

REM a

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

IDENTIFICATION

PRODUCT CODE	AC-8054B-MC
PRODUCT NAME	CFKTH80 PDP 11/34 MEM MGMT
DATE	JUNE 22 1978
MAINTAINER	DIAGNOSTIC PROGRAMMING
AUTHOR	DIAGNOSTIC ENGINEERING

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

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

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

COPYRIGHT (C) 1977, 1978 BY DIGITAL EQUIPMENT CORPORATION

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

PROGRAM HISTORY

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

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

TABLE OF CONTENTS

- 1 0 PROGRAM INFORMATION
 - 1 1 ABSTRACT
 - 1 2 REQUIREMENTS
 - 1 3 RELATED DOCUMENTS AND STANDARDS
 - 1 4 PRELIMINARY PROGRAMS

- 2 0 OPERATING INSTRUCTIONS
 - 2 1 LOADING PROCEDURES
 - 2 2 STARTING PROCEDURES
 - 2 3 OPERATIONAL SWITCH SETTINGS
 - 2 4 LOADING THE SWITCH REGISTER
 - 2 5 EXECUTION TIMES

- 3 0 ERROR INFORMATION
 - 3 1 ERROR REPORTING PROCEDURES
 - 3 2 INTERPRETING ERROR REPORTS
 - 3 3 SAMPLE ERROR REPORT

- 4 0 MISCELLANEOUS INFORMATION
 - 4 1 ACT/APT/XXDP COMPATABILITY
 - 4 2 END-OF-PASS MESSAGE
 - 4 3 T-BIT TRAPPING
 - 4 4 POWER FAILURE HANDLING
 - 4 5 PHYSICAL BUS ADDRESS CONSTRUCTION

- 5 0 PROGRAM DESCRIPTION
 - 5 1 SUBROUTINES USED BY THIS PROGRAM
 - 5 2 PROGRAM LISTING
 - 5 3 USING THE PROGRAM TO DIAGNOSE A FAULT

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

1 0 PROGRAM INFORMATION

1 1 ABSTRACT

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

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

1 2 REQUIREMENTS

A PDP 11/34 PROCESSOR WITH A MINIMUM OF 16K OF MEMORY AND A CONSOLE TERMINAL ARE REQUIRED TO RUN THE PROGRAM UNLESS THE PROGRAM IS RUNNING UNDER APT OR ACT IN WHICH CASE THE CONSOLE TERMINAL IS NOT NECESSARY

1 3 RELATED DOCUMENTS AND STANDARDS

- 1 ACT11/XXDP PROGRAMMING SPECIFICATION
- 2 STANDARD APT SYSTEM TO A PDP11 DIAGNOSTIC INTERFACE
- 3 DIAGNOSTIC ENGINEERING STANDARDS AND CONVENTIONS
- 4 PDP11 MAINDEC SYSMAC PACKAGE
- 5 XXDP USER'S MANUAL

1 4 PRELIMINARY PROGRAMS

BEFORE THIS MEMORY MANAGEMENT DIAGNOSTIC IS RUN, THE FOLLOWING CPU DIAGNOSTICS SHOULD BE RUN

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

ALSO, ONE OF THE MAIN MEMORY DIAGNOSTICS SHOULD BE RUN TO SCAN AT LEAST THE FIRST 16K TO SEE THAT A PROGRAM CAN BE EXECUTED

2 0 OPERATING INSTRUCTIONS

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

2 1 LOADING PROCEDURES

THE PROGRAM IS SUPPLIED ON THE DIAGNOSTIC LOAD MEDIA REFER TO THE XXDP USER'S MANUAL FOR FURTHER INFORMATION FOR USE WITH ACT OR APT, REFER TO THEIR RESPECTIVE DOCUMENTS THE PROGRAM CAN ALSO BE DIRECTLY LOADED USING THE ABSOLUTE LOADER AND THE BINARY PAPER TAPE

2 2 STARTING PROCEDURES

THE PROGRAM IS STARTED BY LOADING ADDRESS 200 AND STARTING IF THE PROCESSOR HAS THE OPTIONAL PROGRAMMER'S CONSOLE, THE SWITCH REGISTER SHOULD BE SET ACCORDING TO SECTION 2.3 BEFORE THE PROGRAM IS STARTED. IF THERE IS NO HARDWARE SWITCH REGISTER, THE PROGRAM WILL USE THE SOFTWARE SWITCH REGISTER AT LOCATION 176 (LOCATION 174 WILL BE USED AS THE SOFTWARE DISPLAY REGISTER) IN THAT CASE THE PROGRAM WILL ASK FOR THE INITIAL SWITCH REGISTER VALUE BY TYPING "SWR= XXXXXX NEW= " AFTER TYPING THE NAME OF THE PROGRAM (XXXXXX = THE OCTAL CONTENTS OF LOCATION 176). (SEE SECTION 2.4)

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

2 3 CONTROL SWITCH SETTINGS

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

216			NORMALLY TAKES PLACE DURING
217			EVERY OTHER PASS STARTING
218			WITH THE THIRD PASS
219			
220	SW11	004000	INHIBIT SUBTEST ITERATIONS
221			THIS SWITCH WHEN SET INHIBITS
222			ITERATIONS OF EACH SUBTEST AFTER
223			THE FIRST PASS IF THIS SWITCH
224			IS NOT SET, EACH SUBTEST IS RUN
225			200 TIMES
226			
227	SW10	002000	BELL ON ERROR
228			THIS SWITCH WHEN SET WILL RING
229			THE CONSOLE TERMINAL BELL WHEN
230			AN ERROR HAS BEEN DETECTED
231			
232	SW9	001000	LOOP ON ERROR
233			THIS SWITCH WHEN SET WILL
234			CAUSE THE PROGRAM TO LOOP ON THE
235			FIRST FAILURE WHICH IS ENCOUNTERED
236			EVEN IF THE FAILURE IS INTERMITTANT
237			
238	SW8	000400	LOOP ON TEST IN SWR<? 0>
239			THIS SWITCH WHEN SET WILL
240			CAUSE THE PROGRAM TO LOOP ON THE
241			TEST WHOSE TEST NUMBER IS SET
242			IN BITS 7-0 OF THE SWITCH REG

2 4 LOADING THE SWITCH REGISTER

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

TO LOAD THE SOFTWARE SWITCH REG. WHILE THE PROGRAM IS RUNNING, A CONTROL G (G) SHOULD BE TYPED ON THE CONSOLE TERMINAL. (THE "SCOPE" AND "ERROR" ROUTINES CHECK TO SEE IF A G HAS BEEN TYPED) THE ORIGINAL VALUE OF THE SOFTWARE SWITCH REG. WILL BE REQUESTED AS MENTIONED IN SECTION 2 2.

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

SWR = XXXXXX NEW -

WHERE "XXXXXX" IS THE CURRENT OCTAL CONTENTS OF LOC 176 THE OPERATOR MAY THEN TYPE ANY ONE OF THE FOLLOWING

XXXXXX<CR> ONE TO SIX OCTAL DIGITS FOLLOWED BY A CARRIAGE RETURN WHICH WILL BE LOADED AS THE NEW VALUE FOR THE SWITCH REG

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

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

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

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

C DIGITS TYPED SO FAR TO BE IGNORED
WILL CAUSE THE PROGRAM TO TYPE THE PRESENT
TEST AND PASS NUMBERS, REQUEST A NEW VALUE
FOR THE SWITCH REG, AND JUMP TO THE END-
OF-PASS ROUTINE SO THE PROGRAM WILL GO DIRECTLY
TO THE NEXT PASS WITH A NEW SW REG. VALUE
<ILL CHAR> ANY CHARACTER TYPED WHICH IS NOT ANY OF THE
ABOVE OR AN OCTAL DIGIT WILL CAUSE THE PROGRAM
TO TYPE A "?<CRLF>" AND REACT AS THOUGH A
U HAD BEEN TYPED

NOTE RECOGNITION OF A G MAY BE HAMPERED BY
----- EXECUTION OF A COUPLE "RESET" INSTRUCTIONS
WITHIN THE PROGRAM

2 5 EXECUTION TIMES

THE RUN TIME FOR A SINGLE PASS WITH NO ITERATIONS
OR TRACE TRAPPING IS APPROXIMATELY 5 SECONDS

THE RUN TIME FOR A SINGLE PASS WITH ITERATIONS
AND TRACE TRAPPING ENABLED IS APPROXIMATELY 3 1/4 MINUTES

3 0 ERROR INFORMATION

3 1 ERROR REPORTING PROCEDURES

IF AN ERROR IS DETECTED, THE PROGRAM WILL TRAP TO THE
ERROR HANDLING ROUTINE (\$ERROR). THE VALUE OF BITS
15, 13, 10, AND 9 IN THE SWITCH REGISTER ARE CONSIDERED
IN REPORTING AN ERROR (SEE SECTION 2 3) THE
ERROR INFORMATION WILL BE TYPED UNLESS SW13 = 1

IF SW15 = 1, THE PROCESSOR WILL HALT AFTER THE ERROR IS
REPORTED. IF THE CONTENTS OF THE SOFTWARE SWITCH REGISTER
ARE TO BE CHANGED, A G SHOULD BE TYPED BEFORE PRESSING
"CONTINUE" TO RESUME TESTING.

IF SW9 = 1 (LOOP ON ERROR), THE PROGRAM WILL GO TO THE
ADDRESS CONTAINED IN LOCATION "\$LPERR" AFTER REPORTING
THE ERROR "\$LPERR" IS SET BY EACH "SCOPE" CALL AND IS
SET DIRECTLY DURING SOME SUBTESTS TO PROVIDE THE SMALLEST
LOOP FOR LOOPING ON ERROR IF SW9 = 0, THE PROGRAM WILL
RETURN TO THE INSTRUCTION FOLLOWING THE ERROR CALL
(SEE SECTION 5 3 FOR MORE ON "LOOP ON ERROR")

3 2 INTERPRETING ERROR REPORTS

EVERY ERROR REPORT TYPES THE NUMBER OF THE TEST IN WHICH
THE ERROR TOOK PLACE (TESTNO) AND THE LOCATION OF THE
ERROR CALL (ERRORPC) THESE TWO VALUES PINPOINT THE
PLACE IN THE CODE THAT THE ERROR OCCURRED BY REFERRING

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

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

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

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

3 3 SAMPLE ERROR REPORT

BELOW IS AN EXAMPLE OF AN ERROR WHICH COULD HAVE OCCURRED DURING EXECUTION OF THE PROGRAM

```
MEM. MGMT REG BITS NOT SET CORRECTLY
REGISTR WRITE READ READ-(BINARY)
ADDRESS (OCTAL) (OCTAL) 5432109876543210 TESTNO ERRORPC
177572 040000 060000 0110000000000000 000012 022060
```

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

4 0 MISCELLANEOUS INFORMATION

4 1 ACT/APT/XXDP COMPATABILITY

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

4 2 END-OF-PASS MESSAGE

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

END OF PASS #2 TOTAL ERRORS SINCE LAST REPORT 0

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

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

4 3 T-BIT TRAPPING

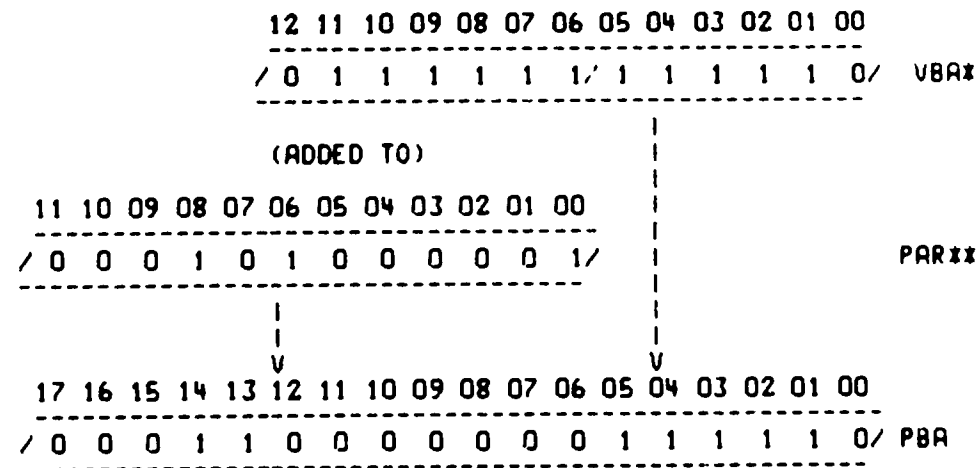
THE "T-BIT" (BIT 4) IN THE PROCESSOR STATUS WORD IS SET BY AN "RTI" IN THE END-OF-PASS ROUTINE FOR EVERY OTHER PASS BEGINNING WITH THE THIRD PASS (PASSES 3,5,7,9) T-BIT TRAPPING CAN BE INHIBITED BY SETTING BIT 12 = 1 IN THE SWITCH REGISTER (SEE SECTION 2.4)

4 4 POWER FAILURE HANDLING

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

4 5 PHYSICAL BUS ADDRESS CONSTRUCTION

BELOW IS A SIMPLIFIED DIAGRAM OF HOW THE MEMORY MANAGEMENT LOGIC CONSTRUCTS A PHYSICAL BUS ADDRESS USING THE VIRTUAL ADDRESS AND THE PAGE ADDRESS REGISTER THE PAGE DESCRIPTOR REGISTER SELECTED WILL CONTAIN THE PAGE EXPANSION, LENGTH, AND ACCESS INFORMATION



* = VBA BITS <15 13> SELECT THE APPROPRIATE PAR AND PDR
 ** = PSW MODE BIT 01 (BIT 15) SELECTS THE USER (=1) OR

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

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

5 0 PROGRAM DESCRIPTION

5 1 SUBROUTINES USED BY THIS PROGRAM

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

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

5 2 PROGRAM LISTING

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

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

5 3 USING THE PROGRAM TO DIAGNOSE A FAULT

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

IF YOU HAVE BEEN RUNNING WITH BIT 15 OF THE SWITCH REG. EQUAL TO "0", THEN YOU ARE ABLE TO LOOK AT ALL THE ERRORS THAT MAY BE RELATED TO THE FAULT YOU ARE DIAGNOSING. A FAULT IN AN EARLIER TEST MAY RESULT IN ERRORS DURING LATER TESTS WHICH MAY GIVE YOU MORE CLUES ABOUT THE NATURE OF THE FAULT. NOW USE THE METHOD OUTLINED IN SECTION 3.2 FOR EACH ERROR TO GATHER AS MUCH INFORMATION AS POSSIBLE

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

NOW TO TEST YOUR IDEAS ON THE CAUSE OF THE FAILURE,
YOU MAY WANT TO SCOPE THIS ERROR CONDITION SET BIT 09
OF THE SWITCH REG. EQUAL TO "1" TO LOOP ON THE ERROR
FOR AN EVEN TIGHTER SCOPE LOOP THE ERROR CALL CAN BE
REPLACED WITH A BRANCH (REFER TO COMMENTS BY ERROR CALLS
IN THE PROGRAM LISTING)

OR YOU COULD LOOP ON THE TEST BY EITHER SETTING BIT 14
OF THE SWITCH REG. EQUAL TO "1" OF BY SETTING BIT 08 OF THE
SWITCH REG. EQUAL TO "1" AND THEN SETTING THE TEST NUMBER
IN BITS 07-00 OF THE SWITCH REG YOU WILL PROBABLY WANT TO
INHIBIT ERROR TYPEOUTS BY SETTING BIT 13 OF THE SWITCH REG
EQUAL TO "1"

2

```

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

```

571 000300 PR6= 300 ..PRIORITY LEVEL 6
 572 000340 PR7= 340 ..PRIORITY LEVEL 7

.S"SWITCH REGISTER" SWITCH DEFINITIONS

574		SW15=	100000
575	100000	SW14=	40000
576	040000	SW13=	20000
577	020000	SW12=	10000
578	010000	SW11=	4000
579	004000	SW10=	2000
580	002000	SW09=	1000
581	001000	SW08=	400
582	000400	SW07=	200
583	000200	SW06=	100
584	000100	SW05=	40
585	000040	SW04=	20
586	000020	SW03=	10
587	000010	SW02=	4
588	000004	SW01=	2
589	000002	SW00=	1
590	000001	EQUIV	SW09, SW9
591		EQUIV	SW08, SW8
592		EQUIV	SW07, SW7
593		EQUIV	SW06, SW6
594		EQUIV	SW05, SW5
595		EQUIV	SW04, SW4
596		EQUIV	SW03, SW3
597		EQUIV	SW02, SW2
598		EQUIV	SW01, SW1
599		EQUIV	SW00, SW0

.S"DATA BIT DEFINITIONS (BIT00 TO BIT15)

601		BIT15=	100000
602		BIT14=	40000
603	100000	BIT13=	20000
604	040000	BIT12=	10000
605	020000	BIT11=	4000
606	010000	BIT10=	2000
607	004000	BIT09=	1000
608	002000	BIT08=	400
609	001000	BIT07=	200
610	000400	BIT06=	100
611	000200	BIT05=	40
612	000100	BIT04=	20
613	000040	BIT03=	10
614	000020	BIT02=	4
615	000010	BIT01=	2
616	000004	BIT00=	1
617	000002	EQUIV	BIT09, BIT9
618	000001	EQUIV	BIT08, BIT8
619		EQUIV	BIT07, BIT7
620		EQUIV	BIT06, BIT6
621		EQUIV	BIT05, BIT5
622		EQUIV	BIT04, BIT4
623		EQUIV	BIT03, BIT3
624		EQUIV	BIT02, BIT2
625			
626			

```

627          EQUIV BIT01,BIT1
628          EQUIV BIT00,BIT0
629
630          ,*BASIC "CPU" TRAP VECTOR ADDRESSES
631          ERRVEC= 4          .. TIME OUT AND OTHER ERRORS
632          RESVEC= 10         .. RESERVED AND ILLEGAL INSTRUCTIONS
633          TBITVEC=14        .. "T" BIT
634          TRTVEC= 14        .. TRACE TRAP
635          BPTVEC= 14        .. BREAKPOINT TRAP (BPT)
636          IOTVEC= 20        .. INPUT/OUTPUT TRAP (IOT) **SCOPE**
637          PWRVEC= 24        .. POWER FAIL
638          EMTVEC= 30        .. EMULATOR TRAP (EMT) **ERROR**
639          TRAPVEC=34        .. "TRAP" TRAP
640          TKVEC= 60          .. TTY KEYBOARD VECTOR
641          TPVEC= 64          .. TTY PRINTER VECTOR
642          PIRQVEC=240       .. PROGRAM INTERRUPT REQUEST VECTOR
643          SBTTL MEMORY MANAGEMENT DEFINITIONS
644
645          ,*KT11 VECTOR ADDRESS
646
647          MMVEC= 250
648
649          ,*KT11 STATUS REGISTER ADDRESSES
650
651          SRO= 177572
652          SR1= 177574
653          SR2= 177576
654          SR3= 172516
655
656          ,*USER "1" PAGE DESCRIPTOR REGISTERS
657
658          UIPDR0= 177600
659          UIPDR1= 177602
660          UIPDR2= 177604
661          UIPDR3= 177606
662          UIPDR4= 177610
663          UIPDR5= 177612
664          UIPDR6= 177614
665          UIPDR7= 177616
666
667          ,*USER "1" PAGE ADDRESS REGISTERS
668
669          UIPAR0= 177640
670          UIPAR1= 177642
671          UIPAR2= 177644
672          UIPAR3= 177646
673          UIPAR4= 177650
674          UIPAR5= 177652
675          UIPAR6= 177654
676          UIPAR7= 177656
677
678          ,*KERNEL "1" PAGE DESCRIPTOR REGISTERS
679
680          KIPDR0= 172300
681          KIPDR1= 172302
682          KIPDR2= 172304
    
```

```

683      172306      KIPDR3= 172306
684      172310      KIPDR4= 172310
685      172312      KIPDR5= 172312
686      172314      KIPDR6= 172314
687      172316      KIPDR7= 172316
688
689      , *KERNEL "I" PAGE ADDRESS REGISTERS
690
691      172340      KIPAR0= 172340
692      172342      KIPAR1= 172342
693      172344      KIPAR2= 172344
694      172346      KIPAR3= 172346
695      172350      KIPAR4= 172350
696      172352      KIPAR5= 172352
697      172354      KIPAR6= 172354
698      172356      KIPAR7= 172356
699
700      EQUIV SP, KSP
701      EQUIV SP, USP
702      EQUIV BIT4, TBIT
703      EQUIV BIT6, WBIT
704      001100      KERSTK= STACK
705      000700      USESTK= STACK-200
706
707      , *ADDITIONAL DEFINITIONS
708      , *
709
710      SBTTL TRAP CATCHER
711
712      =0
713      000000
714      , *ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A "+2, HALT"
715      , *SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
716      , *LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
717      =174
718      000174      000000      DISPREG: .WORD 0      , , SOFTWARE DISPLAY REGISTER
719      000176      000000      SWREG: .WORD 0      , , SOFTWARE SWITCH REGISTER
720
721      000200      000137      020000      SBTTL STARTING ADDRESS(ES)
722      JMP @START , , JUMP TO STARTING ADDRESS OF PROGRAM
723      SBTTL ACT11 HOOKS
724
725      , , *****
726      , *HOOKS REQUIRED BY ACT11
727      $SVPCL=      , , SAVE PC
728      =46
729      SENDAD      , , 1)SET LOC 46 TO ADDRESS OF SENDAD IN SEOP
730      =52
731      WORD 0      , , 2)SET LOC 52 TO ZERO
732      = $SVPCL      , , RESTORE PC
733      SBTTL APT PARAMETER BLOCK
734
735      , , *****
736      , *SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
737      , , *****
738      $X=      , , SAVE CURRENT LOCATION
739      =24      , , SET POWER FAIL TO POINT TO START OF PROGRAM
    
```

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

200 .. FOR APT START UP
 =44 .. POINT TO APT INDIRECT ADDRESS PNTR
 SAPTHDR .. POINT TO APT HEADER BLOCK
 = \$X .. RESET LOCATION COUNTER
 .. *****
 . SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
 . INTERFACE SPEC
 SAPTHD
 \$HIBTS WORD 0 .. TWO HIGH BITS OF 18 BIT MAILBOX ADDR
 \$MADR WORD \$MAIL .. ADDRESS OF APT MAILBOX (BITS 0-15)
 \$TSTM WORD 10 .. RUN TIM OF LONGEST TEST
 \$PASTM WORD 20 .. RUN TIME IN SECS OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
 \$UN TM WORD 5 .. ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
 \$ETEND-\$MAIL/2 .. LENGTH MAILBOX-ETABLE (WORDS)

Address	Value	Label	Description
754		SBTTL	COMMON TAGS
756		*****	
757		*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS	
758		*USED IN THE PROGRAM	
760	001100	=1100	
761	001100	SCMTAG	.. START OF COMMON TAGS
762	001100	WORD	0
763	001102	STSTNM:	BYTE 0 .. CONTAINS THE TEST NUMBER
764	001103	SERFLG:	BYTE 0 .. CONTAINS ERROR FLAG
765	001104	SICNT:	WORD 0 .. CONTAINS SUBTEST ITERATION COUNT
766	001106	SLPADR:	WORD 0 .. CONTAINS SCOPE LOOP ADDRESS
767	001110	SLPERR:	WORD 0 .. CONTAINS SCOPE RETURN FOR ERRORS
768	001112	SERTTL:	WORD 0 .. CONTAINS TOTAL ERRORS DETECTED
769	001114	SITEMB:	BYTE 0 .. CONTAINS ITEM CONTROL BYTE
770	001115	SERMAX:	BYTE 1 .. CONTAINS MAX. ERRORS PER TEST
771	001116	SERRPC:	WORD 0 .. CONTAINS PC OF LAST ERROR INSTRUCTION
772	001120	SGDADR:	WORD 0 .. CONTAINS ADDRESS OF 'GOOD' DATA
773	001122	SBDADR:	WORD 0 .. CONTAINS ADDRESS OF 'BAD' DATA
774	001124	SGDDAT:	WORD 0 .. CONTAINS 'GOOD' DATA
775	001126	SBD DAT:	WORD 0 .. CONTAINS 'BAD' DATA
776	001130	WORD	0 .. RESERVED--NOT TO BE USED
777	001132	WORD	0
778	001134	SAUTOB:	BYTE 0 .. AUTOMATIC MODE INDICATOR
779	001135	SINTAG:	BYTE 0 .. INTERRUPT MODE INDICATOR
780	001136	WORD	0
781	001140	SWR	WORD DSWR .. ADDRESS OF SWITCH REGISTER
782	001142	DISPLAY	WORD DDISP .. ADDRESS OF DISPLAY REGISTER
783	001144	STKS	177560 .. TTY KBD STATUS
784	001146	STKB:	177562 .. TTY KBD BUFFER
785	001150	STPS:	177564 .. TTY PRINTER STATUS REG. ADDRESS
786	001152	STPB:	177566 .. TTY PRINTER BUFFER REG. ADDRESS
787	001154	SNUL:	BYTE 0 .. CONTAINS NULL CHARACTER FOR FILLS
788	001155	SFILLS:	BYTE 2 .. CONTAINS # OF FILLER CHARACTERS REQUIRED
789	001156	SFILLC:	BYTE 12 .. INSERT FILL CHARS. AFTER A "LINE FEED"
790	001157	STPFLG:	BYTE 0 .. "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
791	001160	SREGAD:	WORD 0 .. CONTAINS THE ADDRESS FROM WHICH (\$REG0) WAS OBTAINED
792			
793	001162	SREG0:	WORD 0 .. CONTAINS ((SREGAD)+0)
794	001164	SREG1:	WORD 0 .. CONTAINS ((SREGAD)+2)
795	001166	SREG2:	WORD 0 .. CONTAINS ((SREGAD)+4)
796	001170	SREG3:	WORD 0 .. CONTAINS ((SREGAD)+6)
797	001172	SREG4:	WORD 0 .. CONTAINS ((SREGAD)+10)
798	001174	SREG5:	WORD 0 .. CONTAINS ((SREGAD)+12)
799	001176	STMP0:	WORD 0 .. USER DEFINED
800	001200	STMP1:	WORD 0 .. USER DEFINED
801	001202	STMP2:	WORD 0 .. USER DEFINED
802	001204	STMP3:	WORD 0 .. USER DEFINED
803	001206	STMP4:	WORD 0 .. USER DEFINED
804	001210	STMP5:	WORD 0 .. USER DEFINED
805	001212	STIMES:	0 .. MAX. NUMBER OF ITERATIONS
806	001214	SESCAPE:	0 .. ESCAPE ON ERROR ADDRESS
807	001216	SBELL	ASCIZ <207><377><377> .. CODE FOR BELL
808	001222	SQUES	ASCII /?/ .. QUESTION MARK
809	001223	SCRFL	ASCII <15> .. CARRIAGE RETURN

000377

Line	Address	Value	Field	Code	Description
810	001224	000012	SLF	ASCIZ	<12> ..LINE FEED
811					..*****
812			SBTTL		APT MAILBOX-ETABLE
813					..*****
814					..*****
815			EVEN		
816	001226		SMAIL		..APT MAILBOX
817	001226	000000	SMAGTY	WORD	AMSGTY ..MESSAGE TYPE CODE
818	001230	000000	SFATAL	WORD	AFATAL ..FATAL ERROR NUMBER
819	001232	000000	STESTN	WORD	ATESTN ..TEST NUMBER
820	001234	000000	SPASS	WORD	APASS ..PASS COUNT
821	001236	000000	SDEVCT	WORD	ADEVCT ..DEVICE COUNT
822	001240	000000	SUNIT	WORD	AUNIT ..I/O UNIT NUMBER
823	001242	000000	SMAGAD	WORD	AMSGAD ..MESSAGE ADDRESS
824	001244	000000	SMAGLG	WORD	AMSGLG ..MESSAGE LENGTH
825	001246		SETABLE		..APT ENVIRONMENT TABLE
826	001246	000	SENV	BYTE	AENV ..ENVIRONMENT BYTE
827	001247	000	SENV1	BYTE	AENV1 ..ENVIRONMENT MODE BITS
828	001250	000000	SSWREG	WORD	ASWREG ..APT SWITCH REGISTER
829	001252	000000	SUSWR	WORD	AUSWR ..USER SWITCHES
830	001254	000000	SCPUOP	WORD	ACPUOP ..CPU TYPE, OPTIONS
831			.		BITS 15-11=CPU TYPE
832			.		11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05
833			.		11/70=06, PDQ=07, Q=10
834			.		BIT 10=REAL TIME CLOCK
835			.		BIT 9=FLOATING POINT PROCESSOR
836			.		BIT 8=MEMORY MANAGEMENT
837	001256	000	SMAMS1	BYTE	AMAMS1 ..HIGH ADDRESS, M S BYTE
838	001257	000	SMTYP1	BYTE	AMTYP1 ..MEM TYPE, BLK#1
839			.		MEM TYPE BYTE -- (HIGH BYTE)
840			.		900 MSEC CORE=001
841			.		300 MSEC BIPOLAR=002
842			.		500 MSEC MOS=003
843	001260	000000	SMADR1	WORD	AMADR1 ..HIGH ADDRESS, BLK#1
844			.		MEM LAST ADDR =3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
845	001262	000	SMAMS2	BYTE	AMAMS2 ..HIGH ADDRESS, M S BYTE
846	001263	000	SMTYP2	BYTE	AMTYP2 ..MEM TYPE, BLK#2
847	001264	000000	SMADR2	WORD	AMADR2 ..MEM LAST ADDRESS, BLK#2
848	001266	000	SMAMS3	BYTE	AMAMS3 ..HIGH ADDRESS, M S BYTE
849	001267	000	SMTYP3	BYTE	AMTYP3 ..MEM TYPE, BLK#3
850	001270	000000	SMADR3	WORD	AMADR3 ..MEM LAST ADDRESS, BLK#3
851	001272	000	SMAMS4	BYTE	AMAMS4 ..HIGH ADDRESS, M S BYTE
852	001273	000	SMTYP4	BYTE	AMTYP4 ..MEM TYPE, BLK#4
853	001274	000000	SMADR4	WORD	AMADR4 ..MEM LAST ADDRESS, BLK#4
854	001276	000000	SVECT1	WORD	AVECT1 ..INTERRUPT VECTOR#1, BUS PRIORITY#1
855	001300	000000	SVECT2	WORD	AVECT2 ..INTERRUPT VECTOR#2, BUS PRIORITY#2
856	001302	000000	SBASE	WORD	ABASE ..BASE ADDRESS OF EQUIPMENT UNDER TEST
857	001304	000000	SDEV1	WORD	ADEV1 ..DEVICE MAP
858	001306	000000	SCDW1	WORD	ACDW1 ..CONTROLLER DESCRIPTION WORD#1
859	001310	000000	SCDW2	WORD	ACDW2 ..CONTROLLER DESCRIPTION WORD#2
860	001312	000000	SDDW0	WORD	ADDW0 ..DEVICE DESCRIPTOR WORD#0
861	001314	000000	SDDW1	WORD	ADDW1 ..DEVICE DESCRIPTOR WORD#1
862	001316	000000	SDDW2	WORD	ADDW2 ..DEVICE DESCRIPTOR WORD#2
863	001320	000000	SDDW3	WORD	ADDW3 ..DEVICE DESCRIPTOR WORD#3
864	001322	000000	SDDW4	WORD	ADDW4 ..DEVICE DESCRIPTOR WORD#4
865	001324	000000	SDDW5	WORD	ADDW5 ..DEVICE DESCRIPTOR WORD#5

866	001326	000000	\$DDW6	WORD	ADDW6	.. DEVICE DESCRIPTOR WORD#6
867	001330	000000	\$DDW7	WORD	ADDW7	.. DEVICE DESCRIPTOR WORD#7
868	001332	000000	\$DDW8	WORD	ADDW8	.. DEVICE DESCRIPTOR WORD#8
869	001334	000000	\$DDW9	WORD	ADDW9	.. DEVICE DESCRIPTOR WORD#9
870	001336	000000	\$DDW10	WORD	ADDW10	.. DEVICE DESCRIPTOR WORD#10
871	001340	000000	\$DDW11	WORD	ADDW11	.. DEVICE DESCRIPTOR WORD#11
872	001342	000000	\$DDW12	WORD	ADDW12	.. DEVICE DESCRIPTOR WORD#12
873	001344	000000	\$DD 3	WORD	ADDW13	.. DEVICE DESCRIPTOR WORD#13
874	001346	000000	\$DDW14	WORD	ADDW14	.. DEVICE DESCRIPTOR WORD#14
875	001350	000000	\$DDW15	WORD	ADDW15	.. DEVICE DESCRIPTOR WORD#15
876						
877						
878	001352		\$ETEND			
879						
880						
881	001352	000000	TESTNO	WORD	0	, HOLDS TEST NUMBER FOR TYPEOUTS
882	001354	000000	WASR6	WORD	0	, USED TO STORE THE STACK POINTER AFTER A TRAP
883	001356	000000	TRAPP	WORD	0	, USED TO STORE THE PC OF A TRAP OR ABORT
884	001360	000000	TRAPP:	WORD	0	, USED TO STORE THE PS OF A TRAP OR ABORT
885	001362	000000	CORSR0	WORD	0	, +USED TO STORE THE CORRECT SR0
886	001364	000000	CORSR2	WORD	0	, +USED TO STORE THE CORRECT SR2
887	001366	000000	WASSR0	WORD	0	, USED TO STORE CONTENTS OF SR0
888	001370	000000	WASSR2	WORD	0	, USED TO STORE CONTENTS OF SR2
889	001372	000000	TBITPS	WORD	0	, SAVES THE PSW THAT MAY HAVE ITS T-BIT ON
890	001374	000000	ANDADR	WORD	0	, HOLDS RESULT OF ADDRESSES BEING AND-ED
891	001376	000000	ORADR	WORD	0	, HOLDS RESULT OF ADDRESSES BEING OR-ED
892	001400	000000	TONUM	WORD	0	, HOLDS NUMBER OF TIME-OUTS
893	001402	000000	VIRT1	WORD	0	, HOLDS VIRTUAL ADDRESS TO BE CONVERTED
894	001404	000000	VIRT2	WORD	0	, "
895	001406	000000	PBAL0	WORD	0	, HOLDS BITS <15 00> OF PHYSICAL ADDRESS
896	001410	000000	PBAH1	WORD	0	, HOLDS BITS <17 16> OF PHYSICAL ADDRESS

897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952

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 SITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT
 . NOTE1 IF SITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC)
 . NOTE2 EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS

. X EM .. POINTS TO THE ERROR MESSAGE
 . X DH .. POINTS TO THE DATA HEADER
 . X DT .. POINTS TO THE DATA
 . X DF .. POINTS TO THE DATA FORMAT

\$ERRTB

. ITEM 1

EM1 . UNEXPECTED CPU TRAP TO LOC 004
 DH1 . OLD PC OLD PSW R6 WAS TESTNO ERRORPC
 DT1 . TRAPPC, TRAPPS, WASR6, TESTNO, \$ERPPC, 0
 DF1 . 0, 0, 0, 0

. ITEM 2

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

. ITEM 3

EM3 . PRIORITY BITS SET WRONG IN PSW
 DH3 . WROTE READ TESTNO ERRORPC
 DT3 . \$REG0, \$REG1, TESTNO, \$ERRPC, 0
 DF3 . 0, 0, 0, 0

. ITEM 4

EM4 . MODE BITS SET WRONG IN PSW
 DH3 . WROTE READ TESTNO ERRORPC
 DT3 . \$REG0, \$REG1, TESTNO, \$ERRPC, 0
 DF3 . 0, 0, 0, 0

. ITEM 5

EM5 . DUAL ADDRESSING BETWEEN HI&LO BYTES OF PSW
 DH3 . WROTE READ TESTNO ERRORPC
 DT3 . \$REG0, \$REG1, TESTNO, \$ERRPC, 0
 DF3 . 0, 0, 0, 0

. ITEM 6

EM6 . KERNEL R6 CHANGED BY WRITING USER R6
 DH3 . WROTE READ TESTNO ERRORPC
 DT3 . \$REG0, \$REG1, TESTNO, \$ERRPC, 0
 DF3 . 0, 0, 0, 0

. ITEM 7

EM7 . A MEMORY MGMT REG TIMED OUT
 DH7 . ADDRESS TESTNO ERRORPC
 DT7 . \$REG0, TESTNO, \$ERRPC, 0

001412
001412 041616
001414 045620
001416 051220
001420 052112
001422 041656
001424 045670
001426 051234
001430 052117
001432 041725
001434 045760
001436 051254
001440 052126
001442 041764
001444 045760
001446 051254
001450 052126
001452 042017
001454 045760
001456 051254
001460 052126
001462 042072
001464 045760
001466 051254
001470 052126
001472 042137
001474 046020
001476 051266

Line	Address	Offset	Code	Description
953	001500	052132	DF7	.0.0.0
954				
955			. * ITEM 10	
956	001502	042175	EM10	. SUMMARY OF MEM. MGMT REG TIMEOUTS
957	001504	046050	DH10	. REGISTER-ADDRS NUM OF
958				. AND-ED OR-ED TIMOUTS TESTNO ERRORPC
959	001506	051276	DT10	. ANDADR. ORADR. TONUM. TESTNO. \$ERRPC. 0
960	001510	052135	DF10	. G. 0. 1. 0. 0
961				
962			. * ITEM 11	
963	001512	042241	EM11	. MEM. MGMT REG WOULD NOT CLEAR
964	001514	046150	DH11	. REGISTR READ READ-(BINARY)
965				. ADDRESS (OCTAL) 5432109876543210 TESTNO ERRORPC
966	001516	051312	DT11	. \$REG0. \$REG1. \$REG1. TESTNO. \$ERRPC. 0
967	001520	052142	DF11	. 0. 0. 2. 0. 0
968				
969			. * ITEM 12	
970	001522	042301	EM12	. MEM. MGMT REG BITS NOT SET CORRECTLY
971	001524	046270	DH12	. REGISTR WROTE READ READ
972				. ADDRESS (OCTAL) (OCTAL) (BINARY) TESTNO ERRORPC
973	001526	051326	DT12	. \$REG0. \$REG1. \$REG2. \$REG2. TESTNO. \$ERRPC. 0
974	001530	052147	DF12	. 0. 0. 0. 2. 0. 0
975				
976			. * ITEM 13	
977	001532	042350	EM13	. SRO EFFECTED BY WRITE TO PSW
978	001534	046430	DH13	. READ TESTNO ERRORPC
979	001536	051344	DT13	. \$REG0. TESTNO. \$ERRPC. 0
980	001540	052155	DF13	. 0. 0. 0
981				
982			. * ITEM 14	
983	001542	042405	EM14	. SR1 DID NOT READ ALL ZEROS
984	001544	046430	DH13	. READ TESTNO ERRORPC
985	001546	051344	DT13	. \$REG0. TESTNO. \$ERRPC. 0
986	001550	052155	DF13	. 0. 0. 0
987				
988			. * ITEM 15	
989	001552	042440	EM15	. DUAL ADDRESSING BETWEEN BYTES OF PAR OR PDR
990	001554	046270	DH12	. REGISTER WROTE READ READ
991				. ADDRESS (OCTAL) (OCTAL) (BINARY) TESTNO ERRORPC
992	001556	051326	DT12	. \$REG0. \$REG1. \$REG2. \$REG2. TESTNO. \$ERRPC. 0
993	001560	052147	DF12	. 0. 0. 0. 2. 0. 0
994				
995			. * ITEM 16	
996	001562	042514	EM16	. DUAL ADDRESSING BETWEEN PAR-PDR'S
997	001564	046460	DH16	. PAR-PDR PAR-PDR
998				. CLEARED EFFECTD EXPECTD RECEIVD TESTNO ERRORPC
999	001566	051354	DT16	. \$REG0. \$REG1. \$REG5. \$REG2. TESTNO. \$ERRPC. 0
1000	001570	052160	DF16	. 0. 0. 0. 0. 0. 0
1001				
1002			. * ITEM 17	
1003	001572	042556	EM17	. PHYS ADDR FORMED READ WRONG IN MAINT MODE
1004	001574	046560	DH17	. PHYSICAL VIRTUAL
1005				. ADDRESS ADDRESS KIPAR4 TESTNO ERRORPC
1006	001576	051372	DT17	. PBALO. VIRT1. \$REG4. TESTNO. \$ERRPC. 0
1007	001600	052166	DF17	. 3. 0. 0. 0. 0
1008				

Item No	Code	Address	Pointer	Description
1009			. *ITEM 20	
1010	001602	042626	EM20	. PHYS ADDR FORMED READ WRONG IN RELOCATE MODE
1011	001604	046650	DH20	. PHYSICL PAR 4 PAR 5
1012				. ADDRESS VBA VBA PAR 4 PAR 5 PSW TESTNO
1013	001606	051406	DT20	. PBA0, VIRT1, VIRT2, \$REG4, \$REG5, \$TMP0, TESTNO, \$ERRPC, 0
1014	001610	052173	DF20	. 3, 0, 0, 0, 0, 0, 0, 0
1015				
1016			. *ITEM 21	
1017	001612	042700	EM21	. W-BIT DID NOT GET SET IN PDR
1018	001614	046776	DH21	. PDR VIRTUAL
1019				. TESTED ADDRESS TESTNO ERRORPC
1020	001616	051430	DT21	. \$REG5, \$REG3, TESTNO, \$ERRPC, 0
1021	001620	052203	DF21	. 0, 0, 0, 0
1022				
1023			. *ITEM 22	
1024	001622	042735	EM22	. W-BIT SET IN MORE THAN ONE PDR
1025	001624	047056	DH22	. PDR IN PDR VIRTUAL
1026				. ERROR TESTED ADDRESS TESTNO ERRORPC
1027	001626	051442	DT22	. \$REG0, \$REG5, \$REG3, TESTNO, \$ERRPC, 0
1028	001630	052207	DF22	. 0, 0, 0, 0, 0
1029				
1030			. *ITEM 23	
1031	001632	042774	EM23	. W-BIT NOT CLEARED BY WRITING TO PDR
1032	001634	047155	DH23	. PDR TESTNO ERRORPC
1033	001636	051456	DT23	. \$REG5, TESTNO, \$ERRPC, 0
1034	001640	052214	DF23	. 0, 0, 0
1035				
1036			. *ITEM 24	
1037	001642	043040	EM24	. WRITING SRO SET W-BIT IN KIPDR7
1038	001644	047205	DH24	. PDR WAS EXPECTD TESTNO ERRORPC
1039	001646	051466	DT24	. \$REG2, \$REG1, TESTNO, \$ERRPC, 0
1040	001650	052217	DF24	. 0, 0, 0, 0
1041				
1042			. *ITEM 25	
1043	001652	043100	EM25	. W-BIT GOT SET DURING ODD ADDR ABORT
1044	001654	047205	DH24	. PDR WAS EXPECTD TESTNO ERRORPC
1045	001656	051466	DT24	. \$REG2, \$REG1, TESTNO, \$ERRPC, 0
1046	001660	052217	DF24	. 0, 0, 0, 0
1047				
1048			. *ITEM 26	
1049	001662	043145	EM26	. MEMORY MGMT ACCESS ABORT DID NOT OCCUR
1050	001664	047245	DH26	. PDR 4 PSW TESTNO ERRORPC
1051	001666	051500	DT26	. \$REG2, \$TMP0, TESTNO, \$ERRPC, 0
1052	001670	052217	DF24	. 0, 0, 0, 0
1053				
1054			. *ITEM 27	
1055	001672	043215	EM27	. ACCESS ERROR DID NOT ABORT INSTRUCTION
1056	001674	047245	DH26	. PDR 4 PSW TESTNO ERRORPC
1057	001676	051500	DT26	. \$REG2, \$TMP0, TESTNO, \$ERRPC, 0
1058	001700	052217	DF24	. 0, 0, 0, 0
1059				
1060			. *ITEM 30	
1061	001702	043264	EM30	. SRO DID NOT REPORT ACCESS ERROR CORRECTLY
1062	001704	047305	DH30	. SRO WAS EXPECTD PDR 4 PSW TESTNO ERRORPC
1063	001706	051512	DT30	. WASSRO, \$REG3, \$REG2, \$TMP0, TESTNO, \$ERRPC, 0
1064	001710	052223	DF30	. 0, 0, 0, 0, 0, 0

1065					
1066			. *ITEM 31		
1067	001712	043336	EM31		. SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR
1068	001714	047365	DH31		. SR2 WAS EXPECTD PDR 4 PSW TESTNO ERRORPC
1069	001716	051530	DT31		. WASSR2, \$REG4, \$REG2, \$TMP0, TESTNO, \$ERRPC, 0
1070	001720	052223	DF30		. 0, 0, 0, 0, 0
1071					
1072			. *ITEM 32		
1073	001722	043403	EM32		. PAGE LGTH ABORT OCCURRED WHEN IT SHOULDN'T HAVE
1074	001724	047445	DH32		. V B A KIPDR4 SR0 WAS SR2 WAS TESTNO ERRORPC
1075	001726	051546	DT32		. \$REG0, \$REG4, WASSR0, WASSR2, TESTNO, \$ERRPC, 0
1076	001730	052223	DF30		. 0, 0, 0, 0, 0
1077					
1078			. *ITEM 33		
1079	001732	043464	EM33		. PAGE LGTH ABORT DID NOT OCCUR WHEN IT SHOULD HAVE
1080	001734	047525	DH33		. V B A KIPDR4 TESTNO ERRORPC
1081	001736	051564	DT33		. \$REG0, \$REG4, TESTNO, \$ERRPC, 0
1082	001740	052217	DF24		. 0, 0, 0, 0
1083					
1084			. *ITEM 34		
1085	001742	043547	EM34		. SR0 DID NOT REPORT PAGE LGTH ABORT CORRECTLY
1086	001744	047565	DH34		. V B A KIPDR4 SR0 WAS EXPECTD TESTNO ERRORPC
1087	001746	051576	DT34		. \$REG0, \$REG4, WASSR0, \$REG2, TESTNO, \$ERRPC, 0
1088	001750	052223	DF30		. 0, 0, 0, 0, 0
1089			. *ITEM 35		
1090	001752	043336	EM31		. SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR
1091	001754	047645	DH35		. V B A KIPDR4 SR2 WAS EXPECTD TESTNO ERRORPC
1092	001756	051614	DT35		. \$REG0, \$REG4, WASSR2, \$REG3, TESTNO, \$ERRPC, 0
1093	001760	052223	DF30		. 0, 0, 0, 0, 0
1094					
1095			. *ITEM 36		
1096	001762	043336	EM31		. SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR
1097	001764	047725	DH36		. SR2 WAS EXPECTD TESTNO ERRORPC
1098	001766	051632	DT36		. WASSR2, \$REG1, TESTNO, \$ERRPC, 0
1099	001770	052217	DF24		. 0, 0, 0, 0
1100					
1101			. *ITEM 37		
1102	001772	043625	EM37		. SR0 OR SR2 CHANGED BY A SECOND ABORT
1103	001774	047765	DH37		. FIRST ABORT SECOND ABORT
1104					. SR0 WAS SR2 WAS SR0 WAS SR2 WAS TESTNO ERRORPC
1105	001776	051644	DT37		. \$TMP0, \$TMP2, WASSR0, WASSR2, TESTNO, \$ERRPC, 0
1106	002000	052223	DF30		. 0, 0, 0, 0, 0
1107					
1108			. *ITEM 40		
1109	002002	043672	EM40		. SR0 OR SR2 WAS NOT "RESET" BY A RESET
1110	002004	050102	DH40		. SR0 WAS SR2 WAS TESTNO ERRORPC
1111	002006	051662	DT40		. WASSR0, WASSR2, TESTNO, \$ERRPC, 0
1112	002010	052217	DF24		. 0, 0, 0, 0
1113					
1114			. *ITEM 41		
1115	002012	043741	EM41		. SR2 NOT TRACKING CORRECTLY
1116	002014	047725	DH36		. SR2 WAS EXPECTD TESTNO ERRORPC
1117	002016	051632	DT36		. WASSR2, \$REG1, TESTNO, \$ERRPC, 0
1118	002020	052217	DF24		. 0, 0, 0, 0
1119					
1120			. *ITEM 42		

1121	002022	043774	EM42	. DID NOT TRAP THRU KERNEL SPACE
1122	002024	050142	DH42	. PSW WAS R6 WAS TESTNO ERRORPC
1123	002026	051674	DT42	. \$REG1, \$REG2, TESTNO, \$ERRPC, 0
1124	002030	052217	DF24	. 0, 0, 0, 0
1125				
1126			. X ITEM 43	
1127	002032	044033	EM43	. KT ERROR SERVICED ON ODD ADDR ERROR
1128	002034	047155	DH23	. PDR TESTNO ERRORPC
1129	002036	051456	DT23	. \$REG5, TESTNO, \$ERRPC, 0
1130	002040	052214	DF23	. 0, 0, 0
1131				
1132			. X ITEM 44	
1133	002042	044100	EM44	. SRO OR SR2 CHANGED BY ODD ADDR ERROR
1134	002044	050202	DH44	. EXPECTED RECEIVED
1135			. SRO SR2 SRO WAS SR2 WAS TESTNO ERRORPC	
1136	002046	051706	DT44	. \$REG0, \$REG1, WASSRO, WASSR2, TESTNO, \$ERRPC, 0
1137	002050	052223	DF30	. 0, 0, 0, 0, 0, 0
1138				
1139			. X ITEM 45	
1140	002052	044146	EM45	. ERROR DURING "DOUBLE ERROR" (KT & ODD ADDR)
1141	002054	050314	DH45	. EXPECTED:
1142			. PSW PC SRO SR2	
1143			. 170017 (35+4) 020147 (35)	
1144			. RECEIVED	
1145			. PSW PC SRO SR2 TESTNO ERRORPC	
1146	002056	051724	DT45	. \$REG1, \$REG3, WASSRO, WASSR2, TESTNO, \$ERRPC, 0
1147	002060	052223	DF30	. 0, 0, 0, 0, 0, 0
1148				
1149			. X ITEM 46	
1150	002062	044223	EM46	. MFPI INSTRUCTION PUSHED WRONG DATA
1151	002064	050511	DH46	. DATA DATA
1152			. EXPECTD RECEIVD TESTNO ERRORPC	
1153	002066	051742	DT46	. \$REG0, \$REG1, TESTNO, \$ERRPC, 0
1154	002070	052231	DF46	. 0, 0, 0, 0
1155				
1156			. X ITEM 47	
1157	002072	044266	EM47	. MTP1 INSTRUCTION LOADED WRONG DATA
1158	002074	050511	DH46	. DATA DATA
1159			. EXPECTD RECEIVD TESTNO ERRORPC	
1160	002076	051742	DT46	. \$REG0, \$REG1, TESTNO, \$ERRPC, 0
1161	002100	052231	DF46	. 0, 0, 0, 0
1162				
1163			. X ITEM 50	
1164	002102	044331	EM50	. STACK NOT PUSHED BY MFPI-MTP1
1165	002104	050566	DH50	. TESTNO ERRORPC
1166	002106	051754	DT50	. TESTNO, \$ERRPC, 0
1167	002110	052235	DF50	. 0, 0
1168				
1169			. X ITEM 51	
1170	002112	044367	EM51	. KERNEL PAGE ACCESSED INSTEAD OF USER MFPI-MTP1
1171	002114	050606	DH51	. SRO WAS SR2 WAS TESTNO ERRORPC
1172	002116	051762	DT51	. WASSRO, WASSR2, TESTNO, \$ERRPC, 0
1173	002120	052237	DF51	. 0, 0, 0, 0
1174				
1175			. X ITEM 52	
1176	002122	044445	EM52	. WRONG PDR'S REFERENCED WHILE IN RELOCATE MODE

Line No	Address	PC	Op	Comment
1177	002124	050646	DH52	; PHYSICL PAR 4 ; ADDRESS V. B A PAR 4 SRO WAS SR2 WAS PSW TESTNO ; PBA0, VIRT1, \$REG4, WASSRO, WASSR2, \$TMP0, TESTNO, \$ERRPC, 0 ; 3, 0, 0, 0, 0, 0, 0, 0
1178				
1179	002126	051774	DT52	
1180	002130	052243	DF52	; MFPD INSTRUCTION PUSHED WRONG DATA ; DATA DATA ; EXPECTD RECEIVD TESTNO ERRORPC ; \$REG0, \$REG1, TESTNO, \$ERRPC, 0 ; 0, 0, 0, 0
1181			. * ITEM 53	
1182	002132	044523	EM53	; STACK NOT PUSHED BY MFPD-MTPD ; TESTNO ERRORPC ; TESTNO, \$ERRPC, 0 ; 0, 0
1183	002134	050511	DH46	
1184				; PAR OR PDR WAS CHANGED BY A RESET ; REGISTR READ READ-(BINARY) ; ADDRESS (OCTAL) 5432109876543210 TESTNO ERRORPC ; \$REG0, \$REG1, \$REG1, TESTNO, \$ERRPC, 0 ; 0, 0, 2, 0, 0
1185	002136	051742	DT46	
1186	002140	052231	DF46	; ILLEGAL MODE 01 NOT ABORTED ; TESTNO ERRORPC ; TESTNO, \$ERRPC, 0 ; 0, 0
1187			. * ITEM 54	
1188				; SRO DID NOT REPORT ILLEGAL MODE 01 CORRECTLY ; SRO WAS EXPECTD TESTNO ERRORPC ; WASSRO, \$REG1, TESTNO, \$ERRPC, 0 ; 0, 0, 0, 0
1189	002142	044566	EM54	
1190	002144	050566	DH50	; PSW CHANGED BY AN RTI IN USER MODE ; PSW WAS EXPECTD TESTNO ERRORPC ; \$REG1, \$REG2, TESTNO, \$ERRPC, 0 ; 0, 0, 0, 0
1191	002146	051754	DT50	
1192	002150	052235	DF50	; MAINT MODE (SRO<8>) NOT DISABLED BY A RESET ; TESTNO ERRORPC ; TESTNO, \$ERRPC, 0 ; 0, 0
1193			. * ITEM 55	
1194				; DATA INCORRECT AFTER A MAINT MODE WRITE ; WROTE READ TESTNO ERRORPC ; \$REG0, \$REG1, TESTNO, \$ERRPC, 0 ; 0, 0, 0, 0
1195	002152	044624	EM55	
1196	002154	046150	DH11	; SOURCE RELOCATED IN MAINT MODE
1197			. * ITEM 56	
1198	002156	051312	DT11	
1199	002160	052142	DF11	
1200				
1201			. * ITEM 57	
1202	002162	044662	EM56	
1203	002164	050566	DH50	
1204	002166	051754	DT50	
1205	002170	052235	DF50	
1206				
1207			. * ITEM 60	
1208	002172	044716	EM57	
1209	002174	050764	DH57	
1210	002176	052016	DT57	
1211	002200	052253	DF57	
1212				
1213			. * ITEM 61	
1214	002202	044773	EM60	
1215	002204	051024	DH60	
1216	002206	052030	DT60	
1217	002210	052257	DF60	
1218				
1219			. * ITEM 62	
1220	002212	045036	EM61	
1221	002214	050566	DH50	
1222	002216	051754	DT50	
1223	002220	052235	DF50	
1224				
1225			. * ITEM 63	
1226	002222	045114	EM62	
1227	002224	045760	DH3	
1228	002226	051254	DT3	
1229	002230	052126	DF3	
1230				
1231			. * ITEM 63	
1232	002232	045165	EM63	

Line	Code	Address	Code	Description
1233	002234	045670	DH2	; OLD PC OLD PSW R6 WAS SRO SR2 TESTNO ERRORPC
1234	002236	051234	DT2	; TRAPPC, TRAPPS, WASSR6, WASSRO, WASSR2, TESTNO, SERRPC, 0
1235	002240	052117	DF2	; 0, 0, 0, 0, 0, 0, 0
1236				
1237				; *ITEM 64
1238	002242	043336	EM31	; SR2 DIDNOT LOCKUP CORRECT VIRTUAL ADDR
1239	002244	047725	DH36	; SR2 WAS EXPECTD TESTNO ERRORPC
1240	002246	052042	DT64	; WASSR2, SREG4, TESTNO, SERRPC, 0
1241	002250	052217	DF24	; 0, 0, 0, 0
1242				
1243				; *ITEM 65
1244	002252	045225	EM64	; +NON RESIDENT ABORT DID NOT OCCUR
1245	002254	051064	DH61	; +SRO SR2 TESTNO ERRORPC
1246	002256	052054	DT65	; +WASSRO, WASSR2, TESTNO, SERRPC
1247	002260	052263	DF61	; +0, 0, 0, 0
1248				
1249				; *ITEM 66
1250	002262	045266	EM65	; +ERROR FLAG FOR KR ABORT DID NOT SET
1251	002264	051064	DH61	; +SRO SR2 TESTNO ERRORPC
1252	002266	052054	DT65	; +WASSRO, WASSR2, TESTNO, SERRPC
1253	002270	052263	DF61	; +0, 0, 0, 0
1254				
1255				
1256				; *ITEM67
1257	002272	045351	EM66	; +SR2 DID NOT FREEZE THE VIRTUAL ADRS OF ABORT
1258	002274	051064	DH61	; +SRO SR2 TESTNO ERRORPC
1259	002276	052054	DT65	; +WASSRO, WASSR2, TESTNO, SERRPC
1260	002300	052263	DF61	; +0, 0, 0, 0
1261				
1262				
1263				; *ITEM 70
1264	002302	045441	EM67	; +2ND NON RESIDENT ABORT DID NOT OCCUR
1265	002304	051064	DH61	; +SRO SR2 TESTNO ERRORPC
1266	002306	052054	DT65	; +WASSRO, WASSR2, TESTNO, SERRPC
1267	002310	052263	DF61	; +0, 0, 0, 0
1268				
1269				; *ITEM 71
1270	002312	045506	EM70	; +2ND NR ABORT CHANGED SR2
1271	002314	051122	DH62	; +SR2 EXPT SR2 RECD TESTNO ERRORPC
1272	002316	052066	DT66	; +CORSRO, WASSR2, TESTNO, SERRPC
1273	002320	052263	DF61	; +0, 0, 0, 0
1274				
1275				; *ITEM 72
1276	002322	045562	EM71	; +SRO WAS NOT CLEARED BY INIT
1277	002324	051161	DH63	; +SRO EXPT SRO RECD TESTNO ERRORPC
1278	002326	052100	DT67	; +CORSRO, WASSRO, TESTNO, ERRORPC
1279	002330	052263	DF61	; +0, 0, 0, 0
1280				

```

1281          SBTTL ***** TRAP HANDLING ROUTINES *****
1282
1283          SBTTL CPU TRAP HANDLER ROUTINE
1284          , *****
1285          , *
1286          , * THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS THRU
1287          , * "ERRVEC" (LOC 004) IF THIS SUBROUTINE IS ENTERED BY A
1288          , * SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A HALT IS
1289          , * EXECUTED
1290          , *
1291          , *****
1292 002332 005227 TIMERR INC (PC)+ , MAKE FLAG ZERO IF FIRST TIME THRU
1293 002334 177777 TIMFLG WORD -1 , NEGATIVE ONE FOR "HAVE ENTERED" FLAG
1294 002336 001401 BEQ 15 , BRANCH IF FIRST TIME IN
1295 002340 000000 HALT , STOP! - I'VE ENTERED THIS ROUTINE
1296          , A SECOND TIME BEFORE I FINISHED
1297          , REPORTING THE FIRST ERROR THE
1298          , SECOND ENTRY ADDRESS SHOULD BE ON
1299          , THE KERNEL STACK
1300 002342 012637 001356 15 MOV (KSP)+, TRAPPC , SAVE PC+2 AT TIME OF ABORT
1301 002346 012637 001360 MOV (KSP)+, TRAPPS , SAVE PS AT TIME OF ABORT
1302 002352 010637 001354 MOV KSP, WASSR6 , SAVE STACK POINTER VALUE
1303 002356 104001 ERROR 1 , UNEXPECTED TRAP OR ABORT TO LOC 4
1304 002360 012737 177777 002334 MOV #-1, TIMFLG , MAKE FLAG NEGATIVE ONE FOR NEXT TIME
1305 002366 013746 001360 MOV TRAPPS, -(KSP) , PUT PC & PS OF TRAP ON STACK
1306 002372 013746 001356 MOV TRAPPC, -(KSP)
1307 002376 000006 RTT , RETURN FROM INTERRUPT OR ABORT
1308
1309
1310          SBTTL MEMORY MANAGEMENT TRAP HANDLER ROUTINE
1311          , *****
1312          , *
1313          , * THIS SUBROUTINE WILL HANDLE ALL UNEXPECTED MEMORY MANAGEMENT
1314          , * TRAPS AND ABORTS THRU "MIVEC" (LOC 250) IF THIS SUBROUTINE IS
1315          , * ENTERED BY A SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A
1316          , * HALT IS EXECUTED
1317          , *
1318          , *****
1319 002400 005227 MGMERR INC (PC)+ , MAKE FLAG ZERO IF FIRST TIME THRU
1320 002402 177777 MGMFLG WORD -1 , NEGATIVE ONE FOR "HAVE ENTERED" FLAG
1321 002404 001401 BEQ 15 , BRANCH IF FIRST TIME IN
1322 002406 000000 HALT , STOP! - I'VE ENTERED THIS ROUTINE
1323          , A SECOND TIME BEFORE I FINISHED
1324          , REPORTING THE FIRST ERROR THE
1325          , SECOND ENTRY ADDRESS SHOULD BE ON
1326          , THE KERNEL STACK
1327 002410 012637 001356 15 MOV (KSP)+, TRAPPC , SAVE PC+2 AT TIME OF ABORT
1328 002414 012637 001360 MOV (KSP)+, TRAPPS , SAVE PS AT TIME OF ABORT
1329 002420 010637 001354 MOV KSP, WASSR6 , SAVE STACK POINTER VALUE
1330 002424 013737 177572 001366 MOV SR0, WASSR0 , SAVE CONTENTS OF KT STATUS REG 0
1331 002432 013737 177576 001370 MOV SR2, WASSR2 , SAVE CONTENTS OF KT STATUS REG 2
1332 002440 042737 160000 177572 BIC #160000, SR0 , CLEAR ERROR BITS IN STATUS REG 0
1333 002446 104002 ERROR 2 , UNEXPECTED TRAP OR ABORT TO LOC 250
1334 002450 012737 177777 002402 MOV #-1, MGMFLG , MAKE FLAG NEGATIVE ONE FOR NEXT TIME
1335 002456 013746 001360 MOV TRAPPS, -(KSP) , PUT PC & PS OF TRAP ON STACK
1336 002462 013746 001356 MOV TRAPPC, -(KSP)
    
```

1337 002466 000006
1338

RTT

.RETURN FROM INTERRUPT OR ABORT

```

1339          SBTTL
1340          SBTTL ##### STARTING POINT OF TEST #####
1341          SBTTL ##### STARTING ADDRESS OF 200 #####
1342          020000          =20000
1343
1344          020000          START
1345          SBTTL INITIALIZE THE COMMON TAGS
1346          ..CLEAR THE COMMON TAGS (SCMTAG) AREA
1347          020000 012706 001100          MOV      @SCMTAG,R6          ..FIRST LOCATION TO BE CLEARED
1348          020004 005026          CLR      (R6)+          ..CLEAR MEMORY LOCATION
1349          020006 022706 001140          CMP      @SWR,R6          ..DONE?
1350          020012 001374          BNE     -6          ..LOOP BACK IF NO
1351          020014 012706 001100          MOV      @STACK,SP          ..SETUP THE STACK POINTER
1352          ..INITIALIZE A FEW VECTORS
1353          020020 012737 035040 000020          MOV      @SCOPE,@IOTVEC          ..IOT VECTOR FOR SCOPE ROUTINE
1354          020026 012737 000340 000022          MOV      @340,@IOTVEC+2          ..LEVEL 7
1355          020034 012737 035320 000030          MOV      @ERROR,@EMTVEC          ..EMT VECTOR FOR ERROR ROUTINE
1356          020042 012737 000340 000032          MOV      @340,@EMTVEC+2          ..LEVEL 7
1357          020050 012737 041302 000034          MOV      @STRAP,@TRAPVEC          ..TRAP VECTOR FOR TRAP CALLS
1358          020056 012737 000340 000036          MOV      @340,@TRAPVEC+2          ..LEVEL 7
1359          020064 012737 041370 000024          MOV      @SPWRDN,@PWRVEC          ..POWER FAILURE VECTOR
1360          020072 012737 000340 000026          MOV      @340,@PWRVEC+2          ..LEVEL 7
1361          020100 013737 034606 034600          MOV      @ENDCT,@EOPCT          ..SETUP END-OF-PROGRAM COUNTER
1362          020106 005037 001212          CLR      @TIMES          ..INITIALIZE NUMBER OF ITERATIONS
1363          020112 005037 001214          CLR      @ESCAPE          ..CLEAR THE ESCAPE ON ERROR ADDRESS
1364          020116 112737 000001 001115          MOVB   @1,@SERMAX          ..ALLOW ONE ERROR PER TEST
1365          ..INITIALIZE THE "T-BIT" TRAP VECTOR. THEN LOAD LOCATION "SRTN", IN
1366          ..THE "END-OF-PASS" (EOP) ROUTINE, WITH A "RTI" OR "RTT"
1367          020124 012737 035024 000014          MOV      @SRTN,@TBITVEC          ..SET "T" BIT VECTOR TO SRTN
1368          020132 012737 000340 000016          MOV      @340,@TBITVEC+2          ..LEVEL 7
1369          020140 012737 000002 035024          MOV      @RTI,@SRTN          ..SET SRTN TO A RTI
1370          020146 012737 020174 000010          MOV      @655,@RESVEC          ..TRY TO DO A RTT
1371          020154 005046          CLR      -(SP)          ..DUMMY PS
1372          020156 012746 020164          MOV      @645,-(SP)          ..AND PC
1373          020162 000006          RTT          ..TRY THE RTT
1374          020164 012737 000006 035024 645          MOV      @RTT,@SRTN          ..RTT IS LEGAL--SET SRTN TO A RTT
1375          020172 000402          BR      665
1376          020174 062706 000010          655          ADD      @10,SP          ..RTT ILLEGAL--CLEAN OFF THE STACK
1377          020200 012737 000012 000010 665          MOV      @RESVEC+2,@RESVEC          ..RESTORE TRAP CATCHER
1378          020206 005037 035032          CLR      @TBIT          ..CLEAR "T" BIT SWITCH
1379          020212 012737 020212 001106          MOV      @,@SLPADR          ..INITIALIZE THE LOOP ADDRESS FOR SCOPE
1380          020220 012737 020220 001110          MOV      @,@SLPERR          ..SETUP THE ERROR LOOP ADDRESS
1381          ..SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1382          ..EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER
1383          020226 013746 000004          MOV      @ERRVEC,-(SP)          ..SAVE ERROR VECTOR
1384          020232 012737 020266 000004          MOV      @675,@ERRVEC          ..SET UP ERROR VECTOR
1385          020240 012737 177570 001140          MOV      @DSWR,@SWR          ..SETUP FOR A HARDWARE SWICH REGISTER
1386          020246 012737 177570 001142          MOV      @0DISP,@DISPLAY          ..AND A HARDWARE DISPLAY REGISTER
1387          020254 022777 177777 160656          CMP      @-1,@SWR          ..TRY TO REFERENCE HARDWARE SWR
1388          020262 001012          BNE     695          ..BRANCH IF NO TIMEOUT TRAP OCCURRED
1389          ..AND THE HARDWARE SWR IS NOT = -1
1390          020264 000403          BR      685          ..BRANCH IF NO TIMEOUT
1391          020266 012716 020274          675          MOV      @685,(SP)          ..SET UP FOR TRAP RETURN
1392          020272 000002          RTI
1393          020274 012737 000176 001140 685          MOV      @SWREG,@SWR          ..POINT TO SOFTWARE SWR
1394          020302 012737 000174 001142          MOV      @DISPREG,@DISPLAY
    
```

```

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

```

1436
1437
1438
1439
1440
1441
1442
1443
1444
1445 020550 000004
1446 020552 012737 020562 001110
1447 020560 005000
1448 020562 005001
1449 020564 106400
1450 020566 106701
1451 020570 042701 177437
1452 020574 020001
1453 020576 001401
1454 020600 104003
1455
1456
1457
1458 020602 062700 000040
1459 020606 022700 000400
1460 020612 001363
1461 020614 012737 020552 001110
1462
1463
1464
1465
1466
1467
1468
1469 020622 000004
1470 020624 012737 020634 001110
1471 020632 005000
1472 020634 005037 177776
1473 020640 050037 177776
1474 020644 013701 177776
1475 020660 042701 007777
1476 020664 020001
1477 020666 001403
1478 020660 005037 177776
1479 020664 104004
1480
1481
1482
1483 020666 062700 010000
1484 020672 001360
1485 020674 012737 020624 001110
1486 020702 005037 177776
1487
1488
1489
1490
1491
    .. *****
    *TEST 1          PSW PRIORITY BIT TEST
    *
    * THIS TEST READS AND WRITES THE PROCESSOR STATUS WORD (<7 5> "PRIORITY BITS"
    * TO SEE THAT SOME OF THE BASIC "DATA PATH" LOGIC IS WORKING
    *
    .. *****
    TST1  SCOPE
    15    MOV     #25, $LPERR      ; SET LOOP ON ERROR POINTER TO 25
        CLR     RO              ; INITIALIZE RO WITH PRIORITY=0 DATA
        CLR     R1              ; PREPARE R1 TO ACCEPT DATA READ
    25    MTPS   RO              ; WRITE PRIORITY BITS IN THE PSW
        MFPS   R1              ; READ BACK THE LOW BYTE OF PSW
        BIC    #177437, R1     ; MASK OFF EVERYTHING EXCEPT PRIORITY BITS
        CMP    RO, R1          ; WAS CORRECT PRIORITY SET IN THE PSW?
        BEQ    35              ; BRANCH IF YES
        ERROR  3                ; PRIORITY BITS SET WRONG IN PSW
                                ; FOR TIGHTER SCOPE LOOP
                                ; REPLACE ERROR CALL WITH
                                ; "BR 25" = 000770
    35    ADD     #40, RO        ; CHANGE DATA TO NEXT PRIORITY
        CMP    #400, RO        ; HAVE PRIORITIES 0-7 ALL BEEN CHECKED?
        BNE    25              ; BRANCH IF NO
        MOV    #15, $LPERR     ; RESET LOOP ON ERROR POINTER TO 15
    .. *****
    *TEST 2          PSW MODE BIT TEST
    *
    * THIS TEST READS AND WRITES THE PROCESSOR STATUS WORD (<15 12> "MODE BITS"
    * TO FURTHER CHECK THE BASIC CPU DATA PATHS
    *
    .. *****
    TST2  SCOPE
    15    MOV     #25, $LPERR      ; SET LOOP ON ERROR POINTER TO 25
        CLR     RO              ; INITIALIZE RO WITH MODE BITS = 0000
        CLR     PSW            ; INITIALIZE PSW
    25    BIS     RO, PSW         ; BIT SET THE PSW MODE BITS WITH RO
        MOV    PSW, R1         ; READ BACK THE CONTENTS OF THE PSW
        BIC    #007777, R1     ; MASK OFF EVERYTHING EXCEPT THE MODE BITS
        CMP    RO, R1          ; WERE THE MODE BITS SET CORRECTLY?
        BEQ    35              ; BRANCH IF YES
        CLR     PSW            ; CLEAR PSW FOR ERROR REPORT
        ERROR  4                ; MODE BITS SET WRONG IN PSW
                                ; FOR TIGHTER SCOPE LOOP
                                ; REPLACE ERROR CALL WITH
                                ; "BR 25" = 000763
    35    ADD     #10000, RO     ; CHANGE MODE BIT DATA
        BNE    25              ; BRANCH IF STILL MORE COMBINATIONS
        MOV    #15, $LPERR     ; RESET LOOP ON ERROR POINTER TO 15
        CLR     PSW            ; RESET PSW BEFORE LEAVING
    .. *****
    *TEST 3          BYTE ADDRESSING TEST FOR PSW
    *
    * THIS TEST WRITES THE HIGH AND LOW BYTES OF THE PROCESSOR STATUS WORD
    
```

```

1492 .X AND READS THEM BACK TO BE SURE THEY CAN BE WRITTEN INDEPENDENTLY
1493 .X THIS CHECKS THE PSW PORTION OF THE ADDRESS DETECTION LOGIC.
1494 .X
1495 .X *****
1496 020706 000004 TST3 SCOPE
1497 020710 012737 020716 001110 15 MOV #25,SLPERR ;SET LOOP ON ERROR POINTER TO 25
1498 020716 005037 177776 25 CLR PSW ;CLEAR THE PSW
1499 020722 012700 000360 MOV #360,RO ;PUT THE HIGH BYTE DATA INTO RO
1500 020726 110037 177777 MOVB RO,PSW+1 ;WRITE THE HIGH BYTE OF THE PSW
1501 020732 013701 177776 MOV PSW,R1 ;READ BACK THE ENTIRE PSW
1502 020736 042701 007437 BIC #007437,R1 ;MASK OFF THE T & CC BITS
1503 020742 000300 SWAB RO ;GET DATA WRITTEN IN HIGH BYTE OF RO
1504 020744 020001 CMP RO,R1 ;WAS THE PSW WRITTEN TO CORRECTLY
1505 020746 001403 BEQ 35 ;BRANCH IF YES
1506 020750 005037 177776 CLR PSW ;CLEAR PSW FOR ERROR REPORT
1507 020754 104005 ERROR 5 ;LOW BYTE EFFECTED BY WRITE TO HIGH BYTE OF PSW
1508 ;FOR TIGHTER SCOPE LOOP
1509 ;REPLACE ERROR CALL WITH
1510 ;"BR 25" = 000760
1511 020756 012737 020764 001110 35 MOV #45,SLPERR ;SET LOOP ON ERROR POINTER TO 45
1512 020764 005037 177776 45 CLR PSW ;CLEAR THE PSW
1513 020770 012700 000340 MOV #340,RO ;PUT THE LOW BYTE DATA INTO RO
1514 020774 110037 177776 MOVB RO,PSW ;WRITE THE LOW BYTE OF THE PSW
1515 021000 013701 177776 MOV PSW,R1 ;READ BACK THE ENTIRE PSW
1516 021004 042701 007437 BIC #007437,R1 ;MASK OFF THE T&CC BITS
1517 021010 020001 CMP RO,R1 ;WAS PSW WRITTEN TO CORRECTLY
1518 021012 001403 BEQ 55 ;BRANCH IF YES
1519 021014 005037 177776 CLR PSW ;CLEAR PSW FOR ERROR REPORT
1520 021020 104005 ERROR 5 ;HIGH BYTE EFFECTED BY WRITE TO LOW BYTE OF PSW
1521 ;FOR TIGHTER SCOPE LOOP
1522 ;REPLACE ERROR CALL WITH
1523 ;"BR 25" = 000736
1524 021022 012737 020710 001110 55 MOV #15,SLPERR ;RESET LOOP ON ERROR POINTER TO 15
1525
1526 .X *****
1527 .X TEST 4 TEST AND SETUP OF STACK POINTERS
1528 .X
1529 .X THIS TEST SETS THE USER AND KERNEL STACK POINTERS FOR THE
1530 .X REST OF THE PROGRAM AND MAKES SURE THEY ARE INDEPENDENT OF
1531 .X EACH OTHER KERNEL R6 IS SET TO 1100, USER R6 IS SET TO 700, THEN
1532 .X KERNEL R6 IS READ TO BE SURE ITS STILL 1100
1533 .X
1534 .X *****
1535 021030 000004 TST4 SCOPE
1536 021032 005037 177776 CLR PSW ;GO TO KERNEL MODE
1537 021036 012706 001100 MOV #KERSTK,KSP ;SET KERNEL STACK POINTER TO 1100
1538 021042 012737 140000 177776 MOV #140000,PSW ;GO TO USER MODE
1539 021060 012706 000700 MOV #USESTK,USP ;SET USER STACK POINTER TO 700
1540 021064 005037 177776 CLR PSW ;BACK TO KERNEL MODE
1541 021060 022706 001100 CMP #KERSTK,KSP ;IS KERNEL R6 STILL 1100?
1542 021064 001404 BEQ TST5 ;BRANCH IF KERNEL R6 IS OKAY
1543 021066 012700 001100 MOV #KERSTK,RO ;SAVE DATA WRITTEN FOR ERROR REPORT
1544 021072 010601 MOV KSP,R1 ;SAVE DATA READ AFTER USER R6 WAS WRITTEN
1545 021074 104006 ERROR 6 ;KERNEL R6 CHANGED BY WRITING USER R6
1546 ;FOR TIGHTER SCOPE LOOP
1547 ;REPLACE ERROR CALL WITH
    
```


.000756

```

1548
1549
1550 .. *****
1551 .. *
1552 .. * THE NEXT FIVE (5) TESTS WILL TRY TO ADDRESS ALL OF THE
1553 .. * MEMORY MANAGEMENT REGISTERS (SR0, SR1, SR2, KERNEL & USER PAR/PDR'S)
1554 .. * EVERY TIME A REGISTER TIMES OUT ITS ADDRESS WILL BE REPORTED.
1555 .. * AT THE END OF EACH TEST A SUMMARY OF THE ADDRESSES THAT TIMED
1556 .. * OUT DURING THAT TEST IS GIVEN. THE RESULTS OF "AND-ING" AND "OR-ING"
1557 .. * THEIR ADDRESSES IS GIVEN TO SHOW WHICH ADDRESS LINES MAY BE
1558 .. * STUCK AT 0 OR 1. THE PAR/PDR ADDRESS AND KT MUX'S ARE THE
1559 .. * THINGS BEING CHECKED.
1560 .. *
1561 .. *****
1562
1563 .. *****
1564 .. *
1565 .. * TEST 5 SR0, SR1, SR2 TIMEOUT TEST
1566 .. *
1567 .. * THIS TEST ADDRESSES THE MEMORY MANAGEMENT STATUS REGISTERS
1568 .. * 0, 1, AND 2. STATUS REG. 1 IS NOT USED BUT SHOULD STILL
1569 .. * RESPOND TO ITS UNIBUS ADDRESS. DATA WILL BE WRITTEN OR READ
1570 .. * FROM THESE REGISTERS IN LATER TESTS. THIS TEST JUST CHECK
1571 .. * FOR A RESPONSE.
1572 .. *
1573 .. *****
1574 021076 000004 TST5 SCOPE
1575
1576 021100 012737 021142 001110 15 MOV #25, $LPERR ; SET LOOP ON ERROR POINTER TO 25
1577 021106 012737 021200 000004 MOV #55, @#4 ; SET TIMEOUT VECTOR TO 55
1578 021114 012700 177572 MOV #SR0, R0 ; LOAD R0 WITH ADDRESS OF FIRST REG
1579 021120 012701 000003 MOV #3, R1 ; LOAD R1 WITH THE LOOP COUNT
1580 021124 012737 177777 001374 MOV #-1, ANDADR ; INITIALIZE "AND" OF ADDRS. LOC
1581 021132 005037 001376 CLR ORADR ; INITIALIZE "OR" OF ADDRS. LOC
1582 021136 005037 001400 CLR TONUM ; INITIALIZE "TIMEOUTS" COUNTER
1583 021142 005710 25 TST (R0) ; TRY ADDRESSING A STATUS REGISTER
1584 ; IF IT TIMES OUT GO TO 55
1585 021144 062700 000002 35 ADD #2, R0 ; PUT NEXT ADDRESS IN R0
1586 021150 077104 SOB R1, 25 ; LOOP BACK TO 25 UNTIL ALL TESTED
1587 021152 012737 021100 001110 MOV #15, $LPERR ; RESET LOOP ON ERROR POINTER TO 15
1588 021160 005737 001400 TST TONUM ; DID ANY OF THE STATUS REG.S TIMEOUT?
1589 021164 001401 BEQ 45 ; BRANCH IF NO
1590 021166 104010 ERROR 10 ; SUMMARY OF STATUS REG. TIMEOUTS
1591 021170 012737 002332 000004 45 MOV #TIMERR, @#4 ; RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1592 021176 000414 BR TST6 ; GO TO NEXT TEST
1593
1594 021200 062706 000004 55 ADD #4, KSP ; CLEAN UP THE STACK
1595 021204 104007 ERROR 7 ; ONE OF THE STATUS REGS TIMED OUT
1596 ; FOR TIGHTER SCOPE LOOP
1597 ; REPLACE ERROR CALL WITH
1598 ; "BR 25" = 000756
1599 021206 010002 MOV R0, R2 ; LOAD THE ADDRESS THAT TIMED OUT INTO R2
1600 021210 050237 001376 BIS R2, ORADR ; "OR" IT WITH OTHER ADDRS THAT TIMED OUT
1601 021214 005102 COM R2 ; "AND" IT WITH OTHER ADDRS THAT TIMED OUT
1602 021216 040237 001374 BIC R2, ANDADR
1603 021222 005237 001400 INC TONUM ; INCREMENT THE TIMEOUT COUNTER
    
```

```

1604 021226 000746 BR 35 , BRANCH BACK TO TEST THE NEXT ADDR
1605
1606 , , *****
1607 , *TEST 6 KERNEL PAR'S TIMEOUT TEST
1608 , *
1609 , * THIS TEST ADDRESSES THE EIGHT (8) KERNEL PAGE ADDRESS
1610 , * REGISTERS (KIPAR0-KIPAR7) AND CHECKS THAT SOMETHING
1611 , * RESPONDS TO THEIR ADDRESSES.
1612 , *
1613 , , *****
1614 021230 000004 TST6 SCOPE
1615
1616 021232 012737 021274 001110 15 MOV #25,SLPERR , SET LOOP ON ERROR POINTER TO 25
1617 021240 012737 021332 000004 MOV #55,#4 , SET TIMEOUT VECTOR TO 55
1618 021246 012700 172340 MOV #KIPAR0,R0 , LOAD R0 WITH ADDRESS OF FIRST REG
1619 021252 012737 000010 MOV #10,R1 , LOAD R1 WITH LOOP COUNT (8)
1620 021256 012737 177777 001374 MOV #-1,ANDADR , INITIALIZE "AND" OF ADDR. LOC
1621 021264 005037 001376 CLR ORADR , INITIALIZE "OR" OF ADDR. LOC
1622 021270 005037 001400 CLR TONUM , INITIALIZE "TIMEOUTS" COUNTER
1623 021274 005710 25 TST (R0) , TRY ADDRESSING A KIPAR
1624 , IF IT TIMES OUT, WILL GO TO 55
1625 021276 062700 000002 35 ADD #2,R0 , PUT NEXT KIPAR ADDRESS IN R0
1626 021302 077104 SOB R1,25 , LOOP BACK TO 25 UNTIL ALL TESTED
1627 021304 012737 021232 001110 MOV #15,SLPERR , RESET LOOP ON ERROR POINTER TO 15
1628 021312 005737 001400 TST TONUM , DID ANY OF THE KIPARS TIME OUT?
1629 021316 001401 BEQ 45 , BRANCH IF NO
1630 021320 104010 ERROR 10 , SUMMARY OF KIPAR TIMEOUTS
1631 021322 012737 002332 000004 45 MOV #TIMERR,#4 , RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1632 021330 000414 BR TST7 , GO TO NEXT TEST
1633
1634 021332 062706 000004 55 ADD #4,KSP , CLEAN UP THE STACK
1635 021336 104007 ERROR 7 , ONE OF THE KIPARS TIMED OUT
1636 , FOR TIGHTER SCOPE LOOP
1637 , REPLACE ERROR CALL WITH
1638 , "BR 25" = 000756
1639 021340 010002 MOV R0,R2 , LOAD THE ADDRESS THAT TIMED OUT INTO R2
1640 021342 050237 001376 BIS R2,ORADR , "OR" IT WITH OTHER ADDRS THAT TIMED OUT
1641 021346 005102 COM R2 , "AND" IT WITH OTHER ADDRS THAT TIMED OUT
1642 021350 040237 001374 BIC R2,ANDADR
1643 021354 005237 001400 INC TONUM , INCREMENT THE TIMEOUT COUNTER
1644 021360 000746 BR 35 , BRANCH BACK TO TEST THE NEXT KIPAR
1645
1646 , , *****
1647 , *TEST 7 KERNEL PDR'S TIMEOUT TEST
1648 , *
1649 , * THIS TEST ADDRESSES THE EIGHT (8) KERNEL PAGE DESCRIPTOR
1650 , * REGISTERS (KIPDR0-KIPDR7) AND CHECKS THAT SOMETHING
1651 , * RESPONDS TO THEIR ADDRESSES
1652 , *
1653 , , *****
1654 021362 000004 TST7: SCOPE
1655
1656 021364 012737 021426 001110 15 MOV #25,SLPERR , SET LOOP ON ERROR POINTER TO 25
1657 021372 012737 021464 000004 MOV #55,#4 , SET TIMEOUT VECTOR TO 55
1658 021400 012700 172300 MOV #KIPDR0,R0 , LOAD R0 WITH ADDRESS OF FIRST REG
1659 021404 012701 000010 MOV #10,R1 , LOAD R1 WITH LOOP COUNT (8)
    
```

1660	021410	012737	177777	001374		MOV	#-1, ANDADR	, INITIALIZE "AND" OF ADDR LOC
1661	021416	005037	001376			CLR	ORADR	, INITIALIZE "OR" OF ADDR LOC
1662	021422	005037	001400			CLR	TONUM	, INITIALIZE "TIMEOUTS" COUNTER
1663	021426	005710		25		TST	(R0)	, TRY ADDRESSING A KIPDR
1664								, IF IT TIMES OUT, WILL GO TO 55
1665	021430	062700	000002		35	ADD	#2, R0	, PUT NEXT KIPDR ADDRESS IN R0
1666	021434	077104				SOB	R1, 25	, LOOP BACK TO 25 UNTIL ALL TESTED
1667	021436	012737	021364	001110		MOV	#15, SLPERR	, RESET LOOP ON ERROR POINTER TO 15
1668	021444	005737	001400			TST	TONUM	, DID ANY OF THE KIPDRS TIME OUT?
1669	021450	001401				BEQ	45	, BRANCH IF NO
1670	021452	104010				ERROR	10	, SUMMARY OF KIPDR TIMEOUTS
1671	021454	012737	002332	000004	45	MOV	#TIMERR, 2#4	, RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1672	021462	000414				BR	TST10	, GO TO NEXT TEST
1673								
1674	021464	062706	000004		55	ADD	#4, KSP	, CLEAN UP THE STACK
1675	021470	104007				ERROR	7	, ONE OF THE KIPDRS TIMED OUT
1676								, FOR TIGHTER SCOPE LOOP
1677								, REPLACE ERROR CALL WITH
1678								, "BR 25" = 000756
1679	021472	010002				MOV	R0, R2	, LOAD THE ADDRESS THAT TIMED OUT INTO R2
1680	021474	050237	001376			BIS	R2, ORADR	, "OR" IT WITH OTHER ADDRS THAT TIMED OUT
1681	021500	005102				COM	R2	, "AND" IT WITH OTHER ADDRS THAT TIMED OUT
1682	021502	040237	001374			BIC	R2, ANDADR	
1683	021506	005237	001400			INC	TONUM	, INCREMENT THE TIMEOUT COUNTER
1684	021512	000746				BR	35	, BRANCH BACK TO TEST THE NEXT KIPDR
1685								
1686								
1687								
1688								
1689								
1690								
1691								
1692								
1693								
1694	021514	000004				TST10	SCOPE	
1695								
1696	021516	012737	021560	001110	15	MOV	#25, SLPERR	, SET LOOP ON ERROR POINTER TO 25
1697	021524	012737	021616	000004		MOV	#55, 2#4	, SET TIMEOUT VECTOR TO 55
1698	021532	012700	177640			MOV	#UIPAR0, R0	, LOAD R0 WITH ADDRESS OF FIRST REG
1699	021536	012701	000010			MOV	#10, R1	, LOAD R1 WITH LOOP COUNT (8)
1700	021542	012737	177777	001374		MOV	#-1, ANDADR	, INITIALIZE "AND" OF ADDR LOC
1701	021550	005037	001376			CLR	ORADR	, INITIALIZE "OR" OF ADDR LOC
1702	021554	005037	001400			CLR	TONUM	, INITIALIZE "TIMEOUTS" COUNTER
1703	021560	005710		25		TST	(R0)	, TRY ADDRESSING A UIPAR
1704								, IF IT TIMES OUT, WILL GO TO 55
1705	021562	062700	000002		35	ADD	#2, R0	, PUT NEXT UIPAR ADDRESS IN R0
1706	021566	077104				SOB	R1, 25	, LOOP BACK TO 25 UNTIL ALL TESTED
1707	021570	012737	021516	001110		MOV	#15, SLPERR	, RESET LOOP ON ERROR POINTER TO 15
1708	021576	005737	001400			TST	TONUM	, DID ANY OF THE UIPARS TIME OUT?
1709	021602	001401				BEQ	45	, BRANCH IF NO
1710	021604	104010				ERROR	10	, SUMMARY OF UIPAR TIMEOUTS
1711	021606	012737	002332	000004	45	MOV	#TIMERR, 2#4	, RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1712	021614	000414				BR	TST11	, GO TO NEXT TEST
1713								
1714	021616	062706	000004		55	ADD	#4, KSP	, CLEAN UP THE STACK
1715	021622	104007				ERROR	7	, ONE OF THE UIPARS TIMED OUT

.. *****
 , XTEST 10 USER PAR'S TIMEOUT TEST

, X
 , X THIS TEST ADDRESSES THE EIGHT (8) USER PAGE ADDRESS
 , X REGISTERS (UIPAR0-UIPAR7) AND CHECKS THAT SOMETHING
 , X RESPONDS TO THEIR ADDRESSES

.. *****
 TST10 SCOPE

```

1716                                     ,FOR TIGHTER SCOPE LOOP
1717                                     ,REPLACE ERROR CALL WITH
1718                                     , "BR 25" = 000756
1719 021624 010002                       MOV     R0,R2      ,LOAD THE ADDRESS THAT TIMED OUT INTO R2
1720 021626 050237 001376                BIS     R2,ORADR , "OR" IT WITH OTHER ADDRS THAT TIMED OUT
1721 021632 005102                       COM     R2        , "AND" IT WITH OTHER ADDRS THAT TIMED OUT
1722 021634 040237 001374                BIC     R2,ANDADR
1723 021640 005237 001400                INC     TONUM    , INCREMENT THE TIMEOUT COUNTER
1724 021644 000746                       BR      35      , BRANCH BACK TO TEST THE NEXT UIPAR
1725
1726                                     , , *****
1727 ;*TEST 11          USER PDR'S TIMEOUT TEST
1728 ;*
1729 ;*          THIS TEST ADDRESSES THE EIGHT (8) USER PAGE DESCRIPTOR
1730 ;*          REGISTERS (UIPDR0-UIPDR7) AND CHECKS THAT SOMETHING
1731 ;*          RESPONDS TO THEIR ADDRESSES
1732 ;*
1733                                     , , *****
1734 021646 000004                       TST11. SCOPE
1735
1736 021650 012737 021712 001110 15.     MOV     #25,$LPERR , SET LOOP ON ERROR POINTER TO 25
1737 021656 012737 021750 000004         MOV     #55,$#4    , SET TIMEOUT VECTOR TO 55
1738 021664 012700 177600                 MOV     #UIPDR0,R0 , LOAD R0 WITH ADDRESS OF FIRST REG
1739 021670 012701 000010                 MOV     #10,R1     , LOAD R1 WITH LOOP COUNT (8)
1740 021674 012737 177777 001374         MOV     #-1,ANDADR , INITIALIZE "AND" OF ADDR LOC
1741 021702 005037 001376                 CLR     ORADR      , INITIALIZE "OR" OF ADDR. LOC
1742 021706 005037 001400                 CLR     TONUM     , INITIALIZE "TIMEOUTS" COUNTER
1743 021712 005710                       25     TST     (R0) , TRY ADDRESSING A UIPDR
1744                                     , IF IT TIMES OUT, WILL GO TO 55
1745 021714 062700 000002                 35     ADD     #2,R0 , PUT NEXT UIPDR ADDRESS IN R0
1746 021720 077104                       SOB     R1,25     , LOOP BACK TO 25 UNTIL ALL TESTED
1747 021722 012737 021650 001110         MOV     #15,$LPERR , RESET LOOP ON ERROR POINTER TO 15
1748 021730 005737 001400                 TST     TONUM    , DID ANY OF THE UIPDRS TIME OUT?
1749 021734 001401                       BEQ     45        , BRANCH IF NO
1750 021736 104010                       ERROR  10        , SUMMARY OF UIPDR TIMEOUTS
1751 021740 012737 002332 000004 45     MOV     #TIMERR,$#4 , RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1752 021746 000414                       BR      TST12    , GO TO NEXT TEST
1753
1754 021750 062706 000004                 55     ADD     #4,KSP   , CLEAN UP THE STACK
1755 021754 104007                       ERROR  7         , ONE OF THE UIPDRS TIMED OUT
1756                                     , FOR TIGHTER SCOPE LOOP
1757                                     , REPLACE ERROR CALL WITH
1758                                     , "BR 25" = 000756
1759 021756 010002                       MOV     R0,R2      , LOAD THE ADDRESS THAT TIMED OUT INTO R2
1760 021760 050237 001376                BIS     R2,ORADR , "OR" IT WITH OTHER ADDRS THAT TIMED OUT
1761 021764 005102                       COM     R2        , "AND" IT WITH OTHER ADDRS THAT TIMED OUT
1762 021766 040237 001374                BIC     R2,ANDADR
1763 021772 005237 001400                INC     TONUM    , INCREMENT THE TIMEOUT COUNTER
1764 021776 000746                       BR      35      , BRANCH BACK TO TEST THE NEXT UIPDR
1765
1766                                     , , *****
1767 ;*TEST 12          SRO(15:13) BIT TEST & SR2 TEST
1768 ;*
1769 ;*          THIS TEST CHECKS BITS <15:13> OF STATUS REGISTER 0 TO SEE
1770 ;*          THAT EACH CAN BE SET AND CLEARED AND THAT A "RESET" WILL
1771 ;*          CLEAR ALL OF THEM  A TEST OF THESE THREE ERROR BITS CHECKS
    
```

```

1772 .; PART OF SRO, THE SRO MUX AND THE KTMUX THE REST OF THE
1773 .; BITS IN SRO WILL BE CHECKED LATER
1774 .; ALSO CHECK THAT SR2 IS TRACKING WITH MEM MGMT
1775 .; OFF BUT LOCKS UP WHEN ANY OF SRO ERROR BITS SET
1776 .;
1777 .; *****
1778 022000 000004 TST12 SCOPE
1779
1780 022002 012700 177572 15 MOV #SRO,RO ; LOAD ADDRESS OF SRO INTO RO
1781 022006 012710 160000 MOV #160000,(RO) ; SET BITS <15 13> IN SRO (ERROR BITS)
1782 022012 000005 RESET ; ISSUE AND "INIT" SIGNAL
1783 022014 011001 MOV (RO),R1 ; READ SRO INTO R1 TO SEE IF CLEAR
1784 022016 001404 BEQ 25 ; BRANCH IF SRO<15:13> CLEARED BY "INIT"
1785 022020 104011 ERROR 11 ; SRO<15:13> NOT CLEARED BY A "RESET"
1786 ; FOR TIGHTER SCOPE LOOP
1787 ; REPLACE ERROR CALL WITH
1788 ; "BR 15" = 000770
1789 022022 012737 022030 001110 MOV #25,$LPERR ; SET LOOP ON ERROR POINTER TO 25
1790 022030 013737 177576 001370 25 MOV SR2,WASSR2 ; READ CONTENTS OF SR2
1791 022036 012701 022030 MOV #25,R1 ; LOAD EXPECTED CONTENTS INTO R1
1792 022042 020137 001370 CMP R1,WASSR2 ; IS SR2 TRACKING?
1793 022046 001401 BEQ 35 ; BRANCH IF YES
1794 022050 104041 ERROR 41 ; SR2 NOT "TRACKING" VIRTUAL ADDRESSES
1795 ; FOR TIGHTER SCOPE LOOP
1796 ; REPLACE ERROR CALL WITH
1797 ; "BR 25" = 000767
1798 022052 012737 022070 001110 35 MOV #45,$LPERR ; SET LOOP ON ERROR POINTER TO 45
1799 022060 012701 100000 MOV #BIT15,R1 ; PUT DATA TO BE WRITTEN IN R1
1800 022064 012703 000003 MOV #3,R3 ; SETUP R3 AS A LOOP COUNTER
1801 022070 005010 45 CLR (RO) ; CLEAR SRO
1802 022072 050110 55 BIS R1,(RO) ; SET ONE OF THE ERROR BITS IN SRO
1803 022074 011002 MOV (RO),R2 ; READ SRO INTO R2
1804 022076 020102 CMP R1,R2 ; DID RIGHT ERROR BIT GET SET?
1805 022100 001401 BEQ 65 ; BRANCH IF YES
1806 022102 104012 ERROR 12 ; BITS WERE SET WRONG IN SRO
1807 ; FOR TIGHTER SCOPE LOOP
1808 ; REPLACE ERROR CALL WITH
1809 ; "BR 45" = 000772
1810 022104 012704 022072 65 MOV #55,R4 ; LOAD EXPECTED CONTENTS OF SR2 IN R4
1811 022110 013737 177576 001370 MOV SR2,WASSR2 ; READ SR2
1812 022116 020437 001370 CMP R4,WASSR2 ; DID SR2 LOCK UP WHEN ERROR
1813 ; BIT SET IN SR1?
1814 022122 001401 BEQ 75 ; BRANCH IF YES
1815 022124 104064 ERROR 64 ; SR2 DID NOT LOCK UP
1816 ; FOR TIGHTER SCOPE LOOP
1817 ; REPLACE ERROR CALL WITH
1818 ; "BR 45" = 000761
1819 022126 006001 75 OR R1 ; CHANGE DATA TO CHECK NEXT ERROR BIT
1820 022130 077321 SOB R3,45 ; LOOP BACK UNTIL <15:13> ALL TESTED
1821 022132 005010 CLR (RO) ; CLEAR SRO BEFORE LEAVING
1822 022134 012737 022002 001110 MOV #15,$LPERR ; RESET LOOP ON ERROR POINTER TO 15
1823
1824 .; *****
1825 ;*TEST 13 SRO & PSW DUAL ADDRESSING TEST
1826 .;
1827 .; THIS TEST CHECKS MORE OF THE ADDRESS DETECTION LOGIC BY
    
```

```

1828 ., VERIFYING THAT STATUS REGISTER 0 IS NOT EFFECTED BY WRITING
1829 ., TO THE PSW AND THAT THE LOW BYTE OF STATUS REGISTER 0
1830 ., IS NOT EFFECTED BY WRITING TO ITS HIGH BYTE THIS IS TO
1831 ., SEE IF ADJACENT OUTPUTS ARE SHORTED ON THE ADDRESS DET LOGIC
1832 .,
1833 ., *****
1834 022142 000004 TST13 SCOPE
1835
1836 022144 005037 177776 15 CLR PSW ;CLEAR THE PSW
1837 022150 005037 177572 CLR SRO ;CLEAR STATUS REGISTER 0
1838 022154 106427 000340 MTPS #340 ;SET PRIORITY 7 IN LOW BYTE OF PSW
1839 022160 013700 177572 MOV SRO,RO ;READ STATUS REGISTER 0
1840 022164 001401 BEQ ZS ;BRANCH IF IT WAS STILL 0
1841 022166 104013 ERROR 13 ;SRO EFFECTED BY A WRITE TO THE PSW
1842 ;FOR TIGHTER SCOPE LOOP
1843 ;REPLACE ERROR CALL WITH
1844 ;"BR 15" = 000767
1845 022170 005037 177572 25 CLR SRO ;BE SURE SRO IS 0 BEFORE LEAVING
1846 022174 005037 177776 CLR PSW ;BE SURE PSW IS 0 BEFORE LEAVING
1847
1848 ., *****
1849 ., TEST 14 TEST THAT SR1 READS ALL ZEROS
1850 .,
1851 ., THIS TESTS CHECKS THAT EVEN THOUGH STATUS REGISTER 1
1852 ., IS NON-EXISTENT, ITS ADDRESS SHOULD RESPOND WITH ALL ZEROS,
1853 ., THEREBY CHECK ANOTHER PORTION OF THE ADDRESS DETECTION LOGIC
1854 .,
1855 ., *****
1856 022200 000004 TST14: SCOPE
1857 022202 012700 177777 MOV #-1,RO ;FILL RO WITH ALL ONES
1858 022206 013700 177574 MOV SR1,RO ;READ SR1 INTO RO
1859 022212 001401 BEQ TST15 ;BRANCH IF SR1 READS ALL ZEROS
1860 022214 104014 ERROR 14 ;SR1 DID NOT READ ALL ZEROS
1861 ;FOR TIGHTER SCOPE LOOP
1862 ;REPLACE ERROR CALL WITH
1863 ;000772
1864
1865 ., *****
1866 ., TEST 15 BIT TEST OF KERNEL & USER PAR'S
1867 .,
1868 ., THE FOLLOWING TEST CHECKS THE BITS <11:00> OF BOTH THE KERNEL
1869 ., AND USER PAGE ADDRESS REGISTERS. A "0" IS ROTATED THRU
1870 ., THE REGISTERS FROM LEFT TO RIGHT. THIS CHECKS THE OPERATION
1871 ., OF THE PAR/PDR ADDRESS MUX, THE KT MUX, AND THE PAR
1872 ., OUTPUT DATA LINES.
1873 .,
1874 ., *****
1875 022216 000004 TST15: SCOPE
1876
1877 022220 012700 172340 15: MOV #KIPAR,RO ;LOAD ADDRESS OF FIRST PAR IN RO
1878 022224 012703 000010 25 MOV #10,R3 ;SETUP R3 TO COUNT 8 PAR'S
1879 022230 012737 022236 001110 MOV #35,SLPERR ;SET LOOP ON ERROR POINTER TO 35
1880 022236 005010 35 CLR (RO) ;CLEAR THE PAR
1881 022240 011001 MOV (RO),R1 ;READ THE PAR INTO R1
1882 022242 001401 BEQ 45 ;BRANCH IF PAR CLEARED OK
1883 022244 104011 ERROR 11 ;PAR WOULD NOT CLEAR
    
```



```

1911 // *****
1912 ,*TEST 16 BIT TEST OF KERNEL & USER PDR'S
1913 ,*
1914 ,* THE FOLLOWING TEST CHECKS THE BITS <14:8> AND <3:1> OF BOTH THE
1915 ,* KERNEL AND USER PAGE DESCRIPTOR REGISTERS. A "0" IS ROTATED
1916 ,* THRU THE REGISTERS FROM LEFT TO RIGHT. SOME TEST PATTERNS WILL
1917 ,* BE LOADED MORE THAN ONCE DUE TO THE UNUSED BITS IN THE PDR'S
1918 ,* THE PAR/PDR ADDRESS MUX, KTMUX, AND PDR OUTPUT DATA LINES
1919 ,* ARE BEING CHECKED.
1920 ,*
1921 // *****
1922 022340 000004 TST16 SCOPE
1923
1924 022342 012700 172300 15 MOV #KIPDR0,R0 ,LOAD ADDRESS OF FIRST PDR IN R0
1925 022346 012703 000010 25 MOV #10,R3 ,SETUP R3 TO COUNT 8 PDR'S
1926 022352 012737 022360 001110 MOV #35,$LPERR ,SET LOOP ON ERROR POINTER TO 35
1927 022360 005010 35 CLR (R0) ,CLEAR THE PDR
1928 022362 011001 MOV (R0),R1 ,READ THE PDR INTO R1
1929 022364 001401 BEQ 45 ,BRANCH IF PDR CLEARED OK
1930 022366 104011 ERROR 11 ,PDR WOULD NOT CLEAR
1931 ,* FOR TIGHTER SCOPE LOOP
1932 ,* REPLACE ERROR CALL WITH
1933 ,* "BR 35" = 000774
1934 022370 012704 077777 45 MOV #077777,R4 ,LOAD "WALKING 0" TEST PATTERN IN R4
1935 022374 012737 022402 001110 MOV #55,$LPERR ,SET LOOP ON ERROR POINTER TO 55
1936 022402 005010 55 CLR (R0) ,CLEAR THE PDR BEFORE LOADING DATA
1937 022404 010401 MOV R4,R1 ,LOAD DATA INTO R1
1938 022406 042701 100361 BIC #100361,R1 ,MASK UNUSED BITS OUT OF THE DATA
1939 022412 050110 BIS R1,(R0) ,BIT SET THE TEST PATTERN INTO THE PDR
1940 022414 011002 MOV (R0),R2 ,READ THE PDR INTO R2
1941 022416 020102 CMP R1,R2 ,DOES DATA WRITTEN=DATA READ?
1942 022420 001401 BEQ 65 ,BRANCH IF YES
1943 022422 104012 ERROR 12 ,PDR BITS DID NOT SET CORRECTLY
1944 ,* FOR TIGHTER SCOPE LOOP
1945 ,* REPLACE ERROR CALL WITH
1946 ,* "BR 55" = 000767
1947 022424 000261 65 SEC ,SET THE C-BIT FOR THE ROTATE INST
1948 022426 006004 ROR R4 ,ROTATE THE TEST PATTERN IN R4
1949 022430 103764 BCS 55 ,BRANCH BACK IF MORE BITS TO TEST
1950 022432 062700 000002 ADD #2,R0 ,GET NEXT PDR ADDRESS IN R0
1951 022436 077330 SOB R3,$5 ,BRANCH BACK UNTIL ALL PDR'S TESTED
1952 022440 022700 177620 CMP #UIPDR7+2,R0 ,HAVE USER PDR'S BEEN TESTED?
1953 022444 103003 BHIS 75 ,BRANCH IF YES
1954 022446 012700 177600 MOV #UIPDR0,R0 ,LOAD FIRST USER PDR ADDR IN R0
1955 022452 000735 BR 25 ,BRANCH BACK TO TEST USER PDR'S
1956 022454 012737 022342 001110 75 MOV #15,$LPERR ,RESET LOOP ON ERROR PCINTER TO 15
1957 ,* LEAVE TEST WITH ALL WRITABLE BITS IN
1958 ,* ALL PDR'S = 1
1959
1960 // *****
1961 ,*TEST 17 TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PAR S
1962 ,*
1963 ,* THE FOLLOWING TEST WRITES TO BOTH BYTES OF THE KERNEL & USER
1964 ,* PAR'S SEPERATELY TO SEE THAT WRITING TO ONE DOES NOT EFFECT
1965 ,* THE OTHER THIS FURTHER VERIFIES THE OPERATION OF THE PAR/PDR
1966 ,* ADDR MUX AND THE ADDR DETECTION LOGIC
    
```


1970										
1971	022464	012700	172340		15	MOV	#KIPARG,RO		.LOAD ADDRESS OF FIRST PAR INTO RO	
1972	022470	012737	022502	001110	25	MOV	#3\$,SLPERR		.SET LOOP ON ERROR POINTER TO 3\$	
1973	022476	012703	000010			MOV	#10,R3		.LOAD LOOP COUNTER TO DO 8 PAR'S	
1974	022502	012701	177777		35	MOV	#-1,R1		.LOAD TEST PATTERN INTO R1	
1975	022506	005010				CLR	(RO)		.CLEAR THE PAR	


```

2005 .. *****
2006 . *TEST 20 TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PDR'S
2007 . *
2008 . * THE FOLLOWING TEST WRITES TO BOTH BYTES OF THE KERNEL & USER
2009
2010
2011
2012 . * PDR'S SEPERATELY TO SEE THAT WRITING TO ONE DOES NOT EFFECT
2013 . * THE OTHER THIS FURTHER VERIFIES THE OPERATION OF THE PAR/PDR
2014 . * ADDR MUX AND THE ADDR DETECTION LOGIC
2015 . *
2016 .. *****
2017 022612 000004 TST20 SCOPE
2018
2019 022614 012700 172300 15 MOV #KIPD%J,R0 .LOAD ADDRESS OF FIRST PDR INTO R0
2020 022620 012737 022632 001110 25 MOV #35,SLPERR .SET LOOP ON ERROR POINTER TO 35
2021 022626 012703 000010 MOV #10,R3 .LOAD LOOP COUNTER TO DO 8 PDR'S
2022 022632 012701 177777 35 MOV #-1,R1 .LOAD TEST PATTERN INTO R1
2023 022636 005010 CLR (R0) .CLEAR THE PDR
2024 022640 110110 MOV# R1,(R0) .WRITE 1'S TO THE LOW BYTE OF THE PDR
2025 022642 011002 MOV (R0),R2 .READ THE ENTIRE PDR INTO R2
2026 022644 042701 177761 BIC #177761,R1 .MASK HIGH BYTE & UNUSED BITS OUT OF DATA
2027 022650 020102 CMP R1,R2 .WAS ONLY THE LOW BYTE WRITTEN TO?
2028 022652 001401 BEQ 45 .BRANCH IF YES
2029 022654 104015 ERROR 15 .HIGH BYTE EFFECTED BY WRITING LOW BYTE IN PDR
2030 .FOR TIGHTER SCOPE LOOP
2031 .REPLACE ERROR CALL WITH
2032 . "BR 35" = 000766
2033 022656 012737 022664 001110 45 MOV #55,SLPERR .SET LOOP ON ERROR POINTER TO 55
2034 022664 005010 55 CLR (R0) .CLEAR THE PDR
2035 022666 012701 177777 MOV #-1,R1 .LOAD TEST PATTERN INTO R1
2036 022672 110160 000001 MOV# R1,1(R0) .WRITE 1'S TO THE HIGH BYTE OF THE PDR
2037 022676 011002 MOV (R0),R2 .READ THE ENTIRE PDR INTO R2
2038 022700 042701 100377 BIC #100377,R1 .MASK LOW BYTE & UNUSED BITS OUT OF DATA
2039 022704 020102 CMP R1,R2 .WAS ONLY THE HIGH BYTE WRITTEN TO?
2040 022706 001401 BEQ 65 .BRANCH IF YES
2041 022710 104015 ERROR 15 .LOW BYTE EFFECTED BY WRITING HIGH BYTE IN PDR
2042 .FOR TIGHTER SCOPE LOOP
2043 .REPLACE ERROR CALL WITH
2044 . "BR 55" = 000765
2045 022712 062700 000002 65 ADD #2,R0 .PUT ADDRESS OF NEXT PDR IN R0
2046 022716 077333 SOB R3,35 .BRANCH BACK UNTIL 8 PDR'S TESTED
2047 022720 022700 177620 CMP #UIPDR7+2,R0 .HAVE USER PDR'S BEEN TESTED?
2048 022724 103003 BHIS 75 .BRANCH IF YES
2049 022726 012700 177600 MOV #UIPDRO,R0 .LOAD ADDRESS OF FIRST USER PDR IN R0
2050 022732 000732 BR 25 .BRANCH BACK TO TEST USER PDR'S
2051 022734 012737 022614 001110 75 MOV #15,SLPERR .RESET LOOP ON ERROR POINTER TO 15
2052
    
```

```

2053 // *****
2054 *TEST 21 PAR-PDR DUAL ADDRESSING TEST
2055 *
2056 * THE FOLLOWING TEST SETS ALL OF THE WRITABLE BITS TO 1
2057 * IN THE SIXTEEN (16) PAR'S AND PDR'S USING THE "SETREG"
2058 * SUBROUTINE AND THEN CLEARS JUST ONE OF THEM. THE "CMPREG"
2059 * SUBROUTINE IS USED TO READ ALL OF THE PAR'S AND PDR'S TO SEE
2060 * THAT ONLY ONE REGISTER WAS CLEARED IN RESPONSE TO THAT ONE
2061 * PAR OR PDR ADDRESS. THE "CMPREG" SUBROUTINE REPORTS THE
2062 * ADDRESS OF ANY REGISTER WHOSE BITS DID NOT REMAIN SET WHEN
2063 * ANOTHER REGISTER WAS CLEARED
2064 *
2065 *
2066 * THE PAR AND PDR CHIPS, PAR/PDR ADDR MUX, AND ADDR DETECTION
2067 * LOGIC ARE CHECKED
2068 *
2069 // *****
2070 022742 000004 TST21 SCOPE
2071
2072 022744 012737 022762 001110 15 MOV #25, $LPERR ; SET LOOP ON ERROR POINTER 25
2073 022752 012703 000010 MOV #10, R3 ; LOAD LOOP COUNTER WITH AN 8
2074 022756 012700 172300 MOV #KIPDRO, RO ; LOAD ADDRESS OF FIRST KERNEL PDR AND RO
2075 022762 012706 001100 25 MOV #KERSTK, KSP ; SETUP STACK POINTER
2076 022766 004737 036070 JSR PC, SETREG ; SET ALL BITS IN ALL PAR'S IN PDR'S
2077 022772 005010 CLR (RO) ; CLEAR ONE OF THE KERNEL PDR'S
2078 022774 004737 036162 JSR PC, CMPREG ; SEE IF OTHER PAR/PDR'S WERE EFFECTED
2079 023000 062700 000002 ADD #2, RO ; FORM ADDRESS OF NEXT KERNEL PDR TO CLEAR
2080 023004 077312 SOB R3, 25 ; LOOP TO 25 UNTIL ALL KERNEL PDR'S CHECKED
2081 023006 012737 023024 001110 MOV #35, $LPERR ; SET LOOP ON ERROR POINTER TO 35
2082 023014 012703 000010 MOV #10, R3 ; LOAD LOOP COUNTER WITH AN 8
2083 023020 012700 172340 MOV #KIPARO, RO ; LOAD ADDRESS OF FIRST KERNEL PAR IN RO
2084 023024 012706 001100 35 MOV #KERSTK, KSP ; SETUP STACK POINTER
2085 023030 004737 036070 JSR PC, SETREG ; SET ALL BITS IN ALL PAR'S AND PDR'S
2086 023034 005010 CLR (RO) ; CLEAR ONE OF THE KERNEL PAR'S
2087 023036 004737 036162 JSR PC, CMPREG ; SEE IF OTHER PAR/PDR'S WERE EFFECTED
2088 023042 062700 000002 ADD #2, RO ; FORM ADDRESS OF NEXT KERNEL PAR TO CLEAR
2089 023046 077312 SOB R3, 35 ; LOOP TO 35 UNTIL ALL KERNEL PAR'S CHECKED
2090 023050 012737 023066 001110 MOV #45, $LPERR ; SET LOOP ON ERROR POINTER TO 45
2091 023056 012703 000010 MOV #10, R3 ; LOAD LOOP COUNTER WITH AN 8
2092 023062 012700 177600 MOV #UIPDRO, RO ; LOAD ADDRESS OF FIRST USER PDR IN RO
2093 023066 012706 001100 45 MOV #KERSTK, KSP ; SETUP STACK POINTER
2094 023072 004737 036070 JSR PC, SETREG ; SET ALL BITS IN ALL PAR'S AND PDR'S
2095 023076 005010 CLR (RO) ; CLEAR ONE OF THE USER PDR'S
2096 023100 004737 036162 JSR PC, CMPREG ; SEE IF OTHER PAR/PDR'S WERE EFFECTED
2097 023104 062700 000002 ADD #2, RO ; FORM ADDRESS OF NEXT USER PDR TO CLEAR
2098 023110 077312 SOB R3, 45 ; LOOP TO 45 UNTIL ALL USER PDR'S CHECKED
2099 023112 012737 023130 001110 MOV #55, $LPERR ; SET LOOP ON ERROR POINTER TO 55
2100 023120 012703 000010 MOV #10, R3 ; LOAD LOOP COUNTER WITH AN 8
2101 023124 012700 177640 MOV #UIPARO, RO ; LOAD ADDRESS OF FIRST USER PAR IN RO
2102 023130 012706 001100 55 MOV #KERSTK, KSP ; SETUP STACK POINTER
2103 023134 004737 036070 JSR PC, SETREG ; SET ALL BITS IN ALL PAR'S AND PDR'S
2104 023140 005010 CLR (RO) ; CLEAR ONE OF THE USER PAR'S
2105 023142 004737 036162 JSR PC, CMPREG ; SEE IF OTHER PAR/PDR'S WERE EFFECTED

```

2106	023146	062700	000002		ADD	#2,RO	;FORM ADDRESS OF NEXT USER PAR TO CLEAR
2107	023152	077312			SOB	R3,55	;LOOP TO 55 UNTIL ALL USER PAR'S CHECKED
2108	023154	012737	022744	001110	MOV	#15,\$LPERR	;SET LOOP ON ERROR POINTER TO 15
2109							
2110							*****
2111							TEST 22 TEST THAT PAR-PDR'S NOT AFFECTED BY RESET
2112							*
2113							THIS TEST CHECKS TO SEE THAT THE KERNEL OR USER PAR/PDR'S ARE
2114							NOT AFFECTED BY THE EXECUTION OF A "RESET" INSTRUCTION THE
2115							"SETREG" SUBROUTINE IS USED TO SET ALL WRITEABLE BITS TO A "1" IN
2116							THE PAR/PDR'S THEN THEY ARE READ TO SEE THAT THEY REMAINED
2117							UNCHANGED
2118							*
2119							*****
2120	023162	000004			TST22	SCOPE	
2121							
2122							
2123	023164	004737	036070	15	JSR	PC,SETREG	;SET ALL BITS IN ALL PAR'S AND PDR'S
2124	023170	000005			RESET		;ISSUE AN "INIT" BY EXECUTING A RESET
2125	023172	012700	172300		MOV	#KIPDR,RO	;LOAD ADDRESS OF FIRST KERNEL PDR IN RO
2126	023176	012704	000010		MOV	#10,R4	;LOAD LOOP COUNTER WITH AN 8
2127	023202	011001		25	MOV	(RO),R1	;READ A KERNEL PDR INTO R1
2128	023204	022701	077416		CMP	#77416,R1	;ARE ALL THE BITS STILL SET?
2129	023210	001401			BEQ	35	;BRANCH IF YES
2130	023212	104055			ERROR	55	;KERNEL PDR AFFECTED BY A RESET
2131							;FOR TIGHTER SCOPE LOOP
2132							;REPLACE ERROR CALL WITH
2133							;"BR 25" = 000773
2134	023214	062700	000002	35	ADD	#2,RO	;FORM ADDRESS OF NEXT KERNEL PDR
2135	023220	077410			SOB	R4,25	;LOOP TO 25 UNTIL ALL KERNEL PDR'S CHECKED
2136	023222	012700	172340		MOV	#KIPAR,RO	;LOAD ADDRESS OF FIRST KERNEL PAR IN RO
2137	023226	012704	000010		MOV	#10,R4	;LOAD LOOP COUNTER WITH AN 8
2138	023232	011001		45	MOV	(RO),R1	;READ A KERNEL PAR INTO R1
2139	023234	022701	007777		CMP	#7777,R1	;ARE ALL THE BITS STILL SET?
2140	023240	001401			BEQ	55	;BRANCH IF YES
2141	023242	104055			ERROR	55	;KERNEL PAR AFFECTED BY A RESET
2142							;FOR TIGHTER SCOPE LOOP
2143							;REPLACE ERROR CALL WITH
2144							;"BR 45" = 000773
2145	023244	062700	000002	55	ADD	#2,RO	;FORM ADDRESS OF NEXT KERNEL PAR
2146	023250	077410			SOB	R4,45	;LOOP TO 45 UNTIL ALL KERNEL PAR'S CHECKED
2147	023252	012700	177600		MOV	#UIPDR,RO	;LOAD ADDRESS OF FIRST USER PDR IN RO
2148	023256	012704	000010		MOV	#10,R4	;LOAD LOOP COUNTER WITH AN 8
2149	023262	011001		65	MOV	(RO),R1	;READ A USER PDR INTO R1
2150	023264	022701	077416		CMP	#77416,R1	;ARE ALL THE BITS STILL SET?
2151	023270	001401			BEQ	75	;BRANCH IF YES
2152	023272	104055			ERROR	55	;USER PDR AFFECTED BY A RESET
2153							;FOR TIGHTER SCOPE LOOP
2154							;REPLACE ERROR CALL WITH
2155							;"BR 65" = 000773
2156	023274	062700	000002	75	ADD	#2,RO	;FORM ADDRESS OF NEXT USER PDR
2157	023300	077410			SOB	R4,65	;LOOP TO 65 UNTIL ALL USER PDR'S CHECKED
2158							
2159	023302	012700	177640		MOV	#UIPAR,RO	;LOAD ADDRESS OF FIRST USER PAR IN RO
2160	023306	012704	000010		MOV	#10,R4	;LOAD LOOP COUNTER WITH AN 8
2161	023312	011001		85	MOV	(RO),R1	;READ A USER PAR INTO R1

2218 023440 012737 002400 000250
 2219 023446 012737 000340 000252
 2220 023454 012737 023334 001110
 2221 023462 004737 036036
 2222
 2223
 2224
 2225
 2226
 2227
 2228
 2229
 2230
 2231
 2232
 2233
 2234
 2235
 2236
 2237
 2238
 2239
 2240
 2241
 2242
 2243
 2244
 2245
 2246
 2247
 2248 023466 000004
 2249 023470 004737 036002
 2250 023474 012737 023564 001110
 2251 023502 012737 000600 172346
 2252 023510 012737 000600 172350
 2253 023516 012737 007600 172356
 2254 023524 006037 172306
 2255 023530 012737 077406 172310
 2256 023536 012737 077406 172316
 2257 023544 012737 023644 000250
 2258 023552 012737 000377 060000
 2259 023560 012700 177777
 2260 023564 062737 000400 177572 25
 2261 023572 113737 060000 100001
 2262 023600 000006
 2263 023602 013701 060000
 2264 023606 020001
 2265 023610 001401
 2266 023612 104062
 2267
 2268
 2269
 2270
 2271 023614 012737 002400 000250 35
 2272 023622 012737 077406 172306
 2273 023630 012737 023470 001110

MOV #MGMERR,MMVEC ,RESTORE MEM MGMT TRAP VECTOR
 MOV #340,MMVEC+2 ,RESTORE MEM MGMT VECTOR+2
 MOV #15,\$LPERR ,RESET LOOP ON ERROR POINTER TO 15
 JSR PC,T0N ,TURN T-BIT TRAPPING BACK ON

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

 TST24 SCOPE
 15 JSR PC,TOFF ,TURN OFF T-BIT TRAPPING FOR THIS TEST
 MOV #25,\$LPERR ,SET LOOP ON ERROR POINTER TO 25
 MOV #600,KIPAR3 ,MAP KERNAL PAGE 3 TO 12-16K
 MOV #600,KIPAR4 ,MAP KERNAL PAGE 4 TO 12-16K
 MOV #7600,KIPAR7 ,MAP KERNAL PAGE 7 TO THE I/O PAGE
 CLR KIPDR3 ,MAP KERNAL PAGE 3 "NO ACCESS"
 MOV #77406,KIPDR4 ,MAP KERNAL PAGE 4 R/W, 128 BLKS
 MOV #77406,KIPDR7 ,MAP KERNAL PAGE 7 R/W, 128 BLKS
 MOV #45,MMVEC ,SET M.M. VECTOR TO 45 IN CASE OF ABORT
 MOV #377,@#60000 ,LOAD ALL 1'S IN LOW BYTE OF TEST LOC
 MOV #-1,R0 ,LOAD EXPECTED CONTENTS OF TEST LOC IN R0
 BIS #BIT8,SRO ,TURN ON DEST -ONLY-RELOCATION
 MOVB @#60000,@#100001 ,LOAD HI BYTE OF TEST LOC -ABORT IF SOURCE RELOCATED
 RESET ,TURN OFF DEST.-ONLY-RELOCATION
 MOV @#60000,R1 ,READ CONTENTS OF TEST LOC
 CMP R0,R1 ,WAS TEST LOCATION LOADED PROPERLY?
 BEQ 35 ,BRANCH IF YES
 ERROR 62 ,TEST LOC NOT LOADED - INST WAS ABORTED
 ,WHEN SOURCE WAS RELOCATED
 ,FOR TIGHTER SCOPE LOOP REPLACE
 ,ERROR CALL WITH
 ,"BR 25" = 000764
 MOV #MGMERR,MMVEC ,RESTORE MEM MGMT VECTOR
 MOV #77406,KIPDR3 ,MAP KERNAL PAGE 3 R/W, 128 BLKS
 MOV #15,\$LPERR ,RESET LOOP ON ERROR POINTER TO 15


```

2274 023636 004737 036036          JSR    PC,TON          ,TURN T-BIT TRAPPING BACK ON
2275 023642 000430                    BR     TST25          ,SKIP TO NEXT TEST
2276          ,X IF THE PROGRAM TRIES TO RELOCATE THE SOURCE, IT SHOULD TRAP TO 45
2277
2278 023644 042737 000400 177572 45    BIC    #BIT8,SRO      ,TURN OFF DEST -ONLY-RELOCATION THRU PAR/PDR 7
2279 023652 013737 177572 001366        MOV    SRO,WASSRO    ,SAVE REST OF SRO CONTENTS
2280 023660 013737 177576 001370        MOV    SR2,WASSR2    ,SAVE CONTENTS OF SR2
2281 023666 010637 001354                    MOV    SP,WASR6      ,SAVE VALUE OF THE STACK POINTER
2282 023672 012637 001356                    MOV    (SP)+,TRAPPC ,SAVE PC OF TRAP
2283 023676 012637 001360                    MOV    (SP)+,TRAPPS ,SAVE PSW OF TRAP
2284 023702 042737 160000 177572        BIC    #160000,SRO   ,CLEAR ERROR BITS IN SR1
2285 023710 104063                    ERROR  63            ,SOURCE APPARENTLY RELOCATED IN MAINT MODE
2286          ,FOR TIGHTER SCOPE LOOP
2287          ,REPLACE ERROR CALL WITH A
2288          , "NOP" = 000240
2289 023712 013746 001360                    MOV    TRAPPS,-(SP)  ,PUT PSW OF TRAP BACK ON THE STACK
2290 023716 013746 001356                    MOV    TRAPPC,-(SP) ,PUT PC OF TRAP BACK ON THE STACK
2291 023722 000002                    RTI                    ,RETURN TO TEST
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305          ,*****
2306          ,*TEST 25          RELOCATION & ADDER TEST (NO CARRIES)
2307          ,*
2308          ,* THE FOLLOWING TEST SETS UP THE KERNEL PAR'S AND PDR'S
2309          ,* FOR THE REST OF THE PROGRAM IT THEN USES DIFFERENT
2310          ,* VIRTUAL ADDRESSES AND DIFFERENT VALUES FOR KERNEL PAR 4
2311          ,* TO PUT DIFFERENT PATTERNS AT THE INPUTS OF THE THREE
2312          ,* MEMORY MANAGEMENT ADDER CHIPS. THE VALUES ARE SUCH
2313          ,* THAT NO CARRIES ARE GENERATED OUT OF ANY OF THE ADDER CHIPS
2314          ,*
2315          ,* THE METHOD USED TO SEE THAT THE RIGHT PHYSICAL BUS ADDRESS
2316          ,* IS FORMED BY THE ADDERS IS TO WRITE A PATTERN TO VIRTUAL
2317          ,* LOCATION WITH MEMORY MGMT. IN THE MAINTENANCE MODE, AND
2318          ,* THEN READ THAT LOCATION USING THE PHYSICAL ADDRESS THAT SHOULD
2319          ,* HAVE BEEN FORMED TO SEE IF THE TEST PATTERN GOT THEIR
2320          ,*
2321          ,*****
2322 023724 000004          TST25  SCOPE
2323
2324 023726 012700 172340          15    MOV    #KIPAR0,R0    ,LOAD ADDRESS OF FIRST KERNEL PAR IN R0
2325 023732 005001                    CLR    R1            ,CLEAR R1
2326 023734 012702 000007          25    MOV    #7,R2        ,LOAD LOOP COUNTER WITH A 7
2327 023740 010120                    MOV    R1,(R0)+      ,MAP KERNEL PAR'S TO PAGES 0-6 (4K EACH)
2328 023742 062701 000200          ADD    #200,R1
2329 023746 077204                    SOB    R2,25        ,LOOP UNTIL KIPAR0 - KIPAR6 ARE LOADED
    
```

2330	023750	012710	007600			MOV	#7600, (RO)	. MAP KIPAR7 TO THE I/O PAGE
2331	023754	012700	172300			MOV	#KIPDR0, RO	. LOAD ADDRESS OF FIRST KERNEL PDR IN RO
2332	023760	012701	077406			MOV	#77406, R1	. LOAD PDR DATA INTO R1
2333	023764	012702	000010			MOV	#10, R2	. LOAD LOOP COUNTER WITH AN 8
2334	023770	010120		35		MOV	R1, (RO)+	. MAP ALL 8 PAGES 128 BLOCKS, UPWARD
2335	023772	077202				SQB	R2, 35	EXPANDABLE, READ/WRITE
2336								
2337	023774	012737	023774	001110	45	MOV	#45, \$LPERR	. SET LOOP ON ERROR POINTER TO 45
2338	024002	012700	067776			MOV	#67776, RO	. LOAD PHYSICAL ADDR. PBA INTO RO
2339	024006	012701	107776			MOV	#107776, R1	. LOAD VIRTUAL ADDR. VBA INTO R1
2340	024012	012702	125250			MOV	#125250, R2	. LOAD TEST PATTERN INTO R2
2341	024016	012704	000600			MOV	#600, R4, LOAD R4	WITH PAR VALUE
2342	024022	010437	172350			MOV	R4, KIPAR4	. LOAD KERNEL PAR 4 BITS <11 00>
2343	024026	011037	001176			MOV	(RO), \$TMP0	. SAVE CONTENTS AT TEST LOCATION
2344	024032	052737	000400	177572		BIS	#BIT8, SR0	. TURN ON "DESTINATION-ONLY-RELOCATION"
2345	024040	010211				MOV	R2, (R1)	. LOAD 125250 USING ADDER CHIPS (PAR4 + VIRT ADDR)
2346	024042	011003				MOV	(RO), R3	. READ 125250 BACK WITHOUT USING MEM. MGMT
2347	024044	000005				RESET		. TURN OFF MEMORY MGMT. MAINT. MODE
2348	024046	013710	001176			MOV	\$TMP0, (RO)	. RESTORE ORIGINAL CONTENTS TO TEST LOC
2349	024052	020203				CMF	R2, R3	. WAS SAME PATTERN READ BACK THAT WAS
2350								. WRITTEN USING "DEST-ONLY-RELOC. "?
2351	024054	001405				BEQ	55	. BRANCH IF YES
2352	024056	010137	001402			MOV	R1, VIRT1	. SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2353	024062	004737	036354			JSR	PC, FORMPA	. GO FORM PHYSICAL ADDRESS FOR TYPING
2354	024066	104017				ERROR	17	. TEST LOCATION DID NOT HAVE PATTERN
2355								. THAT SHOULD HAVE BEEN WRITTEN TO IT
2356								. APPARENTLY PHYSICAL ADDR. WAS
2357								. FORMED WRONG BY ADDERS USING
2358								. THE VIRTUAL ADDR. AND KIPAR4
2359								. FOR TIGHTER SCOPE LOOP
2360								. REPLACE ERROR CALL WITH
2361								. "BR 45" = 000742
2362	024070				55			
2363	024070	012737	024070	001110	65	MOV	#65, \$LPERR	. SET LOOP ON ERROR POINTER TO 65
2364	024076	012700	067776			MOV	#67776, RO	. LOAD PHYSICAL ADDR. PBA INTO RO
2365	024102	012701	102576			MOV	#102576, R1	. LOAD VIRTUAL ADDR. VBA INTO R1
2366	024106	012702	125251			MOV	#125251, R2	. LOAD TEST PATTERN INTO R2
2367	024112	012704	000652			MOV	#652, R4, LOAD R4	WITH PAR VALUE
2368	024116	010437	172350			MOV	R4, KIPAR4	. LOAD KERNEL PAR 4 BITS <11 00>
2369	024122	011037	001176			MOV	(RO), \$TMP0	. SAVE CONTENTS AT TEST LOCATION
2370	024126	052737	000400	177572		BIS	#BIT8, SR0	. TURN ON "DESTINATION-ONLY-RELOCATION"
2371	024134	010211				MOV	R2, (R1)	. LOAD 125251 USING ADDER CHIPS (PAR4 + VIRT ADDR)
2372	024136	011003				MOV	(RO), R3	. READ 125251 BACK WITHOUT USING MEM. MGMT
2373	024140	000005				RESET		. TURN OFF MEMORY MGMT. MAINT. MODE
2374	024142	013710	001176			MOV	\$TMP0, (RO)	. RESTORE ORIGINAL CONTENTS TO TEST LOC
2375	024146	020203				CMF	R2, R3	. WAS SAME PATTERN READ BACK THAT WAS
2376								. WRITTEN USING "DEST-ONLY-RELOC. "?
2377	024160	001405				BEQ	75	. BRANCH IF YES
2378	024162	010137	001402			MOV	R1, VIRT1	. SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2379	024166	004737	036354			JSR	PC, FORMPA	. GO FORM PHYSICAL ADDRESS FOR TYPING
2380	024162	104017				ERROR	17	. TEST LOCATION DID NOT HAVE PATTERN
2381								. THAT SHOULD HAVE BEEN WRITTEN TO IT
2382								. APPARENTLY PHYSICAL ADDR. WAS
2383								. FORMED WRONG BY ADDERS USING
2384								. THE VIRTUAL ADDR. AND KIPAR4
2385								. FOR TIGHTER SCOPE LOOP

2442	024366	012701	117776		MOV	#117776,R1	;LOAD VIRTUAL ADDR FOR PSW INTO R1	
2443	024372	012702	030240		MOV	#030240,R2	;LOAD DATA FOR PSW IN R2	
2444	024376	012704	007600		MOV	#7600,R4	;LOAD R4 WITH PAR VALUE	
2445	024402	010437	172350		MOV	R4,KIPAR4	;LOAD KERNEL PAR 4 BITS (<11:00>)	
2446	024406	052737	000400	177572	BIS	#BIT8,SRO	;TURN ON "DESTINATION-ONLY-RELOCATION"	
2447	024414	010211			MOV	R2,(R1)	;LOAD PSW USING ADDER CHIPS (PAR4 + VIRT ADDR)	
2448	024416	011003			MOV	(R0),R3	;READ PSW BACK WITHOUT USING MEM MGMT	
2449	024420	000005			RESET		;TURN OFF MEM MGMT. MAINT. MODE	
2450	024422	005010			CLR	(R0)	;CLEAR THE PSW	
2451	024424	042703	000037		BIC	#37,R3	;MASK T-BIT & CC BITS OUT OF DATA READ	
2452	024430	020203			CMP	R2,R3	;WAS PSW WRITTEN WHILE IN MAINT. MODE?	
2453	024432	001405			BEQ	125	;BRANCH IF YES	
2454	024434	010137	001402		MOV	R1,VIRT1	;SAVE VIRTUAL ADDR. TO FORM PHYSICAL ADDR	
2455	024440	004737	036354		JSR	PC,FORMPA	;GO FORM PHYSICAL ADDR. FOR TYPING	
2456	024444	104017			ERROR	17	;PSW DID NOT HAVE DATA THAT IT SHOULD	
2457							;HAVE, APPARENTLY PHYS. ADDR. OF PSW WAS	
2458							;NOT FORMED BY ADDERS USING THE	
2459							;VIRTUAL ADDR. AND KIPAR4	
2460							;FOR TIGHTER SCOPE LOOP	
2461							;REPLACE ERROR CALL WITH	
2462							; "BR 115" = 000743	
2463	024446	012737	023726	001110	125	MOV	#15,SLPERR	;RESET LOOP ON ERROR POINTER TO 15
2464								
2465							*****	
2466							;TEST 26 RELOCATION & ADDER TEST (WITH CARRIES)	
2467							;*	
2468							;*	
2469							;*	
2470							;*	
2471							;*	
2472							;*	
2473							;*	
2474							;*	
2475							;*	
2476							*****	
2477	024454	000004					TEST 26. SCOPE	
2478								
2479	024456				15:		;KERNEL PAR'S AND PDR'S HAVE BEEN	
2480							;SETUP BY THE PREVIOUS TEST	
2481	024456	012737	024456	001110	25:	MOV	#25,SLPERR	;SET LOOP ON ERROR POINTER TO 25
2482	024464	012700	064276			MOV	#64276,R0	;LOAD PHYSICAL ADDR. PBA INTO R0
2483	024470	012701	102176			MOV	#102176,R1	;LOAD VIRTUAL ADDR. VBA INTO R1
2484	024474	012702	125253			MOV	#125253,R2	;LOAD TEST PATTERN INTO R2
2485	024500	012704	000621			MOV	#621,R4 ;LOAD R4	;LOAD R4 WITH PAR VALUE
2486	024504	010437	172350			MOV	R4,KIPAR4	;LOAD KERNEL PAR 4 BITS (<11:00>)
2487	024510	011037	001176			MOV	(R0),STMP0	;SAVE CONTENTS AT TEST LOCATION
2488	024514	062737	000400	177572		BIS	#BIT8,SRO	;TURN ON "DESTINATION-ONLY-RELOCATION"
2489	024522	010211				MOV	R2,(R1)	;LOAD 125253 USING ADDER CHIPS (PAR4 + VIRT ADDR)
2490	024524	011003				MOV	(R0),R3	;READ 125253 BACK WITHOUT USING MEM MGMT
2491	024526	000005				RESET		;TURN OFF MEMORY MGMT. MAINT. MODE
2492	024530	013710	001176			MOV	STMP0,(R0)	;RESTORE ORIGINAL CONTENTS TO TEST LOC
2493	024534	020203				CMP	R2,R3	;WAS SAME PATTERN READ BACK THAT WAS
2494							;WRITTEN USING "DEST-ONLY-RELOC.?"	
2495	024536	001405				BEQ	35	;BRANCH IF YES
2496	024540	010137	001402			MOV	R1,VIRT1	;SAVE VIRTUAL ADDR. TO FORM PHYS ADDR
2497	024544	004737	036354			JSR	PC,FORMPA	;GO FORM PHYSICAL ADDRESS FOR TYPING

Address	Virtual Addr	Physical Addr	Physical Addr	Op Code	Op Code	Comments
2498	024550	104017		ERROR	17	. TEST LOCATION DID NOT HAVE PATTERN
2499						. THAT SHOULD HAVE BEEN WRITTEN TO IT
2500						. APPARENTLY PHYSICAL ADDR WAS
2501						. FORMED WRONG BY ADDERS USING
2502						. THE VIRTUAL ADDR. AND KIPAR4
2503						. FOR TIGHTER SCOPE LOOP
2504						. REPLACE ERROR CALL WITH
2505						. "BR 25" = 000742
2506	024552				35	
2507	024552	012737	024552	MOV	#45, SLPERR	. SET LOOP ON ERROR POINTER TO 45
2508	024560	012700	064476	MOV	#64476, R0	. LOAD PHYSICAL ADDR. PBA INTO R0
2509	024564	012701	112376	MOV	#112376, R1	. LOAD VIRTUAL ADDR. VBA INTO R1
2510	024570	012702	125254	MOV	#125254, R2	. LOAD TEST PATTERN INTO R2
2511	024574	012704	000521	MOV	#521, R4, LOAD R4	. WITH PAR VALUE
2512	024600	010437	172350	MOV	R4, KIPAR4	. LOAD KERNEL PAR 4 BITS <11:00>
2513	024604	011037	001176	MOV	(R0), STMP0	. SAVE CONTENTS AT TEST LOCATION
2514	024610	052737	000400	BIS	#BIT8, SRO	. TURN ON "DESTINATION-ONLY-RELOCATION"
2515	024616	010211		MOV	R2, (R1)	. LOAD 125254 USING ADDER CHIPS (PAR4 + VIRT ADDR)
2516	024620	011003		MOV	(R0), R3	. READ 125254 BACK WITHOUT USING MEM MGMT
2517	024622	000005		RESET		. TURN OFF MEMORY MGMT. MAINT. MODE
2518	024624	013710	001176	MOV	STMP0, (R0)	. RESTORE ORIGINAL CONTENTS TO TEST LOC
2519	024630	020203		CMP	R2, R3	. WAS SAME PATTERN READ BACK THAT WAS
2520						. WRITTEN USING "DEST-ONLY-RELOC" ?
2521	024632	001405		BEQ	55	. BRANCH IF YES
2522	024634	010137	001402	MOV	R1, VIRT1	. SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2523	024640	004737	036354	JSR	PC, FORMPA	. GO FORM PHYSICAL ADDRESS FOR TYPING
2524	024644	104017		ERROR	17	. TEST LOCATION DID NOT HAVE PATTERN
2525						. THAT SHOULD HAVE BEEN WRITTEN TO IT
2526						. APPARENTLY PHYSICAL ADDR WAS
2527						. FORMED WRONG BY ADDERS USING
2528						. THE VIRTUAL ADDR. AND KIPAR4
2529						. FOR TIGHTER SCOPE LOOP
2530						. REPLACE ERROR CALL WITH
2531						. "BR 45" = 000742
2532	024646				55	
2533	024646	012737	024646	MOV	#65, SLPERR	. SET LOOP ON ERROR POINTER TO 65
2534	024654	012700	061076	MOV	#61076, R0	. LOAD PHYSICAL ADDR PBA INTO R0
2535	024660	012701	106776	MOV	#106776, R1	. LOAD VIRTUAL ADDR VBA INTO R1
2536	024664	012702	125255	MOV	#125255, R2	. LOAD TEST PATTERN INTO R2
2537	024670	012704	000521	MOV	#521, R4, LOAD R4	. WITH PAR VALUE
2538	024674	010437	172350	MOV	R4, KIPAR4	. LOAD KERNEL PAR 4 BITS <11 00>
2539	024700	011037	001176	MOV	(R0), STMP0	. SAVE CONTENTS AT TEST LOCATION
2540	024704	052737	000400	BIS	#BIT8, SRO	. TURN ON "DESTINATION-ONLY-RELOCATION"
2541	024712	010211		MOV	R2, (R1)	. LOAD 125255 USING ADDER CHIPS (PAR4 + VIRT ADDR)
2542	024714	011003		MOV	(R0), R3	. READ 125255 BACK WITHOUT USING MEM MGMT
2543	024716	000005		RESET		. TURN OFF MEMORY MGMT. MAINT. MODE
2544	024720	013710	001176	MOV	STMP0, (R0)	. RESTORE ORIGINAL CONTENTS TO TEST LOC
2545	024724	020203		CMP	R2, R3	. WAS SAME PATTERN READ BACK THAT WAS
2546						. WRITTEN USING "DEST-ONLY-RELOC" ?
2547	024726	001405		BEQ	75	. BRANCH IF YES
2548	024730	010137	001402	MOV	R1, VIRT1	. SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
2549	024734	004737	036354	JSR	PC, FORMPA	. GO FORM PHYSICAL ADDRESS FOR TYPING
2550	024740	104017		ERROR	17	. TEST LOCATION DID NOT HAVE PATTERN
2551						. THAT SHOULD HAVE BEEN WRITTEN TO IT
2552						. APPARENTLY PHYSICAL ADDR WAS
2553						. FORMED WRONG BY ADDERS USING


```

2610 025132
2611 025132 012737 024456 001110 115 MOV #15,SLPERR .RESET LOOP ON ERROR POINTER TO 15
2612
2613 ..*****
2614 .TEST 27 READ AND WRITE WHILE IN RELOCATE MODE
2615 .X
2616 .X THE FOLLOWING TEST TURNS ON MEMORY MANAGEMENT AND THEN
2617 .X READS AND WRITES LOCATIONS BETWEEN PHYSICAL ADDRESSES
2618 .X 060000-067600. ONE LOCATION IN EVERY BLOCK (32 WORDS)
2619 .X IS WRITTEN USING PAR4 AND READ USING PAR5. THIS IS
2620 .X DONE IN BOTH USER AND KERNEL MODES. THE USER PAR/PDR'S
2621 .X ARE SETUP AT THE BEGINNING OF THE TEST AND ONCE MEMORY
2622 .X MANAGEMENT IS TURNED ON IT IS LEFT ON FOR THE REST OF THE
2623 .X OF THE PROGRAM. THE "MODE" INPUT TO THE PAR/PDR ADDRESS MUX
2624 .X IS CHECKED BY READING AND WRITING IN USER MODE. REMEMBER
2625 .X ALSO, THAT SINCE MEMORY MANAGEMENT IS ON (IN RELOCATE
2626 .X MODE) THE PROGRAM ITSELF IS USING ITS VIRTUAL ADDRESSES AND
2627 .X THE PAR/PDR'S TO EXECUTE.
2628 .X
2629 .X WHILE TESTING IN KERNEL MODE, USER PAGES 4 & 5 ARE MAPPED
2630 .X NON-RESIDENT WITH DIFFERENT PAR VALUES THAN THE KERNEL
2631 .X PAR'S TO BE SURE THAT THE KERNEL PAR'S AND PDR'S ARE BEING
2632 .X USED WHEN IN KERNEL MODE (AND VICE VERSA WHILE TESTING IN
2633 .X USER MODE). IF A MEM. MGMT. TRAP OCCURS, THE PROGRAM GOES
2634 .X TO 85 WHERE THE TRAP IS REPORTED
2635 .X
2636 ..*****
2637 025140 000004 TST27 SCOPE
2638
2639 025142 005037 177776 15 CLR PSW .START IN KERNEL MODE
2640 025146 012704 000577 MOV #577,R4 .LOAD R4 WITH VALUE FOR PAR4
2641 025152 012705 000600 MOV #600,R5 .LOAD R5 WITH VALUE FOR PAR5
2642 025156 010437 172350 MOV R4,KIPAR4 .LOAD KERNEL PAR4
2643 025162 010537 172352 MOV R5,KIPAR5 .LOAD KERNEL PAR5
2644 025166 012700 177640 MOV #UIPAR0,R0 .LOAD ADDRESS OF FIRST USER PAR IN R0
2645 025172 005001 CLR R1 .CLEAR R1
2646 025174 012702 000007 MOV #7,R2 .LOAD LOOP COUNTER WITH A 7
2647 025200 010120 25 MOV R1,(R0)+ .MAP USER PAR'S TO PAGES 0-6 (4K EACH)
2648 025202 062701 000200 ADD #200,R1
2649 025206 077204 SOB R2,25 .LOOP UNTIL UIPAR0-UIPAR6 ARE LOADED
2650 025210 012710 007600 MOV #7600,(R0) .MAP USER PAR7 TO THE I/O PAGE
2651 025214 012700 177600 MOV #UIPDR0,R0 .LOAD ADDRESS OF FIRST USER PDR IN R0
2652 025220 012701 077406 MOV #77406,R1 .LOAD PDR DATA INTO R1
2653 025224 012702 000010 MOV #10,R2 .LOAD LOOP COUNTER WITH AN 8
2654 025230 010120 35 MOV R1,(R0)+ .MAP ALL 8 PAGES 128 BLOCKS, UPWARD
2655 025232 077202 SOB R2,35 .EXPANDABLE, READ/WRITE
2656 025234 105037 177610 CLR# UIPDR4 .MAP USER SPACE NON-RESIDENT WHILE
2657 025240 105037 177612 CLR# UIPDR5 .TESTING KERNEL SPACE
2658 025244 010537 177650 MOV R5,UIPAR4 .MAP USER PAR'S OPPOSITE OF KIPAR'S
2659 025250 010437 177652 MOV R4,UIPAR5
2660 025254 012737 000001 177572 MOV #1,SRO .TURN ON MEMORY MANAGEMENT (RELOCATE MODE)
2661 025262 012737 025314 001110 MOV #55,SLPERR .SET LOOP ON ERROR POINTER TO 55
2662 025270 012737 025526 000250 MOV #85,MAVEC .SET M M TRAP VECTOR TO 85
2663 025276 013737 177776 001176 45 MOV PSW,STMP0 .SAVE PSW IN CASE OF ERROR
2664 025304 012700 100100 MOV #100100,R0 .PUT VIRTUAL ADDR THAT USER PAR4 IN R0
2665 025310 012701 120000 MOV #120000,R1 .PUT VIRTUAL ADDR THAT USES PAR5 IN R1
    
```

2666	025314	010010		55	MOV	R0, (R0)	, WRITE TO TEST LOC USING PAR4
2667	025316	011102			MOV	(R1), R2	, READ THE SAME LOC, BUT USING PAR5
2668	025320	020002			CMP	R0, R2	, DID WE READ WHAT WE WROTE?
2669	025322	001411			BEQ	65	, BRANCH IF YES
2670	025324	010137	001404		MOV	R1, VIRT2	, SAVE VIRTUAL ADDR. THAT SELECTED PAR5
2671	025330	010037	001402		MOV	R0, VIRT1	, SAVE VIRTUAL ADDR. THAT SELECTED PAR4
2672	025334	004737	036354		JSR	PC, FORMPA	, GO FORM PHYSICAL ADDRESS BEING USED
2673	025340	104020			ERROR	20	, READING LOC. USING PAR5 AND A VIRT
2674							, ADDR DID NOT FIND DATA WRITTEN WHEN USING
2675							, PAR4 AND VIRT. ADDRESS
2676							, FOR TIGHTER SCOPE LOOP
2677							, REPLACE ERROR CALL WITH
2678							, "BR 55" = 000765
2679	025342	013700	001402		MOV	VIRT1, R0	, RESTORE VBA IN R0
2680	025346	062700	000100	65	ADD	#100, R0	, CHANGE VIRTUAL ADDR TO POINT TO NEXT BLOCK
2681	025352	062701	000100		ADD	#100, R1	
2682	025356	020127	127700		CMP	R1, #127700	, WERE BLOCKS FROM 60000-676000 ALL TRIED?
2683	025362	001354			BNE	55	, BRANCH IF NO
2684	025364	032737	140000	177776	BIT	#140000, PSW	, HAVE WE DONE TEST IN USER MODE YET?
2685	025372	001026			BNE	75	, BRANCH IF YES
2686	025374	010437	177650		MOV	R4, UIPAR4	, LOAD USER PAR4
2687	025400	010537	177652		MOV	R5, UIPAR5	, LOAD USER PAR5
2688	025404	112737	000006	177610	MOVB	#6, UIPDR4	, MAP USER SPACE R/W TO TEST IT
2689	025412	112737	000006	177612	MOVB	#6, UIPDR5	
2690	025420	105037	172310		CLRB	KIPDR4	, MAP KERNEL SPACE NON-RESIDENT WHILE
2691	025424	105037	172312		CLRB	KIPDR5	, TESTING USER SPACE
2692	025430	010537	172350		MOV	R5, KIPAR4	, MAP KERNEL PAR'S OPPOSITE UIPAR'S
2693	025434	010437	172352		MOV	R4, KIPAR5	
2694	025440	012737	140000	177776	MOV	#140000, PSW	, GO TO USER MODE
2695	025446	000713			BR	45	, GO BACK AND READ/WRITE IN USER MODE
2696	025450	005037	177776	75	CLR	PSW	, GO BACK TO KERNEL MODE BEFORE LEAVING
2697	025454	012737	077406	172310	MOV	#77406, KIPDR4	, REMAP KERNEL PAGES READ/WRITE
2698	025462	012737	077406	172312	MOV	#77406, KIPDR5	
2699	025470	010537	172350		MOV	R5, KIPAR4	, MAP KERNEL AND USER PAR'S 4 & 5
2700	025474	010537	172352		MOV	R5, KIPAR5	, BACK TO 12-16K
2701	025500	010537	177650		MOV	R5, UIPAR4	
2702	025504	010537	177652		MOV	R5, UIPAR5	
2703	025510	012737	002400	000250	MOV	#MGHERR, MMVEC	, RESTORE ADDR OF NORMAL M M TRAP ROUTINE
2704	025516	012737	025142	001110	MOV	#15, \$LPERR	, RESET LOOP ON ERROR POINTER TO 15
2705	025524	000427			BR	TST30	, GO TO NEXT TEST
2706							
2707	025526	012637	001356	85	MOV	(KSP)+, TRAPP	, SAVE PC & PS OF TRAP
2708	025532	012637	001360		MOV	(KSP)+, TRAPP5	
2709							, PROGRAM WILL TRAP TO HERE IF TRY
2710							, TO USE USER PDR'S WHEN IN KERNEL MODE
2711							, OR KERNEL PDR'S WHEN IN USER MODE
2712	025536	010037	001402		MOV	R0, VIRT1	, SAVE VIRTUAL ADDRESS FOR ERROR REPORT
2713	025542	004737	036354		JSR	PC, FORMPA	, GO FORM THE PHYSICAL ADDRESS BEING USED
2714	025546	013737	177572	001366	MOV	SRO, WASSRO	, SAVE SRO & SR2 FOR ERROR REPORT
2715	025554	013737	177576	001370	MOV	SR2, WASSR2	
2716	025562	042737	160000	177572	BIC	#160000, SRO	, CLEAR ERROR BITS IN SRO
2717	025570	104052			ERROR	52	, M M TRAP WHILE IN RELOCATE MODE -
2718							, REFERENCED WRONG SET OF PDR'S
2719							, FOR TIGHTER SCOPE LOOP
2720							, REPLACE ERROR CALL WITH
2721							, A "NOP" = 000240


```

2722 025572 013746 001360      MOV      TRAPPS,-(KSP)  ,PUT PC & PS OF TRAP ON STACK
2723 025576 013746 001356      MOV      TRAPPC,-(KSP)
2724 025602 000002                RTI          ,RETURN TO TEST
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744 025604 000004                TST30      SCOPE
2745 025606
2746 025606 004737 036002      JSR      PC,TOFF      ,TURN T-BIT TRAPPING OFF FOR THIS TEST
2747 025612 012702 000004      MOV      #4,R2        ,SET LOOP COUNTER TO 4
2748 025616 012700 172346      MOV      #KIPAR3,RO   ,LOAD ADDRESS OF PAR3 INTO RO
2749 025622 012701 000600      MOV      #600,R1      ,LOAD "12-16K" PAR VALUE INTO R1
2750 025626 010120                25      MOV      R1,(RO)+     ,MAP PARS 3-6 TO 12-16K
2751 025630 077202                SOB      R2,25        ,LOOP TIL ALL 4 OF THEM LOADED
2752 025632 012705 172300      MOV      #KIPDRO,R5   ,LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
2753 025636 012704 000010      MOV      #10,R4       ,SET LOOP COUNTER TO 8
2754 025642 012703 017776      MOV      #17776,R3    ,INITIALIZE VIRTUAL ADDRESS TO BE IN R3
2755 025646 012737 025654 001110      MOV      #35,SLPERR   ,SET LOOP ON ERROR POINTER TO 35
2756 025664 012700 172300 35      MOV      #KIPDRO,RO   ,LOAD ADDR. OF FIRST PDR TO BE SETUP IN RO
2757 025660 012702 000010      MOV      #10,R2       ,SET LOOP COUNTER TO 8
2758 025664 012701 077406      MOV      #77406,R1    ,PUT "W-BIT OFF DATA" INTO R1
2759 025670 010120 45      MOV      R1,(RO)+     ,CLEAR ALL W-BITS BY WRITING TO ALL PDRS
2760 025672 077202                SOB      R2,45        ,LOOP UNTIL ALL OF THEM SETUP
2761 025674 011313                MOV      (R3),(R3)    ,DO "DATA" TO VIRTUAL ADDR -SETTING A W-BIT
2762 025676 031527 000100      BIT      (R5),#WBIT   ,DID THAT CAUSE W-BIT TO BE SET?
2763 025702 001002                BNE      55           ,BRANCH IF YES
2764 025704 104021                ERROR    21          ,W-BIT DID NOT GET SET IN PDR
2765
2766
2767
2768 025706 000422                BR       85           ,FOR TIGHTER SCOPE LOOP
2769 025710 012702 000010 55      MOV      #10,R2       ,REPLACE ERROR CALL WITH
2770 025714 012700 172300                MOV      #KIPDRO,RO   , "BR 35" = 000763
2771 025720 031027 000100 65      BIT      (RO),#WBIT   ,SKIP CHECKING OTHER PDR'S-ERROR WILL SET W-BITS
2772 025724 001403                BEQ      75           ,SET LOOP COUNTER TO 8
2773 025726 020500                CMP      R5,RO        ,LOAD ADDR. OF FIRST PDR TO BE CHECKED IN RO
2774 025730 001401                BEQ      75           ,DID W-BIT IN OTHER PDRS REMAIN CLEAR?
2775 025732 104022                ERROR    22          ,BRANCH IF YES
2776
2777

```

 *TEST 30 W-BIT LOGIC TEST, KERNEL PDR'S
 *
 * THIS TEST WRITES TO EIGHT (8) DIFFERENT VIRTUAL ADDRESSES
 * (VBA'S = 17776, 37776, 57776, 77776, 117776, 137776, 157776, & 177776
 * & PBA'S CONSTRUCTED = 17776, 37776, 57776, 77776, 77776,
 * 77776, 77776, & 777776 RESPECTIVELY).
 * WHICH SHOULD CAUSE THE "W-BIT" TO SET IN EACH OF THE
 * EIGHT (8) KERNEL PAGE DESCRIPTOR REGISTERS THE PDR'S
 * ARE CHECKED TO SEE THAT IT'S W-BIT DOES SET WHEN THE
 * PAGE IT IS MAPPED TO IS WRITTEN TO AND THAT THE W-BIT
 * DOES NOT SET IN ANY OF THE OTHER PDR'S. KERNEL PDR'S 3,4,5,6
 * ARE MAPPED TO 12-16K FOR THIS TEST. ALSO THE W-BIT
 * SHOULD BE CLEARED WHEN THE PDR IS WRITTEN TO THE
 * W-BIT PORTION OF THE PDR'S AND THE PAR/PDR ADRS MUX
 * ARE BEING CHECKED

```

2890 026266 012737 026300 000004      MOV      #4$ ERVEC      .SET UP LOC 4 TO 4$ FOR ODD ADDR  ABORT
2891 026274 005237 060001      INC      60001          .CAUSE ODD ADDRESS ABORT THRU LOC 4
2892 026300 012706 001100      MOV      #KERSTK.KSP   .RESTORE THE STACK POINTER
2893 026304 013702 172306      MOV      KIPDR3,R2     .READ KIPDR3 INTO R2
2894 026310 020102      CMP      R1,R2         .WAS W-BIT LEFT CLEARED?
2895 026312 001401      BEQ      55            .BRANCH IF YES
2896 026314 104025      ERROR    25           .W-BIT GOT SET DURING AN ODD ADDR  ABORT
2897
2898
2899
2900 026316 012737 002332 000004 55      MOV      #TIMERR.ERVEC .RESTORE NORMAL CPU TRAP ROUTINE TO LOC 4
2901 026324 012737 026210 001110      MOV      #15,$LPERR    .RESET LOOP ON ERROR POINTER TO 15
2902 026332 004737 036036      JSR      PC,T0N        .TURN T-BIT TRAPPING BACK ON
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925 026336 000004      TST33   SCOPE
2926
2927 026340 012700 000600      15      MOV      #600,R0        .LOAD DATA FOR PAR'S INTO R0
2928 026344 010037 172346      MOV      R0,KIPAR3     .MAP KERNEL PAR'S 3&4 TO 12-16K
2929 026350 010037 172350      MOV      R0,KIPAR4
2930 026354 010037 177646      MOV      R0,UIPAR3     .MAP USER PAR'S 3&4 TO 12-16K
2931 026360 010037 177650      MOV      R0,UIPAR4
2932 026364 012737 077406 172306      MOV      #77406,KIPDR3 .MAP KERNEL PDR 3 128 BLKS, READ-WRITE
2933 026372 012737 077406 177606      MOV      #77406,UIPDR3 .MAP USER PDR 3 128 BLKS, READ-WRITE
2934 026400 012700 060000      MOV      #60000,R0     .LOAD VIRTUAL ADDR TO REFERENCE PDR3 INTO R0
2935 026404 012701 100000      MOV      #100000,R1    .LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
2936 026410 012703 100011      MOV      #100011,R3    .LOAD R3 WITH WHAT SRO SHOULD READ - N R , KERNEL, PG 4
2937 026414 012702 077400      MOV      #77400,R2     .LOAD ACF=0 (NON-RESIDENT) PDR VALUE IN R2
2938 026420 012737 026462 000250 25      MOV      #55,$MIVEC    .POINT MEM. MGMT. TRAP VECTOR TO 55 BELOW
2939 026426 010237 172310      MOV      R2,KIPDR4     .LOAD ACF TEST VALUE INTO KIPDR4
2940 026432 010237 177610      MOV      R2,UIPDR4     .LOAD ACF TEST VALUE INTO UIPDR4
2941 026436 012737 026444 001110      MOV      #3,$LPERR    .SET LOOP ON ERROR POINTER TO 3$
2942 026444 006010      CLR      (R0)          .CLEAR PHYS. LOC 60000 USING PDR3
2943 026446 013737 177776 001176      MOV      PSW,$TMP0     .SAVE PSW IN CASE OF ERROR
2944 026454 005211      INC      (R1)          .TRY TO REF IT USING PDR4 - SHOULD TRAP TO 55
2945 026456 104026      ERROR    26           .MEM MGMT ABORT DID NOT OCCUR
    
```

THE NEXT THREE (3) TESTS CAUSE MEMORY MANAGEMENT ERRORS TO CHECK THE ABILITY OF STATUS REGISTER 0 TO RECORD KT ERRORS AND THE ABILITY OF STATUS REGISTER 2 TO LOCK UP THE VIRTUAL ADDR. OF THE INSTRUCTION THAT CAUSED THE ERROR THE BITS OF SR2 ARE CHECKED AND BITS <15:13>, <6:5>, AND <3:0> ARE CHECKED IN SRO SO THE SRO AND SR2 LOGIC AND THE KT ERROR LOGIC ARE CHECKED.

TEST 33 NON-RESIDENT ABORT TEST (ACF=0&4)

THIS TEST CHECKS THE ACCESS CONTROL FIELD (ACF) COMPARATOR LOGIC BY CAUSING NON-RESIDENT ABORTS IN BOTH KERNEL AND USER MODES. PDR 4 IS LOADED WITH ACF'S = 0&4 AND THEN PHYSICAL ADDR 60000 IS ACCESSED TO CAUSE THE ABORT

```

2946 ;FOR TIGHTER SCOPE LOOP
2947 ;REPLACE ERROR CALL WITH
2948 ;"BR 35" = 000772
2949 026460 000425 BR 85 ;BRANCH AROUND STATUS REG CHECKS IF NO ABORT
2950 026462 062706 000004 55 ADD #4,SP ;RESTORE STACK POINTER
2951 026466 005710 TST (R0) ;DID INSTRUCTION GET ABORTED & NOT EXECUTE
2952 026470 001401 BEQ 65 ;BRANCH IF YES
2953 026472 104027 ERROR 27 ;INSTRUCTION WAS NOT ABORTED, LOC GOT CHANGED
2954 ;FOR TIGHTER SCOPE LOOP
2955 ;REPLACE ERROR CALL WITH
2956 ;"BR 35" = 000764
2957 026474 013737 177572 001366 65 MOV SRO,WASSRO ;READ STATUS REGISTER 0
2958 026502 013737 177576 001370 MOV SR2,WASSR2 ;READ STATUS REGISTER 2
2959 026510 020337 001366 CMP R3,WASSRO ;DID SRO REPORT NON-RESIDENT ERROR CORRECTLY?
2960 026514 001401 BEQ 75 ;BRANCH IF YES
2961 026516 104030 ERROR 30 ;SRO DID NOT REPORT NON-RES ERROR CORRECTLY
2962 ;FOR TIGHTER SCOPE LOOP
2963 ;REPLACE ERROR CALL WITH
2964 ;"BR 35" = 000752
2965 026520 012704 026454 75 MOV #45,R4 ;LOAD R4 WITH WHAT SR2 SHOULD READ
2966 026524 020437 001370 CMP R4,WASSR2 ;DID SR2 LOCKUP RIGHT VIRTUAL ADDR (=45)?
2967 026530 001401 BEQ 85 ;BRANCH IF YES
2968 026532 104031 ERROR 31 ;SR2 DID NOT LOCK VIRTUAL ADDR OF NON-RES ERROR
2969 ;FOR TIGHTER SCOPE LOOP
2970 ;REPLACE ERROR CALL WITH
2971 ;"BR 35" = 000744
2972 026534 042737 160000 177572 85 BIC #160000,SRO ;CLEAR THE ERROR BITS IN SRO
2973 026542 032737 140000 001176 BIT #140000,$TMP0 ;HAS ACF=0&4 BEEN TESTED IN USER YET
2974 026550 001006 BNE 95 ;BRANCH IF YES
2975 026552 012703 100151 MOV #100151,R3 ;LOAD R3 WITH WHAT SRO SHOULD READ - N R , USER, PG 4
2976 026556 012737 140000 177776 MOV #140000,PSW ;GO TO USER MODE
2977 026564 000715 BR 25 ;REPEAT TEST IN USER MODE
2978 026566 022702 077404 95 CMP #77404,R2 ;HAS ACF=4 BEEN TESTED YET?
2979 026572 001407 BEQ 105 ;BRANCH IF YES
2980 026574 012702 077404 MOV #77404,R2 ;THEN LOAD ACF=4 (NON-RES) PDR VALUE IN R2
2981 026600 012703 100011 MOV #100011,R3 ;LOAD R3 WITH WHAT SRO SHOULD READ-N.R , KERNEL.PG 4
2982 026604 005037 177776 CLR PSW ;GO BACK TO KERNEL MODE
2983 026610 000703 BR 25 ;GO BACK & TEST ACF=4 IN SAME MODE
2984 026612 005037 177776 105 CLR PSW ;GO BACK TO KERNEL MODE BEFORE LEAVING
2985 026616 012737 026340 001110 MOV #15,$LPERR ;RESET LOOP ON ERROR POINTER TO 15
2986 026624 012737 002400 000250 MOV #MGERR,MVVEC ;RESTORE ADDRESS OF NORMAL MEMORY
2987 ;MANAGEMENT ERROR ROUTINE TO MVVEC
2988
2989 .. *****
2990 ;*TEST 34 READ-ONLY ABORT TEST (ACF=2)
2991 ;*
2992 ;* THIS TEST CHECKS THE ACCESS CONTROL FIELD (ACF) COMPARATOR
2993 ;* LOGIC BY CAUSING READ-ONLY ABORTS IN BOTH KERNEL AND
2994 ;* USER MODES. PDR 4 IS LOAD WITH ACF=2 AND THEN
2995 ;* PHYSICAL ADDR 60000 IS WRITTEN TO CAUSE THE ABORT
2996 ;*
2997 .. *****
2998 026632 000004 TST34 SCOPE
2999 026634 15 ;KERNEL & USER PAR'S 3 & 4 AND PDR 3
3000 ;ARE SETUP FROM LAST TEST
3001 026634 012700 060000 MOV #60000,R0 ;LOAD VIRTUAL ADDR TO REFERENCE PDR3 INTO R0
    
```



```

3058 .X
3059 ..XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
3060 027042 000004 TST35 SCOPE
3061 027044 012737 027060 001110 15 MOV #25,SLPERR ;SET LOOP ON ERROR POINTER TO 25
3062 027052 012737 027074 000250 MOV #35,MMVEC ;LOAD MEM. MGMT. TRAP VECTOR WITH 35
3063 027060 012737 040000 177776 25 MOV #40000,PSW ;SET 01 IN PSW CURRENT MODE BITS
3064 027066 000240 NOP ;FETCH OF THIS INSTRUCTION SHOULD CAUSE ABORT
3065 027070 104056 ERROR 56 ;ILLEGAL MODE 01 NOT ABORTED
3066 ;FOR A TIGHTER SCOPE LOOP
3067 ;REPLACE ERROR CALL WITH
3068 ;"BR 25" = 000773
3069 027072 000424 BR 55 ;BRANCH AROUND SRO & SR2 CHECKS
3070 027074 012706 001100 35 MOV #KERSTK,SP ;RESTORE STACK POINTER
3071 027100 012701 100043 MOV #100043,R1 ;LOAD EXPECTED CONTENTS OR SRO INTO R1
3072 027104 013737 177572 001366 MOV SRO,WASSRO ;READ CONTENTS OF SRO
3073
3074 027112 020137 001366 CMP R1,WASSRO ;DID SRO REPORT ILLEGAL MODE CORRECTLY?
3075 027116 001401 BEQ 45 ;BRANCH IF YES
3076 027120 104057 ERROR 57 ;SRO DID NOT REPORT NR ABORT, PG=1, MODE=01
3077 ;FOR TIGHTER SCOPE LOOP
3078 ;REPLACE ERROR CALL WITH
3079 ;"BR 25" = 000757
3080 027122 012701 027060 45 MOV #25,R1 ;LOAD EXPECTED CONTENTS OF SR2 INTO R1
3081 027126 013737 177576 001370 MOV SR2,WASSR2 ;READ CONTENTS OF SR2
3082 027134 020137 001370 CMP R1,WASSR2 ;DID SR2 LOCKUP VIRT ADDR OF ABORTED INST
3083 027140 001401 BEQ 55 ;BRANCH IF YES
3084 027142 104036 ERROR 36 ;SR2 DID NOT LOCKUP VIRT ADDR OF ILL MODE INST
3085 ;FOR TIGHTER SCOPE LOOP
3086 ;REPLACE ERROR CALL WITH
3087 ;"BR 25" = 000746
3088 027144 042737 160000 177572 55 BIC #160000,SRO ;CLEAR POSSIBLE ERROR BITS IN SRO
3089 027152 012737 002400 000250 MOV #MGHERR,MMVEC ;RESTORE MEM MGMT ABORT VECTOR
3090 027160 012737 027044 001110 MOV #15,SLPERR ;RESET LOOP ON ERROR POINTER TO 15
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103 ..XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
3104 .X
3105 .X THE NEXT TWO (2) TESTS WILL BE CHECKING THE PAGE LENGTH
3106 .X COMPARATORS AND SOME MORE OF THE KT ERROR DETECTION
3107 .X AND STATUS LOGIC. THE PAGE LENGTH FIELD (PLF) IN KERNEL
3108 .X PDR 4 IS VARIED AND FOR EVERY PLF, THREE (3) VIRTUAL
3109 .X ADDRESSES ARE READ. WHILE USING BOTH UPWARD & DOWNWARD PAGE
3110 .X EXPANSION, ONE OF THOSE THREE VIRTUAL ADDRESSES WILL CAUSE A
3111 .X "PAGE LENGTH ABORT" WHILE THE OTHER TWO WON'T
3112 .X
3113 .X STATUS REGISTER 0 & 2 ARE CHECKED WHEN THE PAGE LENGTH
    
```

```

3114 .X ABORT DOES OCCUR TO SEE THAT THE ABORT IS REPORTED AND THAT
3115 .X THE VIRTUAL ADDRESS OF THE INSTRUCTION THAT CAUSED THE ABORT
3116 .X IS LOCKED UP
3117 .X
3118 .X *****
3119
3120
3121
3122 .X *****
3123 .X TEST 36 PAGE LENGTH FAULTS-UPWARD EXPANSION
3124 .X
3125 .X THIS TEST VARIES THE PAGE LENGTH FIELD (PLF) IN KERNEL PDR 4
3126 .X FROM 1 TO 177 AND FOR EACH PLF, THREE VIRTUAL ADDRESSES (VBA'S)
3127 .X ARE ACCESSED. WHEN VBA <12:6> IS LESS THAN OR EQUAL TO PDR <14 8>
3128 .X NO ABORT SHOULD OCCUR. WHEN VBA <12:6> IS GREATER THAN PDR <14 8>.
3129 .X A PAGE LENGTH ABORT SHOULD OCCUR AND BE REPORTED BY SRO & SR2
3130 .X THE PAGE EXPANSION DIRECTION IN THIS TEST IS UPWARD. (THE ED BIT
3131 .X (BIT 3) OF PDR 4 = 0).
3132 .X
3133 .X *****
3134 TST36. SCOPE
3135 027166 000004 15 MOV #77406,KIPDR3 ;MAKE SURE PDR3 IS DESCRIBED AS R/W
3136 027170 012737 077406 172306 MOV #77406,KIPDR5 ;MAKE SURE PDR5 IS DESCRIBED AS R/W
3137 027176 012737 077406 172312 MOV #100000,R0 ;LOAD VIRTUAL ADDR. TO SELECT PDR4 INTO R0
3138 027204 012700 100000 MOV #406,R4 ;LOAD FIRST PDR VALUE IN R4 (PLF=1, ACF=6)
3139 027210 012704 000406 MOV #95,MMVEC ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS
3140 027214 012737 027406 000250 25 MOV R4,KIPDR4 ;LOAD KIPDR4 WITH PAGE LENGTH VALUE
3141 027222 010437 172310 MOV #35,SLPERR ;SET LOOP ON ERROR POINTER TO 35
3142 027226 012737 027234 001110 35 MOV #KERSTK,KSP ;MAKE SURE STACK POINTER IS ALL SET UP
3143 027234 012706 001100 MOV (R0),R1 ;ACCESS VIRTUAL ADDR. (VBA < PLF - NO ABORT)
3144 027240 011001 ADD #100,R0 ;FORM NEXT VIRTUAL ADDRESS IN R0
3145 027242 062700 000100 MOV #45,SLPERR ;SET LOOP ON ERROR POINTER TO 45
3146 027246 012737 027254 001110 45 MOV #KERSTK,KSP ;MAKE SURE STACK POINTER IS ALL SET UP
3147 027254 012706 001100 MOV (R0),R1 ;ACCESS VIRTUAL ADDR. (VBA=PLF - NO ABORT)
3148 027262 062700 000100 ADD #100,R0 ;FORM NEXT VIRTUAL ADDR IN R0
3149 027266 020027 117700 CMP R0,#117700 ;HAVE ALL PLF'S BEEN TESTED YET?
3150 027272 001470 BEQ 105 ;BRANCH IF ALL VBA'S & PLF'S HAVE BEEN USED
3151 027274 012737 027310 001110 MOV #55,SLPERR ;SET LOOP ON ERROR POINTER TO 55
3152 027302 012737 027316 000250 MOV #65,MMVEC ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT
3153 027310 011001 55 MOV (R0),R1 ;ACCESS VIRTUAL ADDR. (VBA > PLF - ABORT TO 65)
3154 027312 104034 ERROR 33 ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
3155 ;FOR TIGHTER SCOPE LOOP
3156 ;REPLACE ERROR CALL WITH
3157 ;"BR 55" = 000776
3158 027314 000424 BR 85 ;BRANCH AROUND ABORT CHECKS
3159 027316 012706 001100 65 MOV #KERSTK,KSP ;RESTORE STACK POINTER FOLLOWING ABORT
3160 027322 013737 177572 001366 MOV SRO,WASSRO ;READ M.M. STATUS REG. 0
3161 027330 013737 177576 001370 MOV SR2,WASSR2 ;READ M.M. STATUS REG. 2
3162 027336 012702 040011 MOV #40011,R2 ;PUT EXPECTED SRO CONTENTS IN R2
3163 027342 020237 001366 CMP R2,WASSRO ;DID SRO REPORT PG. LENGTH ABORT, PAGE 4, KERNEL?
3164 027346 001401 BEQ 75 ;BRANCH IF YES
3165 027350 104034 ERROR 34 ;SRO DID NOT REPORT PG. LENGTH ABORT CORRECTLY
3166 ;FOR TIGHTER SCOPE LOOP
3167 ;REPLACE ERROR CALL WITH
3168 ;"BR 55" = 000757
3169 027352 012703 027310 75 MOV #55,R3 ;PUT EXPECTED SR2 CONTENTS IN R3
    
```



```

3170 027356 020337 001370      CMP      R3,WASSR2      ;DID SR2 LOCKUP VIRT ADDR OF ABORTED INSTRUCTION?
3171 027362 001401              BEQ      85              ;BRANCH IF YES
3172 027364 104035              ERROR   35              ;SR2 DID NOT LOCKUP VIRT ADDR OF ABORT CORRECTLY
3173                                ;FOR TIGHTER SCOPE LOOP
3174                                ;REPLACE ERROR CALL WITH
3175                                ;"BR 55" = 000751
3176 027366 042737 160000 177572 85      BIC      #160000,SRO      ;CLEAR ERROR BITS IN SRO
3177 027374 062704 000400              ADD      #400,R4          ;FORM NEXT PLF VALUE FOR KIPDR4
3178 027400 162700 000100              SUB      #100,R0          ;FORM FIRST VIRT. ADDR FOR THAT PLF VALUE
3179 027404 000703              BR       25              ;BRANCH BACK AND ACCESS 3 VBA'S FOR
3180                                ;THE PLF VALUE JUST FORMED
3181 027406 012637 001356      95.     MOV      (KSP)+,TRAPPC      ;SAVE PC & PS OF TRAP
3182 027412 012637 001360              MOV      (KSP)+,TRAPPS
3183 027416 013737 177572 001366              MOV      SRO,WASSR0      ;SAVE CONTENTS OF SRO FOR ERROR
3184 027424 013737 177576 001370              MOV      SR2,WASSR2      ;SAVE CONTENTS OF SR2 FOR ERROR
3185 027432 042737 160000 177572              BIC      #160000,SRO      ;CLEAR ERROR BITS IN SRO
3186 027440 104032              ERROR   32              ;GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
3187                                ;FOR TIGHTER SCOPE LOOP
3188                                ;REPLACE ERROR CALL WITH
3189                                ;A "NOP" = 000240
3190 027442 013746 001360              MOV      TRAPPS,-(KSP)    ;PUT PC & PS OF TRAP ON STACK
3191 027446 013746 001356              MOV      TRAPPC,-(KSP)
3192 027452 000002              RTI                          ;RETURN FROM UNEXPECTED ABORT
3193
3194 027454 012737 027170 001110 105.     MOV      #15,$LPERR      ;RESET LOOP ON ERROR POINTER TO 15
3195 027462 012737 002400 000250              MOV      #MGMERR,MNVEC    ;RESTORE NORMAL M.M. TRAP HANDLER
3196                                ;ADDRESS TO M.M. TRAP VECTOR
3197
3198
3199

```

 ;TEST 37 PAGE LENGTH FAULTS-DOWNWARD EXPANSION

```

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

```

```

3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211 027470 000004              TST37: SCOPE
3212 027472 012700 117700      15:     MOV      #117700,R0      ;LOAD VIRTUAL ADDR. TO SELECT PDR4 INTO R0
3213 027476 012704 077016              MOV      #77016,R4       ;LOAD FIRST PDR VALUE IN R4 (PLF=176,ACF=6)
3214 027502 012737 027674 000250 25:     MOV      #95,MNVEC      ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS
3215 027510 010437 172310              MOV      R4,KIPDR4      ;LOAD KIPDR4 WITH PAGE LENGTH VALUE
3216 027514 012737 027522 001110              MOV      #35,$LPERR     ;SET LOOP ON ERROR POINTER TO 35
3217 027522 012706 001100      35.     MOV      #KERSTK,KSP     ;MAKE SURE STACK POINTER IS ALL SET UP
3218 027526 011001              MOV      (R0),R1        ;ACCESS VIRTUAL ADDR. (VBA > PLF - NO ABORT)
3219 027530 162700 000100              SUB      #100,R0        ;FORM NEXT VIRTUAL ADDRESS IN R0
3220 027534 012737 027542 001110              MOV      #45,$LPERR     ;SET LOOP ON ERROR POINTER TO 45
3221 027542 012706 001100      45.     MOV      #KERSTK,KSP     ;MAKE SURE STACK POINTER IS ALL SET UP
3222 027546 011001              MOV      (R0),R1        ;ACCESS VIRTUAL ADDR. (VBA=PLF - NO ABORT)
3223 027560 162700 000100              SUB      #100,R0        ;FORM NEXT VIRTUAL ADDR. IN R0
3224 027564 020027 100000              CMP      R0,#100000     ;HAVE ALL PLF'S BEEN TESTED YET?
3225 027560 001470              BEQ      105            ;BRANCH IF ALL VBA'S & PLF'S HAVE BEEN USED

```

```

3226 027562 012737 027576 001110      MOV      855,SLPERR      ;SET LOOP ON ERROR POINTER TO 55
3227 027570 012737 027604 000250      MOV      865,MWVEC      ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT
3228 027576 011001          55      MOV      (R0),R1        ;ACCESS VIRTUAL ADDR. (VBA < PLF - ABORT TO 65)
3229 027600 104033          ERROR    33            ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
3230                                     ;FOR TIGHTER SCOPE LOOP
3231                                     ;REPLACE ERROR CALL WITH
3232                                     ;"BR 55" = 000776
3233 027602 000424          65      BP      85            ;BRANCH AROUND ABORT CHECKS
3234 027604 012706 001100      MOV      8KERSTK,KSP    ;RESTORE STACK POINTER FOLLOWING ABORT
3235 027610 013737 177572 001366      MOV      SRO,WASSRO     ;READ M.M. STATUS REG. 0
3236 027616 013737 177576 001370      MOV      SR2,WASSR2     ;READ M.M. STATUS REG. 2
3237 027624 012702 040011      MOV      840011,R2      ;PUT EXPECTED SRO CONTENTS IN R2
3238 027630 020237 001366      CMP      R2,WASSRO      ;DID SRO REPORT PG. LENGTH ABORT, PG. 4, KERNEL?
3239 027634 001401          BEQ      75            ;BRANCH IF YES
3240 027636 104034          ERROR    34            ;SRO DID NOT REPORT PG. LENGTH ABORT CORRECTLY
3241                                     ;FOR TIGHTER SCOPE LOOP
3242                                     ;REPLACE ERROR CALL WITH
3243                                     ;"BR 55" = 000757
3244 027640 012703 027576 75      MOV      855,R3         ;PUT EXPECTED SR2 CONTENTS IN R3
3245 027644 020337 001370      CMP      R3,WASSR2     ;DID SR2 LOCKUP VIRT ADDR OF ABORTED INSTRUCTION?
3246 027650 001401          BEQ      85            ;BRANCH IF YES
3247 027652 104035          ERROR    35            ;SR2 DID NOT LOCKUP VIRT ADDR OF ABORT CORRECTLY
3248                                     ;FOR TIGHTER SCOPE LOOP
3249                                     ;REPLACE ERROR CALL WITH
3250                                     ;"BR 55" = 000751
3251 027654 042737 160000 177572 85      BIC      816000,SRO     ;CLEAR ERROR BITS IN SRO
3252 027662 162704 000400      SUB      8400,R4        ;FORM NEXT PLF VALUE FOR KIPDR4
3253 027666 062700 000100      ADD      8100,R0        ;FORM FIRST VIRT. ADDR. FOR THAT PLF VALUE
3254 027672 000703          BR      25            ;BRANCH BACK AND ACCESS 3 VBA'S FOR
3255                                     ;THE PLF VALUE JUST FORMED
3256 027674 012637 001356 95      MOV      (KSP)+,TRAPPC  ;SAVE PC & PS OF TRAP
3257 027700 012637 001360      MOV      (KSP)+,TRAPPS
3258 027704 013737 177572 001366      MOV      SRO,WASSRO     ;SAVE CONTENTS OF SRO FOR ERROR
3259 027712 013737 177576 001370      MOV      SR2,WASSR2     ;SAVE CONTENTS OF SR2 FOR ERROR
3260 027720 042737 160000 177572      BIC      816000,SRO     ;CLEAR ERROR BITS IN SRO
3261 027726 104032          ERROR    32            ;GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
3262                                     ;FOR TIGHTER SCOPE LOOP
3263                                     ;REPLACE ERROR CALL WITH
3264                                     ;A "NOP" = 000240
3265 027730 013746 001360      MOV      TRAPPS,-(KSP)  ;PUT PC & PS OF TRAP ON STACK
3266 027734 013746 001356      MOV      TRAPPC,-(KSP)
3267 027740 000002          RTI                    ;RETURN FROM UNEXPECTED ABORT
3268
3269 027742 012737 027472 001110 105     MOV      815,SLPERR     ;RESET LOOP ON ERROR POINTER TO 15
3270 027750 012737 002400 000250      MOV      8MGERR,MWVEC   ;RESTORE NORMAL M.M. TRAP HANDLER
3271                                     ;ADDRESS TO M.M. TRAP VECTOR
3272
3273
3274 //*****
3275 *TEST 40      SR2 BIT LIST
3276 *
3277 *
3278 *      THIS TEST CHECKS THE BITS IN MEMORY MANAGEMENT REGISTER 2 BY
3279 *      CAUSING "READ-ONLY ABORTS" AT VIRTUAL ADDRESSES BETWEEN 100000
3280 *      TO 111000 (PHYSICAL ADDRESSES 060000-071000) KIPDR4 IS USED TO EXECUTE
3281 *      THE FOLLOWING FOUR WORDS OF CODE WHICH ARE MOVED THRU MEMORY
3282 *      010727 MOV      PC,(PC)+ ;THIS INSTRUCTION SHOULD CAUSE A R/O ABORT
    
```

```

3282 .x 000000 . ITS VIRTUAL ADDR SHOULD BE LOCKED UP IN SR2
3283 .x 000137 JMP @#35 . THIS INSTRUCTION IS ALSO MOVED THRU MEMORY
3284 .x (ADDR OF 35) . IN CASE A R/O ABORT DOES NOT OCCUR,
3285 .x . IN WHICH CASE SR2 WILL NOT CONTAIN CORRECT ADDR
3286 .x
3287 .x *****
3288 027756 000004 TST40 SCOPE
3289 027760 012737 000600 172346 15 MOV #600,KIPAR3 ;BE SURE PAR3 IS MAPPED TO 12-16K
3290 027766 012737 000600 172350 15 MOV #600,KIPAR4 ;BE SURE PAR4 IS MAPPED TO 12-16K
3291 027774 012737 077406 172306 15 MOV #77406,KIPDR3 ;MAP PAGE 3 128 BLOCKS, R/W
3292 030002 012737 077402 172310 15 MOV #77402,KIPDR4 ;MAP PAGE 4 128 BLOCKS, READ-ONLY
3293 030010 012700 060000 15 MOV #60000,R0 ;LOAD R0 WITH VIRTUAL ADDR. WHICH USES PDR3
3294 030014 012701 100000 15 MOV #100000,R1 ;LOAD R1 WITH VIRTUAL ADDR. WHICH USES PDR4
3295 030020 012737 030054 000250 15 MOV #35,MHVEC ;SET M.M TRAP VECTOR TO 35
3296 030026 012737 030034 001110 15 MOV #25,$LPERR ;SET LOOP ON ERROR POINTER TO 25
3297 030034 012720 010727 25 MOV #010727,(R0)+ ;LOAD "MOV PC,(PC)+" INSTRUCTION AT ADDR
3298 030040 005020 15 CLR (R0)+ ; REACHED THRU PDR/PAR 4.
3299 030042 012720 000137 15 MOV #000137,(R0)+ ;LOAD "JMP @#35" INSTRUCTION AT VIRT. ADDR
3300 030046 012710 030054 15 MOV #35,(R0) ; IN CASE R/O VIOL. DOES NOT ABORT
3301 030052 010107 15 MOV R1,PC ;TRANSFER PROGRAM EXECUTION TO "PAGE 4 INSTRUCTIONS"
3302 030054 012706 001100 35 MOV #KERSTK,KSP ;RESTORE STACK POINTER
3303 030060 013737 177576 001370 15 MOV SR2,WASSR2 ;READ CONTENTS OF STATUS REG 2
3304 030066 020137 001370 15 CMP R1,WASSR2 ;WAS ADDR. OF "RELOCATED - R/O ABORT" LOCKED UP?
3305 030072 001401 15 BEQ 45 ;BRANCH IF YES
3306 030074 104036 15 ERROR 36 ;SR2 DID NOT LOCK UP VIRTUAL ADDR OF R/O VIOL
3307 .x
3308 .x
3309 .x
3310 030076 042737 160000 177572 45 BIC #160000,SRO ;CLEAR THE ERROR BITS IN SRO
3311 030104 162700 000004 15 SUB #4,R0 ;RESET R0 TO POINT TO NEXT VIRT ADDR TO LOAD
3312 030110 062701 000002 15 ADD #2,R1 ;FORM VIRTUAL ADDR. THAT SHOULD BE LOCKED UP NEXT
3313 030114 020127 111002 15 CMP R1,#111002 ;HAVE ALL VBA'S 10000-111000 BEEN TESTED?
3314 030120 103745 15 BLO 25 ;BRANCH IF NO
3315 .x
3316 030122 012737 027760 001110 55 MOV #15,$LPERR ;RESET LOOP ON ERROR POINTER TO 15
3317 030130 012737 077406 172310 15 MOV #77406,KIPDR4 ;RESTORE PDR4 TO R/W ACCESS
3318 030136 012737 002400 000250 15 MOV #BMGMERR,MHVEC ;RESTORE ADDRESS OF NORMAL M.M
3319 .x ;TRAP HANDLER TO M.M VECTOR
3320 .x
3321 .x
    
```

3322
 3323
 3324
 3325
 3326
 3327
 3328
 3329
 3330
 3331
 3332
 3333
 3334
 3335
 3336
 3337
 3338
 3339
 3340
 3341
 3342
 3343
 3344
 3345
 3346
 3347
 3348
 3349
 3350
 3351
 3352
 3353
 3354
 3355
 3356
 3357
 3358
 3359
 3360
 3361
 3362
 3363
 3364
 3365
 3366
 3367
 3368
 3369
 3370
 3371
 3372
 3373
 3374
 3375
 3376
 3377

```

    .. *****
    .TEST 41      MORE CHECKS OF SRO & SR2
    .
    . THIS TEST PERFORMS SOME ADDITIONAL CHECKS OF THE SRO & SR2 LOGIC
    . FIRST IT CHECKS THAT SR2 "TRACKS" ALONG ACTING AS A VIRTUAL ADDRESS
    . PROGRAM COUNTER. ALSO SRO & SR2 ARE LOCKED UP BY A PAGE LENGTH
    . ABORT, THEN WITHOUT CLEARING SRO'S ERROR BITS, A R/O ABORT IS CAUSED
    . SRO & SR2 SHOULD NOT BE CHANGED BY THE SECOND ABORT AND THE
    . INFORMATION ABOUT THE PAGE LENGTH ABORT SHOULD STILL BE LOCKED UP
    . IN ADDITION A "RESET" IS EXECUTED TO VERIFY THAT SRO IS CLEARED
    . AND SR2 IS UNLOCKED BY A RESET. AFTER MEMORY MANAGEMENT IS TURNED BACK ON,
    . SR2 IS CHECKED TO SEE THAT IT IS TRACKING AGAIN.
    .
    .. *****
    TST41  SCOPE
    15     MOV      #600,KIPAR5  ,MAP KERNEL PAGE 5 TO 12-16K
    15     MOV      #406,KIPDR4  ;SETUP PDR4 FOR PAGE LENGTH ABORT
    15     MOV      #77402,KIPDR5 ;SETUP PDR5 FOR R/O ABORT
    25     MOV      #25,$LPERR    ;SET LOOP ON ERROR POINTER TO 25
    25     MOV      SR2,WASSR2    ;READ SR2 TO SEE IF ITS TRACKING
    25     MOV      #25,R1        ;PUT EXPECTED VIRTUAL PC IN R1
    25     CMP      R1,WASSR2     ;DID SR2 CONTAIN VIRTUAL PC AT 25?
    35     BEQ      35            ;BRANCH IF YES
    41     ERROR    41           ;SR2 NOT TRACKING CORRECTLY
    .
    . FOR TIGHTER SCOPE LOOP
    . REPLACE ERROR CALL WITH
    . "BR 25" = 000767
    35     MOV      #45,$LPERR    ;SET LOOP ON ERROR POINTER TO 45
    45     MOV      SR2,WASSR2    ;READ SR2 TO SEE IF ITS TRACKING
    45     MOV      #45,R1        ;PUT EXPECTED VIRTUAL PC IN R1
    45     CMP      R1,WASSR2     ;DID SR2 CONTAIN VIRTUAL PC AT 45
    55     BEQ      55            ;BRANCH IF YES
    41     ERROR    41           ;SR2 NOT TRACKING CORRECTLY
    .
    . FOR TIGHTER SCOPE LOOP
    . REPLACE ERROR CALL WITH
    . "BR 45" = 000767
    55     MOV      #65,$LPERR    ;SET LOOP ON ERROR POINTER TO 65
    65     MOV      #75,MWVEC     ;PUT ADDRESS OF 75 IN M.M TRAP VECTOR
    75     CLR      $TMP1         ;CLEAR ERROR INDICATOR
    75     INC      @#100500      ;CAUSE PAGE LENGTH ABORT - TRAP TO 75
    75     MOV      @KERSTK,KSP   ;RESTORE STACK POINTER AFTER ABORT
    75     MOV      SRO,$TMP0     ;SAVE SRO'S INFORMATION ON PG. LGTH ABORT
    75     MOV      SR2,$TMP2     ;SAVE SR2'S INFORMATION ON PG. LGTH ABORT
    75     MOV      #85,MWVEC     ;PUT ADDRESS OF 85 IN M.M TRAP VECTOR
    85     INC      @#120000      ;CAUSE R/O ABORT - TRAP TO 85
    85     MOV      @KERSTK,KSP   ;RESTORE STACK POINTER AFTER ABORT
    85     MOV      SRO,WASSR0    ;READ SRO FOLLOWING SECOND KT ABORT
    85     MOV      SR2,WASSR2    ;READ SR2 FOLLOWING SECOND KT ABORT
    85     CMP      $TMP0,WASSR0  ;IS SRO STILL HOLDING INFO ON FIRST ABORT?
    95     BEQ      95            ;BRANCH IF YES
    95     INC      $TMP1         ;SET ERROR INDICATOR
    95     CMP      $TMP2,WASSR2  ;DOES SR2 STILL HOLD PC OF FIRST ABORT?
    105    BEQ      105           ;BRANCH IF YES
    105    INC      $TMP1         ;SET ERROR INDICATOR
    
```

3378	030376	005737	001200		105	TST	STMP1	, WERE SRO OR SR2 CHANGED BY A SECOND ABORT?
3379	030402	001401				BEQ	115	, BRANCH IF NO
3380	030404	104037				ERROR	37	, ONE OF STATUS REGS. CHANGED BY SECOND ABORT
3381								, FOR TIGHTER SCOPE LOOP
3382								, REPLACE ERROR CALL WITH
3383								, "BR 65" = 000726
3384	030406	005037	001200		115	CLR	STMP1	, CLEAR ERROR INDICATOR
3385	030412	000005				RESET		, EXECUTE A RESET, APPLYING AN "INIT"
3386	030414	013737	177572	001366		MOV	SRO, WASSRO	, READ SRO
3387	030422	005737	001366			TST	WASSRO	, WAS SRO CLEARED BY THE RESET?
3388	030426	001402				BEQ	125	, BRANCH IF YES
3389	030430	005237	001200			INC	STMP1	, SRO NOT CLEARED BY A RESET
3390	030434	013737	177576	001370	125	MOV	SR2, WASSR2	, READ SR2
3391	030442	022737	030434	001370		CMP	#125, WASSR2	, WAS SR2 UNLOCKED BY A RESET?
3392	030450	001402				BEQ	135	, BRANCH IF YES
3393	030452	005237	001200			INC	STMP1	, SR2 NOT UNLOCKED BY A RESET
3394	030456	005737	001200		135	TST	STMP1	, WERE SRO & SR2 BOTH "RESET" BY A RESET?
3395	030462	001401				BEQ	145	, BRANCH IF YES
3396	030464	104040				ERROR	40	, SRO OR SR2 NOT "RESET" BY A RESET
3397								, FOR TIGHTER SCOPE LOOP
3398								, REPLACE ERROR CALL WITH
3399								, "BR 65" = 000676
3400	030466	005237	177572		145	INC	SRO	, TURN MEMORY MANAGEMENT BACK ON
3401	030472	013737	177576	001370	155	MOV	SR2, WASSR2	, READ SR2 TO SEE IF ITS TRACKING AGAIN
3402	030500	012701	030472			MOV	#155, R1	, PUT EXPECTED VIRTUAL PC IN R1
3403	030504	020137	001370			CMP	R1, WASSR2	, DID SR2 CONTAIN VIRTUAL PC AT 155
3404	030510	001401				BEQ	165	, BRANCH IF YES
3405	030512	104041				ERROR	41	, SR2 NOT TRACKING CORRECTLY
3406								, FOR TIGHTER SCOPE LOOP
3407								, REPLACE ERROR CALL WITH
3408								, "BR 65" = 000663
3409								, *****
3410								,
3411								, ALL CODE WITH A '+' DESIGNATION IN THE COMMENT INDICATES THAT IT HAS BEEN
3412								, ADDED TO THE ORIGINAL REV OF THIS DIAGNOSTIC THESE ENHANCEMENTS OR CORRECTIONS
3413								, ARE MADE BY THE PRODUCT ENHANCEMENT GROUP (DIAGNOSTICS)
3414								, *****
3415								,
3416								,
3417								,
3418								,
3419	030514	012737	030574	000250	165	MOV	#215, MMVEC	, +SET UP KTVEC FOR NON-RESIDENT ABORT
3420	030522	012737	077400	172312		MOV	#77400, KIPDR5	, +SET UP KIPDR5 FOR NR ABORT
3421	030530	012737	030536	001110		MOV	#175, SLPERR	, +SET LOOP ON ERROR POINTER
3422	030536	005037	001200		175	CLR	STMP1	, +CLEAR ERROR INDICATOR
3423	030542	005237	130000		205	INC	#130000	, +CAUSE A NON-RESIDENT ABORT
3424	030546	005237	001200			INC	STMP1	, +SET ERROR INDICATOR
3425	030562	013737	177572	001366		MOV	SRO, WASSRO	, +READ SRO
3426	030560	013737	177576	001370		MOV	SR2, WASSR2	, +READ SR2
3427	030566	104065				ERROR	65	, +NON-RESIDENT ABORT DID NOT OCCUR
3428								,
3429								,
3430	030570	005037	001200			CLR	STMP1	, +CLEAR ERROR INDICATOR
3431	030574	012706	001100		215	MOV	#KERSTK, KSP	, +RESTORE KERNEL STACK POINTER
3432	030600	005737	177572		225	TST	SRO	, +TEST FOR THE NR ABORT IN SRO (BIT 15)
3433	030604	100413				BMI	235	, +IF SET BRANCH

J434	030606	005237	001200			INC	\$TMP1	,+SET ERROR INDICATOR
J435	030612	013737	177572	001366		MOV	SRO,WASSRO	,+READ SRO
J436	030620	013737	177576	001370		MOV	SR2,WASSR2	,+READ SR2
J437	030626	104066				ERROR	66	,+NR ABORT ERROR DID NOT SET IN SRO (BIT 15)
J438								
J439								
J440	030630	005037	001200			CLR	\$TMP1	,+CLEAR ERROR INDICATOR
J441	030634	023727	177576	030542	235	CMP	SR2,#205	,+TEST THAT SR2 FROZE TO THE VIRTUAL
J442								,+ADDRESS OF THE NR ABORT INSTR
J443	030642	001413				BEQ	245	,+IF EQUAL BRANCH
J444	030644	005237	001200			INC	\$TMP1	,+ELSE ERROR
J445	030650	013737	177572	001366		MOV	SRO,WASSRO	,+READ SRO
J446	030656	013737	177576	001370		MOV	SR2,WASSR2	,+READ SR2
J447	030664	104067				ERROR	67	,+SR2 DID NOT CONTAIN THE CORRECT ADDRESS
J448								
J449								
J450	030666	005037	001200			CLR	\$TMP1	,+CLEAR ERROR INDICATOR
J451	030672	012737	030732	000250	245	MOV	#255,MMVEC	,+SET KTVEC FOR 2ND NR ABORT
J452	030700	005237	130000			INC	#130000	,+CAUSE ABORT
J453	030704	005237	001200			INC	\$TMP1	,+SET ERROR INDICATOR
J454	030710	013737	177572	001366		MOV	SRO,WASSRO	,+READ SRO
J455	030716	013737	177576	001370		MOV	SR2,WASSR2	,+READ SR2
J456	030724	104070				ERROR	70	,+2ND NR ABORT DID NOT OCCUR
J457								
J458								
J459	030726	005037	001200			CLR	\$TMP1	,+CLEAR ERROR INDICATOR
J460	030732	012706	001100		255	MOV	#KERSTK,KSP	,+RESTORE KERNEL STACK
J461	030736	013737	177576	001370		MOV	SR2,WASSR2	,+READ SR2
J462	030744	023727	001370	030542		CMP	WASSR2,#205	,+TEST SR2 FOR VIRTUAL ADDRESS OF
J463								,+OF THE FIRST NON-RESIDENT ABORT
J464	030752	001413				BEQ	265	,+IF EQUAL BRANCH
J465								,+ELSE ERROR
J466	030754	005237	001200			INC	\$TMP1	,+SET ERROR INDICATOR
J467	030760	013737	177572	001366		MOV	SRO,WASSRO	,+READ SRO
J468	030766	012737	030542	001364		MOV	#205,CORSR2	,+READ CORRECT SR2 VALUE
J469	030774	104071				ERROR	71	,+SR2 DID NOT CONTAIN THE VALUE OF
J470								,+THE FIRST NR ABORT
J471								
J472								
J473	030776	005037	001200			CLR	\$TMP1	,+CLEAR ERROR INDICATOR
J474	031002	000005			265	RESET		,+ISSUE INIT CONDITIONS
J475	031004	005737	177572			TST	SRO	,+SRO SHOULD BE CLEAR
J476	031010	001412				BEQ	275	,+IF TRUE BRANCH
J477								,+ELSE ERROR
J478	031012	005237	001200			INC	\$TMP1	,+SET ERROR INDICATOR
J479	031016	013737	177572	001366		MOV	SRO,WASSRO	,+READ SRO
J480	031024	005037	001362			CLR	CORSRO	,+SET CORRECT SRO VALUE FOR ERROR MSG
J481	031030	104072				ERROR	72	,+SRO DID NOT CLEAR AFTER INIT
J482								
J483								
J484								
J485	031032	005037	001200			CLR	\$TMP1	,+CLEAR ERROR INDICATOR
J486	031036	005237	177572		275	INC	SRO	,+TURN MEM MANG ON
J487	031042	013737	177576	001370	305	MOV	SR2,WASSR2	,+READ SR2
J488	031050	023727	001370	031042		CMP	WASSR2,#305	,+IS SR2 TRACKING AGAIN
J489	031056	001405				BEQ	315	,+YES, BRANCH

```

3490
3491 031060 005237 001200      INC      $TMP1      ;+ELSE ERROR
3492 031064 012701 031042      MOV      #305,R1   ;+SET ERROR INDICATOR
3493 031070 104041              ERROR    41        ;+SET CORRECT SR2 VALUE FOE ERROR MSG
                                     ;+ERROR,SR2 IS NOT TRACKING CORRECTLY
3494
3495
3496 031072 012737 030146 001110 315  MOV      #15,$LPERR ;RESET LOOP ON ERROR POINTER TO 15
3497 031100 012737 077406 172310      MOV      #77406,KIPDR4 ;RESET PDR4 TO 128 BLKS, R/W
3498 031106 012737 077406 172312      MOV      #77406,KIPDR5 ;RESET PDR5 TO 128 BLKS, R/W
3499 031114 012737 002400 000250      MOV      #8MGERR,MMVE ;RESTORE ADDRESS OF NORMAL MEMORY
3500                                     ;MANAGEMENT TRAP ROUTINE TO M M VECTOR
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515 031122 000004              TST42   SCOPE
3516 031124 004737 036002      15     JSR      PC,TOFF ;TURN OFF T-BIT TRAPPING FOR THIS TEST
3517 031130 012737 031136 001110      MOV      #25,$LPERR ;SET LOOP ON ERROR POINTER TO 25
3518 031136 005037 177776      25     CLR      PSW      ;GO TO KERNEL MODE
3519 031142 012706 001100      MOV      #KERSTK,KSP ;SETUP KERNEL STACK PTR
3520 031146 012737 000600 177640      MOV      #600,UIPARO ;MAP USER PAGE 0 TO 12K
3521 031154 012737 031236 000004      MOV      #45,$#4    ;LOAD KERNEL VECTOR 4 (LOC 4) WITH 45
3522 031162 012737 000340 000006      MOV      #340,$#6   ;LOAD VECTOR+2 WITH NEW PSW
3523 031170 012737 140000 177776      MOV      #140000,PSW ;GO TO USER MODE
3524 031176 012706 000700      MOV      #USESTK,USP ;SETUP USER STACK PTR
3525 031202 012737 031222 000004      MOV      #35,$#4    ;LOAD USER VECTOR 4 (LOC 60004) WITH 35
3526 031210 012737 000340 000006      MOV      #340,$#6   ;LOAD VECTOR+2 WITH NEW PSW
3527 031216 005737 031223      TST     35+1      ;CAUSE ODD ADDR. ERROR TRAP TO "4"
3528                                     ;SHOULD PICK UP NEW PC=45 FROM KERNEL
3529                                     ;LOC. 4, NOT PC=35 FROM USER LOC 4 (=60004)
3530 031222 013701 177776      35     MOV      PSW,R1   ;SAVE PSW FOR ERROR
3531 031226 010602              MOV      SP,R2    ;SAVE VALUE OF STACK POINTER FOR ERROR
3532 031230 005037 177776      CLR      PSW      ;BE SURE BACK IN KERNEL MODE
3533 031234 104042              ERROR    42        ;DID NOT TRAP THRU KERNEL SPACE
3534                                     ;FOR TIGHTER SCOPE LOOP
    
```

*TEST 42 USER ABORT PICKS UP KERNEL SPACE VECTOR

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

TST42 SCOPE
 15 JSR PC,TOFF ;TURN OFF T-BIT TRAPPING FOR THIS TEST
 MOV #25,\$LPERR ;SET LOOP ON ERROR POINTER TO 25
 25 CLR PSW ;GO TO KERNEL MODE
 MOV #KERSTK,KSP ;SETUP KERNEL STACK PTR
 MOV #600,UIPARO ;MAP USER PAGE 0 TO 12K
 MOV #45,\$#4 ;LOAD KERNEL VECTOR 4 (LOC 4) WITH 45
 MOV #340,\$#6 ;LOAD VECTOR+2 WITH NEW PSW
 MOV #140000,PSW ;GO TO USER MODE
 MOV #USESTK,USP ;SETUP USER STACK PTR
 MOV #35,\$#4 ;LOAD USER VECTOR 4 (LOC 60004) WITH 35
 MOV #340,\$#6 ;LOAD VECTOR+2 WITH NEW PSW
 TST 35+1 ;CAUSE ODD ADDR. ERROR TRAP TO "4"
 ;SHOULD PICK UP NEW PC=45 FROM KERNEL
 ;LOC. 4, NOT PC=35 FROM USER LOC 4 (=60004)
 35 MOV PSW,R1 ;SAVE PSW FOR ERROR
 MOV SP,R2 ;SAVE VALUE OF STACK POINTER FOR ERROR
 CLR PSW ;BE SURE BACK IN KERNEL MODE
 ERROR 42 ;DID NOT TRAP THRU KERNEL SPACE
 ;FOR TIGHTER SCOPE LOOP

```

3535
3536
3537 031236 005037 177776      45      CLR      PSW
3538 031242 012706 06110C      MOV      #KERSTK,KSP
3539 031246 005037 177640      CLR      UIPARO
3540 031252 012737 140000 177776      MOV      #140000,PSW
3541 031260 012706 000700      MOV      #USESTK,USP
3542 031264 005037 177776      CLR      PSW
3543 031270 012737 002332 000004      MOV      #TIMERR,204
3544 031276 012737 031124 001110      MOV      #15,$LPERR
3545 031304 004737 036036      JSR      PC,T0N
,REPLACE ERROR CALL WITH
,"BR 25" = 000740
,BE SURE BACK IN KERNEL MODE
,RESTORE KERNEL S.P. IN CASE IT CHANGED
,REMAP USER PAGE 0 TO 0-4K
,GO TO USER MODE
,RESTORE USER STACK POINTER
,GO BACK TO KERNEL MODE
,RESTORE ADDR. OF NORMAL CPU TRAP HANDLER TO 4
,RESET LOOP ON ERROR POINTER TO 15
,TURN T-BIT TRAPPING BACK ON

3546
3547
3548
3549
3550
3551
3552
3553
3554 031310 000004
3555
3556 031312 012737 031324 001110 15      MOV      #25,$LPERR
3557 031320 012702 170000      MOV      #170000,R2
3558 031324 010237 177776      MOV      R2,PSW
3559 031330 012746 000340      MOV      #340,-(SP)
3560 031334 012746 031342      MOV      #35,-(SP)
3561 031340 000002      RTI
3562 031342 013701 177776      MOV      PSW,R1
3563 031346 042701 007437      BIC      #7437,R1
3564 031352 005037 177776      CLR      PSW
3565 031356 020201      CMP      R2,R1
3566 031360 001401      BEQ     45
3567 031362 104060      ERROR   60
,SET LOOP ON ERROR POINTER TO 25
,LOAD "PRESENT & EXPECTED" PSW VALUE INTO R2
,GO TO USER MODE-PRIORITY 0
,PUT A NEW PSW (PRIORITY=7) ON STACK
,PUT NEW PC ON THE STACK
,DO AN RTI FROM USER MODE
,REFD NEW PSW INTO R1
,MASK OFF COND. CODE, T-BIT, AND UNUSED BITS
,GO BACK TO KERNEL MODE
,DID PSW STAY IN USER, PRIORITY=0?
,BRANCH IF YES
,PSW CHANGED BY AN RTI FROM USER
,FOR A TIGHTER SCOPE LOOP
,REPLACE ERROR CALL WITH
,"BR=25" = 000760
,RESET LOOP ON ERROR POINTER TO 15

3568
3569
3570
3571 031364 012737 031312 001110 45      MOV      #15,$LPERR
3572
3573

```



```

3574 .. *****
3575 .. *TEST 44      KT ERROR NOT SERVICED IF ODD ADDR  ERROR
3576 .. *
3577 .. *
3578 .. *      THIS TEST CHECKS TO SEE THAT IF A CERTAIN VIRTUAL ADDRESS THAT
3579 .. *      WOULD CAUSE A MEMORY MANAGEMENT ERROR CAUSES AN ODD ADDRESS
3580 .. *      ERROR FIRST, THE ODD ADDRESS ERROR IS SERVICED BUT THE MEMORY
3581 .. *      MANAGEMENT ERROR ISN'T. THIS MEANS THAT SRO AND SR2
3582 .. *      SHOULD NOT REPORT THE ERROR OR LOCK UP ITS VIRTUAL ADDRESS
3583 .. *      A READ-ONLY VIOLATION IS USED AS THE POTENTIAL MEMORY MANAGEMENT
3584 .. *      ERROR
3585 .. *****
3586 031372 000004 TST44 SCOPE
3587 031374 012737 000600 172350 15 MOV #600,KIPDR4 ;MAP KERNEL PAGE 4 TO 12-16K
3588 031402 012705 077402 MOV #77402,R5 ;LOAD PDR4 DATA INTO R5
3589 031406 010537 172310 MOV R5,KIPDR4 ;MAP PAGE 4 READ-ONLY
3590 031412 012737 031442 000004 MOV #45,284 ;SET CPU TRAP VECTOR TO ADDRESS OF 45
3591 031420 012737 031440 000250 MOV #35,28250 ;SET M.M. TRAP VECTOR TO ADDRESS OF 35
3592 031426 012737 031434 001110 MOV #25,$LPERR ;SET LOOP ON ERROR POINTER TO 25
3593 031434 005237 060001 25 INC 60001 ;CAUSE ODD ADDR. ERROR & POTENTIAL R/O ABORT
3594 031440 104043 35 ERROR 43 ;TRAPPED THRU M.M. VECTOR BUT SHOULDN'T HAVE
3595 ; FOR TIGHTER SCOPE LOOP
3596 ; REPLACE ERROR CALL WITH
3597 ; "BR 25" = 000776
3598 031442 012706 001100 45 MOV #KERSTK,KSP ;RESTORE STACK POINTER AFTER TRAPPING
3599 031446 005037 001200 CLR $TMP1 ;CLEAR ERROR INDICATOR
3600 031452 013737 177572 001366 MOV SRO,WASSRO ;READ STATUS REG. 0
3601 031460 013737 177576 001370 55 MOV SR2,WASSR2 ;READ STATUS REG. 2
3602 031466 012700 000017 MOV #17,RO ;LOAD EXPECTED SRO CONTENTS INTO RO
3603 031472 020037 001366 CMP RO,WASSRO ;SRO ERROR BITS LEFT CLEAR BY TRAPPING?
3604 031476 001402 BEQ 65 ;BRANCH IF YES
3605 031500 005237 001200 INC $TMP1 ;SRO ERROR BITS SET WHEN ODD ADDR SERVICED
3606 031504 012701 031460 65 MOV #55,R1 ;LOAD EXPECTED SR2 CONTENTS INTO R1
3607 031510 020137 001370 CMP R1,WASSR2 ;WAS SR2 LEFT UNLOCKED BY TRAPPING?
3608 031514 001402 BEQ 75 ;BRANCH IF YES
3609 031516 005237 001200 INC $TMP1 ;SR2 LOCKED UP BY ODD ADDR ERROR
3610 031522 005737 001200 75 TST $TMP1 ;WHERE SRO OR SR2 EFFECTED?
3611 031526 001404 BEQ 85 ;BRANCH IF NO
3612 031530 104044 ERROR 44 ;SRO OR SR2 CHANGED BY ODD ADDR ERROR
3613 ; FOR TIGHTER SCOPE LOOP
3614 ; REPLACE ERROR CALL WITH
3615 ; "BR 25" = 000741
3616 031532 042737 160000 177572 BIC #160000,SRO ;CLEAR ERROR BITS THAT MAY BE SET IN SRO
3617 031540 012737 002332 000004 85 MOV #TIMERR,284 ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
3618 031546 012737 002400 000250 MOV #MGERR,28250 ;RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER
3619 031554 012737 077406 172310 MOV #77406,KIPDR4 ;REMAP PAGE 4 TO READ/WRITE
3620 031562 012737 031374 001110 MOV #15,$LPERR ;RESET LOOP ON ERROR POINTER TO 15
3621
3622 .. *****
3623 .. *TEST 45      PC & PSW SAVED FOR KT ERROR DURING SERVICE OF ODD ADDR  ERROR
3624 .. *
3625 .. *
3626 .. *      THIS TEST CHECKS THE PC AND PROCESSOR STATUS WORD SAVED WHEN
3627 .. *      A KT ERROR OCCURS DURING THE SECOND PUSH ON THE STACK DURING
3628 .. *      SERVICING OF AN ODD ADDR ERROR DURING A "DOUBLE ERROR"
3629 .. *      SEQUENCE SUCH AS THIS, THE PSW SAVED WILL BE THE ONE PICKED UP
    
```

```

3630 .X FROM VECTOR+2 (LOC 6 IN THIS CASE) AFTER THE FIRST TRAP.
3631 .X NOT THE PSW PRESENT BEFORE THE FIRST TRAP SRO AND SR2
3632 .X SHOULD RECORD THE KT ERROR (A R/O VIOLATION BY THE USER STACK PTR )
3633 .X
3634 .X NOTE THAT THE PREVIOUS MODE BITS (13 12) OF THE PSW
3635 .X WILL BE SET IN THE PSW THAT IS SAVED
3636 .X
3637 .X *****
3638 TST45 SCOPE
3639 031570 000004 JSR PC.TOFF ,TURN T-BIT TRAPPING OFF FOR THIS TEST
3640 031572 004737 036002 15 MOV #600.UIPAR3 ,MAP USER PAGE 3 TO 12-16K
3641 031576 012737 000600 177646 MOV #600.UIPAR4 ,MAP USER PAGE 4 TO 12-16K
3642 031604 012737 000600 177650 MOV #77402.UIPDR3 ,MAP USER PAGE 3 READ-ONLY
3643 031612 012737 077402 177606 MOV #77406.UIPDR4 ,MAP USER PAGE 4 READ/WRITE
3644 031620 012737 077406 177610 MOV #45.204 ,LOAD ADDRESS OF 45 IN CPU (ODD ADDR ) VECTOR
3645 031626 012737 031702 000004 MOV #140017.206 ,LOAD PSW THAT SHOULD BE PUT ON STACK IN VECTOR+2
3646 031634 012737 140017 000006 MOV #45.20250 ,LOAD ADDRESS OF 45 IN M.M TRAP VECTOR
3647 031642 012737 031702 000250 MOV #340.20252 ,LOAD A KERNEL PSW IN MAVEC+2
3648 031650 012737 000340 000252 MOV #25.$LPERR ,SET LOOP ON ERROR POINTER TO 25
3649 031656 012737 031664 001110 MOV #140000.PSW ,GO TO USER MODE
3650 031664 012737 140000 177776 25 MOV #100002.USP ,SET USER STACK PTR SO SECOND PUSH IS IN PG 3
3651 031672 012706 100002 MOV 100005 ,CAUSE ODD ADDRESS ERROR THAT WILL CAUSE
3652 031676 005237 100005 35 INC ,R/O ERROR WHEN TRY TO SAVE OLD PC
3653 031702 016601 000002 45 MOV 2(KSP),R1 ,PUT PSW SAVED ON KERNEL STACK INTO R1
3654 031706 011603 MOV (KSP),R3 ,PUT PC SAVED ON KERNEL STACK INTO R3
3655 031710 013737 177572 001366 MOV SRO,WASSRO ,READ THE CONTENTS OF M.M STATUS REG 0
3656 031716 013737 177576 001370 MOV SR2,WASSR2 ,READ THE CONTENTS OF M.M STATUS REG 2
3657 031724 042737 160000 177572 BIC #160000.SRO ,CLEAR THE ERROR BITS IN SRO
3658 031732 005037 177776 CLR PSW ,BE SURE IN KERNEL MODE
3659 031736 012706 001100 MOV #KERSTK.KSP ,RESTORE KERNEL STACK POINTER
3660 031742 012737 140000 177776 MOV #140000.PSW ,GO TO USER MODE
3661 031750 012706 000700 MOV #USESTK.USP ,RESTORE USER STACK POINTER
3662 031754 005037 177776 CLR PSW ,GO BACK TO KERNEL MODE
3663 031760 005037 001176 CLR $TMP0 ,CLEAR ERROR INDICATOR
3664 031764 020127 170017 CMP R1,#170017 ,WAS THE PSW SAVED THE ONE PICKED UP BY THE
3665 , ODD ADDR TRAP FROM ERRVEC+2?
3666 , VALUE 170017 = PSW FROM LOC 6 WITH
3667 , PREVIOUS MODE BITS = USER
3668 031770 001402 BEQ 55 ,BRANCH IF YES
3669 031772 005237 001176 INC $TMP0 ,WRONG PSW SAVED DURING "DOUBLE ERROR" SEQUENCE
    
```

```

3670 031776 020327 031702      55    CMP      R3, #35+4      , WAS THE PC AT THE TIME OF THE ODD ADDR ERROR
3671                                     , SAVED ON THE STACK?
3672 032002 001402                                     BEQ      65          , BRANCH IF YES
3673 032004 005237 001176                                     INC      STMPD      , WRONG PC SAVED DURING TRAP SEQUENCE
3674 032010 023727 001366 020147 65    CMP      WASSRO, #20147 , DID SRO REPORT - USER, PAGE 3, R/O ABORT?
3675 032016 001402                                     BEQ      75          , BRANCH IF YES
3676 032020 005237 001176                                     INC      STMPD      , SRO DID NOT REPORT R/O ABORT
3677 032024 023727 001370 031676 75    CMP      WASSR2, #35   , DID SR2 LOCK UP VIRTUAL ADDR OF LAST
3678                                     , INSTRUCTION SUCCESSFULLY FETCHED?
3679 032032 001402                                     BEQ      85          , BRANCH IF YES
3680 032034 005237 001176                                     INC      STMPD      , SR2 DID NOT LOCK UP ADDR. OF ODD ADDR INST
3681 032040 005737 001176      85    TST      STMPD      , ANY "ERRORS" DURING TRAP SEQUENCE?
3682 032044 001401                                     BEQ      95          , BRANCH IF NO
3683 032046 104045                                     ERROR    45          , THE WRONG PC OR PSW WERE SAVED
3684                                     , OR SRO OR SR2 DID NOT REPORT R/O
3685                                     , ERROR DURING ODD ADDR. - KT TRAP
3686                                     , SEQUENCE
3687                                     , FOR TIGHTER SCOPE LOOP
3688                                     , REPLACE ERROR CALL WITH
3689                                     , "BR 25" = 000710
3690 032050 012737 002332 000004 95    MOV      #TIMERR, @#4 , RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
3691 032056 012737 000340 000006      MOV      #340, @#6   , RELOAD ERRVEC+2 WITH KERNEL PSW
3692 032064 012737 002400 000250      MOV      #MGMERR, @#250 , RESTORE ADDRESS OF NORMAL P.L.M. TRAP HANDLER
3693 032072 012737 077406 177606      MOV      #77406, UIPDR3 , REMAP USER PAGE 3 READ/WRITE
3694 032100 012737 031572 001110      MOV      #15, SLPERR  , RESET LOOP ON ERROR POINTER TO 15
3695 032106 004737 036036      JSR      PC, TON     , TURN T-BIT TRAPPING BACK ON
3696                                     , *****
3697                                     , *
3698                                     , * THIS GROUP OF TESTS WILL TEST ALL THE LOGIC ASSOCIATED WITH
3699                                     , * THE "MOVE FROM PREVIOUS" AND MOVE TO PREVIOUS" INSTRUCTIONS
3700                                     , *
3701                                     , *****
3702
3703
3704                                     , *****
3705 *TEST 46 MOVE FROM PREVIOUS (USER) 1-SPACE
3706                                     , *
3707                                     , * THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT THE
3708                                     , * PREVIOUS MODE IS CLOKED CORRECTLY
3709                                     , * THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.
3710                                     , *
3711                                     , *
3712                                     , * IF THE CORRECT MODE (USER) IS NOT ENABLED A NON-RESIDENT ABORT
3713                                     , * WILL OCCUR AND TRAP TO 235, WHERE THE ERRORS ARE REPORTED
3714                                     , *
3715                                     , *****
3716 TST46: SCOPE
3717 032112 000004      15    CLR      KIPAR0      , MAP KERNEL PAGE 0 TO 0-4K
3718 032114 005037 172340      MOV      #200, KIPAR1 , MAP KERNEL PAGE 1 TO 4-8K
3719 032120 012737 000200 172342      MOV      #400, KIPAR2 , MAP KERNEL PAGE 2 TO 8-12K
3720 032126 012737 000400 172344      MOV      #600, KIPAR3 , MAP KERNEL PAGE 3 TO 12-16K
3721 032142 012737 000600 172350      MOV      #600, KIPAR4 , MAP KERNEL PAGE 4 TO 12-16K
3722 032160 012737 007600 172356      MOV      #7600, KIPAR7 , MAP KERNEL PAGE 7 TO THE I/O PAGE
3723 032166 012700 077406      MOV      #77406, RO   , MAKE ALL KERNEL 1-SPACE PAGES RESIDENT
3724                                     , READ/WRITE, LENGTH 200 BLOCKS
3725 032162 012702 000010      MOV      #10, R2     , SET LOOP COUNTER TO 8
    
```

3726	032166	012701	172300		MOV	#KIPDR0,R1	; PUT ADDRESS OF FIRST PDR IN R1
3727	032172	010021		25	MOV	RO,(R1)+	; LOAD PDR WITH 77406
3728	032174	077202			SOB	R2,25	; LOOP TO 25 UNTIL ALL PDRS LOADED
3729	032176	012702	000010		MOV	#10,R2	; SET LOOP COUNTER TO 8
3730	032202	012701	177600		MOV	#UIPDR0,R1	; PUT ADDRESS OF FIRST PDR IN R1
3731	032206	010021		35	MOV	RO,(R1)+	; LOAD PDR WITH 77406
3732	032210	077202			SOB	R2,35	; LOOP TO 35 UNTIL ALL PDRS LOADED
3733	032212	012737	000000	177640	MOV	#000,UIPAR0	; MAP USER 1 PAGE 0 TO 0-4K
3734	032210	012737	000200	177642	MOV	#200,UIPAR1	; MAP USER 1 PAGE 1 TO 4-8K
3735	032226	012737	000400	177644	MOV	#400,UIPAR2	; MAP USER 1 PAGE 2 TO 8-12K
3736	032234	012737	000600	177646	MOV	#600,UIPAR3	; MAP USER 1 PAGE 3 TO 12-16K
3737	032242	012737	007600	177656	MOV	#7600,UIPAR7	; MAP USER 1 PAGE 7 TO THE 1/6 PAGE
3738	032250	012737	032256	001110	MOV	#45,SLPERR	; SET LOOP ON ERROR TO 45
3739	032256						
3740	032256	012737	077406	172310	MOV	#77406,KIPDR4	; KERNEL I-SPACE PAGE 4 READ/WRITE
3741	032264	012737	000600	172350	MOV	#600,KIPAR4	; MAP KERNEL 1 PAGE 4 TO 12K
3742	032272	012737	000600	177650	MOV	#600,UIPAR4	; MAP USER 1 PAGE 4 TO 12K
3743	032300	012700	036514		MOV	#36514,RO	; LOAD DATA PATTERN INTO RO
3744	032304	010037	100000		MOV	RO,#100000	; LOAD DATA PATTERN INTO PHY 60000
3745	032310	012737	032712	000250	MOV	#235,MVEC	; SET M.M VECTOR TO 235
3746	032316	105037	172310		CLRB	KIPDR4	; MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
3747							; THE FOLLOWING WILL TEST DSTN=0 MFPI
3748							
3749	032322	012737	032330	001110	MOV	#55,SLPERR	; SET LOOP ON ERROR POINTER TO 55
3750	032330	012737	030340	177776	MOV	#030340,PSW	; MAKE PREVIOUS MODE USER
3751	032336	006506		65	MFPI	USP	; PUT USER STACK POINTER ON KERNEL
3752							; STACK
3753	032340	022706	001100		CMP	#KERSTK,KSP	; WAS SOMETHING PUSHED ON STACK AT 65

3754	032344	001407	BEQ	75	, BRANCH IF NOTHING WAS PUSHED
3755	032346	012600	MOV	(KSP)+, R0	, POP KERNEL STACK INTO R0
3756	032350	012701 000700	MOV	#USESTK, R1	, EXPECTING TO GET 700 AS USP

3757	032354	020001				CMP	RO,R1		; DID YOU GET THE RIGHT POINTER?
3758	032356	001403				BEQ	85		; BRANCH IF YOU DID
3759	032360	104046				ERROR	46		; WRONG THING WAS PUSHED ON STACK
3760									; FOR TIGHTER SCOPE LOOP
3761									; REPLACE ERROR CALL WITH
3762									; "BR 55" = 000763
3763	032362	000401				BR	85		; BRANCH TO NEXT TRY
3764	032364	104050			75	ERROR	50		; NOTHING PUSHED ON STACK
3765									; FOR TIGHTER SCOPE LOOP
3766									; REPLACE ERROR CALL WITH
3767									; "BR 55" = 000761
3768	032366				85				; THE FOLLOWING WILL TEST DSTH=1 MFPI.
3769	032366	012737	032400	001110		MOV	#95,SLPERR		; SET LOOP ON ERROR POINTER TO 95
3770	032374	012700	036514			MOV	#36514,R0		; RELOAD DATA PATTERN IN R0
3771	032400	012737	030340	177776	95	MOV	#030340,PSW		; MAKE PREVIOUS MODE USER
3772	032406	012702	100000			MOV	#100000,R2		; LOAD VIRTUAL ADDRESS INTO R2
3773	032412	006512				MFPI	(R2)		; READ FROM PHYSICAL 60000
3774	032414	012601				MOV	(KSP)+,R1		; POP KERNEL STACK INTO R1
3775	032416	020001				CMP	RO,R1		; WAS DATA FETCHED SAME AS STORED
3776	032420	001401				BEQ	105		; BRANCH IF CORRECT DATA WAS FETCHED
3777	032422	104046				ERROR	46		; WRONG DATA WAS FETCHED
3778									; FOR TIGHTER SCOPE LOOP
3779									; REPLACE ERROR CALL WITH
3780									; "BR 95" = 000766
3781	032424				105.				; THE FOLLOWING WILL TEST DSTH=2 MFPI.
3782	032424	012737	032432	001110		MOV	#115,SLPERR		; SET LOOP ON ERROR POINTER TO 115
3783	032432	012737	030340	177776	115.	MOV	#030340,PSW		; MAKE PREVIOUS MODE USER
3784	032440	012702	100000			MOV	#100000,R2		; LOAD VIRTUAL ADDRESS INTO R2
3785	032444	006522				MFPI	(R2)+		; READ FROM PHYSICAL 60000
3786	032446	012601				MOV	(KSP)+,R1		; POP KERNEL STACK INTO R1
3787	032450	020001				CMP	RO,R1		; WAS DATA FETCHED SAME AS STORED
3788	032452	001401				BEQ	125		; BRANCH IF CORRECT DATA WAS FETCHED
3789	032454	104046				ERROR	46		; WRONG DATA WAS FETCHED
3790									; FOR TIGHTER SCOPE LOOP
3791									; REPLACE ERROR CALL WITH
3792									; "BR 115" = 000766
3793	032456				125				; THE FOLLOWING WILL TEST DSTH=3 MFPI


```

3859 032674 012737 002400 000250 22$ MOV #MGMERR,MMVEC ;SET M.M VECTOR TO NORMAL ROUTINE
3860 032702 012737 032114 001110 MOV #15,SLPERR ;SET LOOP POINTER TO START OF TEST
3861 032710 000423 BR TST47 ;BRANCH TO NEXT TEST
3862
3863
3864 032712 012637 001356 23$ MOV (KSP)+,TRAPP ;SAVE PC & PS OF TRAP
3865 032716 012637 001360 MOV (KSP)+,TRAPPS
3866 032722 013737 177572 001366 MOV SRO,WASSRO ;SAVE SRO FOR ERROR TYPEOUT
3867 032730 013737 177576 001370 MOV SR2,WASSR2 ;SAVE SR2 FOR ERROR TYPEOUT
3868 032736 042737 160000 177572 BIC #160000,SRO ;CLEAR ERROR BITS IN SRO AND LEAVE
3869 032744 104051 ERROR 51 ;TRIED TO READ NON-RESIDENT PAGE
3870 ;FOR TIGHTER SCOPE LOOP
3871 ;REPLACE ERROR CALL WITH
3872 ;A "NOP" = 000240
3873 032746 013746 001360 MOV TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
3874 032752 013746 001356 MOV TRAPPC,-(KSP)
3875 032756 000002 RTI
3876
3877
3878 ;*****
3879 ;*TEST 47 MOVE TO PREVIOUS (USER) 1-SPACE
3880 ;*
3881 ;* THIS TEST USES THE 'MTP1' INSTRUCTION TO ENSURE THAT THE
3882 ;* PREVIOUS MODE IS CLOKED CORRECTLY
3883 ;* THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.
3884 ;*
3885 ;*
3886 ;* IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
3887 ;* WILL OCCUR AND TRAP TO 20$, WHERE THE ERRORS ARE REPORTED
3888 ;*
3889 ;*****
3890 032760 000004 TST47. SCOPE
3891 032762 012737 077406 172310 1$ MOV #77406,KIPDR4 ;KERNEL 1-SPACE PAGE 4 READ/WRITE
3892 032770 012737 077406 177610 MOV #77406,UIPDR4 ;USER 1-SPACE PAGE 4 READ/WRITE
3893 032776 012737 000600 172350 MOV #600,KIPAR4 ;MAP KERNEL 1 PAGE 4 TO 12K
3894 033004 012737 000600 177650 MOV #600,UIPAR4 ;MAP USER 1 PAGE 4 TO 12K
3895 033012 012737 033572 000250 MOV #20$,MMVEC ;SET M.M VECTOR TO 20$
3896 ;THE FOLLOWING WILL TEST DSTM=0 MTP1
3897
3898 033020 012737 030340 177776 2$ MOV #030340,PSW ;MAKE PREVIOUS MODE USER
3899 033026 012746 007777 MOV #7777,-(KSP) ;PUSH DATA ON KERNEL STACK
3900 033032 006606 MTP1 USP ;LOAD USER STACK POINTER
3901 033034 006506 MFPI USP ;READ USER STACK POINTER
3902 033036 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
3903 033040 022701 007777 CMP #7777,R1 ;WAS USER STACK POINTER CHANGED
3904 033044 001401 BEQ 3$ ;BRANCH IF IT WAS
3905 033046 104050 ERROR 50 ;USER STACK POINTER NOT CHANGED
3906 ;FOR TIGHTER SCOPE LOOP
3907 ;REPLACE ERROR CALL WITH
3908 ;"BR 2$" = 000764
3909 033060 012737 030340 177776 3$ MOV #030340,PSW ;MAKE PREVIOUS MODE USER
3910 033066 012746 000700 MOV #USESTK,-(KSP) ;GET READY TO RESTORE USER S POINT
3911 033062 006606 MTP1 USP ;RESTORE USER STACK POINTER
3912 033064 4$ ;THIS WILL TEST DSTM = 1 MTP1
3913 033064 012737 033102 001110 MOV #5$,SLPERR ;SET LOOP ON ERROR POINTER TO 5$
3914 033072 012702 100000 MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
    
```

3915	033076	012700	125252		MOV	#125252,RO	,LOAD TEST DATA INTO RO
3916	033102	010046		55	MOV	RO,-(KSP)	,PUSH TEST DATA ON KERNEL STACK
3917	033104	105037	172310		CLRB	KIPDR4	,MAKE KERNEL 1 PAGE 4 NON-RESIDENT
3918	033110	006612			MTP1	(R2)	,LOAD TEST DATA INTO PHYSICAL 60000
3919	033112	112737	000006	172310	MOVB	#006,KIPDR4	,MAKE KERNEL PAGE 4 RESIDENT
3920	033120	011201			MOV	(R2),R1	,READ FROM ADDRESS 60000
3921	033122	020001			CMP	RO,R1	,SEE IF DATA WAS STORED AT CORRECT PLACE
3922	033124	001401			BEQ	65	,BRANCH IF STORE WAS CORRECT
3923	033126	104047			ERROR	47	,INCORRECT STORE
3924							,FOR TIGHTER SCOPE LOOP
3925							,REPLACE ERROR CALL WITH
3926							"BR 55" = 000765
3927	033130			65			,THE FOLLOWING WILL TEST DSTN=2 MTP1.
3928							,
3929	033130	012737	033154	001110	MOV	#85,\$LPERR	,SET LOOP ON ERROR POINTER TO 85
3930	033136	012737	030340	177776	MOV	#030340,PSW	,MAKE PREVIOUS MODE USER
3931	033144	012700	125252		MOV	#125252,RO	,LOAD TEST DATA INTO RO
3932	033150	012702	100000		MOV	#100000,R2	,LOAD VIRTUAL ADDRESS INTO R2
3933	033154	010046		85	MOV	RO,-(KSP)	,PUSH TEST DATA ON KERNEL STACK
3934	033156	105037	172310		CLRB	KIPDR4	,MAKE KERNEL PAGE 4 NON-RESIDENT
3935	033162	006612			MTP1	(R2)	,LOAD TEST DATA INTO PHYSICAL 60000
3936	033164	112737	000006	172310	MOVB	#006,KIPDR4	,MAKE KERNEL PAGE 4 RESIDENT
3937	033172	013701	100000		MOV	#100000,R1	,READ FROM ADDRESS 60000
3938	033176	020001			CMP	RO,R1	,SEE IF DATA WAS STORED CORRECTLY
3939	033200	001401			BEQ	95	,BRANCH IF STORE WAS CORRECT
3940	033202	104047			ERROR	47	,INCORRECT STORE
3941							,FOR TIGHTER SCOPE LOOP
3942							,REPLACE ERROR CALL WITH
3943							"BR 85" = 000764
3944	033204			95			,THIS WILL TEST DSTN = 3 MTP1.
3945	033204	012737	033224	001110	MOV	#105,\$LPERR	,SET LOOP ON ERROR POINTER TO 105
3946	033212	012737	030340	177776	MOV	#030340,PSW	,MAKE PREVIOUS MODE USER
3947	033220	012700	052525		MOV	#52525,RO	,LOAD TEST DATA INTO RO
3948	033224	010046		105	MOV	RO,-(KSP)	,PUSH TEST DATA ON KERNEL STACK
3949	033226	105037	172310		CLRB	KIPDR4	,MAKE KERNEL 1 PAGE 4 NON-RESIDENT
3950	033232	006637	100000		MTP1	#100000	,LOAD TEST DATA INTO PHYSICAL 60000
3951	033236	112737	000006	172310	MOVB	#006,KIPDR4	,MAKE KERNEL PAGE 4 RESIDENT
3952	033244	013701	100000		MOV	#100000,R1	,READ FROM ADDRESS 60000
3953	033250	020001			CMP	RO,R1	,SEE IF DATA WAS STORED CORRECTLY

3954	033252	001401				BEG	115		.BRANCH IF STORE WAS CORRECT
3955	033254	104047				ERROR	47		.INCORRECT STORE
3956									.FOR TIGHTER SCOPE LOOP
3957									.REPLACE ERROR CALL WITH
3958									. "BR 105" = 000763
3959	033256				115			.THIS WILL TEST DSTN = 4	MTP1
3960	033256	012737	033276	001110		MOV	#125, \$LPERR		.SET LOOP ON ERROR POINTER TO 125
3961	033264	012737	030340	177776		MOV	#030340, PSW		.MAKE PREVIOUS MODE USER
3962	033272	012700	125252			MOV	#125252, R0		.LOAD TEST DATA INTO R0
3963	033276	010046			125	MOV	R0, -(KSP)		.PUSH TEST DATA ON KERNEL STACK
3964	033300	012702	100002			MOV	#100002, R2		.LOAD VIRTUAL ADDRESS INTO R2

3965	033304	105037	172310		CLRB	KIPDR4	, MAKE KERNEL 1 PAGE 4 NON-RESIDENT
3966	033310	006642			MTP1	-(R2)	, LOAD TEST DATA INTO PHYSICAL 60000
3967	033312	112737	000006	172310	MOVB	#006, KIPDR4	, MAKE KERNEL PAGE 4 RESIDENT
3968	033320	013701	100000		MOV	#100000, R1	, READ FROM ADDRESS 60000
3969	033324	020001			CMP	RO, R1	, SEE IF DATA WAS STORED CORRECTLY
3970	033326	001401			BEQ	135	, BRANCH IF STORE WAS CORRECT
3971	033330	104047			ERROR	47	, INCORRECT STORE
3972							, FOR TIGHTER SCOPE LOOP
3973							, REPLACE ERROR CALL WITH
3974							, "BR 125" = 000762
3975	033332			135			, THE FOLLOWING WILL TEST DSTN=5 MTP1
3976							,
3977	033332	012737	033364	001110	MOV	#145, \$LPERR	, SET LOOP ON ERROR POINTER TO 145
3978	033340	012737	030340	177776	MOV	#030340, PSW	, MAKE PREVIOUS MODE USER
3979	033346	012700	052525		MOV	#52525, RO	, LOAD TEST DATA INTO RO
3980	033352	012702	001204		MOV	#(STMP2+2), R2	, LOAD ADDR. OF LOC. STMP2+2 INTO R2
3981	033356	012737	100000	001202	MOV	#100000, STMP2	, LOAD VIRT. ADDR. OF TEST LOC INTO STMP2
3982	033364	010046		145	MOV	RO, -(KSP)	, PUSH TEST DATA ON KERNEL STACK
3983	033366	105037	172310		CLRB	KIPDR4	, MAKE KERNEL PAGE 4 NON-RESIDENT
3984	033372	006652			MTP1	-(R2)	, LOAD TEST DATA INTO PHYSICAL 60000
3985	033374	112737	000006	172310	MOVB	#006, KIPDR4	, MAKE KERNEL PAGE 4 RESIDENT
3986	033402	013701	100000		MOV	#100000, R1	, READ FROM ADDRESS 60000
3987	033406	020001			CMP	RO, R1	, SEE IF DATA WAS STORED CORRECTLY
3988	033410	001401			BEQ	155	, BRANCH IF STORE WAS CORRECT
3989	033412	104047			ERROR	47	, INCORRECT STORE
3990							, FOR TIGHTER SCOPE LOOP
3991							, REPLACE ERROR CALL WITH
3992							, "BR 145" = 000764
3993	033414			155			, THIS WILL TEST DSTN = 6 MTP1
3994							,
3995	033414	012737	033436	001110	MOV	#165, \$LPERR	, SET LOOP ON ERROR POINTER TO 165
3996	033422	012737	030340	177776	MOV	#030340, PSW	, MAKE PREVIOUS MODE USER
3997	033430	012700	052525		MOV	#52525, RO	, LOAD TEST DATA INTO RO
3998	033434	005002			CLR	R2	, MAKE REGISTER 2 ZERO
3999	033436	010046		165	MOV	RO, -(KSP)	, PUSH TEST DATA ON KERNEL STACK
4000	033440	105037	172310		CLRB	KIPDR4	, MAKE KERNEL 1 PAGE 4 NON-RESIDENT
4001	033444	006662	100000		MTP1	100000(R2)	, LOAD TEST DATA INTO PHYSICAL 60000
4002	033450	112737	000006	172310	MOVB	#006, KIPDR4	, MAKE KERNEL PAGE 4 RESIDENT
4003	033456	013701	100000		MOV	#100000, R1	, READ FROM ADDRESS 60000
4004	033462	020001			CMP	RO, R1	, SEE IF DATA WAS STORED CORRECTLY
4005	033464	001401			BEQ	175	, BRANCH IF STORE WAS CORRECT
4006	033466	104047			ERROR	47	, INCORRECT STORE
4007							, FOR TIGHTER SCOPE LOOP
4008							, REPLACE ERROR CALL WITH
4009							, "BR 165" = 000763
4010	033470			175			, THE FOLLOWING WILL TEST DSTN=7 MTP1
4011							,
4012	033470	012737	033522	001110	MOV	#185, \$LPERR	, SET LOOP ON ERROR POINTER TO 185
4013	033476	012737	030340	177776	MOV	#030340, PSW	, MAKE PREVIOUS MODE USER
4014	033504	012700	125252		MOV	#125252, RO	, LOAD TEST DATA INTO RO
4015	033510	012737	100000	001202	MOV	#100000, STMP2	, LOAD VIRT. ADDR. OF TEST LOCATION
4016							, INTO LOCATION STMP2
4017	033516	012702	001202		MOV	#STMP2, P2	, LOAD ADDRESS OF STMP2 INTO R2
4018	033522	010046		185	MOV	RO, -(KSP)	, PUSH TEST DATA ON KERNEL STACK
4019	033524	105037	172310		CLRB	KIPDR4	, MAKE KERNEL PAGE 4 NON-RESIDENT
4020	033530	006672	000000		MTP1	20(R2)	, LOAD TEST DATA INTO PHYSICAL 60000

```

4021 033534 112737 000006 172310      MOV      #006,KIPDR4      ,MAKE KERNEL PAGE 4 RESIDENT
4022 033542 013701 100000              MOV      #100000,R1      ,READ FROM ADDRESS 60000
4023 033546 020001              CMP      R0,R1          ,SEE IF DATA WAS STORED CORRECTLY
4024 033550 001401              BEQ      195            ,BRANCH IF STORE WAS CORRECT
4025 033552 104047              ERROR    47            ,INCORRECT STORE
4026                                ,FOR TIGHTER SCOPE LOOP
4027                                ,REPLACE ERROR CALL WITH
4028                                , "BR 185" = 000763
4029 033554 012737 032762 001110 195      MOV      #15,$LPERR      ,SET LOOP POINTER TO START OF TEST
4030 033562 012737 002400 000250      MOV      #MGMERR,MMVEC   ,RESTORE M.M VECTOR TO NORMAL ROUTINE
4031 033570 000423              BR       TST50          ,BRANCH TO NEXT TEST
4032
4033
4034 033572 012637 001356      205     MOV      (KSP)+,TRAPPC    ,SAVE PC & PS OF TRAP
4035 033576 012637 001360      MOV      (KSP)+,TRAPPS
4036 033602 013737 177572 001366      MOV      SRO,WASSRO     ,SAVE SRO FOR ERROR TYPEOUT
4037 033610 013737 177576 001370      MOV      SR2,WASSR2     ,SAVE SR2 FOR ERROR TYPEOUT
4038 033616 042737 160000 177572      BIC      #160000,SRO    ,CLEAR ERROR BITS IN SRO
4039 033624 104051              ERROR    51            ,TRIED TO LOAD A N R PAGE 4
4040                                ,FOR TIGHTER SCOPE LOOP
4041                                ,REPLACE ERROR CALL WITH
4042                                , A "NOP" = 000240
4043 033626 013746 001360      MOV      TRAPPS,-(KSP)   ,PUT PC & PS OF TRAP ON STACK
4044 033632 013746 001356      MOV      TRAPPC -(KSP)
4045 033636 000002              RTI                    ,RETURN TO TEST
4046
4047
4048
4049
4050                                , , *****
4051                                , *TEST 50      MOVE FROM PREVIOUS (KERNEL) I-SPACE TO USER MODE
4052                                , *
4053                                , *      THIS TEST CHECKS THAT IF THE PREVIOUS MODE IS KERNEL THE
4054                                , *      FETCH IS FROM KERNEL SPACE
4055                                , *      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.
4056                                , *
4057                                , *      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
4058                                , *      WILL OCCUR AND TRAP TO 215, WHERE THE ERRORS ARE REPORTED
4059                                , *
4060                                , , *****
4061 033640 000004      TST50   SCOPE
4062 033642 012700 077406      15      MOV      #77406,R0      ,MAKE ALL USER I-SPACE PAGES RESIDENT
4063                                ,READ/WRITE, LENGTH 200 BLOCKS
4064 033646 012702 000010      MOV      #10,R2        ,SET LOOP COUNTER TO 8
4065 033662 012701 177600      MOV      #UIPDRO,R1    ,LOAD ADDRESS OF FIRST PDR IN R1
4066 033666 010021      25      MOV      R0,(R1)+      ,LOAD PDR WITH 77406
4067 033660 077202      SOB      R2,25         ,LOOP UNTIL 8 USER PDRS LOADED
4068 033662 012737 033670 001110      MOV      #35,$LPERR    ,SET LOOP ON ERROR TO 35
4069 033670 012737 140340 177776      35      MOV      #140340,PSW    ,GO TO USER MODE FOR THIS TEST
4070 033676 012737 077406 172310      MOV      #77406,KIPDR4 ,KERNEL I-SPACE PAGE 4 READ/WRITE
4071 033704 012737 000600 172350      MOV      #600,KIPDR4   ,MAP KERNEL 1 PAGE 4 TO 12K
4072 033712 012737 000600 177650      MOV      #600,UIPAR4   ,MAP USER 1 PAGE 4 TO 12K
4073 033720 012700 036514      MOV      #36514,R0     ,LOAD DATA PATTERN INTO R0
4074 033724 010037 100000      MOV      R0,#100000    ,LOAD DATA PATTERN INTO PHY 60000
4075 033730 012702 100000      MOV      #100000,R2    ,LOAD VIRTUAL ADDRESS INTO R2
4076                                ,THE FOLLOWING WILL TEST DSTM=0 MFPI
    
```


4133	034124				125	, THE FOLLOWING WILL TEST DSTN=4 MFPI
4134	034124	012737	034132	001110		MOV #135, \$LPERR ; SET LOOP ON ERROR POINTER TO 135
4135	034132	012737	140340	177776	135	MOV #140340, PSW ; MAKE PREVIOUS MODE KERNEL PRESENT USER
4136	034140	012702	100002			MOV #100002, R2 ; LOAD VIRTUAL ADDRESS INTO R2
4137	034144	006542				MFPI -(R2) ; READ FROM PHYSICAL 60000
4138	034146	012601				MOV (USP)+, R1 ; POP USER STACK INTO R1
4139	034150	020001				CMP R0, R1 ; WAS DATA FETCHED SAME AS STORED
4140	034152	001401				BEQ 145 ; BRANCH IF CORRECT DATA WAS FETCHED
4141	034154	104046				ERROR 46 ; WRONG DATA WAS FETCHED
4142						; FOR TIGHTER SCOPE LOOP
4143						; REPLACE ERROR CALL WITH
4144						"BR 135" = 000766
4145	034156				145	, THE FOLLOWING WILL TEST DSTN=5 MFPI.
4146						;
4147	034156	012737	034164	001110		MOV #155, \$LPERR ; SET LOOP ON ERROR POINTER TO 155
4148	034164	012737	140340	177776	155	MOV #140340, PSW ; MAKE PREVIOUS MODE KERNEL PRESENT USER
4149	034172	012737	100000	001202		MOV #100000, \$TMP2 ; LOAD TEST LOC. VIRT. ADDR INTO LOC \$TMP2
4150	034200	012702	001204			MOV #(\$TMP2+2), R2 ; LOAD ADDRESS OF \$TMP2+2 INTO R2
4151	034204	006552				MFPI @-(R2) ; READ FROM PHYSICAL 60000
4152	034206	012601				MOV (USP)+, R1 ; POP USER STACK INTO R1
4153	034210	020001				CMP R0, R1 ; WAS DATA FETCHED SAME AS STORED
4154	034212	001401				BEQ 165 ; BRANCH IF CORRECT DATA FETCHED
4155	034214	104046				ERROR 46 ; WRONG DATA WAS FETCHED
4156						; FOR TIGHTER SCOPE LOOP
4157						; REPLACE ERROR CALL WITH
4158						"BR 155" = 000763
4159	034216				165	, THE FOLLOWING WILL TEST DSTN=6 MFPI.
4160						;
4161	034216	012737	034224	001110		MOV #175, \$LPERR ; SET LOOP ON ERROR POINTER TO 175
4162	034224	012737	140340	177776	175	MOV #140340, PSW ; MAKE PREVIOUS MODE KERNEL PRESENT USER
4163	034232	005002				CLR R2 ; MAKE REGISTER 2 A ZERO
4164	034234	006562	100000			MFPI 100000(R2) ; READ FROM PHYSICAL 60000
4165	034240	012601				MOV (USP)+, R1 ; POP USER STACK INTO R1
4166	034242	020001				CMP R0, R1 ; WAS DATA FETCHED SAME AS STORED
4167	034244	001401				BEQ 185 ; BRANCH IF CORRECT DATA FETCHED
4168	034246	104046				ERROR 46 ; WRONG DATA WAS FETCHED
4169						; FOR TIGHTER SCOPE LOOP
4170						; REPLACE ERROR CALL WITH
4171						"BR 175" = 000766
4172	034250				185	, THE FOLLOWING WILL TEST DSTN=7 MFPI
4173						;
4174	034250	012737	034256	001110		MOV #195, \$LPERR ; SET LOOP ON ERROR POINTER TO 195
4175	034256	012737	140340	177776	195	MOV #140340, PSW ; MAKE PREVIOUS MODE KERNEL PRESENT USER
4176	034264	012737	100000	001202		MOV #100000, \$TMP2 ; LOAD TEST LOC. VIRT. ADDR INTO \$TMP2
4177	034272	012702	001202			MOV #(\$TMP2), R2 ; LOAD ADDRESS OF \$TMP2 INTO R2
4178	034276	006572	000000			MFPI @0(R2) ; READ FROM PHYSICAL 60000
4179	034302	012601				MOV (USP)+, R1 ; POP USER STACK INTO R1
4180	034304	020001				CMP R0, R1 ; WAS DATA FETCHED SAME AS STORED
4181	034306	001401				BEQ 205 ; BRANCH IF CORRECT DATA FETCHED
4182	034310	104046				ERROR 46 ; WRONG DATA WAS FETCHED
4183						; FOR TIGHTER SCOPE LOOP
4184						; REPLACE ERROR CALL WITH
4185						"BR 195" = 000762
4186	034312	012737	002400	000250	205	MOV #MGMERR, MMVEC ; SET M M VECTOR TO NORMAL ROUTINE
4187	034320	012737	000340	177776		MOV #00340, PSW ; GO BACK TO KERNEL MODE, PREVIOUS KERNEL
4188	034326	012737	033642	001110		MOV #15, \$LPERR ; SET LOOP POINTER TO START OF TEST

```

4189 034334 000423          BR      TST51          ,, BRANCH TO NEXT TEXT
4190
4191
4192 034336 012637 001356    21$    MOV      (KSP)+, TRAPPC  , SAVE PC & PS OF TRAP
4193 034342 012637 001360    MOV      (KSP)+, TRAPPS
4194 034346 013737 177572    001366  MOV      SRO, WASSRO    , SAVE SRO FOR ERROR TYPEOUT
4195 034354 013737 177576    001370  MOV      SR2, WASSR2    , SAVE SR2 FOR ERROR TYPEOUT
4196 034362 042737 160000    177572  BIC      #160000, SRO    , CLEAR ERROR BITS IN SRO
4197 034370 104051          ERROR    51            , TRIED TO READ NON-RESIDENT PAGE
4198                                , FOR TIGHTER SCOPE LOOP
4199                                , REPLACE ERROR CALL WITH
4200                                , A "NOP" = 000240
4201 034372 013746 001360          MOV      TRAPPS, -(KSP)  , PUT PC & PS OF TRAP ON STACK
4202 034376 013746 001356    MOV      TRAPPC, -(KSP)
4203 034402 000002          RTI            , RETURN TO TEST
4204
4205                                ,, *****
4206                                , *TEST 51          MOVE FROM/TO D-SPACE = MOVE FROM/TO 1-SPACE
4207                                , *
4208                                , * THIS TEST CHECKS THAT SINCE THERE IS NO DISTINCTION
4209                                , * BETWEEN INSTRUCTION AND DATA SPACE IN THE 11/34
4210                                , * MFPD & MTPD SHOULD BE DECODED THE SAME AS MFPI & MTP1
4211                                , *
4212                                ,, *****
4213 034404 000004          TST51: SCOPE
4214 034406 012737 030340    177776  1$    MOV      #030340, PSW    , MAKE PREVIOUS MODE=USER, CURRENT=KERNEL
4215 034414 106506          MFPD     USP            , MFPD SHOULD ACT LIKE MFPI PUTTING
4216                                , USER STACK POINTER ON THE KERNEL STACK
4217 034416 022706 001100          CMP      #KERSTK, KSP    , WAS SOMETHING PUSHED ON KERNEL STACK?
4218 034422 001407          BEQ      2$            , BRANCH IF NO
4219 034424 012600          MOV      (KSP)+, R0      , POP KERNEL STACK INTO R0
4220 034426 012701 000700          MOV      #USESTK, R1    , EXPECTING TO GET 700 AS USP
4221 034432 020001          CMP      R0, R1         , DID GET RIGHT POINTER VALUE?
4222 034434 001403          BEQ      3$            , BRANCH IF YES
4223 034436 104053          ERROR    53            , WRONG THING WAS PUSHED ON STACK
4224                                , FOR TIGHTER SCOPE LOOP
4225                                , REPLACE ERROR CALL WITH
4226                                , "BR 1$" = 000763
4227 034440 000401          BR      3$            , BRANCH TO NEXT TRY
4228 034442 104054          2$    ERROR    54            , NOTHING PUSHED ON STACK
4229                                , FOR TIGHTER SCOPE LOOP
4230                                , REPLACE ERROR CALL WITH
4231                                , "BR 1$" = 000761
4232 034444 012737 034452    001110  3$    MOV      #4$, $LPERR    , SET LOOP ON ERROR POINTER TO 4$
4233 034452 012746 007777    4$    MOV      #7777, -(KSP)  , PUSH DATA ON KERNEL STACK
4234 034456 106606          MTPD     USP            , LOAD THE USER STACK POINTER
4235 034460 106506          MFPD     USP            , READ USER STACK POINTER
4236 034462 012601          MOV      (KSP)+, R1    , POP KERNEL STACK INTO R1
4237 034464 022701 007777          CMP      #7777, R1     , WAS USER STACK POINTER CHANGED?
4238 034470 001401          BEQ      5$            , BRANCH IF YES
4239 034472 104054          ERROR    54            , USER STACK POINTER NOT CHANGED
4240                                , FOR TIGHTER SCOPE LOOP
4241                                , REPLACE ERROR CALL WITH
4242                                , "BR 4$" = 000767
4243 034474 012746 000700    5$    MOV      #USESTK, -(KSP) , GET READY TO RESTORE USER STK PTR
4244 034500 106606          MTPD     USP            , RESTORE USER STACK POINTER

```



```

4245 034502 012737 034406 001110      MOV      #15,SLPERR      ,SET LOOP POINTER TO START OF TEST
4246
4247 //*****
4248 ,*TEST 52      MOVE FROM PREVIOUS I=SPACE (PREVIOUS=CURRENT=KERNEL)
4249 ,*
4250 ,*      THIS TEST CHECKS THAT IF BOTH PREVIOUS AND CURRENT MODES
4251 ,*      ARE KERNEL, AND THE SOURCE MODE IS 0, THE DESTINATION
4252 ,*      STACK IS NOT DECREMENTED BEFORE ACCESS.
4253 ,*      "MFPI KSP" SHOULD PUSH THE NON-DECREMENTED VALUE
4254 ,*      OF KSP (1100) ONTO THE STACK (AT LOC. 1076).
4255 //*****
4256 034510 000004      TST52   SCOPE
4257 034512 005037 177776 15      CLR      @#PSW          ;SET PREVIOUS = CURRENT = KERNEL
4258 034516 012700 001100      MOV      #STACK,RO     ;SETUP VALUE FOR STACK POINTER
4259 034522 010006      MOV      RO,KSP        ;LOAD STACK POINTER
4260 034524 006506      MFPI     KSP           ;THE VALUE "STACK" SHOULD BE PUSHED
4261                                     ;BEFORE BEING DECREMENTED
4262 034526 011601      MOV      (KSP),R1      ;READ DATA WHICH WAS PUSHED
4263 034530 020001      CMP      RO,R1        ;WAS THE ORIGINAL VALUE OF THE
4264                                     ;STACK POINTER PUSHED?
4265 034532 001401      BEQ      25            ;BRANCH IF YES
4266 034534 104046      ERROR   46            ;MFPI FETCHED WRONG DATA
4267                                     ;FOR TIGHTER SCOPE LOOP
4268                                     ;REPLACE ERROR CALL WITH
4269                                     ;"BR 15" = 000766
4270 034536 005740      25      TST      -(RO)   ;SETUP EXPECTED STACK POINTER VALUE
4271 034540 020600      CMP      KSP,RO       ;WAS THE STACK POINTER DECREMENTED?
4272 034542 001401      BEQ      35            ;BRANCH IF YES
4273 034544 104050      ERROR   50            ;STACK NOT PUSHED BY THE MFPI
4274                                     ;FOR TIGHTER SCOPE LOOP
4275                                     ;REPLACE ERROR CALL WITH
4276                                     ;"BR 15" = 000762
4277 034546 012706 001100      35      MOV      #STACK,KSP ;RESTORE STACK POINTER
4278
4279
4280
4281
4282
4283      SBTTL *****
4284
    
```

```

4285 .SBTTL END OF PASS ROUTINE
4286
4287 ;, *****
4288 ; INCREMENT THE PASS NUMBER ($PASS)
4289 ; TYPE "END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYY"
4290 ; WHERE XXXXX AND YYYY ARE DECIMAL NUMBERS
4291 ; IF SW12=1 INHIBIT TRACE TRAP
4292 ; IF THERES A MONITOR GO TO IT
4293 ; IF THERE ISN'T JUMP TO LOOP
4294
4295 SEOP
4296 034552 SCOPE
4297 034552 000004 CLR $STNM ; ZERO THE TEST NUMBER
4298 034560 005037 001102 CLR $TIMES ; ZERO THE NUMBER OF ITERATIONS
4299 034564 005237 001212 INC $PASS ; INCREMENT THE PASS NUMBER
4300 034570 042737 100000 001234 BIC #100000,$PASS ; DON'T ALLOW A NEG NUMBER
4301 034576 005327 DEC (PC)+ ; LOOP?
4302 034600 000001 SEOPCT. WORD 1
4303 034602 003072 BGT $DOAGN ; YES
4304 034604 012737 MOV (PC)+,@(PC)+ ; RESTORE COUNTER
4305 034606 000001 SENDCT. WORD 1
4306 034610 034600 SEOPCT
4307 034612 104401 034620 TYPE ,655 ; TYPE ASCIZ STRING
4308 034616 000407 BR 645 ; GET OVER THE ASCIZ
4309 ;,655 ASCIZ <12><15>/END PASS #/
4310 645
4311 034636 013746 001234 MOV $PASS,-(SP) ; SAVE $PASS FOR TYPEOUT
4312 ; TYPE PASS NUMBER
4313 034642 104405 TYPDS ; GO TYPE--DECIMAL ASCII WITH SIGN
4314 034644 104401 034652 TYPE ,675 ; TYPE ASCIZ STRING
4315 034650 000421 BR 665 ; GET OVER THE ASCIZ
4316 ;,675. ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
4317 665
4318 034714 013746 011112 MOV $ERTTL,-(SP) ; SAVE $ERTTL FOR TYPEOUT
4319 ; TOTAL NUMBER OF ERRORS
4320 034720 104405 TYPDS ; GO TYPE--DECIMAL ASCII WITH SIGN
4321 034722 104401 001223 TYPE ,SCLRF ; TYPE CARRIAGE RETURN, LINE FEED
4322 034726 005037 001112 CLR $ERTTL ; CLEAR ERROR TOTAL
4323 034732 013700 000042 $GET42. MOV @#42,R0 ; GET MONITOR ADDRESS
4324 034736 001414 BEQ $DOAGN ; BRANCH IF NO MONITOR
4325 034740 005046 CLR -(SP) ; INSURE THE "T" BIT IS CLEAR
4326 034742 012746 034750 MOV #SCLR.T,-(SP) ; SETUP FOR AN RTI OR RTT
4327 034746 000426 BR $RTN ; GO DO AN RTI OR RTT TO LOAD THE PSW
4328 ; WITH A CLEARED "T" BIT
4329
4330 034750 013700 000042 SCLR T. MOV @#42,R0 ; INSURE R0 CONTAINS THE MONITORS
4331 034754 001405 BEQ $DOAGN ; RETURN ADDRESS
4332 034756 000005 RESET ; CLEAR THE WORLD
4333 034760 004710 ENDAD: JSR PC,(R0) ; GO TO MONITOR
4334 034762 000240 NOP ; SAVE ROOM
4335 034764 000240 NOP ; FOR
4336 034766 000240 NOP ; ACT11
4337 034770
4338 034770 104400 SDOAGN: TRAP ; PUSH OLD PSW AND PC ON STACK
4339 034772 042716 000020 BIC #20,(SP) ; CLEAR THE "T" BIT
4340 034776 032777 010000 144134 BIT #BIT12,@SWR ; RUN WITH TRACE TRAP?
    
```

```

4341 035004 001005          BNE 15          ;; BR IF NO
4342 035006 005137 035032  COM $TBIT        ;; IS IT TIME FOR TRACE TRAP
4343 035012 100402          BMI 15          ;; BR IF NO
4344 035014 052716 000020  BIS #20,(SP)    ;; SET TRACE TRAP
4345 035020 012746 035026  15  MOV $SLOOP,-(SP) ;; JUMP TO START OF TEST
4346 035024 000002          $RTRN RTI        ;; RETURN--THIS IS CHANGED TO
4347                                     ;; AN "RTT" IF "RTT" IS A LEGAL
4348                                     ;; INSTRUCTION
4349 035026          $SLOOP.
4350 035026 000137          JMP @PC+        ;; RETURN
4351 035030 020466          $RTNAD .WORD LOOP
4352 035032 000000          $TBIT .WORD 0    ;; "T" BIT STATE INDICATOR
4353 035034 377 377 000 $ENULL: .BYTE -1,-1,0 ;; NULL CHARACTER STRING
4354                                     .EVEN
4355 .SBTTL SCOPE HANDLER ROUTINE
4356
4357 ;; *****
4358 ;; THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
4359 ;; AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG. (DISPLAY<7 0>)
4360 ;; AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15: 08>
4361 ;; THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4362 ;; $SW14=1 LOOP ON TEST
4363 ;; $SW11=1 INHIBIT ITERATIONS
4364 ;; $SW09=1 LOOP ON ERROR
4365 ;; $SW08=1 LOOP ON TEST IN SWR<7 0>
4366 ;; $CALL
4367 ;; SCOPE          ;; SCOPE=10T
4368
4369 $SCOPE:
4370 035040 104410          CKSWR
4371 035042 032777 040000 144070 15: BIT #BIT14,$SWR ;; TEST FOR CHANGE IN SOFT-SWR
4372 035050 001114          BNE $OVER        ;; LOOP ON PRESENT TEST?
4373                                     ;; YES IF SW14=1
4374 035052 000416          ;####$START OF CODE FOR THE XOR TESTER####
4375          $XTSTR: BR 65 ;; IF RUNNING ON THE "XOR" TESTER CHANGE
4376          MOV @ERRVEC,-(SP) ;; THIS INSTRUCTION TO A "NOP" (NOP=240)
4377          MOV #5,$ERRVEC ;; SAVE THE CONTENTS OF THE ERROR VECTOR
4378          TST @177060 ;; SET FOR TIMEOUT
4379          MOV (SP)+,$ERRVEC ;; TIME OUT ON XOR?
4380          BR $SVLAD ;; RESTORE THE ERROR VECTOR
4381          CMP (SP)+,(SP)+ ;; GO TO THE NEXT TEST
4382          MOV (SP)+,$ERRVEC ;; CLEAR THE STACK AFTER A TIME OUT
4383          BR 75 ;; RESTORE THE ERROR VECTOR
4384          BR 75 ;; LOOP ON THE PRESENT TEST
4385          65: ;####$END OF CODE FOR THE XOR TESTER####
4386          BIT #BIT08,$SWR ;; LOOP ON SPEC. TEST?
4387          BEQ 25 ;; BR IF NO
4388          CMPB $SWR,$TSTNM ;; ON THE RIGHT TEST? SWR<7 0>
4389          BEQ $OVER ;; BR IF YES
4390          TSTB $ERFLG ;; HAS AN ERROR OCCURRED?
4391          BEQ 35 ;; BR IF NO
4392          CMPB $ERMAX,$ERFLG ;; MAX. ERRORS FOR THIS TEST OCCURRED?
4393          BMI 35 ;; BR IF NO
4394          BIT #BIT09,$SWR ;; LOOP ON ERROR?
4395          BEQ 45 ;; BR IF NO
4396          MOV $LPERR,$LPADR ;; SET LOOP ADDRESS TO LAST SCOPE
4397          BR $OVER
    
```

```

4397 035166 105037 001103      45  CLR      SERFLG      .. ZERO THE ERROR FLAG
4398 035172 005037 001212      CLR      STIMES      .. CLEAR THE NUMBER OF ITERATIONS TO MAKE
4399 035176 000415      BR       15          .. ESCAPE TO THE NEXT TEST
4400 035200 032777 004000 143732 35  BIT      @BIT11, @SWR .. INHIBIT ITERATIONS?
4401 035206 001011      BNE     15          .. BR IF YES
4402 035210 005737 001234      TST     SPASS      .. IF FIRST PASS OF PROGRAM
4403 035214 001406      BEQ     15          .. INHIBIT ITERATIONS
4404 035216 005237 001104      INC     SICNT      .. INCREMENT ITERATION COUNT
4405 035222 023737 001212 001104      CMP     STIMES, SICNT .. CHECK THE NUMBER OF ITERATIONS MADE
4406 035230 002024      BGE     SOVER      .. BR IF MORE ITERATION REQUIRED
4407 035232 012737 000001 001104 15  MOV     #1, SICNT  .. REINITIALIZE THE ITERATION COUNTER
4408 035240 013737 035316 001212      MOV     SMXCNT, STIMES .. SET NUMBER OF ITERATIONS TO DO
4409 035246 105237 001102      $SVLAD INCB     STSTNM      .. COUNT TEST NUMBERS
4410 035252 113737 001102 001232      MOV     STSTNM, STSTNM .. SET TEST NUMBER IN APT MAILBOX
4411 035260 011637 001106      MOV     (SP), SLPADR .. SAVE SCOPE LOOP ADDRESS
4412 035264 011637 001110      MOV     (SP), SLPERR .. SAVE ERROR LOOP ADDRESS
4413 035270 005037 001214      CLR     $ESCAPE    .. CLEAR THE ESCAPE FROM ERROR ADDRESS
4414 035274 112737 000001 001115      MOV     #1, SERMAX .. ONLY ALLOW ONE(1) ERROR ON NEXT TEST
4415 035302 013777 001102 143632 $OVER  MOV     STSTNM, @DISPLAY .. DISPLAY TEST NUMBER
4416 035310 013716 001106      MOV     RTI, SLPADR, (SP) .. FUDGE RETURN ADDRESS
4417 035314 000002      RTI     .. FIXES PS
4418 035316 000200      SMXCNT 200        .. MAX NUMBER OF ITERATIONS
4419      SBTTL  ERROR HANDLER ROUTINE
4420
4421      .. *****
4422      .. THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT.
4423      .. SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
4424      .. AND GO TO ERRTP ON ERROR
4425      .. THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE
4426      .. $SW15=1      HALT ON ERROR
4427      .. $SW13=1      INHIBIT ERROR TYPEOUTS
4428      .. $SW10=1      BELL ON ERROR
4429      .. $SW09=1      LOOP ON ERROR
4430      .. $CALL
4431      .. $      ERROR      N      .. ERROR=EMT AND N=ERROR ITEM NUMBER
4432
4433      $ERROR
4434      C$SWR      .. TEST FOR CHANGE IN SOFT-SWR
4435      MOV     R0, $REG0 .. SAVE THE CONTENTS OF R0
4436      MOV     R1, $REG1 .. SAVE THE CONTENTS OF R1
4437      MOV     R2, $REG2 .. SAVE THE CONTENTS OF R2
4438      MOV     R3, $REG3 .. SAVE THE CONTENTS OF R3
4439      MOV     R4, $REG4 .. SAVE THE CONTENTS OF R4
4440      MOV     R5, $REG5 .. SAVE THE CONTENTS OF R5
4441      MOV     STSTNM, TESTNO .. SAVE THE TEST NUMBER
4442      75      INCB     SERFLG      .. SET THE ERROR FLAG
4443      BEQ     75      .. DON'T LET THE FLAG GO TO ZERO
4444      MOV     STSTNM, @DISPLAY .. DISPLAY TEST NUMBER AND ERROR FLAG
4445      BIT     @BIT10, @SWR .. BELL ON ERROR?
4446      BEQ     15      .. NO - SKIP
4447      TYPE   , $BELL .. RING BELL
4448      15      INC     $ERTTL .. COUNT THE NUMBER OF ERRORS
4449      MOV     (SP), $ERRPC .. GET ADDRESS OF ERROR INSTRUCTION
4450      SUB     #2, $ERRPC
4451      MOV     @ERRPC, $ITEMB .. STRIP AND SAVE THE ERROR ITEM CODE
4452      BIT     @BIT13, @SWR .. SKIP TYPEOUT IF SET
    
```

```

4453 035442 001004 BNE 20$ ..SKIP TYPEOUTS
4454 035444 004737 035556 JSR PC,ERRTYP ..GO TO USER ERROR ROUTINE
4455 035450 104401 001223 TYPE ,SCLRF
4456 035454 20$
4457 035454 122737 000001 001246 CMPB #APTENV,SENV ..RUNNING IN APT MODE
4458 035462 001007 BNE 2$ ..NO,SKIP APT ERROR REPORT
4459 035464 113737 001114 035476 MOVB $ITEMB,21$ ..SET ITEM NUMBER AS ERROR NUMBER
4460 035472 004737 040110 JSR PC,$ATYH ..REPORT FATAL ERROR TO APT
4461 035476 000 21$ BYTE 0
4462 035477 000 BYTE 0
4463 035500 000777 22$ BR 22$ ..APT ERROR LOOP
4464 035502 005777 143432 2$ TST $SWR ..HALT ON ERROR
4465 035506 100002 BPL 3$ ..SKIP IF CONTINUE
4466 035510 000000 HALT ..HALT ON ERROR!
4467 035512 104410 CKSWR ..TEST FOR CHANGE IN SOFT-SWR
4468 035514 032777 001000 143416 3$ BIT #BIT09,$SWR ..LOOP ON ERROR SWITCH SET?
4469 035522 001402 BEQ 4$ ..BR IF NO
4470 035524 013716 001110 MOV $LPERR,(SP) ..FUDGE RETURN FOR LOOPING
4471 035530 005737 001214 4$ TST $ESCAPE ..CHECK FOR AN ESCAPE ADDRESS
4472 035534 001402 BEQ 5$ ..BR IF NONE
4473 035536 013716 001214 MOV $ESCAPE,(SP) ..FUDGE RETURN ADDRESS FOR ESCAPE
4474 035542 5$
4475 035542 022737 034760 000042 CMP #SENDAD,$#42 ..ACT-11 AUTO-ACCEPT?
4476 035550 001001 BNE 6$ ..BRANCH IF NO
4477 035552 000000 HALT ..YES
4478 035554 6$
4479 035554 000002 RTI ..RETURN

```

SBTTL ERROR MESSAGE TIMEOUT ROUTINE

..*****
 ;THIS ROUTINE USES THE "ITEM CONTROL BYTE" (\$ITEMB) TO DETERMINE WHICH
 ;ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" (\$ERRTB),
 ;AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR
 ;
 ;NOTES:
 ;1) THIS ROUTINE PROVIDES AN AUTOMATIC "CARRIAGE RETURN-LINE FEED"
 ; FOR "EM", "DM", AND "DT".
 ;2) TWO SPACES ARE TYPED AFTER EACH NUMBER FOR "DT"
 ;3) FOR \$ITEMB=0, JUST THE ERROR PC IS TYPED
 ;4) THE AVAILABLE FORMATS FOR TYPING DATA ARE
 ; DF FORMAT
 ; 0 TYPE A 6 DIGIT OCTAL NUMBER (FROM 16-BIT BINARY)
 ; 1 TYPE A DECIMAL NUMBER WITHOUT LEADING ZEROS
 ; 2 TYPE A 16 DIGIT BINARY NUMBER
 ; 3 TYPE A 6 DIGIT OCTAL NUMBER (FROM 18-BIT BINARY)
 ;

```

4500 035556 ERRTYP
4501 035556 104401 001223 TYPE ,SCLRF .."CARRIAGE RETURN" & "LINE FEED"
4502 035562 010046 MOV RO,-(SP) ..SAVE RO
4503 035564 005000 CLR RO ..PICKUP THE ITEM INDEX
4504 035566 153700 001114 BISB $ITEMB,RO
4505 035572 001004 BNE 1$ ..IF ITEM NUMBER IS ZERO JUST
4506 035574 013746 001116 MOV $ERRPC,-(SP) ..SAVE $ERRPC FOR TYPEOUT
4508 ..ERROR ADDRESS

```

```

4509 035600 104402          TYP0C          , GO TYPE--OCTAL ASCII(ALL DIGITS)
4510 035602 000471          BR           13$      , GET OUT
4511 035604 005300          1$ DEC       RO      , ADJUST THE INDEX SO THAT IT WILL
4512 035606 006300          ASL       RO      , WORK FOR THE ERROR TABLE
4513 035610 006300          ASL       RO
4514 035612 006300          ASL       RO
4515 035614 062700 001412  ADD       #ERRTB,RO  , FORM TABLE POINTER
4516 035620 012037 035630  MOV       (RO)+,2$  , PICKUP "ERROR MESSAGE" POINTER
4517 035624 001404          BEQ       3$      , SKIP TIMEOUT IF NO POINTER
4518 035626 104401          TYPE     , TYPE THE "ERROR MESSAGE"
4519 035630 000000          2$ WORD    0      , "ERROR MESSAGE" POINTER GOES HERE
4520 035632 104401 001223  , SCRLF      , "CARRIAGE RETURN" & "LINE FEED"
4521 035636 012037 035646  3$ MOV       (RO)+,4$  , PICKUP "DATA HEADER" POINTER
4522 035642 001404          BEQ       5$      , SKIP TIMEOUT IF 0
4523 035644 104401          TYPE     , TYPE THE "DATA HEADER"
4524 035646 000000          4$ WORD    0      , "DATA HEADER" POINTER GOES HERE
4525 035650 104401 001223  , SCRLF      , "CARRIAGE RETURN" & "LINE FEED"
4526 035654 010146          5$ MOV       R1,-(SP)  , SAVE R1
4527 035656 012001          MOV       (RO)+,R1  , PICKUP "DATA TABLE" POINTER
4528 035660 001441          BEQ       12$     , BR IF NO DATA TO BE TYPED
4529 035662 012000          MOV       (RO)+,RO  , PICKUP "DATA FORMAT" POINTER
4530 035664 105710          6$ TSTB      (RO)     , IS IT FORMAT 0?
4531 035666 001003          BNC       7$      , BR IF NO
4532
4533          , *THIS CODE IS FOR OCTAL (16-BIT) FORMAT (DF=0)
4534 035670 013146          MOV       @ (R1)+,-(SP) , SAVE @ (R1)+ FOR TIMEOUT
4535 035672 104402          TYP0C          , GO TYPE--OCTAL ASCII(ALL DIGITS)
4536 035674 000425          BR           11$
4537
4538          , *THIS CODE IS FOR DECIMAL FORMAT (DF=1)
4539 035676 121027 000001  7$ CMPB      (RO),#1   , IS IT FORMAT 1?
4540 035702 001003          BNE       8$      , BRANCH IF NO
4541 035704 013146          MOV       @ (R1)+,-(SP) , SAVE @ (R1)+ FOR TIMEOUT
4542 035706 104405          TYPDS      , GO TYPE--DECIMAL ASCII WITH SIGN
4543 035710 000417          BR           11$
4544
4545          , *THIS CODE IS FOR BINARY FORMAT (DF=2)
4546 035712 121027 000002  8$ CMPB      (RO),#2   , IS IT FORMAT 2?
4547 035716 001003          BNE       9$      , BRANCH IF NO
4548 035720 013146          MOV       @ (R1)+,-(SP) , SAVE @ (R1)+ FOR TIMEOUT
4549 035722 104406          TYPBN      , GO TYPE--BINARY ASCII
4550 035724 000411          BR           11$
4551
4552          , *THIS CODE IS FOR OCTAL (18-BIT) FORMAT (DF=3)
4553 035726 012146          9$ MOV       (R1)+,-(SP) , PUT ADDRESS OF FIRST LOC ON STACK
4554 035730 004737 041162  JSR       PC,$DB20  , CONVERT TWO LOCS TO AN ASCII STRING
4555 035734 062716 000005  ADD       #5,(SP)   , ONLY NEED 6 CHARACTERS NOT 11
4556 035740 012637 035746  MOV       (SP)+,10$  , PUT ADDRESS OF ASCII CHARS AT 10$
4557 035744 104401          TYPE     , TYPE OCTAL VALUE OF 18-BIT BINARY NO
4558 035746 000000          10$ WORD    0
4559
4560 035750 005711          11$ TST       (R1)     , IS THERE ANOTHER NUMBER?
4561 035752 001404          BEQ       12$     , BR IF NO
4562 035754 104401 035776  TYPE     , TYPE TWO(2) SPACES
4563 035760 105720          TSTB      (RO)+   , POINT TO NEW "DATA FORMAT"
4564 035762 000740          BR           6$      , LOOP
    
```

4565							
4566	035764	012601		125	MOV	(SP)+, R1	, RESTORE R1
4567	035766	012600		135	MOV	(SP)+, R0	, RESTORE R0
4568	035770	104401	001223		TYPE	, \$CRLF	, "CARRIAGE RETURN" & "LINE FEED"
4569	035774	000207			RTS	PC	, RETURN
4570	035776	020040	000	145	ASCIZ	/ /	, TWO(2) SPACES
4571		036002			EVEN		
4572							

```

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



```

4629 036124 012702 000010          MOV      #10,R2          ,LOAD LOOP COUNTER WITH AN 8
4630 036130 012701 177600          MOV      #UIPDR0,R1     ,LOAD ADDRESS OF FIRST PDR INTO R1
4631 036134 012721 177777          MOV      #-1,(R1)+      ,SET BITS IN A USER PDR TO 1
35 4632 036140 077203          SOB      R2,35          ,LOOP TO 35 UNTIL ALL USER PDR'S LOADED
4633 036142 012702 000010          MOV      #10,R2          ,LOAD LOOP COUNTER WITH AN 8
4634 036146 012701 177640          MOV      #UIPAR0,R1     ,LOAD ADDRESS OF FIRST PAR INTO R1
4635 036152 012721 177777          MOV      #-1,(R1)+      ,SET BITS IN A USER PAR TO 1
45 4636 036156 077203          SOB      R2,45          ,LOOP TO 45 UNTIL ALL USER PAR'S LOADED
4637 036160 000207          RTS              ,RETURN TO TEST
4638
4639          SBTTL  READ & COMPARE KERNEL & USER PAR/PDR'S
4640          ,*****
4641          ,*
4642          ,*      THIS SUBROUTINE IS USED BY PAR/PDR DUAL ADDRESSING TEST TO
4643          ,*      READ ALL THE PAR'S AND PDR'S TO SEE THAT ONLY ONE REGISTER
4644          ,*      WAS CLEARED IN RESPONSE TO A SINGLE PAR OR PDR ADDRESS
4645          ,*      ANY FAILURES FOUND BY THE PAR/PDR DUAL ADDRESSING TEST WILL
4646          ,*      BE REPORTED BY THIS SUBROUTINE.
4647          ,*
4648          ,*****
4649          CMPREG
4650 036162 012701 172300          MOV      #KIPDR0,R1     ,LOAD ADDRESS OF FIRST KERNEL PDR IN R1
4651 036166 012704 000010          MOV      #10,R4          ,LOAD LOOP COUNTER WITH AN 8
4652 036172 012705 077416          MOV      #77416,R5      ,PUT EXPECTED PDR CONTENTS IN R5
4653 036176 011102          MOV      (R1),R2        ,READ A KERNEL PDR INTO R2
15 4654 036200 020205          CMP      R2,R5          ,ARE ALL WRITABLE BITS SET AS EXPECTED?
4655 036202 001403          BEQ      25             ,BRANCH IF YES
4656 036204 020100          CMP      R1,R0          ,WAS IT THE REG. THAT WAS CLEARED?
4657 036206 001401          BEQ      25             ,BRANCH IF YES
4658 036210 104016          ERROR    16            ,A PDR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
4659          ,FOR TIGHTER SCOPE LOOP
4660          ,REPLACE ERROR CALL WITH
4661          ,AN "RTS PC" = 000207
4662 036212 062701 J00002          25 ADD      #2,R1          ,FORM NEXT KERNEL PDR ADDRESS
4663 036216 077411          SOB      R4,15          ,LOOP TO 15 UNTIL ALL KERNEL PDR'S CHECKED
4664 036220 012701 172340          MOV      #KIPAR0,R1     ,LOAD ADDRESS OF FIRST KERNEL PAR IN R1
4665 036224 012704 000010          MOV      #10,R4          ,LOAD LOOP COUNTER WITH AN 8
4666 036230 012705 007777          MOV      #7777,R5      ,PUT EXPECTED PAR CONTENTS IN R5
4667 036234 011102          MOV      (R1),R2        ,READ A KERNEL PAR INTO R2
35 4668 036236 020205          CMP      R2,R5          ,ARE ALL WRITABLE BITS SET AS EXPECTED?
4669 036240 001403          BEQ      45             ,BRANCH IF YES
4670 036242 020100          CMP      R1,R0          ,WAS IT THE REG THAT WAS CLEARED?
4671 036244 001401          BEQ      45             ,BRANCH IF YES
4672 036246 104016          ERROR    16            ,A PAR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
4673          ,FOR TIGHTER SCOPE LOOP
4674          ,REPLACE ERROR CALL WITH
4675          ,AN "RTS PC" = 000207
4676 036250 062701 000002          45 ADD      #2,R1          ,FORM NEXT KERNEL PAR ADDRESS
4677 036254 077411          SOB      R4,35          ,LOOP TO 35 UNTIL ALL KERNEL PAR'S CHECKED
4678 036256 012701 177600          MOV      #UIPDR0,R1     ,LOAD ADDRESS OF FIRST USER PDR IN R1
4679 036262 012704 000010          MOV      #10,R4          ,LOAD LOOP COUNTER WITH AN 8
4680 036266 012705 077416          MOV      #77416,R5      ,PUT EXPECTED PDR CONTENTS IN R5
4681 036272 011102          MOV      (R1),R2        ,READ A USER PDR INTO R2
55 4682 036274 020205          CMP      R2,R5          ,ARE ALL WRITABLE BITS SET AS EXPECTED?
4683 036276 001403          BEQ      65             ,BRANCH IF YES
4684 036300 020100          CMP      R1,R0          ,WAS IT THE REG THAT WAS CLEARED?
    
```

```

4685 036302 001401          BEQ      65          ,BRANCH IF YES
4686 036304 104016          ERROR    16          ,A PDR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
4687                                     ,FOR TIGHTER SCOPE LOOP
4688                                     ,REPLACE ERROR CALL WITH
4689                                     ,AN "RTS PC" = 000207
4690 036306 062701 000002    65      ADD      #2,R1      ,FORM NEXT USER PDR ADDRESS
4691 036312 077411          SOB      R4,55         ,LOOP TO 55 UNTIL ALL USER PDR'S CHECKED
4692 036314 012701 177640    MOV      #UIPAR0,R1    ,LOAD ADDRESS OF FIRST USER PAR IN R1
4693 036320 012704 000010    MOV      #10,R4        ,LOAD LOOP COUNTER WITH AN 8
4694 036324 012705 007777    MOV      #777,R5       ,PUT EXPECTED PAR CONTENTS IN R5
4695 036330 011102          MOV      (R1),R2       ,READ A USER PAR INTO R2
4696 036332 020205          CMP      R2,R5         ,ARE ALL WRITABLE BITS SET AS EXPECTED?
4697 036334 001403          BEQ      85           ,BRANCH IF YES
4698 036336 020100          CMP      R1,R0         ,WAS IT THE REG. THAT WAS CLEARED?
4699 036340 001401          BEQ      85           ,BRANCH IF YES
4700 036342 104016          ERROR    16          ,A PAR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
4701                                     ,FOR TIGHTER SCOPE LOOP
4702                                     ,REPLACE ERROR CALL WITH
4703                                     ,AN "RTS PC" = 000207
4704 036344 062701 000002    85      ADD      #2,R1      ,FORM NEXT USER PAR ADDRESS
4705 036350 077411          SOB      R4,75         ,LOOP TO 75 UNTIL ALL USER PAR'S CHECKED
4706 036352 000207          RTS      PC            ,RETURN TO TEST
4707
4708          SBTTL  CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS
4709          ,*****
4710          ,X
4711          ,X      THIS SUBROUTINE IS USED TO FORM AN 18-BIT PHYSICAL ADDRESS
4712          ,X      (PBA) FROM THE 16-BIT VIRTUAL ADDRESS (VBA) AND THE APPROPRIATE
4713          ,X      PAGE ADDRESS REGISTER (PAR).  THE SAME METHOD USED BY THE MEMORY
4714          ,X      MANAGEMENT LOGIC IS USED.  VBA <15:13> SELECTS WHICH PAR/PDR
4715          ,X      IS TO BE USED, VBA <5:0>+PBA <5:0>, AND VBA <12:6> IS ADDED
4716          ,X      TO PAR <11:00> TO GIVE PBA <17:6>  BITS <17:16> OF THE
4717          ,X      PHYSICAL ADDRESS ARE LEFT IN LOC "PBAH1" AND BITS <15:00>
4718          ,X      ARE LEFT IN LOC. "PBA0"  THE PSW'S "CURRENT MODE" BITS
4719          ,X      ARE USED TO SELECT THE KERNEL OR USER PAR/PDR'S  THE ROUTINE
4720          ,X      IS ENTERED WITH LOC "VIRT1" CONTAINING THE 16-BIT VIRTUAL
4721          ,X      ADDRESS
4722          ,X
4723          ,*****
4724
4725 036354 012702 172340    FORMPA  MOV      #KIPARC,R2      ,LOAD ADDRESS OF FIRST KERNEL PAR IN R2
4726 036360 032737 140000 177776  BIT      #140000,PSW     ,IN USER MODE?
4727 036366 001402          BEQ      15           ,BRANCH IF NO
4728 036370 012702 177640    MOV      #UIPAR0,R2    ,LOAD ADDRESS OF FIRST USER PAR IN R2
4729 036374 013700 001402    15      MOV      VIRT1,R0       ,LOAD VIRTUAL ADDR (VBA) INTO R0
4730 036400 072027 177764    ASH      #-14,R0        ,GET BITS <15:13> DOWN TO BITS <3:1>
4731 036404 042700 177761    BIC      #177761,R0     ,MASK OF ALL BITS BUT BITS <3:1>
4732 036410 060002          ADD      R0,R2         ,ADD OFFSET TO BASE PAR ADDRESS
4733 036412 011200          MOV      (R2),R0       ,GET BITS <11:00> FROM APPROPRIATE PAR
4734 036414 010002          MOV      R0,R2         ,COPY PAR BITS <11:00> INTO R2
4735 036416 013737 001402 001406  MOV      VIRT1,PBA0     ,PUT VIRTUAL ADDR IN LOC "PBA0"
4736 036424 042737 160000 001406  BIC      #160000,PBA0   ,CLEAR OFF BITS <15:13> OF ORIGINAL VBA
4737 036432 072227 177766    ASH      #-12,R2       ,GET PAR <11:00> DOWN TO BITS <1:0> OF R2
4738 036436 042702 177774    BIC      #177774,R2     ,CLEAR OFF ALL BITS BUT BITS <1:0>
4739 036442 072027 000006    ASH      #6,R0         ,SHIFT PAR<9:0> TO <15:6> OF R0
4740 036446 042700 000077    BIC      #77,R0        ,CLEAR BITS <5:0> OF R0
    
```

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

```

4747          SBTTL  TTY INPUT ROUTINE
4748
4749          .. *****
4750          ENABL  LSB
4751
4752          .. *****
4753          *SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
4754          *ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
4755          *SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
4756          *WHEN OPERATING IN TTY FLAG MODE
4757 036466 022737 000176 001140 SCKSWR  CMP      #SWREG, SWR      .. IS THE SOFT-SWR SELECTED?
4758 036474 001114          BNE      155          .. BRANCH IF NO
4759 036476 105777 142442          TSTB   @STKS          .. CHAR THERE?
4760 036502 100111          BPL     155          .. IF NO, DON'T WAIT AROUND
4761 036504 117746 142436          MOVB   @STKB, -(SP)   .. SAVE THE CHAR
4762 036510 042716 177600          BIC    # C177, (SP)  .. STRIP-OFF THE ASCII
4763 036514 022726 000007          CMP    #7, (SP)+     .. IS IT A CONTROL G?
4764 036520 001102          BNE    155          .. NO, RETURN TO USER
4765 036522 123727 001134 000001          CMPB   $AUTOB, #1    .. ARE WE RUNNING IN AUTO-MODE?
4766 036530 001476          BEQ    155          .. BRANCH IF YES
4767
4768 036532 104401 037430          SGTSWR TYPE   , $CNTLG      .. ECHO THE CONTROL-G ( G)
4769 036536 104401 037435          TYPE   , $MSWR        .. TYPE CURRENT CONTENTS
4770 036542 013746 000176          MOV    SWREG, -(SP)   .. SAVE SWREG FOR TYPEOUT
4771 036546 104402          TYPOC          .. GO TYPE--OCTAL ASCII(ALL DIGITS)
4772 036550 104401 037446          TYPE   , $MNEW        .. PROMPT FOR NEW SWR
4773 036554 005046          19$    CLR    -(SP)     .. CLEAR COUNTER
4774 036556 005046          CLR    -(SP)         .. THE NEW SWR
4775 036560 105777 142360          7$    TSTB   @STKS          .. CHAR THERE?
4776 036564 100375          BPL    7$           .. IF NOT TRY AGAIN
4777
4778 036566 117746 142354          MOVB   @STKB, -(SP)   .. PICK UP CHAR
4779 036572 042716 177600          BIC    # C177, (SP)  .. MAKE IT 7-BIT ASCII
4780
4781 036576 021627 000003          CMP    (SP), #3       .. IS IT A CONTROL-C?
4782 036602 001015          BNE    9$           .. BRANCH IF NOT
4783 036604 104401 037416          TYPE   , $CNTLC       .. YES, ECHO CONTROL-C ( C)
4784 036610 062706 000006          ADD    #6, SP         .. CLEAN UP STACK
4785 036614 123727 001135 000001          CMPB   $INTAG, #1     .. REENABLE TTY KEYBOARD INTERRUPTS?
4786 036622 001003          BNE    8$           .. BRANCH IF NO
4787 036624 012777 000100 142312          MOV    #100, @STKS   .. ALLOW TTY KEYBOARD INTERRUPTS
4788 036632 000137 037460          8$    JMP    CNTRLC     .. CONTROL-C RESTART
4789
4790
4791 036636 021627 000025          9$    CMP    (SP), #25   .. IS IT A CONTROL-U?
4792 036642 001005          BNE    10$          .. BRANCH IF NOT
4793 036644 104401 037423          TYPE   , $CNTLU       .. YES, ECHO CONTROL-U ( U)
4794 036650 062706 000006          20$   ADD    #6, SP         .. IGNORE PREVIOUS INPUT
4795 036654 000737          BR     19$          .. LET'S TRY IT AGAIN
4796
4797
4798 036656 021617 000015          10$   CMP    (SP), #15     .. IS IT A <CR>?
4799 036662 001012          BNE    16$          .. BRANCH IF NO
4800 036664 005766 000004          TST    4(SP)         .. YES, IS IT THE FIRST CHAR?
4801 036670 001403          BEQ    11$          .. BRANCH IF YES
4802 036672 016677 000002 142240          MOV    2(SP), @SWR   .. SAVE NEW SWR
    
```

```

4803 036700 062706 000006      11$   ADD     #6,SP           ;; CLEAR UP STACK
4804 036704 104401 001223      14$   TYPE   ,SCLRF        ;; ECHO <CR> AND <LF>
4805 036710 123727 001135 000001      CMPB   $INTAG,#1     ;; RE-ENABLE TTY KBD INTERRUPTS?
4806 036716 001003                BNE    15$           ;; BRANCH IF NOT
4807 036720 012777 000100 142216      MOV    #100,2$TKS   ;; RE-ENABLE TTY KBD INTERRUPTS
4808 036726 000002                RTI                    ;; RETURN
4809 036730 004737 040022      15$   JSR    PC,$TYPEC    ;; ECHO CHAR
4810 036734 021627 000060      16$   CMP    (SP),#60     ;; CHAR < 0?
4811 036740 002420                BLT    18$           ;; BRANCH IF YES
4812 036742 021627 000067      CMP    (SP),#67     ;; CHAR > 7?
4813 036746 003015                BGT    18$           ;; BRANCH IF YES
4814 036750 042726 000060      BIC    #60,(SP)+    ;; STRIP-OFF ASCII
4815 036754 005766 000002      TST    2(SP)        ;; IS THIS THE FIRST CHAR
4816 036760 001403                BEQ    17$           ;; BRANCH IF YES
4817 036762 006316                ASL    (SP)         ;; NO, SHIFT PRESENT
4818 036764 006316                ASL    (SP)         ;; CHAR OVER TO MAKE
4819 036766 006316                ASL    (SP)         ;; ROOM FOR NEW ONE
4820 036770 005266 000002      17$   INC    2(SP)        ;; KEEP COUNT OF CHAR
4821 036774 056616 177776      BIS    -2(SP), (SP) ;; SET IN NEW CHAR
4822 037000 000667                BR     7$           ;; GET THE NEXT ONE
4823 037002 104401 001222      18$   TYPE   ,SQUES      ;; TYPE ?<CR><LF>
4824 037006 000720                BR     20$         ;; SIMULATE CONTROL-U
4825                DSABL  LSB
4826
4827
4828                ;; *****
4829                *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
4830                *CALL
4831                *      RDCHR                ;; INPUT A SINGLE CHARACTER FROM THE TTY
4832                *      RETURN HERE        ;; CHARACTER IS ON THE STACK
4833                *                          ;; WITH PARITY BIT STRIPPED OFF
4834                *
4835
4836 037010 011646                $RDCHR MOV    (SP),-(SP) ;; PUSH DOWN THE PC
4837 037012 016666 000004 000002      MOV    4(SP),2(SP) ;; SAVE THE PS
4838 037020 106777 142120      1$   TSTB  2$TKS        ;; WAIT FOR
4839 037024 100375                BPL    1$           ;; A CHARACTER
4840 037026 117746 142114 000004      MOVB  2$TKB,4(SP)  ;; READ THE TTY
4841 037034 042766 177600 000004      BIC    # C(177),4(SP) ;; GET RID OF JUNK IF ANY
4842 037042 026627 000004 000023      CMP    4(SP),#23   ;; IS IT A CONTROL-S?
4843 037050 001013                BNE    3$           ;; BRANCH IF NO
4844 037052 106777 142066      2$   TSTB  2$TKS        ;; WAIT FOR A CHARACTER
4845 037056 100375                BPL    2$           ;; LOOP UNTIL ITS THERE
4846 037060 117746 142062      MOVB  2$TKB,-(SP)  ;; GET CHARACTER
4847 037064 042716 177600      BIC    # C(177),(SP) ;; MAKE IT 7-BIT ASCII
4848 037070 022627 000021      CMP    (SP)+,#21   ;; IS IT A CONTROL-Q?
4849 037074 001366                ANE    2$           ;; IF NOT DISCARD IT
4850 037076 000750                BR     1$           ;; YES, RESUME
4851 037100 026627 000004 000140      3$   CMP    4(SP),#140  ;; IS IT UPPER CASE?
4852 037106 002407                BLT    4$           ;; BRANCH IF YES
4853 037110 026627 000004 000175      CMP    4(SP),#175  ;; IS IT A SPECIAL CHAR?
4854 037116 003003                BGT    4$           ;; BRANCH IF YES
4855 037120 042766 000040 000004      BIC    #40,4(SP)   ;; MAKE IT UPPER CASE
4856 037126 000002                RTI                    ;; GO BACK TO USER
4857                ;; *****
4858                *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
    
```

```

4859          .SCALL
4860          .S      RDLIN
4861          .S      RETURN HERE
4862          .S
4863
4864 037130 010346 $RDLIN MOV R3, -(SP)
4865 037132 005046 CLR -(SP)
4866 037134 012703 037406 15 MOV #STTYIN, R3
4867 037140 022703 037416 25 CMP #STTYIN+8, R3
4868 037144 101467 BLOS 45
4869 037146 104411 RDCHR
4870 037150 112613 MOVB (SP)+, (R3)
4871 037152 122713 000003 CMPB #3, (R3)
4872 037156 001006 BNE 105
4873 037160 104401 037416 TYPE ,SCNTLC
4874 037164 005726 TST (SP)+
4875 037166 012603 MOV (SP)+, R3
4876 037170 000137 037460 JMP CNTRLC
4877 037174 122713 000177 105 CMPB #177, (R3)
4878 037200 001022 BNE 55
4879 037202 005716 TST (SP)
4880 037204 001007 BNE 65
4881 037206 112737 000134 037404 MOVB #' ,95
4882 037214 104401 037404 TYPE ,95
4883 037220 012716 177777 MOV #-1, (SP)
4884 037224 005303 65 DEC R3
4885 037226 020327 037406 CMP R3, #STTYIN
4886 037232 103434 BLO 45
4887 037234 111337 037404 MOVB (R3), 95
4888 037240 104401 037404 TYPE ,95
4889 037244 000735 BR 25
4890 037246 005716 55 TST (SP)
4891 037250 001406 BEQ 75
4892 037252 112737 000134 037404 MOVB #' ,95
4893 037260 104401 037404 TYPE ,95
4894 037264 005016 CLR (SP)
4895 037266 122713 000025 75 CMPB #25, (R3)
4896 037272 001003 BNE 85
4897 037274 104401 037423 TYPE ,SCNTLU
4898 037300 000715 BR 15
4899 037302 122713 000022 85 CMPB #22, (R3)
4900 037306 001011 BNE 35
4901 037310 105013 CLRB (R3)
4902 037312 104401 001223 TYPE ,SCRLF
4903 037316 104401 037406 TYPE ,STTYIN
4904 037322 000706 BR 25
4905 037324 104401 001222 45 TYPE ,SQUES
4906 037330 000701 BR 15
4907 037332 111337 037404 35 MOVB (R3), 95
4908 037336 104401 037404 TYPE ,95
4909 037342 122723 000015 CMPB #15, (R3)+
4910 037346 001274 BNE 25
4911 037350 105063 177777 CLRB -1(R3)
4912 037354 104401 001224 TYPE ,SLF
4913 037360 005726 TST (SP)+
4914 037362 012603 MOV (SP)+, R3
    
```

```

.. INPUT A STRING FROM THE TTY
.. ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
.. TERMINATOR WILL BE A BYTE OF ALL 0'S

.. SAVE R3
.. CLEAR THE RUBOUT KEY
.. GET ADDRESS
.. BUFFER FULL?
.. BR IF YES
.. GO READ ONE CHARACTER FROM THE TTY
.. GET CHARACTER
.. IS IT A CONTROL-C?
.. BRANCH IF NO
.. TYPE A CONTROL-C ( C )
.. CLEAN RUBOUT KEY OFF OF THE STACK
.. RESTORE R3
.. GOTO CONTROL-C RESTART
.. IS IT A RUBOUT
.. BR IF NO
.. IS THIS THE FIRST RUBOUT?
.. BR IF NO
.. TYPE A BACK SLASH

.. SET THE RUBOUT KEY
.. BACKUP BY ONE
.. STACK EMPTY?
.. BR IF YES
.. SETUP TO TYPEOUT THE DELETED CHAR
.. GO TYPE
.. GO READ ANOTHER CHAR
.. RUBOUT KEY SET?
.. BR IF NO
.. TYPE A BACK SLASH

.. CLEAR THE RUBOUT KEY
.. IS CHARACTER A CTRL U?
.. BR IF NO
.. TYPE A CONTROL "U"
.. GO START OVER
.. IS CHARACTER A " R"?
.. BRANCH IF NO
.. CLEAR THE CHARACTER
.. TYPE A "CR" & "LF"
.. TYPE THE INPUT STRING
.. GO PICKUP ANOTHER CHACTER
.. TYPE A '?'
.. CLEAR THE BUFFER AND LOOP
.. ECHO THE CHARACTER

.. CHECK FOR RETURN
.. LOOP IF NOT RETURN
.. CLEAR RETURN (THE 15)
.. TYPE A LINE FEED
.. CLEAN RUBOUT KEY FROM THE STACK
.. RESTORE R3
    
```

```

4915 037364 011646          MOV      (SP), -(SP)      ;; ADJUST THE STACK AND PUT ADDRESS OF THE
4916 037366 016666 000004 000002      MOV      4(SP), 2(SP)    ;; FIRST ASCII CHARACTER ON IT
4917 037374 012766 037406 000004      MOV      @STTYIN, 4(SP)
4918 037402 000002          R11                      ;; RETURN
4919 037404 000          95      BYTE 0      ;; STORAGE FOR ASCII CHAR TO TYPE
4920 037405 000          BYTE 0      ;; TERMINATOR
4921 037406 000010          STTYIN. BLKB 8      ;; RESERVE 8 BYTES FOR TTY INPUT
4922 037416 041536 005015 000      $CNTLC. ASCIZ / C/<15><12>  ;; CONTROL "C"
4923 037423 136 006525 000012      $CNTLU. ASCIZ / U/<15><12>  ;; CONTROL "U"
4924 037430 043536 005015 000      $CNTLG. ASCIZ / G/<15><12>  ;; CONTROL "G"
4925 037435 015 051412 051127      $MSWR.  ASCIZ <15><12>/SWR = /
4926 037442 036440 000040          SMNEW  ASCIZ / NEW = /
4927 037446 020040 042516 020127
4928 037454 020075 000          EVEN
4929 037460
4930
4931          SBTTL CONTROL-C SERVICING ROUTINE
4932
4933          * THE FOLLOWING CODE IS EXECUTED WHEN A CONTROL-C HAS
4934          * BEEN TYPED INSTEAD OF A NEW SWITCH REG VALUE
  
```

```

4935 .X (IN OTHER WORDS, AFTER A CONTROL-G WAS TYPED)
4936 .X A NEW SWITCH REG VALUE WILL BE ASKED FOR.
4937 .X THE TEST NUMBER AND PASS NUMBER WILL BE TYPED,
4938 .X AND THEN THE PROGRAM WILL GO TO "END-OF-PASS" AND CONTINUE
4939
4940 037460 013737 001234 001210 CNTRLC: MOV SPASS, STMP5 ; GET THE VALUE OF "SPASS"
4941 037466 005237 001210 INC STMP5 ; FORM CURRENT PASS NO.
4942 037472 104401 037537 TYPE ,CMMSG ; TYPE THE TEST STOPS MESSAGE
4943 037476 113737 001102 037532 MOVB $STMP, 15 ; SAVE THE TEST NUMBER
4944 037504 013746 037532 MOV 15, -(SP) ; SAVE 15 FOR TYPEOUT
4945 037510 104402 TYPOC ; GO TYPE--OCTAL ASCII (ALL DIGITS)
4946 037512 104401 037534 TYPE ,25 ; TYPE 2 SPACES
4947 037516 013746 001210 MOV STMP5, -(SP) ; SAVE STMP5 FOR TYPEOUT
4948 037522 104405 TYPDS ; GO TYPE--DECIMAL ASCII WITH SIGN
4949 037524 104407 GTSR ; ASK FOR NEW SMR VALUE
4950 037526 000137 034554 JMP $EOP+2 ; CONTINUE AT $EOP+2
4951 037532 000000 15. WORD 0 ; BUFFER FOR TEST NUMBER
4952 037534 020040 000 25. ASCIZ / / ; TWO SPACES AND THE STOP MESSAGE
4953 037537 112 046525 044520 MSG: ASCII / JUMPING TO END-OF-PASS/<15><12>
4954 037544 043516 052040 020117
4955 037562 047106 026504 043117
4956 037560 050055 051501 006523
4957 037566 012
4958 037567 124 051505 047124 .ASCIZ /TESTNO PASSNO/<15><12>
4959 037574 020117 050040 051501
4960 037602 047123 006517 000012
    
```

EVEN
 SBTTL TYPE ROUTINE

```

4961
4962
4963
4964 ; *****
4965 ; *ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE
4966 ; *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED
4967 ; *NOTE1: SNULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER
4968 ; *NOTE2: SFILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED
4969 ; *NOTE3: SFILLC CONTAINS THE CHARACTER TO FILL AFTER
4970 .X
4971 ; *CALL.
4972 ; *1) USING A TRAP INSTRUCTION
4973 .X TYPE ,MESADR ; MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
4974 ; *OR
4975 .X TYPE
4976 .X MESADR
4977 .X
4978
4979 037610 105737 001157 STYPE: TSTB $TPFLG ; IS THERE A TERMINAL?
4980 037614 100002 BPL 15 ; BR IF YES
4981 037616 000000 HALT ; HALT HERE IF NO TERMINAL
4982 037620 000430 BR 35 ; LEAVE
4983 037622 010046 15. MOV RO, -(SP) ; SAVE RO
4984 037624 017600 000002 MOV @2(SP), RO ; GET ADDRESS OF ASCIZ STRING
4985 037630 122737 000001 001246 CMPB @APTENV, $ENV ; RUNNING IN APT MODE
4986 037636 001011 BNE 625 ; NO, GO CHECK FOR APT CONSOLE
4987 037640 132737 000100 001247 BITB @APTSPOOL, $ENV ; SPOOL MESSAGE TO APT
4988 037646 001405 BEQ 625 ; NO, GO CHECK FOR CONSOLE
4989 037660 010037 037660 MOV RO, 615 ; SETUP MESSAGE ADDRESS FOR APT
4990 037664 004737 040100 JSR PC, $ATY3 ; SPOOL MESSAGE TO APT
    
```



```

4991 037660 000000          61$    WORD    0          .. MESSAGE ADDRESS
4992 037662 132737 000040 001247 62$    BITB    #APT(CSUP $ENV#) .. APT CONSOLE SUPPRESSED
4993 037670 001003          60$    BNE     60$          .. YES, SKIP TYPE OUT
4994 037672 112046          2$    MOVB    (RO)+, -(SP) .. PUSH CHARACTER TO BE TYPED ONTO STACK
4995 037674 001005          BNE     4$          .. BR IF IT ISN'T THE TERMINATOR
4996 037676 005726          TST     (SP)+       .. IF TERMINATOR POP IT OFF THE STACK
4997 037700 012630          60$    MOV     (SP)+, RO   .. RESTORE RO
4998 037702 062716 000002    3$    ADD     #2, (SP)    .. ADJUST RETURN PC
4999 037706 000002          RTI                    .. RETURN
5000 037710 122716 000011    4$    CMPB    #HT, (SP)   .. BRANCH IF <HT>
5001 037714 001430          BEQ     8$          ..
5002 037716 122716 000200    CMPB    #CRLF, (SP) .. BRANCH IF NOT <CRLF>
5003 037722 001006          BNE     5$          ..
5004 037724 005726          TST     (SP)+       .. POP <CR><LF> EQUIV
5005 037726 104401          TYPE                    .. TYPE A CR AND LF
5006 037730 001223          $CRLF                    ..
5007 037732 105037 040066    CLRB    $CHARCNT    .. CLEAR CHARACTER COUNT
5008 037736 000755          BR      2$          .. GET NEXT CHARACTER
5009 037740 004737 040022    5$    JSR     PC, $TYPEC  .. GO TYPE THIS CHARACTER
5010 037744 123726 001156    6$    CMPB    $FILLC, (SP)+ .. IS IT TIME FOR FILLER CHARS ?
5011 037750 001350          BNE     2$          .. IF NO GO GET NEXT CHAR.
5012 037752 013746 001154    MOV     $NULL, -(SP) .. GET # OF FILLER CHARS NEEDED
5013                                .. AND THE NULL CHAR
5014 037756 105366 000001    7$    DECB    1(SP)       .. DOES A NULL NEED TO BE TYPED?
5015 037762 002770          BLT     6$          .. BR IF NO--GO POP THE NULL OFF OF STACK
5016 037764 004737 040022    JSR     PC, $TYPEC  .. GO TYPE A NULL
5017 037770 105337 040066    DECB    $CHARCNT    .. DO NOT COUNT AS A COUNT
5018 037774 000770          BR      7$          .. LOOP
5019
5020                                . HORIZONTAL TAB PROCESSOR
5021
5022 037776 112716 000040    8$    MOVB    #' , (SP)   .. REPLACE TAB WITH SPACE
5023 040002 004737 040022    9$    JSR     PC, $TYPEC  .. TYPE A SPACE
5024 040006 132737 000007 040066    BITB    #7, $CHARCNT .. BRANCH IF NOT AT
5025 040014 001372          BNE     9$          .. TAB STOP
5026 040016 005726          TST     (SP)+       .. POP SPACE OFF STACK
5027 040020 000724          BR      2$          .. GET NEXT CHARACTER
5028 040022 105777 141122    $TYPEC  TSTB    #STPS     .. WAIT UNTIL PRINTER IS READY
5029 040026 100375          BPL     $TYPEC      ..
5030 040030 116677 000002 141114    MOVB    2(SP), #STPB .. LOAD CHAR TO BE TYPED INTO DATA REG
5031 040036 122766 000015 000002    CMPB    #CR, 2(SP)  .. IS CHARACTER A CARRIAGE RETURN?
5032 040044 001003          BNE     1$          .. BRANCH IF NO
5033 040046 106037 040066    CLRB    $CHARCNT    .. YES--CLEAR CHARACTER COUNT
5034 040052 000406          BR      $TYPEX      .. EXIT
5035 040054 122766 000012 000002 1$    CMPB    #LF, 2(SP)  .. IS CHARACTER A LINE FEED?
5036 040062 001402          BEQ     $TYPEX      .. BRANCH IF YES
5037 040064 105227          INCB    (PC)+       .. COUNT THE CHARACTER
5038 040066 000000          $CHARCNT: WORD 0    .. CHARACTER COUNT STORAGE
5039 040070 000207          $TYPEX: RTS PC
5040
5041                                SBTTL APT COMMUNICATIONS ROUTINE
5042
5043                                . *****
5044 040072 112737 000001 040336  $ATY1: MOVB    #1, $FFLG .. TO REPORT FATAL ERROR
5045 040100 112737 000001 040334  $ATY3: MOVB    #1, $MFLG .. TO TYPE A MESSAGE
5046 040106 000403          BR      $ATYC
    
```

```

5047 040110 112737 000001 040336 SATY4 MOV8 #1,SFFLG .. TO ONLY REPORT FATAL ERROR
5048 040116 SATYC
5049 040116 010046 MOV RO,-(SP) .. PUSH RO ON STACK
5050 040120 010146 MOV R1,-(SP) .. PUSH R1 ON STACK
5051 040122 105737 040334 TSTB SFFLG .. SHOULD TYPE A MESSAGE?
5052 040126 001450 BEQ 55 .. IF NOT BR
5053 040130 122737 000001 001246 CMPE #APTENV,SENV .. OPERATING UNDER APT?
5054 040136 001031 BNE 35 .. IF NOT BR
5055 040140 132737 000100 001247 BITB #APTPOOL,SENV .. SHOULD SPOOL MESSAGES?
5056 040146 001425 BEQ 35 .. IF NOT BR
5057 040150 017600 000004 MOV #4(SP),RO .. GET MESSAGE ADDR
5058 040154 062766 000002 000004 ADD #2,4(SP) .. BUMP RETURN ADDR
5059 040162 005737 001226 15 TST SMSGTYPE .. SEE IF DONE W/ LAST XMISSION?
5060 040166 001375 BNE 15 .. IF NOT WAIT
5061 040170 010037 001242 MOV RO,SMSGAD .. PUT ADDR IN MAILBOX
5062 040174 105720 25 TSTB (RU)+ .. FIND END OF MESSAGE
5063 040176 001376 BNE 25
5064 040200 163700 001242 SUB SMSGAD,RO .. SUB START OF MESSAGE
5065 040204 006200 ASR RO .. GET MESSAGE LGTH IN WORDS
5066 040206 010037 001244 MOV RO,SMSG LGT .. PUT LENGTH IN MAILBOX
5067 040212 012737 000004 001226 MOV #4,SMSGTYPE .. TELL APT TO TAKE MSG.
5068 040220 000413 BR 55
5069 040222 017637 000004 040246 35 MOV #4(SP),45 .. PUT MSG ADDR IN JSR LINKAGE
5070 040230 062766 000002 000004 ADD #2,4(SP) .. BUMP RETURN ADDRESS
5071 040236 013746 177776 MOV 177776,-(SP) .. PUSH 177776 ON STACK
5072 040242 004737 037610 JSR PC,STYPE .. CALL TYPE MACRO
5073 040246 000000 45 WORD 0
5074 040250 55
5075 040250 105737 040336 105 TSTB SFFLG .. SHOULD REPORT FATAL ERROR?
5076 040254 001416 BEQ 125 .. IF NOT BR
5077 040256 005737 001246 TST SENV .. RUNNING UNDER APT?
5078 040262 001413 BEQ 125 .. IF NOT BR
5079 040264 005737 001226 115 TST SMSGTYPE .. FINISHED LAST MESSAGE?
5080 040270 001375 BNE 115 .. IF NOT WAIT
5081 040272 017637 000004 001230 MOV #4(SP),SFATAL .. GET ERROR #
5082 040300 062766 000002 000004 ADD #2,4(SP) .. BUMP RETURN ADDR
5083 040306 005237 001226 INC SMSGTYPE .. TELL APT TO TAKE ERROR
5084 040312 105037 040336 125 CLRB SFFLG .. CLEAR FATAL FLAG
5085 040316 105037 040335 CLRB SLFLG .. CLEAR LOG FLAG
5086 040322 105037 040334 CLRB SFFLG .. CLEAR MESSAGE FLAG
5087 040326 012601 MOV (SP)+,R1 .. POP STACK INTO R1
5088 040330 012600 MOV (SP)+,RO .. POP STACK INTO RO
5089 040332 000207 RTS PC .. RETURN
5090 040334 000 SFFLG BYTE 0 .. MESSG FLAG
5091 040336 000 SLFLG BYTE 0 .. LOG FLAG
5092 040336 000 SFFLG BYTE 0 .. FATAL FLAG
5093 040340 .EVEN
5094 000200 APTSIZE=200
5095 000001 APTENV=001
5096 000100 APTPOOL=100
5097 000040 APTCSUP=040
5098 SBTTL BINARY TO ASCII AND TYPE ROUTINE
5099
5100 .. *****
5101 .. THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
5102 .. BINARY-ASCII NUMBER AND TYPE IT
    
```

```

5103      .XCALL
5104      .X      MOV      NUMBER, -(SP)      ;; NUMBER TO BE TYPED
5105      .X      TYPBN
5106
5107      040340 010146      STYPBN  MOV      R1, -(SP)      ;; SAVE R1 ON THE STACK
5108      040342 016601 000006      MOV      6(SP), R1      ;; GET THE INPUT NUMBER
5109      040346 000261      SEC
5110      040350 112737 000060 040412 15      MOVBN   #'0, $BIN      ;; SET "C" SO CAN KEEP TRACK OF THE NUMBER OF BITS
5111      040356 006101      ROL     R1      ;; SET CHARACTER TO AN ASCII "0"
5112      040360 001406      BEQ     Z$      ;; GET THIS BIT
5113      040362 105537 040412      RDCB   $BIN      ;; DONE?
5114      040366 104401 040412      TYPE   , $BIN      ;; NO--SET THE CHARACTER EQUAL TO THIS BIT
5115      040372 000241      CLC
5116      040374 000765      BR      15      ;; GO TYPE THIS BIT
5117      040376 012601      BR      25      ;; CLEAR "C" SO CAN KEEP TRACK OF BITS
5118      040400 016666 000002 000004 25      MOV     (SP)+, R1      ;; GO DO THE NEXT BIT
5119      040406 012616      MOV     2(SP), 4(SP)  ;; POP THE STACK INTO R1
5120      040410 000002      MOV     (SP)+, (SP)  ;; ADJUST THE STACK
5121      040412      000      000      RTI      ;; RETURN TO USER
5122      SBIN   BYTE   0,0      ;; STORAGE FOR ASCII CHAR AND TERMINATOR
5123      SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
5124
5125      ;; *****
5126      ;; THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
5127      ;; OCTAL (ASCII) NUMBER AND TYPE IT.
5128      ;; $TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
5129      .XCALL
5130      .X      MOV     NUM, -(SP)      ;; NUMBER TO BE TYPED
5131      .X      TYPOS
5132      .X      BYTE   N      ;; CALL FOR TYPEOUT
5133      .X      BYTE   M      ;; N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
5134      .X
5135      .X      ;; 1=TYPE LEADING ZEROS
5136      .X      ;; 0=SUPPRESS LEADING ZEROS
5137      .X
5138      ;; $STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
5139      ;; $TYPOS OR $TYPOC
5140      .XCALL
5141      .X      MOV     NUM, -(SP)      ;; NUMBER TO BE TYPED
5142      .X      TYPON
5143      .X      ;; CALL FOR TYPEOUT
5144      .X
5145      ;; $TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
5146      .XCALL
5147      .X      MOV     NUM, -(SP)      ;; NUMBER TO BE TYPED
5148      .X      TYPOC
5149      .X      ;; CALL FOR TYPEOUT
5150      .X
5151      040414 017646 000000      STYPOS  MOV     2(SP), -(SP)      ;; PICKUP THE MODE
5152      040420 116637 000001 040637      MOVBN   1(SP), $OFILL      ;; LOAD ZERO FILL SWITCH
5153      040426 112637 040641      MOVBN   (SP)+, $OMODE+1    ;; NUMBER OF DIGITS TO TYPE
5154      040432 062716 000002      ADD     #2, (SP)      ;; ADJUST RETURN ADDRESS
5155      040436 000406      BR      STYPON
5156      040440 112737 000001 040637      STYPOC  MOVBN   #1, $OFILL      ;; SET THE ZERO FILL SWITCH
5157      040446 112737 000006 040641      MOVBN   #6, $OMODE+1    ;; SET FOR SIX(6) DIGITS
5158      040454 112737 000005 040636      STYPON  MOVBN   #5, $OCNT      ;; SET THE ITERATION COUNT
5159      040462 010346      MOV     R3, -(SP)      ;; SAVE R3
5160      040464 010446      MOV     R4, -(SP)      ;; SAVE R4
5161      040466 010546      MOV     R5, -(SP)      ;; SAVE R5
5162      040470 113704 040641      MOVBN   $OMODE+1, R4      ;; GET THE NUMBER OF DIGITS TO TYPE
    
```

```

5159 040474 005404          NEG      R4
5160 040476 062704 000006  ADD      #6,R4          .. SUBTRACT IT FOR MAX ALLOWED
5161 040502 110437 040640  MOVVB   R4,SOMODE     .. SAVE IT FOR USE
5162 040506 113704 040637  MOVVB   $OFILL,R4    .. GET THE ZERO FILL SWITCH
5163 040512 016605 000012  MOV     12(SP),R5    .. PICKUP THE INPUT NUMBER
5164 040516 005003          CLR      R3          .. CLEAR THE OUTPUT WORD
5165 040520 006105          15  ROL   R5          .. ROTATE MSB INTO "C"
5166 040522 000404          BR      3$          .. GO DO MSB
5167 040524 006105          25  ROL   R5          .. FORM THIS DIGIT
5168 040526 006105          ROL   R5
5169 040530 006105          ROL   R5
5170 040532 010503          MOV     R5,R3
5171 040534 006103          35  ROL   R3          .. GET LSB OF THIS DIGIT
5172 040536 105337 040640  DECB   $OMODE       .. TYPE THIS DIGIT?
5173 040542 100016          BPL    7$          .. BR IF NO
5174 040544 042703 177770  BIC    #177770,R3   .. GET RID OF JUNK
5175 040560 001002          BNE    4$          .. TEST FOR 0
5176 040562 005704          TST   R4          .. SUPPRESS THIS 0?
5177 040564 001403          BEQ   5$          .. BR IF YES
5178 040566 005204          45  INC   R4          .. DON'T SUPPRESS ANYMORE 0'S
5179 040568 052703 000060  BIS   #'0,R3       .. MAKE THIS DIGIT ASCII
5180 040564 052703 000040  55  BIS   #' ,R3     .. MAKE ASCII IF NOT ALREADY
5181 040570 110337 040634  MOVVB   R3,8$       .. SAVE FOR TYPING
5182 040574 104401 040634  TYPE   ,8$         .. GO TYPE THIS DIGIT
5183 040600 105337 040636  75  DECB   $OCNT     .. COUNT BY 1
5184 040604 003347          BGT   2$          .. BR IF MORE TO DO
5185 040606 002402          BLT   6$          .. BR IF DONE
5186 040610 005204          INC   R4          .. INSURE LAST DIGIT ISN'T A BLANK
5187 040612 000744          BR    2$          .. GO DO THE LAST DIGIT
5188 040614 012605          65  MOV   (SP)+,R5   .. RESTORE R5
5189 040616 012604          MOV   (SP)+,R4   .. RESTORE R4
5190 040620 012603          MOV   (SP)+,R3   .. RESTORE R3
5191 040622 016666 000002 000004  MOV   2(SP),4(SP) .. SET THE STACK FOR RETURNING
5192 040630 012616          MOV   (SP)+,(SP)
5193 040632 000002          RTI          .. RETURN
5194 040634 000          85  BYTE   0          .. STORAGE FOR ASCII DIGIT
5195 040635 000          BYTE   0          .. TERMINATOR FOR TYPE ROUTINE
5196 040636 000          $OCNT  BYTE   0          .. OCTAL DIGIT COUNTER
5197 040637 000          $OFILL  BYTE   0          .. ZERO FILL SWITCH
5198 040640 000000          $OMODE  WORD   0          .. NUMBER OF DIGITS TO TYPE
5199          SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
5200
5201          .. *****
5202          .. THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
5203          .. SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT DEPENDING ON WHETHER THE
5204          .. NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
5205          .. BEFORE THE FIRST DIGIT OF THE NUMBER LEADING ZEROS WILL ALWAYS BE
5206          .. REPLACED WITH SPACES
5207          .. CALL
5208          .. X      MOV     NUM,-(SP)          .. PUT THE BINARY NUMBER ON THE STACK
5209          .. X      TYPDS          .. GO TO THE ROUTINE
5210
5211          STYPDS.
5212          MOV     R0,-(SP)          .. PUSH R0 ON STACK
5213          MOV     R1,-(SP)          .. PUSH R1 ON STACK
5214          MOV     R2,-(SP)          .. PUSH R2 ON STACK
    
```

```

5215 040650 010346 MOV R3,-(SP) ..PUSH R3 ON STACK
5216 040652 010546 MOV R5,-(SP) ..PUSH R5 ON STACK
5217 040654 012746 020200 MOV #20200,-(SP) ..SET BLANK SWITCH AND SIGN
5218 040660 016605 000020 MOV 20(SP),R5 ..GET THE INPUT NUMBER
5219 040664 100004 BPL 1$ ..BR IF INPUT IS POS.
5220 040666 005405 NEG R5 ..MAKE THE BINARY NUMBER POS
5221 040670 112766 000055 000001 MOVB #'-,1(SP) ..MAKE THE ASCII NUMBER NEG
5222 040676 005000 1$ CLR R0 ..ZERO THE CONSTANTS INDEX
5223 040700 012703 041056 MOV #5DBLK,R3 ..SETUP THE OUTPUT POINTER
5224 040704 112723 000040 MOVB #'',(R3)+ ..SET THE FIRST CHARACTER TO A BLANK
5225 040710 005002 2$ CLR R2 ..CLEAR THE BCD NUMBER
5226 040712 016001 041046 MOV $DTBL(R0),R1 ..GET THE CONSTANT
5227 040716 160105 3$ SUB R1,R5 ..FORM THIS BCD DIGIT
5228 040720 002402 BLT 4$ ..BR IF DONE
5229 040722 005202 INC R2 ..INCREASE THE BCD DIGIT BY 1
5230 040724 000774 BR 3$
5231 040726 060105 4$ ADD R1,R5 ..ADD BACK THE CONSTANT
5232 040730 005702 TST R2 ..CHECK IF BCD DIGIT=0
5233 040732 001002 BNE 5$ ..FALL THROUGH IF 0
5234 040734 105716 TSTB (SP) ..STILL DOING LEADING 0'S?
5235 040736 100407 BMI 7$ ..BR IF YES
5236 040740 106316 5$ ASLB (SP) ..MSD?
5237 040742 103003 BCC 6$ ..BR IF NO
5238 040744 116663 000001 177777 MOVB 1(SP),-1(R3) ..YES-- GET THE SIGN
5239 040752 052702 000060 6$ BIS #'0,R2 ..MAKE THE BCD DIGIT ASCII
5240 040756 052702 000040 7$ BIS #' ,R2 ..MAKE IT A SPACE IF NOT ALREADY A DIGIT
5241 040762 110223 MOVB R2,(R3)+ ..PUT THIS CHARACTER IN THE OUTPUT BUFFER
5242 040764 005720 TST (R0)+ ..JUST INCREMENTING
5243 040766 020027 000010 CMP R0,#10 ..CHECK THE TABLE INDEX
5244 040772 002746 BLT 2$ ..GO DO THE NEXT DIGIT
5245 040774 003002 BGT 8$ ..GO TO EXIT
5246 040776 010502 MOV R5,R2 ..GET THE LSD
5247 041000 000764 BR 6$ ..GO CHANGE TO ASCII
5248 041002 105726 8$ TSTB (SP)+ ..WAS THE LSD THE FIRST NON-ZERO?
5249 041004 100003 BPL 9$ ..BR IF NO
5250 041006 116663 177777 177776 MOVB -1(SP),-2(R3) ..YES--SET THE SIGN FOR TYPING
5251 041014 105013 9$ CLRB (R3) ..SET THE TERMINATOR
5252 041016 012605 MOV (SP)+,R5 ..POP STACK INTO R5
5253 041020 012603 MOV (SP)+,R3 ..POP STACK INTO R3
5254 041022 012602 MOV (SP)+,R2 ..POP STACK INTO R2
5255 041024 012601 MOV (SP)+,R1 ..POP STACK INTO R1
5256 041026 012600 MOV (SP)+,R0 ..POP STACK INTO R0
5257 041030 104401 041056 TYPE ,5DBLK ..NOW TYPE THE NUMBER
5258 041034 016666 000002 000004 MOV 2(SP),4(SP) ..ADJUST THE STACK
5259 041042 012616 MOV (SP)+,(SP)
5260 041044 000002 RTI ..RETURN TO USER
5261 041046 023420 $DTBL 10000
5262 041050 001750 1000
5263 041052 000144 100
5264 041054 000012 10
5265 041056 000004 $DBLK BLKW 4
5266 SBTTL SAVE AND RESTORE R0-R5 ROUTINES
5267
5268 ..*****
5269 *SAVE R0-R5
5270 *CALL
  
```

```

5271 . * SAVREG
5272 . *UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE
5273 . *
5274 . *TOP---(+16)
5275 . * +2---(+18)
5276 . * +4---R5
5277 . * +6---R4
5278 . * +8---R3
5279 . *+10---R2
5280 . *+12---R1
5281 . *+14---R0
5282
5283 041066 $SAVREG
5284 041066 010046 MOV RO,-(SP) .. PUSH RO ON STACK
5285 041070 010146 MOV R1,-(SP) .. PUSH R1 ON STACK
5286 041072 010246 MOV R2,-(SP) .. PUSH R2 ON STACK
5287 041074 010346 MOV R3,-(SP) .. PUSH R3 ON STACK
5288 041076 010446 MOV R4,-(SP) .. PUSH R4 ON STACK
5289 041100 010546 MOV R5,-(SP) .. PUSH R5 ON STACK
5290 041102 016646 000022 MOV 22(SP),-(SP) .. SAVE PS OF MAIN FLOW
5291 041106 016646 000022 MOV 22(SP),-(SP) .. SAVE PC OF MAIN FLOW
5292 041112 016646 000022 MOV 22(SP),-(SP) .. SAVE PS OF CALL
5293 041116 016646 000022 MOV 22(SP),-(SP) .. SAVE PC OF CALL
5294 041122 000002 RTI
5295
5296 . *RESTORE RO-R5
5297 . *CALL
5298 . * RESREG
5299 041124 $RESREG
5300 041124 012666 000022 MOV (SP)+,22(SP) .. RESTORE PC OF CALL
5301 041130 012666 000022 MOV (SP)+,22(SP) .. RESTORE PS OF CALL
5302 041134 012666 000022 MOV (SP)+,22(SP) .. RESTORE PC OF MAIN FLOW
5303 041140 012666 000022 MOV (SP)+,22(SP) .. RESTORE PS OF MAIN FLOW
5304 041144 012605 MOV (SP)+,R5 .. POP STACK INTO R5
5305 041146 012604 MOV (SP)+,R4 .. POP STACK INTO R4
5306 041150 012603 MOV (SP)+,R3 .. POP STACK INTO R3
5307 041152 012602 MOV (SP)+,R2 .. POP STACK INTO R2
5308 041154 012601 MOV (SP)+,R1 .. POP STACK INTO R1
5309 041156 012600 MOV (SP)+,R0 .. POP STACK INTO R0
5310 041160 000002 RTI
5311 SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
5312
5313 . * *****
5314 . * THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
5315 . * UNSIGNED OCTAL ASCII NUMBER
5316 . * CALL
5317 . * MOV #PNTR,-(SP) .. POINTER TO LOW WORD OF BINARY NUMBER
5318 . * JSR PC,@#50B20 .. CALL THE ROUTINE
5319 . * RETURN .. THE ADDRESS OF THE FIRST ASCII CHAR IS ON THE STACK
5320
5321
5322 041162 104413 50B20 SAVREG .. SAVE ALL REGISTERS
5323 041164 016601 000002 MOV 2(SP),R1 .. PICKUP THE POINTER TO LOW WORD
5324 041170 012705 041301 MOV #5OCTVL+13,R5 .. POINTER TO DATA TABLE
5325 041174 012704 000014 MOV #12,R4 .. DO ELEVEN CHARACTERS
5326 041200 012703 177770 MOV #C7,R3 .. MASK

```

```

5327 041204 012100      MOV      (R1)+,R0      ;; LOWER WORD
5328 041206 012101      MOV      (R1)+,R1      ;; HIGH WORD
5329 041210 005002      CLR      R2            ;; TERMINATOR
5330 041212 110245      15      MOVB     R2,-(R5)     ;; PUT CHARACTER IN DATA TABLE
5331 041214 010002      MOV      R0,R2         ;; GET THIS DIGIT
5332 041216 005304      DEC      R4            ;; COUNT THIS CHARACTER
5333 041220 003007      BGT      3$           ;; BR IF NOT THE LAST DIGIT
5334 041222 001405      BEQ      2$           ;; BR IF IT IS THE LAST DIGIT
5335 041224 005205      INC      R5            ;; ALL DIGITS DONE-ADJUST POINTER FOR FIRST
5336 041226 010566      000002  MOV      R5,2(SP)     ;; ASCII CHAR. & PUT IT ON THE STACK
5337 041232 104414      RESREG                    ;; RESTORE ALL REGISTERS
5338 041234 000207      RTS      PC            ;; RETURN TO USER
5339 041236 006203      2$      ASR      R3            ;; POSITION THE MASK FOR THE LAST DIGIT
5340 041240 006001      3$      ROR      R1            ;; POSITION THE BINARY NUMBER FOR
5341 041242 006000      ROR      R0            ;;
5342 041244 006001      ROR      R1            ;;
5343 041246 006000      ROR      R0            ;;
5344 041250 006001      ROR      R1            ;;
5345 041252 006000      ROR      R0            ;;
5346 041254 040302      BIC      R3,R2         ;; MASK OUT ALL JUNK
5347 041256 062702      000060  ADD      #'0,R2         ;; MAKE THIS CHAR ASCII
5348 041262 000753      BR       '5            ;; GO PUT IT IN THE DATA TABLE
5349 041264 000016      SOCTVL  BLKB  14      ;; RESERVE DATA TABLE
    
```

```

5350          SBTTL TRAP DECODER
5351
5352          .. *****
5353          ; THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
5354          ; AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
5355          ; OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
5356          ; GO TO THAT ROUTINE
5357
5358 041302 010046          STRAP  MOV    RO, -(SP)          ;; SAVE RO
5359 041304 016600 000002          MOV    2(SP), RO          ;; GET TRAP ADDRESS
5360 041310 005740          TST    -(RO)          ;; BACKUP BY 2
5361 041312 111000          MOVB  (RO), RO          ;; GET RIGHT BYTE OF TRAP
5362 041314 006300          ASL   RO          ;; POSITION FOR INDEXING
5363 041316 016000 041336          MOV   STRPAD(RO), RO    ;; INDEX TO TABLE
5364 041322 000200          RTS    RO          ;; GO TO ROUTINE
5365
5366
5367          .. THIS IS USE TO HANDLE THE "GETPRI" MACRO
5368
5369 041324 011646          STRAP2 MOV   (SP), -(SP)    ;; MOVE THE PC DOWN
5370 041326 016666 000004 000002          MOV   4(SP), 2(SP)    ;; MOVE THE PSW DOWN
5371 041334 000002          RTI          ;; RESTORE THE PSW
5372
5373          SBTTL TRAP TABLE
5374
5375          ; THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
5376          ; BY THE "TRAP" INSTRUCTION
5377
5378          /
5379          /
5380          /
5381          /
5382          /
5383          /
5384          /
5385          /
5386          /
5387          /
5388 041354 036536          SGTSMR  ;; CALL=GTSMR    TRAP+7(104407)  GET SOFT-SMR SETTING
5389
5390          /
5391          /
5392          /
5393          /
5394          /
5395          /
5396          /
5397          /
5398          /
5399 041370 012737 041546 000024          SPSWRD: MOV   $SILLUP, @SPWRVEC ;; SET FOR FAST UP
5400 041376 012737 000340 000026          MOV   #340, @SPWRVEC+2 ;; PRIO-7
5401 041404 010046          MOV   RO, -(SP)          ;; PUSH RO ON STACK
5402 041406 010146          MOV   R1, -(SP)          ;; PUSH R1 ON STACK
5403 041410 010246          MOV   R2, -(SP)          ;; PUSH R2 ON STACK
5404 041412 010346          MOV   R3, -(SP)          ;; PUSH R3 ON STACK
5405 041414 010446          MOV   R4, -(SP)          ;; PUSH R4 ON STACK
    
```



```

5406 041416 010546          MOV      R5, -(SP)          ;; PUSH R5 ON STACK
5407 041420 017746 137514    MOV      @SWR, -(SP)       ;; PUSH @SWR ON STACK
5408 041424 010637 041552    MOV      SP, $SAVR6       ;; SAVE SP
5409 041430 012737 041442 000024 MOV      @SPWRUP, @@PWRVEC ;; SET UP VECTOR
5410 041436 000000          HALT
5411 041440 000776          BR      -2                ;; HANG UP
5412
5413
5414
5415 041442 012737 041546 000024 SPWRUP MOV      $FILLUP, @PWRVEC ;; SET FOR FAST DOWN
5416 041450 013706 041552    MOV      $SAVR6, SP      ;; GET SP
5417 041454 005037 041552    CLR      $SAVR6         ;; WAIT LOOP FOR THE TTY
5418 041460 005237 041552    15      INC      $SAVR6     ;; WAIT FOR THE INC
5419 041464 001375          BNE     15              ;; OF WORD
5420 041466 012677 137446    MOV      (SP)+, @SWR     ;; POP STACK INTO @SWR
5421 041472 012605          MOV      (SP)+, R5      ;; POP STACK INTO R5
5422 041474 012604          MOV      (SP)+, R4      ;; POP STACK INTO R4
5423 041476 012603          MOV      (SP)+, R3      ;; POP STACK INTO R3
5424 041500 012602          MOV      (SP)+, R2      ;; POP STACK INTO R2
5425 041502 012601          MOV      (SP)+, R1      ;; POP STACK INTO R1
5426 041504 012600          MOV      (SP)+, R0      ;; POP STACK INTO R0
5427 041506 012737 041370 000024 MOV      @SPWRDN, @PWRVEC ;; SET UP THE POWER DOWN VECTOR
5428 041514 012737 000340 000026 MOV      #340, @PWRVEC+2 ;; PRIO: 7
5429 041522 104401          TYPE
5430 041524 041554          SPWRMG WORD PWRMSG      ;; REPORT THE POWER FAILURE
5431 041526 012716          MOV      (PC)+, (SP)    ;; POWER FAIL MESSAGE POINTER
5432 041530 020000          SPWRAD WORD START      ;; RESTART AT START
5433 041532 042766 000020 000002 BIC      #20, 2(SP)     ;; RESTART ADDRESS
5434 041540 005037 035032    CLR      $TBIT         ;; CLEAR "T" BIT
5435 041544 000002          RTI
5436 041546 000000          $FILLUP HALT          ;; THE POWER UP SEQUENCE WAS STARTED
5437 041560 000776          BR      -2                ;; BEFORE THE POWER DOWN WAS COMPLETE
5438 041562 000000          $SAVR6 0              ;; PUT THE SP HERE
5439 041564 006412 050040 053517 PWRMSG  ASCIZ <12><15>? POWER FAILURE - RESTART NG ?<12><15>
5440 041562 051105 043040 044501
5441 041570 052514 042522 026440
5442 041576 051040 051505 040524
5443 041604 052122 047111 020107
5444 041612          000
5445          041616
5446
5447
    EVEN
    
```

Address	Hex 1	Hex 2	Hex 3	Hex 4	Hex 5	Label	Description
5448						SBTTL	ERROR MESSAGES, DATA HEADERS-TABLES & FORMATS
5449	041616	047125	054105	042520	EM1	ASCIZ	/UNEXPECTED CPU TRAP TO LOC 004/
5450	041624	052103	042105	04440			
5451	041632	052520	052040	040522			
5452	041640	020120	047524	046040			
5453	041646	041517	020056	030060			
5454	041654	000064					
5455	041656	047125	054105	042520	EM2	ASCIZ	/UNEXPECTED MEM MGMT TRAP TO LOC 250/
5456	041664	052103	042105	046440			
5457	041672	046505	020056	043515			
5458	041700	052115	020056	051124			
5459	041706	050101	052040	020117			
5460	041714	047514	027103	031040			
5461	041722	030065	000				
5462	041725	120	044522	051117	EM3	ASCIZ	/PRIORITY BITS SET WRONG IN PSW/
5463	041732	052111	020131	044502			
5464	041740	051524	051440	052105			
5465	041746	053440	047522	043516			
5466	041754	044440	020116	051520			
5467	041762	000127					
5468	041764	047515	042504	041040	EM4	ASCIZ	/MODE BITS SET WRONG IN PSW/
5469	041772	052111	020123	042523			
5470	042000	020124	051127	047117			
5471	042006	020107	047111	050040			
5472	042014	053523	000				
5473	042017	104	040525	020114	EM5	ASCIZ	/DUAL ADDRESSING BETWEEN HI&LO BYTES OF PSW/
5474	042024	042101	051104	051505			
5475	042032	044523	043516	041040			
5476	042040	052106	042527	047105			
5477	042046	044040	023111	047514			
5478	042064	041040	052131	051505			
5479	042062	047440	020106	051520			
5480	042070	000127					
5481	042072	042513	047122	046105	EM6	ASCIZ	/KERNEL R6 CHANGED BY WRITING USER R6/
5482	042100	051040	020066	044103			
5483	042106	047101	042507	020104			
5484	042114	054502	053440	044522			
5485	042122	044524	043516	052440			
5486	042130	042523	020122	033122			
5487	042136	000					
5488	042137	101	046440	046505	EM7	ASCIZ	/A MEMORY MGMT REG TIMED OUT/
5489	042144	051117	020131	043515			
5490	042162	052116	020056	042522			
5491	042160	027107	052040	046511			
5492	042166	042106	047440	052125			
5493	042174	000					
5494	042176	123	046526	040515	EM10	ASCIZ	/SUMMARY OF MEM MGMT REG TIMEOUTS/
5495	042202	054522	047440	020106			
5496	042210	042516	027116	046440			
5497	042216	046507	027124	051040			
5498	042224	043506	020056	044524			
5499	042232	042516	052517	051524			
5500	042240	000					
5501	042241	116	046505	020056	EM11	ASCIZ	/MEM MGMT REG WOULD NOT CLEAR/
5502	042246	043515	052115	020056			
5503	042254	042522	027107	053440			

5504	042262	052517	042114	047040				
5505	042270	052117	041440	042514				
5506	042276	051101	000					
5507	042301	115	046505	020056	EM12	ASCIZ	/MEM MGMT REG BITS NOT SET CORRECTLY/	
5508	042306	043515	052115	020056				
5509	042314	042522	027107	041040				
5510	042322	052111	020123	047516				
5511	042330	020124	042523	020124				
5512	042336	047503	051122	041505				
5513	042344	046124	000131					
5514	042350	051123	020060	043105	EM13	ASCIZ	/SRO EFFECTED BY WRITE TO PSW/	
5515	042356	042506	052103	042105				
5516	042364	041040	020131	051127				
5517	042372	052111	020105	047524				
5518	042400	050040	053523	000				
5519	042405	123	030522	042040	EM14	ASCIZ	/SR1 DID NOT READ ALL ZEROS/	
5520	042412	042111	047040	052117				
5521	042420	051040	040505	020104				
5522	042426	046101	020114	042532				
5523	042434	047522	000123					
5524	042440	052504	046101	040440	EM15	ASCIZ	/DUAL ADDRESSING BETWEEN BYTES OF PAR OR PDR/	
5525	042446	042104	042522	051523				
5526	042454	047111	020107	042502				
5527	042462	053524	042505	020116				
5528	042470	054502	042524	020123				
5529	042476	043117	050040	051101				
5530	042504	047440	020122	042120				
5531	042512	000122						
5532	042514	052504	046101	040440	EM16	ASCIZ	/DUAL ADDRESSING BETWEEN PAR-PDR'S/	
5533	042522	042104	042522	051523				
5534	042530	047111	020107	042502				
5535	042536	053524	042505	020116				
5536	042544	040520	026522	042120				
5537	042552	023522	000123					
5538	042556	044120	051531	020056	EM17	ASCIZ	/PHYS ADDR FORMED WRONG IN MAINT MODE/	
5539	042564	042101	051104	020056				
5540	042572	047506	046522	042105				
5541	042600	053440	047522	043516				
5542	042606	044440	020116	040515				
5543	042614	047111	027124	046440				
5544	042622	042117	000105					
5545	042626	044120	051531	020056	EM20	ASCIZ	/PHYS ADDR FORMED WRONG IN RELOCATE MODE/	
5546	042634	042101	051104	020056				
5547	042642	047506	046522	042105				
5548	042650	053440	047522	043516				
5549	042656	044440	020116	042522				
5550	042664	047514	040503	042524				
5551	042672	046440	042117	000105				
5552	042700	026527	044502	020124	EM21	ASCIZ	/W-BIT DID NOT GET SET IN PDR/	
5553	042706	044504	020104	047516				
5554	042714	020124	042507	020124				
5555	042722	042523	020124	047111				
5556	042730	050040	051104	000				
5557	042735	127	041055	052111	EM22	ASCIZ	/W-BIT SET IN MORE THAN ONE PDR/	
5558	042742	051440	052105	044440				
5559	042750	020116	047515	042522				

5560	042756	052040	040510	020116			
5561	042764	047117	020105	042120			
5562	042772	000122					
5563	042774	026527	044502	020124	EM23	ASCIZ	/W-BIT NOT CLEARED BY WRITING TO PDR/
5564	043002	047516	020124	046103			
5565	043010	040505	042522	020104			
5566	043016	044502	053440	044522			
5567	043024	044524	043516	052040			
5568	043032	020117	042120	000122			
5569	043040	051127	052111	047111	EM24	ASCIZ	/WRITING SRO SET W-BIT IN KIPDR/
5570	043046	020107	051123	020060			
5571	043054	042523	020124	026527			
5572	043062	044502	020124	047111			
5573	043070	045440	050111	051104			
5574	043076	000067					
5575	043100	026527	044502	020124	EM25	ASCIZ	/W-BIT GOT SET DURING ODD ADDR ABORT/
5576	043106	047507	020124	042523			
5577	043114	020124	052504	044522			
5578	043122	043516	047440	042104			
5579	043130	040440	042104	027122			
5580	043136	040440	047502	052122			
5581	043144	000					
5582	043146	115	044505	051117	EM26	ASCIZ	/MEMORY MGMT ACCESS ABORT DID NOT OCCUR/
5583	043152	020131	043515	052115			
5584	043160	020056	041501	042503			
5585	043166	051523	040440	047502			
5586	043174	052122	042040	042111			
5587	043202	047040	052117	047440			
5588	043210	041503	051125	000			
5589	043216	101	041503	051505	EM27	ASCIZ	/ACCESS ERROR DID NOT ABORT INSTRUCTION/
5590	043222	020123	051105	047522			
5591	043230	020122	044504	020104			
5592	043236	047516	020124	041101			
5593	043244	051117	020124	047111			
5594	043252	052123	052522	052103			
5595	043260	047511	000116				
5596	043264	051123	020060	044504	EM30	ASCIZ	/SRO DID NOT REPORT ACCESS ERROR CORRECTLY/
5597	043272	020104	047516	020124			
5598	043300	042522	047520	052122			
5599	043306	040440	041503	051505			
5600	043314	020123	051105	047522			
5601	043322	020122	047503	051122			
5602	043330	041505	046124	000131			
5603	043336	044504	020104	047516	EM31	ASCIZ	/DID NOT LOCKUP CORRECT VIRTUAL ADDR /
5604	043344	020124	047514	045503			
5605	043352	050125	041440	051117			
5606	043360	042522	052103	053040			
5607	043366	051111	052524	046101			
5608	043374	040440	042104	027122			
5609	043402	000					
5610	043403	120	043501	020105	EM32	ASCIZ	/PAGE LGTH ABORT OCCURRED WHEN IT SHOULDN'T HAVE/
5611	043410	043514	044124	020056			
5612	043416	041101	051117	020124			
5613	043424	041517	052503	051122			
5614	043432	042105	053440	042510			
5615	043440	020116	052111	051440			

5616	043446	047510	046125	047104		
5617	043454	052047	044040	053101		
5618	043462	000106				
5619	043464	040520	042507	046040	EM33	ASCIZ /PAGE LGTH ABORT DID NOT OCCUR WHEN IT SHOULD HAVE/
5620	043472	062107	027110	040440		
5621	043500	047502	062122	042040		
5622	043506	042111	047040	052117		
5623	043514	047440	041503	051125		
5624	043522	063440	042510	020116		
5625	043530	062111	051440	047510		
5626	043536	046125	020104	040510		
5627	043544	042526	000			
5628	043547	123	030122	042040	EM34	ASCIZ /SRO DID NOT REPORT PAGE LGTH ABORT CORRECTLY/
5629	043554	042111	047040	052117		
5630	043562	061040	060106	051117		
5631	043570	020124	040520	042507		
5632	043576	046040	062107	027110		
5633	043604	040440	047502	052122		
5634	043612	041440	061117	042522		
5635	043620	062103	064514	000		
5636	043626	123	030122	047440	EM37	ASCIZ /SRO OR SR2 CHANGED BY A SECOND ABORT/
5637	043632	020122	061123	020062		
5638	043640	044103	047101	042507		
5639	043646	020104	064502	040440		
5640	043654	061440	041505	047117		
5641	043662	020104	041101	051117		
5642	043670	000124				
5643	043672	061123	020060	051117	EM40	ASCIZ /SRO OR SR2 WERE NOT "RESET" BY A RESET/
5644	043700	061440	031122	053440		
5645	043706	061106	020106	047516		
5646	043714	020124	061042	051505		
5647	043722	062106	020042	064502		
5648	043730	040440	061040	051505		
5649	043736	062106	000			
5650	043741	123	031122	047040	EM41	ASCIZ /SR2 NOT TRACKING CORRECTLY/
5651	043746	062117	062040	040522		
5652	043754	045603	047111	020107		
5653	043762	047502	061122	041505		
5654	043770	046124	000131			
5655	043774	044504	020104	047516	EM42	ASCIZ /DID NOT TRAP THRU KERNEL SPACE/
5656	044002	020124	061124	047516		
5657	044010	062040	061110	020125		
5658	044016	042513	047122	046105		
5659	044024	061440	040520	042503		
5660	044032	000				
5661	044033	113	020124	051105	EM43	ASCIZ /KT ERROR SERVICED ON ODD ADDR ERROR/
5662	044040	047522	020122	042523		
5663	044046	063122	041511	042105		
5664	044054	047440	020116	042117		
5665	044062	020104	042101	051104		
5666	044070	020066	061106	047522		
5667	044076	000122				
5668	044100	061123	020060	051117	EM44	ASCIZ /SRO OR SR2 CHANGED BY ODD ADDR ERROR/
5669	044106	061440	031122	041440		
5670	044114	040510	043516	042105		
5671	044122	041040	020131	042117		

5672	044130	020104	042101	051104		
5673	044136	020056	051105	047522		
5674	044144	000122				
5675	044146	051105	047522	020122	EM45	ASCIZ /ERROR DURING "DOUBLE ERROR" (KT & ODD ADDR) /
5676	044154	052504	044522	043516		
5677	044162	021040	047504	041125		
5678	044170	042514	042440	051122		
5679	044176	051117	020042	045450		
5680	044204	020124	020046	042117		
5681	044212	020104	042101	051104		
5682	044220	024456	000			
5683	044223	115	050106	020111	EM46	ASCIZ /MFPI INSTRUCTION PUSHED WRONG DATA/
5684	044230	047111	052123	052522		
5685	044236	052103	047511	020116		
5686	044244	052520	044123	042105		
5687	044252	053440	047522	043516		
5688	044260	042040	052101	000101		
5689	044266	052115	044520	044440	EM47	ASCIZ /MTP1 INSTRUCTION LOADED WRONG DATA/
5690	044274	051516	051124	041525		
5691	044302	044524	047117	046040		
5692	044310	040517	042504	020104		
5693	044316	051127	047117	020107		
5694	044324	040504	040524	000		
5695	044331	123	040524	045503	EM50	ASCIZ /STACK NOT PUSHED BY MFPI-MTP1/
5696	044336	047040	052117	050040		
5697	044344	051525	042510	020104		
5698	044352	054502	046440	050106		
5699	044360	026511	052115	044520		
5700	044366	000				
5701	044367	113	051105	042516	EM51	ASCIZ /KERNEL PAGE ACCESS INSTEAD OF USER MFPI-MTP1/
5702	044374	020114	040520	042507		
5703	044402	040440	041503	051505		
5704	044410	020123	047111	052123		
5705	044416	040505	020104	043117		
5706	044424	052440	042523	035122		
5707	044432	046440	050106	026511		
5708	044440	052115	044520	000		
5709	044446	127	047522	043516	EM52	ASCIZ /WRONG PDR'S REFERENCED WHILE IN RELOCATE MODE/
5710	044452	050040	051104	051447		
5711	044460	051040	043105	051105		
5712	044466	047105	042503	020104		
5713	044474	044127	046111	020105		
5714	044502	047111	051040	046105		
5715	044510	041517	052101	020105		
5716	044516	047515	042504	000		
5717	044523	115	050106	020104	EM53	ASCIZ /MFPD INSTRUCTION PUSHED WRONG DATA/
5718	044530	047111	052123	052522		
5719	044536	052103	047511	020116		
5720	044544	052520	044123	042105		
5721	044552	053440	047522	043516		
5722	044560	042040	052101	000101		
5723	044566	052123	041501	020113	EM54	ASCIZ /STACK NOT PUSHED BY MFPD-MTPD/
5724	044574	047516	020124	052520		
5725	044602	044123	042105	041040		
5726	044610	020131	043115	042120		
5727	044616	046455	050124	000104		

5728	044624	040520	020122	051117	EM55	ASCIZ /PAR OR PDR CHANGED BY A RESET/
5729	044632	050040	051104	041440		
5730	044640	040510	043516	042105		
5731	044646	041040	020131	020101		
5732	044654	042522	042523	000124		
5733	044662	046111	042514	040507	EM56	ASCIZ /ILLEGAL MODE 01 NOT ABORTED/
5734	044670	020114	047515	042504		
5735	044676	030040	020061	047516		
5736	044704	020124	041101	051117		
5737	044712	042524	000104			
5738	044716	061123	020060	044504	EM57	ASCIZ /SRO DID NOT REPORT ILLEGAL MODE 01 CORRECTLY/
5739	044724	020104	047516	020124		
5740	044732	042522	047520	052122		
5741	044740	044440	046114	043505		
5742	044746	046101	046440	042117		
5743	044754	020106	030460	041440		
5744	044762	051117	042522	052103		
5745	044770	054514	000			
5746	044773	120	053523	041440	EM60	ASCIZ /PSW CHANGED BY AN RTI IN USER MODE/
5747	045000	040610	043516	042105		
5748	045006	041040	020131	047101		
5749	045014	051040	044524	044440		
5750	045022	020116	051525	051105		
5751	045030	046440	042117	000105		
5752	045036	040616	047111	027124	EM61	ASCIZ /MAINT MODE (SRO <8>) NOT DISABLED BY A RESET/
5753	045044	046440	042117	020105		
5754	045052	051450	030122	036040		
5755	045060	037070	020061	047516		
5756	045066	020124	044504	040523		
5757	045074	046102	042105	041040		
5758	045102	020131	020101	042522		
5759	045110	042523	000124			
5760	045114	040604	040624	044440	EM62	ASCIZ /DATA INCORRECT AFTER A MAINT MODE WRITE/
5761	045122	041516	051117	042522		
5762	045130	052103	040440	052106		
5763	045136	051105	040440	046440		
5764	045144	044501	052116	020056		
5765	045152	047516	042504	053440		
5766	045160	044522	042524	000		
5767	045166	123	052517	041522	EM63	ASCIZ /SOURCE RELOCATED IN MAINT MODE/
5768	045172	020105	042522	047514		
5769	045200	040603	042524	020104		
5770	045206	047111	046440	044501		
5771	045214	052116	020066	047515		
5772	045222	042504	000			
5773	045228	116	047117	051040	EM64	ASCIZ /NON RESIDENT ABORT DID NOT OCCUR/
5774	045232	051605	042111	047105		
5775	045240	020124	041101	051117		
5776	045246	020124	044504	020104		
5777	045254	047516	020124	041517		
5778	045262	052503	000122			
5779	045266	051105	047522	020122	EM65	ASCIZ /ERROR FLAG FOR NR ABORT (BIT15) IN SRO DID NOT SET/
5780	045274	046106	043501	043040		
5781	045302	051117	047040	020122		
5782	045310	041101	051117	020124		
5783	045316	041050	052111	032461		

5784	045324	020051	047111	051440					
5785	045332	030122	042040	042111					
5786	045340	047040	052117	051440					
5787	045346	052105	000						
5788	045351	123	031122	042040	EM66	ASCIZ	/SR2 DID NOT FREEZE THE VIRTUAL ADDR OF THE ABORTD INTR/		
5789	045356	042111	047040	052117					
5790	045364	043040	042522	055105					
5791	045372	020105	044124	020105					
5792	045400	044526	052122	040525					
5793	045406	020114	042101	051104					
5794	045414	020123	043117	052040					
5795	045422	042510	040440	047502					
5796	045430	052122	020104	047111					
5797	045436	051124	000						
5798	045441	062	042116	047040	EM67	ASCIZ	/2ND NON RESIDENT ABORT DID NOT OCCUR/		
5799	045446	047117	051040	051505					
5800	045454	042111	047105	020124					
5801	045462	041101	051117	020124					
5802	045470	044504	020104	047516					
5803	045476	020124	041517	052503					
5804	045504	000122							
5805	045506	047062	020104	047516	EM70	ASCIZ	/2ND NON RESIDENT ABORT CAUSED SR2 TO CHANGE/		
5806	045514	020116	042522	044523					
5807	045522	042504	052116	040440					
5808	045530	047502	052122	041440					
5809	045536	052501	042523	020104					
5810	045544	051123	020062	047524					
5811	045552	041440	040510	043516					
5812	045560	000105							
5813	045562	051123	020060	040527	EM71	ASCIZ	/SR0 WAS NOT CLEARED BY INIT /		
5814	045570	020123	047516	020124					
5815	045576	046103	040505	042522					
5816	045604	020104	054502	044440					
5817	045612	044516	027124	000040					
5818									
5819	045620	046117	020104	041520	DH1	ASCIZ	/OLD PC OLD PSW R6 WAS TESTNO ERRORPC/		
5820	045626	020040	046117	020104					
5821	045634	051520	020127	033122					
5822	045642	053440	051501	020040					
5823	045650	042524	052123	047516					
5824	045656	020040	051105	047522					
5825	045664	050122	000103						
5826	045670	046117	020104	041520	DH2	ASCIZ	/OLD PC OLD PSW R6 WAS SR0 SR2 TESTNO ERRORPC/		
5827	045676	020040	046117	020104					
5828	045704	051520	020127	033122					
5829	045712	053440	051501	020040					
5830	045720	051123	020060	020040					
5831	045726	020040	051123	020062					
5832	045734	020040	020040	042524					
5833	045742	052123	047516	020040					
5834	045750	051105	047522	050122					
5835	045756	000103							
5836	045760	051127	052117	020105	DH3	ASCIZ	/WROTE READ TESTNO ERRORPC/		
5837	045766	020040	042522	042101					
5838	045774	020040	020040	042524					
5839	046002	052123	047516	020040					

5840	046010	051105	047522	050122						
5841	046016	000103								
5842	046020	042101	051104	051505	DH7	ASCIZ	/ADDRESS TESTNO	ERRORPC/		
5843	046026	020123	042524	052123						
5844	046034	047516	020040	051105						
5845	046042	047522	050122	000103						
5846	046050	042522	044507	052123	DH10	ASCII	/REGISTER-ADDRS	NUM OF/<<CRLF>		
5847	046056	051105	040455	042104						
5848	046064	051522	020040	052516						
5849	046072	020115	047440	100106						
5850	046100	047101	026504	042105		ASCIZ	/AND-ED OR-ED	TIMOUTS TESTNO	ERRORPC/	
5851	046106	020040	051117	042455						
5852	046114	020104	020040	044524						
5853	046122	047515	052125	020123						
5854	046130	042524	052123	047516						
5855	046136	020040	051105	047522						
5856	046144	050122	000103							
5857	046150	042522	044507	052123	DH11	ASCII	/REGISTR READ	READ-(BINARY)/<<CRLF>		
5858	046156	020122	042522	042101						
5859	046164	020040	020040	042522						
5860	046172	042101	024056	044502						
5861	046200	040516	054522	100051						
5862	046206	042101	051104	051505		ASCIZ	/ADDRESS (OCTAL)	5432109876543210	TESTNO	ERRORPC/
5863	046214	020123	047450	052103						
5864	046222	046101	020051	032065						
5865	046230	031063	030061	034071						
5866	046236	033067	032065	031063						
5867	046244	030061	020040	042524						
5868	046252	052123	047516	020040						
5869	046260	051105	047522	050122						
5870	046266	000103								
5871	046270	042522	044507	052123	DH12	ASCII	/REGISTR WROTE	READ	READ-(BINARY)/<<CRLF>	
5872	046276	020122	051127	052117						
5873	046304	020105	020040	042522						
5874	046312	042101	020040	020040						
5875	046320	042522	042101	024055						
5876	046326	044502	040516	054522						
5877	046334	100051								
5878	046336	042101	051104	051505		ASCIZ	/ADDRESS (OCTAL)	(OCTAL) 5432109876543210	TESTNO	ERRORPC/
5879	046344	020123	047450	052103						
5880	046352	046101	020051	047450						
5881	046360	052103	046101	020051						
5882	046366	032065	031063	030061						
5883	046374	034071	033067	032065						
5884	046402	031063	030061	020040						
5885	046410	042524	052123	047516						
5886	046416	020040	051105	047522						
5887	046424	050122	000103							
5888	046430	042522	042101	020040	DH13	ASCIZ	/READ	TESTNO	ERRORPC/	
5889	046436	020040	042524	052123						
5890	046444	047516	020040	051105						
5891	046452	047522	050122	000103						
5892	046460	040520	026522	042120	DH16	ASCII	/PAR-PDR PAR-FDR/<<CRLF>			
5893	046466	020122	040520	026522						
5894	046474	042120	100122							
5895	046500	046103	040505	042522		ASCIZ	/CLEARED EFFECTD	EXPECTD RECEIVD	TESTNO	ERRORPC/

6008	047652	020056	045440	050111					
6009	047660	051104	020064	051440					
6010	047666	031122	053440	051501					
6011	047674	042440	050130	041505					
6012	047702	042124	052040	051505					
6013	047710	047124	020117	042440					
6014	047716	051122	051117	041520					
6015	047724	000							
6016	047725	123	031122	053440	DH36	ASCIZ	/SR2 WAS EXPECTD	TESTNO	ERRORPC/
6017	047732	051501	042440	050130					
6018	047740	041505	042124	052040					
6019	047746	051505	047124	020117					
6020	047754	042440	051122	051117					
6021	047762	041520	000						
6022	047765	106	051111	052123	DH37	ASC I	/FIRST ABORT	SECOND ABORT/	<<CRLF>
6023	047772	040440	047502	052122					
6024	050000	020040	020040	051440					
6025	050006	041505	047117	020104					
6026	050014	041101	051117	100124					
6027	050022	051123	020060	040527		ASCIZ	/SRO WAS SP2 WAS	SRO WAS SR2 WAS	TESTNO ERRORPC/
6028	050030	020123	051123	020062					
6029	050036	040527	020123	051123					
6030	050044	020060	040527	020123					
6031	050052	051123	020062	040527					
6032	050060	020123	042524	052123					
6033	050066	047516	020040	051105					
6034	050074	047522	050122	000103					
6035	050102	051123	020060	040527	DH40	ASCIZ	/SRO WAS SR2 WAS	TESTNO	ERRORPC/
6036	050110	020123	051123	020062					
6037	050116	040527	020123	042524					
6038	050124	052123	047516	020040					
6039	050132	051105	047522	050122					
6040	050140	000103							
6041	050142	051520	020127	040527	DH42	ASCIZ	/PSW WAS R6 WAS	TESTNO	ERRORPC/
6042	050150	020123	033122	053440					
6043	050156	051501	020040	042524					
6044	050164	052123	047516	020040					
6045	050172	051105	047522	050122					
6046	050200	000103							
6047	050202	054105	042520	052103	DH44	ASCII	/EXPECTED	RECEIVED/	<<CRLF>
6048	050210	042105	020040	020040					
6049	050216	020040	020040	051040					
6050	050224	041505	044505	042526					
6051	050232	100104							
6052	050234	051123	020060	020040		ASCIZ	/SRO	SR2	SRO WAS SR2 WAS TESTNO ERRORPC/
6053	050242	020040	051123	020062					
6054	050250	020040	020040	051123					
6055	050256	020060	040527	020123					
6056	050264	051123	020062	040527					
6057	050272	020123	042524	052123					
6058	050300	047516	020040	051105					
6059	050306	047522	050122	000103					
6060	050314	054105	042520	052103	DH45	ASCII	/EXPECTED	<<CRLF>	
6061	050322	042105	100072						
6062	050326	051520	020127	020040		ASCII	/PSW	PC	SRO SR2/<<CRLF>
6063	050334	020040	041520	020040					

6120	051022	000103									
6121	051024	051520	020127	040527	DM60	ASCIZ	/PSW WAS EXPECTD TESTNO	ERRORPC/			
6122	051032	020123	054105	042520							
6123	051040	052103	020104	042524							
6124	051046	052123	047516	020040							
6125	051054	051105	047522	050122							
6126	051062	000103									
6127	051064	051123	020060	020040	DM61	ASCIZ	/SRO SR2 TESTNO	ERRORPC/			
6128	051072	020040	051123	020062							
6129	051100	020040	042524	052123							
6130	051106	047516	020040	051105							
6131	051114	047522	050122	000103							
6132	051122	051123	020062	054105	DM62	ASCIZ	/SR2 EXP SR2 REC TESTNO	ERRORPC/			
6133	051130	020120	051123	020062							
6134	051136	042522	020103	042524							
6135	051144	052123	047516	042440							
6136	051152	051122	051117	041520							
6137	051160	000									
6138	051161	123	030122	042440	DM63	ASCIZ	/SRO EXP SRO REC TESTNO	ERRORPC/			
6139	051166	050130	051440	030122							
6140	051174	051040	041505	052040							
6141	051202	051505	047124	020117							
6142	051210	051105	047522	050122							
6143	051216	000103									
6144											
6145											
6146											
6147	051220	001356	001360	001354	DT1	WORD	TRAPPC, TRAPPS, WASR6, TESTNO, SERRPC, 0				
6148	051226	001352	001116	000000							
6149	051234	001356	001360	001354	DT2	WORD	TRAPPC, TRAPPS, WASR6, WASSRO, WASSR2, TESTNO, SERRPC, 0				
6150	051242	001366	001370	001352							
6151	051250	001116	000000								
6152	051254	001162	001164	001352	DT3	WORD	\$REG0, \$REG1, TESTNO, SERRPC, 0				
6153	051262	001116	000000								
6154	051266	001162	001352	001116	DT7	WORD	\$REG0, TESTNO, SERRPC, 0				
6155	051274	000000									
6156	051276	001374	001376	001400	DT10	WORD	ANDADR, ORADR, TOMUM, TESTNO, SERRPC, 0				
6157	051304	001352	001116	000000							
6158	051312	001162	001164	001164	DT11	WORD	\$REG0, \$REG1, \$REG1, TESTNO, SERRPC, 0				
6159	051320	001352	001116	000000							
6160	051326	001162	001164	001166	DT12	WORD	\$REG0, \$REG1, \$REG2, \$REG2, TESTNO, SERRPC, 0				
6161	051334	001166	001352	001116							
6162	051342	000000									
6163	051344	001162	001352	001116	DT13	WORD	\$REG0, TESTNO, SERRPC, 0				
6164	051352	000000									
6165	051354	001162	001164	001174	DT16	WORD	\$REG0, \$REG1, \$REG5, \$REG2, TESTNO, SERRPC, 0				
6166	051362	001166	001352	001116							
6167	051370	000000									
6168	051372	001406	001402	001172	DT17	WORD	PBALO, VIRT1, \$REG4, TESTNO, SERRPC, 0				
6169	051400	001352	001116	000000							
6170	051406	001406	001402	001404	DT20	WORD	PBALO, VIRT1, VIRT2, \$REG4, \$REG5, STMPO, TESTNO, SERRPC, 0				
6171	051414	001172	001174	001176							
6172	051422	001352	001116	000000							
6173	051430	001174	001170	001352	DT21	WORD	\$REG5, \$REG3, TESTNO, SERRPC, 0				
6174	051436	001116	000000								
6175	051442	001162	001174	001170	DT22	WORD	\$REG0, \$REG5, \$REG3, TESTNO, SERRPC, 0				

6176	051450	001352	001116	000000					
6177	051456	001174	001352	001116	DT23	WORD	\$REG5, TESTNO, \$ERRPC, 0		
6178	051464	000000							
6179	051466	001166	001164	001352	DT24	WORD	\$REG2, \$REG1, TESTNO, \$ERRPC, 0		
6180	051474	001116	000000						
6181	051500	001166	001176	001352	DT26	WORD	\$REG2, \$TMP0, TESTNO, \$ERRPC, 0		
6182	051506	001116	000000						
6183	051512	001366	001170	001166	DT30	WORD	WASSRO, \$REG3, \$REG2, \$TMP0, TESTNO, \$ERRPC, 0		
6184	051520	001176	001352	001116					
6185	051526	000000							
6186	051530	001370	001172	001166	DT31	WORD	WASSR2, \$REG4, \$REG2, \$TMP0, TESTNO, \$ERRPC, 0		
6187	051536	001176	001352	001116					
6188	051544	000000							
6189	051546	001162	001172	001366	DT32	WORD	\$REG0, \$REG4, WASSRO, WASSR2, TESTNO, \$ERRPC, 0		
6190	051554	001370	001352	001116					
6191	051562	000000							
6192	051564	001162	001172	001352	DT33	WORD	\$REG0, \$REG4, TESTNO, \$ERRPC, 0		
6193	051572	001116	000000						
6194	051576	001162	001172	001366	DT34	WORD	\$REG0, \$REG4, WASSRO, \$REG2, TESTNO, \$ERRPC, 0		
6195	051604	001166	001352	001116					
6196	051612	000000							
6197	051614	001162	001172	001370	DT35	WORD	\$REG0, \$REG4, WASSR2, \$REG3, TESTNO, \$ERRPC, 0		
6198	051622	001170	001352	001116					
6199	051630	000000							
6200	051632	001370	001164	001352	DT36	WORD	WASSR2, \$REG1, TESTNO, \$ERRPC, 0		
6201	051640	001116	000000						
6202	051644	001176	001202	001366	DT37	WORD	\$TMP0, \$TMP2, WASSRO, WASSR2, TESTNO, \$ERRPC, 0		
6203	051662	001370	001352	001116					
6204	051660	000000							
6205	051662	001366	001370	001352	DT40	WORD	WASSRO, WASSR2, TESTNO, \$ERRPC, 0		
6206	051670	001116	000000						
6207	051674	001164	001166	001352	DT42	WORD	\$REG1, \$REG2, TESTNO, \$ERRPC, 0		
6208	051702	001116	000000						
6209	051706	001162	001164	001366	DT44	WORD	\$REG0, \$REG1, WASSRO, WASSR2, TESTNO, \$ERRPC, 0		
6210	051714	001370	001352	001116					
6211	051722	000000							
6212	051724	001164	001170	001366	DT45	WORD	\$REG1, \$REG3, WASSRO, WASSR2, TESTNO, \$ERRPC, 0		
6213	051732	001370	001352	001116					
6214	051740	000000							
6215	051742	001162	001164	001352	DT46	WORD	\$REG0, \$REG1, TESTNO, \$ERRPC, 0		
6216	051760	001116	000000						
6217	051764	001362	001116	000000	DT50	WORD	TESTNO, \$ERRPC, 0		
6218	051762	001366	001370	001352	DT51	WORD	WASSRO, WASSR2, TESTNO, \$ERRPC, 0		
6219	051770	001116	000000						
6220	051774	001406	001402	001172	DT52	WORD	PBFLO, VIRT1, \$REG4, WASSRO, WASSR2, \$TMP0, TESTNO, \$ERRPC, 0		
6221	052002	001366	001370	001176					
6222	052010	001362	001116	000000					
6223	052016	001366	001164	001352	DT57	WORD	WASSRO, \$REG1, TESTNO, \$ERRPC, 0		
6224	052024	001116	000000						
6225	052030	001164	001166	001352	DT60	WORD	\$REG1, \$REG2, TESTNO, \$ERRPC, 0		
6226	052036	001116	000000						
6227	052042	001370	001172	001352	DT64	WORD	WASSR2, \$REG4, TESTNO, \$ERRPC, 0		
6228	052060	001116	000000						
6229	052064	001366	001370	001352	DT65	WORD	WASSRO, WASSR2, TESTNO, \$ERRPC, 0		
6230	052062	001116	000000						
6231	052066	001364	001370	001352	DT66	WORD	CORSR2, WASSR2, TESTNO, \$ERRPC, 0		

6232	052074	001116	000000						
6233	052100	001362	001366	001352	DT67	WORD	CORSRO, WASSRO, TESTNO, \$ERRPC, 0		
6234	052106	001116	000000						
6235									
6236	052112	000	000	000	DF 1	BYTE	0, 0, 0, 0, 0		
6237	052115	000	000						
6238	052117	000	000	000	DF 2	BYTE	0, 0, 0, 0, 0, 0, 0		
6239	052122	000	000	000					
6240	052125	000							
6241	052126	000	000	000	DF 3	BYTE	0, 0, 0, 0		
6242	052131	000							
6243	052132	000	000	000	DF 7	BYTE	0, 0, 0		
6244	052135	000	000	001	DF 10	BYTE	0, 0, 1, 0, 0		
6245	052140	000	000						
6246	052142	000	000	002	DF 11	BYTE	0, 0, 2, 0, 0		
6247	052145	000	000						
6248	052147	000	000	000	DF 12	BYTE	0, 0, 0, 2, 0, 0		
6249	052152	002	000	000					
6250	052155	000	000	000	DF 13	BYTE	0, 0, 0		
6251	052160	000	000	000	DF 16	BYTE	0, 0, 0, 0, 0, 0		
6252	052163	000	000	000					
6253	052166	003	000	000	DF 17.	BYTE	3, 0, 0, 0, 0		
6254	052171	000	000						
6255	052173	003	000	000	DF 20	BYTE	3, 0, 0, 0, 0, 0, 0, 0		
6256	052176	000	000	000					
6257	052201	000	000						
6258	052203	000	000	000	DF 21	BYTE	0, 0, 0, 0		
6259	052206	000							
6260	052207	000	000	000	DF 22	BYTE	0, 0, 0, 0, 0		
6261	052212	000	000						
6262	052214	000	000	000	DF 23	BYTE	0, 0, 0		
6263	052217	000	000	000	DF 24	BYTE	0, 0, 0, 0		
6264	052222	000							
6265	052223	000	000	000	DF 30.	BYTE	0, 0, 0, 0, 0, 0		
6266	052226	000	000	000					
6267	052231	000	000	000	DF 46	BYTE	0, 0, 0, 0		
6268	052234	000							
6269	052235	000	000		DF 50.	BYTE	0, 0		
6270	052237	000	000	000	DF 51	BYTE	0, 0, 0, 0		
6271	052242	000							
6272	052243	003	000	000	DF 52.	BYTE	3, 0, 0, 0, 0, 0, 0, 0		
6273	052246	000	000	000					
6274	052251	000	000						
6275	052253	000	000	000	DF 57	BYTE	0, 0, 0, 0		
6276	052256	000							
6277	052257	000	000	000	DF 60	BYTE	0, 0, 0, 0		
6278	052262	000							
6279	052263	000	000	000	DF 61	BYTE	0, 0, 0, 0		
6280	052266	000							
6281									
6282		000001				END			

DF61	052263	1247	1253	1260	1267	1273	1279	6279#
DF7	052132	953	6243#					
DM1	045620	915	5819#					
DM10	046050	957	5846#					
DM11	046150	964	1196	5857#				
DM12	046270	971	990	5871#				
DM13	046430	978	984	5888#				
DM16	046460	997	5892#					
DM17	046560	1004	5903#					
DM2	045670	921	1233	5826#				
DM20	046650	1011	5913#					
DM21	046776	1018	5928#					
DM22	047056	1025	5937#					
DM23	047155	1032	1128	5948#				
DM24	047205	1038	1044	5953#				
DM26	047245	1050	1056	5959#				
DM3	045760	927	933	939	945	1227	5836#	
DM30	047305	1062	5965#					
DM31	047365	1068	5974#					
DM32	047445	1074	5983#					
DM33	047525	1080	5992#					
DM34	047565	1086	5998#					
DM35	047645	1091	6007#					
DM36	047725	1097	1116	1239	6016#			
DM37	047765	1103	6022#					
DM40	050102	1110	6035#					
DM42	050142	1122	6041#					
DM44	050202	1134	6047#					
DM45	050314	1141	6060#					
DM46	050511	1151	1158	1183	6083#			
DM50	050566	1165	1190	1203	1221	6092#		
DM51	050606	1171	6095#					
DM52	050646	1177	6101#					
DM57	050764	1209	6115#					
DM60	051024	1215	6121#					
DM61	051064	1245	1251	1258	1265	6127#		
DM62	051122	1271	6132#					
DM63	051161	1277	6138#					
DM7	046020	951	5842#					
DISPLA	001142	782#	1386#	1394#	4415#	4444#		
DISPRE	000174	718#	1394					
DSMR	177570	549#	781	1385				
DT1	051220	916	6147#					
DT10	051276	959	6156#					
DT11	0513:2	966	1198	6158#				
DT12	051326	973	992	6160#				
DT13	051344	979	985	6163#				
DT16	051354	999	6165#					
DT17	051372	1006	6168#					
DT2	051234	922	1234	6149#				
DT20	051406	1013	6170#					
DT21	051430	1020	6173#					
DT22	051442	1027	6175#					
DT23	051456	1033	1129	6177#				
DT24	051466	1039	1045	6179#				
DT26	051500	1051	1057	6181#				

DT3	051254	928	934	940	946	1228	6152#
DT30	051512	1063	6183#				
DT31	051530	1069	6186#				
DT32	051546	1075	6189#				
DT33	051564	1081	6192#				
DT34	051576	1087	6194#				
DT35	051614	1092	6197#				
DT36	051632	1098	1117	6200#			
DT37	051644	1105	6202#				
DT40	051662	1111	6205#				
DT42	051674	1123	6207#				
DT44	051706	1136	6209#				
DT45	051724	1146	6212#				
DT46	051742	1153	1160	1185	6215#		
DT50	051754	1166	1191	1204	1222	6217#	
DT51	051762	1172	6218#				
DT52	051774	1179	6220#				
DT57	052016	1210	6223#				
DT60	052030	1216	6225#				
DT64	052042	1240	6227#				
DT65	052054	1246	1252	1259	1266	6229#	
DT66	052066	1272	6231#				
DT67	052100	1278	6233#				
DT7	051266	952	6154#				
EMTVEC =	000030	638#	1355#	1356#			
EM1	041616	514	5449#				
EM10	042175	956	5494#				
EM11	042241	963	5501#				
EM12	042301	970	5507#				
EM13	042350	977	5514#				
EM14	042405	983	5519#				
EM15	042440	989	5524#				
EM16	042514	996	5532#				
EM17	042556	1003	5538#				
EM2	041656	920	5455#				
EM20	042626	1010	5545#				
EM21	042700	1017	5552#				
EM22	042735	1024	5557#				
EM23	042774	1031	5563#				
EM24	043040	1037	5569#				
EM25	043100	1043	5575#				
EM26	043145	1049	5582#				
EM27	043215	1055	5589#				
EM3	041725	926	5462#				
EM30	043264	1061	5596#				
EM31	043336	1067	1090	1096	1238	5603#	
EM32	043403	1073	5610#				
EM33	043464	1079	5619#				
EM34	043547	1085	5628#				
EM37	043625	1102	5636#				
EM4	041764	932	5468#				
EM40	043672	1109	5643#				
EM41	043741	1115	5650#				
EM42	043774	1121	5655#				
EM43	044033	1127	5661#				
EM44	044100	1133	5668#				

\$ARLO	5338		
\$ACT1	18	5158	722
\$APTB	18	8178	
\$APTH	18	5158	732
\$APTY	18	5158	5041
\$ASTA	18		
\$CATC	18	5158	711
\$CHTA	18	5158	754
\$OBZD	12		
\$OBZ0	18	5158	5311
\$OIV	18		
\$EOP	18	5158	4285
\$ERRO	18	5158	4419
\$ERTY	18	5158	
\$MULT	18		
\$POME	18	5158	5395
\$RANO	18		
\$XODE	18		
\$RDOC	18		
\$PEAD	18	5158	4747
\$RZAZ	18		
\$SAVE	18	5158	5266
\$SBZD	18		
\$SBZ0	18		
\$SCOP	18	5158	4355
\$SIZE	18		
\$SUPR	18		
\$TRAP	18	5158	5350
\$TYPB	18	5158	5098
\$TYPD	18	5158	5199
\$TYPE	18	5158	4962
\$TYPO	18	5158	5122
\$4OCA	18		
1170	18		

ABS 052267 000

ERRORS DETECTED 0

CFKTHB.CFKTHB LST/CRF/SOL=CFKTHB SML.CFKTHB P11
RUN-TIME 22 30 2 SECONDS
RUN-TIME RATIO 266/55=4 8
CORE USED 39K (77 PAGES)