

KD11-K

11/6X MEMORY MGT DIAG
MD-11-DQKTA-A

EP-DQKTA-A-DL-A

COPYRIGHT © 1977

FICHE 1 OF 1

APR 1977

digital

MADE IN USA

B01

EOF1000000001
DOKTAA.P11

MA00000000-DOKTAPRO323P 11/6X MEM. PDP11/6X MEM. MACY11/6X MEM. DOKTAA.S80-FEB-77 10:3700000000

770323

.REM 2

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DOKTA-A-D
PRODUCT NAME: PDP-11/6X MEMORY MANAGEMENT DIAGNOSTIC
DATE CREATED: MARCH, 1977
MAINTAINER: DIAGNOSTIC ENGINEERING
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 MANUAL.

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

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

TABLE OF CONTENTS

1)	ABSTRACT AND DESCRIPTION
1.1	ABSTRACT
1.2	DESCRITPION
2)	REQUIREMENTS
2.1	EQUIPMENT
2.2	STORAGE
2.3	PRELIMINARY PROGRAMS
3)	LOADING PROCEDURE
3.1	METHOD
4)	STARTING PROCEDURE
4.1	STARTING ADDRESSES
4.2	PROGRAM AND OPERATOR ACTION
4.3	SPECIAL STARTING PROCEDURE
5)	OPERATING PROCEDURE
5.1	OPERATIONAL SWITCH SETTINGS
5.2	SUB-ROUTINE ABSTRACTS
5.3	PROGRAM AND OPERATOR ACTION
6)	ERRORS
6.1	ERROR HALTS AND DESCRIPTION
6.2	ERROR RECOVERY
6.3	SAMPLE ERROR MESSAGES
7)	RESTRICTIONS
7.1	STARTING RESTRICTIONS
7.2	OPERATING RESTRICTIONS
8)	MISCELLANEOUS
8.1	MEMORY MANAGEMENT STATUS REGISTERS
8.2	EXECTUION TIME
8.3	ACT/APT/XXDP COMPATABILITY
8.4	SOME IDEAS ON HOW TO GET MORE FROM THE PROGRAM

44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94

1. ABSTRACT AND DESCRIPTION

1.1 ABSTRACT

THIS PROGRAM WILL TEST ALL OF THE MEMORY MANAGEMENT LOGIC, INCLUDING THE STACK LIMIT REGISTER LOGIC, AND ENABLE THE FIELD SERVICE REPRESENTATIVE TO ISOLATE THE DETECTED FAILURES TO A REPLACABLE MODULE. IT IS ASSUMED THAT THE CPU HAS BEEN TESTED, OR IS KNOWN TO BE FUNCTIONING CORRECTLY, AND THAT THE PROGRAM IS STARTED FROM ADDRESS 200. THE 11/6X SERIES CACHE IS TURNED OFF FOR THE FIRST PASS OF THE PROGRAM (SEE SECTION 4.3) AND IS TURNED BACK ON FOR THE SECOND AND SUBSEQUENT PASSES. THIS WILL PROVIDE THE EARLIEST DETECTION OF MEMORY MANAGEMENT RELATED ERRORS AND ENABLE LOOPING ON THE ERROR INVOLVING MINIMUM LOGIC. THIS PROGRAM MAY ALSO EXPOSE FAULTS THAT ARE ON THE INTERFACE BETWEEN MEMORY MANAGEMENT AND OTHER SECTIONS OF THE COMPUTER.

1.2 PROGRAM DESCRIPTION

THIS PROGRAM HAS BEEN SEGMENTED IN THE FOLLOWING WAY: ALL DATA TABLES, ERROR MESSAGES, AND TRAP HANDLERS RESIDE IN LOW CORE (VIRTUAL PAGES 0 IE, ADDRESSES 001100 THRU 017776). RIGHT NOW THE END OF THE TRAP HANDLERS IS AROUND 015500, SO THERE IS SOME ROOM FOR FUTURE EXPANSION. THE TEST CODE STARTS AT VIRTUAL PAGE 1 (ADDRESS 020000) AND EXPANDS TOWARD PAGE 3 (ADDRESS 060000). THE END OF THE PROGRAM IS NOW AROUND ADDRESS 044300, SO MODIFICATIONS CAN BE MADE WITHOUT RE-SEGMENTING THE PROGRAM.

THE REASON FOR THIS SEGMENTATION IS TWO-FOLD. FIRST IT ENABLES THE OPERATOR TO TELL FROM THE ADDRESS DISPLAYED EXACTLY WHERE THE PROGRAM HAS HALTED OR "HUNG-UP". THAT IS, DID IT HALT IN THE TRAP ROUTINE BECAUSE OF A CONDITION IMPOSSIBLE TO RECOVER FROM (ON PAGE 0), OR DID IT GET "HUNG-UP" IN THE TEST CODE ON PAGE 1 OR 2. THE OTHER REASON IS THAT CERTAIN MEMORY MANAGEMENT FUNCTIONS LOCK UP THE VIRTUAL PC OF THE INSTRUCTION AND THE PROGRAM, IN ORDER TO OPERATE PROPERLY, ONE MUST KNOW WHERE IT IS AT ALL TIMES. IT SEEMS MUCH SIMPLER FOR THE CODE TO START AT A PREDETERMINED BOUNDARY SO THAT IF THE MESSAGES CHANGE OR A NEW SUBROUTINE IS ADDED THE PAGE THAT THE CODE IS ON WILL REMAIN THE SAME.

EACH TEST WILL SET THE LOOP ON ERROR POINTER (\$LPERR) TO THE MINIMUM NECESSARY SETUP CODE, IF ANY, FOR THE FUNCTION UNDER TEST. IF ON THE 11/6X AN "EXTERNAL SYNC PULSE" ON THE BACK PLANE IS DESIRED, THE MICROBREAK REGISTER SHOULD BE LOADED FROM THE CONSOLE WITH A MICRO-ADDRESS USED IN THE INSTRUCTION THAT TESTS EACH NEW FUNCTION.

SECTION 8.4 OF THIS DOCUMENT CONTAINS SOME IDEAS ON HOW TO EFFECTIVELY UTILIZE IT TO MAKE FAULT ISOLATION EASIER.

IT SHOULD BE NOTED THAT THIS PROGRAM DOES NOT CHECK OUT THE

EO1

MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 4
DQKTA.P11 07-FEB-77 10:30

150
151
152

CONSOLE OR THE CONSOLE CABLES. THE PROGRAM ASSUMES THAT
THOSE COMPONENTS HAVE BEEN TESTED OR ARE KNOWN TO BE GOOD.

153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208

2. REQUIREMENTS

2.1 EQUIPMENT

THE BASIC PDP-11/40 (WITH MEMORY MANAGEMENT-KT11-D), OR A PDP-11/6X SERIES COMPUTER, INCLUDING AN OPERATING CPU AND MEMORY. A CONSOLE TERMINAL IS ALSO NEEDED FOR ERROR MESSAGES, AND END OF PASS REPORTS.

2.2 STORAGE

THIS PROGRAM REQUIRES 12K OF MEMORY TO LOAD AND AT LEAST 16K OF MEMORY TO RUN IN. IT WILL SCAN MEMORY FROM 12K TO 124K ON 2K BOUNDARIES, REPORTING ANY HOLES IT MAY FIND.

2.3 PRELIMINARY PROGRAMS

THE CPU DIAGNOSTICS SHOULD BE RUN BEFORE THIS PROGRAM. MAIN MEMORY SHOULD BE SCANNED FOR AT LEAST THE FIRST 16K TO SEE THAT A PROGRAM WILL EXECUTE CORRECTLY BEFORE ANY PROGRAM IS RUN.

3. LOADING PROCEDURE

3.1 METHOD

THIS PROGRAM CAN BE LOADED FROM ANY DEVICE THAT IS SUPPORTED BY XXDP, AND SHOULD BE LOADED USING THE XXDP PROCEDURE FOR THAT DEVICE OR IT CAN BE LOADED DIRECTLY USING THE ABSOLUTE LOADER AND THE BINARY PAPER TAPE.

IF LOADING A BINARY PAPER TAPE, BE SURE THE SWITCH REGISTER IS CLEARED AFTER THE ABSOLUTE LOADER IS LOADED.

4. STARTING PROCEDURE

4.1 STARTING ADDRESSES

NOTE: CHECK THAT SWITCH REGISTER WAS ZERO IF PROGRAM WAS LOADED FROM PAPER TAPE.

200 THIS ADDRESS WILL RUN THE COMPLETE PROGRAM
204 THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 2
TEST THE READ/WRITE BITS IN THE MEMORY MANAGEMENT
STATUS REGISTERS
210 THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 3
PAGE ADDRESS AND PAGE DESCRIPTOR REGISTER TESTS
214 THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 4
RELOCATION AND ADDER TESTS
220 THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 5
MEMORY MANAGEMENT ABORTS LOGIC TESTS
224 THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 6
W BIT LOGIC TEST AND DUAL MAPPING TESTS

GO1

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 6
DOKTAA.P11 07-FEB-77 10:30

209
210
211
212
213
214
215
216
217
218

230 THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 7
MOVE FROM AND MOVE TO PERVIOUS MODE INSTRUCTION TESTS

4.2 PROGRAM AND OPERATOR ACTION
AFTER THE PROGRAM IS LOADED, THE FIRST TIME IT IS RUN IT WILL
IDENTIFY ITSELF AND RUN A QUICK VERIFY PASS. AT THE END OF
EACH PASS THE PROGRAM WILL TYPE OUT THE PASS NUMBER AND THE
TOTAL NUMBER OF ERRORS FOUND ON THAT PASS, AS FOLLOWS:

"END PASS # 1 TOTAL ERRORS SINCE LAST REPORT 0"

219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274

4.3 SPECIAL STARTING PROCEDURE ON THE 11/6X SERIES
FOR THE FIRST TIME THROUGH THE TESTS THE CACHE WILL BE TURNED
OFF (BITS 2 & 3 - FORCE MISS 0 & 1 - ARE SET IN THE CACHE
CONTROL REGISTER.). CACHE IS TURNED ON AFTER THE FIRST PASS
AND LEFT ON FOR THE REMAINING PASSES (BITS 2 & 3 OF THE CONTROL
REGISTER ARE CLEARED AND BIT 7 - PARITY ERROR ABORT BIT - IS SET
ENABLING CACHE PARITY ERRORS TO BE REPORTED IN THE MEMORY ERROR
REGISTER.)

5. OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

SWITCH	DISPLAY	
SW15=1	100000	HALT ON ERROR
SW14=1	040000	LOOP ON THE TEST THAT YOU ARE IN
SW13=1	020000	INHIBIT ALL ERROR TYPE OUTS
SW12=1	010000	INHIBIT TRACE TRAP
SW11=1	004000	INHIBIT ITERATIONS AFTER FIRST PASS
SW10=1	002000	RING BELL ON ERROR
SW09=1	001000	LOOP ON ERROR
SW08=1	000400	LOOP ON TEST IN SWR(06:00)
SW07=1	000200	INHIBIT MULTIPLE ERROR TYPE OUTS

5.2 SUBROUTINE ABSTRACTS
ALL SUBROUTINE ABSTRACTS APPEAR IN THE CODE, BEFORE THEIR
EXPANSION, THE FOLLOWING IS A LIST OF THEIR TITLES.

5.2.1 MACRO LIBRARY SUBROUTINES (FOUND IN MOST PROGRAMS)

END OF PASS ROUTINE
SCOPE HANDLER ROUTINE
ERROR HANDLER ROUTINE
ERROR MESSAGE TYPE OUT ROUTINE
SAVE & RESTORE R0-R5 ROUTINES
TYPE ROUTINE
BINARY TO OCTAL (ASCII) AND TYPE ROUTINE
CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
TRAP DECODER
POWER DOWN AND UP ROUTINE
DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

5.2.2 SUBROUTINES UNIQUE TO THIS PROGRAM

TURN OFF AND SAVE THE T-BIT
RESTORE T-BIT TO ITS PREVIOUS CONDITION
CLEAR 8 PAR'S OR PDR'S STARTING FROM ADDRESS IN R5
CLEANUP LOCATIONS THAT HOLD LOGICAL 'AND' AND 'OR'
P.A.R. OR P.D.R. ADDRESS TIMED OUT WHEN REFERENCED
DUAL ADDRESSING WHEN LOADING A P.A.R. OR P.D.R.
COUNT PATTERN ERROR IN P.A.R.'S OR P.D.R.'S

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

5.2.3 TRAP AND ABORT HANDLER ROUTINES

CPU TRAP HANDLER
CACHE/MEMORY-PARITY TRAPS AND ABORTS HANDLER
MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER
TRAP ROUTINES FOR ABORT IN USER MODE

6. ERRORS

- 6.1 ERROR HALTS AND DESCRIPTION
WHEN THE PROGRAM DETECTS AN ERROR CONDITION IT ISSUES AN
"ERROR" (EMT) CALL. THIS CAUSES THE CPU TO TRAP TO THE
"ERROR HANDLER ROUTINE" WHICH PRINTS OUT THE ERROR MESSAGE,
IF ANY, AND CHECKS THE SWITCH REGISTER FOR THE MODE SELECTED.
THE PROGRAM WILL REACT AS FOLLOWS:
- | | |
|----------------------------|---------------------|
| HALT ON THE ERROR | IF SW15=1, (100000) |
| INHIBIT ERROR TYPE OUT | IF SW13=1, (020000) |
| RING BELL ON THE ERROR | IF SW10=1, (002000) |
| LOOP ON THE ERROR | IF SW09=1, (001000) |
| INHIBIT MULTIPLE TYPE OUTS | IF SW07=1, (000200) |
- 6.2 ERROR RECOVERY
IF SWITCH 09 IS UP, THE PROGRAM WILL LOOP BACK TO WHERE THE
LOOP ON ERROR POINTER (\$LPERR) IS SET. THIS WILL PROVIDE
THE TIGHTEST POSSIBLE SCOPING LOOP, AND PROVIDES ALL NECESSARY
SETUP CODE TO RECREATE THE ERROR.
- 6.3 SAMPLE ERROR TYPE OUTS
- UNEXPECTED CPU TRAP OR ABORT THRU "ERRVEC" (004)
CPU ERR TESTNO PC AT ABORT
000020 000033 XXXXXX
- THIS ERROR MESSAGE INDICATES THAT THE CPU TIMED OUT OVER THE
UNIBUS WHEN NO TIMEOUTS WERE EXPECTED. THE CONTENTS OF THE CPU
ERROR REGISTER, IF ON AN 11/6X, ARE SHOWN UNDER "CPU ERR",
SHOWING THAT IN FACT A UNIBUS TIMEOUT DID OCCUR (BIT 4 IS SET).
IF ON AN 11/40, JUST THE TEST NO. AND PC AT THE TIME OF THE
ABORT ARE TYPED OUT. THIS TEST HAPPENS TO BE CHECKING THE
CARRY PROPAGATION, THAT IS DETERMINED FROM LOOKING AT THE
INDEX AT THE FRONT OF THE LISTING.

321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376

7. RESTRICTIONS

7.1 STARTING RESTRICTIONS

IF A STARTING POINT OTHER THAN "200" IS USED AND ERRORS ARE REPORTED, THEY MAY BE DUE TO LOGIC THAT IS ASSUMED TO BE WORKING AS A RESULT OF TESTS THAT WERE NOT RUN.

7.2 OPERATING RESTRICTIONS

THE "MED" INSTRUCTION IS USED TO DETERMINE IF THE PROGRAM IS EXECUTING ON AN 11/40 OR AN 11/6X. IF A "MED" TRAPS AS A RESERVED INSTRUCTION, IT IS AUTOMATICALLY ASSUMED THE PROGRAM IS ON AN 11/40.

8. MISCELLANEOUS

8.1 MEMORY MANAGEMENT STATUS REGISTERS

THE SYMBOLS "MMR0,MMR1,MMR2" ARE USE THROUGHOUT THE PROGRAM. THESE REPRESENT THE THREE MEMORY MANAGEMENT STATUS REGISTERS IN THE 11/6X OR THE 11/40.

8.2 EXECUTION TIME

THE TIMES NOTED BELOW ARE "11/6X" TIMES. PROGRAM EXECUTION TIMES WILL BE LONGER IN EXECUTING ON AN 11/40.

THE RUN TIME FOR A SINGLE PASS WITH NO ITERATIONS IS APPROXIMATELY 10 SECONDS.

THE RUN TIME FOR A SINGLE PASS WITH ITERATIONS AND TRACE TRAPPING ENABLED IS APPROXIMATELY 3 MINUTES.

8.3 ACT/APT/XXDP COMPATABILITY

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

8.4 HINTS ON HOW TO GET MORE INFORMATION FROM THE PROGRAM

IF AN ERROR OCCURS THE FIRST THING THAT SHOULD BE NOTED IS WHAT PASS DID THE ERROR OCCUR ON. IF THE PASS HAS AN ODD NUMBER AND SWITCH 12 IS DOWN THEN THE ERROR MIGHT BE T-BIT SENSITIVE. TRY TO RUN AGAIN WITH SWITCH 12 UP, THIS WILL INHIBIT T-BIT TRAPPING.
 IF THE PASS NUMBER IS GREATER THAN ONE THE ERROR MIGHT BE ITERATION SENSITIVE. TRY TO RUN AGAIN WITH SWITCH 11 UP, THIS WILL INHIBIT ITERATIONS.
 NOW THAT YOU HAVE DETERMINED HOW TO MAKE THE MACHINE FAIL LOOK IN THE INDEX AT THE FRONT OF THE LISTING TO FIND THE TITLE OF THE TEST THAT WAS RUNNING WHEN THE ERROR CONDITION OCCURRED. GO TO THE LISTING AND READ THE PARAGRAPH AT THE BEGINNING OF THE TEST SO THAT YOU KNOW WHAT THE TEST IS TRYING TO DO. NOW READ THE ERROR MESSAGE AND IF THERE IS A COLUMN LABELED "ERRORPC" GO TO THAT LOCATION IN THE TEST. THIS IS THE PC

K01

MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 10
DQKTA.P11 07-FEB-77 10:30

377
378
379
380
381
382
383
384
385
386
387
388

OF THE "ERROR" STATEMENT. THE NUMBER IS THE ERROR MESSAGE
NUMBER AND IN THE FRONT OF THE LISTING IS THE "ERROR MESSAGE
POINTER TABLE" WHICH WILL TELL YOU WHAT WORDS WERE TYPED
OUT.
IF YOU WANT TO SCOPE THIS ERROR CONDITION, PUT UP SWITCH 09
(LOOP ON ERROR) OR IF YOU WANT TO LOOP ON THE ENTIRE TEST PUT
UP SWITCH 14. YOU WILL PROBABLY WANT TO INHIBIT THE ERROR TYPE
OUT AT THIS POINT, SWITCH 13 WILL DO THAT.

2

LO1

MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 11
DQKTA.P11 07-FEB-77 10:30

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

```
.TITLE MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG.
*COPYRIGHT (C) OCTOBER, 1976
*DIGITAL EQUIPMENT CORP.
*MAYNARD, MASS. 01754
*
*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
*
SBTTL OPERATIONAL SWITCH SETTINGS
*
*      SWITCH      USE
*      -----
*      15      HALT ON ERROR
*      14      LOOP ON TEST
*      13      INHIBIT ERROR TYPEOUTS
*      12      INHIBIT TRACE TRAP
*      11      INHIBIT ITERATIONS
*      10      BELL ON ERROR
*      9       LOOP ON ERROR
*      8       LOOP ON TEST IN SWR<6:0>
*      7       INHIBIT MULTIPLE ERROR TYPEOUTS
*
*      SWITCH      OCTAL VALUE
*      -----
*      15      100000
*      14      40000
*      13      20000
*      12      10000
*      11      4000
*      10      2000
*      9       1000
*      8       400
*      7       200
*      6       100
*      5       40
*      4       20
*      3       10
*      2       4
*      1       2
*      0       1
```

.SBTTL BASIC DEFINITIONS

;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***

STACK= 1100

.EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL

.EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL

;*MISCELLANEOUS DEFINITIONS

HT= 11 ;;CODE FOR HORIZONTAL TAB

LF= 12 ;;CODE FOR LINE FEED

CR= 15 ;;CODE FOR CARRIAGE RETURN

CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED

PS= 177776 ;;PROCESSOR STATUS WORD

.EQUIV PS,PSW

STKLMT= 177774 ;;STACK LIMIT REGISTER

PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER

DSWR= 177570 ;;HARDWARE SWITCH REGISTER

DDISP= 177570 ;;HARDWARE DISPLAY REGISTER

;*GENERAL PURPOSE REGISTER DEFINITIONS

R0= %0 ;;GENERAL REGISTER

R1= %1 ;;GENERAL REGISTER

R2= %2 ;;GENERAL REGISTER

R3= %3 ;;GENERAL REGISTER

R4= %4 ;;GENERAL REGISTER

R5= %5 ;;GENERAL REGISTER

R6= %6 ;;GENERAL REGISTER

R7= %7 ;;GENERAL REGISTER

SP= %6 ;;STACK POINTER

PC= %7 ;;PROGRAM COUNTER

;*PRIORITY LEVEL DEFINITIONS

PR0= 0 ;;PRIORITY LEVEL 0

PR1= 40 ;;PRIORITY LEVEL 1

PR2= 100 ;;PRIORITY LEVEL 2

PR3= 140 ;;PRIORITY LEVEL 3

PR4= 200 ;;PRIORITY LEVEL 4

PR5= 240 ;;PRIORITY LEVEL 5

PR6= 300 ;;PRIORITY LEVEL 6

PR7= 340 ;;PRIORITY LEVEL 7

;*SWITCH REGISTER" SWITCH DEFINITIONS

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

NO1

MAINDEC-11-DOKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 13
 DOKTAA.P11 07-FEB-77 10:30 BASIC DEFINITIONS

```

486      000002      SW01= 2
487      000001      SW00= 1
488      .EQUIV      SW09,SW9
489      .EQUIV      SW08,SW8
490      .EQUIV      SW07,SW7
491      .EQUIV      SW06,SW6
492      .EQUIV      SW05,SW5
493      .EQUIV      SW04,SW4
494      .EQUIV      SW03,SW3
495      .EQUIV      SW02,SW2
496      .EQUIV      SW01,SW1
497      .EQUIV      SW00,SW0
498
499      ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
500      100000      BIT15= 100000
501      040000      BIT14= 40000
502      020000      BIT13= 20000
503      010000      BIT12= 10000
504      004000      BIT11= 4000
505      002000      BIT10= 2000
506      001000      BIT09= 1000
507      000400      BIT08= 400
508      000200      BIT07= 200
509      000100      BIT06= 100
510      000040      BIT05= 40
511      000020      BIT04= 20
512      000010      BIT03= 10
513      000004      BIT02= 4
514      000002      BIT01= 2
515      000001      BIT00= 1
516      .EQUIV      BIT09,BIT9
517      .EQUIV      BIT08,BIT8
518      .EQUIV      BIT07,BIT7
519      .EQUIV      BIT06,BIT6
520      .EQUIV      BIT05,BIT5
521      .EQUIV      BIT04,BIT4
522      .EQUIV      BIT03,BIT3
523      .EQUIV      BIT02,BIT2
524      .EQUIV      BIT01,BIT1
525      .EQUIV      BIT00,BIT0
526
527      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
528      000004      ERRVEC= 4      ; TIME OUT AND OTHER ERRORS
529      000010      RESVEC= 10     ; RESERVED AND ILLEGAL INSTRUCTIONS
530      000014      TBITVEC= 14    ; "T" BIT
531      000014      TRTVEC= 14     ; TRACE TRAP
532      000014      BPTVEC= 14     ; BREAKPOINT TRAP (BPT)
533      000020      IOTVEC= 20     ; INPUT/OUTPUT TRAP (IOT) **SCOPE**
534      000024      PWRVEC= 24     ; POWER FAIL
535      000030      EMTVEC= 30     ; EMULATOR TRAP (EMT) **ERROR**
536      000034      TRAPVEC= 34    ; "TRAP" TRAP
537      000060      TKVEC= 60      ; TTY KEYBOARD VECTOR
538      000064      TPVEC= 64      ; TTY PRINTER VECTOR
539      000240      PIRQVEC= 240    ; PROGRAM INTERRUPT REQUEST VECTOR
540      .SBTTL      MEMORY MANAGEMENT DEFINITIONS
541

```

```

542          ;*KT11 VECTOR ADDRESS
543
544          000250      MMVEC= 250
545
546          ;*KT11 STATUS REGISTER ADDRESSES
547
548          177572      SR0= 177572
549          177574      SR1= 177574
550          177576      SR2= 177576
551          172516      SR3= 172516
552
553          ;*USER "I" PAGE DESCRIPTOR REGISTERS
554
555          177600      UIPDR0= 177600
556          177602      UIPDR1= 177602
557          177604      UIPDR2= 177604
558          177606      UIPDR3= 177606
559          177610      UIPDR4= 177610
560          177612      UIPDR5= 177612
561          177614      UIPDR6= 177614
562          177616      UIPDR7= 177616
563
564          ;*USER "I" PAGE ADDRESS REGISTERS
565
566          177640      UIPAR0= 177640
567          177642      UIPAR1= 177642
568          177644      UIPAR2= 177644
569          177646      UIPAR3= 177646
570          177650      UIPAR4= 177650
571          177652      UIPAR5= 177652
572          177654      UIPAR6= 177654
573          177656      UIPAR7= 177656
574
575          ;*KERNEL "I" PAGE DESCRIPTOR REGISTERS
576
577          172300      KIPDR0= 172300
578          172302      KIPDR1= 172302
579          172304      KIPDR2= 172304
580          172306      KIPDR3= 172306
581          172310      KIPDR4= 172310
582          172312      KIPDR5= 172312
583          172314      KIPDR6= 172314
584          172316      KIPDR7= 172316
585
586          ;*KERNEL "I" PAGE ADDRESS REGISTERS
587
588          172340      KIPAR0= 172340
589          172342      KIPAR1= 172342
590          172344      KIPAR2= 172344
591          172346      KIPAR3= 172346
592          172350      KIPAR4= 172350
593          172352      KIPAR5= 172352
594          172354      KIPAR6= 172354
595          172356      KIPAR7= 172356
596
597          .EQUIV SP,KSP

```

C02

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 15
 DOKTAA.P11 07-FEB-77 10:30 MEMORY MANAGEMENT DEFINITIONS

```

598 .EQUIV SP,USP
599 .EQUIV STACK,KERSTK
600 .EQUIV SRO,MMRO
601 .EQUIV SRI,MMRI
602 .EQUIV SR2,MMR2
603
604 .ADDITIONAL DEFINITIONS
605
606 USESTK=STACK-200
607 CRLF= 200 ; CODE FOR CARRIAGE RETURN LINE FEED
608 CACHVEC=114 ; CACHE ERROR INTERRUPT VECTOR
609 MED= 76600 ; OP CODE OF MAINTENANCE EXAM/DEP INST.
610 RWHAM= 022 ; READ CODE FOR WHAMI REG.
611 RLJAM= 100 ; READ CODE FOR LOG JAM REGISTER
612 RLSERV= 101 ; READ CODE FOR LOG SERVICE REGISTER
613 RLFGIN= 104 ; READ CODE FOR LOG FLAG/INTERRUPT REG.
614 RLWHAM= 105 ; READ CODE FOR LOG WHAMI REGISTER
615 WHAM= 222 ; WRITE CODE FOR WHAMI REG.
616 WCNSSW= 226 ; WRITE CODE FOR CONSOLE SW. REG.
617 WLWHAM= 305 ; WRITE CODE FOR LOG WHAMI REGISTER
618
619
620
621 .SBTTL TRAP CATCHER
622
623 .=0
624 ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
625 ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
626 ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
627 .=174
628 000174 000000 DISPREG: .WORD 0 ; SOFTWARE DISPLAY REGISTER
629 000176 000000 SWREG: .WORD 0 ; SOFTWARE SWITCH REGISTER
630 .SBTTL STARTING ADDRESS(ES)
631 000200 000137 020000 JMP STRT1 ; JUMP TO STARTING ADDRESS OF PROGRAM
632 ; RUN ENTIRE PROGRAM (ALL TESTS)
633 000204 000137 020010 JMP STRT2 ; STARTING AT ENTRY POINT 2
634 ; MEMORY MANAGEMENT STATUS REGISTERS
635 000210 000137 020020 JMP STRT3 ; STARTING AT ENTRY POINT 3
636 ; PAGE ADDRESS AND DESCRIPTOR REGISTERS
637 000214 000137 020030 JMP STRT4 ; STARTING AT ENTRY POINT 4
638 ; RELOCATION AND ADDER TESTS
639 000220 000137 020040 JMP STRT5 ; STARTING AT ENTRY POINT 5
640 ; MEMORY MANAGEMENT ABORT LOGIC
641 000224 000137 020050 JMP STRT6 ; STARTING AT ENTRY POINT 6
642 ; W BIT LOGIC & DUAL MAPPING
643 000230 000137 020060 JMP STRT7 ; STARTING AT ENTRY POINT 7
644 ; MFP & MTP LOGIC TESTS
645
646 .SBTTL APT PARAMETER BLOCK
647
648 ;*****
649 ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
650 ;*****
651 .SX=. ; SAVE CURRENT LOCATION
652 000024 .=24 ; SET POWER FAIL TO POINT TO START OF PROGRAM
653 200 ; FOR APT START UP
        .=44 ; POINT TO APT INDIRECT ADDRESS PNTR.

```

002

MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 16
DQKTA.P11 07-FEB-77 10:30 APT PARAMETER BLOCK

654	000044	000234	\$APTHDR ; ; POINT TO APT HEADER BLOCK
655		000234	.=.SX ; ; RESET LOCATION COUNTER
656			*****
657			;; SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
658			;; INTERFACE SPEC.
659			
660	000234		\$APTHD:
661	000234	000000	\$SHIBTS: .WORD 0 ; ; TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
662	000236	001226	\$MBAADR: .WORD \$MAIL ; ; ADDRESS OF APT MAILBOX (BITS 0-15)
663	000240	000005	\$TSTM: .WORD 5 ; ; RUN TIM OF LONGEST TEST
664	000242	000015	\$PASTM: .WORD 15 ; ; RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
665	000244	000005	\$UNITM: .WORD 5 ; ; ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
666	000246	000052	.WORD \$ETEND-\$MAIL/2 ; ; LENGTH MAILBOX-ETABLE(WORDS)
667			.SBTTL ACT11 HOOKS
668			
669			*****
670			;; HOOKS REQUIRED BY ACT11
671		000250	\$SVPC= ; ; SAVE PC
672		000046	.=46
673	000046	040762	\$ENDAD ; ; 1) SET LOC.46 TO ADDRESS OF \$ENDAD IN .SEOP
674		000052	.=52
675	000052	000000	.WORD 0 ; ; 2) SET LOC.52 TO ZERO
676		000250	.= \$SVPC ; ; RESTORE PC

E02

MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 17
 DQKTA.P11 07-FEB-77 10:30 COMMON TAGS

.SBTTL COMMON TAGS

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

677
 678
 679
 680
 681
 682
 683 001100
 684 001100
 685 001100 000000
 686 001102 000
 687 001103 000
 688 001104 000000
 689 001106 000000
 690 001110 000000
 691 001112 000000
 692 001114 000
 693 001115 001
 694 001116 000000
 695 001120 000000
 696 001122 000000
 697 001124 000000
 698 001126 000000
 699 001130 000000
 700 001132 000000
 701 001134 000
 702 001135 000
 703 001136 000000
 704 001140 177570
 705 001142 177570
 706 001144 177560
 707 001146 177562
 708 001150 177564
 709 001152 177566
 710 001154 000
 711 001155 002
 712 001156 012
 713 001157 000
 714 001160 000000
 715
 716 001162 000000
 717 001164 000000
 718 001166 000000
 719 001170 000000
 720 001172 000000
 721 001174 000000
 722 001176 000000
 723 001200 000000
 724 001202 000000
 725 001204 000000
 726 001206 000000
 727 001210 000000
 728 001212 000000
 729 001214 000000
 730 001216 177607 000377
 731 001222 077
 732 001223 015

.=1100
 SCMTAG: .WORD 0
 STSTNM: .BYTE 0
 SERFLG: .BYTE 0
 SICNT: .WORD 0
 SLPADR: .WORD 0
 SLPERR: .WORD 0
 SERTTL: .WORD 0
 SITEMB: .BYTE 0
 SERMAX: .BYTE 1
 SERRPC: .WORD 0
 SGDADR: .WORD 0
 SBDADR: .WORD 0
 SGDDAT: .WORD 0
 SBODAT: .WORD 0
 SAUTOB: .BYTE 0
 SINTAG: .BYTE 0
 SWR: .WORD 0
 DISPLAY: .WORD 0
 STKS: 177560
 STKB: 177562
 STPS: 177564
 STPB: 177566
 SNULL: .BYTE 0
 SFILLS: .BYTE 2
 SFILLC: .BYTE 12
 STPFLG: .BYTE 0
 SREGAD: .WORD 0
 SREG0: .WORD 0
 SREG1: .WORD 0
 SREG2: .WORD 0
 SREG3: .WORD 0
 SREG4: .WORD 0
 SREG5: .WORD 0
 STMP0: .WORD 0
 STMP1: .WORD 0
 STMP2: .WORD 0
 STMP3: .WORD 0
 STMP4: .WORD 0
 STMP5: .WORD 0
 STIMES: 0
 SESCAPE: 0
 SBELL: .ASCII <207><377><377>
 SQUES: .ASCII ?/
 SCRLF: .ASCII <15>

;;START OF COMMON TAGS
 ;;CONTAINS THE TEST NUMBER
 ;;CONTAINS ERROR FLAG
 ;;CONTAINS SUBTEST ITERATION COUNT
 ;;CONTAINS SCOPE LOOP ADDRESS
 ;;CONTAINS SCOPE RETURN FOR ERRORS
 ;;CONTAINS TOTAL ERRORS DETECTED
 ;;CONTAINS ITEM CONTROL BYTE
 ;;CONTAINS MAX. ERRORS PER TEST
 ;;CONTAINS PC OF LAST ERROR INSTRUCTION
 ;;CONTAINS ADDRESS OF 'GOOD' DATA
 ;;CONTAINS ADDRESS OF 'BAD' DATA
 ;;CONTAINS 'GOOD' DATA
 ;;CONTAINS 'BAD' DATA
 ;;RESERVED--NOT TO BE USED
 ;;AUTOMATIC MODE INDICATOR
 ;;INTERRUPT MODE INDICATOR
 ;;ADDRESS OF SWITCH REGISTER
 ;;ADDRESS OF DISPLAY REGISTER
 ;;TTY KBD STATUS
 ;;TTY KBD BUFFER
 ;;TTY PRINTER STATUS REG. ADDRESS
 ;;TTY PRINTER BUFFER REG. ADDRESS
 ;;CONTAINS NULL CHARACTER FOR FILLS
 ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
 ;;INSERT FILL CHARS. AFTER A "LINE FEED"
 ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
 ;;CONTAINS THE ADDRESS FROM
 ;;WHICH (SREG0) WAS OBTAINED
 ;;CONTAINS ((SREGAD)+0)
 ;;CONTAINS ((SREGAD)+2)
 ;;CONTAINS ((SREGAD)+4)
 ;;CONTAINS ((SREGAD)+6)
 ;;CONTAINS ((SREGAD)+10)
 ;;CONTAINS ((SREGAD)+12)
 ;;USER DEFINED
 ;;USER DEFINED
 ;;USER DEFINED
 ;;USER DEFINED
 ;;USER DEFINED
 ;;USER DEFINED
 ;;MAX. NUMBER OF ITERATIONS
 ;;ESCAPE ON ERROR ADDRESS
 ;;CODE FOR BELL
 ;;QUESTION MARK
 ;;CARRIAGE RETURN

F02

MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 18
 DQKTA.P11 07-FEB-77 10:30 COMMON TAGS

733	001224	000012	SLF: .ASCIZ (12)	;;LINE FEED
734			*****	
735			.SBTTL APT MAILBOX-ETABLE	
736			*****	
737			*****	
738			.EVEN	
739	001226		\$MAIL: .WORD	APT MAILBOX
740	001226	000000	\$MSGTY: .WORD	MESSAGE TYPE CODE
741	001230	000000	\$FATAL: .WORD	FATAL ERROR NUMBER
742	001232	000000	\$TESTN: .WORD	TEST NUMBER
743	001234	000000	\$PASS: .WORD	PASS COUNT
744	001236	000000	\$DEVCT: .WORD	DEVICE COUNT
745	001240	000000	\$UNIT: .WORD	I/O UNIT NUMBER
746	001242	000000	\$MSGAD: .WORD	MESSAGE ADDRESS
747	001244	000000	\$MSGLG: .WORD	MESSAGE LENGTH
748	001246		\$ETABLE: .WORD	APT ENVIRONMENT TABLE
749	001246	000	\$ENV: .BYTE	ENVIRONMENT BYTE
750	001247	000	\$ENVN: .BYTE	ENVIRONMENT MODE BITS
751	001250	000000	\$SWREG: .WORD	APT SWITCH REGISTER
752	001252	000000	\$USWR: .WORD	USER SWITCHES
753	001254	000000	\$CPUOP: .WORD	CPU TYPE, OPTIONS
754			*****	
755			*****	
756			*****	
757			*****	
758			*****	
759			*****	
760	001256	000	\$MAMS1: .BYTE	AMAMS1
761	001257	000	\$MTYP1: .BYTE	AMTYP1
762			*****	
763			*****	
764			*****	
765			*****	
766	001260	000000	\$MADR1: .WORD	AMADR1
767			*****	
768	001262	000	\$MAMS2: .BYTE	AMAMS2
769	001263	000	\$MTYP2: .BYTE	AMTYP2
770	001264	000000	\$MADR2: .WORD	AMADR2
771	001266	000	\$MAMS3: .BYTE	AMAMS3
772	001267	000	\$MTYP3: .BYTE	AMTYP3
773	001270	000000	\$MADR3: .WORD	AMADR3
774	001272	000	\$MAMS4: .BYTE	AMAMS4
775	001273	000	\$MTYP4: .BYTE	AMTYP4
776	001274	000000	\$MADR4: .WORD	AMADR4
777	001276	000000	\$VECT1: .WORD	AVECT1
778	001300	000000	\$VECT2: .WORD	AVECT2
779	001302	000000	\$BASE: .WORD	ABASE
780	001304	000000	\$DEVN: .WORD	ADEVN
781	001306	000000	\$CDW1: .WORD	ACDW1
782	001310	000000	\$CDW2: .WORD	ACDW2
783	001312	000000	\$DDW0: .WORD	ADDW0
784	001314	000000	\$DDW1: .WORD	ADDW1
785	001316	000000	\$DDW2: .WORD	ADDW2
786	001320	000000	\$DDW3: .WORD	ADDW3
787	001322	000000	\$DDW4: .WORD	ADDW4
788	001324	000000	\$DDW5: .WORD	ADDW5

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
 ;;HIGH ADDRESS, M.S. BYTE
 ;;MEM. TYPE, BLK#1
 MEM. TYPE BYTE -- (HIGH BYTE)
 900 NSEC CORE=001
 300 NSEC BIPOLAR=002
 500 NSEC MOS=003
 ;;HIGH ADDRESS, BLK#1
 MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
 ;;HIGH ADDRESS, M.S. BYTE
 ;;MEM. TYPE, BLK#2
 MEM. LAST ADDRESS, BLK#2
 ;;HIGH ADDRESS, M.S. BYTE
 ;;MEM. TYPE, BLK#3
 MEM. LAST ADDRESS, BLK#3
 ;;HIGH ADDRESS, M.S. BYTE
 ;;MEM. TYPE, BLK#4
 MEM. LAST ADDRESS, BLK#4
 INTERRUPT VECTOR#1, BUS PRIORITY#1
 INTERRUPT VECTOR#2, BUS PRIORITY#2
 BASE ADDRESS OF EQUIPMENT UNDER TEST
 DEVICE MAP
 CONTROLLER DESCRIPTION WORD#1
 CONTROLLER DESCRIPTION WORD#2
 DEVICE DESCRIPTOR WORD#0
 DEVICE DESCRIPTOR WORD#1
 DEVICE DESCRIPTOR WORD#2
 DEVICE DESCRIPTOR WORD#3
 DEVICE DESCRIPTOR WORD#4
 DEVICE DESCRIPTOR WORD#5

G02

MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 19
 DQKTA.P11 07-FEB-77 10:30 APT MAILBOX-ETABLE

789	001326	000000	SDDW6:	.WORD	ADDW6	::: DEVICE	DESCRIPTOR	WORD#6
790	001330	000000	SDDW7:	.WORD	ADDW7	::: DEVICE	DESCRIPTOR	WORD#7
791	001332	000000	SDDW8:	.WORD	ADDW8	::: DEVICE	DESCRIPTOR	WORD#8
792	001334	000000	SDDW9:	.WORD	ADDW9	::: DEVICE	DESCRIPTOR	WORD#9
793	001336	000000	SDDW10:	.WORD	ADDW10	::: DEVICE	DESCRIPTOR	WORD#10
794	001340	000000	SDDW11:	.WORD	ADDW11	::: DEVICE	DESCRIPTOR	WORD#11
795	001342	000000	SDDW12:	.WORD	ADDW12	::: DEVICE	DESCRIPTOR	WORD#12
796	001344	000000	SDDW13:	.WORD	ADDW13	::: DEVICE	DESCRIPTOR	WORD#13
797	001346	000000	SDDW14:	.WORD	ADDW14	::: DEVICE	DESCRIPTOR	WORD#14
798	001350	000000	SDDW15:	.WORD	ADDW15	::: DEVICE	DESCRIPTOR	WORD#15
799								
800								
801	001352		SETEND:					
802								
803								
804	001352	177746	CONTRL:	.WORD	177746		CACHE CONTROL REGISTER	
805	001354	177766	CPUERR:	.WORD	177766		CPU ERROR REGISTER HOLDS	
806							CONDITION THAT CAUSED THE TRAP	
807							TO ERRVEC(000004)	
808	001356	177752	HITMIS:	.WORD	177752		HIT MISS REGISTER "1"	
809							IMPLIES HIT IN CACHE	
810	001360	177744	MEMERR:	.WORD	177744		CACHE ERROR REGISTER	
811	001362	000000	FSTTST:	.WORD	0		HOLDS THE INDEX TO	
812							THE STARTING ADDRESS TABLE	
813	001364	000000	CPUEXP:	.WORD	0		HOLDS EXPECTED CPU ERROR CONDITION	
814	001366	000000	MMEXP:	.WORD			HOLDS EXPECTED MEMORY MANAGEMENT ABORT CONDITION	
815	001370	000000	STKEXP:	.WORD	0		HOLDS TYPE OF STACK VIOLATION EXPECTED	
816	001372	000000	PADRSI:	.WORD	0		HOLDS THE LOWER 16 BITS OF A 18-BIT	
817							ADDRESS GENERATED FOR TYPE OUT	
818	001374	000000	PADRSI:	.WORD	0		HOLDS THE UPPER 6 BITS OF A 18-BIT	
819							ADDRESS GENERATED FOR TYPE OUT	
820	001376	000000	PPARER:	.WORD	0		HOLDS PARITY ERROR REG	
821	001400	000000	PCONTR:	.WORD	0		HOLDS CACHE CONTROL REG	
822	001402	000000	PHITMI:	.WORD	0		HOLDS CACHE HIT MISS REG	
823	001404	000000	PMMRO:	.WORD	0		HOLDS MEMORY MANAGEMENT REGISTER 0	
824	001406	000000	PMMR2:	.WORD	0		HOLDS MEMORY MANAGEMENT REGISTER 2	
825	001410	000000	PCPUER:	.WORD	0		HOLDS CPU ERROR REGISTER.	
826	001412	000000	BADPC:	.WORD	0		HOLDS PC AT ABORT OR TRAP TIME.	
827	001414	000000	TESTNO:	.WORD	0		HOLDS TEST NUMBER OF LAST ERROR.	
828	001416	000000	DATAOR:	.WORD	0		HOLDS LOGICAL OR OF BAD DATA	
829	001420	000000	DATAND:	.WORD	0		HOLDS LOGICAL AND OF BAD DATA	
830	001422	000000	PATTOR:	.WORD	0		HOLDS LOGICAL OR OF PATTERN LOADED	
831	001424	000000	PATAND:	.WORD	0		HOLDS LOGICAL AND OF PATTERN LOADED	
832	001426	000000	ADDROR:	.WORD	0		HOLDS LOGICAL OR OF ADDRESS	
833	001430	000000	ADRAND:	.WORD	0		HOLDS LOGICAL AND OF ADDRESS	
834	001432	000000	ERRCNT:	.WORD	0		HOLDS NUMBER OF ERRORS ON TEST	
835	001434	000000	HOLFLG:	.WORD	0		HOLDS NUMBER OF CONSECUTIVE TIME OUTS	
836							OCCURRING IN A HOLE IN MEMORY	
837	001436	000000	OLDPC:	.WORD	0		HOLDS THE RETURN ADDRESS	
838							IN CASE OF A LOOP ON ERROR	
839	001440	000000	OLDPS:	.WORD	0		HOLDS THE OLD PROCESSOR STATUS	
840							IN CASE OF A LOOP ON ERROR	
841	001442	000340	OLDPSW:	.WORD	000340		HOLDS OLD PSW SO THAT THE T-BIT	
842							CAN BE RESTORED PROPERLY	
843	001444	000000	RETRY:	.WORD	0		FLAG TO DECIDE IF ONE PARITY HAS	
844							HAS BEEN ATTEMPTED	

H02

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 20
DOKTAA.P11 07-FEB-77 10:30 APT MAILBOX-ETABLE

845	001446	000000	NXTTST: .WORD	0	; HOLDS ADDRESS OF THE NEXT TEST
846	001450	000	FORTY: .BYTE	0	; PROCESSOR INDICATOR
847					; IF = 0 RUNNING ON AN 11/6X
848					; IF = 1 RUNNING ON AN 11/40
849	001452		.EVEN		
850					
851			:: THIS AREA STARTS THE DATA TABLE WHERE ANY TABLE REFERENCE WILL		
852			:: REFER TO. ALL DATA WORDS WILL BE LOADED HERE IN ONE SPOT SO THAT		
853			:: IT WILL BE EASIER FOR YOU TO FIND OUT WHAT THAT WORD IS.		
854					
855					
856					
857	001452		PARTAB:		; THIS IS THE TABLE OF THE FIRST
858					; PAR OR PDR OF EACH GROUP. THEY
859					; WILL BE USED FOR A DUAL ADDRESSING
860					; TEST BETWEEN GROUPS.
861	001452	172300	.WORD	172300	; KIPDR0
862	001454	172340	.WORD	172340	; KIPAR0
863	001456	177600	.WORD	177600	; UIPDR0
864	001460	177640	.WORD	177640	; UIPAR0
865	001462	020776	STRTAB: .WORD	TST1	; STARTING ADDRESS OF TEST ONE
866	001464	021510	.WORD	ENTPT2	; ADDRESS OF ENTRY POINT 2
867	001466	022072	.WORD	ENTPT3	; ADDRESS OF ENTRY POINT 3
868	001470	025066	.WORD	ENTPT4	; ADDRESS OF ENTRY POINT 4
869	001472	027204	.WORD	ENTPT5	; ADDRESS OF ENTRY POINT 5
870	001474	033674	.WORD	ENTPT6	; ADDRESS OF ENTRY POINT 6
871	001476	036000	.WORD	ENTPT7	; ADDRESS OF ENTRY POINT 7

.SBTTL ERROR POINTER TABLE

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

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

\$ERRTB:

:ITEM 1

EM1 ;: UNEXPECTED CPU TRAP OR ABORT THRU 'ERRVEC' (004)
 DH1 ;: TESTNO PC AT ABORT
 DT1 ;: TESTNO,BADPC,0
 DF1 ;: 0,0

:ITEM 2

EM1 ;: UNEXPECTED CPU TRAP OR ABORT THRU 'ERRVEC' (004)
 DH2 ;: CPU ERR TESTNO PC AT ABORT
 DT2 ;: PCPUER,TESTNO,BADPC,0
 DF2 ;: 0,0,0

:ITEM 3

EM3 ;: UNEXPECTED CACHE PARITY ERROR THRU 'CACHVEC' (114)
 DH3 ;: MEM ERR CONTROL HITMISS
 ;: REGISTR REGISTR REGISTR TESTNO PC AT ABORT
 DT3 ;: PPARER,PCONTR,PHITMI,TESTNO,BADPC,0
 DF3 ;: 0,0,0,0,0

:ITEM 4

EM4 ;: UNEXPECTED MAIN MEMORY PARITY ERROR THRU 'CACHVEC' (114)
 DH3 ;: MEM ERR CONTROL HITMISS
 ;: REGISTR REGISTR REGISTR TESTNO PC AT ABORT
 DT3 ;: PPARER,PCONTR,PHITMI,TESTNO,BADPC,0
 DF3 ;: 0,0,0,0,0

:ITEM 5

EM5 ;: UNEXPECTED MEMORY MANAGEMENT TRAP OR ABORT
 DH5 ;: ERROR VIRTUAL
 ;: REGISTR ADDRESS TESTNO PC AT ABORT
 DT5 ;: PMMR0,PMMR2,TESTNO,BADPC,0
 DF5 ;: 0,0,0,0

:ITEM 6

EM6 ;: MEMORY MANAGEMENT TRAP OR ABORT HAD INCORRECT CONDITION
 DH6 ;: EXPECTD ERROR VIRTUAL
 ;: CONDITN REGISTR ADDRESS TESTNO PC AT ABORT
 DT6 ;: MMEXP,PMMR0,PMMR2,TESTNO,BADPC,0
 DF6 ;: 0,0,0,0,0

872
 873
 874
 875
 876
 877
 878
 879
 880
 881
 882
 883
 884
 885
 886 001500
 887
 888
 889
 890 001500 002520
 891 001502 007755
 892 001504 013726
 893 001506 014660
 894
 895
 896 001510 002520
 897 001512 010000
 898 001514 013734
 899 001516 014662
 900
 901
 902 001520 002601
 903 001522 010034
 904
 905 001524 013744
 906 001526 014665
 907
 908
 909 001530 002717
 910 001532 010034
 911
 912 001534 013744
 913 001536 014665
 914
 915
 916 001540 003042
 917 001542 010142
 918
 919 001544 013760
 920 001546 014672
 921
 922
 923 001550 003115
 924 001552 010226
 925
 926 001554 013772
 927 001556 014676

928				
929			: ITEM 7	
930	001560	003205	EM7	: MEMORY MANAGEMENT REGISTER 0 WILL NOT CLEAR
931	001562	010332	DH7	: (MMR0) TESTNO ERRORPC
932	001564	014006	DT7	: \$REG1 TESTNO, \$ERRPC, 0
933	001566	014703	DF7	: 0,0,0
934				
935			: ITEM 10	
936	001570	003261	EM10	: CAN'T SET 160000 INTO MMR0
937	001572	010332	DH7	: (MMR0) TESTNO ERRORPC
938	001574	014006	DT7	: \$REG1 TESTNO, \$ERRPC, 0
939	001576	014703	DF7	: 0,0,0
940				
941			: ITEM 11	
942	001600	003312	EM11	: GOT THE WRONG DATA BACK FROM MMR0
943	001602	010362	DH11	: LOADED RECEIVED TESTNO ERRORPC
944	001604	014016	DT11	: \$REG2, \$REG1, TESTNO, \$ERRPC, 0
945	001606	014706	DF11	: 0,0,0,0
946				
947			: ITEM 12	
948	001610	003354	EM12	: MMR2 DID NOT TRACK PROPERLY
949	001612	010422	DH12	: EXPECTED (MMR2) TESTNO ERRORPC
950	001614	014030	DT12	: \$REG3, \$REG2, TESTNO, \$ERRPC
951	001616	014712	DF12	: 0,0,0,0
952				
953			: ITEM 13	
954	001620	003410	EM13	: SUMMARY OF PAR/PDR REFERENCE TIMEOUTS
955	001622	010462	DH13	: ADDROR ADDRAND TESTNO ERRORPC #ERRORS
956	001624	014042	DT13	: ADDROR, ADDRAND, TESTNO, \$ERRPC, \$TMP5, 0
957	001626	014716	DF13	: 0,0,0,0,1
958				
959			: ITEM 14	
960	001630	003456	EM14	: FOLLOWING PAR/PDR WILL NOT ZERO
961	001632	010533	DH14	: ADDRESS DATA TESTNO ERRORPC
962	001634	014056	DT14	: \$REG0, \$REG2, TESTNO, \$ERRPC, 0
963	001636	014723	DF14	: 0,0,0,0
964				
965			: ITEM 15	
966	001640	003517	EM15	: SUMMARY OF DUAL ADDRESSING ERRORS
967	001642	010573	DH15	: ADDROR ADDRAND ADDROR ADDRAND
968				: LOADED LOADED ENABLED ENABLED TESTNO ERRORPC #ERRORS
969	001644	014070	DT15	: ADDROR, ADDRAND, DATAOR, DATAND, TESTNO, \$ERRPC, \$TMP5, 0
970	001646	014727	DF15	: 0,0,0,0,0,0,1
971				
972			: ITEM 16	
973	001650	003561	EM16	: SUMMARY OF COUNT PATTERN FAILURES
974	001652	010722	DH16	: ADDROR ADDRAND PATROR PATRAND DATAOR DATAND TESTNO ERRORPC
975	001654	014110	DT16	: ADDROR ADDRAND, PATTOR, PATAND, DATAOR, DATAND, TESTNO, \$ERRPC, \$TMP5, 0
976	001656	014736	DF16	: 0,0,0,0,0,0,0,0,1
977				
978			: ITEM 17	
979	001660	003623	EM17	: ERROR IN BYTE ADDRESSING OF PAR/PDR
980	001662	011031	DH17	: ADDRESS EXPECTED RECEIVED TESTNO ERRORPC
981	001664	014134	DT17	: \$REG0, \$REG2, \$REG1, TESTNO, \$ERRPC, 0
982	001666	014747	DF17	: 0,0,0,0,0
983				

K02

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006)
 DOKTAA.P11 07-FEB-77 10:30 ERROR POINTER TABLE

07-FEB-77 10:37 PAGE 23

984			: ITEM 20	
985	001670	003667	EM20	: ONE OF THE REGISTERS TIMED OUT
986	001672	011101	DH20	: REGADDR TESTNO ERRORPC
987	001674	014150	DT20	: \$REG0, TESTNO, \$ERRPC, 0
988	001676	014754	DF20	: 0, 0, 0
989				
990			: ITEM 21	
991	001700	003726	EM21	: STACK LIMIT REGISTER WOULD NOT CLEAR
992	001702	011131	DH21	: REGADDR DATA TESTNO ERRORPC
993	001704	014160	DT21	: \$REG0, \$REG1, TESTNO, \$ERRPC, 0
994	001706	014757	DF21	: 0, 0, 0, 0
995				
996			: ITEM 22	
997	001710	003773	EM22	: ERROR LOG REG.DID NOT HAVE CORRECT BIT SET
998	001712	011171	DH22	: REGADDR EXPECTD RECEIVD TESTNO ERRORPC
999	001714	014172	DT22	: \$TMP4, \$TMP5, \$REG0, TESTNO, \$ERRPC, 0
1000	001716	014763	DF22	: 0, 0, 0, 0, 0
1001				
1002			: ITEM 23	
1003	001720	004047	EM23	: LOWER BYTE OF P.S.W. NOT CORRECT
1004	001722	011240	DH23	: REGADDR DATAREC DATAEXP TESTNO ERRORPC
1005	001724	014206	DT23	: \$REG0, \$REG1, \$REG2, TESTNO, \$ERRPC, 0
1006	001726	014770	DF23	: 0, 0, 0, 0, 0
1007				
1008			: ITEM 24	
1009	001730	004110	EM24	: MMRI DID NOT READ ALL ZEROES
1010	001732	011131	DH21	: REGADDR DATA TESTNO ERRORPC
1011	001734	014160	DT21	: \$REG0, \$REG1, TESTNO, \$ERRPC, 0
1012	001736	014757	DF21	: 0, 0, 0, 0
1013				
1014			: ITEM 25	
1015	001740	004145	EM25	: WRONG DATA BACK FROM STACK LIMIT REGISTER
1016	001742	011240	DH23	: REGADDR DATAREC DATAEXP TESTNO ERRORPC
1017	001744	014206	DT23	: \$REG0, \$REG1, \$REG2, TESTNO, \$ERRPC, 0
1018	001746	014770	DF23	: 0, 0, 0, 0, 0
1019				
1020			: ITEM 26	
1021	001750	004217	EM26	: LOADED MORE THAN UPPER BYTE OF STACK LIMIT REGISTER
1022	001752	011240	DH23	: REGADDR DATAREC DATAEXP TESTNO ERRORPC
1023	001754	014206	DT23	: \$REG0, \$REG1, \$REG2, TESTNO, \$ERRPC, 0
1024	001756	014770	DF23	: 0, 0, 0, 0, 0
1025				
1026			: ITEM 27	
1027	001760	004303	EM27	: KERNEL STACK POINTER NOT 1100 AFTER LOADING USP
1028	001762	011310	DH27	: KSP TESTNO ERRORPC
1029	001764	014222	DT27	: \$REG0, TESTNO, \$ERRPC, 0
1030	001766	014775	DF27	: 0, 0, 0
1031				
1032			: ITEM 30	
1033	001770	004363	EM30	: DUAL ADDRESSING BETWEEN PAR/PDR GROUPS
1034	001772	011340	DH30	: INDEX INDEX PAR/PDR
1035				: EXPECTD RECEIVD ADDRREAD TESTNO ERRORPC
1036	001774	014232	DT30	: \$REG0, \$REG1, \$REG2, TESTNO, \$ERRPC, 0
1037	001776	015000	DF30	: 0, 0, 0, 0, 0
1038				
1039			: ITEM 31	

L02

MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 24
 DQKTA.P11 07-FEB-77 10:30 ERROR POINTER TABLE

1040	002000	004432	EM31	:BAD RELOCATION, ON STORING DATA 18-BIT MAPPING
1041	002002	011440	DH31	:VIRTUAL PHYSICAL DATA DATA
1042				:ADDRESS KIPAR4 ADDRESS EXPECTD RECEIVD TESTNO ERRORPC
1043	002004	014246	DT31	:SREG0,\$TMP1,\$TMP2,\$REG1,\$REG2,TESTNO,\$ERRPC,0
1044	002006	015005	DF31	:0,0,0,0,0,0,0
1045				
1046			:ITEM 32	
1047	002010	004511	EM32	:18-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM
1048	002012	011574	DH32	:STRADR FNSHADR TESTNO ERRORPC
1049	002014	014266	DT32	:SREG1,\$TMP2,TESTNO,\$ERRPC,0
1050	002016	015014	DF32	:2,2,0,0
1051				
1052			:ITEM 33	
1053	002020	004572	EM33	:FAULTY CARRY PROPAGATION 18-BIT MAPPING.
1054	002022	011634	DH33	:PATTERN DATA ADDRESS
1055				:LOADED FETCHED INTENDD KIPAR4 KIPARS TESTNO ERRORPC
1056	002024	014300	DT33	:SREG2,\$REG3,\$REG0,\$TMP3,\$TMP4,TESTNO,\$ERRPC,0
1057	002026	015020	DF33	:0,0,0,0,0,0,0
1058				
1059			:ITEM 34	
1060	002030	004643	EM34	:DIDN'T GET WRAP AROUND TO ADDRESS ZERO
1061	002032	011754	DH34	:DATA TESTNO ERRORPC
1062	002034	014320	DT34	:SREG1,TESTNO,\$ERRPC,0
1063	002036	015027	DF34	:0,0,0
1064				
1065			:ITEM 35	
1066	002040	004712	EM35	:PAGE LENGTH ABORT AT WRONG VIRTUAL ADDRESS
1067	002042	012004	DH35	:PGLNFD VABLKNO TESTNO ERRORPC
1068	002044	014330	DT35	:SREG1,\$REG3,TESTNO,\$ERRPC,0
1069	002046	015032	DF35	:0,0,0,0
1070				
1071			:ITEM 36	
1072	002050	004765	EM36	:NO PAGE LENGTH ABORT, WHEN CONDITION CORRECT
1073	002052	012004	DH35	:PGLNFD VABLKNO TESTNO ERRORPC
1074	002054	014330	DT35	:SREG1,\$REG3,TESTNO,\$ERRPC,0
1075	002056	015032	DF35	:0,0,0,0
1076				
1077			:ITEM 37	
1078	002060	005042	EM37	:DID NOT ABORT ON ACCESS OF NON-RESIDENT PAGE THRU KIPDRS
1079	002062	012044	DH37	:KIPDR5 KIPARS VIRTADR TESTNO ERRORPC
1080	002064	014342	DT37	:\$TMP0,\$TMP1,\$TMP2,TESTNO,\$ERRPC,0
1081	002066	015036	DF37	:0,0,0,0,0
1082				
1083			:ITEM 40	
1084	002070	005133	EM40	:DID NOT ABORT ON WRITE TO READ ONLY PAGE THRU KIPDR4
1085	002072	012114	DH40	:KIPDR4 KIPAR4 VIRTADR TESTNO ERRORPC
1086	002074	014342	DT37	:\$TMP0,\$TMP1,\$TMP2,TESTNO,\$ERRPC,0
1087	002076	015036	DF37	:0,0,0,0,0
1088				
1089			:ITEM 41	
1090	002100	005220	EM41	:POINTER VALUE 400 CAUSED STACK VIOLATION
1091	002102	012164	DH41	:STKPTR STKLMT TESTNO ERRORPC
1092	002104	014356	DT41	:SREG2,\$REG3,TESTNO,\$ERRPC,0
1093	002106	015043	DF41	:0,0,0,0
1094				
1095			:ITEM 42	

M02

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006)
 DOKTAA.P11 07-FEB-77 10:30 ERROR POINTER TABLE

07-FEB-77 10:37 PAGE 25

1096	002110	005271	EM42	: YELLOW ZONE VIOLATION EXPECTED-GOT NONE
1097	002112	012164	DH41	: STKPTR STKLMT TESTNO ERRORPC
1098	002114	014356	DT41	: \$REG2, \$REG3, TESTNO, \$ERRPC, 0
1099	002116	015043	DF41	: 0,0,0,0
1100				
1101			: ITEM 43	
1102	002120	005341	EM43	: YELLOW ZONE VIOLATION EXPECTED-GOT RED
1103	002122	012164	DH41	: STKPTR STKLMT TESTNO ERRORPC
1104	002124	014356	DT41	: \$REG2, \$REG3, TESTNO, \$ERRPC, 0
1105	002126	015043	DF41	: 0,0,0,0
1106				
1107			: ITEM 44	
1108	002130	005410	EM44	: RED ZONE VIOLATION EXPECTED-GOT NONE
1109	002132	012164	DH41	: STKPTR STKLMT TESTNO ERRORPC
1110	002134	014356	DT41	: \$REG2, \$REG3, TESTNO, \$ERRPC, 0
1111	002136	015043	DF41	: 0,0,0,0
1112				
1113			: ITEM 45	
1114	002140	005455	EM45	: RED ZONE VIOLATION DID NOT CAUSE ABORT
1115	002142	012164	DH41	: STKPTR STKLMT TESTNO ERRORPC
1116	002144	014356	DT41	: \$REG2, \$REG3, TESTNO, \$ERRPC, 0
1117	002146	015043	DF41	: 0,0,0,0
1118				
1119			: ITEM 46	
1120	002150	005524	EM46	: RED ZONE VIOLATION EXPECTED - GOT YELLOW
1121	002152	012164	DH41	: STKPTR STKLMT TESTNO ERRORPC
1122	002154	014356	DT41	: \$REG2, \$REG3, TESTNO, \$ERRPC, 0
1123	002156	015043	DF41	: 0,0,0,0
1124				
1125			: ITEM 47	
1126	002160	005575	EM47	: ERROR DURING ERROR-ON-ERROR TRAP SEQUENCE
1127				: EXPECTED:
1128	002162	012224	DH47	: PSW STK PTR (MMR0) (MMR2) TESTNO ERRORPC
1129				: 140017 -11/6X- 000000 020151
1130				: 140000 -11/40- 481074
1131				: RECEIVED:
1132				: PSW PC STK PTR (MMR0) (MMR2) TESTNO ERRORPC
1133	002164	014370	DT47	: \$REG2, \$REG3, \$REG1, PMMR0, PMMR2, TESTNO, \$ERRPC, 0
1134	002166	015047	DF47	: 0,0,0,0,0,0,0
1135				
1136			: ITEM 50	
1137	002170	005661	EM50	: AT LEAST ONE M.M. STATUS REGISTERS WAS CLOCKED AFTER BEING LOCK
1138	002172	012467	DH50	: ORIGINAL DATA NEW DATA
1139				: (MMR0) (MMR2) (MMR0) (MMR2) TESTNO ERRORPC
1140	002174	014410	DT50	: PMMR0, PMMR2, STMP0, STMP2, TESTNO, \$ERRPC, 0
1141	002176	015056	DF50	: 0,0,0,0,0,0
1142				
1143			: ITEM 51	
1144	002200	005763	EM51	: DID NOT CHANGE MAPPING TO USER MODE
1145	002202	012044	DH37	: TESTNO ERRORPC
1146	002204	014342	DT37	: TESTNO, \$ERRPC, 0
1147	002206	015036	DF37	: 0,0
1148				
1149			: ITEM 52	
1150	002210	006027	EM52	: ABORT CONDITION INCORRECT EXPECTING 100153
1151	002212	012600	DH52	: RECEIVED (MMR2) TESTNO ERRORPC

N02

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006)
DOKTAA.P11 07-FEB-77 10:30 ERROR POINTER TABLE

07-FEB-77 10:37 PAGE 26

1152	002214	014426	DT52	;PMMR0,PMMR2,TESTNO,\$ERRPC,0
1153	002216	015064	DF52	;0,0,0,0
1154				
1155			: ITEM 53	
1156	002220	006102	EM53	;MMR2 LOCKED UP THE WRONG VIRTUAL ADDRESS
1157	002222	012640	DH53	;VIRTADR (MMR2) TESTNO ERRORPC
1158	002224	014440	DT53	;SREG1,PMMR2,TESTNO,\$ERRPC,0
1159	002226	015070	DF53	;0,0,0,0
1160				
1161			: ITEM 54	
1162	002230	006153	EM54	;MMR0 LOCKED UP THE WRONG PAGE NUMBER
1163	002232	012700	DH54	;EXPECTD (MMR0) TESTNO ERRORPC
1164	002234	014432	DT54	;SREG2,PMMR0,TESTNO,\$ERRPC,0
1165	002236	015074	DF54	;0,0,0,0
1166				
1167			: ITEM 55	
1168	002240	006220	EM55	;W-BIT NOT SET ON WRITE TO PAGE 4
1169	002242	012740	DH55	;KIPDR4 TESTNO ERRORPC
1170	002244	014464	DT55	;STMP0,TESTNO,\$ERRPC,0
1171	002246	015100	DF55	;0,0,0
1172				
1173			: ITEM 56	
1174	002250	006261	EM56	;W-BIT DID NOT REMAIN SET ON M.M. ABORT AT PAGE 4
1175	002252	012740	DH55	;KIPDR4 TESTNO ERRORPC
1176	002254	014464	DT55	;STMP0,TESTNO,\$ERRPC,0
1177	002256	015100	DF55	;0,0,0
1178				
1179			: ITEM 57	
1180	002260	006342	EM57	;W-BIT DID NOT CLEAR ON WRITE TO KIPDR4
1181	002262	012740	DH55	;KIPDR4 TESTNO ERRORPC
1182	002264	014464	DT55	;STMP0,TESTNO,\$ERRPC,0
1183	002266	015100	DF55	;0,0,0
1184				
1185			: ITEM 60	
1186	002270	006411	EM60	;W-BIT WRONG ON WRITE TO PSW
1187	002272	012770	DH60	;KIPDR7 TESTNO ERRORPC
1188	002274	014464	DT55	;STMP0,TESTNO,\$ERRPC,0
1189	002276	015100	DF55	;0,0,0
1190				
1191			: ITEM 61	
1192	002300	006445	EM61	;W-BIT SET AFTER WRITE TO PDR 4
1193	002302	012770	DH60	;KIPDR7 TESTNO ERRORPC
1194	002304	014464	DT55	;STMP0,TESTNO,\$ERRPC,0
1195	002306	015100	DF55	;0,0,0
1196				
1197			: ITEM 62	
1198	002310	006504	EM62	;DUAL MAPPING BETWEEN PAGES
1199	002312	013020	DH62	;GDPAGE BDPAGE TESTNO ERRORPC
1200	002314	014474	DT62	;SREG3,\$REG1,TESTNO,\$ERRPC,0
1201	002316	015103	DF62	;0,0,0,0
1202				
1203			: ITEM 63	
1204	002320	006537	EM63	;NO PAGE HAD ITS BIT SET
1205	002322	013060	DH63	;TSTPAGE CONTENT TESTNO ERRORPC
1206	002324	014506	DT63	;SREG3,\$REG0,TESTNO,\$ERRPC,0
1207	002326	015107	DF63	;0,0,0,0

1208				
1209			: ITEM 64	
1210	002330	006567	EM64	: DID NOT PICK UP CORRECT STACK POINTER
1211	002332	013120	DH64	: EXPECTD RECEIVD TESTNO ERRORPC
1212	002334	014520	DT64	: \$REG2, \$REG1, TESTNO, \$ERRPC, 0
1213	002336	015113	DF64	: 0,0,0,0
1214				
1215			: ITEM 65	
1216	002340	006635	EM65	: CURRENT MODE STACK NOT PUSHED IN MFP INSTRUCTION
1217	002342	012044	DH37	: TESTNO ERRORPC
1218	002344	014342	DT37	: TESTNO, \$ERRPC, 0
1219	002346	015036	DF37	: 0,0
1220				
1221			: ITEM 66	
1222	002350	006716	EM66	: WRONG DATA FETCHED BY MFP INSTRUCTION
1223	002352	013160	DH66	: EXPECTD RECEIVD TESTNO ERRORPC
1224	002354	014532	DT66	: \$REG0, \$REG1, TESTNO, \$ERRPC, 0
1225	002356	015117	DF66	: 0,0,0,0
1226				
1227			: ITEM 67	
1228	002360	006764	EM67	: TRIED TO REFERENCE NON-RESIDENT PAGE
1229	002362	013220	DH67	: (MMR0) (MMR2) TESTNO ERRORPC
1230	002364	014544	DT67	: PMMR0, PMMR2, TESTNO, \$ERRPC, 0
1231	002366	015123	DF67	: 0,0,0,0
1232				
1233			: ITEM 70	
1234	002370	007031	EM70	: STACK POINTER NOT CHANGED BY MTP INSTRUCTION
1235	002372	013260	DH70	: STKPTR TESTNO ERRORPC
1236	002374	014556	DT70	: \$REG1, TESTNO, \$ERRPC, 0
1237	002376	015127	DF70	: 0,0,0
1238				
1239			: ITEM 71	
1240	002400	007106	EM71	: INCORRECT STORE BY MTP INSTRUCTION
1241	002402	013310	DH71	: GDDATA STORED TESTNO ERRORPC
1242	002404	014532	DT66	: \$REG0, \$REG1, TESTNO, \$ERRPC, 0
1243	002406	015117	DF66	: 0,0,0,0
1244				
1245			: ITEM 72	
1246	002410	007151	EM72	: ABORTED THRU RIGHT VECTOR BUT PICKED UP WRONG PSW
1247	002412	013350	DH72	: (PSW) TESTNO ERRORPC EXPECTING XXX340
1248	002414	014566	DT72	: \$REG0, TESTNO, \$ERRPC, 0
1249	002416	015132	DF72	: 0,0,0
1250				
1251			: ITEM 73	
1252	002420	007233	EM73	: ABORTED THRU USER SPACE, USER PSW IS XXX000
1253	002422	013421	DH73	: (PSW) TESTNO ERRORPC
1254	002424	014566	DT72	: \$REG0, TESTNO, \$ERRPC, 0
1255	002426	015132	DF72	: 0,0,0
1256				
1257			: ITEM 74	
1258	002430	007307	EM74	: DID NOT ABORT REFERENCE TO A MEMORY MANAGEMENT REGISTER
1259	002432	013451	DH74	: TESTNO ERRORPC
1260	002434	014576	DT74	: TESTNO, \$ERRPC, 0
1261	002436	015135	DF74	: 0,0
1262				
1263			: ITEM 75	

C03

MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 28
 DQKTA.P11 07-FEB-77 10:30 ERROR POINTER TABLE

1264	002440	007377	EM75	; CPU ERROR REG. DID NOT REPORT YELLOW ZONE
1265	002442	013470	DH75	; STKPTR STKLMT CPUERR TESTNO ERRORPC
1266	002444	014604	DT75	; \$REG2, \$REG3, PCPUER, TESTNO, \$ERRPC, 0
1267	002446	015137	DF75	; 0,0,0,0,0
1268				
1269			: ITEM 76	
1270	002450	007451	EM76	; CPU ERROR REG. DID NOT REPORT RED ZONE
1271	002452	013470	DH75	; STKPTR STKLMT CPUERR TESTNO ERRORPC
1272	002454	014604	DT75	; \$REG2, \$REG3, PCPUER, TESTNO, \$ERRPC, 0
1273	002456	015137	DF75	; 0,0,0,0,0
1274				
1275			: ITEM 77	
1276	002460	002717	EM4	; UNEXPECTED MAIN MEMORY PARITY ERROR THRU 'CACHVEC' (114)
1277	002462	007755	DH1	; TESTNO P` AT ABORT
1278	002464	013726	DT1	; TESTNO, BADPC, 0
1279	002466	014660	DF1	; 0,0
1280				
1281				
1282				
1283	002470		ER200:	; STARTING ADDRESS FOR ITEM 201-377
1284				
1285			: ITEM 201	
1286	002470	007520	EM201	; THE FOLLOWING ARE PAR/PDR REFERENCE TIMEOUTS
1287	002472	013540	DH201	; ADDRESS TESTNO ERRORPC
1288				
1289			: ITEM 301	
1290	002474	014620	DT201	; \$REG0, TESTNO, \$ERRPC, 0
1291	002476	015144	DF201	; 0,0,0
1292				
1293			: ITEM 202	
1294	002500	007575	EM202	; THE FOLLOWING ARE DUAL ADDRESSING ERRORS FOR PAR'S/PDR'S
1295	002502	013567	DH202	; ADDRESS ADDRESS
1296				; LOADED JUST READ TESTNO ERRORPC
1297				
1298			: ITEM 302	
1299	002504	014630	DT202	; \$REG0, \$REG1, TESTNO, \$ERRPC, 0
1300	002506	015147	DF202	; 0,0,0,0
1301				
1302			: ITEM 203	
1303	002510	007666	EM203	; THE FOLLOWING ARE COUNT PATTERN ERRORS FOR PAR'S/PDR'S
1304	002512	013647	DH203	; ADDRESS RECEIVD EXPECTD COUNT TESTNO ERRORPC
1305				
1306			: ITEM 303	
1307	002514	014642	DT203	; \$REG0, \$REG2, \$REG4, \$REG1, TESTNO, \$ERRPC, 0
1308	002516	015153	DF203	; 0,0,0,0,0,0
1309				
1310				
1311			.SBTTL	ERROR TABLE MESSAGES AND DATA POINTERS
1312	002520	047125	054105	042520
1313	002526	052103	042105	041440
1314	002534	052520	052040	040522
1315	002542	020120	051117	040440
1316	002550	047502	052122	052040
1317	002556	051110	020125	042447
1318	002564	051122	042526	023503
1319	002572	024040	030060	024464

EM1: .ASCIZ 'UNEXPECTED CPU TRAP OR ABORT THRU 'ERRVEC' (004)?

003

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 29
 DOKTAA.P11 07-FEB-77 10:30 ERROR TABLE MESSAGES AND DATA POINTERS

1320	002600	000			
1321	002601	125	042516	050130	EM3: .ASCII ?UNEXPECTED CACHE PARITY ERROR THRU 'CACHVEC' (114)?<CRLF>
1322	002606	041505	042524	020104	
1323	002614	040503	044103	020105	
1324	002622	040520	044522	054524	
1325	002630	042440	051122	051117	
1326	002636	052040	051110	020125	
1327	002644	041447	041501	053110	
1328	002652	041505	020047	030450	
1329	002660	032061	100051		
1330	002664	044527	046114	051040	.ASCIZ ?WILL RETRY THIS TEST ONCE.?
1331	002672	052105	054522	052040	
1332	002700	044510	020123	042524	
1333	002706	052123	047440	041516	
1334	002714	027105	000		
1335	002717	125	042516	050130	EM4: .ASCII ?UNEXPECTED MAIN MEMORY PARITY ERROR THRU 'CACHVEC' (114)?<CRLF>
1336	002724	041505	042524	020104	
1337	002732	040515	047111	046440	
1338	002740	046505	051117	020131	
1339	002746	040520	044522	054524	
1340	002754	042440	051122	051117	
1341	002762	052040	051110	020125	
1342	002770	041447	041501	053110	
1343	002776	041505	020047	030450	
1344	003004	032061	100051		
1345	003010	044527	046114	051040	.ASCIZ ?WILL RETRY THIS TEST ONCE?
1346	003016	052105	054522	052040	
1347	003024	044510	020123	042524	
1348	003032	052123	047440	041516	
1349	003040	000105			
1350					
1351	003042	047125	054105	042520	EM5: .ASCIZ ?UNEXPECTED MEMORY MANAGEMENT TRAP OR ABORT?
1352	003050	052103	042105	046440	
1353	003056	046505	051117	020131	
1354	003064	040515	040516	042507	
1355	003072	042515	052116	052040	
1356	003100	040522	020120	051117	
1357	003106	040440	047502	052122	
1358	003114	000			
1359	003115	115	046505	051117	EM6: .ASCIZ ?MEMORY MANAGEMENT TRAP OR ABORT HAD INCORRECT CONDITION?
1360	003122	020131	040515	040516	
1361	003130	042507	042515	052116	
1362	003136	052040	040522	020120	
1363	003144	051117	040440	047502	
1364	003152	052122	044040	042101	
1365	003160	044440	041516	051117	
1366	003166	042522	052103	041440	
1367	003174	047117	044504	044524	
1368	003202	047117	000		
1369	003205	115	046505	051117	EM7: .ASCIZ ?MEMORY MANAGEMENT REGISTER 0 WILL NOT CLEAR?
1370	003212	020131	040515	040516	
1371	003220	042507	042515	052116	
1372	003226	051040	043505	051511	
1373	003234	042524	020122	020060	
1374	003242	044527	046114	047040	
1375	003250	052117	041440	042514	

E03

MAINDEC-11-DQKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 30
 DQKTA.P11 07-FEB-77 10:30 ERROR TABLE MESSAGES AND DATA POINTERS

1376	003256	051101	000		
1377	003261	103	047101	052047	EM10: .ASCIZ 'CAN'T SET 160000 IN MMRO'
1378	003266	051440	052105	030440	
1379	003274	030066	030060	020060	
1380	003302	047111	046440	051115	
1381	003310	000060			
1382	003312	047507	020124	044124	EM11: .ASCIZ 'GOT THE WRONG DATA BACK FROM MMRO'
1383	003320	020105	051127	047117	
1384	003326	020107	040504	040524	
1385	003334	041040	041501	020113	
1386	003342	051106	046517	046440	
1387	003350	051115	000060		
1388	003354	046515	031122	042040	EM12: .ASCIZ 'MMR2 DID NOT TRACK PROPERLY'
1389	003362	042111	047040	052117	
1390	003370	052040	040522	045503	
1391	003376	050040	047522	042520	
1392	003404	046122	000131		
1393	003410	052523	046515	051101	EM13: .ASCIZ 'SUMMARY OF PAR/PDR REFERENCE TIMEOUTS'
1394	003416	020131	043117	050040	
1395	003424	051101	050057	051104	
1396	003432	051040	043105	051105	
1397	003440	047105	042503	052040	
1398	003446	046511	047505	052125	
1399	003454	000123			
1400	003456	047506	046114	053517	EM14: .ASCIZ 'FOLLOWING PAR/PDR WILL NOT CLEAR'
1401	003464	047111	020107	040520	
1402	003472	027522	042120	020122	
1403	003500	044527	046114	047040	
1404	003506	052117	041440	042514	
1405	003514	051101	000		
1406	003517	123	046525	040515	EM15: .ASCIZ 'SUMMARY OF DUAL ADDRESSING ERRORS'
1407	003524	054522	047440	020106	
1408	003532	052504	046101	040440	
1409	003540	042104	042522	051523	
1410	003546	047111	020107	051105	
1411	003554	047522	051522	000	
1412	003561	123	047525	040515	EM16: .ASCIZ 'SUMMARY OF COUNT PATTERN FAILURES'
1413	003566	054522	047440	020106	
1414	003574	047503	047125	020124	
1415	003602	040520	052124	051105	
1416	003610	020116	040506	046111	
1417	003616	051125	051505	000	
1418	003623	105	051122	051117	EM17: .ASCIZ 'ERROR IN BYTE ADDRESSING OF PAR/PDR'
1419	003630	044440	020116	054502	
1420	003636	042524	040440	042104	
1421	003644	042522	051523	047111	
1422	003652	020107	043117	050040	
1423	003660	051101	050057	051104	
1424	003666	000			
1425	003667	117	042516	047440	EM20: .ASCIZ 'ONE OF THE REGISTERS TIMED OUT'
1426	003674	020106	044124	020105	
1427	003702	042522	044507	052123	
1428	003710	051105	020123	044524	
1429	003716	042515	020104	052517	
1430	003724	000124			
1431	003726	052123	041501	020113	EM21: .ASCIZ 'STACK LIMIT REGISTER WOULD NOT CLEAR'

F03

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 31
 DOKTAA.P11 07-FEB-77 10:30 ERROR TABLE MESSAGES AND DATA POINTERS

1432	003734	044514	044515	020124	
1433	003742	042522	044507	052123	
1434	003750	051105	053440	052517	
1435	003756	042114	047040	052117	
1436	003764	041440	042514	051101	
1437	003772	000			
1438	003773	105	051122	051117	EM22: .ASCIZ 'ERROR LOG REG. DID NOT HAVE CORRECT BIT SET'
1439	004000	046040	043517	051040	
1440	004006	043505	020056	044504	
1441	004014	020104	047516	020124	
1442	004022	040510	042526	041440	
1443	004030	051117	042522	052103	
1444	004036	041040	052111	051440	
1445	004044	052105	000		
1446	004047	114	053517	051105	EM23: .ASCIZ 'LOWER BYTE OF P.S.W. NOT CORRECT'
1447	004054	041040	052131	020105	
1448	004062	043117	050040	051456	
1449	004070	053456	020056	047516	
1450	004076	020124	047503	051122	
1451	004104	041505	000124		
1452	004110	046515	030522	042040	EM24: .ASCIZ 'MMR1 DID NOT READ ALL ZEROES'
1453	004116	042111	047040	052117	
1454	004124	051040	040505	020104	
1455	004132	046101	020114	042532	
1456	004140	047522	051505	000	
1457	004145	127	047522	043516	EM25: .ASCIZ 'WRONG DATA BACK FROM STACK LIMIT REGISTER'
1458	004152	042040	052101	020101	
1459	004160	040502	045503	043040	
1460	004166	047522	020115	052123	
1461	004174	041501	020113	044514	
1462	004202	044515	020124	042522	
1463	004210	044507	052123	051105	
1464	004216	000			
1465	004217	114	040517	042504	EM26: .ASCIZ 'LOADED MORE THAN UPPER BYTE OF STACK LIMIT REGISTER'
1466	004224	020104	047515	042522	
1467	004232	052040	040510	020116	
1468	004240	050125	042520	020122	
1469	004246	054502	042524	047440	
1470	004254	020106	052123	041501	
1471	004262	020113	044514	044515	
1472	004270	020124	042522	044507	
1473	004276	052123	051105	000	
1474	004303	113	051105	042516	EM27: .ASCIZ 'KERNEL STACK POINTER NOT 1100 AFTER LOADING USP'
1475	004310	020114	052123	041501	
1476	004316	020113	047520	047111	
1477	004324	042524	020122	047516	
1478	004332	020124	030461	030060	
1479	004340	040440	052106	051105	
1480	004346	046040	040517	044504	
1481	004354	043516	052440	050123	
1482	004362	000			
1483	004363	104	040525	020114	EM30: .ASCIZ 'DUAL ADDRESSING BETWEEN PAR/PDR GROUPS'
1484	004370	042101	051104	051505	
1485	004376	044523	043516	041040	
1486	004404	052105	042527	047105	
1487	004412	050040	051101	050057	

G03

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 32
 DOKTAA.P11 07-FEB-77 10:30 ERROR TABLE MESSAGES AND DATA POINTERS

1488	004420	051104	043440	047522	
1489	004426	050125	000123		
1490	004432	040502	020104	042522	EM31: .ASCIZ 'BAD RELOCATION, ON STORING DATA 18-BIT MAPPING'
1491	004440	047514	040503	044524	
1492	004446	047117	020054	047117	
1493	004454	051440	047524	044522	
1494	004462	043516	042040	052101	
1495	004470	020101	034061	041055	
1496	004476	052111	046440	050101	
1497	004504	044520	043516	000	
1498	004511	061	026470	044502	EM32: .ASCIZ '18-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM'
1499	004516	020124	040515	050120	
1500	004524	047111	020107	047520	
1501	004532	051523	041111	042514	
1502	004540	044040	046117	020105	
1503	004546	047111	046440	044501	
1504	004554	020116	042515	047515	
1505	004562	054522	043040	047522	
1506	004570	000115			
1507	004572	040506	046125	054524	EM33: .ASCIZ 'FAULTY CARRY PROPAGATION 18-BIT MAPPING.'
1508	004600	041440	051101	054522	
1509	004606	050040	047522	040520	
1510	004614	040507	044524	047117	
1511	004622	030440	026470	044502	
1512	004630	020124	040515	050120	
1513	004636	047111	027107	000	
1514	004643	104	042111	023516	EM34: .ASCIZ 'DIDN'T GET WRAP AROUND TO ADDRESS ZERO'
1515	004650	020124	042507	020124	
1516	004656	051127	050101	040440	
1517	004664	047522	047125	020104	
1518	004672	047524	040440	042104	
1519	004700	042522	051523	055040	
1520	004706	051105	000117		
1521	004712	040520	042507	046040	EM35: .ASCIZ 'PAGE LENGTH ABORT AT WRONG VIRTUAL ADDRESS'
1522	004720	047105	052107	020110	
1523	004726	041101	051117	020124	
1524	004734	052101	053440	047522	
1525	004742	043516	053040	051111	
1526	004750	052524	046101	040440	
1527	004756	042104	042522	051523	
1528	004764	000			
1529	004765	116	020117	040520	EM36: .ASCIZ 'NO PAGE LENGTH ABORT, WHEN CONDITION CORRECT'
1530	004772	042507	046040	047105	
1531	005000	052107	020110	041101	
1532	005006	051117	026124	053440	
1533	005014	042510	020116	047503	
1534	005022	042116	052111	047511	
1535	005030	020116	047503	051122	
1536	005036	041505	000124		
1537	005042	044504	020104	047516	EM37: .ASCIZ 'DID NOT ABORT ON ACCESS OF NON-RESIDENT PAGE THRU KIPDR5'
1538	005050	020124	041101	051117	
1539	005056	020124	047117	040440	
1540	005064	041503	051505	020123	
1541	005072	043117	047040	047117	
1542	005100	051055	051505	042111	
1543	005106	047105	020124	040520	

H03

MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 33
 DQKTA.P11 07-FEB-77 10:30 ERROR TABLE MESSAGES AND DATA POINTERS

1544	005114	042507	052040	051110	
1545	005122	020125	044513	042120	
1546	005130	032522	000		
1547	005133	104	042111	047040	EM40: .ASCIZ 'DID NOT ABORT ON WRITE TO READ ONLY PAGE THRU KIPDR4'
1548	005140	052117	040440	047502	
1549	005146	052122	047440	020116	
1550	005154	051127	052111	020105	
1551	005162	047524	051040	040505	
1552	005170	020104	047117	054514	
1553	005176	050040	043501	020105	
1554	005204	044124	052522	045440	
1555	005212	050111	051104	000064	
1556	005220	047520	047111	042524	EM41: .ASCIZ 'POINTER VALUE 400 CAUSED STACK VIOLATION'
1557	005226	020122	040526	052514	
1558	005234	020105	030064	020060	
1559	005242	040503	051525	042105	
1560	005250	051440	040524	045503	
1561	005256	053040	047511	040514	
1562	005264	044524	047117	000	
1563	005271	131	046105	047514	EM42: .ASCIZ 'YELLOW ZONE VIOLATION EXPECTED-GOT NONE'
1564	005276	020127	047532	042516	
1565	005304	053040	047511	040514	
1566	005312	044524	047117	042440	
1567	005320	050130	041505	042524	
1568	005326	026504	047507	020124	
1569	005334	047516	042516	000	
1570	005341	131	046105	047514	EM43: .ASCIZ 'YELLOW ZONE VIOLATION EXPECTED-GOT RED'
1571	005346	020127	047532	042516	
1572	005354	053040	047511	040514	
1573	005362	044524	047117	042440	
1574	005370	050130	041505	042524	
1575	005376	026504	047507	020124	
1576	005404	042522	000104		
1577	005410	042522	020104	047532	EM44: .ASCIZ 'RED ZONE VIOLATION EXPECTED-GOT NONE'
1578	005416	042516	053040	047511	
1579	005424	040514	044524	047117	
1580	005432	042440	050130	041505	
1581	005440	042524	026504	047507	
1582	005446	020124	047516	042516	
1583	005454	000			
1584	005455	122	042105	055040	EM45: .ASCIZ 'RED ZONE VIOLATION DID NOT CAUSE ABORT'
1585	005462	047117	020105	044526	
1586	005470	046117	052101	047511	
1587	005476	020116	044504	020104	
1588	005504	047516	020124	040503	
1589	005512	051525	020105	041101	
1590	005520	051117	000124		
1591	005524	042522	020104	047532	EM46: .ASCIZ 'RED ZONE VIOLATION EXPECTED - GOT YELLOW'
1592	005532	042516	053040	047511	
1593	005540	040514	044524	047117	
1594	005546	042440	050130	041505	
1595	005554	042524	020104	020055	
1596	005562	047507	020124	042531	
1597	005570	046114	053517	000	
1598	005575	105	051122	051117	EM47: .ASCII 'ERROR DURING ERROR-ON-ERROR TRAP SEQUENCE'<CRLF>
1599	005602	042040	051125	047111	

1600	005610	020107	051105	047522	
1601	005616	026522	047117	042455	
1602	005624	051122	051117	052040	
1603	005632	040522	020120	042523	
1604	005640	052521	047105	042503	
1605	005646	200			
1606	005647	105	050130	041505	.ASCIZ ?EXPECTED:?
1607	005654	042524	035104	000	
1608	005661	101	020124	042514	EM50: .ASCIZ ?AT LEAST ONE M.M. STATUS REGISTERS WAS CLOCKED AFTER BEING LOCKED?
1609	005666	051501	020124	047117	
1610	005674	020105	027115	027115	
1611	005702	051440	040524	052524	
1612	005710	020123	042522	044507	
1613	005716	052123	051105	020123	
1614	005724	040527	020123	046103	
1615	005732	041517	042513	020104	
1616	005740	043101	042524	020122	
1617	005746	042502	047111	020107	
1618	005754	047514	045503	042105	
1619	005762	000			
1620	005763	104	042111	047040	EM51: .ASCIZ ?DID NOT CHANGE MAPPING TO USER MODE?
1621	005770	052117	041440	043510	
1622	005776	043516	020105	043515	
1623	006004	050120	047111	020107	
1624	006012	047524	052440	042500	
1625	006020	020122	047515	042504	
1626	006026	000			
1627	006027	101	047502	052122	EM52: .ASCIZ ?ABORT CONDITION INCORRECT EXPECTING 100153?
1628	006034	041440	047117	044504	
1629	006042	044524	047117	044440	
1630	006050	041516	051117	042522	
1631	006056	052103	042440	050130	
1632	006064	041505	044524	043516	
1633	006072	030140	030060	032461	
1634	006100	000063			
1635	006102	046515	031122	046040	EM53: .ASCIZ ?MMR2 LOCKED UP THE WRONG VIRTUAL ADDRESS?
1636	006110	041517	042513	020104	
1637	006116	050125	052040	042510	
1638	006124	053440	047522	043516	
1639	006132	053040	051111	052524	
1640	006140	046101	040440	042104	
1641	006146	042522	051523	000	
1642	006153	115	051115	020060	EM54: .ASCIZ ?MMR0 LOCKED UP THE WRONG PAGE NUMBER?
1643	006160	047514	045503	042105	
1644	006166	052440	020120	044124	
1645	006174	020105	051127	047117	
1646	006202	020107	040520	042507	
1647	006210	047040	046525	042502	
1648	006216	000122			
1649	006220	026527	044502	020124	EM55: .ASCIZ ?W-BIT NOT SET ON WRITE TO PAGE 4?
1650	006226	047516	020124	042523	
1651	006234	020124	047117	053440	
1652	006242	044522	042524	052040	
1653	006250	020117	040520	042507	
1654	006256	032040	000		
1655	006261	127	041055	052111	EM56: .ASCIZ ?W-BIT DID NOT REMAIN SET ON M.M. ABORT AT PAGE 4?

J03

MAINDEC-11-DOKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 35
 DOKTAA.P11 07-FEB-77 10:30 ERROR TABLE MESSAGES AND DATA POINTERS

1656	006266	042040	042111	047040	
1657	006274	052117	051040	046505	
1658	006302	044501	020116	042523	
1659	006310	020124	047117	046440	
1660	006316	046456	020056	041101	
1661	006324	051117	020124	052101	
1662	006332	050040	043501	020105	
1663	006340	000064			
1664	006342	026527	044502	020124	EM57: .ASCIZ ?W-BIT DID NOT CLEAR ON WRITE TO KIPDR4?
1665	006350	044504	020104	047516	
1666	006356	020124	046103	040505	
1667	006364	020122	047117	053440	
1668	006372	044522	042524	052040	
1669	006400	020117	044513	042120	
1670	006406	032122	000		
1671	006411	127	041055	052111	EM60: .ASCIZ ?W-BIT WRONG ON WRITE TO PSW?
1672	006416	053440	047522	043516	
1673	006424	047440	020116	051127	
1674	006432	052111	020105	047524	
1675	006440	050040	053523	000	
1676	006445	127	041055	052111	EM61: .ASCIZ ?W-BIT SET AFTER WRITE TO PDR 4?
1677	006452	051440	052105	040440	
1678	006460	052106	051105	053440	
1679	006466	044522	042524	052040	
1680	006474	020117	042120	020122	
1681	006502	000064			
1682	006504	052504	046101	046440	EM62: .ASCIZ ?DUAL MAPPING BETWEEN PAGES?
1683	006512	050101	044520	043516	
1684	006520	041040	052105	042527	
1685	006526	047105	050040	043501	
1686	006534	051505	000		
1687	006537	116	020117	040520	EM63: .ASCIZ ?NO PAGE HAD ITS BIT SET?
1688	006544	042507	044040	042101	
1689	006552	044440	051524	041040	
1690	006560	052111	051440	052105	
1691	006566	000			
1692	006567	104	042111	047040	EM64: .ASCIZ ?DID NOT PICK UP CORRECT STACK POINTER?
1693	006574	052117	050040	041511	
1694	006602	020113	050125	041440	
1695	006610	051117	042522	052103	
1696	006616	051440	040524	045503	
1697	006624	050040	044517	052116	
1698	006632	051105	000		
1699	006635	103	051125	042522	EM65: .ASCIZ ?CURRENT MODE STACK NOT PUSHED IN MFP INSTRUCTION?
1700	006642	052116	046440	042117	
1701	006650	020105	052123	041501	
1702	006656	020113	047516	020124	
1703	006664	052520	044123	042105	
1704	006672	044440	020116	043115	
1705	006700	020120	047111	052123	
1706	006706	052522	052103	047511	
1707	006714	000116			
1708	006716	051127	047117	020107	EM66: .ASCIZ ?WRONG DATA FETCHED BY MFP INSTRUCTION?
1709	006724	040504	040524	043040	
1710	006732	052105	044103	042105	
1711	006740	041040	020131	043115	

K03

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 36
 DOKTAA.P11 07-FEB-77 10:30 ERROR TABLE MESSAGES AND DATA POINTERS

1712	006746	020120	047111	052123	
1713	006754	052522	052103	047511	
1714	006762	000116			
1715	006764	051124	042511	020104	EM67: .ASCIZ 'TRIED TO REFERENCE NON-RESIDENT PAGE'
1716	006772	047524	051040	043105	
1717	007000	051105	047105	042503	
1718	007006	047040	047117	051055	
1719	007014	051505	042111	047105	
1720	007022	020124	040520	042507	
1721	007030	000			
1722	007031	123	040524	045503	EM70: .ASCIZ 'STACK POINTER NOT CHANGED BY MTP INSTRUCTION'
1723	007036	050040	044517	052116	
1724	007044	051105	047040	052117	
1725	007052	041440	040510	043516	
1726	007060	042105	041040	020131	
1727	007066	052115	020120	047111	
1728	007074	052123	052522	052103	
1729	007102	047511	000116		
1730	007106	047111	047503	051122	EM71: .ASCIZ 'INCORRECT STORE BY MTP INSTRUCTION'
1731	007114	041505	020124	052123	
1732	007122	051117	020105	054502	
1733	007130	046440	050124	044440	
1734	007136	051516	051124	041525	
1735	007144	044524	047117	000	
1736	007151	101	047502	052122	EM72: .ASCIZ 'ABORTED THRU RIGHT VECTOR BUT PICKED UP WRONG PSW'
1737	007156	042105	052040	051110	
1738	007164	020125	044522	044107	
1739	007172	020124	042526	052103	
1740	007200	051117	041040	052125	
1741	007206	050040	041511	042513	
1742	007214	020104	050125	053440	
1743	007222	047522	043516	050040	
1744	007230	053523	000		
1745	007233	101	047502	052122	EM73: .ASCIZ 'ABORTED THRU USER SPACE, USER PSW IS XXX000'
1746	007240	042105	052040	051110	
1747	007246	020125	051525	051105	
1748	007254	051440	040520	042503	
1749	007262	020054	051525	051105	
1750	007270	050040	053523	044440	
1751	007276	020123	054130	030130	
1752	007304	030060	000		
1753	007307	104	042111	047040	EM74: .ASCIZ 'DID NOT ABORT REFERENCE TO A MEMORY MANAGEMENT REGISTER'
1754	007314	052117	040440	047502	
1755	007322	052122	051040	043105	
1756	007330	051105	047103	042503	
1757	007336	052040	020117	020101	
1758	007344	042515	047515	054522	
1759	007352	046440	047101	043501	
1760	007360	046505	047105	020124	
1761	007366	042522	044507	052123	
1762	007374	051105	000		
1763	007377	103	052520	042440	EM75: .ASCIZ 'CPU ERROR REG. DID NOT REPORT YELLOW ZONE'
1764	007404	051122	051117	051040	
1765	007412	043505	020056	044504	
1766	007420	020104	047516	020124	
1767	007426	042522	047520	052122	

L03

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 37
DOKTAA.P11 07-FEB-77 10:30 ERROR TABLE MESSAGES AND DATA POINTERS

1768	007434	054440	046105	047514	
1769	007442	020127	047532	042516	
1770	007450	000			
1771	007451	103	052520	042440	EM76: .ASCIZ 'CPU ERROR REG. DID NOT REPORT RED ZONE'
1772	007456	051122	051117	051040	
1773	007464	043505	020056	044504	
1774	007472	020104	047516	020124	
1775	007500	042522	047520	052122	
1776	007506	051040	042105	055040	
1777	007514	047117	000105		
1778					
1779	007520	044124	020105	047506	EM201: .ASCIZ 'THE FOLLOWING ARE PAR/PDR REFERENCE TIMEOUTS'
1780	007526	046114	053517	047111	
1781	007534	020107	051101	020105	
1782	007542	040520	027522	042120	
1783	007550	020122	042522	042506	
1784	007556	042522	041516	020105	
1785	007564	044524	042515	052517	
1786	007572	051524	000		
1787	007575	124	042510	043040	EM202: .ASCIZ 'THE FOLLOWING ARE DUAL ADDRESSING ERRORS FOR PAR'S/PDR'S'
1788	007602	046117	047514	044527	
1789	007610	043516	040440	042522	
1790	007616	042040	040525	020114	
1791	007624	042101	051104	051505	
1792	007632	044523	043516	042440	
1793	007640	051122	051117	020123	
1794	007646	047506	020122	040520	
1795	007654	023522	027523	042120	
1796	007662	023522	000123		
1797	007666	044124	020105	047506	EM203: .ASCIZ 'THE FOLLOWING ARE COUNT PATTERN ERRORS FOR PAR'S/PDR'S'
1798	007674	046114	053517	047111	
1799	007702	020107	051101	020105	
1800	007710	047503	047125	020124	
1801	007716	040520	052124	051105	
1802	007724	020116	051105	047522	
1803	007732	051522	043040	051117	
1804	007740	050040	051101	051447	
1805	007746	050057	051104	051447	
1806	007754	000			
1807					
1808					
1809	007755	124	051505	047124	DH1: .ASCII 'TESTNO PC AT ABORT'
1810	007762	020117	050040	020103	
1811	007770	052101	040440	047502	
1812	007776	052122			
1813	010000	050103	020125	051105	DH2: .ASCIZ 'CPU ERR TESTNO PC AT ABORT'
1814	010006	020122	042524	052123	
1815	010014	047516	020040	041520	
1816	010022	040440	020124	041101	
1817	010030	051117	000124		
1818	010034	042515	020115	051105	DH3: .ASCII 'MEM ERR CONTROL HITMISS '<CRLF>'
1819	010042	020122	047503	052116	
1820	010050	047522	020114	044510	
1821	010056	046524	051511	020123	
1822	010064	100040			
1823	010066	042522	044507	052123	.ASCIZ 'REGISTR REGISTR REGISTR TESTNO PC AT ABORT'

M03

MAINDEC-11-DOKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 38
 DOKTAA.P11 07-FEB-77 10:30 ERROR TABLE MESSAGES AND DATA POINTERS

1824	010074	020122	042522	044507	
1825	010102	052123	020122	042522	
1826	010110	044507	052123	020122	
1827	010116	042524	052123	047516	
1828	010124	020040	041520	040440	
1829	010132	020124	041101	051117	
1830	010140	000124			
1831	010142	051105	047522	020122	DH5: .ASCII ?ERROR VIRTUAL?(CRLF)
1832	010150	020040	044526	052122	
1833	010156	040525	100114		
1834	010162	042522	044507	052123	.ASCIZ ?REGISTR ADDRESS TESTNO PC AT ABORT?
1835	010170	020122	042101	051104	
1836	010176	051505	020123	042524	
1837	010204	052123	047516	020040	
1838	010212	041520	040440	020124	
1839	010220	041101	051117	000124	
1840	010226	054105	042520	052103	DH6: .ASCII ?EXPECTED ERROR VIRTUAL?(CRLF)
1841	010234	020104	051105	047522	
1842	010242	020122	020040	044526	
1843	010250	052122	040525	100114	
1844	010256	047503	042116	052111	.ASCIZ ?CONDITN REGISTR ADDRESS TESTNO PC AT ABORT?
1845	010264	020116	042522	044507	
1846	010272	052123	020122	042101	
1847	010300	051104	051505	020123	
1848	010306	042524	052123	047516	
1849	010314	020040	041520	040440	
1850	010322	020124	041101	051117	
1851	010330	000124			
1852	010332	046450	051115	024460	DH7: .ASCIZ ?(MMR0) TESTNO ERRORPC?
1853	010340	020040	042524	052123	
1854	010346	047516	020040	051105	
1855	010354	047522	050122	000103	
1856	010362	047514	042101	042105	DH11: .ASCIZ ?LOADED RECEIVD TESTNO ERRORPC?
1857	010370	020040	042522	042503	
1858	010376	053111	020104	042524	
1859	010404	052123	047516	020040	
1860	010412	051105	047522	050122	
1861	010420	000103			
1862	010422	054105	042520	052103	DH12: .ASCIZ ?EXPECTED (MMR2) TESTNO ERRORPC?
1863	010430	020104	046450	051115	
1864	010436	024462	020040	042524	
1865	010444	052123	047516	020040	
1866	010452	051105	047522	050122	
1867	010460	000103			
1868	010462	042101	051104	051117	DH13: .ASCIZ ?ADDROR ADDRAND TESTNO ERRORPC #ERRORS?
1869	010470	020040	042101	051104	
1870	010476	047101	020104	042524	
1871	010504	052123	047516	020040	
1872	010512	051105	047522	050122	
1873	010520	020103	021440	051105	
1874	010526	047522	051522	000	
1875	010533	101	042104	042522	DH14: .ASCIZ ?ADDRESS DATA TESTNO ERRORPC?
1876	010540	051523	042040	052101	
1877	010546	020101	020040	052040	
1878	010554	051505	047124	020117	
1879	010562	042440	051122	051117	

N03

MAINDEC-11-DQKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 39
DQKTA.P11 07-FEB-77 10:30 ERROR TABLE MESSAGES AND DATA POINTERS

1880	010570	041520	000		
1881	010573	101	042104	047522	DH15: .ASCII 'ADDROR ADDRAND ADDROR ADDRAND'<CRLF>
1882	010600	020122	040440	042104	
1883	010606	040522	042116	040440	
1884	010614	042104	047522	020122	
1885	010622	040440	042104	040522	
1886	010630	042116	200		
1887	010633	114	040517	042504	.ASCIZ 'LOADED LOADED ENABLED ENABLED TESTNO ERRORPC #ERRORS'
1888	010640	020104	046040	040517	
1889	010646	042504	020104	042440	
1890	010654	040516	046102	042105	
1891	010662	042440	040516	046102	
1892	010670	042105	052040	051505	
1893	010676	047124	020117	051105	
1894	010704	047522	050122	020103	
1895	010712	042443	051122	051117	
1896	010720	000123			
1897	010722	042101	051104	051117	DH16: .ASCIZ 'ADDROR ADDRAND PATRNOR PATRNAD DATAOR DATAND TESTNO ERRORPC #ERRORS'
1898	010730	020040	042101	051104	
1899	010736	047101	020104	040520	
1900	010744	051124	047516	020122	
1901	010752	040520	051124	040516	
1902	010760	020104	040504	040524	
1903	010766	051117	020040	040504	
1904	010774	040524	042116	020040	
1905	011002	042524	052123	047516	
1906	011010	042440	051122	051117	
1907	011016	041520	021440	051105	
1908	011024	047522	051522	000	
1909	011031	101	042104	042522	DH17: .ASCIZ 'ADDRESS EXPECTD RECEIVD TESTNO ERRORPC'
1910	011036	051523	042440	050130	
1911	011044	041505	042124	051040	
1912	011052	041505	044505	042126	
1913	011060	052040	051505	047124	
1914	011066	020117	042440	051122	
1915	011074	051117	041520	000	
1916	011101	122	043505	042101	DH20: .ASCIZ 'REGADDR TESTNO ERRORPC'
1917	011106	051104	052040	051505	
1918	011114	047124	020117	042440	
1919	011122	051122	051117	041520	
1920	011130	000			
1921	011131	122	043505	042101	DH21: .ASCIZ 'REGADDR DATA TESTNO ERRORPC'
1922	011136	051104	042040	052101	
1923	011144	020101	020040	052040	
1924	011152	051505	047124	020117	
1925	011160	042440	051122	051117	
1926	011166	041520	000		
1927	011171	122	043505	042101	DH22: .ASCIZ 'REGADR EXPECTD RECEIVD TESTNO ERRORPC'
1928	011176	020122	042440	050130	
1929	011204	041505	042124	051040	
1930	011212	041505	044505	042126	
1931	011220	052040	051505	047124	
1932	011226	020117	051105	047522	
1933	011234	050122	000103		
1934	011240	042522	040507	042104	DH23: .ASCIZ 'REGADDR DATAREC DATAEXP TESTNO ERRORPC'
1935	011246	020122	040504	040524	

B04

MAINDEC-11-DOKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 40
 DOKTAA.P11 07-FEB-77 10:30 ERROR TABLE MESSAGES AND DATA POINTERS

1936	011254	042522	020103	040504					
1937	011262	040524	054105	020120					
1938	011270	042524	052123	047516					
1939	011276	020040	051105	047522					
1940	011304	050122	000103						
1941	011310	051513	020120	020040	DH27:	.ASCIZ	'KSP	TESTNO	ERRORPC?
1942	011316	020040	042524	052123					
1943	011324	047516	020040	051105					
1944	011332	047522	050122	000103					
1945	011340	047111	042504	020130	DH30:	.ASCII	'INDEX	INDEX	PAR/PDR?'(CRLF)
1946	011346	020040	047111	042504					
1947	011354	020130	020040	040520					
1948	011362	027522	042120	100122					
1949	011370	054105	042520	052103		.ASCIZ	'EXPECTED	RECEIVD	ADRREAD TESTNO ERRORPC?
1950	011376	020104	042522	042503					
1951	011404	053111	020104	042101					
1952	011412	051122	040505	020104					
1953	011420	042524	052123	047516					
1954	011426	020040	051105	047522					
1955	011434	050122	000103						
1956	011440	044526	052122	040525	DH31:	.ASCII	'VIRTUAL	PHYSICAL DATA	DATA?'(CRLF)
1957	011446	020114	020040	020040					
1958	011454	020040	020040	044120					
1959	011462	051531	040503	020114					
1960	011470	040504	040524	020040					
1961	011476	020040	040504	040524					
1962	011504	230							
1963	011505	101	042104	042522		.ASCIZ	'ADDRESS	KIPAR4	ADDRESS EXPECTD RECEIVD TESTNO ERRORPC?
1964	011512	051523	045440	050111					
1965	011520	051101	020064	040440					
1966	011526	042104	042522	051523					
1967	011534	042440	050130	041505					
1968	011542	042124	051040	041505					
1969	011550	044505	042126	052040					
1970	011556	051505	047124	020117					
1971	011564	051105	047522	050122					
1972	011572	000103							
1973	011574	052123	052122	042101	DH32:	.ASCIZ	'STRTADR	FNSHADR	TESTNO ERRORPC?
1974	011602	020122	047106	044123					
1975	011610	042101	020122	042524					
1976	011616	052123	047516	020040					
1977	011624	051105	047522	050122					
1978	011632	000103							
1979	011634	040520	052124	051105	DH33:	.ASCII	'PATTERN	DATA	ADDRESS?'(CRLF)
1980	011642	020116	040504	040524					
1981	011650	020040	020040	042101					
1982	011656	051104	051505	100123					
1983	011664	047514	042101	042105		.ASCIZ	'LOADED	FETCHED	INTENDD KIPAR4 KIPAR5 TESTNO ERRORPC?
1984	011672	020040	042506	041524					
1985	011700	042510	020104	047111					
1986	011706	042524	042116	020104					
1987	011714	044513	040520	032122					
1988	011722	020040	044513	040520					
1989	011730	032522	020040	042524					
1990	011736	052123	047516	020040					
1991	011744	051105	047522	050122					

MAINDEC-11-DQKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 41
DQKTA.P11 07-FEB-77 10:30 ERROR TABLE MESSAGES AND DATA POINTERS

Year	MMR1	MMR2	MMR3	MMR4	MMR5	MMR6	MMR7	MMR8	MMR9	MMR10	MMR11	MMR12	MMR13	MMR14	MMR15	MMR16	MMR17	MMR18	MMR19	MMR20	MMR21	MMR22	MMR23	MMR24	MMR25	MMR26	MMR27	MMR28	MMR29	MMR30	MMR31	MMR32	MMR33	MMR34	MMR35	MMR36	MMR37	MMR38	MMR39	MMR40	MMR41	MMR42	MMR43	MMR44	MMR45	MMR46	MMR47	MMR48	MMR49	MMR50	MMR51	MMR52	MMR53	MMR54	MMR55	MMR56	MMR57	MMR58	MMR59	MMR60	MMR61	MMR62	MMR63	MMR64	MMR65	MMR66	MMR67	MMR68	MMR69	MMR70	MMR71	MMR72	MMR73	MMR74	MMR75	MMR76	MMR77	MMR78	MMR79	MMR80	MMR81	MMR82	MMR83	MMR84	MMR85	MMR86	MMR87	MMR88	MMR89	MMR90	MMR91	MMR92	MMR93	MMR94	MMR95	MMR96	MMR97	MMR98	MMR99	MMR100	MMR101	MMR102	MMR103	MMR104	MMR105	MMR106	MMR107	MMR108	MMR109	MMR110	MMR111	MMR112	MMR113	MMR114	MMR115	MMR116	MMR117	MMR118	MMR119	MMR120	MMR121	MMR122	MMR123	MMR124	MMR125	MMR126	MMR127	MMR128	MMR129	MMR130	MMR131	MMR132	MMR133	MMR134	MMR135	MMR136	MMR137	MMR138	MMR139	MMR140	MMR141	MMR142	MMR143	MMR144	MMR145	MMR146	MMR147	MMR148	MMR149	MMR150	MMR151	MMR152	MMR153	MMR154	MMR155	MMR156	MMR157	MMR158	MMR159	MMR160	MMR161	MMR162	MMR163	MMR164	MMR165	MMR166	MMR167	MMR168	MMR169	MMR170	MMR171	MMR172	MMR173	MMR174	MMR175	MMR176	MMR177	MMR178	MMR179	MMR180	MMR181	MMR182	MMR183	MMR184	MMR185	MMR186	MMR187	MMR188	MMR189	MMR190	MMR191	MMR192	MMR193	MMR194	MMR195	MMR196	MMR197	MMR198	MMR199	MMR200	MMR201	MMR202	MMR203	MMR204	MMR205	MMR206	MMR207	MMR208	MMR209	MMR210	MMR211	MMR212	MMR213	MMR214	MMR215	MMR216	MMR217	MMR218	MMR219	MMR220	MMR221	MMR222	MMR223	MMR224	MMR225	MMR226	MMR227	MMR228	MMR229	MMR230	MMR231	MMR232	MMR233	MMR234	MMR235	MMR236	MMR237	MMR238	MMR239	MMR240	MMR241	MMR242	MMR243	MMR244	MMR245	MMR246	MMR247	MMR248	MMR249	MMR250	MMR251	MMR252	MMR253	MMR254	MMR255	MMR256	MMR257	MMR258	MMR259	MMR260	MMR261	MMR262	MMR263	MMR264	MMR265	MMR266	MMR267	MMR268	MMR269	MMR270	MMR271	MMR272	MMR273	MMR274	MMR275	MMR276	MMR277	MMR278	MMR279	MMR280	MMR281	MMR282	MMR283	MMR284	MMR285	MMR286	MMR287	MMR288	MMR289	MMR290	MMR291	MMR292	MMR293	MMR294	MMR295	MMR296	MMR297	MMR298	MMR299	MMR300	MMR301	MMR302	MMR303	MMR304	MMR305	MMR306	MMR307	MMR308	MMR309	MMR310	MMR311	MMR312	MMR313	MMR314	MMR315	MMR316	MMR317	MMR318	MMR319	MMR320	MMR321	MMR322	MMR323	MMR324	MMR325	MMR326	MMR327	MMR328	MMR329	MMR330	MMR331	MMR332	MMR333	MMR334	MMR335	MMR336	MMR337	MMR338	MMR339	MMR340	MMR341	MMR342	MMR343	MMR344	MMR345	MMR346	MMR347	MMR348	MMR349	MMR350	MMR351	MMR352	MMR353	MMR354	MMR355	MMR356	MMR357	MMR358	MMR359	MMR360	MMR361	MMR362	MMR363	MMR364	MMR365	MMR366	MMR367	MMR368	MMR369	MMR370	MMR371	MMR372	MMR373	MMR374	MMR375	MMR376	MMR377	MMR378	MMR379	MMR380	MMR381	MMR382	MMR383	MMR384	MMR385	MMR386	MMR387	MMR388	MMR389	MMR390	MMR391	MMR392	MMR393	MMR394	MMR395	MMR396	MMR397	MMR398	MMR399	MMR400	MMR401	MMR402	MMR403	MMR404	MMR405	MMR406	MMR407	MMR408	MMR409	MMR410	MMR411	MMR412	MMR413	MMR414	MMR415	MMR416	MMR417	MMR418	MMR41
------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	-------

004

MAINDEC-11-DOKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 42
 DOKTAA.P11 07-FEB-77 10:30 ERROR TABLE MESSAGES AND DATA POINTERS

2048	012434	020051	024040	046515					
2049	012442	031122	020051	052040					
2050	012450	051505	047124	020117					
2051	012456	042440	051122	051117					
2052	012464	041520	000						
2053	012467	117	044522	044507	DH50:	.ASCII	?ORIGINAL DATA	NEW DATA?<CRLF>	
2054	012474	040516	020114	040504					
2055	012502	040524	004411	047040					
2056	012510	053505	042040	052101					
2057	012516	100101							
2058	012520	046450	051115	024460		.ASCIZ	?(MMR0) (MMR2) (MMR0) (MMR2)	TESTNO ERRORPC?	
2059	012526	020040	046450	051115					
2060	012534	024462	020040	046450					
2061	012542	051115	024460	020040					
2062	012550	046450	051115	024462					
2063	012556	020040	042524	052123					
2064	012564	047516	020040	051105					
2065	012572	047522	050122	000103					
2066	012600	042522	042503	053111	DH52:	.ASCIZ	?RECEIVD (MMR2)	TESTNO ERRORPC?	
2067	012606	020104	046450	051115					
2068	012614	024462	020040	042524					
2069	012622	052123	047516	027040					
2070	012630	051105	047522	050122					
2071	012636	000103							
2072	012640	044526	052122	042101	DH53:	.ASCIZ	?VIRTADR (MMR2)	TESTNO ERRORPC?	
2073	012646	020122	046450	051115					
2074	012654	024462	020040	042524					
2075	012662	052123	047516	020040					
2076	012670	051105	047522	050122					
2077	012676	000103							
2078	012700	054105	042520	052103	DH54:	.ASCIZ	?EXPECTED (MMR0)	TESTNO ERRORPC?	
2079	012706	020104	046450	051115					
2080	012714	024460	020040	042524					
2081	012722	052123	047516	020040					
2082	012730	051105	047522	050122					
2083	012736	000103							
2084	012740	044513	042120	032122	DH55:	.ASCIZ	?KIPDR4	TESTNO ERRORPC?	
2085	012746	020040	042524	052123					
2086	012754	047516	020040	051105					
2087	012762	047522	050122	000103					
2088	012770	044513	042120	033522	DH60:	.ASCIZ	?KIPDR7	TESTNO ERRORPC?	
2089	012776	020040	042524	052123					
2090	013004	047516	020040	051105					
2091	013012	047522	050122	000103					
2092	013020	042107	040520	042507	DH62:	.ASCIZ	?GDPAGE BDPAGE	TESTNO ERRORPC?	
2093	013026	020040	042102	040520					
2094	013034	042507	020040	042524					
2095	013042	052123	047516	020040					
2096	013050	051105	047522	050122					
2097	013056	000103							
2098	013060	051524	050124	043501	DH63:	.ASCIZ	?TSTPAGE CONTENT	TESTNO ERRORPC?	
2099	013066	020105	047503	052116					
2100	013074	047105	020124	042524					
2101	013102	052123	047516	020040					
2102	013110	051105	047522	050122					
2103	013116	000103							

E04

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 43
 DOKTAA.P11 07-FEB-77 10:30 ERROR TABLE MESSAGES AND DATA POINTERS

2104	013120	054105	042520	052103	DH64:	.ASCIZ	'EXPECTED RECEIVD TESTNO	ERRORPC'
2105	013126	020104	042522	042503				
2106	013134	053111	020104	042524				
2107	013142	052123	047516	020040				
2108	013150	051105	047522	050122				
2109	013156	000103						
2110	013160	054105	042520	052103	DH66:	.ASCIZ	'EXPECTED RECEIVD TESTNO	ERRORPC'
2111	013166	020104	042522	042503				
2112	013174	053111	020104	042524				
2113	013202	052123	047516	020040				
2114	013210	051105	047522	050122				
2115	013216	000103						
2116	013220	046450	051115	024460	DH67:	.ASCIZ	'(MMRO) (MMR2) TESTNO	ERRORPC'
2117	013226	020040	046450	051115				
2118	013234	024462	020040	042524				
2119	013242	052123	047516	020040				
2120	013250	051105	047522	050122				
2121	013256	000103						
2122	013260	052123	050113	051124	DH70:	.ASCIZ	'STKPTR TESTNO	ERRORPC'
2123	013266	020040	042524	052123				
2124	013274	047516	020040	051105				
2125	013302	047522	050122	000103				
2126	013310	042107	040504	040524	DH71:	.ASCIZ	'GDDATA STORED TESTNO	ERRORPC'
2127	013316	020040	052123	051117				
2128	013324	042105	020040	042524				
2129	013332	052123	047516	020040				
2130	013340	051105	047522	050122				
2131	013346	000103						
2132	013350	050050	053523	020051	DH72:	.ASCIZ	'(PSW) TESTNO ERRORPC	EXPECTING XXX340'
2133	013356	020040	042524	052123				
2134	013364	047516	020040	051105				
2135	013372	047522	050122	020103				
2136	013400	054105	042520	052103				
2137	013406	047111	020107	054130				
2138	013414	031530	030064	000				
2139	013421	050	051520	024527	DH73:	.ASCIZ	'(PSW) TESTNO	ERRORPC'
2140	013426	020040	052040	051505				
2141	013434	047124	020117	042440				
2142	013442	051122	051117	041520				
2143	013450	000						
2144	013451	124	051505	047124	DH74:	.ASCIZ	'TESTNO ERRORPC'	
2145	013456	020117	051105	047522				
2146	013464	050122	000103					
2147	013470	052123	050113	051124	DH75:	.ASCIZ	'STKPTR STKLMT CPUERR	TESTNO ERRORPC'
2148	013476	020040	052123	046113				
2149	013504	052115	020040	050103				
2150	013512	042525	051122	020040				
2151	013520	042524	052123	047516				
2152	013526	020040	051105	047522				
2153	013534	050122	000103					
2154								
2155								
2156	013540	042101	051104	051505	DH201:	.ASCIZ	'ADDRESS TESTNO	ERRORPC'
2157	013546	020123	042524	052123				
2158	013554	047516	042440	051122				
2159	013562	051117	041520	000				

F04

MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 44
 DQKTA.P11 07-FEB-77 10:30 ERROR TABLE MESSAGES AND DATA POINTERS

2160	013567	101	042104	042522	DH202:	.ASCII	'ADDRESS ADDRESS'<CRLF>
2161	013574	051523	040440	042104			
2162	013602	042522	051523	200			
2163	013607	114	040517	042504		.ASCIZ	'LOADED JUSTREAD TESTNO ERRORPC'
2164	013614	020104	045040	051525			
2165	013622	051124	040505	020104			
2166	013630	042524	052123	047516			
2167	013636	042440	051122	051117			
2168	013644	041520	000				
2169	013647	101	042104	042522	DH203:	.ASCIZ	'ADDRESS RECEIVD EXPECTD COUNT TESTNO ERRORPC'
2170	013654	051523	051040	041505			
2171	013662	044505	042126	042440			
2172	013670	050130	041505	042124			
2173	013676	041440	052517	052116			
2174	013704	020040	052040	051505			
2175	013712	047124	020117	051105			
2176	013720	047522	050122	000103			
2177						.EVEN	
2178							
2179							
2180	013726	001414	001412	000000	DT1:	.WORD	TESTNO,BADPC,0
2181	013734	001410	001414	001412	DT2:	.WORD	PCPUER,TESTNO,BADPC,0
2182	013742	000000					
2183	013744	001376	001400	001402	DT3:	.WORD	PPARER,PCONTR,PHITMI,TESTNO,BADPC,0
2184	013752	001414	001412	000000			
2185	013760	001404	001406	001414	DT5:	.WORD	PMMRO,PMMR2,TESTNO,BADPC,0
2186	013766	001412	000000				
2187	013772	001366	001404	001406	DT6:	.WORD	MMEXP,PMMRO,PMMR2,TESTNO,BADPC,0
2188	014000	001414	001412	000000			
2189	014006	001164	001414	001116	DT7:	.WORD	\$REG1,TESTNO,\$ERRPC,0
2190	014014	000000					
2191	014016	001166	001164	001414	DT11:	.WORD	\$REG2,\$REG1,TESTNO,\$ERRPC,0
2192	014024	001116	000000				
2193	014030	001170	001166	001414	DT12:	.WORD	\$REG3,\$REG2,TESTNO,\$ERRPC,0
2194	014036	001116	000000				
2195	014042	001426	001430	001414	DT13:	.WORD	ADDROR,ADRAND,TESTNO,\$ERRPC,\$TMP5,0
2196	014050	001116	001210	000000			
2197	014056	001162	001166	001414	DT14:	.WORD	\$REG0,\$REG2,TESTNO,\$ERRPC,0
2198	014064	001116	000000				
2199	014070	001426	001430	001416	DT15:	.WORD	ADDROR,ADRAND,DATAOR,DATAND,TESTNO,\$ERRPC,\$TMP5,0
2200	014076	001420	001414	001116			
2201	014104	001210	000000				
2202	014110	001426	001430	001422	DT16:	.WORD	ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,TESTNO,\$ERRPC,\$TMP5,0
2203	014116	001424	001416	001420			
2204	014124	001414	001116	001210			
2205	014132	000000					
2206	014134	001162	001166	001164	DT17:	.WORD	\$REG0,\$REG2,\$REG1,TESTNO,\$ERRPC,0
2207	014142	001414	001116	000000			
2208	014150	001162	001414	001116	DT20:	.WORD	\$REG0,TESTNO,\$ERRPC,0
2209	014156	000000					
2210	014160	001162	001164	001414	DT21:	.WORD	\$REG0,\$REG1,TESTNO,\$ERRPC,0
2211	014166	001116	000000				
2212	014172	001206	001210	001162	DT22:	.WORD	\$TMP4,\$TMP5,\$REG0,TESTNO,\$ERRPC,0
2213	014200	001414	001116	000000			
2214	014206	001162	001164	001166	DT23:	.WORD	\$REG0,\$REG1,\$REG2,TESTNO,\$ERRPC,0
2215	014214	001414	001116	000000			

G04

MAINDEC-11-DOKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 45
 DOKTAA.P11 07-FEB-77 10:30 ERROR TABLE MESSAGES AND DATA POINTERS

2216	014222	001162	001414	001116	DT27:	.WORD	\$REG0, TESTNO, \$ERRPC, 0
2217	014230	000000					
2218	014232	001162	001164	001166	DT30:	.WORD	\$REG0, \$REG1, \$REG2, TESTNO, \$ERRPC, 0
2219	014240	001414	001116	000000			
2220	014246	001162	001200	001202	DT31:	.WORD	\$REG0, \$TMP1, \$TMP2, \$REG1, \$REG2, TESTNO, \$ERRPC, 0
2221	014254	001164	001166	001414			
2222	014262	001116	000000				
2223	014266	001200	001202	001414	DT32:	.WORD	\$TMP1, \$TMP2, TESTNO, \$ERRPC, 0
2224	014274	001116	000000				
2225	014300	001166	001170	001162	DT33:	.WORD	\$REG2, \$REG3, \$REG0, \$TMP3, \$TMP4, TESTNO, \$ERRPC, 0
2226	014306	001204	001206	001414			
2227	014314	001116	000000				
2228	014320	001164	001414	001116	DT34:	.WORD	\$REG1, TESTNO, \$ERRPC, 0
2229	014326	000000					
2230	014330	001164	001170	001414	DT35:	.WORD	\$REG1, \$REG3, TESTNO, \$ERRPC, 0
2231	014336	001116	000000				
2232	014342	001176	001200	001202	DT37:	.WORD	\$TMP0, \$TMP1, \$TMP2, TESTNO, \$ERRPC, 0
2233	014350	001414	001116	000000			
2234	014356	001166	001170	001414	DT41:	.WORD	\$REG2, \$REG3, TESTNO, \$ERRPC, 0
2235	014364	001116	000000				
2236	014370	001166	001170	001164	DT47:	.WORD	\$REG2, \$REG3, \$REG1, \$PMR0, \$PMR2, TESTNO, \$ERRPC, 0
2237	014376	001404	001406	001414			
2238	014404	001116	000000				
2239	014410	001404	001406	001176	DT50:	.WORD	\$PMR0, \$PMR2, \$TMP0, \$TMP2, TESTNO, \$ERRPC, 0
2240	014416	001202	001414	001116			
2241	014424	000000					
2242	014426	001404	001406	001414	DT52:	.WORD	\$PMR0, \$PMR2, TESTNO, \$ERRPC, 0
2243	014434	001116	000000				
2244	014440	001164	001406	001414	DT53:	.WORD	\$REG1, \$PMR2, TESTNO, \$ERRPC, 0
2245	014446	001116	000000				
2246	014452	001166	001404	001414	DT54:	.WORD	\$REG2, \$PMR0, TESTNO, \$ERRPC, 0
2247	014460	001116	000000				
2248	014464	001176	001414	001116	DT55:	.WORD	\$TMP0, TESTNO, \$ERRPC, 0
2249	014472	000000					
2250	014474	001170	001164	001414	DT62:	.WORD	\$REG3, \$REG1, TESTNO, \$ERRPC, 0
2251	014502	001116	000000				
2252	014506	001170	001162	001414	DT63:	.WORD	\$REG3, \$REG0, TESTNO, \$ERRPC, 0
2253	014514	001116	000000				
2254	014520	001166	001164	001414	DT64:	.WORD	\$REG2, \$REG1, TESTNO, \$ERRPC, 0
2255	014526	043300	000000				
2256	014532	001162	001164	001414	DT66:	.WORD	\$REG0, \$REG1, TESTNO, \$ERRPC, 0
2257	014540	001116	000000				
2258	014544	001404	001406	001414	DT67:	.WORD	\$PMR0, \$PMR2, TESTNO, \$ERRPC, 0
2259	014552	001116	000000				
2260	014556	001164	001414	001116	DT70:	.WORD	\$REG1, TESTNO, \$ERRPC, 0
2261	014564	000000					
2262	014566	001162	001414	001116	DT72:	.WORD	\$REG0, TESTNO, \$ERRPC, 0
2263	014574	000000					
2264	014576	001414	001116	000000	DT74:	.WORD	TESTNO, \$ERRPC, 0
2265	014604	001166	001170	001410	DT75:	.WORD	\$REG2, \$REG3, \$PCUER, TESTNO, \$ERRPC, 0
2266	014612	001414	001116	000000			
2267							
2268	014620	001162	001414	001116	DT201:	.WORD	\$REG0, TESTNO, \$ERRPC, 0
2269	014626	000000					
2270	014630	001162	001164	001414	DT202:	.WORD	\$REG0, \$REG1, TESTNO, \$ERRPC, 0
2271	014636	001116	000000				

H04

MAINDEC-11-DOKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 46
 DOKTAA.P11 07-FEB-77 10:30 ERROR TABLE MESSAGES AND DATA POINTERS

Address	Value 1	Value 2	Value 3	Value 4	Label	Value 5	Value 6
2272	014642	001162	001166	001172	DT203:	.WORD	\$REG0, \$REG2, \$REG4, \$REG1, *ESTNO, \$ERRPC, J
2273	014650	001164	001414	001116			
2274	014656	000000					
2275							
2276							
2277	014660	000	000		DF1:	.BYTE	0,0
2278	014662	000	000	000	DF2:	.BYTE	0,0,0
2279	014665	000	000	000	DF3:	.BYTE	0,0,0,0,0
2280	014670	000	000				
2281	014672	000	000	000	DF5:	.BYTE	0,0,0,0
2282	014675	000					
2283	014676	000	000	000	DF6:	.BYTE	0,0,0,0,0
2284	014701	000	000				
2285	014703	000	000	000	DF7:	.BYTE	0,0,0
2286	014706	000	000	000	DF11:	.BYTE	0,0,0,0
2287	014711	000					
2288	014712	000	000	000	DF12:	.BYTE	0,0,0,0
2289	014715	000					
2290	014716	000	000	000	DF13:	.BYTE	0,0,0,0,1
2291	014721	000	001				
2292	014723	000	000	000	DF14:	.BYTE	0,0,0,0
2293	014726	000					
2294	014727	000	000	000	DF15:	.BYTE	0,0,0,0,0,0,1
2295	014732	000	000	000			
2296	014735	001					
2297	014736	000	000	000	DF16:	.BYTE	0,0,0,0,0,0,0,0,1
2298	014741	000	000	000			
2299	014744	000	000	001			
2300	014747	000	000	000	DF17:	.BYTE	0,0,0,0,0
2301	014752	000	000				
2302	014754	000	000	000	DF20:	.BYTE	0,0,0
2303	014757	000	000	000	DF21:	.BYTE	0,0,0,0
2304	014762	000					
2305	014763	000	000	000	DF22:	.BYTE	0,0,0,0,0
2306	014766	000	000				
2307	014770	000	000	000	DF23:	.BYTE	0,0,0,0,0
2308	014773	000	000				
2309	014775	000	000	000	DF27:	.BYTE	0,0,0
2310	015000	000	000	000	DF30:	.BYTE	0,0,0,0,0
2311	015003	000	000				
2312	015005	000	000	000	DF31:	.BYTE	0,0,0,0,0,0,0
2313	015010	000	000	000			
2314	015013	000					
2315	015014	002	002	000	DF32:	.BYTE	2,2,0,0
2316	015017	000					
2317	015020	000	000	000	DF33:	.BYTE	0,0,0,0,0,0,0
2318	015023	000	000	000			
2319	015026	000					
2320	015027	000	000	000	DF34:	.BYTE	0,0,0
2321	015032	000	000	000	DF35:	.BYTE	0,0,0,0
2322	015035	000					
2323	015036	000	000	000	DF37:	.BYTE	0,0,0,0,0
2324	015041	000	000				
2325	015043	000	000	000	DF41:	.BYTE	0,0,0,0
2326	015046	000					
2327	015047	000	000	000	DF47:	.BYTE	0,0,0,0,0,0,0

2328	015052	000	000	000			
2329	015055	000					
2330	015056	000	000	000	DF50:	.BYTE	0,0,0,0,0,0
2331	015061	000	000	000			
2332	015064	000	000	000	DF52:	.BYTE	0,0,0,0
2333	015067	000					
2334	015070	000	000	000	DF53:	.BYTE	0,0,0,0
2335	015073	000					
2336	015074	000	000	000	DF54:	.BYTE	0,0,0,0
2337	015077	000					
2338	015100	000	000	000	DF55:	.BYTE	0,0,0
2339	015103	000	000	000	DF62:	.BYTE	0,0,0,0
2340	015106	000					
2341	015107	000	000	000	DF63:	.BYTE	0,0,0,0
2342	015112	000					
2343	015113	000	000	000	DF64:	.BYTE	0,0,0,0
2344	015116	000					
2345	015117	000	000	000	DF66:	.BYTE	0,0,0,0
2346	015122	000					
2347	015123	000	000	000	DF67:	.BYTE	0,0,0,0
2348	015126	000					
2349	015127	000	000	000	DF70:	.BYTE	0,0,0
2350	015132	000	000	000	DF72:	.BYTE	0,0,0
2351	015135	000	000	000	DF74:	.BYTE	0,0
2352	015137	000	000	000	DF75:	.BYTE	0,0,0,0,0
2353	015142	000	000				
2354							
2355							
2356							
2357	015144	000	000	000	DF201:	.BYTE	0,0,0
2358	015147	000	000	000	DF202:	.BYTE	0,0,0,0
2359	015152	000					
2360	015153	000	000	000	DF203:	.BYTE	0,0,0,0,0,0
2361	015156	000	000	000			
2362		015162				.EVEN	
2363							
2364							
2365							
2366							
2367							

J04

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 48
 DOKTAA.P11 07-FEB-77 10:30 ERROR TABLE MESSAGES AND DATA POINTERS

```

2368
2369
2370 .SBTTL ***** TRAP HANDLING ROUTINES *****
2371
2372 .SBTTL CPU TRAP HANDLER ROUTINE
2373 *****
2374 *
2375 * THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS, THRU
2376 * "ERRVEC" (000004). IF THIS SUBROUTINE IS ENTERED BY A SECOND
2377 * TRAP BEFORE THE FIRST HAS BEEN PROCESSED A HALT IS EXECUTED.
2378 * THE CPU ERROR REGISTER IS READ IF EXECUTING ON AN 11/6X.
2379 * FOR EVERY CASE AN ERROR MESSAGE IS TYPED REPORTING AN
2380 * UNEXPECTED TRAP OR ABORT TO LOC 4. 'PCPUER' CAN
2381 * BE USED AS A FLAG TO INDICATE THAT A TRAP HAS OCCURRED SINCE IT
2382 * IS LOADED WITH THE ERROR REGISTER IF A TRAP VECTORS HERE
2383 *
2384 *****
2385 015162 005227 CPUER: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME
2386 015164 177777 CPFLAG: .WORD -1 ;NEGATIVE ONE FOR A FLAG
2387 015166 001401 BEQ 10$ ;BRANCH IF FIRST TIME IN
2388 015170 000000 HALT ;I HAVE ENTERED THIS ROUTINE BEFORE
2389 ;I FINISHED REPORTING THE FIRST ERROR
2390 ;THE SECOND ENTRY ADDRESS IS ON THE
2391 ;STACK AND THE FIRST ERROR CONDITION
2392 ;IS PROBABLY STILL LOCKED UP
2393 015172 012637 001436 10$: MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF LOOP
2394 015176 012637 001440 MOV (KSP)+,OLDPS ;SAVE OLD PSW IN CASE OF LOOP
2395 015202 013737 001436 001412 MOV OLDPC,BADPC ;SAVE PC+2 AT TIME OF ABORT
2396 015210 105737 001450 TSTB FORTY ;EXECUTING ON AN 11/40?
2397 015214 001402 BEQ 2$ ;BRANCH IF NO
2398 015216 104001 ERROR 1 ;UNEXPECTED CPU TRAP OR ABORT
2399 015220 000404 BR 1$ ;BRANCH TO EXIT
2400 015222 017737 164126 001410 2$: MOV @CPUERR,PCPUER ;SAVE CPU ERROR REGISTER ON AN 11/6X
2401 015230 104002 ERROR 2 ;UNEXPECTED CPU TRAP OR ABORT
2402 015232 012737 177777 015164 1$: MOV #-1,CPFLAG ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
2403 015240 013746 001440 MOV OLDPS,-(KSP) ;PUSH OLD PSW BACK ON STACK
2404 015244 013746 001436 MOV OLDPC,-(KSP) ;PUSH RETURN ADDRESS BACK ON STACK
2405 015250 000006 RTT ;RETURN FROM INTERRUPT OR ABORT
2406
2407
2408
2409 .SBTTL MEMORY/CACHE PARITY TRAPS AND ABORTS HANDLER ROUTINE
2410 *****
2411 *
2412 * THIS SUBROUTINE WILL HANDLE ALL UNEXPECTED PARITY ERRORS. IF
2413 * THE PARITY ERROR IS AN ABORT THE TEST THAT WAS RUNNING WILL
2414 * BE RESTARTED ONCE AFTER THE ABORT CONDITION IS REPORTED. ON THE
2415 * SECOND ABORT IN A SINGLE TEST THE NEXT TEST IS ATTEMPTED
2416 * AFTER THE ABORT IS REPORTED.
2417 *
2418 *****
2419 015252 005227 MEMER: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME
2420 015254 177777 PAFLAG: .WORD -1 ;NEGATIVE ONE FOR A FLAG
2421 015256 001401 BEQ 10$ ;BRANCH IF FIRST TIME IN
2422 015260 000000 HALT ;I HAVE ENTERED THIS ROUTINE BEFORE
2423 ;I FINISHED REPORTING THE FIRST ERROR

```

K04

MAINDEC-11-DOKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 49
 DOKTAA.P11 07-FEB-77 10:30 MEMORY/CACHE PARITY TRAPS AND ABORTS HANDLER ROUTINE

```

2424
2425
2426
2427 015262 012637 001436      10$: MOV      (KSP)+,OLDPC
2428 015266 012637 001440      MOV      (KSP)+,OLDPS
2429 015272 105737 001450      TSTB     FORTY
2430 015276 001026              BNE      6$
2431 015300 017737 164054 001376  MOV      @MEMERR,PPARER
2432 015306 017737 164040 001400  MOV      @CONTRL,PCONTR
2433 015314 017737 164036 001402  MOV      @HITMIS,PHITMI
2434 015322 013737 001436 001412  MOV      OLDPC,BADPC
2435 015330 032777 000014 164014  BIT      #14,@CONTRL
2436 015336 001010              BNE      1$
2437 015340 032777 000340 164030  BIT      #340,@PPARER
2438 015346 001004              BNE      1$
2439 015350 104003              ERROR     3
2440 015352 000403              BR       2$
2441 015354 104077              6$:  ERROR     77
2442 015356 000401              BR       2$
2443 015360 104004              1$:  ERROR     4
2444 015362 005737 001444              2$:  TST      RETRY
2445 015366 001006              BNE      3$
2446 015370 005237 001444              INC      RETRY
2447 015374 013737 001106 001436  MOV      $LPADR,OLDPC
2448 015402 000403              BR       4$
2449 015404 013737 001446 001436  3$:  MOV      NXTTST,OLDPC
2450
2451 015412 012737 177777 015254  4$:  MOV      #-1,PAFLAG
2452 015420 013746 001440              MOV      OLDPS,-(KSP)
2453 015424 013746 001436              MOV      OLDPC,-(KSP)
2454 015430 000006              5$:  RTT
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471 015432 005227      MMTRAP: INC      (PC)+
2472 015434 177777      MMFLAG: .WORD    -1
2473 015436 001401      BEQ      10$
2474 015440 000000      HALT
2475
2476
2477
2478
2479 015442 011637 001412      10$:  MOV      (KSP),BADPC

```

```

; THE SECOND ENTRY ADDRESS IS ON THE
; STACK AND THE FIRST ERROR CONDITION
; IS PROBABLY STILL LOCKED UP
; SAVE RETURN ADDRESS IN CASE OF LOOP
; SAVE OLD PSW IN CASE OF LOOP
; EXECUTING ON AN 11/40?
; BRANCH IF YES
; SAVE MEMORY ERROR REGISTER
; SAVE CONTROL REGISTER FOR TYPE OUT
; SAVE HIT/MISS REGISTER
; SAVE PC+2 AT TIME OF ABORT
; SEE IF CACHE IS ON OR OFF
; IF OFF, MUST BE MEMORY REFERENCE
; WAS THIS A CACHE PARITY ERROR
; BRANCH IF IT WAS A MAIN MEMORY ERROR
; UNEXPECTED CACHE PARITY ERROR
; BRANCH TO EXIT POINT
; UNEXPECTED MAIN MEM. PARITY ERR. 11/40
; BRANCH TO EXIT POINT
; UNEXPECTED MAIN MEMORY PARITY ERROR
; ARE YOU RETRYING THIS TEST?
; BRANCH IF THIS IS SECOND TRY
; SET RETRY FLAG
; RETURN TO START OF THIS TEST
; BRANCH TO EXIT
; RETURN TO START OF NEXT TEST
; SINCE THIS IS THE SECOND ABORT
; RESTORE A NEGATIVE ONE FOR NEXT TIME
; PUSH OLD PSW BACK ON STACK
; PUSH RETURN ADDRESS BACK ON STACK
; RETURN FROM INTERRUPT.

```

```

.SBTTL MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER ROUTINE
;*****
;
; THIS SUBROUTINE WILL HANDLE MOST OF THE MEMORY MANAGEMENT TRAPS
; AND ABORTS THAT ARE GENERATED DURING THIS PROGRAM. ANY M.M.
; ABORTS OR TRAPS THAT OCCUR WHILE 'MMEXP' IS ZERO ARE UNEXPECTED
; AND WILL BE REPORTED AS SUCH. THIS MAY OCCUR BECAUSE SOME LOGIC
; THAT IS TO INHIBIT A TRAP HAS FAILED.
; THERE ARE ALSO TIMES WHEN I AM EXPECTING A CERTAIN CONDITION TO
; OCCUR, AND WHEN THIS CONDITION IN 'MMEXP' DOES NOT MATCH THE
; CONTENTS OF 'PMMRD' (SAME AS 'MMRD') AN ERROR IS REPORTED.
;*****
;*****
MMTRAP: INC      (PC)+
MMFLAG: .WORD    -1
        BEQ      10$
        HALT
; MAKE FLAG ZERO IF FIRST TIME
; FLAG SHOULD BE NEG ONE
; BRANCH IF FIRST TIME INTO ROUTINE
; I HAVE ENTERED THIS ROUTINE BEFORE
; I FINISHED REPORTING THE FIRST ERROR
; THE SECOND ENTRY ADDRESS IS ON THE
; STACK AND THE FIRST ERROR CONDITION
; IS PROBABLY STILL LOCKED UP
; SAVE PC AT TIME OF ABORT OR TRAP

```

L04

MAINDEC-11-DOKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 50
DOKTA.P11 07-FEB-77 10:30 MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER ROUTINE

2480	015446	012637	001436			MOV	(KSP)+,OLDPC	;SAVE RETURN ADDRESS IN CASE OF LOOP
2481	015452	012637	001440			MOV	(KSP)+,OLDPS	;SAVE OLD PSW IN CASE OF LOOP
2482	015456	013737	177572	001404		MOV	MMRO,PMMRO	;SAVE STATUS REGISTER
2483	015464	013737	177576	001406		MOV	MMR2,PMMR2	;SAVE VIRTUAL ADDRESS REGISTER
2484	015472	005737	001366			TST	MMEXP	;SEE IF ANY M.M. TRAPS WERE EXPECTED
2485	015476	001002				BNE	5\$	;BRANCH IF ONE WAS EXPECTED
2486	015500	104005				ERROR	5	;UNEXPECTED M.M. ABORT OR TRAP
2487	015502	000405				BR	1\$	;BRANCH TO EXIT
2488	015504	023737	001366	001404	5\$:	CMP	MMEXP,PMMRO	;SEE IF ABORT OR TRAP WAS EXPECTED
2489	015512	001401				BEQ	1\$	;BRANCH IF CONDITION CORRECT
2490	015514	104006				ERROR	6	;ERROR TYPE OUT ITEM 6
2491	015516	042737	177376	177572	1\$:	BIC	#177376,2#MMRO	;CLEAR ALL BITS EXCEPT 0 AND 8
2492	015524	012737	177777	015434		MOV	#-1,MMFLAG	;RESTORE A NEGATIVE ONE TO FLAG
2493	015532	013746	001440			MOV	OLDPS,-(KSP)	;PUSH OLD PSW ONTO STACK
2494	015536	013746	001436			MOV	OLDPC,-(KSP)	;PUSH RETURN ADDRESS ON STACK
2495	015542	000006				RTT		;RETURN TO MAIN PROGRAM

.SBTTL TRAP ROUTINES FOR ABORT IN USER MODE

THESE NEXT THREE SUBROUTINES ARE USED FOR THE TESTS THAT VERIFY
THE VECTOR IS PICKED UP FROM KERNEL SPACE DURING AN ABORT.
'KERVEC' IS WHERE I SHOULD GO SINCE IT IS AT 250 WHICH IS
KERNEL SPACE VIRTUAL 250.

'USEVEC' IS AT 350 WHICH IS USER SPACE VIRTUAL 250

THEY BOTH READ THE PROCESSOR STATUS WHICH IS DIFFERENT FOR EACH
VECTOR AND THEN JUMP TO 'RDMMR0'. 'RDMMR0' READS MMRO
AND MMR2 AND RETURNS TO THE TEST WHERE THEY ARE TESTED.

2512	015544	013700	177776			KERVEC: MOV	PSW,RO	;PUT PROCESSOR STATUS INTO RO
2513	015550	012637	001436			MOV	(KSP)+,OLDPC	;SAVE RETURN ADDRESS
2514	015554	012637	001440			MOV	(KSP)+,OLDPS	;SAVE OLD PROCESSOR STATUS
2515	015560	122700	000340			CMPB	#340,RO	;SEE IF CORRECT PSW WAS PICKED UP
2516	015564	001401				BEQ	1\$	;BRANCH IF PSW IS CORRECT
2517	015566	104072				ERROR	72	;WRONG PSW PICKED UP
2518	015570	000137	015612		1\$:	JMP	RDMMR0	;JUMP TO READ MMRO
2519								
2520	015574	012637	001436			USEVEC: MOV	(KSP)+,OLDPC	;SAVE RETURN ADDRESS
2521	015600	012637	001440			MOV	(KSP)+,OLDPS	;SAVE OLD PROCESSOR STATUS
2522	015604	013700	177776			MOV	PSW,RO	;READ PSW FOR ERROR PRINT OUT
2523	015610	104073				ERROR	73	;WRONG VECTOR, POSSIBLE WRONG PSW
2524								
2525								
2526	015612	013737	177572	001404		RDMMR0: MOV	MMRO,PMMRO	;READ MMRO TO BE CHECKED LATER
2527	015620	013737	177576	001406		MOV	MMR2,PMMR2	;READ MMR2 FOR POSSIBLE ERROR REPORT
2528	015626	013746	001440			MOV	OLDPS,-(KSP)	;PUSH OLD PROCESSOR STATUS ON STACK
2529	015632	013746	001436			MOV	OLDPC,-(KSP)	;PUSH RETURN ADDRESS ON STACK
2530	015636	000005				RTT		;RETURN TO TEST TO CHECK PMMR0
2531								
2532								

M04

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 51
 DOKTAA.P11 07-FEB-77 10:30

```

2533          .SBTTL
2534          .SBTTL ***** ENTRY POINT 1 --- STARTING ADDRESS 200 *****
2535          .SBTTL ***** TEST CODE STARTS AT ADDRESS 20000 (PAGE 1) *****
2536          .=20000
2537
2538 020000 012737 000000 001362 STRT1: MOV    #0,FSTTST      ;LOAD INDEX TO START TABLE
2539 020006 000427          BR      START                ;BRANCH TO STARTING ADDRESS OF PROGRAM
2540 020010 012737 000002 001362 STRT2: MOV    #2,FSTTST      ;LOAD INDEX TO START TABLE
2541 020016 000423          BR      START                ;BRANCH TO STARTING ADDRESS OF PROGRAM
2542 020020 012737 000004 001362 STRT3: MOV    #4,FSTTST      ;LOAD INDEX TO START TABLE
2543 020026 000417          BR      START                ;BRANCH TO STARTING ADDRESS OF PROGRAM
2544 020030 012737 000006 001362 STRT4: MOV    #6,FSTTST      ;LOAD INDEX TO START TABLE
2545 020036 000413          BR      START                ;BRANCH TO STARTING ADDRESS OF PROGRAM
2546 020040 012737 000010 001362 STRT5: MOV    #10,FSTTST     ;LOAD INDEX TO START TABLE
2547 020046 000407          BR      START                ;BRANCH TO STARTING ADDRESS OF PROGRAM
2548 020050 012737 000012 001362 STRT6: MOV    #12,FSTTST     ;LOAD INDEX TO START TABLE
2549 020056 000403          BR      START                ;BRANCH TO STARTING ADDRESS OF PROGRAM
2550 020060 012737 000014 001362 STRT7: MOV    #14,FSTTST     ;LOAD INDEX TO START TABLE
2551
2552
2553          ;*      NOTE:  THE FIRST THING THAT IS DONE IS TO SEE IF THE
2554          ;*      11/40 OR THE 11/6X MEMORY MGMT. IS BEING TESTED
2555          ;*      AND IF ON AN 11/40 IF THERE IS A STACK LIMIT
2556          ;*      REGISTER.  THE 11/6X "MED" INSTRUCTION IS USED
2557          ;*      TO DETERMINE PROCESSOR TYPE AND THE STACK LIMIT
2558          ;*      REG. IS CHECKED FOR TIMEOUT.
2559
2560          ;*      IF A "MED" DOES NOT EXECUTE PROPERLY ON AN 11/6X
2561          ;*      OR IF THE STACK LIMIT REG. TIMES OUT ON AN 11/40 -
2562          ;*      ERRONEOUS DECISIONS WILL BE MADE DURING TESTING
2563          ;*      WHICH WILL RESULT IN INCOMPLETE TESTING.
2564
2565 020066 012706 001100          START: MOV    #KERSTK,KSP      ;SETUP STACK POINTER
2566 020072 012737 020112 000010 MOV    #15,0#10          ;SETUP ILLEGAL INST. VECTOR
2567 020100 105037 001450          CLR      FORTY              ;CLEAR 11/40 "FLAG LOC."
2568 020104 076600          MED                      ;TRY 11/6X INSTRUCTION
2569 020106 000100          RLJAM                      ;WILL TRAP TO 15 IF ON AN 11/40
2570 020110 000410          BR      3$                    ;WILL BRANCH TO SETUP IF ON 11/6X
2571 020112 062706 000004 1$:  ADD    #4,KSP              ;CLEAN UP STACK IF ON 11/40 AND
2572 020116 105237 001450          INCB     FORTY              ;SET 11/40 "FLAG LOC."
2573 020122 012737 000012 000010 MOV    #12,0#10          ;RESTORE TRAP CATCHER TO LOC. 10
2574 020130 000403          BR      4$                    ;BRANCH TO SETUP
2575 020132 012777 000014 161212 3$: MOV    #14,0#CONTRL      ;TURN 11/6X CACHE OFF FOR FIRST PASS
2576 020140 4$:
2577          .SBTTL INITIALIZE THE COMMON TAGS
2578          ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
2579 020140 012706 001100          MOV    #SCMTAG,R6          ;;FIRST LOCATION TO BE CLEARED
2580 020144 005026          CLR      (R6)+                ;;CLEAR MEMORY LOCATION
2581 020146 022706 001140          CMP     #SWR,R6 ;;DONE?
2582 020152 001374          BNE     -6                    ;;LOOP BACK IF NO
2583 020154 012706 001100          MOV    #STACK,SP          ;;SETUP THE STACK POINTER
2584          ;;INITIALIZE A FEW VECTORS
2585 020160 012737 043004 000020 MOV    #SSCOPE,0#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
2586 020166 012737 000340 000022 MOV    #340,0#IOTVEC+2 ;;LEVEL 7
2587 020174 012737 043300 000030 MOV    #ERROR,0#EMTVEC  ;;EMT VECTOR FOR ERROR ROUTINE
2588 020202 012737 000340 000032 MOV    #340,0#EMTVEC+2 ;;LEVEL 7

```

N04

MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 52
DQKTA.P11 07-FEB-77 10:30 INITIALIZE THE COMMON TAGS

2589	020210	012737	042340	000034	MOV	#STRAP, 2#TRAPVEC	;; TRAP VECTOR FOR TRAP CALLS
2590	020216	012737	000340	000036	MOV	#340, 2#TRAPVEC+2	;; LEVEL 7
2591	020224	012737	042420	000024	MOV	#SPWRDN, 2#PWAVEC	;; POWER FAILURE VECTOR
2592	020232	012737	000340	000026	M V	#340, 2#PWAVEC+2	;; LEVEL 7
2593	020240	013737	040610	040602	MC	SENOCT, SEOPCT	;; SETUP END-OF-PROGRAM COUNTER
2594	020246	005037	001212		CLR	STIMES	;; INITIALIZE NUMBER OF ITERATIONS
2595	020252	005037	001214		CLR	SESCAPE	;; CLEAR THE ESCAPE ON ERROR ADDRESS
2596	020256	112737	000001	001115	MOV	#1, SERMAX	;; ALLOW ONE ERROR PER TEST
2597					;; INITIALIZE THE "1-BIT" TRAP VECTOR. THEN LOAD LOCATION "SRTN", IN		
2598					;; THE "END-OF-PASS" (SEOP) ROUTINE, WITH A "RTI" OR "RTT".		
2599	020264	012737	041026	000014	MOV	#SRTN, 2#TBITVEC	;; SET "T" BIT VECTOR TO SRTN
2600	020272	012737	000340	000016	MOV	#340, 2#TBITVEC+2	;; LEVEL 7
2601	020300	012737	000002	041026	MOV	#RTI, SRTN	;; SET SRTN TO A RTI
2602	020306	012737	020334	000010	MOV	#65\$, 2#RESVEC	;; TRY TO DO A RTT
2603	020314	005046			CLR	-(SP)	;; DUMMY PS
2604	020316	012746	020324		MOV	#64\$, -(SP)	;; AND PC
2605	020322	000006			RTT		;; TRY THE RTT
2606	020324	012737	000006	041026	64\$: MOV	#RTT, SRTN	;; RTT IS LEGAL--SET SRTN TO A RTT
2607	020332	000402			BR	66\$	
2608	020334	062706	000010		65\$: ADD	#10, SP	;; RTT ILLEGAL--CLEAN OFF THE STACK
2609	020340	012737	000012	000010	66\$: MOV	#RESVEC+2, 2#RESVEC	;; RESTORE TRAP CATCHER
2610	020346	005037	041034		CLR	STBIT	;; CLEAR "T" BIT SWITCH
2611	020352	012737	020352	001106	MOV	#, \$LPADR	;; INITIALIZE THE LOOP ADDRESS FOR SCOPE
2612	020360	012737	020360	001110	MOV	#, \$LPERR	;; SETUP THE ERROR LOOP ADDRESS
2613					;; SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS		
2614					;; EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.		
2615	020366	013746	000004		MOV	2#ERRVEC, -(SP)	;; SAVE ERROR VECTOR
2616	020372	012737	020426	000004	MOV	#67\$, 2#ERRVEC	;; SET UP ERROR VECTOR
2617	020400	012737	177570	001140	MOV	#DSWR, SWR	;; SETUP FOR A HARDWARE SWICH REGISTER
2618	020406	012737	177570	001142	MOV	#DDISP, DISPLAY	;; AND A HARDWARE DISPLAY REGISTER
2619	020414	022777	177777	160516	CMP	#-1, 2SWR	;; TRY TO REFERENCE HARDWARE SWR
2620	020422	001012			BNE	69\$	;; BRANCH IF NO TIMEOUT TRAP OCCURRED
2621							;; AND THE HARDWARE SWR IS NOT = -1
2622	020424	000403			BR	68\$	;; BRANCH IF NO TIMEOUT
2623	020426	012716	020434		67\$: MOV	#68\$, (SP)	;; SET UP FOR TRAP RETURN
2624	020432	000002			RTI		
2625	020434	012737	000176	001140	68\$: MOV	#SWREG, SWR	;; POINT TO SOFTWARE SWR
2626	020442	012737	000174	001142	MOV	#DISPREG, DISPLAY	
2627	020450	012637	000004		69\$: MOV	(SP)+, 2#ERRVEC	;; RESTORE ERROR VECTOR
2628							
2629	020454	005037	001234		CLR	\$PASS	;; CLEAR PASS COUNT
2630	020460	132737	000200	001247	BITB	#APTSIZE, \$ENVM	;; TEST USER SIZE UNDER APT
2631	020466	001403			BEQ	70\$	;; YES, USE NON-APT SWITCH
2632	020470	012737	001250	001140	MOV	#SSWREG, SWR	;; NO, USE APT SWITCH REGISTER
2633	020476				70\$:		
2634					.SBTTL	TYPE PROGRAM NAME	
2635					;; TYPE THE NAME OF THE PROGRAM IF FIRST PASS		
2636	020476	005227	177777		INC	#-1	;; FIRST TIME?
2637	020502	001035			BNE	71\$	;; BRANCH IF NO
2638	020504	022737	040762	000042	CMP	#SENDAD, 2#42	;; ACT-11?
2639	020512	001431			BEQ	71\$	;; BRANCH IF YES
2640	020514	104401	020522		TYPE	72\$	;; TYPE ASCIZ STRING
2641	020520	000426			BR	71\$	;; GET OVER THE ASCIZ
2642					72\$: .ASCIZ	<CRLF>'MAINDEC-11-DQKTA-A 11/6X MEM. MGMT. DIAG.'	;; <CRLF>
2643	020576				71\$:		
2644	020576	012737	020576	001110	MOV	#, \$LPERR	;; INITIALIZE LOOP ON ERROR PTR.

```

2645 020604 012737 020604 001106      MOV      #.,$LPADR      ;INITIALIZE LOOP ADDRESS PTR.
2646
2647
2648 020612 005737 001234      LOOP:    TST      $PASS      ;IS THIS THE FIRST PASS?
2649 020616 001406              BEQ      SS          ;BRANCH IF FIRST PASS
2650 020620 105737 001450      TSTB     FORTY      ;EXECUTING ON AN 11/40?
2651 020624 001003              BNE      SS          ;BRANCH IF YES
2652 020626 012777 000200 160516      MOV      #200,$CONTRL ;OTHERWISE TURN CACHE ON AND SET THE
2653                                ;PARITY ABORT BIT
2654 020634 005037 177572      SS:      CLR      MMRO      ;GET INTO 16 BIT MODE ON SECOND PASS
2655 020640 012700 177777      MOV      #-1,RO      ;PUT NEG ONE IN RO TO INITIALIZE FLAGS
2656 020644 010037 015164      MOV      RO,$CPFLAG  ;INITIALIZE CPU ERROR FLAG
2657 020650 010037 015254      MOV      RO,$PFLAG   ;INITIALIZE PARITY ERROR FLAG
2658 020654 010037 015434      MOV      RO,$MMFLAG  ;INITIALIZE MEMORY MANAGEMENT TRAP FLAG
2659 020660 005037 001364      CLR      CPUEXP      ;NOT EXPECTING ANY CPU TRAPS
2660 020664 005037 001366      CLR      MMEXP      ;NOT EXPECTING ANY MEMORY MANAGEMENT TRAPS
2661 020670 012737 015162 000004      MOV      #CPUER,ERRVEC ;LOAD ADDRESS OF CPU TRAP ROUTINE
2662 020676 012737 000340 000006      MOV      #340,ERRVEC+2 ;SET PRIORITY LEVEL 7
2663 020704 012737 015252 000114      MOV      #MEMER,CACHVEC ;LOAD ADDRESS OF PARITY TRAP ROUTINE
2664 020712 012737 000340 000116      MOV      #340,CACHVEC+2 ;SET PRIORITY LEVEL 7
2665 020720 012737 015432 000250      MOV      #MMTRAP,MMVEC ;LOAD ADDRESS OF MEMORY MANAGEMENT TRAP
2666 020726 012737 000340 000252      MOV      #340,MMVEC+2 ;SET PRIORITY LEVEL 7
2667 020734 004737 044202      JSR      PC,CLEANUP   ;INITIALIZE ALL ERROR LOCATIONS
2668 020740 012706 001100      MOV      #KERSTK,KSP ;SET UP KERNEL STACK POINTER
2669 020744 013700 001362      MOV      FSTTST,RO   ;LOAD START TABLE INDEX
2670 020750 001412              BEQ      TST1        ;START WITH TEST ONE IF ZERO
2671 020752 012737 140000 177776      MOV      #140000,PSW  ;GO TO USER MODE
2672 020760 012706 000700      MOV      #USESTK,USP  ;SET UP USER STACK POINTER
2673 020764 012737 000340 177776      MOV      #340,PSW     ;GO TO KERNEL MODE PRIORITY LEVEL 7
2674 020772 000170 001462      JMP      $STRAB(RO)   ;JUMP TO CORRECT ENTRY POINT
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700

```

```

; * THIS FIRST GROUP OF TESTS IS FOR TESTING THE ADDRESS DECODE
; * LOGIC FOR THE INTERNAL REGISTERS AND TO MAKE SOME
; * DETERMINATION ABOUT THE RESPONDING ADDRESSES. IF AN
; * ADDRESS DOES NOT FUNCTION AS THE CORRESPONDING REGISTER SHOULD
; * AN ERROR WILL BE FLAGGED. THE MULTIPLEXERS AND INTERNAL BUS
; * DRIVERS ARE ALSO UTILIZED IN THESE TESTS AND COULD BE
; * THE SOURCE OF ANY PROBLEMS ENCOUNTERED.

```

```

; *****
; *TEST 1 TRY TO READ ALL CPU REGISTERS
; *
; * THIS TEST ENSURES THAT SOMETHING RESPONDS TO SOME INTERNAL ADDRESSES.
; * THIS PROCESSOR STATUS WORD AND STACK LIMIT REG. (IF
; * AVAILABLE) ARE TESTED IF EXECUTING ON AN 11/40. THE CPU
; * ERROR REG. 2 MICROBREAK REG. ARE ALSO CHECKED IF ON AN 11/6X.
; * IF A TRAP TO 'ERRVEC' (004) OCCURS IT IS ASSUMED
; * THAT A REGISTER HAS TIMED OUT, AND AN ERROR IS REPORTED.
; *
; * ERRORS THAT OCCUR IN THIS TEST ARE REPORTED FROM AN AREA
; * AT THE END OF THIS TEST, STARTING AT "50$".
; *****

```

```

; *****
; *TST1:

```


C05

MAINDEC-11-DQKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 54
DQKTA.P11 07-FEB-77 10:30 T1 TRY TO READ ALL CPU REGISTERS

2701	020776	012737	021026	001110		MOV	#20\$,SLPERR	;SET LOOP ON ERROR POINTER TO 20\$
2702	021004	012737	000001	001102		MOV	#1,\$STNM	;LOAD TEST NUMBER INTO MEMORY
2703	021012	013777	001102	160122		MOV	\$STNM,\$DISPLAY	;DISPLAY TEST NUMBER FOR THIS TEST
2704	021020	012737	021130	001446		MOV	#TST2,NXTTST	;SAVE STARTING ADDRESS OF NEXT TEST
2705								;FOR ESCAPE ON PARITY ERRORS
2706	021026	012737	021124	000004	20\$:	MOV	#50\$,ERRVEC	;SET TIME OUT VECTOR TO SPECIAL ROUTINE
2707	021034	012706	001100			MOV	#1100,KSP	;SET KERNEL STACK POINTER TO 1100
2708	021040	012700	177776		1\$:	MOV	#PSW,RO	;SAVE ADDRESS IN CASE OF TIMEOUT
2709	021044	005737	177776			TST	#PSW	;TRY TO READ PROCESSOR STATUS WORD
2710	021050	012700	177774			MOV	#STKLM,RO	;SAVE ADDRESS IN CASE OF TIMEOUT
2711	021054	005737	177774			TST	#STKLM	;TRY TO READ STACK LIMIT REG.
2712								
2713	021060	105737	001450			TSTB	FORTY	;EXECUTING ON AN 11/40?
2714	021064	001010				BNE	2\$	;BRANCH IF YES
2715	021066	013700	001354			MOV	CPUERR,RO	;SAVE ADDRESS IN CASE OF TIMEOUT
2716	021072	005777	160256			TST	CPUERR	;TRY TO READ CPU ERROR REG.
2717	021076	012700	177770			MOV	#177770,RO	;SAVE ADDRESS IN CASE OF TIMEOUT
2718	021102	005737	177770			TST	#177770	;TRY TO READ MICROBREAK REG.
2719								
2720	021106				2\$:			
2721	021106	012737	021026	001110		MOV	#20\$,SLPERR	;SET LOOP POINTER TO START OF TEST
2722	021114	012737	015162	000004		MOV	#CPUERR,ERRVEC	;SET UP C.P.U. ERROR VECTOR
2723	021122	000402				BR	TST2	;TEST OVER BRANCH TO NEXT TEST
2724								
2725	021124	104020			50\$:	ERROR	20	;CPU REGISTER TIMED OUT
2726	021126	000006				RTT		;GO BACK AND FINISH TEST
2727								
2728								
2729								
2730								
2731								
2732								
2733								
2734								
2735								
2736								
2737								
2738								
2739								
2740								
2741								
2742								
2743								
2744								
2745								
2746								
2747								
2748								
2749								
2750								
2751								

TEST 2 STACK LIMIT REGISTER

THE STACK LIMIT REGISTER IS A HIGH(ODD) BYTE ONLY REGISTER, SO THIS TEST TRIES TO LOAD BOTH BYTES OF ADDRESS 177774 AND CHECKS THAT ONLY THE HIGH(ODD) BYTE IS LOADED. THE LOW(EVEN) BYTE WILL ALWAYS BE READ AS ZERO. THIS TEST WILL ALSO BYTE ADDRESS THE STACK LIMIT REGISTER USING ADDR. 177775 AND WRITE/CLEAR EACH BIT OF THE HIGH(ODD) BYTE. A RESET IS ALSO EXEC'ED TO SEE THAT IT WILL CLEAR THE STACK LIMIT REGISTER(DUE TO MICROCODE FOR RESET)

THIS TEST IS SKIPPED IF EXECUTING ON AN 11/40 THAT DOES NOT HAVE A STACK LIMIT REG.

TST2:

2752	021130					SCOPE		
2753	021130	000004				MOV	#TST3,NXTTST	;SAVE STARTING ADDRESS OF NEXT
2754	021132	012737	021340	001446				;TEST FOR ESCAPE ON PARITY ERRORS
2755								;MAKE SURE T-BIT IS OFF FOR THIS TEST
2756	021140	104410				TBITO		

MAINDEC-11-DQKTA-A POP 11/6X MEM. MGMT. DIAG.
DQKTA.P11 07-FEB-77 10:30 T2

MACY11 27(1006) 07-FEB-77 10:37 PAGE 55
STACK LIMIT REGISTER

2757	021142	012737	021154	001110	20\$:	MOV	#1\$,SLPERR	SET LOOP ON ERROR POINTER TO 1\$
2758	021150	012700	177774			MOV	#177774,R0	LOAD ADDRESS OF STACK LIMIT REGISTER
2759	021154	012710	000777		1\$:	MOV	#777,(R0)	ONLY HIGH(ODD) BYTE SHOULD BE LOADED
2760	021160	011001				MOV	(R0),R1	READ STACK LIMIT REGISTER
2761	021162	005010				CLR	(R0)	LEAVE STACK LIMIT REGISTER CLEAR
2762	021164	012702	000400			MOV	#400,R2	LOAD EXPECTED DATA INTO R2
2763	021170	020102				CMP	R1,R2	DID YOU REFERENCE THE STACK LIMIT REG
2764	021172	001401				BEQ	2\$	BRANCH IF IT WAS STACK LIMIT
2765	021174	104026				ERROR	26	NOT STACK LIMIT REGISTER
2766	021176	005200			2\$:	INC	R0	CHANGE ADDRESS TO REFERENCE HIGH(ODD) BYTE
2767	021200	012737	021206	001110		MOV	#3\$,SLPERR	SET LOOP ON ERROR POINTER TO 3\$
2768	021206	112710	000001		3\$:	MOVB	#1,(R0)	SHOULD ONLY LOAD HIGH(ODD) BYTE
2769	021212	005300				DEC	R0	CHANGE ADDRESS TO REFERENCE WORD
2770	021214	011001				MOV	(R0),R1	READ STACK LIMIT REGISTER
2771	021216	005010				CLR	(R0)	LEAVE STACK LIMIT REGISTER CLEAR
2772	021220	012702	000400			MOV	#400,R2	LOAD EXPECTED DATA INTO R2
2773	021224	020102				CMP	R1,R2	DID YOU REFERENCE HIGH(ODD) BYTE
2774	021226	001403				BEQ	4\$	BRANCH IF DATA MATCHES
2775	021230	005200				INC	R0	CHANGE ADDRESS TO ODD BYTE FOR ERROR
2776	021232	104025				ERROR	25	WRONG DATA BACK FROM STACK LIMIT REG
2777	021234	005300				DEC	R0	CHANGE ADDRESS BACK TO WORD ADDRESS
2778	021236	012737	021250	001110	4\$:	MOV	#5\$,SLPERR	SET LOOP ON ERROR POINTER TO 5\$
2779	021244	012702	000400			MOV	#400,R2	LOAD TEST PATTERN INTO R2
2780	021250	010210			5\$:	MOV	R2,(R0)	LOAD TEST PATTERN INTO STACK LIMIT REG.
2781	021252	011001				MOV	(R0),R1	READ STACK LIMIT REGISTER
2782	021254	020201				CMP	R2,R1	WAS CORRECT DATA READ
2783	021256	001402				BEQ	6\$	BRANCH IF DATA MATCHES
2784	021260	005010				CLR	(R0)	CLEAR STK. LMT. REG. FOR ERROR CALL
2785	021262	104025				ERROR	25	WRONG DATA BACK FROM STACK LIMIT REG.
2786	021264	012737	021264	001110	6\$:	MOV	#6\$,SLPERR	SET LOOP ON ERROR POINTER TO 6\$
2787	021272	005010				CLR	(R0)	TRY TO CLEAR THE STACK LIMIT REG
2788	021274	011001				MOV	(R0),R1	READ THE STACK LIMIT REGISTER
2789	021276	001401				BEQ	7\$	BRANCH IF STACK LIMIT REG WAS CLEARED
2790	021300	104021				ERROR	21	STACK LIMIT REG WOULD NOT CLEAR
2791	021302	006302			7\$:	ASL	R2	CHANGE TEST DATA TO TEST NEXT BIT
32	021304	012737	021250	001110		MOV	#5\$,SLPERR	SET LOOP ON ERROR POINTER TO 5\$
93	021312	103356				BCC	5\$	BRANCH UNTIL BITS ARE ALL TESTED
2794	021314	012710	177400			MOV	#177400,(R0)	LOAD STACK LIMIT REG. WITH ALL 1'S
2795	021320	000005				RESET		TRY TO CLEAR WITH A RESET INST.
2796	021322	011001				MOV	(R0),R1	READ THE STACK LIMIT REG.
2797	021324	001402				BEQ	8\$	BRANCH IF WAS CLEARED
2798	021326	005010				CLR	(R0)	CLEAR STK. LMT. REG. FOR ERROR CALL
2799	021330	104021				ERROR	21	STACK LIMIT REG. WOULD NOT CLEAR
2800	021332	012737	021142	001110	8\$:	MOV	#20\$,SLPERR	SET LOOP POINTER TO START OF TEST

```

*****
*TEST 3          PROCESSOR STATUS WORD
*
*      THIS TEST SETS THE PRIORITY LEVEL TO 7 AND CLEARS THE
*      CONDITION CODES AND CHECKS TO SEE THAT ADDRESS 177776
*      HAS 340 IN ITS LOW BYTE.
*****

```

†ST3:

2812 021340

E05

 MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG.
 DOKTAA.P11 07-FEB-77 10:30 T3

 MACY11 27(1006) 07-FEB-77 10:37 PAGE 56
 PROCESSOR STATUS WORD

2813	021340	000004			SCOPE		
2814	021342	012737	021430	001446	MOV	#TST4,NXTTST	;SAVE STARTING ADDRESS OF NEXT
2815							;TEST FOR ESCAPE ON PARITY ERRORS
2816	021350	104411			TBITR		;RESTORE THE T-BIT IF IT WAS ON
2817							;BEFORE THE LAST TEST
2818	021352	012737	021360	001106	MOV	#20\$,SLPADR	;SET LOOP ADDRESS POINTER TO 20\$
2819							;FOR ITERATION PASS
2820	021360	012737	021376	001110	20\$: MOV	#1\$,SLPERR	;SET LOOP ON ERROR POINTER TO 1\$
2821	021366	012700	177776		MOV	#177776,R0	;LOAD ADDRESS OF P.S.W. INTO R0
2822	021372	012702	000340		MOV	#340,R2	;LOAD EXPECTED DATA INTO R2
2823	021376	005010			1\$: CLR	(R0)	;CLEAR THE PSW
2824	021400	052710	000340		BIS	#340,(R0)	;SET PRIORITY TO LEVEL SEVEN
2825	021404	000257			CCC		;CLEAR ALL CONDITION CODES
2826	021406	111001			MOV	(R0),R1	;LOAD LOWER BYTE OF P.S.W. INTO R1
2827	021410	042701	177420		BIC	#177420,R1	;CLEAR "REG. SIGN EXTEND" AND T-BIT
2828	021414	020201			CMP	R2,R1	;LOWER BYTE OF PSW SHOULD BE 340
2829	021416	001401			BEQ	2\$	;BRANCH IF PSW IS CORRECT
2830	021420	104023			ERROR	23	;MUST HAVE READ SOME OTHER REGISTER
2831	021422	012737	021360	001110	2\$: MOV	#20\$,SLPERR	;SET LOOP POINTER TO START OF TEST

```

*****
;TEST 4      SET UP THE STACK POINTERS FOR REST OF TESTS
;
;      THIS TEST IS USED TO SET THE KERNEL STACK POINTER AT
;      1100 AND THE USER STACK POINTER AT 700. IT THEN RETURNS
;      TO KERNEL MODE AND VERIFIES THAT THE KERNEL STACK
;      POINTER IS STILL AT 1100.
;
*****

```

```

*****
;TST4:
SCOPE
MOV      #TST5,NXTTST      ;SAVE STARTING ADDRESS OF NEXT
                                ;TEST FOR ESCAPE ON PARITY ERRORS
                                ;GET TO KERNEL MODE, PRIORITY 0
                                ;SETUP KERNEL STACK POINTER
                                ;GET INTO USER MODE
                                ;SET UP USER STACK POINTER
                                ;GET BACK INTO KERNEL MODE
                                ;READ KSRNL STACK POINTER
                                ;SEE IF KERNEL STACK IS STILL 1100
                                ;BRANCH IF KERNEL STACK IS OKAY
                                ;KERNEL STACK POINTER IS NOT RIGHT
CLR      PSW
MOV      #KERSTK,KSP
MOV      #140000,PSW
MOV      #USESTK,USP
CLR      PSW
MOV      KSP,R0
CMP      #KERSTK,R0
BEQ      TST5
ERROR    27

```

```

.SBTTL ***** ENTRY POINT 2 --- STARTING ADDRESS 204 *****
.SBTTL ** TEST READ/WRITE BITS IN MEMORY MANAGEMENT STATUS REGISTERS **

```

```

;
;      THIS GROUP OF TESTS EXERCISES ALL OF THE READ/WRITE BITS
;      IN MMRO, AND TRIES A FEW VIRTUAL ADDRESSES IN MMR2.
;      THESE TESTS SHOW THAT ONCE THE MMR LOCK UP SIGNAL GOES LOW
;      MMR2 STOPS TRACKING THE C.P.U. OPERATIONS.
;

```

```

;*****

```

2868

F05

MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 57
 DQKTA.P11 07-FEB-77 10:30 TS BIT TEST OF MEMORY MANAGEMENT REGISTER 0

```

2869      *TEST 5      BIT TEST OF MEMORY MANAGEMENT REGISTER 0
2870      *
2871      *      THIS TEST TRIES TO SET AND CLEAR BITS <15:13> OF
2872      *      MMRO. 'INIT' IS ISSUED TO SEE THAT IT WILL CLEAR THESE BITS.
2873      *      THE OTHER BITS OF MMRO ARE CLOCKED ONLY ON MEMORY
2874      *      MANAGEMENT ERROR CONDITIONS IF RELOCATION IS ENABLED.
2875      *      BIT <00> ENABLES FULL RELOCATION AND BIT <08> ENABLES
2876      *      RELOCATION ON THE DESTINATION CYCLE ONLY. THESE BITS WILL
2877      *      BE TESTED IN A LATER TEST.
2878      *
2879      *
2880      *
2881      *
2882      *
2883      *
2884      *
2885      *
2886      *
2887      *
2888      *
2889      *
2890      *
2891      *
2892      *
2893      *
2894      *
2895      *
2896      *
2897      *
2898      *
2899      *
2900      *
2901      *
2902      *
2903      *
2904      *
2905      *
2906      *
2907      *
2908      *
2909      *
2910      *
2911      *
2912      *
2913      *
2914      *
2915      *
2916      *
2917      *
2918      *
2919      *
2920      *
2921      *
2922      *
2923      *
2924      *

```

TS5: SCOPE

MOV #TST6,NXTTST ;SAVE STARTING ADDRESS OF NEXT

ENTPT2: ;TEST FOR ESCAPE ON PARITY ERRORS

MOV #20\$,SLPERR ;SET LOOP ON ERROR POINTER TO 20\$

MOV #5\$,STSTNM ;LOAD TEST NUMBER INTO MEMORY

STSTNM,20\$DISPLAY ;DISPLAY TEST NUMBER FOR THIS TEST

20\$: MOV #MMRO,R0 ;PUT ADDRESS OF MMRO IN R0

MOV #160000,(R0) ;LOAD WRITABLE BITS IN MMRO

RESET ;'INIT' SHOULD CLEAR ALL BITS OF MMRO

MOV (R0),R1 ;READ MMRO

BEQ 1\$;BRANCH IF MMRO IS CLEAR

ERROR 7 ;MMRO WON'T CLEAR

1\$: MOV #11\$,SLPERR ;SET LOOP ON ERROR POINTER TO 11\$

11\$: MOV #160000,(R0) ;SET BITS <15:13> OF MMRO

MOV (R0),R1 ;READ MMRO

CMP #160000,R1 ;SEE IF BITS <15:13> ARE SET

BEQ 2\$;BRANCH IF CORRECT BITS ARE SET

ERROR 10 ;CAN'T SET 160000 IN MMRO

2\$: MOV #12\$,SLPERR ;SET LOOP ON ERROR POINTER TO 12\$

12\$: CLR (R0) ;CLEAR UPPER 3 BITS OF MMRO

MOV (R0),R1 ;READ MMRO

BEQ 3\$;BRANCH IF MMRO IS CLEAR

ERROR 7 ;MMRO WON'T CLEAR

3\$: MOV #5\$,SLPERR ;SET LOOP ON ERROR POINTER TO 5\$

MOV #BIT15,R2 ;SET BIT IN R2 TO FLOAT THRU MMRO

MOV #3,R3 ;PUT LOOP COUNT IN R3

5\$: MOV R2,(R0) ;LOAD R2 INTO MMRO

MOV (R0),R1 ;READ MMRO IN R1

CLR (R0) ;CLEAR MMRO

MOV R2,R4 ;PUT PATTERN IN R4

BIC #017777,R4 ;MASK OFF BITS <12:00>

CMP R4,R1 ;COMPARE PATTERN WITH MMRO

BEQ 4\$;BRANCH IF DATA MATCHES

ERROR 11 ;GOT WRONG DATA FROM MMRO

4\$: ROR R2 ;SHIFT BIT TO RIGHT

SOB R3,5\$;LOOP BACK TWICE

MOV #20\$,SLPERR ;SET LOOP POINTER TO START OF TEST

*TEST 6 SEE THAT MEMORY MANAGEMENT REGISTER 1 READS 0'S

*

G05

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG.
DOKTAA.P11 07-FEB-77 10:30MACY11 27(1006) 07-FEB-77 10:37 PAGE 58
T6 SEE THAT MEMORY MANAGEMENT REGISTER 1 READS 0'S

```

2925      *      THIS TEST WILL ENSURE THAT ALTHOUGH MEMORY MANAGEMENT
2926      *      REGISTER #1 (777574) IS NON-EXSISTENT, THAT ADDRESS
2927      *      WILL RESPOND AND READ AS THOUGH THERE WAS A REGISTER
2928      *      THERE AND THAT IT READS AS ALL ZEROES.
2929      *
2930      * *****
2931      TST6:
2932      SCOPE
2933      MOV      #TST7,NXTTST      ;SAVE STARTING ADDRESS OF NEXT
2934      ;TEST FOR ESCAPE ON PARITY ERRORS
2935
2936      20$:  MOV      #MMR1,R0      ;PUT ADDRESS OF MMR1 IN R0
2937      MOV      #-1,R1      ;LOAD R1 WITH ALL ONES
2938      MOV      (R0),R1      ;READ MMR1 INTO R1
2939      TST      R1      ;DID MMR1 READ AS ALL ZEROES?
2940      BEQ      1$      ;BRANCH IF YES
2941      ERROR    24      ;MMR1 DID NOT READ ALL ZEROES
2942
2943      1$:
2944
2945      ; *****
2946      ; TEST 7      BIT TEST OF MEMORY MANAGEMENT REGISTER 2
2947      ;
2948      ; HERE MMR2 WILL BE READ SEVERAL DIFFERENT TIMES TO
2949      ; SEE THAT IT IS TRACKING PROPERLY. THEN IT WILL BE
2950      ; LOCKED UP AND READ IT TO SEE THAT IT DOES NOT CHANGE AFTER
2951      ; IT IS ONCE LOCKED UP. NOT ALL BITS WILL BE TESTED IN THIS
2952      ; TEST BUT A LATER TEST RUNS A COUNT PATTERN THROUGH MMR2. THIS
2953      ; CANNOT BE DONE HERE SINCE IT REQUIRES THAT THE ABORT LOGIC IS
2954      ; FUNCTIONING PROPERLY.
2955      ; *****
2956      ; *****
2957      TST7:
2958      SCOPE
2959      MOV      #TST10,NXTTST      ;SAVE STARTING ADDRESS OF NEXT
2960      ;TEST FOR ESCAPE ON PARITY ERRORS
2961
2962      20$:  MOV      #MMR0,R0      ;PUT ADDRESS OF MMR0 IN R0
2963      MOV      #MMR2,R1      ;PUT ADDRESS OF MMR2 IN R1
2964      MOV      #21$,SLPERR      ;SET LOOP ON ERROR POINTER TO 21$
2965      CLR      @MMR0      ;MAKE SURE THAT MMR0 IS CLEAR
2966      21$:  MOV      (R1),R2      ;READ MMR2 TO R2
2967      MOV      #21$,R3      ;PUT ADDRESS OF LAST INST IN R3
2968      CMP      R2,R3      ;SEE IF MMR2 HELD CORRECT ADDRESS
2969      BEQ      1$      ;BRANCH IF DATA MATCHES
2970      ERROR    12      ;MMR2 DID NOT TRACK PROPERLY
2971      1$:  MOV      #22$,SLPERR      ;SET LOOP ON ERROR POINTER TO 22$
2972      22$:  MOV      @MMR2,R2      ;READ MMR2 TO R2
2973      MOV      #22$,R3      ;PUT ADDRESS OF LAST INST IN R3
2974      CMP      R2,R3      ;SEE IF MMR2 HELD CORRECT ADDRESS
2975      BEQ      2$      ;BRANCH IF DATA MATCHES
2976      ERROR    12      ;MMR2 DID NOT TRACK PROPERLY
2977      2$:  MOV      #23$,SLPERR      ;SET LOOP ON ERROR POINTER TO 23$
2978      23$:  MOV      #40000,(R0)      ;LOCK UP MMR2
2979      MOV      (R1),R2      ;READ MMR2 INTO R2
2980      MOV      #23$,R3      ;PUT LOCKING INST'S ADDRESS IN R3

```

H05

MAINDEC-11-DQKTA-A POP 11/6X MEM. MGMT. DIAG.
DQKTA.P11 07-FEB-77 10:30 T7

MACY11 27(1006) 07-FEB-77 10:37 PAGE 59
BIT TEST OF MEMORY MANAGEMENT REGISTER 2

2981	022030	020203				CMP	R2,R3	;SEE IF MMR2 HOLDS RIGHT ADDRESS
2982	022032	001401				BEQ	3\$	;BRANCH IF DATA MATCHES
2983	022034	104012				ERROR	12	;MMR2 DID NOT TRACK PROPERLY
2984	022036	012737	022044	001110	3\$:	MOV	#14\$,SLPERR	;SET LOOP ON ERROR POINTER TO 14\$
2985	022044	000005			14\$:	RESET		;ISSUE INIT TO CLEAR MMR2
2986	022046	011102			25\$:	MOV	(R1),R2	;READ MMR2 INTO R2
2987	022050	012703	022046			MOV	#25\$,R3	;PUT ADDRESS OF LAST INST IN R3
2988	022054	020203				CMP	R2,R3	;SEE IF MMR2 IS STILL TRACKING
2989	022056	001401				BEQ	TST10	;GO TO NEXT TEST IF IT IS
2990	022060	104012				ERROR	12	;MMR2 DID NOT TRACK PROPERLY
2991								
2992								
2993								
2994						.SBTTL	***** ENTRY POINT 3 --- STARTING ADDRESS 210 *****	
2995						.SBTTL	***** PAGE ADDRESS AND DESCRIPTOR REGISTER TESTS *****	
2996						*		
2997						*		
2998						*	THIS GROUP OF TESTS IS USED TO CHECK ALL THE PAR'S AND	
2999						*	PDR'S; THEIR ADDRESS DECODING ON DIRECT REFERENCES, THEIR	
3000						*	READ/WRITE BITS, AND DUAL ADDRESSING WITHIN A GROUP OR BETWEEN	
3001						*	TWO GROUPS.	
3002						*		
3003								
3004						.SBTTL	TEST THAT ALL P.A.R.'S & P.D.R.'S RESPOND	
3005						*	THESE NEXT FOUR (4) TESTS VERIFY THAT THERE ARE 4 GROUPS OF	
3006						*	8 CONTIGUOUS ADDRESSES IN THE I/O PAGE THAT WILL RESPOND	
3007						*	WITHOUT TIMING OUT. AT THIS POINT THE TEST IS NOT CONCERNED	
3008						*	WITH EXACTLY WHAT IS RESPONDING, JUST THAT SOMETHING RESPONDS.	
3009						*		
3010						*	*****	
3011						*TEST 10	READ ALL KERNEL PAGE ADDRESS REGISTERS, CHECK FOR TIMEOUT	
3012						*		
3013						*	THIS TEST DOES A READ FROM ALL THE KERNEL PAGE ADDRESS REGISTERS	
3014						*	'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE TEST AND	
3015						*	IF ANY OF THE 8 KERNEL ADDRESS REGISTERS TIME OUT THEIR I/O	
3016						*	PAGE ADDRESS IS REPORTED AND LOGGED. AT THE END OF THE TEST A	
3017						*	SUMMARY OF THE ERRORS IS GIVEN AND 'ERRVEC' IS RE-SET TO 'CPUER'.	
3018						*		
3019						*	ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE "TIMEOUT"	
3020						*		
3021						*	*****	
3022	022062					TST10:		
3023	022062	000004				SCOPE		
3024	022064	012737	022206	001446		MOV	#TST11,NXTTST	;SAVE STARTING ADDRESS OF NEXT
3025								;TEST FOR ESCAPE ON PARITY ERRORS
3026	022072					ENTPT3:		
3027	022072	012737	022114	001110		MOV	#20\$,SLPERR	;SET LOOP ON ERROR POINTER TO 20\$
3028	022100	012737	000010	001102		MOV	#10\$,TSTNM	;LOAD TEST NUMBER INTO MEMORY
3029	022106	013777	001102	157026		MOV	\$TSTNM,DISP	;DISPLAY TEST NUMBER FOR THIS TEST
3030	022114	004737	044202		20\$:	JSR	PC,CLEANUP	;INITIALIZE ERROR LOCATIONS
3031	022120	012737	044240	000004		MOV	#TIMEOUT,ERRVEC	;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
3032	022126	012737	022140	001110		MOV	#1\$,SLPERR	;SET LOOP ON ERROR POINTER TO 1\$
3033	022134	012700	172340			MOV	#KIPAR0,R0	;PUT ADDRESS OF FIRST PAR IN R0
3034	022140	011001			1\$:	MOV	(R0),R1	;THIS WILL TIMEOUT IF REGISTER DECODING BAD
3035	022142	062700	000002			ADD	#2,R0	;POINT TO NEXT REGISTER
3036	022146	022700	172356			CMP	#KIPAR7,R0	;SEE IF KIPAR7 HAS BEEN TRIED

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG.
DOKTAA.P11 07-FEB-77 10:30 T10

MACY11 27(1006) 07-FEB-77 10:37 PAGE 60
READ ALL KERNEL PAGE ADDRESS REGISTERS, CHECK FOR TIMEOUT

3037	022152	103372			BHIS	15	;BRANCH IF NOT DONE
3038	022154	012737	022114	001110	MOV	#20\$, \$LPERR	;PUT LOOP ON ERROR POINTER TO START OF TEST
3039	022162	012737	015162	000004	MOV	#CPUER, ERRVEC	;RE-SET NORMAL CPU TRAP SERVICE ROUTINE
3040	022170	005737	001432		TST	ERRCNT	;SEE IF ANY ERRORS OCCURRED ON THIS TEST
3041	022174	001404			BEQ	TST11	;BRANCH TO NEXT TEST IF NO ERRORS
3042	022176	013737	001432	001210	MOV	ERRCNT, \$TMP5	;SAVE NUMBER OF ERRORS FOR TYPEOUT
3043	022204	104013			ERROR	13	;SUMMARY OF PAR ERRORS

3044
3045
3046

*TEST 11 READ ALL USER PAGE ADDRESS REGISTERS, CHECK FOR TIMEOUT

THIS TEST DOES A READ FROM ALL THE USER PAGE ADDRESS
REGISTERS. 'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE
TEST, AND IF ANY OF THE 8 USER ADDRESS REGISTERS TIME OUT
THEIR I/O PAGE ADDRESS IS REPORTED AND LOGGED. AT THE END
OF THE TEST A SUMMARY OF ERRORS IS GIVEN AND 'ERRVEC' IS SET TO
'CPUER'.
ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE "TIMEOUT"

3055
3056
3057

TST11:

3058	022206			
3059	022206	000004		
3060	022210	012737	022310	001446

	SCOPE		
	MOV	#TST12, NXTTST	;SAVE STARTING ADDRESS OF NEXT
			;TEST FOR ESCAPE ON PARITY ERRORS
20\$:	JSR	PC, CLEANUP	;INITIALIZE ERROR LOCATIONS
	MOV	#TIMEOUT, ERRVEC	;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
	MOV	#1\$, \$LPERR	;SET LOOP ON ERROR POINTER TO 15
	MOV	#UIPAR0, R0	;PUT ADDRESS OF FIRST PAR IN R0
1\$:	MOV	(R0), R1	;THIS WILL TIMEOUT IF REGISTER DECODING BAD
	ADD	#2, R0	;POINT TO NEXT REGISTER
	CMP	#UIPAR7, R0	;SEE IF UIPAR7 HAS BEEN TRIED
	BHIS	15	;BRANCH IF NOT DONE
	MOV	#20\$, \$LPERR	;PUT LOOP ON ERROR POINTER TO START OF TEST
	MOV	#CPUER, ERRVEC	;RE-SET NORMAL CPU TRAP SERVICE ROUTINE
	TST	ERRCNT	;SEE IF ANY ERRORS OCCURRED ON THIS TEST
	BEQ	TST12	;BRANCH TO NEXT TEST IF NO ERRORS
	MOV	ERRCNT, \$TMP5	;SAVE NUMBER OF ERRORS FOR TYPEOUT
	ERROR	13	;SUMMARY OF PAR ERRORS

3061				
3062	022216	004737	044202	
3063	022222	012737	044240	000004
3064	022230	012737	022242	001110
3065	022236	012700	177640	

3066	022242	011001		
3067	022244	062700	000002	
3068	022250	022700	177656	

3069	022254	103372		
3070	022256	012737	022216	001110
3071	022264	012737	015162	000004

3072	022272	005737	001432	
3073	022276	001404		
3074	022300	013737	001432	001210
3075	022306	104013		

3076
3077

*TEST 12 READ ALL KERNEL PAGE DESCRIPTOR REGISTERS, CHECK FOR TIMEOUT

THIS TEST DOES A READ FROM ALL THE KERNEL PAGE DESCRIPTOR
REGISTERS. 'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE
TEST, AND IF ANY OF THE 8 KERNEL DESCRIPTOR REGISTERS TIME OUT
THEIR I/O PAGE ADDRESS IS REPORTED AND LOGGED. AT THE END
OF THE TEST A SUMMARY OF ERRORS IS GIVEN AND 'ERRVEC' IS SET TO
'CPUER'.
ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE "TIMEOUT"

3078
3079

3080
3081

3082
3083

3084
3085

3086
3087

3088
3089

3089	022310			
3090	022310	000004		
3091	022312	012737	022412	001446

TST12:

	SCOPE		
	MOV	#TST13, NXTTST	;SAVE STARTING ADDRESS OF NEXT
			;TEST FOR ESCAPE ON PARITY ERRORS

3092

J05

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG.
DOKTAA.P11 07-FEB-77 10:30 T12

MACY11 27(1006) 07-FEB-77 10:37 PAGE 61
READ ALL KERNEL PAGE DESCRIPTOR REGISTERS, CHECK FOR TIMEOUT

3093	022320	004737	044202		205:	JSR	PC,CLEANUP	; INITIALIZE ERROR LOCATIONS
3094	022324	012737	044240	000004		MOV	#TIMEOUT,ERRVEC	; SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
3095	022332	012737	022344	001110		MOV	#15,\$LPERR	; SET LOOP ON ERROR POINTER TO 15
3096	022340	012700	172300			MOV	#KIPDR0,R0	; PUT ADDRESS OF FIRST PDR IN R0
3097	022344	011001			15:	MOV	(R0),R1	; THIS WILL TIMEOUT IF REGISTER DECODING BAD
3098	022346	062700	000002			ADD	#2,R0	; POINT TO NEXT REGISTER
3099	022352	022700	172316			CMP	#KIPDR7,R0	; SEE IF KIPDR7 HAS BEEN TRIED
3100	022356	103372				BHIS	15	; BRANCH IF NOT DONE
3101	022360	012737	022320	001110		MOV	#205,\$LPERR	; PUT LOOP ON ERROR POINTER TO START OF TEST
3102	022366	012737	015162	000004		MOV	#CPUER,ERRVEC	; RE-SET NORMAL CPU TRAP SERVICE ROUTINE
3103	022374	005737	001432			TST	ERRCNT	; SEE IF ANY ERRORS OCCURRED ON THIS TEST
3104	022400	001404				BEQ	TST13	; BRANCH TO NEXT TEST IF NO ERRORS
3105	022402	013737	001432	001210		MOV	ERRCNT,\$TMP5	; SAVE NUMBER OF ERRORS FOR TYPEOUT
3106	022410	104013				ERROR	13	; SUMMARY OF PDR ERRORS

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
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143

*TEST 13 READ ALL USER PAGE DESCRIPTOR REGISTERS, CHECK FOR TIMEOUT

THIS TEST DOES A READ FROM ALL THE USER PAGE DESCRIPTOR
REGISTERS. 'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE
TEST, AND IF ANY OF THE 8 USER DESCRIPTOR REGISTERS TIME OUT
THEIR I/O PAGE ADDRESS IS REPORTED AND LOGGED. AT THE END
OF THE TEST A SUMMARY OF ERRORS IS GIVEN AND 'ERRVEC' IS SET TO
'CPUER'.
ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE "TIMEOUT"

*TST13:

					SCOPE			
					MOV	#TST14,NXTTST		; SAVE STARTING ADDRESS OF NEXT
								; TEST FOR ESCAPE ON PARITY ERRORS
					205:	JSR	PC,CLEANUP	; INITIALIZE ERROR LOCATIONS
					MOV	#TIMEOUT,ERRVEC		; SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
					MOV	#15,\$LPERR		; SET LOOP ON ERROR POINTER TO 15
					MOV	#UIPDR0,R0		; PUT ADDRESS OF FIRST PDR IN R0
				15:	MOV	(R0),R1		; THIS WILL TIMEOUT IF REGISTER DECODING BAD
					ADD	#2,R0		; POINT TO NEXT REGISTER
					CMP	#UIPDR7,R0		; SEE IF UIPDR7 HAS BEEN TRIED
					BHIS	15		; BRANCH IF NOT DONE
					MOV	#205,\$LPERR		; PUT LOOP ON ERROR POINTER TO START OF TEST
					MOV	#CPUER,ERRVEC		; RE-SET NORMAL CPU TRAP SERVICE ROUTINE
					TST	ERRCNT		; SEE IF ANY ERRORS OCCURRED ON THIS TEST
					BEQ	TST14		; BRANCH TO NEXT TEST IF NO ERRORS
					MOV	ERRCNT,\$TMP5		; SAVE NUMBER OF ERRORS FOR TYPEOUT
					ERROR	13		; SUMMARY OF PDR ERRORS

*.SBTTL TEST FOR DUAL ADDRESSING ON LOAD WITHIN GROUPS
* THESE NEXT FOUR (4) TESTS WILL CHECK FOR DUAL ADDRESSING WITHIN A
* GROUP OF PAR'S OR PDR'S. FIRST ALL OF THE REGISTERS IN A GROUP
* ARE CLEARED AND READ TO SEE THAT THEY CAN EACH HOLD ZEROES, AND

K05

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG.
DOKTAA.P11 07-FEB-77 10:30

MACY11 27(1006) 07-FEB-77 10:37 PAGE 62
TEST FOR DUAL ADDRESSING ON LOAD WITHIN GROUPS

3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163 022514
3164 022514 000004
3165 022516 012737 022720 001446
3166
3167 022524 004737 044202
3168 022530 012737 022556 001110
3169 022536 012705 172340
3170 022542 004737 044164
3171 022546 012700 172340
3172 022552 012701 000010
3173 022556 011002
3174 022560 001401
3175 022562 104014
3176 022564 062700 000002
3177 022570 077106
3178
3179
3180
3181
3182
3183 022572 012737 022604 001110
3184 022600 012700 172340
3185 022604 012705 172340
3186 022610 004737 044164
3187 022614 012701 172356
3188 022620 012710 177777
3189 022624 005037 001176
3190 022630 011102
3191 022632 001406
3192 022634 020001
3193
3194 022636 001402
3195 022640 004737 044334
3196 022644 005237 001176
3197 022650 162701 000002
3198 022654 022701 172340
3199 022660 101761

```

*      THAT THE DATA PATH DOES NOT HAVE A BIT STUCK AT ONE.
*      THEN ONE REGISTER AT A TIME, WITHIN THAT GROUP, IS LOADED
*      WITH A NEGATIVE ONE WHILE ALL REGISTERS ARE READ TO SEE
*      THAT ONLY THE REGISTER UNDER TEST WAS NOT ZERO.
*****
*TEST 14      DUAL ADDRESS KERNEL PAGE ADDRESS REGISTERS, ON LOADING
*****
*      THIS TEST FIRST CLEARS ALL THE KERNEL PAGE ADDRESS REGISTERS,
*      AND CHECKS TO SEE THAT THEY EACH HOLD ZERO. THEN, STARTING
*      WITH ADDRESS REGISTER ZERO, ONE REGISTER AT A TIME
*      IS LOADED WITH A NEGATIVE ONE. ALL KERNEL ADDRESS REGISTERS
*      ARE NOW READ TO SEE THAT ONLY THE REGISTER UNDER TEST IS NON-
*      ZERO. INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM
*      IS GIVEN AT THE END OF THIS TEST.
*
*      ALL ERRORS IN THIS TEST ARE REPORTED AND ANALYZED BY SUBROUTINE
*      "DUALADR".
*****
*ST14:
SCOPE
MOV      #TST15,NXTTST      ;SAVE STARTING ADDRESS OF NEXT
                             ;TEST FOR ESCAPE ON PARITY ERRORS
20$:     JSR      PC,CLEANUP  ;INITIALIZE ERROR LOCATIONS
MOV      #1$,SLPERR         ;SET LOOP ON ERROR POINTER TO 1$
MOV      #KIPAR0,R5         ;PUT ADDRESS OF FIRST PAR IN R5
JSR      PC,CLRREG          ;CLEAR 8 REGISTERS POINTED TO BY R5
MOV      #KIPAR0,R0         ;PUT ADDRESS OF FIRST PAR IN R0
MOV      #10,R1             ;BRANCH COUNT IS 8 DECIMAL
1$:      MOV      (R0),R2     ;READ PAR TO R2
BEQ      2$                 ;BRANCH IF PAR IS 0
ERROR    14                 ;PAR NOT ZERO
2$:      ADD      #2,R0       ;POINT TO NEXT REGISTER
SOB      R1,1$             ;BRANCH BACK TO 1$ 7 TIMES
;
;
;      NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE
;      REGISTER AND READING THE REST TO SEE ONLY THAT REGISTER
;      IS NON-ZERO. (DROPPED BITS WILL BE FOUND IN NEXT TEST.)
;
MOV      #3$,SLPERR         ;SET LOOP ON ERROR POINTER TO 3$
MOV      #KIPAR0,R0         ;PUT ADDRESS OF FIRST PAR IS R0
3$:      MOV      #KIPAR0,R5 ;LOAD STARTING ADDRESS INTO R5
JSR      PC,CLRREG          ;CLEAR 8 REGISTERS POINTED TO BY R5
MOV      #KIPAR7,R1         ;PUT KIPAR7 ADDRESS IN R1
MOV      #-1,(R0)           ;LOAD REGISTER UNDER TEST
4$:      CLR      $TMP0      ;FLAG TO INDICATE THERE WAS A MATCH
MOV      (R1),R2            ;READ ALL REGISTERS
BEQ      6$                 ;BRANCH IF REGISTER IS 0
CMP      R0,R1              ;IS ADDRESS OF NON-ZERO REGISTER SAME
                             ;AS THAT OF REGISTER UNDER TEST
BEQ      5$                 ;BRANCH IF ADDRESSES MATCH
JSR      PC,DUALADR         ;LOG AND REPORT ERRORS
INC      $TMP0              ;SET FLAG WHEN ADDRESSES MATCH
5$:      SUB      #2,R1      ;POINT TO NEXT REGISTER
CMP      #KIPAR0,R1         ;SEE IF ALL REGISTERS HAVE BEEN READ
BLOS     4$                 ;BRANCH IF MORE TO READ
6$:

```

MAINDEC-11-DQKTA-A POP 11/6X MEM. MGMT. DIAG.
DQKTAAR.P11 07-FEB-77 10:30 T14

MACY11 27(1006) 07-FEB-77 10:37 PAGE 63
DUAL ADDRESS KERNEL PAGE ADDRESS REGISTERS, ON LOADING

3200	022662	062700	000002		ADD	#2,RO	:NOW LOAD NEXT REGISTER
3201	022666	022700	172356		CMP	#KIPAR7,RO	:SEE IF THERE ARE MORE REGISTERS TO TEST
3202	022672	103344			BHIS	3\$	:BRANCH IF MORE REGISTERS TO TEST
3203	022674	012737	022524	001110	MOV	#20\$,SLPERR	:SET LOOP ON ERROR POINTER TO START OF TEST
3204	022702	005737	001432		TST	ERRCNT	:SEE IF THERE WERE ANY ERRORS
3205	022706	001404			BEQ	TST15	:BRANCH TO NEXT TEST IF NO ERRORS
3206	022710	013737	001432	001210	MOV	ERRCNT,\$TMP5	:SAVE NUMBER OF ERRORS FOR PAR OUT
3207	022716	104015			ERROR	15	:SUMMARY OF DUAL ADDRESSING ERRORS

3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255

*TEST 15 DUAL ADDRESS USER PAGE ADDRESS REGISTERS, ON LOADING

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

ALL ERRORS IN THIS TEST ARE REPORTED AND ANALYZED BY SUBROUTINE "DUALADR".

*TST15:

	SCOPE		
	MOV	#TST16,NXTTST	:SAVE STARTING ADDRESS OF NEXT
			:TEST FOR ESCAPE ON PARITY ERRORS
20\$:	JSR	PC,CLEANUP	:INITIALIZE ERROR LOCATIONS
	MOV	#1\$,SLPERR	:SET LOOP ON ERROR POINTER TO 1\$
	MOV	#UIPAR0,R5	:PUT ADDRESS OF FIRST PAR IN R5
	JSR	PC,CLRREG	:CLEAR 8 REGISTERS POINTED TO BY R5
	MOV	#UIPAR0,RO	:PUT ADDRESS OF FIRST PAR IN RO
	MOV	#10,R1	:BRANCH COUNT IS 8 DECIMAL
1\$:	MOV	(RO),R2	:READ PAR TO R2
	BEQ	2\$	:BRANCH IF PAR IS 0
	ERROR	14	:PAR NOT ZERO
2\$:	ADD	#2,RO	:POINT TO NEXT REGISTER
	SUB	R1,1\$	:BRANCH BACK TO 1\$ 7 TIMES

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

	MOV	#3\$,SLPERR	:SET LOOP ON ERROR POINTER TO 3\$
	MOV	#UIPAR0,RO	:PUT ADDRESS OF FIRST PAR IS RO
3\$:	MOV	#UIPAR0,R5	:LOAD STARTING ADDRESS INTO R5
	JSR	PC,CLRREG	:CLEAR 8 REGISTERS POINTED TO BY R5
	MOV	#UIPAR7,R1	:PUT UIPAR7 ADDRESS IN R1
	MOV	#-1,(RO)	:LOAD REGISTER UNDER TEST
4\$:	CLR	\$TMP0	:FLAG TO INDICATE THERE WAS A MATCH
	MOV	(R1),R2	:READ ALL REGISTERS
	BEQ	6\$	:BRANCH IF REGISTER IS 0
	CMP	RO,R1	:IS ADDRESS OF NON-ZERO REGISTER SAME
			:AS THAT OF REGISTER UNDER TEST
	BEQ	5\$	:BRANCH IF ADDRESSES MATCH

M05

MAINDEC-11-DQKTA-A POP 11/6X MEM. MGMT. DIAG.
DQKTAAR.P11 07-FEB-77 10:30 T15

MACY11 27(1006) 07-FEB-77 10:37 PAGE 64
DUAL ADDRESS USER PAGE ADDRESS REGISTERS, ON LOADING

3256	023044	004737	044334		JSR	PC,DUALADR	: LOG AND REPORT ERRORS
3257	023050	005237	001176	5\$:	INC	\$TMP0	: SET FLAG WHEN ADDRESSES MATCH
3258	023054	162701	000002	6\$:	SUB	#2,R1	: POINT TO NEXT REGISTER
3259	023060	022701	177640		CMP	#UIPAR0,R1	: SEE IF ALL REGISTERS HAVE BEEN READ
3260	023064	101761			BLOS	4\$	: BRANCH IF MORE TO READ
3261	023066	062700	000002		ADD	#2,R0	: NOW LOAD NEXT REGISTER
3262	023072	022700	177656		CMP	#UIPAR7,R0	: SEE IF THERE ARE MORE REGISTERS TO TEST
3263	023076	103344			BHIS	3\$	: BRANCH IF MORE REGISTERS TO TEST
3264	023100	012737	022730	001110	MOV	#20\$,\$LPERR	: SET LOOP ON ERROR POINTER TO START OF TEST
3265	023106	005737	001432		TST	ERRCNT	: SEE IF THERE WERE ANY ERRORS
3266	023112	001404			BEQ	TST16	: BRANCH TO NEXT TEST IF NO ERRORS
3267	023114	013737	001432	001210	MOV	ERRCNT,\$TMP5	: SAVE NUMBER OF ERRORS FOR PAR OUT
3268	023122	104015			ERROR	15	: SUMMARY OF DUAL ADDRESSING ERRORS
3269							
3270							
3271							
3272							
3273							
3274							
3275							
3276							
3277							
3278							
3279							
3280							
3281							
3282							
3283							
3284	023124						
3285	023124	000004					
3286	023126	012737	023330	001446			
3287							
3288	023134	004737	044202				
3289	023140	012737	023166	001110	20\$:	JSR	PC,CLEANUP
3290	023146	012705	172300			MOV	#1\$,\$LPERR
3291	023152	004737	044164			MOV	#KIPDR0,R5
3292	023156	012700	172300			JSR	PC,CLRRÉG
3293	023162	012701	000010			MOV	#KIPDR0,R0
3294	023166	011002				MOV	#10,R1
3295	023170	001401			1\$:	MOV	(R0),R2
3296	023172	104014				BEQ	2\$
3297	023174	062700	000002			ERROR	14
3298	023200	077106			2\$:	ADD	#2,R0
3299						SUB	R1,1\$
3300							
3301							
3302							
3303							
3304	023202	012737	023214	001110		MOV	#3\$,\$LPERR
3305	023210	012700	172300			MOV	#KIPDR0,R0
3306	023214	012705	172300		3\$:	MOV	#KIPDR0,R5
3307	023220	004737	044164			JSR	PC,CLRRÉG
3308	023224	012701	172316			MOV	#KIPDR7,R1
3309	023230	012710	177777			MOV	#-1,(R0)
3310	023234	005037	001176		4\$:	CLR	\$TMP0
3311	023240	011102				MOV	(R1),R2

: TEST 16 DUAL ADDRESS KERNEL PAGE DESCRIPTOR REGISTERS, ON LOADING

: THIS TEST FIRST CLEARS ALL THE KERNEL PAGE DESCRIPTOR REGISTERS
: AND CHECKS TO SEE THAT THEY EACH HOLD ZERO. THEN, STARTING WITH
: DESCRIPTOR REGISTER ZERO, ONE REGISTER AT A TIME IS
: LOADED WITH A NEGATIVE ONE. ALL KERNEL DESCRIPTOR REGISTERS
: ARE NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
: INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN AT
: THE END OF THIS TEST.
: ALL ERRORS IN THIS TEST ARE REPORTED AND ANALYZED BY SUBROUTINE
: "DUALADR".

: TST16:
: SCOPE
: MOV #TST17,NXTTST : SAVE STARTING ADDRESS OF NEXT
: : TEST FOR ESCAPE ON PARITY ERRORS
: : INITIALIZE ERROR LOCATIONS
: : SET LOOP ON ERROR POINTER TO 1\$
: : PUT ADDRESS OF FIRST PDR IN R5
: : CLEAR 8 REGISTERS POINTED TO BY R5
: : PUT ADDRESS OF FIRST PDR IN R0
: : BRANCH COUNT IS 8 DECIMAL
: : READ PDR TO R2
: : BRANCH IF PDR IS 0
: : PDR NOT ZERO
: : POINT TO NEXT REGISTER
: : BRANCH BACK TO 1\$ 7 TIMES
: :
: : NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE
: : REGISTER AND READING THE REST TO SEE ONLY THAT REGISTER
: : IS NON-ZERO. (DROPPED BITS WILL BE FOUND IN NEXT TEST.)
: :
: : SET LOOP ON ERROR POINTER TO 3\$
: : PUT ADDRESS OF FIRST PDR IN R0
: : LOAD STARTING ADDRESS INTO R5
: : CLEAR 8 REGISTERS POINTED TO BY R5
: : PUT KIPDR7 ADDRESS IN R1
: : LOAD REGISTER UNDER TEST
: : FLAG TO INDICATE THERE WAS A MATCH
: : READ ALL REGISTERS

N05

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG.
DOKTAA.P11 07-FEB-77 10:30 T16

MACY11 27(1006) 07-FEB-77 10:37 PAGE 65
DUAL ADDRESS KERNEL PAGE DESCRIPTOR REGISTERS, ON LOADING

3312	023242	001406		BEQ	6\$		: BRANCH IF REGISTER IS 0
3313	023244	020001		CMP	R0,R1		: IS ADDRESS OF NON-ZERO REGISTER SAME
3314							: AS THAT OF REGISTER UNDER TEST
3315	023246	001402		BEQ	5\$		: BRANCH IF ADDRESSES MATCH
3316	023250	004737	044334	JSR	PC,DUALADR		: LOG AND REPORT ERRORS
3317	023254	005237	001176	INC	\$TMP0		: SET FLAG WHEN ADDRESSES MATCH
3318	023260	162701	000002	SUB	#2,R1		: POINT TO NEXT REGISTER
3319	023264	022701	172300	CMP	#KIPDR0,R1		: SEE IF ALL REGISTERS HAVE BEEN READ
3320	023270	101761		BLOS	4\$		: BRANCH IF MORE TO READ
3321	023272	062700	000002	ADD	#2,R0		: NOW LOAD NEXT REGISTER
3322	023276	022700	172316	CMP	#KIPDR7,R0		: SEE IF THERE ARE MORE REGISTERS TO TEST
3323	023302	103344		BHIS	3\$		: BRANCH IF MORE REGISTERS TO TEST
3324	023304	012737	023134 001110	MOV	#20\$,SLPERR		: SET LOOP ON ERROR POINTER TO START OF TEST
3325	023312	005737	001432	TST	ERRCNT		: SEE IF THERE WERE ANY ERRORS
3326	023316	001404		BEQ	TST17		: BRANCH TO NEXT TEST IF NO ERRORS
3327	023320	013737	001432 001210	MOV	ERRCNT,\$TMP5		: SAVE NUMBER OF ERRORS FOR PDR OUT
3328	023326	104015		ERROR	15		: SUMMARY OF DUAL ADDRESSING ERRORS

: TEST 17 DUAL ADDRESS USER PAGE DESCRIPTOR REGISTERS, ON LOADING

: THIS TEST FIRST CLEARS ALL THE USER PAGE DESCRIPTOR REGISTERS
: AND CHECKS TO SEE THAT THEY EACH HOLD ZERO. THEN, STARTING WITH
: DESCRIPTOR REGISTER ZERO, ONE REGISTER AT A TIME IS
: LOADED WITH A NEGATIVE ONE. ALL USER DESCRIPTOR REGISTERS
: ARE NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
: INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN AT
: THE END OF THIS TEST.

: ALL ERRORS IN THIS TEST ARE REPORTED AND ANALYZED BY SUBROUTINE
: "DUALADR".

: TST17.

3345	023330			SCOPE			
3346	023330	000004		MOV	#TST20,NXTTST		: SAVE STARTING ADDRESS OF NEXT
3347	023332	012737	023534 001446				: TEST FOR ESCAPE ON PARITY ERRORS
3348							: INITIALIZE ERROR LOCATIONS
3349	023340	004737	044202	20\$: JSR	PC,CLEANUP		: SET LOOP ON ERROR POINTER TO 1\$
3350	023344	012737	023372 001110	MOV	#1\$,SLPERR		: PUT ADDRESS OF FIRST PDR IN R5
3351	023352	012705	177600	MOV	#UIPDR0,R5		: CLEAR 8 REGISTERS POINTED TO BY R5
3352	023356	004737	044164	JSR	PC,CLRREG		: PUT ADDRESS OF FIRST PDR IN R0
3353	023362	012700	177600	MOV	#UIPDR0,R0		: BRANCH COUNT IS 8 DECIMAL
3354	023366	012701	000010	MOV	#10,R1		: READ PDR TO R2
3355	023372	011002		1\$: MOV	(R0),R2		: BRANCH IF PDR IS 0
3356	023374	001401		BEQ	2\$		: PDR NOT ZERO
3357	023376	104014		ERROR	14		: POINT TO NEXT REGISTER
3358	023400	062700	000002	2\$: ADD	#2,R0		: BRANCH BACK TO 1\$ 7 TIMES
3359	023404	077106		SUB	R1,1\$		
3360							
3361							
3362							
3363							
3364							
3365	023406	012737	023420 001110	MOV	#3\$,SLPERR		: SET LOOP ON ERROR POINTER TO 3\$
3366	023414	012700	177600	MOV	#UIPDR0,P0		: PUT ADDRESS OF FIRST PDR IN R0
3367	023420	012705	177600	3\$: MOV	#UIPDR0,R5		: LOAD STARTING ADDRESS INTO R5

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

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG.
DOKTAA.P11 07-FEB-77 10:30 T17

MACY11 27(1006) 07-FEB-77 10:37 PAGE 66
DUAL ADDRESS USER PAGE DESCRIPTOR REGISTERS, ON LOADING

3368	023424	004737	044164		JSR	PC,CLRREG	:CLEAR 8 REGISTERS POINTED TO BY R5
3369	023430	012701	177616		MOV	#UIPDR7,R1	:PUT UIPDR7 ADDRESS IN R1
3370	023434	012710	177777		MOV	#-1,(R0)	:LOAD REGISTER UNDER TEST
3371	023440	005037	001176	4\$:	CLR	\$TMPD	:FLAG TO INDICATE THERE WAS A MATCH
3372	023444	011102			MOV	(R1),R2	:READ ALL REGISTERS
3373	023446	001406			BEQ	6\$	:BRANCH IF REGISTER IS 0
3374	023450	020001			CMP	R0,R1	:IS ADDRESS OF NON-ZERO REGISTER SAME
3375							:AS THAT OF REGISTER UNDER TEST
3376	023452	001402			BEQ	5\$	:BRANCH IF ADDRESSES MATCH
3377	023454	004737	044334		JSR	PC,DUALADR	:LOG AND REPORT ERRORS
3378	023460	005237	001176	5\$:	INC	\$TMPD	:SET FLAG WHEN ADDRESSES MATCH
3379	023464	162701	000002	6\$:	SUB	#2,R1	:POINT TO NEXT REGISTER
3380	023470	022701	177600		CMP	#UIPDR0,R1	:SEE IF ALL REGISTERS HAVE BEEN READ
3381	023474	101761			BLOS	4\$	:BRANCH IF MORE TO READ
3382	023476	062700	000002		ADD	#2,R0	:NOW LOAD NEXT REGISTER
3383	023502	022700	177616		CMP	#UIPDR7,R0	:SEE IF THERE ARE MORE REGISTERS TO TEST
3384	023506	103344			BHIS	3\$	:BRANCH IF MORE REGISTERS TO TEST
3385	023510	012737	023340	001110	MOV	#20\$,\$LPERR	:SET LOOP ON ERROR POINTER TO START OF TEST
3386	023516	005737	001432		TST	ERRCNT	:SEE IF THERE WERE ANY ERRORS
3387	023522	001404			BEQ	TST20	:BRANCH TO NEXT TEST IF NO ERRORS
3388	023524	013737	001432	001210	MOV	ERRCNT,\$TMP5	:SAVE NUMBER OF ERRORS FOR PDR OUT
3389	023532	104015			ERROR	15	:SUMMARY OF DUAL ADDRESSING ERRORS

SBTTL

TEST FOR BAD READ/WRITE BITS IN P.A.R.'S & P.D.R.'S
THESE NEXT FOUR (4) TESTS CHECK FOR BAD BITS IN THE MEMORY CHIPS
THAT MAKE UP THE PAR'S AND PDR'S. THE REGISTERS ARE LOADED WITH
ZERO AND MODIFIED BY "401" UNTIL 177777 IS REACHED IN EACH ONE
(THE NUMBER 401 WAS CHOSEN FOR FASTER RUN-TIME). THE BITS THAT
ARE NOT IMPLEMENTED OR THAT ARE NOT READ/WRITE ARE MASKED OUT
OF THE DATA COMPARE. A LOG OF MULTIPLE ERRORS IS KEPT FOR EACH
GROUP AND IT IS REPORTED AT THE CONCLUSION OF EACH TEST.

*TEST 20 COUNT PATTERN IN KERNEL PAGE ADDRESS REGISTERS

THIS TEST RUNS A COUNT PATTERN THRU THE KERNEL PAGE ADDRESS
REGISTERS. SINCE BITS (15:12) ARE NOT IMPLEMENTED THEY ARE
MASKED OUT OF THE DATA COMPARE. THESE BITS ARE STILL SENT
TO THE ADDRESS REGISTER TO FIND BAD ETCHES. IF THE COUNT PATTERN
DOES NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS, DATA
PATTERN, AND BAD DATA ARE REPORTED. AT THE END OF THE TEST A
SUMMARY OF THE ERRORS IS GIVEN.

ALL ERRORS FOUND BY THIS TEST ARE REPORTED AND ANALYZED BY
SUBROUTINE "PARCOUNT".

*TST20:

3415	023534				SCOPE		
3416	023534	000004			MOV	#TST21,NXTTST	:SAVE STARTING ADDRESS OF NEXT
3417	023536	012737	023666	001446			:TEST FOR ESCAPE ON PARITY ERRORS
3418							
3419	023544	012737	000012	001212	MOV	#12,\$TIMES	:DO 12 ITERATIONS
3420	023552	004737	044202		JSR	PC,CLEANUP	:INITIALIZE ERROR LOCATIONS
3421	023556	012737	023572	001110	MOV	#2\$,\$LPERR	:SET LOOP ON ERROR POINTER TO 1\$
3422	023564	005001			CLR	R1	:CLEAR REGISTER TO HOLD COUNT PATTERN
3423	023566	012700	172340		MOV	#KIPAR0,R0	:PUT ADDRESS OF FIRST REGISTER 'S R0

C06

 MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG.
 DOKTAA.P11 07-FEB-77 10:30 T20

 MACY11 27(1006) 07-FEB-77 10:37 PAGE 67
 COUNT PATTERN IN KERNEL PAGE ADDRESS REGISTERS

3424	023572	010110		2\$:	MOV	R1,(R0)	;LOAD COUNT INTO REGISTER
3425	023574	011002			MOV	(R0),R2	;READ REGISTER BACK TO R2
3426	023576	010104			MOV	R1,R4	;PUT PATTERN IS R4
3427	023600	042704	170000		BIC	#170000,R4	;CLEAR BITS NOT FOUND IN REGISTER.
3428	023604	020402			CMP	R4,R2	;SEE IF DATA MATCHES PATTERN
3429	023606	001402			BEQ	3\$	;BRANCH IF DATA IS GOOD
3430	023610	004737	044410		JSR	PC,PARCOUNT	;LOG AND REPORT COUNT ERROR
3431	023614	062700	000002	3\$:	ADD	#2,R0	;POINT TO NEXT REGISTER
3432	023620	022700	172356		CMP	#KIPAR7,R0	;SEE IF YOU PASSED THE KIPAR7 PAR
3433	023624	103362			BHIS	2\$	;BRANCH IF MORE PAR'S TO TEST
3434	023626	022701	177777		CMP	#177777,R1	;SEE IF COUNT HAS REACHED 177777
3435	023632	001403			BEQ	4\$	;BRANCH IF COUNT IS 177777
3436	023634	062701	000401		ADD	#401,R1	;INCREASE COUNT PATTERN
3437	023640	000752			BR	1\$	;BRANCH TO CONTINUE TEST
3438	023642	012737	023552 001110	4\$:	MOV	#20\$,\$LPERR	;SET LOOP POINTER TO START OF TEST
3439	023650	005737	001432		TST	ERRCNT	;SEE IF THERE WERE ANY ERRORS
3440	023654	001404			BEQ	TST21	;BRANCH TO NEXT TEST IF NO ERRORS
3441	023656	013737	001432 001210		MOV	ERRCNT,\$TMP5	;SAVE NUMBER OF ERRORS FOR TYPEOUT
3442	023664	104016			ERROR	16	;SUMMARY OF COUNT PATTERN FAILURES

 ;TEST 21 COUNT PATTERN IN USER PAGE ADDRESS REGISTERS

 THIS TEST RUNS A COUNT PATTERN THRU THE USER PAGE ADDRESS
 REGISTERS. SINCE BITS <15:12> ARE NOT IMPLEMENTED THEY ARE
 MASKED OUT OF THE DATA COMPARE. THESE BITS ARE STILL SENT
 TO THE ADDRESS REGISTER TO FIND BAD ETCHES. IF THE COUNT PATTERN
 DOES NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS, DATA
 PATTERN, AND BAD DATA ARE REPORTED. AT THE END OF THE TEST A
 SUMMARY OF THE ERRORS IS GIVEN.

 ALL ERRORS FOUND BY THIS TEST ARE REPORTED AND ANALYZED BY
 SUBROUTINE "PARCOUNT".

 TST21:

3460	023666	000004			SCOPE		
3461	023666	012737	024020 001446		MOV	#TST22,NXTTST	;SAVE STARTING ADDRESS OF NEXT
3462	023670						;TEST FOR ESCAPE ON PARITY ERRORS
3463							;DO 12 ITERATIONS
3464	023676	012737	000012 001212	20\$:	MOV	#12,\$TIMES	;INITIALIZE ERROR LOCATIONS
3465	023704	004737	044202		JSR	PC,CLEANUP	;SET LOOP ON ERROR POINTER TO 1\$
3466	023710	012737	023724 001110		MOV	#2\$,\$LPERR	;CLEAR REGISTER TO HOLD COUNT PATTERN
3467	023716	005001			CLR	R1	;PUT ADDRESS OF FIRST REGISTER IS R0
3468	023720	012700	177640	1\$:	MOV	#UIPAR0,R0	;LOAD COUNT INTO REGISTER
3469	023724	010110		2\$:	MOV	R1,(R0)	;READ REGISTER BACK TO R2
3470	023726	011002			MOV	(R0),R2	;PUT PATTERN IS R4
3471	023730	010104			MOV	R1,R4	;CLEAR BITS NOT FOUND IN REGISTER.
3472	023732	042704	170000		BIC	#170000,R4	;SEE IF DATA MATCHES PATTERN
3473	023736	020402			CMP	R4,R2	;BRANCH IF DATA IS GOOD
3474	023740	001402			BEQ	3\$	;LOG AND REPORT COUNT ERROR
3475	023742	004737	044410	3\$:	JSR	PC,PARCOUNT	;POINT TO NEXT REGISTER
3476	023746	062700	000002		ADD	#2,R0	;SEE IF YOU PASSED THE UIPAR7 PAR
3477	023752	022700	177656		CMP	#UIPAR7,R0	;BRANCH IF MORE PAR'S TO TEST
3478	023756	103362			BHIS	2\$	;SEE IF COUNT HAS REACHED 177777
3479	023760	022701	177777		CMP	#177777,R1	

MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG.
DQKTA.P11 07-FEB-77 10:30 T21

MACY11 27(1006) 07-FEB-77 10:37 PAGE 68
COUNT PATTERN IN USER PAGE ADDRESS REGISTERS

3480	023764	001403				BEQ	4\$	: BRANCH IF COUNT IS 177777
3481	023766	062701	000401			ADD	#401,R1	: INCREASE COUNT PATTERN
3482	023772	000752				BR	1\$	: BRANCH TO CONTINUE TEST
3483	023774	012737	023704	001110	4\$:	MOV	#20\$,SLPERR	: SET LOOP POINTER TO START OF TEST
3484	024002	005737	001432			TST	ERRCNT	: SEE IF THERE WERE ANY ERRORS
3485	024006	001404				BEQ	TST22	: BRANCH TO NEXT TEST IF NO ERRORS
3486	024010	013737	001432	001210		MOV	ERRCNT,\$TMP5	: SAVE NUMBER OF ERRORS FOR TYPEOUT
3487	024016	104016				ERROR	16	: SUMMARY OF COUNT PATTERN FAILURES

: TEST 22 COUNT PATTERN IN KERNEL PAGE DESCRIPTOR REGISTERS

:
: THIS TEST RUNS A COUNT PATTERN THRU THE KERNEL PAGE DESCRIPTOR
: REGISTERS. SINCE BITS <15>,<07>,<05:04> AND <00> ARE NOT IMPLEMENTED
: AND BITS <06> CANNOT BE SET BY DIRECT LOAD, THEY ARE MASKED
: OUT OF THE DATA COMPARE. THESE BITS ARE STILL SENT TO THE
: DESCRIPTOR REGISTER TO FIND BAD ETCHES. IF THE COUNT PATTERN
: DOES NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS, DATA
: PATTERN, AND BAD DATA ARE REPORTED. AT THE END OF THE TEST A
: SUMMARY OF THE ERRORS IS GIVEN.
: *****

: TST22:

3502	024020	000004				SCOPE		
3503	024020	012737	024152	001446		MOV	#TST23,NXTTST	: SAVE STARTING ADDRESS OF NEXT
3504	024022							: TEST FOR ESCAPE ON PARITY ERRORS
3505								: DO 12 ITERATIONS
3506	024030	012737	000012	001212		MOV	#12,\$TIMES	: INITIALIZE ERROR LOCATIONS
3507	024036	004737	044202		20\$:	JSR	PC,CLEANUP	: SET LOOP ON ERROR POINTER TO 1\$
3508	024042	012737	024056	001110		MOV	#2\$,SLPERR	: CLEAR REGISTER TO HOLD COUNT PATTERN
3509	024050	005001				CLR	R1	: PUT ADDRESS OF FIRST REGISTER IS R0
3510	024052	012700	172300		1\$:	MOV	#KIPDR0,R0	: LOAD COUNT INTO REGISTER
3511	024056	010110			2\$:	MOV	R1,(R0)	: READ REGISTER BACK TO R2
3512	024060	011002				MOV	(R0),R2	: PUT PATTERN IS R4
3513	024062	010104				MOV	R1,R4	: CLEAR BITS NOT FOUND IN REGISTER.
3514	024064	042704	100361			BIC	#100361,R4	: SEE IF DATA MATCHES PATTERN
3515	024070	020402				CMP	R4,R2	: BRANCH IF DATA IS GOOD
3516	024072	001402				BEQ	3\$	: LOG AND REPORT COUNT ERROR
3517	024074	004737	044410			JSR	PC,PARCOUNT	: POINT TO NEXT REGISTER
3518	024100	062700	000002		3\$:	ADD	#2,R0	: SEE IF YOU PASSED THE KIPDR7 PDR
3519	024104	022700	172316			CMP	#KIPDR7,R0	: BRANCH IF MORE PDR'S TO TEST
3520	024110	103362				BHIS	2\$	: SEE IF COUNT HAS REACHED 177777
3521	024112	022701	177777			CMP	#177777,R1	: BRANCH IF COUNT IS 177777
3522	024116	001403				BEQ	4\$	: INCREASE COUNT PATTERN
3523	024120	062701	000401			ADD	#401,R1	: BRANCH TO CONTINUE TEST
3524	024124	000752				BR	1\$	: SET LOOP POINTER TO START OF TEST
3525	024126	012737	024036	001110	4\$:	MOV	#20\$,SLPERR	: SEE IF THERE WERE ANY ERRORS
3526	024134	005737	001432			TST	ERRCNT	: BRANCH TO NEXT TEST IF NO ERRORS
3527	024140	001404				BEQ	TST23	: SAVE NUMBER OF ERRORS FOR TYPEOUT
3528	024142	013737	001432	001210		MOV	ERRCNT,\$TMP5	: SUMMARY OF COUNT PATTERN FAILURES
3529	024150	104016				ERROR	16	

: TEST 23 COUNT PATTERN IN USER PAGE DESCRIPTOR REGISTERS

:
: THIS TEST RUNS A COUNT PATTERN THRU THE USER PAGE DESCRIPTOR

E06

MAINDEC-11-DQKTA-A POP 11/6X MEM. MGMT. DIAG.
DQKTA.P11 07-FEB-77 10:30 T23

MACY11 27(1006) 07-FEB-77 10:37 PAGE 69
COUNT PATTERN IN USER PAGE DESCRIPTOR REGISTERS

REGISTERS. SINCE BITS <15>, <07>, <05:04>, AND <00> ARE NOT IMPLEMENTED AND BITS <06> CANNOT BE SET BY DIRECT LOAD, THEY ARE MASKED OUT OF THE DATA COMPARE. THESE BITS ARE STILL SENT TO THE DESCRIPTOR REGISTER TO FIND BAD ETCHES. IF THE COUNT PATTERN DOES NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS, DATA PATTERN, AND BAD DATA ARE REPORTED. AT THE END OF THE TEST A SUMMARY OF THE ERRORS IS GIVEN.

TST23:

SCOPE	MOV	#TST24,NXTTST	SAVE STARTING ADDRESS OF NEXT
			TEST FOR ESCAPE ON PARITY ERRORS
			DO 12 ITERATIONS
			INITIALIZE ERROR LOCATIONS
			SET LOOP ON ERROR POINTER TO 1\$
			CLEAR REGISTER TO HOLD COUNT PATTERN
			PUT ADDRESS OF FIRST REGISTER IS R0
			LOAD COUNT INTO REGISTER
			READ REGISTER BACK TO R2
			PUT PATTERN IS R4
			CLEAR BITS NOT FOUND IN REGISTER.
			SEE IF DATA MATCHES PATTERN
			BRANCH IF DATA IS GOOD
			LOG AND REPORT COUNT ERROR
			POINT TO NEXT REGISTER
			SEE IF YOU PASSED THE UIPDR7 PDR
			BRANCH IF MORE PDR'S TO TEST
			SEE IF COUNT HAS REACHED 177777
			BRANCH IF COUNT IS 177777
			INCREASE COUNT PATTERN
			BRANCH TO CONTINUE TEST
			SET LOOP POINTER TO START OF TEST
			SEE IF THERE WERE ANY ERRORS
			BRANCH TO NEXT TEST IF NO ERRORS
			SAVE NUMBER OF ERRORS FOR TYPEOUT
			SUMMARY OF COUNT PATTERN FAILURES

SBTTL TEST FOR CORRECT BYTE ADDRESSING OF P.A.R.'S & P.D.R.'S
THESE NEXT FOUR (4) TESTS ARE USED TO TEST THE LOGIC THAT
ALLOWS BYTE ADDRESSING OF THE PAR'S AND PDR'S. IN EACH
CASE REGISTER 0 IS USED, SINCE IT IS REALLY
THE WRITE PULSE TO THE REGISTER SET THAT IS BEING TESTED.
IT IS ASSUMED THAT IF ONE REGISTER OF A GROUP FUNCTIONS
PROPERLY THAT THEY ALL WILL, SINCE ALL REGISTERS HAVE BEEN
BIT TESTED EARLIER.

TEST 24 BYTE ADDRESSING OF KERNEL PAGE ADDRESS REGISTERS

THIS TEST CHECKS THE 'WRITE PAR LOW' AND 'WRITE PAR HIGH'
SIGNAL GENERATION FOR KERNEL PAGE ADDRESS REGISTERS.
BOTH SIGNALS ARE CONTROLLED BY THE WRITE REG. DECODE LOGIC.

TST24:

3536					
3537					
3538					
3539					
3540					
3541					
3542					
3543					
3544					
3545	024152				
3546	024152	000004			
3547	024154	012737	024304	001446	
3548					
3549	024162	012737	000012	001212	
3550	024170	004737	044202		20\$:
3551	024174	012737	024210	001110	
3552	024202	005001			
3553	024204	012700	177600		1\$:
3554	024210	010110			2\$:
3555	024212	011002			
3556	024214	010104			
3557	024216	042701	100361		
3558	024222	020402			
3559	024224	001402			
3560	024226	004737	044410		
3561	024232	062700	000002		3\$:
3562	024236	022700	177616		
3563	024242	103362			
3564	024244	022701	177777		
3565	024250	001403			
3566	024252	062701	000401		
3567	024256	000752			
3568	024260	012737	024170	001110	4\$:
3569	024266	005737	001432		
3570	024272	001404			
3571	024274	013737	001432	001210	
3572	024302	104016			
3573					
3574					
3575					
3576					
3577					
3578					
3579					
3580					
3581					
3582					
3583					
3584					
3585					
3586					
3587					
3588					
3589					
3590					
3591	024304				

F06

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG.
DOKTAA.P11 07-FEB-77 10:30 T24

MACY11 27(1006) 07-FEB-77 10:37 PAGE 70
BYTE ADDRESSING OF KERNEL PAGE ADDRESS REGISTERS

```

3592 024304 000004 SCOPE
3593 024306 012737 024416 001446 MOV #TST25,NXTTST ;SAVE STARTING ADDRESS OF NEXT
3594 ;TEST FOR ESCAPE ON PARITY ERRORS
3595 024314 012737 024332 001110 20$: MOV #11$,SLPERR ;SET LOOP ON ERROR POINTER TO 11$
3596 024322 012702 000016 MOV #16,R2 ;PUT EXPECTED DATA IN R2
3597 024326 012700 172340 MOV #KIPAR0,R0 ;PUT ADDRESS OF REGISTER IN R0
3598 024332 005037 172340 11$: CLR KIPAR0 ;CLEAR THE REGISTER UNDER TEST
3599 024336 112710 172016 MOV #172016,(R0) ;LOAD LOWER BYTE OF REGISTER
3600 024342 013701 172340 MOV KIPAR0,R1 ;READ REGISTER INTO R1
3601 024346 020102 CMP R1,R2 ;SEE IF ONLY LOWER BYTE WAS WRITTEN
3602 024350 001401 BEQ 1$ ;BRANCH IF DATA MATCHES
3603 024352 104017 ERROR 17 ;DIDN'T LOAD CORRECT BYTE
3604 024354 012737 024372 001110 1$: MOV #12$,SLPERR ;SET LOOP ON ERROR POINTER TO 12$
3605 024356 012700 172341 MOV #KIPAR0+1,R0 ;POINT TO UPPER BYTE OF REGISTER
3606 024358 012702 007416 MOV #007416,R2 ;LOAD EXPECTED DATA IN R2
3607 024372 112710 000017 12$: MOV #17,(R0) ;WRITE THE UPPER BYTE OF REGISTER
3608 024376 013701 172340 MOV KIPAR0,R1 ;READ REGISTER INTO R1
3609 024402 020102 CMP R1,R2 ;SEE IF REGISTER HOLDS CORRECT DATA
3610 024404 001401 BEQ 2$ ;BRANCH TO EXIT IF DATA RIGHT
3611 024406 104017 ERROR 17 ;DIDN'T LOAD CORRECT BYTE
3612
3613 024410 012737 024314 001110 2$: MOV #20$,SLPERR ;SET LOOP POINTER TO START OF TEST
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647

```

```

*****
;TEST 25      BYTE ADDRESSING OF USER PAGE ADDRESS REGISTERS
;
;THIS TEST CHECKS THE 'WRITE PAR LOW' AND 'WRITE PAR HIGH'
;SIGNAL GENERATION FOR USER PAGE ADDRESS REGISTERS.
;BOTH SIGNALS ARE CONTROLLED BY THE WRITE REG. DECODE LOGIC.
*****

```

```

3626 024416 000004
3627 024416 012737 024530 001446 SCOPE
3628 024420 012737 024530 001446 MOV #TST26,NXTTST ;SAVE STARTING ADDRESS OF NEXT
3629 ;TEST FOR ESCAPE ON PARITY ERRORS
3630 024426 012737 024444 001110 20$: MOV #11$,SLPERR ;SET LOOP ON ERROR POINTER TO 11$
3631 024434 012702 000016 MOV #16,R2 ;PUT EXPECTED DATA IN R2
3632 024440 012700 177640 MOV #UIPAR0,R0 ;PUT ADDRESS OF REGISTER IN R0
3633 024444 005037 177640 11$: CLR UIPAR0 ;CLEAR THE REGISTER UNDER TEST
3634 024450 112710 172016 MOV #172016,(R0) ;LOAD LOWER BYTE OF REGISTER
3635 024454 013701 177640 MOV UIPAR0,R1 ;READ REGISTER INTO R1
3636 024460 020102 CMP R1,R2 ;SEE IF ONLY LOWER BYTE WAS WRITTEN
3637 024462 001401 BEQ 1$ ;BRANCH IF DATA MATCHES
3638 024464 104017 ERROR 17 ;DIDN'T LOAD CORRECT BYTE
3639 024466 012737 024504 001110 1$: MOV #12$,SLPERR ;SET LOOP ON ERROR POINTER TO 12$
3640 024474 012700 177641 MOV #UIPAR0+1,R0 ;POINT TO UPPER BYTE OF REGISTER
3641 024500 012702 007416 MOV #007416,R2 ;LOAD EXPECTED DATA IN R2
3642 024504 112710 000017 12$: MOV #17,(R0) ;WRITE THE UPPER BYTE OF REGISTER
3643 024510 013701 177640 MOV UIPAR0,R1 ;READ REGISTER INTO R1
3644 024514 020102 CMP R1,R2 ;SEE IF REGISTER HOLDS CORRECT DATA
3645 024516 001401 BEQ 2$ ;BRANCH TO EXIT IF DATA RIGHT
3646 024520 104017 ERROR 17 ;DIDN'T LOAD CORRECT BYTE
3647

```

```

3648 024522 012737 024426 001110 25:  MOV    #20$,SLPERR    ;SET LOOP POINTER TO START OF TEST
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659 024530
3660 024530 000004
3661 024532 012737 024642 001446
3662
3663 024540 012737 024556 001110 20$:  MOV    #11$,SLPERR    ;SAVE STARTING ADDRESS OF NEXT
3664 024546 012702 000016                MOV    #16,R2          ;TEST FOR ESCAPE ON PARITY ERRORS
3665 024552 012700 172300                MOV    #KIPDR0,R0       ;SET LOOP ON ERROR POINTER TO 11$
3666 024556 005037 172300                CLR     KIPDR0         ;PUT EXPECTED DATA IN R2
3667 024562 112710 172016 11$:  MOV     #172016,(R0)    ;PUT ADDRESS OF REGISTER IN R0
3668 024566 013701 172300                MOV     KIPDR0,R1     ;CLEAR THE REGISTER UNDER TEST
3669 024572 020102                CMP     R1,R2         ;LOAD LOWER BYTE OF REGISTER
3670 024574 001401                BEQ     1$            ;READ REGISTER INTO R1
3671 024576 104017                ERROR   17           ;SEE IF ONLY LOWER BYTE WAS WRITTEN
3672 024600 012737 024616 001110 1$:  MOV     #12$,SLPERR    ;BRANCH IF DATA MATCHES
3673 024606 012700 172301                MOV     #KIPDR0+1,R0  ;DIDN'T LOAD CORRECT BYTE
3674 024612 012702 007416                MOV     #007416,R2    ;SET LOOP ON ERROR POINTER TO 12$
3675 024616 112710 000017 12$:  MOV     #17,(R0)       ;POINT TO UPPER BYTE OF REGISTER
3676 024622 013701 172300                MOV     KIPDR0,R1     ;LOAD EXPECTED DATA IN R2
3677 024626 020102                CMP     R1,R2         ;WRITE THE UPPER BYTE OF REGISTER
3678 024630 001401                BEQ     2$            ;READ REGISTER INTO R1
3679 024632 104017                ERROR   17           ;SEE IF REGISTER HOLDS CORRECT DATA
3680
3681 024634 012737 024540 001110 2$:  MOV     #20$,SLPERR    ;BRANCH TO EXIT IF DATA RIGHT
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694 024642
3695 024642 000004
3696 024644 012737 024754 001446
3697
3698 024652 012737 024670 001110 20$:  MOV     #11$,SLPERR    ;SAVE STARTING ADDRESS OF NEXT
3699 024660 012702 000016                MOV     #16,R2          ;TEST FOR ESCAPE ON PARITY ERRORS
3700 024664 012700 177600                MOV     #UIPDRO,R0     ;SET LOOP ON ERROR POINTER TO 11$
3701 024670 005037 177600                CLR     UIPDR0         ;PUT EXPECTED DATA IN R2
3702 024674 112710 172016 11$:  MOV     #172016,(R0)    ;PUT ADDRESS OF REGISTER IN R0
3703 024700 013701 177600                MOV     UIPDR0,R1     ;CLEAR THE REGISTER UNDER TEST

```

 :TEST 26 BYTE ADDRESSING OF KERNEL PAGE DESCRIPTOR REGISTERS
 :
 : THIS TEST CHECKS THE 'WRITE PDR LOW' AND 'WRITE PDR HIGH'
 : SIGNAL GENERATION FOR KERNEL PAGE ADDRESS REGISTERS.
 : BOTH SIGNALS ARE CONTROLLED BY THE WRITE REG. DECODE LOGIC.
 :
 : *****
 :TST26:

 :TEST 27 BYTE ADDRESSING OF USER PAGE DESCRIPTOR REGISTERS
 :
 : THIS TEST CHECKS THE 'WRITE PDR LOW' AND 'WRITE PDR HIGH'
 : SIGNAL GENERATION FOR USER PAGE ADDRESS REGISTERS.
 : BOTH SIGNALS ARE CONTROLLED BY THE WRITE REG. DECODE LOGIC.
 :
 : *****
 :TST27:

H06

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG.
DOKTAA.P11 07-FEB-77 10:30 T27

MACY11 27(1006) 07-FEB-77 10:37 PAGE 72
BYTE ADDRESSING OF USER PAGE DESCRIPTOR REGISTERS

```

3704 024704 020102      CMP      R1,R2      ;SEE IF ONLY LOWER BYTE WAS WRITTEN
3705 024706 001401      BEQ      15          ;BRANCH IF DATA MATCHES
3706 024710 104017      ERROR    17          ;DIDN'T LOAD CORRECT BYTE
3707 024712 012737 024730 001110 15:  MOV      #12$,SLPERR      ;SET LOOP ON ERROR POINTER TO 12$
3708 024720 012700 177601      MOV      UIPOR0+1,R0      ;POINT TO UPPER BYTE OF REGISTER
3709 024724 012702 007416      MOV      #007416,R2      ;LOAD EXPECTED DATA IN R2
3710 024730 112710 000017      12$:  MOVB     #17,(R0)      ;WRITE THE UPPER BYTE OF REGISTER
3711 024734 013701 177600      MOV      UIPOR0,R1      ;READ REGISTER INTO R1
3712 024740 020102      CMP      R1,R2      ;SEE IF REGISTER HOLDS CORRECT DATA
3713 024742 001401      BEQ      25          ;BRANCH TO EXIT IF DATA RIGHT
3714 024744 104017      ERROR    17          ;DIDN'T LOAD CORRECT BYTE
3715
3716 024746 012737 024652 001110 25:  MOV      #20$,SLPERR      ;SET LOOP POINTER TO START OF TEST
3717
3718
3719      .SBTTL      DUAL ADDRESSING BETWEEN GROUPS OF REGISTERS
3720      *****
3721      *TEST 30      DUAL ADDRESSING FOR ALL PAR'S AND PDR'S
3722      *
3723      *      THIS TEST WILL ENSURE THAT THERE IS NO DUAL ADDRESSING BETWEEN
3724      *      GROUPS OF PAR'S AND PDR'S. THAT IS, WHEN YOU REFERENCE A KERNEL
3725      *      PAR YOU ARE REALLY REFERENCING THAT PAR. FIRST EACH
3726      *      PAR0 OR PDR0 IS LOADED WITH A UNIQUE NUMBER 0-6 AND
3727      *      THEN THEY ARE EACH CHECKED FOR THE CORRECT DATA.
3728      *      EACH GROUP HAS ALREADY BEEN CHECKED FOR DUAL ADDRESSING WITHIN
3729      *      ITS OWN GROUP, SO ONLY ONE REGISTER FROM EACH GROUP NEEDS TO
3730      *      BE TESTED HERE.
3731      *
3732      *****
3733      *ST30:
3734 024754 000004      SCOPE
3735 024756 012737 025056 001446      MOV      #TST31,NXTTST      ;SAVE STARTING ADDRESS OF NEXT
3736      ;TEST FOR ESCAPE ON PARITY ERRORS
3737 024764 012737 025000 001110 20$:  MOV      #1$,SLPERR      ;SET LOOP ON ERROR POINTER TO 1$
3738 024772 005000      CLR      R0          ;THIS WILL HOLD THE INDEX OF EACH REGISTER
3739      ;AND THE COUNT LOADED
3740 024774 012704 000004      MOV      #4,R4          ;THIS IS THE NUMBER OF TIMES TO DO THIS LOOP
3741 025000 010070 001452      1$:  MOV      R0,2PARTAB(R0)      ;LOAD PAR OR PDR WITH INDEX NUMBER
3742 025004 062700 000002      ADD      #2,R0          ;CHANGE INDEX TO POINT TO NEXT REGISTER
3743 025010 077405      SOB      R4,1$          ;BRANCH BACK 3 TIMES
3744 025012 012704 000004      MOV      #4,R4          ;DO NEXT LOOP 4 TIMES ALSO
3745 025016 012737 025030 001110      MOV      #10$,SLPERR      ;SET LOOP ON ERROR POINTER TO 10$
3746 025024 162700 000002      2$:  SUB      #2,R0          ;ADJUST INDEX FOR REGISTER DESIRED
3747 025030 017001 001452      10$:  MOV      2PARTAB(R0),R1      ;READ PAR OR PDR INTO R1
3748 025034 020001      CMP      R0,R1          ;SEE IF INDEX EQUALS DATA
3749 025036 001403      BEQ      3$          ;BRANCH IF CORRECT REGISTER WAS READ.
3750 025040 016002 001452      MOV      PARTAB(R0),R2      ;SAVE ADDRESS OF BAD REGISTER
3751 025044 104030      ERROR    30          ;DUAL ADDRESSING
3752 025046 077412      3$:  SOB      R4,2$          ;BRANCH BACK 3 TIMES
3753 025050 012737 024764 001110      MOV      #20$,SLPERR      ;SET LOOP ON ERROR POINTER TO START OF TEST
3754
3755
3756      .SBTTL      ***** ENTRY POINT 4 --- STARTING ADDRESS 214 *****
3757      .SBTTL      ***** RELOCATION AND ADDER TESTS *****
3758      *
3759      *      THIS GROUP OF TESTS CHECKS MEMORY MANAGEMENT'S RELOCATION ADDER

```

MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 73
DQKTA.P11 07-FEB-77 10:30

***** RELOCATION AND ADDER TESTS *****

3760 : * FIRST USING DESTINATION ONLY RELOCATION, THEN WITH FULL 18-BIT
3761 : * RELOCATION .
3762 : *
3763 : *

3766 : *TEST 31 18-BIT MAPPING ADDER TESTING

3767 : *
3768 : * THIS TEST USES 'DESTINATION ONLY' RELOCATION TO CHECK THE
3769 : * FULL ADD PROPERTIES OF THE RELOCATION ADDER. TWELVE PAIRS
3770 : * OF NUMBERS ARE ADDED, GENERATING PHYSICAL ADDRESSES ABOVE
3771 : * 12K.
3772 : *

3773 : *
3774 : *
3775 : *
3776 : *
3777 : *
3778 : *
3779 : *
3780 : *
3781 : *
3782 : *

3774 : *
3775 : *
3776 : *
3777 : *
3778 : *
3779 : *
3780 : *
3781 : *
3782 : *

3774 : *
3775 : *
3776 : *
3777 : *
3778 : *
3779 : *
3780 : *
3781 : *
3782 : *

3774 : *
3775 : *
3776 : *
3777 : *
3778 : *
3779 : *
3780 : *
3781 : *
3782 : *

3783 : *
3784 : *
3785 : *
3786 : *
3787 : *
3788 : *
3789 : *
3790 : *
3791 : *
3792 : *
3793 : *
3794 : *
3795 : *
3796 : *
3797 : *
3798 : *
3799 : *
3800 : *
3801 : *
3802 : *
3803 : *
3804 : *
3805 : *
3806 : *
3807 : *
3808 : *
3809 : *
3810 : *
3811 : *
3812 : *
3813 : *
3814 : *
3815 : *

3783 : *
3784 : *
3785 : *
3786 : *
3787 : *
3788 : *
3789 : *
3790 : *
3791 : *
3792 : *
3793 : *
3794 : *
3795 : *
3796 : *
3797 : *
3798 : *
3799 : *
3800 : *
3801 : *
3802 : *
3803 : *
3804 : *
3805 : *
3806 : *
3807 : *
3808 : *
3809 : *
3810 : *
3811 : *
3812 : *
3813 : *
3814 : *
3815 : *

3760 : *
3761 : *
3762 : *
3763 : *
3764 : *
3765 : *
3766 : *
3767 : *
3768 : *
3769 : *
3770 : *
3771 : *
3772 : *
3773 : *
3774 : *
3775 : *
3776 : *
3777 : *
3778 : *
3779 : *
3780 : *
3781 : *
3782 : *
3783 : *
3784 : *
3785 : *
3786 : *
3787 : *
3788 : *
3789 : *
3790 : *
3791 : *
3792 : *
3793 : *
3794 : *
3795 : *
3796 : *
3797 : *
3798 : *
3799 : *
3800 : *
3801 : *
3802 : *
3803 : *
3804 : *
3805 : *
3806 : *
3807 : *
3808 : *
3809 : *
3810 : *
3811 : *
3812 : *
3813 : *
3814 : *
3815 : *

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG.
DOKTAA.P11 07-FEB-77 10:30 T31

MACY11 27(1006) 07-FEB-77 10:37 PAGE 74
18-BIT MAPPING ADDER TESTING

3816	025312	020102				CMP	R1,R2	SEE IF DATA MATCHES PATTERN
3817	025314	001401				BEQ	25	BRANCH IF DATA MATCHES
3818	025316	104031				ERROR	31	RELOCATION FAILED
3819	025320	013737	001176	060000	25:	MOV	\$TMP0,2#60000	RESTORE ORIGINAL DATA TO TEST LOCATION
3820	025326	012737	025374	001110		MOV	#75,\$LPERR	SET LOOP ON ERROR POINTER TO 75
3821	025334	013737	062000	001176		MOV	2#62000,\$TMP0	SAVE DATA AT TEST LOCATION
3822	025342	012737	000610	172350		MOV	#610,2#KIPAR4	LOAD PAR4 WITH 610
3823	025350	012737	000610	001200		MOV	#610,\$TMP1	SAVE 610 FOR ERROR REPORT
3824	025356	012737	062000	001202		MOV	#62000,\$TMP2	SAVE 62000 FOR ERROR REPORT
3825	025364	012700	101000			MOV	#101000,R0	PUT VIRTUAL ADDRESS IN R0
3826	025370	012701	125203			MOV	#125203,R1	PUT DATA PATTERN IN R1
3827	025374	052737	000400	177572	75:	BIS	#BIT8,2#MMR0	TURN ON DESTINATION ONLY RELOCATION
3828	025402	010110				MOV	R1,(R0)	TRY TO LOAD DATA PATTERN INTO 62000
3829	025404	013702	062000			MOV	2#62000,R2	READ (62000) INTO R2
3830	025410	000005				RESET		CLEAR MMR0
3831	025412	020102				CMP	R1,R2	SEE IF DATA MATCHES PATTERN
3832	025414	001401				BEQ	85	BRANCH IF DATA MATCHES
3833	025416	104031				ERROR	31	RELOCATION FAILED
3834	025420	013737	001176	062000	85:	MOV	\$TMP0,2#62000	RESTORE ORIGINAL DATA TO TEST LOCATION
3835	025426	012737	025474	001110		MOV	#95,\$LPERR	SET LOOP ON ERROR POINTER TO 95
3836	025434	013737	062700	001176		MOV	2#62000,\$TMP0	SAVE DATA AT TEST LOCATION
3837	025442	012737	000514	172350		MOV	#514,2#KIPAR4	LOAD PAR4 WITH 514
3838	025450	012737	000514	001200		MOV	#514,\$TMP1	SAVE 514 FOR ERROR REPORT
3839	025456	012737	062000	001202		MOV	#62000,\$TMP2	SAVE 62000 FOR ERROR REPORT
3840	025464	012700	110400			MOV	#110400,R0	PUT VIRTUAL ADDRESS IN R0
3841	025470	012701	125204			MOV	#125204,R1	PUT DATA PATTERN IN R1
3842	025474	052737	000400	177572	95:	BIS	#BIT8,2#MMR0	TURN ON DESTINATION ONLY RELOCATION
3843	025502	010110				MOV	R1,(R0)	TRY TO LOAD DATA PATTERN INTO 62000
3844	025504	013702	062000			MOV	2#62000,R2	READ (62000) INTO R2
3845	025510	000005				RESET		CLEAR MMR0
3846	025512	020102				CMP	R1,R2	SEE IF DATA MATCHES PATTERN
3847	025514	001401				BEQ	105	BRANCH IF DATA MATCHES
3848	025516	104031				ERROR	31	RELOCATION FAILED
3849	025520	013737	001176	062000	105:	MOV	\$TMP0,2#62000	RESTORE ORIGINAL DATA TO TEST LOCATION
3850	025526	012737	025574	001110		MOV	#115,\$LPERR	SET LOOP ON ERROR POINTER TO 115
3851	025534	013737	062000	001176		MOV	2#62000,\$TMP0	SAVE DATA AT TEST LOCATION
3852	025542	012737	000504	172350		MOV	#504,2#KIPAR4	LOAD PAR4 WITH 504
3853	025550	012737	000504	001200		MOV	#504,\$TMP1	SAVE 504 FOR ERROR REPORT
3854	025556	012737	062000	001202		MOV	#62000,\$TMP2	SAVE 62000 FOR ERROR REPORT
3855	025564	012700	111400			MOV	#111400,R0	PUT VIRTUAL ADDRESS IN R0
3856	025570	012701	125205			MOV	#125205,R1	PUT DATA PATTERN IN R1
3857	025574	052737	000400	177572	115:	BIS	#BIT8,2#MMR0	TURN ON DESTINATION ONLY RELOCATION
3858	025602	010110				MOV	R1,(R0)	TRY TO LOAD DATA PATTERN INTO 62000
3859	025604	013702	062000			MOV	2#62000,R2	READ (62000) INTO R2
3860	025610	000005				RESET		CLEAR MMR0
3861	025612	020102				CMP	R1,R2	SEE IF DATA MATCHES PATTERN
3862	025614	001401				BEQ	125	BRANCH IF DATA MATCHES
3863	025616	104031				ERROR	31	RELOCATION FAILED
3864	025620	013737	001176	062000	125:	MOV	\$TMP0,2#62000	RESTORE ORIGINAL DATA TO TEST LOCATION
3865	025626	012737	025674	001110		MOV	#135,\$LPERR	SET LOOP ON ERROR POINTER TO 135
3866	025634	013737	062000	001176		MOV	2#62000,\$TMP0	SAVE DATA AT TEST LOCATION
3867	025642	012737	000556	172350		MOV	#556,2#KIPAR4	LOAD PAR4 WITH 556
3868	025650	012737	000556	001200		MOV	#556,\$TMP1	SAVE 556 FOR ERROR REPORT
3869	025656	012737	062000	001202		MOV	#62000,\$TMP2	SAVE 62000 FOR ERROR REPORT
3870	025664	012700	104200			MOV	#104200,R0	PUT VIRTUAL ADDRESS IN R0
3871	025670	012701	125206			MOV	#125206,R1	PUT DATA PATTERN IN R1

K06

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG.
DOKTAA.P11 07-FEB-77 10:30 T31

MACY11 27(1006) 07-FEB-77 10:37 PAGE 75
18-BIT MAPPING ADDER TESTING

3872	025674	052737	000400	177572	13\$:	BIS	#BIT8,2#MMR0	:TURN ON DESTINATION ONLY RELOCATION
3873	025702	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 62000
3874	025704	013702	062000			MOV	2#62000,R2	:READ (62000) INTO R2
3875	025710	000005				RESET		:CLEAR MMR0
3876	025712	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
3877	025714	001401				BEQ	14\$	:BRANCH IF DATA MATCHES
3878	025716	104031				ERROR	31	:RELOCATION FAILED
3879	025720	013737	001176	062000	14\$:	MOV	\$TMP0,2#62000	:RESTORE ORIGINAL DATA TO TEST LOCATION
3880	025726	012737	025774	001110		MOV	#15\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 15\$
3881	025734	013737	062000	001176		MOV	2#62000,\$TMP0	:SAVE DATA AT TEST LOCATION
3882	025742	012737	000442	172350		MOV	#442,2#KIPAR4	:LOAD PAR4 WITH 442
3883	025750	012737	000442	001200		MOV	#442,\$TMP1	:SAVE 442 FOR ERROR REPORT
3884	025756	012737	062000	001202		MOV	#62000,\$TMP2	:SAVE 62000 FOR ERROR REPORT
3885	025764	012700	115600			MOV	#115600,R0	:PUT VIRTUAL ADDRESS IN R0
3886	025770	012701	125207			MOV	#125207,R1	:PUT DATA PATTERN IN R1
3887	025774	052737	000400	177572	15\$:	BIS	#BIT8,2#MMR0	:TURN ON DESTINATION ONLY RELOCATION
3888	026002	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 62000
3889	026004	013702	062000			MOV	2#62000,R2	:READ (62000) INTO R2
3890	026010	000005				RESET		:CLEAR MMR0
3891	026012	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
3892	026014	001401				BEQ	16\$	:BRANCH IF DATA MATCHES
3893	026016	104031				ERROR	31	:RELOCATION FAILED
3894	026020	013737	001176	062000	16\$:	MOV	\$TMP0,2#62000	:RESTORE ORIGINAL DATA TO TEST LOCATION
3895	026026	012737	026074	001110		MOV	#17\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 17\$
3896	026034	013737	062000	001176		MOV	2#62000,\$TMP0	:SAVE DATA AT TEST LOCATION
3897	026042	012737	000577	172350		MOV	#577,2#KIPAR4	:LOAD PAR4 WITH 577
3898	026050	012737	000577	001200		MOV	#577,\$TMP1	:SAVE 577 FOR ERROR REPORT
3899	026056	012737	062000	001202		MOV	#62000,\$TMP2	:SAVE 62000 FOR ERROR REPORT
3900	026064	012700	102100			MOV	#102100,R0	:PUT VIRTUAL ADDRESS IN R0
3901	026070	012701	125210			MOV	#125210,R1	:PUT DATA PATTERN IN R1
3902	026074	052737	000400	177572	17\$:	BIS	#BIT8,2#MMR0	:TURN ON DESTINATION ONLY RELOCATION
3903	026102	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 62000
3904	026104	013702	062000			MOV	2#62000,R2	:READ (62000) INTO R2
3905	026110	000005				RESET		:CLEAR MMR0
3906	026112	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
3907	026114	001401				BEQ	18\$	:BRANCH IF DATA MATCHES
3908	026116	104031				ERROR	31	:RELOCATION FAILED
3909	026120	013737	001176	062000	18\$:	MOV	\$TMP0,2#62000	:RESTORE ORIGINAL DATA TO TEST LOCATION
3910	026126	012737	026174	001110		MOV	#21\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 21\$
3911	026134	013737	062000	001176		MOV	2#62000,\$TMP0	:SAVE DATA AT TEST LOCATION
3912	026142	012737	000421	172350		MOV	#421,2#KIPAR4	:LOAD PAR4 WITH 421
3913	026150	012737	000421	001200		MOV	#421,\$TMP1	:SAVE 421 FOR ERROR REPORT
3914	026156	012737	062000	001202		MOV	#62000,\$TMP2	:SAVE 62000 FOR ERROR REPORT
3915	026164	012700	117700			MOV	#117700,R0	:PUT VIRTUAL ADDRESS IN R0
3916	026170	012701	125211			MOV	#125211,R1	:PUT DATA PATTERN IN R1
3917	026174	052737	000400	177572	21\$:	BIS	#BIT8,2#MMR0	:TURN ON DESTINATION ONLY RELOCATION
3918	026202	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 62000
3919	026204	013702	062000			MOV	2#62000,R2	:READ (62000) INTO R2
3920	026210	000005				RESET		:CLEAR MMR0
3921	026212	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
3922	026214	001401				BEQ	22\$	:BRANCH IF DATA MATCHES
3923	026216	104031				ERROR	31	:RELOCATION FAILED
3924	026220	013737	001176	062000	22\$:	MOV	\$TMP0,2#62000	:RESTORE ORIGINAL DATA TO TEST LOCATION
3925	026226	012737	026274	001110		MOV	#23\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 23\$
3926	026234	013737	060000	001176		MOV	2#60000,\$TMP0	:SAVE DATA AT TEST LOCATION
3927	026242	012737	000577	172350		MOV	#577,2#KIPAR4	:LOAD PAR4 WITH 577

MAINDEC-11-DOKTA-A POP 11/6X MEM. MGMT. DIAG. T31
DOKTAA.P11 07-FEB-77 10:30

MACY11 27(1006) 07-FEB-77 10:37 PAGE 76
18-BIT MAPPING ADDER TESTING

3928	026250	012737	000577	001200		MOV	#577,STMP1	;SAVE 577 FOR ERROR REPORT
3929	026256	012737	060000	001202		MOV	#60000,STMP2	;SAVE 60000 FOR ERROR REPORT
3930	026264	012700	100100			MOV	#100100,R0	;PUT VIRTUAL ADDRESS IN R0
3931	026270	012701	125212			MOV	#125212,R1	;PUT DATA PATTERN IN R1
3932	026274	052737	000400	177572	23\$:	BIS	#BIT8,2#MMRO	;TURN ON DESTINATION ONLY RELOCATION
3933	026302	010110				MOV	R1,(R0)	;TRY TO LOAD DATA PATTERN INTO 60000
3934	026304	013702	060000			MOV	2#60000,R2	;READ (60000) INTO R2
3935	026310	000005				RESET		;CLEAR MMRO
3936	026312	020102				CMP	R1,R2	;SEE IF DATA MATCHES PATTERN
3937	026314	001401				BEQ	24\$	;BRANCH IF DATA MATCHES
3938	026316	104031				ERROR	31	;RELOCATION FAILED
3939	026320	013737	001176	060000	24\$:	MOV	STMP0,2#60000	;RESTORE ORIGINAL DATA TO TEST LOCATION
3940	026326	012737	026374	001110		MOV	#25\$,SLPERR	;SET LOOP ON ERROR POINTER TO 25\$
3941	026334	013737	060000	001176		MOV	2#60000,STMP0	;SAVE DATA AT TEST LOCATION
3942	026342	012737	000570	172350		MOV	#570,2#KIPAR4	;LOAD PAR4 WITH 570
3943	026350	012737	000570	001200		MOV	#570,STMP1	;SAVE 570 FOR ERROR REPORT
3944	026356	012737	060000	001202		MOV	#60000,STMP2	;SAVE 60000 FOR ERROR REPORT
3945	026364	012700	101000			MOV	#101000,R0	;PUT VIRTUAL ADDRESS IN R0
3946	026370	012701	125213			MOV	#125213,R1	;PUT DATA PATTERN IN R1
3947	026374	052737	000400	177572	25\$:	BIS	#BIT8,2#MMRO	;TURN ON DESTINATION ONLY RELOCATION
3948	026402	010110				MOV	R1,(R0)	;TRY TO LOAD DATA PATTERN INTO 60000
3949	026404	013702	060000			MOV	2#60000,R2	;READ (60000) INTO R2
3950	026410	000005				RESET		;CLEAR MMRO
3951	026412	020102				CMP	R1,R2	;SEE IF DATA MATCHES PATTERN
3952	026414	001401				BEQ	26\$	;BRANCH IF DATA MATCHES
3953	026416	104031				ERROR	31	;RELOCATION FAILED
3954	026420	013737	001176	060000	26\$:	MOV	STMP0,2#60000	;RESTORE ORIGINAL DATA TO TEST LOCATION
3955								
3956								
3957								
3958								
3959								
3960								
3961								
3962	026426	012737	026466	001110		MOV	#27\$,SLPERR	;SET LOOP ON ERROR POINTER TO 27\$
3963	026434	012737	007600	172350		MOV	#7600,KIPAR4	;LOAD PAR4 WITH 7600
3964	026442	012737	007600	001200		MOV	#7600,STMP1	;SAVE PAR4 DATA FOR ERROR REPORT
3965	026450	012737	177776	001202		MOV	#177776,STMP2	;SAVE PHYS. ADDR. FOR ERROR REPORT
3966	026456	012700	117776			MOV	#117776,R0	;PUT VIRTUAL ADDR. IN R0
3967	026462	012701	000100			MOV	#100,R1	;PUT DATA PATTERN IN R1
3968	026466	052737	000400	177572	27\$:	BIS	#BIT8,2#MMRO	;TURN ON DEST.-ONLY-RELOCATION
3969	026474	010110				MOV	R1,(R0)	;LOAD DATA INTO PSW USING PAR4
3970	026476	013702	177776			MOV	2#177776,R2	;READ PSW INTO R2
3971	026502	000005				RESET		;CLEAR MMRO
3972	026504	042702	000020			BIC	#20,R2	;CLEAR T-BIT IF SET IN DATA READ
3973	026510	020102				CMP	R1,R2	;SEE IF DATA MATCHES PATTERN WRITTEN
3974	026512	001401				BEQ	28\$	;BRANCH IF DATA MATCHES
3975	026514	104031				ERROR	31	;RELOCATION FAILED
3976	026516	012737	026556	001110	28\$:	MOV	#29\$,SLPERR	;SET LOOP ON ERROR POINTER TO 29\$
3977	026524	012737	007777	172350		MOV	#7777,KIPAR4	;LOAD PAR4 WITH 7777
3978	026532	012737	007777	001200		MOV	#7777,STMP1	;SAVE PAR4 DATA FOR ERROR REPORT
3979	026540	012737	177776	001202		MOV	#177776,STMP2	;SAVE PHYS. ADDR. FOR ERROR REPORT
3980	026546	012700	100076			MOV	#100076,R0	;PUT VIRTUAL ADDR. R0
3981	026552	012701	000200			MOV	#200,R1	;PUT DATA PATTERN IN R1
3982	026556	052737	000400	177572	29\$:	BIS	#BIT8,2#MMRO	;TURN ON DEST.-ONLY-RELOCATION
3983	026564	010110				MOV	R1,(R0)	;LOAD DATA INTO PSW USING PAR4

THE FOLLOWING CODE PERFORMS A COUPLE RELOCATION ADDER CHECKS
USING THE PROCESSOR STATUS WORD AS THE PHYSICAL ADDRESS
WHICH IS GENERATED TO CHECK THE UPPER BITS OF THE "PHYSICAL
BUS ADDRESS ALU". SPECIAL DATA PATTERNS ARE USED DUE TO
THE CONDITION CODE BITS.

M06

MAINDEC-11-DOKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 77
 DOKTAA.P11 07-FEB-77 10:30 T31 18-BIT MAPPING ADDER TESTING

3984	026566	013702	177776		MOV	2#177776,R2	;READ PSW INTO R2
3985	026572	000005			RESET		;CLEAR MMRO
3986	026574	042702	000020		BIC	#20,R2	;CLEAR T-BIT IF SET IN DATA READ
3987	026600	020102			CMP	R1,R2	;SEE IF DATA MATCHES PATTERN
3988	026602	001401			BEQ	30\$	;BRANCH IF DATA MATCHES
3989	026604	104031			ERROR	31	;RELOCATION FAILED
3990	026606			30\$:			
3991	026606	005037	177776		CLR	2#PSW	;CLEAR PSW BEFORE LEAVING TEST
3992	026612	012737	025150	001110	MOV	#20\$,SLPERR	;SET LOOP ON ERROR POINTER TO START OF TEST
3993							
3994							
3995							
3996							
3997							
3998							
3999							
4000							
4001							
4002							
4003							
4004							
4005	026620						
4006	026620	000004					
4007	026622	012737	027166	001446			
4008							
4009	026630	005037	001434				
4010	026634	005037	001200				
4011	026640	012737	027156	000004			
4012	026646	105737	001450				
4013	026652	001403					
4014	026654	012737	027146	000004			
4015	026662						
4016	026662	012737	000577	172350			
4017	026670	012737	000600	172352			
4018	026676	012700	100100				
4019	026702	012701	120000				
4020	026706	012702	001000				
4021	026712	012737	000001	177572			
4022							
4023							
4024							
4025	026720	012737	027016	001110	1\$:		
4026	026726	005037	001410				
4027	026732	011137	001176				
4028	026736	032737	000020	001410			
4029	026744	001411					
4030	026746	005237	001434				
4031	026752	005737	001200				
4032	026756	001034					
4033	026760	013737	172352	001200			
4034	026766	000430					
4035	026770	005737	001434		2\$:		
4036	026774	001410					
4037	026776	013737	172352	001202			
4038	027004	104032					
4039	027006	005037	001434				

;TEST 32 18-BIT MAPPING CARRY PROPAGATION							

; THIS TEST USES FULL 18-BIT RELOCATION TO CHECK THE CARRY							
; PROPAGATION OF THE RELOCATION ADDER. SINCE THIS TEST SCANS							
; MEMORY FROM 060000 TO 750000 ON 2K BOUNDARIES, IT WILL							
; REPORT ANY HOLES THAT IT FINDS IN MEMORY UP TO THE 124K							
; THE INFORMATION GIVEN WILL BE THE ADDRESS WHERE							
; THE HOLE WAS DISCOVERED AND THE FIRST GOOD ADDRESS AFTER THE							
; HOLE.							

;TST32:							
SCOPE							
MOV #TST33,NXTTST ;SAVE STARTING ADDRESS OF NEXT							
;TEST FOR ESCAPE ON PARITY ERRORS							
20\$:							
CLR HOLFLG ;MAKE SURE HOLE FLAG STARTS AT 0							
CLR \$TMP1 ;CLEAR SPOT TO SAVE HOLE ADDRESS							
MOV #11\$,ERRVEC ;SET ERROR POINTER TO 11\$ FOR 11/6X							
TSTB FORTY ;EXECUTING ON AN 11/40							
BEQ 7\$;BRANCH IF NO							
MOV #10\$,ERRVEC ;SET ERROR POINTER TO 10\$ FOR 11/40							
7\$:							
MOV #577,KIPAR4 ;LOAD PAR4 WITH STARTING BASE							
MOV #600,KIPAR5 ;START ADDRESSING WITH 12K							
MOV #100100,R0 ;LOAD VIRTUAL ADDR FOR PAR4 IN R0							
MOV #120000,R1 ;LOAD VIRTUAL ADDR FOR PAR5 IN R1							
MOV #1000,R2 ;LOAD DATA PATTERN INTO R2							
MOV #1,2#MMRO ;TURN ON 18-BIT MAPPING							
;FULL RELOCATION STARTS HERE AND CONTINUES FOR REST OF PROGRAM							
;							
MOV #3\$,SLPERR ;SET LOOP ON ERROR POINTER TO 3\$							
CLR PCPUER ;CLEAR CPU ERROR FLAG							
MOV (R1),\$TMP0 ;SAVE DATA AT TEST LOCATION							
BIT #BIT4,PCPUER ;SEE IF THERE WAS A CPU TRAP							
BEQ 2\$;BRANCH IF NO TIMEOUT							
INC HOLFLG ;SET FLAG IF THERE WAS TIMEOUT							
TST \$TMP1 ;HAVE I SAVED STARTING ADDR. OF HOLE							
BNE 5\$;IF I HAVE I'LL CONTINUE							
MOV KIPAR5,\$TMP1 ;SAVE PAR-POSSIBLE START OF HOLE							
BR 5\$;BRANCH TO CHANGE BASE ADDRESS							
2\$:							
TST HOLFLG ;IF HOLFLG CLEAR-NO TIMEOUTS YET							
BEQ 3\$;OTHERWISE HAVE FOUND MEMORY HOLE							
MOV KIPAR5,\$TMP2 ;SAVE PAR THAT POINTS TO END OF HOLE							
ERROR 32 ;HOLE IN MEMORY FROM \$TMP1 TO \$TMP2							
CLR HOLFLG ;CLEAR HOLE FLAG IN CASE THERES MORE							

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG.
DOKTAA.P11 07-FEB-77 10:30 T32

MACY11 27(1006) 07-FEB-77 10:37 PAGE 78
18-BIT MAPPING CARRY PROPAGATION

```

4040 027012 005037 001200          CLR      STMP1          ;CLEAR STARTING ADDR. LOC. ALSO
4041 027016 010210          MOV      R2,(R0)          ;LOAD TEST PATTERN INTO TEST LOCATION
4042 027020 011103          MOV      (R1),R3          ;READ TEST LOCATION VIA DIFFERENT PAR
4043 027022 020203          CMP      R2,R3          ;SEE IF CORRECT LOCATION REFERENCED
4044 027024 001407          BEQ      4$          ;BRANCH IF CORRECT DATA OBTAINED
4045 027026 013737 172350 001204  MOV      KIPAR4,STMP3          ;SAVE PAR4 FOR ERROR REPORT
4046 027034 013737 172352 001206  MOV      KIPAR5,STMP4          ;SAVE PAR5 FOR ERROR REPORT
4047 027042 104033          ERROR 33          ;OTHERWISE BAD RELOCATION
4048 027044 013711 001176          MOV      STMP0,(R1)          ;RESTORE ORIGINAL DATA TO TEST LOCATION
4049 027050 062737 000100 172350 5$: ADD      #100,KIPAR4          ;CHANGE BASE ADDRESS
4050 027056 062737 000100 172352 5$: ADD      #100,KIPAR5          ;CHANGE BASE ADDRESS
4051 027064 005202          INC      R2          ;CHANGE DATA PATTERN
4052 027066 022737 007500 172352  CMP      #7500,KIPAR5          ;SEE IF PAST 122K-LAST ADDRESS
4053 027074 103311          BHIS     1$          ;BRANCH IF NOT PAST LAST ADDRESS
4054          ;*
4055          ;* ADDRESS 000000 IS GENERATED BY CARRY PROPAGATION, USING
4056          ;* '7777' IN KIPAR4 WITH A VIRTUAL ADDRESS OF '100100'.
4057          ;*
4058          ;*
4059          ;*
4060          ;*
4061          ;*
4062          ;*
4063 027076 012737 027116 001110 6$: MOV      #16$,$LPERR          ;SET LOOP ON ERROR POINTER TO 16$
4064 027104 012737 007777 172350  MOV      #7777,KIPAR4          ;LOAD PAR 4 WITH HIGHEST VALUE POSSIBLE
4065 027112 005037 000000          CLR      #000000          ;CLEAR ADDRESS ZERO
4066 027116 013701 100100          16$: MOV      #100100,R1          ;THIS SHOULD READ ADDRESS ZERO INTO R1
4067 027122 005701          TST      R1          ;SEE IF YOU REALLY READ ADDRESS ZERO
4068 027124 001401          BEQ      8$          ;BRANCH IF YOU READ ADDRESS ZERO
4069 027126 104034          ERROR 34          ;DIDN'T READ ADDRESS ZERO
4070 027130 012737 015162 000004 8$: MOV      #CPUER,ERRVEC          ;RESTORE NORMAL CPU TRAP ROUTINE
4071 027136 012737 026630 001110  MOV      #20$,$LPERR          ;SET LOOP POINTER TO START OF TEST
4072 027144 000410          BR       TST33          ;BRANCH TO NEXT TEST
4073
4074          ;;***** TRAP TO HERE THRU ERRVEC *****
4075
4076 027146 012737 000020 001410 10$: MOV      #BIT4,PCPUER          ;SET "TRAP-TO-4" FLAG
4077 027154 000002          RTI
4078 027156 017737 152172 001410 11$: MOV      #CPUERR,PCPUER          ;SAVE CPU ERROR REGISTER
4079 027164 000002          RTI          ;RETURN TO TEST AND CONTINUE
4080
4081
4082
4083
4084
4085
4086          .SBTTL ***** ENTRY POINT 5 --- STARTING ADDRESS 220 *****
4087          .SBTTL ***** MEMORY MANAGEMENT ABORTS LOGIC TESTS *****
4088          ;*
4089          ;* THIS GROUP OF TESTS CHECKS OUT THE MEMORY MANAGEMENT ABORT LOGIC, AND
4090          ;* ALSO CHECK OUT BITS <06:05> AND <03:00> OF MMRO, AND ALL BITS OF MMR2.
4091          ;*
4092
4093          ;*****
4094          ;TEST 33 PAGE LENGTH FAULTS - UPWARD EXPANSION
4095          ;*

```

4096
4097
4098
4099
4100
4101
4102
4103
4104
4105
4106
4107
4108
4109
4110
4111
4112
4113
4114
4115
4116
4117
4118
4119
4120
4121
4122
4123
4124
4125
4126
4127
4128
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4150
4151

• ✱ • ✱ • ✱ • ✱ • ✱ •

THIS TEST CHECKS OUT THE PAGE LENGTH COMPARATORS AND THE LOGIC THAT GENERATES "PAGE LTH ERR". IT TRIES EVERY PAGE LENGTH FIELD FROM 1 BLOCK TO 200(8) BLOCKS. THE TEST THEN TRIES A REFERENCE AT EVERY 100 BYTES UNTIL AN ABORT OCCURS. IF THE ABORT HAPPENS TOO SOON OR IF NO ABORT HAPPENS AT THE CORRECT BLOCK NUMBER AN ERROR IS REPORTED.

```

*****
+ST33:

```

```

SCOPE
MOV      #TST34,NXTTST      ;SAVE STARTING ADDRESS OF NEXT
                                ;TEST FOR ESCAPE ON PARITY ERRORS
                                ;;DO 24 ITERATIONS
ENTPT5:  MOV      #24,$TIMES
MOV      #20$,SLPERR        ;SET LOOP ON ERROR POINTER TO 20$
MOV      #33,$TSTNM         ;LOAD TEST NUMBER INTO MEMORY
MOV      $TSTNM,$DISP1AY    ;DISPLAY TEST NUMBER FOR THIS TEST
MOV      #77406,KIPDR0      ;MAKE KERNEL I PAGE 0 200 BLOCKS, R/W
MOV      #77406,KIPDR1      ;MAKE KERNEL I PAGE 1 200 BLOCKS, R/W
MOV      #77406,KIPDR2      ;MAKE KERNEL I PAGE 2 200 BLOCKS, R/W
MOV      #77406,KIPDR3      ;MAKE KERNEL I PAGE 3 200 BLOCKS, R/W
MOV      #77406,KIPDR7      ;MAKE KERNEL I PAGE 7 200 BLOCKS, R/W
MOV      #000,KIPAR0         ;MAP KERNEL I PAGE 0 TO 0 - 4K
MOV      #200,KIPAR1         ;MAP KERNEL I PAGE 1 TO 4K - 8K
MOV      #400,KIPAR2         ;MAP KERNEL I PAGE 2 TO 8K - 12K
MOV      #600,KIPAR3         ;MAP KERNEL I PAGE 3 TO 12K - 16K
MOV      #7600,KIPAR7        ;MAP KERNEL I PAGE 7 TO THE I/O PAGE
MOV      #BIT0,MMRO          ;ENABLE 18-BIT RELOCATION IF NOT ON
20$:     MOV      #000006,$KIPDR4 ;LOAD PDR4 FOR PAGE LENGTH OF 1
MOV      #600,KIPAR4         ;MAP PAGE 4 TO 12K
MOV      #KIPDR4+1,R0        ;PUT ADDRESS OF PDR 4'S UPPER BYTE IN R0
1$:      MOV      #3$,SLPERR    ;SET LOOP ON ERROR POINTER TO 3$
CLR      PMMR0               ;CLEAR LOCATION THAT HOLDS MMRO
MOV      #100000,R2           ;PUT VIRTUAL ADDRESS INTO R2
2$:      MOV      R2,R3        ;PUT VIRTUAL ADDRESS INTO R3 TOO
        HCR      R3           ;RIGHT SHIFT 6 BITS SO IT WILL
        ASR      R3           ;MATCH THE PLF
        ASR      R3
        ASR      R3
        ASR      R3
        ASR      R3
        BIC      #.77600,R3    ;CLEAR BITS THAT ARE SET IN UPPER BYTE
MOV      (R0),R1              ;SAVE PLF IN R1 FOR LATER COMPARISON
3$:      MOV      #040011,MMEXP ;POTENTIAL ABORT CONDITION: PAGE LENGTH
                                ;ERROR KERNEL, I-SPACE, PAGE 4
MOV      (R2),R4              ;READ USING V.A. IN R2
CLR      MMEXP                ;CLEAR EXPECTED ABORT CONDITION
TST      PMMR0                ;SEE IF ABORT OCCURRED YET
BEQ      4$                   ;BRANCH IF NO ABORT YET, CHANGE V.A.
DEC      R3                    ;MAKE R3 EQUAL TO PLF
CMP      R1,R3                 ;SEE IF PDR 4'S PLF=R3
BEQ      6$                   ;BRANCH IF ABORT HAPPENS AT RIGHT PLACE
ERROR    3$                   ;ABORT WRONG PLACE
BR       6$                   ;BRANCH TO CHANGE PLF
4$:      CMP      R1,R3        ;SEE IF PAGE LENGTH FAULT SHOULD HAVE OCCURRED
        BHS      5$           ;BRANCH IF NO PAGE LENGTH FAULT CONDITION

```

C07

 MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG.
 DQKTA.A.P11 07-FEB-77 10:30 T33

 MACY11 27(1006) 07-FEB-77 10:37 PAGE 80
 PAGE LENGTH FAULTS - UPWARD EXPANSION

4152	027452	104036			ERROR	36		; NO ABORT, IT SHOULD HAVE HAPPENED THIS TIME
4153	027454	000406			BR	6\$		; BRANCH TO CHANGE PLF
4154	027456	120327	000177	5\$:	CMPB	R3, #177		; SEE IF V.A. IS 177
4155	027462	001403			BEQ	6\$		; BRANCH TO CHECK PLF
4156	027464	062702	000100		ADD	#100, R2		; CHANGE VIRTUAL ADDRESS
4157	027470	000736			BK	2\$		; GO TRY THE NEXT VIRTUAL ADDRESS
4158	027472	122710	000177	6\$:	CMPB	#177, (R0)		; SEE IF PDR 4'S PLF IS 177
4159	027476	001402			BEQ	7\$		; BRANCH TO EXIT IF PLF IS 177
4160	027500	105210			INCB	(R0)		; STEP PLF OF PDR 4 UP BY 1
4161	027502	000725			BR	1\$		; BRANCH TO START WITH V.A. OF 0
4162	027504	012737	027330	7\$:	MOV	#20\$, \$LPERR		; SET LOOP POINTER TO START OF TEST

 ; TEST 34 PAGE LENGTH FAULTS - DOWNWARD EXPANSION

 ;
 ; THIS TEST CHECKS OUT THE PAGE LENGTH COMPARATORS
 ; AND THE LOGIC THAT GENERATES "PAGE LTH ERR". IT TRIES
 ; EVERY PAGE LENGTH FIELD FROM 1 BLOCK TO 200(8) BLOCKS. THE
 ; TEST THEN TRIES A REFERENCE AT EVERY 100 BYTES UNTIL AN ABORT
 ; OCCURS. IF THE ABORT HAPPENS TOO SOON OR IF NO ABORT HAPPENS AT
 ; THE CORRECT BLOCK NUMBER AN ERROR IS REPORTED.
 ;

 ; ST34:

4175	027512				SCOPE			
4176	027512	000004			MOV	#TST35, NXTTST		; SAVE STARTING ADDRESS OF NEXT
4177	027514	012737	027712	001446				; TEST FOR ESCAPE ON PARITY ERRORS
4178								; DO 24 ITERATIONS
4179	027522	012737	000024	001212	MOV	#24, \$TIMES		; LOAD PDR4 FOR PAGE LENGTH OF 1
4180	027530	012737	077416	172310	20\$:	MOV	#77416, 2#KIPDR4	; MAP PAGE 4 TO 12K
4181	027536	012737	000600	172350	MOV	#600, KIPAR4		; PUT ADDRESS OF PDR 4'S UPPER BYTE IN R0
4182	027544	012700	172311		MOV	#KIPDR4+1, R0		; SET LOOP ON ERROR POINTER TO 3\$
4183	027550	012737	027612	001110	MOV	#3\$, \$LPERR		; CLEAR LOCATION THAT HOLDS MMRO
4184	027556	005037	001404		1\$:	CLR	PMMRO	; PUT VIRTUAL ADDRESS INTO R2
4185	027562	012702	117700		MOV	#117700, R2		; PUT VIRTUAL ADDRESS INTO R3 TOO
4186	027566	010203			2\$:	MOV	R2, R3	; RIGHT SHIFT 6 BITS SO IT WILL
4187	027570	006203			ASR	R3		; MATCH THE PLF
4188	027572	006203			ASR	R3		
4189	027574	006203			ASR	R3		
4190	027576	006203			ASR	R3		
4191	027600	006203			ASR	R3		
4192	027602	006203			ASR	R3		
4193	027604	042703	177600		BIC	#177600, R3		; CLEAR BITS THAT ARE SET IN UPPER BYTE
4194	027610	111001			MOV	(R0), R1		; SAVE PLF IN R1 FOR LATER COMPARISON
4195	027612	012737	040011	001366	3\$:	MOV	#040011, MMEXP	; POTENTIAL ABORT CONDITION: PAGE LENGTH
4196								; ERROR KERNEL, I-SPACE, PAGE 4
4197	027620	011204			MOV	(R2), R4		; READ USING V.A. IN R2
4198	027622	005037	001366		CLR	MMEXP		; CLEAR EXPECTED ABORT CONDITION
4199	027626	005737	001404		TST	PMMRO		; SEE IF ABORT OCCURRED YET
4200	027632	001405			BEQ	4\$		; BRANCH IF NO ABORT YET, CHANGE V.A.
4201	027634	005203			INC	R3		; MAKE R3 EQUAL TO PLF
4202	027636	020103			CMP	R1, R3		; SEE IF PDR 4'S PLF=R3
4203	027640	001414			BEQ	6\$		; BRANCH IF ABORT HAPPENS AT RIGHT PLACE
4204	027642	104035			ERROR	3\$		; ABORT WRONG PLACE
4205	027644	000412			BR	6\$		; BRANCH TO CHANGE PLF
4206	027646	020103		4\$:	CMP	R1, R3		; SEE IF PAGE LENGTH FAULT SHOULD HAVE OCCURRED
4207	027650	101402			BLOS	5\$		; BRANCH IF NO PAGE LENGTH FAULT CONDITION

007

 MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG.
 DQKTA.P11 07-FEB-77 10:30 T34

 MACY11 27(1006) 07-FEB-77 10:37 PAGE 81
 PAGE LENGTH FAULTS - DOWNWARD EXPANSION

```

4208 027652 104036 ERROR 36 ; NO ABORT, IT SHOULD HAVE HAPPENED THIS TIME
4209 027654 000406 BR 6$ ; BRANCH TO CHANGE PLF
4210 027656 120327 000000 5$: CMPB R3, #000 ; SEE IF V.A. IS 000
4211 027662 001403 BEQ 6$ ; BRANCH TO CHECK PLF
4212 027664 162702 000100 SUB #100, R2 ; CHANGE VIRTUAL ADDRESS
4213 027670 000736 BR 2$ ; GO TRY THE NEXT VIRTUAL ADDRESS
4214 027672 122710 000000 6$: CMPB #000, (R0) ; SEE IF PDR 4'S PLF IS 000
4215 027676 001402 BEQ 7$ ; BRANCH TO EXIT IF PLF IS 000
4216 027700 105310 DECB (R0) ; STEP PLF OF PDR 4 UP BY 1
4217 027702 000725 BR 1$ ; BRANCH TO START WITH V.A. OF 0
4218 027704 012737 027530 001110 7$: MOV #20$, $LPERR ; SET LOOP POINTER TO START OF TEST
4219
4220
4221
4222
4223
4224
4225
4226
4227
4228
4229
4230
4231 027712
4232 027712 000004
4233 027714 012737 030210 001446
4234
4235 027722 012737 077406 172310 20$: MOV #77406, @#KIPDR4 ; SAVE STARTING ADDRESS OF NEXT
4236 027730 012737 077400 172312 MOV #77400, @#KIPDR5 ; TEST FOR ESCAPE ON PARITY ERRORS
4237
4238 027736 012737 000600 172350 MOV #600, @#KIPAR4 ; LOAD 12K INTO PAR4
4239 027744 012737 000600 172352 MOV #600, @#KIPAR5 ; LOAD 12K INTO PAR5
4240 027752 012700 100000 MOV #100000, R0 ; LOAD VIRTUAL ADDRESS FOR PAR4 INTO R0
4241 027756 012701 120000 MOV #120000, R1 ; LOAD VIRTUAL ADDRESS FOR PAR5 INTO R1
4242 027762 012702 161457 MOV #161457, R2 ; LOAD DATA PATTERN INTO R2
4243 027766 012737 027776 001110 11$: MOV #1$, $LPERR ; SET LOOP ON ERROR POINTER TO 1$
4244 027774 005010 CLR (R0) ; CLEAR MEMORY LOCATION 60000
4245 027776 012737 100013 001366 1$: MOV #100013, MMEXP ; LOAD EXPECTED ABORT CONDITION: NON-RESIDENT,
4246 ; KERNEL, PAGE 5, FULL RELOCATION
4247 030004 010211 8$: MOV R2, (R1) ; WRITE TO NON-RESIDENT OR UNUSED ACF
4248 ; SHOULD CAUSE ABORT
4249 030006 005037 001366 CLR MMEXP ; CLEAR EXPECTED ABORT CONDITION
4250
4251 030012 005710 TST (R0) ; SEE IF (60000) IS STILL ZERO
4252 030014 001411 BEQ 2$ ; BRANCH IF (60000) IS ZERO
4253 030016 013737 172312 001176 MOV KIPDR5, $TMP0 ; SAVE KIPDR5 FOR ERROR REPORT
4254 030024 013737 172352 001200 MOV KIPAR5, $TMP1 ; SAVE KIPAR5 FOR ERROR REPORT
4255 030032 010137 001202 MOV R1, $TMP2 ; SAVE VIRT ADR. FOR ERROR REPORT
4256 030036 104037 ERROR 37 ; ABORT DID NOT HAPPEN
4257 030040 012737 030052 001110 2$: MOV #4$, $LPERR ; SET LOOP ON ERROR POINTER TO 4$
4258 030046 010210 MOV R2, (R0) ; LOAD DATA PATTERN INTO 60000
4259 030050 005004 CLR R4 ; CLEAR REGISTER TO RECEIVE DATA
4260 030052 012737 100013 001366 4$: MOV #100013, MMEXP ; LOAD EXPECTED ABORT CONDITION: NON-RESIDENT,
4261 ; KERNEL, I-SPACE, PAGE 5, FULL RELOCATION
4262 030060 011104 9$: MOV (R1), R4 ; TRY TO READ (100000) INTO R4
4263 ; THIS SHOULD ABORT

```

```

*****
;TEST 35 ACCESS CONTROL FIELD = 0, OR 4 (ABORT ALL ACCESSES)

```

```

;
; THESE A.C.F.'S ARE ALL NON-RESIDENT, ANY REFERENCE (READ
; OR WRITE) TO A NON-RESIDENT PAGE SHOULD SET BIT 15 IN MMRO.
; BITS <06:05> AND <03:00> IN MMRO WILL LOCK UP ALL AVAILABLE INFORMATION
; ON THAT PAGE. IN THIS CASE THE PAGE THAT CAUSES THE ABORT
; IS KERNEL PAGE 5. THE EXPECTED ERROR CODE IS 100013.

```

```

*****
;T35:

```

```

SCOPE
MOV #TST36, NXXTST ; SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
MOV #77406, @#KIPDR4 ; LOAD ACF 6 INTO PDR 4
MOV #77400, @#KIPDR5 ; LOAD ACF 0 INTO PDR5
MOV #600, @#KIPAR4 ; LOAD 12K INTO PAR4
MOV #600, @#KIPAR5 ; LOAD 12K INTO PAR5
MOV #100000, R0 ; LOAD VIRTUAL ADDRESS FOR PAR4 INTO R0
MOV #120000, R1 ; LOAD VIRTUAL ADDRESS FOR PAR5 INTO R1
MOV #161457, R2 ; LOAD DATA PATTERN INTO R2
MOV #1$, $LPERR ; SET LOOP ON ERROR POINTER TO 1$
CLR (R0) ; CLEAR MEMORY LOCATION 60000
MOV #100013, MMEXP ; LOAD EXPECTED ABORT CONDITION: NON-RESIDENT,
; KERNEL, PAGE 5, FULL RELOCATION
MOV R2, (R1) ; WRITE TO NON-RESIDENT OR UNUSED ACF
; SHOULD CAUSE ABORT
CLR MMEXP ; CLEAR EXPECTED ABORT CONDITION
TST (R0) ; SEE IF (60000) IS STILL ZERO
BEQ 2$ ; BRANCH IF (60000) IS ZERO
MOV KIPDR5, $TMP0 ; SAVE KIPDR5 FOR ERROR REPORT
MOV KIPAR5, $TMP1 ; SAVE KIPAR5 FOR ERROR REPORT
MOV R1, $TMP2 ; SAVE VIRT ADR. FOR ERROR REPORT
ERROR 37 ; ABORT DID NOT HAPPEN
MOV #4$, $LPERR ; SET LOOP ON ERROR POINTER TO 4$
MOV R2, (R0) ; LOAD DATA PATTERN INTO 60000
CLR R4 ; CLEAR REGISTER TO RECEIVE DATA
MOV #100013, MMEXP ; LOAD EXPECTED ABORT CONDITION: NON-RESIDENT,
; KERNEL, I-SPACE, PAGE 5, FULL RELOCATION
MOV (R1), R4 ; TRY TO READ (100000) INTO R4
; THIS SHOULD ABORT

```


E07

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG.
DOKTAA.P11 07-FEB-77 10:30 T35

MACY11 27(1006) 07-FEB-77 10:37 PAGE 82
ACCESS CONTROL FIELD = 0, OR 4 (ABORT ALL ACCESSES)

```

4264 030062 005037 001366 CLR MMEXP ;EXPECTED ABORT CONDITION
4265
4266 030066 005704 TST R4 ;MAKE SURE R4 IS STILL 0
4267 030070 001411 BEQ 5$ ;BRANCH IF R4 IS 0
4268 030072 013737 172312 001176 MOV KIPDR5,$TMP0 ;SAVE KIPDR5 FOR ERROR REPORT
4269 030100 013737 172352 001200 MOV KIPAR5,$TMP1 ;SAVE KIPAR5 FOR ERROR REPORT
4270 030106 010137 001202 MOV R1,$TMP2 ;SAVE VIRT ADR. FOR ERROR REPORT
4271 030112 104037 ERROR 37 ;ABORT DID NOT HAPPEN
4272 030114
4273 030114 023727 172312 077404 5$: CMP KIPDR5,#77404 ;SEE IF PDR 5'S ACF=4
4274 030122 001404 BEQ 12$ ;TEST OVER IF ACF=4
4275 030124 012737 077404 172312 10$: MOV #77404,KIPDR5 ;MAKE PDR 5'S ACF=4
4276 030132 000715 BR 11$ ;REPEAT TEST WITH ACF=4
4277
4278
4279
4280
4281 030134 012737 030162 001110 12$: MOV #13$,$LPERR ;SET LOOP ON ERROR POINTER TO 13$
4282 030142 012737 007600 172352 MOV #7600,KIPAR5 ;MAP PAGE 5 TO I/O PAGE
4283 030150 012701 132350 MOV #132350,R1 ;LOAD VIRTUAL ADDRESS TO REFERENCE
4284 KIPAR4
4285 030154 012737 100013 001366 MOV #100013,MMEXP ;EXPECTING NON-RESIDENT ABORT
4286
4287 030162 005011 13$: CLR (R1) ;KERNEL I PAGE 5
4288 ;THIS INSTRUCTION SHOULD ABORT
4289 ;DURING THE ABORT 'NON RES ERR' IS
4290 ;ASSERTED TO STOP THE CLOCKING OF THE
4291 ;FLIP/FLOPS
4292 030164 022737 000600 172350 CMP #600,KIPAR4 ;KIPAR4 SHOULD STILL BE 600
4293 030172 001401 BEQ 14$ ;BRANCH IF KIPAR4 IS STILL 600
4294 030174 104074 ERROR 74 ;ABORT DID NOT STOP CLR'ING OF KIPAR4
4295 030176 012737 027722 001110 14$: MOV #20$,$LPERR ;SET LOOP POINTER TO START OF TEST
4296 030204 005037 001366 CLR MMEXP ;NOT EXPECTING ANY TRAPS
4297
4298
4299
4300
4301
4302
4303
4304
4305
4306
4307
4308 030210
4309 030210 000004
4310 030212 012737 030454 001446
4311
4312 030220
4313 030220 012737 000600 172352 20$: MOV #600,KIPAR5 ;MAP PAGE 5 TO 12K
4314 030226 012737 000600 172350 MOV #600,KIPAR4 ;MAP PAGE 4 TO 12K
4315 030234 012737 077406 172312 MOV #77406,KIPDR5 ;PAGE 5 IS 200 BLOCKS LONG
4316 ;EXPANDS UPWARD, AND IS READ/WRITE
4317 ;WITH NO TRAPPING
4318 030242 012737 077402 172310 MOV #77402,KIPDR4 ;LOAD ACF 2 INTO PDR 4
4319 030250 005037 001366 CLR MMEXP ;NOT EXPECTING ANY TRAPS OR ABORTS YET

```

;*****
 ;TEST 36 ACCESS CONTROL FIELD = 2 (ABORT ON WRITE)
 ;*
 ;* THIS IS A READ ONLY A.C.F. ANY WRITE ATTEMPT TO THIS PAGE
 ;* WILL ABORT AND SET BIT 13 OF MMRO.
 ;* BITS <06:05> AND <03:01> IN MMRO WILL LOCK UP ALL AVAILABLE INFORMATION
 ;* ON THAT PAGE. IN THIS CASE THE WRITE ATTEMPT WILL BE TO
 ;* KERNEL SPACE PAGE 4. THE EXPECTED ERROR CODE IS 020011.
 ;*
 ;*****
 ;TST36:

F07

 MAINDEC-11-DQKTA-A POP 11/6X MEM. MGMT. DIAG.
 DQKTA.P11 07-FEB-77 10:30 T36

 MACY11 27(1006) 07-FEB-77 10:37 PAGE 83
 ACCESS CONTROL FIELD = 2 (ABORT ON WRITE)

4320	030254	012737	002222	120000		MOV	#2222,2#120000	: LOAD DATA INTO 12K
4321	030262	013700	100000			MOV	2#100000,RO	: READ DATA THRU PAGE 4
4322	030266	005037	120000			CLR	2#120000	: CLEAR TEST LOCATION THRU PAGE 5
4323	030272	012737	020011	001366		MOV	#20011,MMEXP	: EXPECTING READ ONLY FAULT, PAGE 4
4324	030300	012737	017777	100000		MOV	#17777,2#100000	: TRY TO WRITE THRU PAGE 4
4325	030306	005037	001366			CLR	MMEXP	: NO MORE TRAPS EXPECTED
4326	030312	005737	120000			TST	2#120000	: SEE IF TEST LOCATION IS STILL ZERO
4327	030316	001412				BEG	2\$	: BRANCH IF WORD IS STILL ZERO
4328	030320	013737	17210	001176		MOV	KIPDR4,\$TMP0	: SAVE KIPDR4 FOR ERROR REPORT
4329	030326	013737	172350	001200		MOV	KIPAR4,\$TMP1	: SAVE KIPAR4 FOR ERROR REPORT
4330	030334	012737	100000	001202		MOV	#100000,\$TMP2	: SAVE VIRT ADR. FOR ERROR REPORT
4331	030342	104040				ERROR	40	: NO ABORT ON PAGE 4
4332	030344	105737	001450		2\$:	TSTB	FORTY	: EXECUTING ON AN 11/40
4333	030350	001041				BNE	5\$	: BRANCH IF YES
4334	030352	076600				MED		: READ THE LOG JAM REG.
4335	030354	000100				RLJAM		
4336	030356	030027	000020			BIT	RO,#BIT4	: WAS BIT 4 SET IN LOG JAM?
4337	030362	001007				BNE	4\$	: BRANCH IF YES
4338	030364	012737	000020	001210		MOV	#BIT4,\$TMP5	: SAVE EXPECTED CONTENTS FOR ERROR
4339	030372	012737	000100	001206		MOV	#RLJAM,\$TMP4	: SAVE LOG JAM ADDRESS FOR ERROR
4340	030400	104022				ERROR	22	: BIT4 NOT SET IN LOG JAM
4341								: TO INDICATE KT ABORT
4342	030402				4\$:			
4343	030402	012737	030414	000004		MOV	#3\$,ERRVEC	: SET TRAP TO 4 VECTOR TO 3\$
4344	030410	005737	000001			TST	2#000001	: CAUSE TRAP TO 4 WITH ODD ADDR. ERROR
4345	030414	022626			3\$:	CMP	(SP)+,(SP)+	: CLEAN UP STACK
4346	030416	012737	001354	000004		MOV	#CPUERR,ERRVEC	: RESTORE NORMAL CPU TRAP ROUTINE
4347								: NOW CHECK FOR 250 AS LAST VEC. LOGGED
4348	030424	076600				MED		: READ LOG FLAG/INTRPT REG.
4349	030426	000104				RLFGIN		
4350	030430	030027	000250			BIT	RO,#250	: WAS VECTOR=250 RECORDED
4351	030434	001007				BNE	5\$	: BRANCH IF YES
4352	030436	012737	000250	001210		MOV	#250,\$TMP5	: SAVE EXPECTED CONTENTS FOR ERROR
4353	030444	012737	000104	001206		MOV	#RLFGIN,\$TMP4	: SAVE LOG FLAG INT ADDRESS FOR ERROR
4354	030452	104022				ERROR	22	: VECTOR 250 WAS NOT LOGGIN IN
4355	030454				5\$:			: LOG FLAG/INTERRUPT REG.
4356								
4357								
4358								
4359								
4360								
4361								
4362								
4363								
4364								
4365								
4366								
4367								
4368								
4369								
4370	030454							
4371	030454	000004				SCOPE		
4372	030456	012737	031562	001446		MOV	#TST40,NXTTST	: SAVE STARTING ADDRESS OF NEXT
4373								: TEST FOR ESCAPE ON PARITY ERRORS
4374	030464	104410				TBITO		: MAKE SURE T-BIT IS OFF FOR THIS TEST
4375	030466	042737	000001	177572	20\$:	BIC	#BITO,2#MMRO	: DISABLE MEMORY MANAGEMENT FOR THIS TEST

```

*****
*TEST 37      RED & YELLOW ZONE STACK VIOLATIONS
*
*  THE FOLLOWING TEST CHECKS THE 'RED ZONE' AND 'YELLOW ZONE' LOGIC
*  ON PAGE K506.  BITS <15:08> OF THE STACK LIMIT REGISTER AND
*  THE STACK POINTER ARE INCREMENTED FROM ZERO UP TO THE LAST
*  EXISTING MEMORY LOCATION.  FOR EACH VALUE IN BITS <15:08>
*  OF THE STACK POINTER ALL FOUR 'STACK BOUNDARY VALUES' ARE TESTED
*  FOR THE CORRECT INDICATION.  MEMORY MANAGEMENT IS DISABLED FOR
*  THIS TEST ONLY.  RELOCATION IS REENABLED FOR THE REST OF THE PROGRAM.
*****

```

†ST37:

G07

 MAINDEC-11-DOKTA-A POP 11/6X MEM. MGMT. DIAG.
 DOKTAA.P11 07-FEB-77 10:30 T37

 MACY11 27(1006) 07-FEB-77 10:37 PAGE 84
 RED & YELLOW ZONE STACK VIOLATIONS

4376	030474	005037	001202		CLR	\$TMP2	INITIALIZE ERROR PRINT COUNTER
4377	030500	012702	000010		MOV	#10, R2	INITIALIZE STACK POINTER VALUE
4378	030504	012703	177400		MOV	#-400, R3	INITIALIZE STACK LIMIT REG. VALUE
4379	030510	062702	000400	15:	ADD	#400, R2	LOAD NEW STACK POINTER VALUE IN R2
4380	030514	062703	000400		ADD	#400, R3	LOAD NEW STACK LIMIT REG VALUE IN R3
4381	030520	020227	030466		CMP	R2, #205	IS SP VALUE LESS THAN LOC. OF THIS
4382							TEST CODE?
4383	030524	002404			BLT	65	BRANCH IF YES
4384	030526	020227	031530		CMP	R2, #105	IS SP VALUE GREATER THAN LOC. OF THIS
4385							TEST CODE?
4386	030532	003001			BGT	65	BRANCH IF YES
4387	030534	000765			BR	15	OTHERWISE GO BACK AND INCREASE VALUE OF
4388							SP SO CODE IS NOT AFFECTED
4389	030536	005037	000000	65:	CLR	#0	CLEAR LOCATION 0 FOR RED ZONE CHECK
4390	030542	005037	001370		CLR	STKEXP	NO STACK VIOLATIONS EXPECTED(STKEXP=0)
4391	030546	012706	001100		MOV	#STACK, SP	INITIALIZE STACK POINTER
4392	030552	012737	031530	000004	MOV	#105, ERRVEC	SET ERROR POINTER TO 105
4393	030560	016204	177776		MOV	-2(R2), R4	SAVE STACK LOCATIONS TO BE USED
4394	030564	016205	177774		MOV	-4(R2), R5	EXIT TEST IF EITHER LOC. TIMES OUT
4395	030570	012737	030576	001110	MOV	#115, \$LPERR	SET LOOP ON ERROR POINTER TO 115
4396	030576	012737	030762	000004	MOV	#505, ERRVEC	SET ERROR POINTER TO 505
4397	030604	010206			MOV	R2, SP	LOAD STACK POINTER
4398	030606	010337	177774		MOV	R3, STKLMT	LOAD STACK LIMIT REGISTER
4399	030612	016666	177770	177770	MOV	-10(SP), -10(SP)	REFERENCE LAST 'NO VIOLATION' ADDRESS
4400	030620	005037	000000		CLR	#0	CLEAR LOCATION 0 FOR RED ZONE CHECK
4401	030624	012737	000001	001370	MOV	#1, STKEXP	EXPECT YELLOW ZONE VIOLATION(STKEXP=1)
4402	030632	012737	030640	001110	MOV	#125, \$LPERR	SET LOOP ON ERROR POINTER TO 125
4403	030640	010206			MOV	R2, SP	LOAD STACK POINTER
4404	030642	016666	177766	177766	MOV	-12(SP), -12(SP)	REFERENCE FIRST 'YELLOW ZONE' ADDRESS
4405	030650	000137	031442		JMP	605	YELLOW ZONE VIOL. EXPECTED-GOT NONE
4406	030654	012737	030662	001110	MOV	#135, \$LPERR	SET LOOP ON ERROR POINTER TO 135
4407	030662	010206			MOV	R2, SP	LOAD STACK POINTER
4408	030664	005037	000000		CLR	#0	CLEAR LOCATION 0 FOR RED ZONE CHECK
4409	030670	016666	177730	177730	MOV	-50(SP), -50(SP)	REFERENCE LAST 'YELLOW ZONE' ADDRESS
4410	030676	000137	031464		JMP	615	YELLOW ZONE VIOL. EXPECTED-GOT NONE
4411	030702	012737	030710	001110	MOV	#145, \$LPERR	SET LOOP ON ERROR POINTER TO 145
4412	030710	012737	000002	001370	MOV	#2, STKEXP	EXPECT RED ZONE VIOLATION(STKEXP=2)
4413	030716	010206			MOV	R2, SP	LOAD STACK POINTER
4414	030720	005037	000000		CLR	#0	CLEAR LOCATION 0 FOR RED ZONE CHECK
4415	030724	016237	177726	001200	MOV	-52(R2), \$TMP1	SAVE CONTENTS OF TEST LOC.
4416	030732	005166	177726		COM	-52(SP)	REFERENCE FIRST 'RED ZONE' ADDRESS
4417	030736	000137	031506		JMP	625	RED ZONE VIOL. EXPECTED-GOT NONE
4418	030742	013762	001200	177726	MOV	\$TMP1, -52(R2)	RESTORE CONTENTS OF TEST LOC.
4419	030750	010462	177776		MOV	R4, -2(R2)	RESTORE STACK LOCATION USED
4420	030754	010562	177774		MOV	R5, -4(R2)	RESTORE STACK LOCATION USED
4421	030760	000653			BR	15	BRANCH UNTIL REACH NON EXSIST. MEM.

THE CODE BELOW IS RESPONSIBLE FOR ALL OF
 THE ERROR REPORTING IN THIS TEST
 THE TEST VALUES OF THE STACK LIMIT REGISTER, STACK POINTER, AND
 THE STACK LOCATIONS TESTED ARE RESET TO THEIR NORMAL VALUES BUT
 THEIR TEST VALUES ARE RESTORED BEFORE RETURNING TO THE TEST.

H07

MAINDEC-11-DOKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 85
 DOKTAA.P11 07-FEB-77 10:30 T37 RED & YELLOW ZONE STACK VIOLATIONS

4433	030762	010637	001176		50\$:	MOV	SP,STMP0	:SAVE TEST VALUE OF STACK POINTER
4434	030766	016200	177776			MOV	-2(R2),R0	:SAVE TEST VALUES IN TEST LOC.S
4435	030772	016201	177774			MOV	-4(R2),R1	
4436	030776	010462	177776			MOV	R4,-2(R2)	:RESTORE ORIGINAL VALUES TO
4437	031002	010562	177774			MOV	R5,-4(R2)	:TEST LOCATIONS
4438	031006	005037	177774			CLR	STKLMT	:CLEAR STACK LIMIT REG. FOR ERROR REPORT
4439	031012	012706	001100			MOV	#STACK,SP	:RESET STACK POINTER FOR ERROR REPORTING
4440	031016	005737	001370			TST	STKEXP	:WAS A VIOLATION EXPECTED
4441	031022	001012				BNE	51\$	:IF SO GO DECIDE WHAT KIND
4442	031024	104041				ERROR	41	:OTHERWISE ERROR - NO VIOLATION EXPECTED
4443	031026	013706	001176			MOV	STMP0,SP	:RESTORE STACK POINTER VALUE
4444	031032	010062	177776			MOV	R0,-2(R2)	:RESTORE TEST VALUES TO TEST LOC.S
4445	031036	010162	177774			MOV	R1,-4(R2)	
4446	031042	010337	177774			MOV	R3,STKLMT	:RESTORE STACK LIMIT REG. TEST VALUE
4447	031046	000006				RTT		:RETURN FROM ERROR
4448								
4449								
4450								
4451	031050	022737	000001	001370	51\$:	CMP	#1,STKEXP	:WAS A YELLOW ZONE VIOLATION EXPECTED
4452	031056	001076				BNE	54\$	:IF NOT WE WERE EXPECTING A RED ZONE
4453	031060	005737	000000			TST	#0	:WAS LOCATION 0 CHANGED
4454	031064	001401				BEQ	52\$	:BRANCH IF IT WASN'T
4455	031066	104043				ERROR	43	:EXPECTING YELLOW ZONE - GOT RED
4456	031070	022737	000005	001202	52\$:	CMP	#5,STMP2	:HAVE 5 "ERROR LOG" ERRORS BEEN TYPED?
4457	031076	003447				BLE	53\$	:IF YES DON'T TYPE ANY MORE
4458	031100	105737	001450			TSTB	FORTY	:EXECUTING ON AN 11/40?
4459	031104	001044				BNE	53\$	:BRANCH AROUND LOG READS IF YES
4460								:SFE IF YELLOW ZONE LOGGED CORRECTLY
4461	031106	010037	001204			MOV	R0,STMP3	:SAVE THE CONTENTS OF R0
4462	031112	012737	000100	001206		MOV	#RLJAM,STMP4	:SAVE THE ADDRESS OF THE REG. IN CASE
4463								:OF AN ERROR
4464	031120	012737	010000	001210		MOV	#BIT12,STMP5	:ALSO SAVE THE EXPECTED CONTENTS
4465	031126	076600				MED		:READ THE CONTENTS OF THE ERROR LOG REG.
4466	031130	000100				RLJAM		:SPECIFIED BY "RLJAM"
4467	031132	032700	010000			BIT	#BIT12,R0	:WAS THE RIGHT BIT SET
4468	031136	001003				BNE	.+10	:BRANCH IF IT WAS
4469	031140	005237	001202			INC	STMP2	:INCREMENT ERROR PRINT COUNT
4470	031144	104022				ERROR	22	:ERROR LOG REG. DID NOT HAVE CORRECT BIT SET
4471	031146	013700	001204			MOV	STMP3,R0	:RESTORE THE CONTENTS OF R0
4472	031152	010037	001204			MOV	R0,STMP3	:SAVE THE CONTENTS OF R0
4473	031156	012737	000101	001206		MOV	#RLSERV,STMP4	:SAVE THE ADDRESS OF THE REG. IN CASE
4474								:OF AN ERROR
4475	031164	012737	000001	001210		MOV	#BIT0,STMP5	:ALSO SAVE THE EXPECTED CONTENTS
4476	031172	076600				MED		:READ THE CONTENTS OF THE ERROR LOG REG.
4477	031174	000101				RLSERV		:SPECIFIED BY "RLSERV"
4478	031176	032700	000001			BIT	#BIT0,R0	:WAS THE RIGHT BIT SET
4479	031202	001003				BNE	.+10	:BRANCH IF IT WAS
4480	031204	005237	001202			INC	STMP2	:INCREMENT ERROR PRINT COUNT
4481	031210	104022				ERROR	22	:ERROR LOG REG. DID NOT HAVE CORRECT BIT SET
4482	031212	013700	001204			MOV	STMP3,R0	:RESTORE THE CONTENTS OF R0
4483	031216	005737	000000		53\$:	TST	#0	:DID YOU GET RED ZONE?
4484	031222	001070				BNE	57\$	:RETURN DIFFERENTLY IF YES
4485	031224	013706	001176			MOV	STMP0,SP	:RESTORE STACK POINTER VALUE
4486	031230	010062	177776			MOV	R0,-2(R2)	:RESTORE TEST VALUES TO TEST LOC.S
4487	031234	010162	177774			MOV	R1,-4(R2)	

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. T37
DOKTAA.P11 07-FEB-77 10:30

MACY11 27(1006) 07-FEB-77 10:37 PAGE 86
RED & YELLOW ZONE STACK VIOLATIONS

```

4488 031240 010337 177774      MOV      R3,STKLMT      ;RESTORE STACK LIMIT REG. TEST VALUE
4489 031244 062762 000004 177774  ADD      #4,-4(R2)      ;FUDGE RETURN FROM TRAP
4490 031252 000006      RTT      ;RETURN FROM TRAP
4491
4492
4493
4494 031254 023762 001200 177726 54$:  CMP      $TMP1,-52(R2)  ;WAS INSTRUCTION ABORTED ON 'RED' VIOL.
4495 031262 001401      BEQ      55$      ;BRANCH IF IT WAS EXECUTED
4496 031264 104045      ERROR    45      ;RED ZONE DID NOT CAUSE ABORT
4497 031266 005737 001176      55$:  TST      $TMP0      ;IS TEST VALUE OF SP EQUAL TO LOC. 0
4498 031272 001401      BEQ      56$      ;STACK SHOULD BE BUILT AT 0 ON A 'RED'
4499 031274 104046      ERROR    46      ;EXPECTED RED ZONE - GOT YELLOW
4500 031276 022737 000005 001202 56$:  CMP      #5,$TMP2      ;HAVE 5 "ERROR LOG" ERRORS BEEN TYPED?
4501 031304 003437      BLE      57$      ;IF YES, DON'T TYPE ANY MORE
4502 031306 105737 001450      TSTB     FORTY      ;EXECUTING ON AN 11/40?
4503 031312 001034      BNE      57$      ;BRANCH AROUND LOG READS IF YES
4504
4505 031314 010037 001204      MOV      R0,$TMP3      ;SEE IF RED ZONE LOGGED CORRECTLY
4506 031320 012737 000100 001206  MOV      #RLJAM,$TMP4  ;SAVE THE CONTENTS OF R0
4507
4508 031326 012737 004010 001210  MOV      #004010,$TMP5 ;SAVE THE ADDRESS OF THE REG. IN CASE
4509 031334 076600      MED      ;OF AN ERROR
4510 031336 000100      RLJAM    ;ALSO SAVE THE EXPECTED CONTENTS
4511 031340 032700 004010      BIT      #004010,R0  ;READ THE CONTENTS OF THE ERROR LOG REG.
4512 031344 001003      BNE      .+10      ;SPECIFIED BY "RLJAM"
4513 031346 005237 001202      INC      $TMP2      ;WAS THE RIGHT BIT SET
4514 031352 104022      ERROR    22      ;BRANCH IF IT WAS
4515 031354 013700 001204      MOV      $TMP3,R0      ;INCREMENT ERROR PRINT COUNT
4516 031360 017737 147770 001410  MOV      #CPUERP,PCPUER ;ERROR LOG REG. DID NOT HAVE CORRECT BIT SET
4517 031366 032737 000004 001410  BIT      #BIT2,PCPUER  ;RESTORE THE CONTENTS OF R0
4518 031374 001003      BNE      57$      ;READ THE CPU ERROR REGISTER
4519 031376 005237 001202      INC      $TMP2      ;LOOK AT CONTENTS OF CPU ERROR REG.
4520 031402 104076      ERROR    76      ;WAS RED ZONE VIOLATION REPORTED
4521 031404 005737 000000      57$:  TST      #0      ;INCREMENT ERROR TYPE COUNT
4522 031410 001702      BEQ      53$      ;CPU ERROR REG.DID NOT REPORT RED ZONE
4523 031412 013706 001176      MOV      $TMP0,SP      ;DID WE GET RED ZONE?
4524 031416 010062 177776      MOV      R0,-2(R2)    ;RETURN DIFFERENTLY IF NO
4525 031422 010162 177774      MOV      R1,-4(R2)    ;RESTORE STACK POINTER VALUE
4526 031426 010337 177774      MOV      R3,STKLMT    ;RESTORE TEST VALUES TO TEST LOC.5
4527 031432 062737 000004 000000  ADD      #4,#0      ;RESTORE STACK LIMIT REG. TEST VALUE
4528 031440 000006      RTT      ;FUDGE RETURN FROM TRAP
4529
4530
4531
4532
4533 031442 005037 177774      60$:  CLR      STKLMT      ;THIS GROUP OF CODING IS ENTERED WHEN A STACK VIOLATION
4534 031446 012706 001100      MOV      #STACK,SP  ;WAS EXPECTED BUT NONE OCCURRED
4535 031452 104042      ERROR    42      ;CLEAR STACK LIMIT REG. FOR ERROR REPORT
4536 031454 010337 177774      MOV      R3,STKLMT  ;RESET STACK POINTER FOR ERROR REPORT
4537 031460 000137 030654      JMP      3$          ;YELLOW ZONE EXPECTED - GOT NONE
4538 031464 005037 177774      61$:  CLR      STKLMT      ;RESTORE STACK LIMIT REG. TEST VALUE
4539 031470 012706 001100      MOV      #STACK,SP  ;RETRUN TO TEST AT 3$
4540 031474 104042      ERROR    42      ;CLEAR STACK LIMIT REG. FOR ERROR REPORT
4541 031476 010337 177774      MOV      R3,STKLMT  ;RESET STACK POINTER FOR ERROR REPORT
4542 031502 000137 030702      JMP      4$          ;YELLOW ZONE EXPECTED - GOT NONE
4543 031506 005037 177774      62$:  CLR      STKLMT      ;RESTORE STACK LIMIT REG. TEST VALUE

```

J07

MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 87
 DQKTA.P11 07-FEB-77 10:30 T37 RED & YELLOW ZONE STACK VIOLATIONS

```

4544 031512 012706 001100      MOV      #STACK,SP      ;RESET STACK POINTER FOR ERROR REPORT
4545 031516 104044      ERROR      44      ;RED ZONE EXPECTED - GOT NONE
4546 031520 010337 177774      MOV      R3,STKLMT      ;RESTORE STACK LIMIT REG. TEST VALUE
4547 031524 000137 030742      JMP      5$      ;RETURN TO TEST AT 5$
4548
4549
4550
4551
4552

```

ONCE A NONEXISTENT MEMORY LOCATION IS REACHED,
 THE STACK POINTER IS RESET, MEMORY MANAGEMENT IS
 REENABLED, AND THE TEST IS OVER

```

4553 031530 012737 015162 000004 10$: MOV      #CPUER,ERRVEC ;RESTORE NORMAL CPU TRAP ROUTINE
4554                                ;GOT TIMEOUT, TEST IS OVER
4555 031536 012706 001100      MOV      #STACK,SP      ;RESET THE STACK POINTER
4556 031542 052737 000001 177572      BIS      #BIT0,#MM1RC ;REENABLE MEMORY MANAGEMENT
4557 031550 005037 177774      CLR      #STKLMT      ;LEAVE STACK LIMIT REG. CLEAR
4558 031554 012737 030466 001110      MOV      #20$,SLPERR ;SET LOOP POINTER TO START OF TEST
4559 031562
4560
4561
4562
4563
4564
4565
4566
4567
4568
4569
4570
4571
4572
4573
4574
4575
4576

```

 TEST 40 MEMORY MANAGEMENT REGISTERS ONLY CLOCKED ONCE IF MMRO NOT CLEARED

AS LONG AS THE 'ABORT' OR 'LOCKUP' SIGNAL IS ASSERTED (THAT IS AFTER
 AN ABORT AND UNTIL MMRO BITS <15:13> ARE CLEARED) MMRO,
 AND MM12 SHOULD NOT BE CLOCKED. THIS TEST CAUSES A NON-RESIDENT
 ABORT, SAVES THE STATUS REGISTERS, CHANGES THE VECTOR TO 10\$,
 AND THEN CAUSES A PAGE LENGTH ABORT. AT THE SECOND ABORT THE
 STATUS REGISTERS ARE COMPARED WITH THEIR FIRST CONDITIONS. IF ANY
 OF THEM CHANGE, THE OLD AND THE NEW CONDITIONS WILL BE REPORTED.

```

4577 031562
4578 031562 000004
4579 031564 012737 032002 001446      SCOPE
4580                                MOV      #TST41,NXTTST ;SAVE STARTING ADDRESS OF NEXT
4581 031572 104411      TBTR      ;TEST FOR ESCAPE ON PARITY ERRORS
4582                                ;RESTORE T-BIT IF IT WAS ON BEFORE
4583 031574 012737 031602 001106      MOV      #20$,SLPADR ;THE LAST TEST
4584                                ;SET LOOP ADDRESS POINTER TO 20$
4585 031602 012737 000000 172310 20$: MOV      #000000,KIPDR4 ;FOR ITERATION PASS
4586                                ;MAP PAGE 4 NON-RESIDENT, AND PAGE
4587 031610 012737 031650 000250      MOV      #5$,MMVEC ;LENGTH OF 1 BLOCK
4588 031616 012737 000340 000252      MOV      #340,MMVEC+2 ;SET M.M. TRAP VECTOR TO 5$
4589 031624 012737 031632 001110      MOV      #1$,SLPERR ;SET PRIORITY TO 7 GOTO KERNEL MODE
4590 031632 013700 100000 1$: MOV      #100000,R0 ;SET LOOP ON ERROR POINTER TO 1$
4591                                ;TRY TO READ THRU PAGE 4
4592                                ;THIS PAGE NON-RESIDENT SHOULD CAUSE
4593 031636 012737 031672 000250 2$: MOV      #10$,MMVEC ;ABORT AND TRAP TO 5$.
4594 031644 013700 100100      MOV      #100100,R0 ;SET M.M. TRAP VECTOR TO 10$
4595                                ;TRY TO READ FROM BLOCK 2 OF PAGE 4
4596                                ;THIS SHOULD ABORT AGAIN BUT NONE
4597                                ;OF THE MEMORY MANAGEMENT STATUS
4598                                ;REGISTERS SHOULD BE CLOCKED THIS TIME.
4599

```

K07

MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG.
 DQKTA.P11 07-FEB-77 10:30 T40

MACY11 27(1006) 07-FEB-77 10:37 PAGE 88
 MEMORY MANAGEMENT REGISTERS ONLY CLOCKED ONCE IF MMRO NOT CLEARED

4600	031650	013737	177572	001404	55:	MOV	MMRO, PMMR0	; READ MEMORY MANAGEMENT REGISTER 0
4601	031656	013737	177576	001406		MOV	MMR2, PMMR2	; READ MEMORY MANAGEMENT REGISTER 2
4602	031664	012716	031636			MOV	#25, (KSP)	; CHANGE RETURN ADDRESS TO 25
4603	031670	000006				RTT		; GO BACK TO 25 AND CAUSE PAGE FAULT
4604								
4605								
4606	031672	012716	031700		105:	MOV	#165, (KSP)	; CHANGE RETURN ADDRESS TO 165
4607	031676	000006				RTT		; RETURN TO 165 AND CONTINUE PROGRAM
4608	031700	005037	001204		165:	CLR	STMP3	; ERROR COUNTER, 3 POSSIBLE CMP FAILURES
4609	031704	013737	177572	001176		MOV	MMRO, STMP0	; READ MEMORY MANAGEMENT REGISTER 0
4610	031712	013737	177576	001202		MOV	MMR2, STMP2	; READ MEMORY MANAGEMENT REGISTER 2
4611	031720	023737	001202	001406		CMP	STMP2, PMMR2	; SEE IF MMR2 CHANGED
4612	031726	001402				BEQ	125	; BRANCH IF MMR2 DIDN'T CHANGE
4613	031730	005237	001204			INC	STMP3	; ONE COMPARE FAILURE
4614	031734	023737	001176	001404	125:	CMP	STMP0, PMMR0	; SEE IF MMRO CHANGED
4615	031742	001402				BEQ	135	; BRANCH IF MMRO DIDN'T CHANGE
4616	031744	005237	001204			INC	STMP3	; ANOTHER COMPARE FAILURE
4617	031750	005737	001204		135:	TST	STMP3	; WERE THERE ANY ERRORS ON THIS TEST
4618	031754	001401				BEQ	195	; BRANCH IF NO ERRORS
4619	031756	104050				ERROR	50	; AT LEAST ONE M.M. REG CHANGED
4620	031760	042737	160556	177572	195:	BIC	#160556, MMRO	; CLEAR ALL ERROR BITS IN MMRO
4621	031766	012737	015432	000250		MOV	#MMTRAP, MMVEC	; PUT BACK REGULAR M.M. TRAP ROUTINE
4622	031774	012737	031602	001110		MOV	#205, SLPERR	; SET LOOP POINTER TO START OF TEST

L07

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 89
 DOKTAA.P11 07-FEB-77 10:30 T41 ERROR-ON-ERROR, KT ABORT & RED ZONE

4623
4624
4625
4626
4627
4628
4629
4630
4631
4632
4633
4634
4635
4636
4637
4638
4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4660
4661
4662
4663
4664
4665
4666
4667
4668
4669
4670
4671
4672
4673
4674
4675
4676
4677
4678

 *TEST 41 ERROR-ON-ERROR, KT ABORT & RED ZONE
 * THIS TEST CREATES AN "ERROR-ON-ERROR" SITUATION BY CAUSING
 * A KT ABORT (READ-ONLY VIOL.) IN USER MODE FOLLOWED BY
 * ODD ADDRESS VIOL. BY USER STK. PTR. THE PSW PICKED UP FROM
 * "MMVEC+2" PUTS IT IN USER MODE. THEN WHEN TRYING TO
 * PUSH THE OLD PSW AND PC ON THE USER STACK, AN ODD ADDR.
 * VIOLATION OCCURS (KSP=1100, STACK LIMIT REG. = 703).
 * THE KERNEL STACK POINTER SHOULD BE FORCED TO THE VALUE 4,
 * BUILDING A STACK AT LOC.S 082 ON THE 11/6X. ON THE 11/40
 * THE USER STACK POINTER IS FORCED TO 4, BUT THE KERNEL STACK
 * POINTER IS USED TO BUILD THE STACK AT LOCATIONS 1074 &
 * 1076. THE PSW THAT WILL BE SAVED ON THE STACK
 * SHOULD BE THE ONE PRESENT AT THE TIME OF THE SECOND ERROR
 * (THE ONE FROM "MMVEC+2") FOR THE 11/6X, OR THE PSW PRESENT
 * AT THE TIME OF THE FIRST ERROR FOR THE 11/40
 * THE PC SAVED ON THE STACK, IN BOTH CASES
 * SHOULD BE THE UPDATED PC FROM THE FIRST ERROR (THE KT
 * ABORT INSTRUCTION).
 * ALSO, MMRO SHOULD RECORD A R/O VIOL. & USER PAGE 0,
 * MMR2 SHOULD LOCK UP THE PC OF THE KT ABORT INSTRUCTION,
 * AND THE WHAMI AND LOG WHAMI REGISTERS SHOULD RECORD AN
 * "ERROR-ON-ERROR" ON THE 11/6X.

TST41:

SCOPE
 MOV #TST42,NXTTST
 .EQUIV BIT4,TBIT
 TBIT0
 MOV #340,ERRVEC+2
 MOV #000,UIPAR0
 MOV #200,UIPAR1
 MOV #400,UIPAR2
 MOV #600,UIPAR3
 MOV #7600,UIPAR7
 MOV #77406,UIPDR0
 MOV #77406,UIPDR1
 MOV #77406,UIPDR2
 MOV #77406,UIPDR3
 MOV #77406,UIPDR7
 MOV #15,SLPERR
 MOV #105,ERRVEC
 MOV #140017,MMVEC+2
 MOV #600,UIPAR4
 MOV #77402,UIPDR4
 MOV #140000,PSW
 MOV #703,USP

SAVE STARTING ADDRESS OF NEXT
 TEST FOR ESCAPE ON PARITY ERRORS
 BIT 4 OF P.S. IS T-BIT TRAPPING BIT
 MAKE SURE T-BIT IS OFF FOR THE NEXT 3 TESTS
 SET PRIORITY OF ERRVEC TO 7
 AND MAKE NEW MODE KERNEL
 MAP USER PAGE 0 TO PHYS. 0
 MAP USER PAGE 1 TO 4K - 8K
 MAP USER PAGE 2 TO 8K - 12K
 MAP USER PAGE 3 TO 12K - 16K
 MAP USER PAGE 7 TO I/O PAGE
 MAKE USER PAGE 0 200 BLKS, R/W
 MAKE USER PAGE 1 200 BLKS, R/W
 MAKE USER PAGE 2 200 BLKS, R/W
 MAKE USER PAGE 3 200 BLKS, R/W
 MAKE USER PAGE 7 200 BLOCKS, R/W
 SET LOOP ON ERROR POINTER TO 15
 SET ERRVEC TO 105
 LOAD NEW PSW INTO MMVEC+2
 MAP USER PAGE 4 TO 12K
 MAKE USER PAGE 4 200 BLOCKS, READ-ONLY
 GO TO USER MODE
 SETUP USER STACK POINTER
 TO CAUSE ODD ADDR. ERROR
 CREATE KT ABORT (R/O VIOL.) WHICH
 PICKS UP A USER PSW AND THEN
 CAUSES A ODD ADDR. VIOLATION.
 THE DOUBLE ERROR SHOULD BE PROCESSED
 AND THEN THE PROGRAM SHOULD CONTINUE

15:

INC 2#100000

PC	Instruction	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418
----	-------------	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

N07

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG.
DOKTAA.P11 07-FEB-77 10:30 T4.

MACY11 27(1006) 07-FEB-77 10:37 PAGE 91
ERROR-ON-ERROR, KT ABORT & RED ZONE

4735	032454	052737	140000	177776		BIS	#140000,PSW	:GO TO USER MODE
4736	032462	010604				MOV	USP,R4	:READ THE USER STACK POINTER
4737	032464	042737	140000	177776		BIC	#140000,PSW	:GO BACK TO KERNEL MODE
4738	032472	022704	000004			CMP	#4,R4	:WAS USER STK. PTR. FORCED TO 4 ON 11/40?
4739	032476	001404				BEQ	22\$	:BRANCH IF YES
4740	032500	005237	001200			INC	\$TMP1	:USP ON 11/40 IS NOT CORRECT
4741	032504	010401				MOV	R4,R1	:SAVE VALUE OF USP FOR TYPEOUT
4742	032506	000413				BR	13\$	:DON'T BOTHER CHECKING KSP IF USP WAS WRONG
4743	032510	022701	001074		22\$:	CMP	#KERSTK-4,R1	:WAS PC & PSW SAVED ON 11/40 KERNEL STACK?
4744	032514	001410				BEQ	13\$	:BRANCH IF YES
4745	032516	005237	001200			INC	\$TMP1	:PC & PSW NOT SAVED ON 11/40 KERNEL STACK
4746	032522	000405				BR	13\$	:BRANCH AROUND 11/6X KSP CHECK
4747	032524				23\$:			
4748	032524	022701	000000			CMP	#0,R1	:THE KER. STK. POINTER
4749								:SHOULD HAVE BEEN = 0
4750								:AFTER ERR-ON-ERR TRAP TO LOC. 4
4751	032530	001402				BEQ	13\$	:BRANCH IF KSP IS CORRECT
4752	032532	005237	001200			INC	\$TMP1	:KSP IS NOT CORRECT
4753	032536	022737	032166	001406	13\$:	CMP	#1\$,PMMR2	:SEE IF PMMR2 EQUALS ADDR. OF KT ABORT
4754	032544	001402				BEQ	14\$	:BRANCH IF ADDRESS IS CORRECT
4755	032546	005237	001200			INC	\$TMP1	:PMMR2 HAS THE WRONG ADDRESS
4756	032552	005737	001200		14\$:	TST	\$TMP1	:DID ANY OF THE COMPARES FAIL?
4757	032556	001401				BEQ	15\$	:BRANCH IF ALL COMPARES SUCCEEDED
4758	032560	104047				ERROR	47	:AT LEAST ONE COMPARE FAILED
4759								:IF USER STK. PTR. WRONG ON 11/40
4760								:THE KERNEL STK. PTR. WAS NOT CHECKED
4761	032562	052737	000340	177776	15\$:	BIS	#340,PSW	:NORMAL PRIORITY IS 7
4762	032570	052737	140000	177776		BIS	#140000,PSW	:GO TO USER MODE TO RESET PTR
4763	032576	012706	000700			MOV	#700,USP	:RESTORE USER STK PTR TO 700
4764	032602	042737	140000	177776		BIC	#140000,PSW	:RETURN TO KERNEL MODE
4765	032610	012737	000340	000006		MOV	#340,ERRVEC+2	:RESTORE CORRECT PS TO ERROR VECTOR
4766	032616	012737	032014	001110		MOV	#20\$,SLPERR	:SET LOOP POINTER TO START OF TEST
4767	032624	042737	160556	177572		BIC	#160556,MMR0	:CLEAR ERROR CONDITION IN MMR0
4768	032632	012737	015162	000004		MOV	#CPUER,ERRVEC	:RESTORE NORMAL CPU TRAP ROUTINE

4769
4770
4771
4772
4773
4774
4775
4776
4777
4778
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799
4800
4801
4802
4803
4804
4805
4806
4807
4808
4809
4810
4811
4812
4813
4814
4815
4816
4817
4818
4819
4820
4821
4822
4823
4824

[illegible]

[illegible]

```

;*****
;TEST 43          SECOND BIT TEST OF MEMORY MANAGEMENT REGISTER 2
;
;  THIS TEST MAPS USER PAGES 3,4,5, AND 6 TO 12K READ-ONLY.
;  A VIRTUAL ADDRESS IS THEN FORMED IN REGISTER 1 USING BITS
;  <15:13> TO SELECT PAR/PDRS 3 THRU 6 AND A "1" WHICH IS SHIFTED
;  THRU BITS <12:1>.  THE INSTRUCTION "INC (R1)" (CODED AS 005211)
;  IS THEN PLACED AT THE VIRTUAL ADDRESS IN R1 CAUSING A KT-
;  READ ONLY VIOLATION WHEN EXECUTED.  THE CONTENTS OF MMR2 ARE
;  TESTED ALONG WITH BITS <06:05> AND <03:01> OF MMRO.

```

```

*****
TST43:
      SCOPE
      MOV      #TST44,NXTTST      ;SAVE STARTING ADDRESS OF NEXT
                                   ;TEST FOR ESCAPE ON PARITY ERRORS
      MOV      #2,$TIMES          ;;DO 2 ITERATIONS

20$:
      MOV      #77406,KIPDR0      ;MAKE KERNEL I PAGE 0 200 BLOCKS, R/W
      MOV      #77406,KIPDR1      ;MAKE KERNEL I PAGE 1 200 BLOCKS, R/W
      MOV      #77406,KIPDR2      ;MAKE KERNEL I PAGE 2 200 BLOCKS, R/W
      MOV      #77406,KIPDR3      ;MAKE KERNEL I PAGE 3 200 BLOCKS, R/W
      MOV      #77406,KIPDR7      ;MAKE KERNEL I PAGE 7 200 BLOCKS, R/W
      MOV      #000,KIPAR0        ;MAP KERNEL I PAGE 0 TO 0 - 4K
      MOV      #200,KIPAR1        ;MAP KERNEL I PAGE 1 TO 4K - 8K
      MOV      #400,KIPAR2        ;MAP KERNEL I PAGE 2 TO 8K - 12K
      MOV      #600,KIPAR3        ;MAP KERNEL I PAGE 3 TO 12K - 16K
      MOV      #7600,KIPAR7       ;MAP KERNEL I PAGE 7 TO THE I/O PAGE

```

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG.
DOKTAA.P11 07-FEB-77 10:30 T43

MACY11 27(1006) 07-FEB-77 10:37 PAGE 94
SECOND BIT TEST OF MEMORY MANAGEMENT REGISTER 2

4881	033270	012737	000001	177572	MOV	#BIT0,MMR0	;ENABLE 18-BIT RELOCATION IF NOT ON
4882	033276	012700	077406		MOV	#77406,R0	
4883	033302	010037	172310		MOV	R0,KIPDR4	;MAP KERNEL PAGE 4 READ-WRITE
4884	033306	010037	172312		MOV	R0,KIPDR5	;MAP KERNEL PAGE 5 READ-WRITE
4885	033312	010037	172314		MOV	R0,KIPDR6	;MAP KERNEL PAGE 6 READ-WRITE
4886	033316	012700	000600		MOV	#600,R0	
4887	033322	010037	172350		MOV	R0,KIPAR4	;MAP KERNEL PAGE 4 TO 12K
4888	033326	010037	172352		MOV	R0,KIPAR5	;MAP KERNEL PAGE 5 TO 12K
4889	033332	010037	172354		MOV	R0,KIPAR6	;MAP KERNEL PAGE 6 TO 12K
4890	033336	012737	077402	177606	MOV	#77402,UIPDR3	;MAP USER PAGE 3 READ-ONLY
4891	033344	012737	077402	177610	MOV	#77402,UIPDR4	;MAP USER PAGE 4 READ-ONLY
4892	033352	012737	077402	177612	MOV	#77402,UIPDR5	;MAP USER PAGE 5 READ-ONLY
4893	033360	012737	077402	177614	MOV	#77402,UIPDR6	;MAP USER PAGE 6 READ-ONLY
4894	033366	012737	000000	177640	MOV	#000,UIPAR0	;MAP USER PAGE 0 TO 0-4K
4895	033374	012737	000200	177642	MOV	#200,UIPAR1	;MAP USER PAGE 1 TO 4K
4896	033402	012737	000400	177644	MOV	#400,UIPAR2	;MAP USER PAGE 2 TO 8K
4897	033410	012737	000600	177646	MOV	#600,UIPAR3	;MAP USER PAGE 3 TO 12K
4898	033416	012737	000600	177650	MOV	#600,UIPAR4	;MAP USER PAGE 4 TO 12K
4899	033424	012737	000600	177652	MOV	#600,UIPAR5	;MAP USER PAGE 5 TO 12K
4900	033432	012737	000600	177654	MOV	#600,UIPAR6	;MAP USER PAGE 6 TO 12K
4901	033440	012737	033476	001110	MOV	#3\$, \$LPERR	;SET LOOP ON ERROR POINTER TO 3\$
4902	033446	012700	140000		MOV	#140000,R0	;THIS WILL FORCE USER MODE WHEN USED
4903							AS A PROCESSOR STATUS
4904	033452	012704	060000		MOV	#60000,R4	;SETUP BITS <15:13> IN R4 SO V.A. WILL
4905							INITIALLY SELECT PAR/PDR 3
4906	033456	012737	033522	000250	MOV	#10\$,MMVEC	;SET M.M. TRAP VECTOR TO 10\$
4907	033464	012737	000340	000252	MOV	#340,MMVEC+2	;MAKE SURE TRAP TAKES YOU TO KERNEL
4908	033472	012701	000002		MOV	#2,R1	;PUT TEST BIT AT STARTING POSITION(BIT 1)
4909	033476	042701	160000		BIC	#160000,R1	;CLEAR BITS <15:13> SO APP SELECT BITS CAN
4910							BE RESET
4911	033502	050401			BIS	R4,R1	;SET BITS <15:13> TO SELECT PAR/PDR BEING USED
4912	033504	011137	001176		MOV	(R1), \$TMP0	;SAVE CONTENTS OF TEST LOCATION
4913	033510	012711	005211		MOV	#5211,(R1)	;PUT "ABORT INSTRUCTION" AT TEST LOC.
4914	033514	010046			MOV	R0,-(KSP)	;PUSH USER PS ON STACK
4915	033516	010146			MOV	R1,-(KSP)	;PUSH USER'S VIRTUAL PC ON STACK
4916	033520	000002			RTI		;RETURN TO USER MODE
4917							;THE FIRST INSTRUCTION FETCH WILL
4918							;CAUSE A NON-RESIDENT ABORT AND LOCK
4919							;MMR2 SO THAT ALL BITS CAN BE CHECKED.
4920							
4921							
4922							
4923	033522	062706	000004		ADD	#4,KSP	;CLEAN UP STACK FOR NEXT TIME
4924	033526	013711	001176		MOV	\$TMP0,(R1)	;RESTORE CONTENTS OF TEST LOCATION
4925	033532	013737	177576	001406	MOV	MMR2,PMMR2	;READ MMR2 TO TEMP LOCATION
4926	033540	013737	177572	001404	MOV	MMR0,PMMR0	;READ MMR0 TO TEMP LOCATION
4927	033546	020137	001406		CMP	R1,PMMR2	;SEE IF MMR2 LOCKED CORRECT V. A.
4928	033552	001401			BEQ	11\$	;BRANCH IF MMR2 HAS RIGHT V.A.
4929	033554	104053			ERROR	53	;WRONG V.A.
4930	033556	005002			CLR	R2	;R2 WILL GET THE VIRTUAL PAGE NO.
4931	033560	010103			MOV	R1,R3	;COPY VIRTUAL ADDRESS INTO R3
4932	033562	012705	000003		MOV	#3,R5	;COMBINED LEFT SHIFT <R2,R3> 3 BITS
4933	033566	000241			CLC		
4934	033570	006303			ASL	R3	
4935	033572	006102			ROL	R2	
4936	033574	077504			SOB	R5,4\$	

E08

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG.
DOKTAA.P11 07-FEB-77 10:30 T43

MACY11 27(1006) 07-FEB-77 10:37 PAGE 95
SECOND BIT TEST OF MEMORY MANAGEMENT REGISTER 2

4937	033576	006102				ROL	R2	;ADJUST PAGE NUMBER
4938	033600	052702	020141			BIS	#020141,R2	;SET OTHER EXPECTED BITS IN MMRO
4939	033604	020237	001404			CMP	R2,PMRO	;SEE IF MMRO RECORDED CORRECT PAGE NO.
4940	033610	001401				BEQ	12\$	;BRANCH IF PAGE NUMBER WAS CORRECT
4941	033612	104054				ERROR	54	;WRONG PAGE NO. IN MMRO
4942	033614	042737	160556	177572	12\$:	BIC	#160556,MMRO	;CLEAR ALL ERROR BITS IN MMRO
4943	033622	032701	010000			BIT	#BIT12,R1	;HAS BIT BEEN SHIFTED ALL THE WAY THRU
4944	033626	001002				BNE	13\$	;BRANCH IF IT WAS
4945	033630	006301				ASL	R1	;SHIFT BIT ONE POSITION LEFT
4946	033632	000721				BR	3\$	;GO AND TRY THIS VIRTUAL ADDRESS
4947	033634	062704	020000		13\$:	ADD	#20000,R4	;INCREASE R4 TO SELECT NEXT PAR/PDR
4948	033640	022704	160000			CMP	#160000,R4	;HAVE ALL PAGES(3,4,5,6) BEEN USED
4949	033644	001312				BNE	2\$	;CONTINUE IF THEY HAVEN'T
4950	033646	012737	015432	000250		MOV	#MMTRAP,MMVEC	;PUT BACK REGULAR M.M. TRAP ROUTINE
4951	033654	012737	033174	001110		MOV	#20\$,SLPERR	;SET LOOP POINTER TO START OF TEST

.SBTTL ***** ENTRY POINT 6 --- STARTING ADDRESS 224 *****
.SBTTL ***** W BIT LOGIC TEST AND DUAL MAPPING TESTS *****

THIS GROUP OF TESTS CHECKS OUT THE W-BIT LOGIC
AND THEN USES THAT LOGIC TO VERIFY THAT THERE IS NO
DUAL MAPPING OF PAR/PDR PAIRS BETWEEN GROUPS OR INSIDE A GROUP.
ALL W-BITS ARE SET TO ENSURE THAT THERE ISN'T A BAD
CHIP IN THE PDR'S.

TEST 44 TEST W-BIT LOGIC

THIS TEST CHECKS ALL THE LOGIC FOR THE W-BIT ON
'K504'. THE BITS ARE SET ONE AT A TIME THEN A REFERENCE
TO THAT PAGE CAUSING AN ABORT IS MADE TO SEE IF THE BIT REMAINS
SET. LAST EITHER THE PAR OR PDR IS WRITTEN TO SEE THAT THE
BIT IS CLEARED DURING A PAR OR PDR LOAD.

TST44:

4977	033662					SCOPE		
4978	033662	000004				MOV	#TST45,NXTTST	;SAVE STARTING ADDRESS OF NEXT
4979	033664	012737	034342	001446				;TEST FOR ESCAPE ON PARITY ERRORS
4980								;RESTORE T-BIT IF IT WAS ON BEFORE THE
4981	033672	104411				TBITR		;LAST 3 TESTS
4982								
4983	033674	012737	034026	001106	ENTPT6:	MOV	#20\$,SLPADR	;SET LOOP ADDRESS POINTER TO 20\$
4984	033702	012737	034026	001110		MOV	#20\$,SLPERR	;SET LOOP ON ERROR POINTER TO 20\$
4985	033710	012737	000044	001102		MOV	#44,\$TSTNM	;LOAD TEST NUMBER INTO MEMORY
4986	033716	013777	001102	145216		MOV	\$TSTNM,\$DISPLAY	;DISPLAY TEST NUMBER FOR THIS TEST
4987	033724	012737	077406	172300		MOV	#77406,KIPDR0	;MAKE KERNEL I PAGE 0 200 BLOCKS, R/W
4988	033732	012737	077406	172302		MOV	#77406,KIPDR1	;MAKE KERNEL I PAGE 1 200 BLOCKS, R/W
4989	033740	012737	077406	172304		MOV	#77406,KIPDR2	;MAKE KERNEL I PAGE 2 200 BLOCKS, R/W
4990	033746	012737	077406	172306		MOV	#77406,KIPDR3	;MAKE KERNEL I PAGE 3 200 BLOCKS, R/W
4991	033754	012737	077406	172316		MOV	#77406,KIPDR7	;MAKE KERNEL I PAGE 7 200 BLOCKS, R/W
4992	033762	012737	000000	172340		MOV	#000,KIPAR0	;MAP KERNEL I PAGE 0 TO 0 - 4K

MAINDEC-11-DOKTA-A POP 11/6X MEM. MGMT. DIAG.
DOKTAA.P11 07-FEB-77 10:30 T44

MACY11 27(1006) 07-FEB-77 10:37 PAGE 96
TEST 4-BIT LOGIC

4993	033770	012737	000200	172342	
4994	033776	012737	000400	172344	
4995	034004	012737	000600	172346	
4996	034012	012737	007600	172356	
4997	034020	012737	000001	177572	
4998					
4999	034026	012737	000006	172310	20\$:
5000	034034	012737	000600	172350	
5001	034042	012737	034332	000250	
5002	034050	012737	034056	001110	
5003	034056	012737	012345	100000	1\$:
5004	034064	013737	172310	001176	
5005	034072	032737	000100	001176	
5006	034100	001001			
5007	034102	104055			
5008	034104	012737	034112	001110	2\$:
5009	034112	012737	012345	100100	12\$:
5010	034120	013737	172310	001176	
5011	034126	032737	000100	001176	
5012	034134	001001			
5013	034136	104056			
5014	034140	012737	034146	001110	3\$:
5015	034146	112737	000004	172310	13\$:
5016	034154	013737	172310	001176	
5017	034162	022737	000004	001176	
5018	034170	001401			
5019	034172	104057			
5020	034174	012737	034202	001110	7\$:
5021	034202	013737	177776	001176	17\$:
5022	034210	013737	001176	177776	
5023	034216	013737	172316	001176	
5024	034224	105737	001450		
5025	034230	001406			
5026	034232	032737	000100	001176	
5027	034240	001007			
5028	034242	104060			
5029	034244	000405			
5030	034246				21\$:
5031	034246	032737	000100	001176	
5032	034254	001401			
5033	034256	104060			
5034	034260	012737	034266	001110	8\$:
5035	034266	012737	077406	172310	18\$:
5036					
5037	034274	013737	172316	001176	
5038	034302	032737	000100	001176	
5039	034310	001401			
5040	034312	104061			
5041	034314	012737	034026	001110	9\$:
5042	034322	012737	015432	000250	
5043	034330	000404			
5044					
5045	034332	042737	160556	177572	10\$:
5046	034340	000002			
5047					
5048					

```

MOV      #200,KIPAR1
MOV      #400,KIPAR2
MOV      #600,KIPAR3
MOV      #7600,KIPAR7
MOV      #BIT0,MMRO
.EQUIV   BIT6,4BIT
MOV      #00006,KIPDR4
MOV      #600,KIPAR4
MOV      #10$,MMVEC
MOV      #1$,SLPERR
MOV      #12345,2#100000
MOV      KIPDR4,$TMP0
BIT      #4BIT,$TMP0
BNE      2$
ERROR    5$
MOV      #12$,SLPERR
MOV      #12345,2#100100
MOV      KIPDR4,$TMP0
BIT      #4BIT,$TMP0
BNE      3$
ERROR    5$
MOV      #13$,SLPERR
MOV      #004,KIPDR4
MOV      KIPDR4,$TMP0
CMP      #00004,$TMP0
BEQ      7$
ERROR    5$
MOV      #17$,SLPERR
MOV      PSW,$TMP0
MOV      $TMP0,PSW
MOV      KIPDR7,$TMP0
TSTB     FORTY
BEQ      21$
BIT      #4BIT,$TMP0
BNE      8$
ERROR    6$
BR       8$

BIT      #4BIT,$TMP0
BEQ      8$
ERROR    6$
MOV      #18$,SLPERR
MOV      #77406,KIPDR4

MOV      KIPDR7,$TMP0
BIT      #4BIT,$TMP0
BEQ      9$
ERROR    6$
MOV      #20$,SLPERR
MOV      #MMTRAP,MMVEC
BR       TST45

BIC      #160556,MMRO
RTI

```

```

:MAP KERNEL 1 PAGE 1 TO 4K - 8K
:MAP KERNEL 1 PAGE 2 TO 8K - 12K
:MAP KERNEL 1 PAGE 3 TO 12K - 16K
:MAP KERNEL 1 PAGE 7 TO THE I/O PAGE
:ENABLE 18-BIT RELOCATION IF NOT ON
:LABEL 'WBIT' IS EQUIVALENT TO BIT 6
:PAGE 4 HAS A PAGE LENGTH OF 1
:MAP PAGE 4 TO 12K
:SET M.M. TRAP VECTOR TO 105
:SET LOOP ON ERROR POINTER TO 115
:WRITE INTO PAGE 4
:READ PDR 4 TO CHECK W-BIT
:SEE IF W-BIT IS SET
:BRANCH IF W-BIT IS SET
:W-BIT NOT SET
:SET LOOP ON ERROR POINTER TO 125
:WRITE INTO PAGE 4, BLOCK 2
:READ PDR 4 TO SEE IF W-BIT REMAINED SET
:SEE IF W-BIT REMAINED SET
:BRANCH IF W-BIT IS SET
:W-BIT DID NOT REMAIN SET
:SET LOOP ON ERROR POINTER TO 135
:CHANGE PAGE 4'S ACCESS CONTROL FIELD
:READ PDR 4 TO CHECK W-BIT CLEARING
:SEE IF W-BIT GOT CLEARED ON PDR WRITE
:BRANCH IF W-BIT GOT CLEARED
:W-BIT DIDN'T GET CLEARED ON PDR WRITE
:SET LOOP ON ERROR POINTER TO 175
:READ PROCESSOR STATUS WORD
:WRITE TO PSW TO CHECK 'GOOD DATA L'
:READ PDR 7 TO CHECK W-BIT
:EXECUTING ON AN 11/40?
:BRANCH IF NO
:DID W-BIT SET FOR 11/40 PSW WRITE?
:BRANCH IF W-BIT SET
:W-BIT DIDN'T SET ON 11/40 P~W WRITE

:;DID W-BIT STAY 0 FOR 11/6X PSW WRITE?
:BRANCH IF W-BIT WAS NOT SET
:W-BIT DID GET SET ON 11/6X
:SET LOOP ON ERROR POINTER TO 185
:LOAD AN INTERNAL REGISTER, SHOULDN'T
:AFFECT PAGE 7'S W-BIT
:READ PDR 7 TO CHECK W-BIT AGAIN
:SEE IF W-BIT CLEAR
:BRANCH IF W-BIT IS STILL CLEAR
:W-BIT SET AFTER WRITING PDR 4 ON 11/6X
:SET LOOP POINTER TO START OF TEST
:RESTORE NORMAL M.M. TRAP ROUTINE
:;BRANCH TO NEXT TEST

:;CLEAR MMIO FOR NEXT READ OR WRITE
:RETURN TO TEST AND CHECK A OR W BIT~

```


G08

 MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG.
 DOKTAA.P11 07-FEB-77 10:30 T44

 MACY11 27(1006) 07-FEB-77 10:37 PAGE 97
 TEST W-BIT LOGIC

 5049
 5050
 5051
 5052
 5053
 5054
 5055
 5056
 5057
 5058
 5059
 5060
 5061
 5062
 5063
 5064
 5065
 5066
 5067
 5068
 5069
 5070
 5071
 5072
 5073
 5074
 5075
 5076
 5077
 5078
 5079
 5080
 5081
 5082
 5083
 5084
 5085
 5086
 5087
 5088
 5089
 5090
 5091
 5092
 5093
 5094
 5095
 5096
 5097
 5098
 5099
 5100
 5101
 5102
 5103
 5104

034342			
034342	000004		
034344	012737	035240	001446
034352	012737	000000	172340
034360	012737	000200	172342
034366	012737	000400	172344
034374	012737	000600	172346
034402	012737	000600	172350
034410	012737	000600	172352
034416	012737	000600	172354
034424	012737	000600	172356
034432	012737	000000	177640
034440	012737	000200	177642
034446	012737	000400	177644
034454	012737	000600	177646
034462	012737	000600	177650
034470	012737	000600	177652
034476	012737	000600	177654
034504	012737	007600	177656
034512	012737	034522	001110
034520	104410		
034522	012737	034612	001110
034530	012737	000000	177776
034536	012703	172300	
034542	012704	000010	

```

*
* THESE NEXT TWO (2) TESTS
* CHECK FOR DUAL MAPPING
* AMONG GROUPS OR INSIDE A GROUP OF PAR/PDR'S. THIS IS DONE BY
* WRITING ONLY ONE PAGE IN A MODE OF OPERATION THEN CHECKING ALL
* PDR'S TO SEE THAT ONLY THE ONE UNDER TEST HAS ITS W
* BIT SET. THIS TEST IS RUN IN ALL MODES,
* SO THAT ALL THE REGISTER
* PAIRS ARE TESTED.
*
*****
*TEST 45      DUAL MAPPING KERNEL MODE SPACE
*
* THIS TEST STARTS BY LOADING ALL THE P.D.R.'S WITH 77406
* (4K PAGE READ/WRITE). THEN THE VIRTUAL ADDRESS IS SET TO
* 010200 AND THAT WORD IS WRITTEN INTO ITSELF. THIS WRITE WILL
* SET THE W BIT IN THE KERNEL SPACE P.D.R. UNDER TEST. NOW ALL OF THE
* P.D.R.'S ARE COMPARED WITH 77406 AND ANY THAT MATCH, EXCEPT THE
* ONE THAT IS UNDER TEST, ARE REPORTED AS DUAL MAPPING ERRORS.
* KERNEL PAR7 IS MAPPED TO I/O PAGE DURING THE WRITE SINCE
* WRITING TO REGISTERS INTERNAL TO THE CPU ON THE 11/6X WILL NOT SET THE W BIT
* AND ANY OTHER EXISTING ADDRESS CAN'T BE GUARANTEED FOR THE I/O PAGE
*
*****
*ST45:
* SCOPE
* MOV      #TST46,NXTTST      ;SAVE STARTING ADDRESS OF NEXT
*                               ;TEST FOR ESCAPE ON PARITY ERRORS
*
* THE FOLLOWING CODE IS USED TO INITIALIZE THE PAR'S FOR
* THE NEXT TESTS, SO THAT THE CODE WILL RUN WITH MEMORY
* MANAGEMENT FULLY ENABLED.
*
* MOV      #0,KIPAR0
* MOV      #200,KIPAR1
* MOV      #400,KIPAR2
* MOV      #600,KIPAR3
* MOV      #600,KIPAR4
* MOV      #600,KIPAR5
* MOV      #600,KIPAR6
* MOV      #7600,KIPAR7
* MOV      #0,UIPAR0
* MOV      #200,UIPAR1
* MOV      #400,UIPAR2
* MOV      #600,UIPAR3
* MOV      #600,UIPAR4
* MOV      #600,UIPAR5
* MOV      #600,UIPAR6
* MOV      #7600,UIPAR7
* MOV      #20$,SLPERR
* TBITO
*
* SET LOOP ON ERROR POINTER TO 20$
* MAKE SURE THAT T-BIT IS OFF FOR
* THE DUAL MAPPING TESTS
* SET LOOP ON ERROR POINTER TO 21$
* GO TO KERNEL MODE
* LOAD FIRST ADDRESS OF KERNEL PDR'S
* THAT WILL BE TESTED IN THIS TEST
* TEST THE NEXT EIGHT PDR'S
*
* MOV      #21$,SLPERR
* MOV      #00000,PSW
* MOV      #KIPDR0,R3
*
* MOV      #10,R4

```

H08

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG.
DOKTAA.P11 07-FEB-77 10:30 T45

MACY11 27(1006) 07-FEB-77 10:37 PAGE 98
DUAL MAPPING KERNEL MODE SPACE

5105	034546	012705	010200		MOV	#10200,R5	;LOAD STARTING VIRTUAL ADDRESS INTO R5
5106	034552	012700	077406	19\$:	MOV	#77406,R0	;ALL PAGES WILL BE ACF=6
5107	034556	005037	001176		CLR	\$TMP0	;CLEAR CORRECT PDR SET INDICATOR
5108	034562	012702	000010		MOV	#10,R2	;SET COUNT TO LOAD 8 ADDRESSES
5109	034566	012701	172300		MOV	#KIPDR0,R1	;PUT ADDRESS OF FIRST PDR IN R1
5110	034572	010021		1\$:	MOV	R0,(R1)+	;LOAD R0 INTO PDR ADDRESSED BY R1
5111	034574	077202			SQB	R2,1\$	;BRANCH BACK TO 1\$ IF R2 IS NOT ZERO
5112	034576	012702	000010		MOV	#10,R2	;SET COUNT TO LOAD 8 ADDRESSES
5113	034602	012701	177600		MOV	#UIPDR0,R1	;PUT ADDRESS OF FIRST PDR IN R1
5114	034606	010021		2\$:	MOV	R0,(R1)+	;LOAD R0 INTO PDR ADDRESSED BY R1
5115	034610	077202			SQB	R2,2\$	;BRANCH BACK TO 2\$ IF R2 IS NOT ZERO
5116	034612	012700	077406	21\$:	MOV	#77406,R0	;MUST RE-INIT THESE PDRS IF ERROR
5117	034616	010037	172300		MOV	R0,KIPDR0	;LOAD KERNEL PDR0
5118	034622	010037	172302		MOV	R0,KIPDR1	;LOAD KERNEL PDR1
5119	034626	010037	172304		MOV	R0,KIPDR2	;LOAD KERNEL PDR2
5120	034632	010037	172316		MOV	R0,KIPDR7	;LOAD KERNEL PDR7
5121	034636	010037	172300		MOV	R0,KIPDR0	;LOAD PDR0 OF PRESENT SPACE
5122	034642	010037	172316		MOV	R0,KIPDR7	;LOAD PDR7 OF PRESENT SPACE
5123	034646	020527	170200		CMP	R5,#170200	;IS PDR7 BEING TESTED?
5124	034652	001032			BNE	22\$	;BRANCH IF IT ISN'T
5125	034654	012737	007600 172354		MOV	#7600,KIPAR6	;MAP PAR6 TO THE I/O PAGE
5126	034662	012737	000600 172356		MOV	#600,KIPAR7	;RE-MAP PAR7 TO 12K
5127	034670	011515			MOV	(R5),(R5)	;NOW WRITE USING PAR7
5128	034672	042737	000001 157572		BIC	#BIT0,#157572	;TURN OFF MEMORY MANAGEMENT USING
5129							;PAR6-ADDRESS OF MMIO
5130	034700	022737	077506 172316		CMP	#77506,KIPDR7	;CHECK PDR 7 BEFORE RESTORING PDR/PAR'S
5131	034706	001005			BNE	24\$	;DID W-BIT GET SET?
5132							;BRANCH IF NO
5133	034710	005237	001176		INC	\$TMP0	;SET CORRECT PDR SET INDICATOR
5134	034714	012737	077406 172300		MOV	#77406,KIPDR0	;RESTORE PDR0 AFTER WRITING \$TMP0
5135	034722	012737	077406 172314	24\$:	MOV	#77406,KIPDR6	;RESET PDR6
5136	034730	012737	000600 172354		MOV	#600,KIPAR6	;RE-MAP PAR6 TO 12K
5137	034736	000401			BR	23\$	;AND NOW COMPARE TO OTHER PDR'S
5138	034740	011515		22\$:	MOV	(R5),(R5)	;WRITE INTO PAGE UNDER TEST
5139	034742			23\$:			
5140	034742	012702	000010		MOV	#10,R2	;SET COUNTER TO READ NEXT 10 REGISTERS
5141	034746	012701	172300		MOV	#KIPDR0,R1	;LOAD ADDRESS OF BEGINNING PDR
5142	034752	011100		7\$:	MOV	(R1),R0	;READ PDR INTO R0
5143	034754	022700	077506		CMP	#77506,R0	;SEE IF THIS WAS THE PDR WITH ITS W BIT ON
5144	034760	001030			BNE	9\$	;BRANCH IF THIS IS NOT THE ONE
5145	034762	020103			CMP	R1,R3	;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
5146	034764	001424			BEQ	8\$	;BRANCH IF ADDRESS IS CORRECT
5147	034766	104062			ERROR	62	;W BIT GOT SET IN WRONG PDR
5148	034770	012700	077406		MOV	#77406,R0	;RE-SET PAGES MODIFIED BY ERROR
5149	034774	010037	172300		MOV	R0,KIPDR0	;RELOAD KERNEL PDR0
5150	035000	010037	172302		MOV	R0,KIPDR1	;RELOAD KERNEL PDR1
5151	035004	010037	172304		MOV	R0,KIPDR2	;RELOAD KERNEL PDR2
5152	035010	010037	172316		MOV	R0,KIPDR7	;RELOAD KERNEL PDR7
5153	035014	010037	172300		MOV	R0,KIPDR0	;RELOAD PAGE 0 OF PRESENT SPACE
5154	035020	010037	172316		MOV	R0,KIPDR7	;RE-LOAD I/O PAGE PDR IF ERROR
5155	035024	020527	170200		CMP	R5,#170200	;SEE IF TESTING PDR7
5156	035030	001404			BEQ	9\$	;DON'T WRITE IF TESTING PDR7
5157	035032	011515			MOV	(R5),(R5)	;TRY WRITE AGAIN, IN CASE YOU
5158							;WERE NOT TESTING PAGE SEVEN
5159	035034	000402			BR	9\$	;GO UPDATE R1 FOR NEXT READ
5160	035036	005237	001176	8\$:	INC	\$TMP0	;SET FLAG SINCE ADDRESSES MATCHED

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG.
DOKTAA.P11 07-FEB-77 10:30 T45

MACY11 27(1006) 07-FEB-77 10:37 PAGE 99
DUAL MAPPING KERNEL MODE SPACE

5161	035042	062701	000002	95:	ADD	#2,R1	:POINT TO NEXT PDR TO BE READ
5162	035046	077237			SUB	R2,75	:BRANCH TO 75 IF ALL PDR'S NOT READ
5163	035050	012702	000010		MOV	#10,R2	:SET COUNTER TO READ NEXT 10 REGISTERS
5164	035054	012701	177600		MOV	#UIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR
5165	035060	011100		135:	MOV	(R1),R0	:READ PDR INTO R0
5166	035062	022700	077506		CMP	#77506,R0	:SEE IF THIS WAS THE PDR WITH ITS W BIT ON
5167	035066	001030			BNE	155	:BRANCH IF THIS IS NOT THE ONE
5168	035070	020103			CMP	R1,R3	:SEE IF THE ADDRESS MATCHES PDR UNDER TEST
5169	035072	001424			BEQ	145	:BRANCH IF ADDRESS IS CORRECT
5170	035074	104062			ERROR	62	:W BIT GOT SET IN WRONG PDR
5171	035076	012700	077406		MOV	#77406,R0	:RE-SET PAGES MODIFIED BY ERROR
5172	035102	010037	172300		MOV	R0,KIPDR0	:RELOAD KERNEL PDR0
5173	035106	010037	172302		MOV	R0,KIPDR1	:RELOAD KERNEL PDR1
5174	035112	010037	172304		MOV	R0,KIPDR2	:RELOAD KERNEL PDR2
5175	035116	010037	172316		MOV	R0,KIPDR3	:RELOAD KERNEL PDR3
5176	035122	010037	172300		MOV	R0,KIPDR0	:RELOAD PAGE 0 OF PRESENT SPACE
5177	035126	010037	172316		MOV	R0,KIPDR3	:RE-LOAD I/O PAGE PDR IF ERROR
5178	035132	020527	170200		CMP	R5,#170200	:SEE IF TESTING PDR?
5179	035136	001404			BEQ	155	:DON'T WRITE IF TESTING PDR?
5180	035140	011515			MOV	(R5),(R5)	:TRY WRITE AGAIN, IN CASE YOU
5181							:WERE NOT TESTING PAGE SEVEN
5182	035142	000402			BR	155	:GO UPDATE R1 FOR NEXT READ
5183	035144	005237	001176	145:	INC	\$TMP0	:SET FLAG SINCE ADDRESSES MATCHED
5184	035150	062701	000002	155:	ADD	#2,R1	:POINT TO NEXT PDR TO BE READ
5185	035154	077237			SUB	R2,135	:BRANCH TO 135 IF ALL PDR'S NOT READ
5186	035156	005737	001176		TST	\$TMP0	:SEE IF THERE WAS A CORRECT PDR
5187	035162	001002			BNE	165	:BRANCH IF THERE WAS
5188	035164	011300			MOV	(R3),R0	:SAVE CONTENTS OF PDR UNDER TEST
5189	035166	104063			ERROR	63	:NO PDR ADDRESSES MATCHED
5190	035170	062703	000002	165:	ADD	#2,R3	:POINT TO NEXT PDR UNDER TEST
5191	035174	062705	020000		ADD	#20000,R5	:CHANGE PAGE NUMBER IN VIRT. ADDR.
5192	035200	005304			DEC	R4	:DECREMENT COUNTER
5193	035202	001402			BEQ	175	:BRANCH IF COUNTER IS ZERO
5194	035204	000137	034552		JMP	195	:JUMP TO LOAD PDR'S AGAIN
5195	035210	012737	007600 172356	175:	MOV	#7600,KIPAR7	:REMAP PAR7 TO THE I/O PAGE
5196	035216	052737	000001 177572		BIS	#BIT0,MMR0	:REENABLE MEMORY MANAGEMENT AFTER
5197							:TESTING PDR?
5198	035224	012737	000340 177776		MOV	#340,PSW	:RETURN TO KERNEL MODE, PRIORITY 7
5199	035232	012737	034522 001110		MOV	#205,\$LPERR	:SET LOOP POINTER TO START OF TEST

*TEST 46 DUAL MAPPING USER MODE SPACE

THIS TEST STARTS BY LOADING ALL THE P.D.R.'S WITH 77406
(4K PAGE, READ/WRITE). THEN THE VIRTUAL ADDRESS IS SET TO
010200 AND THAT WORD IS WRITTEN INTO ITSELF. THIS WRITE WILL
SET THE W BIT IN THE USER SPACE P.D.R. UNDER TEST. NOW ALL OF THE
P.D.R.'S ARE COMPARED WITH 77706 AND ANY THAT MATCH, EXCEPT THE
ONE THAT IS UNDER TEST, ARE REPORTED AS DUAL MAPPING ERRORS.
USER PAR7 IS MAPPED TO 12K AND USER PAR6 TO I/O PAGE DURING THE WRITE SINCE
WRITING TO REGISTERS INTERNAL TO THE CPU ON THE 11/6X WILL NOT SET THE W BIT
AND ANY OTHER EXISTING ADDRESS CAN'T BE GUARANTEED FOR THE I/O PAGE

J08

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 100
 DOKTAA.P11 07-FEB-77 10:30 T46 DUAL MAPPING USER MODE SPACE

```

5217          ::*****
5218          †ST46:
5219          SCOPE
5220          MOV      #TST47,NXTTST      ;SAVE STARTING ADDRESS OF NEXT
5221          ;TEST FOR ESCAPE ON PARITY ERRORS
5222          20$:    MOV      #21$,SLPERR      ;SET LOOP ON ERROR POINTER TO 21$
5223          MOV      #140000,PSW      ;GO TO USER MODE
5224          MOV      #UIPDRO,R3      ;LOAD FIRST ADDRESS OF USER PDR'S
5225          ;THAT WILL BE TESTED IN THIS TEST
5226          MOV      #10,R4      ;TEST THE NEXT EIGHT PDR'S
5227          MOV      #10200,R5      ;LOAD STARTING VIRTUAL ADDRESS INTO R5
5228          19$:    MOV      #77406,R0      ;ALL PAGES WILL BE ACF=6
5229          CLR      $TMP0      ;CLEAR CORRECT PDR SET INDICATOR
5230          MOV      #10,R2      ;SET COUNT TO LOAD 8 ADDRESSES
5231          MOV      #KIPDR0,R1      ;PUT ADDRESS OF FIRST PDR IN R1
5232          1$:    MOV      R0,(R1)†      ;LOAD R0 INTO PDR ADDRESSED BY R1
5233          SOB      R2,1$      ;BRANCH BACK TO 1$ IF R2 IS NOT ZERO
5234          MOV      #10,R2      ;SET COUNT TO LOAD 8 ADDRESSES
5235          MOV      #UIPDRO,R1      ;PUT ADDRESS OF FIRST PDR IN R1
5236          2$:    MOV      R0,(R1)†      ;LOAD R0 INTO PDR ADDRESSED BY R1
5237          SOB      R2,2$      ;BRANCH BACK TO 2$ IF R2 IS NOT ZERO
5238          21$:   MOV      #77406,R0      ;MUST RE-INIT THESE PDRS IF ERROR
5239          MOV      R0,KIPDR0      ;LOAD KERNEL PDR0
5240          MOV      R0,KIPDR1      ;LOAD KERNEL PDR1
5241          MOV      R0,KIPDR2      ;LOAD KERNEL PDR2
5242          MOV      R0,KIPDR7      ;LOAD KERNEL PDR7
5243          MOV      R0,UIPDRO      ;LOAD PDR0 OF PRESENT SPACE
5244          MOV      R0,UIPDR7      ;LOAD PDR7 OF PRESENT SPACE
5245          CMP      R5,#170200      ;IS PDR7 BEING TESTED?
5246          BNE      22$      ;BRANCH IF IT ISN'T
5247          MOV      #7600,UIPAR6      ;MAP PAR6 TO THE I/O PAGE
5248          MOV      #600,UIPAR7      ;RE-MAP PAR7 TO 12K
5249          MOV      (R5),(R5)      ;NOW WRITE USING PAR7
5250          BIC      #BIT0,#157572      ;TURN OFF MEMORY MANAGEMENT USING
5251          ;PAR6-ADDRESS OF MMRO
5252          CMP      #77506,UIPDR7      ;CHECK PDR 7 BEFORE RESTORING PDR/PAR'S
5253          BNE      24$      ;DID W-BIT GET SET?
5254          ;BRANCH IF NO
5255          INC      $TMP0      ;SET CORRECT PDR SET INDICATOR
5256          MOV      #77406,UIPDRO      ;RESTORE PDR0 AFTER WRITING $TMP0
5257          24$:    MOV      #77406,UIPDR6      ;RESET PDR6
5258          MOV      #600,UIPAR6      ;RE-MAP PAR6 TO 12K
5259          BR       23$      ;AND NOW COMPARE TO OTHER PDR'S
5260          22$:    MOV      (R5),(R5)      ;WRITE INTO PAGE UNDER TEST
5261          23$:
5262          MOV      #10,R2      ;SET COUNTER TO READ NEXT 10 REGISTERS
5263          MOV      #KIPDR0,R1      ;LOAD ADDRESS OF BEGINNING PDR
5264          7$:    MOV      (R1),R0      ;READ PDR INTO R0
5265          CMP      #77506,R0      ;SEE IF THIS WAS THE PDR WITH ITS W BIT ON
5266          BNE      9$      ;BRANCH IF THIS IS NOT THE ONE
5267          CMP      R1,R3      ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
5268          BEQ      8$      ;BRANCH IF ADDRESS IS CORRECT
5269          ERROR    62      ;W BIT GOT SET IN WRONG PDR
5270          MOV      #77406,R0      ;RE-SET PAGES MODIFIED BY ERROR
5271          MOV      R0,KIPDR0      ;RELOAD KERNEL PDR0
5272          MOV      R0,KIPDR1      ;RELOAD KERNEL PDR1

```

K08

MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG. T46
DQKTA.P11 07-FEB-77 10:30

MACY11 27(1006) 07-FEB-77 10:37 PAGE 101
DUAL MAPPING USER MODE SPACE

5273	035532	010037	172304		MOV	R0, KIPDR2	; RELOAD KERNEL PDR2
5274	035536	010037	172316		MOV	R0, KIPDR7	; RELOAD KERNEL PDR7
5275	035542	010037	177600		MOV	R0, UIPDR0	; RELOAD PAGE 0 OF PRESENT SPACE
5276	035546	010037	177616		MOV	R0, UIPDR7	; RE-LOAD I/O PAGE PDR IF ERROR
5277	035552	020527	170200		CMP	R5, #170200	; SEE IF TESTING PDR7
5278	035556	001404			BEQ	9\$	; DON'T WRITE IF TESTING PDR7
5279	035560	011515			MOV	(R5), (R5)	; TRY WRITE AGAIN, IN CASE YOU
5280							; WERE NOT TESTING PAGE SEVEN
5281	035562	000402			BR	9\$	; GO UPDATE R1 FOR NEXT READ
5282	035564	005237	001176	8\$:	INC	\$TMP0	; SET FLAG SINCE ADDRESSES MATCHED
5283	035570	062701	000002	9\$:	ADD	#2, R1	; POINT TO NEXT PDR TO BE READ
5284	035574	077237			SQB	R2, 7\$	; BRANCH TO 7\$ IF ALL PDR'S NOT READ
5285	035576	012702	000010		MOV	#10, R2	; SET COUNTER TO READ NEXT 10 REGISTERS
5286	035602	012701	177600		MOV	#UIPDR0, R1	; LOAD ADDRESS OF BEGINNING PDR
5287	035606	011100		13\$:	MOV	(R1), R0	; READ PDR INTO R0
5288	035610	022700	077506		CMP	#77506, R0	; SEE IF THIS WAS THE PDR WITH ITS W BIT ON
5289	035614	001030			BNE	15\$	; BRANCH IF THIS IS NOT THE ONE
5290	035616	020103			CMP	R1, R3	; SEE IF THE ADDRESS MATCHES PDR UNDER TEST
5291	035620	001424			BEQ	14\$	; BRANCH IF ADDRESS IS CORRECT
5292	035622	104062			ERROR	62	; W BIT GOT SET IN WRONG PDR
5293	035624	012700	077406		MOV	#77406, R0	; RE-SET PAGES MODIFIED BY ERROR
5294	035630	010037	172300		MOV	R0, KIPDR0	; RELOAD KERNEL PDR0
5295	035634	010037	172302		MOV	R0, KIPDR1	; RELOAD KERNEL PDR1
5296	035640	010037	172304		MOV	R0, KIPDR2	; RELOAD KERNEL PDR2
5297	035644	010037	172316		MOV	R0, KIPDR7	; RELOAD KERNEL PDR7
5298	035650	010037	177600		MOV	R0, UIPDR0	; RELOAD PAGE 0 OF PRESENT SPACE
5299	035654	010037	177616		MOV	R0, UIPDR7	; RE-LOAD I/O PAGE PDR IF ERROR
5300	035660	020527	170200		CMP	R5, #170200	; SEE IF TESTING PDR7
5301	035664	001404			BEQ	15\$	; DON'T WRITE IF TESTING PDR7
5302	035666	011515			MOV	(R5), (R5)	; TRY WRITE AGAIN, IN CASE YOU
5303							; WERE NOT TESTING PAGE SEVEN
5304	035670	000402			BR	15\$	; GO UPDATE R1 FOR NEXT READ
5305	035672	005237	001176	14\$:	INC	\$TMP0	; SET FLAG SINCE ADDRESSES MATCHED
5306	035676	062701	000002	15\$:	ADD	#2, R1	; POINT TO NEXT PDR TO BE READ
5307	035702	077237			SQB	R2, 13\$	; BRANCH TO 13\$ IF ALL PDR'S NOT READ
5308	035704	005737	001176		TST	\$TMP0	; SEE IF THERE WAS A CORRECT PDR
5309	035710	001002			BNE	16\$	; BRANCH IF THERE WAS
5310	035712	011300			MOV	(R3), R0	; SAVE CONTENTS OF PDR UNDER TEST
5311	035714	104063			ERROR	63	; NO PDR ADDRESSES MATCHED
5312	035716	062703	000002	16\$:	ADD	#2, R3	; POINT TO NEXT PDR UNDER TEST
5313	035722	062705	020000		ADD	#20000, R5	; CHANGE PAGE NUMBER IN VIRT. ADDR.
5314	035726	005304			DEC	R4	; DECREMENT COUNTER
5315	035730	001402			BEQ	17\$	; BRANCH IF COUNTER IS ZERO
5316	035732	000137	035300		JMP	19\$	; JUMP TO LOAD PDR'S AGAIN
5317	035736	012737	007600	17\$:	MOV	#7600, UIPAR7	; REMAP PAR7 TO THE I/O PAGE
5318	035744	052737	000001		BIS	#BIT0, MMRO	; REENABLE MEMORY MANAGEMENT AFTER
5319							; TESTING PDR7
5320	035752	012737	000340		MOV	#340, PSW	; RETURN TO KERNEL MODE, PRIORITY 7
5321	035760	012737	035250	001110	MOV	#20\$, \$LPERR	; SET LOOP POINTER TO START OF TEST
5322							
5323							
5324							
5325							
5326							
5327							
5328							

SBTTL ***** ENTRY POINT 7 --- STARTING ADDRESS 230 *****
SBTTL ***** MOVE FROM AND MOVE TO PREVIOUS MODE INSTRUCTION TEST *****
;*

Address	Instruction	Comments
5329	THIS GROUP OF TESTS WILL TEST ALL THE LOGIC ASSOCIATED WITH	
5330	THE "MOVE FROM PREVIOUS" AND MOVE TO PREVIOUS INSTRUCTIONS.	
5331		
5332		
5333		
5334	*****	
5335	TEST 47 MOVE FROM PREVIOUS (USER) I-SPACE	
5336		
5337	THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT THE	
5338	PREVIOUS MODE IS CLOCKED CORRECTLY	
5339	THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,	
5340		
5341		
5342	IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT	
5343	WILL OCCUR AND TRAP TO 10\$, WHERE THE ERRORS ARE REPORTED.	
5344		
5345	*****	
5346	TST47:	
5347	SCOPE	
5348	MOV #TST50,NXTTST	SAVE STARTING ADDRESS OF NEXT
5349		TEST FOR ESCAPE ON PARITY ERRORS
5350	TBITR	RESTORE T-BIT IF IT WAS ON BEFORE
5351		THE DUAL MAPPING TESTS
5352	ENT-17: MOV #20\$,SLPADR	SET LOOP ADDRESS POINTER TO 20\$
5353	MOV #20\$,SLPERR	SET LOOP ON ERROR POINTER TO 20\$
5354	MOV #47,\$TSTNM	LOAD TEST NUMBER INTO MEMORY
5355	MOV \$TSTNM,\$DISPLAY	DISPLAY TEST NUMBER FOR THIS TEST
5356	MOV #77406,KIPDR0	MAKE KERNEL I PAGE 0 200 BLOCKS, R/W
5357	MOV #77406,KIPDR1	MAKE KERNEL I PAGE 1 200 BLOCKS, R/W
5358	MOV #77406,KIPDR2	MAKE KERNEL I PAGE 2 200 BLOCKS, R/W
5359	MOV #77406,KIPDR3	MAKE KERNEL I PAGE 3 200 BLOCKS, R/W
5360	MOV #77406,KIPDR7	MAKE KERNEL I PAGE 7 200 BLOCKS, R/W
5361	MOV #000,KIPAR0	MAP KERNEL I PAGE 0 TO 0 - 4K
5362	MOV #200,KIPAR1	MAP KERNEL I PAGE 1 TO 4K - 8K
5363	MOV #400,KIPAR2	MAP KERNEL I PAGE 2 TO 8K - 12K
5364	MOV #600,KIPAR3	MAP KERNEL I PAGE 3 TO 12K - 16K
5365	MOV #7600,KIPAR7	MAP KERNEL I PAGE 7 TO THE I/O PAGE
5366	MOV #810,MAR0	ENABLE 18-BIT RELOCATION IF NOT ON
5367	MOV #77406,R0	MAKE ALL KERNEL I-SPACE PAGES RESIDENT
5368		READ/WRITE, LENGTH 200 BLOCKS
5369	MOV #10,R2	SET COUNT TO LOAD 8 ADDRESSES
5370	MOV #KIPDR0,R1	PUT ADDRESS OF FIRST PDR IN R1
5371	MOV R0,(R1)+	LOAD R0 INTO PDR ADDRESSED BY R1
5372	SOB R2,19\$	BRANCH BACK TO 19\$ IF R2 IS NOT ZERO
5373	MOV #10,R2	SET COUNT TO LOAD 8 ADDRESSES
5374	MOV #UIPDR0,R1	PUT ADDRESS OF FIRST PDR IN R1
5375	MOV R0,(R1)+	LOAD R0 INTO PDR ADDRESSED BY R1
5376	SOB R2,30\$	BRANCH BACK TO 30\$ IF R2 IS NOT ZERO
5377	MOV #000,UIPAR0	MAP USER I PAGE 0 TO 0-4K
5378	MOV #200,UIPAR1	MAP USER I PAGE 1 TO 4-8K
5379	MOV #400,UIPAR2	MAP USER I PAGE 2 TO 8-12K
5380	MOV #600,UIPAR3	MAP USER I PAGE 3 TO 12-16K
5381	MOV #7600,UIPAR7	MAP USER I PAGE 7 TO THE I/O PAGE
5382	MOV #20\$,SLPERR	SET LOOP ON ERROR TO 20\$
5383		
5384	MOV #77406,KIPDR4	KERNEL I-SPACE PAGE 4 READ/WRITE

M08

MAINDEX-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG.
DOKTAA.P11 07-FEB-77 10:30 T47

MACY11 27(1006) 07-FEB-77 10:37 PAGE 103
MOVE FROM PREVIOUS (USEF) I-SPACE

5385	036240	012737	000600	172350		MOV	#600,KIPAR4	MAP KERNEL I PAGE 4 TO 12K
5386	036246	012737	000600	177650		MOV	#600,UIPAR4	MAP USER I PAGE 4 TO 12K
5387	036254	012700	036514			MOV	#36514,R0	LOAD DATA PATTERN INTO R0
5388	036260	010037	100000			MOV	R0,#100000	LOAD DATA PATTERN INTO PHY 60000
5389	036264	012737	036662	000250		MOV	#105,MVVEC	SET M.M. VECTOR TO 105
5390	036272	105037	172310			CLRB	KIPAR4	MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
5391								DSTM=0 MFPI
5392								THE FOLLOWING WILL TEST
5393	036276	012737	036304	001110		MOV	#115,\$LPERR	SET LOOP ON ERROR POINTER TO 115
5394	036304	012737	030340	177776	115:	MOV	#030340,PSW	MAKE PREVIOUS MODE USER
5395	036312	006506			15:	MFPI	USP	PUT USER STACK POINTER ON KERNEL
5396								STACK
5397	036314	022706	001100			CMP	#KERSTK,KSP	WAS SOMETHING PUSHED ON STACK AT 15
5398	036320	001407				BEQ	35	BRANCH IF NOTHING WAS PUSHED
5399	036322	012601				MOV	(KSP)+,R1	POP KERNEL STACK INTO R1
5400	036324	012702	000700			MOV	#USESTK,R2	EXPECTING TO GET 700 AS USP
5401	036330	020201				CMP	R2,R1	DID YOU GET THE RIGHT POINTER?
5402	036332	001403				BEQ	25	BRANCH IF YOU DID
5403	036334	104064				ERROR	64	WRONG THING WAS PUSHED ON STACK
5404	036336	000401				BR	25	BRANCH TO NEXT TRY
5405	036340	104065			35:	ERROR	65	NOTHING PUSHED ON STACK
5406	036342				25:			THE FOLLOWING WILL TEST DSTM=1 MFPI.
5407	036342	012737	036350	001110		MOV	#125,\$LPERR	SET LOOP ON ERROR POINTER TO 125
5408	036350	012737	030340	177776	125:	MOV	#030340,PSW	MAKE PREVIOUS MODE USER
5409	036356	012702	100000			MOV	#100000,R2	LOAD VIRTUAL ADDRESS INTO R2
5410	036362	006512				MFPI	(R2)	READ FROM PHYSICAL 60000
5411	036364	012501				MOV	(KSP)+,R1	POP KERNEL STACK INTO R1
5412	036366	020001				CMP	R0,R1	WAS DATA FETCHED SAME AS STORED
5413	036370	001401				BEQ	45	BRANCH IF CORRECT DATA WAS FETCHED
5414	036372	104066				ERROR	66	WRONG DATA WAS FETCHED
5415	036374				45:			THE FOLLOWING WILL TEST DSTM=2 MFPI.
5416	036374	012737	036402	001110		MOV	#145,\$LPERR	SET LOOP ON ERROR POINTER TO 145
5417	036402	012737	030340	177776	145:	MOV	#030340,PSW	MAKE PREVIOUS MODE USER
5418	036410	012702	100000			MOV	#100000,R2	LOAD VIRTUAL ADDRESS INTO R2
5419	036414	006522				MFPI	(R2)+	READ FROM PHYSICAL 60000
5420	036416	012601				MOV	(KSP)+,R1	POP KERNEL STACK INTO R1
5421	036420	020001				CMP	R0,R1	WAS DATA FETCHED SAME AS STORED
5422	036422	001401				BEQ	55	BRANCH IF CORRECT DATA WAS FETCHED
5423	036424	104066				ERROR	66	WRONG DATA WAS FETCHED
5424	036426				55:			THE FOLLOWING WILL TEST DSTM=3 MFPI.
5425	036426	012737	036434	001110		MOV	#155,\$LPERR	SET LOOP ON ERROR POINTER TO 155
5426	036434	012737	030340	177776	155:	MOV	#030340,PSW	MAKE PREVIOUS MODE USER
5427	036442	006537	100000			MFPI	#100000	READ FROM PHYSICAL 60000
5428	036446	012601				MOV	(KSP)+,R1	POP KERNEL STACK INTO R1
5429	036450	020001				CMP	R0,R1	WAS DATA FETCHED SAME AS STORED
5430	036452	001401				BEQ	65	BRANCH IF CORRECT DATA WAS FETCHED
5431	036454	104066				ERROR	66	WRONG DATA WAS FETCHED
5432	036456				65:			THE FOLLOWING WILL TEST DSTM=4 MFPI.
5433	036456	012737	036464	001110		MOV	#165,\$LPERR	SET LOOP ON ERROR POINTER TO 165
5434	036464	012737	030340	177776	165:	MOV	#030340,PSW	MAKE PREVIOUS MODE USER
5435	036472	012702	100002			MOV	#100002,R2	LOAD VIRTUAL ADDRESS INTO R2
5436	036476	006542				MFPI	-(R2)	READ FROM PHYSICAL 60000
5437	036500	012601				MOV	(KSP)+,R1	POP KERNEL STACK INTO R1
5438	036502	020001				CMP	R0,R1	WAS DATA FETCHED SAME AS STORED
5439	036504	001401				BEQ	75	BRANCH IF CORRECT DATA WAS FETCHED
5440	036506	104066				ERROR	66	WRONG DATA WAS FETCHED

N08

NRINDEC-11-DOKTA-A POP 11/6X MEM. MGMT. DIAG.
DOKTAA.P11 07-FEB-77 10:30 T47

MACY11 27(1006) 07-FEB-77 10:37 PAGE 104
MOVE FROM PREVIOUS (USER) I-SPACE

```

5441 036510 7S: ;THE FOLLOWING WILL TEST DSTN=5 MFPI.
5442
5443
5444 036510 012737 036516 001110 MOV #17S,SLPERR ;SET LOOP ON ERROR POINTER TO 17S
5445 036516 012737 030340 177776 17S: MOV #030340,PSW ;MAKE PREVIOUS MODE USER
5446 036524 012737 100000 001202 MOV #100000,STMP2 ;LOAD TEST LOC. VIRT. ADDR INTO LOC. STMP2
5447 036532 012702 001204 MOV #(<STMP2+2>),R2 ;LOAD ADDR. OF STMP2+2 INTO R2
5448 036536 006552 MFPI 2-(R2) ;READ FROM PHYSICAL 60000
5449 036540 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
5450 036542 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
5451 036544 001401 BEQ 8S ;BRANCH IF CORRECT DATA WAS FETCHED
5452 036546 104066 ERROR 66 ;WRONG DATA WAS FETCHED
5453 036550 8S: ;THE FOLLOWING WILL TEST DSTN=6 MFPI.
5454
5455 036550 012737 036556 001110 MOV #18S,SLPERR ;SET LOOP ON ERROR POINTER TO 18S
5456 036556 012737 030340 177776 18S: MOV #030340,PSW ;MAKE PREVIOUS MODE USER
5457 036564 005002 CLR R2 ;MAKE REGISTER 2 A ZERO
5458 036566 006562 100000 MFPI 100000(R2) ;READ FROM PHYSICAL 60000
5459 036572 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
5460 036574 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
5461 036576 001401 BEQ 9S ;BRANCH IF CORRECT DATA WAS FETCHED
5462 036600 104066 ERROR 66 ;WRONG DATA WAS FETCHED
5463 036602 9S: ;THE FOLLOWING WILL TEST DSTN=7 MFPI.
5464
5465 036602 012737 036610 001110 MOV #22S,SLPERR ;SET LOOP ON ERROR POINTER TO 22S
5466 036610 012737 030340 177776 22S: MOV #030340,PSW ;MAKE PREVIOUS MODE USER
5467 036616 012737 100000 001202 MOV #100000,STMP2 ;LOAD TEST LOC. V.A. INTO STMP2
5468 036624 012702 001202 MOV #STMP2,R2 ;LOAD ADDRESS OF STMP2 INTO R2
5469 036630 006572 000000 MFPI 20(R2) ;USE STMP2 TO FETCH VIRTUAL
5470 ;ADDRESS OF 60000
5471 036634 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
5472 036636 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
5473 036640 001401 BEQ 25S ;BRANCH IF CORRECT DATA WAS FETCHED
5474 036642 104066 ERROR 66 ;WRONG DATA WAS FETCHED
5475 036644 012737 015432 000250 25S: MOV #MMTRAP,MMVEC ;SET M.M. VECTOR TO NORMAL ROUTINE
5476 036652 012737 036232 001110 MOV #20S,SLPERR ;SET LOOP POINTER TO START OF TEST
5477 036660 000423 BR 15150 ;BRANCH TO NEXT TEST
5478
5479
5480 036662 012637 001436 10S: MOV (KSP)+,OLDPC ;SAVE PC & PS OF TRAP
5481 036666 012637 001440 MOV (KSP)+,OLDPS
5482 036672 013737 177572 001404 MOV MMRO,PMRO ;SAVE MMRO FOR ERROR TYPEOUT
5483 036700 013737 177576 001406 MOV MMW2,PMW2 ;SAVE MMW2 FOR ERROR TYPEOUT
5484 036706 042737 160000 177572 BIC #160000,MMRO ;CLEAR ERROR BITS IN MMRO
5485 036714 104067 ERROR 67 ;TRIED TO READ NON-RESIDENT PAGE
5486 036716 013746 001440 MOV OLDPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
5487 036722 013746 001436 MOV OLDPC,-(KSP)
5488 036726 000002 PTI
5489
5490
5491 ;*****
5492 ;TEST 50 MOVE TO PREVIOUS (USER) I-SPACE
5493 ;
5494 ; THIS TEST USES THE 'MTPI' INSTRUCTION TO ENSURE THAT THE
5495 ; PREVIOUS MODE IS CLOKED CORRECTLY
5496 ; THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,

```

```

5497
5498
5499
5500
5501
5502
5503 036730
5504 036730 000004
5505 036732 012737 037616 001446
5506
5507 036740 012737 077406 172310 20$:
5508 036746 012737 077406 177610
5509 036754 012737 000600 172350
5510 036762 012737 000600 177650
5511 036770 012737 037550 000250
5512
5513
5514 036776 012737 030340 177776 1$:
5515 037004 012746 007777
5516 037010 006606
5517 037012 006506
5518 037014 012601
5519 037016 022701 007777
5520 037022 001401
5521 037024 104070
5522 037026 012737 030340 177776 2$:
5523 037034 012746 000700
5524 037040 006606
5525 037042
5526 037042 012737 037060 001110 3$:
5527 037050 012702 100000
5528 037054 012700 125252
5529 037060 010046
5530 037062 105037 172310 13$:
5531 037066 006612
5532 037070 112737 000006 172310
5533 037076 011201
5534 037100 020001
5535 037102 001401
5536 037104 104071
5537 037106
5538 037106 012737 037126 001110 4$:
5539 037114 012737 030340 177776
5540 037122 012700 052525
5541 037126 010046
5542 037130 105037 172310 14$:
5543 037134 006637 100000
5544 037140 112737 000006 172310
5545 037146 013701 100000
5546 037152 020001
5547 037154 001401
5548 037156 104071
5549 037160
5550 037160 012737 037200 001110 5$:
5551 037166 012737 030340 177776
5552 037174 012700 125252

```

IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
 WILL OCCUR AND TRAP TO 10\$, WHERE THE ERRORS ARE REPORTED.

 T50:

SCOPE
 MOV #TST51,NXTTST ;SAVE STARTING ADDRESS OF NEXT
 ;TEST FOR ESCAPE ON PARITY ERRORS
 MOV #77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE
 MOV #77406,UIPDR4 ;USER I-SPACE PAGE 4 READ/WRITE
 MOV #600,KIPAR4 ;MAP KERNEL I PAGE 4 TO 12K
 MOV #600,UIPAR4 ;MAP USER I PAGE 4 TO 12K
 MOV #10\$,MMVEC ;SET M.M. VECTOR TO 10\$
 ;THE FOLLOWING WILL TEST DSTM=0 MTPI

1\$: MOV #030340,PSW ;MAKE PREVIOUS MODE USER
 MOV #7777,-(KSP) ;PUSH DATA ON KERNEL STACK
 MTPI USP ;LOAD USER STACK POINTER
 MFPI USP ;READ USER STACK POINTER
 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
 CMP #7777,R1 ;WAS USER STACK POINTER CHANGED
 BEQ 2\$;BRANCH IF IT WAS
 ERROR 70 ;USER STACK POINTER NOT CHANGED

2\$: MOV #030340,PSW ;MAKE PREVIOUS MODE USER
 MOV #USESTK,-(KSP) ;GET READY TO RESTORE USER S. POINT
 MTPI USP ;RESTORE USER STACK POINTER

3\$: ;THIS WILL TEST DSTM = 1 MTPI.
 MOV #13\$,SLPERR ;SET LOOP ON ERROR POINTER TO 13\$
 MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
 MOV #125252,R0 ;LOAD TEST DATA INTO R0
 MOV R0,-(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 R0,R1 ;SEE IF DATA WAS STORED AT CORRECT PLACE
 BEQ 4\$;BRANCH IF STORE WAS CORRECT
 ERROR 71 ;INCORRECT STORE

4\$: ;THIS WILL TEST DSTM = 3 MTPI.
 MOV #14\$,SLPERR ;SET LOOP ON ERROR POINTER TO 14\$
 MOV #030340,PSW ;MAKE PREVIOUS MODE USER
 MOV #52525,R0 ;LOAD TEST DATA INTO R0
 MOV R0,-(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 R0,R1 ;SEE IF DATA WAS STORED CORRECTLY
 BEQ 5\$;BRANCH IF STORE WAS CORRECT
 ERROR 71 ;INCORRECT STORE

5\$: ;THIS WILL TEST DSTM = 4 MTPI.
 MOV #15\$,SLPERR ;SET LOOP ON ERROR POINTER TO 15\$
 MOV #030340,PSW ;MAKE PREVIOUS MODE USER
 MOV #125252,R0 ;LOAD TEST DATA INTO R0

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG.
DOKTAA.P11 07-FEB-77 10:30 T50

MACY11 27(1006) 07-FEB-77 10:37 PAGE 106
MOVE TO PREVIOUS (USER) I-SPACE

5553	037200	010046		15\$:	MOV	R0, -(KSP)	PUSH TEST DATA ON KERNEL STACK
5554	037202	012702	100002		MOV	#100002, R2	LOAD VIRTUAL ADDRESS INTO R2
5555	037206	105037	172310		CLRB	KIPDR4	MAKE KERNEL PAGE 4 NON-RESIDENT
5556	037212	006642			MTPI	-(R2)	LOAD TEST DATA INTO PHYSICAL 60000
5557	037214	112737	000006	172310	MOVB	#006, KIPDR4	MAKE KERNEL PAGE 4 RESIDENT
5558	037222	013701	100000		MOV	#100000, R1	READ FROM ADDRESS 60000
5559	037226	020001			CMP	R0, R1	SEE IF DATA WAS STORED CORRECTLY
5560	037230	001401			BEQ	6\$	BRANCH IF STORE WAS CORRECT
5561	037232	104071			ERROR	71	INCORRECT STORE
5562	037234			6\$:	; THIS WILL TEST DSTN = 6 MTPI. BELOW ARE THE		
5563							
5564	037234	012737	037256	001110	MOV	#16\$, \$LPERR	SET LOOP ON ERROR POINTER TO 16\$
5565	037242	012737	030340	177776	MOV	#030340, PSW	MAKE PREVIOUS MODE USER
5566	037250	012700	052525		MOV	#52525, R0	LOAD TEST DATA INTO R0
5567	037254	005002			CLR	R2	MAKE REGISTER 2 ZERO
5568	037256	010046		16\$:	MOV	R0, -(KSP)	PUSH TEST DATA ON KERNEL STACK
5569	037260	105037	172310		CLRB	KIPDR4	MAKE KERNEL PAGE 4 NON-RESIDENT
5570	037264	006662	100000		MTPI	100000(R2)	LOAD TEST DATA INTO PHYSICAL 60000
5571	037270	112737	000006	172310	MOVB	#006, KIPDR4	MAKE KERNEL PAGE 4 RESIDENT
5572	037276	013701	100000		MOV	#100000, R1	READ FROM ADDRESS 60000
5573	037302	020001			CMP	R0, R1	SEE IF DATA WAS STORED CORRECTLY
5574	037304	001401			BEQ	7\$	BRANCH IF STORE WAS CORRECT
5575	037306	104071			ERROR	71	INCORRECT STORE
5576	037310			7\$:	; THE FOLLOWING WILL TEST DSTN=2 MTPI.		
5577							
5578	037310	012737	037334	001110	MOV	#17\$, \$LPERR	SET LOOP ON ERROR POINTER TO 17\$
5579	037316	012737	030340	177776	MOV	#030340, PSW	MAKE PREVIOUS MODE USER
5580	037324	012700	125252		MOV	#125252, R0	LOAD TEST DATA INTO R0
5581	037330	012702	100000		MOV	#100000, R2	LOAD VIRTUAL ADDRESS INTO R2
5582	037334	010046		17\$:	MOV	R0, -(KSP)	PUSH TEST DATA ON KERNEL STACK
5583	037336	105037	172310		CLRB	KIPDR4	MAKE KERNEL PAGE 4 NON-RESIDENT
5584	037342	006612			MTPI	(R2)	LOAD TEST DATA INTO PHYSICAL 60000
5585	037344	112737	000006	172310	MOVB	#006, KIPDR4	MAKE KERNEL PAGE 4 RESIDENT
5586	037352	013701	100000		MOV	#100000, R1	READ FROM ADDRESS 60000
5587	037356	020001			CMP	R0, R1	SEE IF DATA WAS STORED CORRECTLY
5588	037360	001401			BEQ	8\$	BRANCH IF STORE WAS CORRECT
5589	037362	104071			ERROR	71	INCORRECT STORE
5590	037364			8\$:	; THE FOLLOWING WILL TEST DSTN=5 MTPI.		
5591							
5592	037364	012737	037416	001110	MOV	#18\$, \$LPERR	SET LOOP ON ERROR POINTER TO 18\$
5593	037372	012737	030340	177776	MOV	#030340, PSW	MAKE PREVIOUS MODE USER
5594	037400	012700	052525		MOV	#52525, R0	LOAD TEST DATA INTO R0
5595	037404	012702	001204		MOV	#(STMP2+2), R2	LOAD ADDR. OF LOC. STMP2+2 INTO R2
5596	037410	012737	100000	001202	MOV	#100000, STMP2	LOAD VIRT. ADDR. OF TEST LOC. INTO STMP2
5597	037416	010046		18\$:	MOV	R0, -(KSP)	PUSH TEST DATA ON KERNEL STACK
5598	037420	105037	172310		CLRB	KIPDR4	MAKE KERNEL PAGE 4 NON-RESIDENT
5599	037424	006652			MTPI	2-(R2)	LOAD TEST DATA INTO PHYSICAL 60000
5600	037426	112737	000006	172310	MOVB	#006, KIPDR4	MAKE KERNEL PAGE 4 RESIDENT
5601	037434	013701	100000		MOV	#100000, R1	READ FROM ADDRESS 60000
5602	037440	020001			CMP	R0, R1	SEE IF DATA WAS STORED CORRECTLY
5603	037442	001401			BEQ	9\$	BRANCH IF STORE WAS CORRECT
5604	037444	104071			ERROR	71	INCORRECT STORE
5605	037446			9\$:	; THE FOLLOWING WILL TEST DSTN=7 MTPI.		
5606							
5607	037446	012737	037500	001110	MOV	#19\$, \$LPERR	SET LOOP ON ERROR POINTER TO 19\$
5608	037454	012737	030340	177776	MOV	#030340, PSW	MAKE PREVIOUS MODE USER

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 107
 DOKTAA.P11 07-FEB-77 10:30 T50 MOVE TO PREVIOUS (USER) I-SPACE

5609	037462	012700	125252		MOV	#125252,R0	;LOAD TEST DATA INTO R0
5610	037466	012737	100000	001202	MOV	#100000,\$TMP2	;LOAD VIRT. ADDR. OF TEST LOCATION
5611							;INTO LOCATION \$TMP2
5612	037474	012702	001202		MOV	\$TMP2,R2	;LOAD ADDRESS OF \$TMP2 INTO R2
5613	037500	010046			19\$: MOV	R0,-(KSP)	;PUSH TEST DATA ON KERNEL STACK
5614	037502	105037	172310		CLRB	KIPDR4	;MAKE KERNEL PAGE 4 NON-RESIDENT
5615	037506	006672	000000		MTPI	20(R2)	;LOAD TEST DATA INTO PHYSICAL 60000
5616	037512	112737	000006	172310	MOV8	#006,KIPDR4	;MAKE KERNEL PAGE 4 RESIDENT
5617	037520	013701	100000		MOV	2#100000,R1	;READ FROM ADDRESS 60000
5618	037524	020001			CMP	R0,R1	;SEE IF DATA WAS STORED CORRECTLY
5619	037526	001401			BEQ	25\$	;BRANCH IF STORE WAS CORRECT
5620	037530	104071			ERROR	71	;INCORRECT STORE
5621	037532	012737	036740	001110	25\$: MOV	#20\$,SLPERR	;SET LOOP POINTER TO START OF TEST
5622	037540	012737	015432	000250	MOV	\$MMTRAP,\$MVEC	;RESTORE M.M. VECTOR TO NORMAL ROUTINE
5623	037546	000423			BR	TST51	;BRANCH TO NEXT TEST
5624							
5625							
5626	037550	012637	001436		10\$: MOV	(KSP)+,OLDPC	;SAVE PC & PS OF TRAP
5627	037554	012637	001440		MOV	(KSP)+,OLDPS	
5628	037560	013737	177572	001404	MOV	\$MMR0,\$PMR0	;SAVE \$MMR0 FOR ERROR TYPEOUT
5629	037566	013737	177576	001406	MOV	\$MMR2,\$PMR2	;SAVE \$MMR2 FOR ERROR TYPEOUT
5630	037574	042737	160000	177572	BIC	#160000,\$MMR0	;CLEAR ERROR BITS IN \$MMR0
5631	037602	104067			ERROR	67	;TRIED TO LOAD A N.R. PAGE 4
5632	037604	013746	001440		MOV	OLDPS,-(KSP)	;PUT PC & PS OF TRAP ON STACK
5633	037610	013746	001436		MOV	OLDPC,-(KSP)	
5634	037614	000002			RTI		;RETURN TO TEST
5635							
5636							
5637							
5638							
5639							
5640							
5641							
5642							
5643							
5644							
5645							
5646							
5647							
5648							
5649							
5650							
5651	037616						
5652	037616	000004			TST51:	SCOPE	
5653	037620	012737	040364	001446	MOV	#TST52,NXTTST	;SAVE STARTING ADDRESS OF NEXT
5654							;TEST FOR ESCAPE ON PARITY ERRORS
5655	037626	012700	077406		MOV	#77406,R0	;MAKE ALL USER I-SPACE PAGES RESIDENT
5656							;READ/WRITE LENGTH 200 BLOCKS
5657	037632	012702	000010		MOV	#10,R2	;SET COUNT TO LOAD 8 ADDRESSES
5658	037636	012701	177600		MOV	\$UIPDR0,R1	;PUT ADDRESS OF FIRST PDR IN R1
5659	037642	010021			19\$: MOV	R0,(R1)+	;LOAD R0 INTO PDR ADDRESSED BY R1
5660	037644	077202			SOB	R2,19\$	;BRANCH BACK TO 19\$ IF R2 IS NOT ZERO
5661	037646	012737	037654	001110	MOV	#20\$,SLPEPR	;SET LOOP ON ERROR TO 20\$
5662	037654	012737	140340	177776	20\$: MOV	#140340,\$PSW	;GO TO USER MODE FOR THIS TEST
5663	037662	012737	077406	172310	MOV	#77406,KIPDR4	;KERNEL I-SPACE PAGE 4 READ/WRITE
5664	037670	012737	000600	172350	MOV	#600,KIPAR4	;MAP KERNEL I PAGE 4 TO 12K

```

*****
;TEST 51      MOVE FROM PREVIOUS (KERNEL) I-SPACE TO USER MODE
;
;      THIS TEST CHECKS THAT IF THE PREVIOUS MODE IS KERNEL THE
;      FETCH IS FROM
;      KERNEL MODE.
;      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 10$, WHERE THE ERRORS ARE REPORTED.
*****

```

E09

 MAINDEC-11-DOKTA-A POP 11/6X MEM. MGMT. DIAG.
 DOKTAA.P11 07-FEB-77 10:30 TSI

 MACY11 27(1006) 07-FEB-77 10:37 PAGE 108
 MOVE FROM PREVIOUS (KERNEL) I-SPACE TO USER MODE

5665	037676	012737	000600	177650	MOV	#600,UIPAR4	MAP USER I PAGE 4 TO 12K
5666	037704	012700	036514		MOV	#36514,R0	LOAD DATA PATTERN INTO R0
5667	037710	010037	100000		MOV	R0,#100000	LOAD DATA PATTERN INTO PHY 60000
5668	037714	012702	100000		MOV	#100000,R2	LOAD VIRTUAL ADDRESS INTO R2
5669							
5670							;THE FOLLOWING WILL TEST DSTM=0 MFPI
5671	037720	012737	040316	000250	MOV	#105,MMVEC	SET M.M. VECTOR TO 105
5672	037726	105037	177610		CLRB	UIPAR4	MAKE USER I-SPACE PAGE 4 NON-RESIDENT
5673	037732	012737	140340	177776	MOV	#140340,PSW	MAKE PREVIOUS MODE KERNEL PRESENT USER
5674	037740	006506			MFPI	KSP	PUT KERNEL STACK POINTER ON USER STACK
5675	037742	022706	000700		CMP	#USESTK,USP	WAS SOMETHING PUSHED ON STACK AT 15
5676	037746	001407			BEQ	35	BRANCH IF NOTHING WAS PUSHED
5677	037750	012601			MOV	(USP)+,R1	POP USER STACK INTO R1
5678	037752	012702	001100		MOV	#KERSTK,R2	EXPECTING 1100 AS KSP
5679	037756	020201			CMP	R2,R1	DID YOU GET THE RIGHT POINTER?
5680	037760	001403			BEQ	25	BRANCH IF YOU DID
5681	037762	104064			ERROR	64	WRONG THING WAS PUSHED ON STACK
5682	037764	000401			BR	25	BRANCH TO NEXT TRY
5683	037766	104065			ERROR	65	NOTHING PUSHED ON STACK
5684	037770						;THE FOLLOWING WILL TEST DSTM=1 MFPI.
5685	037770	012737	037776	001110	MOV	#125,SLPERR	SET LOOP ON ERROR POINTER TO 125
5686	037776	012737	140340	177776	MOV	#140340,PSW	MAKE PREVIOUS MODE KERNEL PRESENT USER
5687	040004	012702	100000		MOV	#100000,R2	LOAD VIRTUAL ADDRESS INTO R2
5688	040010	006512			MFPI	(R2)	READ FROM PHYSICAL 60000
5689	040012	012601			MOV	(USP)+,R1	POP USER STACK INTO R1
5690	040014	020001			CMP	R0,R1	WAS DATA FETCHED SAME AS STORED
5691	040016	001401			BEQ	45	BRANCH IF CORRECT DATA WAS FETCHED
5692	040020	104066			ERROR	66	WRONG DATA WAS FETCHED
5693	040022						;THE FOLLOWING WILL TEST DSM=2 MFPI.
5694	040022	012737	040030	001110	MOV	#145,SLPERR	SET LOOP ON ERROR POINTER TO 145
5695	040030	012737	140340	177776	MOV	#140340,PSW	MAKE PREVIOUS MODE KERNEL PRESENT USER
5696	040036	012702	100000		MOV	#100000,R2	LOAD VIRTUAL ADDRESS INTO R2
5697	040042	006522			MFPI	(R2)+	READ FROM PHYSICAL 60000
5698	040044	012601			MOV	(USP)+,R1	POP USER STACK INTO R1
5699	040046	020001			CMP	R0,R1	WAS DATA FETCHED SAME AS STORED
5700	040050	001401			BEQ	55	BRANCH IF CORRECT DATA WAS FETCHED
5701	040052	104066			ERROR	66	WRONG DATA WAS FETCHED
5702	040054						;THE FOLLOWING WILL TEST DSTM=3 MFPI.
5703	040054	012737	040062	001110	MOV	#155,SLPERR	SET LOOP ON ERROR POINTER TO 155
5704	040062	012737	140340	177776	MOV	#140340,PSW	MAKE PREVIOUS MODE KERNEL PRESENT USER
5705	040070	006537	100000		MFPI	#100000	READ FROM PHYSICAL 60000
5706	040074	012601			MOV	(USP)+,R1	POP USER STACK INTO R1
5707	040076	020001			CMP	R0,R1	WAS DATA FETCHED SAME AS STORED
5708	040100	001401			BEQ	65	BRANCH IF CORRECT DATA WAS FETCHED
5709	040102	104066			ERROR	66	WRONG DATA WAS FETCHED
5710	040104						;THE FOLLOWING WILL TEST DSTM=4 MFPI.
5711	040104	012737	040112	001110	MOV	#165,SLPERR	SET LOOP ON ERROR POINTER TO 165
5712	040112	012737	140340	177776	MOV	#140340,PSW	MAKE PREVIOUS MODE DERNEL PRESENT USER
5713	040120	012702	100002		MOV	#100002,R2	LOAD VIRTUAL ADDRESS INTO R2
5714	040124	006542			MFPI	-(R2)	READ FROM PHYSICAL 60000
5715	040126	012601			MOV	(USP)+,R1	POP USER STACK INTO R1
5716	040130	020001			CMP	R0,R1	WAS DATA FETCHED SAME AS STORED
5717	040132	001401			BEQ	75	BRANCH IF CORRECT DATA WAS FETCHED
5718	040134	104066			ERROR	66	WRONG DATA WAS FETCHED
5719	040136						;THE FOLLOWING WILL TEST DSTM=5 MFPI.
5720							

F09

MAINDEC-11-DOKTA-A POP 11/6X MEM. MGMT. DIAG.
DOKTAA.P11 07-FEB-77 10:30 T51

MACY11 27(1006) 07-FEB-77 10:37 PAGE 109
MOVE FROM PREVIOUS (KERNEL) I-SPACE TO USER MODE

```

5721 040136 012737 040144 001110      MOV      #17$,SLPERR      ;SET LOOP ON ERROR POINTER TO 17$
5722 040144 012737 140340 177776 17$: MOV      #140340,PSW      ;MAKE PREVIOUS MODE KERNEL PRESENT USER
5723 040152 012737 100000 001202      MOV      #100000,$TMP2    ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
5724 040160 012702 001204              MOV      #($TMP2+2),R2    ;LOAD ADDRESS OF $TMP2+2 INTO R2
5725 040164 006552              MFPI      2-(R2)              ;READ FROM PHYSICAL 60000
5726 040166 012601              MOV      (USP)+,R1            ;POP USER STACK INTO R1
5727 040170 020001              CMP      R0,R1              ;WAS DATA FETCHED SAME AS STORED
5728 040172 001401              BEQ      B$                ;BRANCH IF CORRECT DATA FETCHED
5729 040174 104066              ERROR    66                ;WRONG DATA WAS FETCHED
5730 040176              8$: ;THE FOLLOWING WILL TEST DSTN=6 MFPI.
5731
5732 040176 012737 040204 001110      MOV      #18$,SLPERR      ;SET LOOP ON ERROR POINTER TO 18$.
5733 040204 012737 140340 177776 18$: MOV      #140340,PSW      ;MAKE PREVIOUS MODE KERNEL PRESENT USER
5734 040212 005002              CLR      R2                ;MAKE REGISTER 2 A ZERO
5735 040214 006562 100000              MFPI      100000(R2)        ;READ FROM PHYSICAL 60000
5736 040220 012601              MOV      (USP)+,R1            ;POP USER STACK INTO R1
5737 040222 020001              CMP      R0,R1              ;WAS DATA FETCHED SAME AS STORED
5738 040224 001401              BEQ      9$                ;BRANCH IF CORRECT DATA FETCHED
5739 040226 104066              ERROR    66                ;WRONG DATA WAS FETCHED
5740 040230              9$: ;THE FOLLOWING WILL TEST DSTN=7 MFPI.
5741
5742 040230 012737 040236 001110      MOV      #22$,SLPERR      ;SET LOOP ON ERROR POINTER TO 22$.
5743 040236 012737 140340 177776 22$: MOV      #140340,PSW      ;MAKE PREVIOUS MODE KERNEL PRESENT USER
5744 040244 012737 100000 001202      MOV      #100000,$TMP2    ;LOAD TEST LOC. VIRT. ADDR INTO $TMP2
5745 040252 012702 001202              MOV      #($TMP2),R2        ;LOAD ADDRESS OF $TMP2 INTO R2
5746 040256 006572 000000              MFPI      20(R2)              ;READ FROM PHYSICAL 60000
5747 040262 012601              MOV      (USP)+,R1            ;POP USER STACK INTO R1
5748 040264 020001              CMP      R0,R1              ;WAS DATA FETCHED SAME AS STORED
5749 040266 001401              BEQ      25$              ;BRANCH IF CORRECT DATA FETCHED
5750 040270 104066              ERROR    66                ;WRONG DATA WAS FETCHED
5751 040272 012737 015432 000250 25$: MOV      #MMTRAP,MMVEC      ;SET M.M. VECTOR TO NORMAL ROUTINE
5752 040300 012737 000340 177776      MOV      #00340,PSW      ;GO BACK TO KERNEL MODE, PREVIOUS KERNEL
5753 040306 012737 037654 001110      MOV      #20$,SLPERR      ;SET LOOP POINTER TO START OF TEST
5754 040314 000423              BR       TST52              ;BRANCH TO NEXT TEXT
5755
5756
5757 040316 012637 001436              10$: MOV      (KSP)+,OLDPC      ;SAVE PC & PS OF TRAP
5758 040322 012637 001440              MOV      (KSP)+,OLDPS
5759 040326 013737 177572 001404      MOV      MMRO,PMRO        ;SAVE MMRO FOR ERROR TYPEOUT
5760 040334 013737 177576 001406      MOV      MMR2,PMMR2      ;SAVE MMR2 FOR ERROR TYPEOUT
5761 040342 042737 160000 177572      BIC      #160000,MMRO     ;CLEAR ERROR BITS IN MMRO
5762 040350 104067              ERROR    67                ;TRIED TO READ NON-RESIDENT PAGE
5763 040352 013746 001440              MOV      OLDPS,-(KSP)    ;PUT PC & PS OF TRAP ON STACK
5764 040356 013746 001436              MOV      OLDPC,-(KSP)
5765 040362 000002              RTI                ;RETURN TO TEST
5766
5767
5768
5769
5770
5771
5772
5773
5774
5775 040364
5776 040364 000004

```

 *TEST 52 MOVE FROM/TO D-SPACE = MOVE FROM/TO I-SPACE
 *
 * THIS TEST CHECKS THAT SINCE THERE IS NO DISTINCTION
 * BETWEEN INSTRUCTION AND DATA SPACE IN THE 11/6X & 11/40
 * MFPD & MTPD SHOULD BE DECODED THE SAME AS MFPI & MTPI.
 *

TST52: SCOPE

5777	040366	012737	040476	001446		MOV	#TST53,NXTTST	:SAVE STARTING ADDRESS OF NEXT
5778								:TEST FOR ESCAPE ON PARITY ERRORS
5779	040374	012737	030340	177776	20\$:	MOV	#030340,PSW	:MAKE PREVIOUS MODE=USER,CURRENT=KERNEL
5780	040402	106506				MFPD	USP	:MFPD SHOULD ACT LIKE MFPI PUTTING
5781								:USER STACK POINTER ON THE KERNEL STACK
5782	040404	022706	001100			CMP	#KERSTK,KSP	:WAS SOMETHING PUSHED ON KERNEL STACK?
5783	040410	001407				BEQ	2\$	:BRANCH IF NO
5784	040412	012601				MOV	(KSP)+,R1	:POP KERNEL STACK INTO R1
5785	040414	012702	000700			MOV	#USESTK,R2	:EXPECTING TO GET 700 AS USP
5786	040420	020201				CMP	R2,R1	:DID GET RIGHT POINTER VALUE?
5787	040422	001403				BEQ	3\$	:BRANCH IF YES
5788	040424	104064				ERROR	64	:WRONG THING WAS PUSHED ON STACK
5789	040426	000401				BR	3\$	:BRANCH TO NEXT TRY
5790	040430	104065			2\$:	ERROR	65	:NOTHING PUSHED ON STACK
5791	040432	012737	040440	001110	3\$:	MOV	#4\$,SLPERR	:SET LOOP ON ERROR POINTER TO 4\$
5792	040440	012746	007777		4\$:	MOV	#7777,-(KSP)	:PUSH DATA ON KERNEL STACK
5793	040444	106606				MTPD	USP	:LOAD THE USER STACK POINTER
5794	040446	106506				MFPD	USP	:READ USER STACK POINTER
5795	040450	012601				MOV	(KSP)+,R1	:POP KERNEL STACK INTO R1
5796	040452	022701	007777			CMP	#7777,R1	:WAS USER STACK POINTER CHANGED?
5797	040456	001401				BEQ	5\$	:BRANCH IF YES
5798	040460	104070				ERROR	70	:USER STACK POINTER NOT CHANGED
5799	040462	012746	000700		5\$:	MOV	#USESTK,-(KSP)	:GET READY TO RESTORE USER STK. PTR.
5800	040466	106606				MTPD	USP	:RESTORE USER STACK POINTER
5801	040470	012737	040374	001110		MOV	#20\$,SLPERR	:SET LOOP POINTER TO START OF TEST
5802								
5803								:*****
5804								:*TEST 53 MOVE FROM PREVIOUS I=SPACE (PREVIOUS=CURRENT=KERNEL)
5805								:*
5806								:* THIS TEST CHECKS THAT IF BOTH PREVIOUS AND CURRENT MODES
5807								:* ARE KERNEL, AND THE SOURCE MODE IS 0, THE DESTINATION
5808								:* STACK IS NOT DECREMENTED BEFORE ACCESS.
5809								:* (FOR 11/40 THIS IS PURPOSE OF 'HOOK MUX')
5810								:* "MFPI KSP" SHOULD PUSH THE NON-DECREMENTED VALUE
5811								:* OF KSP (1100) ONTO THE STACK (AT LOC. 1076).
5812								:*****
5813	040476	000004			TST53:	SCOPE		
5814	040500	012737	040554	001446		MOV	#SEOP,NXTTST	:SET ESCAPE POINTER TO EOP ROUTINE
5815	040506	012737	040514	001110		MOV	#20\$,SLPERR	:SET LOOP ON ERROR POINTER TO 20\$
5816	040514	005037	177776		20\$:	CLR	#PSW	:SET PREVIOUS = CURRENT = KERNEL
5817	040520	012700	001100			MOV	#STACK,RO	:SETUP VALUE FOR STACK POINTER
5818	040524	010006				MOV	RO,KSP	:LOAD STACK POINTER
5819	040526	006506				MFPI	KSP	:THE VALUE "STACK" SHOULD BE PUSHED
5820								:BEFORE BEING DECREMENTED
5821	040530	011601				MOV	(KSP),R1	:READ DATA WHICH WAS PUSHED
5822	040532	020001				CMP	RO,R1	:WAS THE ORIGINAL VALUE OF THE
5823								:STACK POINTER PUSHED?
5824	040534	001401				BEQ	1\$	:BRANCH IF YES
5825	040536	104066				ERROR	66	:MFPI FETCHED WRONG DATA
5826	040540	005740			1\$:	TST	-(RO)	:SETUP EXPECTED STACK POINTER VALUE
5827	040542	020600				CMP	KSP,RO	:WAS THE STACK POINTER DECREMENTED?
5828	040544	001401				BEQ	2\$	:BRANCH IF YES
5829	040546	104065				ERROR	65	:STACK NOT PUSHED BY THE MFPI
5830	040550	012706	001100		2\$:	MOV	#STACK,KSP	:RESTORE STACK POINTER
5831								
5832								

H09

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 111
 DOKTAA.P11 07-FEB-77 10:30 TS3 MOVE FROM PREVIOUS I=SPACE (PREVIOUS=CURRENT=KERNEL)

```

5833
5834
5835
5836
5837
5838
5839
5840
5841
5842
5843
5844
5845
5846
5847
5848
5849 040554
5850 040554 000004
5851 040556 005037 001102
5852 040562 005037 001212
5853 040566 005237 001234
5854 040572 042737 100000 001234
5855 040600 005327
5856 040602 000001
5857 040604 003072
5858 040606 012737
5859 040610 000001
5860 040612 040602
5861 040614 104401 040622
5862 040620 000407
5863
5864 040640
5865 040640 013746 001234
5866
5867 040644 104405
5868 040646 104401 040654
5869 040652 000421
5870
5871 040716
5872 040716 013746 001112
5873
5874 040722 104405
5875 040724 104401 001223
5876 040730 005037 001112
5877 040734 013700 000042
5878 040740 001414
5879 040742 005046
5880 040744 012746 040752
5881 040750 000426
5882
5883 040752
5884 040752 013700 000042
5885 040756 001405
5886 040760 000005
5887 040762 004710
5888 040764 000240

.SBTTL *****

.SBTTL END OF PASS ROUTINE

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

$EOP:
      SCOPE
      CLR $TSTNM          ;; ZERO THE TEST NUMBER
      CLR $TIMES          ;; ZERO THE NUMBER OF ITERATIONS
      INC $PASS           ;; INCREMENT THE PASS NUMBER
      BIC $100000,$PASS   ;; DON'T ALLOW A NEG. NUMBER
      DEC (PC)+           ;; LOOP?
$EOPCT: .WORD 1
      BGT $DOAGN          ;; YES
      MOV (PC)+,$2(PC)+   ;; RESTORE COUNTER
$ENDCT: .WORD 1
      $EOPCT
      TYPE 65$           ;; TYPE ASCIZ STRING
      BR 64$             ;; GET OVER THE ASCIZ
;; 65$: .ASCIZ <12><15>/END PASS #/
64$:
      MOV $PASS,-(SP)     ;; SAVE $PASS FOR TYPEOUT
      TYPDS              ;; TYPE PASS NUMBER
      TYPE 67$           ;; GO TYPE--DECIMAL ASCII WITH SIGN
      BR 66$             ;; TYPE ASCIZ STRING
;; 67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
66$:
      MOV $ERTTL,-(SP)    ;; SAVE $ERTTL FOR TYPEOUT
      TYPDS              ;; TOTAL NUMBER OF ERRORS
      TYPE $SCLF         ;; GO TYPE--DECIMAL ASCII WITH SIGN
      CLR $ERTTL         ;; TYPE CARRIAGE RETURN, LINE FEED
      MOV $242,R0        ;; CLEAR ERROR TOTAL
      BEQ $DOAGN         ;; GET MONITOR ADDRESS
      CLR -(SP)          ;; BRANCH IF NO MONITOR
      MOV $SCLR.T,-(SP)   ;; INSURE THE "T" BIT IS CLEAR
      BR $RTAN           ;; SETUP FOR AN RTI OR RTT
      ;; GO DO AN RTI OR RTT TO LOAD THE PSW
      ;; WITH A CLEARED "T" BIT

$SCLR.T:
      MOV $242,R0        ;; INSURE R0 CONTAINS THE MONITORS
      BEQ $DOAGN         ;; RETURN ADDRESS
      RESET             ;; CLEAR THE WORLD
$SENDAD: JSR PC,(R0)     ;; GO TO MONITOR
      NOP               ;; SAVE ROOM

```

```

5889 040766 000240      NOP      ;;FOR
5890 040770 000240      NOP      ;;ACT11
5891 040772      SDOAGN:      TRAP      ;;PUSH OLD PSW AND PC ON STACK
5892 040772 104400      BIC      #20,(SP)      ;;CLEAR THE "T" BIT
5893 040774 042716 000020      BIT      #BIT12,2SWR      ;;RUN WITH TRACE TRAP?
5894 041000 032777 010000 140132      BNE      1$      ;;BR IF NO
5895 041006 001005      COM      $TBIT      ;;IS IT TIME FOR TRACE TRAP
5896 041010 005137 041034      BMI      1$      ;;BR IF NO
5897 041014 100402      BIS      #20,(SP)      ;;SET TRACE TRAP
5898 041016 052716 000020      IS:      MOV      $SLOOP,-(SP)      ;;JUMP TO START OF TEST
5899 041022 012746 041030      $RTN:      RTI      ;;RETURN--THIS IS CHANGED TO
5900 041026 000002      ;;AN "RTT" IF "RTT" IS A LEGAL
5901      ;;INSTRUCTION
5902
5903 041030      $LOOP:      JMP      2(PC)+      ;;RETURN
5904 041030 000137      $RTNAD:      .WORD      LOOP
5905 041032 020612      $TBIT:      .WORD      0      ;; "T" BIT STATE INDICATOR
5906 041034 000000      $ENULL:      .BYTE      -1,-1,0      ;;NULL CHARACTER STRING
5907 041036      377      000
5908 041042
5909
5910      .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
5911
5912      ;*****
5913      ;SAVE R0-R5
5914      ;CALL:
5915      ;      SAVREG
5916      ;UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
5917      ;
5918      ;TOP---(+16)
5919      ;+2---(+18)
5920      ;+4---R5
5921      ;+6---R4
5922      ;+8---R3
5923      ;+10---R2
5924      ;+12---R1
5925      ;+14---R0
5926
5927      $SAVREG:      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
5928 041042 010046      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
5929 041044 010146      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
5930 041046 010246      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
5931 041050 010346      MOV      R4,-(SP)      ;;PUSH R4 ON STACK
5932 041052 010446      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
5933 041054 010546      MOV      22(SP),-(SP)      ;;SAVE PS OF MAIN FLOW
5934 041056 016646 000022      MOV      22(SP),-(SP)      ;;SAVE PC OF MAIN FLOW
5935 041062 016646 000022      MOV      22(SP),-(SP)      ;;SAVE PS OF CALL
5936 041066 016646 000022      MOV      22(SP),-(SP)      ;;SAVE PC OF CALL
5937 041072 016646 000022      RTI
5938 041076 000002
5939
5940      ;*RESTORE R0-R5
5941      ;*CALL:
5942      ;      RESREG
5943 041100      $RESREG:      MOV      (SP)+,22(SP)      ;;RESTORE PC OF CALL
5944 041100 012666 000022

```

```

5945 041104 012666 000022      MOV      (SP)+,22(SP)      ;;RESTORE PS OF CALL
5946 041110 012666 000022      MOV      (SP)+,22(SP)      ;;RESTORE PC OF MAIN FLOW
5947 041114 012666 000022      MOV      (SP)+,22(SP)      ;;RESTORE PS OF MAIN FLOW
5948 041120 012605              MOV      (SP)+,R5          ;;POP STACK INTO R5
5949 041122 012604              MOV      (SP)+,R4          ;;POP STACK INTO R4
5950 041124 012603              MOV      (SP)+,R3          ;;POP STACK INTO R3
5951 041126 012602              MOV      (SP)+,R2          ;;POP STACK INTO R2
5952 041130 012601              MOV      (SP)+,R1          ;;POP STACK INTO R1
5953 041132 012600              MOV      (SP)+,R0          ;;POP STACK INTO R0
5954 041134 000002              RTI
5955                               .SBTTL  TYPE ROUTINE
5956
5957                               ;*****
5958                               ;ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
5959                               ;THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
5960                               ;NOTE1:          $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
5961                               ;NOTE2:          $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
5962                               ;NOTE3:          $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
5963                               ;
5964                               ;CALL:
5965                               ;1) USING A TRAP INSTRUCTION
5966                               ;      TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
5967                               ;OR
5968                               ;      TYPE
5969                               ;      MESADR
5970                               ;
5971
5972 041136 105737 001157      $TYPE:  TSTB      $TPFLG      ;; IS THERE A TERMINAL?
5973 041142 100002              BPL          1$            ;; BR IF YES
5974 041144 000000              HALT                    ;; HALT HERE IF NO TERMINAL
5975 041146 000430              BR          3$            ;; LEAVE
5976 041150 010046              1$:  MOV      R0,-(SP)      ;; SAVE R0
5977 041152 017600 000002      MOV      22(SP),R0      ;; GET ADDRESS OF ASCIZ STRING
5978 041156 122737 000001 001246      CMPB      #APTENV,$ENV      ;; RUNNING IN APT MODE
5979 041164 001011              BNE          62$          ;; NO GO CHECK FOR APT CONSOLE
5980 041166 132737 000100 001247      BITB      #APTSPool,$ENV      ;; SPOOL MESSAGE TO APT
5981 041174 001405              BEQ          62$          ;; NO GO CHECK FOR CONSOLE
5982 041176 010037 041206      MOV      R0,61$          ;; SETUP MESSAGE ADDRESS FOR APT
5983 041202 004737 041426      JSR          PC,$ATY3      ;; SPOOL MESSAGE TO APT
5984 041206 000000              .WORD      0            ;; MESSAGE ADDRESS
5985 041210 132737 000040 001247      61$:  BITB      #APTCsup,$ENV      ;; APT CONSOLE SUPPRESSED
5986 041216 001003              BNE          60$          ;; YES, SKIP TYPE OUT
5987 041220 112046              2$:  MOVB      (R0)+,-(SP)      ;; PUSH CHARACTER TO BE TYPED ONTO STACK
5988 041222 001005              BNE          4$            ;; BR IF IT ISN'T THE TERMINATOR
5989 041224 005726              TST      (SP)+          ;; IF TERMINATOR POP IT OFF THE STACK
5990 041226 012600              60$:  MOV      (SP)+,R0      ;; RESTORE R0
5991 041230 062716 000002      3$:  ADD      #2,(SP)      ;; ADJUST RETURN PC
5992 041234 000002              RTI
5993 041236 122716 000011      4$:  CMPB      #HT,(SP)      ;; BRANCH IF <HT>
5994 041242 001430              BEQ          8$            ;;
5995 041244 122716 000200      CMPB      #CRLF,(SP)      ;; BRANCH IF NOT <CRLF>
5996 041250 001006              BNE          5$            ;;
5997 041252 005726              TST      (SP)+          ;; POP <CR><LF> EQUIV
5998 041254 104401              TYPE                    ;; TYPE A CR AND LF
5999 041256 001223              $CRLF
6000 041260 105037 041414      CLRB      $CHARCNT      ;; CLEAR CHARACTER COUNT

```

K09

MAINDEC-11-DQKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 114
 DQKTA.P11 07-FEB-77 10:30 TYPE ROUTINE

```

6001 041264 000755          BR      2$          ;; GET NEXT CHARACTER
6002 041266 004737 041350  5$: JSR    PC,$TYPEC  ;; GO TYPE THIS CHARACTER
6003 041272 123726 001156  6$: CMPB  $FILLC,(SP)+  ;; IS IT TIME FOR FILLER CHARS.?
6004 041276 001350          BNE     2$          ;; IF NO GO GET NEXT CHAR.
6005 041300 013746 001154          MOV    $NULL,-(SP)  ;; GET # OF FILLER CHARS. NEEDED
6006                                AND THE NULL CHAR.
6007 041304 105366 000001  7$: DECB   1(SP)  ;; DOES A NULL NEED TO BE TYPED?
6008 041310 002770          BLT     6$          ;; BR IF NO--GO POP THE NULL OFF OF STACK
6009 041312 004737 041350          JSR    PC,$TYPEC  ;; GO TYPE A NULL
6010 041316 105337 041414          DECB  $CHARCNT  ;; DO NOT COUNT AS A COUNT
6011 041322 000770          BR      7$          ;; LOOP
6012
6013                                ;HORIZONTAL TAB PROCESSOR
6014
6015 041324 112716 000040  8$: MOVB   #' (SP)  ;; REPLACE TAB WITH SPACE
6016 041330 004737 041350  9$: JSR    PC,$TYPEC  ;; TYPE A SPACE
6017 041334 132737 000007 041414  BITB   #7,$CHARCNT  ;; BRANCH IF NOT AT
6018 041342 001372          BNE     9$          ;; TAB STOP
6019 041344 005726          TST     (SP)+  ;; POP SPACE OFF STACK
6020 041346 000724          BR      2$          ;; GET NEXT CHARACTER
6021 041350 105777 137574  $TYPEC: TSTB  2$TPS  ;; WAIT UNTIL PRINTER IS READY
6022 041354 100375          BPL     $TYPEC
6023 041356 116677 000002 137566  MOVB   2(SP),2$TPB  ;; LOAD CHAR TO BE TYPED INTO DATA REG.
6024 041364 122766 000015 000002  CMPB   #CR,2(SP)  ;; IS CHARACTER A CARRIAGE RETURN?
6025 041372 001003          BNE     1$          ;; BRANCH IF NO
6026 041374 105037 041414          CLRB   $CHARCNT  ;; YES--CLEAR CHARACTER COUNT
6027 041400 000406          BR      $TYPEX  ;; EXIT
6028 041402 122766 000012 000002  1$: CMPB  #LF,2(SP)  ;; IS CHARACTER A LINE FEED?
6029 041410 001402          BEQ     $TYPEX  ;; BRANCH IF YES
6030 041412 105227          INCB   (PC)+  ;; COUNT THE CHARACTER
6031 041414 000000          $CHARCNT: WORD 0  ;; CHARACTER COUNT STORAGE
6032 041416 000207          $TYPEX: RTS   PC
6033
6034                                .SBTTL  APT COMMUNICATIONS ROUTINE
6035
6036                                ;*****
6037 041420 112737 000001 041664  $ATY1: MOVB   #1,$FFLG  ;; TO REPORT FATAL ERROR
6038 041426 112737 000001 041662  $ATY3: MOVB   #1,$MFLG  ;; TO TYPE A MESSAGE
6039 041434 000403          BR      $ATYC
6040 041436 112737 000001 041664  $ATY4: MOVB   #1,$FFLG  ;; TO ONLY REPORT FATAL ERROR
6041 041444          $ATYC:
6042 041444 010046          MOV    R0,-(SP)  ;; PUSH R0 ON STACK
6043 041446 010146          MOV    R1,-(SP)  ;; PUSH R1 ON STACK
6044 041450 105737 041662          TSTB  $MFLG  ;; SHOULD TYPE A MESSAGE?
6045 041454 001450          BEQ     5$          ;; IF NOT: BR
6046 041456 122737 000001 001246  CMPB   #APTENV,$ENV  ;; OPERATING UNDER APT?
6047 041464 001031          BNE     3$          ;; IF NOT: BR
6048 041466 132737 000100 001247  BITB   #APTSPool,$ENVM  ;; SHOULD SPOOL MESSAGES?
6049 041474 001425          BEQ     3$          ;; IF NOT: BR
6050 041476 017600 000004          MOV    24(SP),R0  ;; GET MESSAGE ADDR.
6051 041502 062766 000002 000004  ADD     #2,4(SP)  ;; BUMP RETURN ADDR.
6052 041510 005737 001226  1$: TST     $MSGTYPE  ;; SEE IF DONE W/ LAST XMISSION?
6053 041514 001375          BNE     1$          ;; IF NOT: WAIT
6054 041516 010037 001242          MOV    R0,$MSGAD  ;; PUT ADDR IN MAILBOX
6055 041522 105720          2$: TSTB  (R0)+  ;; FIND END OF MESSAGE
6056 041524 001376          BNE     2$

```

```

6057 041526 163700 001242      SUB      $MSGAD,RO      ;; SUB START OF MESSAGE
6058 041532 006200              ASR      RO              ;; GET MESSAGE LNTH IN WORDS
6059 041534 010037 001244      MOV      RO,$MSGGLT      ;; PUT LENGTH IN MAILBOX
6060 041540 012737 000004 001226  MOV      #4,$MSGTYPE      ;; TELL APT TO TAKE MSG.
6061 041546 000413              BR      5$
6062 041550 017637 000004 041574 3$:      MOV      24(SP),4$      ;; PUT MSG ADDR IN JSR LINKAGE
6063 041556 062766 000002 000004      ADD      #2,4(SP)      ;; BUMP RETURN ADDRESS
6064 041564 013746 177776      MOV      177776,-(SP)      ;; PUSH 177776 ON STACK
6065 041570 004737 041136      JSR      PC,$TYPE      ;; CALL TYPE MACRO
6066 041574 000000              4$:      .WORD      0
6067 041576              5$:
6068 041576 105737 041664      10$:      TSTB      $FFLG      ;; SHOULD REPORT FATAL ERROR?
6069 041602 001416              BEQ      12$      ;; IF NOT: BR
6070 041604 005737 001246      TST      $ENV      ;; RUNNING UNDER APT?
6071 041610 001413              BEQ      12$      ;; IF NOT: BR
6072 041612 005737 001226      11$:      TST      $MSGTYPE      ;; FINISHED LAST MESSAGE?
6073 041616 001375              BNE      11$      ;; IF NOT: WAIT
6074 041620 017637 000004 001230      MOV      24(SP),$FATAL      ;; GET ERROR #
6075 041626 062766 000002 000004      ADD      #2,4(SP)      ;; BUMP RETURN ADDR.
6076 041634 005237 001226      INC      $MSGTYPE      ;; TELL APT TO TAKE ERROR
6077 041640 105037 041664      12$:      CLRB      $FFLG      ;; CLEAR FATAL FLAG
6078 041644 105037 041663      CLRB      $LFLG      ;; CLEAR LOG FLAG
6079 041650 105037 041662      CLRB      $MFLG      ;; CLEAR MESSAGE FLAG
6080 041654 012601              MOV      (SP)+,R1      ;; POP STACK INTO R1
6081 041656 012600              MOV      (SP)+,RO      ;; POP STACK INTO RO
6082 041660 000207              RTS      PC      ;; RETURN
6083 041662 000              $MFLG: .BYTE      0      ;; MESSG. FLAG
6084 041663 000              $LFLG: .BYTE      0      ;; LOG FLAG
6085 041664 000              $FFLG: .BYTE      0      ;; FATAL FLAG
6086              .EVEN
6087              APTSIZE=200
6088              APTENV=001
6089              APTSPool=100
6090              APTCSUP=040
6091              .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
6092
6093 *****
6094 *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
6095 *OCTAL (ASCII) NUMBER AND TYPE IT.
6096 *$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
6097 *CALL:
6098 *      MOV      NUM,-(SP)      ;; NUMBER TO BE TYPED
6099 *      TYPOS      ;; CALL FOR TYPEOUT
6100 *      .BYTE      N      ;; N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
6101 *      .BYTE      M      ;; M=1 OR 0
6102 *      ;;1=TYPE LEADING ZEROS
6103 *      ;;0=SUPPRESS LEADING ZEROS
6104 *
6105 *$STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
6106 *$TYPOS OR $TYPOC
6107 *CALL:
6108 *      MOV      NUM,-(SP)      ;; NUMBER TO BE TYPED
6109 *      TYPON      ;; CALL FOR TYPEOUT
6110 *
6111 *$STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
6112 *CALL:

```

M09

MAINDEC-11-DQKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 116
 DQKTA.P11 07-FEB-77 10:30 BINARY TO OCTAL (ASCII) AND TYPE

6113					;	MOV	NUM,-(SP)	;;	NUMBER TO BE TYPED
6114					;	TYPOC		;;	CALL FOR TYPEOUT
6115									
6116	041666	017646	000000		STYPOS:	MOV	2(SP),-(SP)	;;	PICKUP THE MODE
6117	041672	116637	000001	042111		MOVB	1(SP),SOFILL	;;	LOAD ZERO FILL SWITCH
6118	041700	112637	042113			MOVB	(SP)+,SOMODE+1	;;	NUMBER OF DIGITS TO TYPE
6119	041704	062716	000002			ADD	#2,(SP)	;;	ADJUST RETURN ADDRESS
6120	041710	000406				BR	STYPON		
6121	041712	112737	000001	042111	STYPOC:	MOVB	#1,SOFILL	;;	SET THE ZERO FILL SWITCH
6122	041720	112737	000006	042113		MOVB	#6,SOMODE+1	;;	SET FOR SIX(6) DIGITS
6123	041726	112737	000005	042110	STYPON:	MOVB	#5,SOCNT	;;	SET THE ITERATION COUNT
6124	041734	010346				MOV	R3,-(SP)	;;	SAVE R3
6125	041736	010446				MOV	R4,-(SP)	;;	SAVE R4
6126	041740	010546				MOV	R5,-(SP)	;;	SAVE R5
6127	041742	113704	042113			MOVB	SOMODE+1,R4	;;	GET THE NUMBER OF DIGITS TO TYPE
6128	041746	005404				NEG	R4		
6129	041750	062704	000006			ADD	#6,R4	;;	SUBTRACT IT FOR MAX. ALLOWED
6130	041754	110437	042112			MOVB	R4,SOMODE	;;	SAVE IT FOR USE
6131	041760	113704	042111			MOVB	SOFILL,R4	;;	GET THE ZERO FILL SWITCH
6132	041764	016605	000012			MOV	12(SP),R5	;;	PICKUP THE INPUT NUMBER
6133	041770	005003				CLR	R3	;;	CLEAR THE OUTPUT WORD
6134	041772	006105			1\$:	ROL	R5	;;	ROTATE MSB INTO "C"
6135	041774	000404				BR	3\$	;;	GO DO MSB
6136	041776	006105			2\$:	ROL	R5	;;	FORM THIS DIGIT
6137	042000	006105				ROL	R5		
6138	042002	006105				ROL	R5		
6139	042004	010503				MOV	R5,R3		
6140	042006	006103			3\$:	ROL	R3	;;	GET LSB OF THIS DIGIT
6141	042010	105337	042112			DECB	SOMODE	;;	TYPE THIS DIGIT?
6142	042014	100016				BPL	7\$	;;	BR IF NO
6143	042016	042703	177770			BIC	#177770,R3	;;	GET RID OF JUNK
6144	042022	001002				BNE	4\$	;;	TEST FOR 0
6145	042024	005704				TST	R4	;;	SUPPRESS THIS 0?
6146	042026	001403				BEQ	5\$	;;	BR IF YES
6147	042030	005204			4\$:	INC	R4	;;	DON'T SUPPRESS ANYMORE 0'S
6148	042032	052703	000060			BIS	#'0,R3	;;	MAKE THIS DIGIT ASCII
6149	042036	052703	000040		5\$:	BIS	#' ,R3	;;	MAKE ASCII IF NOT ALREADY
6150	042042	110337	042106			MOVB	R3,8\$	;;	SAVE FOR TYPING
6151	042046	104401	042106			TYPE	8\$	;;	GO TYPE THIS DIGIT
6152	042052	105337	042110		7\$:	DECB	SOCNT	;;	COUNT BY 1
6153	042056	003347				BGT	2\$	;;	BR IF MORE TO DO
6154	042060	002402				BLT	6\$	;;	BR IF DONE
6155	042062	005204				INC	R4	;;	INSURE LAST DIGIT ISN'T A BLANK
6156	042064	000744				BR	2\$	;;	GO DO THE LAST DIGIT
6157	042066	012605			6\$:	MOV	(SP)+,R5	;;	RESTORE R5
6158	042070	012604				MOV	(SP)+,R4	;;	RESTORE R4
6159	042072	012603				MOV	(SP)+,R3	;;	RESTORE R3
6160	042074	016666	000002	000004		MOV	2(SP),4(SP)	;;	SET THE STACK FOR RETURNING
6161	042102	012616				MOV	(SP)+,(SP)		
6162	042104	000002				RTI		;;	RETURN
6163	042106	000			8\$:	.BYTE	0	;;	STORAGE FOR ASCII DIGIT
6164	042107	000				.BYTE	0	;;	TERMINATOR FOR TYPE ROUTINE
6165	042110	000			SOCNT:	.BYTE	0	;;	OCTAL DIGIT COUNTER
6166	042111	000			SOFILL:	.BYTE	0	;;	ZERO FILL SWITCH
6167	042112	000000			SOMODE:	.WORD	0	;;	NUMBER OF DIGITS TO TYPE
6168					.SBTTL	CONVERT	BINARY TO DECIMAL AND TYPE ROUTINE		


```

6169
6170
6171
6172
6173
6174
6175
6176
6177
6178
6179
6180 042114
6181 042114 010046
6182 042116 010146
6183 042120 010246
6184 042122 010346
6185 042124 010546
6186 042126 012746 020200
6187 042132 016605 000020
6188 042136 100004
6189 042140 005405
6190 042142 112766 000055 000001
6191 042150 005000
6192 042152 012703 042330
6193 042156 112723 000040
6194 042162 005002
6195 042164 016001 042320
6196 042170 160105
6197 042172 002402
6198 042174 005202
6199 042176 000774
6200 042200 060105
6201 042202 005702
6202 042204 001002
6203 042206 105716
6204 042210 100407
6205 042212 106316
6206 042214 103003
6207 042216 116663 000001 177777
6208 042224 052702 000060
6209 042230 052702 000040
6210 042234 110223
6211 042236 005720
6212 042240 020027 000010
6213 042244 002746
6214 042246 003002
6215 042250 010502
6216 042252 000764
6217 042254 105726
6218 042256 100003
6219 042260 116663 177777 177776
6220 042266 105013
6221 042270 012605
6222 042272 012603
6223 042274 012602
6224 042276 012601

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

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

```


B10

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 118
 DOKTAA.P11 07-FEB-77 10:30 CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

6225 042300 012600      MOV      (SP)+,RO      ;;POP STACK INTO RO
6226 042302 104401 042330  TYPE      $DBLK      ;;NOW TYPE THE NUMBER
6227 042306 016666 000002 000004  MOV      2(SP),4(SP)      ;;ADJUST THE STACK
6228 042314 012616      MOV      (SP)+,(SP)
6229 042316 000002      RTI                      ;;RETURN TO USER
6230 042320 023420      $DTBL: 10000.
6231 042322 001750      1000.
6232 042324 000144      100.
6233 042326 000012      10.
6234 042330 000004      $DBLK: .BLKW 4
        .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.
6243 042340 010046      $TRAP: MOV      RO,-(SP)      ;;SAVE RO
6244 042342 016600 000002  MOV      2(SP),RO      ;;GET TRAP ADDRESS
6245 042346 005740      TST      -(RO)      ;;BACKUP BY 2
6246 042350 111000      MOV      (RO),RO      ;;GET RIGHT BYTE OF TRAP
6247 042352 006300      ASL      RO      ;;POSITION FOR INDEXING
6248 042354 016000 042374  MOV      $TRAPD(RO),RO      ;;INDEX TO TABLE
6249 042360 000200      RTS      RO      ;;GO TO ROUTINE

        ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
6254 042362 011646      $TRAP2: MOV      (SP),-(SP)      ;;MOVE THE PC DOWN
6255 042364 016666 000004 000002  MOV      4(SP),2(SP)      ;;MOVE THE PSW DOWN
6256 042372 000002      RTI                      ;;RESTORE THE PSW

        .SBTTL TRAP TABLE

        ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
        ;*BY THE "TRAP" INSTRUCTION.
        ;
        ; ROUTINE
        ;-----
6265 042374 042362      $TRPAD: .WORD      $TRAP2
6266 042376 041136      $TYPE      ;;CALL=TYPE      TRAP+1(104401) TTY TYPEOUT ROUTINE
6267 042400 041712      $TYPOC     ;;CALL=TYPOC      TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
6268 042402 041666      $TYPOS     ;;CALL=TYPOS      TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
6269 042404 041726      $TYPON     ;;CALL=TYPON      TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
6270 042406 042114      $TYPDS     ;;CALL=TYPDS      TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
6271
6272
6273 042410 041042      $SAVREG     ;;CALL=SAVREG      TRAP+6(104406) SAVE RO-R5 ROUTINE
6274 042412 041100      $RESREG     ;;CALL=RESREG      TRAP+7(104407) RESTORE RO-R5 ROUTINE
6275 042414 044120      $TBITOFF    ;;CALL=TBITO      TRAP+10(104410) TURN OFF T-BIT TRAPPING
6276 042416 044146      $TBITRESTORE ;;CALL=TBITR      TRAP+11(104411) RETURN T-BIT TO ITS PREVIOUS CON
6277
6278 .SBTTL POWER DOWN AND UP ROUTINES
6279
6280 ;*****

```

```

6281      :POWER DOWN ROUTINE
6282 042420 012737 042612 000024 $PWRDN: MOV    $SILLUP,2#PWRVEC    ;;SET FOR FAST UP
6283 042426 012737 000340 000026      MOV    #340,2#PWRVEC+2    ;;PRIO:7
6284 042434 010046      MOV    R0,-(SP)      ;;PUSH R0 ON STACK
6285 042436 010146      MOV    R1,-(SP)      ;;PUSH R1 ON STACK
6286 042440 010246      MOV    R2,-(SP)      ;;PUSH R2 ON STACK
6287 042442 010346      MOV    R3,-(SP)      ;;PUSH R3 ON STACK
6288 042444 010446      MOV    R4,-(SP)      ;;PUSH R4 ON STACK
6289 042446 010546      MOV    R5,-(SP)      ;;PUSH R5 ON STACK
6290 042450 017746 136464      MOV    2SWR,-(SP)      ;;PUSH 2SWR ON STACK
6291 042454 010637 042616      MOV    SP,$SAVR6      ;;SAVE SP
6292 042460 012737 042472 000024      MOV    $PWRUP,2#PWRVEC    ;;SET UP VECTOR
6293 042466 000000      HALT
6294 042470 000776      BR      -2          ;;HANG UP
6295
6296      ;*****
6297      :POWER UP ROUTINE
6298 042472 012737 042612 000024 $PWRUP: MOV    $SILLUP,2#PWRVEC    ;;SET FOR FAST DOWN
6299 042500 013706 042616      MOV    $SAVR6,SP      ;;GET SP
6300 042504 005037 042616      CLR    $SAVR6      ;;WAIT LOOP FOR THE TTY
6301 042510 005237 042616      1$: INC    $SAVR6      ;;WAIT FOR THE INC
6302 042514 001375      BNE    1$      ;;OF WORD
6303 042516 105737 001450      TSTB   FORTY      ;;EXECUTING ON AN 11/40?
6304 042522 001003      BNE    2$      ;;BRANCH IF YES
6305 042524 011600      MOV    (SP),R0      ;;GET OLD SW. REG. CONTENTS
6306 042526 076600      MED      ;;WRITE THE CONSOLE SW. REG.
6307 042530 000226      WCNSSW      ;;WITH ITS PREVIOUS VALUE
6308 042532
6309 042532 012677 136402      2$: MOV    (SP)+,2SWR      ;;POP STACK INTO 2SWR
6310 042536 012605      MOV    (SP)+,R5      ;;POP STACK INTO R5
6311 042540 012604      MOV    (SP)+,R4      ;;POP STACK INTO R4
6312 042542 012603      MOV    (SP)+,R3      ;;POP STACK INTO R3
6313 042544 012602      MOV    (SP)+,R2      ;;POP STACK INTO R2
6314 042546 012601      MOV    (SP)+,R1      ;;POP STACK INTO R1
6315 042550 012600      MOV    (SP)+,R0      ;;POP STACK INTO R0
6316 042552 012737 042420 000024      MOV    $PWRDN,2#PWRVEC    ;;SET UP THE POWER DOWN VECTOR
6317 042560 012737 000340 000026      MOV    #340,2#PWRVEC+2    ;;PRIO:7
6318 042566 104401      TYPE      ;;REPORT THE POWER FAILURE
6319 042570 042620      SPWRMG: .WORD  PWRMSG      ;;POWER FAIL MESSAGE POINTER
6320 042572 012716      MOV    (PC)+,(SP)      ;;RESTART AT START
6321 042574 020066      SPWRAD: .WORD  START      ;;RESTART ADDRESS
6322 042576 042766 000020 000002      BIC    #20,2(SP)      ;;CLEAR "T" BIT
6323 042604 005037 041034      CLR    $TBIT      ;;CLEAR THE "T" BIT FLAG
6324 042610 000002      RTI
6325 042612 000000      $SILLUP: HALT      ;;THE POWER UP SEQUENCE WAS STARTED
6326 042614 000776      BR      -2          ;;BEFORE THE POWER DOWN WAS COMPLETE
6327 042616 000000      $SAVR6: 0      ;;PUT THE SP HERE
6328 042620 006412 047520 042527      PWRMSG: .ASCII <12><15>'POWER FAILURE, RESTARTING PROGRAM'
6329 042626 020122 040506 046111
6330 042634 051125 026105 051040
6331 042642 051505 040524 052122
6332 042650 047111 020107 051120
6333 042656 043517 040522 000115
6334      .EVEN
6335
6336      .SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

```

```

6337
6338
6339
6340
6341
6342
6343
6344
6345
6346
6347 042664 104406
6348 042666 016601 000002
6349 042672 012705 043003
6350 042676 012704 000014
6351 042702 012703 177770
6352 042706 012100
6353 042710 012101
6354 042712 005002
6355 042714 110245
6356 042716 010002
6357 042720 005304
6358 042722 003007
6359 042724 001405
6360 042726 005205
6361 042730 010566 000002
6362 042734 104407
6363 042736 000207
6364 042740 006203
6365 042742 006001
6366 042744 006000
6367 042746 006001
6368 042750 006000
6369 042752 006001
6370 042754 006000
6371 042756 040302
6372 042760 062702 000060
6373 042764 000753
6374 042766 000016
6375
6376
6377
6378
6379
6380
6381
6382
6383
6384
6385
6386
6387
6388
6389
6390 043004
6391 043004 005037 001444
6392 043010 005037 001432

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

.SBTTL SCOPE HANDLER ROUTINE

;*****
;THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
;AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
;AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;SW14=1      LOOP ON TEST
;SW11=1      INHIBIT ITERATIONS
;SW09=1      LOOP ON ERROR
;SW08=1      LOOP ON TEST IN SWR<6:0>
;CALL
;      SCOPE                      ;;SCOPE=IOT

$SCOPE:
      CLR      RETRY              ;;CLEAR THE RETRY FLAG BEFORE THIS TEST
      CLR      ERRCNT            ;;CLEAR THE MULTIPLE ERROR COUNTER

```

E10

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 121
 DOKTAA.P11 07-FEB-77 10:30 SCOPE HANDLER ROUTINE

```

6393 043014 032777 040000 136116 1$: BIT #BIT14,2SWR ;: LOOP ON PRESENT TEST?
6394 043022 001117 BNE $OVER ;: YES IF SW14=1
6395 ;: *****START OF CODE FOR THE XOR TESTER*****
6396 043024 000416 $XTSTR: BR 6$ ;: IF RUNNING ON THE "XOR" TESTER CHANGE
6397 ;: THIS INSTRUCTION TO A "NOP" (NOP=240)
6398 043026 013746 000004 MOV 2$ERRVEC, -(SP) ;: SAVE THE CONTENTS OF THE ERROR VECTOR
6399 043032 012737 043052 000004 MOV 5$, 2$ERRVEC ;: SET FOR TIMEOUT
6400 043040 005737 177060 TST 2$177060 ;: TIME OUT ON XOR?
6401 043044 012637 000004 MOV (SP)+, 2$ERRVEC ;: RESTORE THE ERROR VECTOR
6402 043050 000466 BR $SVLAD ;: GO TO THE NEXT TEST
6403 043052 022626 5$: CMP (SP)+, (SP)+ ;: CLEAR THE STACK AFTER A TIME OUT
6404 043054 012637 000004 MOV (SP)+, 2$ERRVEC ;: RESTORE THE ERROR VECTOR
6405 043060 000426 BR 7$ ;: LOOP ON THE PRESENT TEST
6406 043062 6$: ;: *****END OF CODE FOR THE XOR TESTER*****
6407 043062 032777 000400 136050 BIT #BIT08,2SWR ;: LOOP ON SPEC. TEST?
6408 043070 001407 BEQ 2$ ;: BR IF NO
6409 043072 017746 136042 MOV 2$SWR, -(SP) ;: SET DESIRED TEST NUM. FROM SWR
6410 043076 042716 000200 BIC 5$SWRMK, (SP) ;: STRIP AWAY UNDESIRED BITS
6411 043102 122637 001102 CMPB (SP)+, $TSTNM ;: ON THE RIGHT TEST?
6412 043106 001465 BEQ $OVER ;: BR IF YES
6413 043110 105737 001103 2$: TSTB $ERFLG ;: HAS AN ERROR OCCURRED?
6414 043114 001421 BEQ 3$ ;: BR IF NO
6415 043116 123737 001115 001103 CMPB $ERMAX, $ERFLG ;: MAX. ERRORS FOR THIS TEST OCCURRED?
6416 043124 101015 BHI 3$ ;: BR IF NO
6417 043126 032777 001000 136004 BIT #BIT09,2SWR ;: LOOP ON ERROR?
6418 043134 001404 BEQ 4$ ;: BR IF NO
6419 043136 013737 001110 001106 7$: MOV $LPERR, $LPADR ;: SET LOOP ADDRESS TO LAST SCOPE
6420 043144 000446 BR $OVER
6421 043146 105037 001103 4$: CLRB $ERFLG ;: ZERO THE ERROR FLAG
6422 043152 005037 001212 CLR $TIMES ;: CLEAR THE NUMBER OF ITERATIONS TO MAKE
6423 043156 000415 BR 1$ ;: ESCAPE TO THE NEXT TEST
6424 043160 032777 004000 135752 3$: BIT #BIT11,2SWR ;: INHIBIT ITERATIONS?
6425 043166 001011 BNE 1$ ;: BR IF YES
6426 043170 005737 001234 TST $PASS ;: IF FIRST PASS OF PROGRAM
6427 043174 001406 BEQ 1$ ;: INHIBIT ITERATIONS
6428 043176 005237 001104 INC $ICNT ;: INCREMENT ITERATION COUNT
6429 043202 023737 001212 001104 CMP $TIMES, $ICNT ;: CHECK THE NUMBER OF ITERATIONS MADE
6430 043210 002024 BGE $OVER ;: BR IF MORE ITERATION REQUIRED
6431 043212 012737 000001 001104 1$: MOV #1, $ICNT ;: REINITIALIZE THE ITERATION COUNTER
6432 043220 013737 043276 001212 MOV $MXCNT, $TIMES ;: SET NUMBER OF ITERATIONS TO DO
6433 043226 105237 001102 $SVLAD: INCB $TSTNM ;: COUNT TEST NUMBERS
6434 043232 113737 001102 001232 MOVB $TSTNM, $TESTN ;: SET TEST NUMBER IN APT MAILBOX
6435 043240 011637 001106 MOV (SP), $LPADR ;: SAVE SCOPE LOOP ADDRESS
6436 043244 011637 001110 MOV (SP), $LPERR ;: SAVE ERROR LOOP ADDRESS
6437 043250 005037 001214 CLR $ESCAPE ;: CLEAR THE ESCAPE FROM ERROR ADDRESS
6438 043254 112737 000001 001115 MOVB #1, $ERMAX ;: ONLY ALLOW ONE(1) ERROR ON NEXT TEST
6439 043262 013777 001102 135652 $OVER: MOV $TSTNM, 2$DISPLAY ;: DISPLAY TEST NUMBER
6440 043270 013716 001106 MOV $LPADR, (SP) ;: FUDGE RETURN ADDRESS
6441 043274 000002 RTI ;: FIXES PS
6442 043276 000310 $MXCNT: 200 ;: MAX. NUMBER OF ITERATIONS
6443 .SBTTL ERROR HANDLER ROUTINE
6444 ;: *****
6445 ;: *THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
6446 ;: *SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
6447 ;: *AND GO TO ERTYPE ON ERROR
6448

```

F10

MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 122
 DQKTA.P11 07-FEB-77 10:30 ERROR HANDLER ROUTINE

```

6449      ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
6450      ;*SW15=1      HALT ON ERROR
6451      ;*SW13=1      INHIBIT ERROR TYPEOUTS
6452      ;*SW10=1      BELL ON ERROR
6453      ;*SW09=1      LOOP ON ERROR
6454      ;*CALL
6455      ;*      ERROR      N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
6456
6457      $ERROR:
6458      043300      113737      001102      001414      MOVB      $TSTNM,TESTNO      ;SAVE TEST NUMBER FOR ERROR TYPE OUT
6459      043306      005237      001432      INC      ERRCNT      ;KEEP COUNT OF MULTIPLE ERRORS
6460      043312      010037      001162      MOV      R0,$REG0      ;SAVE R0
6461      043316      010137      001164      MOV      R1,$REG1      ;SAVE R1
6462      043322      010237      001166      MOV      R2,$REG2      ;SAVE R2
6463      043326      010337      001170      MOV      R3,$REG3      ;SAVE R3
6464      043332      010437      001172      MOV      R4,$REG4      ;SAVE R4
6465      043336      010537      001174      MOV      R5,$REG5      ;SAVE R5
6466      043342      105237      001103      7$:      INCB      $ERFLG      ;SET THE ERROR FLAG
6467      043346      001775      BEQ      7$      ;DON'T LET THE FLAG GO TO ZERO
6468      043350      013777      001102      135564      MOV      $TSTNM,$DISPLAY      ;DISPLAY TEST NUMBER AND ERROR FLAG
6469      043356      032777      002000      135554      BIT      $BIT10,$SWR      ;BELL ON ERROR?
6470      043364      001402      BEQ      1$      ;NO - SKIP
6471      043366      104401      001216      TYPE      $BELL      ;RING BELL
6472      043372      005237      001112      1$:      INC      $ERTTL      ;COUNT THE NUMBER OF ERRORS
6473      043376      011637      001116      MOV      (SP),$ERRPC      ;GET ADDRESS OF ERROR INSTRUCTION
6474      043402      162737      000002      001116      SUB      #2,$ERRPC
6475      043410      117737      135502      001114      MOVB      $ERRPC,$ITEMB      ;STRIP AND SAVE THE ERROR ITEM CODE
6476      043416      032777      020000      135514      BIT      $BIT13,$SWR      ;SKIP TYPEOUT IF SET
6477      043424      001004      BNE      20$      ;SKIP TYPEOUTS
6478      043426      004737      043576      JSR      PC,ERTYPE      ;GO TO USER ERROR ROUTINE
6479      043432      104401      001223      TYPE      ,$CRLF
6480      043436      127737      000001      001246      20$:      CMPB      $APTENV,$ENV      ;RUNNING IN APT MODE
6481      043444      001007      BNE      2$      ;NO SKIP APT ERROR REPORT
6482      043446      113737      001114      043460      MOVB      $ITEMB,21$      ;SET ITEM NUMBER AS ERROR NUMBER
6483      043454      004737      041436      JSR      PC,$ATY4      ;REPORT FATAL ERROR TO APT
6484      043460      000      21$:      .BYTE      0
6485      043461      000      .BYTE      0
6486      043462      000777      22$:      BR      22$      ;APT ERROR LOOP
6487      043464      005777      135450      2$:      TST      $SWR      ;HALT ON ERROR
6488      043470      100001      BPL      3$      ;SKIP IF CONTINUE
6489      043472      000000      HALT      ;HALT ON ERROR!
6490      043474      032777      001000      135436      3$:      BIT      $BIT09,$SWR      ;LOOP ON ERROR SWITCH SET?
6491      043502      001402      BEQ      4$      ;BR IF NO
6492      043504      013716      001110      MOV      $LPERR,(SP)      ;FUDGE RETURN FOR LOOPING
6493      043510      005737      001214      4$:      TST      $ESCAPE      ;CHECK FOR AN ESCAPE ADDRESS
6494      043514      001402      BEQ      5$      ;BR IF NONE
6495      043516      013716      001214      MOV      $ESCAPE,(SP)      ;FUDGE RETURN ADDRESS FOR ESCAPE
6496      043522      022737      040762      000042      5$:      CMP      $SENDAD,$#42      ;ACT-11 AUTO-ACCEPT?
6497      043530      001001      BNE      6$      ;BRANCH IF NO
6498      043532      000000      HALT      ;YES
6499      043534      032737      001000      001140      6$:      BIT      $SW9,$SWR      ;IS THE LOOP ON ERROR SWITCH UP?
6500      043542      001414      BEQ      EREXIT      ;BRANCH IF NOT LOOPING ON ERROR
6501      043544      042737      177776      177572      BIC      $177776,$MMRO      ;CLEAR MEMORY MANAGEMENT REG 0

```

G10

MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 123
 DQKTA.P11 07-FEB-77 10:30 ERROR HANDLER ROUTINE

```

6505                                     ;LEAVE BIT0 UNCHANGED
6506 043552 012737 177777 015164      MOV    #-1,CPFLAG      ;RE-INITIALIZE CP TRAP FLAG
6507 043560 012737 177777 015254      MOV    #-1,PAFLAG      ;RE-INITIALIZE PARITY TRAP FLAG
6508 043566 012737 177777 015434      MOV    #-1,MMFLAG      ;RE-INITIALIZE MEMORY MANAGE. TRAP FLAG
6509 043574 000002                      EREXIT: RTI           ;RETURN TO TEST AFTER ERROR
6510
6511
6512                                     .SBTTL  ERROR MESSAGE TYPE OUT ROUTINE
6513
6514
6515                                     ;*
6516                                     ;* THIS SUBROUTINE IS CALLED BY THE ERROR HANDLER TO TYPE
6517                                     ;* THE ERROR MESSAGES. IT PICKS UP THE ITEM BYTE ($ITEMB) NUMBER
6518                                     ;* AND USES THAT TO INDEX THROUGH THE ERROR TABLE. THE ERROR
6519                                     ;* TABLE STARTS AT "SERRTB" AND HAS FOUR (4) POINTERS FOR EACH
6520                                     ;* ENTRY 'EM' 'DH' 'DT' 'DF'. THE 'EM' POINTS TO THE ERROR
6521                                     ;* MESSAGE WHICH IS AN ASCIZ STRING. THE 'DH' POINTS TO THE DATA
6522                                     ;* HEADER WHICH IS ANOTHER ASCIZ STRING. THE 'DT' POINTS TO THE
6523                                     ;* DATA TABLE WHICH IS A GROUP OF WORDS CONTAINING THE ADDRESSES
6524                                     ;* OF THE DATA TO BY TYPED. THE FORMAT OF THIS DATA IS
6525                                     ;* CONTROLLED BY THE 'DF' WHICH IS THE POINTER TO THE DATA FORMAT.
6526                                     ;* THE DATA FORMAT IS A GROUP OF BYTES WHICH CONTAIN NUMBERS
6527                                     ;* THAT CORRESPOND TO DIFFERENT TYPING FORMATS.
6528                                     ;*
6529                                     ;* 0 -16 BIT OCTAL FORMAT
6530                                     ;* 1 -DECIMAL FORMAT
6531                                     ;* 2 -18 BIT OCTAL FORMAT. DATA IS A P.A.R. AND THE
6532                                     ;* OUTPUT IS THE BASE ADDRESS THAT THE P.A.R. POINTS TO.
6533
6534 043576 010046                      ERTYPE: MOV    RO,-(KSP)      ;SAVE RO ON STACK
6535 043600 005000                      CLR     RO              ;CLEAR RO
6536 043602 113700 001114              MOV     2($ITEMB,RO)      ;PUT ITEM NUMBER IN RO
6537 043606 001004                      BNE     1$              ;BRANCH IF IT IS NON-ZERO
6538 043610 013746 001116              MOV     $ERRPC,-(KSP)      ;PUT ERROR PC ON STACK FOR TYPING
6539 043614 104402                      TYPOC   ;TYPE FAILING PC
6540 043616 000534                      BR      13$              ;GO TO RETURN
6541 043620 104401 001223              1$:   TYPE   $CRLF        ;TYPE CRLF
6542 043624 005300                      DEC     RO              ;ADJUST ITEM NUMBER TO GET A POINTER
6543 043626 006300                      ASL     RO              ;LEFT SHIFT ITEM NO. 3 PLACES
6544 043630 006300                      ASL     RO
6545 043632 006300                      ASL     RO
6546 043634 100024                      BPL     22$              ;BRANCH IF ITEM #LESS THAN 200
6547 043636 032700 001000              BIT     $BIT9,RO        ;SEE IF ITEM # WAS 3XX
6548 043642 001415                      BEQ     21$              ;BRANCH IF ITEM # WAS 2XX
6549                                     ;AND TYPE ERROR MESSAGE
6550 043644 032737 000200 001140        BIT     $SW7,$SWR        ;SEE IF SWITCH 7 IS UP
6551 043652 001404                      BEQ     20$              ;BRANCH IF SWITCH IS NOT UP
6552                                     ;AND TYPE DATA, ON MULTIPLE ERRORS
6553 043654 062766 000004 000002        ADD     #4,2(KSP)        ;SKIP 'TYPE $CRLF' IF SW 7 IS UP
6554                                     ;INHIBIT MULTIPLE ERROR TYPEOUTS
6555 043662 000512                      BR      13$              ;BRANCH TO EXIT
6556                                     ;
6557 043664                      20$:   BIC     #177000,RO        ;CLEAR UPPER BYTE OF RO
6558 043670 062700 002474              ADD     $ER200+4,RO        ;POINT TO DATA TABLE ENTRY
6559 043674 000424                      BR      5$              ;GO TYPE DATA TABLE
6560 043676 042700 177000              21$:  BIC     #177000,RO        ;CLEAR UPPER BYTE OF RO
6561 043702 062700 000770              ADD     #($ER200-$ERRTB),RO ;ADD DIFFERENCE BETWEEN
6562                                     ;ITEM 1 AND ITEM 201
6563                                     ;
6564                                     ; GET POINTER TO ERROR MESSAGE AND TYPE IT

```


H10

MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 124
 DQKTA.P11 07-FEB-77 10:30 ERROR MESSAGE TYPE OUT ROUTINE

```

6561      IF THE POINTER IS NOT ZERO
6562 043706 062700 001500      22$: ADD    #ERRTB,RO      ;ADD BASE OF ERROR TABLE
6563 043712 012037 043722      MOV    (RO)+,2$      ;PUT MESSAGE POINTER IN TYPE STATEMENT
6564 043716 001404              BEQ    3$              ;BRANCH IF NO ERROR MESSAGE
6565 043720 104401              TYPE   TYPE ERROR MESSAGE
6566 043722 000000      2$: .WORD 0      ;POINTER TO ERROR MESSAGE
6567 043724 104401 001220      TYPE   ,SCRLF      ;TYPE CRLF
6568      GET THE POINTER TO THE DATA HEADER AND
6569      TYPE IT IF THE POINTER IS NOT ZERO
6570 043730 012037 043740      3$: MOV    (RO)+,4$      ;PUT HEADER POINTER IN TYPE STATEMENT
6571 043734 001404              BEQ    5$              ;BRANCH IF NO DATA HEADER
6572 043736 104401              TYPE   TYPE THE DATA HEADER
6573 043740 000000      4$: .WORD 0      ;POINTER TO DATA HEADER
6574 043742 104401 001223      TYPE   ,SCRLF      ;TYPE CRLF
6575      THIS IS THE START OF THE DATA OUTPUT IF THE
6576      DATA POINTER IS NOT ZERO. RO POINTS TO THE
6577      DATA FORMAT. R1 POINTS TO THE ADDRESS OF
6578      THE DATA WORDS.
6579 043746 010146      5$: MOV    R1,-(KSP)      ;SAVE R1 ON THE STACK
6580 043750 012001      MOV    (RO)+,R1      ;PUT DATA TABLE POINTER IN R1
6581 043752 001455      BEQ    12$      ;BRANCH IF NO DATA TABLE
6582 043754 012000      MOV    (RO)+,RO      ;PICK UP DATA FORMAT POINTER
6583 043756 105710      6$: TSTB  (RO)      ;IS THIS WORD OCTAL
6584 043760 001003      BNE    7$      ;BRANCH IF NOT 16-BIT OCTAL
6585      THIS IS 16 BIT OCTAL FORMAT (DF = 0)
6586 043762 013146      MOV    2(R1)+,-(KSP)      ;PUT WORD ON STACK FOR TYPING
6587 043764 104402      TPOC      ;TYPE THE WORD ON STACK AS 16 BIT OCTAL
6588 043766 000441      BR    11$      ;GET READY FOR NEXT WORD
6589 043770 122710 000001      7$: CMPB  #1,(RO)      ;IS THE WORD DECIMAL
6590 043774 001003      BNE    10$      ;BRANCH IF NOT DECIMAL
6591      THIS IS DECIMAL FORMAT (DF = 1)
6592 043776 013146      MOV    2(R1)+,-(KSP)      ;PUT WORD ON STACK FOR TYPING
6593 044000 104405      TYPDS      ;TYPE THE WORD ON STACK AS DECIMAL
6594 044002 000433      BR    11$      ;GET READY FOR NEXT WORD
6595      THIS IS FORMAT 2. DATA WORD IS A P.A.R.
6596      OUTPUT WILL BE 18-BIT. PAR LEFT SHIFTED 2.
6597 044004 010246      10$: MOV    R2,-(KSP)      ;SAVE R2 ON THE STACK
6598 044006 010346      MOV    R3,-(KSP)      ;SAVE R3 ON THE STACK
6599 044010 013103      MOV    2(R1)+,R3      ;LOAD DATA WORD INTO R3
6600 044012 005002      CLR    R2      ;R2 WILL HOLD UPPER SIX BITS OF NUMBER
6601 044014 012705 000006      MOV    #6,R5      ;LEFT SHIFT (R2:R3) 6 PLACES
6602 044020 000241      23$: CLC      ;
6603 044022 006303      ASL    R3      ;
6604 044024 006102      ROL    R2      ;
6605 044026 077504      SOB    R5,23$      ;
6606 044030 010237 001374      MOV    R2,PADRSB      ;STORE UPPER BITS OF ADDRESS
6607 044034 010337 001372      MOV    R3,PADRSL      ;STORE LOWER 16 BITS OF ADDRESS
6608 044040 012746 001372      MOV    #PADRSL,-(KSP)      ;PUT ADDRESS OF LOWER 16 BITS ON STACK
6609 044044 004737 042664      JSR    PC,$DB20      ;CONVERT TWO WORDS TO ASCIZ
6610 044050 062716 000005      ADD    #5,(KSP)      ;I ONLY WANT 6 DIGITS
6611 044054 012637 044062      MOV    (KSP)+,31$      ;PUT POINTER AFTER TYPE CALL
6612 044060 104401      TYPE   ;POINTER TO ASCIZ STRING FOLLOWS
6613 044062 000000      31$: .WORD 0      ;POINTER TO ASCIZ STRING
6614 044064 012603      MOV    (KSP)+,R3      ;RESTORE R3 FROM STACK
6615 044066 012602      MOV    (KSP)+,R2      ;RESTORE R2 FROM STACK
6616 044070 000400      BR    11$      ;GET READY FOR NEXT WORD

```


6617	044072	005200			115:	INC	RO		;POINT TO NEXT FORMAT BYTE
6618	044074	104401	044114			TYPE	325		;TYPE TWO SPACES
6619	044100	005711				TST	(R1)		;IS THERE ANOTHER WORD?
6620	044102	001401				BEQ	125		;BRANCH IF ALL DONE
6621	044104	000724				BR	65		;GO BACK FOR NEXT NUMBER
6622	044106	012601			125:	MOV	(KSP)+,R1		;RESTORE R1
6623	044110	012600			135:	MOV	(KSP)+,RO		;RESTORE RO
6624	044112	000207				RTS	PC		;RETURN TO ERROR ROUTINE
6625	044114	020040	000		325:	.ASCIZ	' '		;TWO SPACES
6626		044120				.EVEN			
6627									
6628									
6629									
6630									
6631									
6632									
6633									
6634									
6635									
6636									
6637									
6638									
6639									
6640									
6641									
6642									
6643									
6644									
6645									
6646									
6647	044120								
6648	044120	032766	000020	000002					
6649	044126	001406							
6650	044130	016637	000002	001442					
6651	044136	042766	000020	000002					
6652	044144	000006							
6653									
6654									
6655									
6656									
6657									
6658									
6659									
6660									
6661									
6662									
6663	044146								
6664	044146	013766	001442	000002					
6665	044154	042737	000020	001442					
6666									
6667	044162	000006							
6668									
6669									
6670									
6671									
6672									

.SBTTL ***** SUBROUTINES UNIQUE TO THIS PROGRAM *****

 .SBTTL TURN OFF AND SAVE THE T-BIT
 ;*
 ;* THIS SUBROUTINE IS REACHED BY THE TRAP CALL "TBITO". IT IS
 ;* USED TO TURN OFF THE T-BIT IF IT IS ON. THE PROCESSOR STATUS
 ;* IS SAVED IN "OLDPSW" SO THAT THE T-BIT CAN BE RESTORED TO ITS
 ;* PREVIOUS STATUS WHEN CONDITIONS WARRANT
 ;*
 .EQUIV BIT4,TBIT
 TBITOFF:
 BIT #TBIT,2(KSP) ;IS THE T-BIT ON?
 BEQ 15 ;BRANCH IF ITS NOT
 MOV 2(KSP),OLDPSW ;SAVE OLD PSW FOR RESTORING LATER
 BIC #TBIT,2(KSP) ;CLEAR THE T-BIT
 15: RTT ;RETURN TO THE PROGRAM

 .SBTTL RESTORE T-BIT TO ITS PREVIOUS CONDITION
 ;*
 ;* THIS SUBROUTINE IS REACHED BY THE TRAP CALL "TBITR". IT IS
 ;* USED TO RESTORE THE T-BIT AFTER A PARTICULAR TEST THAT
 ;* CANNOT BE RUN WITH THE T-BIT ON. IT USES THE PROCESSOR
 ;* STATUS STORED IN "OLDPSW" BY "TBITO", REPLACES THE PS ON THE
 ;* STACK WITH IT AND DOES AN "RTT".
 ;*
 TBITRESTORE:
 MOV OLDPSW,2(KSP) ;PUT OLD PSW ON THE STACK
 BIC #TBIT,OLDPSW ;CLEAR T-BIT IN "OLDPSW" SO THAT IT
 ;* WON'T BE TURNED ON BY ACCIDENT
 RTT ;RETURN TO PROGRAM AND INHIBIT T-BIT
 ;TRAPPING AFTER THIS INSTRUCTION

 .SBTTL CLEAR 8 PARS OR PDRS STARTING FROM ADDRESS IN R5
 ;*****

J10

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 126
 DOKTAA.P11 07-FEB-77 10:30 CLEAR 8 PARS OR PDRS STARTING FROM ADDRESS IN R5

```

6673      ;*      SUBROUTINE CLRREG
6674      ;*
6675      ;*      THIS SUBROUTINE CLEARS 8 CONSECUTIVE MEMORY MANAGEMENT
6676      ;*      REGISTERS STARTING WITH THE REGISTER POINTED TO BY R5.
6677      ;*      IT SAVES R0 ON THE STACK AND LOADS A COUNT IN R0.
6678      ;*      CLEARS THE PAR'S OR PDR'S, AND THEN RESTORES R0 BEFORE
6679      ;*      RETURNING.
6680      ;*      THE CALLING SEQUENCE IS:
6681      ;*      MOV      #KIPAR,R5      ;PUT ADDRESS OF FIRST KERNEL PAR IN R5
6682      ;*      JSR      PC,CLRREG      ;GO CLEAR ALL KERNEL PAR'S
6683      ;*
6684      ;*      *****
6685
6686      CLRREG: MOV      R0,-(KSP)      ;SAVE R0 ON STACK
6687      MOV      #10,R0      ;PUT COUNT IN R0
6688      1$: CLR      (R5)+      ;CLEAR PAR ORPDR POINTED TO BY R5
6689      SOB      R0,1$      ;BRANCH BACK 7 DECIMAL TIMES.
6690      MOV      (KSP)+,R0      ;RESTORE R0 FROM STACK
6691      RTS      PC      ;RETURN TO TEST
6692
6693      .SBTTL  CLEANUP LOCATIONS THAT HOLD LOGICAL 'AND' AND 'OR'
6694      ;*      *****
6695      ;*      SUBROUTINE CLEANUP
6696      ;*
6697      ;*      THIS SUBROUTINE IS USED TO INITIALIZE ALL THE LOCATIONS THAT
6698      ;*      HOLD THE "LOGICAL AND" AND "LOGICAL OR" OF THE DATA AND ADDRESSES
6699      ;*      THAT FAILED DURING THE EXECUTION OF A TEST.
6700      ;*
6701      ;*      *****
6702
6703      CLEANUP:      ;STARTING ADDRESS OF SUBROUTINE.
6704      CLR      DATACR      ;LOCATION FOR LOGICAL OR OF BAD DATA
6705      CLR      ADDRJR      ;LOCATION FOR LOGICAL OR OF ADDRESS
6706      CLR      PATTOR      ;LOCATION FOR LOGICAL OR OF PATTERN LOADED
6707      MOV      #-1,R0      ;LOAD -1 INTO R0 TO INITIALIZE LOGICAL AND LOCS
6708      MOV      R0,DATAND      ;LOCATION FOR LOGICAL AND OF BAD DATA
6709      MOV      R0,DRAND      ;LOCATION FOR LOGICAL AND OF ADDRESS
6710      MOV      R0,PATAND      ;LOCATION FOR LOGICAL AND OF PATTERN LOADED
6711      RTS      PC      ;RETURN TO TEST
6712
6713      .SBTTL  P.A.R. OR P.D.R. ADDRESS TIMED OUT WHEN REFERENCED
6714      ;*      *****
6715      ;*
6716      ;*      THIS SUBROUTINE IS USED TO LOG AND REPORT THE FACT THAT A
6717      ;*      REFERENCE TO A P.A.R. OR A P.D.R. TIMED OUT ON THE UNIBUS. IT
6718      ;*      KEEPS A LOGICAL AND AND A LOGICAL OR OF EACH ADDRESS THAT TIMES
6719      ;*      OUT.
6720      ;*      *****
6721      TIMEOUT:      ;STARTING ADDRESS OF SUBROUTINE
6722      INC      (PC)+      ;INCREMENT ONE TIME GATE
6723      .WORD      -1      ;ONE TIME ENTRANCE FLAG
6724      BEQ      10$      ;BRANCH IF FLAG IS NOW ZERO
6725      HALT      ;I HAVE ENTERED THIS ROUTINE BEFORE
6726      ;I FINISHED REPORTING THE FIRST ERROR
6727      ;THE SECOND ENTRY ADDRESS IS ON THE
6728      ;STACK AND THE FIRST ERROR CONDITION

```

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 127
DOKTA.A.P11 07-FEB-77 10:30 P.A.R. OR P.D.R. ADDRESS TIMED OUT WHEN REFERENCED

```

6729 044250 012637 001436 10$: MOV (KSP)+,OLDPC ;IS PROBABLY STILL LOCKED UP .
6730 044254 012637 001440 10$: MOV (KSP)+,OLDPS ;SAVE RETURN ADDRESS
6731 044260 012706 001100 10$: MOV #KERSTK,KSP ;SAVE OLD PSW
6732 044264 050037 001426 15$: BIS RO,ADDROR ;RESTORE STACK POINTER
6733 044270 005100 001426 15$: COM RO ;PERFORM LOGICAL OR OF FAILING ADDRESS
6734 044272 040037 001430 15$: BIC RO,ADRAND ;GET RO READY FOR AND
6735 044276 005100 001430 15$: COM RO ;PERFORM LOGICAL AND
6736 044300 105737 001103 15$: TSTB $ERFLG ;PUT RO BACK AS IT WAS
6737 044304 001002 001103 15$: BNE 2$ ;IS THIS THE FIRST ERROR
6738 044306 104201 001103 15$: ERROR 201 ;BRANCH IF NOT FIRST ERROR
6739 044310 000401 001103 15$: BR 3$ ;NO REGISTER RESPONSE.
6740 044312 104301 001103 15$: ERROR 301 ;BRANCH TO EXIT
6741 044314 012737 177777 044242 25$: MOV #-1,TOFLAG ;CONTINUE NO RESPONSE TABLE
6742 044322 013746 001440 35$: MOV OLDPS,-(KSP) ;RESET ONE TIME GATE
6743 044326 013746 001436 35$: MOV OLDPC,-(KSP) ;RESTORE OLD PSW
6744 044332 000006 001436 35$: RTT ;PUSH RETURN ADDRESS BACK ON THE STACK
6745 044334 000006 001436 35$: ;RETURN TO THE TEST
6746
6747 .SBTTL DUAL ADDRESSING WHEN LOADING A P.A.R. OR P.D.R.
6748 *****
6749 *
6750 * THIS SUBROUTINE WILL LOG AND REPORT ALL DUAL ADDRESSING ERRORS
6751 * FOUND IN BOTH P.A.R.'S AND P.D.R.'S. A "LOGICAL OR" AND A
6752 * "LOGICAL AND" OF THE WRITTEN ADDRESSES WILL BE MAINTAINED IN
6753 * 'ADDROR' AND 'ADRAND'. THE LOG ON THE ADDITIONAL OR FAILING,
6754 * ADDRESSES WILL BE MAINTAINED IN 'DATAOR' AND 'DATAND'.
6755 *
6756 *****
6757 DUALADR: MOV (KSP)+,OLDPC ;STARTING LOCATION OF SUBROUTINE
6758 044334 012637 001436 10$: BIS RO,ADDROR ;SAVE RETURN ADDRESS IN CASE OF LOOP
6759 044340 050037 001426 10$: COM RO ;LOGICAL OR OF WRITTEN ADDRESS
6760 044344 005100 001426 10$: BIC RO,ADRAND ;GET RO READY FOR AND
6761 044346 040037 001430 10$: COM RO ;PERFORM LOGICAL AND
6762 044352 005100 001430 10$: BIS R1,DATAOR ;PUT RO BACK AS IT WAS
6763 044354 050137 001416 10$: COM R1 ;LOGICAL OR OF DUALED ADDRESS
6764 044360 005101 001416 10$: BIC R1,DATAND ;GET R1 READY FOR AND
6765 044362 040137 001420 10$: COM R1 ;PERFORM LOGICAL AND
6766 044366 005101 001420 10$: TSTB $ERFLG ;PUT R1 BACK AS IT WAS
6767 044370 105737 001103 15$: BNE 1$ ;SEE IF THIS IS FIRST ERROR
6768 044374 001002 001103 15$: ERROR 202 ;BRANCH IF NOT FIRST ERROR
6769 044376 104202 001103 15$: BR 2$ ;BRANCH TO EXIT
6770 044400 000401 001103 15$: ERROR 302 ;BRANCH TO EXIT
6771 044402 104302 001103 15$: JMP 3OLDPC ;RETURN TO TEST
6772 044404 000177 135026 25$:
6773
6774 .SBTTL COUNT PATTERN ERRORS IN P.A.R.'S OR P.D.R.'S
6775 *****
6776 *
6777 * THIS SUBROUTINE IS USED TO LOG AND REPORT THE COUNT PATTERN
6778

```

L10

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 128
DOKTAA.P11 07-FEB-77 10:30 COUNT PATTERN ERRORS IN P.A.R.'S OR P.D.R.'S

6779
6780
6781
6782
6783
6784
6785
6786

:* ERRORS OCCURRING WHEN TESTING THE P.A.R.'S AND P.D.R.'S.
:* THE "LOGICAL OR" AND "LOGICAL AND" OF VARIOUS DATA WILL BE
:* MAINTAINED AS FOLLOWS:
:* ADDRESSES OF FAILING REGISTERS IN 'ADDROR' AND 'ADRAND'
:* DATA FETCHED FROM REGISTERS IN 'DATAOR' AND 'DATAND'
:* PATTERN LOADED INTO THE REGISTERS IN 'PATTOR' AND 'PATAND'.
:* *****

M10

MAINDEC-11-DQKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 129
 DQKTA.P11 07-FEB-77 10:30 COUNT PATTERN ERRORS IN P.A.R.'S OR P.D.R.'S

6787	044410			PARCOUNT:	MOV	(KSP)+,OLDPC	;STARTING ADDRESS OF THIS SUBROUTINE
6788	044410	012637	001436		BIS	R0,ADDROR	;SAVE RETURN ADDRESS IN CASE OF LOOP
6789	044414	050037	001426		COM	R0	;LOGICAL OR OF FAILING ADDRESS
6790	044420	005100			BIC	R0,ADRAND	;GET R0 READY FOR AND
6791	044422	040037	001430		COM	R0	;PERFORM LOGICAL AND
6792	044426	005100			BIS	R1,PATTOR	;PUT R0 BACK AS IT WAS
6793	044430	050137	001422		COM	R1	;LOGICAL OR OF PATTERN LOADED
6794	044434	005101			BIC	R1,PATAND	;GET R1 READY FOR AND
6795	044436	040137	001424		COM	R1	;PERFORM LOGICAL AND
6796	044442	005101			BIS	R2,DATAOR	;PUT R1 BACK AS IT WAS
6797	044444	050237	001416		COM	R2	;LOGICAL OR OF DATA FETCHED
6798	044450	005102			BIC	R2,DATAND	;GET R2 READY FOR AND
6799	044452	040237	001420		COM	R2	;PERFORM LOGICAL AND
6800	044456	005102			TSTB	\$ERFLG	;PUT R2 BACK AS IT WAS
6801	044460	105737	001103		BNE	1\$	;SEE IF THIS IS THE FIRST ERROR
6802	044464	001002			ERROR	203	;BRANCH IF NOT FIRST ERROR
6803	044466	104203			BR	2\$	;BRANCH TO EXIT
6804	044470	000401		1\$:	ERROR	303	
6805	044472	104303		2\$:	JMP	2OLDPC	;RETURN TO TEST
6806	044474	000177	134736				
6807							
6808							
6809							
6810							
6811							
6812		000001		.END			

N10

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 131
 DOKTAA.P11 07-FEB-77 10:30 CROSS REFERENCE TABLE -- USER SYMBOLS

ABASE = 000000	738	779																		
ACDM1 = 000000	738	781																		
ACDM2 = 000000	738	782																		
ACPUOP = 000000	738	753																		
ADDROR 001426	832#	2195	2199	2202	6705*	6733*	6759*	6789*												
ADDMD = 000000	738	783																		
ADDH1 = 000000	738	784																		
ADDH10 = 000000	738	793																		
ADDH11 = 000000	738	794																		
ADDH12 = 000000	738	795																		
ADDH13 = 000000	738	796																		
ADDH14 = 000000	738	797																		
ADDH15 = 000000	738	798																		
ADDH2 = 000000	738	785																		
ADDH3 = 000000	738	786																		
ADDH4 = 000000	738	787																		
ADDH5 = 000000	738	788																		
ADDH6 = 000000	738	789																		
ADDH7 = 000000	738	790																		
ADDH8 = 000000	738	791																		
ADDH9 = 000000	738	792																		
ADEVCT = 000000	738	744																		
ADEVH = 000000	738	780																		
ADRAND 001430	833#	2195	2199	2202	6709*	6735*	6761*	6791*												
AENV = 000000	738	749																		
AENVH = 000000	738	750																		
AFATAL = 000000	738	741																		
AMADR1 = 000000	738	766																		
AMADR2 = 000000	738	770																		
AMADR3 = 000000	738	773																		
AMADR4 = 000000	738	776																		
AMAMS1 = 000000	738	760																		
AMAMS2 = 000000	738	768																		
AMAMS3 = 000000	738	771																		
AMAMS4 = 000000	738	774																		
AMSGAD = 000000	738	746																		
AMTGLC = 000000	738	747																		
AMSGTY = 000000	738	740																		
AMTYP1 = 000000	738	761																		
AMTYP2 = 000000	738	769																		
AMTYP3 = 000000	738	772																		
AMTYP4 = 000000	738	775																		
APASS = 000000	738	743																		
APRIOR = 000000	738																			
APTCSU = 000040	5985	6090#																		
APTENV = 000001	5978	6046	6088#	6481																
APTSIZ = 000200	2630	6087#																		
APTSP0 = 000100	5980	6048	6089#																	
ASWREG = 000000	738	751																		
ATESTN = 000000	738	742																		
AUNIT = 000000	738	745																		
AUSMR = 000000	738	752																		
AVECT1 = 000000	738	777																		
AVECT2 = 000000	738	778																		
BAOPC 001412	826#	2180	2181	2183	2185	2187	2395*	2434*	2479*											
BIT0 = 000001	525#	4123	4375	4475	4478	4556	4881	4997	5128	5196	5250	5318	5366							

[illegible]

C11

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 133
 DOKTAA.P11 07-FEB-77 10:30 CROSS REFERENCE TABLE -- USER SYMBOLS

DF23	014770	1006	1018	1024	2307#				
DF27	014775	1030	2309#						
DF3	014665	906	913	2279#					
DF30	015000	1037	2310#						
DF31	015005	1044	2312#						
DF32	015014	1050	2315#						
DF33	015020	1057	2317#						
DF34	015027	1063	2320#						
DF35	015032	1069	1075	2321#					
DF37	015036	1081	1087	1147	1219	2323#			
DF41	015043	1093	1099	1105	1111	1117	1123	2325#	
DF47	015047	1134	2327#						
DF5	014672	920	2281#						
DF50	015056	1141	2330#						
DF52	015064	1153	2332#						
DF53	015070	1159	2334#						
DF54	015074	1165	2336#						
DF55	015100	1171	1177	1183	1189	1195	2338#		
DF6	014676	927	2283#						
DF62	015103	1201	2339#						
DF63	015107	1207	2341#						
DF64	015113	1213	2343#						
DF66	015117	1225	1243	2345#					
DF67	015123	1231	2347#						
DF7	014703	933	939	2285#					
DF70	015127	1237	2349#						
DF72	015132	1249	1255	2350#					
DF74	015135	1261	2351#						
DF75	015137	1267	1273	2352#					
DH1	007755	891	1277	1809#					
DH11	010362	943	1856#						
DH12	010422	949	1862#						
DH13	010462	955	1868#						
DH14	010533	961	1875#						
DH15	010573	967	1881#						
DH16	010722	974	1897#						
DH17	011031	980	1909#						
DH2	010000	897	1813#						
DH20	011101	986	1916#						
DH201	013540	1287	2156#						
DH202	013567	1295	2160#						
DH203	013647	1304	2169#						
DH21	011131	992	1010	1921#					
DH22	011171	998	1927#						
DH23	011240	1004	1016	1022	1934#				
DH27	011310	1028	1941#						
DH3	010034	903	910	1818#					
DH30	011340	1034	1945#						
DH31	011440	1041	1956#						
DH32	011574	1048	1973#						
DH33	011634	1054	1979#						
DH34	011754	1061	1993#						
DH35	012004	1067	1073	1997#					
DH37	012044	1079	1145	1217	2003#				
DH40	012114	1085	2010#						
DH41	012164	1091	1097	1103	1109	1115	1121	2017#	

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 134
DOKTAA.P11 07-FEB-77 10:30 CROSS REFERENCE TABLE -- USER SYMBOLS

DH7	012224	1128	2023#														
DH5	010142	917	1831#														
DH50	012467	1138	2053#														
DH52	012600	1151	2066#														
DH53	012640	1157	2072#														
DH54	012700	1163	2078#														
DH55	012740	1169	1175	1181	2084#												
DH6	010226	924	1840#														
DH60	012770	1187	1193	2088#													
DH62	013020	1199	2092#														
DH63	013060	1205	2098#														
DH64	013120	1211	2104#														
DH66	013160	1223	2110#														
DH67	013220	1229	2116#														
DH7	010332	931	937	1852#													
DH70	013260	1235	2122#														
DH71	013310	1241	2126#														
DH72	013350	1247	2132#														
DH73	013421	1253	2139#														
DH74	013451	1259	2144#														
DH75	013470	1265	1271	2147#													
DISPLA	001142	705#	2618#	2626*	2703*	2887*	3029*	3786*	4112*	4986*	5355*	6439*	6468*				
DISPRE	000174	628#	2626														
DSMR =	177570	446#	704	2617													
DT1	013726	892	1278	2180#													
DT11	014016	944	2191#														
DT12	014030	950	2193#														
DT13	014042	956	2195#														
DT14	014056	962	2197#														
DT15	014070	969	2199#														
DT16	014110	975	2202#														
DT17	014134	981	2206#														
DT2	013734	898	2181#														
DT20	014150	987	2208#														
DT201	014620	1290	2268#														
DT202	014630	1299	2270#														
DT203	014642	1307	2272#														
DT21	014160	993	1011	2210#													
DT22	014172	999	2212#														
DT23	014206	1005	1017	1023	2214#												
DT27	014222	1029	2216#			</											

E11

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 135
 DOKTAA.P11 07-FEB-77 10:30 CROSS REFERENCE TABLE -- USER SYMBOLS

DT55	014464	1170	1176	1182	1188	1194	2248#
DT6	013772	926	2187#				
DT62	014474	1200	2250#				
DT63	014506	1206	2252#				
DT64	014520	1212	2254#				
DT66	014532	1224	1242	2256#			
DT67	014544	1230	2258#				
DT7	014006	932	938	2189#			
DT70	014556	1236	2260#				
DT72	014566	1248	1254	2262#			
DT74	014576	1260	2264#				
DT75	014604	1266	1272	2265#			
DUALAD	044334	3195	3256	3316	3377	6757#	
EMTVEC=	000030	535#	2587#	2588#			
EM1	002520	890	896	1312#			
EM10	003261	936	1377#				
EM11	003312	942	1382#				
EM12	003354	948	1388#				
EM13	003410	954	1393#				
EM14	003456	960	1400#				
EM15	003517	966	1406#				
EM16	003561	973	1412#				
EM17	003623	979	1418#				
EM20	003667	985	1425#				
EM201	007520	1286	1779#				
EM202	007575	1294	1787#				
EM203	007666	1303	1797#				
EM21	003726	991	1431#				
EM22	003773	997	1438#				
EM23	004047	1003	1446#				
EM24	004110	1009	1452#				
EM25	004145	1015	1457#				
EM26	004217	1021	1465#				
EM27	004303	1027	1474#				
EM3	002601	902	1321#				
EM30	004363	1033	1483#				
EM31	004432	1040	1490#				
EM32	004511	1047	1498#				
EM33	004572	1053	1507#				
EM34	004643	1060	1514#				
EM35	004712	1066	1521#				
EM36	004765	1072	1529#				
EM37	005042	1078	1537#				
EM4	002717	909	1276	1335#			
EM40	005133	1084	1547#				
EM41	005220	1090	1556#				
EM42	005271	1096	1563#				
EM43	005341	1102	1570#				
EM44	005410	1108	1577#				
EM45	005455	1114	1584#				
EM46	005524	1120	1591#				
EM47	005575	1126	1598#				
EM5	003042	916	1351#				
EM50	005661	1137	1608#				
EM51	005763	1144	1620#				
EM52	006027	1150	1627#				

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 136
DOKTAA.P11 07-FEB-77 10:30 CROSS REFERENCE TABLE -- USER SYMBOLS

EM53	006102	1156	1635#											
EM54	006153	1162	1642#											
EM55	006220	1168	1649#											
EM56	006261	1174	1655#											
EM57	006342	1180	1664#											
EM6	003115	923	1359#											
EM60	006411	1186	1671#											
EM61	006445	1192	1676#											
EM62	006504	1198	1682#											
EM63	006537	1204	1687#											
EM64	006567	1210	1692#											
EM65	006635	1216	1699#											
EM66	006716	1222	1708#											
EM67	006764	1228	1715#											
EM7	003205	930	1369#											
EM70	007031	1234	1722#											
EM71	007106	1240	1730#											
EM72	007151	1246	1736#											
EM73	007233	1252	1745#											
EM74	007307	1258	1753#											
EM75	007377	1264	1763#											
EM76	007451	1270	1771#											
ENTPT2	021510	866	2884#											
ENTPT3	022072	867	3026#											
ENTPT4	025066	868	3783#											
ENTPT5	027204	869	4109#											
ENTPT6	033674	870	4983#											
ENTPT7	036000	871	5352#											
EREXIT	043574	6503	6509#											
ERRCNT	001432	834#	3040	3042	3072	3074	3103	3105	3135	3137	3204	3206	3265	3267
		3325	3327	3386	3388	3439	3441	3484	3486	3526	3528	3569	3571	6392#
		6459#												
ERRVEC=	000004	528#	2615	2616#	2627#	2661#	2662#	2706#	2722#	3031#	3039#	3063#	3071#	3094#
		3102#	3126#	3134#	4011#	4014#	4070#	4343#	4346#	4392#	4396#	4553#	4654#	4667#
		4765#	4768#	6398	6399#	6401#	6404#							
ERTYPE	043576	6478	6531#											
ER200	002470	1283#	6555	6558										
FORTY	001450	846#	2396	2429	2567#	2572#	2650	2713	4012	4332	4458	4502	4692	4707
		4733	5024	6303										
FSTTST	001362	811#	2538#	2540#	2542#	2544#	2546#	2548#	2550#	2669				
GNS =	***** U	627	2642	5863	5870	6266	6267	6268	6269	6270	6273	6274	6275	6276
HITMIS	001356	808#	2433											
HOLFLG	001434	835#	4009#	4030#	4035	4039#								
HT =	000011	438#	599											

G11

MAINDEC-11-DQKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 137
 DQKTA.P11 07-FEB-77 10:30 CROSS REFERENCE TABLE -- USER SYMBOLS

KIPAR6= 172354	5086*													
KIPAR7= 172356	594*	4889*	5087*	5125*	5136*									
KIPDR0= 172300	595*	3036	3187	3201	3432	3804*	4122*	4880*	4996*	5088*	5126*	5195*	5365*	
	577*	3096	3290	3292	3305	3306	3319	3510	3665	3666*	3668	3673	3676	
	3787	3797	4113*	4871*	4987*	5102	5109	5117*	5121*	5134*	5141	5149*	5153*	
	5172*	5176*	5231	5239*	5263	5271*	5294*	5356*	5370					
KIPDR1= 172302	578*	4114*	4872*	4988*	5118*	5150*	5173*	5240*	5272*	5295*	5357*			
KIPDR2= 172304	579*	4115*	4873*	4989*	5119*	5151*	5174*	5241*	5273*	5296*	5358*			
KIPDR3= 172306	580*	4116*	4874*	4990*	5359*									
KIPDR4= 172310	581*	4124*	4126	4180*	4182	4235*	4318*	4328	4585*	4883*	4999*	5004	5010	
	5015*	5016	5035*	5384*	5390*	5507*	5530*	5532*	5542*	5544*	5555*	5557*	5569*	
	5571*	5583*	5585*	5598*	5600*	5614*	5616*	5663*						
KIPDR5= 172312	582*	4236*	4253	4268	4273	4275*	4315*	4884*						
KIPDR6= 172314	583*	4885*	5135*											
KIPDR7= 172316	584*	3099	3308	3322	3519	4117*	4875*	4991*	5023	5037	5120*	5122*	5130	
	5152*	5154*	5175*	5177*	5242*	5274*	5297*	5360*						
LF = 000012	439*	6028	6034											
LOOP = 020612	2648*	5905												
MED = 076600	609*	2568	4334	4348	4465	4476	4509	4710	4720	4726	6306			
MEMER 015252	2419*	2663												
MEMERR 001360	810*	2431												
MMEXP 001366	814*	2187	2484	2488	2660*	4139*	4142*	4195*	4198*	4245*	4249*	4260*	4264*	
	4285*	4295*	4319*	4323*	4325*	4788*								
MMFLAG 015434	2472*	2492*	2658*	6508*										
MMRO = 177572	600*	2482	2491*	2526	2654*	2888	2962	2965*	3812*	3827*	3842*	3857*	3872*	
	3887*	3902*	3917*	3932*	3947*	3968*	3982*	4021*	4123*	4375*	4556*	4600	4609	
	4620*	4705	4767*	4847*	4881*	4926	4942*	4997*	5045*	5196*	5318*	5366*	5482	
	5484*	5628	5630*	5759	5761*	6504*								
MMR1 = 177574	601*	2936												
MMR2 = 177576	602*	2483	2527	2963	2972	4601	4610	4706	4925	5483	5629	5760		
MMTRAP 015432	2471*	2665	4621	4828	4950	5042	5475	5622	5751					
MMVEC = 000250	544*	2665*	2666*	4587*	4588*	4593*	4621*	4668*	4828*	4835*	4836*	4906*	4907*	
	4950*	5001*	5042*	5389*	5475*	5511*	5622*	5671*	5751*					
NXTTST 001446	845*	2449	2704*	2754*	2814*	2845*	2882*	2933*	2960*	3024*	3060*	3091*	3123*	
	3165*	3226*	3286*	3347*	3417*	3462*	3504*	3547*	3593*	3628*	3661*	3696*	3735*	
	3776*	4007*	4106*	4177*	4233*	4310*	4372*	4579*	4650*	4786*	4867*	4979*	5075*	
	5220*	5348*	5505*	5653*	5777*	5814*								
OLDPC 001436	837*	2393*	2395	2404	2427*	2434	2447*	2449*	2453	2480*	2494	2513*	2520*	
	2529	5480*	5487	5626*	5633	5757*	5764	6730*	6744	6758*	6772	6788*	6806	
OLDPS 001440	839*	2394*	2403	2428*	2452	2481*	2493	2514*	2521*	2528	5481*	5486	5627*	
	5632	5758*	5763	6731*	6743									
OLDPSW 001442	841*	6650*	6664	6665*										
PAORSH 001374	818*	6606*												
PAORSL 001372	816*	6607*	6608											
PAFLAG 015254	2420*	2451*	2657*	6507*										
PARCOU 044410	3430	3475	3517	3560	6787*									
PARTAB 001452	857*	3741*	3747	3750										
PATAND 001424	831*	2202	6710*	6795*										
PATTOR 001422	830*	2202	6706*	6793*										
PCONTR 001400	821*	2183	2432*											
PCPUER 001410	825*	2181	2265	2400*	4026*	4028	4076*	4078*	4516*	4517				
PHITMI 001402	822*	2183	2433*											
PIRQ = 177772	445*													
PIRQVE= 000240	539*													
PMMRO 001404	823*	2185	2187	2236	2239	2242	2246	2258	2482*	2488	2526*	4128*	4143	
	4184*	4199	4600*	4614	4705*	4729	4829	4926*	4939	5482*	5628*	5759*		

MAINDEC-11-DOKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 138
DOKTAA.P11 07-FEB-77 10:30 CROSS REFERENCE TABLE -- USER SYMBOLS

PWR2	001406	824# 4753	2185 4925*	2187 4927	2236 5483*	2239 5629*	2242 5760*	2244	2258	2483*	2527*	4601*	4611	4706*
PPARER	001376	820#	2183	2431*	2437									
PRO =	000000	462#												
PR1 =	000040	463#												
PR2 =	000100	464#												
PR3 =	000140	465#												
PR4 =	000200	466#												
PRS =	000240	467#												
PR6 =	000300	468#												
PR7 =	000340	469#												
-S =	177776	442#	443											
PSM =	177776	443#	2512	2522	2671*	2673*	2708	2709	2847*	2849*	2851*	3991*	4671*	4735*
		4737*	4761*	4762*	4764*	4806*	4811*	4848*	5021	5022*	5101*	5198*	5223*	5320*
		5394*	5408*	5417*	5426*	5434*	5445*	5456*	5466*	5514*	5522*	5539*	5551*	5565*
		5579*	5593*	5608*	5662*	5673*	5686*	5695*	5704*	5712*	5722*	5733*	5743*	5752*
		5779*	5816*											
PWRMSG	042620	6319	6328#											
PWVEC=	000024	534#	2591*	2592*	6282*	6283*	6292*	6298*	6316*	6317*				
RMMRO	015612	2518	2526#											
RESREG=	104407	6274#	6362											
RESVEC=	000010	529#	2602*	2609*										
RETRY	001444	843#	2444	2446*	6391*									
RLFGIN=	000104	613#	4349	4353										
RLJAM =	000100	611#	2569	4335	4339	4462	4466	4506	4510					
RLSERV=	000101	612#	4473	4477										
RLHAM=	000105	614#	4711	4715										
RHAM =	000022	610#	4718	4721										
SAVREG=	104406	6273#	6347											
SRO =	177572	548#	600											
SR1 =	177574	549#	601											
SR2 =	177576	550#	602											
SR3 =	172516	551#												
STACK =	001100	433#	599	606	2583	4391	4439	4534	4539	4544	4555	4688	5817	5830
START	020066	2539	2541	2543	2545	2547	2549	2565*	6321					
STKEXP	001370	815#	4390*	4401*	4412*	4440	4451							
STKLMT=	177774	444#	2710	2711	4398*	4438*	4446*	4498*	4526*	4533*	4536*	4538*	4541*	4543*
		4546*	4557*											
STRTAB	001462	865#	2674											
STRT1	020000	631	2538#											
STRT2	020010	633	2540#											
STRT3	020020	635	2542#											
STRT4	020030	637	2544#											
STRT5	020040	639	2546#											

[illegible]

MAINDEC-11-DOKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 140
DOKTAA.P11 07-FEB-77 10:30 CROSS REFERENCE TABLE -- USER SYMBOLS

[illegible]

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 141
DOKTAA.P11 07-FEB-77 10:30 CROSS REFERENCE TABLE -- USER SYMBOLS

[illegible]

MAINDEC-11-DOKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 142
DOKTAA.P11 07-FEB-77 10:30 CROSS REFERENCE TABLE -- USER SYMBOLS

[illegible]

M11

MAINDEC-11-DOKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 143
 DOKTAA.P11 07-FEB-77 10:30 CROSS REFERENCE TABLE -- USER SYMBOLS

		2948	3010	3012	3046	3048	3077	3079	3109	3111	3149	3151	3210	3212
		3270	3272	3331	3333	3400	3402	3445	3447	3489	3491	3532	3534	3583
		3585	3618	3620	3651	3653	3686	3688	3720	3722	3765	3767	3994	3996
		4093	4095	4164	4166	4221	4223	4298	4300	4358	4360	4565	4567	4623
		4625	4771	4773	4853	4855	4967	4969	5059	5061	5204	5206	5334	5336
		5491	5493	5638	5640	5767	5769	5803	5805					
\$OCNT	042110	6123	6152	6165										
\$OCTVL	042766	6349	6374											
\$OMODE	042112	6118	6122	6127	6130	6141	6167							
\$OVER	043262	6394	6412	6420	6430	6439								
\$PASS	001234	743	2629	2648	5853	5854	5865	5906	6426	6443				
\$PASTH	000242	664												
\$PWRAD	042574	6321												
\$PWRON	042420	2591	6282	6316										
\$PWRMG	042570	6319												
\$PWRUP	042472	6292	6298											
\$QUES	001222	731	6034	6510										
\$ROCHR=	***** U	6273												
\$RODEC=	***** U	6273												
\$ROLIN=	***** U	6273												
\$RODOCT=	***** U	6273												
\$REGAD	001160	714												
\$REGO	001162	716	2197	2206	2208	2210	2212	2214	2216	2218	2220	2225	2252	2256
		2262	2268	2270	2272	6460*								
\$REG1	001164	717	2189	2191	2206	2210	2214	2218	2220	2228	2230	2236	2244	2250
		2254	2256	2260	2270	2272	6461*							
\$REG2	001166	718	2191	2193	2197	2206	2214	2218	2220	2225	2234	2236	2246	2254
		2265	2272	6462*										
\$REG3	001170	719	2193	2225	2230	2234	2236	2250	2252	2265	6463*			
\$REG4	001172	720	2272	6464*										
\$REG5	001174	721	6465*											
\$RESRE	041100	5943	6274											
\$RTNAD	041032	5905												
\$RTRN	041026	2599	2601	2606	5881	5900								
\$R2A =	***** U	6275												
\$SAVRE	041042	5927	6273											
\$SAVR6	042616	6291	6299	6300	6301	6327								
\$SCOPE	043004	2585	6390											
\$SETUP=	000037	621	2584	2585	2587	2589	2591	2593	2594	2595	2597	2611	2638	2641
		5851	6323	6391	6458	6491	6498							
\$STUP =	177777	621												
\$SVLAD	043226	6402	6433											
\$SVPC =	000250	671	676											
\$SMR =	177400	389	398	402	403	404	405	406	407	408	409	728	729	730
		2594	2595	2597	2611	2612	2704	2756	2816	2847	2884	2935	2962	3026
		3062	3093	3125	3167	3228	3288	3349	3419	3464	3506	3549	3595	3630
		3663	3698	3737	3778	4009	4108	4179	4235	4312	4374	4581	4652	4788
		4869	4981	5077	5222	5350	5507	5655	5779	5814	5845	5852	5879	5892
		5906	6322	6382	6383	6384	6385	6386	6393	6405	6407	6408	6413	6414
		6415	6422	6423	6424	6436	6439	6442	6449	6450	6451	6452	6453	6469
		6476	6488	6491	6510									
\$SWREG	001250	711	2632											
\$SWRMK=	000200	389	409	410	6386	6387	6409	6410						
\$TBIT	041034	2610	5896	5906	6323									
\$TESTN	001232	742	6434											
\$TIMES	001212	728	2594	3419	3464	3506	3549	4108	4179	4869	5852	6422	6429	6432

MAINDEC-11-DOKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 144
DOKTAA.P11 07-FEB-77 10:30 CROSS REFERENCE TABLE -- USER SYMBOLS

[illegible]

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 145
DOKTAA.P11 07-FEB-77 10:30 CROSS REFERENCE TABLE -- USER SYMBOLS

[illegible]

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 147
DOKTAA.P11 07-FEB-77 10:30 CROSS REFERENCE TABLE -- MACRO NAMES

[illegible]

MSG43	4771#	4773													
MSG44	4853#	4855													
MSG45	4967#	4969													
MSG46	5059#	5061													
MSG47	5204#	5206													
MSG5	2804#	2806													
MSG50	5334#	5336													
MSG51	5491#	5493													
MSG52	5638#	5640													
MSG53	5767#	5769													
MSG54	5803#	5805													
MSG6	2834#	2836													
MSG7	2868#	2870													
MSG7A	2922#	2924													
MULT	540#														
NEWST	540#	2686	2734	2804	2834	2868	2922	2946	3010	3046	3077	3109	3149	3210	3270
	3331	3400	3445	3489	3532	3583	3618	3651	3686	3720	3765	3994	4093	4164	4221
	4298	4358	4565	4623	4771	4853	4967	5059	5204	5334	5491	5638	5767	5803	
PAGEF	2364#	4124	4180												
PARMSG	3444#	3447													
PDRMSG	3531#	3534													
POP	540#	5948	6080	6081	6221	6309	6310								
PUSH	540#	5928	6041	6043	6064	6180	6284	6290							
READLG	2364#	4461	4472	4505											
REPORT	540#														
RESPMG	3045#	3048	3079	3111											
RESPON	2364#	3030	3062	3093	3125										
SAVTST	2364#	6458													
SCOPE	435#	2753	2813	2844	2881	2932	2959	3023	3059	3090	3122	3164	3225	3285	3346
	3416	3461	3503	3546	3592	3627	3660	3695	3734	3775	4006	4105	4176	4232	4309
	4371	4578	4649	4785	4866	4978	5074	5219	5347	5504	5652	5776	5813	5850	
SETACF	2364#	4312													
SETPRI	540#														
SETTRA	6258#	6267	6268	6269	6270	6273	6274	6275	6276						
SETUP	540#	2577													
SKIP	540#	2670	2723	2854	2989	3041	3073	3104	3136	3205	3266	3326	3387	3440	3485
	3527	3570	4072	4834	5043	5477	5623	5754							
SLASH	540#														
SPACE	540#														
SSCOPE	2364#	6391													
STARS	540#	647	649	656	669	679	734	737	2686	2699	2734	2751	2804	2811	2834
	2842	2868	2879	2922	2930	2946	2957	3010	3021	3046	3057	3077	3088	3109	3120
	3149	3162	3210	3223	3270	3283	3331	3344	3400	3414	3445	3459	3489	3501	3532
	3544	3583	3590	3618	3625	3651	3658	3686	3693	3720	3732	3765	3773	3994	4004
	4093	4103	4164	4174	4221	4230	4298	4307	4358	4369	4565	4576	4623	4647	4771
	4783	4853	4864	4967	4976	5059	5072	5204	5217	5334	5345	5491	5502	5638	5650
	5767	5774	5803	5812	5841	5912	5957	6036	6093	6170	6237	6280	6296	6338	6378
	6445														
SWRSU	540#	2613#													
TRACK	2364#	2752	2812	2843	2880	2931	2958	3022	3058	3089	3121	3163	3224	3284	3345
	3415	3460	3502	3545	3591	3626	3659	3694	3733	3774	4005	4104	4175	4231	4308
	4370	4577	4648	4784	4865	4977	5073	5218	5346	5503	5651	5775			
TRACK1	2364#	2753	2813	2844	2881	2932	2959	3023	3059	3090	3122	3164	3225	3285	3346
	3416	3461	3503	3546	3592	3627	3660	3695	3734	3775	4006	4105	4176	4232	4309
	4371	4578	4649	4785	4866	4978	5074	5219	5347	5504	5652	5776			
TRMTRP	6258#														

E12

MAINDEC-11-DQKTA-A POP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 149
 DQKTA.P11 07-FEB-77 10:30 CROSS REFERENCE TABLE -- MACRO NAMES

TYPBIN	540#														
TYPDEC	540#	5865	5872												
TYPNAM	540#	2634													
TYPNUM	540#														
TYPOCS	540#														
TYPOCT	540#														
TYPTXT	540#	5861	5868												
UPCODE	2364#	6303													
USER	620#	803													
ZEROER	2364#	6502													
SSCMRE	677#	716	717	718	719	720	721								
SSCMTH	677#	722	723	724	725	726	727								
SSESCA	540#														
SSNEWT	540#	2686	2734	2804	2834	2868	2922	2946	3010	3046	3077	3109	3149	3210	3270
	3331	3400	3445	3489	3532	3583	3618	3651	3686	3720	3765	3994	4093	4164	4221
	4248	4358	4565	4623	4771	4853	4967	5059	5204	5334	5491	5638	5767	5803	
SSSET	6258#	6267	6268	6269	6270	6273	6274	6275	6276						
SSSETH	2629#														
SSSETU	2597#														
SSSKIP	540#	2670	2723	2854	2989	3041	3073	3104	3136	3205	3266	3326	3387	3440	3485
	3527	3570	4072	4834	5043	5477	5623	5754							
.EQUAT	389#	430													
.HEADE	389#														
.KT11	389#	540													
.SETUP	389#	621													
.SWRHI	389#	398													
.SWRLO	410#														
.SACT1	389#	667													
.SAPT8	389#	735#													
.SAPTH	389#	645													
.SAPTY	389#	6034													
.SCATC	389#	621													
.SCMTA	389#	677													
.SDB20	389#	6336													
.SEOP	389#	5839													
.SERRO	389#	6443													
.SPOWE	389#	6278													
.SSAVE	389#	5910													
.SSCOP	389#	6376													
.STRAP	389#	6235													
.STYPD	389#	6168													
.STYPE	389#	5955													
.STYPO	389#	6091													

. ABS. 044500 000

ERRORS DETECTED: 0
 DEFAULT GLOBALS GENERATED: 0

DSKZ:DQKTA, DSKZ:DQKTA.SEG/CRF/SOL=DSKZ:DQKTA.P11
 RUN-TIME: 24 21 2 SECONDS
 RUN-TIME RATIO: 628/48=12.9
 CORE USED: 32K (64 PAGES)

F12

MAINDEC-11-DOKTA-A PDP 11/6X MEM. MGMT. DIAG. MACY11 27(1006) 07-FEB-77 10:37 PAGE 150
DOKTAA.P11 07-FEB-77 10:30 CROSS REFERENCE TABLE -- MACRO NAMES