	030412	104037				ERROR	37	:ONE OF STATUS REGS. CHANGED BY SECOND ABORT
3343								:FOR TIGHTER SCOPE LOOP
3344								:REPLACE ERROR CALL WITH
3345								"BR 6\$" = 000726
3346	030414	005037	001200		11\$:	CLR	STMP1	:CLEAR ERROR INDICATOR
3347	030420	000005				RESET		:EXECUTE A RESET, APPLYING AN "INIT"
3348	030422	013737	177572	001362		MOV	SR0,WASSR0	:READ SR0
3349	030430	005737	001362			TST	WASSR0	:WAS SR0 CLEARED BY THE RESET?
3350	030434	001402				BEQ	12\$:BRANCH IF YES
3351	030436	005237	001200			INC	STMP1	:SR0 NOT CLEARED BY A RESET
3352	030442	013737	177576	001364	12\$:	MOV	SR2,WASSR2	:READ SR2
3353	030450	022737	030442	001364		CMP	#12\$,WASSR2	:WAS SR2 UNLOCKED BY A RESET?
3354	030456	001402				BEQ	13\$:BRANCH IF YES
3355	030460	005237	001200			INC	STMP1	:SR2 NOT UNLOCKED BY A RESET
3356	030464	005737	001200		13\$:	TST	STMP1	:WERE SR0 & SR2 BOTH "RESET" BY A RESET?
3357	030470	001401				BEQ	14\$:BRANCH IF YES

```

3358 030472 104040
3359
3360
3361
3362 030474 005237 177572
3363 030500 013737 177576 001364
3364 030506 012701 030500
3365 030512 020137 001364
3366 030516 001401
3367 030520 104041
3368
3369
3370
3371 030522 012737 030154 001110
3372 030530 012737 077406 172310
3373 030536 012737 077406 172312
3374 030544 012737 002314 000250
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390 030552 000004
3391 030554 004737 035432
3392 030560 012737 030566 001110
3393 030566 005037 177776
3394 030572 012706 001100
3395 030576 012737 000600 177640
3396 030604 012737 030666 000004
3397 030612 012737 000340 000006
3398 030620 012737 140000 177776
3399 030626 012706 000700
3400 030632 012737 030652 000004
3401 030640 012737 000340 000006
3402 030646 005737 030653
3403
3404
3405 030652 013701 177776
3406 030656 010602
3407 030660 005037 177776
3408 030664 104042
3409

```

```

ERROR 40
14$: INC SRO
15$: MOV SR2,WASSR2
MOV #15$,R1
CMP R1,WASSR2
BEQ 16$
ERROR 41
16$: MOV #1$,SLPERR
MOV #77406,KIPDR4
MOV #77406,KIPDR5
MOV #MGMERR,MMVEC

```

```

;SRO OR SR2 NOT "RESET" BY A RESET
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 6$" = 000676
;TURN MEMORY MANAGEMENT BACK ON
;READ SR2 TO SEE IF ITS TRACKING AGAIN
;PUT EXPECTED VIRTUAL PC IN R1
;DID SR2 CONTAIN VIRTUAL PC AT 15$
;BRANCH IF YES
;SR2 NOT TRACKING CORRECTLY
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 6$" = 000663
;RESET LOOP ON ERROR POINTER TO 1$
;RESET PDR4 TO 128 BLKS, R/W
;RESET PDR5 TO 128 BLKS, R/W
;RESTORE ADDRESS OF NORMAL MEMORY
;MANAGEMENT TRAP ROUTINE TO M.M. VECTOR

```

```

*****
;TEST 42 USER ABORT PICKS UP KERNEL SPACE VECTOR
;
; THIS TEST CHECKS TO BE SURE THAT WHEN AN ABORT OCCURS WHILE IN
; USER MODE, THE TRAP VECTOR INFORMATION FETCHED IS TAKEN FROM
; KERNEL SPACE. USER PAGE 0 IS MAPPED TO 12K (60000-77776) SO
; THAT IF USER SPACE IS USED INSTEAD OF KERNEL, THE NEW PC THAT
; WAS LOADED AT LOC. 060004 IS USED INSTEAD OF THE NEW PC THAT
; SHOULD BE PICKED UP FROM LOC. 000004. AN ODD ADDRESS ERROR IS USED
; TO CAUSE A TRAP TO "4".
*****

```

```

*****
TST42: SCOPE
1$: JSR PC,TOFF ;TURN OFF T-BIT TRAPPING FOR THIS TEST
MOV #2$,SLPERR ;SET LOOP ON ERROR POINTER TO 2$
2$: CLR PSW ;GO TO KERNEL MODE
MOV #KERSTK,KSP ;SETUP KERNEL STACK PTR.
MOV #600,UIPAR0 ;MAP USER PAGE 0 TO 12K
MOV #4$,2#4 ;LOAD KERNEL VECTOR 4 (LOC.4) WITH 4$
MOV #340,2#6 ;LOAD VECTOR+2 WITH NEW PSW
MOV #140000,PSW ;GO TO USER MODE
MOV #USESTK,USP ;SETUP USER STACK PTR.
MOV #3$,2#4 ;LOAD USER VECTOR 4 (LOC. 60004) WITH 3$
MOV #340,2#6 ;LOAD VECTOR+2 WITH NEW PSW
3$: TST 3$+1 ;CAUSE ODD ADDR. ERROR TRAP TO "4"
;SHOULD PICK UP NEW PC=4$ FROM KERNEL
;LOC. 4, NOT PC=3$ FROM USER LOC. 4 (=60004)
MOV PSW,R1 ;SAVE PSW FOR ERROR
MOV SP,R2 ;SAVE VALUE OF STACK POINTER FOR ERROR
CLR PSW ;BE SURE BACK IN KERNEL MODE
ERROR 42 ;DID NOT TRAP THRU KERNEL SPACE
;FOR TIGHTER SCOPE LOOP

```

F06

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 70
T42 USER ABORT PICKS UP KERNEL SPACE VECTOR

```

3410                                     ;REPLACE ERROR CALL WITH
3411                                     ;"BR 2$" = 000740
3412 030666 005037 177776 4$: CLR PSW ;BE SURE BACK IN KERNEL MODE
3413 030672 012706 001100 MOV #KERSTK,KSP ;RESTORE KERNEL S.P. IN CASE IT CHANGED
3414 030676 005037 177640 CLR UIPARO ;REMAP USER PAGE 0 TO 0-4K
3415 030702 012737 140000 177776 MOV #140000,PSW ;GO TO USER MODE
3416 030710 012706 000700 MOV #USESTK,USP ;RESTORE USER STACK POINTER
3417 030714 005037 177776 CLR PSW ;GO BACK TO KERNEL MODE
3418 030720 012737 002246 000004 MOV #TIMERR,2#4 ;RESTORE ADDR. OF NORMAL CPU TRAP HANDLER TO 4
3419 030726 012737 030554 001110 MOV #1$,SLPERR ;RESET LOOP ON ERROR POINTER TO 1$
3420 030734 004737 035466 JSR PC,TON ;TURN T-BIT TRAPPING BACK ON
3421
3422 ;*****
3423 ;TEST 43 RTI IN USER MODE DOES NOT CHANGE PSW
3424 ;*
3425 ;* THIS TEST CHECKS TO SEE THAT WHEN AN RTI IS EXECUTED IN USER
3426 ;* MODE, THE MODE OR PRIORITY BITS OF THE PSW ARE NOT CHANGED.
3427 ;*
3428 ;*****
3429 030740 000004 †ST43: SCOPE
3430
3431 030742 012737 030754 001110 1$: MOV #2$,SLPERR ;SET LOOP ON ERROR POINTER TO 2$
3432 030750 012702 170000 MOV #170000,R2 ;LOAD "PRESENT & EXPECTED" PSW VALUE INTO R2
3433 030754 010237 177776 2$: MOV R2,PSW ;GO TO USER MODE-PRIORITY 0
3434 030760 012746 000340 MOV #340,-(SP) ;PUT A NEW PSW (PRIORITY=7) ON STACK
3435 030764 012746 030772 MOV #3$,-(SP) ;PUT NEW PC ON THE STACK
3436 030770 000002 RTI ;DO AN RTI FROM USER MODE
3437 030772 013701 177776 3$: MOV PSW,R1 ;READ NEW PSW INTO R1
3438 030776 042701 007437 BIC #7437,R1 ;MASK OFF COND. CODE, T-BIT, AND UNUSED BITS
3439 031002 005037 177776 CLR PSW ;GO BACK TO KERNEL MODE
3440 031006 020201 CMP R2,R1 ;DID PSW STAY IN USER, PRIORITY=0?
3441 031010 001401 BEQ 4$ ;BRANCH IF YES
3442 031012 104060 ERROR 60 ;PSW CHANGED BY AN RTI FROM USER
3443 ;FOR A TIGHTER SCOPE LOOP
3444 ;REPLACE ERROR CALL WITH
3445 ;"BR=2$" = 000760
3446 031014 012737 030742 001110 4$: MOV #1$,SLPERR ;RESET LOOP ON ERROR POINTER TO 1$
3447
3448

```

3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504

031022 000004
031024 012737 000600 172350
031032 012705 077402
031036 010537 172310
031042 012737 031072 000004
031050 012737 031070 000250
031056 012737 031064 001110
031064 005237 060001
031070 104043

031072 012706 001100
031076 005037 001200
031102 013737 177572 001362
031110 013737 177576 001364
031116 012700 000017
031122 020037 001362
031126 001402
031130 005237 001200
031134 012701 031110
031140 020137 001364
031144 001402
031146 005237 001200
031152 005737 001200
031156 001404
031160 104044

031162 042737 160000 177572
031170 012737 002246 000004
031176 012737 002314 000250
031204 012737 077406 172310
031212 012737 031024 001110

```
*****  
*TEST 44          KT ERROR NOT SERVICED IF ODD ADDR. ERROR  
*  
* THIS TEST CHECKS TO SEE THAT IF A CERTAIN VIRTUAL ADDRESS THAT  
* WOULD CAUSE A MEMORY MANAGEMENT ERROR CAUSES AN ODD ADDRESS  
* ERROR FIRST, THE ODD ADDRESS ERROR IS SERVICED BUT THE MEMORY  
* MANAGEMENT ERROR ISN'T. THIS MEANS THAT SR0 AND SR2  
* SHOULD NOT REPORT THE ERROR OR LOCK UP ITS VIRTUAL ADDRESS.  
* A READ-ONLY VIOLATION IS USED AS THE POTENTIAL MEMORY MANAGEMENT  
* ERROR  
*  
*****  
*ST44: SCOPE  
1$:  MOV      #600,KIPDR4 ;MAP KERNEL PAGE 4 TO 12-16K  
    MOV      #77402,R5 ;LOAD PDR4 DATA INTO R5  
    MOV      R5,KIPDR4 ;MAP PAGE 4 READ-ONLY  
    MOV      #4$,R#4 ;SET CPU TRAP VECTOR TO ADDRESS OF 4$  
    MOV      #3$,R#250 ;SET M.M. TRAP VECTOR TO ADDRESS OF 3$  
    MOV      #2$,SLPERR ;SET LOOP ON ERROR POINTER TO 2$  
2$:  INC      60001 ;CAUSE ODD ADDR. ERROR & POTENTIAL R/O ABORT  
3$:  ERROR   43 ;TRAPPED THRU M.M. VECTOR BUT SHOULDN'T HAVE  
    ;FOR TIGHTER SCOPE LOOP  
    ;REPLACE ERROR CALL WITH  
    ;"BR 2$" = 000776  
4$:  MOV      #KERSTK,KSP ;RESTORE STACK POINTER AFTER TRAPPING  
    CLR      $TMP1 ;CLEAR ERROR INDICATOR  
    MOV      SR0,WASSR0 ;READ STATUS REG. 0  
5$:  MOV      SR2,WASSR2 ;READ STATUS REG. 2  
    MOV      #17,R0 ;LOAD EXPECTED SR0 CONTENTS INTO R0  
    CMP      R0,WASSR0 ;SR0 ERROR BITS LEFT CLEAR BY TRAPPING?  
    BEQ      6$ ;BRANCH IF YES  
    INC      $TMP1 ;SR0 ERROR BITS SET WHEN ODD ADDR. SERVICED  
6$:  MOV      #5$,R1 ;LOAD EXPECTED SR2 CONTENTS INTO R1  
    CMP      R1,WASSR2 ;WAS SR2 LEFT UNLOCKED BY TRAPPING?  
    BEQ      7$ ;BRANCH IF YES  
    INC      $TMP1 ;SR2 LOCKED UP BY ODD ADDR. ERROR  
7$:  TST      $TMP1 ;WHERE SR0 OR SR2 EFFECTED?  
    BEQ      8$ ;BRANCH IF NO  
    ERROR   44 ;SR0 OR SR2 CHANGED BY ODD ADDR. ERROR  
    ;FOR TIGHTER SCOPE LOOP  
    ;REPLACE ERROR CALL WITH  
    ;"BR 2$" = 000741  
8$:  BIC      #160000,SR0 ;CLEAR ERROR BITS THAT MAY BE SET IN SR0  
    MOV      #TIMERR,R#4 ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER  
    MOV      #MGERR,R#250 ;RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER  
    MOV      #77406,KIPDR4 ;REMAP PAGE 4 TO READ/WRITE  
    MOV      #1$,SLPERR ;RESET LOOP ON ERROR POINTER TO 1$  
*****
```

```
*****  
*TEST 45          PC & PSW SAVED FOR KT ERROR DURING SERVICE OF ODD ADDR. ERROR  
*  
* THIS TEST CHECKS THE PC AND PROCESSOR STATUS WORD SAVED WHEN  
* A KT ERROR OCCURS DURING THE SECOND PUSH ON THE STACK DURING  
* SERVICING OF AN ODD ADDR. ERROR. DURING A "DOUBLE ERROR"  
* SEQUENCE SUCH AS THIS, THE PSW SAVED WILL BE THE ONE PICKED UP  
*  
*****
```


H06

MD-11-DFKTH-A PDP 11/34 NEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 72
T45 PC & PSW SAVED FOR KT ERROR DURING SERVICE OF ODD ADDR. ERROR

```

3505
3506
3507
3508
3509
3510
3511
3512
3513 031220 000004
3514 031222 004737 035432
3515 031226 012737 000600 177646
3516 031234 012737 000600 177650
3517 031242 012737 077402 177606
3518 031250 012737 077406 177610
3519 031256 012737 031332 000004
3520 031264 012737 140017 000006
3521 031272 012737 031332 000250
3522 031300 012737 000340 000252
3523 031306 012737 031314 001110
3524 031314 012737 140000 177776
3525 031322 012706 100002
3526 031326 005237 100005
3527
3528 031332 016601 000002
3529 031336 011603
3530 031340 013737 177572 001362
3531 031346 013737 177576 001364
3532 031354 042737 160000 177572
3533 031362 005037 177776
3534 031366 012706 001100
3535 031372 012737 140000 177776
3536 031400 012706 000700
3537 031404 005037 177776
3538 031410 005037 001176
3539 031414 020127 170017
3540
3541
3542
3543 031420 001402
3544 031422 005237 001176
    
```

```

: * FROM VECTOR+2 (LOC. 6 IN THIS CASE) AFTER THE FIRST TRAP.
: * NOT THE PSW PRESENT BEFORE THE FIRST TRAP. SRO AND SR2
: * SHOULD RECORD THE KT ERROR (A R/O VIOLATION BY THE USER STACK PTR.)
: *
: * NOTE THAT THE PREVIOUS MODE BITS (13:12) OF THE PSW
: * WILL BE SET IN THE PSW THAT IS SAVED.
: *
: *****
↑ST45: SCOPE
1$: JSR PC TOFF ; TURN T-BIT TRAPPING OFF FOR THIS TEST
MOV #600,UIPAR3 ; MAP USER PAGE 3 TO 12-16K
MOV #600,UIPAR4 ; MAP USER PAGE 4 TO 12-16K
MOV #77402,UIPDR3 ; MAP USER PAGE 3 READ-ONLY
MOV #77406,UIPDR4 ; MAP USER PAGE 4 READ/WRITE
MOV #45,2#4 ; LOAD ADDRESS OF 4$ IN CPU (ODD ADDR.) VECTOR
MOV #140017,2#6 ; LOAD PSW THAT SHOULD BE PUT ON STACK IN VECTOR+2
MOV #45,2#250 ; LOAD ADDRESS OF 4$ IN M.M. TRAP VECTOR
MOV #340,2#252 ; LOAD A KERNEL PSW IN MMVEC+2
MOV #2$, $LPERR ; SET LOOP ON ERROR POINTER TO 2$
2$: MOV #140000,PSW ; GO TO USER MODE
MOV #100002,USP ; SET USER STACK PTR. SO SECOND PUSH IS IN PG. 3
3$: INC 100005 ; CAUSE ODD ADDRESS ERROR THAT WILL CAUSE
; R/O ERROR WHEN TRY TO SAVE OLD PC
4$: MOV 2(KSP),R1 ; PUT PSW SAVED ON KERNEL STACK INTO R1
MOV (KSP),R3 ; PUT PC SAVED ON KERNEL STACK INTO R3
MOV SRO,WASSRO ; READ THE CONTENTS OF M.M. STATUS REG. 0
MOV SR2,WASSR2 ; READ THE CONTENTS OF M.M. STATUS REG. 2
BIC #160000,SRO ; CLEAR THE ERROR BITS IN SRO
CLR PSW ; BE SURE IN KERNEL MODE
MOV #KERSTK,KSP ; RESTORE KERNEL STACK POINTER
MOV #140000,PSW ; GO TO USER MODE
MOV #USESTK,USP ; RESTORE USER STACK POINTER
CLR PSW ; GO BACK TO KERNEL MODE
CLR $TMPD ; CLEAR ERROR INDICATOR
CMP R1,#170017 ; WAS THE PSW SAVED THE ONE PICKED UP BY THE
; ODD ADDR. TRAP FROM ERRVEC+2?
; VALUE 170017 = PSW FROM LOC. 6 WITH
; PREVIOUS MODE BITS = USER
BEQ $$ ; BRANCH IF YES
INC $TMPD ; WRONG PSW SAVED DURING "DOUBLE ERROR" SEQUENCE
    
```

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 10:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 73
T45 PC & PSW SAVED FOR KT ERROR DURING SERVICE OF ODD ADDR. ERROR

3545	031426	020327	031332		55:	CMP	R3, #35+4		: WAS THE PC AT THE TIME OF THE ODD ADDR. ERROR
3546									: SAVED ON THE STACK?
3547	031432	001402				BEQ	65		: BRANCH IF YES
3548	031434	005237	001176			INC	\$TMP0		: WRONG PC SAVED DURING TRAP SEQUENCE
3549	031440	023727	001362	020147	65:	CMP	WASSR0, #20147		: DID SR0 REPORT - USER, PAGE 3, R/O ABORT?
3550	031446	001402				BEQ	75		: BRANCH IF YES
3551	031450	005237	001176			INC	\$TMP0		: SR0 DID NOT REPORT R/O ABORT
3552	031454	023727	001364	031326	75:	CMP	WASSR2, #35		: DID SR2 LOCK UP VIRTUAL ADDR. OF LAST
3553									: INSTRUCTION SUCCESSFULLY FETCHED?
3554	031462	001402				BEQ	85		: BRANCH IF YES
3555	031464	005237	001176			INC	\$TMP0		: SR2 DID NOT LOCK UP ADDR. OF ODD ADDR. INST.
3556	031470	005737	001176		85:	TST	\$TMP0		: ANY "ERRORS" DURING TRAP SEQUENCE?
3557	031474	001401				BEQ	95		: BRANCH IF NO
3558	031476	104045				ERROR	45		: THE WRONG PC OR PSW WERE SAVED
3559									: OR SR0 OR SR2 DID NOT REPORT R/O
3560									: ERROR DURING ODD ADDR. - KT TRAP
3561									: SEQUENCE
3562									: FOR TIGHTER SCOPE LOOP
3563									: REPLACE ERROR CALL WITH
3564									: "BR 23" = 000710
3565	031500	012737	002246	000004	95:	MOV	#TIMERR, #4		: RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
3566	031506	012737	000340	000006		MOV	#340, #6		: RELOAD ERRVEC+2 WITH KERNEL PSW
3567	031514	012737	002314	000250		MOV	#MGERR, #250		: RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER
3568	031522	012737	077406	177606		MOV	#77406, UIPDR3		: REMAP USER PAGE 3 READ/WRITE
3569	031530	012737	031222	001110		MOV	#15, SLPERR		: RESET LOOP ON ERROR POINTER TO 15
3570	031536	004737	035466			JSR	PC, 10N		: TURN T-BIT TRAPPING BACK ON

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

 * TEST 46 MOVE FROM PREVIOUS (USER) I-SPACE
 *
 * THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT THE
 * PREVIOUS MODE IS CLOKED CORRECTLY
 * THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
 *
 * IF THE CORRECT MODE (USER) IS NOT ENABLED A NON-RESIDENT ABORT
 * WILL OCCUR AND TRAP TO 235, WHERE THE ERRORS ARE REPORTED.
 *

3591	031542	000004			TST46:	SCOPE			
3592	031544	005037	172340		15:	CLR	KIPAR0		: MAP KERNEL PAGE 0 TO 0-4K
3593	031550	012737	000200	172342		MOV	#200, KIPAR1		: MAP KERNEL PAGE 1 TO 4-8K
3594	031556	012737	000400	172344		MOV	#400, KIPAR2		: MAP KERNEL PAGE 2 TO 8-12K
3595	031564	012737	000600	172346		MOV	#600, KIPAR3		: MAP KERNEL PAGE 3 TO 12-16K
3596	031572	012737	000600	172350		MOV	#600, KIPAR4		: MAP KERNEL PAGE 4 TO 12-16K
3597	031600	012737	007600	172356		MOV	#7600, KIPAR7		: MAP KERNEL PAGE 7 TO THE I/O PAGE
3598	031606	012700	077406			MOV	#77406, R0		: MAKE ALL KERNEL I-SPACE PAGES RESIDENT
3599									: READ/WRITE, LENGTH 200 BLOCKS
3600	031612	012702	000010			MOV	#10, R2		: SET LOOP COUNTER TO 8

JOB

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 74
T46 MOVE FROM PREVIOUS (USER) I-SPACE

3601	031616	012701	172300		MOV	#KIPDR0,R1	;PUT ADDRESS OF FIRST PDR IN R1
3602	031622	010021			MOV	R0,(R1)+	;LOAD PDR WITH 77406
3603	031624	077202			SQB	R2,2\$;LOOP TO 2\$ UNTIL ALL PDRS LOADED
3604	031626	012702	000010		MOV	#10,R2	;SET LOOP COUNTER TO 8
3605	031632	012701	177600		MOV	#UIPDR0,R1	;PUT ADDRESS OF FIRST PDR IN R1
3606	031636	010021			MOV	R0,(R1)+	;LOAD PDR WITH 77406
3607	031640	077202			SQB	R2,3\$;LOOP TO 3\$ UNTIL ALL PDRS LOADED
3608	031642	012737	000000	177640	MOV	#000,UIPAR0	;MAP USER I PAGE 0 TO 0-4K
3609	031650	012737	000200	177642	MOV	#200,UIPAR1	;MAP USER I PAGE 1 TO 4-8K
3610	031656	012737	000400	177644	MOV	#400,UIPAR2	;MAP USER I PAGE 2 TO 8-12K
3611	031664	012737	000600	177646	MOV	#600,UIPAR3	;MAP USER I PAGE 3 TO 12-16K
3612	031672	012737	007600	177656	MOV	#7600,UIPAR7	;MAP USER I PAGE 7 TO THE I/O PAGE
3613	031700	012737	031706	001110	MOV	#4\$,SLPERR	;SET LOOP ON ERROR TO 4\$
3614	031706						
3615	031706	012737	077406	172310	MOV	#77406,KIPDR4	;KERNEL I-SPACE PAGE 4 READ/WRITE
3616	031714	012737	000600	172350	MOV	#600,KIPAR4	;MAP KERNEL I PAGE 4 TO 12K
3617	031722	012737	000600	177650	MOV	#600,UIPAR4	;MAP USER I PAGE 4 TO 12K
3618	031730	012700	036514		MOV	#36514,R0	;LOAD DATA PATTERN INTO R0
3619	031734	010037	100000		MOV	R0,#100000	;LOAD DATA PATTERN INTO PHY 60000
3620	031740	012737	032342	000250	MOV	#23\$,MMVEC	;SET M.M. VECTOR TO 23\$
3621	031746	105037	172310		CLRB	KIPDR4	;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
3622						;THE FOLLOWING WILL TEST	DSTM=0 MFPI
3623							
3624	031752	012737	031760	001110	MOV	#5\$,SLPERR	;SET LOOP ON ERROR POINTER TO 5\$
3625	031760	012737	030340	177776	MOV	#030340,PSW	;MAKE PREVIOUS MODE USER
3626	031766	006506			MFPI	USP	;PUT USER STACK POINTER ON KERNEL
3627							;STACK
3628	031770	022706	001100		CMP	#KERSTK,KSP	;WAS SOMETHING PUSHED ON STACK AT 6\$

K06

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 75
T46 MOVE FROM PREVIOUS (USER) I-SPACE

3629 031774 001407
3630 031776 012600
3631 032000 012701 000700

BEQ 7\$
MOV (KSP)+,R0
MOV #USESTK,R1

;BRANCH IF NOTHING WAS PUSHED
;POP KERNEL STACK INTO R0
;EXPECTING TO GET 700 AS USP

3632	032004	020001				CMP	RO,R1		:DID YOU GET THE RIGHT POINTER?
3633	032006	001403				BEQ	8\$:BRANCH IF YOU DID
3634	032010	104046				ERROR	46		:WRONG THING WAS PUSHED ON STACK
3635									:FOR TIGHTER SCOPE LOOP
3636									:REPLACE ERROR CALL WITH
3637									: "BR 5\$" = 000763
3638	032012	000401				BR	8\$:BRANCH TO NEXT TRY
3639	032014	104050			7\$:	ERROR	50		:NOTHING PUSHED ON STACK
3640									:FOR TIGHTER SCOPE LOOP
3641									:REPLACE ERROR CALL WITH
3642									: "BR 5\$" = 000761
3643	032016				8\$:			:THE FOLLOWING WILL TEST	DSTM=1 MFPI.
3644	032016	012737	032030	001110		MOV	#9\$,SLPERR		:SET LOOP ON ERROR POINTER TO 9\$
3645	032024	012700	036514			MOV	#36514,R0		:RELOAD DATA PATTERN IN R0
3646	032030	012737	030340	177776	9\$:	MOV	#030340,PSW		:MAKE PREVIOUS MODE USER
3647	032036	012702	100000			MOV	#100000,R2		:LOAD VIRTUAL ADDRESS INTO R2
3648	032042	006512				MFPI	(R2)		:READ FROM PHYSICAL 60000
3649	032044	012601				MOV	(KSP)+,R1		:POP KERNEL STACK INTO R1
3650	032046	020001				CMP	RO,R1		:WAS DATA FETCHED SAME AS STORED
3651	032050	001401				BEQ	10\$:BRANCH IF CORRECT DATA WAS FETCHED
3652	032052	104046				ERROR	46		:WRONG DATA WAS FETCHED
3653									:FOR TIGHTER SCOPE LOOP
3654									:REPLACE ERROR CALL WITH
3655									: "BR 9\$" = 000766
3656	032054				10\$:			:THE FOLLOWING WILL TEST	DSTM=2 MFPI.
3657	032054	012737	032062	001110		MOV	#11\$,SLPERR		:SET LOOP ON ERROR POINTER TO 11\$
3658	032062	012737	030340	177776	11\$:	MOV	#030340,PSW		:MAKE PREVIOUS MODE USER
3659	032070	012702	100000			MOV	#100000,R2		:LOAD VIRTUAL ADDRESS INTO R2
3660	032074	006522				MFPI	(R2)+		:READ FROM PHYSICAL 60000
3661	032076	012601				MOV	(KSP)+,R1		:POP KERNEL STACK INTO R1
3662	032100	020001				CMP	RO,R1		:WAS DATA FETCHED SAME AS STORED
3663	032102	001401				BEQ	12\$:BRANCH IF CORRECT DATA WAS FETCHED
3664	032104	104046				ERROR	46		:WRONG DATA WAS FETCHED
3665									:FOR TIGHTER SCOPE LOOP
3666									:REPLACE ERROR CALL WITH
3667									: "BR 11\$" = 000766
3668	032106				12\$:			:THE FOLLOWING WILL TEST	DSTM=3 MFPI.

M06

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 77
T46 MOVE FROM PREVIOUS (USER) I-SPACE

3669	032106	012737	032114	001110	
3670	032114	012737	030340	177776	13\$:
3671	032122	006537	100000		
3672	032126	012601			
3673	032130	020001			
3674	032132	001401			
3675	032134	104046			
3676					
3677					

MOV	#13\$, \$LPERR	;SET LOOP ON ERROR POINTER TO 13\$
MOV	#030340, PSW	;MAKE PREVIOUS MODE USER
MFPI	#100000	;READ FROM PHYSICAL 60000
MOV	(KSP)+, R1	;POP KERNEL STACK INTO R1
CMP	R0, R1	;WAS DATA FETCHED SAME AS STORED
BEQ	14\$;BRANCH IF CORRECT DATA WAS FETCHED
ERROR	46	;WRONG DATA WAS FETCHED
		;FOR TIGHTER SCOPE LOOP
		;REPLACE ERROR CALL WITH

```

3678
3679 032136
3680 032136 012737 032144 001110
3681 032144 012737 030340 177776
3682 032152 012702 100002
3683 032156 006542
3684 032160 012601
3685 032162 020001
3686 032164 001401
3687 032166 104046
3688
3689
3690
3691 032170
3692
3693
3694 032170 012737 032176 001110
3695 032 76 012737 030340 177776
3696 032 704 012737 100000 001202
3697 032 212 012702 001204
3698 03 216 006552
3699 03 220 012601
3700 0 3222 020001
3701 32224 001401
3702 J32226 104046
3703
3704
3705
3706 032230
3707
3708 032230 012737 032236 001110
3709 032236 012737 030340 177776
3710 032244 005002
3711 032246 006562 100000
3712 032252 012601
3713 032254 020001
3714 032256 001401
3715 032260 104046
3716
3717
3718
3719 032262
3720
3721 032262 012737 032270 001110
3722 032270 012737 030340 177776
3723 032276 012737 100000 001202
3724 032304 012702 001202
3725 032310 006572 000000
3726
3727 032314 012601
3728 032316 020001
3729 032320 001401
3730 032322 104046
3731
3732
3733

```

```

14$: ;THE FOLLOWING WILL TEST
MOV #15$,SLPERR
MOV #030340,PSW
MOV #100002,R2
MFPI -(R2)
MOV (KSP)+,R1
CMP R0,R1
BEQ 16$
ERROR 46
; "BR 13$" = 000767
; DSTM=4 MFPI.
; SET LOOP ON ERROR POINTER TO 15$
; MAKE PREVIOUS MODE USER
; LOAD VIRTUAL ADDRESS INTO R2
; READ FROM PHYSICAL 60000
; POP KERNEL STACK INTO R1
; WAS DATA FETCHED SAME AS STORED
; BRANCH IF CORRECT DATA WAS FETCHED
; WRONG DATA WAS FETCHED
; FOR TIGHTER SCOPE LOOP
; REPLACE ERROR CALL WITH
; "BR 15$" = 000766

15$: ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
MOV #17$,SLPERR
MOV #030340,PSW
MOV #100000,$TMP2
MOV #($TMP2+2),R2
MFPI 2-(R2)
MOV (KSP)+,R1
CMP R0,R1
BEQ 18$
ERROR 46
; SET LOOP ON ERROR POINTER TO 17$
; MAKE PREVIOUS MODE USER
; LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
; LOAD ADDR. OF $TMP2+2 INTO R2
; READ FROM PHYSICAL 60000
; POP KERNEL STACK INTO R1
; WAS DATA FETCHED SAME AS STORED
; BRANCH IF CORRECT DATA WAS FETCHED
; WRONG DATA WAS FETCHED
; FOR TIGHTER SCOPE LOOP
; REPLACE ERROR CALL WITH
; "BR 17$" = 000763

18$: ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
MOV #19$,SLPERR
MOV #030340,PSW
CLR R2
MFPI 100000(R2)
MOV (KSP)+,R1
CMP R0,R1
BEQ 20$
ERROR 46
; SET LOOP ON ERROR POINTER TO 19$
; MAKE PREVIOUS MODE USER
; MAKE REGISTER 2 A ZERO
; READ FROM PHYSICAL 60000
; POP KERNEL STACK INTO R1
; WAS DATA FETCHED SAME AS STORED
; BRANCH IF CORRECT DATA WAS FETCHED
; WRONG DATA WAS FETCHED
; FOR TIGHTER SCOPE LOOP
; REPLACE ERROR CALL WITH
; "BR 19$" = 000766

20$: ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
MOV #21$,SLPERR
MOV #030340,PSW
MOV #100000,$TMP2
MOV # $TMP2,R2
MFPI 20(R2)
MOV (KSP)+,R1
CMP R0,R1
BEQ 22$
ERROR 46
; SET LOOP ON ERROR POINTER TO 21$
; MAKE PREVIOUS MODE USER
; LOAD TEST LOC. V.A. INTO $TMP2
; LOAD ADDRESS OF $TMP2 INTO R2
; USE $TMP2 TO FETCH VIRTUAL
; ADDRESS OF 60000
; POP KERNEL STACK INTO R1
; WAS DATA FETCHED SAME AS STORED
; BRANCH IF CORRECT DATA WAS FETCHED
; WRONG DATA WAS FETCHED
; FOR TIGHTER SCOPE LOOP
; REPLACE ERROR CALL WITH
; "BR 21$" = 000762

```

```

3734 032324 012737 002314 000250 22$: MOV #MGMERR,MMVEC ;SET M.M. VECTOR TO NORMAL ROUTINE
3735 032332 012737 031544 001110 MOV #1$,SLPERR ;SET LOOP POINTER TO START OF TEST
3736 032340 000423 BR TST47 ;;BRANCH TO NEXT TEST
3737
3738
3739 032342 012637 001356 23$: MOV (KSP)+,TRAPPC ;SAVE PC & PS OF TRAP
3740 032346 012637 001360 MOV (KSP)+,TRAPPS
3741 032352 013737 177572 001362 MOV SRO,WASSRO ;SAVE SRO FOR ERROR TYPEOUT
3742 032360 013737 177576 001364 MOV SR2,WASSR2 ;SAVE SR2 FOR ERROR TYPEOUT
3743 032366 042737 160000 177572 BIC #160000,SRO ;CLEAR ERROR BITS IN SRO AND LEAVE
3744 032374 104051 ERROR 51 ;TRIED TO READ NON-RESIDENT PAGE
3745 ;FOR TIGHTER SCOPE LOOP
3746 ;REPLACE ERROR CALL WITH
3747 ;A "NOP" = 000240
3748 032376 013746 001360 MOV TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
3749 032402 013746 001356 MOV TRAPPC,-(KSP)
3750 032406 000002 RTI
3751
3752
3753 ;*****
3754 ;TEST 47 MOVE TO PREVIOUS (USER) I-SPACE
3755 ;
3756 ; THIS TEST USES THE 'MTPI' INSTRUCTION TO ENSURE THAT THE
3757 ; PREVIOUS MODE IS CLOKED CORRECTLY
3758 ; THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
3759 ;
3760 ;
3761 ; IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
3762 ; WILL OCCUR AND TRAP TO 20$, WHERE THE ERRORS ARE REPORTED.
3763 ;
3764 ;*****
3765 ;TST47: SCOPE
3766 032410 000004 1S: MOV #77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE
3767 032412 012737 077406 172310 MOV #77406,UIPDR4 ;USER I-SPACE PAGE 4 READ/WRITE
3768 032420 012737 077406 177610 MOV #600,KIPAR4 ;MAP KERNEL I PAGE 4 TO 12K
3769 032426 012737 000600 172350 MOV #600,UIPAR4 ;MAP USER I PAGE 4 TO 12K
3770 032434 012737 000600 177650 MOV #20$,MMVEC ;SET M.M. VECTOR TO 20$
3771 032442 012737 033222 000250 ;THE FOLLOWING WILL TEST DSTM=0 MTPI
3772
3773 032450 012737 030340 177776 2$: MOV #030340,PSW ;MAKE PREVIOUS MODE USER
3774 032456 012746 007777 MOV #7777,-(KSP) ;PUSH DATA ON KERNEL STACK
3775 032462 006606 MTPI USP ;LOAD USER STACK POINTER
3776 032464 006506 MFPI USP ;READ USER STACK POINTER
3777 032466 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
3778 032470 022701 007777 CMP #7777,R1 ;WAS USER STACK POINTER CHANGED
3779 032474 001401 BEQ 3$ ;BRANCH IF IT WAS
3780 032476 104050 ERROR 50 ;USER STACK POINTER NOT CHANGED
3781 ;FOR TIGHTER SCOPE LOOP
3782 ;REPLACE ERROR CALL WITH
3783 ;"BR 2$" = 000764
3784 032500 012737 030340 177776 3$: MOV #030340,PSW ;MAKE PREVIOUS MODE USER
3785 032506 012746 000700 MOV #USESTK,-(KSP) ;GET READY TO RESTORE USER S. POINT
3786 032512 006606 MTPI USP ;RESTORE USER STACK POINTER
3787 032514 ;THIS WILL TEST DSTM = 1 MTPI.
3788 032514 012737 032532 001110 4$: MOV #5$,SLPERR ;SET LOOP ON ERROR POINTER TO 5$
3789 032522 012702 100000 MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2

```


3790	032526	012700	125252	
3791	032532	010046		
3792	032534	105037	172310	
3793	032540	006612		
3794	032542	112737	000006	172310
3795	032550	011201		
3796	032552	020001		
3797	032554	001401		
3798	032556	104047		
3799				
3800				
3801				
3802	032560			
3803				
3804	032560	012737	032604	001110
3805	032566	012737	030340	177776
3806	032574	012700	125252	
3807	032600	012702	100000	
3808	032604	010046		
3809	032606	105037	172310	
3810	032612	006612		
3811	032614	112737	000006	172310
3812	032622	013701	100000	
3813	032626	020001		
3814	032630	001401		
3815	032632	104047		
3816				
3817				
3818				
3819	032634			
3820	032634	012737	032654	001110
3821	032642	012737	030340	177776
3822	032650	012700	052525	
3823	032654	010046		
3824	032656	105037	172310	
3825	032662	006637	100000	
3826	032666	112737	000006	172310
3827	032674	013701	100000	
3828	032700	020001		

```

5$:  MOV      #125252,RO      ;LOAD TEST DATA INTO RO
      MOV      RO,-(KSP)      ;PUSH TEST DATA ON KERNEL STACK
      CLRB    KIPDR4         ;MAKE KERNEL I PAGE 4 NON-RESIDENT
      MTPI    (R2)           ;LOAD TEST DATA INTO PHYSICAL 60000
      MOVB   #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
      MOV     (R2),R1        ;READ FROM ADDRESS 60000
      CMP    RO,R1          ;SEE IF DATA WAS STORED AT CORRECT PLACE
      BEQ    6$             ;BRANCH IF STORE WAS CORRECT
      ERROR  47             ;INCORRECT STORE
                               ;FOR TIGHTER SCOPE LOOP
                               ;REPLACE ERROR CALL WITH
                               ;"BR 5$" = 000765

6$:  ;THE FOLLOWING WILL TEST DSTM=2 MTPI.
      MOV     #8$,SLPERR     ;SET LOOP ON ERROR POINTER TO 8$
      MOV     #030340,PSW    ;MAKE PREVIOUS MODE USER
      MOV     #125252,RO     ;LOAD TEST DATA INTO RO
      MOV     #100000,R2     ;LOAD VIRTUAL ADDRESS INTO R2
      MOV     RO,-(KSP)      ;PUSH TEST DATA ON KERNEL STACK
      CLRB    KIPDR4         ;MAKE KERNEL PAGE 4 NON-RESIDENT
      MTPI    (R2)           ;LOAD TEST DATA INTO PHYSICAL 60000
      MOVB   #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
      MOV     #100000,R1     ;READ FROM ADDRESS 60000
      CMP    RO,R1          ;SEE IF DATA WAS STORED CORRECTLY
      BEQ    9$             ;BRANCH IF STORE WAS CORRECT
      ERROR  47             ;INCORRECT STORE
                               ;FOR TIGHTER SCOPE LOOP
                               ;REPLACE ERROR CALL WITH
                               ;"BR 8$" = 000764

9$:  ;THIS WILL TEST DSTM = 3 MTPI.
      MOV     #10$,SLPERR    ;SET LOOP ON ERROR POINTER TO 10$
      MOV     #030340,PSW    ;MAKE PREVIOUS MODE USER
      MOV     #52525,RO      ;LOAD TEST DATA INTO RO
      MOV     RO,-(KSP)      ;PUSH TEST DATA ON KERNEL STACK
      CLRB    KIPDR4         ;MAKE KERNEL I PAGE 4 NON-RESIDENT
      MTPI    #100000        ;LOAD TEST DATA INTO PHYSICAL 60000
      MOVB   #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
      MOV     #100000,R1     ;READ FROM ADDRESS 60000
      CMP    RO,R1          ;SEE IF DATA WAS STORED CORRECTLY

```

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 81
T47 MOVE TO PREVIOUS (USER) I-SPACE

3829 032702 001401
3830 032704 104047

BEQ 11\$
ERROR 47

; BRANCH IF STORE WAS CORRECT
; INCORRECT STORE
; FOR TIGHTER SCOPE LOOP
; REPLACE ERROR CALL WITH
; "BR 10\$" = 000763

3831
3832
3833
3834 032706
3835 032706 012737 032726 001110
3836 032714 012737 030340 177776
3837 032722 012700 125252
3838 032726 010046
3839 032730 012702 100002

11\$: ; THIS WILL TEST DSTM = 4

MOV #12\$, \$LPERR
MOV #030340, PSW
MOV #125252, R0
12\$: MOV R0, -(KSP)
MOV #100002, R2

MTPI.
; SET LOOP ON ERROR POINTER TO 12\$
; MAKE PREVIOUS MODE USER
; LOAD TEST DATA INTO R0
; PUSH TEST DATA ON KERNEL STACK
; LOAD VIRTUAL ADDRESS INTO R2

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 82
T47 MOVE TO PREVIOUS (USER) I-SPACE

3840	032734	105037	172310		CLRB	KIPDR4	; MAKE KERNEL I PAGE 4 NON-RESIDENT	
3841	032740	006642			MTPI	-(R2)	; LOAD TEST DATA INTO PHYSICAL 60000	
3842	032742	112737	000006	172310	MOV	#006, KIPDR4	; MAKE KERNEL PAGE 4 RESIDENT	
3843	032750	013701	100000		MOV	#100000, R1	; READ FROM ADDRESS 60000	
3844	032754	020001			CMP	R0, R1	; SEE IF DATA WAS STORED CORRECTLY	
3845	032756	001401			BEQ	13\$; BRANCH IF STORE WAS CORRECT	
3846	032760	104047			ERROR	47	; INCORRECT STORE	
3847							; FOR TIGHTER SCOPE LOOP	
3848							; REPLACE ERROR CALL WITH	
3849							"BR 12\$" = 000762	
3850	032762				13\$:	; THE FOLLOWING WILL TEST	DSTM=5 MTPI.	
3851								
3852	032762	012737	033014	001110	MOV	#14\$, \$LPERR	; SET LOOP ON ERROR POINTER TO 14\$	
3853	032770	012737	030340	177776	MOV	#030340, PSW	; MAKE PREVIOUS MODE USER	
3854	032776	012700	052525		MOV	#52525, R0	; LOAD TEST DATA INTO R0	
3855	033002	012702	001204		MOV	#(<\$TMP2+2>, R2	; LOAD ADDR. OF LOC. \$TMP2+2 INTO R2	
3856	033006	012737	100000	001202	MOV	#100000, \$TMP2	; LOAD VIRT. ADDR. OF TEST LOC. INTO \$TMP2	
3857	033014	010046			14\$:	MOV	R0, -(KSP)	; PUSH TEST DATA ON KERNEL STACK
3858	033016	105037	172310		CLRB	KIPDR4	; MAKE KERNEL PAGE 4 NON-RESIDENT	
3859	033022	006652			MTPI	-(R2)	; LOAD TEST DATA INTO PHYSICAL 60000	
3860	033024	112737	000006	172310	MOV	#006, KIPDR4	; MAKE KERNEL PAGE 4 RESIDENT	
3861	033032	013701	100000		MOV	#100000, R1	; READ FROM ADDRESS 60000	
3862	033036	020001			CMP	R0, R1	; SEE IF DATA WAS STORED CORRECTLY	
3863	033040	001401			BEQ	15\$; BRANCH IF STORE WAS CORRECT	
3864	033042	104047			ERROR	47	; INCORRECT STORE	
3865							; FOR TIGHTER SCOPE LOOP	
3866							; REPLACE ERROR CALL WITH	
3867							"BR 14\$" = 000764	
3868	033044				15\$:	; THIS WILL TEST	DSTM = 6 MTPI.	
3869								
3870	033044	012737	033066	001110	MOV	#16\$, \$LPERR	; SET LOOP ON ERROR POINTER TO 16\$	
3871	033052	012737	030340	177776	MOV	#030340, PSW	; MAKE PREVIOUS MODE USER	
3872	033060	012700	052525		MOV	#52525, R0	; LOAD TEST DATA INTO R0	
3873	033064	005002			CLR	R2	; MAKE REGISTER 2 ZERO	
3874	033066	010046			16\$:	MOV	R0, -(KSP)	; PUSH TEST DATA ON KERNEL STACK
3875	033070	105037	172310		CLRB	KIPDR4	; MAKE KERNEL I PAGE 4 NON-RESIDENT	
3876	033074	006662	100000		MTPI	100000(R2)	; LOAD TEST DATA INTO PHYSICAL 60000	
3877	033100	112737	000006	172310	MOV	#006, KIPDR4	; MAKE KERNEL PAGE 4 RESIDENT	
3878	033106	013701	100000		MOV	#100000, R1	; READ FROM ADDRESS 60000	
3879	033112	020001			CMP	R0, R1	; SEE IF DATA WAS STORED CORRECTLY	
3880	033114	001401			BEQ	17\$; BRANCH IF STORE WAS CORRECT	
3881	033116	104047			ERROR	47	; INCORRECT STORE	
3882							; FOR TIGHTER SCOPE LOOP	
3883							; REPLACE ERROR CALL WITH	
3884							"BR 16\$" = 000763	
3885	033120				17\$:	; THE FOLLOWING WILL TEST	DSTM=7 MTPI.	
3886								
3887	033120	012737	033152	001110	MOV	#18\$, \$LPERR	; SET LOOP ON ERROR POINTER TO 18\$	
3888	033126	012737	030340	177776	MOV	#030340, PSW	; MAKE PREVIOUS MODE USER	
3889	033134	012700	125252		MOV	#125252, R0	; LOAD TEST DATA INTO R0	
3890	033140	012737	100000	001202	MOV	#100000, \$TMP2	; LOAD VIRT. ADDR. OF TEST LOCATION	
3891							; INTO LOCATION \$TMP2	
3892	033146	012702	001202		MOV	#\$TMP2, R2	; LOAD ADDRESS OF \$TMP2 INTO R2	
3893	033152	010046			18\$:	MOV	R0, -(KSP)	; PUSH TEST DATA ON KERNEL STACK
3894	033154	105037	172310		CLRB	KIPDR4	; MAKE KERNEL PAGE 4 NON-RESIDENT	
3895	033160	006672	000000		MTPI	-(R2)	; LOAD TEST DATA INTO PHYSICAL 60000	

F07

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 83
T47 MOVE TO PREVIOUS (USER) I-SPACE

3896	033164	112737	000006	172310		MOVB	#006, KIPDR4	; MAKE KERNEL PAGE 4 RESIDENT
3897	033172	013701	100000			MOV	#100000, R1	; READ FROM ADDRESS 60000
3898	033176	020001				CMP	RO, R1	; SEE IF DATA WAS STORED CORRECTLY
3899	033200	001401				BEQ	19\$; BRANCH IF STORE WAS CORRECT
3900	033202	104047				ERROR	47	; INCORRECT STORE
3901								; FOR TIGHTER SCOPE LOOP
3902								; REPLACE ERROR CALL WITH
3903								; "BR 18\$" = 000763
3904	033204	012737	032412	001110	19\$:	MOV	#1\$, \$LPERR	; SET LOOP POINTER TO START OF TEST
3905	033212	012737	002314	000250		MOV	#MGMERR, MMVEC	; RESTORE M.M. VECTOR TO NORMAL ROUTINE
3906	033220	000423				BR	TST50	; BRANCH TO NEXT TEST
3907								
3908								
3909	033222	012637	001356		20\$:	MOV	(KSP)+, TRAPPC	; SAVE PC & PS OF TRAP
3910	033226	012637	001360			MOV	(KSP)+, TRAPPS	
3911	033232	013737	177572	001362		MOV	SRO, WASSRO	; SAVE SRO FOR ERROR TYPEOUT
3912	033240	013737	177576	001364		MOV	SR2, WASSR2	; SAVE SR2 FOR ERROR TYPEOUT
3913	033246	042737	160000	177572		BIC	#160000, SRO	; CLEAR ERROR BITS IN SRO
3914	033254	104051				ERROR	51	; TRIED TO LOAD A N.R. PAGE 4
3915								; FOR TIGHTER SCOPE LOOP
3916								; REPLACE ERROR CALL WITH
3917								; A "NOP" = 000240
3918	033256	013746	001360			MOV	TRAPPS, -(KSP)	; PUT PC & PS OF TRAP ON STACK
3919	033262	013746	001356			MOV	TRAPPC, -(KSP)	
3920	033266	000002				RTI		; RETURN TO TEST
3921								
3922								
3923								
3924								
3925								
3926								
3927								
3928								
3929								
3930								
3931								
3932								
3933								
3934								
3935								

```

*****
;TEST 50      MOVE FROM PREVIOUS (KERNEL) I-SPACE TO USER MODE
;
; THIS TEST CHECKS THAT IF THE PREVIOUS MODE IS KERNEL THE
; FETCH IS FROM KERNEL SPACE.
; THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
;
; IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
; WILL OCCUR AND TRAP TO 21$, WHERE THE ERRORS ARE REPORTED.
*****

```

3936	033270	000004				TST50:	SCOPE	
3937	033272	012700	077406		1\$:	MOV	#77406, RO	; MAKE ALL USER I-SPACE PAGES RESIDENT
3938								; READ/WRITE, LENGTH 200 BLOCKS
3939	033276	012702	000010			MOV	#10, R2	; SET LOOP COUNTER TO 8
3940	033302	012701	177600			MOV	#UIPDR0, R1	; LOAD ADDRESS OF FIRST PDR IN R1
3941	033306	010021			2\$:	MOV	RO, (R1)+	; LOAD PDR WITH 77406
3942	033310	077202				SQB	R2, 2\$; LOOP UNTIL 8 USER PDRS LOADED
3943	033312	012737	033320	001110		MOV	#3\$, \$LPERR	; SET LOOP ON ERROR TO 3\$
3944	033320	012737	140340	177776	3\$:	MOV	#140340, PSW	; GO TO USER MODE FOR THIS TEST
3945	033326	012737	077406	172310		MOV	#77406, KIPDR4	; KERNEL I-SPACE PAGE 4 READ/WRITE
3946	033334	012737	000600	172350		MOV	#600, KIPAR4	; MAP KERNEL I PAGE 4 TO 12K
3947	033342	012737	000600	177650		MOV	#600, UIPAR4	; MAP USER I PAGE 4 TO 12K
3948	033350	012700	036514			MOV	#36514, RO	; LOAD DATA PATTERN INTO RO
3949	033354	010037	100000			MOV	RO, #100000	; LOAD DATA PATTERN INTO PHY 60000
3950	033360	012702	100000			MOV	#100000, R2	; LOAD VIRTUAL ADDRESS INTO R2
3951								; THE FOLLOWING WILL TEST DSTN=0 MFPI

Address	OpCode	Operand 1	Operand 2	Operand 3	Operand 4	Comments
3952						
3953	033364	012737	033766	000250		MOV #215,MMVEC ; SET M.M. VECTOR TO 215
3954	033372	105037	177610			CLRB UIPDR4 ; MAKE USER I-SPACE PAGE 4 NON-RESIDENT
3955	033376	012737	140340	177776		MOV #140340,PSW ; MAKE PREVIOUS MODE KERNEL PRESENT USER
3956	033404	006506			45:	MFPI KSP ; PUT KERNEL STACK POINTER ON USER STACK
3957	033406	022706	000700			CMP #USESTK,USP ; WAS SOMETHING PUSHED ON STACK AT 15
3958	033412	001407				BEQ 55 ; BRANCH IF NOTHING WAS PUSHED
3959	033414	012600				MOV (USP)+,R0 ; POP USER STACK INTO R0
3960	033416	012701	001100			MOV #KERSTK,R1 ; EXPECTING 1100 AS KSP
3961	033422	020001				CMP R0,R1 ; DID YOU GET THE RIGHT POINTER?
3962	033424	001403				BEQ 65 ; BRANCH IF YOU DID
3963	033426	104046				ERROR 46 ; WRONG THING WAS PUSHED ON STACK
3964						
3965						
3966						
3967	033430	000401				BR 65 ; BRANCH TO NEXT TRY
3968	033432	104050			55:	ERROR 50 ; NOTHING PUSHED ON STACK
3969						
3970						
3971						
3972	033434				65:	; THE FOLLOWING WILL TEST DSTM=1 MFPI.
3973	033434	012737	033442	001110		MOV #75,SLPERR ; SET LOOP ON ERROR POINTER TO 75
3974	033442	012737	140340	177776	75:	MOV #140340,PSW ; MAKE PREVIOUS MODE KERNEL PRESENT USER
3975	033450	012700	036514			MOV #36514,R0 ; LOAD DATA EXPECTED INTO R0
3976	033454	012702	100000			MOV #100000,R2 ; LOAD VIRTUAL ADDRESS INTO R2
3977	033460	006512				MFPI (R2) ; READ FROM PHYSICAL 60000
3978	033462	012601				MOV (USP)+,R1 ; POP USER STACK INTO R1
3979	033464	020001				CMP R0,R1 ; WAS DATA FETCHED SAME AS STORED
3980	033466	001401				BEQ 85 ; BRANCH IF CORRECT DATA WAS FETCHED
3981	033470	104046				ERROR 46 ; WRONG DATA WAS FETCHED
3982						
3983						
3984						
3985	033472				85:	; THE FOLLOWING WILL TEST DSM=2 MFPI.
3986	033472	012737	033500	001110		MOV #95,SLPERR ; SET LOOP ON ERROR POINTER TO 95
3987	033500	012737	140340	177776	95:	MOV #140340,PSW ; MAKE PREVIOUS MODE KERNEL PRESENT USER
3988	033506	012702	100000			MOV #100000,R2 ; LOAD VIRTUAL ADDRESS INTO R2
3989	033512	006522				MFPI (R2)+ ; READ FROM PHYSICAL 60000
3990	033514	012601				MOV (USP)+,R1 ; POP USER STACK INTO R1
3991	033516	020001				CMP R0,R1 ; WAS DATA FETCHED SAME AS STORED
3992	033520	001401				BEQ 105 ; BRANCH IF CORRECT DATA WAS FETCHED
3993	033522	104046				ERROR 46 ; WRONG DATA WAS FETCHED
3994						
3995						
3996						
3997	033524				105:	; THE FOLLOWING WILL TEST DSTM=3 MFPI.
3998	033524	012737	033532	001110		MOV #115,SLPERR ; SET LOOP ON ERROR POINTER TO 115
3999	033532	012737	140340	177776	115:	MOV #140340,PSW ; MAKE PREVIOUS MODE KERNEL PRESENT USER
4000	033540	006537	100000			MFPI #100000 ; READ FROM PHYSICAL 60000
4001	033544	012601				MOV (USP)+,R1 ; POP USER STACK INTO R1
4002	033546	020001				CMP R0,R1 ; WAS DATA FETCHED SAME AS STORED
4003	033550	001401				BEQ 125 ; BRANCH IF CORRECT DATA WAS FETCHED
4004	033552	104046				ERROR 46 ; WRONG DATA WAS FETCHED
4005						
4006						
4007						

H07

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 85
TSO MOVE FROM PREVIOUS (KERNEL) I-SPACE TO USER MODE

4008	033554				12S:	; THE FOLLOWING WILL TEST DSTM=4 MFPI.
4009	033554	012737	033562	001110		MOV #13\$,SLPERR ; SET LOOP ON ERROR POINTER TO 13\$
4010	033562	012737	140340	177776	13S:	MOV #140340,PSW ; MAKE PREVIOUS MODE KERNEL PRESENT USER
4011	033570	012702	100002			MOV #100002,R2 ; LOAD VIRTUAL ADDRESS INTO R2
4012	033574	006542				MFPI -(R2) ; READ FROM PHYSICAL 60000
4013	033576	012601				MOV (USP)+,R1 ; POP USER STACK INTO R1
4014	033600	020001				CMP R0,R1 ; WAS DATA FETCHED SAME AS STORED
4015	033602	001401				BEQ 14\$; BRANCH IF CORRECT DATA WAS FETCHED
4016	033604	104046				ERROR 46 ; WRONG DATA WAS FETCHED
4017						; FOR TIGHTER SCOPE LOOP
4018						; REPLACE ERROR CALL WITH
4019						"BR 13\$" = 000766
4020	033606				14S:	; THE FOLLOWING WILL TEST DSTM=5 MFPI.
4021						
4022	033606	012737	033614	001110		MOV #15\$,SLPERR ; SET LOOP ON ERROR POINTER TO 15\$
4023	033614	012737	140340	177776	15S:	MOV #140340,PSW ; MAKE PREVIOUS MODE KERNEL PRESENT USER
4024	033622	012737	100000	001202		MOV #100000,\$TMP2 ; LOAD TEST LOC. VIRT. ADDR INTO LOC. \$TMP2
4025	033630	012702	001204			MOV #(\$TMP2+2),R2 ; LOAD ADDRESS OF \$TMP2+2 INTO R2
4026	033634	006552				MFPI 2-(R2) ; READ FROM PHYSICAL 60000
4027	033636	012601				MOV (USP)+,R1 ; POP USER STACK INTO R1
4028	033640	020001				CMP R0,R1 ; WAS DATA FETCHED SAME AS STORED
4029	033642	001401				BEQ 16\$; BRANCH IF CORRECT DATA FETCHED
4030	033644	104046				ERROR 46 ; WRONG DATA WAS FETCHED
4031						; FOR TIGHTER SCOPE LOOP
4032						; REPLACE ERROR CALL WITH
4033						"BR 15\$" = 000763
4034	033646				16S:	; THE FOLLOWING WILL TEST DSTM=6 MFPI.
4035						
4036	033646	012737	033654	001110		MOV #17\$,SLPERR ; SET LOOP ON ERROR POINTER TO 17\$.
4037	033654	012737	140340	177776	17S:	MOV #140340,PSW ; MAKE PREVIOUS MODE KERNEL PRESENT USER
4038	033662	005002				CLR R2 ; MAKE REGISTER 2 A ZERO
4039	033664	006562	100000			MFPI 100000(R2) ; READ FROM PHYSICAL 60000
4040	033670	012601				MOV (USP)+,R1 ; POP USER STACK INTO R1
4041	033672	020001				CMP R0,R1 ; WAS DATA FETCHED SAME AS STORED
4042	033674	001401				BEQ 18\$; BRANCH IF CORRECT DATA FETCHED
4043	033676	104046				ERROR 46 ; WRONG DATA WAS FETCHED
4044						; FOR TIGHTER SCOPE LOOP
4045						; REPLACE ERROR CALL WITH
4046						"BR 17\$" = 000766
4047	033700				18S:	; THE FOLLOWING WILL TEST DSTM=7 MFPI.
4048						
4049	033700	012737	033706	001110		MOV #19\$,SLPERR ; SET LOOP ON ERROR POINTER TO 19\$
4050	033706	012737	140340	177776	19S:	MOV #140340,PSW ; MAKE PREVIOUS MODE KERNEL PRESENT USER
4051	033714	012737	100000	001202		MOV #100000,\$TMP2 ; LOAD TEST LOC. VIRT. ADDR. INTO \$TMP2
4052	033722	012702	001202			MOV # \$TMP2,R2 ; LOAD ADDRESS OF \$TMP2 INTO R2
4053	033726	006572	000000			MFPI 20(R2) ; READ FROM PHYSICAL 60000
4054	033732	012601				MOV (USP)+,R1 ; POP USER STACK INTO R1
4055	033734	020001				CMP R0,R1 ; WAS DATA FETCHED SAME AS STORED
4056	033736	001401				BEQ 20\$; BRANCH IF CORRECT DATA FETCHED
4057	033740	104046				ERROR 46 ; WRONG DATA WAS FETCHED
4058						; FOR TIGHTER SCOPE LOOP
4059						; REPLACE ERROR CALL WITH
4060						"BR 19\$" = 000762
4061	033742	012737	002314	000250	20S:	MOV #MGMERR,MMVEC ; SET M.M. VECTOR TO NORMAL ROUTINE
4062	033750	012737	000340	177776		MOV #00340,PSW ; GO BACK TO KERNEL MODE, PREVIOUS KERNEL
4063	033756	012737	033272	001110		MOV #1\$,SLPERR ; SET LOOP POINTER TO START OF TEST

```

4064 033764 000423          BR      T5T51          ;; BRANCH TO NEXT TEXT
4065
4066
4067 033766 012637 001356    21$:  MOV      (KSP)+,TRAPPC ;SAVE PC & PS OF TRAP
4068 033772 012637 001360      MOV      (KSP)+,TRAPPS
4069 033776 013737 177572 001362    MOV      SR0,WASSR0    ;SAVE SR0 FOR ERROR TYPEOUT
4070 034004 013737 177576 001364    MOV      SR2,WASSR2    ;SAVE SR2 FOR ERROR TYPEOUT
4071 034012 042737 160000 177572    BIC      #160000,SR0   ;CLEAR ERROR BITS IN SR0
4072 034020 104051          ERROR    51           ;TRIED TO READ NON-RESILIENT PAGE
4073
4074
4075
4076 034022 013746 001360      MOV      TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
4077 034026 013746 001356      MOV      TRAPPC,-(KSP)
4078 034032 000002          RTI              ;RETURN TO TEST
4079
4080
4081
4082
4083
4084
4085
4086
4087
4088 034034 000004          *ST51: SCOPE
4089 034036 012737 030340 177776 1$:  MOV      #030340,PSW   ;MAKE PREVIOUS MODE=USER,CURRENT=KERNEL
4090 034044 106506          MFPD     USP         ;MFPD SHOULD ACT LIKE MFPI PUTTING
4091
4092 034046 022706 001100          CMP      #KERSTK,KSP  ;USER STACK POINTER ON THE KERNEL STACK?
4093 034052 001407          BEQ     2$           ;WAS SOMETHING PUSHED ON KERNEL STACK?
4094 034054 012600          MOV     (KSP)+,R0    ;BRANCH IF NO
4095 034056 012701 000700          MOV     #USESTK,R1   ;POP KERNEL STACK INTO R0
4096 034062 020001          CMP     R0,R1        ;EXPECTING TO GET 700 AS USP
4097 034064 001403          BEQ     3$           ;DID GET RIGHT POINTER VALUE?
4098 034066 104053          ERROR   53         ;BRANCH IF YES
4099
4100
4101
4102 034070 000401          BR      3$           ;WRONG THING WAS PUSHED ON STACK
4103 034072 104054          2$:  ERROR    54         ;FOR TIGHTER SCOPE LOOP
4104
4105
4106
4107 034074 012737 034102 001110 3$:  MOV      #4$,SLPERR   ;REPLACE ERROR CALL WITH
4108 034102 012746 007777 4$:  MOV      #7777,-(KSP) ;"BR 1$" = 000763
4109 034106 106606          MTPD    USP         ;BRANCH TO NEXT TRY
4110 034110 106506          MFPD    USP         ;NOTHING PUSHED ON STACK
4111 034112 012601          MOV     (KSP)+,R1   ;FOR TIGHTER SCOPE LOOP
4112 034114 022701 007777          CMP     #7777,R1    ;REPLACE ERROR CALL WITH
4113 034120 001401          BEQ     5$           ;"BR 1$" = 000761
4114 034122 104054          ERROR   54         ;SET LOOP ON ERROR POINTER TO 4$
4115
4116
4117
4118 034124 012746 000700 5$:  MOV      #USESTK,-(KSP) ;PUSH DATA ON KERNEL STACK
4119 034130 106606          MTPD    USP         ;LOAD THE USER STACK POINTER
                        ;READ USER STACK POINTER
                        ;POP KERNEL STACK INTO R1
                        ;WAS USER STACK POINTER CHANGED?
                        ;BRANCH IF YES
                        ;USER STACK POINTER NOT CHANGED
                        ;FOR TIGHTER SCOPE LOOP
                        ;REPLACE ERROR CALL WITH
                        ;"BR 4$" = 000767
                        ;GET READY TO RESTORE USER STK. PTR.
                        ;RESTORE USER STACK POINTER

```

4120 034132 012737 034036 001110
4121
4122
4123
4124
4125
4126
4127
4128
4129
4130
4131 034140 000004
4132 034142 005037 177776
4133 034146 012700 001100
4134 034152 010006
4135 034154 006506
4136
4137 034156 011601
4138 03416C 020001
4139
4140 034162 001401
4141 034164 104046
4142
4143
4144
4145 034166 005740
4146 034170 020600
4147 034172 001401
4148 034174 104050
4149
4150
4151
4152 034176 012706 001100
4153
4154
4155
4156
4157
4158
4159

```
MOV #1$, $LPERR ;SET LOOP POINTER TO START OF TEST
*****
*TEST 52 MOVE FROM PREVIOUS I=SPACE (PREVIOUS=CURRENT=KERNEL)
*
* THIS TEST CHECKS THAT IF BOTH PREVIOUS AND CURRENT MODES
* ARE KERNEL, AND THE SOURCE MODE IS 0, THE DESTINATION
* STACK IS NOT DECREMENTED BEFORE ACCESS.
* "MFPI KSP" SHOULD PUSH THE NON-DECREMENTED VALUE
* OF KSP (1100) ONTO THE STACK (AT LOC. 1076).
*****
1$: SCOPE
CLR @PSW ;SET PREVIOUS = CURRENT = KERNEL
MOV #STACK, RO ;SETUP VALUE FOR STACK POINTER
MOV RO, KSP ;LOAD STACK POINTER
MFPI KSP ;THE VALUE "STACK" SHOULD BE PUSHED
;BEFORE BEING DECREMENTED
MOV (KSP), R1 ;READ DATA WHICH WAS PUSHED
CMP RO, R1 ;WAS THE ORIGINAL VALUE OF THE
;STACK POINTER PUSHED?
BEQ 2$ ;BRANCH IF YES
ERROR 46 ;MFPI FETCHED WRONG DATA
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 1$" = 000766
2$: TST -(RO) ;SETUP EXPECTED STACK POINTER VALUE
CMP KSP, RO ;WAS THE STACK POINTER DECREMENTED?
BEQ 3$ ;BRANCH IF YES
ERROR 50 ;STACK NOT PUSHED BY THE MFPI
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 1$" = 000762
3$: MOV #STACK, KSP ;RESTORE STACK POINTER
.SBTTL *****
```



```

4160
4161
4162
4163
4164
4165
4166
4167
4168
4169
4170 034202
4171 034202 000004
4172 034204 005037 001102
4173 034210 005037 001212
4174 034214 005237 001234
4175 034220 042737 100000 001234
4176 034226 005327
4177 034230 000001
4178 034232 003072
4179 034234 012737
4180 034236 000001
4181 034240 034230
4182 034242 104401 034250
4183 034246 000407
4184
4185 034266
4186 034266 013746 001234
4187
4188 034272 104405
4189 034274 104401 034302
4190 034300 000421
4191
4192 034344
4193 034344 013746 001112
4194
4195 034350 104405
4196 034352 104401 001223
4197 034356 005037 001112
4198 034362 013700 000042
4199 034366 001414
4200 034370 005046
4201 034372 012746 034400
4202 034376 000426
4203
4204 034400
4205 034400 013700 000042
4206 034404 001405
4207 034406 000005
4208 034410 004710
4209 034412 000240
4210 034414 000240
4211 034416 000240
4212 034420
4213 034420 104400
4214 034422 042716 000020
4215 034426 032777 010000 144504

```

```

.SBTTL END OF PASS ROUTINE

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

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

SEOPCT: .WORD 1
BGT $DOAGN ;; YES
MOV (PC)+,2(PC)+ ;; RESTORE COUNTER

SENDCT: .WORD 1
TYPE $DOAGN ;; TYPE ASCIZ STRING
BR 65S ;; GET OVER THE ASCIZ
;; 65S: .ASCIZ <12><15>/END PASS #/
64S: MOV $PASS,-(SP) ;; SAVE $PASS FOR TYPEOUT
;; TYPE PASS NUMBER
TYPDS ;; GO TYPE--DECIMAL ASCII WITH SIGN
TYPE 67S ;; TYPE ASCIZ STRING
BR 66S ;; GET OVER THE ASCIZ
;; 67S: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
66S: MOV $ERTTL,-(SP) ;; SAVE $ERTTL FOR TYPEOUT
;; TOTAL NUMBER OF ERRORS
TYPDS ;; GO TYPE--DECIMAL ASCII WITH SIGN
TYPE $SCLF ;; TYPE CARRIAGE RETURN, LINE FEED
CLR $ERTTL ;; CLEAR ERROR TOTAL
SGET42: MOV 2#42,R0 ;; GET MONITOR ADDRESS
BEQ $DOAGN ;; BRANCH IF NO MONITOR
CLR -(SP) ;; INSURE THE "T" BIT IS CLEAR
MOV #SCLR.T,-(SP) ;; SETUP FOR AN RTI OR RTT
BR $RTN ;; GO DO AN RTI OR RTT TO LOAD THE PSW
;; WITH A CLEARED "T" BIT

SCLR.T: MOV 2#42,R0 ;; INSURE R0 CONTAINS THE MONITORS
BEQ $DOAGN ;; RETURN ADDRESS
RESET ;; CLEAR THE WORLD
SENDAD: JSR PC,(R0) ;; GO TO MONITOR
NOP ;; SAVE ROOM
NOP ;; FOR
NOP ;; ACT11

SDOAGN: TRAP ;; PUSH OLD PSW AND PC ON STACK
BIC #20,(SP) ;; CLEAR THE "T" BIT
BIT #BIT12,$SWR ;; RUN WITH TRACE TRAP?

```

```

4216 034434 001005      BNE      1$          ;; BR IF NO
4217 034436 005137 034462    COM      $TBIT      ;; IS IT TIME FOR TRACE TRAP
4218 034442 100402      BMI      1$          ;; BR IF NO
4219 034444 052716 000020    BIS      #20,(SP)   ;; SET TRACE TRAP
4220 034450 012746 034456    1$:     MOV      #SLOOP,-(SP) ;; JUMP TO START OF TEST
4221 034454 000002    $RTN:   RTI          ;; RETURN--THIS IS CHANGED TO
4222                                     ;; AN "RTT" IF "RTT" IS A LEGAL
4223                                     ;; INSTRUCTION
4224 034456      SLOOP:          ;;
4225 034456 000137      JMP      @PC+        ;; RETURN
4226 034460 020474    $RTNAD: .WORD     LOOP
4227 034462 000000    $TBIT:  .WORD     0          ;; "T" BIT STATE INDICATOR
4228 034464 377 377 000 $ENULL: .BYTE     -1,-1,0    ;; NULL CHARACTER STRING
4229 034470      .EVEN
4230      .SBTTL SCOPE HANDLER ROUTINE
4231
4232      ;*****
4233      ;THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
4234      ;AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
4235      ;AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
4236      ;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4237      ;SW14=1      LOOP ON TEST
4238      ;SW11=1      INHIBIT ITERATIONS
4239      ;SW09=1      LOOP ON ERROR
4240      ;SW08=1      LOOP ON TEST IN SWR<7:0>
4241      ;CALL
4242      ;*          SCOPE          ;;SCOPE=IOT
4243
4244      $SCOPE:
4245 034470 104410      CKSWR
4246 034472 032777 040000 144440 1$:     BIT      #BIT14,@SWR    ;; TEST FOR CHANGE IN SOFT-SWR
4247 034500 001114      BNE      $OVER      ;; LOOP ON PRESENT TEST?
4248                                     ;; YES IF SW14=1
4249 034502 000416      $XTSTR: BR      6$    ;;TESTER*****
4250                                     ;; IF RUNNING ON THE "XOR" TESTER CHANGE
4251 034504 013746 000004      MOV      @ERRVEC,-(SP) ;; THIS INSTRUCTION TO A "NOP" (NOP=240)
4252 034510 012737 034530 000004      MOV      #5,@ERRVEC  ;; SAVE THE CONTENTS OF THE ERROR VECTOR
4253 034516 005737 177060      TST      @177060     ;; SET FOR TIMEOUT
4254 034522 012637 000004      MOV      (SP)+,@ERRVEC ;; TIME OUT ON XOR?
4255 034526 000463      BR      $SVLAD      ;; RESTORE THE ERROR VECTOR
4256 034530 022626      5$:     CMP      (SP)+,(SP)+ ;; GO TO THE NEXT TEST
4257 034532 012637 000004      MOV      (SP)+,@ERRVEC ;; CLEAR THE STACK AFTER A TIME OUT
4258 034536 000423      BR      7$          ;; RESTORE THE ERROR VECTOR
4259 034540      6$: ; *****END OF CODE FOR THE XOR TESTER***** ;; LOOP ON THE PRESENT TEST
4260 034540 032777 000400 144372      BIT      #BIT08,@SWR  ;; LOOP ON SPEC. TEST?
4261 034546 001404      BEQ      2$          ;; BR IF NO
4262 034550 127737 144364 001102      CMPB    @SWR,$TSTNM  ;; ON THE RIGHT TEST? SWR<7:0>
4263 034556 001465      BEQ      $OVER      ;; BR IF YES
4264 034560 105737 001103      2$:     TSTB    $ERFLG  ;; HAS AN ERROR OCCURRED?
4265 034564 001421      BEQ      3$          ;; BR IF NO
4266 034566 123737 001115 001103      CMPB    $ERMAX,$ERFLG ;; MAX. ERRORS FOR THIS TEST OCCURRED?
4267 034574 101015      BHI      3$          ;; BR IF NO
4268 034576 032777 001000 144334      BIT      #BIT09,@SWR  ;; LOOP ON ERROR?
4269 034604 001404      BEQ      4$          ;; BR IF NO
4270 034606 013737 001110 001106      7$:     MOV      $LPERR,$LPAOR ;; SET LOOP ADDRESS TO LAST SCOPE
4271 034614 000446      BR      $OVER

```

```

4272 034616 105037 001103
4273 034622 005037 001212
4274 034626 000415
4275 034630 032777 004000 144302
4276 034636 001011
4277 034640 005737 001234
4278 034644 001406
4279 034646 005237 001104
4280 034652 023737 001212 001104
4281 034660 002024
4282 034662 012737 000001 001104
4283 034670 013737 034746 001212
4284 034676 105237 001102
4285 034702 113737 001102 001232
4286 034710 011637 001106
4287 034714 011637 001110
4288 034720 005037 001214
4289 034724 112737 000001 001115
4290 034732 013777 001102 144202
4291 034740 013716 001106
4292 034744 000002
4293 034746 000200
4294
4295
4296
4297
4298
4299
4300
4301
4302
4303
4304
4305
4306
4307
4308 034750
4309 034750 104410
4310 034752 010037 001162
4311 034756 010137 001164
4312 034762 010237 001166
4313 034766 010337 001170
4314 034772 010437 001172
4315 034776 010537 001174
4316 035002 113737 001102 001352
4317 035010 105237 001103
4318 035014 001775
4319 035016 013777 001102 144116
4320 035024 032777 002000 144106
4321 035032 001402
4322 035034 104401 001216
4323 035040 005237 001112
4324 035044 011637 001116
4325 035050 162737 000002 001116
4326 035056 117737 144034 001114
4327 035064 032777 020000 144046

```

```

45:  CLR  SERFLG      ;; ZERO THE ERROR FLAG
      CLR  $TIMES     ;; CLEAR THE NUMBER OF ITERATIONS TO MAKE
      BR   $          ;; ESCAPE TO THE NEXT TEST
35:  BIT   $BIT11,$SWR  ;; INHIBIT ITERATIONS?
      BNE  $          ;; BR IF YES
      TST  $PASS      ;; IF FIRST PASS OF PROGRAM
      BEQ  $          ;; INHIBIT ITERATIONS
      INC  $ICNT      ;; INCREMENT ITERATION COUNT
      CMP  $TIMES,$ICNT ;; CHECK THE NUMBER OF ITERATIONS MADE
      BGE  $OVER      ;; BR IF MORE ITERATION REQUIRED
15:  MOV  $1,$ICNT     ;; REINITIALIZE THE ITERATION COUNTER
      MOV  $SMXCNT,$TIMES ;; SET NUMBER OF ITERATIONS TO DO
$SVLAD: INCB $STSTN    ;; COUNT TEST NUMBERS
      MOV  $STSTN,$STSTN ;; SET TEST NUMBER IN APT MAILBOX
      MOV  ($SP),$SLPADR ;; SAVE SCOPE LOOP ADDRESS
      MOV  ($SP),$SLPERR ;; SAVE ERROR LOOP ADDRESS
      CLR  $ESCAPE     ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
      MOV  $1,$SERMAX  ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
$OVER:  MOV  $STSTN,$DISPLAY ;; DISPLAY TEST NUMBER
      MOV  $SLPADR,($SP) ;; FUDGE RETURN ADDRESS
      RTI  $          ;; FIXES PS
$SMXCNT: 200          ;; MAX. NUMBER OF ITERATIONS
.SBTTL  ERROR HANDLER ROUTINE

```

```

*****
*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
*AND GO TO ERRYP ON ERROR
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW15=1      HALT ON ERROR
*SW13=1      INHIBIT ERROR TYPEOUTS
*SW10=1      BELL ON ERROR
*SW09=1      LOOP ON ERROR
*CALL
*      ERROR  N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER

```

```

$ERROR:
      CKSWR
      MOV  R0,$REG0    ;; TEST FOR CHANGE IN SOFT-SWR
      MOV  R1,$REG1    ;; SAVE THE CONTENTS OF R0
      MOV  R2,$REG2    ;; SAVE THE CONTENTS OF R1
      MOV  R3,$REG3    ;; SAVE THE CONTENTS OF R2
      MOV  R4,$REG4    ;; SAVE THE CONTENTS OF R3
      MOV  R5,$REG5    ;; SAVE THE CONTENTS OF R4
      MOV  $STSTN,$TESTNO ;; SAVE THE TEST NUMBER
75:  INCB  $SERFLG     ;; SET THE ERROR FLAG
      BEQ  $          ;; DON'T LET THE FLAG GO TO ZERO
      MOV  $STSTN,$DISPLAY ;; DISPLAY TEST NUMBER AND ERROR FLAG
      BIT  $BIT10,$SWR  ;; BELL ON ERROR?
      BEQ  $          ;; NO - SKIP
      TYPE $BELL       ;; RING BELL
15:  INC  $ERTTL      ;; COUNT THE NUMBER OF ERRORS
      MOV  ($SP),$ERRPC ;; GET ADDRESS OF ERROR INSTRUCTION
      SUB  $2,$ERRPC
      MOV  $ERRPC,$ITEMB ;; STRIP AND SAVE THE ERROR ITEM CODE
      BIT  $BIT13,$SWR  ;; SKIP TYPEOUT IF SET

```

```

4328 035072 001004
4329 035074 004737 035206
4330 035100 104401 001223
4331 035104
4332 035104 122737 000001 001246
4333 035112 001007
4334 035114 113737 001114 035126
4335 035122 004737 037540
4336 035126 000
4337 035127 000
4338 035130 000777
4339 035132 005777 144002
4340 035136 100002
4341 035140 000000
4342 035142 104410
4343 035144 032777 001000 143766
4344 035152 001402
4345 035154 013716 001110
4346 035160 005737 001214
4347 035164 001402
4348 035166 013716 001214
4349 035172
4350 035172 022737 034410 000042
4351 035200 001001
4352 035202 000000
4353 035204
4354 035204 0010002
4355
4356
4357
4358
4359
4360
4361
4362
4363
4364
4365
4366
4367
4368
4369
4370
4371
4372
4373
4374
4375 035206
4376 035206 104401 001223
4377 035212 010046
4378 035214 005000
4379 035216 153700 001114
4380 035222 001004
4381
4382 035224 013746 001116
4383

```

```

BNE 20S ;: SKIP TYPEOUTS
JSR PC,ERRTYP ;: GO TO USER ERROR ROUTINE
TYPE ,SCRFL

20S: CMPB #APTENV,SENV ;: RUNNING IN APT MODE
BNE 2S ;: NO SKIP APT ERROR REPORT
MOVB $ITEMB,21S ;: SET ITEM NUMBER AS ERROR NUMBER
JSR PC,$ATY4 ;: REPORT FATAL ERROR TO APT

21S: .BYTE 0
.BYTE 0

22S: BR 22S ;: APT ERROR LOOP
2S: TST $SWR ;: HALT ON ERROR
BPL 3S ;: SKIP IF CONTINUE
HALT ;: HALT ON ERROR!
CKSWR ;: TEST FOR CHANGE IN SOFT-SWR
3S: BIT #BIT0,$SWR ;: LOOP ON ERROR SWITCH SET?
BEQ 4S ;: BR IF NO
MOV $LPERR,(SP) ;: FUDGE RETURN FOR LOOPING
4S: TST $ESCAPE ;: CHECK FOR AN ESCAPE ADDRESS
BEQ 5S ;: BR IF NONE
MOV $ESCAPE,(SP) ;: FUDGE RETURN ADDRESS FOR ESCAPE

5S: CMP #SENDAD,$#42 ;: ACT-11 AUTO-ACCEPT?
BNE 6S ;: BRANCH IF NO
HALT ;: YES

6S: RTI ;: RETURN
.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

```

```

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

```

```

*NOTES:
*1) THIS ROUTINE PROVIDES AN AUTOMATIC "CARRIAGE RETURN-LINE FEED"
*OR "EM", "DM", AND "DT".
*2) TWO SPACES ARE TYPED AFTER EACH NUMBER FOR "DT"
*3) FOR $ITEMB=0, JUST THE ERROR PC IS TYPED
*4) THE AVAILABLE FORMATS FOR TYPING DATA ARE:
* DF FORMAT
* 0 TYPE A 6 DIGIT OCTAL NUMBER (FROM 16-BIT BINARY)
* 1 TYPE A DECIMAL NUMBER WITHOUT LEADING ZEROS
* 2 TYPE A 16 DIGIT BINARY NUMBER
* 3 TYPE A 6 DIGIT OCTAL NUMBER (FROM 18-BIT BINARY)
*

```

```

ERRTYP:
TYPE ,SCRFL ;: "CARRIAGE RETURN" & "LINE FEED"
MOV RO,-(SP) ;: SAVE RO
CLR RO ;: PICKUP THE ITEM INDEX
BISB $#ITEMB,RO
BNE 1S ;: IF ITEM NUMBER IS ZERO, JUST
MOV $ERRPC,-(SP) ;: TYPE THE PC OF THE ERROR
;: SAVE $ERRPC FOR TYPEOUT
;: ERROR ADDRESS

```

```

4384 035230 104402          TYPCC          ;GO TYPE--OCTAL ASCII(ALL DIGITS)
4385 035232 000471          BR          13$          ;GET OUT
4386 035234 005300          1$: DEC      RO          ;ADJUST THE INDEX SO THAT IT WILL
4387 035236 006300          ASL      RO          ;WORK FOR THE ERROR TABLE
4388 035240 006300          ASL      RO
4389 035242 006300          ASL      RO
4390 035244 062700 001406      ADD      #ERRTB,RO      ;FORM TABLE POINTER
4391 035250 012037 035260      MOV      (RO)+,2$      ;PICKUP "ERROR MESSAGE" POINTER
4392 035254 001404          BEQ      3$            ;SKIP TYPEOUT IF NO POINTER
4393 035256 104401          TYPE     ;TYPE THE "ERROR MESSAGE"
4394 035260 000000          2$: .WORD  0          ;"ERROR MESSAGE" POINTER GOES HERE
4395 035262 104401 001223      TYPE     $CRLF        ;"CARRIAGE RETURN" & "LINE FEED"
4396 035266 012037 035276      3$: MOV      (RO)+,4$      ;PICKUP "DATA HEADER" POINTER
4397 035272 001404          BEQ      5$            ;SKIP TYPEOUT IF 0
4398 035274 104401          TYPE     ;TYPE THE "DATA HEADER"
4399 035276 000000          4$: .WORD  0          ;"DATA HEADER" POINTER GOES HERE
4400 035300 104401 001223      TYPE     $CRLF        ;"CARRIAGE RETURN" & "LINE FEED"
4401 035304 010146          5$: MOV      R1,-(SP)    ;SAVE R1
4402 035306 012001          MOV      (RO)+,R1     ;PICKUP "DATA TABLE" POINTER
4403 035310 001441          BEQ      12$          ;BR IF NO DATA TO BE TYPED
4404 035312 012000          MOV      (RO)+,RO     ;PICKUP "DATA FORMAT" POINTER
4405 035314 105710          6$: TSTB   (RO)        ;IS IT FORMAT 0?
4406 035316 001003          BNE      7$          ;BR IF NO
4407
4408          ;*THIS CODE IS FOR OCTAL (16-BIT) FORMAT (DF=0)
4409 035320 013146          MOV      2(R1)+,-(SP) ;SAVE 2(R1)+ FOR TYPEOUT
4410 035322 104402          TYPCC          ;GO TYPE--OCTAL ASCII(ALL DIGITS)
4411 035324 000425          BR          11$
4412
4413          ;*THIS CODE IS FOR DECIMAL FORMAT (DF=1)
4414 035326 121027 000001      7$: CMPB   (RO),#1     ;IS IT FORMAT 1?
4415 035332 001003          BNE      8$          ;BRANCH IF NO
4416 035334 013146          MOV      2(R1)+,-(SP) ;SAVE 2(R1)+ FOR TYPEOUT
4417 035336 104405          TYPDS          ;GO TYPE--DECIMAL ASCII WITH SIGN
4418 035340 000417          BR          11$
4419
4420          ;*THIS CODE IS FOR BINARY FORMAT (DF=2)
4421 035342 121027 000002      8$: CMPB   (RO),#2     ;IS IT FORMAT 2?
4422 035346 001003          BNE      9$          ;BRANCH IF NO
4423 035350 013146          MOV      2(R1)+,-(SP) ;SAVE 2(R1)+ FOR TYPEOUT
4424 035352 104406          TYPBN          ;GO TYPE--BINARY ASCII
4425 035354 000411          BR          11$
4426
4427          ;*THIS CODE IS FOR OCTAL (18-BIT) FORMAT (DF=3)
4428 035356 012146          9$: MOV      (R1)+,-(SP) ;PUT ADDRESS OF FIRST LOC. ON STACK
4429 035360 004737 040612      JSR      PC,$DB20     ;CONVERT TWO LOCS. TO AN ASCII STRING
4430 035364 062716 000005      ADD      #5,(SP)      ;ONLY NEED 6 CHARACTERS NOT 11
4431 035370 012637 035376      MOV      (SP)+,10$    ;PUT ADDRESS OF ASCII CHARS. AT 10$
4432 035374 104401          TYPE     ;TYPE OCTAL VALUE OF 18-BIT BINARY NO.
4433 035376 000000          10$: .WORD  0
4434
4435 035400 005711          11$: TST     (R1)        ;IS THERE ANOTHER NUMBER?
4436 035402 001404          BEQ      12$          ;BR IF NO
4437 035404 104401 035426      TYPE     14$         ;TYPE TWO(2) SPACES
4438 035410 105720          TSTB   (RO)+        ;POINT TO NEW "DATA FORMAT"
4439 035412 000740          BR          6$          ;LOOP

```

4440			
4441	035414	012601	
4442	035416	012600	
4443	035420	104401	001223
4444	035424	000207	
4445	035426	020040	000
4446		035432	
4447			

12\$:	MOV	(SP)+,R1	;RESTORE R1
13\$:	MOV	(SP)+,RO	;RESTORE RO
	TYPE	\$CRLF	;"CARRIAGE RETURN" & "LINE FEED"
	RTS	PC	;RETURN
14\$:	.ASCIZ	/ /	;TWO(2) SPACES
	.EVEN		

```

4448
4449
4450
4451
4452
4453
4454
4455
4456
4457
4458
4459 035432 033727 177776 000020
4460 035440 001411
4461 035442 013746 177776
4462 035446 011637 001366
4463
4464 035452 042716 000020
4465 035456 012746 035464
4466 035462 000006
4467 035464 000207
4468
4469
4470
4471
4472
4473
4474
4475
4476
4477 035466 033727 001366 000020
4478 035474 001410
4479 035476 013746 001366
4480 035502 012737 000340 001366
4481 035510 012746 035516
4482 035514 000006
4483 035516 000207
4484
4485
4486
4487
4488
4489
4490
4491
4492
4493
4494
4495
4496 035520 012702 000010
4497 035524 012701 172300
4498 035530 012721 177777
4499 035534 077203
4500 035536 012702 000010
4501 035542 012701 172340
4502 035546 012721 177777
4503 035552 077203

```

```

.SBTTL ***** SUBROUTINES USED BY THIS PROGRAM *****

.SBTTL TURN OFF T-BIT AND SAVE CURRENT PSW
*****
*
* THIS SUBROUTINE IS USED TO TURN OFF THE TRACE TRAP BIT IN THE PSW
* IF IT IS ON. THE PROCESSOR STATUS IS SAVED IN "TBITPS" SO THAT
* THE PSW CAN BE RESTORED TO ITS PREVIOUS CONDITION WHEN CONDITIONS
* WARRANT T-BIT TRAPPING.
*
*****
TOFF: BIT PSW,#TBIT ;IS THE T-BIT SET IN THE PSW?
      BEQ 1$ ;EXIT IF NO
      MOV PSW,-(SP) ;PUSH PRESENT PSW ON THE STACK
      MOV (SP),TBITPS ;ALSO SAVE IT IN "TBITPS" FOR
                        ;RESTORING LATER
      BIC #TBIT,(SP) ;CLEAR THE T-BIT (BIT 4) IN THE PSW
      MOV #1$,-(SP) ;PUSH PC OF "RTS" ON STACK
      RTT ;"RETURN" TO 1$ WITH T-BIT OFF
1$: RTS PC ;RETURN TO PROGRAM

.SBTTL TURN ON T-BIT AND RESTORE PREVIOUS PSW
*****
*
* THIS SUBROUTINE IS USED TO RESTORE THE PROCESSOR STATUS TO ITS
* PREVIOUS CONDITION BY RESTORING THE "T-BIT PSW" SAVED BY THE
* "TOFF" SUBROUTINE IN THE "TBITPS" LOCATION.
*
*****
TON: BIT TBITPS,#TBIT ;WAS T-BIT ON IN THE PREVIOUS PSW?
      BEQ 1$ ;EXIT IF NO
      MOV TBITPS,-(SP) ;PUSH PREVIOUS PSW ON THE STACK
      MOV #340,TBITPS ;RESET THE "TBITPS" LOCATION
      MOV #1$,-(SP) ;PUSH PC OF "RTS" ON STACK
      RTT ;"RETURN" TO 1$ WITH T-BIT RESTORED
1$: RTS PC ;RETURN TO PROGRAM

.SBTTL SET ALL WRITEABLE BITS IN ALL PAR/PDR'S
*****
*
* THIS SUBROUTINE IS USED BY THE PAR/PDR DUAL ADDRESSING TEST
* TO SET ALL WRITEABLE BITS IN ALL KERNEL AND USE PAR'S AND
* PDR'S TO A 1. THE "INITIAL STATE" OF HAVING ALL BITS=1 IS
* USED TO SEE THAT ONLY ONE REGISTER IS CLEARED IN RESPONSE TO
* A SINGLE PAR OR PDR ADDRESS.
*
*****
SETREG: MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8
        MOV #KIPDR0,R1 ;LOAD ADDRESS OF FIRST PDR INTO R1
1$: MOV #-1,(R1)+ ;SET BITS IN KERNEL PDR TO 1
      SOB R2,1$ ;LOOP TO 1$ UNTIL ALL KERNEL PDR'S LOADED
        MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8
        MOV #KIPAR0,R1 ;LOAD ADDRESS OF FIRST PAR INTO R1
2$: MOV #-1,(R1)+ ;SET BITS IN A KERNEL PAR TO 1
      SOB R2,2$ ;LOOP TO 2$ UNTIL ALL KERNEL PAR'S LOADED

```

E08

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 95
SET ALL WRITEABLE BITS IN ALL PAR/PDR'S

4504	035554	012702	000010
4505	035560	012701	177600
4506	035564	012721	177777
4507	035570	077203	
4508	035572	012702	000010
4509	035576	012701	177640
4510	035602	012721	177777
4511	035606	077203	
4512	035610	000207	
4513			
4514			
4515			
4516			
4517			
4518			
4519			
4520			
4521			
4522			
4523			
4524	035612		
4525	035612	012701	172300
4526	035616	012704	000010
4527	035622	012705	077416
4528	035626	011102	
4529	035630	020205	
4530	035632	001403	
4531	035634	020100	
4532	035636	001401	
4533	035640	104016	
4534			
4535			
4536			
4537	035642	062701	000002
4538	035646	077411	
4539	035650	012701	172340
4540	035654	012704	000010
4541	035660	012705	007777
4542	035664	011102	
4543	035666	020205	
4544	035670	001403	
4545	035672	020100	
4546	035674	001401	
4547	035676	104016	
4548			
4549			
4550			
4551	035700	062701	000002
4552	035704	077411	
4553	035706	012701	177600
4554	035712	012704	000010
4555	035716	012705	077416
4556	035722	011102	
4557	035724	020205	
4558	035726	001403	
4559	035730	020100	

```

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

.SBTTL READ & COMPARE KERNEL & USER PAR/PDR'S
*****
*
* THIS SUBROUTINE IS USED BY PAR/PDR DUAL ADDRESSING TEST TO
* READ ALL THE PAR'S AND PDR'S TO SEE THAT ONLY ONE REGISTER
* WAS CLEARED IN RESPONSE TO A SINGLE PAR OR PDR ADDRESS.
* ANY FAILURES FOUND BY THE PAR/PDR DUAL ADDRESSING TEST WILL
* BE REPORTED BY THIS SUBROUTINE.
*
*****
CMPREG:
MOV #KIPDR0,R1 ;LOAD ADDRESS OF FIRST KERNEL PDR IN R1
MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
MOV #77416,R5 ;PUT EXPECTED PDR CONTENTS IN R5
1$: MOV (R1),R2 ;READ A KERNEL PDR INTO R2
CMP R2,R5 ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
BEQ 2$ ;BRANCH IF YES
CMP R1,RO ;WAS IT THE REG. THAT WAS CLEARED?
BEQ 2$ ;BRANCH IF YES
ERROR 16 ;A PDR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;AN "RTS PC" = 000207
2$: ADD #2,R1 ;FORM NEXT KERNEL PDR ADDRESS
SOB R4,1$ ;LOOP TO 1$ UNTIL ALL KERNEL PDR'S CHECKED
MOV #KIPAR0,R1 ;LOAD ADDRESS OF FIRST KERNEL PAR IN R1
MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
MOV #7777,R5 ;PUT EXPECTED PAR CONTENTS IN R5
3$: MOV (R1),R2 ;READ A KERNEL PAR INTO R2
CMP R2,R5 ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
BEQ 4$ ;BRANCH IF YES
CMP R1,RO ;WAS IT THE REG. THAT WAS CLEARED?
BEQ 4$ ;BRANCH IF YES
ERROR 16 ;A PAR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;AN "RTS PC" = 000207
4$: ADD #2,R1 ;FORM NEXT KERNEL PAR ADDRESS
SOB R4,3$ ;LOOP TO 3$ UNTIL ALL KERNEL PAR'S CHECKED
MOV #UIPDR0,R1 ;LOAD ADDRESS OF FIRST USER PDR IN R1
MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
MOV #77416,R5 ;PUT EXPECTED PDR CONTENTS IN R5
5$: MOV (R1),R2 ;READ A USER PDR INTO R2
CMP R2,R5 ;ARE ALL WRITABLE BITS SET AS EXPECTED?
BEQ 6$ ;BRANCH IF YES
CMP R1,RO ;WAS IT THE REG. THAT WAS CLEARED?

```


F08

MD-11-DFKTH-A POP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 96
READ & COMPARE KERNEL & USER PAR/PDR'S

```

4560 035732 001401
4561 035734 104016
4562
4563
4564
4565 035736 062701 000002
4566 035742 077411
4567 035744 012701 177640
4568 035750 012704 000010
4569 035754 012705 007777
4570 035760 011102
4571 035762 020205
4572 035764 001403
4573 035766 020100
4574 035770 001401
4575 035772 104016
4576
4577
4578
4579 035774 062701 000002
4580 036000 077411
4581 036002 000207
4582
4583
4584
4585
4586
4587
4588
4589
4590
4591
4592
4593
4594
4595
4596
4597
4598
4599
4600 036004 012702 172340
4601 036010 032737 140000 177776
4602 036016 001402
4603 036020 012702 177640
4604 036024 013700 001376
4605 036030 072027 177764
4606 036034 042700 177761
4607 036040 060002
4608 036042 011200
4609 036044 010002
4610 036046 013737 001376 001402
4611 036054 042737 160000 001402
4612 036062 072227 177766
4613 036066 042702 177774
4614 036072 072027 000006
4615 036076 042700 000077
    
```

```

        BEQ      6$
        ERROR   16
;BRANCH IF YES
;A PDR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;AN "RTS PC" = 000207
;FORM NEXT USER PDR ADDRESS
;LOOP TO 5$ UNTIL ALL USER PDR'S CHECKED
;LOAD ADDRESS OF FIRST USER PAR IN R1
;LOAD LOOP COUNTER WITH AN 8
;PUT EXPECTED PAR CONTENTS IN R5
;READ A USER PAR INTO R2
;ARE ALL WRITABLE BITS SET AS EXPECTED?
;BRANCH IF YES
;WAS IT THE REG. THAT WAS CLEARED?
;BRANCH IF YES
;A PAR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;AN "RTS PC" = 000207
;FORM NEXT USER PAR ADDRESS
;LOOP TO 7$ UNTIL ALL USER PAR'S CHECKED
;RETURN TO TEST

6$:     ADD      #2,R1
        SOB     R4,5$
        MOV     #UIPAR0,R1
        MOV     #10,R4
        MOV     #7777,R5
7$:     MOV     (R1),R2
        CMP     R2,R5
        BEQ     8$
        CMP     R1,R0
        BEQ     8$
        ERROR   16

8$:     ADD      #2,R1
        SOB     R4,7$
        RTS     PC
    
```

.SBTTL CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS

```

*****
*
* THIS SUBROUTINE IS USED TO FORM AN 18-BIT PHYSICAL ADDRESS
* (PBA) FROM THE 16-BIT VIRTUAL ADDRESS (VBA) AND THE APPROPRIATE
* PAGE ADDRESS REGISTER (PAR). THE SAME METHOD USED BY THE MEMORY
* MANAGEMENT LOGIC IS USED. VBA <15:13> SELECTS WHICH PAR/PDR
* IS TO BE USED, VBA <5:0>+PBA <5:0>, AND VBA <12:6> IS ADDED
* TO PAR <11:00> TO GIVE PBA <17:6>. BITS <17:16> OF THE
* PHYSICAL ADDRESS ARE LEFT IN LOC. "PBAHI" AND BITS <15:00>
* ARE LEFT IN LOC. "PBALO". THE PSW'S "CURRENT MODE" BITS
* ARE USED TO SELECT THE KERNEL OR USER PAR/PDR'S. THE ROUTINE
* IS ENTERED WITH LOC. "VIRT1" CONTAINING THE 16-BIT VIRTUAL
* ADDRESS.
*****
    
```

```

FORMPA: MOV     #KIPAR0,R2
        BIT     #140000,PSW
        BEQ     1$
        MOV     #UIPAR0,R2
1$:     MOV     VIRT1,R0
        ASH     #-14,R0
        BIC     #177761,R0
        ADD     R0,R2
        MOV     (R2),R0
        MOV     R0,R2
        MOV     VIRT1,PBALO
        BIC     #160000,PBALO
        ASH     #-12,R2
        BIC     #177774,R2
        ASH     #6,R0
        BIC     #77,R0
;LOAD ADDRESS OF FIRST KERNEL PAR IN R2
;IN USER MODE?
;BRANCH IF NO
;LOAD ADDRESS OF FIRST USER PAR IN R2
;LOAD VIRTUAL ADDR. (VBA) INTO R0
;GET BITS <15:13> DOWN TO BITS <3:1>
;MASK OF ALL BITS BUT BITS <3:1>
;ADD OFFSET TO BASE PAR ADDRESS
;GET BITS <11:00> FROM APPROPRIATE PAR
;COPY PAR BITS <11:00> INTO R2
;PUT VIRTUAL ADDR. IN LOC. "PBALO"
;CLEAR OFF BITS <15:13> OF ORIGINAL VBA
;GET PAR <11:00> DOWN TO BITS <1:0> OF R2
;CLEAR OFF ALL BITS BUT BITS <1:0>
;SHIFT PAR<9:0> TO <15:6> OF R0
;CLEAR BITS <5:0> OF R0
    
```

G08

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 97
CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS

4616	036102	060037	001402	ADD	RO,PBALO	; IN EFFECT, ADD VBA<12:0> TO PAR<9:0>
4617						; (PAR<9:0> IN BITS <15:6> OF RO)
4618	036106	005502		ADC	R2	; ADD ANY CARRY TO R2
4619	036110	010237	001404	MOV	R2,PBAHI	; PUT BITS <17:16> OF PHYSICAL ADDR. IN PBAHI
4620	036114	000207		RTS	PC	; RETURN TO PROGRAM
4621						

.SBTTL TTY INPUT ROUTINE

4622
4623
4624
4625
4626
4627
4628
4629
4630
4631
4632 036116 022737 000176 001140
4633 036124 001114
4634 036126 105777 143012
4635 036132 100111
4636 036134 117746 143006
4637 036140 042716 177600
4638 036144 022726 000007
4639 036150 001102
4640 036152 123727 001134 000001
4641 036160 001476
4642
4643 036162 104401 037060
4644 036166 104401 037065
4645 036172 013746 000176
4646 036176 104402
4647 036200 104401 037076
4648 036204 005046
4649 036206 005046
4650 036210 105777 142730
4651 036214 100375
4652
4653 036216 117746 142724
4654 036222 042716 177600
4655
4656 036226 021627 000003
4657 036232 001015
4658 036234 104401 037046
4659 036240 062706 000006
4660 036244 123727 001135 000001
4661 036252 001003
4662 036254 012777 000100 142662
4663 036262 000137 037110
4664
4665
4666 036266 021627 000025
4667 036272 001005
4668 036274 104401 037053
4669 036300 062706 000006
4670 036304 000737
4671
4672
4673 036306 021627 000015
4674 036312 001022
4675 036314 005766 000004
4676 036320 001403
4677 036322 016677 000002 142610

```
*****
ENABL LSB
*****
SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
WHEN OPERATING IN TTY FLAG MODE.
$CKSWR: CMP      #SWREG, SWR      ;; IS THE SOFT-SWR SELECTED?
        BNE      15$             ;; BRANCH IF NO
        TSTB     #STKS           ;; CHAR THERE?
        BPL      15$             ;; IF NO, DON'T WAIT AROUND
        MOVB     #STKB, -(SP)    ;; SAVE THE CHAR
        BIC      #177, (SP)     ;; STRIP-OFF THE ASCII
        CMP      #7, (SP)+      ;; IS IT A CONTROL G?
        BNE      15$             ;; NO, RETURN TO USER
        CMPB     $AUTOB, #1      ;; ARE WE RUNNING IN AUTO-MODE?
        BEQ      15$             ;; BRANCH IF YES

$GTSWR: TYPE     , SCNTLG        ;; ECHO THE CONTROL-G (1G)
        TYPE     , SMSWR         ;; TYPE CURRENT CONTENTS
        MOV      SWREG, -(SP)    ;; SAVE SWREG FOR TYPEOUT
        TYPOC    , SMNEW         ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
        TYPE     , SMNEW         ;; PROMPT FOR NEW SWR
19$:   CLR      -(SP)           ;; CLEAR COUNTER
        CLR      -(SP)           ;; THE NEW SWR
7$:   TSTB     #STKS           ;; CHAR THERE?
        BPL      7$             ;; IF NOT TRY AGAIN

        MOVB     #STKB, -(SP)    ;; PICK UP CHAR
        BIC      #177, (SP)     ;; MAKE IT 7-BIT ASCII

        CMP      (SP), #3        ;; IS IT A CONTROL-C?
        BNE      9$             ;; BRANCH IF NOT
        TYPE     , SCNTLC        ;; YES, ECHO CONTROL-C (1C)
        ADD      #6, SP          ;; CLEAN UP STACK
        CMPB     $INTAG, #1      ;; REENABLE TTY KEYBOARD INTERRUPTS?
        BNE      8$             ;; BRANCH IF NO
        MOV      #100, #STKS     ;; ALLOW TTY KEYBOARD INTERRUPTS
        JMP      CNTRLC          ;; CONTROL-C RESTART

9$:   CMP      (SP), #25        ;; IS IT A CONTROL-U?
        BNE      10$            ;; BRANCH IF NOT
        TYPE     , SCNTLU        ;; YES, ECHO CONTROL-U (1U)
20$:  ADD      #6, SP          ;; IGNORE PREVIOUS INPUT
        BR      19$            ;; LET'S TRY IT AGAIN

10$:  CMP      (SP), #15        ;; IS IT A <CR>?
        BNE      16$            ;; BRANCH IF NO
        TST      4(SP)          ;; YES, IS IT THE FIRST CHAR?
        BEQ      11$            ;; BRANCH IF YES
        MOV      2(SP), #SWR     ;; SAVE NEW SWR
```

```

4678 036330 062706 000006
4679 036334 104401 001223
4680 036340 123727 001135 000001
4681 036346 001003
4682 036350 012777 000100 142566
4683 036356 000002
4684 036360 004737 037452
4685 036364 021627 000060
4686 036370 002420
4687 036372 021627 000067
4688 036376 003015
4689 036400 042726 000060
4690 036404 005766 000002
4691 036410 001403
4692 036412 006316
4693 036414 006316
4694 036416 006316
4695 036420 005266 000002
4696 036424 056616 177776
4697 036430 000667
4698 036432 104401 001222
4699 036436 000720
4700
4701
4702
4703
4704
4705
4706
4707
4708
4709
4710
4711 036440 011646
4712 036442 016666 000004 000002
4713 036450 105777 142470
4714 036454 100375
4715 036456 117766 142464 000004
4716 036464 042766 177600 000004
4717 036472 026627 000004 000023
4718 036500 001013
4719 036502 105777 142436
4720 036506 100375
4721 036510 117746 142432
4722 036514 042716 177600
4723 036520 022627 000021
4724 036524 001366
4725 036526 000750
4726 036530 026627 000004 000140
4727 036536 002407
4728 036540 026627 000004 000175
4729 036546 003003
4730 036550 042766 000040 000004
4731 036556 000002
4732
4733

```

```

11$: ADD #6,SP ; CLEAR UP STACK
14$: TYPE $CRLF ; ECHO <CR> AND <LF>
; CMPB $INTAG,#1 ; RE-ENABLE TTY KBD INTERRUPTS?
; BNE 15$ ; BRANCH IF NOT
; MOV #100,2$TKS ; RE-ENABLE TTY KBD INTERRUPTS
15$: RTI ; RETURN
16$: JSR PC,$TYPEC ; ECHO CHAR
; CMP (SP),#60 ; CHAR < 0?
; BLT 18$ ; BRANCH IF YES
; CMP (SP),#67 ; CHAR > 7?
; BGT 18$ ; BRANCH IF YES
; BIC #60,(SP)+ ; STRIP-OFF ASCII
; TST 2(SP) ; IS THIS THE FIRST CHAR
; BEQ 17$ ; BRANCH IF YES
; ASL (SP) ; NO, SHIFT PRESENT
; ASL (SP) ; CHAR OVER TO MAKE
; ASL (SP) ; ROOM FOR NEW ONE.
17$: INC 2(SP) ; KEEP COUNT OF CHAR
; BIS -2(SP),(SP) ; SET IN NEW CHAR
; BR 7$ ; GET THE NEXT ONE
18$: TYPE $QUES ; TYPE ?<CR><LF>
; BR 20$ ; SIMULATE CONTROL-U
.DSABL LSB

```

```

;*****
;THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
;CALL:
; RDCHR ; INPUT A SINGLE CHARACTER FROM THE TTY
; RETURN HERE ; CHARACTER IS ON THE STACK
; ; WITH PARITY BIT STRIPPED OFF
;

```

```

$RDCHR: MOV (SP),-(SP) ; PUSH DOWN THE PC
; MOV 4(SP),2(SP) ; SAVE THE PS
1$: TSTB 2$TKS ; WAIT FOR
; BPL 1$ ; A CHARACTER
; MOVB 2$TKB,4(SP) ; READ THE TTY
; BIC #177,4(SP) ; GET RID OF JUNK IF ANY
; CMP 4(SP),#23 ; IS IT A CONTROL-5?
; BNE 3$ ; BRANCH IF NO
2$: TSTB 2$TKS ; WAIT FOR A CHARACTER
; BPL 2$ ; LOOP UNTIL ITS THERE
; MOVB 2$TKB,-(SP) ; GET CHARACTER
; BIC #177,(SP) ; MAKE IT 7-BIT ASCII
; CMP (SP)+,#21 ; IS IT A CONTROL-Q?
; BNE 2$ ; IF NOT DISCARD IT
; BR 1$ ; YES, RESUME
3$: CMP 4(SP),#140 ; IS IT UPPER CASE?
; BLT 4$ ; BRANCH IF YES
; CMP 4(SP),#175 ; IS IT A SPECIAL CHAR?
; BGT 4$ ; BRANCH IF YES
; BIC #40,4(SP) ; MAKE IT UPPER CASE
4$: RTI ; GO BACK TO USER

```

```

;*****
;THIS ROUTINE WILL INPUT A STRING FROM THE TTY

```

JOB

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 100
TTY INPUT ROUTINE

4734									
4735									
4736									
4737									
4738									
4739	036560	010346							
4740	036562	005046							
4741	036564	012703	037036						
4742	036570	022703	037046						
4743	036574	101467							
4744	036576	104411							
4745	036600	112613							
4746	036602	122713	000003						
4747	036606	001006							
4748	036610	104401	037046						
4749	036614	005726							
4750	036616	012603							
4751	036620	000137	037110						
4752	036624	122713	000177						
4753	036630	001022							
4754	036632	005716							
4755	036634	001007							
4756	036636	112737	000134	037034					
4757	036644	104401	037034						
4758	036650	012716	177777						
4759	036654	005303							
4760	036656	020327	037036						
4761	036662	103434							
4762	036664	111337	037034						
4763	036670	104401	037034						
4764	036674	000735							
4765	036676	005716							
4766	036700	001406							
4767	036702	112737	000134	037034					
4768	036710	104401	037034						
4769	036714	005016							
4770	036716	122713	000025						
4771	036722	001003							
4772	036724	104401	037053						
4773	036730	000715							
4774	036732	122713	000022						
4775	036736	001011							
4776	036740	105013							
4777	036742	104401	001223						
4778	036746	104401	037036						
4779	036752	000706							
4780	036754	104401	001222						
4781	036760	000701							
4782	036762	111337	037034						
4783	036766	104401	037034						
4784	036772	122723	000015						
4785	036776	001274							
4786	037000	105063	177777						
4787	037004	104401	001224						
4788	037010	005726							
4789	037012	012603							

```

;*CALL:
;*      RDLIN
;*      RETURN HERE
;*
SRDLIN: MOV      R3 -(SP)
          CLR      -(SP)
1$:      MOV      #STTYIN,R3
2$:      CMP      #STTYIN+8.,R3
          BLOS     4$
          RDCHR
          MOV      (SP)+,(R3)
          CMP      #3,(R3)
          BNE     10$
          TYPE     ,SCNTLC
          TST      (SP)+
          MOV      (SP)+,R3
          JMP      CNTRLC
10$:     CMP      #177,(R3)
          BNE     5$
          TST      (SP)
          BNE     6$
          MOV      #' \ ,9$
          TYPE     ,9$
          MOV      #-1,(SP)
6$:      DEC      R3
          CMP      R3,#STTYIN
          BLO     4$
          MOV      (R3),9$
          TYPE     ,9$
          BR      2$
5$:      TST      (SP)
          BEQ     7$
          MOV      #' \ ,9$
          TYPE     ,9$
          CLR      (SP)
7$:      CMP      #25,(R3)
          BNE     8$
          TYPE     ,SCNTLU
          BR      1$
8$:      CMP      #22,(R3)
          BNE     3$
          CLRB    (R3)
          TYPE     ,SCRLF
          TYPE     ,STTYIN
          BR      2$
4$:      TYPE     ,SQUES
          BR      1$
3$:      MOV      (R3),9$
          TYPE     ,9$
          CMP      #15,(R3)+
          BNE     2$
          CLRB    -1(R3)
          TYPE     ,SLF
          TST      (SP)+
          MOV      (SP)+,R3
          ;; INPUT A STRING FROM THE TTY
          ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
          ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
          ;; SAVE R3
          ;; CLEAR THE RUBOUT KEY
          ;; GET ADDRESS
          ;; BUFFER FULL?
          ;; BR IF YES
          ;; GO READ ONE CHARACTER FROM THE TTY
          ;; GET CHARACTER
          ;; IS IT A CONTROL-C?
          ;; BRANCH IF NO
          ;; TYPE A CONTROL-C (↑C)
          ;; CLEAN RUBOUT KEY OFF OF THE STACK
          ;; RESTORE R3
          ;; GOTO CONTROL-C RESTART
          ;; IS IT A RUBOUT
          ;; BR IF NO
          ;; IS THIS THE FIRST RUBOUT?
          ;; BR IF NO
          ;; TYPE A BACK SLASH
          ;; SET THE RUBOUT KEY
          ;; BACKUP BY ONE
          ;; STACK EMPTY?
          ;; BR IF YES
          ;; SETUP TO TYPEOUT THE DELETED CHAR.
          ;; GO TYPE
          ;; GO READ ANOTHER CHAR.
          ;; RUBOUT KEY SET?
          ;; BR IF NO
          ;; TYPE A BACK SLASH
          ;; CLEAR THE RUBOUT KEY
          ;; IS CHARACTER A CTRL U?
          ;; BR IF NO
          ;; TYPE A CONTROL "U"
          ;; GO START OVER
          ;; IS CHARACTER A "↑R"?
          ;; BRANCH IF NO
          ;; CLEAR THE CHARACTER
          ;; TYPE A "CR" & "LF"
          ;; TYPE THE INPUT STRING
          ;; GO PICKUP ANOTHER CHARACTER
          ;; TYPE A ' '
          ;; CLEAR THE BUFFER AND LOOP
          ;; ECHO THE CHARACTER
          ;; CHECK FOR RETURN
          ;; LOOP IF NOT RETURN
          ;; CLEAR RETURN (THE 15)
          ;; TYPE A LINE FEED
          ;; CLEAN RUBOUT KEY FROM THE STACK
          ;; RESTORE R3

```

```

4790 037014 011646
4791 037016 016666 000004 000002
4792 037024 012766 037036 000004
4793 037032 000002
4794 037034 000
4795 037035 000
4796 037036 000010
4797 037046 041536 005015 000
4798 037053 136 006525 000012
4799 037060 043536 005015 000
4800 037065 015 051412 051127
4801 037072 036440 000040
4802 037076 020040 042516 020127
4803 037104 020075 000
4804 037110
4805
4806
4807
4808
4809

```

```

MOV (SP), -(SP) ;; ADJUST THE STACK AND PUT ADDRESS OF THE
MOV 4(SP), 2(SP) ;; FIRST ASCII CHARACTER ON IT
MOV #STTYIN, 4(SP)
RTI ;; RETURN
9S: .BYTE 0 ;; STORAGE FOR ASCII CHAR. TO TYPE
.BYTE 0 ;; TERMINATOR
$TTYIN: .BLKB 8. ;; RESERVE 8 BYTES FOR TTY INPUT
$CNTLC: .ASCIZ /↑C/<15><12> ;; CONTROL "C"
$CNTLU: .ASCIZ /↑U/<15><12> ;; CONTROL "U"
$CNTLG: .ASCIZ /↑G/<15><12> ;; CONTROL "G"
$MSWR: .ASCIZ <15><12>/SWR = /
$MNEW: .ASCIZ / NEW = /

.EVEN

.SBTTL CONTROL-C SERVICING ROUTINE

;* THE FOLLOWING CODE IS EXECUTED WHEN A CONTROL-C HAS
;* BEEN TYPED INSTEAD OF A NEW SWITCH REG. VALUE.

```

```

4810      ;*      (IN OTHER WORDS, AFTER A CONTROL-G WAS TYPED).
4811      ;*      A NEW SWITCH REG. VALUE WILL BE ASKED FOR,
4812      ;*      THE TEST NUMBER AND PASS NUMBER WILL BE TYPED,
4813      ;*      AND THEN THE PROGRAM WILL GO TO "END-OF-PASS" AND CONTINUE
4814
4815 037110 013737 001234 001210 CNTRLC: MOV      $PASS,$STMP5 ;GET THE VALUE OF "$PASS"
4816 037116 005237 001210          INC      $STMP5 ;FORM CURRENT PASS NO.
4817 037122 104401 037167          TYPE     ,CMMSG ;TYPE THE TEST STOPS MESSAGE
4818 037126 113737 001102 037162 MOVVB   $STNM,1$ ;SAVE THE TEST NUMBER
4819 037134 013746 037162          MOV      1$,-(SP) ;SAVE 1$ FOR TYPEOUT
4820 037140 104402          TYPOC   ;GO TYPE--OCTAL ASCII(ALL DIGITS)
4821 037142 104401 037164          TYPE     2$ ;TYPE 2 SPACES
4822 037146 013746 001210          MOV      $STMP5,-(SP) ;SAVE $STMP5 FOR TYPEOUT
4823 037152 104405          TYPDS   ;GO TYPE--DECIMAL ASCII WITH SIGN
4824 037154 104407          GTSWR   ;ASK FOR NEW SWR VALUE
4825 037156 000137 034204          JMP      $EOP+2 ;CONTINUE AT $EOP+2
4826 037162 000000          1$:      .WORD   0 ;BUFFER FOR TEST NUMBER
4827 037164 020040          2$:      .ASCII  / / ;TWO SPACES AND THE STOP MESSAGE
4828 037167          112 046525 044520 MSG:      .ASCII  /JUMPING TO END-OF-PASS/<15><12>
4829 037174 043516 052040          000
4830 037202 047105 026504          000
4831 037210 050055 051501 006523          .ASCII  /TESTNO PASSNO/<15><12>
4832 037216          012
4833 037217          124 051505 047124          .ASCII  /TESTNO PASSNO/<15><12>
4834 037224 020117 050040 051501
4835 037232 047123 006517 000012
4836
4837      .EVEN
4838      .SBTTL TYPE ROUTINE
4839
4840      ;*****
4841      ;ROUTINE TO TYPE ASCII MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
4842      ;THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
4843      ;NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
4844      ;NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
4845      ;NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
4846      ;
4847      ;CALL:
4848      ;1) USING A TRAP INSTRUCTION
4849      ;      TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCII STRING
4850      ;OR
4851      ;      TYPE
4852      ;      MESADR
4853      ;
4854 037240 105737 001157 $TYPE: TSTB   $TFPLG ;: IS THERE A TERMINAL?
4855 037244 100002          BPL     1$ ;: BR IF YES
4856 037246 000000          HALT   ;: HALT HERE IF NO TERMINAL
4857 037250 000430          BR     3$ ;: LEAVE
4858 037252 010046          1$:      MOV     RO,-(SP) ;: SAVE RO
4859 037254 017600 000002          MOV     @2(SP),RO ;: GET ADDRESS OF ASCII STRING
4860 037260 122737 000001 001246          CMPB   #APTENV,$ENV ;: RUNNING IN APT MODE
4861 61 037266 001011          BNE    62$ ;: NO, GO CHECK FOR APT CONSOLE
4862 037270 132737 000100 001247          BITB   #APTPOOL,$ENVM ;: SPOOL MESSAGE TO APT
4863 037276 001405          BEQ    62$ ;: NO, GO CHECK FOR CONSOLE
4864 037300 010037 037310          MOV     RO,61$ ;: SETUP MESSAGE ADDRESS FOR APT
4865 037304 004737 037530          JSR    PC,$ATY3 ;: SPOOL MESSAGE TO APT

```

```

4866 037310 000000
4867 037312 132737 000040 001247 61$: WORD 0      ;; MESSAGE ADDRESS
4868 037320 001003 62$: BITB #APTC SUP, $ENV M ;; APT CONSOLE SUPPRESSED
4869 037322 112046 2$: MOV BNE 60$      ;; YES, SKIP TYPE OUT
4870 037324 001005 4$: (RD)+, -(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
4871 037326 005726 TST 4$      ;; BR IF IT ISN'T THE TERMINATOR
4872 037330 012600 60$: (SP)+, RD ;; IF TERMINATOR POP IT OFF THE STACK
4873 037332 062716 000002 3$: ADD #2, (SP) ;; RESTORE RD
4874 037336 000002 RTI      ;; ADJUST RETURN PC
4875 037340 122716 000011 4$: CMPB #HT, (SP) ;; RETURN
4876 037344 001430 BEQ 8$      ;; BRANCH IF <HT>
4877 037346 122716 000200 5$: CMPB #CRLF, (SP) ;; BRANCH IF NOT <CRLF>
4878 037352 001006 BNE 5$
4879 037354 005726 TST (SP)+ ;; POP <CR><LF> EQUIV
4880 037356 104401 TYPE ;; TYPE A CR AND LF
4881 037360 001223 $CRLF
4882 037362 105037 037516 CLR B $CHARCNT ;; CLEAR CHARACTER COUNT
4883 037366 000755 BR 2$      ;; GET NEXT CHARACTER
4884 037370 004737 037452 5$: JSR PC, $TYPE C ;; GO TYPE THIS CHARACTER
4885 037374 123726 001156 6$: CMPB $FILLC, (SP)+ ;; IS IT TIME FOR FILLER CHARS.?
4886 037400 001350 BNE 2$      ;; IF NO GO GET NEXT CHAR.
4887 037402 013746 001154 MOV #NULL, -(SP) ;; GET # OF FILLER CHARS. NEEDED
4888 037406 105366 000001 7$: DECB 1(SP) ;; AND THE NULL CHAR.
4889 037412 002770 BLT 6$      ;; DOES A NULL NEED TO BE TYPED?
4890 037414 004737 037452 JSR PC, $TYPE C ;; BR IF NO--GO POP THE NULL OFF OF STACK
4891 037420 105337 037516 DECB $CHARCNT ;; GO TYPE A NULL
4892 037424 000770 BR 7$      ;; DO NOT COUNT AS A COUNT
4893
4894
4895
4896
4897 037426 112716 000040 ; HORIZONTAL TAB PROCESSOR
4898 037432 004737 037452 8$: MOV B #', (SP) ;; REPLACE TAB WITH SPACE
4899 037436 132737 000007 037516 9$: JSR PC, $TYPE C ;; TYPE A SPACE
4900 037444 001372 BNE 9$      ;; BRANCH IF NOT AT
4901 037446 005726 TST (SP)+ ;; TAB STOP
4902 037450 000724 BR 2$      ;; POP SPACE OFF STACK
4903 037452 105777 141472 $TYPE C: TSTB $STPS ;; GET NEXT CHARACTER
4904 037456 100375 BPL $TYPE C ;; WAIT UNTIL PRINTER IS READY
4905 037460 116677 000002 141464 MOV B 2(SP), $STPB ;; LOAD CHAR TO BE TYPED INTO DATA REG.
4906 037466 122766 000015 000002 CMPB #CR, 2(SP) ;; IS CHARACTER A CARRIAGE RETURN?
4907 037474 001003 BNE 1$      ;; BRANCH IF NO
4908 037476 105037 037516 CLR B $CHARCNT ;; YES--CLEAR CHARACTER COUNT
4909 037502 000406 BR $TYPE X ;; EXIT
4910 037504 122766 000012 000002 1$: CMPB #LF, 2(SP) ;; IS CHARACTER A LINE FEED?
4911 037512 001402 BEQ $TYPE X ;; BRANCH IF YES
4912 037514 105227 INCB (PC)+ ;; COUNT THE CHARACTER
4913 037516 000000 $CHARCNT: WORD 0 ;; CHARACTER COUNT STORAGE
4914 037520 000207 $TYPE X: RTS PC
4915
4916
4917
4918
4919 037522 112737 000001 037766 .SBTTL APT COMMUNICATIONS ROUTINE
4920 037530 112737 000001 037764 ;; *****
SATY1: MOV B #1, $FFLG ;; TO REPORT FATAL ERROR
SATY3: MOV B #1, $MFLG ;; TO TYPE A MESSAGE
4921 037536 000403 BR $ATYC

```



```

4922 037540 112737 000001 037766 $ATY4:  MOVB  #1,$FFLG  ;; TO ONLY REPORT FATAL ERROR
4923 037546 010046 000000 037766 $ATYC:  MOV  RO,-(SP)  ;; PUSH RO ON STACK
4924 037546 010146 000000 037766 MOV  RI,-(SP)  ;; PUSH RI ON STACK
4925 037550 105737 037764 TSTB $MFLG  ;; SHOULD TYPE A MESSAGE?
4926 037552 001450 000000 037764 BEQ  5$  ;; IF NOT: BR
4927 037556 122737 000001 001246 CMPB $APTENV,$ENV  ;; OPERATING UNDER APT?
4928 037560 001031 000000 001246 BNE  3$  ;; IF NOT: BR
4929 037570 132737 000100 001247 BITB $APTSPool,$ENVm  ;; SHOULD SPOOL MESSAGES?
4930 037576 001425 000000 001247 BEQ  3$  ;; IF NOT: BR
4931 037600 017600 000004 000004 MOV  #4(SP),RO  ;; GET MESSAGE ADDR.
4932 037604 062766 000002 000004 ADD  #2,4(SP)  ;; BUMP RETURN ADDR.
4933 037612 005737 001226 15:  TST  $MSGTYPE  ;; SEE IF DONE W/ LAST XMISSION?
4934 037616 001375 001242 15:  BNE  1$  ;; IF NOT: WAIT
4935 037620 010037 001242 25:  MOV  RO,$MSGAD  ;; PUT ADDR IN MAILBOX
4936 037624 105720 001242 25:  TSTB (RO)+  ;; FIND END OF MESSAGE
4937 037626 001376 001242 25:  BNE  2$  ;; SUB START OF MESSAGE
4938 037630 163700 001242 25:  SUB  $MSGAD,RO  ;; GET MESSAGE LNTH IN WORDS
4939 037634 006200 001244 25:  ASR  RO  ;; PUT LENGTH IN MAILBOX
4940 037636 010037 000004 001226 25:  MOV  RO,$MSGGLT  ;; TELL APT TO TAKE MSG.
4941 037642 012737 000004 001226 25:  MOV  #4,$MSGTYPE
4942 037642 000413 000004 037676 35:  BR  5$  ;; PUT MSG ADDR IN JSR LINKAGE
4943 037650 017637 000002 000004 35:  MOV  #4(SP),4$  ;; BUMP RETURN ADDRESS
4944 037652 062766 177776 037240 35:  ADD  #2,4(SP)  ;; PUSH 177776 ON STACK
4945 037660 013746 037240 35:  MOV  177776,-(SP)  ;; CALL TYPE MACRO
4946 037672 004737 000000 35:  JSR  PC,$TYPE
4947 037676 000000 35:  .WORD 0
4948 037700 105737 037766 105:  TSTB $FFLG  ;; SHOULD REPORT FATAL ERROR?
4949 037704 001416 001246 105:  BEQ  12$  ;; IF NOT: BR
4950 037706 005737 001226 105:  TST  $ENV  ;; RUNNING UNDER APT?
4951 037712 001413 001226 105:  BEQ  12$  ;; IF NOT: BR
4952 037714 005737 000004 001230 115:  TST  $MSGTYPE  ;; FINISHED LAST MESSAGE?
4953 037720 001375 000002 000004 115:  BNE  11$  ;; IF NOT: WAIT
4954 037722 017637 000002 000004 115:  MOV  #4(SP),$FATAL  ;; GET ERROR #
4955 037730 062766 001226 000004 115:  ADD  #2,4(SP)  ;; BUMP RETURN ADDR.
4956 037736 005237 001226 125:  INC  $MSGTYPE  ;; TELL APT TO TAKE ERROR
4957 037742 105037 037766 125:  CLRB $FFLG  ;; CLEAR FATAL FLAG
4958 037746 105037 037765 125:  CLRB $LFLG  ;; CLEAR LOG FLAG
4959 037752 105037 037764 125:  CLRB $MFLG  ;; CLEAR MESSAGE FLAG
4960 037756 012601 037764 125:  MOV  (SP)+,R1  ;; POP STACK INTO R1
4961 037760 012600 037764 125:  MOV  (SP)+,RO  ;; POP STACK INTO RO
4962 037762 000207 037764 125:  RTS  PC  ;; RETURN
4963 037764 000 037764 125:  $MFLG: .BYTE 0  ;; MESSG. FLAG
4964 037765 000 037764 125:  $LFLG: .BYTE 0  ;; LOG FLAG
4965 037766 000 037764 125:  $FFLG: .BYTE 0  ;; FATAL FLAG
4966 037770 000 037770 125:  .EVEN
4967 000200 APTSIZE=200
4968 000001 APTENV=001
4969 000100 APTSPool=100
4970 000040 APTCSUP=040
4971 .SBTTL BINARY TO ASCII AND TYPE ROUTINE
4972 *****
4973 ; THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
4974 ; BINARY-ASCII NUMBER AND TYPE IT.

```

```

4978 ;*CALL:
4979 ;*      MOV      NUMBER,-(SP)      ;;NUMBER TO BE TYPED
4980 ;*      TYPBN
4981 ;*
4982 $TYPBN: MOV      R1,-(SP)          ;;SAVE R1 ON THE STACK
4983         MOV      6(SP),R1         ;;GET THE INPUT NUMBER
4984         SEC
4985         MOVVB   #'0,$BIN          ;;SET "C" SO CAN KEEP TRACK OF THE NUMBER OF BITS
4986         ROL      R1               ;;SET CHARACTER TO AN ASCII "0".
4987         BEQ     2$                ;;GET THIS BIT
4988         ADCB    $BIN              ;;DONE?
4989         TYPE    , $BIN            ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
4990         CLC
4991         BR      1$                ;;GO TYPE THIS BIT
4992         MOV     (SP)+,R1           ;;CLEAR "C" SO CAN KEEP TRACK OF BITS
4993         MOV     2(SP),4(SP)       ;;POP THE STACK INTO R1
4994         MOV     (SP)+,(SP)        ;;ADJUST THE STACK
4995         RTI
4996 $BIN:   .BYTE   0,0               ;;RETURN TO USER
4997         .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
4998
4999 *****
5000 *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
5001 *OCTAL (ASCII) NUMBER AND TYPE IT.
5002 *$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
5003 *CALL:
5004 *      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
5005 *      TYPOS
5006 *      .BYTE   N                   ;;CALL FOR TYPEOUT
5007 *      .BYTE   M                   ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
5008 *                                     ;;M=1 OR 0
5009 *                                     ;;1=TYPE LEADING ZEROS
5010 *                                     ;;0=SUPPRESS LEADING ZEROS
5011 *
5012 *$STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
5013 *$TYPOS OR $TYPOC
5014 *CALL:
5015 *      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
5016 *      TYPON
5017 *                                     ;;CALL FOR TYPEOUT
5018 *
5019 *$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
5020 *CALL:
5021 *      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
5022 *      TYPOC
5023 *                                     ;;CALL FOR TYPEOUT
5024
5025 $TYPOS: MOV      2(SP),-(SP)        ;;PICKUP THE MODE
5026         MOVVB   1(SP),$OFILL      ;;LOAD ZERO FILL SWITCH
5027         MOVVB   (SP)+,$OMODE+1    ;;NUMBER OF DIGITS TO TYPE
5028         ADD     #2,(SP)           ;;ADJUST RETURN ADDRESS
5029         RR      $TYPON
5030
5031 $TYPOC: MOVVB   #1,$OFILL          ;;SET THE ZERO FILL SWITCH
5032         MOVVB   #6,$OMODE+1       ;;SET FOR SIX(6) DIGITS
5033         MOVVB   #5,$OCNT          ;;SET #E ITERATION COUNT
5034         MOV     R3,-(SP)           ;;SAVE R3
5035         MOV     R4,-(SP)           ;;SAVE R4
5036         MOV     R5,-(SP)           ;;SAVE R5
5037         MOVVB   $OMODE+1,R4       ;;GET THE NUMBER OF DIGITS TO TYPE

```

```

5034 040124 005404          NEG      R4
5035 040126 062704 000006  ADD      #6,R4          ;; SUBTRACT IT FOR MAX. ALLOWED
5036 040132 110437 040270  MOVVB   R4,$OMODE     ;; SAVE IT FOR USE
5037 040136 113704 040267  MOVVB   $OFILL,R4     ;; GET THE ZERO FILL SWITCH
5038 040142 016605 000012  MOV     12(SP),R5     ;; PICKUP THE INPUT NUMBER
5039 040146 005003          CLR     R3           ;; CLEAR THE OUTPUT WORD
5040 040150 006105          15:    ROL     R5           ;; ROTATE MSB INTO "C"
5041 040152 000404          BR     3$           ;; GO DO MSB
5042 040154 006105          25:    ROL     R5           ;; FORM THIS DIGIT
5043 040156 006105          ROL     R5
5044 040160 006105          ROL     R5
5045 040162 010503          MOV     R5,R3
5046 040164 006103          35:    ROL     R3           ;; GET LSB OF THIS DIGIT
5047 040166 105337 040270  DECB   $OMODE         ;; TYPE THIS DIGIT?
5048 040172 100016          BPL    7$           ;; BR IF NO
5049 040174 042703 177770  BIC    #177770,R3     ;; GET RID OF JUNK
5050 040200 001002          BNE    4$           ;; TEST FOR 0
5051 040202 005704          TST   R4           ;; SUPPRESS THIS 0?
5052 040204 001403          BEQ   5$           ;; BR IF YES
5053 040206 005204          45:    INC   R4           ;; DON'T SUPPRESS ANYMORE 0'S
5054 040210 052703 000060  BIS   #'0,R3         ;; MAKE THIS DIGIT ASCII
5055 040214 052703 000040  55:    BIS   #' ,R3     ;; MAKE ASCII IF NOT ALREADY
5056 040220 110337 040264  MOVVB  R3,$S         ;; SAVE FOR TYPING
5057 040224 104401 040264  TYPE   $S           ;; GO TYPE THIS DIGIT
5058 040230 105337 040266  75:    DECB  $OCNT     ;; COUNT BY 1
5059 040234 003347          BGT   2$           ;; BR IF MORE TO DO
5060 040236 002402          BLT   6$           ;; BR IF DONE
5061 040240 005204          INC   R4           ;; INSURE LAST DIGIT ISN'T A BLANK
5062 040242 000744          BR    2$           ;; GO DO THE LAST DIGIT
5063 040244 012605          65:    MOV   (SP)+,R5     ;; RESTORE R5
5064 040246 012604          MOV   (SP)+,R4     ;; RESTORE R4
5065 040250 012603          MOV   (SP)+,R3     ;; RESTORE R3
5066 040252 016666 000002 000004  MOV   2(SP),4(SP)  ;; SET THE STACK FOR RETURNING
5067 040260 012616          MOV   (SP)+,(SP)
5068 040262 000002          RTI                    ;; RETURN
5069 040264 000          85:    .BYTE 0          ;; STORAGE FOR ASCII DIGIT
5070 040265 000          .BYTE 0          ;; TERMINATOR FOR TYPE ROUTINE
5071 040266 000          $OCNT: .BYTE 0     ;; OCTAL DIGIT COUNTER
5072 040267 000          $OFILL: .BYTE 0    ;; ZERO FILL SWITCH
5073 040270 000000          $OMODE: .WORD 0     ;; NUMBER OF DIGITS TO TYPE
5074          .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
5075
5076          ;; *****
5077          ;; *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
5078          ;; *SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
5079          ;; *NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
5080          ;; *BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
5081          ;; *REPLACED WITH SPACES.
5082          ;; *CALL:
5083          ;; *   MOV     NUM,-(SP)          ;; PUT THE BINARY NUMBER ON THE STACK
5084          ;; *   TYPDS          ;; GO TO THE ROUTINE
5085
5086          $TYPDS:
5087          MOV     R0,-(SP)          ;; PUSH R0 ON STACK
5088          MOV     R1,-(SP)          ;; PUSH R1 ON STACK
5089          MOV     R2,-(SP)          ;; PUSH R2 ON STACK

```

```

5090 040300 010346          MOV      R3,-(SP)          ;; PUSH R3 ON STACK
5091 040302 010546          MOV      R5,-(SP)          ;; PUSH R5 ON STACK
5092 040304 012746 020200    MOV      #20200,-(SP)      ;; SET BLANK SWITCH AND SIGN
5093 040310 016605 000020    MOV      20(SP),R5        ;; GET THE INPUT NUMBER
5094 040314 100004          BPL      1$                ;; BR IF INPUT IS POS.
5095 040316 005405          NEG      R5                ;; MAKE THE BINARY NUMBER POS.
5096 040320 112766 000055 000001  MOVB     #'-,1(SP)        ;; MAKE THE ASCII NUMBER NEG.
5097 040326 005000          CLR      R0                ;; ZERO THE CONSTANTS INDEX
5098 040330 012703 040506 1$:    MOV      #SDBLK,R3        ;; SETUP THE OUTPUT POINTER
5099 040334 112723 000040    MOVB     #'',(R3)+        ;; SET THE FIRST CHARACTER TO A BLANK
5100 040340 005002          CLR      R2                ;; CLEAR THE BCD NUMBER
5101 040342 016001 040476 2$:    MOV      $DTBL(R0),R1     ;; GET THE CONSTANT
5102 040346 160105 3$:    SUB      R1,R5            ;; FORM THIS BCD DIGIT
5103 040350 002402          BLT     4$                ;; BR IF DONE
5104 040352 005202          INC     R2                ;; INCREASE THE BCD DIGIT BY 1
5105 040354 000774          BR      3$
5106 040356 060105 4$:    ADD      R1,R5            ;; ADD BACK THE CONSTANT
5107 040360 005702          TST     R2                ;; CHECK IF BCD DIGIT=0
5108 040362 001002          BNE     5$                ;; FALL THROUGH IF 0
5109 040364 105716          TSTB   (SP)              ;; STILL DOING LEADING 0'S?
5110 040366 100407          BMI     7$                ;; BR IF YES
5111 040370 106316 5$:    ASLB   (SP)              ;; MSD?
5112 040372 103003          BCC     6$                ;; BR IF NO
5113 040374 116663 000001 177777  MOVB     1(SP),-1(R3)     ;; YES--SET THE SIGN
5114 040402 052702 000060 6$:    BIS     #'0,R2           ;; MAKE THE BCD DIGIT ASCII
5115 040406 052702 000040 7$:    BIS     #' ,R2           ;; MAKE IT A SPACE IF NOT ALREADY A DIGIT
5116 040412 110223          MOVB   R2,(R3)+          ;; PUT THIS CHARACTER IN THE OUTPUT BUFFER
5117 040414 005720          TST    (R0)+            ;; JUST INCREMENTING
5118 040416 020027 000010    CMP     R0,#10          ;; CHECK THE TABLE INDEX
5119 040422 002746          BLT    2$                ;; GO DO THE NEXT DIGIT
5120 040424 003002          BGT    8$                ;; GO TO EXIT
5121 040426 010502          MOV    R5,R2            ;; GET THE LSD
5122 040430 000764          BR     6$                ;; GO CHANGE TO ASCII
5123 040432 105726 8$:    TSTB  (SP)+            ;; WAS THE LSD THE FIRST NON-ZERO?
5124 040434 100003          BPL    9$                ;; BR IF NO
5125 040436 116663 177777 177776  MOVB   -1(SP),-2(R3)     ;; YES--SET THE SIGN FOR TYPING
5126 040444 105013 9$:    CLRB  (R3)              ;; SET THE TERMINATOR
5127 040446 012605          MOV    (SP)+,R5         ;; POP STACK INTO R5
5128 040450 012603          MOV    (SP)+,R3         ;; POP STACK INTO R3
5129 040452 012602          MOV    (SP)+,R2         ;; POP STACK INTO R2
5130 040454 012601          MOV    (SP)+,R1         ;; POP STACK INTO R1
5131 040456 012600          MOV    (SP)+,R0         ;; POP STACK INTO R0
5132 040460 104401 040506    TYPE   $SDBLK           ;; NOW TYPE THE NUMBER
5133 040464 016666 000002 000004  MOV     2(SP),4(SP)      ;; ADJUST THE STACK
5134 040472 012616          MOV    (SP)+,(SP)
5135 040474 000002          RTI
5136 040476 023420          $DTBL: 10000.
5137 040500 001750          1000.
5138 040502 000144          100.
5139 040504 000012          10.
5140 040506 000004          $SDBLK: .BLKW 4
5141          .SBTTL SAVE AND RESTORE R0-R5 ROUTINES
5142
5143          ;; *****
5144          ;; *SAVE R0-R5
5145          ;; *CALL:

```

```

5146
5147
5148
5149
5150
5151
5152
5153
5154
5155
5156
5157
5158 040516
5159 040516 010046
5160 040520 010146
5161 040522 010246
5162 040524 010346
5163 040526 010446
5164 040530 010546
5165 040532 016646 000022
5166 040536 016646 000022
5167 040542 016646 000022
5168 040546 016646 000022
5169 040552 000002
5170
5171
5172
5173
5174 040554
5175 040554 012666 000022
5176 040560 012666 000022
5177 040564 012666 000022
5178 040570 012666 000022
5179 040574 012605
5180 040576 012604
5181 040600 012603
5182 040602 012602
5183 040604 012601
5184 040606 012600
5185 040610 000002
5186
5187
5188
5189
5190
5191
5192
5193
5194
5195
5196
5197 040612 104413
5198 040614 016601 000002
5199 040620 012705 040731
5200 040624 012704 000014
5201 040630 012703 177770

```

```

;* SAVREG
;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
;*
;*TOP---(+16)
;* +2---(+18)
;* +4---R5
;* +6---R4
;* +8---R3
;*+10---R2
;*+12---R1
;*+14---R0

```

```

$SAVREG:
MOV RO, -(SP) ;; PUSH RO ON STACK
MOV R1, -(SP) ;; PUSH R1 ON STACK
MOV R2, -(SP) ;; PUSH R2 ON STACK
MOV R3, -(SP) ;; PUSH R3 ON STACK
MOV R4, -(SP) ;; PUSH R4 ON STACK
MOV R5, -(SP) ;; PUSH R5 ON STACK
MOV 22(SP), -(SP) ;; SAVE PS OF MAIN FLOW
MOV 22(SP), -(SP) ;; SAVE PC OF MAIN FLOW
MOV 22(SP), -(SP) ;; SAVE PS OF CALL
MOV 22(SP), -(SP) ;; SAVE PC OF CALL
RTI

```

```

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

```

.SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

```

;*****
;THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
;UNSIGNED OCTAL ASCII NUMBER.
;*CALL
;* MOV #PNTR, -(SP) ;; POINTER TO LOW WORD OF BINARY NUMBER
;* JSR PC, #SDB20 ;; CALL THE ROUTINE
;* RETURN ;; THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK

```

```

$SDB20: SAVREG ;; SAVE ALL REGISTERS
MOV 2(SP), R1 ;; PICKUP THE POINTER TO LOW WORD
MOV #SDB20+13., R5 ;; POINTER TO DATA TABLE
MOV #12., R4 ;; DO ELEVEN CHARACTERS
MOV #17., R3 ;; MASK

```

5202	040634	012100		MOV	(R1)+,R0	:: LOWER WORD
5203	040636	012101		MOV	(R1)+,R1	:: HIGH WORD
5204	040640	005002		CLR	R2	:: TERMINATOR
5205	040642	110245		1S: MOVB	R2,-(R5)	:: PUT CHARACTER IN DATA TABLE
5206	040644	010002		MOV	R0,R2	:: GET THIS DIGIT
5207	040646	005304		DEC	R4	:: COUNT THIS CHARACTER
5208	040650	003007		BGT	3\$:: BR IF NOT THE LAST DIGIT
5209	040652	001405		BEQ	2\$:: BR IF IT IS THE LAST DIGIT
5210	040654	005205		INC	R5	:: ALL DIGITS DONE-ADJUST POINTER FOR FIRST
5211	040656	010566	000002	MOV	R5,2(SP)	:: ASCIZ CHAR. & PUT IT ON THE STACK
5212	040662	104414		RESREG		:: RESTORE ALL REGISTERS
5213	040664	000207		RTS	PC	:: RETURN TO USER
5214	040666	006203		2\$: ASR	R3	:: POSITION THE MASK FOR THE LAST DIGIT
5215	040670	006001		3\$: ROR	R1	:: POSITION THE BINARY NUMBER FOR
5216	040672	006000		ROR	R0	:: THE NEXT OCTAL DIGIT
5217	040674	006001		ROR	R1	
5218	040676	006000		ROR	R0	
5219	040700	006001		ROR	R1	
5220	040702	006000		ROR	R0	
5221	040704	040302		BIC	R3,R2	:: MASK OUT ALL JUNK
5222	040706	062702	000060	ADD	#0,R2	:: MAKE THIS CHAR. ASCII
5223	040712	000753		BR	1\$:: GO PUT IT IN THE DATA TABLE
5224	040714	000016		\$OCTVL: .BLKB	14.	:: RESERVE DATA TABLE

.SBTTL TRAP DECODER

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

5225
5226
5227
5228
5229
5230
5231
5232
5233 040732 010046
5234 040734 016600 000002
5235 040740 005740
5236 040742 111000
5237 040744 006300
5238 040746 016000 040756
5239 040752 000200
5240
5241
5242
5243
5244 040754 011646
5245 040756 016666 000004 000002
5246 040764 000002
5247
5248
5249
5250
5251
5252
5253
5254
5255 040766 040754
5256 040770 037240
5257 040772 040070
5258 040774 040044
5259 040776 040104
5260 041000 040272
5261 041002 037770
5262
5263 041004 036166
5264
5265 041006 036116
5266 041010 036440
5267 041012 036560
5268 041014 040516
5269 041016 040554
5270
5271
5272
5273
5274 041020 012737 041176 000024
5275 041026 012737 000340 000026
5276 041034 010046
5277 041036 010146
5278 041040 010246
5279 041042 010346
5280 041044 010446

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

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

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

.SBTTL TRAP TABLE

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

ROUTINE

\$TRPAD: .WORD \$TRAP2
\$TYPE ;: CALL=TYPE TRAP+1(104401) TTY TIMEOUT ROUTINE
\$TYPOC ;: CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
\$TYPOS ;: CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
\$TYPON ;: CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
\$TYPDS ;: CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
\$TYPBN ;: CALL=TYPBN TRAP+6(104406) TYPE BINARY (ASCII) NUMBER

\$GTSWR ;: CALL=GTSWR TRAP+7(104407) GET SOFT-SWR SETTING

\$CKSWR ;: CALL=CKSWR TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
\$RDCHR ;: CALL=RDCHR TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
\$RDLIN ;: CALL=RDLIN TRAP+12(104412) TTY TYPEIN STRING ROUTINE
\$SAVREG ;: CALL=SAVREG TRAP+13(104413) SAVE RO-R5 ROUTINE
\$RESREG ;: CALL=RESREG TRAP+14(104414) RESTORE RO-R5 ROUTINE

.SBTTL POWER DOWN AND UP ROUTINES

: POWER DOWN ROUTINE

\$PWRDN: MOV \$SILLUP, 2*\$PWRVEC ;: SET FOR FAST UP
MOV #340, 2*\$PWRVEC+2 ;: PRIO:7
MOV RO, -(SP) ;: PUSH RO ON STACK
MOV R1, -(SP) ;: PUSH R1 ON STACK
MOV R2, -(SP) ;: PUSH R2 ON STACK
MOV R3, -(SP) ;: PUSH R3 ON STACK
MOV R4, -(SP) ;: PUSH R4 ON STACK

```

5281 041046 010546          MOV      R5, -(SP)          ;; PUSH R5 ON STACK
5282 041050 017746 140064    MOV      @SWR, -(SP)       ;; PUSH @SWR ON STACK
5283 041054 010637 041202    MOV      SP, $$SAVR6      ;; SAVE SP
5284 041060 012737 041072 000024  MOV      @SPWRUP, @PWRVEC ;; SET UP VECTOR
5285 041066 000000          HALT
5286 041070 000776          BR      .-2                ;; HANG UP
5287
5288
5289
5290 041072 012737 041176 000024  $PWRUP: MOV      @SILLUP, @PWRVEC ;; SET FOR FAST DOWN
5291 041100 013706 041202          MOV      $$SAVR6, SP      ;; GET SP
5292 041104 005037 041202          CLR      $$SAVR6         ;; WAIT LOOP FOR THE TTY
5293 041110 005237 041202          IS:    INC      $$SAVR6    ;; WAIT FOR THE INC
5294 041114 001375          BNE     IS               ;; OF WORD
5295 041116 012677 140016    MOV      (SP)+, @SWR      ;; POP STACK INTO @SWR
5296 041122 012605          MOV      (SP)+, R5       ;; POP STACK INTO R5
5297 041124 012604          MOV      (SP)+, R4       ;; POP STACK INTO R4
5298 041126 012603          MOV      (SP)+, R3       ;; POP STACK INTO R3
5299 041130 012602          MOV      (SP)+, R2       ;; POP STACK INTO R2
5300 041132 012601          MOV      (SP)+, R1       ;; POP STACK INTO R1
5301 041134 012600          MOV      (SP)+, R0       ;; POP STACK INTO R0
5302 041136 012737 041020 000024  MOV      @SPWRDN, @PWRVEC ;; SET UP THE POWER DOWN VECTOR
5303 041144 012737 000340 000026  MOV      @340, @PWRVEC+2 ;; PRIO:7
5304 041152 104401          TYPE
5305 041154 041204          SPWRMG: .WORD  PWRMSG      ;; REPORT THE POWER FAILURE
5306 041156 012716          MOV      (PC)+, (SP)     ;; POWER FAIL MESSAGE POINTER
5307 041160 020000          SPWRAD: .WORD  START     ;; RESTART AT START
5308 041162 042766 000020 000002  BIC      #20, 2(SP)      ;; RESTART ADDRESS
5309 041170 005037 034462          CLR      $TBIT          ;; CLEAR "T" BIT
5310 041174 000002          RTI
5311 041176 000000          $SILLUP: HALT           ;; THE POWER UP SEQUENCE WAS STARTED
5312 041200 000776          BR      .-2                ;; BEFORE THE POWER DOWN WAS COMPLETE
5313 041202 000000          $$SAVR6: 0                ;; PUT THE SP HERE
5314 041204 006412 050040 053517  PWRMSG: .ASCIZ <12><15>? POWER FAILURE - RESTARTING ?<12><15>
5315 041212 051105 043040 044501
5316 041220 052514 042522 026440
5317 041226 051040 051505 040524
5318 041234 052122 047111 020107
5319 041242 006412 000
5320
5321
5322

```

.EVEN

MD-11-DFKTH-A POP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 112
ERROR MESSAGES, DATA HEADERS-TABLES & FORMATS

5323					.SBTTL	ERROR MESSAGES, DATA HEADERS-TABLES & FORMATS
5324	041246	047125	054105	042520	EM1:	.ASCIZ /UNEXPECTED CPU TRAP TO LOC. 004/
5325	041254	052103	042105	041440		
5326	041262	052520	052040	040522		
5327	041270	020120	047524	046040		
5328	041276	041517	020056	030060		
5329	041304	000064				
5330	041306	047125	054105	042520	EM2:	.ASCIZ /UNEXPECTED MEM. MGMT. TRAP TO LOC. 250/
5331	041314	052103	042105	046440		
5332	041322	046505	020056	043515		
5333	041330	052115	020056	051124		
5334	041336	050101	052040	020117		
5335	041344	047514	027103	031040		
5336	041352	030065	000			
5337	041355	120	044522	051117	EM3:	.ASCIZ /PRIORITY BITS SET WRONG IN PSW/
5338	041362	052111	020131	044502		
5339	041370	051524	051440	052105		
5340	041376	053440	047522	043516		
5341	041404	044440	020116	051520		
5342	041412	000127				
5343	041414	047515	042504	041040	EM4:	.ASCIZ /MODE BITS SET WRONG IN PSW/
5344	041422	052111	020123	042523		
5345	041430	020124	051127	047117		
5346	041436	020107	047111	050040		
5347	041444	053523	000			
5348	041447	104	040525	020114	EM5:	.ASCIZ /DUAL ADDRESSING BETWEEN HI&LO BYTES OF PSW/
5349	041454	042101	051104	051505		
5350	041462	044523	043516	041040		
5351	041470	052105	042527	047105		
5352	041476	044040	023111	047514		
5353	041504	041040	052131	051505		
5354	041512	047440	020106	051520		
5355	041520	000127				
5356	041522	042513	047122	046105	EM6:	.ASCIZ /KERNEL R6 CHANGED BY WRITING USER R6/
5357	041530	051040	020066	044103		
5358	041536	047101	042507	020104		
5359	041544	054502	053440	044522		
5360	041552	044524	043516	052440		
5361	041560	042523	020122	033122		
5362	041566	000				
5363	041567	101	046440	046505	EM7:	.ASCIZ /A MEMORY MGMT. REG. TIMED OUT/
5364	041574	051117	020131	043515		
5365	041602	052115	020056	042522		
5366	041610	027107	052040	046511		
5367	041616	042105	047440	052125		
5368	041624	000				
5369	041625	123	046525	040515	EM10:	.ASCIZ /SUMMARY OF MEM. MGMT. REG. TIMEOUTS/
5370	041632	054522	047440	020106		
5371	041640	042515	027115	046440		
5372	041646	046507	027124	051040		
5373	041654	043505	020056	044524		
5374	041662	042515	052517	051524		
5375	041670	000				
5376	041671	11	046505	020056	EM11:	.ASCIZ /MEM. MGMT. REG. WOULD NOT CLEAR/
5377	041676	043515	052115	020056		
5378	041704	042522	027107	053440		

5379	041712	052517	042114	047040	
5380	041720	052117	041440	042514	
5381	041726	051101	000		
5382	041731	115	046505	020056	EM12: .ASCIZ /MEM. MGMT. REG. BITS NOT SET CORRECTLY/
5383	041736	043515	052115	020056	
5384	041744	042522	027107	041040	
5385	041752	052111	020123	047516	
5386	041760	020124	042523	020124	
5387	041766	047503	051122	041505	
5388	041774	046124	000131		
5389	042000	051123	020060	043105	EM13: .ASCIZ /SR0 EFFECTED BY WRITE TO PSW/
5390	042006	042506	052103	042105	
5391	042014	041040	020131	051127	
5392	042022	052111	020105	047524	
5393	042030	050040	053523	000	
5394	042035	123	030522	042040	EM14: .ASCIZ /SR1 DID NOT READ ALL ZEROS/
5395	042042	042111	047040	052117	
5396	042050	051040	040505	020104	
5397	042056	046101	020114	042532	
5398	042064	047522	000123		
5399	042070	052504	046101	040440	EM15: .ASCIZ /DUAL ADDRESSING BETWEEN BYTES OF PAR OR PDR/
5400	042076	042104	042522	051523	
5401	042104	047111	020107	042502	
5402	042112	053524	042505	020116	
5403	042120	054502	042524	020123	
5404	042126	043117	050040	051101	
5405	042134	047440	020122	042120	
5406	042142	000122			
5407	042144	052504	046101	040440	EM16: .ASCIZ /DUAL ADDRESSING BETWEEN PAR-PDR'S/
5408	042152	042104	042522	051523	
5409	042160	047111	020107	042502	
5410	042166	053524	042505	020116	
5411	042174	040520	026522	042120	
5412	042202	023522	000123		
5413	042206	044120	051531	020056	EM17: .ASCIZ /PHYS. ADDR. FORMED WRONG IN MAINT. MODE/
5414	042214	042101	051104	020056	
5415	042222	047506	046522	042105	
5416	042230	053440	047522	043516	
5417	042236	044440	020116	040515	
5418	042244	047111	027124	046440	
5419	042252	042117	000105		
5420	042256	044120	051531	020056	EM20: .ASCIZ /PHYS. ADDR. FORMED WRONG IN RELOCATE MODE/
5421	042264	042101	051104	020056	
5422	042272	047506	046522	042105	
5423	042300	053440	047522	043516	
5424	042306	044440	020116	042522	
5425	042314	047514	040503	042524	
5426	042322	046440	042117	000105	
5427	042330	026527	044502	020124	EM21: .ASCIZ /W-BIT DID NOT GET SET IN PDR/
5428	042336	044504	020104	047516	
5429	042344	020124	042507	020124	
5430	042352	042523	020124	047111	
5431	042360	050040	051104	000	
5432	042365	127	041055	052111	EM22: .ASCIZ /W-BIT SET IN MORE THAN ONE PDR/
5433	042372	051440	052105	044440	
5434	042400	020116	047515	042522	

5435	042406	052040	040510	020116	
5436	042414	047117	020105	042120	
5437	042422	000122			
5438	042424	026527	044502	020124	EM23: .ASCIZ /W-BIT NOT CLEARED BY WRITING TO PDR/
5439	042432	047516	020124	046103	
5440	042440	040505	042522	020104	
5441	042446	054502	053440	044522	
5442	042454	044524	043516	052040	
5443	042462	020117	042120	000122	
5444	042470	051127	052111	047111	EM24: .ASCIZ /WRITING SRO SET W-BIT IN KIPDR7/
5445	042476	020107	051123	020060	
5446	042504	042523	020124	026527	
5447	042512	044502	020124	047111	
5448	042520	045440	050111	051104	
5449	042526	000067			
5450	042530	026527	044502	020124	EM25: .ASCIZ /W-BIT GOT SET DURING ODD ADDR. ABORT/
5451	042536	047507	020124	042523	
5452	042544	020124	052504	044522	
5453	042552	043516	047440	042104	
5454	042560	040440	042104	027122	
5455	042566	040440	047502	052122	
5456	042574	000			
5457	042575	115	046505	051117	EM26: .ASCIZ /MEMORY MGMT. ACCESS ABORT DID NOT OCCUR/
5458	042602	020131	043515	052115	
5459	042610	020056	041501	042503	
5460	042616	051523	040440	047502	
5461	042624	052122	042040	042111	
5462	042632	047040	052117	047440	
5463	042640	041503	051125	000	
5464	042645	101	041503	051505	EM27: .ASCIZ /ACCESS ERROR DID NOT ABORT INSTRUCTION/
5465	042652	020123	051105	047522	
5466	042660	020122	044504	020104	
5467	042666	047516	020124	041101	
5468	042674	051117	020124	047111	
5469	042702	052123	052522	052103	
5470	042710	047511	000116		
5471	042714	051123	020060	044504	EM30: .ASCIZ /SRO DID NOT REPORT ACCESS ERROR CORRECTLY/
5472	042722	020104	047516	020124	
5473	042730	042522	047520	052122	
5474	042736	040440	041503	051505	
5475	042744	020123	051105	047522	
5476	042752	020122	047503	051122	
5477	042760	041505	046124	000131	
5478	042766	044504	020104	047516	EM31: .ASCIZ /DID NOT LOCKUP CORRECT VIRTUAL ADDR./
5479	042774	020124	047514	045503	
5480	043002	050125	041440	051117	
5481	043010	042522	052103	053040	
5482	043016	051111	052524	046101	
5483	043024	040440	042104	027122	
5484	043032	000			
5485	043033	120	043501	020105	EM32: .ASCIZ /PAGE LGTH. ABORT OCCURRED WHEN IT SHOULDN'T HAVE/
5486	043040	043514	044124	020056	
5487	043046	041101	051117	020124	
5488	043054	041517	052503	051122	
5489	043062	042105	053440	042510	
5490	043070	020116	052111	051440	

5491	043076	047510	046125	047104	
5492	043104	052047	044040	053101	
5493	043112	000105			
5494	043114	040520	042507	046040	EM33: .ASCIZ /PAGE LGTH. ABORT DID NOT OCCUR WHEN IT SHOULD HAVE/
5495	043122	052107	027110	040440	
5496	043130	047502	052122	042040	
5497	043136	042111	047040	052117	
5498	043144	047440	041503	051125	
5499	043152	053440	042510	020116	
5500	043160	052111	051440	047510	
5501	043166	046125	020104	040510	
5502	043174	042526	000		
5503	043177	123	030122	042040	EM34: .ASCIZ /SRO DID NOT REPORT PAGE LGTH. ABORT CORRECTLY/
5504	043204	042111	047040	052117	
5505	043212	051040	050105	051117	
5506	043220	020124	040520	042507	
5507	043226	046040	052107	027110	
5508	043234	040440	047502	052122	
5509	043242	041440	051117	042522	
5510	043250	052103	054514	000	
5511	043255	123	030122	047440	EM37: .ASCIZ /SRO OR SR2 CHANGED BY A SECOND ABORT/
5512	043262	020122	051123	020062	
5513	043270	044103	047101	042507	
5514	043276	020104	054502	040440	
5515	043304	051440	041505	047117	
5516	043312	020104	041101	051117	
5517	043320	000124			
5518	043322	051123	020060	051117	EM40: .ASCIZ /SRO OR SR2 WERE NOT "RESET" BY A RESET/
5519	043330	051440	031122	053440	
5520	043336	051105	020105	047516	
5521	043344	020124	051042	051505	
5522	043352	052105	020042	054502	
5523	043360	040440	051040	051505	
5524	043366	052105	000		
5525	043371	123	031122	047040	EM41: .ASCIZ /SR2 NOT TRACKING CORRECTLY/
5526	043376	052117	052040	040522	
5527	043404	045503	047111	020107	
5528	043412	047503	051122	041505	
5529	043420	046124	000131		
5530	043424	044504	020104	047516	EM42: .ASCIZ /DID NOT TRAP THRU KERNEL SPACE/
5531	043432	020124	051124	050101	
5532	043440	052040	051110	020125	
5533	043446	042513	047122	046105	
5534	043454	051440	040520	042503	
5535	043462	000			
5536	043463	113	020124	051105	EM43: .ASCIZ /KT ERROR SERVICED ON ODD ADDR. ERROR/
5537	043470	047522	020122	042523	
5538	043476	053122	041511	042105	
5539	043504	047440	020116	042117	
5540	043512	020104	042101	051104	
5541	043520	020056	051105	047522	
5542	043526	000122			
5543	043530	051123	020060	051117	EM44: .ASCIZ /SRO OR SR2 CHANGED BY ODD ADDR. ERROR/
5544	043536	051440	031122	041440	
5545	043544	040510	043516	042105	
5546	043552	041040	020131	042117	

5547	043560	020104	042101	051104	
5548	043566	020056	051105	047522	
5549	043574	000122			
5550	043576	051105	047522	020122	EM45: .ASCIZ /ERROR DURING "DOUBLE ERROR" (KT & ODD ADDR.)/
5551	043604	052504	044522	043516	
5552	043612	021040	047504	041125	
5553	043620	042514	042440	051122	
5554	043626	051117	020042	045450	
5555	043634	020124	020046	042117	
5556	043642	020104	042101	051104	
5557	043650	024456	000		
5558	043653	115	050106	020111	EM46: .ASCIZ /MFPI INSTRUCTION PUSHED WRONG DATA/
5559	043660	047111	052123	052522	
5560	043666	052103	047511	020116	
5561	043674	052520	044123	042105	
5562	043702	053440	047522	043516	
5563	043710	042040	052101	000101	
5564	043716	052115	044520	044440	EM47: .ASCIZ /MTPI INSTRUCTION LOADED WRONG DATA/
5565	043724	051516	051124	041525	
5566	043732	044524	047117	046040	
5567	043740	040517	042504	020104	
5568	043746	051127	047117	020107	
5569	043754	040504	040524	000	
5570	043761	123	040524	045503	EM50: .ASCIZ /STACK NOT PUSHED BY MFPI-MTPI/
5571	043766	047040	052117	050040	
5572	043774	051525	042510	020104	
5573	044002	054502	046440	050106	
5574	044010	026511	052115	044520	
5575	044016	000			
5576	044017	113	051105	042516	EM51: .ASCIZ /KERNEL PAGE ACCESS INSTEAD OF USER: MFPI-MTPI/
5577	044024	020114	040520	042507	
5578	044032	040440	041503	051505	
5579	044040	020123	047111	052123	
5580	044046	040505	020104	043117	
5581	044054	052440	042523	035122	
5582	044062	046440	050106	026511	
5583	044070	052115	044520	000	
5584	044075	127	047522	043516	EM52: .ASCIZ /WRONG PDR'S REFERENCED WHILE IN RELOCATE MODE/
5585	044102	050040	051104	051447	
5586	044110	051040	043105	051105	
5587	044116	047105	042503	020104	
5588	044124	044127	046111	020105	
5589	044132	047111	051040	046105	
5590	044140	041517	052101	020105	
5591	044146	047515	042504	000	
5592	044153	115	050106	020104	EM53: .ASCIZ /MFPD INSTRUCTION PUSHED WRONG DATA/
5593	044160	047111	052123	052522	
5594	044166	052103	047511	020116	
5595	044174	052520	044123	042105	
5596	044202	053440	047522	043516	
5597	044210	042040	052101	000101	
5598	044216	052123	041501	020113	EM54: .ASCIZ /STACK NOT PUSHED BY MFPD-MTPD/
5599	044224	047516	020124	052520	
5600	044232	044123	042105	041040	
5601	044240	020131	043115	042120	
5602	044246	046455	050124	000104	

5603	044254	040520	020122	051117
5604	044262	050040	051104	041440
5605	044270	040510	043516	042105
5606	044276	041040	020131	020101
5607	044304	042522	042523	000124
5608	044312	046111	042514	040507
5609	044320	020114	047515	042504
5610	044326	030040	020061	047516
5611	044334	020124	041101	051117
5612	044342	042524	000104	
5613	044346	051123	020060	044504
5614	044354	020104	047516	020124
5615	044362	042522	047520	052122
5616	044370	044440	046114	043505
5617	044376	046101	046440	042117
5618	044404	020105	030460	041440
5619	044412	051117	042522	052103
5620	044420	054514	000	
5621	044423	120	053523	041440
5622	044430	040510	043516	042105
5623	044436	041040	020131	047101
5624	044444	051040	044524	044440
5625	044452	020116	051525	051105
5626	044460	046440	042117	000105
5627	044466	040515	047111	027124
5628	044474	046440	042117	020105
5629	044502	051450	030122	036040
5630	044510	037070	020051	047516
5631	044516	020124	044504	040523
5632	044524	046102	042105	041040
5633	044532	020131	020101	042522
5634	044540	042523	000124	
5635	044544	040524	040524	044440
5636	044552	041516	051117	042522
5637	044560	052103	040440	052106
5638	044566	051105	040440	046440
5639	044574	044501	052116	020056
5640	044602	047515	042504	053440
5641	044610	044522	042524	000
5642	044615	123	052517	041522
5643	044622	020105	042522	047514
5644	044630	040503	042524	020104
5645	044636	047111	046440	044501
5646	044644	052116	020056	047515
5647	044652	042504	000	
5648				
5649	044655	117	042114	050040
5650	044662	020103	047440	042114
5651	044670	050040	053523	051040
5652	044676	020066	040527	020123
5653	044704	052040	051505	047124
5654	044712	020117	042440	051122
5655	044720	051117	041520	000
5656	044725	117	042114	050040
5657	044732	020103	047440	042114
5658	044740	050040	053523	051040

EM55: .ASCIZ /PAR OR PDR CHANGED BY A RESET/

EM56: .ASCIZ /ILLEGAL MODE 01 NOT ABORTED/

EM57: .ASCIZ /SRO DID NOT REPORT ILLEGAL MODE 01 CORRECTLY/

EM60: .ASCIZ /PSW CHANGED BY AN RTI IN USER MODE/

EM61: .ASCIZ /MAINT. MODE (SRO <8>) NOT DISABLED BY A RESET/

EM62: .ASCIZ /DATA INCORRECT AFTER A MAINT. MODE WRITE/

EM63: .ASCIZ /SOURCE RELOCATED IN MAINT. MODE/

DH1: .ASCIZ /OLD PC OLD PSW R6 WAS TESTNO ERRORPC/

DH2: .ASCIZ /OLD PC OLD PSW R6 WAS SRO SR2 TESTNO ERRORPC/

5659	044746	020066	040527	020123	
5660	044754	051440	030122	020040	
5661	044762	020040	051440	031122	
5662	044770	020040	020040	052040	
5663	044776	051505	047124	020117	
5664	045004	042440	051122	051117	
5665	045012	041520	000		
5666	045015	127	047522	042524	DH3: .ASCIZ /WROTE READ TESTNO ERRORPC/
5667	045022	020040	051040	040505	
5668	045030	020104	020040	052040	
5669	045036	051505	047124	020117	
5670	045044	042440	051122	051117	
5671	045052	041520	000		
5672	045055	101	042104	042522	DH7: .ASCIZ /ADDRESS TESTNO ERRORPC/
5673	045062	051523	052040	051505	
5674	045070	047124	020117	042440	
5675	045076	051122	051117	041520	
5676	045104	000			
5677	045105	122	043505	051511	DH10: .ASCII /REGISTER-ADDRS NUM OF/<CRLF>
5678	045112	042524	026522	042101	
5679	045120	051104	020123	047040	
5680	045126	046525	020040	043117	
5681	045134	200			
5682	045135	101	042116	042455	.ASCIZ /AND-ED OR-ED TIMOUTS TESTNO ERRORPC/
5683	045142	020104	047440	026522	
5684	045150	042105	020040	052040	
5685	045156	046511	052517	051524	
5686	045164	052040	051505	047124	
5687	045172	020117	042440	051122	
5688	045200	051117	041520	000	
5689	045205	122	043505	051511	DH11: .ASCII /REGISTR READ READ-(BINARY)/<CRLF>
5690	045212	051124	051040	040505	
5691	045220	020104	020040	051040	
5692	045226	040505	026504	041050	
5693	045234	047111	051101	024531	
5694	045242	200			
5695	045243	101	042104	042522	.ASCIZ /ADDRESS (OCTAL) 5432109876543210 TESTNO ERRORPC/
5696	045250	051523	024040	041517	
5697	045256	040524	024514	032440	
5698	045264	031464	030462	034460	
5699	045272	033470	032466	031464	
5700	045300	030462	020060	052040	
5701	045306	051505	047124	020117	
5702	045314	042440	051122	051117	
5703	045322	041520	000		
5704	045325	122	043505	051511	DH12: .ASCII /REGISTR WROTE READ READ-(BINARY)/<CRLF>
5705	045332	051124	053440	047522	
5706	045340	042524	020040	051040	
5707	045346	040505	020104	020040	
5708	045354	051040	040505	026504	
5709	045362	041050	047111	051101	
5710	045370	024531	200		
5711	045373	101	042104	042522	.ASCIZ /ADDRESS (OCTAL) (OCTAL) 5432109876543210 TESTNO ERRORPC/
5712	045400	051523	024040	041517	
5713	045406	040524	024514	024040	
5714	045414	041517	040524	024514	

5715	045422	032440	031464	030462	
5716	045430	034460	033470	032466	
5717	045436	031464	030462	020060	
5718	045444	052040	051505	047124	
5719	045452	020117	042440	051122	
5720	045460	051117	041520	000	
5721	045465	122	040505	020104	DH13: .ASCIZ /READ TESTNO ERRORPC/
5722	045472	020040	052040	051505	
5723	045500	047124	020117	042440	
5724	045506	051122	051117	041520	
5725	045514	000			
5726	045515	120	051101	050055	DH16: .ASCII /PAR-PDR PAR-PDR/<CRLF>
5727	045522	051104	050040	051101	
5728	045530	050055	051104	200	
5729	045535	103	042514	051101	.ASCIZ /CLEARED EFFECTD EXPECTD RECEIVD TESTNO ERRORPC/
5730	045542	042105	042440	043106	
5731	045550	041505	042124	042440	
5732	045556	050130	041505	042124	
5733	045564	051040	041505	044505	
5734	045572	042126	052040	051505	
5735	045600	047124	020117	042440	
5736	045606	051122	051117	041520	
5737	045614	000			
5738	045615	120	054510	044523	DH17: .ASCII /PHYSICL VIRTUAL/<CRLF>
5739	045622	046103	053040	051111	
5740	045630	052524	046101	200	
5741	045635	101	042104	042522	.ASCIZ /ADDRESS ADDRESS KIPAR4 TESTNO ERRORPC/
5742	045642	051523	040440	042104	
5743	045650	042522	051523	045440	
5744	045656	050111	051101	020064	
5745	045664	052040	051505	047124	
5746	045672	020117	042440	051122	
5747	045700	051117	041520	000	
5748	045705	120	054510	044523	DH20: .ASCII /PHYSICL PAR 4 PAR 5/<CRLF>
5749	045712	046103	050040	051101	
5750	045720	032040	020040	050040	
5751	045726	051101	032440	200	
5752	045733	101	042104	042522	.ASCIZ /ADDRESS VBA VBA PAR 4 PAR 5 PSW TESTNO ERRORPC/
5753	045740	051523	053040	040502	
5754	045746	020040	020040	053040	
5755	045754	040502	020040	020040	
5756	045762	050040	051101	032040	
5757	045770	020040	050040	051101	
5758	045776	032440	020040	050040	
5759	046004	053523	020040	020040	
5760	046012	052040	051505	047124	
5761	046020	020117	042440	051122	
5762	046026	051117	041520	000	
5763	046033	120	051104	020040	DH21: .ASCII /PDR VIRTUAL/<CRLF>
5764	046040	020040	053040	051111	
5765	046046	052524	046101	200	
5766	046053	124	051505	042524	.ASCIZ /TESTED ADDRESS TESTNO ERRORPC/
5767	046060	020104	040440	042104	
5768	046066	042522	051523	052040	
5769	046074	051505	047124	020117	
5770	046102	042440	051122	051117	

5771	046110	041520	000	
5772	046113	120	051104	044440
5773	046120	020116	050040	051104
5774	046126	020040	020040	053040
5775	046134	051111	052524	046101
5776	046142	051105	047522	020122
5777	046150	020040	042524	052123
5778	046156	042105	020040	042101
5779	046164	051104	051505	020123
5780	046172	042524	052123	047516
5781	046200	020040	051105	047522
5782	046206	050122	000103	
5783	046212	042120	020122	020040
5784	046220	020040	042524	052123
5785	046226	047516	020040	051105
5786	046234	047522	050122	000103
5787	046242	042120	020122	040527
5788	046250	020123	054105	042520
5789	046256	052103	020104	042524
5790	046264	052123	047516	020040
5791	046272	051105	047522	050122
5792	046300	000103		
5793	046302	042120	020122	020064
5794	046310	020040	051520	020127
5795	046316	020040	020040	042524
5796	046324	052123	047516	020040
5797	046332	051105	047522	050122
5798	046340	000103		
5799	046342	051123	020060	040527
5800	046350	020123	054105	042520
5801	046356	052103	020104	042120
5802	046364	020122	020064	020040
5803	046372	051520	020127	020040
5804	046400	020040	042524	052123
5805	046406	047516	020040	051105
5806	046414	047522	050122	000103
5807	046422	051123	020062	040527
5808	046430	020123	054105	042520
5809	046436	052103	020104	042120
5810	046444	020122	020064	020040
5811	046452	051520	020127	020040
5812	046460	020040	042524	052123
5813	046466	047516	020040	051105
5814	046474	047522	050122	000103
5815	046502	027126	027102	027101
5816	046510	020040	044513	042120
5817	046516	032122	020040	051123
5818	046524	020060	040527	020123
5819	046532	051123	020062	040527
5820	046540	020123	042524	052123
5821	046546	047516	020040	051105
5822	046554	047522	050122	000103
5823	046562	027126	027102	027101
5824	046570	020040	044513	042120
5825	046576	032122	020040	042524
5826	046604	052123	047516	020040

DH22: .ASCII /PDR IN PDR VIRTUAL/

.ASCIZ /ERROR TESTED ADDRESS TESTNO ERRORPC/

DH23: .ASCIZ /PDR TESTNO ERRORPC/

DH24: .ASCIZ /PDR WAS EXPECTD TESTNO ERRORPC/

DH26: .ASCIZ /PDR 4 PSW TESTNO ERRORPC/

DH30: .ASCIZ /SRO WAS EXPECTD PDR 4 PSW TESTNO ERRORPC/

DH31: .ASCIZ /SR2 WAS EXPECTD PDR 4 PSW TESTNO ERRORPC/

DH32: .ASCIZ /V.B.A. KIPDR4 SRO WAS SR2 WAS TESTNO ERRORPC/

DH33: .ASCIZ /V.B.A. KIPDR4 TESTNO ERRORPC/

E10

MD-11-DFKTH-A POP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 121
ERROR MESSAGES, DATA HEADERS-TABLES & FORMATS

5827	046612	051105	047522	050122					
5828	046620	000103							
5829	046622	027126	027102	027101	DH34:	.ASCIZ	/V.B.A.	KIPDR4	SRO WAS EXPECTD TESTNO ERRORPC/
5830	046630	020040	044513	042120					
5831	046636	032122	020040	051123					
5832	046644	020060	040527	020123					
5833	046652	054105	042520	052103					
5834	046660	020104	042524	052123					
5835	046666	047516	020040	051105					
5836	046674	047522	050122	000103					
5837	046702	027126	027102	027101	DH35:	.ASCIZ	/V.B.A.	KIPDR4	SR2 WAS EXPECTD TESTNO ERRORPC/
5838	046710	020040	044513	042120					
5839	046716	032122	020040	051123					
5840	046724	020062	040527	020123					
5841	046732	054105	042520	052103					
5842	046740	020104	042524	052123					
5843	046746	047516	020040	051105					
5844	046754	047522	050122	000103					
5845	046762	051123	020062	040527	DH36:	.ASCIZ	/SR2 WAS EXPECTD TESTNO	ERRORPC/	
5846	046770	020123	054105	042520					
5847	046776	052103	020104	042524					
5848	047004	052123	047516	020040					
5849	047012	051105	047522	050122					
5850	047020	000103							
5851	047022	044506	051522	020124	DH37:	.ASCII	/FIRST ABORT	SECOND ABORT/<CRLF>	
5852	047030	041101	051117	020124					
5853	047036	020040	020040	042523					
5854	047044	047503	042116	040440					
5855	047052	047502	052122	200					
5856	047057	123	030122	053440		.ASCIZ	/SRO WAS SR2 WAS SRO WAS SR2 WAS TESTNO	ERRORPC/	
5857	047064	051501	051440	031122					
5858	047072	053440	051501	051440					
5859	047100	030122	053440	051501					
5860	047106	051440	031122	053440					
5861	047114	051501	052040	051505					
5862	047122	047124	020117	042440					
5863	047130	051122	051117	041520					
5864	047136	000							
5865	047137	123	030122	053440	DH40:	.ASCIZ	/SRO WAS SR2 WAS TESTNO	ERRORPC/	
5866	047144	051501	051440	031122					
5867	047152	053440	051501	052040					
5868	047160	051505	047124	020117					
5869	047166	042440	051122	051117					
5870	047174	041520	000						
5871	047177	120	053523	053440	DH42:	.ASCIZ	/PSW WAS R6 WAS TESTNO	ERRORPC/	
5872	047204	051501	051040	020066					
5873	047212	040527	020123	052040					
5874	047220	051505	047124	020117					
5875	047226	042440	051122	051117					
5876	047234	041520	000						
5877	047237	105	050130	041505	DH44:	.ASCII	/EXPECTED	RECEIVED/<CRLF>	
5878	047244	042524	020104	020040					
5879	047252	020040	020040	020040					
5880	047260	042522	042503	053111					
5881	047266	042105	200						
5882	047271	123	030122	020040		.ASCIZ	/SRO SR2 SRO WAS SR2 WAS TESTNO	ERRORPC/	

F10

MO-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
 DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 122
 ERROR MESSAGES, DATA HEADERS-TABLES & FORMATS

5883	047276	020040	051440	031122
5884	047304	020040	020040	051440
5885	047312	030122	053440	051501
5886	047320	051440	031122	053440
5887	047326	051501	052040	051505
5888	047334	047124	020117	042440
5889	047342	051122	051117	041520
5890	047350	000		
5891	047351	105	050130	041505
5892	047356	042524	035104	200
5893	047363	120	053523	020040
5894	047370	020040	050040	020103
5895	047376	020040	020040	051440
5896	047404	030122	020040	020040
5897	047412	051440	031122	200
5898	047417	061	030067	030460
5899	047424	020067	024040	022063
5900	047432	032053	020051	030040
5901	047440	030062	032061	020067
5902	047446	024040	022063	100051
5903	047454	042522	042503	053111
5904	047462	042105	100072	
5905	047466	051520	020127	020040
5906	047474	020040	041520	020040
5907	047502	020040	020040	051123
5908	047510	020060	020040	020040
5909	047516	051123	020062	020040
5910	047524	020040	042524	052123
5911	047532	047516	020040	051105
5912	047540	047522	050122	000103
5913	047546	040504	040524	020040
5914	047554	020040	040504	040524
5915	047562	200		
5916	047563	105	050130	041505
5917	047570	042124	051040	041505
5918	047576	044505	042126	052040
5919	047604	051505	047124	020117
5920	047612	042440	051122	051117
5921	047620	041520	000	
5922	047623	124	051505	047124
5923	047630	020117	042440	051122
5924	047636	051117	041520	000
5925	047643	123	030122	053440
5926	047650	051501	051440	031122
5927	047656	053440	051501	052040
5928	047664	051505	047124	020117
5929	047672	042440	051122	051117
5930	047700	041520	000	
5931	047703	120	054510	044523
5932	047710	046103	050040	051101
5933	047716	032040	200	
5934	047721	101	042104	042522
5935	047726	051523	053040	041056
5936	047734	040456	020056	050040
5937	047742	051101	032040	020040
5938	047750	051440	030122	053440

DH45: .ASCII /EXPECTED:/(CRLF)

.ASCII /PSW PC SR0 SR2/(CRLF)

.ASCII /170017 (35+4) 020147 (35)/(CRLF)

.ASCII /RECEIVED:/(CRLF)

.ASCIZ /PSW PC SR0 SR2 TESTNO ERRORPC/

DH46: .ASCII /DATA DATA/(CRLF)

.ASCIZ /EXPECTD RECEIVD TESTNO ERRORPC/

DH50: .ASCIZ /TESTNO ERRORPC/

DH51: .ASCIZ /SR0 WAS SR2 WAS TESTNO ERRORPC/

DH52: .ASCII /PHYSICL PAR 4/(CRLF)

.ASCIZ /ADDRESS V.B.A. PAR 4 SR0 WAS SR2 WAS PSW TESTNO ERRORPC/

G10

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 123
ERROR MESSAGES, DATA HEADERS-TABLES & FORMATS

5939	047756	051501	051440	031122			
5940	047764	053440	051501	050040			
5941	047772	053523	020040	020040			
5942	050000	052040	051505	047124			
5943	050006	020117	042440	051122			
5944	050014	051117	041520	000			
5945	050021	123	030122	053440	DH57:	.ASCIZ	/SRD WAS EXPECTD TESTNO ERRORPC/
5946	050026	051501	042440	050130			
5947	050034	041505	042124	052040			
5948	050042	051505	047124	020117			
5949	050050	042440	051122	051117			
5950	050056	041520	000				
5951	050061	120	053523	053440	DH60:	.ASCIZ	/PSW WAS EXPECTD TESTNO ERRORPC/
5952	050066	051501	042440	050130			
5953	050074	041505	042124	052040			
5954	050102	051505	047124	020117			
5955	050110	042440	051122	051117			
5956	050116	041520	000				
5957							
5958		050122				.EVEN	
5959							
5960	050122	001356	001360	001354	DT1:	.WORD	TRAPPC, TRAPPS, WASR6, TESTNO, \$ERRPC, 0
5961	050130	001352	001116	000000			
5962	050136	001356	001360	001354	DT2:	.WORD	TRAPPC, TRAPPS, WASR6, WASSR0, WASSR2, TESTNO, \$ERRPC, 0
5963	050144	001362	001364	001352			
5964	050152	001116	000000				
5965	050156	001162	001164	001352	DT3:	.WORD	\$REG0, \$REG1, TESTNO, \$ERRPC, 0
5966	050164	001116	000000				
5967	050170	001162	001352	001116	DT7:	.WORD	\$REG0, TESTNO, \$ERRPC, 0
5968	050176	000000					
5969	050200	001370	001372	001374	DT10:	.WORD	ANDADR, ORADR, TONUM, TESTNO, \$ERRPC, 0
5970	050206	001352	001116	000000			
5971	050214	001162	001164	001164	DT11:	.WORD	\$REG0, \$REG1, \$REG1, TESTNO, \$ERRPC, 0
5972	050222	001352	001116	000000			
5973	050230	001162	001164	001166	DT12:	.WORD	\$REG0, \$REG1, \$REG2, \$REG2, TESTNO, \$ERRPC, 0
5974	050236	001166	001352	001116			
5975	050244	000000					
5976	050246	001162	001352	001116	DT13:	.WORD	\$REG0, TESTNO, \$ERRPC, 0
5977	050254	000000					
5978	050256	001162	001164	001174	DT16:	.WORD	\$REG0, \$REG1, \$REG5, \$REG2, TESTNO, \$ERRPC, 0
5979	050264	001166	001352	001116			
5980	050272	000000					
5981	050274	001402	001376	001172	DT17:	.WORD	PBALO, VIRT1, \$REG4, TESTNO, \$ERRPC, 0
5982	050302	001352	001116	000000			
5983	050310	001402	001376	001400	DT20:	.WORD	PBALO, VIRT1, VIRT2, \$REG4, \$REG5, \$TMPO, TESTNO, \$ERRPC, 0
5984	050316	001172	001174	001176			
5985	050324	001352	001116	000000			
5986	050332	001174	001170	001352	DT21:	.WORD	\$REG5, \$REG3, TESTNO, \$ERRPC, 0
5987	050340	001116	000000				
5988	050344	001162	001174	001170	DT22:	.WORD	\$REG0, \$REG5, \$REG3, TESTNO, \$ERRPC, 0
5989	050352	001352	001116	000000			
5990	050360	001174	001352	001116	DT23:	.WORD	\$REG5, TESTNO, \$ERRPC, 0
5991	050366	000000					
5992	050370	001166	001164	001352	DT24:	.WORD	\$REG2, \$REG1, TESTNO, \$ERRPC, 0
5993	050376	001116	000000				
5994	050402	001166	001176	001352	DT26:	.WORD	\$REG2, \$TMPO, TESTNO, \$ERRPC, 0

H10

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 124
ERROR MESSAGES, DATA HEADERS-TABLES & FORMATS

5995	050410	001116	000000						
5996	050414	001362	001170	001166	DT30:	.WORD	WASSRO, \$REG3, \$REG2, \$TMPO, TESTNO, \$ERRPC, 0		
5997	050422	001176	001352	001116					
5998	050430	000000							
5999	050432	001364	001172	001166	DT31:	.WORD	WASSR2, \$REG4, \$REG2, \$TMPO, TESTNO, \$ERRPC, 0		
6000	050440	001176	001352	001116					
6001	050446	000000							
6002	050450	001162	001172	001362	DT32:	.WORD	\$REG0, \$REG4, WASSRO, WASSR2, TESTNO, \$ERRPC, 0		
6003	050456	001364	001352	001116					
6004	050464	000000							
6005	050466	001162	001172	001352	DT33:	.WORD	\$REG0, \$REG4, TESTNO, \$ERRPC, 0		
6006	050474	001116	000000						
6007	050500	001162	001172	001362	DT34:	.WORD	\$REG0, \$REG4, WASSRO, \$REG2, TESTNO, \$ERRPC, 0		
6008	050506	001166	001352	001116					
6009	050514	000000							
6010	050516	001162	001172	001364	DT35:	.WORD	\$REG0, \$REG4, WASSR2, \$REG3, TESTNO, \$ERRPC, 0		
6011	050524	001170	001352	001116					
6012	050532	000000							
6013	050534	001364	001164	001352	DT36:	.WORD	WASSR2, \$REG1, TESTNO, \$ERRPC, 0		
6014	050542	001116	000000						
6015	050546	001176	001202	001362	DT37:	.WORD	\$TMPO, \$TMP2, WASSRO, WASSR2, TESTNO, \$ERRPC, 0		
6016	050554	001364	001352	001116					
6017	050562	000000							
6018	050564	001362	001364	001352	DT40:	.WORD	WASSRO, WASSR2, TESTNO, \$ERRPC, 0		
6019	050572	001116	000000						
6020	050576	001164	001166	001352	DT42:	.WORD	\$REG1, \$REG2, TESTNO, \$ERRPC, 0		
6021	050604	001116	000000						
6022	050610	001162	001164	001362	DT44:	.WORD	\$REG0, \$REG1, WASSRO, WASSR2, TESTNO, \$ERRPC, 0		
6023	050616	001364	001352	001116					
6024	050624	000000							
6025	050626	001164	001170	001362	DT45:	.WORD	\$REG1, \$REG3, WASSRO, WASSR2, TESTNO, \$ERRPC, 0		
6026	050634	001364	001352	001116					
6027	050642	000000							
6028	050644	001162	001164	001352	DT46:	.WORD	\$REG0, \$REG1, TESTNO, \$ERRPC, 0		
6029	050652	001116	000000						
6030	050656	001352	001116	000000	DT50:	.WORD	TESTNO, \$ERRPC, 0		
6031	050664	001362	001364	001352	DT51:	.WORD	WASSRO, WASSR2, TESTNO, \$ERRPC, 0		
6032	050672	001116	000000						
6033	050676	001402	001376	001172	DT52:	.WORD	PBALO, VIRT1, \$REG4, WASSRO, WASSR2, \$TMPO, TESTNO, \$ERRPC, 0		
6034	050704	001362	001364	001176					
6035	050712	001352	001116	000000					
6036	050720	001362	001164	001352	DT57:	.WORD	WASSRO, \$REG1, TESTNO, \$ERRPC, 0		
6037	050726	001116	000000						
6038	050732	001164	001166	001352	DT60:	.WORD	\$REG1, \$REG2, TESTNO, \$ERRPC, 0		
6039	050740	001116	000000						
6040	050744	001364	001172	001352	DT64:	.WORD	WASSR2, \$REG4, TESTNO, \$ERRPC, 0		
6041	050752	001116	000000						
6042									
6043	050756	000	000	000	DF1:	.BYTE	0,0,0,0,0		
6044	050761	000	000						
6045	050763	000	000	000	DF2:	.BYTE	0,0,0,0,0,0,0		
6046	050766	000	000	000					
6047	050771	000							
6048	050772	J30	000	000	DF3:	.BYTE	0,0,0,0		
6049	050775	000							
6050	05077E	000	000	000	DF7:	.BYTE	0,0,0		

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 125
ERROR MESSAGES, DATA HEADERS-TABLES & FORMATS

6051	051001	000	000	001	DF10:	.BYTE	0,0,1,0,0
6052	051004	000	000				
6053	051006	000	000	002	DF11:	.BYTE	0,0,2,0,0
6054	051011	000	000				
6055	051013	000	000	000	DF12:	.BYTE	0,0,0,2,0,0
6056	051016	002	000	000			
6057	051021	000	000	000	DF13:	.BYTE	0,0,0
6058	051024	000	000	000	DF16:	.BYTE	0,0,0,0,0,0
6059	051027	000	000	000			
6060	051032	003	000	000	DF17:	.BYTE	3,0,0,0,0
6061	051035	000	000				
6062	051037	003	000	000	DF20:	.BYTE	3,0,0,0,0,0,0,0
6063	051042	000	000	000			
6064	051045	000	000				
6065	051047	000	000	000	DF21:	.BYTE	0,0,0,0
6066	051052	000	000				
6067	051053	000	000	000	DF22:	.BYTE	0,0,0,0,0
6068	051056	000	000				
6069	051060	000	000	000	DF23:	.BYTE	0,0,0
6070	051063	000	000	000	DF24:	.BYTE	0,0,0,0
6071	051066	000	000				
6072	051067	000	000	000	DF30:	.BYTE	0,0,0,0,0,0
6073	051072	000	000	000			
6074	051075	000	000	000	DF46:	.BYTE	0,0,0,0
6075	051100	000	000				
6076	051101	000	000		DF50:	.BYTE	0,0
6077	051103	000	000	000	DF51:	.BYTE	0,0,0,0
6078	051106	000	000				
6079	051107	003	000	000	DF52:	.BYTE	3,0,0,0,0,0,0,0
6080	051112	000	000	000			
6081	051115	000	000				
6082	051117	000	000	000	DF57:	.BYTE	0,0,0,0
6083	051122	000	000				
6084	051123	000	000	000	DF60:	.BYTE	0,0,0,0
6085	051126	000	000				
6086							
6087		000001				.END	

DH10	045105	959	5677#				
DH11	045205	966	1198	5689#			
DH12	045325	973	992	5704#			
DH13	045465	980	986	5721#			
DH16	045515	999	5726#				
DH17	045615	1006	5738#				
DH2	044725	923	1235	5656#			
DH20	045705	1013	5748#				
DH21	046033	1020	5763#				
DH22	046113	1027	5772#				
DH23	046212	1034	1130	5783#			
DH24	046242	1040	1046	5787#			
DH26	046302	1052	1058	5793#			
DH3	045015	929	935	941	947	1229	5666#
DH30	046342	1064	5799#				
DH31	046422	1070	5807#				
DH32	046502	1076	5815#				
DH33	046562	1082	5823#				
DH34	046622	1088	5829#				
DH35	046702	1093	5837#				
DH36	046762	1099	1118	1241	5845#		
DH37	047022	1105	5851#				
DH40	047137	1112	5865#				
DH42	047177	1124	5871#				
DH44	047237	1136	5877#				
DH45	047351	1143	5891#				
DH46	047546	1153	1160	1185	5913#		
DH50	047623	1167	1192	1205	1223	5922#	
DH51	047643	1173	5925#				
DH52	047703	1179	5931#				
DH57	050021	1211	5945#				
DH60	050061	1217	5951#				
DH7	045055	953	5672#				
DISPLA	001142	786#	1350#	1358#	4290#	4319#	
DISPRE	000174	722#	1358				
DSMR =	177570	553#	785	1349			
DT1	050122	918	5960#				
DT10	050200	961	5969#				
DT11	050214	968	1200	5971#			
DT12	050230	975	994	5973#			
DT13	050246	981	987	5976#			
DT16	050256	1001	5978#				
DT17	050274	1008	5981#				
DT2	050136	924	1236	5962#			
DT20	050310	1015	5983#				
DT21	050332	1022	598E)				
DT22	050344	1029	5988#				
DT23	050360	1035	1131	5990#			
DT24	050370	1041	1047	5992#			
DT26	050402	1053	1059	5994#			
DT3	050156	930	936	942	948	1230	5965#
DT30	050414	1065	5996#				
DT31	050432	1071	5999#				
DT32	050450	1077	6002#				
DT33	050466	1083	6005#				
DT34	050500	1089	6007#				

M10

NO-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 130
CROSS REFERENCE TABLE -- USER SYMBOLS

DT35	050516	1094	6010#			
DT36	050534	1100	1119	6013#		
DT37	050546	1107	6015#			
DT40	050564	1113	6018#			
DT42	050576	1125	6020#			
DT44	050610	1138	6022#			
DT45	050626	1148	6025#			
DT46	050644	1155	1162	1187	6028#	
DT50	050656	1168	1193	1206	1224	6030#
DT51	050664	1174	6031#			
DT52	050676	1181	6033#			
DT57	050720	1212	6036#			
DT60	050732	1218	6038#			
DT64	050744	1242	6040#			
DT7	050170	954	5967#			
EMTVEC=	000030	642#	1319*	1320*		
EM1	041246	916	5324#			
EM10	041625	958	5369#			
EM11	041671	965	5376#			
EM12	041731	972	5382#			
EM13	042000	979	5389#			
EM14	042035	985	5394#			
EM15	042070	991	5399#			
EM16	042144	998	5407#			
EM17	042206	1005	5413#			
EM2	041306	922	5330#			
EM20	042256	1012	5420#			
EM21	042330	1019	5427#			
EM22	042365	1026	5432#			
EM23	042424	1033	5438#			
EM24	042470	1039	5444#			
EM25	042530	1045	5450#			
EM26	042575	1051	5457#			
EM27	042645	1057	5464#			
EM3	041355	928	5337#			
EM30	042714	1063	5471#			
EM31	042766	1069	1092	1098	1240	5478#
EM32	043033	1075	5485#			
EM33	043114	1081	5494#			
EM34	043177	1087	5503#			
EM37	043255	1104	5511#			
EM4	041414	934	5343#			
EM40	043322	1111	5518#			
EM41	043371	1117	5525#			
EM42	043424	1123	5530#			
EM43	043463	1129	5536#			
EM44	043530	1135	5543#			
EM45	043576	1142	5550#			
EM46	043653	1152	5558#			
EM47	043716	1159	5564#			
EM5	041447	940	5348#			
EM50	043761	1166	5570#			
EM51	044017	1172	5576#			
EM52	044075	1178	5584#			
EM53	044153	1184	5592#			
EM54	044216	1191	5598#			

SAUTOB	001134	782#	1382*	4640	4804																
SBASE	001302	860#																			
SBOADR	001122	777#																			
SBOOAT	001126	779#																			
SBELL	001216	811#	4322	4355																	
SBIN	040042	4985*	4988*	4989	4996#																
SCDW1	001306	862#																			
SCDW2	001310	863#																			
SCHARC	037516	4882*	4892*	4899	4908*	4913#															
SCKSWR	036116	4632#	5265																		
SCLR.T	034400	4201	4204#																		
SCMTAG	001100	765#	1310	1311	1319	1325	1326	1327													
SCM1 =	000006	797#	798#	799#	800#	801#	802#	803#													
SCM2 =	000014	797#	798#	799#	800#	801#	802#	803#													
SCM3 =	000006	795#	797																		
SCM4 =	000006	803#	804#	805#	806#	807#	808#	809#													
SCNTLC	037046	4658	4748	4797#																	
SCNTLG	037060	4643	4799#																		
SCNTLU	037053	4668	4772	4798#																	
SCPUOF	001254	834#																			
SCRFL	001223	813#	4196	4330	4355	4376	4395	4400	4443	4679	4777	4797	4881	4916							
SDBLK	040506	5098	5132	5140#																	
SDB20	040612	4429	5197#																		
SDDW0	001312	864#																			
SDDW1	001314	865#																			
SDDW10	001336	874#																			
SDDW11	001340	875#																			
SDDW12	001342	876#																			
SDDW13	001344	877#																			
SDDW14	001346	878#																			
SDDW15	001350	879#																			
SDDW2	001316	866#																			
SDDW3	001320	867#																			
SDDW4	001322	868#																			
SDDW5	001324	869#																			
SDDW6	001326	870#																			
SDDW7	001330	871#																			
SDDW8	001332	872#																			
SDDW9	001334	873#																			
SDEVCT	001236	825#																			
SDEVH	001304	861#																			
SDDAGN	034420	4178	4199	4206	4212#																
SOTBL	040476	5101	5136#																		
SENDAD	034410	732	1370	4208#	4350																
SENDCT	034236	1325	4180#																		
SENULL	034464	4228#																			
SENV	001246	830#	1376	4332	4860	4928	4952														
SENVH	001247	831#	1362	4862	4867	4930															
SEOP	034202	4170#	4825																		
SEOPCT	034230	1325*	4177#	4181																	
SERFLG	001103	768#	4235	4264	4266	4272*	4294	4317*	4355												
SERMAX	001115	774#	1328*	4266	4289*	4294															
SERROR	034750	1319	4308#																		
SERRPC	001116	775#	4324*	4325*	4326	4355	4382	5960	5962	5965	5967	5969	5971	5973							
		5976	5978	5981	5983	5986	5988	5990	5992	5994	5996	5999	6002	6005							
		6007	6010	6013	6015	6018	6020	6022	6025	6028	6030	6031	6033	6036							

K11

MD-11-DFKTH-A PDP 11/34 MEM MGMT DIAG
DFKTHA.P11 19-JAN-77 16:02

MACY11 27(1006) 27-JAN-77 08:51 PAGE 142
CROSS REFERENCE TABLE -- MACRO NAMES

.SWRLO	537#	
.SACT1	515#	726
.SAPT8	816#	
.SAPTH	515#	736
.SAPTY	515#	4916
.SCATC	515#	715
.SCMTA	515#	758
.SOB20	515#	5186
.SEOP	515#	4160
.SERRO	515#	4294
.SERRT	515#	
.SPOWE	515#	5270
.SREAO	515#	4622
.SSAVE	515#	5141
.SSCOP	515#	4230
.STRAP	515#	5225
.STYP8	515#	4973
.STYPO	515#	5074
.STYPE	515#	4837
.STYPO	515#	4997

. ABS. 051127 000

ERRORS DETECTED: 0
DEFAULT GLOBALS GENERATED: 0

DSKZ:DFKTHA,DSKZ:DFKTHA.SEQ/CRF/SOL=DFKTHA.P11
RUN-TIME: 27 18 1 SECONDS
RUN-TIME RATIO: 310/48=6.4
CORE USED: 33K (66 PAGES)