

KD11-K

PDP11/60 MEMORY MANAGEMENT AH-8108B-MC
CQKTAB0

COPYRIGHT © 77-78
FICHE 1 OF 1

MAR 1978
digital
MADE IN USA

This microfiche card contains a grid of frames, each displaying a small portion of data. The data is organized into columns and rows, with some frames containing headers or labels. The overall layout is a dense grid of information, typical of microfiche storage.

95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
1491. 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 B.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

E01

CQKTA-80 PDP 11/6X MEM. MGMT. DIAG.
CQKTAB.P11 30-DEC-77 14:49

MACY11 30A(1052) 03-JAN-78 08:09 PAGE 4

SEQ 0004

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

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 RO-RS 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 B 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

CQKTA-80 PDP 11/6X MEM. MGMT. DIAG.
CQKTAB.P11 30-DEC-77 14:49

MACY11 30A(1052) 03-JAN-78 08:09 PAGE 10

SEQ 0010

377
378
379
380
381
382
383
384
385
386
387

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

388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428

```

.TITLE CQKTA-80 PDP 11/6X MEM. MGMT. DIAG.
*COPYRIGHT (C) JANUARY, 1978
*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

```

```
429
430
431
432      001100
433
434
435
436
437      000011
438      000012
439      000015
440      000200
441      177776
442
443      177774
444      177772
445      177570
446      177570
447
448
449      000000
450      000001
451      000002
452      000003
453      000004
454      000005
455      000006
456      000007
457      000006
458      000007
459
460
461      000000
462      000040
463      000100
464      000140
465      000200
466      000240
467      000300
468      000340
469
470
471      100000
472      040000
473      020000
474      010000
475      004000
476      002000
477      001000
478      000400
479      000200
480      000100
481      000040
482      000020
483      000010
484      000004

.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
```

485 000002
 486 000001
 487
 488
 489
 490
 491
 492
 493
 494
 495
 496
 497
 498
 499 100000
 500 040000
 501 020000
 502 010000
 503 004000
 504 002000
 505 001000
 506 000400
 507 000200
 508 000100
 509 000040
 510 000020
 511 000010
 512 000004
 513 000002
 514 000001
 515
 516
 517
 518
 519
 520
 521
 522
 523
 524
 525
 526
 527 000004
 528 000010
 529 000014
 530 000014
 531 000014
 532 000020
 533 000024
 534 000030
 535 000034
 536 000060
 537 000064
 538 000240
 539
 540

SW01= 2
 SW00= 1
 .EQUIV SW09,SW9
 .EQUIV SW08,SW8
 .EQUIV SW07,SW7
 .EQUIV SW06,SW6
 .EQUIV SW05,SW5
 .EQUIV SW04,SW4
 .EQUIV SW03,SW3
 .EQUIV SW02,SW2
 .EQUIV SW01,SW1
 .EQUIV SW00,SW0

.*DATA BIT DEFINITIONS (BIT00 TO BIT15)

BIT15= 100000
 BIT14= 40000
 BIT13= 20000
 BIT12= 10000
 BIT11= 4000
 BIT10= 2000
 BIT09= 1000
 BIT08= 400
 BIT07= 200
 BIT06= 100
 BIT05= 40
 BIT04= 20
 BIT03= 10
 BIT02= 4
 BIT01= 2
 BIT00= 1
 .EQUIV BIT09,BIT9
 .EQUIV BIT08,BIT8
 .EQUIV BIT07,BIT7
 .EQUIV BIT06,BIT6
 .EQUIV BIT05,BIT5
 .EQUIV BIT04,BIT4
 .EQUIV BIT03,BIT3
 .EQUIV BIT02,BIT2
 .EQUIV BIT01,BIT1
 .EQUIV BIT00,BIT0

.*BASIC "CPU" TRAP VECTOR ADDRESSES

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

541
542
543 000250
544
545
546
547 177572
548 177574
549 177576
550 172516
551
552
553
554 177600
555 177602
556 177604
557 177606
558 177610
559 177612
560 177614
561 177616
562
563
564
565 177640
566 177642
567 177644
568 177646
569 177650
570 177652
571 177654
572 177656
573
574
575
576 172300
577 172302
578 172304
579 172306
580 172310
581 172312
582 172314
583 172316
584
585
586
587 172340
588 172342
589 172344
590 172346
591 172350
592 172352
593 172354
594 172356
595
596

```

;*KT11 VECTOR ADDRESS
MMVEC= 250

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

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

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

;*KERNEL "I" PAGE DESCRIPTOR REGISTERS
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316

;*KERNEL "I" PAGE ADDRESS REGISTERS
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356

```

.EQUIV SP,KSP

597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652

000700
000200
000114
076600
000022
000100
000101
000104
000105
000222
000226
000305
000300
000301
000304
177766

.EQUIV SP, USP
.EQUIV STACK, KERSTK
.EQUIV SRO, MMRO
.EQUIV SR1, MMR1
.EQUIV SR2, MMR2
:
:ADDITIONAL DEFINITIONS
:
:USESTK=STACK-200
:CRLF= 200
:CACHVEC=114
:MED= 76600
:RWHAM= 022
:RLJAM= 100
:RLSERV= 101
:RLFGIN= 104
:RLWHAM= 105
:WVHAM= 222
:WCNSSL= 226
:WLWHAM= 305
:WLJAM= 300
:WLSERV= 301
:WVFGIN= 304
:CPUERR=177766

; CODE FOR CARRIAGE RETURN LINE FEED
; CACHE ERROR INTERRUPT VECTOR
; OP CODE OF MAINTENANCE EXAM/DEP INST.
; READ CODE FOR WHAMI REG.
; READ CODE FOR LOG JAM REGISTER
; READ CODE FOR LOG SERVICE REGISTER
; READ CODE FOR LOG FLAG/INTERRUPT REG.
; READ CODE FOR LOG WHAMI REGISTER
; WRITE CODE FOR WHAMI REG.
; WRITE CODE FOR CONSOLE SW. REG.
; WRITE CODE FOR LOG WHAMI REGISTER

;XXXXXXX

.SBTTL TRAP CATCHER

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

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

.SBTTL STARTING ADDRESS(ES)
JMP @#STRT1 ; JUMP TO STARTING ADDRESS OF PROGRAM
; RUN ENTIRE PROGRAM (ALL TESTS)
JMP STRT2 ; STARTING AT ENTRY POINT 2
; MEMORY MANAGEMENT STATUS REGISTERS
JMP STRT3 ; STARTING AT ENTRY POINT 3
; PAGE ADDRESS AND DESCRIPTOR REGISTERS
JMP STRT4 ; STARTING AT ENTRY POINT 4
; RELOCATION AND ADDER TESTS
JMP STRT5 ; STARTING AT ENTRY POINT 5
; MEMORY MANAGEMENT ABORT LOGIC
JMP STRT6 ; STARTING AT ENTRY POINT 6
; W BIT LOGIC & DUAL MAPPING
JMP STRT7 ; STARTING AT ENTRY POINT 7
; MFP & MTP LOGIC TESTS

.SBTTL APT PARAMETER BLOCK

;
; *****
; SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
; *****
;

653		000234
654		000024
655	000024	000200
656		000044
657	000044	000234
658		000234
659		
660		
661		
662		
663	000234	
664	000234	000000
665	000236	001226
666	000240	000005
667	000242	000015
668	000244	000005
669	000246	000052
670		
671		
672		
673		
674		000250
675		000046
676	000046	041034
677		000052
678	000052	000000
679		000250

```

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

$APTHD:
$HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR: .WORD $MAIL  ;;ADDRESS OF APT MAILBOX (BITS 0-15)
$TSTM:  .WORD 5      ;;RUN TIM OF LONGEST TEST
$PASTM: .WORD 15     ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD 5      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
        .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
.SBTTL ACT11 HOOKS

;*****
;HOOKS REQUIRED BY ACT11
$SVPC=.   ;;SAVE PC
.=46     ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
$ENDAD
.=52     ;;2)SET LOC.52 TO ZERO
.WORD 0
.= $SVPC  ;; RESTORE PC

```

```

680
681
682
683
684
685
686
687 001100 001100
688 001100 000000
689 001102 000
690 001103 000
691 001104 000000
692 001106 000000
693 001110 000000
694 001112 000000
695 001114 000
696 001115 001
697 001116 000000
698 001120 000000
699 001122 000000
700 001124 000000
701 001126 000000
702 001130 000000
703 001132 000000
704 001134 000
705 001135 000
706 001136 000000
707 001140 177570
708 001142 177570
709 001144 177560
710 001146 177562
711 001150 177564
712 001152 177566
713 001154 000
714 001155 002
715 001156 012
716 001157 000
717 001160 000000
718
719 001162 000000
720 001164 000000
721 001166 000000
722 001170 000000
723 001172 000000
724 001174 000000
725 001176 000000
726 001200 000000
727 001202 000000
728 001204 000000
729 001206 000000
730 001210 000000
731 001212 000000
732 001214 000000
733 001216 177607 000377
734 001222 077
735 001223 015

```

.SBTTL COMMON TAGS

```

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

```

SCMTAG: .=1100

```

$CMTAG: .WORD 0
$STNM: .BYTE 0
$ERFLG: .BYTE 0
$ICNT: .WORD 0
$LPADR: .WORD 0
$LPERR: .WORD 0
$ERTTL: .WORD 0
$ITEMB: .BYTE 0
$ERMAX: .BYTE 1
$ERRPC: .WORD 0
$GDADR: .WORD 0
$BDADR: .WORD 0
$GDDAT: .WORD 0
$BDDAT: .WORD 0
$SAUTOB: .BYTE 0
$INTAG: .BYTE 0
$SWR: .WORD DSWR
DISPLAY: .WORD DDISP
$TKS: 177560
$TKB: 177562
$TPS: 177564
$TPB: 177566
$NULL: .BYTE 0
$FILLS: .BYTE 2
$FILLC: .BYTE 12
$TPFLG: .BYTE 0
$REGAD: .WORD 0
$REG0: .WORD 0
$REG1: .WORD 0
$REG2: .WORD 0
$REG3: .WORD 0
$REG4: .WORD 0
$REG5: .WORD 0
$TMP0: .WORD 0
$TMP1: .WORD 0
$TMP2: .WORD 0
$TMP3: .WORD 0
$TMP4: .WORD 0
$TMP5: .WORD 0
$TIMES: 0
$ESCAPE: 0
$BELL: .ASCIZ <207><377><377>
$QUES: .ASCII /?/
$CRLF: .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 ($REG0) WAS OBTAINED
;; CONTAINS (($REGAD)+0)
;; CONTAINS (($REGAD)+2)
;; CONTAINS (($REGAD)+4)
;; CONTAINS (($REGAD)+6)
;; CONTAINS (($REGAD)+10)
;; CONTAINS (($REGAD)+12)
;; USER DEFINED
;; 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

```

736 001224 000012
737
738
739
740
741
742 001226
743 001226 000000
744 001230 000000
745 001232 000000
746 001234 000000
747 001236 000000
748 001240 000000
749 001242 000000
750 001244 000000
751 001246
752 001246 000
753 001247 000
754 001250 000000
755 001252 000000
756 001254 000000
757
758
759
760
761
762
763 001256 000
764 001257 000
765
766
767
768
769 001260 000000
770
771 001262 000
772 001263 000
773 001264 000000
774 001266 000
775 001267 000
776 001270 000000
777 001272 000
778 001273 000
779 001274 000000
780 001276 000000
781 001300 000000
782 001302 000000
783 001304 000000
784 001306 000000
785 001310 000000
786 001312 000000
787 001314 000000
788 001316 000000
789 001320 000000
790 001322 000000
791 001324 000000

```

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

```

792	001326	000000	\$DDW6:	.WORD	ADDW6	::: DEVICE	DESCRIPTOR	WORD#6
793	001330	000000	\$DDW7:	.WORD	ADDW7	::: DEVICE	DESCRIPTOR	WORD#7
794	001332	000000	\$DDW8:	.WORD	ADDW8	::: DEVICE	DESCRIPTOR	WORD#8
795	001334	000000	\$DDW9:	.WORD	ADDW9	::: DEVICE	DESCRIPTOR	WORD#9
796	001336	000000	\$DDW10:	.WORD	ADDW10	::: DEVICE	DESCRIPTOR	WORD#10
797	001340	000000	\$DDW11:	.WORD	ADDW11	::: DEVICE	DESCRIPTOR	WORD#11
798	001342	000000	\$DDW12:	.WORD	ADDW12	::: DEVICE	DESCRIPTOR	WORD#12
799	001344	000000	\$DDW13:	.WORD	ADDW13	::: DEVICE	DESCRIPTOR	WORD#13
800	001346	000000	\$DDW14:	.WORD	ADDW14	::: DEVICE	DESCRIPTOR	WORD#14
801	001350	000000	\$DDW15:	.WORD	ADDW15	::: DEVICE	DESCRIPTOR	WORD#15
802								
803								
804	001352		\$ETEND:					
805								
806								
807	001352	177746	CONTRL:	.WORD	177746		: CACHE CONTROL REGISTER	
808							: CONDITION THAT CAUSED THE TRAP	
809							: TO ERRVEC(000004)	
810	001354	177752	HITMIS:	.WORD	177752		: HIT MISS REGISTER "1"	
811							: IMPLIES HIT IN CACHE	
812	001356	177744	MEMERR:	.WORD	177744		: CACHE ERROR REGISTER	
813	001360	000000	FSTTST:	.WORD	0		: HOLDS THE INDEX TO	
814							: THE STARTING ADDRESS TABLE	
815	001362	000000	CPUEXP:	.WORD	0		: HOLDS EXPECTED CPU ERROR CONDITION	
816	001364	000000	MMEXP:	.WORD	0		: HOLDS EXPECTED MEMORY MANAGEMENT ABORT CONDITION	
817	001366	000000	STKEXP:	.WORD	0		: HOLDS TYPE OF STACK VIOLATION EXPECTED	
818	001370	000000	PADRSL:	.WORD	0		: HOLDS THE LOWER 16 BITS OF A 18-BIT	
819							: ADDRESS GENERATED FOR TYPE OUT	
820	001372	000000	PADRSH:	.WORD	0		: HOLDS THE UPPER 6 BITS OF A 18-BIT	
821							: ADDRESS GENERATED FOR TYPE OUT	
822	001374	000000	PPARER:	.WORD	0		: HOLDS PARITY ERROR REG	
823	001376	000000	PCONTR:	.WORD	0		: HOLDS CACHE CONTROL REG	
824	001400	000000	PHITMI:	.WORD	0		: HOLDS CACHE HIT MISS REG	
825	001402	000000	PMMR0:	.WORD	0		: HOLDS MEMORY MANAGEMENT REGISTER 0	
826	001404	000000	PMMR2:	.WORD	0		: HOLDS MEMORY MANAGEMENT REGISTER 2	
827	001406	000000	PCPUER:	.WORD	0		: HOLDS CPU ERROR REGISTER.	
828	001410	000000	BADPC:	.WORD	0		: HOLDS PC AT ABORT OR TRAP TIME.	
829	001412	000000	TESTNO:	.WORD	0		: HOLDS TEST NUMBER OF LAST ERROR.	
830	001414	000000	DATAOR:	.WORD	0		: HOLDS LOGICAL OR OF BAD DATA	
831	001416	000000	DATAND:	.WORD	0		: HOLDS LOGICAL AND OF BAD DATA	
832	001420	000000	PATTOR:	.WORD	0		: HOLDS LOGICAL OR OF PATTERN LOADED	
833	001422	000000	PATAND:	.WORD	0		: HOLDS LOGICAL AND OF PATTERN LOADED	
834	001424	000000	ADDROR:	.WORD	0		: HOLDS LOGICAL OR OF ADDRESS	
835	001426	000000	ADRAND:	.WORD	0		: HOLDS LOGICAL AND OF ADDRESS	
836	001430	000000	ERRCNT:	.WORD	0		: HOLDS NUMBER OF ERRORS ON TEST	
837	001432	000000	HOLFLG:	.WORD	0		: HOLDS NUMBER OF CONSECUTIVE TIME OUTS	
838							: OCCURRING IN A HOLE IN MEMORY	
839	001434	000000	OLDPC:	.WORD	0		: HOLDS THE RETURN ADDRESS	
840							: IN CASE OF A LOOP ON ERROR	
841	001436	000000	OLDPS:	.WORD	0		: HOLDS THE OLD PROCESSOR STATUS	
842							: IN CASE OF A LOOP ON ERROR	
843	001440	000340	OLDPSW:	.WORD	000340		: HOLDS OLD PSW SO THAT THE T-BIT	
844							: CAN BE RESTORED PROPERLY	
845	001442	000000	RETRY:	.WORD	0		: FLAG TO DECIDE IF ONE PARITY HAS	
846							: BEEN ATTEMPTED	
847	001444	000000	NXTTST:	.WORD	0		: HOLDS ADDRESS OF THE NEXT TEST	

848 001446 000

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

001446

000

001450

001450

001450

172300

001452

172340

001454

177600

001456

177640

001460

021010

001462

021522

001464

022104

001466

025100

001470

027216

001472

033706

001474

036012

FORTY: .BYTE 0

.EVEN

:: THIS AREA STARTS THE DATA TABLE WHERE ANY TABLE REFERENCE WILL
:: REFER TO. ALL DATA WORDS WILL BE LOADED HERE IN ONE SPOT SO THAT
:: IT WILL BE EASIER FOR YOU TO FIND OUT WHAT THAT WORD IS.

PARTAB:

STRTAB:

.WORD 172300
.WORD 172340
.WORD 177600
.WORD 177640
.WORD TST1
.WORD ENTPT2
.WORD ENTPT3
.WORD ENTPT4
.WORD ENTPT5
.WORD ENTPT6
.WORD ENTPT7

;PROCESSOR INDICATOR
;IF = 0 RUNNING ON AN 11/6X
;IF = 1 RUNNING ON AN 11/40

;THIS IS THE TABLE OF THE FIRST
;PAR OR PDR OF EACH GROUP. THEY
;WILL BE USED FOR A DUAL ADDRESSING
;TEST BETWEEN GROUPS.
;KIPDR0
;KIPAR0
;UIPDR0
;UIPAR0
;STARTING ADDRESS OF TEST ONE
;ADDRESS OF ENTRY POINT 2
;ADDRESS OF ENTRY POINT 3
;ADDRESS OF ENTRY POINT 4
;ADDRESS OF ENTRY POINT 5
;ADDRESS OF ENTRY POINT 6
;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
    
```

874				
875				
876				
877				
878				
879				
880				
881				
882				
883				
884				
885				
886				
887				
888	001476			\$ERRTB:
889				
890				
891				: ITEM 1
892	001476	002516	EM1	: UNEXPECTED CPU TRAP OR ABORT THRU 'ERRVEC' (004)
893	001500	007753	DH1	: TESTNO PC AT ABORT
894	001502	013724	DT1	: TESTNO,BADPC,0
895	001504	014656	DF1	: 0,0
896				
897				: ITEM 2
898	001506	002516	EM1	: UNEXPECTED CPU TRAP OR ABORT THRU 'ERRVEC' (004)
899	001510	007776	DH2	: CPU ERR TESTNO PC AT ABORT
900	001512	013732	DT2	: PCPUER,TESTNO,BADPC,0
901	001514	014660	DF2	: 0,0,0
902				
903				: ITEM 3
904	001516	002577	EM3	: UNEXPECTED CACHE PARITY ERROR THRU 'CACHVEC' (114)
905	001520	010032	DH3	: MEM ERR CONTROL HITMISS
906				: REGISTR REGISTR REGISTR TESTNO PC AT ABORT
907	001522	013742	DT3	: PPARER,PCONTR,PHITMI,TESTNO,BADPC,0
908	001524	014663	DF3	: 0,0,0,0,0
909				
910				: ITEM 4
911	001526	002715	EM4	: UNEXPECTED MAIN MEMORY PARITY ERROR THRU 'CACHVEC' (114)
912	001530	010032	DH3	: MEM ERR CONTROL HITMISS
913				: REGISTR REGISTR REGISTR TESTNO PC AT ABORT
914	001532	013742	DT3	: PPARER,PCONTR,PHITMI,TESTNO,BADPC,0
915	001534	014663	DF3	: 0,0,0,0,0
916				
917				: ITEM 5
918	001536	003040	EM5	: UNEXPECTED MEMORY MANAGEMENT TRAP OR ABORT
919	001540	010140	DH5	: ERROR VIRTUAL
920				: REGISTR ADDRESS TESTNO PC AT ABORT
921	001542	013756	DT5	: PMMR0,PMMR2,TESTNO,BADPC,0
922	001544	014670	DF5	: 0,0,0,0
923				
924				: ITEM 6
925	001546	003113	EM6	: MEMORY MANAGEMENT TRAP OR ABORT HAD INCORRECT CONDITION
926	001550	010224	DH6	: EXPECTD ERROR VIRTUAL
927				: CONDITN REGISTR ADDRESS TESTNO PC AT ABORT
928	001552	013770	DT6	: MMEXP,PMMR0,PMMR2,TESTNO,BADPC,0
929	001554	014674	DF6	: 0,0,0,0,0

930					
931			: ITEM 7		
932	001556	003203	EM7	:	MEMORY MANAGEMENT REGISTER 0 WILL NOT CLEAR
933	001560	010330	DH7	:	(MMR0) TESTNO ERRORPC
934	001562	014004	DT7	:	\$REG1, TESTNO, \$ERRPC, 0
935	001564	014701	DF7	:	0, 0, 0
936					
937			: ITEM 10		
938	001566	003257	EM10	:	CAN'T SET 160000 INTO MMR0
939	001570	010330	DH7	:	(MMR0) TESTNO ERRORPC
940	001572	014004	DT7	:	\$REG1, TESTNO, \$ERRPC, 0
941	001574	014701	DF7	:	0, 0, 0
942					
943			: ITEM 11		
944	001576	003310	EM11	:	GOT THE WRONG DATA BACK FROM MMR0
945	001600	010360	DH11	:	LOADED RECEIVD TESTNO ERRORPC
946	001602	014014	DT11	:	\$REG2, \$REG1, TESTNO, \$ERRPC, 0
947	001604	014704	DF11	:	0, 0, 0, 0
948					
949			: ITEM 12		
950	001606	003352	EM12	:	MMR2 DID NOT TRACK PROPERLY
951	001610	010420	DH12	:	EXPECTD (MMR2) TESTNO ERRORPC
952	001612	014026	DT12	:	\$REG3, \$REG2, TESTNO, \$ERRPC
953	001614	014710	DF12	:	0, 0, 0, 0
954					
955			: ITEM 13		
956	001616	003406	EM13	:	SUMMARY OF PAR/PDR REFERENCE TIMEOUTS
957	001620	010460	DH13	:	ADDROR ADDRAND TESTNO ERRORPC #ERRORS
958	001622	014040	DT13	:	ADDROR, ADRAND, TESTNO, \$ERRPC, \$TMP5, 0
959	001624	014714	DF13	:	0, 0, 0, 0, 1
960					
961			: ITEM 14		
962	001626	003454	EM14	:	FOLLOWING PAR/PDR WILL NOT ZERO
963	001630	010531	DH14	:	ADDRESS DATA TESTNO ERRORPC
964	001632	014054	DT14	:	\$REG0, \$REG2, TESTNO, \$ERRPC, 0
965	001634	014721	DF14	:	0, 0, 0, 0
966					
967			: ITEM 15		
968	001636	003515	EM15	:	SUMMARY OF DUAL ADDRESSING ERRORS
969	001640	010571	DH15	:	ADDROR ADDRAND ADDROR ADDRAND
970				:	LOADED LOADED ENABLED ENABLED TESTNO ERRORPC #ERRORS
971	001642	014066	DT15	:	ADDROR, ADRAND, DATAOR, DATAND, TESTNO, \$ERRPC, \$TMP5, 0
972	001644	014725	DF15	:	0, 0, 0, 0, 0, 0, 1
973					
974			: ITEM 16		
975	001646	003557	EM16	:	SUMMARY OF COUNT PATTERN FAILURES
976	001650	010720	DH16	:	ADDROR ADDRAND PATRNOR PATRNAD DATAOR DATAAND TESTNO ERRORPC
977	001652	014106	DT16	:	ADDROR ADRAND, PATTOR, PATAMD, DATAOR, DATAND, TESTNO, \$ERRPC, \$TMP5, 0
978	001654	014734	DF16	:	0, 0, 0, 0, 0, 0, 0, 1
979					
980			: ITEM 17		
981	001656	003621	EM17	:	ERROR IN BYTE ADDRESSING OF PAR/PDR
982	001660	011027	DH17	:	ADDRESS EXPECTD RECEIVD TESTNO ERRORPC
983	001662	014132	DT17	:	\$REG0, \$REG2, \$REG1, TESTNO, \$ERRPC, 0
984	001664	014745	DF17	:	0, 0, 0, 0, 0
985					

986			: ITEM 20	
987	001666	003665	EM20	: ONE OF THE REGISTERS TIMED OUT
988	001670	011077	DH20	: REGADDR TESTNO ERRORPC
989	001672	014146	DT20	: \$REG0, TESTNO, \$ERRPC, 0
990	001674	014752	DF20	: 0, 0, 0
991				
992			: ITEM 21	
993	001676	003724	EM21	: STACK LIMIT REGISTER WOULD NOT CLEAR
994	001700	011127	DH21	: REGADDR DATA TESTNO ERRORPC
995	001702	014156	DT21	: \$REG0, \$REG1, TESTNO, \$ERRPC, 0
996	001704	014755	DF21	: 0, 0, 0, 0
997				
998			: ITEM 22	
999	001706	003771	EM22	: ERROR LOG REG.DID NOT HAVE CORRECT BIT SET
1000	001710	011167	DH22	: REGADR EXPECTD RECEIVD TESTNO ERRORPC
1001	001712	014170	DT22	: \$TMP4, \$TMP5, \$REG0, TESTNO, \$ERRPC, 0
1002	001714	014761	DF22	: 0, 0, 0, 0, 0
1003				
1004			: ITEM 23	
1005	001716	004045	EM23	: LOWER BYTE OF P.S.W. NOT CORRECT
1006	001720	011236	DH23	: REGADDR DATAREC DATAEXP TESTNO ERRORPC
1007	001722	014204	DT23	: \$REG0, \$REG1, \$REG2, TESTNO, \$ERRPC, 0
1008	001724	014766	DF23	: 0, 0, 0, 0, 0
1009				
1010			: ITEM 24	
1011	001726	004106	EM24	: MMR1 DID NOT READ ALL ZEROES
1012	001730	011127	DH21	: REGADDR DATA TESTNO ERRORPC
1013	001732	014156	DT21	: \$REG0, \$REG1, TESTNO, \$ERRPC, 0
1014	001734	014755	DF21	: 0, 0, 0, 0
1015				
1016			: ITEM 25	
1017	001736	004143	EM25	: WRONG DATA BACK FROM STACK LIMIT REGISTER
1018	001740	011236	DH23	: REGADDR DATAREC DATAEXP TESTNO ERRORPC
1019	001742	014204	DT23	: \$REG0, \$REG1, \$REG2, TESTNO, \$ERRPC, 0
1020	001744	014766	DF23	: 0, 0, 0, 0, 0
1021				
1022			: ITEM 26	
1023	001746	004215	EM26	: LOADED MORE THAN UPPER BYTE OF STACK LIMIT REGISTER
1024	001750	011236	DH23	: REGADDR DATAREC DATAEXP TESTNO ERRORPC
1025	001752	014204	DT23	: \$REG0, \$REG1, \$REG2, TESTNO, \$ERRPC, 0
1026	001754	014766	DF23	: 0, 0, 0, 0, 0
1027				
1028			: ITEM 27	
1029	001756	004301	EM27	: KERNEL STACK POINTER NOT 1100 AFTER LOADING USP
1030	001760	011306	DH27	: KSP TESTNO ERRORPC
1031	001762	014220	DT27	: \$REG0, TESTNO, \$ERRPC, 0
1032	001764	014773	DF27	: 0, 0, 0
1033				
1034			: ITEM 30	
1035	001766	004361	EM30	: DUAL ADDRESSING BETWEEN PAR/PDR GROUPS
1036	001770	011336	DH30	: INDEX INDEX PAR/PDR
1037				: EXPECTD RECEIVD ADDRREAD TESTNO ERRORPC
1038	001772	014230	DT30	: \$REG0, \$REG1, \$REG2, TESTNO, \$ERRPC, 0
1039	001774	014776	DF30	: 0, 0, 0, 0, 0
1040				
1041			: ITEM 31	

1042	001776	004430	EM31	:BAD RELOCATION, ON STORING DATA 18-BIT MAPPING
1043	002000	011436	DH31	:VIRTUAL PHYSICAL DATA DATA
1044				:ADDRESS KIPAR4 ADDRESS EXPECTD RECEIVD TESTNO ERRORPC
1045	002002	014244	DT31	:\$REG0,\$TMP1,\$TMP2,\$REG1,\$REG2,TESTNO,\$ERRPC,0
1046	002004	015003	DF31	:0,0,0,0,0,0,0
1047				
1048			: ITEM 32	
1049	002006	004507	EM32	:18-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM
1050	002010	011572	DH32	:STARTADR FNSHADR TESTNO ERRORPC
1051	002012	014264	DT32	:\$TMP1,\$TMP2,TESTNO,\$ERRPC,0
1052	002014	015012	DF32	:2,2,0,0
1053				
1054			: ITEM 33	
1055	002016	004570	EM33	:FAULTY CARRY PROPAGATION 18-BIT MAPPING.
1056	002020	011632	DH33	:PATTERN DATA ADDRESS
1057				:LOADED FETCHED INTENDD KIPAR4 KIPAR5 TESTNO ERRORPC
1058	002022	014276	DT33	:\$REG2,\$REG3,\$REG0,\$TMP3,\$TMP4,TESTNO,\$ERRPC,0
1059	002024	015016	DF33	:0,0,0,0,0,0,0
1060				
1061			: ITEM 34	
1062	002026	004641	EM34	:DIDN'T GET WRAP AROUND TO ADDRESS ZERO
1063	002030	011752	DH34	:DATA TESTNO ERRORPC
1064	002032	014316	DT34	:\$REG1,TESTNO,\$ERRPC,0
1065	002034	015025	DF34	:0,0,0
1066				
1067			: ITEM 35	
1068	002036	004710	EM35	:PAGE LENGTH ABORT AT WRONG VIRTUAL ADDRESS
1069	002040	012002	DH35	:PGLNFD VABLKNO TESTNO ERRORPC
1070	002042	014326	DT35	:\$REG1,\$REG3,TESTNO,\$ERRPC,0
1071	002044	015030	DF35	:0,0,0,0
1072				
1073			: ITEM 36	
1074	002046	004763	EM36	:NO PAGE LENGTH ABORT, WHEN CONDITION CORRECT
1075	002050	012002	DH36	:PGLNFD VABLKNO TESTNO ERRORPC
1076	002052	014326	DT36	:\$REG1,\$REG3,TESTNO,\$ERRPC,0
1077	002054	015030	DF36	:0,0,0,0
1078				
1079			: ITEM 37	
1080	002056	005040	EM37	:DID NOT ABORT ON ACCESS OF NON-RESIDENT PAGE THRU KIPDR5
1081	002060	012042	DH37	:KIPDR5 KIPAR5 VIRTADR TESTNO ERRORPC
1082	002062	014340	DT37	:\$TMP0,\$TMP1,\$TMP2,TESTNO,\$ERRPC,0
1083	002064	015034	DF37	:0,0,0,0,0
1084				
1085			: ITEM 40	
1086	002066	005131	EM40	:DID NOT ABORT ON WRITE TO READ ONLY PAGE THRU KIPDR4
1087	002070	012112	DH40	:KIPDR4 KIPAR4 VIRTADR TESTNO ERRORPC
1088	002072	014340	DT37	:\$TMP0,\$TMP1,\$TMP2,TESTNO,\$ERRPC,0
1089	002074	015034	DF37	:0,0,0,0,0
1090				
1091			: ITEM 41	
1092	002076	005216	EM41	:POINTER VALUE 400 CAUSED STACK VIOLATION
1093	002100	012162	DH41	:STKPTR STKLMT TESTNO ERRORPC
1094	002102	014354	DT41	:\$REG2,\$REG3,TESTNO,\$ERRPC,0
1095	002104	015041	DF41	:0,0,0,0
1096				
1097			: ITEM 42	

1098	002106	005267	EM42	;YELLOW ZONE VIOLATION EXPECTED-GOT NONE
1099	002110	012162	DH41	;STKPTR STKLMT TESTNO ERRORPC
1100	002112	014354	DT41	;\$REG2,\$REG3,TESTNO,\$ERRPC,0
1101	002114	015041	DF41	;0,0,0,0
1102				
1103			; ITEM 43	
1104	002116	005337	EM43	;YELLOW ZONE VIOLATION EXPECTED-GOT RED
1105	002120	012162	DH41	;STKPTR STKLMT TESTNO ERRORPC
1106	002122	014354	DT41	;\$REG2,\$REG3,TESTNO,\$ERRPC,0
1107	002124	015041	DF41	;0,0,0,0
1108				
1109			; ITEM 44	
1110	002126	005406	EM44	;RED ZONE VIOLATION EXPECTED-GOT NONE
1111	002130	012162	DH41	;STKPTR STKLMT TESTNO ERRORPC
1112	002132	014354	DT41	;\$REG2,\$REG3,TESTNO,\$ERRPC,0
1113	002134	015041	DF41	;0,0,0,0
1114				
1115			; ITEM 45	
1116	002136	005453	EM45	;RED ZONE VIOLATION DID NOT CAUSE ABORT
1117	002140	012162	DH41	;STKPTR STKLMT TESTNO ERRORPC
1118	002142	014354	DT41	;\$REG2,\$REG3,TESTNO,\$ERRPC,0
1119	002144	015041	DF41	;0,0,0,0
1120				
1121			; ITEM 46	
1122	002146	005522	EM46	;RED ZONE VIOLATION EXPECTED - GOT YELLOW
1123	002150	012162	DH41	;STKPTR STKLMT TESTNO ERRORPC
1124	002152	014354	DT41	;\$REG2,\$REG3,TESTNO,\$ERRPC,0
1125	002154	015041	DF41	;0,0,0,0
1126				
1127			; ITEM 47	
1128	002156	005573	EM47	;ERROR DURING ERROR-ON-ERROR TRAP SEQUENCE
1129				;EXPECTED:
1130	002160	012222	DH47	;PSW STK PTR (MMR0) (MMR2) TESTNO ERRORPC
1131				;140017 -11/6X- 000000 020151
1132				;140000 -11/40- 4&1074
1133				;RECEIVED:
1134				;PSW PC STK PTR (MMR0) (MMR2) TESTNO ERRORPC
1135	002162	014366	DT47	;\$REG2,\$REG3,\$REG1,PMR0,PMR2,TESTNO,\$ERRPC,0
1136	002164	015045	DF47	;0,0,0,0,0,0,0
1137				
1138			; ITEM 50	
1139	002166	005657	EM50	;AT LEAST ONE M.M. STATUS REGISTERS WAS CLOKED AFTER BEING LOCK
1140	002170	012465	DH50	;ORIGINAL DATA NEW DATA
1141				;(MMR0) (MMR2) (MMR0) (MMR2) TESTNO ERRORPC
1142	002172	014406	DT50	;PMR0,PMR2,\$TMP0,\$TMP2,TESTNO,\$ERRPC,0
1143	002174	015054	DF50	;0,0,0,0,0,0
1144				
1145			; ITEM 51	
1146	002176	005761	EM51	;DID NOT CHANGE MAPPING TO USER MODE
1147	002200	012042	DH37	;TESTNO ERRORPC
1148	002202	014340	DT37	;TESTNO,\$ERRPC,0
1149	002204	015034	DF37	;0,0
1150				
1151			; ITEM 52	
1152	002206	006025	EM52	;ABORT CONDITION INCORRECT EXPECTING 100153
1153	002210	012576	DH52	;RECEIVD (MMR2) TESTNO ERRORPC

1154	002212	014424	DT52	:PMMR0,PMMR2,TESTNO,\$ERRPC,0
1155	002214	015062	DF52	:0,0,0,0
1156				
1157			: ITEM 53	
1158	002216	006100	EM53	:MMR2 LOCKED UP THE WRONG VIRTUAL ADDRESS
1159	002220	012636	DH53	:VIRTADR (MMR2) TESTNO ERRORPC
1160	002222	014436	DT53	:\$REG1,PMMR2,TESTNO,\$ERRPC,0
1161	002224	015066	DF53	:0,0,0,0
1162				
1163			: ITEM 54	
1164	002226	006151	EM54	:MMR0 LOCKED UP THE WRONG PAGE NUMBER
1165	002230	012676	DH54	:EXPECTED (MMR0) TESTNO ERRORPC
1166	002232	014450	DT54	:\$REG2,PMMR0,TESTNO,\$ERRPC,0
1167	002234	015072	DF54	:0,0,0,0
1168				
1169			: ITEM 55	
1170	002236	006216	EM55	:W-BIT NOT SET ON WRITE TO PAGE 4
1171	002240	012736	DH55	:KIPDR4 TESTNO ERRORPC
1172	002242	014462	DT55	:\$TMP0,TESTNO,\$ERRPC,0
1173	002244	015076	DF55	:0,0,0
1174				
1175			: ITEM 56	
1176	002246	006257	EM56	:W-BIT DID NOT REMAIN SET ON M.M. ABORT AT PAGE 4
1177	002250	012736	DH55	:KIPDR4 TESTNO ERRORPC
1178	002252	014462	DT55	:\$TMP0,TESTNO,\$ERRPC,0
1179	002254	015076	DF55	:0,0,0
1180				
1181			: ITEM 57	
1182	002256	006340	EM57	:W-BIT DID NOT CLEAR ON WRITE TO KIPDR4
1183	002260	012736	DH55	:KIPDR4 TESTNO ERRORPC
1184	002262	014462	DT55	:\$TMP0,TESTNO,\$ERRPC,0
1185	002264	015076	DF55	:0,0,0
1186				
1187			: ITEM 60	
1188	002266	006407	EM60	:W-BIT WRONG ON WRITE TO PSW
1189	002270	012766	DH60	:KIPDR7 TESTNO ERRORPC
1190	002272	014462	DT55	:\$TMP0,TESTNO,\$ERRPC,0
1191	002274	015076	DF55	:0,0,0
1192				
1193			: ITEM 61	
1194	002276	006443	EM61	:W-BIT SET AFTER WRITE TO PDR 4
1195	002300	012766	DH60	:KIPDR7 TESTNO ERRORPC
1196	002302	014462	DT55	:\$TMP0,TESTNO,\$ERRPC,0
1197	002304	015076	DF55	:0,0,0
1198				
1199			: ITEM 62	
1200	002306	006502	EM62	:DUAL MAPPING BETWEEN PAGES
1201	002310	013016	DH62	:GDPAGE BDPAGE TESTNO ERRORPC
1202	002312	014472	DT62	:\$REG3,\$REG1,TESTNO,\$ERRPC,0
1203	002314	015101	DF62	:0,0,0,0
1204				
1205			: ITEM 63	
1206	002316	006535	EM63	:NO PAGE HAD ITS BIT SET
1207	002320	013056	DH63	:TSTPAGE CONTENT TESTNO ERRORPC
1208	002322	014504	DT63	:\$REG3,\$REG0,TESTNO,\$ERRPC,0
1209	002324	015105	DF63	:0,0,0,0

1210				
1211			: ITEM 64	
1212	002326	006565	EM64	: DID NOT PICK UP CORRECT STACK POINTER
1213	002330	013116	DH64	: EXPECTD RECEIVD TESTNO ERRORPC
1214	002332	014516	DT64	: \$REG2, \$REG1, TESTNO, \$ERRPC, 0
1215	002334	015111	DF64	: 0, 0, 0, 0
1216				
1217			: ITEM 65	
1218	002336	006633	EM65	: CURRENT MODE STACK NOT PUSHED IN MFP INSTRUCTION
1219	002340	012042	DH37	: TESTNO ERRORPC
1220	002342	014340	DT37	: TESTNO, \$ERRPC, 0
1221	002344	015034	DF37	: 0, 0
1222				
1223			: ITEM 66	
1224	002346	006714	EM66	: WRONG DATA FETCHED BY MFP INSTRUCTION
1225	002350	013156	DH66	: EXPECTD RECEIVD TESTNO ERRORPC
1226	002352	014530	DT66	: \$REG0, \$REG1, TESTNO, \$ERRPC, 0
1227	002354	015115	DF66	: 0, 0, 0, 0
1228				
1229			: ITEM 67	
1230	002356	006762	EM67	: TRIED TO REFERENCE NON-RESIDENT PAGE
1231	002360	013216	DH67	: (MMR0) (MMR2) TESTNO ERRORPC
1232	002362	014542	DT67	: PMMR0, PMMR2, TESTNO, \$ERRPC, 0
1233	002364	015121	DF67	: 0, 0, 0, 0
1234				
1235			: ITEM 70	
1236	002366	007027	EM70	: STACK POINTER NOT CHANGED BY MTP INSTRUCTION
1237	002370	013256	DH70	: STKPTR TESTNO ERRORPC
1238	002372	014554	DT70	: \$REG1, TESTNO, \$ERRPC, 0
1239	002374	015125	DF70	: 0, 0, 0
1240				
1241			: ITEM 71	
1242	002376	007104	EM71	: INCORRECT STORE BY MTP INSTRUCTION
1243	002400	013306	DH71	: GDDATA STORED TESTNO ERRORPC
1244	002402	014530	DT66	: \$REG0, \$REG1, TESTNO, \$ERRPC, 0
1245	002404	015115	DF66	: 0, 0, 0, 0
1246				
1247			: ITEM 72	
1248	002406	007147	EM72	: ABORTED THRU RIGHT VECTOR BUT PICKED UP WRONG PSW
1249	002410	013346	DH72	: (PSW) TESTNO ERRORPC EXPECTING XXX340
1250	002412	014564	DT72	: \$REG0, TESTNO, \$ERRPC, 0
1251	002414	015130	DF72	: 0, 0, 0
1252				
1253			: ITEM 73	
1254	002416	007231	EM73	: ABORTED THRU USER SPACE, USER PSW IS XXX000
1255	002420	013417	DH73	: (PSW) TESTNO ERRORPC
1256	002422	014564	DT72	: \$REG0, TESTNO, \$ERRPC, 0
1257	002424	015130	DF72	: 0, 0, 0
1258				
1259			: ITEM 74	
1260	002426	007305	EM74	: DID NOT ABORT REFERENCE TO A MEMORY MANAGEMENT REGISTER
1261	002430	013447	DH74	: TESTNO ERRORPC
1262	002432	014574	DT74	: TESTNO, \$ERRPC, 0
1263	002434	015133	DF74	: 0, 0
1264				
1265			: ITEM 75	

1266	002436	007375			EM75					;CPU ERROR REG. DID NOT REPORT YELLOW ZONE
1267	002440	013466			DH75					;STKPTR STKLMT CPUERR TESTNO ERRORPC
1268	002442	014602			DT75					;\$REG2,\$REG3,PCPUER,TESTNO,\$ERRPC,0
1269	002444	015135			DF75					;0,0,0,0,0
1270										
1271										;ITEM 76
1272	002446	007447			EM76					;CPU ERROR REG. DID NOT REPORT RED ZONE
1273	002450	013466			DH75					;STKPTR STKLMT CPUERR TESTNO ERRORPC
1274	002452	014602			DT75					;\$REG2,\$REG3,PCPUER,TESTNO,\$ERRPC,0
1275	002454	015135			DF75					;0,0,0,0,0
1276										
1277										;ITEM 77
1278	002456	002715			EM4					;UNEXPECTED MAIN MEMORY PARITY ERROR THRU 'CACHVEC' (114)
1279	002460	007753			DH1					;TESTNO PC AT ABORT
1280	002462	013724			DT1					;TESTNO,BADPC,0
1281	002464	014656			DF1					;0,0
1282										
1283										
1284										
1285	002466				ER200:					;STARTING ADDRESS FOR ITEM 201-377
1286										
1287										;ITEM 201
1288	002466	007516			EM201					;THE FOLLOWING ARE PAR/PDR REFERENCE TIMEOUTS
1289	002470	013536			DH201					;ADDRESS TESTNO ERRORPC
1290										
1291										;ITEM 301
1292	002472	014616			DT201					;\$REG0,TESTNO,\$ERRPC,0
1293	002474	015142			DF201					;0,0,0
1294										
1295										;ITEM 202
1296	002476	007573			EM202					;THE FOLLOWING ARE DUAL ADDRESSING ERRORS FOR PAR'S/PDR'S
1297	002500	013565			DH202					;ADDRESS ADDRESS
1298										;LOADED JUST READ TESTNO ERRORPC
1299										
1300										;ITEM 302
1301	002502	014626			DT202					;\$REG0,\$REG1,TESTNO,\$ERRPC,0
1302	002504	015145			DF202					;0,0,0,0
1303										
1304										;ITEM 203
1305	002506	007664			EM203					;THE FOLLOWING ARE COUNT PATTERN ERRORS FOR PAR'S/PDR'S
1306	002510	013645			DH203					;ADDRESS RECEIVD EXPECTD COUNT TESTNO ERRORPC
1307										
1308										;ITEM 303
1309	002512	014640			DT203					;\$REG0,\$REG2,\$REG4,\$REG1,TESTNO,\$ERRPC,0
1310	002514	015151			DF203					;0,0,0,0,0,0
1311										
1312										
1313										.SETTL ERROR TABLE MESSAGES AND DATA POINTERS
1314	002516	047125	054105	042520	EM1:					.ASCIZ ?UNEXPECTED CPU TRAP OR ABORT THRU 'ERRVEC' (004)?
1315	002524	052103	042105	041440						
1316	002532	052520	052040	040522						
1317	002540	020120	051117	040440						
1318	002546	047502	052122	052040						
1319	002554	051110	020125	042447						
1320	002562	051122	042526	023503						
1321	002570	024040	030060	024464						

1322	002576	000				
1323	002577	125	042516	050130	EM3:	.ASCII ?UNEXPECTED CACHE PARITY ERROR THRU 'CACHVEC' (114)?<CRLF>
1324	002604	041505	042524	020104		
1325	002612	040503	044103	020105		
1326	002620	040520	044522	054524		
1327	002626	042440	051122	051117		
1328	002634	052040	051110	020125		
1329	002642	041447	041501	053110		
1330	002650	041505	020047	030450		
1331	002656	032061	100051			
1332	002662	044527	046114	051040		.ASCIZ ?WILL RETRY THIS TEST ONCE.?
1333	002670	052105	054522	052040		
1334	002676	044510	020123	042524		
1335	002704	052123	047440	041516		
1336	002712	027105	000			
1337	002715	125	042516	050130	EM4:	.ASCII ?UNEXPECTED MAIN MEMORY PARITY ERROR THRU 'CACHVEC' (114)?<CRLF>
1338	002722	041505	042524	020104		
1339	002730	040515	047111	046440		
1340	002736	046505	051117	020131		
1341	002744	040520	044522	054524		
1342	002752	042440	051122	051117		
1343	002760	052040	051110	020125		
1344	002766	041447	041501	053110		
1345	002774	041505	020047	030450		
1346	003002	032061	100051			
1347	003006	044527	046114	051040		.ASCIZ ?WILL RETRY THIS TEST ONCE?
1348	003014	052105	054522	052040		
1349	003022	044510	020123	042524		
1350	003030	052123	047440	041516		
1351	003036	000105				
1352						
1353	003040	047125	054105	042520	EM5:	.ASCIZ ?UNEXPECTED MEMORY MANAGEMENT TRAP OR ABORT?
1354	003046	052103	042105	046440		
1355	003054	046505	051117	020131		
1356	003062	040515	040516	042507		
1357	003070	042515	052116	052040		
1358	003076	040522	020120	051117		
1359	003104	040440	047502	052122		
1360	003112	000				
1361	003113	115	046505	051117	EM6:	.ASCIZ ?MEMORY MANAGEMENT TRAP OR ABORT HAD INCORRECT CONDITION?
1362	003120	020131	040515	040516		
1363	003126	042507	042515	052116		
1364	003134	052040	040522	020120		
1365	003142	051117	040440	047502		
1366	003150	052122	044040	042101		
1367	003156	044440	041516	051117		
1368	003164	042522	052103	041440		
1369	003172	047117	044504	044524		
1370	003200	047117	000			
1371	003203	115	046505	051117	EM7:	.ASCIZ ?MEMORY MANAGEMENT REGISTER 0 WILL NOT CLEAR?
1372	003210	020131	040515	040516		
1373	003216	042507	042515	052116		
1374	003224	051040	043505	051511		
1375	003232	042524	020122	020060		
1376	003240	044527	046114	047040		
1377	003246	052117	041440	042514		

E03

CQKTA-BO PDP 11/6X MEM. MGMT. DIAG.
CQKTAB.P11 30-DEC-77 14:49MACY11 30A(1052) 03-JAN-78 08:09 PAGE 30
ERROR TABLE MESSAGES AND DATA POINTERS

SEQ 0030

1378	003254	051101	000		
1379	003257	103	047101	052047	EM10: .ASCIZ ?CAN'T SET 160000 IN MMRO?
1380	003264	051440	052105	030440	
1381	003272	030066	030060	020060	
1382	003300	047111	046440	051115	
1383	003306	000060			
1384	003310	047507	020124	044124	EM11: .ASCIZ ?GOT THE WRONG DATA BACK FROM MMRO?
1385	003316	020105	051127	047117	
1386	003324	020107	040504	040524	
1387	003332	041040	041501	020113	
1388	003340	051106	046517	046440	
1389	003346	051115	000060		
1390	003352	046515	031122	042040	EM12: .ASCIZ ?MMR2 DID NOT TRACK PROPERLY?
1391	003360	042111	047040	052117	
1392	003366	052040	040522	045503	
1393	003374	050040	047522	042520	
1394	003402	046122	000131		
1395	003406	052523	046515	051101	EM13: .ASCIZ ?SUMMARY OF PAR/PDR REFERENCE TIMEOUTS?
1396	003414	020131	043117	050040	
1397	003422	051101	050057	051104	
1398	003430	051040	043105	051105	
1399	003436	047105	042503	052040	
1400	003444	046511	047505	052125	
1401	003452	000123			
1402	003454	047506	046114	053517	EM14: .ASCIZ ?FOLLOWING PAR/PDR WILL NOT CLEAR?
1403	003462	047111	020107	040520	
1404	003470	027522	042120	020122	
1405	003476	044527	046114	047040	
1406	003504	052117	041440	042514	
1407	003512	051101	000		
1408	003515	123	046525	040515	EM15: .ASCIZ ?SUMMARY OF DUAL ADDRESSING ERRORS?
1409	003522	054522	047440	020106	
1410	003530	052504	046101	040440	
1411	003536	042104	042522	051523	
1412	003544	047111	020107	051105	
1413	003552	047522	051522	000	
1414	003557	123	046525	040515	EM16: .ASCIZ ?SUMMARY OF COUNT PATTERN FAILURES?
1415	003564	054522	047440	020106	
1416	003572	047503	047125	020124	
1417	003600	040520	052124	051105	
1418	003606	020116	040506	046111	
1419	003614	051125	051505	000	
1420	003621	105	051122	051117	EM17: .ASCIZ ?ERROR IN BYTE ADDRESSING OF PAR/PDR?
1421	003626	044440	020116	054502	
1422	003634	042524	040440	042104	
1423	003642	042522	051523	047111	
1424	003650	020107	043117	050040	
1425	003656	051101	050057	051104	
1426	003664	000			
1427	003665	117	042516	047440	EM20: .ASCIZ ?ONE OF THE REGISTERS TIMED OUT?
1428	003672	020106	044124	020105	
1429	003700	042522	044507	052123	
1430	003706	051105	020123	044524	
1431	003714	042515	020104	052517	
1432	003722	000124			
1433	003724	052123	041501	020113	EM21: .ASCIZ ?STACK LIMIT REGISTER WOULD NOT CLEAR?

F03

CQKTA-80 PDP 11/6X MEM. MGMT. DIAG.
CQKTAB.P11 30-DEC-77 14:49

MACY11 30A(1052) 03-JAN-78 08:09 PAGE 31
ERROR TABLE MESSAGES AND DATA POINTERS

SEQ 0031

1434	003732	044514	044515	020124	
1435	003740	042522	044507	052123	
1436	003746	051105	053440	052517	
1437	003754	042114	047040	052117	
1438	003762	041440	042514	051101	
1439	003770	000			
1440	003771	105	051122	051117	EM22: .ASCIZ ?ERROR LOG REG. DID NOT HAVE CORRECT BIT SET?
1441	003776	046040	043517	051040	
1442	004004	043505	020056	044504	
1443	004012	020104	047516	020124	
1444	004020	040510	042526	041440	
1445	004026	051117	042522	052103	
1446	004034	041040	052111	051440	
1447	004042	052105	000		
1448	004045	114	053517	051105	EM23: .ASCIZ ?LOWER BYTE OF P.S.W. NOT CORRECT?
1449	004052	041040	052131	020105	
1450	004060	043117	050040	051456	
1451	004066	053456	020056	047516	
1452	004074	020124	047503	051122	
1453	004102	041505	000124		
1454	004106	046515	030522	042040	EM24: .ASCIZ ?MMR1 DID NOT READ ALL ZEROES?
1455	004114	042111	047040	052117	
1456	004122	051040	040505	020104	
1457	004130	046101	020114	042532	
1458	004136	047522	051505	000	
1459	004143	127	047522	043516	EM25: .ASCIZ ?WRONG DATA BACK FROM STACK LIMIT REGISTER?
1460	004150	042040	052101	020101	
1461	004156	040502	045503	043040	
1462	004164	047522	020115	052123	
1463	004172	041501	020113	044514	
1464	004200	044515	020124	042522	
1465	004206	044507	052123	051105	
1466	004214	000			
1467	004215	114	040517	042504	EM26: .ASCIZ ?LOADED MORE THAN UPPER BYTE OF STACK LIMIT REGISTER?
1468	004222	020104	047515	042522	
1469	004230	052040	040510	020116	
1470	004236	050125	042520	020122	
1471	004244	054502	042524	047440	
1472	004252	020106	052123	041501	
1473	004260	020113	044514	044515	
1474	004266	020124	042522	044507	
1475	004274	052123	051105	000	
1476	004301	113	051105	042516	EM27: .ASCIZ ?KERNEL STACK POINTER NOT 1100 AFTER LOADING USP?
1477	004306	020114	052123	041501	
1478	004314	020113	047520	047111	
1479	004322	042524	020122	047516	
1480	004330	020124	030461	030060	
1481	004336	040440	052106	051105	
1482	004344	046040	040517	044504	
1483	004352	043516	052440	050123	
1484	004360	000			
1485	004361	104	040525	020114	EM30: .ASCIZ ?DUAL ADDRESSING BETWEEN PAR/PDR GROUPS?
1486	004366	042101	051104	051505	
1487	004374	044523	043516	041040	
1488	004402	052105	042527	047105	
1489	004410	050040	051101	050057	

1490	004416	051104	043440	047522	
1491	004424	050125	000123		
1492	004430	040502	020104	042522	EM31: .ASCIZ ?BAD RELOCATION, ON STORING DATA 18-BIT MAPPING?
1493	004436	047514	040503	044524	
1494	004444	047117	020054	047117	
1495	004452	051440	047524	044522	
1496	004460	043516	042040	052101	
1497	004466	020101	034061	041055	
1498	004474	052111	046440	050101	
1499	004502	044520	043516	000	
1500	004507	061	026470	044502	EM32: .ASCIZ ?18-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM?
1501	004514	020124	040515	050120	
1502	004522	047111	020107	047520	
1503	004530	051523	041111	042514	
1504	004536	044040	046117	020105	
1505	004544	047111	046440	044501	
1506	004552	020116	042515	047515	
1507	004560	054522	043040	047522	
1508	004566	000115			
1509	004570	040506	046125	054524	EM33: .ASCIZ ?FAULTY CARRY PROPAGATION 18-BIT MAPPING.?
1510	004576	041440	051101	054522	
1511	004604	050040	047522	040520	
1512	004612	040507	044524	047117	
1513	004620	030440	026470	044502	
1514	004626	020124	040515	050120	
1515	004634	047111	027107	000	
1516	004641	104	042111	023516	EM34: .ASCIZ ?DIDN'T GET WRAP AROUND TO ADDRESS ZERO?
1517	004646	020124	042507	020124	
1518	004654	051127	050101	040440	
1519	004662	047522	047125	020104	
1520	004670	047524	040440	042104	
1521	004676	042522	051523	055040	
1522	004704	051105	000117		
1523	004710	040520	042507	046040	EM35: .ASCIZ ?PAGE LENGTH ABORT AT WRONG VIRTUAL ADDRESS?
1524	004716	047105	052107	020110	
1525	004724	041101	051117	020124	
1526	004732	052101	053440	047522	
1527	004740	043516	053040	051111	
1528	004746	052524	046101	040440	
1529	004754	042104	042522	051523	
1530	004762	000			
1531	004763	116	020117	040520	EM36: .ASCIZ ?NO PAGE LENGTH ABORT, WHEN CONDITION CORRECT?
1532	004770	042507	046040	047105	
1533	004776	052107	020110	041101	
1534	005004	051117	026124	053440	
1535	005012	042510	020116	047503	
1536	005020	042116	052111	047511	
1537	005026	020116	047503	051122	
1538	005034	041505	000124		
1539	005040	044504	020104	047516	EM37: .ASCIZ ?DID NOT ABORT ON ACCESS OF NON-RESIDENT PAGE THRU KIPDRS?
1540	005046	020124	041101	051117	
1541	005054	020124	047117	040440	
1542	005062	041503	051505	020123	
1543	005070	043117	047040	047117	
1544	005076	051055	051505	042111	
1545	005104	047105	020124	040520	

1546	005112	042507	052040	051110	
1547	005120	020125	044513	042120	
1548	005126	032522	000		
1549	005131	104	042111	047040	EM40: .ASCIZ ?DID NOT ABORT ON WRITE TO READ ONLY PAGE THRU KIPDR4?
1550	005136	052117	040440	047502	
1551	005144	052122	047440	020116	
1552	005152	051127	052111	020105	
1553	005160	047524	051040	040505	
1554	005166	020104	047117	054514	
1555	005174	050040	043501	020105	
1556	005202	044124	052522	045440	
1557	005210	050111	051104	000064	
1558	005216	047520	047111	042524	EM41: .ASCIZ ?POINTER VALUE 400 CAUSED STACK VIOLATION?
1559	005224	020122	040526	052514	
1560	005232	020105	030064	020060	
1561	005240	040503	051525	042105	
1562	005246	051440	040524	045503	
1563	005254	053040	047511	040514	
1564	005262	044524	047117	000	
1565	005267	131	046105	047514	EM42: .ASCIZ ?YELLOW ZONE VIOLATION EXPECTED-GOT NONE?
1566	005274	020127	047532	042516	
1567	005302	053040	047511	040514	
1568	005310	044524	047117	042440	
1569	005316	050130	041505	042524	
1570	005324	026504	047507	020124	
1571	005332	047516	042516	000	
1572	005337	131	046105	047514	EM43: .ASCIZ ?YELLOW ZONE VIOLATION EXPECTED-GOT RED?
1573	005344	020127	047532	042516	
1574	005352	053040	047511	040514	
1575	005360	044524	047117	042440	
1576	005366	050130	041505	042524	
1577	005374	026504	047507	020124	
1578	005402	042522	000104		
1579	005406	042522	020104	047532	EM44: .ASCIZ ?RED ZONE VIOLATION EXPECTED-GOT NONE?
1580	005414	042516	053040	047511	
1581	005422	040514	044524	047117	
1582	005430	042440	050130	041505	
1583	005436	042524	026504	047507	
1584	005444	020124	047516	042516	
1585	005452	000			
1586	005453	122	042105	055040	EM45: .ASCIZ ?RED ZONE VIOLATION DID NOT CAUSE ABORT?
1587	005460	047117	020105	044526	
1588	005466	046117	052101	047511	
1589	005474	020116	044504	020104	
1590	005502	047516	020124	040503	
1591	005510	051525	020105	041101	
1592	005516	051117	000124		
1593	005522	042522	020104	047532	EM46: .ASCIZ ?RED ZONE VIOLATION EXPECTED - GOT YELLOW?
1594	005530	042516	053040	047511	
1595	005536	040514	044524	047117	
1596	005544	042440	050130	041505	
1597	005552	042524	020104	020055	
1598	005560	047507	020124	042531	
1599	005566	046114	053517	000	
1600	005573	105	051122	051117	EM47: .ASCII ?ERROR DURING ERROR-ON-ERROR TRAP SEQUENCE?<CRLF>
1601	005600	042040	051125	047111	

1602	005606	020107	051105	047522
1603	005614	026522	047117	042455
1604	005622	051122	051117	052040
1605	005630	040522	020120	042523
1606	005636	052521	047105	042503
1607	005644	200		
1608	005645	105	050130	041505
1609	005652	042524	035104	000
1610	005657	101	020124	042514
1611	005664	051501	020124	047117
1612	005672	020105	027115	027115
1613	005700	051440	040524	052524
1614	005706	020123	042522	044507
1615	005714	052123	051105	020123
1616	005722	040527	020123	046103
1617	005730	041517	042513	020104
1618	005736	043101	042524	020122
1619	005744	042502	047111	020107
1620	005752	047514	045503	042105
1621	005760	000		
1622	005761	104	042111	047040
1623	005766	052117	041440	040510
1624	005774	043516	020105	040515
1625	006002	050120	047111	020107
1626	006010	047524	052440	042523
1627	006016	020122	047515	042504
1628	006024	000		
1629	006025	101	047502	052122
1630	006032	041440	047117	044504
1631	006040	044524	047117	044440
1632	006046	041516	051117	042522
1633	006054	052103	042440	050130
1634	006062	041505	044524	043516
1635	006070	030440	030060	032461
1636	006076	000063		
1637	006100	046515	031122	046040
1638	006106	041517	042513	020104
1639	006114	050125	052040	042510
1640	006122	053440	047522	043516
1641	006130	053040	051111	052524
1642	006136	046101	040440	042104
1643	006144	042522	051523	000
1644	006151	115	051115	020060
1645	006156	047514	045503	042105
1646	006164	052440	020120	044124
1647	006172	020105	051127	047117
1648	006200	020107	040520	042507
1649	006206	047040	046525	042502
1650	006214	000122		
1651	006216	026527	044502	020124
1652	006224	047516	020124	042523
1653	006232	020124	047117	053440
1654	006240	044522	042524	052040
1655	006246	020117	040520	042507
1656	006254	032040	000	
1657	006257	127	041055	052111

.ASCIZ ?EXPECTED:?

EM50: .ASCIZ ?AT LEAST ONE M.M. STATUS REGISTERS WAS CLOCKED AFTER BEING LOCKED?

EM51: .ASCIZ ?DID NOT CHANGE MAPPING TO USER MODE?

EM52: .ASCIZ ?ABORT CONDITION INCORRECT EXPECTING 100153?

EM53: .ASCIZ ?MMR2 LOCKED UP THE WRONG VIRTUAL ADDRESS?

EM54: .ASCIZ ?MMR0 LOCKED UP THE WRONG PAGE NUMBER?

EM55: .ASCIZ ?W-BIT NOT SET ON WRITE TO PAGE 4?

EM56: .ASCIZ ?W-BIT DID NOT REMAIN SET ON M.M. ABORT AT PAGE 4?

1658	006264	042040	042111	047040	
1659	006272	052117	051040	046505	
1660	006300	044501	020116	042523	
1661	006306	020124	047117	046440	
1662	006314	046456	020056	041101	
1663	006322	051117	020124	052101	
1664	006330	050040	043501	020105	
1665	006336	000064			
1666	006340	026527	044502	020124	EM57: .ASCIZ ?W-BIT DID NOT CLEAR ON WRITE TO KIPDR4?
1667	006346	044504	020104	047516	
1668	006354	020124	046103	040505	
1669	006362	020122	047117	053440	
1670	006370	044522	042524	052040	
1671	006376	020117	044513	042120	
1672	006404	032122	000		
1673	006407	127	041055	052111	EM60: .ASCIZ ?W-BIT WRONG ON WRITE TO PSW?
1674	006414	053440	047522	043516	
1675	006422	047440	020116	051127	
1676	006430	052111	020105	047524	
1677	006436	050040	053523	000	
1678	006443	127	041055	052111	EM61: .ASCIZ ?W-BIT SET AFTER WRITE TO PDR 4?
1679	006450	051440	052105	040440	
1680	006456	052106	051105	053440	
1681	006464	044522	042524	052040	
1682	006472	020117	042120	020122	
1683	006500	000064			
1684	006502	052504	046101	046440	EM62: .ASCIZ ?DUAL MAPPING BETWEEN PAGES?
1685	006510	050101	044520	043516	
1686	006516	041040	052105	042527	
1687	006524	047105	050040	043501	
1688	006532	051505	000		
1689	006535	116	020117	040520	EM63: .ASCIZ ?NO PAGE HAD ITS BIT SET?
1690	006542	042507	044040	042101	
1691	006550	044440	051524	041040	
1692	006556	052111	051440	052105	
1693	006564	000			
1694	006565	104	042111	047040	EM64: .ASCIZ ?DID NOT PICK UP CORRECT STACK POINTER?
1695	006572	052117	050040	041511	
1696	006600	020113	050125	041440	
1697	006606	051117	042522	052103	
1698	006614	051440	040524	045503	
1699	006622	050040	044517	052116	
1700	006630	051105	000		
1701	006633	103	051125	042522	EM65: .ASCIZ ?CURRENT MODE STACK NOT PUSHED IN MFP INSTRUCTION?
1702	006640	052116	046440	042117	
1703	006646	020105	052123	041501	
1704	006654	020113	047516	020124	
1705	006662	052520	044123	042105	
1706	006670	044440	020116	043115	
1707	006676	020120	047111	052123	
1708	006704	052522	052103	047511	
1709	006712	000116			
1710	006714	051127	047117	020107	EM66: .ASCIZ ?WRONG DATA FETCHED BY MFP INSTRUCTION?
1711	006722	040504	040524	043040	
1712	006730	052105	044103	042105	
1713	006736	041040	020131	043115	

K03

CQKTA-BO PDP 11/6X MEM. MGMT. DIAG.
CQKTAB.P11 30-DEC-77 14:49

MACY11 30A(1052) 03-JAN-78 08:09 PAGE 36
ERROR TABLE MESSAGES AND DATA POINTERS

SEQ 0036

1714	006744	020120	047111	052123	
1715	006752	052522	052103	047511	
1716	006760	000116			
1717	006762	051124	042511	020104	EM67: .ASCIZ ?TRIED TO REFERENCE NON-RESIDENT PAGE?
1718	006770	047524	051040	043105	
1719	006776	051105	047105	042503	
1720	007004	047040	047117	051055	
1721	007012	051505	042111	047105	
1722	007020	020124	040520	042507	
1723	007026	000			
1724	007027	123	040524	045503	EM70: .ASCIZ ?STACK POINTER NOT CHANGED BY MTP INSTRUCTION?
1725	007034	050040	044517	052116	
1726	007042	051105	047040	052117	
1727	007050	041440	040510	043516	
1728	007056	042105	041040	020131	
1729	007064	052115	020120	047111	
1730	007072	052123	052522	052103	
1731	007100	047511	000116		
1732	007104	047111	047503	051122	EM71: .ASCIZ ?INCORRECT STORE BY MTP INSTRUCTION?
1733	007112	041505	020124	052123	
1734	007120	051117	020105	054502	
1735	007126	046440	050124	044440	
1736	007134	051516	051124	041525	
1737	007142	044524	047117	000	
1738	007147	101	047502	052122	EM72: .ASCIZ ?ABORTED THRU RIGHT VECTOR BUT PICKED UP WRONG PSW?
1739	007154	042105	052040	051110	
1740	007162	020125	044522	044107	
1741	007170	020124	042526	052103	
1742	007176	051117	041040	052125	
1743	007204	050040	041511	042513	
1744	007212	020104	050125	053440	
1745	007220	047522	043516	050040	
1746	007226	053523	000		
1747	007231	101	047502	052122	EM73: .ASCIZ ?ABORTED THRU USER SPACE, USER PSW IS XXX000?
1748	007236	042105	052040	051110	
1749	007244	020125	051525	051105	
1750	007252	051440	040520	042503	
1751	007260	020054	051525	051105	
1752	007266	050040	053523	044440	
1753	007274	020123	054130	030130	
1754	007302	030060	000		
1755	007305	104	042111	047040	EM74: .ASCIZ ?DID NOT ABORT REFERENCE TO A MEMORY MANAGEMENT REGISTER?
1756	007312	052117	040440	047502	
1757	007320	052122	051040	043105	
1758	007326	051105	047105	042503	
1759	007334	052040	020117	020101	
1760	007342	042515	047515	054522	
1761	007350	046440	047101	043501	
1762	007356	046505	047105	020124	
1763	007364	042522	044507	052123	
1764	007372	051105	000		
1765	007375	103	052520	042440	EM75: .ASCIZ ?CPU ERROR REG. DID NOT REPORT YELLOW ZONE?
1766	007402	051122	051117	051040	
1767	007410	043505	020056	044504	
1768	007416	020104	047516	020124	
1769	007424	042522	047520	052122	

1770	007432	054440	046105	047514	
1771	007440	020127	047532	042516	
1772	007446	000			
1773	007447	103	052520	042440	EM76: .ASCIZ ?CPU ERROR REG. DID NOT REPORT RED ZONE?
1774	007454	051122	051117	051040	
1775	007462	043505	020056	044504	
1776	007470	020104	047516	020124	
1777	007476	042522	047520	052122	
1778	007504	051040	042105	055040	
1779	007512	047117	000105		
1780					
1781	007516	044124	020105	047506	EM201: .ASCIZ ?THE FOLLOWING ARE PAR/PDR REFERENCE TIMEOUTS?
1782	007524	046114	053517	047111	
1783	007532	020107	051101	020105	
1784	007540	040520	027522	042120	
1785	007546	020122	042522	042506	
1786	007554	042522	041516	020105	
1787	007562	044524	042515	052517	
1788	007570	051524	000		
1789	007573	124	042510	043040	EM202: .ASCIZ ?THE FOLLOWING ARE DUAL ADDRESSING ERRORS FOR PAR'S/PDR'S?
1790	007600	046117	047514	044527	
1791	007606	043516	040440	042522	
1792	007614	042040	040525	020114	
1793	007622	042101	051104	051505	
1794	007630	044523	043516	042440	
1795	007636	051122	051117	020123	
1796	007644	047506	020122	040520	
1797	007652	023522	027523	042120	
1798	007660	023522	000123		
1799	007664	044124	020105	047506	EM203: .ASCIZ ?THE FOLLOWING ARE COUNT PATTERN ERRORS FOR PAR'S/PDR'S?
1800	007672	046114	053517	047111	
1801	007700	020107	051101	020105	
1802	007706	047503	047125	020124	
1803	007714	040520	052124	051105	
1804	007722	020116	051105	047522	
1805	007730	051522	043040	051117	
1806	007736	050040	051101	051447	
1807	007744	050057	051104	051447	
1808	007752	000			
1809					
1810					
1811	007753	124	051505	047124	DH1: .ASCII ?TESTNO PC AT ABORT?
1812	007760	020117	050040	020103	
1813	007766	052101	040440	047502	
1814	007774	052122			
1815	007776	050103	020125	051105	DH2: .ASCIZ ?CPU ERR TESTNO PC AT ABORT?
1816	010004	020122	042524	052123	
1817	010012	047516	020040	041520	
1818	010020	040440	020124	041101	
1819	010026	051117	000124		
1820	010032	042515	020115	051105	DH3: .ASCII ?MEM ERR CONTROL HITMISS ?<CRLF>
1821	010040	020122	047503	052116	
1822	010046	047522	020114	044510	
1823	010054	046524	051511	020123	
1824	010062	100040			
1825	010064	042522	044507	052123	.ASCIZ ?REGISTR REGISTR REGISTR TESTNO PC AT ABORT?

1826	010072	020122	042522	044507					
1827	010100	052123	020122	042522					
1828	010106	044507	052123	020122					
1829	010114	042524	052123	047516					
1830	010122	020040	041520	040440					
1831	010130	020124	041101	051117					
1832	010136	000124							
1833	010140	051105	047522	020122	DH5:	.ASCII	?ERROR	VIRTUAL?<CRLF>	
1834	010146	020040	044526	052122					
1835	010154	040525	100114						
1836	010160	042522	044507	052123					
1837	010166	020122	042101	051104					
1838	010174	051505	020123	042524					
1839	010202	052123	047516	020040					
1840	010210	041520	040440	020124					
1841	010216	041101	051117	000124					
1842	010224	054105	042520	052103	DH6:	.ASCII	?EXPECTD ERROR	VIRTUAL?<CRLF>	
1843	010232	020104	051105	047522					
1844	010240	020122	020040	044526					
1845	010246	052122	040525	100114					
1846	010254	047503	042116	052111					
1847	010262	020116	042522	044507					
1848	010270	052123	020122	042101					
1849	010276	051104	051505	020123					
1850	010304	042524	052123	047516					
1851	010312	020040	041520	040440					
1852	010320	020124	041101	051117					
1853	010326	000124							
1854	010330	046450	051115	024460	DH7:	.ASCIZ	?(MMR0)	TESTNO ERRORPC?	
1855	010336	020040	042524	052123					
1856	010344	047516	020040	051105					
1857	010352	047522	050122	000103					
1858	010360	047514	042101	042105	DH11:	.ASCIZ	?LOADED RECEIVD	TESTNO ERRORPC?	
1859	010366	020040	042522	042503					
1860	010374	053111	020104	042524					
1861	010402	052123	047516	020040					
1862	010410	051105	047522	050122					
1863	010416	000103							
1864	010420	054105	042520	052103	DH12:	.ASCIZ	?EXPECTD (MMR2)	TESTNO ERRORPC?	
1865	010426	020104	046450	051115					
1866	010434	024462	020040	042524					
1867	010442	052123	047516	020040					
1868	010450	051105	047522	050122					
1869	010456	000103							
1870	010460	042101	051104	051117	DH13:	.ASCIZ	?ADDROR ADDRAND	TESTNO ERRORPC #ERRORS?	
1871	010466	020040	042101	051104					
1872	010474	047101	020104	042524					
1873	010502	052123	047516	020040					
1874	010510	051105	047522	050122					
1875	010516	020103	021440	051105					
1876	010524	047522	051522	000					
1877	010531	101	042104	042522	DH14:	.ASCIZ	?ADDRESS DATA	TESTNO ERRORPC?	
1878	010536	051523	042040	052101					
1879	010544	020101	020040	052040					
1880	010552	051505	047124	020117					
1881	010560	042440	051122	051117					

1882	010566	041520	000							
1883	010571	101	042104	047522	DH15:	.ASCII	?ADDROR	ADDRAND	ADDROR	ADDRAND?<CRLF>
1884	010576	020122	040440	042104						
1885	010604	040522	042116	040440						
1886	010612	042104	047522	020122						
1887	010620	040440	042104	040522						
1888	010626	042116	200							
1889	010631	114	040517	042504		.ASCIZ	?LOADED	LOADED	ENABLED	ENABLED TESTNO ERRORPC #ERRORS?
1890	010636	020104	046040	040517						
1891	010644	042504	020104	042440						
1892	010652	040516	046102	042105						
1893	010660	042440	040516	046102						
1894	010666	042105	052040	051505						
1895	010674	047124	020117	051105						
1896	010702	047522	050122	020103						
1897	010710	042443	051122	051117						
1898	010716	000123								
1899	010720	042101	051104	051117	DH16:	.ASCIZ	?ADDROR	ADDRAND	PATRNOR	PATRNAD DATAOR DATAND TESTNO ERRORPC #ERRORS?
1900	010726	020040	042101	051104						
1901	010734	047101	020104	040520						
1902	010742	051124	047516	020122						
1903	010750	040520	051124	040516						
1904	010756	020104	040504	040524						
1905	010764	051117	020040	040504						
1906	010772	040524	042116	020040						
1907	011000	042524	052123	047516						
1908	011006	042440	051122	051117						
1909	011014	041520	021440	051105						
1910	011022	047522	051522	000						
1911	011027	101	042104	042522	DH17:	.ASCIZ	?ADDRESS	EXPECTD	RECEIVD	TESTNO ERRORPC?
1912	011034	051523	042440	050130						
1913	011042	041505	042124	051040						
1914	011050	041505	044505	042126						
1915	011056	052040	051505	047124						
1916	011064	020117	042440	051122						
1917	011072	051117	041520	000						
1918	011077	122	043505	042101	DH20:	.ASCIZ	?REGADDR	TESTNO	ERRORPC?	
1919	011104	051104	052040	051505						
1920	011112	047124	020117	042440						
1921	011120	051122	051117	041520						
1922	011126	000								
1923	011127	122	043505	042101	DH21:	.ASCIZ	?REGADDR	DATA	TESTNO	ERRORPC?
1924	011134	051104	042040	052101						
1925	011142	020101	020040	052040						
1926	011150	051505	047124	020117						
1927	011156	042440	051122	051117						
1928	011164	041520	000							
1929	011167	122	043505	042101	DH22:	.ASCIZ	?REGADR	EXPECTD	RECEIVD	TESTNO ERRORPC?
1930	011174	020122	042440	050130						
1931	011202	041505	042124	051040						
1932	011210	041505	044505	042126						
1933	011216	052040	051505	047124						
1934	011224	020117	051105	047522						
1935	011232	050122	000103							
1936	011236	042522	040507	042104	DH23:	.ASCIZ	?REGADDR	DATAREC	DATAEXP	TESTNO ERRORPC?
1937	011244	020122	040504	040524						

1938	011252	042522	020103	040504					
1939	011260	040524	054105	020120					
1940	011266	042524	052123	047516					
1941	011274	020040	051105	047522					
1942	011302	050122	000103						
1943	011306	051513	020120	020040	DH27:	.ASCIZ	?KSP	TESTNO	ERRORPC?
1944	011314	020040	042524	052123					
1945	011322	047516	020040	051105					
1946	011330	047522	050122	000103					
1947	011336	047111	042504	020130	DH30:	.ASCII	?INDEX	INDEX	PAR/PDR?<CRLF>
1948	011344	020040	047111	042504					
1949	011352	020130	020040	040520					
1950	011360	027522	042120	100122					
1951	011366	054105	042520	052103		.ASCIZ	?EXPECTD	RECEIVD	ADRREAD TESTNO ERRORPC?
1952	011374	020104	042522	042503					
1953	011402	053111	020104	042101					
1954	011410	051122	040505	020104					
1955	011416	042524	052123	047516					
1956	011424	020040	051105	047522					
1957	011432	050122	000103						
1958	011436	044526	052122	040525	DH31:	.ASCII	?VIRTUAL	PHYSICAL DATA	DATA?<CRLF>
1959	011444	020114	020040	020040					
1960	011452	020040	020040	044120					
1961	011460	051531	040503	020114					
1962	011466	040504	040524	020040					
1963	011474	020040	040504	040524					
1964	011502	200							
1965	011503	101	042104	042522		.ASCIZ	?ADDRESS	KIPAR4	ADDRESS EXPECTD RECEIVD TESTNO ERRORPC?
1966	011510	051523	045440	050111					
1967	011516	051101	020064	040440					
1968	011524	042104	042522	051523					
1969	011532	042440	050130	041505					
1970	011540	042124	051040	041505					
1971	011546	044505	042126	052040					
1972	011554	051505	047124	020117					
1973	011562	051105	047522	050122					
1974	011570	000103							
1975	011572	052123	052122	042101	DH32:	.ASCIZ	?STRTADR	FNSHADR	TESTNO ERRORPC?
1976	011600	020122	047106	044123					
1977	011606	042101	020122	042524					
1978	011614	052123	047516	020040					
1979	011622	051105	047522	050122					
1980	011630	000103							
1981	011632	040520	052124	051105	DH33:	.ASCII	?PATTERN	DATA	ADDRESS?<CRLF>
1982	011640	020116	040504	040524					
1983	011646	020040	020040	042101					
1984	011654	051104	051505	100123					
1985	011662	047514	042101	042105		.ASCIZ	?LOADED	FETCHED	INTENDD KIPAR4 KIPAR5 TESTNO ERRORPC?
1986	011670	020040	042506	041524					
1987	011676	042510	020104	047111					
1988	011704	042524	042116	020104					
1989	011712	044513	040520	032122					
1990	011720	020040	044513	040520					
1991	011726	032522	020040	042524					
1992	011734	052123	047516	020040					
1993	011742	051105	047522	050122					

2050	012432	020051	024040	046515						
2051	012440	031122	020051	052040						
2052	012446	051505	047124	020117						
2053	012454	042440	051122	051117						
2054	012462	041520	000							
2055	012465	117	044522	044507	DH50:	.ASCII	?ORIGINAL DATA		NEW DATA?<CRLF>	
2056	012472	040516	020114	040504						
2057	012500	040524	004411	047040						
2058	012506	053505	042040	052101						
2059	012514	100101								
2060	012516	046450	051115	024460		.ASCIZ	?(MMR0) (MMR2) (MMR0) (MMR2)	TESTNO	ERRORPC?	
2061	012524	020040	046450	051115						
2062	012532	024462	020040	046450						
2063	012540	051115	024460	020040						
2064	012546	046450	051115	024462						
2065	012554	020040	042524	052123						
2066	012562	047516	020040	051105						
2067	012570	047522	050122	000103						
2068	012576	042522	042503	053111	DH52:	.ASCIZ	?RECEIVD (MMR2)	TESTNO	ERRORPC?	
2069	012604	020104	046450	051115						
2070	012612	024462	020040	042524						
2071	012620	052123	047516	020040						
2072	012626	051105	047522	050122						
2073	012634	000103								
2074	012636	044526	052122	042101	DH53:	.ASCIZ	?VIRTADR (MMR2)	TESTNO	ERRORPC?	
2075	012644	020122	046450	051115						
2076	012652	024462	020040	042524						
2077	012660	052123	047516	020040						
2078	012666	051105	047522	050122						
2079	012674	000103								
2080	012676	054105	042520	052103	DH54:	.ASCIZ	?EXPECTD (MMR0)	TESTNO	ERRORPC?	
2081	012704	020104	046450	051115						
2082	012712	024460	020040	042524						
2083	012720	052123	047516	020040						
2084	012726	051105	047522	050122						
2085	012734	000103								
2086	012736	044513	042120	032122	DH55:	.ASCIZ	?KIPDR4	TESTNO	ERRORPC?	
2087	012744	020040	042524	052123						
2088	012752	047516	020040	051105						
2089	012760	047522	050122	000103						
2090	012766	044513	042120	033522	DH60:	.ASCIZ	?KIPDR7	TESTNO	ERRORPC?	
2091	012774	020040	042524	052123						
2092	013002	047516	020040	051105						
2093	013010	047522	050122	000103						
2094	013016	042107	040520	042507	DH62:	.ASCIZ	?GDPAGE BDPAGE	TESTNO	ERRORPC?	
2095	013024	020040	042102	040520						
2096	013032	042507	020040	042524						
2097	013040	052123	047516	020040						
2098	013046	051105	047522	050122						
2099	013054	000103								
2100	013056	051524	050124	043501	DH63:	.ASCIZ	?TSTPAGE CONTENT	TESTNO	ERRORPC?	
2101	013064	020105	047503	052116						
2102	013072	047105	020124	042524						
2103	013100	052123	047516	020040						
2104	013106	051105	047522	050122						
2105	013114	000103								

2106	013116	054105	042520	052103	DH64:	.ASCIZ	?EXPECTD RECEIVD TESTNO	ERRORPC?	
2107	013124	020104	042522	042503					
2108	013132	053111	020104	042524					
2109	013140	052123	047516	020040					
2110	013146	051105	047522	050122					
2111	013154	000103							
2112	013156	054105	042520	052103	DH66:	.ASCIZ	?EXFECTD RECEIVD TESTNO	ERRORPC?	
2113	013164	020104	042522	042503					
2114	013172	053111	020104	042524					
2115	013200	052123	047516	020040					
2116	013206	051105	047522	050122					
2117	013214	000103							
2118	013216	046450	051115	024460	DH67:	.ASCIZ	?(MMR0) (MMR2) TESTNO	ERRORPC?	
2119	013224	020040	046450	051115					
2120	013232	024462	020040	042524					
2121	013240	052123	047516	020040					
2122	013246	051105	047522	050122					
2123	013254	000103							
2124	013256	052123	050113	051124	DH70:	.ASCIZ	?STKPTR TESTNO	ERRORPC?	
2125	013264	020040	042524	052123					
2126	013272	047516	020040	051105					
2127	013300	047522	050122	000103					
2128	013306	042107	040504	040524	DH71:	.ASCIZ	?GDDATA STORED TESTNO	ERRORPC?	
2129	013314	020040	052123	051117					
2130	013322	042105	020040	042524					
2131	013330	052123	047516	020040					
2132	013336	051105	047522	050122					
2133	013344	000103							
2134	013346	050050	053523	020051	DH72:	.ASCIZ	?(PSW) TESTNO	ERRORPC EXPECTING XXX340?	
2135	013354	020040	042524	052123					
2136	013362	047516	020040	051105					
2137	013370	047522	050122	020103					
2138	013376	054105	042520	052103					
2139	013404	047111	020107	054130					
2140	013412	031530	030064	000					
2141	013417	050	051520	024527	DH73:	.ASCIZ	?(PSW) TESTNO	ERRORPC?	
2142	013424	020040	052040	051505					
2143	013432	047124	020117	042440					
2144	013440	051122	051117	041520					
2145	013446	000							
2146	013447	124	051505	047124	DH74:	.ASCIZ	?TESTNO	ERRORPC?	
2147	013454	020117	051105	047522					
2148	013462	050122	000103						
2149	013466	052123	050113	051124	DH75:	.ASCIZ	?STKPTR STKLMT CPUERR	TESTNO	ERRORPC?
2150	013474	020040	052123	046113					
2151	013502	052115	020040	050103					
2152	013510	042525	051122	020040					
2153	013516	042524	052123	047516					
2154	013524	020040	051105	047522					
2155	013532	050122	000103						
2156									
2157									
2158	013536	042101	051104	051505	DH201:	.ASCIZ	?ADDRESS TESTNO	ERRORPC?	
2159	013544	020123	042524	052123					
2160	013552	047516	042440	051122					
2161	013560	051117	041520	000					

2162	013565	101	042104	042522	DH202:	.ASCII	?ADDRESS ADDRESS?<CRLF>
2163	013572	051523	040440	042104			
2164	013600	042522	051523	200			
2165	013605	114	040517	042504		.ASCIZ	?LOADED JUSTREAD TESTNO ERRORPC?
2166	013612	020104	045040	051525			
2167	013620	051124	040505	020104			
2168	013626	042524	052123	047516			
2169	013634	042440	051122	051117			
2170	013642	041520	000				
2171	013645	101	042104	042522	DH203:	.ASCIZ	?ADDRESS RECEIVD EXPECTD COUNT TESTNO ERRORPC?
2172	013652	051523	051040	041505			
2173	013660	044505	042126	042440			
2174	013666	050130	041505	042124			
2175	013674	041440	052517	052116			
2176	013702	020040	052040	051505			
2177	013710	047124	020117	051105			
2178	013716	047522	050122	000103			
2179						.EVEN	
2180							
2181							
2182	013724	001412	001410	000000	DT1:	.WORD	TESTNO,BADPC,0
2183	013732	001406	001412	001410	DT2:	.WORD	PCPUER,TESTNO,BADPC,0
2184	013740	000000					
2185	013742	001374	001376	001400	DT3:	.WORD	PPARER,PCONTR,PHITMI,TESTNO,BADPC,0
2186	013750	001412	001410	000000			
2187	013756	001402	001404	001412	DT5:	.WORD	PMMR0,PMMR2,TESTNO,BADPC,0
2188	013764	001410	000000				
2189	013770	001364	001402	001404	DT6:	.WORD	MMEXP,PMMR0,PMMR2,TESTNO,BADPC,0
2190	013776	001412	001410	000000			
2191	014004	001164	001412	001116	DT7:	.WORD	\$REG1,TESTNO,\$ERRPC,0
2192	014012	000000					
2193	014014	001166	001164	001412	DT11:	.WORD	\$REG2,\$REG1,TESTNO,\$ERRPC,0
2194	014022	001116	000000				
2195	014026	001170	001166	001412	DT12:	.WORD	\$REG3,\$REG2,TESTNO,\$ERRPC,0
2196	014034	001116	000000				
2197	014040	001424	001426	001412	DT13:	.WORD	ADDROR,ADRAND,TESTNO,\$ERRPC,\$TMP5,0
2198	014046	001116	001210	000000			
2199	014054	001162	001166	001412	DT14:	.WORD	\$REG0,\$REG2,TESTNO,\$ERRPC,0
2200	014062	001116	000000				
2201	014066	001424	001426	001414	DT15:	.WORD	ADDROR,ADRAND,DATAOR,DATAND,TESTNO,\$ERRPC,\$TMP5,0
2202	014074	001416	001412	001116			
2203	014102	001210	000000				
2204	014106	001424	001426	001420	DT16:	.WORD	ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,TESTNO,\$ERRPC,\$TMP5,0
2205	014114	001422	001414	001416			
2206	014122	001412	001116	001210			
2207	014130	000000					
2208	014132	001162	001166	001164	DT17:	.WORD	\$REG0,\$REG2,\$REG1,TESTNO,\$ERRPC,0
2209	014140	001412	001116	000000			
2210	014146	001162	001412	001116	DT20:	.WORD	\$REG0,TESTNO,\$ERRPC,0
2211	014154	000000					
2212	014156	001162	001164	001412	DT21:	.WORD	\$REG0,\$REG1,TESTNO,\$ERRPC,0
2213	014164	001116	000000				
2214	014170	001206	001210	001162	DT22:	.WORD	\$TMP4,\$TMP5,\$REG0,TESTNO,\$ERRPC,0
2215	014176	001412	001116	000000			
2216	014204	001162	001164	001166	DT23:	.WORD	\$REG0,\$REG1,\$REG2,TESTNO,\$ERRPC,0
2217	014212	001412	001116	000000			

2218	014220	001162	001412	001116	DT27:	.WORD	\$REGO, TESTNO, \$ERRPC, 0
2219	014226	000000					
2220	014230	001162	001164	001166	DT30:	.WORD	\$REGO, \$REG1, \$REG2, TESTNO, \$ERRPC, 0
2221	014236	001412	001116	000000			
2222	014244	001162	001200	001202	DT31:	.WORD	\$REGO, \$TMP1, \$TMP2, \$REG1, \$REG2, TESTNO, \$ERRPC, 0
2223	014252	001164	001166	001412			
2224	014260	001116	000000				
2225	014264	001200	001202	001412	DT32:	.WORD	\$TMP1, \$TMP2, TESTNO, \$ERRPC, 0
2226	014272	001116	000000				
2227	014276	001166	001170	001162	DT33:	.WORD	\$REG2, \$REG3, \$REGO, \$TMP3, \$TMP4, TESTNO, \$ERRPC, 0
2228	014304	001204	001206	001412			
2229	014312	001116	000000				
2230	014316	001164	001412	001116	DT34:	.WORD	\$REG1, TESTNO, \$ERRPC, 0
2231	014324	000000					
2232	014326	001164	001170	001412	DT35:	.WORD	\$REG1, \$REG3, TESTNO, \$ERRPC, 0
2233	014334	001116	000000				
2234	014340	001176	001200	001202	DT37:	.WORD	\$TMP0, \$TMP1, \$TMP2, TESTNO, \$ERRPC, 0
2235	014346	001412	001116	000000			
2236	014354	001166	001170	001412	DT41:	.WORD	\$REG2, \$REG3, TESTNO, \$ERRPC, 0
2237	014362	001116	000000				
2238	014366	001166	001170	001164	DT47:	.WORD	\$REG2, \$REG3, \$REG1, PMMR0, PMMR2, TESTNO, \$ERRPC, 0
2239	014374	001402	001404	001412			
2240	014402	001116	000000				
2241	014406	001402	001404	001176	DT50:	.WORD	PMMR0, PMMR2, \$TMP0, \$TMP2, TESTNO, \$ERRPC, 0
2242	014414	001202	001412	001116			
2243	014422	000000					
2244	014424	001402	001404	001412	DT52:	.WORD	PMMR0, PMMR2, TESTNO, \$ERRPC, 0
2245	014432	001116	000000				
2246	014436	001164	001404	001412	DT53:	.WORD	\$REG1, PMMR2, TESTNO, \$ERRPC, 0
2247	014444	001116	000000				
2248	014450	001166	001402	001412	DT54:	.WORD	\$REG2, PMMR0, TESTNO, \$ERRPC, 0
2249	014456	001116	000000				
2250	014462	001176	001412	001116	DT55:	.WORD	\$TMP0, TESTNO, \$ERRPC, 0
2251	014470	000000					
2252	014472	001170	001164	001412	DT62:	.WORD	\$REG3, \$REG1, TESTNO, \$ERRPC, 0
2253	014500	001116	000000				
2254	014504	001170	001162	001412	DT63:	.WORD	\$REG3, \$REGO, TESTNO, \$ERRPC, 0
2255	014512	001116	000000				
2256	014516	001166	001164	001412	DT64:	.WORD	\$REG2, \$REG1, TESTNO, \$ERRPC, 0
2257	014524	043352	000000				
2258	014530	001162	001164	001412	DT66:	.WORD	\$REGO, \$REG1, TESTNO, \$ERRPC, 0
2259	014536	001116	000000				
2260	014542	001402	001404	001412	DT67:	.WORD	PMMR0, PMMR2, TESTNO, \$ERRPC, 0
2261	014550	001116	000000				
2262	014554	001164	001412	001116	DT70:	.WORD	\$REG1, TESTNO, \$ERRPC, 0
2263	014562	000000					
2264	014564	001162	001412	001116	DT72:	.WORD	\$REGO, TESTNO, \$ERRPC, 0
2265	014572	000000					
2266	014574	001412	001116	000000	DT74:	.WORD	TESTNO, \$ERRPC, 0
2267	014602	001166	001170	001406	DT75:	.WORD	\$REG2, \$REG3, PCPUER, TESTNO, \$ERRPC, 0
2268	014610	001412	001116	000000			
2269							
2270	014616	001162	001412	001116	DT201:	.WORD	\$REGO, TESTNO, \$ERRPC, 0
2271	014624	000000					
2272	014626	001162	001164	001412	DT202:	.WORD	\$REGO, \$REG1, TESTNO, \$ERRPC, 0
2273	014634	001116	000000				

2274	014640	001162	001166	001172	DT203:	.WORD	\$REG0,\$REG2,\$REG4,\$REG1,TESTNO,\$ERRPC,0
2275	014646	001164	001412	001116			
2276	014654	000000					
2277							
2278							
2279	014656	000	000		DF1:	.BYTE	0,0
2280	014660	000	000	000	DF2:	.BYTE	0,0,0
2281	014663	000	000	000	DF3:	.BYTE	0,0,0,0,0
2282	014666	000	000				
2283	014670	000	000	000	DF5:	.BYTE	0,0,0,0
2284	014673	000					
2285	014674	000	000	000	DF6:	.BYTE	0,0,0,0,0
2286	014677	000	000				
2287	014701	000	000	000	DF7:	.BYTE	0,0,0
2288	014704	000	000	000	DF11:	.BYTE	0,0,0,0
2289	014707	000					
2290	014710	000	000	000	DF12:	.BYTE	0,0,0,0
2291	014713	000					
2292	014714	000	000	000	DF13:	.BYTE	0,0,0,0,1
2293	014717	000	001				
2294	014721	000	000	000	DF14:	.BYTE	0,0,0,0
2295	014724	000					
2296	014725	000	000	000	DF15:	.BYTE	0,0,0,0,0,0,1
2297	014730	000	000	000			
2298	014733	001					
2299	014734	000	000	000	DF16:	.BYTE	0,0,0,0,0,0,0,0,1
2300	014737	000	000	000			
2301	014742	000	000	001			
2302	014745	000	000	000	DF17:	.BYTE	0,0,0,0,0
2303	014750	000	000				
2304	014752	000	000	000	DF20:	.BYTE	0,0,0
2305	014755	000	000	000	DF21:	.BYTE	0,0,0,0
2306	014760	000					
2307	014761	000	000	000	DF22:	.BYTE	0,0,0,0,0
2308	014764	000	000				
2309	014766	000	000	000	DF23:	.BYTE	0,0,0,0,0
2310	014771	000	000				
2311	014773	000	000	000	DF27:	.BYTE	0,0,0
2312	014776	000	000	000	DF30:	.BYTE	0,0,0,0,0
2313	015001	000	000				
2314	015003	000	000	000	DF31:	.BYTE	0,0,0,0,0,0,0
2315	015006	000	000	000			
2316	015011	000					
2317	015012	002	002	000	DF32:	.BYTE	2,2,0,0
2318	015015	000					
2319	015016	000	000	000	DF33:	.BYTE	0,0,0,0,0,0,0
2320	015021	000	000	000			
2321	015024	000					
2322	015025	000	000	000	DF34:	.BYTE	0,0,0
2323	015030	000	000	000	DF35:	.BYTE	0,0,0,0
2324	015033	000					
2325	015034	000	000	000	DF37:	.BYTE	0,0,0,0,0
2326	015037	000	000				
2327	015041	000	000	000	DF41:	.BYTE	0,0,0,0
2328	015044	000					
2329	015045	000	000	000	DF47:	.BYTE	0,0,0,0,0,0,0

2330	015050	000	000	000				
2331	015053	000						
2332	015054	000	000	000	DF50:	.BYTE	0,0,0,0,0,0	
2333	015057	000	000	000				
2334	015062	000	000	000	DF52:	.BYTE	0,0,0,0	
2335	015065	000						
2336	015066	000	000	000	DF53:	.BYTE	0,0,0,0	
2337	015071	000						
2338	015072	000	000	000	DF54:	.BYTE	0,0,0,0	
2339	015075	000						
2340	015076	000	000	000	DF55:	.BYTE	0,0,0	
2341	015101	000	000	000	DF62:	.BYTE	0,0,0,0	
2342	015104	000						
2343	015105	000	000	000	DF63:	.BYTE	0,0,0,0	
2344	015110	000						
2345	015111	000	000	000	DF64:	.BYTE	0,0,0,0	
2346	015114	000						
2347	015115	000	000	000	DF66:	.BYTE	0,0,0,0	
2348	015120	000						
2349	015121	000	000	000	DF67:	.BYTE	0,0,0,0	
2350	015124	000						
2351	015125	000	000	000	DF70:	.BYTE	0,0,0	
2352	015130	000	000	000	DF72:	.BYTE	0,0,0	
2353	015133	000	000	000	DF74:	.BYTE	0,0	
2354	015135	000	000	000	DF75:	.BYTE	0,0,0,0,0	
2355	015140	000	000					
2356								
2357								
2358								
2359	015142	000	000	000	DF201:	.BYTE	0,0,0	
2360	015145	000	000	000	DF202:	.BYTE	0,0,0,0	
2361	015150	000						
2362	015151	000	000	000	DF203:	.BYTE	0,0,0,0,0,0	
2363	015154	000	000	000				
2364		015160				.EVEN		
2365								
2366								
2367								
2368								
2369								

2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425

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

.SBTTL CPU TRAP HANDLER ROUTINE

* THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS, THRU
* "ERRVEC" (000004). IF THIS SUBROUTINE IS ENTERED BY A SECOND
* TRAP BEFORE THE FIRST HAS BEEN PROCESSED A HALT IS EXECUTED.
* THE CPU ERROR REGISTER IS READ IF EXECUTING ON AN 11/6X.
* FOR EVERY CASE AN ERROR MESSAGE IS TYPED REPORTING AN
* UNEXPECTED TRAP OR ABORT TO LOC 4. 'PCPUER' CAN
* BE USED AS A FLAG TO INDICATE THAT A TRAP HAS OCCURRED SINCE IT
* IS LOADED WITH THE ERROR REGISTER IF A TRAP VECTORS HERE

CPUER: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME
CPFLAG: .WORD -1 ;NEGATIVE ONE FOR A FLAG
BEQ 10\$;BRANCH IF FIRST TIME IN
HALT ;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
10\$: MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF LOOP
MOV (KSP)+,OLDPS ;SAVE OLD PSW IN CASE OF LOOP
MOV OLDPC,BADPC ;SAVE PC+2 AT TIME OF ABORT
TSTB FORTY ;EXECUTING ON AN 11/40?
BEQ 2\$;BRANCH IF NO
ERROR 1 ;UNEXPECTED CPU TRAP OR ABORT
BR 1\$;BRANCH TO EXIT
2\$: MOV @#CPUERR,PCPUER ;SAVE CPU ERROR REGISTER ON AN 11/6X
ERROR 2 ;UNEXPECTED CPU TRAP OR ABORT
1\$: MOV #-1,CPFLAG ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
MOV OLDPS,-(KSP) ;PUSH OLD PSW BACK ON STACK
MOV OLDPC,-(KSP) ;PUSH RETURN ADDRESS BACK ON STACK
RTT ;RETURN FROM INTERRUPT OR ABORT

.SBTTL MEMORY/CACHE PARITY TRAPS AND ABORTS HANDLER ROUTINE

* THIS SUBROUTINE WILL HANDLE ALL UNEXPECTED PARITY ERRORS. IF
* THE PARITY ERROR IS AN ABORT THE TEST THAT WAS RUNNING WILL
* BE RESTARTED ONCE AFTER THE ABORT CONDITION IS REPORTED. ON THE
* SECOND ABORT IN A SINGLE TEST THE NEXT TEST IS ATTEMPTED
* AFTER THE ABORT IS REPORTED.

MEMER: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME
PAFLAG: .WORD -1 ;NEGATIVE ONE FOR A FLAG
BEQ 10\$;BRANCH IF FIRST TIME IN
HALT ;I HAVE ENTERED THIS ROUTINE BEFORE
; I FINISHED REPORTING THE FIRST ERROR

```

2426                                     ; THE SECOND ENTRY ADDRESS IS ON THE
2427                                     ; STACK AND THE FIRST ERROR CONDITION
2428                                     ; IS PROBABLY STILL LOCKED UP
2429 015260 012637 001434 10$: MOV (KSP)+,OLDPC ; SAVE RETURN ADDRESS IN CASE OF LOOP
2430 015264 012637 001436 MOV (KSP)+,OLDPS ; SAVE OLD PSW IN CASE OF LOOP
2431 015270 105737 001446 TSTB FORTY ; EXECUTING ON AN 11/40?
2432 015274 001026 BNE 6$ ; BRANCH IF YES
2433 015276 017737 164054 001374 MOV @MEMERR,PPARER ; SAVE MEMORY ERROR REGISTER
2434 015304 017737 164042 001376 MOV @CONTRL,PCONTR ; SAVE CONTROL REGISTER FOR TYPE OUT
2435 015312 017737 164036 001400 MOV @HITMIS,PHITMI ; SAVE HIT/MISS REGISTER
2436 015320 013737 001434 001410 MOV OLDPC,BADPC ; SAVE PC+2 AT TIME OF ABORT
2437 015326 032777 000014 164016 BIT #14,@CONTRL ; SEE IF CACHE IS ON OR OFF
2438 015334 001010 BNE 1$ ; IF OFF, MUST BE MEMORY REFERENCE
2439 015336 032777 000340 164030 BIT #340,@PPARER ; WAS THIS A CACHE PARITY ERROR
2440 015344 001004 BNE 1$ ; BRANCH IF IT WAS A MAIN MEMORY ERROR
2441 015346 104003 ERROR 3 ; UNEXPECTED CACHE PARITY ERROR
2442 015350 000403 BR 2$ ; BRANCH TO EXIT POINT
2443 015352 104077 6$: ERROR 77 ; UNEXPECTED MAIN MEM. PARITY ERR. 11/40
2444 015354 000401 BR 2$ ; BRANCH TO EXIT POINT
2445 015356 104004 1$: ERROR 4 ; UNEXPECTED MAIN MEMORY PARITY ERROR
2446 015360 005737 001442 2$: TST RETRY ; ARE YOU RETRYING THIS TEST?
2447 015364 001006 BNE 3$ ; BRANCH IF THIS IS SECOND TRY
2448 015366 005237 001442 INC RETRY ; SET RETRY FLAG
2449 015372 013737 001106 001434 MOV $LPADR,OLDPC ; RETURN TO START OF THIS TEST
2450 015400 000403 BR 4$ ; BRANCH TO EXIT
2451 015402 013737 001444 001434 3$: MOV NXTTST,OLDPC ; RETURN TO START OF NEXT TEST
2452                                     ; SINCE THIS IS THE SECOND ABORT
2453 015410 012737 177777 015252 4$: MOV #-1,PAFLAG ; RESTORE A NEGATIVE ONE FOR NEXT TIME
2454 015416 013746 001436 MOV OLDPS,-(KSP) ; PUSH OLD PSW BACK ON STACK
2455 015422 013746 001434 MOV OLDPC,-(KSP) ; PUSH RETURN ADDRESS BACK ON STACK
2456 015426 000006 5$: RTT ; RETURN FROM INTERRUPT.
2457
2458
2459
2460 .SBTTL MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER ROUTINE
2461 *****
2462 *
2463 * THIS SUBROUTINE WILL HANDLE MOST OF THE MEMORY MANAGEMENT TRAPS
2464 * AND ABORTS THAT ARE GENERATED DURING THIS PROGRAM. ANY M.M.
2465 * ABORTS OR TRAPS THAT OCCUR WHILE 'MMEXP' IS ZERO ARE UNEXPECTED
2466 * AND WILL BE REPORTED AS SUCH. THIS MAY OCCUR BECAUSE SOME LOGIC
2467 * THAT IS TO INHIBIT A TRAP HAS FAILED.
2468 * THERE ARE ALSO TIMES WHEN I AM EXPECTING A CERTAIN CONDITION TO
2469 * OCCUR, AND WHEN THIS CONDITION IN 'MMEXP' DOES NOT MATCH THE
2470 * CONTENTS OF 'PMMRO' (SAME AS 'MMRO') AN ERROR IS REPORTED.
2471 *
2472 *****
2473 015430 005227 MMTRAP: INC (PC)+ ; MAKE FLAG ZERO IF FIRST TIME
2474 015432 177777 MMFLAG: .WORD -1 ; FLAG SHOULD BE NEG ONE
2475 015434 001401 BEQ 10$ ; BRANCH IF FIRST TIME INTO ROUTINE
2476 015436 000000 HALT ; I HAVE ENTERED THIS ROUTINE BEFORE
2477 ; I FINISHED REPORTING THE FIRST ERROR
2478 ; THE SECOND ENTRY ADDRESS IS ON THE
2479 ; STACK AND THE FIRST ERROR CONDITION
2480 ; IS PROBABLY STILL LOCKED UP
2481 015440 011637 001410 10$: MOV (KSP),BADPC ; SAVE PC AT TIME OF ABORT OR TRAP

```

```

2482 015444 012637 001434      MOV      (KSP)+,OLDPC      ;SAVE RETURN ADDRESS IN CASE OF LOOP
2483 015450 012637 001436      MOV      (KSP)+,OLDPS     ;SAVE OLD PSW IN CASE OF LOOP
2484 015454 013737 177572 001402  MOV      MMR0,PMMR0       ;SAVE STATUS REGISTER
2485 015462 013737 177576 001404  MOV      MMR2,PMMR2       ;SAVE VIRTUAL ADDRESS REGISTER
2486 015470 005737 001364      TST      MMEXP            ;SEE IF ANY M.M. TRAPS WERE EXPECTED
2487 015474 001002                    BNE      5$               ;BRANCH IF ONE WAS EXPECTED
2488 015476 104005                    ERROR    5                ;UNEXPECTED M.M. ABORT OR TRAP
2489 015500 000405                    BR       1$               ;BRANCH TO EXIT
2490 015502 023737 001364 001402 5$:    CMP      MMEXP,PMMR0     ;SEE IF ABORT OR TRAP WAS EXPECTED
2491 015510 001401                    BEQ      1$               ;BRANCH IF CONDITION CORRECT
2492 015512 104006                    ERROR    6                ;ERROR TYPE OUT ITEM 6
2493 015514 042737 177376 177572 1$:    BIC      #177376,#MMR0   ;CLEAR ALL BITS EXCEPT 0 AND 8
2494 015522 012737 177777 015432  MOV      #-1,MMFLAG      ;RESTORE A NEGATIVE ONE TO FLAG
2495 015530 013746 001436      MOV      OLDPS,-(KSP)    ;PUSH OLD PSW ONTO STACK
2496 015534 013746 001434      MOV      OLDPC,-(KSP)    ;PUSH RETURN ADDRESS ON STACK
2497 015540 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 MMR0
* AND MMR2 AND RETURNS TO THE TEST WHERE THEY ARE TESTED.
*
*****

```

```

2514 015542 013700 177776      KERVEC: MOV      PSW,R0      ;PUT PROCESSOR STATUS INTO R0
2515 015546 012637 001434      MOV      (KSP)+,OLDPC    ;SAVE RETURN ADDRESS
2516 015552 012637 001436      MOV      (KSP)+,OLDPS    ;SAVE OLD PROCESSOR STATUS
2517 015556 122700 000340      CMPB    #340,R0         ;SEE IF CORRECT PSW WAS PICKED UP
2518 015562 001401                    BEQ      1$               ;BRANCH IF PSW IS CORRECT
2519 015564 104072                    ERROR    72              ;WRONG PSW PICKED UP
2520 015566 000137 015610 1$:    JMP      RDMMR0         ;JUMP TO READ MMR0
2521
2522 015572 012637 001434      USEVEC: MOV      (KSP)+,OLDPC ;SAVE RETURN ADDRESS
2523 015576 012637 001436      MOV      (KSP)+,OLDPS    ;SAVE OLD PROCESSOR STATUS
2524 015602 013700 177776      MOV      PSW,R0         ;READ PSW FOR ERROR PRINT OUT
2525 015606 104073                    ERROR    73              ;WRONG VECTOR, POSSIBLE WRONG PSW
2526
2527 015610 013737 177572 001402  RDMMR0: MOV      MMR0,PMMR0 ;READ MMR0 TO BE CHECKED LATER
2528 015616 013737 177576 001404  MOV      MMR2,PMMR2     ;READ MMR2 FOR POSSIBLE ERROR REPORT
2529 015624 013746 001436      MOV      OLDPS,-(KSP)   ;PUSH OLD PROCESSOR STATUS ON STACK
2530 015630 013746 001434      MOV      OLDPC,-(KSP)   ;PUSH RETURN ADDRESS ON STACK
2531 015634 000006      RTT                      ;RETURN TO TEST TO CHECK PMMR0
2532
2533
2534

```

```

2535 .SBTTL
2536 .SBTTL ***** ENTRY POINT 1 --- STARTING ADDRESS 200 *****
2537 .SBTTL ***** TEST CODE STARTS AT ADDRESS 20000 (PAGE 1) *****
2538 020000
2539
2540 020000 012737 000000 001360 STRT1: MOV #0,FSTTST ;LOAD INDEX TO START TABLE
2541 020006 000427 BR START ;BRANCH TO STARTING ADDRESS OF PROGRAM
2542 020010 012737 000002 001360 STRT2: MOV #2,FSTTST ;LOAD INDEX TO START TABLE
2543 020016 000423 BR START ;BRANCH TO STARTING ADDRESS OF PROGRAM
2544 020020 012737 000004 001360 STRT3: MOV #4,FSTTST ;LOAD INDEX TO START TABLE
2545 020026 000417 BR START ;BRANCH TO STARTING ADDRESS OF PROGRAM
2546 020030 012737 000006 001360 STRT4: MOV #6,FSTTST ;LOAD INDEX TO START TABLE
2547 020036 000413 BR START ;BRANCH TO STARTING ADDRESS OF PROGRAM
2548 020040 012737 000010 001360 STRT5: MOV #10,FSTTST ;LOAD INDEX TO START TABLE
2549 020046 000407 BR START ;BRANCH TO STARTING ADDRESS OF PROGRAM
2550 020050 012737 000012 001360 STRT6: MOV #12,FSTTST ;LOAD INDEX TO START TABLE
2551 020056 000403 BR START ;BRANCH TO STARTING ADDRESS OF PROGRAM
2552 020060 012737 000014 001360 STRT7: MOV #14,FSTTST ;LOAD INDEX TO START TABLE
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567 020066 012706 001100 START: MOV #KERSTK,KSP ;SETUP STACK POINTER
2568 020072 012737 020112 000010 MOV #15,2#10 ;SETUP ILLEGAL INST. VECTOR
2569 020100 105037 001446 CLRB FORTY ;CLEAR 11/40 "FLAG LOC."
2570 020104 076600 MED ;TRY 11/6X INSTRUCTION
2571 020106 000100 RLJAM ;WILL TRAP TO 1$ IF ON AN 11/40
2572 020110 000410 BR 3$ ;WILL BRANCH TO SETUP IF ON 11/6X
2573 020112 062706 000004 1$: ADD #4,KSP ;CLEAN UP STACK IF ON 11/40 AND
2574 020116 105237 001446 INCB FORTY ;SET 11/40 "FLAG LOC."
2575 020122 012737 000012 000010 MOV #12,2#10 ;RESTORE TRAP CATCHER TO LOC. 10
2576 020130 000413 BR 4$ ;BRANCH TO SETUP
2577 020132 012777 000014 161212 3$: MOV #14,2CONTRL ;TURN 11/6X CACHE OFF FOR FIRST PASS
2578 020140 052700 100001 BIS #100001,R0 ;XXXXXX
2579 020144 076600 MED
2580 020146 000305 WLWHAM
2581 020150 052700 100001 BIS #100001,R0
2582 020154 076600 MED
2583 020156 000222 WWHAM
2584 020160
2585
2586
2587 .SBTTL INITIALIZE THE COMMON TAGS
2588 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
2589 MOV #SCMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
2590 CLR (R6)+ ;;CLEAR MEMORY LOCATION
2591 CMP #SWR,R6 ;;DONE?
2592 BNE .-6 ;;LOOP BACK IF NO

```

```

;* NOTE: THE FIRST THING THAT IS DONE IS TO SEE IF THE
;* 11/40 OR THE 11/6X MEMORY MGMT. IS BEING TESTED
;* AND IF ON AN 11/40 IF THERE IS A STACK LIMIT
;* REGISTER. THE 11/6X "MED" INSTRUCTION IS USED
;* TO DETERMINE PROCESSOR TYPE AND THE STACK LIMIT
;* REG. IS CHECKED FOR TIMEOUT.
;*
;* IF A "MED" DOES NOT EXECUTE PROPERLY ON AN 11/6X
;* OR IF THE STACK LIMIT REG. TIMES OUT ON AN 11/40 -
;* ERRONEOUS DECISIONS WILL BE MADE DURING TESTING
;* WHICH WILL RESULT IN INCOMPLETE TESTING.

```



```

2647 020532 001424 BEQ 71$ ;:BRANCH IF YES
2648 020534 104401 020542 TYPE 72$ ;:TYPE ASCIZ STRING
2649 020540 000421 BR 71$ ;:GET OVER THE ASCIZ
2650 ;:72$: .ASCIZ <CRLF>?CQKTA-BO 11/6X MEM. MGMT. DIAG.?<CRLF>
2651 ;:1$:
2652 020604 012737 020604 001110 MOV #.,$LPERR ;:INITIALIZE LOOP ON ERROR PTR.
2653 020612 012737 020612 001106 MOV #.,$LPADR ;:INITIALIZE LOOP ADDRESS PTR.
2654
2655
2656 020620 005737 001234 LOOP: TST $PASS ;:IS THIS THE FIRST PASS?
2657 020624 001406 BEQ 5$ ;:BRANCH IF FIRST PASS
2658 020626 105737 001446 TSTB FORTY ;:EXECUTING ON AN 11/40?
2659 020632 001003 BNE 5$ ;:BRANCH IF YES
2660 020634 012777 000200 160510 MOV #200,$CONTRL ;:OTHERWISE TURN CACHE ON AND SET THE
2661 ;:PARITY ABORT BIT
2662 020642 005037 177572 5$: CLR MMRO ;:GET INTO 16 BIT MODE ON SECOND PASS
2663 020646 004737 040570 JSR PC,CLRUW ;:INITIALIZE U REG'S
2664 020652 012700 177777 MOV #-1,RO ;:PUT NEG ONE IN RO TO INITIALIZE FLAGS
2665 020656 010037 015162 MOV RO,$PFLAG ;:INITIALIZE CPU ERROR FLAG
2666 020662 010037 015252 MOV RO,$PAFLAG ;:INITIALIZE PARITY ERROR FLAG
2667 020666 010037 015432 MOV RO,$MFLAG ;:INITIALIZE MEMORY MANAGEMENT TRAP FLAG
2668 020672 005037 001362 CLR CPUEXP ;:NOT EXPECTING ANY CPU TRAPS
2669 020676 005037 001364 CLR MMEXP ;:NOT EXPECTING ANY MEMORY MANAGEMENT TRAPS
2670 020702 012737 015160 000004 MOV #CPUER,ERRVEC ;:LOAD ADDRESS OF CPU TRAP ROUTINE
2671 020710 012737 000340 000006 MOV #340,ERRVEC+2 ;:SET PRIORITY LEVEL 7
2672 020716 012737 015250 000114 MOV #MEMER,CACHVEC ;:LOAD ADDRESS OF PARITY TRAP ROUTINE
2673 020724 012737 000340 000116 MOV #340,CACHVEC+2 ;:SET PRIORITY LEVEL 7
2674 020732 012737 015430 000250 MOV #MMTRAP,MMVEC ;:LOAD ADDRESS OF MEMORY MANAGEMENT TRAP
2675 020740 012737 000340 000252 MOV #340,MMVEC+2 ;:SET PRIORITY LEVEL 7
2676 020746 004737 044254 JSR PC,CLEANUP ;:INITIALIZE ALL ERROR LOCATIONS
2677 020752 012706 001100 MOV #KERSTK,KSP ;:SET UP KERNEL STACK POINTER
2678 020756 013700 001360 MOV FSTTST,RO ;:LOAD START TABLE INDEX
2679 020762 001412 BEQ TST1 ;:START WITH TEST ONE IF ZERO
2680 020764 012737 140000 177776 MOV #140000,$PSW ;:GO TO USER MODE
2681 020772 012706 000700 MOV #USESTK,$USP ;:SET UP USER STACK POINTER
2682 020776 012737 000340 177776 MOV #340,$PSW ;:GO TO KERNEL MODE PRIORITY LEVEL 7
2683 021004 000170 001460 JMP @STRTAB(RO) ;:JUMP TO CORRECT ENTRY POINT

```

```

2684
2685
2686 ;: * THIS FIRST GROUP OF TESTS IS FOR TESTING THE ADDRESS DECODE
2687 ;: * LOGIC FOR THE INTERNAL REGISTERS AND TO MAKE SOME
2688 ;: * DETERMINATION ABOUT THE RESPONDING ADDRESSES. IF AN
2689 ;: * ADDRESS DOES NOT FUNCTION AS THE CORRESPONDING REGISTER SHOULD
2690 ;: * AN ERROR WILL BE FLAGGED. THE MULTIPLEXERS AND INTERNAL BUS
2691 ;: * DRIVERS ARE ALSO UTILIZED IN THESE TESTS AND COULD BE
2692 ;: * THE SOURCE OF ANY PROBLEMS ENCOUNTERED.
2693
2694
2695 ;: *****

```

```

2696 ;: *TEST 1 TRY TO READ ALL CPU REGISTERS
2697 ;: *
2698 ;: * THIS TEST ENSURES THAT SOMETHING RESPONDS TO SOME INTERNAL ADDRESSES.
2699 ;: * THIS PROCESSOR STATUS WORD AND STACK LIMIT REG. (IF
2700 ;: * AVAILABLE) ARE TESTED IF EXECUTING ON AN 11/40. THE CPU
2701 ;: * ERROR REG. 2 MICROBREAK REG. ARE ALSO CHECKED IF ON AN 11/6X.
2702 ;: * IF A TRAP TO 'ERRVEC' (004) OCCURS IT IS ASSUMED

```

```

2703
2704
2705
2706
2707
2708
2709 021010
2710 021010 012737 021040 001110
2711 021016 012737 000001 001102
2712 021024 013777 001102 160110
2713 021032 012737 021142 001444
2714
2715 021040 012737 021136 000004
2716 021046 012706 001100
2717 021052 012700 177776
2718 021056 005737 177776
2719 021062 012700 177774
2720 021066 005737 177774
2721
2722 021072 105737 001446
2723 021076 001010
2724 021100 012700 177766
2725 021104 005737 177766
2726 021110 012700 177770
2727 021114 005737 177770
2728
2729 021120
2730 021120 012737 021040 001110
2731 021126 012737 015160 000004
2732 021134 000402
2733
2734 021136 104020
2735 021140 000006
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758

```

```

;* 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$".
*****
TST1:
MOV #20$, $LPERR ; SET LOOP ON ERROR POINTER TO 20$
MOV #1, $TSTNM ; LOAD TEST NUMBER INTO MEMORY
MOV $TSTNM, @DISPLAY ; DISPLAY TEST NUMBER FOR THIS TEST
MOV #TST2, $NXTTST ; SAVE STARTING ADDRESS OF NEXT TEST
; FOR ESCAPE ON PARITY ERRORS
20$: MOV #50$, ERRVEC ; SET TIME OUT VECTOR TO SPECIAL ROUTINE
MOV #1100, KSP ; SET KERNEL STACK POINTER TO 1100
1$: MOV #PSW, R0 ; SAVE ADDRESS IN CASE OF TIMEOUT
TST @#PSW ; TRY TO READ PROCESSOR STATUS WORD
MOV #STKLMT, R0 ; SAVE ADDRESS IN CASE OF TIMEOUT
TST @#STKLMT ; TRY TO READ STACK LIMIT REG.
; EXECUTING ON AN 11/40?
TSTB FORTY
BNE 2$ ; BRANCH IF YES
MOV #CPUERR, R0 ; SAVE ADDRESS IN CASE OF TIMEOUT
TST @#CPUERR ; TRY TO READ CPU ERROR REG.
MOV #177770, R0 ; SAVE ADDRESS IN CASE OF TIMEOUT
TST @#177770 ; TRY TO READ MICROBREAK REG.
2$: MOV #20$, $LPERR ; SET LOOP POINTER TO START OF TEST
MOV #CPUER, ERRVEC ; SET UP C.P.U. ERROR VECTOR
BR TST2 ; TEST OVER BRANCH TO NEXT TEST
50$: ERROR 20 ; CPU REGISTER TIMED OUT
RTT ; GO BACK AND FINISH TEST

```

```

*****
*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 EXECUTED 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.

```

2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814

021142
021142
021144
021152
021154
021162
021166
021172
021174
021176
021202
021204
021206
021210
021212
021220
021224
021226
021230
021232
021236
021240
021242
021244
021246
021250
021256
021262
021264
021266
021270
021272
021274
021276
021304
021306
021310
021312
021314
021316
021324
021326
021332
021334
021336
021340
021342
021344

000004
012737
021352
001444
104410
012737
021166
177774
000777
011001
005010
012702
000400
020102
001401
104026
005200
021220
001110
000001
112710
000001
005300
011001
005010
012702
000400
020102
001403
005200
104025
005300
012737
021262
001110
000400
010210
011001
020201
001402
005010
104025
021276
001110
005010
011001
001401
104021
006302
012737
021262
001110
103356
012710
177400
000005
011001
001402
005010
104021
021154
001110

↑ST2:
SCOPE
MOV #TST3,NXTTST
TBITO
MOV #1\$, \$LPERR
MOV #177774, R0
MOV #777, (R0)
MOV (R0), R1
CLR (R0)
MOV #400, R2
CMP R1, R2
BEQ 2\$
ERROR 26
INC R0
MOV #3\$, \$LPERR
MOVB #1, (R0)
DEC R0
MOV (R0), R1
CLR (R0)
MOV #400, R2
CMP R1, R2
BEQ 4\$
INC R0
ERROR 25
DEC R0
MOV #5\$, \$LPERR
MOV #400, R2
MOV R2, (R0)
MOV (R0), R1
CMP R2, R1
BEQ 6\$
CLR (R0)
ERROR 25
MOV #6\$, \$LPERR
CLR (R0)
MOV (R0), R1
BEQ 7\$
ERROR 21
ASL R2
MOV #5\$, \$LPERR
BCC 5\$
MOV #177400, (R0)
RESET
MOV (R0), R1
BEQ 8\$
CLR (R0)
ERROR 21
MOV #20\$, \$LPERR

: SAVE STARTING ADDRESS OF NEXT
: TEST FOR ESCAPE ON PARITY ERRORS
: MAKE SURE T-BIT IS OFF FOR THIS TEST
: SET LOOP ON ERROR POINTER TO 1\$
: LOAD ADDRESS OF STACK LIMIT REGISTER
: ONLY HIGH(ODD) BYTE SHOULD BE LOADED
: READ STACK LIMIT REGISTER
: LEAVE STACK LIMIT REGISTER CLEAR
: LOAD EXPECTED DATA INTO R2
: DID YOU REFERENCE THE STACK LIMIT REG
: BRANCH IF IT WAS STACK LIMIT
: NOT STACK LIMIT REGISTER
: CHANGE ADDRESS TO REFERENCE HIGH(ODD) BYTE
: SET LOOP ON ERROR POINTER TO 3\$
: SHOULD ONLY LOAD HIGH(ODD) BYTE
: CHANGE ADDRESS TO REFERENCE WORD
: READ STACK LIMIT REGISTER
: LEAVE STACK LIMIT REGISTER CLEAR
: LOAD EXPECTED DATA INTO R2
: DID YOU REFERENCE HIGH(ODD) BYTE
: BRANCH IF DATA MATCHES
: CHANGE ADDRESS TO ODD BYTE FOR ERROR
: WRONG DATA BACK FROM STACK LIMIT REG
: CHANGE ADDRESS BACK TO WORD ADDRESS
: SET LOOP ON ERROR POINTER TO 5\$
: LOAD TEST PATTERN INTO R2
: LOAD TEST PATTERN INTO STACK LIMIT REG.
: READ STACK LIMIT REGISTER
: WAS CORRECT DATA READ
: BRANCH IF DATA MATCHES
: CLEAR STK. LMT. REG. FOR ERROR CALL
: WRONG DATA BACK FROM STACK LIMIT REG.
: SET LOOP ON ERROR POINTER TO 6\$
: TRY TO CLEAR THE STACK LIMIT REG
: READ THE STACK LIMIT REGISTER
: BRANCH IF STACK LIMIT REG WAS CLEARED
: STACK LIMIT REG WOULD NOT CLEAR
: CHANGE TEST DATA TO TEST NEXT BIT
: SET LOOP ON ERROR POINTER TO 5\$
: BRANCH UNTIL BITS ARE ALL TESTED
: LOAD STACK LIMIT REG. WITH ALL 1'S
: TRY TO CLEAR WITH A RESET INST.
: READ THE STACK LIMIT REG.
: BRANCH IF WAS CLEARED
: CLEAR STK. LMT. REG. FOR ERROR CALL
: STACK LIMIT REG. WOULD NOT CLEAR
: SET LOOP POINTER TO START OF TEST

: *TEST 3 PROCESSOR STATUS WORD

E05

2815
2816
2817
2818
2819
2820
2821 021352
2822 021352 000004
2823 021354 012737 021442 001444
2824
2825 021362 104411
2826
2827 021364 012737 021372 001106
2828
2829 021372 012737 021410 001110 20\$:
2830 021400 012700 177776
2831 021404 012702 000340
2832 021410 005010 1\$:
2833 021412 052710 000340
2834 021416 000257
2835 021420 111001
2836 021422 042701 177420
2837 021426 020201
2838 021430 001401
2839 021432 104023
2840 021434 012737 021372 001110 2\$:
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852 021442
2853 021442 000004
2854 021444 012737 021512 001444
2855
2856 021452 005037 177776
2857 021456 012706 001100
2858 021462 012737 140000 177776
2859 021470 012706 000700
2860 021474 005037 177776
2861 021500 010600
2862 021502 022700 001100
2863 021506 001401
2864 021510 104027
2865
2866
2867
2868
2869
2870

```

;*
;* 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.
;*
*****
TST3:
SCOPE
MOV #TST4,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
TBITR ;RESTORE THE T-BIT IF IT WAS ON
;BEFORE THE LAST TEST
MOV #20$,$LPADR ;SET LOOP ADDRESS POINTER TO 20$
;FOR ITERATION PASS
20$: MOV #1$,$LPERR ;SET LOOP ON ERROR POINTER TO 1$
MOV #177776,R0 ;LOAD ADDRESS OF P.S.W. INTO R0
MOV #340,R2 ;LOAD EXPECTED DATA INTO R2
1$: CLR (R0) ;CLEAR THE PSW
BIS #340,(R0) ;SET PRIORITY TO LEVEL SEVEN
CCC ;CLEAR ALL CONDITION CODES
MOVB (R0),R1 ;LOAD LOWER BYTE OF P.S.W. INTO R1
BIC #177420,R1 ;CLEAR "REG. SIGN EXTEND" AND T-BIT
CMP R2,R1 ;LOWER BYTE OF PSW SHOULD BE 340
BEQ 2$ ;BRANCH IF PSW IS CORRECT
ERROR 23 ;MUST HAVE READ SOME OTHER REGISTER
2$: MOV #20$,$LPERR ;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
CLR PSW ;GET TO KERNEL MODE, PRIORITY 0
MOV #KERSTK,KSP ;SETUP KERNEL STACK POINTER
MOV #140000,PSW ;GET INTO USER MODE
MOV #USESTK,USP ;SET UP USER STACK POINTER
CLR PSW ;GET BACK INTO KERNEL MODE
MOV KSP,R0 ;READ KSRNEL STACK POINTER
CMP #KERSTK,R0 ;SEE IF KERNEL STACK IS STILL 1100
BEQ TST5 ;BRANCH IF KERNEL STACK IS OKAY
ERROR 27 ;KERNEL STACK POINTER IS NOT RIGHT

```

```

.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

```

F05

CQKTA-BO PDP 11/6X MEM. MGMT. DIAG.
CQKTAB.P11 30-DEC-77 14:49

MACY11 30A(1052) 03-JAN-78 08:09 PAGE 57
** TEST READ/WRITE BITS IN MEMORY MANAGEMENT STATUS REGISTERS **

SEQ 0057

```

2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889 021512
2890 021512 000004
2891 021514 012737 021700 001444
2892
2893 021522
2894 021522 012737 021544 001110
2895 021530 012737 000005 001102
2896 021536 013777 001102 157376
2897 021544 012700 177572
2898 021550 012710 160000
2899 021554 000005
2900 021556 011001
2901 021560 001401
2902 021562 104007
2903 021564 012737 021572 001110
2904 021572 012710 160000
2905 021576 011001
2906 021600 022701 160000
2907 021604 001401
2908 021606 104010
2909 021610 012737 021616 001110
2910 021616 005010
2911 021620 011001
2912 021622 001401
2913 021624 104007
2914 021626 012737 021644 001110
2915 021634 012702 100000
2916 021640 012703 000003
2917 021644 010210
2918 021646 011001
2919 021650 005010
2920 021652 010204
2921 021654 042704 017777
2922 021660 020401
2923 021662 001401
2924 021664 104011
2925 021666 006002
2926 021670 077313
    
```

```

;*
;*      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.
;*
    
```

```

*****
*TEST 5      BIT TEST OF MEMORY MANAGEMENT REGISTER 0
    
```

```

;*
;*      THIS TEST TRIES TO SET AND CLEAR BITS <15:13> OF
;*      MMRO. 'INIT' IS ISSUED TO SEE THAT IT WILL CLEAR THESE BITS.
;*      THE OTHER BITS OF MMRO ARE CLOCKED ONLY ON MEMORY
;*      MANAGEMENT ERROR CONDITIONS IF RELOCATION IS ENABLED.
;*      BIT <00> ENABLES FULL RELOCATION AND BIT <08> ENABLES
;*      RELOCATION ON THE DESTINATION CYCLE ONLY. THESE BITS WILL
;*      BE TESTED IN A LATER TEST.
    
```

```

*****
†ST5:
    
```

```

        SCOPE
        MOV      #TST6,NXTTST      ;SAVE STARTING ADDRESS OF NEXT
        ;TEST FOR ESCAPE ON PARITY ERRORS
ENTPT2:
        MOV      #20$, $LPERR      ;SET LOOP ON ERROR POINTER TO 20$
        MOV      #5, $TSTNM        ;LOAD TEST NUMBER INTO MEMORY
        MOV      $TSTNM, @DISPLAY  ;DISPLAY TEST NUMBER FOR THIS TEST
20$:    MOV      #MMRO, R0          ;PUT ADDRESS OF MMRO IN R0
        MOV      #160000, (R0)     ;LOAD WRITABLE BITS IN MMRO
        RESET
        MOV      (R0), R1          ;'INIT' SHOULD CLEAR ALL BITS OF MMRO
        BEQ     1$                ;READ MMRO
        ERROR   7                  ;BRANCH IF MMRO IS CLEAR
        MOV      #11$, $LPERR      ;MMRO WON'T CLEAR
        MOV      #160000, (R0)     ;SET LOOP ON ERROR POINTER TO 11$
11$:    MOV      (R0), R1          ;SET BITS <15:13> OF MMRO
        CMP     #160000, R1        ;READ MMRO
        BEQ     2$                ;SEE IF BITS <15:13> ARE SET
        ERROR   10                 ;BRANCH IF CORRECT BITS ARE SET
        MOV      #12$, $LPERR      ;CAN'T SET 160000 IN MMRO
        CLR     (R0)              ;SET LOOP ON ERROR POINTER TO 12$
12$:    MOV      (R0), R1          ;CLEAR UPPER 3 BITS OF MMRO
        BEQ     3$                ;READ MMRO
        ERROR   7                  ;BRANCH IF MMRO IS CLEAR
        MOV      #5$, $LPERR       ;MMRO WON'T CLEAR
        MOV      #BIT15, R2        ;SET LOOP ON ERROR POINTER TO 5$
        MOV      #3, R3            ;SET BIT IN R2 TO FLOAT THRU MMRO
        MOV      R2, (R0)          ;PUT LOOP COUNT IN R3
        CLR     (R0)              ;LOAD R2 INTO MMRO
        MOV      R2, R4            ;READ MMRO IN R1
        BIC     #017777, R4        ;CLEAR MMRO
        CMP     R4, R1            ;PUT PATTERN IN R4
        BEQ     4$                ;MASK OFF BITS <12:00>
        ERROR   11                 ;COMPARE PATTERN WITH MMRO
        ROR     R2                ;BRANCH IF DATA MATCHES
        SOB    R3, 5$             ;GOT WRONG DATA FROM MMRO
        ;SHIFT BIT TO RIGHT
        ;LOOP BACK TWICE
    
```

2927 021672 012737 021544 001110
 2928
 2929
 2930
 2931
 2932
 2933
 2934
 2935
 2936
 2937
 2938
 2939
 2940 021700
 2941 021700 000004
 2942 021702 012737 021730 001444
 2943
 2944
 2945 021710 012700 177574
 2946 021714 012701 177777
 2947 021720 011001
 2948 021722 005701
 2949 021724 001401
 2950 021726 104024
 2951 021730
 2952
 2953
 2954
 2955
 2956
 2957
 2958
 2959
 2960
 2961
 2962
 2963
 2964
 2965
 2966
 2967 021730
 2968 021730 000004
 2969 021732 012737 022074 001444
 2970
 2971 021740 012700 177572
 2972 021744 012701 177576
 2973 021750 012737 021762 001110
 2974 021756 005037 177572
 2975 021762 011102
 2976 021764 012703 021762
 2977 021770 020203
 2978 021772 001401
 2979 021774 104012
 2980 021776 012737 022004 001110
 2981 022004 013702 177576
 2982 022010 012703 022004

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

 *TEST 6 SEE THAT MEMORY MANAGEMENT REGISTER 1 READS 0'S
 *
 * THIS TEST WILL ENSURE THAT ALTHOUGH MEMORY MANAGEMENT
 * REGISTER #1 (777574) IS NON-EXSISTENT, THAT ADDRESS
 * WILL RESPOND AND READ AS THOUGH THERE WAS A REGISTER
 * THERE AND THAT IT READS AS ALL ZEROES.
 *

TST6:
 SCOPE
 MOV #TST7,NXTTST ;SAVE STARTING ADDRESS OF NEXT
 ;TEST FOR ESCAPE ON PARITY ERRORS
 20\$: MOV #MMR1,R0 ;PUT ADDRESS OF MMR1 IN R0
 MOV #-1,R1 ;LOAD R1 WITH ALL ONES
 MOV (R0),R1 ;READ MMR1 INTO R1
 TST R1 ;DID MMR1 READ AS ALL ZEROES?
 BEQ 1\$;BRANCH IF YES
 ERROR 24 ;MMR1 DID NOT READ ALL ZEROES
 1\$:

 *TEST 7 BIT TEST OF MEMORY MANAGEMENT REGISTER 2
 *
 * HERE MMR2 WILL BE READ SEVERAL DIFFERENT TIMES TO
 * SEE THAT IT IS TRACKING PROPERLY. THEN IT WILL BE
 * LOCKED UP AND READ IT TO SEE THAT IT DOES NOT CHANGE AFTER
 * IT IS ONCE LOCKED UP. NOT ALL BITS WILL BE TESTED IN THIS
 * TEST BUT A LATER TEST RUNS A COUNT PATTERN THROUGH MMR2. THIS
 * CANNOT BE DONE HERE SINCE IT REQUIRES THAT THE ABORT LOGIC IS
 * FUNCTIONING PROPERLY.
 *

TST7:
 SCOPE
 MOV #TST10,NXTTST ;SAVE STARTING ADDRESS OF NEXT
 ;TEST FOR ESCAPE ON PARITY ERRORS
 20\$: MOV #MMR0,R0 ;PUT ADDRESS OF MMR0 IN R0
 MOV #MMR2,R1 ;PUT ADDRESS OF MMR2 IN R1
 MOV #21\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 21\$
 CLR @MMR0 ;MAKE SURE THAT MMR0 IS CLEAR
 21\$: MOV (R1),R2 ;READ MMR2 TO R2
 MOV #21\$,R3 ;PUT ADDRESS OF LAST INST IN R3
 CMP R2,R3 ;SEE IF MMR2 HELD CORRECT ADDRESS
 BEQ 1\$;BRANCH IF DATA MATCHES
 ERROR 12 ;MMR2 DID NOT TRACK PROPERLY
 1\$: MOV #22\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 22\$
 22\$: MOV @MMR2,R2 ;READ MMR2 TO R2
 MOV #22\$,R3 ;PUT ADDRESS OF LAST INST IN R3

H05

CQKTA-B0 PDP 11/6X MEM. MGMT. DIAG.
CQKTAB.P11 30-DEC-77 14:49

MACY11 30A(1052) 03-JAN-78 08:09 PAGE 59
T7 BIT TEST OF MEMORY MANAGEMENT REGISTER 2

SEQ 0059

```
2983 022014 020203      CMP      R2,R3      ;SEE IF MMR2 HELD CORRECT ADDRESS
2984 022016 001401      BEQ      12         ;BRANCH IF DATA MATCHES
2985 022020 104012      ERROR    12         ;MMR2 DID NOT TRACK PROPERLY
2986 022022 012737 022030 001110 2$:      MOV      #23$, $LPERR ;SET LOOP ON ERROR POINTER TO 23$
2987 022030 012710 040000 23$:      MOV      #40000,(R0) ;LOCK UP MMR2
2988 022034 011102      MOV      (R1),R2    ;READ MMR2 INTO R2
2989 022036 012703 022030      MOV      #23$,R3    ;PUT LOCKING INST'S ADDRESS IN R3
2990 022042 020203      CMP      R2,R3      ;SEE IF MMR2 HOLDS RIGHT ADDRESS
2991 022044 001401      BEQ      12         ;BRANCH IF DATA MATCHES
2992 022046 104012      ERROR    12         ;MMR2 DID NOT TRACK PROPERLY
2993 022050 012737 022056 001110 3$:      MOV      #14$, $LPERR ;SET LOOP ON ERROR POINTER TO 14$
2994 022056 000005      RESET   ;ISSUE INIT TO CLEAR MMR2
2995 022060 011102      MOV      (R1),R2    ;READ MMR2 INTO R2
2996 022062 012703 022060      MOV      #25$,R3    ;PUT ADDRESS OF LAST INST IN R3
2997 022066 020203      CMP      R2,R3      ;SEE IF MMR2 IS STILL TRACKING
2998 022070 001401      BEQ      TST10     ;GO TO NEXT TEST IF IT IS
2999 022072 104012      ERROR    12         ;MMR2 DID NOT TRACK PROPERLY
```

```
3000
3001
3002
3003 .SBTTL ***** ENTRY POINT 3 --- STARTING ADDRESS 210 *****
3004 .SBTTL ***** PAGE ADDRESS AND DESCRIPTOR REGISTER TESTS *****
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
```

```
.SBTTL TEST THAT ALL P.A.R.'S & P.D.R.'S RESPOND
* THESE NEXT FOUR (4) TESTS VERIFY THAT THERE ARE 4 GROUPS OF
* 8 CONTIGUOUS ADDRESSES IN THE I/O PAGE THAT WILL RESPOND
* WITHOUT TIMING OUT. AT THIS POINT THE TEST IS NOT CONCERNED
* WITH EXACTLY WHAT IS RESPONDING, JUST THAT SOMETHING RESPONDS.
*
* *****
* TEST 10 READ ALL KERNEL PAGE ADDRESS REGISTERS, CHECK FOR TIMEOUT
*
* THIS TEST DOES A READ FROM ALL THE KERNEL PAGE ADDRESS REGISTERS
* 'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE TEST AND
* IF ANY OF THE 8 KERNEL ADDRESS REGISTERS TIME OUT THEIR I/O
* PAGE ADDRESS IS REPORTED AND LOGGED. AT THE END OF THE TEST A
* SUMMARY OF THE ERRORS IS GIVEN AND 'ERRVEC' IS RE-SET TO 'CPUER'.
*
* ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE "TIMEOUT"
*
* *****
```

```
3031 022074 000004      TST10:
3032 022074 012737 022220 001444      SCOPE
3033 022076 012737 022220 001444      MOV      #TST11,NXTTST ;SAVE STARTING ADDRESS OF NEXT
3034
3035 ENTPT3:
3036 022104 012737 022126 001110      MOV      #20$, $LPERR ;SET LOOP ON ERROR POINTER TO 20$
3037 022112 012737 000010 001102      MOV      #10,$TSTNM   ;LOAD TEST NUMBER INTO MEMORY
3038 022120 013777 001102 157014      MOV      $TSTNM,$DISPLAY ;DISPLAY TEST NUMBER FOR THIS TEST
```

3039 022126 004737 044254
3040 022132 012737 044312 000004
3041 022140 012737 022152 001110
3042 022146 012700 172340
3043 022152 011001
3044 022154 062700 000002
3045 022160 022700 172356
3046 022164 103372
3047 022166 012737 022126 001110
3048 022174 012737 015160 000004
3049 022202 005737 001430
3050 022206 001404
3051 022210 013737 001430 001210
3052 022216 104013
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067 022220
3068 022220 000004
3069 022222 012737 022322 001444
3070
3071 022230 004737 044254
3072 022234 012737 044312 000004
3073 022242 012737 022254 001110
3074 022250 012700 177640
3075 022254 011001
3076 022256 062700 000002
3077 022262 022700 177656
3078 022266 103372
3079 022270 012737 022230 001110
3080 022276 012737 015160 000004
3081 022304 005737 001430
3082 022310 001404
3083 022312 013737 001430 001210
3084 022320 104013
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094

20\$: JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
MOV #TIMEOUT,ERRVEC ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
MOV #1\$,SLPERR ;SET LOOP ON ERROR POINTER TO 1\$
MOV #KIPAR0,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 #KIPAR7,R0 ;SEE IF KIPAR7 HAS BEEN TRIED
BHS 1\$;BRANCH IF NOT DONE
MOV #20\$,SLPERR ;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 TST11 ;BRANCH TO NEXT TEST IF NO ERRORS
MOV ERRCNT,\$TMP5 ;SAVE NUMBER OF ERRORS FOR TYPEOUT
ERROR 13 ;SUMMARY OF PAR ERRORS

*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"

TST11: SCOPE
MOV #TST12,NXTTST ;SAVE STARTING ADDRESS OF NEXT
20\$: JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
MOV #TIMEOUT,ERRVEC ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
MOV #1\$,SLPERR ;SET LOOP ON ERROR POINTER TO 1\$
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
BHS 1\$;BRANCH IF NOT DONE
MOV #20\$,SLPERR ;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

*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'.

```

3095
3096
3097
3098 022322
3099 022322 000004
3100 022324 012737 022424 001444
3101
3102 022332 004737 044254
3103 022336 012737 044312 000004
3104 022344 012737 022356 001110
3105 022352 012700 172300
3106 022356 011001
3107 022360 062700 000002
3108 022364 022700 172316
3109 022370 103372
3110 022372 012737 022332 001110
3111 022400 012737 015160 000004
3112 022406 005737 001430
3113 022412 001404
3114 022414 013737 001430 001210
3115 022422 104013
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130 022424
3131 022424 000004
3132 022426 012737 022526 001444
3133
3134 022434 004737 044254
3135 022440 012737 044312 000004
3136 022446 012737 022460 001110
3137 022454 012700 177600
3138 022460 011001
3139 022462 062700 000002
3140 022466 022700 177616
3141 022472 103372
3142 022474 012737 022434 001110
3143 022502 012737 015160 000004
3144 022510 005737 001430
3145 022514 001404
3146 022516 013737 001430 001210
3147 022524 104013
3148
3149
3150

```

```

:: ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE "TIMEOUT"
::
:: *****
TST12:
SCOPE
MOV #TST13,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
;INITIALIZE ERROR LOCATIONS
20$: JSR PC.CLEANUP ;INITIALIZE ERROR LOCATIONS
MOV #TIMEOUT,ERRVEC ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
MOV #1$, $LPERR ;SET LOOP ON ERROR POINTER TO 1$
MOV #KIPDR0,RO ;PUT ADDRESS OF FIRST PDR IN RO
1$: MOV (RO),R1 ;THIS WILL TIMEOUT IF REGISTER DECODING BAD
ADD #2,RO ;POINT TO NEXT REGISTER
CMP #KIPDR7,RO ;SEE IF KIPDR7 HAS BEEN TRIED
BHS 1$ ;BRANCH IF NOT DONE
MOV #20$, $LPERR ;PUT LOOP ON ERROR POINTER TO START OF TEST
MOV #CPUER,ERRVEC ;RE-SET NORMAL CPU TRAP SERVICE ROUTINE
TST ERRCNT ;SEE IF ANY ERRORS OCCURRED ON THIS TEST
BEQ TST13 ;BRANCH TO NEXT TEST IF NO ERRORS
MOV ERRCNT,$TMP5 ;SAVE NUMBER OF ERRORS FOR TYPEOUT
ERROR 13 ;SUMMARY OF PDR ERRORS

```

```

:: *****
*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
;INITIALIZE ERROR LOCATIONS
20$: JSR PC.CLEANUP ;INITIALIZE ERROR LOCATIONS
MOV #TIMEOUT,ERRVEC ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
MOV #1$, $LPERR ;SET LOOP ON ERROR POINTER TO 1$
MOV #UIPDR0,RO ;PUT ADDRESS OF FIRST PDR IN RO
1$: MOV (RO),R1 ;THIS WILL TIMEOUT IF REGISTER DECODING BAD
ADD #2,RO ;POINT TO NEXT REGISTER
CMP #UIPDR7,RO ;SEE IF UIPDR7 HAS BEEN TRIED
BHS 1$ ;BRANCH IF NOT DONE
MOV #20$, $LPERR ;PUT LOOP ON ERROR POINTER TO START OF TEST
MOV #CPUER,ERRVEC ;RE-SET NORMAL CPU TRAP SERVICE ROUTINE
TST ERRCNT ;SEE IF ANY ERRORS OCCURRED ON THIS TEST
BEQ TST14 ;BRANCH TO NEXT TEST IF NO ERRORS
MOV ERRCNT,$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

```

K05

CQKTA-BO PDP 11/6X MEM. MGMT. DIAG.
CQKTAB.P11 30-DEC-77 14:49

MACY11 30A(1052) 03-JAN-78 08:09 PAGE 62
TEST FOR DUAL ADDRESSING ON LOAD WITHIN GROUPS

SEQ 0062

3151
3152

;* 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

```

3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172 022526
3173 022526 000004
3174 022530 012737 022732 001444
3175
3176 022536 004737 044254
3177 022542 012737 022570 001110
3178 022550 012705 172340
3179 022554 004737 044236
3180 022560 012700 172340
3181 022564 012701 000010
3182 022570 011002
3183 022572 001401
3184 022574 104014
3185 022576 062700 000002
3186 022602 077106
3187
3188
3189
3190
3191
3192 022604 012737 022616 001110
3193 022612 012700 172340
3194 022616 012705 172340
3195 022622 004737 044236
3196 022626 012701 172356
3197 022632 012710 177777
3198 022636 005037 001176
3199 022642 011102
3200 022644 001406
3201 022646 020001
3202
3203 022650 001402
3204 022652 004737 044406
3205 022656 005237 001176
3206 022662 162701 000002
3207 022666 022701 172340
3208 022672 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
: * 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,CLRRÉG ;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
: * MOV (R0),R2 ;READ PAR TO R2
: * BEQ 2$ ;BRANCH IF PAR IS 0
: * ERROR 14 ;PAR NOT ZERO
: * 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
: * MOV #KIPAR0,R5 ;LOAD STARTING ADDRESS INTO R5
: * JSR PC,CLRRÉG ;CLEAR 8 REGISTERS POINTED TO BY R5
: * MOV #KIPAR7,R1 ;PUT KIPAR7 ADDRESS IN R1
: * MOV #-1,(R0) ;LOAD REGISTER UNDER TEST
: * 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 $TMP0 ;SET FLAG WHEN ADDRESSES MATCH
: * SUB #2,R1 ;POINT TO NEXT REGISTER
: * CMP #KIPAR0,R1 ;SEE IF ALL REGISTERS HAVE BEEN READ
: * BLOS 4$ ;BRANCH IF MORE TO READ

```


3209	022674	062700	000002		ADD	#2,RO	;	NOW LOAD NEXT REGISTER
3210	022700	022700	172356		CMP	#RIPAR7,RO	;	SEE IF THERE ARE MORE REGISTERS TO TEST
3211	022704	103344			BHIS	3\$;	BRANCH IF MORE REGISTERS TO TEST
3212	022706	012737	022536	001110	MOV	#20\$,SLPERR	;	SET LOOP ON ERROR POINTER TO START OF TEST
3213	022714	005737	001430		TST	ERRCNT	;	SEE IF THERE WERE ANY ERRORS
3214	022720	001404			BEG	TST15	;	BRANCH TO NEXT TEST IF NO ERRORS
3215	022722	013737	001430	001210	MOV	ERRCNT,\$TMP5	;	SAVE NUMBER OF ERRORS FOR PAR OUT
3216	022730	104015			ERROR	15	;	SUMMARY OF DUAL ADDRESSING ERRORS

 *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:

3233	022732				SCOPE				
3234	022732	000004			MOV	#TST16,NXTTST	;	SAVE STARTING ADDRESS OF NEXT	
3235	022734	012737	023136	001444			;	TEST FOR ESCAPE ON PARITY ERRORS	
3236							;	INITIALIZE ERROR LOCATIONS	
3237	022742	004737	044254		20\$:	JSR	PC,CLEANUP	;	SET LOOP ON ERROR POINTER TO 1\$
3238	022746	012737	022774	001110	MOV	#1\$,SLPERR	;	PUT ADDRESS OF FIRST PAR IN R5	
3239	022754	012705	177640		MOV	#UIPARO,R5	;	CLEAR 8 REGISTERS POINTED TO BY R5	
3240	022760	004737	044236		JSR	PC,CLRRÉG	;	PUT ADDRESS OF FIRST PAR IN RO	
3241	022764	012700	177640		MOV	#UIPARO,RO	;	BRANCH COUNT IS 8 DECIMAL	
3242	022770	012701	000010		MOV	#10,R1	;	READ PAR TO R2	
3243	022774	011002			1\$:	MOV	(RO),R2	;	BRANCH IF PAR IS 0
3244	022776	001401			BEG	2\$;	PAR NOT ZERO	
3245	023000	104014			ERROR	14	;	POINT TO NEXT REGISTER	
3246	023002	062700	000002		2\$:	ADD	#2,RO	;	BRANCH BACK TO 1\$ 7 TIMES
3247	023006	077106			SOB	R1,1\$			

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.)

3253	023010	012737	023022	001110	MOV	#3\$,SLPERR	;	SET LOOP ON ERROR POINTER TO 3\$	
3254	023016	012700	177640		MOV	#UIPARO,RO	;	PUT ADDRESS OF FIRST PAR IS RO	
3255	023022	012705	177640		3\$:	MOV	#UIPARO,R5	;	LOAD STARTING ADDRESS INTO R5
3256	023026	004737	044236		JSR	PC,CLRRÉG	;	CLEAR 8 REGISTERS POINTED TO BY R5	
3257	023032	012701	177656		MOV	#UIPAR7,R1	;	PUT UIPAR7 ADDRESS IN R1	
3258	023036	012710	177777		MOV	#-1,(RO)	;	LOAD REGISTER UNDER TEST	
3259	023042	005037	001176		4\$:	CLR	\$TMP0	;	FLAG TO INDICATE THERE WAS A MATCH
3260	023046	011102			MOV	(R1),R2	;	READ ALL REGISTERS	
3261	023050	001406			BEG	6\$;	BRANCH IF REGISTER IS 0	
3262	023052	020001			CMP	RO,R1	;	IS ADDRESS OF NON-ZERO REGISTER SAME	
3263							;	AS THAT OF REGISTER UNDER TEST	
3264	023054	001402			BEG	5\$;	BRANCH IF ADDRESSES MATCH	

3265 023056 004737 044406
 3266 023062 005237 001176
 3267 023066 162701 000002
 3268 023072 022701 177640
 3269 023076 101761
 3270 023100 062700 000002
 3271 023104 022700 177656
 3272 023110 103344
 3273 023112 012737 022742 001110
 3274 023120 005737 001430
 3275 023124 001404
 3276 023126 013737 001430 001210
 3277 023134 104015

```

JSR PC,DUALADR ;LOG AND REPORT ERRORS
5$: INC $TMPD ;SET FLAG WHEN ADDRESSES MATCH
6$: SUB #2,R1 ;POINT TO NEXT REGISTER
CMP #UIPARO,R1 ;SEE IF ALL REGISTERS HAVE BEEN READ
BLOS 4$ ;BRANCH IF MORE TO READ
ADD #2,RO ;NOW LOAD NEXT REGISTER
CMP #UIPAR7,RO ;SEE IF THERE ARE MORE REGISTERS TO TEST
BHIS 3$ ;BRANCH IF MORE REGISTERS TO TEST
MOV #20$,$LPERR ;SET LOOP ON ERROR POINTER TO START OF TEST
TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
BEQ TST16 ;BRANCH TO NEXT TEST IF NO ERRORS
MOV ERRCNT,$TMP5 ;SAVE NUMBER OF ERRORS FOR PAR OUT
ERROR 15 ;SUMMARY OF DUAL ADDRESSING ERRORS

```

 *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:

3293 023136
 3294 023136 000004
 3295 023140 012737 023342 001444
 3296
 3297 023146 004737 044254
 3298 023152 012737 023200 001110
 3299 023160 012705 172300
 3300 023164 004737 044236
 3301 023170 012700 172300
 3302 023174 012701 000010
 3303 023200 011002
 3304 023202 001401
 3305 023204 104014
 3306 023206 062700 000002
 3307 023212 077106
 3308
 3309
 3310
 3311
 3312
 3313 023214 012737 023226 001110
 3314 023222 012700 172300
 3315 023226 012705 172300
 3316 023232 004737 044236
 3317 023236 012701 172316
 3318 023242 012710 177777
 3319 023246 005037 001176
 3320 023252 011102

```

MOV #TST17,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
20$: JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
MOV #1$,$LPERR ;SET LOOP ON ERROR POINTER TO 1$
MOV #KIPDR,R5 ;PUT ADDRESS OF FIRST PDR IN R5
JSR PC,CLRREG ;CLEAR 8 REGISTERS POINTED TO BY R5
MOV #KIPDR,RO ;PUT ADDRESS OF FIRST PDR IN RO
MOV #10,R1 ;BRANCH COUNT IS 8 DECIMAL
1$: MOV (RO),R2 ;READ PDR TO R2
BEQ 2$ ;BRANCH IF PDR IS 0
ERROR 14 ;PDR NOT ZERO
2$: ADD #2,RO ;POINT TO NEXT REGISTER
SOB R1,1$ ;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$,$LPERR ;SET LOOP ON ERROR POINTER TO 3$
MOV #KIPDR,RO ;PUT ADDRESS OF FIRST PDR IS RO
3$: MOV #KIPDR,R5 ;LOAD STARTING ADDRESS INTO R5
JSR PC,CLRREG ;CLEAR 8 REGISTERS POINTED TO BY R5
MOV #KIPDR7,R1 ;PUT KIPDR7 ADDRESS IN R1
MOV #-1,(RO) ;LOAD REGISTER UNDER TEST
4$: CLR $TMPD ;FLAG TO INDICATE THERE WAS A MATCH
MOV (R1),R2 ;READ ALL REGISTERS

```

3321 023254 001406 BEQ 6\$;BRANCH IF REGISTER IS 0
3322 023256 020001 CMP RO,R1 ;IS ADDRESS OF NON-ZERO REGISTER SAME
3323 ;AS THAT OF REGISTER UNDER TEST
3324 023260 001402 BEQ 5\$;BRANCH IF ADDRESSES MATCH
3325 023262 004737 044406 JSR PC,DUALADR ;LOG AND REPORT ERRORS
3326 023266 005237 001176 5\$: INC \$TMPD ;SET FLAG WHEN ADDRESSES MATCH
3327 023272 162701 000002 6\$: SUB #2,R1 ;POINT TO NEXT REGISTER
3328 023276 022701 172300 CMP #KIPDR0,R1 ;SEE IF ALL REGISTERS HAVE BEEN READ
3329 023302 101761 BLOS 4\$;BRANCH IF MORE TO READ
3330 023304 062700 000002 ADD #2,RO ;NOW LOAD NEXT REGISTER
3331 023310 022700 172316 CMP #KIPDR7,RO ;SEE IF THERE ARE MORE REGISTERS TO TEST
3332 023314 103344 BHIS 3\$;BRANCH IF MORE REGISTERS TO TEST
3333 023316 012737 023146 001110 MOV #20\$, \$LPERR ;SET LOOP ON ERROR POINTER TO START OF TEST
3334 023324 005737 001430 TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
3335 023330 001404 BEQ TST17 ;BRANCH TO NEXT TEST IF NO ERRORS
3336 023332 013737 001430 001210 MOV ERRCNT,\$TMP5 ;SAVE NUMBER OF ERRORS FOR PDR OUT
3337 023340 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:

3354 023342 000004 SCOPE
3355 023342 012737 023546 001444 MOV #TST20,NXTTST ;SAVE STARTING ADDRESS OF NEXT
3356 023344 012737 023546 001444 ;TEST FOR ESCAPE ON PARITY ERRORS
3357 ;INITIALIZE ERROR LOCATIONS
3358 023352 004737 044254 20\$: JSR PC,CLEANUP ;SET LOOP ON ERROR POINTER TO 1\$
3359 023356 012737 023404 001110 MOV #1\$, \$LPERR ;PUT ADDRESS OF FIRST PDR IN R5
3360 023364 012705 177600 MOV #UIPDR0,R5 ;CLEAR 8 REGISTERS POINTED TO BY R5
3361 023370 004737 044236 JSR PC,CLAREG ;PUT ADDRESS OF FIRST PDR IN R0
3362 023374 012700 177600 MOV #UIPDR0,R0 ;BRANCH COUNT IS 8 DECIMAL
3363 023400 012701 000010 MOV #10,R1 ;READ PDR TO R2
3364 023404 011002 1\$: MOV (R0),R2 ;BRANCH IF PDR IS 0
3365 023406 001401 BEQ 2\$;PDR NOT ZERO
3366 023410 104014 ERROR 14 ;POINT TO NEXT REGISTER
3367 023412 062700 000002 2\$: ADD #2,RO ;BRANCH BACK TO 1\$ 7 TIMES
3368 023416 077106 SOB R1,1\$

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.)

3374 023420 012737 023432 001110 MOV #3\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 3\$
3375 023426 012700 177600 MOV #UIPDR0,R0 ;PUT ADDRESS OF FIRST PDR IN R0
3376 023432 012705 177600 3\$: MOV #UIPDR0,R5 ;LOAD STARTING ADDRESS INTO R5

3377	023436	004737	044236		JSR	PC, CLRREG	; CLEAR 8 REGISTERS POINTED TO BY R5
3378	023442	012701	177616		MOV	#UIPDR7, R1	; PUT UIPDR7 ADDRESS IN R1
3379	023446	012710	177777		MOV	#-1, (R0)	; LOAD REGISTER UNDER TEST
3380	023452	005037	001176	4\$:	CLR	\$TMPD	; FLAG TO INDICATE THERE WAS A MATCH
3381	023456	011102			MOV	(R1), R2	; READ ALL REGISTERS
3382	023460	001406			BEQ	6\$; BRANCH IF REGISTER IS 0
3383	023462	020001			CMP	RO, R1	; IS ADDRESS OF NON-ZERO REGISTER SAME
3384							; AS THAT OF REGISTER UNDER TEST
3385	023464	001402			BEQ	5\$; BRANCH IF ADDRESSES MATCH
3386	023466	004737	044406		JSR	PC, DUALADR	; LOG AND REPORT ERRORS
3387	023472	005237	001176	5\$:	INC	\$TMPD	; SET FLAG WHEN ADDRESSES MATCH
3388	023476	162701	000002	6\$:	SUB	#2, R1	; POINT TO NEXT REGISTER
3389	023502	022701	177600		CMP	#UIPDR0, R1	; SEE IF ALL REGISTERS HAVE BEEN READ
3390	023506	101761			BLOS	4\$; BRANCH IF MORE TO READ
3391	023510	062700	000002		ADD	#2, RO	; NOW LOAD NEXT REGISTER
3392	023514	022700	177616		CMP	#UIPDR7, RO	; SEE IF THERE ARE MORE REGISTERS TO TEST
3393	023520	103344			BHIS	3\$; BRANCH IF MORE REGISTERS TO TEST
3394	023522	012737	023352	001110	MOV	#20\$, \$LPERR	; SET LOOP ON ERROR POINTER TO START OF TEST
3395	023530	005737	001430		TST	ERRCNT	; SEE IF THERE WERE ANY ERRORS
3396	023534	001404			BEQ	TST20	; BRANCH TO NEXT TEST IF NO ERRORS
3397	023536	013737	001430	001210	MOV	ERRCNT, \$TMP5	; SAVE NUMBER OF ERRORS FOR PDR OUT
3398	023544	104015			ERROR	15	; SUMMARY OF DUAL ADDRESSING ERRORS

3399
3400 .SBTTL TEST FOR BAD READ/WRITE BITS IN P.A.R.'S & P.D.R.'S
3401 * THESE NEXT FOUR (4) TESTS CHECK FOR BAD BITS IN THE MEMORY CHIPS
3402 * THAT MAKE UP THE PAR'S AND PDR'S. THE REGISTERS ARE LOADED WITH
3403 * ZERO AND MODIFIED BY "401" UNTIL 177777 IS REACHED IN EACH ONE
3404 * (THE NUMBER 401 WAS CHOSEN FOR FASTER RUN-TIME). THE BITS THAT
3405 * ARE NOT IMPLEMENTED OR THAT ARE NOT READ/WRITE ARE MASKED OUT
3406 * OF THE DATA COMPARE. A LOG OF MULTIPLE ERRORS IS KEPT FOR EACH
3407 * GROUP AND IT IS REPORTED AT THE CONCLUSION OF EACH TEST.
3408 *
3409 *
3410 * *****
3411 * TEST 20 COUNT PATTERN IN KERNEL PAGE ADDRESS REGISTERS
3412 *
3413 * THIS TEST RUNS A COUNT PATTERN THRU THE KERNEL PAGE ADDRESS
3414 * REGISTERS. SINCE BITS <15:12> ARE NOT IMPLEMENTED THEY ARE
3415 * MASKED OUT OF THE DATA COMPARE. THESE BITS ARE STILL SENT
3416 * TO THE ADDRESS REGISTER TO FIND BAD ETCHES. IF THE COUNT PATTERN
3417 * DOES NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS, DATA
3418 * PATTERN, AND BAD DATA ARE REPORTED. AT THE END OF THE TEST A
3419 * SUMMARY OF THE ERRORS IS GIVEN.
3420 *
3421 * ALL ERRORS FOUND BY THIS TEST ARE REPORTED AND ANALYZED BY
3422 * SUBROUTINE "PARCOUNT".
3423 *
3424 * *****

3424	023546				TST20:		
3425	023546	000004			SCOPE		
3426	023550	012737	023700	001444	MOV	#TST21, NXXTST	; SAVE STARTING ADDRESS OF NEXT
3427							; TEST FOR ESCAPE ON PARITY ERRORS
3428	023556	012737	000012	001212	MOV	#12, \$TIMES	; DO 12 ITERATIONS
3429	023564	004737	044254	20\$:	JSR	PC, CLEANUP	; INITIALIZE ERROR LOCATIONS
3430	023570	012737	023604	001110	MOV	#2\$, \$LPERR	; SET LOOP ON ERROR POINTER TO 15
3431	023576	005001			CLR	R1	; CLEAR REGISTER TO HOLD COUNT PATTERN
3432	023600	012700	172340	1\$:	MOV	#KIPAR0, RO	; PUT ADDRESS OF FIRST REGISTER IS RO

3433	023604	010110			2\$:	MOV	R1,(R0)	;LOAD COUNT INTO REGISTER
3434	023606	011002				MOV	(R0),R2	;READ REGISTER BACK TO R2
3435	023610	010104				MOV	R1,R4	;PUT PATTERN IS R4
3436	023612	042704	170000			BIC	#170000,R4	;CLEAR BITS NOT FOUND IN REGISTER.
3437	023616	020402				CMP	R4,R2	;SEE IF DATA MATCHES PATTERN
3438	023620	001402				BEQ	3\$;BRANCH IF DATA IS GOOD
3439	023622	004737	044462			JSR	PC,PARCOUNT	;LOG AND REPORT COUNT ERROR
3440	023626	062700	000002		3\$:	ADD	#2,R0	;POINT TO NEXT REGISTER
3441	023632	022700	172356			CMP	#KIPAR7,R0	;SEE IF YOU PASSED THE KIPAR7 PAR
3442	023636	103362				BHIS	2\$;BRANCH IF MORE PAR'S TO TEST
3443	023640	022701	177777			CMP	#177777,R1	;SEE IF COUNT HAS REACHED 177777
3444	023644	001403				BEQ	4\$;BRANCH IF COUNT IS 177777
3445	023646	062701	000401			ADD	#401,R1	;INCREASE COUNT PATTERN
3446	023652	000752				BR	1\$;BRANCH TO CONTINUE TEST
3447	023654	012737	023564	001110	4\$:	MOV	#20\$,\$LPERR	;SET LOOP POINTER TO START OF TEST
3448	023662	005737	001430			TST	ERRCNT	;SEE IF THERE WERE ANY ERRORS
3449	023666	001404				BEQ	TST21	;BRANCH TO NEXT TEST IF NO ERRORS
3450	023670	013737	001430	001210		MOV	ERRCNT,\$TMP5	;SAVE NUMBER OF ERRORS FOR TYPEOUT
3451	023676	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".
*****

```

3469	023700				TST21:	SCOPE		
3470	023700	000004				MOV	#TST22,NXTTST	;SAVE STARTING ADDRESS OF NEXT
3471	023702	012737	024032	001444				;TEST FOR ESCAPE ON PARITY ERRORS
3472								;DO 12 ITERATIONS
3473	023710	012737	000012	001212		MOV	#12,\$TIMES	;INITIALIZE ERROR LOCATIONS
3474	023716	004737	044254		20\$:	JSR	PC,CLEANUP	;SET LOOP ON ERROR POINTER TO 1\$
3475	023722	012737	023736	001110		MOV	#2\$,\$LPERR	;CLEAR REGISTER TO HOLD COUNT PATTERN
3476	023730	005001				CLR	R1	;PUT ADDRESS OF FIRST REGISTER IS R0
3477	023732	012700	177640		1\$:	MOV	#UIPAR0,R0	;LOAD COUNT INTO REGISTER
3478	023736	010110			2\$:	MOV	R1,(R0)	;READ REGISTER BACK TO R2
3479	023740	011002				MOV	(R0),R2	;PUT PATTERN IS R4
3480	023742	010104				MOV	R1,R4	;CLEAR BITS NOT FOUND IN REGISTER.
3481	023744	042704	170000			BIC	#170000,R4	;SEE IF DATA MATCHES PATTERN
3482	023750	020402				CMP	R4,R2	;BRANCH IF DATA IS GOOD
3483	023752	001402				BEQ	3\$;LOG AND REPORT COUNT ERROR
3484	023754	004737	044462			JSR	PC,PARCOUNT	;POINT TO NEXT REGISTER
3485	023760	062700	000002		3\$:	ADD	#2,R0	;SEE IF YOU PASSED THE UIPAR7 PAR
3486	023764	022700	177656			CMP	#UIPAR7,R0	;BRANCH IF MORE PAR'S TO TEST
3487	023770	103362				BHIS	2\$;SEE IF COUNT HAS REACHED 177777
3488	023772	022701	177777			CMP	#177777,R1	

3489	023776	001403				BEQ	4\$;BRANCH IF COUNT IS 177777
3490	024000	062701	000401			ADD	#401,R1	;INCREASE COUNT PATTERN
3491	024004	000752				BR	1\$;BRANCH TO CONTINUE TEST
3492	024006	012737	023716	001110	4\$:	MOV	#20\$, \$LPERR	;SET LOOP POINTER TO START OF TEST
3493	024014	005737	001430			TST	ERRCNT	;SEE IF THERE WERE ANY ERRORS
3494	024020	001404				BEQ	TST22	;BRANCH TO NEXT TEST IF NO ERRORS
3495	024022	013737	001430	001210		MOV	ERRCNT, \$TMP5	;SAVE NUMBER OF ERRORS FOR TYPEOUT
3496	024030	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.
*****

```

3510						TST22:		
3511	024032					SCOPE		
3512	024032	000004				MOV	#TST23, NXTTST	;SAVE STARTING ADDRESS OF NEXT
3513	024034	012737	024164	001444				;TEST FOR ESCAPE ON PARITY ERRORS
3514								;DO 12 ITERATIONS
3515	024042	012737	000012	001212		MOV	#12, \$TIMES	;INITIALIZE ERROR LOCATIONS
3516	024050	004737	044254		20\$:	JSR	PC, CLEANUP	;SET LOOP ON ERROR POINTER TO 1\$
3517	024054	012737	024070	001110		MOV	#2\$, \$LPERR	;CLEAR REGISTER TO HOLD COUNT PATTERN
3518	024062	005001				CLR	R1	;PUT ADDRESS OF FIRST REGISTER IS R0
3519	024064	012700	172300		1\$:	MOV	#KIPDR0, R0	;LOAD COUNT INTO REGISTER
3520	024070	010110			2\$:	MOV	R1, (R0)	;READ REGISTER BACK TO R2
3521	024072	011002				MOV	(R0), R2	;PUT PATTERN IS R4
3522	024074	010104				MOV	R1, R4	;CLEAR BITS NOT FOUND IN REGISTER.
3523	024076	042704	100361			BIC	#100361, R4	;SEE IF DATA MATCHES PATTERN
3524	024102	020402				CMP	R4, R2	;BRANCH IF DATA IS GOOD
3525	024104	001402				BEQ	3\$;LOG AND REPORT COUNT ERROR
3526	024106	004737	044462			JSR	PC, PARCOUNT	;POINT TO NEXT REGISTER
3527	024112	062700	000002		3\$:	ADD	#2, R0	;SEE IF YOU PASSED THE KIPDR7 PDR
3528	024116	022700	172316			CMP	#KIPDR7, R0	;BRANCH IF MORE PDR'S TO TEST
3529	024122	103362				BHIS	2\$;SEE IF COUNT HAS REACHED 177777
3530	024124	022701	177777			CMP	#177777, R1	;BRANCH IF COUNT IS 177777
3531	024130	001403				BEQ	4\$;INCREASE COUNT PATTERN
3532	024132	062701	000401			ADD	#401, R1	;BRANCH TO CONTINUE TEST
3533	024136	000752				BR	1\$;SET LOOP POINTER TO START OF TEST
3534	024140	012737	024050	001110	4\$:	MOV	#20\$, \$LPERR	;SEE IF THERE WERE ANY ERRORS
3535	024146	005737	001430			TST	ERRCNT	;BRANCH TO NEXT TEST IF NO ERRORS
3536	024152	001404				BEQ	TST23	;SAVE NUMBER OF ERRORS FOR TYPEOUT
3537	024154	013737	001430	001210		MOV	ERRCNT, \$TMP5	;SUMMARY OF COUNT PATTERN FAILURES
3538	024162	104016				ERROR	16	

```

*****
*TEST 23          COUNT PATTERN IN USER PAGE DESCRIPTOR REGISTERS
*
*   THIS TEST RUNS A COUNT PATTERN THRU THE USER PAGE DESCRIPTOR
*****

```

```

3545
3546
3547
3548
3549
3550
3551
3552
3553
3554 024164
3555 024164 000004
3556 024166 012737 024316 001444
3557
3558 024174 012737 000012 001212
3559 024202 004737 044254
3560 024206 012737 024222 001110
3561 024214 005001
3562 024216 012700 177600
3563 024222 010110
3564 024224 011002
3565 024226 010104
3566 024230 042704 100361
3567 024234 020402
3568 024236 001402
3569 024240 004737 044462
3570 024244 062700 000002
3571 024250 022700 177616
3572 024254 103362
3573 024256 022701 177777
3574 024262 001403
3575 024264 062701 000401
3576 024270 000752
3577 024272 012737 024202 001110
3578 024300 005737 001430
3579 024304 001404
3580 024306 013737 001430 001210
3581 024314 104016
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600 024316

```

```

:: 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.

```

```

*****
†ST23:

```

```

SCOPE
MOV #TST24,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
;DO 12 ITERATIONS
20$: JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 1$
CLR R1 ;CLEAR REGISTER TO HOLD COUNT PATTERN
1$: MOV #UIPDR0,R0 ;PUT ADDRESS OF FIRST REGISTER IS R0
2$: MOV R1,(R0) ;LOAD COUNT INTO REGISTER
MOV (R0),R2 ;READ REGISTER BACK TO R2
MOV R1,R4 ;PUT PATTERN IS R4
BIC #100361,R4 ;CLEAR BITS NOT FOUND IN REGISTER.
CMP R4,R2 ;SEE IF DATA MATCHES PATTERN
BEQ 3$ ;BRANCH IF DATA IS GOOD
3$: JSR PC,PARCOUNT ;LOG AND REPORT COUNT ERROR
ADD #2,R0 ;POINT TO NEXT REGISTER
CMP #UIPDR7,R0 ;SEE IF YOU PASSED THE UIPDR7 PDR
BHS 2$ ;BRANCH IF MORE PDR'S TO TEST
CMP #177777,R1 ;SEE IF COUNT HAS REACHED 177777
BEQ 4$ ;BRANCH IF COUNT IS 177777
ADD #401,R1 ;INCREASE COUNT PATTERN
BR 1$ ;BRANCH TO CONTINUE TEST
4$: MOV #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
BEQ TST24 ;BRANCH TO NEXT TEST IF NO ERRORS
MOV ERRCNT,$TMP5 ;SAVE NUMBER OF ERRORS FOR TYPEOUT
ERROR 16 ;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.

```

```

*****
†ST24:

```

```

3601 024316 000004
3602 024320 012737 024430 001444
3603
3604 024326 012737 024344 001110 20$: MOV #11$, $LPERR
3605 024334 012702 000016 MOV #16, R2
3606 024340 012700 172340 MOV #KIPARO, R0
3607 024344 005037 172340 11$: CLR KIPARO
3608 024350 112710 172016 MOV #172016, (R0)
3609 024354 013701 172340 MOV KIPARO, R1
3610 024360 020102 CMP R1, R2
3611 024362 001401 BEQ 1$
3612 024364 104017 ERROR 17
3613 024366 012737 024404 001110 1$: MOV #12$, $LPERR
3614 024374 012700 172341 MOV #KIPARO+1, R0
3615 024400 012702 007416 MOV #007416, R2
3616 024404 112710 000017 12$: MOV #17, (R0)
3617 024410 013701 172340 MOV KIPARO, R1
3618 024414 020102 CMP R1, R2
3619 024416 001401 BEQ 2$
3620 024420 104017 ERROR 17
3621
3622 024422 012737 024326 001110 2$: MOV #20$, $LPERR
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
3648
3649
3650
3651
3652
3653
3654
3655
3656

```

```

*****
*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.
:
*****

```

```

TST25:
3635 024430 000004
3636 024430 012737 024542 001444
3637 024432
3638
3639 024440 012737 024456 001110 20$: MOV #11$, $LPERR
3640 024446 012702 000016 MOV #16, R2
3641 024452 012700 177640 MOV #UIPARO, R0
3642 024456 005037 177640 11$: CLR UIPARO
3643 024462 112710 172016 MOV #172016, (R0)
3644 024466 013701 177640 MOV UIPARO, R1
3645 024472 020102 CMP R1, R2
3646 024474 001401 BEQ 1$
3647 024476 104017 ERROR 17
3648 024500 012737 024516 001110 1$: MOV #12$, $LPERR
3649 024506 012700 177641 MOV #UIPARO+1, R0
3650 024512 012702 007416 MOV #007416, R2
3651 024516 112710 000017 12$: MOV #17, (R0)
3652 024522 013701 177640 MOV UIPARO, R1
3653 024526 020102 CMP R1, R2
3654 024530 001401 BEQ 2$
3655 024532 104017 ERROR 17
3656

```


3657 024534 012737 024440 001110 2\$: MOV #20\$, \$LPERR ;SET LOOP POINTER TO START OF TEST

3658
3659
3660

*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.

3661
3662
3663
3664
3665
3666
3667
3668

†ST26:

3669 024542 000004 SCOPE
3670 024544 012737 024654 001444 MOV #TST27, NXXTST ;SAVE STARTING ADDRESS OF NEXT

3671
3672 024552 012737 024570 001110 20\$: MOV #11\$, \$LPERR ;TEST FOR ESCAPE ON PARITY ERRORS

3673 024560 012702 000016 MOV #16, R2 ;SET LOOP ON ERROR POINTER TO 11\$

3674 024564 012700 172300 MOV #KIPDR0, R0 ;PUT EXPECTED DATA IN R2

3675 024570 005037 172300 11\$: CLR KIPDR0 ;PUT ADDRESS OF REGISTER IN R0

3676 024574 112710 172016 MOVB #172016, (R0) ;CLEAR THE REGISTER UNDER TEST

3677 024600 013701 172300 MOV KIPDR0, R1 ;LOAD LOWER BYTE OF REGISTER

3678 024604 020102 CMP R1, R2 ;READ REGISTER INTO R1

3679 024606 001401 BEQ 1\$;SEE IF ONLY LOWER BYTE WAS WRITTEN

3680 024610 104017 ERROR 17 ;BRANCH IF DATA MATCHES

3681 024612 012737 024630 001110 1\$: MOV #12\$, \$LPERR ;DIDN'T LOAD CORRECT BYTE

3682 024620 012700 172301 MOV #KIPDR0+1, R0 ;SET LOOP ON ERROR POINTER TO 12\$

3683 024624 012702 007416 MOV #007416, R2 ;POINT TO UPPER BYTE OF REGISTER

3684 024630 112710 000017 12\$: MOVB #17, (R0) ;LOAD EXPECTED DATA IN R2

3685 024634 013701 172300 MOV KIPDR0, R1 ;WRITE THE UPPER BYTE OF REGISTER

3686 024640 020102 CMP R1, R2 ;READ REGISTER INTO R1

3687 024642 001401 BEQ 2\$;SEE IF REGISTER HOLDS CORRECT DATA

3688 024644 104017 ERROR 17 ;BRANCH TO EXIT IF DATA RIGHT

3689
3690 024646 012737 024552 001110 2\$: MOV #20\$, \$LPERR ;DIDN'T LOAD CORRECT BYTE

3691
3692
3693
3694
3695

*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.

3696
3697
3698
3699
3700
3701
3702

†ST27:

3703 024654 000004 SCOPE
3704 024654 012737 024766 001444 MOV #TST30, NXXTST ;SAVE STARTING ADDRESS OF NEXT

3705 024656 012737 024702 001110 20\$: MOV #11\$, \$LPERR ;TEST FOR ESCAPE ON PARITY ERRORS

3706 024672 012702 000016 MOV #16, R2 ;SET LOOP ON ERROR POINTER TO 11\$

3707 024676 012700 177600 MOV #UIPDR0, R0 ;PUT EXPECTED DATA IN R2

3708 024702 005037 177600 11\$: CLR UIPDR0 ;PUT ADDRESS OF REGISTER IN R0

3709 024706 112710 172016 MOVB #172016, (R0) ;CLEAR THE REGISTER UNDER TEST

3710 024712 013701 177600 MOV UIPDR0, R1 ;LOAD LOWER BYTE OF REGISTER

3711
3712

3712 024712 013701 177600 MOV UIPDR0, R1 ;READ REGISTER INTO R1

```

3713 024716 020102          CMP      R1,R2          ;SEE IF ONLY LOWER BYTE WAS WRITTEN
3714 024720 001401          BEQ      15$           ;BRANCH IF DATA MATCHES
3715 024722 104017          ERROR   17            ;DIDN'T LOAD CORRECT BYTE
3716 024724 012737 024742 001110 15$:  MOV     #12$, $LPERR   ;SET LOOP ON ERROR POINTER TO 12$
3717 024732 012700 177601          MOV     #UIPDR0+1,R0   ;POINT TO UPPER BYTE OF REGISTER
3718 024736 012702 007416          MOV     #007416,R2     ;LOAD EXPECTED DATA IN R2
3719 024742 112710 000017          MOVVB  #17,(R0)        ;WRITE THE UPPER BYTE OF REGISTER
3720 024746 013701 177600          MOV     UIPDR0,R1      ;READ REGISTER INTO R1
3721 024752 020102          CMP     R1,R2          ;SEE IF REGISTER HOLDS CORRECT DATA
3722 024754 001401          BEQ     25$           ;BRANCH TO EXIT IF DATA RIGHT
3723 024756 104017          ERROR   17            ;DIDN'T LOAD CORRECT BYTE
3724
3725 024760 012737 024664 001110 25$:  MOV     #20$, $LPERR   ;SET LOOP POINTER TO START OF TEST
3726
3727
3728 .SBTTL      DUAL ADDRESSING BETWEEN GROUPS OF REGISTERS
3729 :*****
3730 :*TEST 30      DUAL ADDRESSING FOR ALL PAR'S AND PDR'S
3731 :*
3732 :*          THIS TEST WILL ENSURE THAT THERE IS NO DUAL ADDRESSING BETWEEN
3733 :*          GROUPS OF PAR'S AND PDR'S. THAT IS, WHEN YOU REFERENCE A KERNEL
3734 :*          PAR YOU ARE REALLY REFERENCING THAT PAR. FIRST EACH
3735 :*          PAR0 OR PDR0 IS LOADED WITH A UNIQUE NUMBER 0-6 AND
3736 :*          THEN THEY ARE EACH CHECKED FOR THE CORRECT DATA.
3737 :*          EACH GROUP HAS ALREADY BEEN CHECKED FOR DUAL ADDRESSING WITHIN
3738 :*          ITS OWN GROUP, SO ONLY ONE REGISTER FROM EACH GROUP NEEDS TO
3739 :*          BE TESTED HERE.
3740 :*
3741 :*****
3742 024766          †ST30:
3743 024766 000004          SCOPE
3744 024770 012737 025070 001444          MOV     #TST31,NXTTST ;SAVE STARTING ADDRESS OF NEXT
3745 :*          ;TEST FOR ESCAPE ON PARITY ERRORS
3746 024776 012737 025012 001110 20$:  MOV     #1$, $LPERR   ;SET LOOP ON ERROR POINTER TO 1$
3747 025004 005000          CLR     R0            ;THIS WILL HOLD THE INDEX OF EACH REGISTER
3748 :*          ;AND THE COUNT LOADED
3749 025006 012704 000004          MOV     #4,R4         ;THIS IS THE NUMBER OF TIMES TO DO THIS LOOP
3750 025012 010070 001450          MOV     R0,$PARTAB(R0);LOAD PAR OR PDR WITH INDEX NUMBER
3751 025016 062700 000002          ADD     #2,R0         ;CHANGE INDEX TO POINT TO NEXT REGISTER
3752 025022 077405          SOB    R4,1$         ;BRANCH BACK 3 TIMES
3753 025024 012704 000004          MOV     #4,R4         ;DO NEXT LOOP 4 TIMES ALSO
3754 025030 012737 025042 001110          MOV     #10$, $LPERR  ;SET LOOP ON ERROR POINTER TO 10$
3755 025036 162700 000002          SUB     #2,R0         ;ADJUST INDEX FOR REGISTER DESIRED
3756 025042 017001 001450          MOV     $PARTAB(R0),R1;READ PAR OR PDR INTO R1
3757 025046 020001          CMP     R0,R1         ;SEE IF INDEX EQUALS DATA
3758 025050 001403          BEQ     35$           ;BRANCH IFCORRECT REGISTER WAS READ.
3759 025052 016002 001450          MOV     PARTAB(R0),R2 ;SAVE ADDRESS OF BAD REGISTER
3760 025056 104030          ERROR   30            ;DUAL ADDRESSING
3761 025060 077412          SOB    R4,25$        ;BRANCH BACK 3 TIMES
3762 025062 012737 024776 001110 35$:  MOV     #20$, $LPERR  ;SET LOOP ON ERROR POINTER TO START OF TEST
3763
3764
3765 .SBTTL      ***** ENTRY POINT 4 --- STARTING ADDRESS 214 *****
3766 .SBTTL      ***** RELOCATION AND ADDER TESTS *****
3767 :*
3768 :*          THIS GROUP OF TESTS CHECKS MEMORY MANAGEMENT'S RELOCATION ADDER

```

```

3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783 025070
3784 025070 000004
3785 025072 012737 026632 001444
3786
3787
3788
3789
3790
3791
3792 025100
3793 025100 012737 025162 001110
3794 025106 012737 000031 001102
3795 025114 013777 001102 154020
3796 025122 012705 172300
3797 025126 004737 044236
3798 025132 012705 172340
3799 025136 004737 044236
3800 025142 012705 177600
3801 025146 004737 044236
3802 025152 012705 177640
3803 025156 004737 044236
3804 025162 012700 077406 20$:
3805 025166 012702 000010
3806 025172 012701 172300
3807 025176 010021 64$:
3808 025200 077202
3809 025202 012737 000000 172340
3810 025210 012737 000200 172342
3811 025216 012737 000400 172344
3812 025224 012737 000600 172346
3813 025232 012737 007600 172356
3814 025240 012737 025306 001110
3815 025246 013737 060000 001176
3816 025254 012737 000600 172350
3817 025262 012737 000600 001200
3818 025270 012737 060000 001202
3819 025276 012700 100000
3820 025302 012701 125200
3821 025306 052737 000400 177572 1$:
3822 025314 010110
3823 025316 013702 060000
3824 025322 000005

```

```

:* FIRST USING DESTINATION ONLY RELOCATION, THEN WITH FULL 18-BIT
:* RELOCATION .
:*

```

```

*****
:TEST 31 18-BIT MAPPING ADDER TESTING

```

```

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

```

```

*****
TST31:

```

```

SCOPE
MOV #TST32,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS

```

```

:
: THE FOLLOWING CODE WILL CLEAR ALL PAR'S AND PDR'S SO THAT
: THE RELOCATION ADDERS CAN BE CHECKED, ONLY THE KERNEL
: PDR'S AND PAR'S WILL BE MAPPED RESIDENT AND READ WRITE.

```

```

ENTPT4:

```

```

MOV #20$,SLPERR ;SET LOOP ON ERROR POINTER TO 20$
MOV #31,$STSTNM ;LOAD TEST NUMBER INTO MEMORY
MOV $STSTNM,$DISPLAY ;DISPLAY TEST NUMBER FOR THIS TEST
MOV #KIPDR0,R5 ;PUT ADDRESS OF KIPDR0 IN R5
JSR PC,CLRREG ;CLEAR KERNEL PDR'S
MOV #KIPAR0,R5 ;PUT ADDRESS OF KIPAR0 IN R5
JSR PC,CLRREG ;CLEAR KERNEL PAR'S
MOV #UIPDR0,R5 ;PUT ADDRESS OF UIPDR0 IN R5
JSR PC,CLRREG ;CLEAR USER PDR'S
MOV #UIPAR0,R5 ;PUT ADDRESS OF UIPAR0 IN R5
JSR PC,CLRREG ;CLEAR USER PAR'S
20$: MOV #77406,R0 ;MAKE KERNEL I PAGES 4K, R/W, EXPAND UP
MOV #10,R2 ;SET COUNT TO LOAD 8 ADDRESSES
MOV #KIPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
64$: MOV R0,(R1)+ ;LOAD R0 INTO PDR ADDRESSED BY R1
SOB R2,64$ ;BRANCH BACK TO 64$ IF R2 IS NOT ZERO
MOV #000,KIPAR0 ;MAP PAGE 0 TO 0 - 4K
MOV #200,KIPAR1 ;MAP PAGE 1 TO 4K - 8K
MOV #400,KIPAR2 ;MAP PAGE 2 TO 8K - 12K
MOV #600,KIPAR3 ;MAP PAGE 3 TO 12K - 16K
MOV #7600,KIPAR7 ;MAP PAGE 7 TO I/O PAGE
MOV #1$,SLPERR ;SET LOOP ON ERROR POINTER TO 1$
MOV @#60000,$TMP0 ;SAVE DATA AT TEST LOCATION
MOV #600,@#KIPAR4 ;LOAD PAR4 WITH 600
MOV #600,$TMP1 ;SAVE 600 FOR ERROR REPORT
MOV #60000,$TMP2 ;SAVE 60000 FOR ERROR REPORT
MOV #100000,R0 ;PUT VIRTUAL ADDRESS IN R0
MOV #125200,R1 ;PUT DATA PATTERN IN R1
BIS #BIT8,@#MMRO ;TURN ON DESTINATION ONLY RELOCATION
MOV R1,(R0) ;TRY TO LOAD DATA PATTERN INTO 60000
MOV @#60000,R2 ;READ (60000) INTO R2
RESET ;CLEAR MMRO

```

K06

3825	025324	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
3826	025326	001401				BEQ	2\$:BRANCH IF DATA MATCHES
3827	025330	104031				ERROR	31	:RELOCATION FAILED
3828	025332	013737	001176	060000	2\$:	MOV	\$TMP0,2#60000	:RESTORE ORIGINAL DATA TO TEST LOCATION
3829	025340	012737	025406	001110		MOV	#7\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 7\$
3830	025346	013737	062000	001176		MOV	2#62000, \$TMP0	:SAVE DATA AT TEST LOCATION
3831	025354	012737	000610	172350		MOV	#610,2#KIPAR4	:LOAD PAR4 WITH 610
3832	025362	012737	000610	001200		MOV	#610, \$TMP1	:SAVE 610 FOR ERROR REPORT
3833	025370	012737	062000	001202		MOV	#62000, \$TMP2	:SAVE 62000 FOR ERROR REPORT
3834	025376	012700	101000			MOV	#101000, R0	:PUT VIRTUAL ADDRESS IN R0
3835	025402	012701	125203			MOV	#125203, R1	:PUT DATA PATTERN IN R1
3836	025406	052737	000400	177572	7\$:	BIS	#BIT8,2#MMRO	:TURN ON DESTINATION ONLY RELOCATION
3837	025414	010110				MOV	R1, (R0)	:TRY TO LOAD DATA PATTERN INTO 62000
3838	025416	013702	062000			MOV	2#62000, R2	:READ (62000) INTO R2
3839	025422	000005				RESET		:CLEAR MMRO
3840	025424	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
3841	025426	001401				BEQ	8\$:BRANCH IF DATA MATCHES
3842	025430	104031				ERROR	31	:RELOCATION FAILED
3843	025432	013737	001176	062000	8\$:	MOV	\$TMP0,2#62000	:RESTORE ORIGINAL DATA TO TEST LOCATION
3844	025440	012737	025506	001110		MOV	#9\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 9\$
3845	025446	013737	062000	001176		MOV	2#62000, \$TMP0	:SAVE DATA AT TEST LOCATION
3846	025454	012737	000514	172350		MOV	#514,2#KIPAR4	:LOAD PAR4 WITH 514
3847	025462	012737	000514	001200		MOV	#514, \$TMP1	:SAVE 514 FOR ERROR REPORT
3848	025470	012737	062000	001202		MOV	#62000, \$TMP2	:SAVE 62000 FOR ERROR REPORT
3849	025476	012700	110400			MOV	#110400, R0	:PUT VIRTUAL ADDRESS IN R0
3850	025502	012701	125204			MOV	#125204, R1	:PUT DATA PATTERN IN R1
3851	025506	052737	000400	177572	9\$:	BIS	#BIT8,2#MMRO	:TURN ON DESTINATION ONLY RELOCATION
3852	025514	010110				MOV	R1, (R0)	:TRY TO LOAD DATA PATTERN INTO 62000
3853	025516	013702	062000			MOV	2#62000, R2	:READ (62000) INTO R2
3854	025522	000005				RESET		:CLEAR MMRO
3855	025524	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
3856	025526	001401				BEQ	10\$:BRANCH IF DATA MATCHES
3857	025530	104031				ERROR	31	:RELOCATION FAILED
3858	025532	013737	001176	062000	10\$:	MOV	\$TMP0,2#62000	:RESTORE ORIGINAL DATA TO TEST LOCATION
3859	025540	012737	025606	001110		MOV	#11\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 11\$
3860	025546	013737	062000	001176		MOV	2#62000, \$TMP0	:SAVE DATA AT TEST LOCATION
3861	025554	012737	000504	172350		MOV	#504,2#KIPAR4	:LOAD PAR4 WITH 504
3862	025562	012737	000504	001200		MOV	#504, \$TMP1	:SAVE 504 FOR ERROR REPORT
3863	025570	012737	062000	001202		MOV	#62000, \$TMP2	:SAVE 62000 FOR ERROR REPORT
3864	025576	012700	111400			MOV	#111400, R0	:PUT VIRTUAL ADDRESS IN R0
3865	025602	012701	125205			MOV	#125205, R1	:PUT DATA PATTERN IN R1
3866	025606	052737	000400	177572	11\$:	BIS	#BIT8,2#MMRO	:TURN ON DESTINATION ONLY RELOCATION
3867	025614	010110				MOV	R1, (R0)	:TRY TO LOAD DATA PATTERN INTO 62000
3868	025616	013702	062000			MOV	2#62000, R2	:READ (62000) INTO R2
3869	025622	000005				RESET		:CLEAR MMRO
3870	025624	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
3871	025626	001401				BEQ	12\$:BRANCH IF DATA MATCHES
3872	025630	104031				ERROR	31	:RELOCATION FAILED
3873	025632	013737	001176	062000	12\$:	MOV	\$TMP0,2#62000	:RESTORE ORIGINAL DATA TO TEST LOCATION
3874	025640	012737	025706	001110		MOV	#13\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 13\$
3875	025646	013737	062000	001176		MOV	2#62000, \$TMP0	:SAVE DATA AT TEST LOCATION
3876	025654	012737	000556	172350		MOV	#556,2#KIPAR4	:LOAD PAR4 WITH 556
3877	025662	012737	000556	001200		MOV	#556, \$TMP1	:SAVE 556 FOR ERROR REPORT
3878	025670	012737	062000	001202		MOV	#62000, \$TMP2	:SAVE 62000 FOR ERROR REPORT
3879	025676	012700	104200			MOV	#104200, R0	:PUT VIRTUAL ADDRESS IN R0
3880	025702	012701	125206			MOV	#125206, R1	:PUT DATA PATTERN IN R1

3881	025706	052737	000400	177572	13\$:	BIS	#BIT8, @#MMRO	:TURN ON DESTINATION ONLY RELOCATION
3882	025714	010110				MOV	R1, (R0)	:TRY TO LOAD DATA PATTERN INTO 62000
3883	025716	013702	062000			MOV	@#62000, R2	:READ (62000) INTO R2
3884	025722	000005				RESET		:CLEAR MMRO
3885	025724	020102				CMP	R1, R2	:SEE IF DATA MATCHES PATTERN
3886	025726	001401				BEQ	14\$:BRANCH IF DATA MATCHES
3887	025730	104031				ERROR	31	:RELOCATION FAILED
3888	025732	013737	001176	062000	14\$:	MOV	\$TMP0, @#62000	:RESTORE ORIGINAL DATA TO TEST LOCATION
3889	025740	012737	026006	001110		MOV	#15\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 15\$
3890	025746	013737	062000	001176		MOV	@#62000, \$TMP0	:SAVE DATA AT TEST LOCATION
3891	025754	012737	000442	172350		MOV	#442, @#KIPAR4	:LOAD PAR4 WITH 442
3892	025762	012737	000442	001200		MOV	#442, \$TMP1	:SAVE 442 FOR ERROR REPORT
3893	025770	012737	062000	001202		MOV	#62000, \$TMP2	:SAVE 62000 FOR ERROR REPORT
3894	025776	012700	115600			MOV	#115600, R0	:PUT VIRTUAL ADDRESS IN R0
3895	026002	012701	125207			MOV	#125207, R1	:PUT DATA PATTERN IN R1
3896	026006	052737	000400	177572	15\$:	BIS	#BIT8, @#MMRO	:TURN ON DESTINATION ONLY RELOCATION
3897	026014	010110				MOV	R1, (R0)	:TRY TO LOAD DATA PATTERN INTO 62000
3898	026016	013702	062000			MOV	@#62000, R2	:READ (62000) INTO R2
3899	026022	000005				RESET		:CLEAR MMRO
3900	026024	020102				CMP	R1, R2	:SEE IF DATA MATCHES PATTERN
3901	026026	001401				BEQ	16\$:BRANCH IF DATA MATCHES
3902	026030	104031				ERROR	31	:RELOCATION FAILED
3903	026032	013737	001176	062000	16\$:	MOV	\$TMP0, @#62000	:RESTORE ORIGINAL DATA TO TEST LOCATION
3904	026040	012737	026106	001110		MOV	#17\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 17\$
3905	026046	013737	062000	001176		MOV	@#62000, \$TMP0	:SAVE DATA AT TEST LOCATION
3906	026054	012737	000577	172350		MOV	#577, @#KIPAR4	:LOAD PAR4 WITH 577
3907	026062	012737	000577	001200		MOV	#577, \$TMP1	:SAVE 577 FOR ERROR REPORT
3908	026070	012737	062000	001202		MOV	#62000, \$TMP2	:SAVE 62000 FOR ERROR REPORT
3909	026076	012700	102100			MOV	#102100, R0	:PUT VIRTUAL ADDRESS IN R0
3910	026102	012701	125210			MOV	#125210, R1	:PUT DATA PATTERN IN R1
3911	026106	052737	000400	177572	17\$:	BIS	#BIT8, @#MMRO	:TURN ON DESTINATION ONLY RELOCATION
3912	026114	010110				MOV	R1, (R0)	:TRY TO LOAD DATA PATTERN INTO 62000
3913	026116	013702	062000			MOV	@#62000, R2	:READ (62000) INTO R2
3914	026122	000005				RESET		:CLEAR MMRO
3915	026124	020102				CMP	R1, R2	:SEE IF DATA MATCHES PATTERN
3916	026126	001401				BEQ	18\$:BRANCH IF DATA MATCHES
3917	026130	104031				ERROR	31	:RELOCATION FAILED
3918	026132	013737	001176	062000	18\$:	MOV	\$TMP0, @#62000	:RESTORE ORIGINAL DATA TO TEST LOCATION
3919	026140	012737	026206	001110		MOV	#21\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 21\$
3920	026146	013737	062000	001176		MOV	@#62000, \$TMP0	:SAVE DATA AT TEST LOCATION
3921	026154	012737	000421	172350		MOV	#421, @#KIPAR4	:LOAD PAR4 WITH 421
3922	026162	012737	000421	001200		MOV	#421, \$TMP1	:SAVE 421 FOR ERROR REPORT
3923	026170	012737	062000	001202		MOV	#62000, \$TMP2	:SAVE 62000 FOR ERROR REPORT
3924	026176	012700	117700			MOV	#117700, R0	:PUT VIRTUAL ADDRESS IN R0
3925	026202	012701	125211			MOV	#125211, R1	:PUT DATA PATTERN IN R1
3926	026206	052737	000400	177572	21\$:	BIS	#BIT8, @#MMRO	:TURN ON DESTINATION ONLY RELOCATION
3927	026214	010110				MOV	R1, (R0)	:TRY TO LOAD DATA PATTERN INTO 62000
3928	026216	013702	062000			MOV	@#62000, R2	:READ (62000) INTO R2
3929	026222	000005				RESET		:CLEAR MMRO
3930	026224	020102				CMP	R1, R2	:SEE IF DATA MATCHES PATTERN
3931	026226	001401				BEQ	22\$:BRANCH IF DATA MATCHES
3932	026230	104031				ERROR	31	:RELOCATION FAILED
3933	026232	013737	001176	062000	22\$:	MOV	\$TMP0, @#62000	:RESTORE ORIGINAL DATA TO TEST LOCATION
3934	026240	012737	026306	001110		MOV	#23\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 23\$
3935	026246	013737	060000	001176		MOV	@#60000, \$TMP0	:SAVE DATA AT TEST LOCATION
3936	026254	012737	000577	172350		MOV	#577, @#KIPAR4	:LOAD PAR4 WITH 577

3937	026262	012737	000577	001200		MOV	#577,\$TMP1	SAVE 577 FOR ERROR REPORT
3938	026270	012737	060000	001202		MOV	#60000,\$TMP2	SAVE 60000 FOR ERROR REPORT
3939	026276	012700	100100			MOV	#100100,R0	PUT VIRTUAL ADDRESS IN R0
3940	026302	012701	125212			MOV	#125212,R1	PUT DATA PATTERN IN R1
3941	026306	052737	000400	177572	23\$:	BIS	#BITS,\$MMRO	TURN ON DESTINATION ONLY RELOCATION
3942	026314	010110				MOV	R1,(R0)	TRY TO LOAD DATA PATTERN INTO 60000
3943	026316	013702	060000			MOV	\$#60000,R2	READ (60000) INTO R2
3944	026322	000005				RESET		CLEAR MMRO
3945	026324	020102				CMP	R1,R2	SEE IF DATA MATCHES PATTERN
3946	026326	001401				BEQ	24\$	BRANCH IF DATA MATCHES
3947	026330	104031				ERROR	31	RELOCATION FAILED
3948	026332	013737	001176	060000	24\$:	MOV	\$TMP0,\$#60000	RESTORE ORIGINAL DATA TO TEST LOCATION
3949	026340	012737	026406	001110		MOV	#25\$,\$LPERR	SET LOOP ON ERROR POINTER TO 25\$
3950	026346	012737	060000	001176		MOV	\$#60000,\$TMP0	SAVE DATA AT TEST LOCATION
3951	026354	012737	000570	172350		MOV	#570,\$KIPAR4	LOAD PAR4 WITH 570
3952	026362	012737	000570	001200		MOV	#570,\$TMP1	SAVE 570 FOR ERROR REPORT
3953	026370	012737	060000	001202		MOV	#60000,\$TMP2	SAVE 60000 FOR ERROR REPORT
3954	026376	012700	101000			MOV	#101000,R0	PUT VIRTUAL ADDRESS IN R0
3955	026402	012701	125213			MOV	#125213,R1	PUT DATA PATTERN IN R1
3956	026406	052737	000400	177572	25\$:	BIS	#BITS,\$MMRO	TURN ON DESTINATION ONLY RELOCATION
3957	026414	010110				MOV	R1,(R0)	TRY TO LOAD DATA PATTERN INTO 60000
3958	026416	013702	060000			MOV	\$#60000,R2	READ (60000) INTO R2
3959	026422	000005				RESET		CLEAR MMRO
3960	026424	020102				CMP	R1,R2	SEE IF DATA MATCHES PATTERN
3961	026426	001401				BEQ	26\$	BRANCH IF DATA MATCHES
3962	026430	104031				ERROR	31	RELOCATION FAILED
3963	026432	013737	001176	060000	26\$:	MOV	\$TMP0,\$#60000	RESTORE ORIGINAL DATA TO TEST LOCATION
3964						***		
3965						***		
3966						***		
3967						***		
3968						***		
3969						***		
3970						***		
3971	026440	012737	026500	001110		MOV	#27\$,\$LPERR	SET LOOP ON ERROR POINTER TO 27\$
3972	026446	012737	007600	172350		MOV	#7600,\$KIPAR4	LOAD PAR4 WITH 7600
3973	026454	012737	007600	001200		MOV	#7600,\$TMP1	SAVE PAR4 DATA FOR ERROR REPORT
3974	026462	012737	177776	001202		MOV	#177776,\$TMP2	SAVE PHYS. ADDR. FOR ERROR REPORT
3975	026470	012700	117776			MOV	#117776,R0	PUT VIRTUAL ADDR. IN R0
3976	026474	012701	000100			MOV	#100,R1	PUT DATA PATTERN IN R1
3977	026500	052737	000400	177572	27\$:	BIS	#BITS,\$MMRO	TURN ON DEST.-ONLY-RELOCATION
3978	026506	010110				MOV	R1,(R0)	LOAD DATA INTO PSW USING PAR4
3979	026510	013702	177776			MOV	\$#177776,R2	READ PSW INTO R2
3980	026514	000005				RESET		CLEAR MMRO
3981	026516	042702	000020			BIC	#20,R2	CLEAR T-BIT IF SET IN DATA READ
3982	026522	020102				CMP	R1,R2	SEE IF DATA MATCHES PATTERN WRITTEN
3983	026524	001401				BEQ	28\$	BRANCH IF DATA MATCHES
3984	026526	104031				ERROR	31	RELOCATION FAILED
3985	026530	012737	026570	001110	28\$:	MOV	#29\$,\$LPERR	SET LOOP ON ERROR POINTER TO 29\$
3986	026536	012737	007777	172350		MOV	#7777,\$KIPAR4	LOAD PAR4 WITH 7777
3987	026544	012737	007777	001200		MOV	#7777,\$TMP1	SAVE PAR4 DATA FOR ERROR REPORT
3988	026552	012737	177776	001202		MOV	#177776,\$TMP2	SAVE PHYS. ADDR. FOR ERROR REPORT
3989	026560	012700	100076			MOV	#100076,R0	PUT VIRTUAL ADDR. R0
3990	026564	012701	000200			MOV	#200,R1	PUT DATA PATTERN IN R1
3991	026570	052737	000400	177572	29\$:	BIS	#BITS,\$MMRO	TURN ON DEST.-ONLY-RELOCATION
3992	026576	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.

3993	026600	013702	177776		MOV	2#177776,R2	;READ PSW INTO R2
3994	026604	000005			RESET		;CLEAR MMRO
3995	026606	042702	000020		BIC	#20,R2	;CLEAR T-BIT IF SET IN DATA READ
3996	026612	020102			CMP	R1,R2	;SEE IF DATA MATCHES PATTERN
3997	026614	001401			BEQ	30\$;BRANCH IF DATA MATCHES
3998	026616	104031			ERROR	31	;RELOCATION FAILED
3999	026620						
4000	026620	005037	177776		30\$: CLR	2#PSW	;CLEAR PSW BEFORE LEAVING TEST
4001	026624	012737	025162	001110	MOV	#20\$,SLPERR	;SET LOOP ON ERROR POINTER TO START OF TEST
4002							
4003							
4004							
4005							
4006							
4007							
4008							
4009							
4010							
4011							
4012							
4013							
4014	026632						
4015	026632	000004					
4016	026634	012737	027200	001444			
4017							
4018	026642	005037	001432		20\$: CLR	HOLFLG	;SAVE STARTING ADDRESS OF NEXT
4019	026646	005037	001200		CLR	\$TMP1	;TEST FOR ESCAPE ON PARITY ERRORS
4020	026652	012737	027170	000004	MOV	#11\$,ERRVEC	;MAKE SURE HOLE FLAG STARTS AT 0
4021	026660	105737	001446		TSTB	FORTY	;CLEAR SPOT TO SAVE HOLE ADDRESS
4022	026664	001403			BEQ	7\$;SET ERROR POINTER TO 11\$ FOR 11/6X
4023	026666	012737	027160	000004	MOV	#10\$,ERRVEC	;EXECUTING ON AN 11/40
4024	026674						;BRANCH IF NO
4025	026674	012737	000577	172350	7\$: MOV	#577,KIPAR4	;SET ERROR POINTER TO 10\$ FOR 11/40
4026	026702	012737	000600	172352	MOV	#600,KIPAR5	;LOAD PAR4 WITH STARTING BASE
4027	026710	012700	100100		MOV	#100100,R0	;START ADDRESSING WITH 12K
4028	026714	012701	120000		MOV	#120000,R1	;LOAD VIRTUAL ADDR FOR PAR4 IN R0
4029	026720	012702	001000		MOV	#1000,R2	;LOAD VIRTUAL ADDR FOR PAR5 IN R1
4030	026724	012737	000001	177572	MOV	#1,2#MMRO	;LOAD DATA PATTERN INTO R2
4031							;TURN ON 18-BIT MAPPING
4032							
4033							
4034	026732	012737	027030	001110	1\$: MOV	#3\$,SLPERR	;FULL RELOCATION STARTS HERE AND CONTINUES FOR REST OF PROGRAM
4035	026740	005037	001406		CLR	PCPUER	;SET LOOP ON ERROR POINTER TO 3\$
4036	026744	011137	001176		MOV	(R1), \$TMP0	;CLEAR CPU ERROR FLAG
4037	026750	032737	000020	001406	BIT	#BIT4,PCPUER	;SAVE DATA AT TEST LOCATION
4038	026756	001411			BEQ	2\$;SEE IF THERE WAS A CPU TRAP
4039	026760	005237	001432		INC	HOLFLG	;BRANCH IF NO TIMEOUT
4040	026764	005737	001200		TST	\$TMP1	;SET FLAG IF THERE WAS TIMEOUT
4041	026770	001034			BNE	5\$;HAVE I SAVED STARTING ADDR. OF HOLE
4042	026772	013737	172352	001200	MOV	KIPAR5,\$TMP1	;IF I HAVE I'LL CONTINUE
4043	027000	000430			BR	5\$;SAVE PAR-POSSIBLE START OF HOLE
4044	027002	005737	001432		2\$: TST	HOLFLG	;BRANCH TO CHANGE BASE ADDRESS
4045	027006	001410			BEQ	3\$;IF HOLFLG CLEAR-NO TIMEOUTS YET
4046	027010	013737	172352	001202	MOV	KIPAR5,\$TMP2	;OTHERWISE HAVE FOUND MEMORY HOLE
4047	027016	104032			ERROR	32	;SAVE PAR THAT POINTS TO END OF HOLE
4048	027020	005037	001432		CLR	HOLFLG	;HOLE IN MEMORY FROM \$TMP1 TO \$TMP2

```

4049 027024 005037 001200
4050 027030 010210
4051 027032 011103
4052 027034 020203
4053 027036 001407
4054 027040 013737 172350 001204
4055 027046 013737 172352 001206
4056 027054 104033
4057 027056 013711 001176
4058 027062 062737 000100 172350
4059 027070 062737 000100 172352
4060 027076 005202
4061 027100 022737 007500 172352
4062 027106 103311
4063
4064
4065
4066
4067
4068
4069
4070
4071
4072 027110 012737 027130 001110
4073 027116 012737 007777 172350
4074 027124 005037 000000
4075 027130 013701 100100
4076 027134 005701
4077 027136 001401
4078 027140 104034
4079 027142 012737 015160 000004
4080 027150 012737 026642 001110
4081 027156 000410
4082
4083
4084
4085 027160 012737 000020 001406
4086 027166 000002
4087 027170 013737 177766 001406
4088 027176 000002
4089
4090
4091
4092
4093
4094
4095
4096
4097
4098
4099
4100
4101
4102
4103
4104

3$: CLR $TMP1 ;CLEAR STARTING ADDR. LOC. ALSO
MOV R2,(R0) ;LOAD TEST PATTERN INTO TEST LOCATION
MOV (R1),R3 ;READ TEST LOCATION VIA DIFFERENT PAR
CMP R2,R3 ;SEE IF CORRECT LOCATION REFERENCED
BEQ 4$ ;BRANCH IF CORRECT DATA OBTAINED
MOV KIPAR4,$TMP3 ;SAVE PAR4 FOR ERROR REPORT
MOV KIPAR5,$TMP4 ;SAVE PAR5 FOR ERROR REPORT
ERROR 33 ;OTHERWISE BAD RELOCATION
4$: MOV $TMP0,(R1) ;RESTORE ORIGINAL DATA TO TEST LOCATION
5$: ADD #100,KIPAR4 ;CHANGE BASE ADDRESS
ADD #100,KIPAR5 ;CHANGE BASE ADDRESS
INC R2 ;CHANGE DATA PATTERN
CMP #7500,KIPAR5 ;SEE IF PAST 122K-LAST ADDRESS
BHS 1$ ;BRANCH IF NOT PAST LAST ADDRESS

;*
;* ADDRESS 000000 IS GENERATED BY CARRY PROPAGATION, USING
;* '7777' IN KIPAR4 WITH A VIRTUAL ADDRESS OF '100100'.
;*
;*
;*
;*
;*
;*
6$: MOV #16$,$LPERR ;SET LOOP ON ERROR POINTER TO 16$
MOV #7777,KIPAR4 ;LOAD PAR 4 WITH HIGHEST VALUE POSSIBLE
CLR #000000 ;CLEAR ADDRESS ZERO
16$: MOV @#100100,R1 ;THIS SHOULD READ ADDRESS ZERO INTO R1
TST R1 ;SEE IF YOU REALLY READ ADDRESS ZERO
BEQ 8$ ;BRANCH IF YOU READ ADDRESS ZERO
ERROR 34 ;DIDN'T READ ADDRESS ZERO
8$: MOV #CPUER,ERRVEC ;RESTORE NORMAL CPU TRAP ROUTINE
MOV #20$,$LPERR ;SET LOOP POINTER TO START OF TEST
BR TST33 ;BRANCH TO NEXT TEST

;***** TRAP TO HERE THRU ERRVEC *****

10$: MOV #BIT4,PCPUER ;SET "TRAP-TO-4" FLAG
RTI
11$: MOV @#CPUERR,PCPUER ;SAVE CPU ERROR REGISTER
RTI ;RETURN TO TEST AND CONTINUE

.SBTTL ***** ENTRY POINT 5 --- STARTING ADDRESS 220 *****
.SBTTL ***** MEMORY MANAGEMENT ABORTS LOGIC TESTS *****
;*
;* THIS GROUP OF TESTS CHECKS OUT THE MEMORY MANAGEMENT ABORT LOGIC, AND
;* ALSO CHECK OUT BITS <06:05> AND <03:00> OF MMRO, AND ALL BITS OF MMR2.
;*
;*****
;*TEST 33 PAGE LENGTH FAULTS - UPWARD EXPANSION
;*
```



```

4105
4106
4107
4108
4109
4110
4111
4112
4113 027200
4114 027200 000004
4115 027202 012737 027524 001444
4116
4117 027210 012737 000024 001212
4118 027216
4119 027216 012737 027342 001110
4120 027224 012737 000033 001102
4121 027232 013777 001102 151702
4122 027240 012737 077406 172300
4123 027246 012737 077406 172302
4124 027254 012737 077406 172304
4125 027262 012737 077406 172306
4126 027270 012737 077406 172316
4127 027276 012737 000000 172340
4128 027304 012737 000200 172342
4129 027312 012737 000400 172344
4130 027320 012737 000600 172346
4131 027326 012737 007600 172356
4132 027334 012737 000001 177572
4133 027342 012737 000006 172310
4134 027350 012737 000600 172350
4135 027356 012700 172311
4136 027362 012737 027424 001110
4137 027370 005037 001402
4138 027374 012702 100000
4139 027400 010203
4140 027402 006203
4141 027404 006203
4142 027406 006203
4143 027410 006203
4144 027412 006203
4145 027414 006203
4146 027416 042703 177600
4147 027422 111001
4148 027424 012737 040011 001364
4149
4150 027432 011204
4151 027434 005037 001364
4152 027440 005737 001402
4153 027444 001405
4154 027446 005303
4155 027450 020103
4156 027452 001414
4157 027454 104035
4158 027456 000412
4159 027460 020103
4160 027462 103002
    
```

```

; * 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.
; *
; * *****
; * TST33:
; * SCOPE
; * MOV #TST34,NXTTST ;SAVE STARTING ADDRESS OF NEXT
; * ;TEST FOR ESCAPE ON PARITY ERRORS
; * MOV #24,$TIMES ;DO 24 ITERATIONS
; * ENTPTS:
; * MOV #20$, $LPERR ;SET LOOP ON ERROR POINTER TO 20$
; * MOV #33,$STNM ;LOAD TEST NUMBER INTO MEMORY
; * STNM, @DISPLAY ;DISPLAY TEST NUMBER FOR THIS TEST
; * MOV #77406, KIPDR0 ;MAKE KERNEL I PAGE 0 200 BLOCKS, R/W
; * MOV #77406, KIPDR1 ;MAKE KERNEL I PAGE 1 200 BLOCKS, R/W
; * MOV #77406, KIPDR2 ;MAKE KERNEL I PAGE 2 200 BLOCKS, R/W
; * MOV #77406, KIPDR3 ;MAKE KERNEL I PAGE 3 200 BLOCKS, R/W
; * MOV #77406, KIPDR7 ;MAKE KERNEL I PAGE 7 200 BLOCKS, R/W
; * MOV #000, KIPAR0 ;MAP KERNEL I PAGE 0 TO 0 - 4K
; * MOV #200, KIPAR1 ;MAP KERNEL I PAGE 1 TO 4K - 8K
; * MOV #400, KIPAR2 ;MAP KERNEL I PAGE 2 TO 8K - 12K
; * MOV #600, KIPAR3 ;MAP KERNEL I PAGE 3 TO 12K - 16K
; * MOV #7600, KIPAR7 ;MAP KERNEL I PAGE 7 TO THE I/O PAGE
; * MOV #BIT0, MMRO ;ENABLE 18-BIT RELOCATION IF NOT ON
; * 20$: 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$: MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
; * CLR PMMRO ;CLEAR LOCATION THAT HOLDS MMRO
; * MOV #100000, R2 ;PUT VIRTUAL ADDRESS INTO R2
; * 2$: MOV R2, R3 ;PUT VIRTUAL ADDRESS INTO R3 TOO
; * ASR R3 ;RIGHT SHIFT 6 BITS SO IT WILL
; * ASR R3 ;MATCH THE PLF
; * ASR R3
; * ASR R3
; * ASR R3
; * BIC #177600, R3 ;CLEAR BITS THAT ARE SET IN UPPER BYTE
; * MOV (R0), R1 ;SAVE PLF IN R1 FOR LATER COMPARISON
; * 3$: 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 PMMRO ;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 35 ;ABORT WRONG PLACE
; * BR 6$ ;BRANCH TO CHANGE PLF
; * 4$: CMP R1, R3 ;SEE IF PAGE LENGTH FAULT SHOULD HAVE OCCURRED
; * BHIS 5$ ;BRANCH IF NO PAGE LENGTH FAULT CONDITION
    
```

```

4161 027464 104036          ERROR 36          ;NO ABORT, IT SHOULD HAVE HAPPENED THIS TIME
4162 027466 000406          BR      6$          ;BRANCH TO CHANGE PLF
4163 027470 120327 000177 5$:  CMPB   R3,#177    ;SEE IF V.A. IS 177
4164 027474 001403          BEQ    6$          ;BRANCH TO CHECK PLF
4165 027476 062702 000100          ADD    #100,R2     ;CHANGE VIRTUAL ADDRESS
4166 027502 000736          BR      2$          ;GO TRY THE NEXT VIRTUAL ADDRESS
4167 027504 122710 000177 6$:  CMPB   #177,(R0)   ;SEE IF PDR 4'S PLF IS 177
4168 027510 001402          BEQ    7$          ;BRANCH TO EXIT IF PLF IS 177
4169 027512 105210          INCB   (R0)        ;STEP PLF OF PDR 4 UP BY 1
4170 027514 000725          BR      1$          ;BRANCH TO START WITH V.A. OF 0
4171 027516 012737 027342 001110 7$:  MOV    #20$,$LPERR ;SET LOOP POINTER TO START OF TEST
4172
4173
4174 ;*****
4175 ;TEST 34          PAGE LENGTH FAULTS - DOWNWARD EXPANSION
4176 ;
4177 ;THIS TEST CHECKS OUT THE PAGE LENGTH COMPARATORS
4178 ;AND THE LOGIC THAT GENERATES "PAGE LTH ERR". IT TRIES
4179 ;EVERY PAGE LENGTH FIELD FROM 1 BLOCK TO 200(8) BLOCKS. THE
4180 ;TEST THEN TRIES A REFERENCE AT EVERY 100 BYTES UNTIL AN ABORT
4181 ;OCCURS. IF THE ABORT HAPPENS TOO SOON OR IF NO ABORT HAPPENS AT
4182 ;THE CORRECT BLOCK NUMBER AN ERROR IS REPORTED.
4183 ;*****
4184 ;TST34:
4185 027524 000004          SCOPE
4186 027526 012737 027724 001444  MOV    #TST35,NXTTST ;SAVE STARTING ADDRESS OF NEXT
4187 ;TEST FOR ESCAPE ON PARITY ERRORS
4188 027534 012737 000024 001212  MOV    #24,$TIMES    ;DO 24 ITERATIONS
4189 027542 012737 077416 172310 20$:  MOV    #77416,$KIPDR4 ;LOAD PDR4 FOR PAGE LENGTH OF 1
4190 027550 012737 000600 172350  MOV    #600,KIPAR4   ;MAP PAGE 4 TO 12K
4191 027556 012700 172311  MOV    #KIPDR4+1,R0  ;PUT ADDRESS OF PDR 4'S UPPER BYTE IN R0
4192 027562 012737 027624 001110  MOV    #3$,$LPERR    ;SET LOOP ON ERROR POINTER TO 3$
4193 027570 005037 001402 1$:  CLR    PMMR0        ;CLEAR LOCATION THAT HOLDS MMRO
4194 027574 012702 117700  MOV    #117700,R2    ;PUT VIRTUAL ADDRESS INTO R2
4195 027600 010203 2$:  MOV    R2,R3        ;PUT VIRTUAL ADDRESS INTO R3 TOO
4196 027602 006203  ASR    R3            ;RIGHT SHIFT 6 BITS SO IT WILL
4197 027604 006203  ASR    R3            ;MATCH THE PLF
4198 027606 006203  ASR    R3
4199 027610 006203  ASR    R3
4200 027612 006203  ASR    R3
4201 027614 006203  ASR    R3
4202 027616 042703 177600  BIC    #177600,R3    ;CLEAR BITS THAT ARE SET IN UPPER BYTE
4203 027622 111001  MOVVB  (R0),R1        ;SAVE PLF IN R1 FOR LATER COMPARISON
4204 027624 012737 040011 001364 3$:  MOV    #040011,MMEXP ;POTENTIAL ABORT CONDITION: PAGE LENGTH
4205 ;ERROR KERNEL, I-SPACE, PAGE 4
4206 027632 011204  MOV    (R2),R4        ;READ USING V.A. IN R2
4207 027634 005037 001364  CLR    MMEXP        ;CLEAR EXPECTED ABORT CONDITION
4208 027640 005737 001402  TST    PMMR0        ;SEE IF ABORT OCCURRED YET
4209 027644 001405  BEQ    4$            ;BRANCH IF NO ABORT YET, CHANGE V.A.
4210 027646 005203  INC    R3            ;MAKE R3 EQUAL TO PLF
4211 027650 020103  CMP    R1,R3        ;SEE IF PDR 4'S PLF=R3
4212 027652 001414  BEQ    6$            ;BRANCH IF ABORT HAPPENS AT RIGHT PLACE
4213 027654 104035  ERROR 35            ;ABORT WRONG PLACE
4214 027656 000412  BR      6$            ;BRANCH TO CHANGE PLF
4215 027660 020103 4$:  CMP    R1,R3        ;SEE IF PAGE LENGTH FAULT SHOULD HAVE OCCURRED
4216 027662 101402  BLOS  5$            ;BRANCH IF NO PAGE LENGTH FAULT CONDITION

```

```

4217 027664 104036          ERROR 36          ;NO ABORT, IT SHOULD HAVE HAPPENED THIS TIME
4218 027666 000406          BR      6$          ;BRANCH TO CHANGE PLF
4219 027670 120327 000000 5$:  CMPB   R3,#000    ;SEE IF V.A. IS 000
4220 027674 001403          BEQ    6$          ;BRANCH TO CHECK PLF
4221 027676 162702 000100  SUB    #100,R2    ;CHANGE VIRTUAL ADDRESS
4222 027702 000736          BR      2$          ;GO TRY THE NEXT VIRTUAL ADDRESS
4223 027704 122710 000000 6$:  CMPB   #00C,(R0) ;SEE IF PDR 4'S PLF IS 000
4224 027710 001402          BEQ    7$          ;BRANCH TO EXIT IF PLF IS 000
4225 027712 105310          DECB   (R0)       ;STEP PLF OF PDR 4 UP BY 1
4226 027714 000725          BR      1$          ;BRANCH TO START WITH V.A. OF 0
4227 027716 012737 027542 001110 7$:  MOV    #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
4228
4229
4230
4231          ;*****
4232          ;TEST 35      ACCESS CONTROL FIELD = 0, OR 4 (ABORT ALL ACCESSES)
4233          ;
4234          ;      THESE A.C.F.'S ARE ALL NON-RESIDENT, ANY REFERENCE (READ
4235          ;      OR WRITE) TO A NON-RESIDENT PAGE SHOULD SET BIT 15 IN MMRO.
4236          ;      BITS <06:05> AND <03:00> IN MMRO WILL LOCK UP ALL AVAILABLE INFORMATION
4237          ;      ON THAT PAGE.  IN THIS CASE THE PAGE THAT CAUSES THE ABORT
4238          ;      IS KERNEL PAGE 5.  THE EXPECTED ERROR CODE IS 100013.
4239          ;*****
4240          ;TST35:
4241 027724          SCOPE
4242 027726 000004 012737 030222 001444  MOV    #TST36,NXTTST ;SAVE STARTING ADDRESS OF NEXT
4243          ;TEST FOR ESCAPE ON PARITY ERRORS
4244 027734 012737 077406 172310 20$:  MOV    #77406,#KIPDR4 ;LOAD ACF 6 INTO PDR 4
4245 027742 012737 077400 172312  MOV    #77400,#KIPDR5 ;LOAD ACF 0 INTO PDR5
4246
4247 027750 012737 000600 172350  MOV    #600,#KIPPAR4 ;LOAD 12K INTO PAR4
4248 027756 012737 000600 172352  MOV    #600,#KIPPAR5 ;LOAD 12K INTO PAR5
4249 027764 012700 100000  MOV    #100000,R0    ;LOAD VIRTUAL ADDRESS FOR PAR4 INTO R0
4250 027770 012701 120000  MOV    #120000,R1    ;LOAD VIRTUAL ADDRESS FOR PAR5 INTO R1
4251 027774 012702 161457  MOV    #161457,R2    ;LOAD DATA PATTERN INTO R2
4252 030000 012737 030010 001110 11$:  MOV    #1$, $LPERR  ;SET LOOP ON ERROR POINTER TO 1$
4253 030006 005010          CLR    (R0)       ;CLEAR MEMORY LOCATION 60000
4254 030010 012737 100013 001364 1$:  MOV    #100013,MMEXP ;LOAD EXPECTED ABORT CONDITION: NON-RESIDENT,
4255          ;KERNEL PAGE 5, FULL RELOCATION
4256 030016 010211          8$:  MOV    R2,(R1)    ;WRITE TO NON-RESIDENT OR UNUSED ACF
4257          ;SHOULD CAUSE ABORT
4258 030020 005037 001364          CLR    MMEXP     ;CLEAR EXPECTED ABORT CONDITION
4259
4260 030024 005710          TST    (R0)       ;SEE IF (60000) IS STILL ZERO
4261 030026 001411          BEQ    2$          ;BRANCH IF (60000) IS ZERO
4262 030030 013737 172312 001176  MOV    KIPDR5,$TMP0 ;SAVE KIPDR5 FOR ERROR REPORT
4263 030036 013737 172352 001200  MOV    KIPARS,$TMP1 ;SAVE KIPARS FOR ERROR REPORT
4264 030044 010137 001202  MOV    R1,$TMP2    ;SAVE VIRT ADR. FOR ERROR REPORT
4265 030050 104037          ERROR 37          ;ABORT DID NOT HAPPEN
4266 030052 012737 030064 001110 2$:  MOV    #4$, $LPERR  ;SET LOOP ON ERROR POINTER TO 4$
4267 030060 010210          MOV    R2,(R0)    ;LOAD DATA PATTERN INTO 60000
4268 030062 005004          CLR    R4         ;CLEAR REGISTER TO RECEIVE DATA
4269 030064 012737 100013 001364 4$:  MOV    #100013,MMEXP ;LOAD EXPECTED ABORT CONDITION: NON-RESIDENT,
4270          ;KERNEL, I-SPACE, PAGE 5, FULL RELOCATION
4271 030072 011104          9$:  MOV    (R1),R4    ;TRY TO READ (100000) INTO R4
4272          ;THIS SHOULD ABORT

```

```

4273 030074 005037 001364 CLR MMEXP ;EXPECTED ABORT CONDITION
4274
4275 030100 005704 TST R4 ;MAKE SURE R4 IS STILL 0
4276 030102 001411 BEQ 5$ ;BRANCH IF R4 IS 0
4277 030104 013737 172312 001176 MOV KIPDR5,$TMP0 ;SAVE KIPDR5 FOR ERROR REPORT
4278 030112 013737 172352 001200 MOV KIPARS,$TMP1 ;SAVE KIPARS FOR ERROR REPORT
4279 030120 010137 001202 MOV R1,$TMP2 ;SAVE VIRT ADR. FOR ERROR REPORT
4280 030124 104037 ERROR 37 ;ABORT DID NOT HAPPEN
4281 030126 5$:
4282 030126 023727 172312 077404 CMP KIPDR5,#77404 ;SEE IF PDR 5'S ACF=4
4283 030134 001404 BEQ 12$ ;TEST OVER IF ACF=4
4284 030136 012737 077404 172312 10$: MOV #77404,KIPDR5 ;MAKE PDR 5'S ACF=4
4285 030144 000715 BR 11$ ;REPEAT TEST WITH ACF=4
4286
4287
4288
4289
4290 030146 012737 030174 001110 12$: MOV #13$,$LPERR ;SET LOOP ON ERROR POINTER TO 13$
4291 030154 012737 007600 172352 MOV #7600,KIPARS ;MAP PAGE 5 TO I/O PAGE
4292 030162 012701 132350 MOV #132350,R1 ;LOAD VIRTUAL ADDRESS TO REFERENCE
4293
4294 030166 012737 100013 001364 MOV #100013,MMEXP ;KIPAR4
4295
4296 030174 005011 13$: CLR (R1) ;EXPECTING NON-RESIDENT ABORT
4297
4298
4299
4300 030176 022737 000600 172350 CMP #600,KIPAR4 ;KERNEL I PAGE 5
4301 030204 001401 BEQ 14$ ;THIS INSTRUCTION SHOULD ABORT
4302 030206 104074 ERROR 74 ;DURING THE ABORT 'NON RES ERR' IS
4303 030210 012737 027734 001110 14$: MOV #20$,$LPERR ;ASSERTED TO STOP THE CLOCKING OF THE
4304 030216 005037 001364 CLR MMEXP ;FLIP/FLOPS
4305
4306
4307
4308
4309
4310
4311
4312
4313
4314
4315
4316
4317 030222
4318 030222 000004 TST36: SCOPE
4319 030224 012737 030466 001444 MOV #TST37,NXTTST ;SAVE STARTING ADDRESS OF NEXT
4320
4321 030232 20$:
4322 030232 012737 000600 172352 MOV #600,KIPARS ;TEST FOR ESCAPE ON PARITY ERRORS
4323 030240 012737 000600 172350 MOV #600,KIPAR4 ;MAP PAGE 5 TO 12K
4324 030246 012737 077406 172312 MOV #77406,KIPDR5 ;MAP PAGE 4 TO 12K
4325
4326
4327 030254 012737 077402 172310 MOV #77402,KIPDR4 ;PAGE 5 IS 200 BLOCKS LONG
4328 030262 005037 001364 CLR MMEXP ;EXPANDS UPWARD, AND IS READ/WRITE
;WITH NO TRAPPING
;LOAD ACF 2 INTO PDR 4
;NOT EXPECTING ANY TRAPS OR ABORTS YET

```

THIS SECTION OF CODE VERIFIES THAT YOU WON'T MODIFY A MEMORY MANAGEMENT REGISTER ON A M.M. ABORT REFERENCE.

*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.

4329	030266	012737	002222	120000		MOV	#2222,2#120000	;LOAD DATA INTO 12K
4330	030274	013700	100000			MOV	2#100000,RO	;READ DATA THRU PAGE 4
4331	030300	005037	120000			CLR	2#120000	;CLEAR TEST LOCATION THRU PAGE 5
4332	030304	012737	020011	001364		MOV	#20011,MMEXP	;EXPECTING READ ONLY FAULT, PAGE 4
4333	030312	012737	017777	100000		MOV	#17777,2#100000	;TRY TO WRITE THRU PAGE 4
4334	030320	005037	001364			CLR	MMEXP	;NO MORE TRAPS EXPECTED
4335	030324	005737	120000			TST	2#120000	;SEE IF TEST LOCATION IS STILL ZERO
4336	030330	001412				BEQ	2\$;BRANCH IF WORD IS STILL ZERO
4337	030332	013737	172310	001176		MOV	KIPDR4,\$TMP0	;SAVE KIPDR4 FOR ERROR REPORT
4338	030340	013737	172350	001200		MOV	KIPAR4,\$TMP1	;SAVE KIPAR4 FOR ERROR REPORT
4339	030346	012737	100000	001202		MOV	#100000,\$TMP2	;SAVE VIRT ADR. FOR ERROR REPORT
4340	030354	104040				ERROR	40	;NO ABORT ON PAGE 4
4341	030356	105737	001446		2\$:	TSTB	FORTY	;EXECUTING ON AN 11/40
4342	030362	001041				BNE	5\$;BRANCH IF YES
4343	030364	076600				MED		;READ THE LOG JAM REG.
4344	030366	000100				RLJAM		
4345	030370	030027	000020			BIT	RO,#BIT4	;WAS BIT 4 SET IN LOG JAM?
4346	030374	001007				BNE	4\$;BRANCH IF YES
4347	030376	012737	000020	001210		MOV	#BIT4,\$TMP5	;SAVE EXPECTED CONTENTS FOR ERROR
4348	030404	012737	000100	001206		MOV	#RLJAM,\$TMP4	;SAVE LOG JAM ADDRESS FOR ERROR
4349	030412	104022				ERROR	22	;BIT4 NOT SET IN LOG JAM
4350								;TO INDICATE KT ABORT
4351	030414				4\$:			
4352	030414	012737	030426	000004		MOV	#3\$,ERRVEC	;SET TRAP TO 4 VECTOR TO 3\$
4353	030422	005737	000001			TST	2#000001	;CAUSE TRAP TO 4 WITH ODD ADDR. ERROR
4354	030426	022626			3\$:	CMP	(SP)+,(SP)+	;CLEAN UP STACK
4355	030430	013737	177766	000004		MOV	CPUEA,ERRVEC	;RESTORE NORMAL CPU TRAP ROUTINE
4356								;NOW CHECK FOR 250 AS LAST VEC. LOGGED
4357	030436	076600				MED		;READ LOG FLAG/INTRPT REG.
4358	030440	000104				RLFGIN		
4359	030442	030027	000250			BIT	RO,#250	;WAS VECTOR=250 RECORDED
4360	030446	001007				BNE	5\$;BRANCH IF YES
4361	030450	012737	000250	001210		MOV	#250,\$TMP5	;SAVE EXPECTED CONTENTS FOR ERROR
4362	030456	012737	000104	001206		MOV	#RLFGIN,\$TMP4	;SAVE LOG FLAG INT ADDRESS FOR ERROR
4363	030464	104022				ERROR	22	;VECTOR 250 WAS NOT LOGGIN IN
4364	030466				5\$:			;LOG FLAG/INTERRUPT REG.

```

*****
*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.
*****

```

4378								
4379	030466					TST37:		
4380	030466	000004				SCOPE		
4381	030470	012737	031574	001444		MOV	#TST40,NXTTST	;SAVE STARTING ADDRESS OF NEXT
4382								;TEST FOR ESCAPE ON PARITY ERRORS
4383	030476	104410				TBITO		;MAKE SURE T-BIT IS OFF FOR THIS TEST
4384	030500	042737	000001	177572	20\$:	BIC	#BITO,2#MMRO	;DISABLE MEMORY MANAGEMENT FOR THIS TEST

4385	030506	005037	001202			CLR	\$TMP2	INITIALIZE ERROR PRINT COUNTER
4386	030512	012702	000010			MOV	#10,R2	INITIALIZE STACK POINTER VALUE
4387	030516	012703	177400			MOV	#-400,R3	INITIALIZE STACK LIMIT REG. VALUE
4388	030522	062702	000400		1\$:	ADD	#400,R2	LOAD NEW STACK POINTER VALUE IN R2
4389	030526	062703	000400			ADD	#400,R3	LOAD NEW STACK LIMIT REG VALUE IN R3
4390	030532	020227	030500			CMP	R2,#20\$	IS SP VALUE LESS THAN LOC. OF THIS
4391								TEST CODE?
4392	030536	002404				BLT	6\$	BRANCH IF YES
4393	030540	020227	031542			CMP	R2,#10\$	IS SP VALUE GREATER THAN LOC. OF THIS
4394								TEST CODE?
4395	030544	003001				BGT	6\$	BRANCH IF YES
4396	030546	000765				BR	1\$	OTHERWISE GO BACK AND INCREASE VALUE OF
4397								SP SO CODE IS NOT AFFECTED
4398	030550	005037	000000		6\$:	CLR	0#0	CLEAR LOCATION 0 FOR RED ZONE CHECK
4399	030554	005037	001366			CLR	STKEXP	NO STACK VIOLATIONS EXPECTED(STKEXP=0)
4400	030560	012706	001100			MOV	#STACK,SP	INITIALIZE STACK POINTER
4401	030564	012737	031542	000004	2\$:	MOV	#10\$,ERRVEC	SET ERROR POINTER TO 10\$
4402	030572	016204	177776			MOV	-2(R2),R4	SAVE STACK LOCATIONS TO BE USED
4403	030576	016205	177774			MOV	-4(R2),R5	EXIT TEST IF EITHER LOC. TIMES OUT
4404	030602	012737	030610	001110		MOV	#11\$,SLPERR	SET LOOP ON ERROR POINTER TO 11\$
4405	030610	012737	030774	000004	11\$:	MOV	#50\$,ERRVEC	SET ERROR POINTER TO 50\$
4406	030616	010206				MOV	R2,SP	LOAD STACK POINTER
4407	030620	010337	177774			MOV	R3,STKLMT	LOAD STACK LIMIT REGISTER
4408	030624	016666	177770	177770		MOV	-10(SP),-10(SP)	REFERENCE LAST 'NO VIOLATION' ADDRESS
4409	030632	005037	000000			CLR	0#0	CLEAR LOCATION 0 FOR RED ZONE CHECK
4410	030636	012737	000001	001366		MOV	#1,STKEXP	EXPECT YELLOW ZONE VIOLATION(STKEXP=1)
4411	030644	012737	030652	001110		MOV	#12\$,SLPERR	SET LOOP ON ERROR POINTER TO 12\$
4412	030652	010206			12\$:	MOV	R2,SP	LOAD STACK POINTER
4413	030654	016666	177766	177766		MOV	-12(SP),-12(SP)	REFERENCE FIRST 'YELLOW ZONE' ADDRESS
4414	030662	000137	031454			JMP	60\$	YELLOW ZONE VIOL. EXPECTED-GOT NONE
4415	030666	012737	030674	001110	3\$:	MOV	#13\$,SLPERR	SET LOOP ON ERROR POINTER TO 13\$
4416	030674	010206			13\$:	MOV	R2,SP	LOAD STACK POINTER
4417	030676	005037	000000			CLR	0#0	CLEAR LOCATION 0 FOR RED ZONE CHECK
4418	030702	016666	177730	177730		MOV	-50(SP),-50(SP)	REFERENCE LAST 'YELLOW ZONE' ADDRESS
4419	030710	000137	031476			JMP	61\$	YELLOW ZONE VIOL. EXPECTED-GOT NONE
4420	030714	012737	030722	001110	4\$:	MOV	#14\$,SLPERR	SET LOOP ON ERROR POINTER TO 14\$
4421	030722	012737	000002	001366	14\$:	MOV	#2,STKEXP	EXPECT RED ZONE VIOLATION(STKEXP=2)
4422	030730	010206				MOV	R2,SP	LOAD STACK POINTER
4423	030732	005037	000000			CLR	0#0	CLEAR LOCATION 0 FOR RED ZONE CHECK
4424	030736	016237	177726	001200		MOV	-52(R2),\$TMP1	SAVE CONTENTS OF TEST LOC.
4425	030744	005166	177726			COM	-52(SP)	REFERENCE FIRST 'RED ZONE' ADDRESS
4426	030750	000137	031520			JMP	62\$	RED ZONE VIOL. EXPECTED-GOT NONE
4427	030754	013762	001200	177726	5\$:	MOV	\$TMP1,-52(R2)	RESTORE CONTENTS OF TEST LOC.
4428	030762	010462	177776			MOV	R4,-2(R2)	RESTORE STACK LOCATION USED
4429	030766	010562	177774			MOV	R5,-4(R2)	RESTORE STACK LOCATION USED
4430	030772	000653				BR	1\$	BRANCH UNTIL REACH NON EXSIST. MEM.

4431
4432
4433
4434
4435
4436
4437
4438
4439
4440

```

;
; 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.
;

```

4441											
4442	030774	010637	001176		50\$:	MOV	SP,\$TMP0				;SAVE TEST VALUE OF STACK POINTER
4443	031000	016200	177776			MOV	-2(R2),RO				;SAVE TEST VALUES IN TEST LOC.S
4444	031004	016201	177774			MOV	-4(R2),R1				
4445	031010	010462	177776			MOV	R4,-2(R2)				;RESTORE ORIGINAL VALUES TO
4446	031014	010562	177774			MOV	R5,-4(R2)				TEST LOCATIONS
4447	031020	005037	177774			CLR	STKLMT				;CLEAR STACK LIMIT REG. FOR ERROR REPORT
4448	031024	012706	001100			MOV	#STACK,SP				;RESET STACK POINTER FOR ERROR REPORTING
4449	031030	005737	001366			TST	STKEXP				;WAS A VIOLATION EXPECTED
4450	031034	001012				BNE	51\$;IF SO GO DECIDE WHAT KIND
4451	031036	104041				ERROR	41				;OTHERWISE ERROR - NO VIOLATION EXPECTED
4452	031040	013706	001176			MOV	\$TMP0,SP				;RESTORE STACK POINTER VALUE
4453	031044	010062	177776			MOV	RO,-2(R2)				;RESTORE TEST VALUES TO TEST LOC.S
4454	031050	010162	177774			MOV	R1,-4(R2)				
4455	031054	010337	177774			MOV	R3,STKLMT				;RESTORE STACK LIMIT REG. TEST VALUE
4456	031060	000006				RTT					;RETURN FROM ERROR
4457											
4458											
4459											
4460	031062	022737	000001	001366	51\$:	CMP	#1,STKEXP				;WAS A YELLOW ZONE VIOLATION EXPECTED
4461	031070	001076				BNE	54\$;IF NOT WE WERE EXPECTING A RED ZONE
4462	031072	005737	000000			TST	#0				;WAS LOCATION 0 CHANGED
4463	031076	001401				BEQ	52\$;BRANCH IF IT WASN'T
4464	031100	104043				ERROR	43				;EXPECTING YELLOW ZONE - GOT RED
4465	031102	022737	000005	001202	52\$:	CMP	#5,\$TMP2				;HAVE 5 "ERROR LOG" ERRORS BEEN TYPED?
4466	031110	003447				BLE	53\$;IF YES,DON'T TYPE ANY MORE
4467	031112	105737	001446			TSTB	FORTY				;EXECUTING ON AN 11/40?
4468	031116	001044				BNE	53\$;BRANCH AROUND LOG READS IF YES
4469											;SEE IF YELLOW ZONE LOGGED CORRECTLY
4470	031120	010037	001204			MOV	RO,\$TMP3				;SAVE THE CONTENTS OF RO
4471	031124	012737	000100	001206		MOV	#RLJAM,\$TMP4				;SAVE THE ADDRESS OF THE REG. IN CASE
4472											;OF AN ERROR
4473	031132	012737	010000	001210		MOV	#BIT12,\$TMP5				;ALSO SAVE THE EXPECTED CONTENTS
4474	031140	076600				MED					;READ THE CONTENTS OF THE ERROR LOG REG.
4475	031142	000100				RLJAM					;SPECIFIED BY "RLJAM"
4476	031144	032700	010000			BIT	#BIT12,RO				;WAS THE RIGHT BIT SET
4477	031150	001003				BNE	.+10				;BRANCH IF IT WAS
4478	031152	005237	001202			INC	\$TMP2				;INCREMENT ERROR PRINT COUNT
4479	031156	104022				ERROR	22				;ERROR LOG REG. DID NOT HAVE CORRECT BIT SET
4480	031160	013700	001204			MOV	\$TMP3,RO				;RESTORE THE CONTENTS OF RO
4481	031164	010037	001204			MOV	RO,\$TMP3				;SAVE THE CONTENTS OF RO
4482	031170	012737	000101	001206		MOV	#RLSERV,\$TMP4				;SAVE THE ADDRESS OF THE REG. IN CASE
4483											;OF AN ERROR
4484	031176	012737	000001	001210		MOV	#BIT0,\$TMP5				;ALSO SAVE THE EXPECTED CONTENTS
4485	031204	076600				MED					;READ THE CONTENTS OF THE ERROR LOG REG.
4486	031206	000101				RLSERV					;SPECIFIED BY "RLSERV"
4487	031210	032700	000001			BIT	#BIT0,RO				;WAS THE RIGHT BIT SET
4488	031214	001003				BNE	.+10				;BRANCH IF IT WAS
4489	031216	005237	001202			INC	\$TMP2				;INCREMENT ERROR PRINT COUNT
4490	031222	104022				ERROR	22				;ERROR LOG REG. DID NOT HAVE CORRECT BIT SET
4491	031224	013700	001204			MOV	\$TMP3,RO				;RESTORE THE CONTENTS OF RO
4492	031230	005737	000000		53\$:	TST	#0				;DID YOU GET RED ZONE?
4493	031234	001070				BNE	57\$;RETURN DIFFERENTLY IF YES
4494	031236	013706	001176			MOV	\$TMP0,SP				;RESTORE STACK POINTER VALUE
4495	031242	010062	177776			MOV	RO,-2(R2)				;RESTORE TEST VALUES TO TEST LOC.S
4496	031246	010162	177774			MOV	R1,-4(R2)				

```

4497 031252 010337 177774      MOV      R3,STKLMT      ;RESTORE STACK LIMIT REG. TEST VALUE
4498 031256 062762 000004 177774  ADD      #4,-4(R2)      ;FUDGE RETURN FROM TRAP
4499 031264 000006      RTT                    ;RETURN FROM TRAP
4500
4501
4502
4503 031266 023762 001200 177726 54$:  CMP      $TMF1,-52(R2)  ;WAS INSTRUCTION ABORTED ON 'RED' VIOL.
4504 031274 001401      BEQ      55$           ;BRANCH IF IT WAS EXECUTED
4505 031276 104045      ERROR   45           ;RED ZONE DID NOT CAUSE ABORT
4506 031300 005737 001176      TST      $TMP0        ;IS TEST VALUE OF SP EQUAL TO LOC. 0
4507 031304 001401      BEQ      56$           ;STACK SHOULD BE BUILT AT 0 ON A 'RED'
4508 031306 104046      ERROR   46           ;EXPECTED RED ZONE - GOT YELLOW
4509 031310 022737 000005 001202 56$:  CMP      #5,$TMP2      ;HAVE 5 "ERROR LOG" ERRORS BEEN TYPED?
4510 031316 003437      BLE     57$           ;IF YES, DON'T TYPE ANY MORE
4511 031320 105737 001446      TSTB    FORTY        ;EXECUTING ON AN 11/40?
4512 031324 001034      BNE     57$           ;BRANCH AROUND LOG READS IF YES
4513
4514 031326 010037 001204      MOV      R0,$TMP3     ;SEE IF RED ZONE LOGGED CORRECTLY
4515 031332 012737 000100 001206  MOV      #RLJAM,$TMP4  ;SAVE THE CONTENTS OF R0
4516
4517 031340 012737 004010 001210  MOV      #004010,$TMP5 ;SAVE THE ADDRESS OF THE REG. IN CASE
4518 031346 076600      MED                    ;OF AN ERROR
4519 031350 000100      RLJAM                    ;ALSO SAVE THE EXPECTED CONTENTS
4520 031352 032700 004010      BIT      #004010,R0   ;READ THE CONTENTS OF THE ERROR LOG REG.
4521 031356 001003      BNE     .+10          ;SPECIFIED BY "RLJAM"
4522 031360 005237 001202      INC     $TMP2        ;WAS THE RIGHT BIT SET
4523 031364 104022      ERROR   22          ;BRANCH IF IT WAS
4524 031366 013700 001204      MOV     $TMP3,R0     ;INCREMENT ERROR PRINT COUNT
4525 031372 013737 177766 001406  MOV     @#CPUERR,PCPUER ;ERROR LOG REG. DID NOT HAVE CORRECT BIT SET
4526 031400 032737 000004 001406  BIT     #BIT2,PCPUER ;RESTORE THE CONTENTS OF R0
4527 031406 001003      BNE     57$          ;READ THE CPU ERROR REGISTER
4528 031410 005237 001202      INC     $TMP2        ;LOOK AT CONTENTS OF CPU ERROR REG.
4529 031414 104076      ERROR   76          ;WAS RED ZONE VIOLATION REPORTED
4530 031416 005737 000000 57$:  TST     @#0          ;INCREMENT ERROR TYPE COUNT
4531 031422 001702      BEQ     53$          ;CPU ERROR REG. DID NOT REPORT RED ZONE
4532 031424 013706 001176      MOV     $TMP0,SP     ;DID WE GET RED ZONE?
4533 031430 010062 177776      MOV     R0,-2(R2)   ;RETURN DIFFERENTLY IF NO
4534 031434 010162 177774      MOV     R1,-4(R2)   ;RESTORE STACK POINTER VALUE
4535 031440 010337 177774      MOV     R3,STKLMT   ;RESTORE TEST VALUES TO TEST LOC.S
4536 031444 062737 000004 000000  ADD     #4,@#0      ;RESTORE STACK LIMIT REG. TEST VALUE
4537 031452 000006      RTT                    ;FUDGE RETURN FROM TRAP
4538
4539
4540
4541
4542 031454 005037 177774 60$:  CLR     STKLMT      ;THIS GROUP OF CODING IS ENTERED WHEN A STACK VIOLATION
4543 031460 012706 001100      MOV     #STACK,SP   ;WAS EXPECTED BUT NONE OCCURRED
4544 031464 104042      ERROR   42          ;CLEAR STACK LIMIT REG. FOR ERROR REPORT
4545 031466 010337 177774      MOV     R3,STKLMT   ;RESET STACK POINTER FOR ERROR REPORT
4546 031472 000137 030666      JMP     3$          ;YELLOW ZONE EXPECTED - GOT NONE
4547 031476 005037 177774 61$:  CLR     STKLMT      ;RESTORE STACK LIMIT REG. TEST VALUE
4548 031502 012706 001100      MOV     #STACK,SP   ;RETRUN TO TEST AT 3$
4549 031506 104042      ERROR   42          ;CLEAR STACK LIMIT REG. FOR ERROR REPORT
4550 031510 010337 177774      MOV     R3,STKLMT   ;RESET STACK POINTER FOR ERROR REPORT
4551 031514 000137 030714      JMP     4$          ;YELLOW ZONE EXPECTED - GOT NONE
4552 031520 005037 177774 62$:  CLR     STKLMT      ;RESTORE STACK LIMIT REG. TEST VALUE

```


4553 031524 012706 001100 MOV #STACK,SP ;RESET STACK POINTER FOR ERROR REPORT
4554 031530 104044 ERROR 44 ;RED ZONE EXPECTED - GOT NONE
4555 031532 010337 177774 MOV R3,STKLMT ;RESTORE STACK LIMIT REG. TEST VALUE
4556 031536 000137 030754 JMP 5\$;RETURN TO TEST AT 5\$

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

4562 031542 012737 015160 000004 10\$: MOV #CPUER,ERRVEC ;RESTORE NORMAL CPU TRAP ROUTINE
4563 ;GOT TIMEOUT, TEST IS OVER
4564 031550 012706 001100 MOV #STACK,SP ;RESET THE STACK POINTER
4565 031554 052737 000001 177572 BIS #BIT0,@#MMRO ;REENABLE MEMORY MANAGEMENT
4566 031562 005037 177774 CLR @#STKLMT ;LEAVE STACK LIMIT REG. CLEAR
4567 031566 012737 030500 001110 MOV #20\$, \$LPERR ;SET LOOP POINTER TO START OF TEST
4568 031574

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 MMR2 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.

4586 031574
4587 031574 000004
4588 031576 012737 032014 001444 SCOPE
4589 MOV #TST41,NXTTST ;SAVE STARTING ADDRESS OF NEXT
4590 031604 104411 TBITR ;TEST FOR ESCAPE ON PARITY ERRORS
4591 ;RESTORE T-BIT IF IT WAS ON BEFORE
4592 031606 012737 031614 001106 MOV #20\$, \$LPADR ;THE LAST TEST
4593 ;SET LOOP ADDRESS POINTER TO 20\$
4594 031614 012737 000000 172310 20\$: MOV #000000,KIPDR4 ;FOR ITERATION PASS
4595 ;MAP PAGE 4 NON-RESIDENT, AND PAGE
4596 031622 012737 031662 000250 MOV #5\$, MMVEC ;LENGTH OF 1 BLOCK
4597 031630 012737 000340 000252 MOV #340, MMVEC+2 ;SET M.M. TRAP VECTOR TO 5\$
4598 031636 012737 031644 001110 MOV #1\$, \$LPERR ;SET PRIORITY TO 7 GOTO KERNEL MODE
4599 031644 013700 100000 1\$: MOV @#100000,R0 ;SET LOOP ON ERROR POINTER TO 1\$
4600 ;TRY TO READ THRU PAGE 4
4601 ;THIS PAGE NON-RESIDENT SHOULD CAUSE
4602 031650 012737 031704 000250 2\$: MOV #10\$, MMVEC ;ABORT AND TRAP TO 5\$.
4603 031656 013700 100100 MOV @#100100,R0 ;SET M.M. TRAP VECTOR TO 10\$
4604 ;TRY TO READ FROM BLOCK 2 OF PAGE 4
4605 ;THIS SHOULD ABORT AGAIN BUT NONE
4606 ;OF THE MEMORY MANAGEMENT STATUS
4607 ;REGISTERS SHOULD BE CLOCKED THIS TIME.
4608

4609	031662	013737	177572	001402	5\$:	MOV	MMRO, PMMRO	; READ MEMORY MANAGEMENT REGISTER 0
4610	031670	013737	177576	001404		MOV	MMR2, PMMR2	; READ MEMORY MANAGEMENT REGISTER 2
4611	031676	012716	031650			MOV	#2\$, (KSP)	; CHANGE RETURN ADDRESS TO 2\$
4612	031702	000006				RTT		; GO BACK TO 2\$ AND CAUSE PAGE FAULT
4613								
4614								
4615	031704	012716	031712		10\$:	MOV	#16\$, (KSP)	; CHANGE RETURN ADDRESS TO 16\$
4616	031710	000006				RTT		; RETURN TO 16\$ AND CONTINUE PROGRAM
4617	031712	005037	001204		16\$:	CLR	\$TMP3	; ERROR COUNTER, 3 POSSIBLE CMP FAILURES
4618	031716	013737	177572	001176		MOV	MMRO, \$TMP0	; READ MEMORY MANAGEMENT REGISTER 0
4619	031724	013737	177576	001202		MOV	MMR2, \$TMP2	; READ MEMORY MANAGEMENT REGISTER 2
4620	031732	023737	001202	001404		CMP	\$TMP2, PMMR2	; SEE IF MMR2 CHANGED
4621	031740	001402				BEQ	12\$; BRANCH IF MMR2 DIDN'T CHANGE
4622	031742	005237	001204			INC	\$TMP3	; ONE COMPARE FAILURE
4623	031746	023737	001176	001402	12\$:	CMP	\$TMP0, PMMRO	; SEE IF MMRO CHANGED
4624	031754	001402				BEQ	13\$; BRANCH IF MMRO DIDN'T CHANGE
4625	031756	005237	001204			INC	\$TMP3	; ANOTHER COMPARE FAILURE
4626	031762	005737	001204		13\$:	TST	\$TMP3	; WERE THERE ANY ERRORS ON THIS TEST
4627	031766	001401				BEQ	19\$; BRANCH IF NO ERRORS
4628	031770	104050				ERROR	50	; AT LEAST ONE M.M. REG CHANGED
4629	031772	042737	160556	177572	19\$:	BIC	#160556, MMRO	; CLEAR ALL ERROR BITS IN MMRO
4630	032000	012737	015430	000250		MOV	#MMTRAP, MMVEC	; PUT BACK REGULAR M.M. TRAP ROUTINE
4631	032006	012737	031614	001110		MOV	#20\$, \$LPERR	; SET LOOP POINTER TO START OF TEST

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
4679
4680
4681
4682
4683
4684
4685
4686
4687

```

*****
*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 0&2 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 ;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
EQUIV BIT4,TBIT
TBITO
MOV #340,ERRVEC+2 ;SET PRIORITY OF ERRVEC TO 7
;AND MAKE NEW MODE KERNEL
MOV #000,UIPARO ;MAP USER PAGE 0 TO PHYS. 0
MOV #200,UIPAR1 ;MAP USER PAGE 1 TO 4K -8K
MOV #400,UIPAR2 ;MAP USER PAGE 2 TO 8K - 12K
MOV #600,UIPAR3 ;MAP USER PAGE 3 TO 12K - 16K
MOV #7600,UIPAR7 ;MAP USER PAGE 7 TO I/O PAGE
MOV #77406,UIPDR0 ;MAKE USER PAGE 0 200 BLCKS, R/W
MOV #77406,UIPDR1 ;MAKE USER PAGE 1 200 BLCKS, R/W
MOV #77406,UIPDR2 ;MAKE USER PAGE 2 200 BLCKS, R/W
MOV #77406,UIPDR3 ;MAKE USER PAGE 3 200 BLCKS, R/W
MOV #77406,UIPDR7 ;MAKE USER PAGE 7 200 BLOCKS, R/W
MOV #1$,SLPERR ;SET LOOP ON ERROR POINTER TO 1$
MOV #10$,ERRVEC ;SET ERRVEC TO 10$
MOV #140017,MMVEC+2 ;LOAD NEW PSW INTO MMVEC+2
MOV #600,UIPAR4 ;MAP USER PAGE 4 TO 12K
MOV #77402,UIPDR4 ;MAKE USER PAGE 4 200 BLOCKS, READ-ONLY
MOV #140000,PSW ;GO TO USER MODE
MOV #703,USP ;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

```

```

032014
032014 000004
032016 012737 032652 001444
032024 104410
032026 012737 000340 000006 20$:
032034 012737 000000 177640
032042 012737 000200 177642
032050 012737 000400 177644
032056 012737 000600 177646
032064 012737 007600 177656
032072 012737 077406 177600
032100 012737 077406 177602
032106 012737 077406 177604
032114 012737 077406 177606
032122 012737 077406 177616
032130 012737 032200 001110
032136 012737 032204 000004
032144 012737 140017 000252
032152 012737 000600 177650
032160 012737 077402 177610
032166 012737 140000 177776
032174 012706 000703
032200 005237 100000 1$: INC @#100000

```

Address	OpCode	Operand 1	Operand 2	Operand 3	Operand 4	Label	Instruction	Comments
4688								: AT 10\$ POINTED TO BY "ERRVEC"
4689	032204	016602	000002			10\$:	MOV 2(KSP),R2	:SAVE PSW IN R2
4690	032210	011603					MOV (KSP),R3	:SAVE PC IN R3
4691	032212	012737	077406	177610			MOV #77406,UIPDR4	:MAKE USER PAGE 4 READ & WRITE
4692	032220	010601					MOV KSP,R1	:SAVE THE KERNEL STACK POINTER
4693	032222	012711	032236				MOV #16\$(R1)	:SET RETURN PC TO 16\$
4694	032226	012761	000340	000002			MOV #340,2(R1)	:SET RETURN PSW TO KERNEL MODE
4695	032234	000006					RTT	:RETURN FROM "ERR-ON-ERR" SERVICE
4696	032236	042761	000020	000002		16\$:	BIC #TBIT,2(R1)	:CLEAR T BIT IN SAVED PSW
4697	032244	012706	001100				MOV #STACK,KSP	:RESTORE THE STACK POINTER
4698	032250	005037	001200				STMP1	:THIS IS THE ERROR COUNTER FLAG
4699	032254	012737	140017	001202			MOV #140017,STMP2	:LOAD EXPECTED PS SAVED FOR 11/6X
4700								: (PS FROM MMVEC+2 FETCHED BY KT ABORT)
4701	032262	105737	001446				TSTB FORTY	:EXECUTING ON AN 11/40?
4702	032266	001403					BEQ 19\$:BRANCH IF NO
4703	032270	012737	140000	001202			MOV #140000,STMP2	:LOAD EXPECTED PS SAVED FOR 11/40
4704								: (PS AT TIME OF FIRST (KT) ERROR)
4705	032276	023702	001202			19\$:	CMP STMP2,R2	:WAS CORRECT PS SAVED?
4706	032302	001402					BEQ 18\$:BRANCH IF PS IS CORRECT
4707	032304	005237	001200				INC STMP1	:RIGHT PS IS ON STACK
4708	032310	022703	032204			18\$:	CMP #10\$,R3	:THE PC THAT SHOULD HAVE BEEN
4709								: ON THE STACK IS THE UPDATED
4710								: PC FROM THE KT ABORT INST.
4711								: (10\$ = 1\$+4)
4712	032314	001402					BEQ 11\$:BRANCH IF PC CORRECT
4713	032316	005237	001200				INC STMP1	:WRONG PC ON THE STACK
4714	032322	013737	177572	001402		11\$:	MOV MMRO,PMR0	:SAVE MMRO FOR COMPARE
4715	032330	013737	177576	001404			MOV MMR2,PMR2	:SAVE MMR2, IT SHOULD EQUAL #10
4716	032336	105737	001446				TSTB FORTY	:EXECUTING ON AN 11/40?
4717	032342	001040					BNE 21\$:BRANCH IF YES
4718	032344	010037	001204				MOV RO,STMP3	:SAVE CONTENTS OF RO
4719	032350	076600					MED	:READ LOG WHAMI REG.
4720	032352	000105					RLWHAM	
4721	032354	030027	000004				BIT RO,#BIT2	:WAS ERROR-ON-ERROR BIT SET?
4722	032360	001007					BNE 17\$:BRANCH IF YES
4723	032362	012737	000004	001210			MOV #BIT2,STMP5	:SAVE EXPECTED CONTENTS FOR ERROR
4724	032370	012737	000105	001206			MOV #RLWHAM,STMP4	:SAVE ADDRESS FOR ERROR
4725	032376	104022					ERROR 22	:ERR-ON-ERR BIT NOT SET IN
4726	032400					17\$:		: LOG WHAMI REG.
4727	032400	012737	000022	001206			MOV #RWAM,STMP4	:SAVE ADDR. FOR ERROR
4728	032406	012737	000004	001210			MOV #BIT2,STMP5	:SAVE EXP. DATA FOR ERROR
4729	032414	076600					MED	:MED READ THE WHAMI REG.
4730	032416	000022					RWHAM	
4731	032420	032700	000004				BIT #BIT2,RO	:WAS "ERR-ON-ERR" BIT SET?
4732	032424	001001					BNE .+4	:BRANCH IF YES
4733	032426	104022					ERROR 22	: "ERR-ON-ERR" BIT NOT SET
4734	032430	042700	000004				BIC #BIT2,RO	:CLEAR BIT2 IN RO
4735	032434	076600					MED	:WRITE ORIGINAL CONTENTS
4736	032436	000222					WVHAM	:TO WHAMI WITH BIT2=0
4737	032440	013700	001204				MOV STMP3,RO	:RESTORE CONTENTS OF RO
4738	032444	022737	020151	001402		21\$:	CMP #020151,PMR0	:SHOULD HAVE READ-ONLY VIO.
4739								: COMPLETED, PAGE 4, USER I-SPACE.
4740	032452	001402					BEQ 12\$:BRANCH IF CORRECT CONDITION
4741	032454	005237	001200				INC STMP1	:WRONG ABORT CONDITION
4742	032460	105737	001446			12\$:	TSTB FORTY	:EXECUTING ON AN 11/40?
4743	032464	001424					BEQ 23\$:BRANCH IF NO AROUND 11/40 CHECKS

4744	032466	052737	140000	177776		BIS	#140000,PSW	:GO TO USER MODE
4745	032474	010604				MOV	USP,R4	:READ THE USER STACK POINTER
4746	032476	042737	140000	177776		BIC	#140000,PSW	:GO BACK TO KERNEL MODE
4747	032504	022704	000004			CMP	#4,R4	:WAS USER STK. PTR. FORCED TO 4 ON 11/40?
4748	032510	001404				BEQ	22\$:BRANCH IF YES
4749	032512	005237	001200			INC	\$TMP1	:USP ON 11/40 IS NOT CORRECT
4750	032516	010401				MOV	R4,R1	:SAVE VALUE OF USP FOR TYPEOUT
4751	032520	000413				BR	13\$:DON'T BOTHER CHECKING KSP IF USP WAS WRONG
4752	032522	022701	001074		22\$:	CMP	#KERSTK-4,R1	:WAS PC & PSW SAVED ON 11/40 KERNEL STACK?
4753	032526	001410				BEQ	13\$:BRANCH IF YES
4754	032530	005237	001200			INC	\$TMP1	:PC & PSW NOT SAVED ON 11/40 KERNEL STACK
4755	032534	000405				BR	13\$:BRANCH AROUND 11/6X KSP CHECK
4756	032536				23\$:			
4757	032536	022701	000000			CMP	#0,R1	:THE KER. STK. POINTER
4758								:SHOULD HAVE BEEN = 0
4759								:AFTER ERR-ON-ERR TRAP TO LOC. 4
4760	032542	001402				BEQ	13\$:BRANCH IF KSP IS CORRECT
4761	032544	005237	001200			INC	\$TMP1	:KSP IS NOT CORRECT
4762	032550	022737	032200	001404	13\$:	CMP	#1\$,PMMR2	:SEE IF MMR2 EQUALS ADDRS. OF KT ABORT
4763	032556	001402				BEQ	14\$:BRANCH IF ADDRESS IS CORRECT
4764	032560	005237	001200			INC	\$TMP1	:MMR2 HAS THE WRONG ADDRESS
4765	032564	005737	001200		14\$:	TST	\$TMP1	:DID ANY OF THE COMPARES FAIL?
4766	032570	001401				BEQ	15\$:BRANCH IF ALL COMPARES SUCCEEDED
4767	032572	104047				ERROR	47	:AT LEAST ONE COMPARE FAILED
4768								:IF USER STK. PTR. WRONG ON 11/40,
4769								:THE KERNEL STK. PTR. WAS NOT CHECKED
4770	032574	052737	000340	177776	15\$:	BIS	#340,PSW	:NORMAL PRIORITY IS 7
4771	032602	052737	140000	177776		BIS	#140000,PSW	:GO TO USER MODE TO RESET PTR
4772	032610	012706	000700			MOV	#700,USP	:RESTORE USER STK PTR TO 700
4773	032614	042737	140000	177776		BIC	#140000,PSW	:RETURN TO KERNEL MODE
4774	032622	012737	000340	000006		MOV	#340,ERRVEC+2	:RESTORE CORRECT PS TO ERROR VECTOR
4775	032630	012737	032026	001110		MOV	#20\$,SLPERR	:SET LOOP POINTER TO START OF TEST
4776	032636	042737	160556	177572		BIC	#160556,MMR0	:CLEAR ERROR CONDITION IN MMR0
4777	032644	012737	015160	000004		MOV	#CPUER,ERRVEC	:RESTORE NORMAL CPU TRAP ROUTINE


```

4834 033062 000000 HALT ;EVER BE REACHED
4835 033064 000000 HALT ;EVER BE REACHED
4836 033066 000000 HALT ;EVER BE REACHED
4837 033070 012737 015430 000250 2$: MOV #MMTRAP,MMVEC ;RESTORE NORMAL M.M. TRAP ROUTINE
4838 033076 022737 100153 001402 CMP #100153,PMR0 ;EXPECTING NON-RESIDENT PAGE 5
4839 ;ABORT IN USER MODE I-SPACE
4840 033104 001401 BEQ 11$ ;BRANCH IF CORRECT COND
4841 033106 104052 ERROR 52 ;ABORT CONDITION INCORRECT
4842 033110 012737 032662 001110 11$: MOV #20$,SLPERR ;SET LOOP POINTER TO START OF TEST
4843 033116 000424 BR TST43 ;BRANCH TO NEXT TEST
4844 033120 012737 015542 000150 3$: MOV #KERVEC,<MMVEC-100> ;SET UP KERNEL SPACE VECTOR
4845 033126 012737 000340 000152 MOV #340,<MMVEC+2-100> ;KERNEL SPACE PSW = 340
4846 033134 013700 120000 MOV #120000,R0 ;READ FROM PAGE 5, USER SPACE
4847 033140 000405 BR 10$ ;GO SEE IF GOT EXPECTED ABORT
4848 ;AFTER WE'VE JUST RETURNED
4849 ;FROM "KERVEC"
4850 ;THESE HALTS SHOULDN'T
4851 033142 000000 HALT ;EVER BE REACHED
4852 033144 000000 HALT
4853 033146 000000 HALT
4854 033150 000000 HALT
4855 033152 000000 HALT
4856 033154 042737 160556 177572 10$: BIC #160556,MMR0 ;CLEAR MMRO FOR NEXT TEST
4857 033162 012737 000340 177776 MOV #340,PSW ;GO BACK INTO KERNEL MODE NOW
4858 ;THE NEXT INSTRUCTION TO BE EXECUTED
4859 ;IS AT LABEL 2$
4860
4861
4862
4863
4864
4865
4866
4867
4868
4869
4870
4871
4872
4873
4874
4875
4876
4877
4878
4879
4880
4881
4882
4883
4884
4885
4886
4887
4888
4889

```

```

*****
*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.
*****

```

```

4874 033170 TST43:
4875 033170 000004 SCOPE
4876 033172 012737 033674 001444 MOV #TST44,NXTTST ;SAVE STARTING ADDRESS OF NEXT
4877 ;TEST FOR ESCAPE ON PARITY ERRORS
4878 033200 012737 000002 001212 20$: MOV #2,$TIMES ;DO 2 ITERATIONS
4879 033206
4880 033206 012737 077406 172300 MOV #77406,KIPDR0 ;MAKE KERNEL I PAGE 0 200 BLOCKS, R/W
4881 033214 012737 077406 172302 MOV #77406,KIPDR1 ;MAKE KERNEL I PAGE 1 200 BLOCKS, R/W
4882 033222 012737 077406 172304 MOV #77406,KIPDR2 ;MAKE KERNEL I PAGE 2 200 BLOCKS, R/W
4883 033230 012737 077406 172306 MOV #77406,KIPDR3 ;MAKE KERNEL I PAGE 3 200 BLOCKS, R/W
4884 033236 012737 077406 172316 MOV #77406,KIPDR7 ;MAKE KERNEL I PAGE 7 200 BLOCKS, R/W
4885 033244 012737 000000 172340 MOV #000,KIPAR0 ;MAP KERNEL I PAGE 0 TO 0 - 4K
4886 033252 012737 000200 172342 MOV #200,KIPAR1 ;MAP KERNEL I PAGE 1 TO 4K - 8K
4887 033260 012737 000400 172344 MOV #400,KIPAR2 ;MAP KERNEL I PAGE 2 TO 8K - 12K
4888 033266 012737 000600 172346 MOV #600,KIPAR3 ;MAP KERNEL I PAGE 3 TO 12K - 16K
4889 033274 012737 007600 172356 MOV #7600,KIPAR7 ;MAP KERNEL I PAGE 7 TO THE I/O PAGE

```

4890	033302	012737	000001	177572		MOV	#BIT0,MMR0	;ENABLE 18-BIT RELOCATION IF NOT ON
4891	033310	012700	077406			MOV	#77406,R0	
4892	033314	010037	172310			MOV	R0,KIPDR4	;MAP KERNEL PAGE 4 READ-WRITE
4893	033320	010037	172312			MOV	R0,KIPDR5	;MAP KERNEL PAGE 5 READ-WRITE
4894	033324	010037	172314			MOV	R0,KIPDR6	;MAP KERNEL PAGE 6 READ-WRITE
4895	033330	012700	000600			MOV	#600,R0	
4896	033334	010037	172350			MOV	R0,KIPAR4	;MAP KERNEL PAGE 4 TO 12K
4897	033340	010037	172352			MOV	R0,KIPAR5	;MAP KERNEL PAGE 5 TO 12K
4898	033344	010037	172354			MOV	R0,KIPAR6	;MAP KERNEL PAGE 6 TO 12K
4899	033350	012737	077402	177606		MOV	#77402,UIPDR3	;MAP USER PAGE 3 READ-ONLY
4900	033356	012737	077402	177610		MOV	#77402,UIPDR4	;MAP USER PAGE 4 READ-ONLY
4901	033364	012737	077402	177612		MOV	#77402,UIPDR5	;MAP USER PAGE 5 READ-ONLY
4902	033372	012737	077402	177614		MOV	#77402,UIPDR6	;MAP USER PAGE 6 READ-ONLY
4903	033400	012737	000000	177640		MOV	#000,UIPAR0	;MAP USER PAGE 0 TO 0-4K
4904	033406	012737	000200	177642		MOV	#200,UIPAR1	;MAP USER PAGE 1 TO 4K
4905	033414	012737	000400	177644		MOV	#400,UIPAR2	;MAP USER PAGE 2 TO 8K
4906	033422	012737	000600	177646		MOV	#600,UIPAR3	;MAP USER PAGE 3 TO 12K
4907	033430	012737	000600	177650		MOV	#600,UIPAR4	;MAP USER PAGE 4 TO 12K
4908	033436	012737	000600	177652		MOV	#600,UIPAR5	;MAP USER PAGE 5 TO 12K
4909	033444	012737	000600	177654		MOV	#600,UIPAR6	;MAP USER PAGE 6 TO 12K
4910	033452	012737	033510	001110		MOV	#3\$, \$LPERR	;SET LOOP ON ERROR POINTER TO 3\$
4911	033460	012700	140000			MOV	#140000,R0	;THIS WILL FORCE USER MODE WHEN USED
4912								;AS A PROCESSOR STATUS
4913	033464	012704	060000			MOV	#60000,R4	;SETUP BITS <15:13> IN R4 SO V.A. WILL
4914								;INITIALLY SELECT PAR/PDR 3
4915	033470	012737	033534	000250		MOV	#10\$,MMVEC	;SET M.M. TRAP VECTOR TO 10\$
4916	033476	012737	000340	000252		MOV	#340,MMVEC+2	;MAKE SURE TRAP TAKES YOU TO KERNEL
4917	033504	012701	000002		2\$:	MOV	#2,R1	;PUT TEST BIT AT STARTING POSITION(BIT 1)
4918	033510	042701	160000		3\$:	BIC	#160000,R1	;CLEAR BITS <15:13> SO APF SELECT BITS CAN
4919								;BE RESET
4920	033514	050401				BIS	R4,R1	;SET BITS <15:13> TO SELECT PAR/PDR BEING USED
4921	033516	011137	001176			MOV	(R1)\$TMPD	;SAVE CONTENTS OF TEST LOCATION
4922	033522	012711	005211			MOV	#5211,(R1)	;PUT "ABORT INSTRUCTION" AT TEST LOC.
4923	033526	010046				MOV	R0,-(KSP)	;PUSH USER PS ON STACK
4924	033530	010146				MOV	R1,-(KSP)	;PUSH USER'S VIRTUAL PC ON STACK
4925	033532	000002				RTI		;RETURN TO USER MODE
4926								;THE FIRST INSTRUCTION FETCH WILL
4927								;CAUSE A NON-RESIDENT ABORT AND LOCK
4928								;MMR2 SO THAT ALL BITS CAN BE CHECKED.
4929								
4930								
4931								
4932	033534	062706	000004		10\$:	ADD	#4,KSP	;CLEAN UP STACK FOR NEXT TIME
4933	033540	013711	001176			MOV	\$TMPD,(R1)	;RESTORE CONTENTS OF TEST LOCATION
4934	033544	013737	177576	001404		MOV	MMR2,PMR2	;READ MMR2 TO TEMP LOCATION
4935	033552	013737	177572	001402		MOV	MMR0,PMR0	;READ MMR0 TO TEMP LOCATION
4936	033560	020137	001404			CMP	R1,PMR2	;SEE IF MMR2 LOCKED CORRECT V. A.
4937	033564	001401				BEQ	11\$;BRANCH IF MMR2 HAS RIGHT V.A.
4938	033566	104053				ERROR	53	;WRONG V.A.
4939	033570	005002			11\$:	CLR	R2	;R2 WILL GET THE VIRTUAL PAGE NO.
4940	033572	010103				MOV	R1,R3	;COPY VIRTUAL ADDRESS INTO R3
4941	033574	012705	000003			MOV	#3,R5	;COMBINED LEFT SHIFT <R2,R3> 3 BITS
4942	033600	000241			4\$:	CLC		
4943	033602	006303				ASL	R3	
4944	033604	006102				ROL	R2	
4945	033606	077504				SOB	R5,4\$	


```

4946 033610 006102          ROL      R2          ;ADJUST PAGE NUMBER
4947 033612 052702 020141  BIS      #020141,R2    ;SET OTHER EXPECTED BITS IN MMRO
4948 033616 020237 001402  CMP      R2,PMR0     ;SEE IF MMRO RECORDED CORRECT PAGE NO.
4949 033622 001401          BEQ      12$         ;BRANCH IF PAGE NUMBER WAS CORRECT
4950 033624 104054          ERROR    54         ;WRONG PAGE NO. IN MMRO
4951 033626 042737 160556 177572 12$:  BIC      #160556,MMRO ;CLEAR ALL ERROR BITS IN MMRO
4952 033634 032701 010000  BIT      #BIT12,R1   ;HAS BIT BEEN SHIFTED ALL THE WAY THRU
4953 033640 001002          BNE      13$         ;BRANCH IF IT WAS
4954 033642 006301          ASL      R1          ;SHIFT BIT ONE POSITION LEFT
4955 033644 000721          BR       3$         ;GO AND TRY THIS VIRTUAL ADDRESS
4956 033646 062704 020000 13$:  ADD      #20000,R4    ;INCREASE R4 TO SELECT NEXT PAR/PDR
4957 033652 022704 160000  CMP      #160000,R4  ;HAVE ALL PAGES(3,4,5,6) BEEN USED
4958 033656 001312          BNE      2$         ;CONTINUE IF THEY HAVEN'T
4959 033660 012737 015430 000250 MOV      #MMTRAP,MMVEC ;PUT BACK REGULAR M.M. TRAP ROUTINE
4960 033666 012737 033206 001110 MOV      #20$,SLPERR ;SET LOOP POINTER TO START OF TEST

```

4961
4962
4963
4964
4965
4966
4967
4968
4969
4970
4971
4972
4973
4974
4975
4976
4977
4978
4979
4980
4981
4982
4983
4984
4985

```

.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.
*****

```

```

4986 033674          TST44:  SCOPE
4987 033674 000004          MOV      #TST45,NXTTST ;SAVE STARTING ADDRESS OF NEXT
4988 033676 012737 034354 001444 TBITR    ;TEST FOR ESCAPE ON PARITY ERRORS
4989          ;RESTORE T-BIT IF IT WAS ON BEFORE THE
4990 033704 104411          ;LAST 3 TESTS
4991          ENTPT6: MOV      #20$,SLPADR ;SET LOOP ADDRESS POINTER TO 20$
4992 033706 012737 034040 001106 MOV      #20$,SLPERR ;SET LOOP ON ERROR POINTER TO 20$
4993 033714 012737 034040 001110 MOV      #44,$TSTNM ;LOAD TEST NUMBER INTO MEMORY
4994 033722 012737 000044 001102 MOV      $TSTNM,@DISPLAY ;DISPLAY TEST NUMBER FOR THIS TEST
4995 033730 013777 001102 145204 MOV      #77406,KIPDR0 ;MAKE KERNEL I PAGE 0 200 BLOCKS, R/W
4996 033736 012737 077406 172300 MOV      #77406,KIPDR1 ;MAKE KERNEL I PAGE 1 200 BLOCKS, R/W
4997 033744 012737 077406 172302 MOV      #77406,KIPDR2 ;MAKE KERNEL I PAGE 2 200 BLOCKS, R/W
4998 033752 012737 077406 172304 MOV      #77406,KIPDR3 ;MAKE KERNEL I PAGE 3 200 BLOCKS, R/W
4999 033760 012737 077406 172306 MOV      #77406,KIPDR7 ;MAKE KERNEL I PAGE 7 200 BLOCKS, R/W
5000 033766 012737 077406 172316 MOV      #000,KIPAR0 ;MAP KERNEL I PAGE 0 TO 0 - 4K
5001 033774 012737 000000 172340

```



```

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 034354
5083 034354 000004
5084 034356 012737 035252 001444
5085
5086
5087
5088
5089
5090 034364 012737 000000 172340
5091 034372 012737 000200 172342
5092 034400 012737 000400 172344
5093 034406 012737 000600 172346
5094 034414 012737 000600 172350
5095 034422 012737 000600 172352
5096 034430 012737 000600 172354
5097 034436 012737 007600 172356
5098 034444 012737 000000 177640
5099 034452 012737 000200 177642
5100 034460 012737 000400 177644
5101 034466 012737 000600 177646
5102 034474 012737 000600 177650
5103 034502 012737 000600 177652
5104 034510 012737 000600 177654
5105 034516 012737 007600 177656
5106 034524 012737 034534 001110
5107 034532 104410
5108
5109 034534 012737 034624 001110 20$:
5110 034542 012737 000000 177776
5111 034550 012703 172300
5112
5113 034554 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 77706 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 EXSISTING ADDRESS CAN'T BE GUARANTEED FOR THE I/O PAGE
*

```

```

*****
TST45:

```

```

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$, $LPERR
TBITO
;SET LOOP ON ERROR POINTER TO 20$
;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

```

5114	034560	012705	010200		MOV	#10200,R5	;;LOAD STARTING VIRTUAL ADDRESS INTO R5
5115	034564	012700	077406	19\$:	MOV	#77406,R0	;;ALL PAGES WILL BE ACF=6
5116	034570	005037	001176		CLR	\$TMP0	;;CLEAR CORRECT PDR SET INDICATOR
5117	034574	012702	000010		MOV	#10,R2	;;SET COUNT TO LOAD 8 ADDRESSES
5118	034600	012701	172300		MOV	#KIPDR0,R1	;;PUT ADDRESS OF FIRST PDR IN R1
5119	034604	010021		1\$:	MOV	R0,(R1)+	;;LOAD R0 INTO PDR ADDRESSED BY R1
5120	034606	077202			SOB	R2,1\$;;BRANCH BACK TO 1\$ IF R2 IS NOT ZERO
5121	034610	012702	000010		MOV	#10,R2	;;SET COUNT TO LOAD 8 ADDRESSES
5122	034614	012701	177600		MOV	#UIPDR0,R1	;;PUT ADDRESS OF FIRST PDR IN R1
5123	034620	010021		2\$:	MOV	R0,(R1)+	;;LOAD R0 INTO PDR ADDRESSED BY R1
5124	034622	077202			SOB	R2,2\$;;BRANCH BACK TO 2\$ IF R2 IS NOT ZERO
5125	034624	012700	077406	21\$:	MOV	#77406,R0	;;MUST RE-INIT THESE PDRS IF ERROR
5126	034630	010037	172300		MOV	R0,KIPDR0	;;LOAD KERNEL PDR0
5127	034634	010037	172302		MOV	R0,KIPDR1	;;LOAD KERNEL PDR1
5128	034640	010037	172304		MOV	R0,KIPDR2	;;LOAD KERNEL PDR2
5129	034644	010037	172316		MOV	R0,KIPDR7	;;LOAD KERNEL PDR7
5130	034650	010037	172300		MOV	R0,KIPDR0	;;LOAD PDR0 OF PRESENT SPACE
5131	034654	010037	172316		MOV	R0,KIPDR7	;;LOAD PDR7 OF PRESENT SPACE
5132	034660	020527	170200		CMP	R5,#170200	;;IS PDR7 BEING TESTED?
5133	034664	001032			BNE	22\$;;BRANCH IF IT ISN'T
5134	034666	012737	007600 172354		MOV	#7600,KIPAR6	;;MAP PAR6 TO THE I/O PAGE
5135	034674	012737	000600 172356		MOV	#600,KIPAR7	;;RE-MAP PAR7 TO 12K
5136	034702	011515			MOV	(R5),(R5)	;;NOW WRITE USING PAR7
5137	034704	042737	000001 157572		BIC	#BIT0,#157572	;;TURN OFF MEMORY MANAGMENT USING
5138							;;PAR6-ADDRESS OF MMRO
5139	034712	022737	077506 172316		CMP	#77506,KIPDR7	;;CHECK PDR 7 BEFORE RESTORING PDR/PAR'S
5140	034720	001005			BNE	24\$;;DID W-BIT GET SET?
5141							;;BRANCH IF NO
5142	034722	005237	001176		INC	\$TMP0	;;SET CORRECT PDR SET INDICATOR
5143	034726	012737	077406 172300		MOV	#77406,KIPDR0	;;RESTORE PDR0 AFTER WRITING \$TMP0
5144	034734	012737	077406 172314	24\$:	MOV	#77406,KIPDR6	;;RESET PDR6
5145	034742	012737	000600 172354		MOV	#600,KIPAR6	;;RE-MAP PAR6 TO 12K
5146	034750	000401			BR	23\$;;AND NOW COMPARE TO OTHER PDR'S
5147	034752	011515		22\$:	MOV	(R5),(R5)	;;WRITE INTO PAGE UNDER TEST
5148	034754			23\$:			
5149	034754	012702	000010		MOV	#10,R2	;;SET COUNTER TO READ NEXT 10 REGISTERS
5150	034760	012701	172300		MOV	#KIPDR0,R1	;;LOAD ADDRESS OF BEGINNING PDR
5151	034764	011100		7\$:	MOV	(R1),R0	;;READ PDR INTO R0
5152	034766	022700	077506		CMP	#77506,R0	;;SEE IF THIS WAS THE PDR WITH ITS W BIT ON
5153	034772	001030			BNE	9\$;;BRANCH IF THIS IS NOT THE ONE
5154	034774	020103			CMP	R1,R3	;;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
5155	034776	001424			BEQ	8\$;;BRANCH IF ADDRESS IS CORRECT
5156	035000	104062			ERROR	62	;;W BIT GOT SET IN WRONG PDR
5157	035002	012700	077406		MOV	#77406,R0	;;RE-SET PAGES MODIFIED BY ERROR
5158	035006	010037	172300		MOV	R0,KIPDR0	;;RELOAD KERNEL PDR0
5159	035012	010037	172302		MOV	R0,KIPDR1	;;RELOAD KERNEL PDR1
5160	035016	010037	172304		MOV	R0,KIPDR2	;;RELOAD KERNEL PDR2
5161	035022	010037	172316		MOV	R0,KIPDR7	;;RELOAD KERNEL PDR7
5162	035026	010037	172300		MOV	R0,KIPDR0	;;RELOAD PAGE 0 OF PRESENT SPACE
5163	035032	010037	172316		MOV	R0,KIPDR7	;;RE-LOAD I/O PAGE PDR IF ERROR
5164	035036	020527	170200		CMP	R5,#170200	;;SEE IF TESTING PDR7
5165	035042	001404			BEQ	9\$;;DON'T WRITE IF TESTING PDR7
5166	035044	011515			MOV	(R5),(R5)	;;TRY WRITE AGAIN, IN CASE YOU
5167							;;WERE NOT TESTING PAGE SEVEN
5168	035046	000402			BR	9\$;;GO UPDATE R1 FOR NEXT READ
5169	035050	005237	001176	8\$:	INC	\$TMP0	;;SET FLAG SINCE ADDRESSES MATCHED

JOB

CQKTA-80 PDP 11/6X MEM. MGMT. DIAG.
CQKTAB.P11 30-DEC-77 14:49

MACY11 30A(1052) 03-JAN-78 08:09 PAGE 100
T45 DUAL MAPPING KERNEL MODE SPACE

SEQ 0100

5170	035054	062701	000002		9\$:	ADD #2,R1	; POINT TO NEXT PDR TO BE READ
5171	035060	077237				SOB R2,7\$; BRANCH TO 7\$ IF ALL PDR'S NOT READ
5172	035062	012702	000010			MOV #10,R2	; SET COUNTER TO READ NEXT 10 REGISTERS
5173	035066	012701	177600			MOV #UIPDR0,R1	; LOAD ADDRESS OF BEGINNING PDR
5174	035072	011100			13\$:	MOV (R1),R0	; READ PDR INTO R0
5175	035074	022700	077506			CMP #77506,R0	; SEE IF THIS WAS THE PDR WITH ITS W BIT ON
5176	035100	001030				BNE 15\$; BRANCH IF THIS IS NOT THE ONE
5177	035102	020103				CMP R1,R3	; SEE IF THE ADDRESS MATCHES PDR UNDER TEST
5178	035104	001424				BEQ 14\$; BRANCH IF ADDRESS IS CORRECT
5179	035106	104062				ERROR 62	; W BIT GOT SET IN WRONG PDR
5180	035110	012700	077406			MOV #77406,R0	; RE-SET PAGES MODIFIED BY ERROR
5181	035114	010037	172300			MOV R0,KIPDR0	; RELOAD KERNEL PDR0
5182	035120	010037	172302			MOV R0,KIPDR1	; RELOAD KERNEL PDR1
5183	035124	010037	172304			MOV R0,KIPDR2	; RELOAD KERNEL PDR2
5184	035130	010037	172316			MOV R0,KIPDR7	; RELOAD KERNEL PDR7
5185	035134	010037	172300			MOV R0,KIPDR0	; RELOAD PAGE 0 OF PRESENT SPACE
5186	035140	010037	172316			MOV R0,KIPDR7	; RE-LOAD I/O PAGE PDR IF ERROR
5187	035144	020527	170200			CMP R5,#170200	; SEE IF TESTING PDR?
5188	035150	001404				BEQ 15\$; DON'T WRITE IF TESTING PDR?
5189	035152	011515				MOV (R5),(R5)	; TRY WRITE AGAIN, IN CASE YOU
5190							; WERE NOT TESTING PAGE SEVEN
5191	035154	000402				BR 15\$; GO UPDATE R1 FOR NEXT READ
5192	035156	005237	001176		14\$:	INC \$TMP0	; SET FLAG SINCE ADDRESSES MATCHED
5193	035162	062701	000002		15\$:	ADD #2,R1	; POINT TO NEXT PDR TO BE READ
5194	035166	077237				SOB R2,13\$; BRANCH TO 13\$ IF ALL PDR'S NOT READ
5195	035170	005737	001176			TST \$TMP0	; SEE IF THERE WAS A CORRECT PDR
5196	035174	001002				BNE 16\$; BRANCH IF THERE WAS
5197	035176	011300				MOV (R3),R0	; SAVE CONTENTS OF PDR UNDER TEST
5198	035200	104063				ERROR 63	; NO PDR ADDRESSES MATCHED
5199	035202	062703	000002		16\$:	ADD #2,R3	; POINT TO NEXT PDR UNDER TEST
5200	035206	062705	020000			ADD #20000,R5	; CHANGE PAGE NUMBER IN VIRT. ADDR.
5201	035212	005304				DEC R4	; DECREMENT COUNTER
5202	035214	001402				BEQ 17\$; BRANCH IF COUNTER IS ZERO
5203	035216	000137	034564			JMP 19\$; JUMP TO LOAD PDR'S AGAIN
5204	035222	012737	007600 172356		17\$:	MOV #7600,KIPAR7	; REMAP PAR7 TO THE I/O PAGE
5205	035230	052737	000001 177572			BIS #BIT0,MMR0	; REENABLE MEMORY MANAGEMENT AFTER
5206							; TESTING PDR?
5207	035236	012737	000340 177776			MOV #340,PSW	; RETURN TO KERNEL MODE, PRIORITY 7
5208	035244	012737	034534 001110			MOV #20\$,SLPERR	; 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 EXSISTING ADDRESS CAN'T BE GUARANTEED FOR THE I/O PAGE
    
```

5226 035252 000004 036000 001444 SCOPE
5227 035252 012737 036000 001444 MOV #TST47,NXTTST
5228 035252 012737 036000 001444 MOV #21\$,SLPERR
5229 035254 012737 140000 177776 MOV #140000,PSW
5230 035276 012703 177600 MOV #UIPDR0,R3
5231 035302 012704 000010 MOV #10,R4
5232 035306 012705 010200 MOV #10200,R5
5233 035312 012700 077406 19\$: MOV #77406,R0
5234 035316 005037 001176 CLR \$TMPO
5235 035322 012702 000010 MOV #10,R2
5236 035326 012701 172300 MOV #KIPDR0,R1
5237 035332 010021 172300 1\$: MOV RO,(R1)+
5238 035334 077202 SOB R2,1\$
5239 035336 012702 000010 MOV #10,R2
5240 035342 012701 177600 MOV #UIPDR0,R1
5241 035346 010021 177600 2\$: MOV RO,(R1)+
5242 035350 077202 SOB R2,2\$
5243 035352 012700 077406 21\$: MOV #77406,R0
5244 035356 010037 172300 MOV RO,KIPDR0
5245 035362 010037 172302 MOV RO,KIPDR1
5246 035366 010037 172304 MOV RO,KIPDR2
5247 035372 010037 172316 MOV RO,KIPDR7
5248 035376 010037 177600 MOV RO,UIPDR0
5249 035402 010037 177616 MOV RO,UIPDR7
5250 035406 020527 170200 CMP R5,#170200
5251 035412 001032 BNE 22\$
5252 035414 012737 007600 177654 MOV #7600,UIPAR6
5253 035422 012737 000600 177656 MOV #600,UIPAR7
5254 035430 011515 MOV (R5),(R5)
5255 035432 042737 000001 157572 BIC #BIT0,#157572
5256 035440 022737 077506 177616 CMP #77506,UIPDR7
5257 035446 001005 BNE 24\$
5258 035450 005237 001176 INC \$TMPO
5259 035454 012737 077406 177600 MOV #77406,UIPDR0
5260 035462 012737 077406 177614 24\$: MOV #77406,UIPDR6
5261 035470 012737 000600 177654 MOV #600,UIPAR6
5262 035476 000401 BR 23\$
5263 035500 011515 22\$: MOV (R5),(R5)
5264 035502 012702 000010 23\$: MOV #10,R2
5265 035506 012701 172300 MOV #KIPDR0,R1
5266 035512 011100 7\$: MOV (R1),RO
5267 035514 022700 077506 CMP #77506,RO
5268 035520 001030 BNE 9\$
5269 035522 020103 CMP R1,R3
5270 035524 001424 BEQ 8\$
5271 035526 104062 ERROR 62
5272 035530 012700 077406 MOV #77406,RO
5273 035534 010037 172300 MOV RO,KIPDR0
5274 035540 010037 172302 MOV RO,KIPDR1

5282	035544	010037	172304		MOV	R0,KIPDR2	;RELOAD KERNEL PDR2
5283	035550	010037	172316		MOV	R0,KIPDR7	;RELOAD KERNEL PDR7
5284	035554	010037	177600		MOV	R0,UIPDR0	;RELOAD PAGE 0 OF PRESENT SPACE
5285	035560	010037	177616		MOV	R0,UIPDR7	;RE-LOAD I/O PAGE PDR IF ERROR
5286	035564	020527	170200		CMP	R5,#170200	;SEE IF TESTING PDR7
5287	035570	001404			BEQ	9\$;DON'T WRITE IF TESTING PDR7
5288	035572	011515			MOV	(R5),(R5)	;TRY WRITE AGAIN, IN CASE YOU
5289							;WERE NOT TESTING PAGE SEVEN
5290	035574	000402			BR	9\$;GO UPDATE R1 FOR NEXT READ
5291	035576	005237	001176	8\$:	INC	\$TMP0	;SET FLAG SINCE ADDRESSES MATCHED
5292	035602	062701	000002	9\$:	ADD	#2,R1	;POINT TO NEXT PDR TO BE READ
5293	035606	077237			SOB	R2,7\$;BRANCH TO 7\$ IF ALL PDR'S NOT READ
5294	035610	012702	000010		MOV	#10,R2	;SET COUNTER TO READ NEXT 10 REGISTERS
5295	035614	012701	177600		MOV	#UIPDR0,R1	;LOAD ADDRESS OF BEGINNING PDR
5296	035620	011100		13\$:	MOV	(R1),R0	;READ PDR INTO R0
5297	035622	022700	077506		CMP	#77506,R0	;SEE IF THIS WAS THE PDR WITH ITS W BIT ON
5298	035626	001030			BNE	15\$;BRANCH IF THIS IS NOT THE ONE
5299	035630	020103			CMP	R1,R3	;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
5300	035632	001424			BEQ	14\$;BRANCH IF ADDRESS IS CORRECT
5301	035634	104062			ERROR	62	;W BIT GOT SET IN WRONG PDR
5302	035636	012700	077406		MOV	#77406,R0	;RE-SET PAGES MODIFIED BY ERROR
5303	035642	010037	172300		MOV	R0,KIPDR0	;RELOAD KERNEL PDR0
5304	035646	010037	172302		MOV	R0,KIPDR1	;RELOAD KERNEL PDR1
5305	035652	010037	172304		MOV	R0,KIPDR2	;RELOAD KERNEL PDR2
5306	035656	010037	172316		MOV	R0,KIPDR7	;RELOAD KERNEL PDR7
5307	035662	010037	177600		MOV	R0,UIPDR0	;RELOAD PAGE 0 OF PRESENT SPACE
5308	035666	010037	177616		MOV	R0,UIPDR7	;RE-LOAD I/O PAGE PDR IF ERROR
5309	035672	020527	170200		CMP	R5,#170200	;SEE IF TESTING PDR7
5310	035676	001404			BEQ	15\$;DON'T WRITE IF TESTING PDR7
5311	035700	011515			MOV	(R5),(R5)	;TRY WRITE AGAIN, IN CASE YOU
5312							;WERE NOT TESTING PAGE SEVEN
5313	035702	000402			BR	15\$;GO UPDATE R1 FOR NEXT READ
5314	035704	005237	001176	14\$:	INC	\$TMP0	;SET FLAG SINCE ADDRESSES MATCHED
5315	035710	062701	000002	15\$:	ADD	#2,R1	;POINT TO NEXT PDR TO BE READ
5316	035714	077237			SOB	R2,13\$;BRANCH TO 13\$ IF ALL PDR'S NOT READ
5317	035716	005737	001176		TST	\$TMP0	;SEE IF THERE WAS A CORRECT PDR
5318	035722	001002			BNE	16\$;BRANCH IF THERE WAS
5319	035724	011300			MOV	(R3),R0	;SAVE CONTENTS OF PDR UNDER TEST
5320	035726	104063			ERROR	63	;NO PDR ADDRESSES MATCHED
5321	035730	062703	000002	16\$:	ADD	#2,R3	;POINT TO NEXT PDR UNDER TEST
5322	035734	062705	020000		ADD	#20000,R5	;CHANGE PAGE NUMBER IN VIRT. ADDR.
5323	035740	005304			DEC	R4	;DECREMENT COUNTER
5324	035742	001402			BEQ	17\$;BRANCH IF COUNTER IS ZERO
5325	035744	000137	035312		JMP	19\$;JUMP TO LOAD PDR'S AGAIN
5326	035750	012737	007600	177656	MOV	#7600,UIPAR7	;REMAP PAR7 TO THE I/O PAGE
5327	035756	052737	000001	177572	BIS	#BIT0,MMR0	;REENABLE MEMORY MANAGEMENT AFTER
5328							;TESTING PDR7
5329	035764	012737	000340	177776	MOV	#340,PSW	;RETURN TO KERNEL MODE, PRIORITY 7
5330	035772	012737	035262	001110	MOV	#20\$,SLPERR	;SET LOOP POINTER TO START OF TEST
5331							
5332							
5333							
5334							
5335							
5336							
5337							

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

5394	036252	012737	000600	172350		MOV	#600,KIPAR4	;MAP KERNEL I PAGE 4 TO 12K
5395	036260	012737	000600	177650		MOV	#600,UIPAR4	;MAP USER I PAGE 4 TO 12K
5396	036266	012700	036514			MOV	#36514,R0	;LOAD DATA PATTERN INTO R0
5397	036272	010037	100000			MOV	R0,#100000	;LOAD DATA PATTERN INTO PHY 60000
5398	036276	012737	036674	000250		MOV	#10\$,MMVEC	;SET M.M. VECTOR TO 10\$
5399	036304	105037	172310			CLRB	KIPDR4	;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
5400								;THE FOLLOWING WILL TEST DSTM=0 MFPI
5401								
5402	036310	012737	036316	001110		MOV	#11\$,SLPERR	;SET LOOP ON ERROR POINTER TO 11\$
5403	036316	012737	030340	177776	11\$:	MOV	#030340,PSW	;MAKE PREVIOUS MODE USER
5404	036324	006506			1\$:	MFPI	USP	;PUT USER STACK POINTER ON KERNEL
5405								;STACK
5406	036326	022706	001100			CMP	#KERSTK,KSP	;WAS SOMETHING PUSHED ON STACK AT 1\$
5407	036332	001407				BEQ	3\$;BRANCH IF NOTHING WAS PUSHED
5408	036334	012601				MOV	(KSP)+,R1	;POP KERNEL STACK INTO R1
5409	036336	012702	000700			MOV	#UVESTK,R2	;EXPECTING TO GET 700 AS USP
5410	036342	020201				CMP	R2,R1	;DID YOU GET THE RIGHT POINTER?
5411	036344	001403				BEQ	2\$;BRANCH IF YOU DID
5412	036346	104064				ERROR	64	;WRONG THING WAS PUSHED ON STACK
5413	036350	000401				BR	2\$;BRANCH TO NEXT TRY
5414	036352	104065				ERROR	65	;NOTHING PUSHED ON STACK
5415	036354				3\$:			;THE FOLLOWING WILL TEST DSTM=1 MFPI.
5416	036354	012737	036362	001110		MOV	#12\$,SLPERR	;SET LOOP ON ERROR POINTER TO 12\$
5417	036362	012737	030340	177776	12\$:	MOV	#030340,PSW	;MAKE PREVIOUS MODE USER
5418	036370	012702	100000			MOV	#100000,R2	;LOAD VIRTUAL ADDRESS INTO R2
5419	036374	006512				MFPI	(R2)	;READ FROM PHYSICAL 60000
5420	036376	012601				MOV	(KSP)+,R1	;POP KERNEL STACK INTO R1
5421	036400	020001				CMP	R0,R1	;WAS DATA FETCHED SAME AS STORED
5422	036402	001401				BEQ	4\$;BRANCH IF CORRECT DATA WAS FETCHED
5423	036404	104066				ERROR	66	;WRONG DATA WAS FETCHED
5424	036406				4\$:			;THE FOLLOWING WILL TEST DSTM=2 MFPI.
5425	036406	012737	036414	001110		MOV	#14\$,SLPERR	;SET LOOP ON ERROR POINTER TO 14\$
5426	036414	012737	030340	177776	14\$:	MOV	#030340,PSW	;MAKE PREVIOUS MODE USER
5427	036422	012702	100000			MOV	#100000,R2	;LOAD VIRTUAL ADDRESS INTO R2
5428	036426	006522				MFPI	(R2)+	;READ FROM PHYSICAL 60000
5429	036430	012601				MOV	(KSP)+,R1	;POP KERNEL STACK INTO R1
5430	036432	020001				CMP	R0,R1	;WAS DATA FETCHED SAME AS STORED
5431	036434	001401				BEQ	5\$;BRANCH IF CORRECT DATA WAS FETCHED
5432	036436	104066				ERROR	66	;WRONG DATA WAS FETCHED
5433	036440				5\$:			;THE FOLLOWING WILL TEST DSTM=3 MFPI.
5434	036440	012737	036446	001110		MOV	#15\$,SLPERR	;SET LOOP ON ERROR POINTER TO 15\$
5435	036446	012737	030340	177776	15\$:	MOV	#030340,PSW	;MAKE PREVIOUS MODE USER
5436	036454	006537	100000			MFPI	#100000	;READ FROM PHYSICAL 60000
5437	036460	012601				MOV	(KSP)+,R1	;POP KERNEL STACK INTO R1
5438	036462	020001				CMP	R0,R1	;WAS DATA FETCHED SAME AS STORED
5439	036464	001401				BEQ	6\$;BRANCH IF CORRECT DATA WAS FETCHED
5440	036466	104066				ERROR	66	;WRONG DATA WAS FETCHED
5441	036470				6\$:			;THE FOLLOWING WILL TEST DSTM=4 MFPI.
5442	036470	012737	036476	001110		MOV	#16\$,SLPERR	;SET LOOP ON ERROR POINTER TO 16\$
5443	036476	012737	030340	177776	16\$:	MOV	#030340,PSW	;MAKE PREVIOUS MODE USER
5444	036504	012702	100002			MOV	#100002,R2	;LOAD VIRTUAL ADDRESS INTO R2
5445	036510	006542				MFPI	-(R2)	;READ FROM PHYSICAL 60000
5446	036512	012601				MOV	(KSP)+,R1	;POP KERNEL STACK INTO R1
5447	036514	020001				CMP	R0,R1	;WAS DATA FETCHED SAME AS STORED
5448	036516	001401				BEQ	7\$;BRANCH IF CORRECT DATA WAS FETCHED
5449	036520	104066				ERROR	66	;WRONG DATA WAS FETCHED

```

5450 036522 7$:
5451 ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
5452 ;
5453 036522 012737 036530 001110 MOV #17$, $LPERR ;SET LOOP ON ERROR POINTER TO 17$
5454 036530 012737 030340 177776 17$: MOV #030340, PSW ;MAKE PREVIOUS MODE USER
5455 036536 012737 100000 001202 MOV #100000, $TMP2 ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
5456 036544 012702 001204 MOV #<$TMP2+2>, R2 ;LOAD ADDR. OF $TMP2+2 INTO R2
5457 036550 006552 MFPI 2-(R2) ;READ FROM PHYSICAL 60000
5458 036552 012601 MOV (KSP)+, R1 ;POP KERNEL STACK INTO R1
5459 036554 020001 CMP R0, R1 ;WAS DATA FETCHED SAME AS STORED
5460 036556 001401 BEQ 8$ ;BRANCH IF CORRECT DATA WAS FETCHED
5461 036560 104066 ERROR 66 ;WRONG DATA WAS FETCHED
5462 036562 8$: ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
5463 ;
5464 036562 012737 036570 001110 MOV #18$, $LPERR ;SET LOOP ON ERROR POINTER TO 18$
5465 036570 012737 030340 177776 18$: MOV #030340, PSW ;MAKE PREVIOUS MODE USER
5466 036576 005002 CLR R2 ;MAKE REGISTER 2 A ZERO
5467 036600 006562 100000 MFPI 100000(R2) ;READ FROM PHYSICAL 60000
5468 036604 012601 MOV (KSP)+, R1 ;POP KERNEL STACK INTO R1
5469 036606 020001 CMP R0, R1 ;WAS DATA FETCHED SAME AS STORED
5470 036610 001401 BEQ 9$ ;BRANCH IF CORRECT DATA WAS FETCHED
5471 036612 104066 ERROR 66 ;WRONG DATA WAS FETCHED
5472 036614 9$: ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
5473 ;
5474 036614 012737 036622 001110 MOV #22$, $LPERR ;SET LOOP ON ERROR POINTER TO 22$
5475 036622 012737 030340 177776 22$: MOV #030340, PSW ;MAKE PREVIOUS MODE USER
5476 036630 012737 100000 001202 MOV #100000, $TMP2 ;LOAD TEST LOC. V.A. INTO $TMP2
5477 036636 012702 001202 MOV $TMP2, R2 ;LOAD ADDRESS OF $TMP2 INTO R2
5478 036642 006572 000000 MFPI 20(R2) ;USE $TMP2 TO FETCH VIRTUAL
5479 ; ADDRESS OF 60000
5480 036646 012601 MOV (KSP)+, R1 ;POP KERNEL STACK INTO R1
5481 036650 020001 CMP R0, R1 ;WAS DATA FETCHED SAME AS STORED
5482 036652 001401 BEQ 25$ ;BRANCH IF CORRECT DATA WAS FETCHED
5483 036654 104066 ERROR 66 ;WRONG DATA WAS FETCHED
5484 036656 012737 015430 000250 25$: MOV #MMTRAP, MMVEC ;SET M.M. VECTOR TO NORMAL ROUTINE
5485 036664 012737 036244 001110 MOV #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
5486 036672 000423 BR TST50 ;BRANCH TO NEXT TEST
5487
5488
5489 036674 012637 001434 10$: MOV (KSP)+, OLDPC ;SAVE PC & PS OF TRAP
5490 036700 012637 001436 MOV (KSP)+, OLDPS
5491 036704 013737 177572 001432 MOV MMRO, PMMR0 ;SAVE MMRO FOR ERROR TYPEOUT
5492 036712 013737 177576 001434 MOV MMR2, PMMR2 ;SAVE MMR2 FOR ERROR TYPEOUT
5493 036720 042737 160000 177572 BIC #160000, MMRO ;CLEAR ERROR BITS IN MMRO
5494 036726 104067 ERROR 67 ;TRIED TO READ NON-RESIDENT PAGE
5495 036730 013746 001436 MOV OLDPS, -(KSP) ;PUT PC & PS OF TRAP ON STACK
5496 036734 013746 001434 MOV OLDPC, -(KSP)
5497 036740 000002 RTI
5498
5499
5500
5501 *****
5502 *TEST 50 MOVE TO PREVIOUS (USER) I-SPACE
5503 *
5504 * THIS TEST USES THE 'MTPI' INSTRUCTION TO ENSURE THAT THE
5505 * PREVIOUS MODE IS CLOKED CORRECTLY
5506 * THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,

```

```

5506 ;*
5507 ;*
5508 ;* IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
5509 ;* WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
5510 ;*
5511 ;*
5512 ;*****
5513 TST50: SCOPE
5514 036742 000004 MOV #TST51,NXTTST ;SAVE STARTING ADDRESS OF NEXT
5515 036744 012737 037630 001444 ;TEST FOR ESCAPE ON PARITY ERRORS
5516 036752 012737 077406 172310 20$: MOV #77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE
5517 036760 012737 077406 177610 MOV #77406,UIPDR4 ;USER I-SPACE PAGE 4 READ/WRITE
5518 036766 012737 000600 172350 MOV #600,KIPAR4 ;MAP KERNEL I PAGE 4 TO 12K
5519 036774 012737 000600 177650 MOV #600,UIPAR4 ;MAP USER I PAGE 4 TO 12K
5520 037002 012737 037562 000250 MOV #10$,MMVEC ;SET M.M. VECTOR TO 10$
5521 ;THE FOLLOWING WILL TEST DSTM=0 MTPI
5522 ;
5523 037010 012737 030340 177776 1$: MOV #030340,PSW ;MAKE PREVIOUS MODE USER
5524 037016 012746 007777 MOV #7777,-(KSP) ;PUSH DATA ON KERNEL STACK
5525 037022 006606 MTPI USP ;LOAD USER STACK POINTER
5526 037024 006506 MFPI USP ;READ USER STACK POINTER
5527 037026 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
5528 037030 022701 007777 CMP #7777,R1 ;WAS USER STACK POINTER CHANGED
5529 037034 001401 BEQ 2$ ;BRANCH IF IT WAS
5530 037036 104070 ERROR 70 ;USER STACK POINTER NOT CHANGED
5531 037040 012737 030340 177776 2$: MOV #030340,PSW ;MAKE PREVIOUS MODE USER
5532 037046 012746 000700 MOV #USESTK,-(KSP) ;GET READY TO RESTORE USER S. POINT
5533 037052 006606 MTPI USP ;RESTORE USER STACK POINTER
5534 037054 ;THIS WILL TEST DSTM = 1 MTPI.
5535 037054 012737 037072 001110 3$: MOV #13$,SLPERR ;SET LOOP ON ERROR POINTER TO 13$
5536 037062 012702 100000 MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
5537 037066 012700 125252 MOV #125252,R0 ;LOAD TEST DATA INTO R0
5538 037072 010046 13$: MOV R0,-(KSP) ;PUSH TEST DATA ON KERNEL STACK
5539 037074 105037 172310 CLRB KIPDR4 ;MAKE KERNEL I PAGE 4 NON-RESIDENT
5540 037100 006612 MTPI (R2) ;LOAD TEST DATA INTO PHYSICAL 60000
5541 037102 112737 000006 172310 MOVB #006,KIPDR4 ;MAKE KERNEL PAGE 4 RESIDENT
5542 037110 011201 MOV (R2),R1 ;READ FROM ADDRESS 60000
5543 037112 020001 CMP R0,R1 ;SEE IF DATA WAS STORED AT CORRECT PLACE
5544 037114 001401 BEQ 4$ ;BRANCH IF STORE WAS CORRECT
5545 037116 104071 ERROR 71 ;INCORRECT STORE
5546 037120 4$: ;THIS WILL TEST DSTM = 3 MTPI.
5547 037120 012737 037140 001110 MOV #14$,SLPERR ;SET LOOP ON ERROR POINTER TO 14$
5548 037126 012737 030340 177776 MOV #030340,PSW ;MAKE PREVIOUS MODE USER
5549 037134 012700 052525 MOV #52525,R0 ;LOAD TEST DATA INTO R0
5550 037140 010046 14$: MOV R0,-(KSP) ;PUSH TEST DATA ON KERNEL STACK
5551 037142 105037 172310 CLRB KIPDR4 ;MAKE KERNEL I PAGE 4 NON-RESIDENT
5552 037146 006637 100000 MTPI @#100000 ;LOAD TEST DATA INTO PHYSICAL 60000
5553 037152 112737 000006 172310 MOVB #006,KIPDR4 ;MAKE KERNEL PAGE 4 RESIDENT
5554 037160 013701 100000 MOV @#100000,R1 ;READ FROM ADDRESS 60000
5555 037164 020001 CMP R0,R1 ;SEE IF DATA WAS STORED CORRECTLY
5556 037166 001401 BEQ 5$ ;BRANCH IF STORE WAS CORRECT
5557 037170 104071 ERROR 71 ;INCORRECT STORE
5558 037172 5$: ;THIS WILL TEST DSTM = 4 MTPI.
5559 037172 012737 037212 001110 MOV #15$,SLPERR ;SET LOOP ON ERROR POINTER TO 15$
5560 037200 012737 030340 177776 MOV #030340,PSW ;MAKE PREVIOUS MODE USER
5561 037206 012700 125252 MOV #125252,R0 ;LOAD TEST DATA INTO R0

```

```

5562 037212 010046          15$:  MOV      RO, -(KSP)          ; PUSH TEST DATA ON KERNEL STACK
5563 037214 012702 100002    MOV      #100002, R2         ; LOAD VIRTUAL ADDRESS INTO R2
5564 037220 105037 172310    CLR      KIPDR4            ; MAKE KERNEL I PAGE 4 NON-RESIDENT
5565 037224 006642          MTPI     -(R2)              ; LOAD TEST DATA INTO PHYSICAL 60000
5566 037226 112737 000006 172310  MOV      #006, KIPDR4       ; MAKE KERNEL PAGE 4 RESIDENT
5567 037234 013701 100000    MOV      @#100000, R1       ; READ FROM ADDRESS 60000
5568 037240 020001          CMP      RO, R1            ; SEE IF DATA WAS STORED CORRECTLY
5569 037242 001401          BEQ      6$                ; BRANCH IF STORE WAS CORRECT
5570 037244 104071          ERROR   71                ; INCORRECT STORE
5571 037246          ; THIS WILL TEST DSTM = 6 MTPI. BELOW ARE THE
5572          ;
5573 037246 012737 037270 001110  MOV      #16$, $LPERR       ; SET LOOP ON ERROR POINTER TO 16$
5574 037254 012737 030340 177776  MOV      #030340, PSW       ; MAKE PREVIOUS MODE USER
5575 037262 012700 052525    MOV      #52525, RO        ; LOAD TEST DATA INTO RO
5576 037266 005002          CLR      R2                ; MAKE REGISTER 2 ZERO
5577 037270 010046          16$:  MOV      RO, -(KSP)          ; PUSH TEST DATA ON KERNEL STACK
5578 037272 105037 172310    CLR      KIPDR4            ; MAKE KERNEL I PAGE 4 NON-RESIDENT
5579 037276 006662 100000    MTPI     100000(R2)        ; LOAD TEST DATA INTO PHYSICAL 60000
5580 037302 112737 000006 172310  MOV      #006, KIPDR4       ; MAKE KERNEL PAGE 4 RESIDENT
5581 037310 013701 100000    MOV      @#100000, R1       ; READ FROM ADDRESS 60000
5582 037314 020001          CMP      RO, R1            ; SEE IF DATA WAS STORED CORRECTLY
5583 037316 001401          BEQ      7$                ; BRANCH IF STORE WAS CORRECT
5584 037320 104071          ERROR   71                ; INCORRECT STORE
5585 037322          ; THE FOLLOWING WILL TEST DSTM=2 MTPI.
5586          ;
5587 037322 012737 037346 001110  MOV      #17$, $LPERR       ; SET LOOP ON ERROR POINTER TO 17$
5588 037330 012737 030340 177776  MOV      #030340, PSW       ; MAKE PREVIOUS MODE USER
5589 037336 012700 125252    MOV      #125252, RO        ; LOAD TEST DATA INTO RO
5590 037342 012702 100000    MOV      #100000, R2        ; LOAD VIRTUAL ADDRESS INTO R2
5591 037346 010046          17$:  MOV      RO, -(KSP)          ; PUSH TEST DATA ON KERNEL STACK
5592 037350 105037 172310    CLR      KIPDR4            ; MAKE KERNEL PAGE 4 NON-RESIDENT
5593 037354 006612          MTPI     (R2)              ; LOAD TEST DATA INTO PHYSICAL 60000
5594 037356 112737 000006 172310  MOV      #006, KIPDR4       ; MAKE KERNEL PAGE 4 RESIDENT
5595 037364 013701 100000    MOV      @#100000, R1       ; READ FROM ADDRESS 60000
5596 037370 020001          CMP      RO, R1            ; SEE IF DATA WAS STORED CORRECTLY
5597 037372 001401          BEQ      8$                ; BRANCH IF STORE WAS CORRECT
5598 037374 104071          ERROR   71                ; INCORRECT STORE
5599 037376          ; THE FOLLOWING WILL TEST DSTM=5 MTPI.
5600          ;
5601 037376 012737 037430 001110  MOV      #18$, $LPERR       ; SET LOOP ON ERROR POINTER TO 18$
5602 037404 012737 030340 177776  MOV      #030340, PSW       ; MAKE PREVIOUS MODE USER
5603 037412 012700 052525    MOV      #52525, RO        ; LOAD TEST DATA INTO RO
5604 037416 012702 001204          MOV      #(<$TMP2+2>), R2   ; LOAD ADDR. OF LOC. $TMP2+2 INTO R2
5605 037422 012737 100000 001202  MOV      #100000, $TMP2     ; LOAD VIRT. ADDR. OF TEST LOC. INTO $TMP2
5606 037430 010046          18$:  MOV      RO, -(KSP)          ; PUSH TEST DATA ON KERNEL STACK
5607 037432 105037 172310    CLR      KIPDR4            ; MAKE KERNEL PAGE 4 NON-RESIDENT
5608 037436 006652          MTPI     @-(R2)            ; LOAD TEST DATA INTO PHYSICAL 60000
5609 037440 112737 000006 172310  MOV      #006, KIPDR4       ; MAKE KERNEL PAGE 4 RESIDENT
5610 037446 013701 100000    MOV      @#100000, R1       ; READ FROM ADDRESS 60000
5611 037452 020001          CMP      RO, R1            ; SEE IF DATA WAS STORED CORRECTLY
5612 037454 001401          BEQ      9$                ; BRANCH IF STORE WAS CORRECT
5613 037456 104071          ERROR   71                ; INCORRECT STORE
5614 037460          ; THE FOLLOWING WILL TEST DSTM=7 MTPI.
5615          ;
5616 037460 012737 037512 001110  MOV      #19$, $LPERR       ; SET LOOP ON ERROR POINTER TO 19$
5617 037466 012737 030340 177776  MOV      #030340, PSW       ; MAKE PREVIOUS MODE USER

```

5618	037474	012700	125252			MOV	#125252,R0	;LOAD TEST DATA INTO R0
5619	037500	012737	100000	001202		MOV	#100000,\$TMP2	;LOAD VIRT. ADDR. OF TEST LOCATION
5620								;INTO LOCATION \$TMP2
5621	037506	012702	001202			MOV	#\$TMP2,R2	;LOAD ADDRESS OF \$TMP2 INTO R2
5622	037512	010046			19\$:	MOV	R0,-(KSP)	;PUSH TEST DATA ON KERNEL STACK
5623	037514	105037	172310			CLRB	KIPDR4	;MAKE KERNEL PAGE 4 NON-RESIDENT
5624	037520	006672	000000			MTPI	20(R2)	;LOAD TEST DATA INTO PHYSICAL 60000
5625	037524	112737	000006	172310		MOVB	#006,KIPDR4	;MAKE KERNEL PAGE 4 RESIDENT
5626	037532	013701	100000			MOV	2#100000,R1	;READ FROM ADDRESS 60000
5627	037536	020001				CMP	R0,R1	;SEE IF DATA WAS STORED CORRECTLY
5628	037540	001401				BEQ	25\$;BRANCH IF STORE WAS CORRECT
5629	037542	104071				ERROR	71	;INCORRECT STORE
5630	037544	012737	036752	001110	25\$:	MOV	#20\$,\$LPERR	;SET LOOP POINTER TO START OF TEST
5631	037552	012737	015430	000250		MOV	#MMTRAP,MMVEC	;RESTORE M.M. VECTOR TO NORMAL ROUTINE
5632	037560	000423				BR	TST51	;BRANCH TO NEXT TEST
5633								
5634								
5635	037562	012637	001434		10\$:	MOV	(KSP)+,OLDPC	;SAVE PC & PS OF TRAP
5636	037566	012637	001436			MOV	(KSP)+,OLDPS	
5637	037572	013737	177572	001402		MOV	MMR0,PMMR0	;SAVE MMR0 FOR ERROR TYPEOUT
5638	037600	013737	177576	001404		MOV	MMR2,PMMR2	;SAVE MMR2 FOR ERROR TYPEOUT
5639	037606	042737	160000	177572		BIC	#160000,MMR0	;CLEAR ERROR BITS IN MMR0
5640	037614	104067				ERROR	67	;TRIED TO LOAD A N.R. PAGE 4
5641	037616	013746	001436			MOV	OLDPS,-(KSP)	;PUT PC & PS OF TRAP ON STACK
5642	037622	013746	001434			MOV	OLDPC,-(KSP)	
5643	037626	000002				RTI		;RETURN TO TEST
5644								
5645								
5646								
5647								
5648								
5649								
5650								
5651								
5652								
5653								
5654								
5655								
5656								
5657								
5658								
5659								
5660	037630					TST51:		
5661	037630	000004				SCOPE		
5662	037632	012737	040376	001444		MOV	#TST52,NXTTST	;SAVE STARTING ADDRESS OF NEXT
5663								;TEST FOR ESCAPE ON PARITY ERRORS
5664	037640	012700	077406			MOV	#77406,R0	;MAKE ALL USER I-SPACE PAGES RESIDENT
5665								;READ/WRITE, LENGTH 200 BLOCKS
5666	037644	012702	000010			MOV	#10,R2	;SET COUNT TO LOAD 8 ADDRESSES
5667	037650	012701	177600			MOV	#UIPDR0,R1	;PUT ADDRESS OF FIRST PDR IN R1
5668	037654	010021			19\$:	MOV	R0,(R1)+	;LOAD R0 INTO PDR ADDRESSED BY R1
5669	037656	077202				SOB	R2,19\$;BRANCH BACK TO 19\$ IF R2 IS NOT ZERO
5670	037660	012737	037666	001110		MOV	#20\$,\$LPERR	;SET LOOP ON ERROR TO 20\$
5671	037666	012737	140340	177776	20\$:	MOV	#140340,PSW	;GO TO USER MODE FOR THIS TEST
5672	037674	012737	077406	172310		MOV	#77406,KIPDR4	;KERNEL I-SPACE PAGE 4 READ/WRITE
5673	037702	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.
*****

```



```

5730 040150 012737 040156 001110      MOV      #17$, $LPERR      ; SET LOOP ON ERROR POINTER TO 17$
5731 040156 012737 140340 177776 17$:  MOV      #140340, PSW      ; MAKE PREVIOUS MODE KERNEL PRESENT USER
5732 040164 012737 100000 001202      MOV      #100000, $TMP2    ; LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
5733 040172 012702 001204      MOV      <$TMP2+2>, R2     ; LOAD ADDRESS OF $TMP2+2 INTO R2
5734 040176 006552      MFPI     2-(R2)            ; READ FROM PHYSICAL 60000
5735 040200 012601      MOV      (USP)+, R1        ; POP USER STACK INTO R1
5736 040202 020001      CMP      R0, R1           ; WAS DATA FETCHED SAME AS STORED
5737 040204 001401      BEQ      8$               ; BRANCH IF CORRECT DATA FETCHED
5738 040206 104066      ERROR   66                ; WRONG DATA WAS FETCHED
5739 040210      ; THE FOLLOWING WILL TEST DSTM=6 MFPI.
5740
5741 040210 012737 040216 001110      MOV      #18$, $LPERR      ; SET LOOP ON ERROR POINTER TO 18$.
5742 040216 012737 140340 177776 18$:  MOV      #140340, PSW      ; MAKE PREVIOUS MODE KERNEL PRESENT USER
5743 040224 005002      CLR      R2               ; MAKE REGISTER 2 A ZERO
5744 040226 006562 100000      MFPI     100000(R2)        ; READ FROM PHYSICAL 60000
5745 040232 012601      MOV      (USP)+, R1        ; POP USER STACK INTO R1
5746 040234 020001      CMP      R0, R1           ; WAS DATA FETCHED SAME AS STORED
5747 040236 001401      BEQ      9$               ; BRANCH IF CORRECT DATA FETCHED
5748 040240 104066      ERROR   66                ; WRONG DATA WAS FETCHED
5749 040242      ; THE FOLLOWING WILL TEST DSTM=7 MFPI.
5750
5751 040242 012737 040250 001110      MOV      #22$, $LPERR      ; SET LOOP ON ERROR POINTER TO 22$
5752 040250 012737 140340 177776 22$:  MOV      #140340, PSW      ; MAKE PREVIOUS MODE KERNEL PRESENT USER
5753 040256 012737 100000 001202      MOV      #100000, $TMP2    ; LOAD TEST LOC. VIRT. ADDR. INTO $TMP2
5754 040264 012702 001202      MOV      $TMP2, R2         ; LOAD ADDRESS OF $TMP2 INTO R2
5755 040270 006572 000000      MFPI     20(R2)           ; READ FROM PHYSICAL 60000
5756 040274 012601      MOV      (USP)+, R1        ; POP USER STACK INTO R1
5757 040276 020001      CMP      R0, R1           ; WAS DATA FETCHED SAME AS STORED
5758 040300 001401      BEQ      25$              ; BRANCH IF CORRECT DATA FETCHED
5759 040302 104066      ERROR   66                ; WRONG DATA WAS FETCHED
5760 040304 012737 015430 000250 25$:  MOV      #MMTRAP, MMVEC    ; SET M.M. VECTOR TO NORMAL ROUTINE
5761 040312 012737 000340 177776      MOV      #00340, PSW      ; GO BACK TO KERNEL MODE, PREVIOUS KERNEL
5762 040320 012737 037666 001110      MOV      #20$, $LPERR      ; SET LOOP POINTER TO START OF TEST
5763 040326 000423      BR      TST52             ; BRANCH TO NEXT TEXT
5764
5765
5766 040330 012637 001434      MOV      (KSP)+, OLDPC     ; SAVE PC & PS OF TRAP
5767 040334 012637 001436      MOV      (KSP)+, OLDPS     ;
5768 040340 013737 177572 001402      MOV      MMR0, PMMR0       ; SAVE MMR0 FOR ERROR TYPEOUT
5769 040346 013737 177576 001404      MOV      MMR2, PMMR2       ; SAVE MMR2 FOR ERROR TYPEOUT
5770 040354 042737 160000 177572      BIC      #160000, MMR0     ; CLEAR ERROR BITS IN MMR0
5771 040362 104067      ERROR   67                ; TRIED TO READ NON-RESIDENT PAGE
5772 040364 013746 001436      MOV      OLDPS, -(KSP)     ; PUT PC & PS OF TRAP ON STACK
5773 040370 013746 001434      MOV      OLDPC, -(KSP)     ;
5774 040374 000002      RTI                       ; RETURN TO TEST
5775
5776
5777
5778
5779
5780
5781
5782
5783
5784 040376
5785 040376 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
: * MFPI & MTPD SHOULD BE DECODED THE SAME AS MFPI & MTPI.
: *

†TST52: SCOPE

```

5786 040400 012737 040510 001444      MOV      #TST53,NXTTST      ;SAVE STARTING ADDRESS OF NEXT
5787                                     ;TEST FOR ESCAPE ON PARITY ERRORS
5788 040406 012737 030340 177776 20$:  MOV      #030340,PSW      ;MAKE PREVIOUS MODE=USER,CURRENT=KERNEL
5789 040414 106506                                     MFPD     USP              ;MFPD SHOULD ACT LIKE MFPI PUTTING
5790                                     ;USER STACK POINTER ON THE KERNEL STACK
5791 040416 022706 001100      CMP      #KERSTK,KSP      ;WAS SOMETHING PUSHED ON KERNEL STACK?
5792 040422 001407      BEQ     2$                ;BRANCH IF NO
5793 040424 012601      MOV     (KSP)+,R1        ;POP KERNEL STACK INTO R1
5794 040426 012702 000700      MOV     #USESTK,R2      ;EXPECTING TO GET 700 AS USP
5795 040432 020201      CMP     R2,R1           ;DID GET RIGHT POINTER VALUE?
5796 040434 001403      BEQ     3$                ;BRANCH IF YES
5797 040436 104064      ERROR  64                ;WRONG THING WAS PUSHED ON STACK
5798 040440 000401      BR     3$                ;BRANCH TO NEXT TRY
5799 040442 104065      ERROR  65                ;NOTHING PUSHED ON STACK
5800 040444 012737 040452 001110 2$:  MOV     #4$, $LPERR      ;SET LOOP ON ERROR POINTER TO 4$
5801 040452 012746 007777 4$:  MOV     #7777,-(KSP)    ;PUSH DATA ON KERNEL STACK
5802 040456 106606      MTPD   USP              ;LOAD THE USER STACK POINTER
5803 040460 106506      MFPD   USP              ;READ USER STACK POINTER
5804 040462 012601      MOV     (KSP)+,R1        ;POP KERNEL STACK INTO R1
5805 040464 022701 007777      CMP     #7777,R1        ;WAS USER STACK POINTER CHANGED?
5806 040470 001401      BEQ     5$                ;BRANCH IF YES
5807 040472 104070      ERROR  70                ;USER STACK POINTER NOT CHANGED
5808 040474 012746 000700 5$:  MOV     #USESTK,-(KSP)  ;GET READY TO RESTORE USER STK. PTR.
5809 040500 106606      MTPD   USP              ;RESTORE USER STACK POINTER
5810 040502 012737 040406 001110  MOV     #20$, $LPERR    ;SET LOOP POINTER TO START OF TEST

```

```

5811
5812 ;*****
5813 ;*TEST 53      MOVE FROM PREVIOUS I-SPACE (PREVIOUS=CURRENT=KERNEL)
5814 ;*
5815 ;*      THIS TEST CHECKS THAT IF BOTH PREVIOUS AND CURRENT MODES
5816 ;*      ARE KERNEL, AND THE SOURCE MODE IS 0, THE DESTINATION
5817 ;*      STACK IS NOT DECREMENTED BEFORE ACCESS.
5818 ;*      (FOR 11/40 THIS IS PURPOSE OF 'HOOK MUX')
5819 ;*      "MFPI KSP" SHOULD PUSH THE NON-DECREMENTED VALUE
5820 ;*      OF KSP (1100) ONTO THE STACK (AT LOC. 1076).
5821 ;*****

```

```

5822 040510 000004      TST53: SCOPE
5823 040512 012737 040626 001444      MOV     #EOP,NXTTST      ;SET ESCAPE POINTER TO EOP ROUTINE
5824 040520 012737 040526 001110      MOV     #20$, $LPERR    ;SET LOOP ON ERROR POINTER TO 20$
5825 040526 005037 177776      CLR     0#PSW           ;SET PREVIOUS = CURRENT = KERNEL
5826 040532 012700 001100      MOV     #STACK,RO      ;SETUP VALUE FOR STACK POINTER
5827 040536 010006      MOV     RO,KSP         ;LOAD STACK POINTER
5828 040540 006506      MFPI   KSP             ;THE VALUE "STACK" SHOULD BE PUSHED
5829                                     ;BEFORE BEING DECREMENTED
5830 040542 011601      MOV     (KSP),R1       ;READ DATA WHICH WAS PUSHED
5831 040544 020001      CMP     RO,R1          ;WAS THE ORIGINAL VALUE OF THE
5832                                     ;STACK POINTER PUSHED?
5833 040546 001401      BEQ     1$                ;BRANCH IF YES
5834 040550 104066      ERROR  66                ;MFPI FETCHED WRONG DATA
5835 040552 005740 1$:  TST     -(RO)           ;SETUP EXPECTED STACK POINTER VALUE
5836 040554 020600      CMP     KSP,RO         ;WAS THE STACK POINTER DECREMENTED?
5837 040556 001401      BEQ     2$                ;BRANCH IF YES
5838 040560 104065      ERROR  65                ;STACK NOT PUSHED BY THE MFPI
5839 040562 012706 001100 2$:  MOV     #STACK,KSP     ;RESTORE STACK POINTER
5840 040566 000417      BR     $EOP
5841 ;*****

```



```

5842
5843
5844
5845 040570 012700 000000
5846 040574 076600
5847 040576 000300
5848 040600 076600
5849 040602 000301
5850 040604 076600
5851 040606 000304
5852 040610 076600
5853 040612 000222
5854 040614 076600
5855 040616 000226
5856 040620 076600
5857 040622 000305
5858 040624 000207
5859
5860
5861
5862
5863
5864
5865
5866
5867
5868
5869
5870
5871
5872
5873
5874
5875
5876
5877
5878 040626
5879 040626 000004
5880 040630 005037 001102
5881 040634 005037 001212
5882 040640 005237 001234
5883 040644 042737 100000 001234
5884 040652 005327
5885 040654 000001
5886 040656 003072
5887 040660 012737
5888 040662 000001
5889 040664 040654
5890 040666 104401 040674
5891 040672 000407
5892
5893 040712
5894 040712 013746 001234
5895
5896 040716 104405
5897 040720 104401 040726

```

```

; ROUTINE TO CLR MED REGISTERS
; NOT ALL MED REGISTERS ARE CLEARED
; ONLY THOSE USED BY THIS DIAGNOSTIC
CLRUIW: MOV #0,RO ;CLR RO TO WRITE INTO MED'S
MED
WLJAM ;LOG JAM REG
MED
WLSERV ;LOG SERVICE REG
MED
WLFGIN ;LOG FLAG INTERRUPT REG
MED
WVHAM ;WHAMI REG
MED
WCNSSW ;CONSOLE SWITCH REG
MED
WLWHAM ;LOG WHAMI REG
RTS PC
;*****

.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 $STNM ;: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
TYPE $EOPCT
TYPE ,65$ ;:TYPE ASCIZ STRING
BR ,64$ ;:GET OVER THE ASCIZ
;:65$: .ASCIZ <12><15>/END PASS #/
;:64$: MOV $PASS,-(SP) ;:SAVE $PASS FOR TYPEOUT
;:TYPE PASS NUMBER
;:GO TYPE--DECIMAL ASCII WITH SIGN
;:TYPE ASCIZ STRING

```

5898 040724 000421
5899
5900 040770
5901 040770 013746 001112
5902
5903 040774 104405
5904 040776 104401 001223
5905 041002 005037 001112
5906 041006 013700 000042
5907 041012 001414
5908 041014 005046
5909 041016 012746 041024
5910 041022 000426
5911
5912 041024
5913 041024 013700 000042
5914 041030 001405
5915 041032 000005
5916 041034 004710
5917 041036 000240
5918 041040 000240
5919 041042 000240
5920 041044
5921 041044 104400
5922 041046 042716 000020
5923 041052 032777 010000 140060
5924 041060 001005
5925 041062 005137 041106
5926 041066 100402
5927 041070 052716 000020
5928 041074 012746 041102
5929 041100 000002
5930
5931
5932 041102
5933 041102 000137
5934 041104 020620
5935 041106 000000
5936 041110 377 000
5937 041114
5938
5939
5940
5941
5942
5943
5944
5945
5946
5947
5948
5949
5950
5951
5952
5953

BR 66\$::GET OVER THE ASCIZ
::67\$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
66\$: MOV \$ERTTL,-(SP) ::SAVE \$ERTTL FOR TYPEOUT
::TOTAL NUMBER OF ERRORS
TYPDS ::GO TYPE--DECIMAL ASCII WITH SIGN
TYPE \$CRLF ::TYPE CARRIAGE RETURN, LINE FEED
CLR \$ERTTL ::CLEAR ERROR TOTAL
\$GET42: MOV @#42,RO ::GET MONITOR ADDRESS
BEQ \$DOAGN ::BRANCH IF NO MONITOR
CLR -(SP) ::INSURE THE "T" BIT IS CLEAR
MOV #SCLR.T,-(SP) ::SETUP FOR AN RTI OR RTT
BR \$RTRN ::GO DO AN RTI OR RTT TO LOAD THE PSW
::WITH A CLEARED "T" BIT
SCLR.T: MOV @#42,RO ::INSURE RO CONTAINS THE MONITORS
BEQ \$DOAGN ::RETURN ADDRESS
RESET ::CLEAR THE WORLD
\$ENDAD: JSR PC,(RO) ::GO TO MONITOR
NOP ::SAVE ROOM
NOP ::FOR
NOP ::ACT11
\$DOAGN: TRAP ::PUSH OLD PSW AND PC ON STACK
BIC #20,(SP) ::CLEAR THE "T" BIT
BIT #BIT12,@SWR ::RUN WITH TRACE TRAP?
BNE 1\$::BR IF NO
COM \$TBIT ::IS IT TIME FOR TRACE TRAP
BMI 1\$::BR IF NO
BIS #20,(SP) ::SET TRACE TRAP
1\$: MOV #SLOOP,-(SP) ::JUMP TO START OF TEST
\$RTRN: RTI ::RETURN--THIS IS CHANGED TO
::AN "RTT" IF "RTT" IS A LEGAL
::INSTRUCTION
SLOOP: JMP @PC+ ::RETURN
\$RTNAD: .WORD LOOP
\$TBIT: .WORD 0 ::"T" BIT STATE INDICATOR
\$ENULL: .BYTE -1,-1,0 ::NULL CHARACTER STRING
.EVEN
.SBTTL SAVE AND RESTORE RO-R5 ROUTINES

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

```

5954
5955
5956 041114
5957 041114 010046
5958 041116 010146
5959 041120 010246
5960 041122 010346
5961 041124 010446
5962 041126 010546
5963 041130 016646 000022
5964 041134 016646 000022
5965 041140 016646 000022
5966 041144 016646 000022
5967 041150 000002
5968
5969
5970
5971
5972 041152
5973 041152 012666 000022
5974 041156 012666 000022
5975 041162 012666 000022
5976 041166 012666 000022
5977 041172 012605
5978 041174 012604
5979 041176 012603
5980 041200 012602
5981 041202 012601
5982 041204 012600
5983 041206 000002
5984
5985
5986
5987
5988
5989
5990
5991
5992
5993
5994
5995
5996
5997
5998
5999
6000
6001 041210 105737 001157
6002 041214 100002
6003 041216 000000
6004 041220 000430
6005 041222 010046
6006 041224 017600 000002
6007 041230 122737 000001 001246
6008 041236 001011
6009 041240 132737 000100 001247

```

```

; *+14---RO
$SAVREG:
MOV RO, -(SP) ;: PUSH RO ON STACK
MOV R1, -(SP) ;: PUSH R1 ON STACK
MOV R2, -(SP) ;: PUSH R2 ON STACK
MOV R3, -(SP) ;: PUSH R3 ON STACK
MOV R4, -(SP) ;: PUSH R4 ON STACK
MOV R5, -(SP) ;: PUSH R5 ON STACK
MOV 22(SP), -(SP) ;: SAVE PS OF MAIN FLOW
MOV 22(SP), -(SP) ;: SAVE PC OF MAIN FLOW
MOV 22(SP), -(SP) ;: SAVE PS OF CALL
MOV 22(SP), -(SP) ;: SAVE PC OF CALL
RTI

```

```

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

```

.SBTTL TYPE ROUTINE

```

; *****
; *ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
; *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
; *NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
; *NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
; *NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
; *
; *CALL:
; *1) USING A TRAP INSTRUCTION
; * TYPE ,MESADR ;: MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
; *OR
; * TYPE
; * MESADR
; *

```

```

$TYPE: TSTB $TPFLG ;: IS THERE A TERMINAL?
BPL 1$ ;: BR IF YES
HALT ;: HALT HERE IF NO TERMINAL
BR 3$ ;: LEAVE
1$: MOV RO, -(SP) ;: SAVE RO
MOV 22(SP), RO ;: GET ADDRESS OF ASCIZ STRING
CMPB #APTENV, $ENV ;: RUNNING IN APT MODE
BNE 62$ ;: NO, GO CHECK FOR APT CONSOLE
BITB #APTPOOL, $ENVM ;: SPOOL MESSAGE TO APT

```

```

6010 041246 001405 BEQ 62$ ; NO GO CHECK FOR CONSOLE
6011 041250 010037 041260 MOV RO,61$ ; SETUP MESSAGE ADDRESS FOR APT
6012 041254 004737 041500 JSR PC,$ATY3 ; SPOOL MESSAGE TO APT
6013 041260 000000 .WORD 0 ; MESSAGE ADDRESS
6014 041262 132737 000040 001247 62$: BITB #APTCSUP,$ENVM ; APT CONSOLE SUPPRESSED
6015 041270 001003 BNE 60$ ; YES, SKIP TYPE OUT
6016 041272 112046 2$: MOVB (RO)+,-(SP) ; PUSH CHARACTER TO BE TYPED ONTO STACK
6017 041274 001005 BNE 4$ ; BR IF IT ISN'T THE TERMINATOR
6018 041276 005726 TST (SP)+ ; IF TERMINATOR POP IT OFF THE STACK
6019 041300 012600 60$: MOV (SP)+,RO ; RESTORE RO
6020 041302 062716 000002 3$: ADD #2,(SP) ; ADJUST RETURN PC
6021 041306 000002 RTI ; RETURN
6022 041310 122716 000011 4$: CMPB #HT,(SP) ; BRANCH IF <HT>
6023 041314 001430 BEQ 8$ ; BRANCH IF NOT <CRLF>
6024 041316 122716 000200 CMPB #CRLF,(SP) ; BRANCH IF NOT <CRLF>
6025 041322 001006 BNE 5$ ; POP <CR><LF> EQUIV
6026 041324 005726 TST (SP)+ ; TYPE A CR AND LF
6027 041326 104401 TYPE $CRLF ; CLEAR CHARACTER COUNT
6028 041330 001223 $CRLF CLRB $CHARCNT ; GET NEXT CHARACTER
6029 041332 105037 041466 BR 2$ ; GO TYPE THIS CHARACTER
6030 041336 000755 JSR PC,$TYPEC ; IS IT TIME FOR FILLER CHARS.?
6031 041340 004737 041422 5$: JSR PC,$TYPEC ; IS IT TIME FOR FILLER CHARS.?
6032 041344 123726 001156 6$: CMPB $FILLC,(SP)+ ; IF NO GO GET NEXT CHAR.
6033 041350 001350 BNE 2$ ; GET # OF FILLER CHARS. NEEDED
6034 041352 013746 001154 MOV $NULL,-(SP) ; AND THE NULL CHAR.
6035 041356 105366 000001 7$: DECB 1(SP) ; DOES A NULL NEED TO BE TYPED?
6036 041362 002770 BLT 6$ ; BR IF NO--GO POP THE NULL OFF OF STACK
6037 041362 002770 JSR PC,$TYPEC ; GO TYPE A NULL
6038 041364 004737 041422 DECB $CHARCNT ; DO NOT COUNT AS A COUNT
6039 041370 105337 041466 BR 7$ ; LOOP
6040 041374 000770
6041
6042 ;HORIZONTAL TAB PROCESSOR
6043
6044 041376 112716 000040 8$: MOVB #' (SP) ; REPLACE TAB WITH SPACE
6045 041402 004737 041422 9$: JSR PC,$TYPEC ; TYPE A SPACE
6046 041406 132737 000007 041466 BITB #7,$CHARCNT ; BRANCH IF NOT AT
6047 041414 001372 BNE 9$ ; TAB STOP
6048 041416 005726 TST (SP)+ ; POP SPACE OFF STACK
6049 041420 000724 BR 2$ ; GET NEXT CHARACTER
6050 041422 105777 137522 $TYPEC: TSTB $STPS ; WAIT UNTIL PRINTER IS READY
6051 041426 100375 BPL $TYPEC
6052 041430 116677 000002 137514 MOVB 2(SP),$STPB ; LOAD CHAR TO BE TYPED INTO DATA REG.
6053 041436 122766 000015 000002 CMPB #CR,2(SP) ; IS CHARACTER A CARRIAGE RETURN?
6054 041444 001003 BNE 1$ ; BRANCH IF NO
6055 041446 105037 041466 CLRB $CHARCNT ; YES--CLEAR CHARACTER COUNT
6056 041452 000406 BR $TYPEX ; EXIT
6057 041454 122766 000012 000002 1$: CMPB #LF,2(SP) ; IS CHARACTER A LINE FEED?
6058 041462 001402 BEQ $TYPEX ; BRANCH IF YES
6059 041464 105227 INCB (PC)+ ; COUNT THE CHARACTER
6060 041466 000000 $CHARCNT: .WORD 0 ; CHARACTER COUNT STORAGE
6061 041470 000207 $TYPEX: RTS PC
6062
6063 .SBTTL APT COMMUNICATIONS ROUTINE
6064
6065 ;*****

```

```

6066 041472 112737 000001 041736 $ATY1: MOVB #1,$FFLG ;; TO REPORT FATAL ERROR
6067 041500 112737 000001 041734 $ATY3: MOVB #1,$MFLG ;; TO TYPE A MESSAGE
6068 041506 000403 BR $ATYC
6069 041510 112737 000001 041736 $ATY4: MOVB #1,$FFLG ;; TO ONLY REPORT FATAL ERROR
6070 041516 $ATYC:
6071 041516 010046 MOV RO,-(SP) ;; PUSH RO ON STACK
6072 041520 010146 MOV R1,-(SP) ;; PUSH R1 ON STACK
6073 041522 105737 041734 TSTB $MFLG ;; SHOULD TYPE A MESSAGE?
6074 041526 001450 BEQ 5$ ;; IF NOT: BR
6075 041530 122737 000001 001246 CMPB #APTENV,$ENV ;; OPERATING UNDER APT?
6076 041536 001031 BNE 3$ ;; IF NOT: BR
6077 041540 132737 000100 001247 BITB #APTPOOL,$ENVM ;; SHOULD SPOOL MESSAGES?
6078 041546 001425 BEQ 3$ ;; IF NOT: BR
6079 041550 017600 000004 MOV #4(SP),RO ;; GET MESSAGE ADDR.
6080 041554 062766 000002 000004 ADD #2,4(SP) ;; BUMP RETURN ADDR.
6081 041562 005737 001226 1$: TST $MSGTYPE ;; SEE IF DONE W/ LAST XMISSION?
6082 041566 001375 BNE 1$ ;; IF NOT: WAIT
6083 041570 010037 001242 MOV RO,$MSGAD ;; PUT ADDR IN MAILBOX
6084 041574 105720 2$: TSTB (RO)+ ;; FIND END OF MESSAGE
6085 041576 001376 BNE 2$
6086 041600 163700 001242 SUB $MSGAD,RO ;; SUB START OF MESSAGE
6087 041604 006200 ASR RO ;; GET MESSAGE LNTH IN WORDS
6088 041606 010037 001244 MOV RO,$MSG LGT ;; PUT LENGTH IN MAILBOX
6089 041612 012737 000004 001226 MOV #4,$MSGTYPE ;; TELL APT TO TAKE MSG.
6090 041620 000413 BR 5$
6091 041622 017637 000004 041646 3$: MOV #4(SP),4$ ;; PUT MSG ADDR IN JSR LINKAGE
6092 041630 062766 000002 000004 ADD #2,4(SP) ;; BUMP RETURN ADDRESS
6093 041636 013746 177776 MOV 177776,-(SP) ;; PUSH 177776 ON STACK
6094 041642 004737 041210 JSR PC,$TYPE ;; CALL TYPE MACRO
6095 041646 000000 4$: .WORD 0
6096 041650 5$:
6097 041650 105737 041736 10$: TSTB $FFLG ;; SHOULD REPORT FATAL ERROR?
6098 041654 001416 BEQ 12$ ;; IF NOT: BR
6099 041656 005737 001246 TST $ENV ;; RUNNING UNDER APT?
6100 041662 001413 BEQ 12$ ;; IF NOT: BR
6101 041664 005737 001226 11$: TST $MSGTYPE ;; FINISHED LAST MESSAGE?
6102 041670 001375 BNE 11$ ;; IF NOT: WAIT
6103 041672 017637 000004 001230 MOV #4(SP),$FATAL ;; GET ERROR #
6104 041700 062766 000002 000004 ADD #2,4(SP) ;; BUMP RETURN ADDR.
6105 041706 005237 001226 INC $MSGTYPE ;; TELL APT TO TAKE ERROR
6106 041712 105037 041736 12$: CLRB $FFLG ;; CLEAR FATAL FLAG
6107 041716 105037 041735 CLRB $LFLG ;; CLEAR LOG FLAG
6108 041722 105037 041734 CLRB $MFLG ;; CLEAR MESSAGE FLAG
6109 041726 012601 MOV (SP)+,R1 ;; POP STACK INTO R1
6110 041730 012600 MOV (SP)+,RO ;; POP STACK INTO RO
6111 041732 000207 RTS PC ;; RETURN
6112 041734 000 $MFLG: .BYTE 0 ;; MESSG. FLAG
6113 041735 000 $LFLG: .BYTE 0 ;; LOG FLAG
6114 041736 000 $FFLG: .BYTE 0 ;; FATAL FLAG
6115 041740 .EVEN
6116 000200 APTSIZE=200
6117 000001 APTENV=001
6118 000100 APTPOOL=100
6119 000040 APTCSUP=040
6120 .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
6121

```

6122
6123
6124
6125
6126
6127
6128
6129
6130
6131
6132
6133
6134
6135
6136
6137
6138
6139
6140
6141
6142
6143
6144
6145
6146
6147
6148
6149
6150
6151
6152
6153
6154
6155
6156
6157
6158
6159
6160
6161
6162
6163
6164
6165
6166
6167
6168
6169
6170
6171
6172
6173
6174
6175
6176
6177

041740	017646	000000	
041744	116637	000001	042163
041752	112637	042165	
041756	062716	000002	
041762	000406		
041764	112737	000001	042163
041772	112737	000006	042165
042000	112737	000005	042162
042006	010346		
042010	010446		
042012	010546		
042014	113704	042165	
042020	005404		
042022	062704	000006	
042026	110437	042164	
042032	113704	042163	
042036	016605	000012	
042042	005003		
042044	006105		
042046	000404		
042050	006105		
042052	006105		
042054	006105		
042056	010503		
042060	006103		
042062	105337	042164	
042066	100016		
042070	042703	177770	
042074	001002		
042076	005704		
042100	001403		
042102	005204		
042104	052703	000060	

```

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPOS   N              ;;CALL FOR TYPEOUT
*   .BYTE  M              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*   .BYTE  M              ;;M=1 OR 0
*                               ;;1=TYPE LEADING ZEROS
*                               ;;0=SUPPRESS LEADING ZEROS
*$STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC
*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPON   N              ;;CALL FOR TYPEOUT
*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPOC   N              ;;CALL FOR TYPEOUT
*$TYPOS:
*   MOV     2(SP),-(SP)    ;;PICKUP THE MODE
*   MOV     1(SP),SOFILL   ;;LOAD ZERO FILL SWITCH
*   MOV     (SP)+,SOMODE+1 ;;NUMBER OF DIGITS TO TYPE
*   ADD     #2,(SP)        ;;ADJUST RETURN ADDRESS
*   BR     $TYPON
*$TYPOC:
*   MOV     #1,SOFILL      ;;SET THE ZERO FILL SWITCH
*   MOV     #6,SOMODE+1    ;;SET FOR SIX(6) DIGITS
*$TYPON:
*   MOV     #5,SOCNT       ;;SET THE ITERATION COUNT
*   MOV     R3,-(SP)       ;;SAVE R3
*   MOV     R4,-(SP)       ;;SAVE R4
*   MOV     R5,-(SP)       ;;SAVE R5
*   MOV     SOMODE+1,R4    ;;GET THE NUMBER OF DIGITS TO TYPE
*   NEG     R4
*   ADD     #6,R4          ;;SUBTRACT IT FOR MAX. ALLOWED
*   MOV     R4,SOMODE      ;;SAVE IT FOR USE
*   MOV     SOFILL,R4      ;;GET THE ZERO FILL SWITCH
*   MOV     12(SP),R5      ;;PICKUP THE INPUT NUMBER
*   CLR     R3             ;;CLEAR THE OUTPUT WORD
*   ROL     R5             ;;ROTATE MSB INTO "C"
*   BR     3$             ;;GO DO MSB
*   ROL     R5             ;;FORM THIS DIGIT
*   ROL     R5
*   ROL     R5
*   MOV     R5,R3
*   ROL     R3             ;;GET LSB OF THIS DIGIT
*   DECB   $SOMODE        ;;TYPE THIS DIGIT?
*   BPL    7$             ;;BR IF NO
*   BIC    #177770,R3     ;;GET RID OF JUNK
*   BNE   4$             ;;TEST FOR 0
*   TST   R4             ;;SUPPRESS THIS 0?
*   BEQ   5$             ;;BR IF YES
*   INC   R4             ;;DON'T SUPPRESS ANYMORE 0'S
*   BIS   #'0,R3         ;;MAKE THIS DIGIT ASCII
  
```

```

6178 042110 052703 000040
6179 042114 110337 042160
6180 042120 104401 042160
6181 042124 105337 042162
6182 042130 003347
6183 042132 002402
6184 042134 005204
6185 042136 000744
6186 042140 012605
6187 042142 012604
6188 042144 012603
6189 042146 016666 000002 000004
6190 042154 012616
6191 042156 000002
6192 042160 000
6193 042161 000
6194 042162 000
6195 042163 000
6196 042164 000000
6197
6198
6199
6200
6201
6202
6203
6204
6205
6206
6207
6208
6209 042166
6210 042166 010046
6211 042170 010146
6212 042172 010246
6213 042174 010346
6214 042176 010546
6215 042200 012746 020200
6216 042204 016605 000020
6217 042210 100004
6218 042212 005405
6219 042214 112766 000055 000001
6220 042222 005000
6221 042224 012703 042402
6222 042230 112723 000040
6223 042234 005002
6224 042236 016001 042372
6225 042242 160105
6226 042244 002402
6227 042246 005202
6228 042250 000774
6229 042252 060105
6230 042254 005702
6231 042256 001002
6232 042260 105716
6233 042262 100407

```

```

5$: BIS #' R3 ;; MAKE ASCII IF NOT ALREADY
   MOV R3, R3 ;; SAVE FOR TYPING
   TYPE R3, R3 ;; GO TYPE THIS DIGIT
7$: DECB $OCNT ;; COUNT BY 1
   BGT 2$ ;; BR IF MORE TO DO
   BLT 6$ ;; BR IF DONE
   INC R4 ;; INSURE LAST DIGIT ISN'T A BLANK
   BR 2$ ;; GO DO THE LAST DIGIT
6$: MOV (SP)+, R5 ;; RESTORE R5
   MOV (SP)+, R4 ;; RESTORE R4
   MOV (SP)+, R3 ;; RESTORE R3
   MOV 2(SP), 4(SP) ;; SET THE STACK FOR RETURNING
   MOV (SP)+, (SP)
   RTI ;; RETURN
8$: .BYTE 0 ;; STORAGE FOR ASCII DIGIT
   .BYTE 0 ;; TERMINATOR FOR TYPE ROUTINE
$OCNT: .BYTE 0 ;; OCTAL DIGIT COUNTER
$OFILL: .BYTE 0 ;; ZERO FILL SWITCH
$OMODE: .WORD 0 ;; NUMBER OF DIGITS TO TYPE
.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

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

$TYPDS: MOV R0, -(SP) ;; PUSH R0 ON STACK
        MOV R1, -(SP) ;; PUSH R1 ON STACK
        MOV R2, -(SP) ;; PUSH R2 ON STACK
        MOV R3, -(SP) ;; PUSH R3 ON STACK
        MOV R5, -(SP) ;; PUSH R5 ON STACK
        MOV #20200, -(SP) ;; SET BLANK SWITCH AND SIGN
        MOV 20(SP), R5 ;; GET THE INPUT NUMBER
        BPL 1$ ;; BR IF INPUT IS POS.
        NEG R5 ;; MAKE THE BINARY NUMBER POS.
        MOV #'-, 1(SP) ;; MAKE THE ASCII NUMBER NEG.
1$: CLR R0 ;; ZERO THE CONSTANTS INDEX
   MOV #SDBLK, R3 ;; SETUP THE OUTPUT POINTER
   MOV #', (R3)+ ;; SET THE FIRST CHARACTER TO A BLANK
2$: CLR R2 ;; CLEAR THE BCD NUMBER
   MOV $DTBL(R0), R1 ;; GET THE CONSTANT
3$: SUB R1, R5 ;; FORM THIS BCD DIGIT
   BLT 4$ ;; BR IF DONE
   INC R2 ;; INCREASE THE BCD DIGIT BY 1
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

```

```

6234 042264 106316
6235 042266 103003
6236 042270 116663 000001 177777
6237 042276 052702 000060
6238 042302 052702 000040
6239 042306 110223
6240 042310 005720
6241 042312 020027 000010
6242 042316 002746
6243 042320 003002
6244 042322 010502
6245 042324 000764
6246 042326 105726
6247 042330 100003
6248 042332 116663 177777 177776
6249 042340 105013
6250 042342 012605
6251 042344 012603
6252 042346 012602
6253 042350 012601
6254 042352 012600
6255 042354 104401 042402
6256 042360 016666 000002 000004
6257 042366 012616
6258 042370 000002
6259 042372 023420
6260 042374 001750
6261 042376 000144
6262 042400 000012
6263 042402 000004
6264
6265
6266
6267
6268
6269
6270
6271
6272 042412 010046
6273 042414 016600 000002
6274 042420 005740
6275 042422 111000
6276 042424 006300
6277 042426 016000 042446
6278 042432 000200
6279
6280
6281
6282
6283 042434 011646
6284 042436 016666 000004 000002
6285 042444 000002
6286
6287
6288
6289

```

```

5$: ASLB (SP) ; MSD?
BCC 6$ ; BR IF NO
MOV 1(SP),-1(R3) ; YES--SET THE SIGN
6$: BIS #'0,R2 ; MAKE THE BCD DIGIT ASCII
7$: BIS #' ,R2 ; MAKE IT A SPACE IF NOT ALREADY A DIGIT
MOV R2,(R3)+ ; PUT THIS CHARACTER IN THE OUTPUT BUFFER
TST (R0)+ ; JUST INCREMENTING
CMP RO,#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
9$: 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
MOV (SP)+,R0 ; POP STACK INTO R0
TYPE $DBLK ; NOW TYPE THE NUMBER
MOV 2(SP),4(SP) ; ADJUST THE STACK
MOV (SP)+,(SP)
RTI ; RETURN TO USER

$DTBL: 10000.
1000.
100.
10.

$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.

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

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

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

.SBTTL TRAP TABLE
;THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED

```



```

6290
6291
6292
6293
6294 042446 042434
6295 042450 041210
6296 042452 041764
6297 042454 041740
6298 042456 042000
6299 042460 042166
6300
6301
6302 042462 041114
6303 042464 041152
6304 042466 044172
6305 042470 044220
6306
6307
6308
6309
6310
6311 042472 012737 042664 000024
6312 042500 012737 000340 000026
6313 042506 010046
6314 042510 010146
6315 042512 010246
6316 042514 010346
6317 042516 010446
6318 042520 010546
6319 042522 017746 136412
6320 042526 010637 042670
6321 042532 012737 042544 000024
6322 042540 000000
6323 042542 000776
6324
6325
6326
6327 042544 012737 042664 000024
6328 042552 013706 042670
6329 042556 005037 042670
6330 042562 005237 042670
6331 042566 001375
6332 042570 105737 001446
6333 042574 001003
6334 042576 011600
6335 042600 076600
6336 042602 000226
6337 042604
6338 042604 012677 136330
6339 042610 012605
6340 042612 012604
6341 042614 012603
6342 042616 012602
6343 042620 012601
6344 042622 012600
6345 042624 012737 042472 000024

```

; *BY THE "TRAP" INSTRUCTION.

```

; ROUTINE
;-----
$TRPAD: .WORD $STRAP2
$TYPE ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
$TYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
$TYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
$TYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
$TYPDS ;;CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)

$SAVREG ;;CALL=SAVREG TRAP+6(104406) SAVE R0-R5 ROUTINE
$RESREG ;;CALL=RESREG TRAP+7(104407) RESTORE R0-R5 ROUTINE
TBITOFF ;;CALL=TBITO TRAP+10(104410) TURN OFF T-BIT TRAPPING
TBITRESTORE ;;CALL=TBITR TRAP+11(104411) RETURN T-BIT TO ITS PREVIOUS CON

```

.SBTTL POWER DOWN AND UP ROUTINES

```

;POWER DOWN ROUTINE
$PWRDN: MOV $SILLUP, @#PWRVEC ;;SET FOR FAST UP
MOV #340, @#PWRVEC+2 ;;PRIO:7
MOV R0, -(SP) ;;PUSH R0 ON STACK
MOV R1, -(SP) ;;PUSH R1 ON STACK
MOV R2, -(SP) ;;PUSH R2 ON STACK
MOV R3, -(SP) ;;PUSH R3 ON STACK
MOV R4, -(SP) ;;PUSH R4 ON STACK
MOV R5, -(SP) ;;PUSH R5 ON STACK
MOV @SWR, -(SP) ;;PUSH @SWR ON STACK
MOV SP, $SAVR6 ;;SAVE SP
MOV $PWRUP, @#PWRVEC ;;SET UP VECTOR
HALT
BR .-2 ;;HANG UP

```

```

;POWER UP ROUTINE
$PWRUP: MOV $SILLUP, @#PWRVEC ;;SET FOR FAST DOWN
MOV $SAVR6, SP ;;GET SP
CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
1$: INC $SAVR6 ;;WAIT FOR THE INC
BNE 1$ ;;OF WORD
TSTB FORTY ;;EXECUTING ON AN 11/40?
BNE 2$ ;;BRANCH IF YES
MOV (SP), R0 ;;GET OLD SW. REG. CONTENTS
MED ;;WRITE THE CONSOLE SW. REG.
WCNSSW ;;WITH ITS PREVIOUS VALUE
2$: MOV (SP)+, @SWR ;;POP STACK INTO @SWR
MOV (SP)+, R5 ;;POP STACK INTO R5
MOV (SP)+, R4 ;;POP STACK INTO R4
MOV (SP)+, R3 ;;POP STACK INTO R3
MOV (SP)+, R2 ;;POP STACK INTO R2
MOV (SP)+, R1 ;;POP STACK INTO R1
MOV (SP)+, R0 ;;POP STACK INTO R0
MOV $PWRDN, @#PWRVEC ;;SET UP THE POWER DOWN VECTOR

```

```

6346 042632 012737 000340 000026
6347 042640 104401
6348 042642 042672
6349 042644 012716
6350 042646 020066
6351 042650 042766 000020 000002
6352 042656 005037 041106
6353 042662 000002
6354 042664 000000
6355 042666 000776
6356 042670 000000
6357 042672 006412 047520 042527
6358 042700 020122 040506 046111
6359 042706 051125 026105 051040
6360 042714 051505 040524 052122
6361 042722 047111 020107 051120
6362 042730 043517 040522 000115
6363
6364
6365
6366
6367
6368
6369
6370
6371
6372
6373
6374
6375
6376 042736 104406
6377 042740 016601 000002
6378 042744 012705 043055
6379 042750 012704 000014
6380 042754 012703 177770
6381 042760 012100
6382 042762 012101
6383 042764 005002
6384 042766 110245
6385 042770 010002
6386 042772 005304
6387 042774 003007
6388 042776 001405
6389 043000 005205
6390 043002 010566 000002
6391 043006 104407
6392 043010 000207
6393 043012 006203
6394 043014 006001
6395 043016 006000
6396 043020 006001
6397 043022 006000
6398 043024 006001
6399 043026 006000
6400 043030 040302
6401 043032 062702 000060

```

```

MOV #340,2#PWRVEC+2 ;;PRIO:7
TYPE ;;REPORT THE POWER FAILURE
SPWRMG: .WORD PWRMSG ;;POWER FAIL MESSAGE POINTER
MOV (PC)+,(SP) ;;RESTART AT START
SPWRAD: .WORD START ;;RESTART ADDRESS
BIC #20,2(SP) ;;CLEAR "T" BIT
CLR $TBIT ;;CLEAR THE "T" BIT FLAG
RTI
$ILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
BR -.2 ;;BEFORE THE POWER DOWN WAS COMPLETE
$SAVR6: 0 ;;PUT THE SP HERE
PWRMSG: .ASCIZ <12><15>?POWER FAILURE, RESTARTING PROGRAM?

.EVEN

.SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

;*****
;THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
;UNSIGNED OCTAL ASCIZ NUMBER.
;CALL
;* MOV #PNTR,-(SP) ;; POINTER TO LOW WORD OF BINARY NUMBER
;* JSR PC,2#$DB20 ;; CALL THE ROUTINE
;* RETURN ;; THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACK

$DB20: SAVREG ;;SAVE ALL REGISTERS
MOV 2(SP),R1 ;;PICKUP THE POINTER TO LOW WORD
MOV #SOCTVL+13.,R5 ;; POINTER TO DATA TABLE
MOV #12.,R4 ;;DO ELEVEN CHARACTERS
MOV #C7,R3 ;;MASK
MOV (R1)+,R0 ;;LOWER WORD
MOV (R1)+,R1 ;;HIGH WORD
CLR R2 ;;TERMINATOR
1$: MOVB 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) ;;ASCIZ 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
ROR R1
ROR R0
BIC R3,R2 ;;MASK OUT ALL JUNK
ADD #0,R2 ;;MAKE THIS CHAR. ASCII

```

F10

CQKTA-BO PDP 11/6X MEM. MGMT. DIAG.
CQKTAB.P11 30-DEC-77 14:49

MACY11 30A(1052) 03-JAN-78 08:09 PAGE 122
DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

SEQ 0122

```

6402 043036 000753
6403 043040 000016
6404
6405
6406
6407
6408
6409
6410
6411
6412
6413
6414
6415
6416
6417
6418
6419 043056
6420 043056 005037 001442
6421 043062 005037 001430
6422 043066 032777 040000 136044
6423 043074 001117
6424
6425 043076 000416
6426
6427 043100 013746 000004
6428 043104 012737 043124 000004
6429 043112 005737 177060
6430 043116 012637 000004
6431 043122 000466
6432 043124 022626
6433 043126 012637 000004
6434 043132 000426
6435 043134
6436 043134 032777 000400 135776
6437 043142 001407
6438 043144 017746 135770
6439 043150 042716 000200
6440 043154 122637 001102
6441 043160 001465
6442 043162 105737 001103
6443 043166 001421
6444 043170 123737 001115 001103
6445 043176 101015
6446 043200 032777 001000 135732
6447 043206 001404
6448 043210 013737 001110 001106
6449 043216 000446
6450 043220 105037 001103
6451 043224 005037 001212
6452 043230 000415
6453 043232 032777 004000 135700
6454 043240 001011
6455 043242 005737 001234
6456 043246 001406
6457 043250 005237 001104
    
```

```

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

$SCOPE:
CLR RETRY ;CLEAR THE RETRY FLAG BEFORE THIS TEST
CLR ERRCNT ;CLEAR THE MULTIPLE ERROR COUNTER
1$: BIT #BIT14,$SWR ;LOOP ON PRESENT TEST?
BNE $OVER ;YES IF SW14=1
;*****START OF CODE FOR THE XOR TESTER*****
$XTSTR: BR 6$ ;IF RUNNING ON THE "XOR" TESTER CHANGE
;THIS INSTRUCTION TO A "NOP" (NOP=240)
MOV @#ERRVEC,-(SP) ;SAVE THE CONTENTS OF THE ERROR VECTOR
MOV #5,$@#ERRVEC ;SET FOR TIMEOUT
TST @#177060 ;TIME OUT ON XOR?
MOV (SP)+,@#ERRVEC ;RESTORE THE ERROR VECTOR
BR $SVLAD ;GO TO THE NEXT TEST
5$: CMP (SP)+,(SP)+ ;CLEAR THE STACK AFTER A TIME OUT
MOV (SP)+,@#ERRVEC ;RESTORE THE ERROR VECTOR
BR 7$ ;LOOP ON THE PRESENT TEST
6$; ;*****END OF CODE FOR THE XOR TESTER*****
BIT #BIT08,$SWR ;LOOP ON SPEC. TEST?
BEQ 2$ ;BR IF NO
MOV @SWR,-(SP) ;SET DESIRED TEST NUM. FROM SWR
BIC #$$SWRMK,(SP) ;STRIP AWAY UNDESIRED BITS
CMPB (SP)+,$STNM ;ON THE RIGHT TEST?
BEQ $OVER ;BR IF YES
2$: TSTB $ERFLG ;HAS AN ERROR OCCURRED?
BEQ 3$ ;BR IF NO
CMPB $ERMAX,$ERFLG ;MAX. ERRORS FOR THIS TEST OCCURRED?
BHI 3$ ;BR IF NO
BIT #BIT09,$SWR ;LOOP ON ERROR?
BEQ 4$ ;BR IF NO
7$: MOV $LPERR,$LPADR ;SET LOOP ADDRESS TO LAST SCOPE
BR $OVER
4$: CLRB $ERFLG ;ZERO THE ERROR FLAG
CLR $TIMES ;CLEAR THE NUMBER OF ITERATIONS TO MAKE
BR 1$ ;ESCAPE TO THE NEXT TEST
3$: BIT #BIT11,$SWR ;INHIBIT ITERATIONS?
BNE 1$ ;BR IF YES
TST $PASS ;IF FIRST PASS OF PROGRAM
BEQ 1$ ;INHIBIT ITERATIONS
INC $ICNT ;INCREMENT ITERATION COUNT
    
```

```

6458 043254 023737 001212 001104      CMP      $TIMES,$ICNT      ;; CHECK THE NUMBER OF ITERATIONS MADE
6459 043262 002024                      BGE      $OVER            ;; BR IF MORE ITERATION REQUIRED
6460 043264 012737 000001 001104 1$:      MOV      #1,$ICNT         ;; REINITIALIZE THE ITERATION COUNTER
6461 043272 013737 043350 001212      MOV      $MXCNT,$TIMES    ;; SET NUMBER OF ITERATIONS TO DO
6462 043300 105237 001102                      $SVLAD: INCB     $STNM      ;; COUNT TEST NUMBERS
6463 043304 113737 001102 001232      MOV      $STNM,$TESTN     ;; SET TEST NUMBER IN APT MAILBOX
6464 043312 011637 001106                      MOV      (SP),$LPADR      ;; SAVE SCOPE LOOP ADDRESS
6465 043316 011637 001110                      MOV      (SP),$LPERR     ;; SAVE ERROR LOOP ADDRESS
6466 043322 005037 001214                      CLR      $ESCAPE         ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
6467 043326 112737 000001 001115      MOV      #1,$ERMAX       ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
6468 043334 013777 001102 135600 $OVER:  MOV      $STNM,$DISPLAY ;; DISPLAY TEST NUMBER
6469 043342 013716 001106                      MOV      $LPADR,(SP)     ;; FUDGE RETURN ADDRESS
6470 043346 000002                      RTI                       ;; FIXES PS
6471 043350 000310 $MXCNT: 200.                ;; MAX. NUMBER OF ITERATIONS
6472                      .SBTTL  ERROR HANDLER ROUTINE
6473
6474                      ;; *****
6475                      ;; THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
6476                      ;; SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
6477                      ;; AND GO TO ERTYPE ON ERROR
6478                      ;; THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
6479                      ;; *SW15=1      HALT ON ERROR
6480                      ;; *SW13=1      INHIBIT ERROR TYPEOUTS
6481                      ;; *SW10=1     BELL ON ERROR
6482                      ;; *SW09=1     LOOP ON ERROR
6483                      ;; *CALL
6484                      ;; *      ERROR      N      ;; ERROR=EMT AND N=ERROR ITEM NUMBER
6485
6486 043352                      $ERROR:
6487 043352 113737 001102 001412      MOV      $STNM,TESTNO    ;; SAVE TEST NUMBER FOR ERROR TYPE OUT
6488 043360 005237 001430                      INC      ERRCNT          ;; KEEP COUNT OF MULTIPLE ERRORS
6489 043364 010037 001162                      MOV      R0,$REG0        ;; SAVE R0
6490 043370 010137 001164                      MOV      R1,$REG1        ;; SAVE R1
6491 043374 010237 001166                      MOV      R2,$REG2        ;; SAVE R2
6492 043400 010337 001170                      MOV      R3,$REG3        ;; SAVE R3
6493 043404 010437 001172                      MOV      R4,$REG4        ;; SAVE R4
6494 043410 010537 001174                      MOV      R5,$REG5        ;; SAVE R5
6495 043414 105237 001103 7$:      INCB     $ERFLG         ;; SET THE ERROR FLAG
6496 043420 001775                      BEQ      7$              ;; DON'T LET THE FLAG GO TO ZERO
6497 043422 013777 001102 135512      MOV      $STNM,$DISPLAY  ;; DISPLAY TEST NUMBER AND ERROR FLAG
6498 043430 032777 002000 135502      BIT      #BIT10,$SWR     ;; BELL ON ERROR?
6499 043436 001402                      BEQ      1$              ;; NO - SKIP
6500 043440 104401                      TYPE     $BELL           ;; RING BELL
6501 043444 005237 001112 1$:      INC      $ERTTL         ;; COUNT THE NUMBER OF ERRORS
6502 043450 011637 001116                      MOV      (SP),$ERRPC     ;; GET ADDRESS OF ERROR INSTRUCTION
6503 043454 162737 000002 001116      SUB      #2,$ERRPC
6504 043462 117737 135430 001114      MOV      $ERRPC,$ITEMB  ;; STRIP AND SAVE THE ERROR ITEM CODE
6505 043470 032777 020000 135442      BIT      #BIT13,$SWR    ;; SKIP TYPEOUT IF SET
6506 043476 001004                      BNE     20$             ;; SKIP TYPEOUTS
6507 043500 004737 043650                      JSR     PC,ERTYPE       ;; GO TO USER ERROR ROUTINE
6508 043504 104401 001223                      TYPE     , $CRLF
6509 043510
6510 043510 122737 000001 001246 20$:  CMP      #APTENV,$ENV    ;; RUNNING IN APT MODE
6511 043516 001007                      BNE     2$              ;; NO, SKIP APT ERROR REPORT
6512 043520 113737 001114 043532      MOV      $ITEMB,21$     ;; SET ITEM NUMBER AS ERROR NUMBER
6513 043526 004737 041510                      JSR     PC,$ATY4        ;; REPORT FATAL ERROR TO APT

```

```

6514 043532 000
6515 043533 000
6516 043534 000777
6517 043536 005777 135376
6518 043542 100001
6519 043544 000000
6520 043546 032777 001000 135364
6521 043554 001402
6522 043556 013716 001110
6523 043562 005737 001214
6524 043566 001402
6525 043570 013716 001214
6526 043574
6527 043574 022737 041034 000042
6528 043602 001001
6529 043604 000000
6530 043606
6531 043606 032737 001000 001140
6532 043614 001414
6533 043616 042737 177776 177572
6534
6535 043624 012737 177777 015162
6536 043632 012737 177777 015252
6537 043640 012737 177777 015432
6538 043646 000002
6539
6540
6541
6542
6543
6544
6545
6546
6547
6548
6549
6550
6551
6552
6553
6554
6555
6556
6557
6558
6559
6560 043650 010046
6561 043652 005000
6562 043654 113700 001114
6563 043660 001004
6564 043662 013746 001116
6565 043666 104402
6566 043670 000534
6567 043672 104401 001223
6568 043676 005300
6569 043700 006300

```

```

21$: .BYTE 0
      .BYTE 0
22$: BR 22$ ;: APT ERROR LOOP
2$: TST 2$SWR ;: HALT ON ERROR
      BPL 3$ ;: SKIP IF CONTINUE
      HALT ;: HALT ON ERROR!
3$: BIT #BIT09,2$SWR ;: LOOP ON ERROR SWITCH SET?
      BEQ 4$ ;: BR IF NO
      MOV $LPERR,(SP) ;: FUDGE RETURN FOR LOOPING
4$: TST $ESCAPE ;: CHECK FOR AN ESCAPE ADDRESS
      BEQ 5$ ;: BR IF NONE
      MOV $ESCAPE,(SP) ;: FUDGE RETURN ADDRESS FOR ESCAPE
5$: CMP #2ENDAD,2#42 ;: ACT-11 AUTO-ACCEPT?
      BNE 6$ ;: BRANCH IF NO
      HALT ;: YES
6$: BIT #SW9,SWR ;: IS THE LOOP ON ERROR SWITCH UP?
      BEQ EREXIT ;: BRANCH IF NOT LOOPING ON ERROR
      BIC #177776,MMRO ;: CLEAR MEMORY MANAGEMENT REG 0
      MOV #-1,CPFLAG ;: RE-INITIALIZE CP TRAP FLAG
      MOV #-1,PAFLAG ;: RE-INITIALIZE PARITY TRAP FLAG
      MOV #-1,MMFLAG ;: RE-INITIALIZE MEMORY MANAGE. TRAP FLAG
EREXIT: RTI ;: RETURN TO TEST AFTER ERROR

```

.SBTTL ERROR MESSAGE TYPE OUT ROUTINE

```

*: THIS SUBROUTINE IS CALLED BY THE ERROR HANDLER TO TYPE
*: THE ERROR MESSAGES. IT PICKS UP THE ITEM BYTE ($ITEMB) NUMBER
*: AND USES THAT TO INDEX THROUGH THE ERROR TABLE. THE ERROR
*: TABLE STARTS AT "$ERRTB" AND HAS FOUR (4) POINTERS FOR EACH
*: ENTRY. 'EM' 'DH' 'DT' 'DF'. THE 'EM' POINTS TO THE ERROR
*: MESSAGE WHICH IS AN ASCIZ STRING. THE 'DH' POINTS TO THE DATA
*: HEADER WHICH IS ANOTHER ASCIZ STRING. THE 'DT' POINTS TO THE
*: DATA TABLE WHICH IS A GROUP OF WORDS CONTAINING THE ADDRESSES
*: OF THE DATA TO BE TYPED. THE FORMAT OF THIS DATA IS
*: CONTROLLED BY THE 'DF' WHICH IS THE POINTER TO THE DATA FORMAT.
*: THE DATA FORMAT IS A GROUP OF BYTES WHICH CONTAIN NUMBERS
*: THAT CORRESPOND TO DIFFERENT TYPING FORMATS.
*: 0 -16 BIT OCTAL FORMAT
*: 1 -DECIMAL FORMAT
*: 2 -18 BIT OCTAL FORMAT. DATA IS A P.A.R. AND THE
*: OUTPUT IS THE BASE ADDRESS THAT THE P.A.R. POINTS TO.
ERTYPE: MOV RO,-(KSP) ;: SAVE RO ON STACK
      CLR RO ;: CLEAR RO
      MOVB 2#$ITEMB,RO ;: PUT ITEM NUMBER IN RO
      BNE 1$ ;: BRANCH IF IT IS NON-ZERO
      MOV $ERRPC,-(KSP) ;: PUT ERROR PC ON STACK FOR TYPING
      TYPOC ;: TYPE FAILING PC
      BR 13$ ;: GO TO RETURN
1$: TYPE $CRLF ;: TYPE CRLF
      DEC RO ;: ADJUST ITEM NUMBER TO BE A POINTER
      ASL RO ;: LEFT SHIFT ITEM NO. 3 PLACES

```

```

6570 043702 006300 ASL R0
6571 043704 006300 ASL R0
6572 043706 100024 BPL 22$ ; BRANCH IF ITEM #LESS THAN 200
6573 043710 032700 001000 BIT #BIT9,R0 ; SEE IF ITEM # WAS 3XX
6574 043714 001415 BEQ 21$ ; BRANCH IF ITEM # WAS 2XX
6575 ; AND TYPE ERROR MESSAGE
6576 043716 032737 000200 001140 BIT #SW7,@#SWR ; SEE IF SWITCH 7 IS UP
6577 043724 001404 BEQ 20$ ; BRANCH IF SWITCH IS NOT UP
6578 ; AND TYPE DATA ON MULTIPLE ERRORS
6579 043726 062766 000004 000002 ADD #4,2(KSP) ; SKIP 'TYPE $CRLF' IF SW 7 IS UP
6580 ; INHIBIT MULTIPLE ERROR TYPEOUTS
6581 043734 000512 BR 13$ ; BRANCH TO EXIT
6582 043736 20$: BIC #177000,R0 ; CLEAR UPPER BYTE OF R0
6583 043736 042700 177000 ADD #ER200+4,R0 ; POINT TO DATA TABLE ENTRY
6584 043742 062700 002472 BR 5$ ; GO TYPE DATA TABLE
6585 043746 000424 BR 5$ ; GO TYPE DATA TABLE
6586 043750 042700 177000 21$: BIC #177000,R0 ; CLEAR UPPER BYTE OF R0
6587 043754 062700 000770 ADD #<ER200-$ERRTB>,R0 ; ADD DIFFERENCE BETWEEN
6588 ; ITEM 1 AND ITEM 201
6589 ; GET POINTER TO ERROR MESSAGE AND TYPE IT
6590 ; IF THE POINTER IS NOT ZERO
6591 043760 062700 001476 22$: ADD #$ERRTB,R0 ; ADD BASE OF ERROR TABLE
6592 043764 012037 043774 MOV (R0)+,2$ ; PUT MESSAGE POINTER IN TYPE STATEMENT
6593 043770 001404 BEQ 3$ ; BRANCH IF NO ERROR MESSAGE
6594 043772 104401 TYPE ; TYPE ERROR MESSAGE
6595 043774 000000 2$: .WORD 0 ; POINTER TO ERROR MESSAGE
6596 043776 104401 001223 TYPE , $CRLF ; TYPE CRLF
6597 ; GET THE POINTER TO THE DATA HEADER AND
6598 ; TYPE IT IF THE POINTER IS NOT ZERO
6599 044002 012037 044012 3$: MOV (R0)+,4$ ; PUT HEADER POINTER IN TYPE STATEMENT
6600 044006 001404 BEQ 5$ ; BRANCH IF NO DATA HEADER
6601 044010 104401 TYPE ; TYPE THE DATA HEADER
6602 044012 000000 4$: .WORD 0 ; POINTER TO DATA HEADER
6603 044014 104401 001223 TYPE , $CRLF ; TYPE CRLF
6604 ; THIS IS THE START OF THE DATA OUTPUT IF THE
6605 ; DATA POINTER IS NOT ZERO. R0 POINTS TO THE
6606 ; DATA FORMAT. R1 POINTS TO THE ADDRESS OF
6607 ; THE DATA WORDS.
6608 044020 010146 5$: MOV R1,-(KSP) ; SAVE R1 ON THE STACK
6609 044022 012001 MOV (R0)+,R1 ; PUT DATA TABLE POINTER IN R1
6610 044024 001455 BEQ 12$ ; BRANCH IF NO DATA TABLE
6611 044026 012000 MOV (R0)+,R0 ; PICK UP DATA FORMAT POINTER
6612 044030 105710 6$: TSTB (R0) ; IS THIS WORD OCTAL
6613 044032 001003 BNE 7$ ; BRANCH IF NOT 16-BIT OCTAL
6614 ; THIS IS 16 BIT OCTAL FORMAT (DF = 0)
6615 044034 013146 MOV @ (R1)+,-(KSP) ; PUT WORD ON STACK FOR TYPING
6616 044036 104402 TYPOC ; TYPE THE WORD ON STACK AS 16 BIT OCTAL
6617 044040 000441 BR 11$ ; GET READY FOR NEXT WORD
6618 044042 122710 7$: CMPB #1,(R0) ; IS THE WORD DECIMAL
6619 044046 001003 BNE 10$ ; BRANCH IF NOT DECIMAL
6620 ; THIS IS DECIMAL FORMAT (DF = 1)
6621 044050 013146 MOV @ (R1)+,-(KSP) ; PUT WORD ON STACK FOR TYPING
6622 044052 104405 TYPDS ; TYPE THE WORD ON STACK AS DECIMAL
6623 044054 000433 BR 11$ ; GET READY FOR NEXT WORD
6624 ; THIS IS FORMAT 2. DATA WORD IS A P.A.R.
6625 ; OUTPUT WILL BE 18-BIT. PAR LEFT SHIFTED 2.

```

```

6626 044056 010246
6627 044060 010346
6628 044062 013103
6629 044064 005002
6630 044066 012705 000006
6631 044072 000241
6632 044074 006303
6633 044076 006102
6634 044100 077504
6635 044102 010237 001372
6636 044106 010337 001370
6637 044112 012746 001370
6638 044116 004737 042736
6639 044122 062716 000005
6640 044126 012637 044134
6641 044132 104401
6642 044134 000000
6643 044136 012603
6644 044140 012602
6645 044142 000400
6646 044144 005200
6647 044146 104401 044166
6648 044152 005711
6649 044154 001401
6650 044156 000724
6651 044160 012601
6652 044162 012600
6653 044164 000207
6654 044166 020040 000
6655 044172
6656
6657
6658
6659
6660
6661
6662
6663
6664
6665
6666
6667
6668
6669
6670
6671
6672
6673
6674
6675
6676 044172
6677 044172 032766 000020 000002
6678 044200 001406
6679 044202 016637 000002 001440
6680 044210 042766 000020 000002
6681 044216 000006

```

```

10$: MOV R2,-(KSP) ;SAVE R2 ON THE STACK
      MOV R3,-(KSP) ;SAVE R3 ON THE STACK
      MOV 2(R1)+,P3 ;LOAD DATA WORD INTO R3
      CLR R2 ;R2 WILL HOLD UPPER SIX BITS OF NUMBER
      MOV #6,R5 ;LEFT SHIFT <R2:R3> 6 PLACES
23$: CLC
      ASL R3
      ROL R2
      SOB R5,23$
      MOV R2,PADRSR ;STORE UPPER BITS OF ADDRESS
      MOV R3,PADRSL ;STORE LOWER 16 BITS OF ADDRESS
      MOV #PADRSL,-(KSP) ;PUT ADDRESS OF LOWER 16 BITS ON STACK
      JSR PC,$DB20 ;CONVERT TWO WORDS TO ASCIZ
      ADD #5,(KSP) ;I ONLY WANT 6 DIGITS
      MOV (KSP)+,31$ ;PUT POINTER AFTER TYPE CALL
      TYPE ;POINTER TO ASCIZ STRING FOLLOWS
31$: .WORD 0 ;POINTER TO ASCIZ STRING
      MOV (KSP)+,R3 ;RESTORE R3 FROM STACK
      MOV (KSP)+,R2 ;RESTORE R2 FROM STACK
      BR 11$ ;GET READY FOR NEXT WORD
11$: INC R0 ;POINT TO NEXT FORMAT BYTE
      TYPE ,32$ ;TYPE TWO SPACES
      TST (R1) ;IS THERE ANOTHER WORD?
      BEQ 12$ ;BRANCH IF ALL DONE
      BR 6$ ;GO BACK FOR NEXT NUMBER
12$: MOV (KSP)+,R1 ;RESTORE R1
13$: MOV (KSP)+,R0 ;RESTORE R0
      RTS PC ;RETURN TO ERROR ROUTINE
32$: .ASCIZ ? ? ;TWO SPACES
      .EVEN

```

.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 1$ ;BRANCH IF ITS NOT
          MOV 2(KSP),OLDPSW ;SAVE OLD PSW FOR RESTORING LATER
          BIC #TBIT,2(KSP) ;CLEAR THE T-BIT
1$: RTT ;RETURN TO THE PROGRAM

```

6682
6683
6684
6685
6686
6687
6688
6689
6690
6691
6692
6693
6694
6695
6696
6697
6698
6699
6700
6701
6702
6703
6704
6705
6706
6707
6708
6709
6710
6711
6712
6713
6714
6715
6716
6717
6718
6719
6720
6721
6722
6723
6724
6725
6726
6727
6728
6729
6730
6731
6732
6733
6734
6735
6736
6737

044220
044220 013766 001440 000002
044226 042737 000020 001440

044234 000006

044236 010046
044240 012700 000010
044244 005025
044246 077002
044250 012600
044252 000207

044254
044254 005037 001414
044260 005037 001424
044264 005037 001420
044270 012700 177777
044274 010037 001416

```
.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
;*****
SUBROUTINE CLRREG
;*
;* THIS SUBROUTINE CLEARS 8 CONSECUTIVE MEMORY MANAGEMENT
;* REGISTERS STARTING WITH THE REGISTER POINTED TO BY R5.
;* IT SAVES R0 ON THE STACK AND LOADS A COUNT IN R0,
;* CLEARS THE PAR'S OR PDR'S, AND THEN RESTORES R0 BEFORE
;* RETURNING.
;* THE CALLING SEQUENCE IS:
;* MOV #KIPAR,R5 ;PUT ADDRESS OF FIRST KERNEL PAR IN R5
;* JSR PC,CLRREG ;GO CLEAR ALL KERNEL PAR'S
;*
;*****
CLRREG: MOV R0,-(KSP) ;SAVE R0 ON STACK
MOV #10,R0 ;PUT COUNT IN R0
1$: CLR (R5)+ ;CLEAR PAR ORPDR POINTED TO BY R5
SOB R0,1$ ;BRANCH BACK 7 DECIMAL TIMES.
MOV (KSP)+,R0 ;RESTORE R0 FROM STACK
RTS PC ;RETURN TO TEST

.SBTTL CLEANUP LOCATIONS THAT HOLD LOGICAL 'AND' AND 'OR'
;*****
SUBROUTINE CLEANUP
;*
;* THIS SUBROUTINE IS USED TO INITIALIZE ALL THE LOCATIONS THAT
;* HOLD THE "LOGICAL AND" AND "LOGICAL OR" OF THE DATA AND ADDRESSES
;* THAT FAILED DURING THE EXECUTION OF A TEST.
;*
;*****
CLEANUP: ;STARTING ADDRESS OF SUBROUTINE.
CLR DATAOR ;LOCATION FOR LOGICAL OR OF BAD DATA
CLR ADDROR ;LOCATION FOR LOGICAL OR OF ADDRESS
CLR PATTOR ;LOCATION FOR LOGICAL OR OF PATTERN LOADED
MOV #-1,R0 ;LOAD -1 INTO R0 TO INITIALIZE LOGICAL AND LOCS
MOV R0,DATAND ;LOCATION FOR LOGICAL AND OF BAD DATA
```



```

6738 044300 010037 001426
6739 044304 010037 001422
6740 044310 000207
6741
6742
6743
6744
6745
6746
6747
6748
6749
6750 044312
6751 044312 005227
6752 044314 177777
6753 044316 001401
6754 044320 000000
6755
6756
6757
6758
6759 044322 012637 001434
6760 044326 012637 001436
6761 044332 012706 001100
6762 044336 050037 001424
6763 044342 005100
6764 044344 040037 001426
6765 044350 005100
6766 044352 105737 001103
6767 044356 001002
6768 044360 104201
6769 044362 000401
6770 044364 104301
6771 044366 012737 177777 044314
6772 044374 013746 001436
6773 044400 013746 001434
6774 044404 000006
6775
6776
6777
6778
6779
6780
6781
6782
6783
6784
6785
6786 044406
6787 044406 012637 001434
6788 044412 050037 001424
6789 044416 005100
6790 044420 040037 001426
6791 044424 005100
6792 044426 050137 001414
6793 044432 005101

```

```

MOV RO,ADRAND ;LOCATION FOR LOGICAL AND OF ADDRESS
MOV RO,PATAND ;LOCATION FOR LOGICAL AND OF PATTERN LOADED
RTS PC ;RETURN TO TEST

.SBTTL P.A.R. OR P.D.R. ADDRESS TIMED OUT WHEN REFERENCED
*****
*
* THIS SUBROUTINE IS USED TO LOG AND REPORT THE FACT THAT A
* REFERENCE TO A P.A.R. OR A P.D.R. TIMED OUT ON THE UNIBUS. IT
* KEEPS A LOGICAL AND AND A LOGICAL OR OF EACH ADDRESS THAT TIMES
* OUT.
*****
TIMEOUT: ;STARTING ADDRESS OF SUBROUTINE
INC (PC)+ ;INCREMENT ONE TIME GATE
TOFLAG: .WORD -1 ;ONE TIME ENTRANCE FLAG
BEQ 10$ ;BRANCH IF FLAG IS NOW ZERO
HALT ;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 .
10$: MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS
MOV (KSP)+,OLDPS ;SAVE OLD PSW
MOV #KERSTK,KSP ;RESTORE STACK POINTER
1$: BIS RO,ADDROR ;PERFORM LOGICAL OR OF FAILING ADDRESS
COM RO ;GET RO READY FOR AND
BIC RO,ADRAND ;PERFORM LOGICAL AND
COM RO ;PUT RO BACK AS IT WAS
TSTB $ERFLG ;IS THIS THE FIRST ERROR
BNE 2$ ;BRANCH IF NOT FIRST ERROR
ERROR 201 ;NO REGISTER RESPONSE.
BR 3$ ;BRANCH TO EXIT
2$: ERROR 301 ;CONTINUE NO RESPONSE TABLE
3$: MOV #-1,TOFLAG ;RESET ONE TIME GATE
MOV OLDPS,-(KSP) ;RESTORE OLD PSW
MOV OLDPC,-(KSP) ;PUSH RETURN ADDRESS BACK ON THE STACK
RTT ;RETURN TO THE TEST

.SBTTL DUAL ADDRESSING WHEN LOADING A P.A.R. OR P.D.R.
*****
*
* THIS SUBROUTINE WILL LOG AND REPORT ALL DUAL ADDRESSING ERRORS
* FOUND IN BOTH P.A.R.'S AND P.D.R.'S. A "LOGICAL OR" AND A
* "LOGICAL AND" OF THE WRITTEN ADDRESSES WILL BE MAINTAINED IN
* 'ADDROR' AND 'ADRAND'. THE LOG ON THE ADDITIONAL OR FAILING,
* ADDRESSES WILL BE MAINTAINED IN 'DATAOR' AND 'DATAND'.
*****
DUALADR: ;STARTING LOCATION OF SUBROUTINE
MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF LOOP
BIS RO,ADDROR ;LOGICAL OR OF WRITTEN ADDRESS
COM RO ;GET RO READY FOR AND
BIC RO,ADRAND ;PERFORM LOGICAL AND
COM RO ;PUT RO BACK AS IT WAS
BIS R1,DATAOR ;LOGICAL OR OF DUALED ADDRESS
COM R1 ;GET R1 READY FOR AND

```

6794	044434	040137	001416
6795	044440	005101	
6796	044442	105737	001103
6797	044446	001002	
6798	044450	104202	
6799	044452	000401	
6800	044454	104302	
6801	044456	000177	134752
6802			
6803			
6804			
6805			
6806			
6807			

BIC	R1,DATAND		;PERFORM LOGICAL AND
COM	R1		;PUT R1 BACK AS IT WAS
TSTB	\$ERFLG		;SEE IF THIS IS FIRST ERROR
BNE	1\$;BRANCH IF NOT FIRST ERROR
ERROR	202		
BR	2\$;BRANCH TO EXIT
1\$: ERROR	302		
2\$: JMP	\$OLDPC		;RETURN TO TEST

```

.SBTTL COUNT PATTERN ERRORS IN P.A.R.'S OR P.D.R.'S
:*****
:*
:* THIS SUBROUTINE IS USED TO LOG AND REPORT THE COUNT PATTERN

```

6808
6809
6810
6811
6812
6813
6814
6815

```

:* 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'.
:*
:*****

```

```

6816 044462
6817 044462 012637 001434
6818 044466 050037 001424
6819 044472 005100
6820 044474 040037 001426
6821 044500 005100
6822 044502 050137 001420
6823 044506 005101
6824 044510 040137 001422
6825 044514 005101
6826 044516 050237 001414
6827 044522 005102
6828 044524 040237 001416
6829 044530 005102
6830 044532 105737 001103
6831 044536 001002
6832 044540 104203
6833 044542 000401
6834 044544 104303
6835 044546 000177 134662
6836
6837
6838
6839
6840
6841 000001

```

```

PARCOUNT:
MOV (KSP)+,OLDPC
BIS R0,ADDRCR
COM R0
BIC R0,ADRAND
COM R0
BIS R1,FATTOR
COM R1
BIC R1,PATAND
COM R1
BIS R2,DATAOR
COM R2
BIC R2,DATAND
COM R2
TSTB $ERFLG
BNE 1$
ERROR 203
BR 2$
ERROR 303
JMP @OLDPC
1$:
2$:

```

```

;STARTING ADDRESS OF THIS SUBROUTINE
;SAVE RETURN ADDRESS IN CASE OF LOOP
;LOGICAL OR OF FAILING ADDRESS
;GET R0 READY FOR AND
;PERFORM LOGICAL AND
;PUT R0 BACK AS IT WAS
;LOGICAL OR OF PATTERN LOADED
;GET R1 READY FOR AND
;PERFORM LOGICAL AND
;PUT R1 BACK AS IT WAS
;LOGICAL OR OF DATA FETCHED
;GET R2 READY FOR AND
;PERFORM LOGICAL AND
;PUT R2 BACK AS IT WAS
;SEE IF THIS IS THE FIRST ERROR
;BRANCH IF NOT FIRST ERROR
;BRANCH TO EXIT
;RETURN TO TEST

```

.END

ABASE = 000000	741	782																		
ACDW1 = 000000	741	784																		
ACDW2 = 000000	741	785																		
ACPUOP = 000000	741	756																		
ADDROR = 001424	834#	2197	2201	2204	6734*	6762*	6788*	6818*												
ADDW0 = 000000	741	786																		
ADDW1 = 000000	741	787																		
ADDW10 = 000000	741	796																		
ADDW11 = 000000	741	797																		
ADDW12 = 000000	741	798																		
ADDW13 = 000000	741	799																		
ADDW14 = 000000	741	800																		
ADDW15 = 000000	741	801																		
ADDW2 = 000000	741	788																		
ADDW3 = 000000	741	789																		
ADDW4 = 000000	741	790																		
ADDW5 = 000000	741	791																		
ADDW6 = 000000	741	792																		
ADDW7 = 000000	741	793																		
ADDW8 = 000000	741	794																		
ADDW9 = 000000	741	795																		
ADEVCT = 000000	741	747																		
ADEVN = 000000	741	783																		
ADRAND = 001426	835#	2197	2201	2204	6738*	6764*	6790*	6820*												
AENV = 000000	741	752																		
AENVN = 000000	741	753																		
AFATAL = 000000	741	744																		
AMADR1 = 000000	741	769																		
AMADR2 = 000000	741	773																		
AMADR3 = 000000	741	776																		
AMADR4 = 000000	741	779																		
AMAMS1 = 000000	741	763																		
AMAMS2 = 000000	741	771																		
AMAMS3 = 000000	741	774																		
AMAMS4 = 000000	741	777																		
AMSGAD = 000000	741	749																		
AMSGLG = 000000	741	750																		
AMSGTY = 000000	741	743																		
AMTYP1 = 000000	741	764																		
AMTYP2 = 000000	741	772																		
AMTYP3 = 000000	741	775																		
AMTYP4 = 000000	741	778																		
APASS = 000000	741	746																		
APRIOR = 000000	741																			
APTCSU = 000040	6014	6119#																		
APTENV = 000001	6007	6075	6117#	6510																
APTSIZ = 000200	2638	6116#																		
APTSPO = 000100	6009	6077	6118#																	
ASWREG = 000000	741	754																		
ATESTN = 000000	741	745																		
AUNIT = 000000	741	748																		
AUSWR = 000000	741	755																		
AVECT1 = 000000	741	780																		
AVECT2 = 000000	741	781																		
BADPC = 001410	828#	2182	2183	2185	2187	2189	2397*	2436*	2481*											
BITO = 000001	524#	4132	4384	4484	4487	4565	4890	5006	5137	5205	5259	5327	5375							

DF22	014761	1002	2307#						
DF23	014766	1008	1020	1026	2309#				
DF27	014773	1032	2311#						
DF3	014663	908	915	2281#					
DF30	014776	1039	2312#						
DF31	015003	1046	2314#						
DF32	015012	1052	2317#						
DF33	015016	1059	2319#						
DF34	015025	1065	2322#						
DF35	015030	1071	1077	2323#					
DF37	015034	1083	1089	1149	1221	2325#			
DF41	015041	1095	1101	1107	1113	1119	1125	2327#	
DF47	015045	1136	2329#						
DF5	014670	922	2283#						
DF50	015054	1143	2332#						
DF52	015062	1155	2334#						
DF53	015066	1161	2336#						
DF54	015072	1167	2338#						
DF55	015076	1173	1179	1185	1191	1197	2340#		
DF6	014674	929	2285#						
DF62	015101	1203	2341#						
DF63	015105	1209	2343#						
DF64	015111	1215	2345#						
DF66	015115	1227	1245	2347#					
DF67	015121	1233	2349#						
DF7	014701	935	941	2287#					
DF70	015125	1239	2351#						
DF72	015130	1251	1257	2352#					
DF74	015133	1263	2353#						
DF75	015135	1269	1275	2354#					
DH1	007753	893	1279	1811#					
DH11	010360	945	1858#						
DH12	010420	951	1864#						
DH13	010460	957	1870#						
DH14	010531	963	1877#						
DH15	010571	969	1883#						
DH16	010720	976	1899#						
DH17	011027	982	1911#						
DH2	007776	899	1815#						
DH20	011077	988	1918#						
DH201	013536	1289	2158#						
DH202	013565	1297	2162#						
DH203	013645	1306	2171#						
DH21	011127	994	1012	1923#					
DH22	011167	1000	1929#						
DH23	011236	1006	1018	1024	1936#				
DH27	011306	1030	1943#						
DH3	010032	905	912	1820#					
DH30	011336	1036	1947#						
DH31	011436	1043	1958#						
DH32	011572	1050	1975#						
DH33	011632	1056	1981#						
DH34	011752	1063	1995#						
DH35	012002	1069	1075	1999#					
DH37	012042	1081	1147	1219	2005#				
DH40	012112	1087	2012#						

DT54	014450	1166	2248#				
DT55	014462	1172	1178	1184	1190	1196	2250#
DT6	013770	928	2189#				
DT62	014472	1202	2252#				
DT63	014504	1208	2254#				
DT64	014516	1214	2256#				
DT66	014530	1226	1244	2258#			
DT67	014542	1232	2260#				
DT7	014004	934	940	2191#			
DT70	014554	1238	2262#				
DT72	014564	1250	1256	2264#			
DT74	014574	1262	2266#				
DT75	014602	1268	1274	2267#			
DUALAD	044406	3204	3265	3325	3386	6786#	
EMTVEC=	000030	534#	2595*	2596*			
EM1	002516	892	898	1314#			
EM10	003257	938	1379#				
EM11	003310	944	1384#				
EM12	003352	950	1390#				
EM13	003406	956	1395#				
EM14	003454	962	1402#				
EM15	003515	968	1408#				
EM16	003557	975	1414#				
EM17	003621	981	1420#				
EM20	003665	987	1427#				
EM201	007516	1288	1781#				
EM202	007573	1296	1789#				
EM203	007664	1305	1799#				
EM21	003724	993	1433#				
EM22	003771	999	1440#				
EM23	004045	1005	1448#				
EM24	004106	1011	1454#				
EM25	004143	1017	1459#				
EM26	004215	1023	1467#				
EM27	004301	1029	1476#				
EM3	002577	904	1323#				
EM30	004361	1035	1485#				
EM31	004430	1042	1492#				
EM32	004507	1049	1500#				
EM33	004570	1055	1509#				
EM34	004641	1062	1516#				
EM35	004710	1068	1523#				
EM36	004763	1074	1531#				
EM37	005040	1080	1539#				
EM4	002715	911	1278	1337#			
EM40	005131	1086	1549#				
EM41	005216	1092	1558#				
EM42	005267	1098	1565#				
EM43	005337	1104	1572#				
EM44	005406	1110	1579#				
EM45	005453	1116	1586#				
EM46	005522	1122	1593#				
EM47	005573	1128	1600#				
EM5	003040	918	1353#				
EM50	005657	1139	1610#				
EM51	005761	1146	1622#				

TYPBIN	539#														
TYPDEC	539#	5894	5901												
TYPNAM	539#	2642													
TYPNUM	539#														
TYPDCS	539#														
TYPDCT	539#														
TYPTXT	539#	5890	5897												
UPCODE	2366#	6332													
USER	623#	806													
ZEROER	2366#	6531													
\$\$CMRE	680#	719	720	721	722	723	724								
\$\$CMTM	680#	725	726	727	728	729	730								
\$\$ESCA	539#														
\$\$NEWT	539#	2695	2743	2813	2843	2877	2931	2955	3019	3055	3086	3118	3158	3219	3279
	3340	3409	3454	3498	3541	3592	3627	3660	3695	3729	3774	4003	4102	4173	4230
	4307	4367	4574	4632	4780	4862	4976	5068	5213	5343	5500	5647	5776	5812	
\$\$SET	6287#	6296	6297	6298	6299	6302	6303	6304	6305						
\$\$SETM	2637#														
\$\$SETU	2605#														
\$\$SKIP	539#	2679	2732	2863	2998	3050	3082	3113	3145	3214	3275	3335	3396	3449	3494
	3536	3579	4081	4843	5052	5486	5632	5763							
.EQUAT	388#	429													
.HEADE	388#														
.KT11	388#	539													
.SETUP	388#	624													
.SWRHI	388#	397													
.SWRLO	409#														
.\$ACT1	388#	670													
.\$APT8	388#	738#													
.\$APTH	388#	648													
.\$APTY	388#	6063													
.\$CATC	388#	624													
.\$CMTA	388#	680													
.\$DB20	388#	6365													
.\$EOP	388#	5868													
.\$ERRO	388#	6472													
.\$POWE	388#	6307													
.\$SAVE	388#	5939													
.\$SCOP	388#	6405													
.\$STRAP	388#	6264													
.\$TYPD	388#	6197													
.\$TYPE	388#	5984													
.\$TYPO	388#	6120													

. ABS. 044552 000

ERRORS DETECTED: 0

CQKTAB.BIN,CQKTAB.LST/CRF/SOL=CQKTAB.P11
RUN-TIME: 31 23 2 SECONDS
RUN-TIME RATIO: 151/57=2.6
CORE USED: 31K (61 PAGES)

H12