

KD11-Z

11/44 MEM MGMT PRT B
CKKTBA0

AH-F614A-MC
COPYRIGHT 1980
FICHE 1 OF 1

JAN 1980
digital
MADE IN USA

This microfiche card contains a grid of frames. The frames are arranged in approximately 15 rows and 10 columns. Each frame contains a small, high-contrast image of a document page, likely a management report or memorandum. The text within the frames is too small to be legible. The frames are separated by dark lines, and the overall card has a dark, textured background.



.REM @

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

IDENTIFICATION

PRODUCT CODE: AC-F612A-MC
PRODUCT NAME: CKKTBAO 11/44 MEM MGMT PRT B
DATE: OCTOBER, 1979
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: J. PETER BRADY

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) 1979 BY DIGITAL EQUIPMENT CORPORATION

44
45
46
47
48
49
50
51
52
53
54

PROGRAM HISTORY

<u>DATE</u>	<u>REVISION</u>	<u>REASON FOR REVISION</u>
OCTOBER, 1979	A	FIRST RELEASE

56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91

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

93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149

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.

PART A 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. PART B BEGINS BY TESTING THE ABORT AND STATUS SEGMENTS OF LOGIC. PART B THEN CHECKS THE SPECIAL ABORT SEQUENCES, THE MFPI/MTPI INSTRUCTIONS AND THE CSM INSTRUCTION.

1.2 REQUIREMENTS

A PDP 11/44 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 DIAGNOSTICS SHOULD BE RUN:

CKKAAA0 11/4 CPU/EIS
CKKABAO TRAPS

ALSO, THE MAIN MEMORY DIAGNOSTIC (CZMSD) SHOULD BE RUN TO SCAN AT LEAST THE FIRST 16K TO SEE THAT A PROGRAM CAN BE EXECUTED.

2.0 OPERATING INSTRUCTIONS

2.1 LOADING PROCEDURES

150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206

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. THE SWITCH REGISTER SHOULD BE SET ACCORDING TO SECTION 2.3 BEFORE THE PROGRAM IS STARTED. HOWEVER, IF DESIRED, 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 CONSOLE SWITCH REG. IS PRESENT BY LOADING '177777' INTO THE CONSOLE SWITCH REG. BEFORE STARTING THE PROGRAM.

2.3 CONTROL SWITCH SETTINGS

SWITCH	OCTAL VALUE	USE
SW15	100000	HALT ON ERROR THIS SWITCH WHEN SET WILL HALT THE PROCESSOR WHEN AN ERROR IS DETECTED AFTER THE ERROR MESSAGE HAS BEEN TYPED. PRESSING CONTINUE WILL RESUME TESTING (SEE SECTION 3.1 ABOUT LOADING THE SWITCH REG BEFORE CONTINUING).
SW14	040000	LOOP ON TEST THIS SWITCH WHEN SET WILL CAUSE THE PROGRAM TO LOOP ON THE CURRENT SUBTEST.
SW13	020000	INHIBIT ERROR TYPEOUTS THIS SWITCH WHEN SET WILL INHIBIT THE TYPING OF ERROR MESSAGES.
SW12	010000	INHIBIT TRACE TRAP THIS SWITCH WHEN SET WILL INHIBIT T-BIT TRAPPING WHICH NORMALLY TAKES PLACE DURING EVERY OTHER PASS STARTING WITH THE THIRD PASS.
SW10	002000	BELI ON ERROR

207
208
209
210
211
212
213
214
215
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

THIS SWITCH WHEN SET WILL RING THE CONSOLE TERMINAL BELL WHEN AN ERROR HAS BEEN DETECTED.

SW9	001000	LOOP ON ERROR	THIS SWITCH WHEN SET WILL CAUSE THE PROGRAM TO LOOP ON THE FIRST FAILURE WHICH IS ENCOUNTERED EVEN IF THE FAILURE IS INTERMITTANT
SW8	000400	LOOP ON TEST IN SWR<7:0>	THIS SWITCH WHEN SET WILL CAUSE THE PROGRAM TO LOOP ON THE TEST WHOSE TEST NUMBER IS SET IN BITS 7-0 OF THE SWITCH REG.

2.4 LOADING THE SWITCH REGISTER

THE CONSOLE SWITCH REGISTER PROVIDED IS LOADED DIRECTLY FROM THE CONSOLE BY TYPING A CONTROL P (^P), THEN WHEN THE CONSOLE PROMPT IS RECEIVED, TYPE 'D SW XXXXXX', WHERE 'XXXXXX' IS THE INTENDED VALUE OF THE SWITCH REGISTER. THE VALUE OF THE CONSOLE 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 DIGITS TYPED SO FAR TO BE IGNORED.
- ^C 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

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

----- EXECUTION OF A COUPLE 'RESET' INSTRUCTIONS
WITHIN THE PROGRAM.

2.5 EXECUTION TIMES

THE RUN TIME FOR A SINGLE PASS WITH TRACE TRAPPING
ENABLED IS APPROXIMATELY 5 SECONDS WITH CACHE.

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
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.

321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377

3.3 SAMPLE ERROR REPORT

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

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

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

4.0 MISCELLANEOUS INFORMATION

4.1 ACT/APT/XXDP COMPATABILITY

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

4.2 END-OF-PASS MESSAGE

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

END OF PASS #2 TOTAL ERRORS SINCE LAST REPORT 0

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).

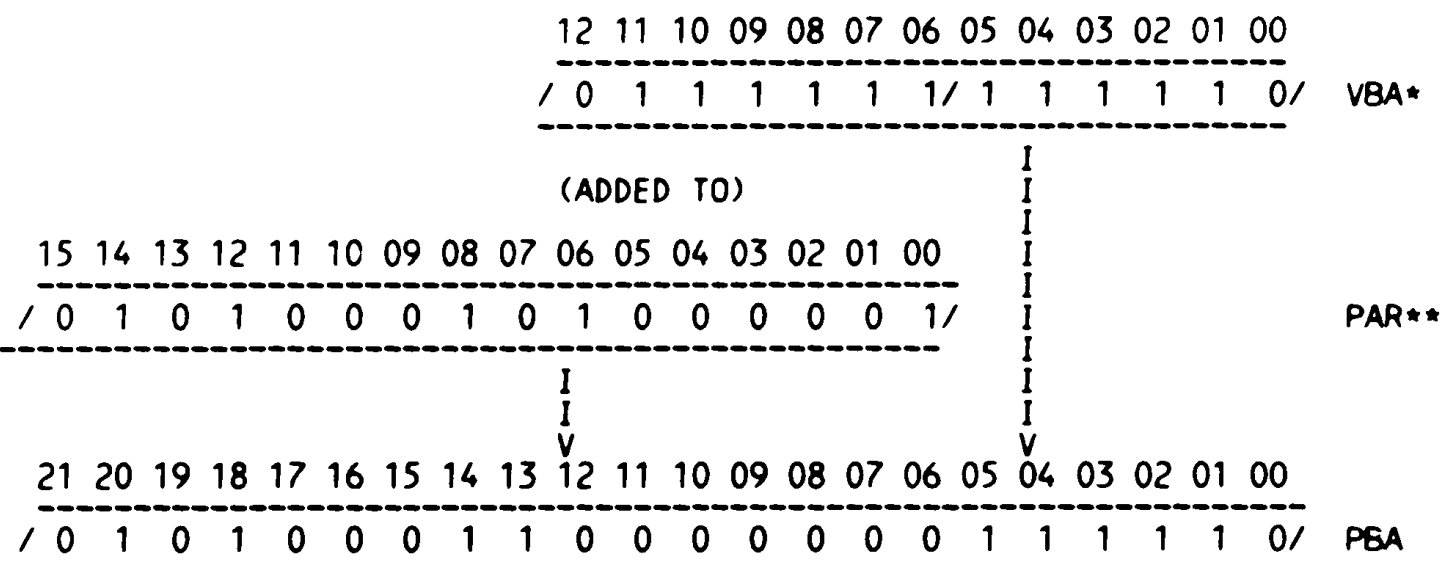
4.4 POWER FAILURE HANDLING

378
379
380
381
382
383
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

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.

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 BITS <15:14> SELECTS THE USER (=11), SUPERVISOR (=01) OR KERNEL (=00) SET OF PAR'S/PDR'S

5.0 PROGRAM DESCRIPTION

5.1 SUBROUTINES USED BY THIS PROGRAM

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

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

435
436
437
438
439
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

NOTE ALSO THAT THE MACRO LIBRARY USED TO ASSEMBLE THIS PROGRAM HAS OTHER SPECIAL ROUTINES APPENDED TO THE SYSMAC MACRO PACKAGE; THIS LIBRARY MUST BE USED TO ASSEMBLE EITHER PART A OR PART B CORRECTLY.

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. THIS 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.

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'.

@

522
523

```
.TITLE CKKTBAO 11/44 MEM MGMT PRT B
.*COPYRIGHT (C) 1979
.*DIGITAL EQUIPMENT CORP.
.*MAYNARD, MASS. 01754
.*
.*
.*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
.*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
.*
```

524

```
.SBTTL OPERATIONAL SWITCH SETTINGS
.*
.*      SWITCH          USE
.*      -----
.*      15             HALT ON ERROR
.*      14             LOOP ON TEST
.*      13             INHIBIT ERROR TYPEOUTS
.*      12             INHIBIT TRACE TRAP
.*      10             BELL ON ERROR
.*      9              LOOP ON ERROR
.*      8              LOOP ON TEST IN SWR<7:0>
```

525

```
.SBTTL BASIC DEFINITIONS
.*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100
        ERROR=EMT
        SCOPE=IOT

.*MISCELLANEOUS DEFINITIONS
000011 HT= 11          ;;CODE FOR HORIZONTAL TAB
000012 LF= 12          ;;CODE FOR LINE FEED
000015 CR= 15          ;;CODE FOR CARRIAGE RETURN
000200 CRLF= 200       ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776 PS= 177776     ;;PROCESSOR STATUS WORD
177776 PSW=PS
177774 STKLMT= 177774 ;;STACK LIMIT REGISTER
177772 PIRQ= 177772   ;;PROGRAM INTERRUPT REQUEST REGISTER
177570 DSWR= 177570  ;;HARDWARE SWITCH REGISTER
177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER

.*GENERAL PURPOSE REGISTER DEFINITIONS
000000 R0= %0          ;;GENERAL REGISTER
000001 R1= %1          ;;GENERAL REGISTER
000002 R2= %2          ;;GENERAL REGISTER
000003 R3= %3          ;;GENERAL REGISTER
000004 R4= %4          ;;GENERAL REGISTER
000005 R5= %5          ;;GENERAL REGISTER
000006 R6= %6          ;;GENERAL REGISTER
000007 R7= %7          ;;GENERAL REGISTER
000006 SP= %6          ;;STACK POINTER
000007 PC= %7          ;;PROGRAM COUNTER

.*PRIORITY LEVEL DEFINITIONS
000000 PR0= 0          ;;PRIORITY LEVEL 0
000040 PR1= 40         ;;PRIORITY LEVEL 1
000100 PR2= 100        ;;PRIORITY LEVEL 2
000140 PR3= 140        ;;PRIORITY LEVEL 3
000200 PR4= 200        ;;PRIORITY LEVEL 4
000240 PR5= 240        ;;PRIORITY LEVEL 5
000300 PR6= 300        ;;PRIORITY LEVEL 6
000340 PR7= 340        ;;PRIORITY LEVEL 7
```

```

100000
040000
020000
010000
004000
002000
001000
000400
000200
000100
000040
000020
000010
000004
000002
000001
001000
000400
000200
000100
000040
000020
000010
000004
000002
000001
100000
040000
020000
010000
004000
002000
001000
000400
000200
000100
000040
000020
000010
000004
000002
000001
001000
000400
000200
000100
000040
000020
000010
000004
000002
000001
000004
000010

; *SWITCH REGISTER' SWITCH DEFINITIONS
SW15= 100000
SW14= 40000
SW13= 20000
SW12= 10000
SW11= 4000
SW10= 2000
SW09= 1000
SW08= 400
SW07= 200
SW06= 100
SW05= 40
SW04= 20
SW03= 10
SW02= 4
SW01= 2
SW00= 1
SW9=SW09
SW8=SW08
SW7=SW07
SW6=SW06
SW5=SW05
SW4=SW04
SW3=SW03
SW2=SW02
SW1=SW01
SW0=SW00

; *DATA BIT DEFINITIONS (BIT00 TO BIT15)
BIT15= 100000
BIT14= 40000
BIT13= 20000
BIT12= 10000
BIT11= 4000
BIT10= 2000
BIT09= 1000
BIT08= 400
BIT07= 200
BIT06= 100
BIT05= 40
BIT04= 20
BIT03= 10
BIT02= 4
BIT01= 2
BIT00= 1
BIT9=BIT09
BIT8=BIT08
BIT7=BIT07
BIT6=BIT06
BIT5=BIT05
BIT4=BIT04
BIT3=BIT03
BIT2=BIT02
BIT1=BIT01
BIT0=BIT00

; *BASIC 'CPU' TRAP VECTOR ADDRESSES
ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
```

526

```
000014 TBITVEC=14 ;:'T' BIT
000014 TRTVEC= 14 ;:TRACE TRAP
000014 BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)
000020 IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024 PWRVEC= 24 ;:POWER FAIL
000030 EMTVEC= 30 ;:EMULATOR TRAP (EMT) **ERROR**
000034 TRAPVEC=34 ;:'TRAP' TRAP
000060 TKVEC= 60 ;:TTY KEYBOARD VECTOR
000064 TPVEC= 64 ;:TTY PRINTER VECTOR
000240 PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR
.SBTTL MEMORY MANAGEMENT DEFINITIONS
;*KT11 VECTOR ADDRESS
000250 MMVEC= 250
;*KT11 STATUS REGISTER ADDRESSES
177572 SR0= 177572
177574 SR1= 177574
177576 SR2= 177576
172516 SR3= 172516
;*USER 'I' PAGE DESCRIPTOR REGISTERS
177600 UIPDR0= 177600
177602 UIPDR1= 177602
177604 UIPDR2= 177604
177606 UIPDR3= 177606
177610 UIPDR4= 177610
177612 UIPDR5= 177612
177614 UIPDR6= 177614
177616 UIPDR7= 177616
;*USER 'D' PAGE DESCRIPTOR REGISTERS
177620 UDPDR0= 177620
177622 UDPDR1= 177622
177624 UDPDR2= 177624
177626 UDPDR3= 177626
177630 UDPDR4= 177630
177632 UDPDR5= 177632
177634 UDPDR6= 177634
177636 UDPDR7= 177636
;*USER 'I' PAGE ADDRESS REGISTERS
177640 UIPAR0= 177640
177642 UIPAR1= 177642
177644 UIPAR2= 177644
177646 UIPAR3= 177646
177650 UIPAR4= 177650
177652 UIPAR5= 177652
177654 UIPAR6= 177654
177656 UIPAR7= 177656
;*USER 'D' PAGE ADDRESS REGISTERS
177660 UDPAR0= 177660
177662 UDPAR1= 177662
177664 UDPAR2= 177664
177666 UDPAR3= 177666
177670 UDPAR4= 177670
177672 UDPAR5= 177672
177674 UDPAR6= 177674
177676 UDPAR7= 177676
;*SUPERVISOR 'I' PAGE DESCRIPTOR REGISTERS
172200 SIPDR0= 172200
172202 SIPDR1= 172202
```

172204	SIPDR2= 172204
172206	SIPDR3= 172206
172210	SIPDR4= 172210
172212	SIPDR5= 172212
172214	SIPDR6= 172214
172216	SIPDR7= 172216
	;*SUPERVISOR 'D' PAGE DESCRIPTOR REGISTERS
172220	SDPDR0= 172220
172222	SDPDR1= 172222
172224	SDPDR2= 172224
172226	SDPDR3= 172226
172230	SDPDR4= 172230
172232	SDPDR5= 172232
172234	SDPDR6= 172234
172236	SDPDR7= 172236
	;*SUPERVISOR 'I' PAGE ADDRESS REGISTERS
172240	SIPAR0= 172240
172242	SIPAR1= 172242
172244	SIPAR2= 172244
172246	SIPAR3= 172246
172250	SIPAR4= 172250
172252	SIPAR5= 172252
172254	SIPAR6= 172254
172256	SIPAR7= 172256
	;*SUPERVISOR 'D' PAGE ADDRESS REGISTERS
172260	SDPAR0= 172260
172262	SDPAR1= 172262
172264	SDPAR2= 172264
172266	SDPAR3= 172266
172270	SDPAR4= 172270
172272	SDPAR5= 172272
172274	SDPAR6= 172274
172276	SDPAR7= 172276
	;*KERNEL 'I' PAGE DESCRIPTOR REGISTERS
172300	KIPDR0= 172300
172302	KIPDR1= 172302
172304	KIPDR2= 172304
172306	KIPDR3= 172306
172310	KIPDR4= 172310
172312	KIPDR5= 172312
172314	KIPDR6= 172314
172316	KIPDR7= 172316
	;*KERNEL 'D' PAGE DESCRIPTOR REGISTERS
172320	KDPDR0= 172320
172322	KDPDR1= 172322
172324	KDPDR2= 172324
172326	KDPDR3= 172326
172330	KDPDR4= 172330
172332	KDPDR5= 172332
172334	KDPDR6= 172334
172336	KDPDR7= 172336
	;*KERNEL 'I' PAGE ADDRESS REGISTERS
172340	KIPAR0= 172340
172342	KIPAR1= 172342
172344	KIPAR2= 172344
172346	KIPAR3= 172346
172350	KIPAR4= 172350

```
172352 KIPAR5= 172352
172354 KIPAR6= 172354
172356 KIPAR7= 172356
;*KERNEL 'D' PAGE ADDRESS REGISTERS
172360 KDPAR0= 172360
172362 KDPAR1= 172362
172364 KDPAR2= 172364
172366 KDPAR3= 172366
172370 KDPAR4= 172370
172372 KDPAR5= 172372
172374 KDPAR6= 172374
172376 KDPAR7= 172376
;*ADDITIONAL DEFINITIONS
;*
527 MMR0=SR0
528 MMR1=SR1
529 MMR2=SR2
530 MMR3=SR3
531 KSP=SP
532 SSP=SP
533 USP=SP
534 TBIT=BIT4
535 WBIT=BIT6
536 CPUERR=177766
537 KERSTK=STACK
538 SUPSTK=STACK-200
539 USESTK=STACK-300
540
541
542
572
```



```

575          000000          .SBTTL TRAP CATCHER
                                =0
                                ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A '+2,HALT'
                                ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
                                =174
000174      000174          DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
000176      000000          SWREG:   .WORD 0          ;;SOFTWARE SWITCH REGISTER
                                .SBTTL STARTING ADDRESS(ES)
                                JMP     @*START ;;JUMP TO STARTING ADDRESS OF PROGRAM
576          000200      000137      020000          .SBTTL ACT11 HOOKS
                                ;*****
                                ;HOOKS REQUIRED BY ACT11
                                $SVPC=.          ;SAVE PC
                                =46
                                $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
000046      036056          =52
000052      000000          .WORD 0          ;;2)SET LOC.52 TO ZERO
                                =.$SVPC          ;; RESTORE PC
577          000204          .SBTTL APT PARAMETER BLOCK
                                ;*****
                                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                ;*****
                                000204          .$X=.          ;;SAVE CURRENT LOCATION
                                000024          =24          ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000024      000200          200          ;;FOR APT START UP
                                000044          =44          ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044      000204          $APTHDR ;;POINT TO APT HEADER BLOCK
                                000204          -. $X          ;;RESET LOCATION COUNTER
                                ;*****
                                ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                ;INTERFACE SPEC.
000204          $APTHD:
000204      000000          $HIBTS: .WORD 0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
000206      001224          $MBADR: .WORD $MAIL      ;;ADDRESS OF APT MAILBOX (BITS 0-15)
000210      000002          $STMT:  .WORD 2          ;;RUN TIM OF LONGEST TEST
000212      000005          $PASTM: .WORD 5          ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
000214      000005          $UNITM: .WORD 5          ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
000216      000014          .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)

```

:SBTTL APT MAILBOX-ETABLE

001224
001224 000000
001226 000000
001230 000000
001232 000000
001234 000000
001236 000000
001240 000000
001242 000000
001244
001244 000
001245 000
001246 000000
001250 000000
001252 000000

.EVEN
\$MAIL: ::: APT MAILBOX
\$MSGTY: .WORD AMSGTY ::: MESSAGE TYPE CODE
\$FATAL: .WORD AFATAL ::: FATAL ERROR NUMBER
\$TESTN: .WORD ATESTN ::: TEST NUMBER
\$PASS: .WORD APASS ::: PASS COUNT
\$DEVCT: .WORD ADEVCT ::: DEVICE COUNT
\$UNIT: .WORD AUNIT ::: I/O UNIT NUMBER
\$MSGAD: .WORD AMSGAD ::: MESSAGE ADDRESS
\$MSGLG: .WORD AMSLG ::: MESSAGE LENGTH
\$ETABLE: ::: APT ENVIRONMENT TABLE
\$ENV: .BYTE AENV ::: ENVIRONMENT BYTE
\$ENVM: .BYTE AENVM ::: ENVIRONMENT MODE BITS
\$SWREG: .WORD ASWREG ::: APT SWITCH REGISTER
\$USWR: .WORD AUSWR ::: USER SWITCHES
\$CPUOP: .WORD ACPUOP ::: CPU TYPE, OPTIONS

BIT 15-11=CPU TYPE
 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
 11/70=06,PDQ=07,Q=10
BIT 10=REAL TIME CLOCK
BIT 9=FLOATING POINT PROCESSOR
BIT 8=MEMORY MANAGEMENT

001254

\$ETEND:
.MEXIT

001254 000000
001256 000000
001260 000000
001262 000000
001264 000000
001266 000000
001270 000000
001272 000000
001274 000000
001276 000000
001300 000000
001302 000000
001304 000000
001306 000200
001310 000000
001312 136 103
001315 012 000

TESTNO: .WORD 0 ::: HOLDS TEST NUMBER FOR TYPEOUTS
WASR6: .WORD 0 ::: USED TO STORE THE STACK POINTER AFTER A TRAP
TRAPP: .WORD 0 ::: USED TO STORE THE PC OF A TRAP OR ABORT
TRAPPS: .WORD 0 ::: USED TO STORE THE PS OF A TRAP OR ABORT
WASSR0: .WORD 0 ::: USED TO STORE CONTENTS OF SR0
WASSR1: .WORD 0 ::: USED TO STORE CONTENTS OF SR1
WASSR2: .WORD 0 ::: USED TO STORE CONTENTS OF SR2
WASSR3: .WORD 0 ::: USED TO STORE CONTENTS OF SR3
TBITPS: .WORD 0 ::: SAVES THE PSW THAT MAY HAVE ITS T-BIT ON
VIRT1: .WORD 0 ::: HOLDS VIRTUAL ADDRESS TO BE CONVERTED
PBALO: .WORD 0 ::: HOLDS BITS <15:00> OF PHYSICAL ADDRESS
PBAHI: .WORD 0 ::: HOLDS BITS <21:16> OF PHYSICAL ADDRESS
BADPC: .WORD 0 ::: HOLDS PC FROM ABORT OR TRAP
\$MXCNT: .WORD 200 ::: HOLD MAX. NUMBER OF LOOP ITERATIONS
\$TBIT: .WORD 0 ::: 'T' BIT STATE INDICATOR
\$CNTLC: .ASCIIZ /^C/<15><12> ::: CONTROL C

.EVEN

```
.SBTTL ERROR POINTER TABLE
:*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
:*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
:*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
:*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
:*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
:*      EM      ;;POINTS TO THE ERROR MESSAGE
:*      DH      ;;POINTS TO THE DATA HEADER
:*      DT      ;;POINTS TO THE DATA
:*      DF      ;;POINTS TO THE DATA FORMAT
$ERRTB:
```

001320

```
579
580
581 001320 042264      EM1      :UNEXPECTED CPU TRAP TO LOC. 004
582 001322 045026      DH1      :OLD PC OLD PSW R6 WAS CPUERR TESTNO ERRORPC
583 001324 047336      DT1      :TRAPPC,TRAPPS,WASR6,CPUERR,TESTNO,$ERRPC,0
584 001326 050051      DF12     :0,0,0,0,0,0
585
586
587 001330 042324      EM2      :UNEXPECTED MEM. MGMT. TRAP TO LOC. 250
588 001332 045106      DH2      :OLD PC OLD PSW R6 WAS SR0 SR2 TESTNO ERRORPC
589 001334 047354      DT2      :TRAPPC,TRAPPS,WASR6,WASSR0,WASSR2,TESTNO,$ERRPC,0
590 001336 050033      DF2      :0,0,0,0,0,0
591
592
593 001340 042373      EM10     :MEMORY MGMT. ACCESS ABORT DID NOT OCCUR
594 001342 045176      DH10     :PDR 4 PSW TESTNO ERRORPC
595 001344 047374      DT10     :$REG2,$TMP0,TESTNO,$ERRPC,0
596 001346 050042      DF3      :0,0,0,0
597
598
599 001350 042443      EM11     :ACCESS ERROR DID NOT ABORT INSTRUCTION
600 001352 045176      DH10     :PDR 4 PSW TESTNO ERRORPC
601 001354 047374      DT10     :$REG2,$TMP0,TESTNO,$ERRPC,0
602 001356 050042      DF3      :0,0,0,0
603
604
605 001360 042512      EM12     :SR0 DID NOT REPORT ACCESS ERROR CORRECTLY
606 001362 045236      DH12     :SR0 WAS EXPECTD PDR 4 PSW TESTNO ERRORPC
607 001364 047406      DT12     :WASSR0,$REG3,$REG2,$TMP0,TESTNO,$ERRPC,0
608 001366 050051      DF12     :0,0,0,0,0,0
609
610
611 001370 042564      EM13     :SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR.
612 001372 045316      DH13     :SR2 WAS EXPECTD PDR 4 PSW TESTNO ERRORPC
613 001374 047424      DT13     :WASSR2,$REG4,$REG2,$TMP0,TESTNO,$ERRPC,0
614 001376 050051      DF12     :0,0,0,0,0,0
615
616
617 001400 042635      EM14     :PAGE LGTH. ABORT OCCURRED WHEN IT SHOULDN'T HAVE
618 001402 045376      DH14     :V.B.A. KIPDR4 SR0 WAS SR2 WAS TESTNO ERRORPC
619 001404 047442      DT14     :$REG0,$REG4,WASSR0,WASSR2,TESTNO,$ERRPC,0
620 001406 050051      DF12     :0,0,0,0,0,0
621
622
623 001410 042716      EM15     :PAGE LGTH. ABORT DID NOT OCCUR WHEN IT SHOULD HAVE
624 001412 045456      DH15     :V.B.A. KIPDR4 TESTNO ERRORPC
```

625	001414	047460	DT15	:\$REG0,\$REG4,TESTNO,\$ERRPC,0
626	001416	050042	DF3	:0,0,0,0
627				
628			;*ITEM 11	
629	001420	043001	EM16	:SR0 DID NOT REPORT PAGE LGTH. ABORT CORRECTLY
630	001422	045516	DH16	:V.B.A. KIPDR4 SR0 WAS EXPECTD TESTNO ERRORPC
631	001424	047472	DT16	:\$REG0,\$REG4,WASSR0,\$REG2,TESTNO,\$ERRPC,0
632	001426	050051	DF12	:0,0,0,0,0,0
633				
634			;*ITEM 12	
635	001430	042564	EM13	:SR2 DID NOT LOCKUP CORRECT VIRUAL ADDR.
636	001432	045576	DH17	:V.B.A. KIPDR4 SR2 WAS EXPECTD TESTNO ERRORPC
637	001434	047510	DT17	:\$REG0,\$REG4,WASSR2,\$REG3,TESTNO,\$ERRPC,0
638	001436	050051	DF12	:0,0,0,0,0,0
639				
640			;*ITEM 13	
641	001440	042564	EM13	:SR2 DID NOT LOCKUP CORRECT VIRUAL ADDR.
642	001442	045656	DH20	:SR2 WAS EXPECTD TESTNO ERRORPC
643	001444	047526	DT20	:WASSR2,\$REG1,TESTNO,\$ERRPC,0
644	001446	050042	DF3	:0,0,0,0
645				
646			;*ITEM 14	
647	001450	043057	EM21	:SR0 OR SR2 CHANGED BY A SECOND ABORT
648	001452	045716	DH21	:FIRST ABORT SECOND ABORT
649				:SR0 WAS SR2 WAS SR0 WAS SR2 WAS TESTNO ERRORPC
650	001454	047540	DT21	:\$TMP0,\$TMP2,WASSR0,WASSR2,TESTNO,\$ERRPC,0
651	001456	050051	DF12	:0,0,0,0,0,0
652				
653			;*ITEM 15	
654	001460	043124	EM22	:SR0 OR SR2 WAS NOT 'RESET' BY A RESET
655	001462	046033	DH22	:SR0 WAS SR2 WAS TESTNO ERRORPC
656	001464	047556	DT22	:WASSR0,WASSR2,TESTNO,\$ERRPC,0
657	001466	050042	DF3	:0,0,0,0
658				
659			;*ITEM 16	
660	001470	043173	EM23	:SR2 NOT TRACKING CORRECTLY
661	001472	045656	DH20	:SR2 WAS EXPECTD TESTNO ERRORPC
662	001474	047526	DT20	:WASSR2,\$REG1,TESTNO,\$ERRPC,0
663	001476	050042	DF3	:0,0,0,0
664				
665			;*ITEM 17	
666	001500	043226	EM24	:DID NOT TRAP THRU KERNEL SPACE
667	001502	046073	DH24	:PSW WAS R6 WAS TESTNO ERRORPC
668	001504	047570	DT24	:\$REG1,\$REG2,TESTNO,\$ERRPC,0
669	001506	050042	DF3	:0,0,0,0
670				
671			;*ITEM 20	
672	001510	043265	EM25	:KT ERROR SERVICED ON ODD ADDR. ERROR
673	001512	045656	DH20	:PDR TESTNO ERRORPC
674	001514	047526	DT20	:\$REG5,TESTNO,\$ERRPC,0
675	001516	050046	DF5	:0,0,0
676				
677			;*ITEM 21	
678	001520	043332	EM26	:SR0 OR SR2 CHANGED BY ODD ADDR. ERROR
679	001522	046133	DH26	:EXPECTED RECEIVED
680				:SR0 SR2 SR0 WAS SR2 WAS TESTNO ERRORPC
681	001524	047602	DT26	:\$REG0,\$REG1,WASSR0,WASSR2,TESTNO,\$ERRPC,0

682	001526	050051	DF12	:0,0,0,0,0,0
683				
684			;*ITEM 22	
685	001530	043400	EM27	:ERROR DURING 'DOUBLE ERROR' (KT & ODD ADDR.)
686	001532	046245	DH27	:EXPECTED:
687				:PSW PC SRO SR2
688				:170017 (3\$+4) 020147 (3\$)
689				:RECEIVED
690				:PSW PC SRO SR2 TESTNO ERRORPC
691	001534	047620	DT27	:\$REG1,\$REG3,WASSRO,WASSR2,TESTNO,\$ERRPC,0
692	001536	050051	DF12	:0,0,0,0,0,0
693				
694			;*ITEM 23	
695	001540	043455	EM30	:MFPI INSTRUCTION PUSHED WRONG DATA
696	001542	046442	DH30	:DATA DATA
697				:EXPECTD RECEIVD TESTNO ERRORPC
698	001544	047636	DT30	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
699	001546	050042	DF3	:0,0,0,0
700				
701			;*ITEM 24	
702	001550	043520	EM31	:MTPI INSTRUCTION LOADED WRONG DATA
703	001552	046442	DH30	:DATA DATA
704				:EXPECTD RECEIVD TESTNO ERRORPC
705	001554	047636	DT30	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
706	001556	050042	DF3	:0,0,0,0
707				
708			;*ITEM 25	
709	001560	043563	EM32	:STACK NOT PUSHED BY MFPI-MTPI
710	001562	046517	DH32	:TESTNO ERRORPC
711	001564	047650	DT32	:TESTNO,\$ERRPC,0
712	001566	050057	DF32	:0,0
713				
714			;*ITEM 26	
715	001570	043621	EM33	:KERNEL PAGE ACCESSED INSTEAD OF USER: MFPI-MTPI
716	001572	046537	DH33	:SRO WAS SR2 WAS TESTNO ERRORPC
717	001574	047556	DT22	:WASSRO,WASSR2,TESTNO,\$ERRPC,0
718	001576	050042	DF3	:0,0,0,0
719				
720			;*ITEM 27	
721	001600	043677	EM34	:M.M. ABORT IN KERNAL D-SPACE HAD WRONG CONDITION
722	001602	046577	DH34	:(MMR0) (MMR1) (MMR2) TESTNO ERRORPC EXPECTING 020031
723	001604	047656	DT34	:\$REG1,\$REG2,\$REG3,TESTNO,\$ERRPC,0
724	001606	050026	DF1	:0,0,0,0,0
725				
726			;*ITEM 30	
727	001610	043760	EM35	:ILLEGAL MODE 10 NOT ABORTED
728	001612	046517	DH32	:TESTNO ERRORPC
729	001614	047650	DT32	:TESTNO,\$ERRPC,0
730	001616	050057	DF32	:0,0
731				
732			;*ITEM 31	
733	001620	044014	EM36	:SRO DID NOT REPORT ILLEGAL MODE 10 CORRECTLY
734	001622	046670	DH36	:SRO WAS EXPECTD TESTNO ERRORPC
735	001624	047672	DT36	:WASSRO,\$REG1,TESTNO,\$ERRPC,0
736	001626	050042	DF3	:0,0,0,0
737				
738			;*ITEM 32	

739	001630	044071	EM37	:PSW CHANGED BY AN RTI IN USER MODE
740	001632	046730	DH37	:PSW WAS EXPECTD TESTNO ERRORPC
741	001634	047570	DT24	:\$REG1,\$REG2,TESTNO,\$ERRPC,0
742	001636	050042	DF3	:0,0,0,0
743				
744			;*ITEM 33	
745	001640	044134	EM40	:ABORT IN KERNAL D-SPACE PICKED UP VECTOR FROM I-SPACE
746	001642	046770	DH40	:(PSW) TESTNO ERRORPC EXPECTING XXX340
747	001644	047704	DT40	:\$REG0,TESTNO,\$ERRPC,0
748	001646	050046	DF5	:0,0,0
749				
750			;*ITEM 34	
751	001650	044221	EM41	:D SPACE ENABLE CIRCUITRY HAS FAILED
752	001652	047041	DH41	:ERROR AUTOI/D VIRTUAL
753				:REGISTR REGISTR ADDRESS TESTNO PC AT ABORT
754	001654	047714	DT41	:WASSR0,WASSR1,WASSR2,TESTNO,BADPC,0
755	001656	050026	DF1	:0,0,0,0,0
756				
757			;*ITEM 35	
758	001660	044265	EM42	:INCORRECT STORE BY MTP INSTRUCTION
759	001662	047145	DH42	:GDDATA STORED TESTNO ERRORPC
760	001664	047730	DT42	:\$REG3,\$REG4,TESTNO,\$ERRPC,0
761	001666	050042	DF3	:0,0,0,0
762				
763			;*ITEM 36	
764	001670	044330	EM43	:TRIED TO REFERENCE NON-RESIDENT PAGE
765	001672	047205	DH43	:(MMR0) (MMR1) (MMR2) TESTNO ERRORPC
766	001674	047742	DT43	:\$REG0,\$REG1,\$REG2,TESTNO,\$ERRPC,0
767	001676	050026	DF1	
768				
769			;*ITEM 37	
770	001700	044375	EM44	:WRONG DATA FETCHED BY MFP INSTRUCTION
771	001702	046442	DH30	:DATA DATA
772				:EXPECTD RECEIVD TESTNO ERRORPC
773	001704	047636	DT30	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
774	001706	050042	DF3	:0,0,0,0
775				
776			;*ITEM 40	
777	001710	044330	EM43	:TRIED TO REFERENCE NON-RESIDENT PAGE
778	001712	047205	DH43	:(MMR0) (MMR1) (MMR2) TESTNO ERRORPC
779	001714	047756	DT45	:WASSR0,WASSR1,WASSR2,TESTNO,\$ERRPC,0
780	001716	050026	DF1	:0,0,0,0,0
781				
782			;*ITEM 41	
783	001720	044443	EM45	:ILLEGAL CSM DID NOT TRAP TO 10
784	001722	046517	DH32	:TESTNO ERRORPC
785	001724	047650	DT32	:TESTNO,\$ERRPC,0
786	001726	050033	DF2	:0,0
787				
788			;*ITEM 42	
789	001730	044502	EM46	:CSM DID NOT ENTER SUPERVISOR MODE
790	001732	047255	DH44	:(PSW) TESTNO ERRORPC
791	001734	047772	DT46	:WASR6,TESTNO,\$ERRPC,0
792	001736	050033	DF2	:0,0,0
793				
794			;*ITEM 43	
795	001740	044544	EM47	:CSM SET UP WRONG PREVIOUS MODE

796	001742	047255	DH44	:(PSW) TESTNO ERRORPC
797	001744	047772	DT46	:WASR6,TESTNO,\$ERRPC,0
798	001746	050033	DF2	:0,0,0
799				
800			;*ITEM 44	
801	001750	044603	EM50	:CSM SET UP STACK WRONG
802	001752	046442	DH30	:DATA DATA
803				:EXPECTD RECEIVD TESTNO ERRORPC
804	001754	047636	DT30	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
805	001756	050042	DF3	:0,0,0,0
806				
807			;*ITEM 45	
808	001760	044632	EM51	:CSM PUSHED INCORRECT ARGUMENT
809	001762	046442	DH30	:DATA DATA
810				:EXPECTD RECEIVD TESTNO ERRORPC
811	001764	050002	DT47	:\$REG5,WASR6,TESTNO,\$ERRPC,0
812	001766	050042	DF3	:0,0,0,0
813				
814			;*ITEM 46	
815	001770	044670	EM52	:CSM PUSHED WRONG PC
816	001772	046442	DH30	:DATA DATA
817				:EXPECTD RECEIVD TESTNO ERRORPC
818	001774	050014	DT50	:\$TMP0,WASR6,TESTNO,\$ERRPC,0
819	001776	050042	DF3	:0,0,0,0
820				
821			;*ITEM 47	
822	002000	044714	EM53	:CSM DID NOT CLEAR OLD PSW BITS <3:0>
823	002002	047255	DH44	:OLDPSW TESTNO ERRORPC
824	002004	047772	DT46	:WASR6,TESTNO,\$ERRPC,0
825	002006	050033	DF2	:0,0,0
826				
827			;*ITEM 50	
828	002010	044761	EM54	:CSM ACCESSED WRONG SUPERVISOR SPACE
829	002012	046517	DH32	:TESTNO ERRORPC
830	002014	047650	DT32	:TESTNO,\$ERRPC,0
831	002016	050033	DF2	:0,0


```

833      .SBTTL ***** SUBROUTINES UNIQUE TO THIS PROGRAM *****
834
835      .SBTTL INITIALIZE ALL PAR'S AND PDR'S
836      :*****
837      :
838      :   THIS ROUTINE WILL INITIALIZE ALL KERNAL, SUPERVISOR, AND
839      :   USER PAR'S AND PDR'S TO THEIR USUAL INITIAL VALUE
840      :
841      :*****
842      APRINIT:
843      002020 012700 077406      MOV      #77406,R0      ;MAKE ALL PDR'S 4K, READ/WRITE, UPWARDS
844                                     ;EXPANDING, 200 BLOCKS
845      002024 012702 172300      MOV      #KIPDR0,R2    ;LOAD THE ADDRESS OF THE FIRST KERNAL PDR
846      002030 012701 000020      1$:     MOV      #20,R1    ;LOAD R1 WITH 16
847      002034 010022                2$:     MOV      R0,(R2)+    ;LOAD EACH PDR IN TURN
848      002036 077102                SOB      R1,2$         ;LOOP UNTIL ALL ARE LOADED
849      002040 020227 172340      CMP      R2,#KDPDR7+2 ;HAVE WE LOADED ALL KERNAL PDR'S
850      002044 001003                BNE      3$           ;BRANCH IF KERNAL & SUPER HAVE BEEN LOADED
851      002046 012702 172200      MOV      #SIPDR0,R2   ;LOAD ALL SUPERVISOR PDR'S
852      002052 000766                BR       1$           ;BRANCH TO LOOP
853      002054 020227 172240      3$:     CMP      R2,#SDPDR7+2 ;HAVE USER PDR'S BEEN DONE
854      002060 001003                BNE      4$           ;BRANCH IF THEY HAVE
855      002062 012702 177600      MOV      #UIPDR0,R2   ;LOAD ALL USER PDR'S
856      002066 000760                BR       1$           ;BRANCH TO LOOP
857      002070 012701 172340      4$:     MOV      #KIPAR0,R1  ;LOAD R1 WITH ADDRESS OF KIPAR0
858      002074 012702 172360      MOV      #KDPAR0,R2  ;LOAD R2 WITH ADDRESS OF KDPAR0
859      002100 012703 000007      5$:     MOV      #7,R3        ;LOAD LOOP COUNTER WITH 7
860      002104 005000                CLR      R0           ;CLEAR PAR VALUE REGISTER
861      002106 010021                6$:     MOV      R0,(R1)+    ;LOAD AN I-SPACE PAR
862      002110 010022                MOV      R0,(R2)+    ;LOAD A D-SPACE PAR
863      002112 062700 000200      ADD      #200,R0      ;INCREASE THE PAR VALUE BY 200
864      002116 077305                SOB      R3,6$       ;LOOP UNTIL 7 PAR'S ARE LOADED
865      002120 012711 177600      MOV      #177600,(R1) ;MAP I-SPACE PAR7 TO I/O PAGE
866      002124 012712 177600      MOV      #177600,(R2) ;MAP D-SPACE PAR7 TO I/O PAGE
867      002130 020127 172356      CMP      R1,#KIPAR7
868      002134 001005                BNE      7$
869      002136 012701 172240      MOV      #SIPAR0,R1
870      002142 012702 172260      MOV      #SDPAR0,R2
871      002146 000754                BR       5$
872      002150 020127 172256      7$:     CMP      R1,#SIPAR7
873      002154 001401                BEQ      8$
874      002156 000207                RTS      PC           ;BRANCH TO USER LOAD ROUTINE
875      002160 012701 177640      8$:     MOV      #UIPAR0,R1 ;RETURN TO CALLING ROUTINE
876      002164 012702 177660      MOV      #UDPAR0,R2
877      002170 000743                BR       5$
878

```

```

880 .SBITL D-SPACE TESTS MEMORY MANAGEMENT ABORT SERVICE ROUTINE
881 :*****
882 :* THIS ROUTINE WILL BE ENTERED IF A MEMORY MANAGEMENT ABORT OCCURS
883 :* DURING THE D-SPACE ENABLE TESTS. IF THE ABORT IS A NON-RESIDENT
884 :* ABORT, THE PROBLEM IS PROBABLY IN THE D-SPACE ENABLE LOGIC. IN
885 :* ALL OF THE D-SPACE ENABLE TESTS, D-SPACE PAGES 1 & 3 ARE MAPPED
886 :* NON-RESIDENT AND I-SPACE PAGE 3 IS MAPPED NON-RESIDENT. ALL
887 :* OTHER PAGES ARE MAPPED RESIDENT, 4K, READ/WRITE. THEREFORE, IF
888 :* THE NON-RESIDENT PAGE IS 1 OR 3 YOU ARE NOT FORCING I-SPACE WHEN
889 :* YOU SHOULD. IF THE NON-RESIDENT PAGE IS 3, AND YOU ARE IN TEST
890 :* 15, YOU ARE PROBABLY FORCING I-SPACE WHEN YOU SHOULD BE ALLOWING
891 :* D-SPACE.
892 :*****
893 002172 NODSPAC: ;STARTING ADDRESS FOR ABORT SERVICE ROUTINE
894 002172 042767 000004 170316 BIC #BIT2,MMR3 ;TURN OFF D-SPACE BEFORE DOING ROUTINE
895 002200 005227 INC (PC)+ ;MAKE FLAG ZERO IF THE FIRST TIME
896 002202 177777 NDFLAG: .WORD -1 ;FLAG SHOULD BE -1
897 002204 001401 BEQ 10$ ;BRANCH IF FIRST TIME IN ROUTINE
898 002206 000000 HALT ;I HAVE ENTERED THIS ROUTINE BEFORE
899 ;THE FIRST ERROR IS REPORTED; THE SECOND
900 ;ENTRY ADDRESS IS ON THE STACK, AND THE
901 ;FIRST ERROR CONDITION IS PROBABLY STILL
902 ;LOCKED UP.
903 002210 011667 177070 10$: MOV (KSP),BADPC ;SAVE PC AT TIME OF ABORT OR TRAP
904 002214 012667 177040 MOV (KSP)+,TRAPPC ;SAVE RETURN ADDRESS IN CASE OF LOOP
905 002220 012667 177036 MOV (KSP)+,TRAPPS ;SAVE OLD PSW IN CASE OF LOOP
906 002224 016767 175342 177032 MOV MMR0,WASSR0 ;SAVE STATUS REGISTER
907 002232 016767 175336 177026 MOV MMR1,WASSR1 ;SAVE AUTO INCR/DECR REGISTER
908 002240 016767 175332 177022 MOV MMR2,WASSR2 ;SAVE VIRTUAL ADDRESS REGISTER
909 002246 005767 177012 TST WASSR0 ;WAS ABORT NON-RESIDENT?
910 002252 100002 BPL 1$ ;BRANCH IF ABORT NOT EXPECTED
911 002254 104034 ERROR +34 ;D-SPACE ENABLE FAULTY
912 002256 000401 BR 2$ ;BRANCH TO EXIT
913 002260 104002 1$: ERROR +2 ;UNEXPECTED M.M. ABORT
914 002262 042767 177376 175302 2$: BIC #177376,MMR0 ;CLEAR ALL BITS EXCEPT 0 AND 8
915 002270 012767 177777 177704 MOV #-1,NDFLAG ;MOVE A -1 TO THE FLAG
916 002276 016746 176760 MOV TRAPPS,-(KSP) ;PUSH OLD PSW ONTO STACK
917 002302 016746 176752 MOV TRAPPC,-(KSP) ;PUSH OLD PC ONTO STACK
918 002306 052767 000004 170202 BIS #BIT2,MMR3 ;TURN D-SPACE BACK ON
919 002314 000006 RTT ;RETURN TO MAIN PROGRAM
920

```

```

922 .SBTTL TURN OFF T-BIT AND SAVE CURRENT PSW
923 :*****
924 :
925 : THIS SUBROUTINE IS USED TO TURN OFF THE TRACE TRAP BIT IN
926 : THE PSW IF IT IS ON. THE PROCESSOR STATUS IS SAVED IN
927 : 'TBITPS' SO THAT THE PSW CAN BE RESTORED TO ITS PREVIOUS
928 : CONDITION WHEN CONDITIONS WARRANT T-BIT TRAPPING.
929 :
930 :*****
931 002316 036727 175454 000020 TOFF: BIT PSW,#TBIT ;IS THE T-BIT SET IN THE PSW?
932 002324 001411 BEQ 1$ ;EXIT IF NO
933 002326 016746 175444 MOV PSW,-(SP) ;PUSH PRESENT PSW ON THE STACK
934 002332 011667 176736 MOV (SP),TBITPS ;ALSO SAVE IT IN 'TBITPS' FOR
935 ;RESTORING LATER
936 002336 042716 000020 BIC #TBIT,(SP) ;CLEAR THE T-BIT (BIT 4) IN THE PSW
937 002342 012746 002350 MOV #1$,-(SP) ;PUSH PC OF 'RTS' ON STACK
938 002346 000006 RTT ;'RETURN' TO 1$ WITH T-BIT OFF
939 002350 000207 1$: RTS PC ;RETURN TO PROGRAM
940
941 .SBTTL TURN ON T-BIT AND RESTORE PREVIOUS PSW
942 :*****
943 :
944 : THIS SUBROUTINE IS USED TO RESTORE THE PROCESSOR STATUS
945 : TO ITS PREVIOUS CONDITION BY RESTORING THE 'T-BIT PSW'
946 : SAVED BY THE 'TOFF' SUBROUTINE IN THE 'TBITPS' LOCATION.
947 :
948 :*****
949 002352 036727 176716 000020 TON: BIT TBITPS,#TBIT ;WAS T-BIT ON IN THE PREVIOUS PSW?
950 002360 001410 BEQ 1$ ;EXIT IF NO
951 002362 016746 176706 MOV TBITPS,-(SP) ;PUSH PREVIOUS PSW ON THE STACK
952 002366 012767 000340 176700 MOV #340,TBITPS ;RESET THE 'TBITPS' LOCATION
953 002374 012746 002402 MOV #1$,-(SP) ;PUSH PC OF 'RTS' ON STACK
954 002400 000006 RTT ;'RETURN' TO 1$ WITH T-BIT RESTORED
955 002402 000207 1$: RTS PC ;RETURN TO PROGRAM
956
957
958
    
```

960
 961
 962
 963
 964
 965
 966
 967
 968
 969
 970
 971 002404 005227
 972 002406 177777
 973 002410 001401
 974 002412 000000
 975
 976
 977
 978
 979 002414 012667 176640
 980 002420 012667 176636
 981 002424 010667 176626
 982 002430 104001
 983 002432 012767 177777 177746
 984 002440 005067 175322
 985 002444 016746 176612
 986 002450 016746 176604
 987 002454 000006
 988
 989
 990
 991
 992
 993
 994
 995
 996
 997
 998
 999 002456 005227
 1000 002460 177777
 1001 002462 001401
 1002 002464 000000
 1003
 1004
 1005
 1006
 1007 002466 012667 176566
 1008 002472 012667 176564
 1009 002476 010667 176554
 1010 002502 016767 175064 176554
 1011 002510 016767 175062 176552
 1012 002516 042767 160000 175046
 1013 002524 104002
 1014 002526 012767 177777 177724
 1015 002534 016746 176522
 1016 002540 016746 176514

```

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

.SBTTL CPU TRAP HANDLER ROUTINE
:*****
:*
:* THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS THRU
:* 'ERRVEC' (LOC. 004). IF THIS SUBROUTINE IS ENTERED BY A
:* SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A HALT IS
:* EXECUTED.
:*****
TIMERR: INC      (PC)+      ;MAKE FLAG ZERO IF FIRST TIME THRU
TIMFLG: .WORD   -1        ;NEGATIVE ONE FOR 'HAVE ENTERED' FLAG
        BEQ     1$        ;BRANCH IF FIRST TIME IN
        HALT           ;STOP! - I'VE ENTERED THIS ROUTINE
                           ;A SECOND TIME BEFORE I FINISHED
                           ;REPORTING THE FIRST ERROR. THE
                           ;SECOND ENTRY ADDRESS SHOULD BE ON
                           ;THE KERNEL STACK.
1$:     MOV     (KSP)+,TRAPPC ;SAVE PC+2 AT TIME OF ABORT
        MOV     (KSP)+,TRAPPS ;SAVE PS AT TIME OF ABORT
        MOV     KSP,WASR6    ;SAVE STACK POINTER VALUE
        ERROR   +1         ;UNEXPECTED TRAP OR ABORT TO LOC. 4
        MOV     #-1,TIMFLG  ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
        CLR     CPUERR      ;CLEAR THE CPU ERROR REGISTER
        MOV     TRAPPS,-(KSP);PUT PC & PS OF TRAP ON STACK
        MOV     TRAPPC,-(KSP)
        RTT              ;RETURN FROM INTERRUPT OR ABORT
  
```

```

.SBTTL MEMORY MANAGEMENT TRAP HANDLER ROUTINE
:*****
:*
:* THIS SUBROUTINE WILL HANDLE ALL UNEXPECTED MEMORY MANAGEMENT
:* TRAPS AND ABORTS THRU 'MMVEC' (LOC. 250). IF THIS SUBROUTINE IS
:* ENTERED BY A SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A
:* HALT IS EXECUTED.
:*****
MGMERR: INC      (PC)+      ;MAKE FLAG ZERO IF FIRST TIME THRU
MGMFLG: .WORD   -1        ;NEGATIVE ONE FOR 'HAVE ENTERED' FLAG
        BEQ     1$        ;BRANCH IF FIRST TIME IN
        HALT           ;STOP! - I'VE ENTERED THIS ROUTINE
                           ;A SECOND TIME BEFORE I FINISHED
                           ;REPORTING THE FIRST ERROR. THE
                           ;SECOND ENTRY ADDRESS SHOULD BE ON
                           ;THE KERNEL STACK.
1$:     MOV     (KSP)+,TRAPPC ;SAVE PC+2 AT TIME OF ABORT
        MOV     (KSP)+,TRAPPS ;SAVE PS AT TIME OF ABORT
        MOV     KSP,WASR6    ;SAVE STACK POINTER VALUE
        MOV     SR0,WASSR0   ;SAVE CONTENTS OF KT STATUS REG. 0
        MOV     SR2,WASSR2   ;SAVE CONTENTS OF KT STATUS REG. 2
        BIC     #160000,SR0  ;CLEAR ERROR BITS IN STATUS REG 0
        ERROR   +2         ;UNEXPECTED TRAP OR ABORT TO LOC. 250
        MOV     #-1,MGMFLG  ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
        MOV     TRAPPS,-(KSP);PUT PC & PS OF TRAP ON STACK
        MOV     TRAPPC,-(KSP)
  
```

1017 002544 000006
1018

RTT

;RETURN FROM INTERRUPT OR ABORT.

1020
1021
1022
1023
1024 020000

020000

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

START:

.SBTTL INITIALIZE THE COMMON TAGS
;;CLEAR THE COMMON TAGS (\$CMTAG) AREA

020000 012706 001100
020004 005026
020006 022706 001140
020012 001374
020014 012706 001100

MOV #CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
CLR (R6)+ ;;CLEAR MEMORY LOCATION
CMP #SWR,R6 ;;DONE?
BNE -6 ;;LOOP BACK IF NO
MOV #STACK,SP ;;SETUP THE STACK POINTER

;;INITIALIZE A FEW VECTORS

020020 012737 036134 000020
020026 012737 000340 000022
020034 012737 036326 000030
020042 012737 000340 000032
020050 012737 041750 000034
020056 012737 000340 000036
020064 012737 042036 000024
020072 012737 000340 000026
020100 016767 015600 015570
020106 005067 161100
020112 112767 000001 160775

MOV #SCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
MOV #340,@IOTVEC+2 ;;LEVEL 7
MOV #ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
MOV #340,@EMTVEC+2 ;;LEVEL 7
MOV #STRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
MOV #340,@TRAPVEC+2;LEVEL 7
MOV #SPWRDN,@PWRVEC ;;POWER FAILURE VECTOR
MOV #340,@PWRVEC+2 ;;LEVEL 7
MOV SENDCT,\$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
CLR \$ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
MOVB #1,\$ERMAX ;;ALLOW ONE ERROR PER TEST

;;INITIALIZE THE 'T-BIT' TRAP VECTOR. THEN LOAD LOCATION '\$RTRN', IN
;;THE 'END-OF-PASS' (\$EOP) ROUTINE, WITH A 'RTI' OR 'RTT'.

020120 012737 036122 000014
020126 012737 000340 000016
020134 012767 000002 015760
020142 012737 020170 000010
020150 005046
020152 012746 020160
020156 000006
020160 012767 000006 015734 64\$:
020166 000402
020170 062706 000010 65\$:
020174 012737 000012 000010 66\$:
020202 005067 161102
020206 012767 020206 160672
020214 012767 020214 160666

MOV #RTRN,@TBITVEC ;;SET 'T' BIT VECTOR TO \$RTRN
MOV #340,@TBITVEC+2 ;;LEVEL 7
MOV #RTI,\$RTRN ;;SET \$RTRN TO A RTI
MOV #65\$,@RESVEC ;;TRY TO DO A RTT
CLR -(SP) ;;DUMMY PS
MOV #64\$,-(SP) ;;AND PC
RTT ;;TRY THE RTT
MOV #RTT,\$RTRN ;;RTT IS LEGAL--SET \$RTRN TO A RTT
BR 66\$
ADD #10,SP ;;RTT ILLEGAL--CLEAN OFF THE STACK
MOV #RESVEC+2,@RESVEC ;;RESTORE TRAP CATCHER
CLR \$TBIT ;;CLEAR 'T' BIT SWITCH
MOV #,\$LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
MOV #,\$LPERR ;;SETUP THE ERROR LOOP ADDRESS

;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
;;EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.

020222 013746 000004
020226 012737 020262 000004
020234 012767 177570 160676
020242 012767 177570 160672
020250 022777 177777 160662
020256 001012

MOV @ERRVEC,-(SP) ;;SAVE ERROR VECTOR
MOV #67\$,@ERRVEC ;;SET UP ERROR VECTOR
MOV #DSWR,\$SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
MOV #DDISP,\$DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
CMP #-1,\$SWR ;;TRY TO REFERENCE HARDWARE SWR
BNE 69\$;;BRANCH IF NO TIMEOUT TRAP OCCURRED
;;AND THE HARDWARE SWR IS NOT -1

020260 000403
020262 012716 020270 67\$:
020266 000002
020270 012767 000176 160642 68\$:
020276 012767 000174 160636
020304 012637 000004 69\$:
020310 005067 160716
020314 132767 000200 160723

BR 68\$;;BRANCH IF NO TIMEOUT
MOV #68\$,(SP) ;;SET UP FOR TRAP RETURN
RTI
MOV #SWREG,\$SWR ;;POINT TO SOFTWARE SWR
MOV #DISPREG,\$DISPLAY
MOV (SP)+,@ERRVEC ;;RESTORE ERROR VECTOR
CLR \$PASS ;;CLEAR PASS COUNT
BITB #APTSIZE,\$ENVM ;;TEST USER SIZE UNDER APT

```

020322 001403          BEQ    70$          ;;YES,USE NON-APT SWITCH
020324 012767 001246 160606  MOV    #$$SWREG,SWR      ;;NO,USE APT SWITCH REGISTER
020332
1025 70$:
.SBTTL  TYPE PROGRAM NAME
;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
020332 005227 177777      INC    #-1              ;;FIRST TIME?
020336 001047          BNE    71$              ;;BRANCH IF NO
020340 022737 036056 000042  CMP    #SENDAD,@#42     ;;ACT-11?
020346 001443          BEQ    71$              ;;BRANCH IF YES
020350 104401 020416      TYPE    ,72$            ;;TYPE ASCIZ STRING
.SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
020354 005737 000042      TST    @#42             ;;ARE WE RUNNING UNDER XXDP/ACT?
020360 001012          BNE    73$              ;;BRANCH IF YES
020362 126727 160656 000001  CMPB   $ENV,#1          ;;ARE WE RUNNING UNDER APT?
020370 001406          BEQ    73$              ;;BRANCH IF YES
020372 026727 160542 000176  CMP    SWR,#SWREG       ;;SOFTWARE SWITCH REG SELECTED?
020400 001005          BNE    74$              ;;BRANCH IF NO
020402 104407          GTSWR                    ;;GET SOFT-SWR SETTINGS
020404 000403          BR     74$
020406 112767 000001 160520 73$:  MOVB   #1,$AUTOB        ;;SET AUTO-MODE INDICATOR
020414          74$:
020414 000420          BR     71$              ;;GET OVER THE ASCIZ
1026 020456          ;;72$: .ASCIZ <CRLF>#CKKTBAO 11/44 MEM MGMT PRT B#<CRLF>
1027          71$:
1028          LOOP:
1028 020456 012706 001100      MOV    #STACK,KSP      ;;INITIALIZE THE STACK POINTER
1029 020462 012767 040000 157306  MOV    #4000,PSW       ;;TURN ON SUPERVISOR MODE
1030 020470 012706 000700      MOV    #SUPSTK,SSP     ;;INITIALIZE SUPER. STACK POINTER
1031 020474 012767 140000 157274  MOV    #140000,PSW     ;;TURN ON USER MODE
1032 020502 012706 000600      MOV    #USESTK,USP     ;;INITIALIZE USER STACK POINTER
1033 020506 005067 157264      CLR    PSW             ;;RETURN TO KERNAL MODE
1034 020512 012767 002404 157264  MOV    #TIMERR,ERRVEC  ;;LOAD CPU SERVICE ROUTINE INTO TRAP VECTOR
1035 020520 012767 000340 157260  MOV    #340,ERRVEC+2   ;;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
1036 020526 012767 002456 157514  MOV    #MGMEERR,MMVEC  ;;LOAD MEMORY MANAGENT ROUTINE INTO VECTOR
1037 020534 012767 000340 157510  MOV    #340,MMVEC+2    ;;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
1038 020542 012700 177777      MOV    #-1,R0          ;;PUT -1 INTO R0 TO INITIALIZE FLAGS
1039 020546 010067 161634      MOV    R0,TIMFLG      ;;INITIALIZE CPU ERROR FLAG
1040 020552 010067 161702      MOV    R0,MGMFLG      ;;INITIALIZE MEMORY MANAGEMENT ERROR FLAG
1041 020556 012767 000340 160510  MOV    #340,TBITPS     ;;INITIALIZE LOG THAT HOLDS T-BIT PSW
1042 020564 005067 157002      INIT: CLR    MMR0        ;;BE SURE MEM. MGMT IS OFF TO START WITH,
1043 020570 012767 000020 151720  MOV    #BIT4,MMR3      ;;BUT TURN ON 22-BIT ADDRESSING MODE
1044 020576 004767 161216      JSR    PC,APRINIT     ;;JUMP TO PAR/PDR INIT ROUTINE

```

1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1071

:/.\./.
:SBTTL GROUP 5 ACCESS CONTROL FIELD/SPECIAL ABORTS TESTS
:/.\./.

* 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 SR0. SO THE SR0 AND SR2 LOGIC AND THE
* KT ERROR LOGIC ARE CHECKED.

* TEST 1 NON-RESIDENT ABORT TEST (ACF=084)
* THIS TEST CHECKS THE ACCESS CONTROL FIELD (ACF) COMPARATOR
* LOGIC BY CAUSING NON-RESIDENT ABORTS IN KERNEL, SUPERVISOR
* AND USER MODES. PDR 4 IS LOADED WITH ACF'S = 084 AND
* THEN PHYSICAL ADDR. 60000 IS ACCESSED TO CAUSE THE ABORT.

1072 020602 000004
1073 020604 012700 000600
1074 020610 010067 151532
1075 020614 010067 151530
1076 020620 010067 151422
1077 020624 010067 151420
1078 020630 010067 157012
1079 020634 010067 157010
1080 020640 012700 060000
1081 020644 012701 100000
1082 020650 012703 100011
1083 020654 012702 077400
1084 020660 012767 020734 157362
1085 020666 010267 151416
1086 020672 010267 151312
1087 020676 010267 156706
1088 020702 012767 020716 160200
1089 020710 012767 000001 156654
1090 020716 005010
1091 020720 016767 157052 160250
1092 020726 005211
1093 020730 104003
1094
1095
1096
1097 020732 000425
1098 020734 062706 000004
1099 020740 005710
1100 020742 001401
1101 020744 104004

TST1: SCOPE
1\$: MOV #600,R0 ;LOAD DATA FOR PAR'S INTO R0
MOV R0,KIPAR3 ;MAP KERNEL PAR'S 384 TO 12-16K
MOV R0,KIPAR4
MOV R0,SIPAR3 ;MAP SUPERVISOR PAR'S 384 TO 12-16K
MOV R0,SIPAR4
MOV R0,UIPAR3 ;MAP USER PAR'S 384 TO 12-16K
MOV R0,UIPAR4
MOV #60000,R0 ;LOAD VIRTUAL ADDR. TO REFERENCE PDR3 INTO R0
MOV #100000,R1 ;LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
MOV #100011,R3 ;LOAD R3 WITH WHAT SR0 SHOULD READ - N.R., KERNEL, PG.4
MOV #77100,R2 ;LOAD ACF=0 (NON-RESIDENT) PDR VALUE IN R2
2\$: MOV #5\$,MMVEC ;POINT MEM. MGMT. TRAP VECTOR TO 5\$ BELOW
MOV R2,KIPDR4 ;LOAD ACF TEST VALUE INTO KIPDR4
MOV R2,SIPDR4 ;LOAD ACF TEST VALUE INTO SIPDR4
MOV R2,UIPDR4 ;LOAD ACF TEST VALUE INTO UIPDR4
MOV #3\$,SLPERR ;SET LOOP ON ERROR POINTER TO 3\$
MOV #1,MMRO ;TURN ON MEMORY MANAGEMENT
3\$: CLR (R0) ;CLEAR PHYS. LOC. 60000 USING PDR3
MOV PSW,\$TMP0 ;SAVE PSW IN CASE OF ERROR
4\$: INC (R1) ;TRY TO REF. IT USING PDR4 - SHOULD TRAP TO 5\$
ERROR +3 ;MEM. MGMT. ABORT DID NOT OCCUR
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 3\$' = 000772
5\$: BR 8\$;BRANCH AROUND STATUS REG. CHECKS IF NO ABORT
ADD #4,SP ;RESTORE STACK POINTER
TST (R0) ;DID INSTRUCTION GET ABORTED & NOT EXECUTE
BEQ 6\$;BRANCH IF YES
ERROR +4 ;INSTRUCTION WAS NOT ABORTED, LOC. GOT CHANGED

1151

```
*****
*TEST 2          READ-ONLY ABORT TEST (ACF=2)
*
*   THIS TEST CHECKS THE ACCESS CONTROL FIELD (ACF) COMPARATOR
*   LOGIC BY CAUSING READ-ONLY ABORTS IN KERNEL, SUPERVISOR AND
*   USER MODES.  PDR 4 IS LOAD WITH ACF=2 AND THEN
*   PHYSICAL ADDR. 60000 IS WRITTEN TO CAUSE THE ABORT.
*
*****
```

```

1152 021126 000004
1153 021130
1154 021130 012700 060000      MOV      #60000,R0
1155 021134 012701 100000      MOV      #100000,R1
1156 021140 012703 020011      MOV      #20011,R3
1157 021144 012702 077402      MOV      #77402,R2
1158 021150 012767 021222 157072 2$:  MOV      #5$,MMVEC
1159 021156 010267 151126      MOV      R2,KIPDR4
1160 021162 010267 151022      MOV      R2,SIPDR4
1161 021166 010267 156416      MOV      R2,UIPDR4
1162 021172 012767 021204 157710      MOV      #3$,SLPERR
1163 021200 005267 156366      INC      MMRO
1164 021204 005010              CLR      (R0)
1165 021206 016767 156564 157762 3$:  MOV      PSW,$TMP0
1166 021214 005211              INC      (R1)
1167 021216 104003              ERROR   +3
1168
1169
1170
1171 021220 000425              BR      8$
1172 021222 062706 000004 5$:  ADD      #4,SP
1173 021226 005710              TST     (R0)
1174 021230 001401              BEQ    6$
1175 021232 104004              ERROR   +4

```

```

;KERNEL, SUPERVISOR AND USER PAR'S 3 & 4,
;AND PDR 3 ARE SETUP FROM LAST TEST
;LOAD VIRTUAL ADDR. TO REFERENCE PDR3 INTO R0
;LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
;LOAD R3 WITH WHAT SRO SHOULD READ - R/O, KERNEL, PG.4
;LOAD ACF=2 (READ-ONLY) PDR VALUE IN R2
;POINT MEM. MGMT. TRAP VECTOR TO 5$ BELOW
;LOAD ACF=2 INTO KIPDR4
;LOAD ACF=2 INTO SIPDR4
;LOAD ACF=2 INTO UIPDR4
;SET LOOP ON ERROR POINTER TO 3$
;TURN ON MEMORY MANAGEMENT
;CLEAR PHYS. LOC. 60000 USING PDR3
;SAVE PSW IN CASE OF ERROR
;TRY TO WRITE USING PDR4 - SHOULD TRAP TO 5$
;MEM. MGMT. ABORT DID NOT OCCUR
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 3$' = 000772
;BRANCH AROUND STATUS REG. CHECKS IF NO ABORT
;RESTORE STACK POINTER
;DID INSTRUCTION GET ABORTED & NOT EXECUTE
;BRANCH IF YES
;INSTRUCTION WAS NOT ABORTED, LOC. GOT CHANGED

```

```

1177
1178
1179
1180 021234 016767 156332 160022 6$: MOV SRO,WASSRO ;FOR TIGHTER SCOPE LOOP
1181 021242 016767 156330 160020 MOV SR2,WASSR2 ;REPLACE ERROR CALL WITH
1182 021250 020367 160010 CMP R3,WASSRO ;'BR 3$' = 000764
1183 021254 001401 BEQ 7$ ;READ STATUS REG. 0
1184 021256 104005 ERROR +5 ;READ STATUS REG. 2
1185 ;DID SRO REPORT READ-ONLY ERROR CORRECTLY?
1186 ;BRANCH IF YES
1187 ;SRO DID NOT REPORT R/O ERROR CORRECTLY
1188 021260 012704 021214 7$: MOV #4$,R4 ;FOR TIGHTER SCOPE LOOP
1189 021264 020467 160000 CMP R4,WASSR2 ;REPLACE ERROR CALL WITH
1190 021270 001401 BEQ 8$ ;'BR 3$' = 000752
1191 021272 104006 ERROR +6 ;LOAD R4 WITH WHAT SR2 SHOULD READ
1192 ;DID SR2 LOCKUP RIGHT VIRTUAL ADDR. (=4$)?
1193 ;BRANCH IF YES
1194 ;SR2 DID NOT LOCKUP VIRTUAL ADDR. OF R/O ERROR
1195 021274 005067 156272 8$: CLR MMRO ;FOR TIGHTER SCOPE LOOP
1196 021300 032767 140000 157670 BIT #140000,$TMP0 ;REPLACE ERROR CALL WITH
1197 021306 001006 BNE 11$ ;'BR 3$' = 000744
1198 021310 012703 020151 MOV #20151,R3 ;TURN OFF MEMORY MANAGEMENT
1199 021314 012767 140000 156454 MOV #140000,PSW ;HAS ACF=2 BEEN TESTED IN USER MODE?
1200 021322 000712 BR 2$ ;BRANCH IF YES
1201 021324 022767 040000 157644 11$: CMP #40000,$TMP0 ;LOAD R3 WITH WHAT SRO SHOULD READ-R/O, USER, PG.4
1202 021332 001006 BNE 9$ ;GO TO USER MODE
1203 021334 012703 020051 MOV #20051,R3 ;REPEAT TEST IN USER MODE
1204 021340 012767 040000 156430 MOV #40000,PSW ;HAS ACF=2 BEEN TESTED IN SUPERVISOR MODE?
1205 021346 000700 BR 2$ ;BRANCH IF YES
1206 021350 005067 156422 9$: CLR PSW ;LOAD R3 WITH WHAT SRO SHOULD READ-R/O, SUPERVISOR, PG.4
1207 021354 012767 021130 157526 MOV #1$,$LPERR ;GO TO SUPERVISOR MODE
1208 021362 012767 002456 156660 MOV #MMGMERR,$MVEEC ;REPEAT TEST IN SUPERVISOR MODE
1209 ;GO BACK TO KERNEL MODE BEFORE LEAVING
1220 ;RESET LOOP ON ERROR POINTER TO 1$
;RESTORE ADDRESS OF NORMAL MEMORY
;MANAGEMENT ERROR ROUTINE TO MMVEC.

```

```

:*****
:*TEST 3 TEST ILLEGAL MODE '10'
:*
:* THIS TEST CHECKS TO SEE THAT A 10 IN THE CURRENT MODE BITS OF THE
:* PSW WHILE MEMORY MANAGEMENT IS ON IS ILLEGAL. A
:* MEMORY MANAGEMENT ABORT SHOULD OCCUR AND STATUS REGISTER 0
:* SHOULD REPORT NON-RESIDENT ABORT, MODE = 10, PAGE = 1 (100103). STATUS
:* REGISTER 2 SHOULD LOCKUP THE ADDRESS OF THE INSTRUCTION
:* THAT LOADED THE PSW.
:*****

```

```

1221 021370 000004 TST3: SCOPE
1221 021372 012767 021414 157510 1$: MOV #2$,$LPERR ;SET LOOP ON ERROR POINTER TO 2$
1222 021400 012767 021426 156642 MOV #3$,$MVEEC ;LOAD MEM. MGMT. TRAP VECTOR WITH 3$
1223 021406 012767 000001 156156 MOV #1,$MMRO ;TURN ON MEMORY MANAGEMENT
1224 021414 012767 100000 156354 2$: MOV #100000,PSW ;SET 10 IN PSW CURRENT MODE BITS
1225 021422 104030 ERROR +30 ;ILLEGAL MODE 10 NOT ABORTED
1226 ;FOR A TIGHTER SCOPE LOOP
1227 ;REPLACE ERROR CALL WITH
1228 ;'BR 2$' = 000773
1229 021424 000424 BR 5$ ;BRANCH AROUND SRO & SR2 CHECKS
1230 021426 012706 001100 3$: MOV #KERSTK,$SP ;RESTORE STACK POINTER
1231 021432 012701 100103 MOV #100103,R1 ;LOAD EXPECTED CONTENTS OR SRO INTO R1
1232 021436 016767 156130 157620 MOV SRO,WASSRO ;READ CONTENTS OF SRO

```

```

1233
1234 021444 020167 157614      CMP      R1,WASSRO      ;DID SRO REPORT ILLEGAL MODE CORRECTLY?
1235 021450 001401              BEQ      4$              ;BRANCH IF YES
1236 021452 104031              ERROR   +31             ;SRO DID NOT REPORT NR ABORT, PG=1, MODE=10
1237                                ;FOR TIGHTER SCOPE LOOP
1238                                ;REPLACE ERROR CALL WITH
1239                                ;'BR 2$' = 000757
1240 021454 012701 021414      4$:  MOV      #2$,R1        ;LOAD EXPECTED CONTENTS OF SR2 INTO R1
1241 021460 016767 156112 157602  MOV      SR2,WASSR2     ;READ CONTENTS OF SR2
1242 021466 020167 157576      CMP      R1,WASSR2     ;DID SR2 LOCKUP VIRT. ADDR OF ABORTED INST.
1243 021472 001401              BEQ      5$              ;BRANCH IF YES
1244 021474 104013              ERROR   +13            ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ILL. MODE INST.
1245                                ;FOR TIGHTER SCOPE LOOP
1246                                ;REPLACE ERROR CALL WITH
1247                                ;'BR 2$' = 000746
1248 021476 042767 160000 156066 5$:  BIC      #160000,SRO    ;CLEAR POSSIBLE ERROR BITS IN SRO
1249 021504 012767 002456 156536  MOV      #MMGMERR,MMVEC ;RESTORE MEM. MGMT. ABORT VECTOR
1250 021512 012767 021372 157370  MOV      #1$, $LPERR    ;RESET LOOP ON ERROR POINTER TO 1$
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1282
1283

```

```

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

```

```

:*****
:*TEST 4 PAGE LENGTH FAULTS-UPWARD EXPANSION
:*
:* THIS TEST VARIES THE PAGE LENGTH FIELD (PLF) IN KERNEL PDR 4
:* FROM 1 TO 177 AND FOR EACH PLF, THREE VIRTUAL ADDRESSES (VBA'S)
:* ARE ACCESSED. WHEN VBA <12:6> IS LESS THAN OR EQUAL TO PDR <14:8>
:* NO ABORT SHOULD OCCUR. WHEN VBA <12:6> IS GREATER THAN PDR <14:8>,
:* A PAGE LENGTH ABORT SHOULD OCCUR AND BE REPORTED BY SRO & SR2.
:* THE PAGE EXPANSION DIRECTION IN THIS TEST IS UPWARD, (THE ED BIT
:* (BIT 3) OF PDR 4 = 0).
:*
:*****

```

```

1284 021520 000004              TST4: SCOPE
1284 021522 012767 077406 150556 1$:  MOV      #77406,KIPDR3 ;MAKE SURE PDR3 IS DESCRIBED AS R/W
1285 021530 012767 077406 150554  MOV      #77406,KIPDR5 ;MAKE SURE PDR5 IS DESCRIBED AS R/W
1286 021536 012700 100000      MOV      #100000,R0    ;LOAD VIRTUAL ADDR. TO SELECT PDR4 INTO R0
1287 021542 012704 000406      MOV      #406,R4       ;LOAD FIRST PDR VALUE IN R4 (PLF=1, ACF=6)
1288 021546 012767 021740 156474 2$:  MOV      #9$,MMVEC     ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS

```

1289	021554	010467	150530			MOV	R4,KIPDR4	:LOAD KIPDR4 WITH PAGE LENGTH VALUE
1290	021560	012767	021566	157322		MOV	#3\$,SLPERR	:SET LOOP ON ERROR POINTER TO 3\$
1291	021566	012706	001100		3\$:	MOV	#KERSTK,KSP	:MAKE SURE STACK POINTER IS ALL SET UP
1292	021572	011001				MOV	(R0),R1	:ACCESS VIRTUAL ADDR. (VBA < PLF - NO ABORT)
1293	021574	062700	000100			ADD	#100,R0	:FORM NEXT VIRTUAL ADDRESS IN R0
1294	021600	012767	021606	157302		MOV	#4\$,SLPERR	:SET LOOP ON ERROR POINTER TO 4\$
1295	021606	012706	001100		4\$:	MOV	#KERSTK,KSP	:MAKE SURE STACK POINTER IS ALL SET UP
1296	021612	011001				MOV	(R0),R1	:ACCESS VIRTUAL ADDR. (VBA=PLF - NO ABORT)
1297	021614	062700	000100			ADD	#100,R0	:FORM NEXT VIRTUAL ADDR IN R0
1298	021620	020027	117700			CMP	R0,#117700	:HAVE ALL PLF'S BEEN TESTED YET?
1299	021624	001470				BEQ	10\$:BRANCH IF ALL VBA'S & PLF'S HAVE BEEN USED
1300	021626	012767	021642	157254		MOV	#5\$,SLPERR	:SET LOOP ON ERROR POINTER TO 5\$
1301	021634	012767	021650	156406		MOV	#6\$,MMVEC	:SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT
1302	021642	011001			5\$:	MOV	(R0),R1	:ACCESS VIRTUAL ADDR. (VBA > PLF - ABORT TO 6\$)
1303	021644	104010				ERROR	+10	:EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
1304								:FOR TIGHTER SCOPE LOOP
1305								:REPLACE ERROR CALL WITH
1306								: 'BR 5\$' = 000776
1307	021646	000424				BR	8\$:BRANCH AROUND ABORT CHECKS
1308	021650	012706	001100		6\$:	MOV	#KERSTK,KSP	:RESTORE STACK POINTER FOLLOWING ABORT
1309	021654	016767	155712	157402		MOV	SRO,WASSRO	:READ M.M. STATUS REG. 0
1310	021662	016767	155710	157400		MOV	SR2,WASSR2	:READ M.M. STATUS REG. 2
1311	021670	012702	040011			MOV	#40011,R2	:PUT EXPECTED SRO CONTENTS IN R2
1312	021674	020267	157364			CMP	R2,WASSRO	:DID SRO REPORT PG. LENGTH ABORT, PAGE 4, KERNEL?
1313	021700	001401				BEQ	7\$:BRANCH IF YES
1314	021702	104011				ERROR	+11	:SRO DID NOT REPORT PG. LENGTH ABORT CORRECTLY
1315								:FOR TIGHTER SCOPE LOOP
1316								:REPLACE ERROR CALL WITH
1317								: 'BR 5\$' = 000757
1318	021704	012703	021642		7\$:	MOV	#5\$,R3	:PUT EXPECTED SR2 CONTENTS IN R3
1319	021710	020367	157354			CMP	R3,WASSR2	:DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?
1320	021714	001401				BEQ	8\$:BRANCH IF YES
1321	021716	104012				ERROR	+12	:SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY
1322								:FOR TIGHTER SCOPE LOOP
1323								:REPLACE ERROR CALL WITH
1324								: 'BR 5\$' = 000751
1325	021720	042767	160000	155644	8\$:	BIC	#160000,SRO	:CLEAR ERROR BITS IN SRO
1326	021726	062704	000400			ADD	#400,R4	:FORM NEXT PLF VALUE FOR KIPDR4
1327	021732	162700	000100			SUB	#100,R0	:FORM FIRST VIRT. ADDR FOR THAT PLF VALUE
1328	021736	000703				BR	2\$:BRANCH BACK AND ACCESS 3 VBA'S FOR
1329								:THE PLF VALUE JUST FORMED
1330	021740	012667	157314		9\$:	MOV	(KSP)+,TRAPPC	:SAVE PC & PS OF TRAP
1331	021744	012667	157312			MOV	(KSP)+,TRAPPS	
1332	021750	016767	155616	157306		MOV	SRO,WASSRO	:SAVE CONTENTS OF SRO FOR ERROR
1333	021756	016767	155614	157304		MOV	SR2,WASSR2	:SAVE CONTENTS OF SR2 FOR ERROR
1334	021764	042767	160000	155600		BIC	#160000,SRO	:CLEAR ERROR BITS IN SRO
1335	021772	104007				ERROR	+7	:GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
1336								:FOR TIGHTER SCOPE LOOP
1337								:REPLACE ERROR CALL WITH
1338								: A 'NOP' = 000240
1339	021774	016746	157262			MOV	TRAPPS,-(KSP)	:PUT PC & PS OF TRAP ON STACK
1340	022000	016746	157254			MOV	TRAPPC,-(KSP)	
1341	022004	000002				RTI		:RETURN FROM UNEXPECTED ABORT
1342								
1343	022006	012767	021522	157074	10\$:	MOV	#1\$,SLPERR	:RESET LOOP ON ERROR POINTER TO 1\$
1344	022014	012767	002456	156226		MOV	#MMGERR,MMVEC	:RESTORE NORMAL M.M. TRAP HANDLER
1345								:ADDRESS TO M.M. TRAP VECTOR

1346
 1358
 1359

```

*****
*TEST 5 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).
*
*****
    
```

```

TST5: SCOPE
1360 022022 000004 117700 1$: MOV #117700,R0 ;LOAD VIRTUAL ADDR. TO SELECT PDR4 INTO R0
1361 022024 012700 077016 MOV #77016,R4 ;LOAD FIRST PDR VALUE IN R4 (PLF=176,ACF=6)
1362 022034 012767 022226 156206 2$: MOV #9$,MMVEC ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS
1363 022042 010467 150242 MOV R4,KIPDR4 ;LOAD KIPDR4 WITH PAGE LENGTH VALUE
1364 022046 012767 022054 157034 MOV #3$,SLPERR ;SET LOOP ON ERROR POINTER TO 3$
1365 022054 012706 001100 3$: MOV #KERSTK,KSP ;MAKE SURE STACK POINTER IS ALL SET UP
1366 022060 011001 MOV (R0),R1 ;ACCESS VIRTUAL ADDR. (VBA > PLF - NO ABORT)
1367 022062 162700 000100 SUB #100,R0 ;FORM NEXT VIRTUAL ADDRESS IN R0
1368 022066 012767 022074 157014 MOV #4$,SLPERR ;SET LOOP ON ERROR POINTER TO 4$
1369 022074 012706 001100 4$: MOV #KERSTK,KSP ;MAKE SURE STACK POINTER IS ALL SET UP
1370 022100 011001 MOV (R0),R1 ;ACCESS VIRTUAL ADDR. (VBA=PLF - NO ABORT)
1371 022102 162700 000100 SUB #100,R0 ;FORM NEXT VIRTUAL ADDR. IN R0
1372 022106 020027 100000 CMP R0,#100000 ;HAVE ALL PLF'S BEEN TESTED YET?
1373 022112 001470 BEQ 10$ ;BRANCH IF ALL VBA'S & PLF'S HAVE BEEN USED
1374 022114 012767 022130 156766 MOV #5$,SLPERR ;SET LOOP ON ERROR POINTER TO 5$
1375 022122 012767 022136 156120 5$: MOV #6$,MMVEC ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT
1376 022130 011001 MOV (R0),R1 ;ACCESS VIRTUAL ADDR. (VBA < PLF - ABORT TO 6$)
1377 022132 104010 ERROR +10 ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
1378 ;FOR TIGHTER SCOPE LOOP
1379 ;REPLACE ERROR CALL WITH
1380 ;'BR 5$' = 000776
1381 022134 000424 BR 8$ ;BRANCH AROUND ABORT CHECKS
1382 022136 012706 001100 6$: MOV #KERSTK,KSP ;RESTORE STACK POINTER FOLLOWING ABORT
1383 022142 016767 155424 157114 MOV SRO,WASSRO ;READ M.M. STATUS REG. 0
1384 022150 016767 155422 157112 MOV SR2,WASSR2 ;READ M.M. STATUS REG. 2
1385 022156 012702 040011 MOV #40011,R2 ;PUT EXPECTED SRO CONTENTS IN R2
1386 022162 020267 157076 CMP R2,WASSRO ;DID SRO REPORT PG. LENGTH ABORT, PG. 4, KERNEL?
1387 022166 001401 BEQ 7$ ;BRANCH IF YES
1388 022170 104011 ERROR +11 ;SRO DID NOT REPORT PG. LENGTH ABORT CORRECTLY
1389 ;FOR TIGHTER SCOPE LOOP
1390 ;REPLACE ERROR CALL WITH
1391 ;'BR 5$' = 000757
1392 022172 012703 022130 7$: MOV #5$,R3 ;PUT EXPECTED SR2 CONTENTS IN R3
1393 022176 020367 157066 CMP R3,WASSR2 ;DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?
1394 022202 001401 BEQ 8$ ;BRANCH IF YES
1395 022204 104012 ERROR +12 ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY
1396 ;FOR TIGHTER SCOPE LOOP
1397 ;REPLACE ERROR CALL WITH
1398 ;'BR 5$' = 000751
1399 022206 042767 160000 155356 8$: BIC #160000,SRO ;CLEAR ERROR BITS IN SRO
1400 022214 162704 000400 SUB #400,R4 ;FORM NEXT PLF VALUE FOR KIPDR4
1401 022220 062700 000100 ADD #100,R0 ;FORM FIRST VIRT. ADDR. FOR THAT PLF VALUE
    
```

```

1402 022224 000703          BR      2$          ;BRANCH BACK AND ACCESS 3 VBA'S FOR
1403                          ;THE PLF VALUE JUST FORMED
1404 022226 012667 157026    9$:      MOV      (KSP)+,TRAPPC ;SAVE PC & PS OF TRAP
1405 022232 012667 157024      MOV      (KSP)+,TRAPPS
1406 022236 016767 155330 157020      MOV      SRO,WASSRO      ;SAVE CONTENTS OF SRO FOR ERROR
1407 022244 016767 155326 157016      MOV      SR2,WASSR2      ;SAVE CONTENTS OF SR2 FOR ERROR
1408 022252 042767 160000 155312      BIC      #160000,SRO      ;CLEAR ERROR BITS IN SRO
1409 022260 104007          ERROR    +7          ;GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
1410                          ;FOR TIGHTER SCOPE LOOP
1411                          ;REPLACE ERROR CALL WITH
1412                          ;A 'NOP' = 000240
1413 022262 016746 156774          MOV      TRAPPS,-(KSP)    ;PUT PC & PS OF TRAP ON STACK
1414 022266 016746 156766          MOV      TRAPPC,-(KSP)
1415 022272 000002          RTI              ;RETURN FROM UNEXPECTED ABORT
1416
1417 022274 012767 022024 156606 10$:     MOV      #1$,$LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
1418 022302 012767 002456 155740      MOV      #MMGMERR,MMVEC ;RESTORE NORMAL M.M. TRAP HANDLER
1419                          ;ADDRESS TO M.M. TRAP VECTOR
1420
1434
1435
    
```

```

*****
*TEST 6          SR2 BIT TEST
*
* THIS TEST CHECKS THE BITS IN MEMORY MANAGEMENT REGISTER 2 BY
* CAUSING 'READ-ONLY ABORTS' AT VIRTUAL ADDRESSES BETWEEN 100000
* TO 111000 (PHYSICAL ADDRESSES 060000-071000). KIPDR4 IS USED TO EXECUTE
* THE FOLLOWING FOUR WORDS OF CODE WHICH ARE MOVED THRU MEMORY:
* 010727 MOV      PC,(PC)+ ;THIS INSTRUCTION SHOULD CAUSE A R/O ABORT
* 000000          ;ITS VIRTUAL ADDR. SHOULD BE LOCKED UP IN SR2
* 000137 JMP      @#3$      ;THIS INSTRUCTION IS ALSO MOVED THRU MEMORY
* (ADDR. OF 3$)          ;IN CASE A R/O ABORT DOES NOT OCCUR,
*                          ;IN WHICH CASE SR2 WILL NOT CONTAIN CORRECT ADDR.
*****
    
```

```

022310 000004
1436 022312 012767 000600 150026 TST6:  SCOPE
1437 022320 012767 000600 150022 1$:      MOV      #600,KIPAR3      ;BE SURE PAR3 IS MAPPED TO 12-16K
1438 022326 012767 077406 147752      MOV      #600,KIPAR4      ;BE SURE PAR4 IS MAPPED TO 12-16K
1439 022334 012767 077402 147746      MOV      #77406,KIPDR3    ;MAP PAGE 3 128 BLOCKS, R/W
1440 022342 012700 060000          MOV      #77402,KIPDR4    ;MAP PAGE 4 128 BLOCKS, READ-ONLY
1441 022346 012701 100000          MOV      #60000,R0        ;LOAD R0 WITH VIRTUAL ADDR. WHICH USES PDR3
1442 022352 012767 022406 155670      MOV      #100000,R1       ;LOAD R1 WITH VIRTUAL ADDR. WHICH USES PDR4
1443 022360 012767 022366 156522      MOV      #3$,MMVEC        ;SET M.M. TRAP VECTOR TO 3$
1444 022366 012720 010727          MOV      #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
1445 022372 005020          MOV      #010727,(R0)+    ;LOAD 'MOV PC,(PC)+' INSTRUCTION AT ADDR.
1446 022374 012720 000137          CLR      (R0)+           ;REACHED THRU PDR/PAR 4.
1447 022400 012710 022406          MOV      #000137,(R0)+   ;LOAD 'JMP @#3$' INSTRUCTION AT VIRT. ADDR.
1448 022404 010107          MOV      #3$,(R0)        ;IN CASE R/O VIOL. DOES NOT ABORT
1449 022406 012706 001100          MOV      R1,PC           ;TRANSFER PROGRAM EXECUTION TO 'PAGE 4 INSTRUCTIONS'
1450 022412 016767 155160 156650 3$:      MOV      #KERSTK,KSP      ;RESTORE STACK POINTER
1451 022420 020167 156644          MOV      SR2,WASSR2      ;READ CONTENTS OF STATUS REG 2
1452 022424 001401          CMP      R1,WASSR2       ;WAS ADDR. OF 'RELOCATED - R/O ABORT' LOCKED UP?
1453 022426 104013          BEQ      4$              ;BRANCH IF YES
1454                          ;SR2 DID NOT LOCK UP VIRTUAL ADDR. OF R/O VIOL.
1455                          ;FOR TIGHTER SCOPE LOOP
1456                          ;REPLACE ERROR CALL WITH
1457 022430 042767 160000 155134 4$:      BIC      #160000,SRO      ;CLEAR THE ERROR BITS IN SRO
    
```

```

1458 022436 162700 000004          SUB    #4,R0          ;RESET R0 TO POINT TO NEXT VIRT. ADDR. TO LOAD
1459 022442 062701 000002          ADD    #2,R1          ;FORM VIRTUAL ADDR. THAT SHOULD BE LOCKED UP NEXT
1460 022446 020127 111002          CMP    R1,#111002    ;HAVE ALL VBA'S 100000-111000 BEEN TESTED?
1461 022452 103745          BLO    2$            ;BRANCH IF NO
1462
1463 022454 012767 022312 156426 5$:  MOV    #1$,SLPERR    ;RESET LOOP ON ERROR POINTER TO 1$
1464 022462 012767 077406 147620    MOV    #77406,KIPDR4 ;RESTORE PDR4 TO R/W ACCESS
1465 022470 012767 002456 155552    MOV    #MMGMERR,MMVEC ;RESTORE ADDRESS OF NORMAL M.M.
1466                                     ;TRAP HANDLER TO M.M. VECTOR
1467
1481
1482
  
```

 :TEST 7 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.

```

022476 000004
1483 022500 012767 000600 147644 TST7: SCOPE
1484 022506 012767 000406 147574 1$: MOV    #600,KIPARS    ;MAP KERNEL PAGE 5 TO 12-16K
1485 022514 012767 077402 147570    MOV    #406,KIPDR4    ;SETUP PDR4 FOR PAGE LENGTH ABORT
1486 022522 012767 022530 156360    MOV    #77402,KIPDR5 ;SETUP PDR5 FOR R/O ABORT
1487 022530 016767 155042 156532 2$: MOV    #2$,SLPERR    ;SET LOOP ON ERROR POINTER TO 2$
1488 022536 012701 022530    MOV    SR2,WASSR2    ;READ SR2 TO SEE IF ITS TRACKING
1489 022542 020167 156522    MOV    #2$,R1        ;PUT EXPECTED VIRTUAL PC IN R1
1490 022546 001401    CMP    R1,WASSR2    ;DID SR2 CONTAIN VIRTUAL PC AT 2$?
1491 022550 104016    BEQ    3$            ;BRANCH IF YES
1492                                     ;SR2 NOT TRACKING CORRECTLY
1493                                     ;FOR TIGHTER SCOPE LOOP
1494                                     ;REPLACE ERROR CALL WITH
1495                                     ;'BR 2$' = 000767
1495 022552 012767 022560 156330 3$: MOV    #4$,SLPERR    ;SET LOOP ON ERROR POINTER TO 4$
1496 022560 016767 155012 156502 4$: MOV    SR2,WASSR2    ;READ SR2 TO SEE IF ITS TRACKING
1497 022566 012701 022560    MOV    #4$,R1        ;PUT EXPECTED VIRTUAL PC IN R1
1498 022572 020167 156472    CMP    R1,WASSR2    ;DID SR2 CONTAIN VIRTUAL PC AT 4$
1499 022576 001401    BEQ    5$            ;BRANCH IF YES
1500 022600 104016    ERROR  +16          ;SR2 NOT TRACKING CORRECTLY
1501                                     ;FOR TIGHTER SCOPE LOOP
1502                                     ;REPLACE ERROR CALL WITH
1503                                     ;'BR 4$' = 000767
1504 022602 012767 022610 156300 5$: MOV    #6$,SLPERR    ;SET LOOP ON ERROR POINTER TO 6$
1505 022610 012767 022626 155432 6$: MOV    #7$,MMVEC    ;PUT ADDRESS OF 7$ IN M.M. TRAP VECTOR
1506 022616 005067 156356    CLR    $TMP1        ;CLEAR ERROR INDICATOR
1507 022622 005237 100500    INC    @#100500     ;CAUSE PAGE LENGTH ABORT - TRAP TO 7$
1508 022626 012706 001100    MOV    #KERSTK,KSP  ;RESTORE STACK POINTER AFTER ABORT
1509 022632 016767 154734 156336    MOV    SRO,$TMP0    ;SAVE SRO'S INFORMATION ON PG. LGTH. ABORT
1510 022640 016767 154732 156334    MOV    SR2,$TMP2    ;SAVE SR2'S INFORMATION ON PG. LGTH. ABORT
1511 022646 012767 022660 155374    MOV    #8$,MMVEC    ;PUT ADDRESS OF 8$ IN M.M. TRAP VECTOR
1512 022654 005237 120000    INC    @#120000     ;CAUSE R/O ABORT - TRAP TO 8$
1513 022660 012706 001100    8$: MOV    #KERSTK,KSP ;RESTORE STACK POINTER AFTER ABORT
  
```


1514	022664	016767	154702	156372		MOV	SRO,WASSRO	:READ SRO FOLLOWING SECOND KT ABORT
1515	022672	016767	154700	156370		MOV	SR2,WASSR2	:READ SR2 FOLLOWING SECOND KT ABORT
1516	022700	026767	156272	156356		CMP	\$TMP0,WASSRO	:IS SRO STILL HOLDING INFO ON FIRST ABORT?
1517	022706	001402				BEQ	9\$:BRANCH IF YES
1518	022710	005267	156264			INC	\$TMP1	:SET ERROR INDICATOR
1519	022714	026767	156262	156346	9\$:	CMP	\$TMP2,WASSR2	:DOES SR2 STILL HOLD PC OF FIRST ABORT?
1520	022722	001402				BEQ	10\$:BRANCH IF YES
1521	022724	005267	156250			INC	\$TMP1	:SET ERROR INDICATOR
1522	022730	005767	156244		10\$:	TST	\$TMP1	:WERE SRO OR SR2 CHANGED BY A SECOND ABORT?
1523	022734	001401				BEQ	11\$:BRANCH IF NO
1524	022736	104014				ERROR	+14	:ONE OF STATUS REGS. CHANGED BY SECOND ABORT
1525								:FOR TIGHTER SCOPE LOOP
1526								:REPLACE ERROR CALL WITH
1527								: 'BR 6\$' = 000726
1528	022740	005067	156234		11\$:	CLR	\$TMP1	:CLEAR ERROR INDICATOR
1529	022744	000005				RESET		:EXECUTE A RESET, APPLYING AN 'INIT'
1530	022746	016767	154620	156310		MOV	SRO,WASSRO	:READ SRO
1531	022754	005767	156304			TST	WASSRO	:WAS SRO CLEARED BY THE RESET?
1532	022760	001402				BEQ	12\$:BRANCH IF YES
1533	022762	005267	156212			INC	\$TMP1	:SRO NOT CLEARED BY A RESET
1534	022766	016767	154604	156274	12\$:	MOV	SR2,WASSR2	:READ SR2
1535	022774	022767	022766	156266		CMP	#12\$,WASSR2	:WAS SR2 UNLOCKED BY A RESET?
1536	023002	001402				BEQ	13\$:BRANCH IF YES
1537	023004	005267	156170			INC	\$TMP1	:SR2 NOT UNLOCKED BY A RESET
1538	023010	005767	156164		13\$:	TST	\$TMP1	:WERE SRO & SR2 BOTH 'RESET' BY A RESET?
1539	023014	001401				BEQ	14\$:BRANCH IF YES
1540	023016	104015				ERROR	+15	:SRO OR SR2 NOT 'RESET' BY A RESET
1541								:FOR TIGHTER SCOPE LOOP
1542								:REPLACE ERROR CALL WITH
1543								: 'BR 6\$' = 000676
1544	023020	005267	154546		14\$:	INC	SRO	:TURN MEMORY MANAGEMENT BACK ON
1545	023024	016767	154546	156236	15\$:	MOV	SR2,WASSR2	:READ SR2 TO SEE IF ITS TRACKING AGAIN
1546	023032	012701	023024			MOV	#15\$,R1	:PUT EXPECTED VIRTUAL PC IN R1
1547	023036	020167	156226			CMP	R1,WASSR2	:DID SR2 CONTAIN VIRTUAL PC AT 15\$
1548	023042	001401				BEQ	16\$:BRANCH IF YES
1549	023044	104016				ERROR	+16	:SR2 NOT TRACKING CORRECTLY
1550								:FOR TIGHTER SCOPE LOOP
1551								:REPLACE ERROR CALL WITH
1552								: 'BR 6\$' = 000663
1553	023046	012767	022500	156034	16\$:	MOV	#1\$, \$LPERR	:RESET LOOP ON ERROR POINTER TO 1\$
1554	023054	012767	077406	147226		MOV	#77406,KIPDR4	:RESET PDR4 TO 128 BLKS, R/W
1555	023062	012767	077406	147222		MOV	#77406,KIPDR5	:RESET PDR5 TO 128 BLKS, R/W
1556	023070	012767	002456	155152		MOV	#MGMERR,MMVEC	:RESTORE ADDRESS OF NORMAL MEMORY
1557								:MANAGEMENT TRAP ROUTINE TO M.M. VECTOR
1558								
1572								

1574

```

:*****
:TEST 10      SUPER/USER ABORT PICKS UP KERNEL VECTOR
:
:  THIS TEST CHECKS TO BE SURE THAT WHEN AN ABORT OCCURS WHILE
:  IN SUPERVISOR OR 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.
:  THE SUPERVISOR PAGE 0 IS THEN MAPPED TO 12K, AND THE TEST
:  IS REPEATED FOR SUPERVISOR MODE.  AN ODD ADDRESS ERROR IS
:  USED TO CAUSE A TRAP TO '4'.
:*****
  
```

```

1575 023076 000004
1576 023100 004767 157212
1577 023104 012767 023112 155776
1578 023112 005067 154660
1579 023116 012706 001100
1580
1581
1582 023122 012767 000600 154510
1583 023130 012737 023212 000004
1584 023136 012737 000340 000006
1585 023144 012767 140000 154624
1586 023152 012706 000600
1587 023156 012737 023176 000004
1588 023164 012737 000340 000006
1589 023172 005767 000001
1590
1591
1592 023176 016701 154574
1593 023202 010602
1594 023204 005067 154566
1595 023210 104017
1596
1597
1598
1599 023212 005067 154560
1600 023216 012706 001100
1601 023222 005067 154412
1602 023226 012767 140000 154542
1603 023234 012706 000600
1604 023240 005067 154532
1605
1606
1607
1608 023244 012767 000600 146766
1609 023252 012737 023334 000004
1610 023260 012737 000340 000006
1611 023266 012767 040000 154502
1612 023274 012706 000700
1613 023300 012737 023320 000004
1614 023306 012737 000340 000006
1615 023314 005767 000001
1616
  
```

```

TST10: SCOPE
1$: JSR PC,TOFF ;TURN OFF T-BIT TRAPPING FOR THIS TEST
    MOV #2$,SLPERR ;SET LOOP ON ERROR POINTER TO 2$
2$: CLR PSW ;GO TO KERNEL MODE
    MOV #KERSTK,KSP ;SETUP KERNEL STACK PTR.
:
: TEST USER MODE ABORT
:
    MOV #600,UIPARO ;MAP USER PAGE 0 TO 12K
    MOV #4$,@#4 ;LOAD KERNEL VECTOR 4 (LOC.4) WITH 4$
    MOV #340,@#6 ;LOAD VECTOR+2 WITH NEW PSW
    MOV #140000,PSW ;GO TO USER MODE
    MOV #USESTK,USP ;SETUP USER STACK PTR.
    MOV #3$,@#4 ;LOAD USER VECTOR 4 (LOC. 60004) WITH 3$
    MOV #340,@#6 ;LOAD VECTOR+2 WITH NEW PSW
    TST 3$+1 ;CAUSE ODD ADDR. ERROR TRAP TO '4'
                    ;SHOULD PICK UP NEW PC=4$ FROM KERNEL
                    ;LOC. 4, NOT PC=3$ FROM USER LOC. 4 (=60004)
3$: MOV PSW,R1 ;SAVE PSW FOR ERROR
    MOV SP,R2 ;SAVE VALUE OF STACK POINTER FOR ERROR
    CLR PSW ;BE SURE BACK IN KERNEL MODE
    ERROR +17 ;DID NOT TRAP THRU KERNEL SPACE
                    ;FOR TIGHTER SCOPE LOOP
                    ;REPLACE ERROR CALL WITH
                    ;'BR 2$' = 000740
4$: CLR PSW ;BE SURE BACK IN KERNEL MODE
    MOV #KERSTK,KSP ;RESTORE KERNEL S.P. IN CASE IT CHANGED
    CLR UIPARO ;REMAP USER PAGE 0 TO 0-4K
    MOV #140000,PSW ;GO TO USER MODE
    MOV #USESTK,USP ;RESTORE USER STACK POINTER
    CLR PSW ;GO BACK TO KERNEL MODE
:
: NOW TEST THE SUPERVISOR MODE ABORT
:
    MOV #600,SIPARO ;MAP SUPERVISOR PAGE 0 TO 12K
    MOV #6$,@#4 ;LOAD KERNAL VECTOR 4 WITH 6$
    MOV #340,@#6 ;LOAD VECTOR+2 WITH NEW PSW
    MOV #40000,PSW ;GO TO SUPERVISOR MODE
    MOV #SUPSTK,SSP ;SETUP SUPERVISOR STACK PTR.
    MOV #5$,@#4 ;LOAD SUPERVISOR VECTOR 4 (LOC. 60004) WITH 5$
    MOV #340,@#6 ;LOAD VECTOR+2 WITH NEW PSW
    TST 5$+1 ;CAUSE ODD ADDR. ERROR TRAP TO '4'
                    ;SHOULD PICK UP NEW PC=6$ FROM KERNEL
  
```

```

1617                                     ;LOC. 4, NOT PC=5$ FROM SUPERVISOR LOC. 4 (=60004)
1618 023320 016701 154452 5$:      MOV    PSW,R1      ;SAVE PSW FOR ERROR
1619 023324 010602                MOV    SP,R2      ;SAVE VALUE OF STACK POINTER FOR ERROR
1620 023326 005067 154444        CLR    PSW        ;BE SURE BACK IN KERNEL MODE
1621 023332 104017                ERROR  +17       ;DID NOT TRAP THRU KERNEL SPACE
1622                                     ;FOR TIGHTER SCOPE LOOP
1623                                     ;REPLACE ERROR CALL WITH
1624                                     ;'BR 2$' = 000740
1625 023334 005067 154436 6$:      CLR    PSW        ;BE SURE BACK IN KERNEL MODE
1626 023340 012706 001100        MOV    #KERSTK,KSP ;RESTORE KERNEL S.P. IN CASE IT CHANGED
1627 023344 005067 146670        CLR    SIPARO     ;REMAP SUPERVISOR PAGE 0 TO 0-4K
1628 023350 012767 040000 154420  MOV    #40000,PSW ;GO TO SUPERVISOR MODE
1629 023356 012706 000700        MOV    #SUPSTK,SSP ;RESTORE SUPERVISOR STACK POINTER
1630 023362 005067 154410        CLR    PSW        ;GO BACK TO KERNEL MODE
1631 023366 012737 002404 000004  MOV    #TIMERR,@#4 ;RESTORE ADDR. OF NORMAL CPU TRAP HANDLER TO 4
1632 023374 012767 023100 155506  MOV    #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
1633 023402 004767 156744        JSR    PC,T0N     ;TURN T-BIT TRAPPING BACK ON
1634
1641

```

```

*****
*TEST 11      RTI IN SUPER/USER MODE DOES NOT CHANGE PSW
*
*      THIS TEST CHECKS TO SEE THAT WHEN AN RTI IS EXECUTED IN SUPERVISOR
*      OR USER MODE, THE MODE OR PRIORITY BITS OF THE PSW ARE NOT CHANGED.
*
*****

```

```

023406 000004
1642
1643 023410 012767 023422 155472 1$:      MOV    #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
1644 023416 012702 170000        MOV    #170000,R2 ;LOAD 'PRESENT & EXPECTED' PSW VALUE INTO R2
1645 023422 010267 154350 2$:      MOV    R2,PSW     ;GO TO PRESENT MODE-PRIORITY 0
1646 023426 012746 000340        MOV    #340,-(SP) ;PUT A NEW PSW (PRIORITY=7) ON STACK
1647 023432 012746 023440        MOV    #3$,-(SP) ;PUT NEW PC ON THE STACK
1648 023436 000002                RTI                    ;DO AN RTI FROM PRESENT MODE
1649 023440 016701 154332 3$:      MOV    PSW,R1     ;READ NEW PSW INTO R1
1650 023444 042701 007437        BIC    #7437,R1    ;MASK OFF COND. CODE, T-BIT, AND UNUSED BITS
1651 023450 005067 154322        CLR    PSW        ;GO BACK TO KERNEL MODE
1652 023454 020201                CMP    R2,R1      ;DID PSW STAY IN PRESENT MODE, PRIORITY=0?
1653 023456 001401                BEQ    4$         ;BRANCH IF YES
1654 023460 104032                ERROR  +32       ;PSW CHANGED BY AN RTI FROM USER
1655                                     ;FOR A TIGHTER SCOPE LOOP
1656                                     ;REPLACE ERROR CALL WITH
1657                                     ;'BR=2$' = 000760
1658 023462 022702 050000 4$:      CMP    #50000,R2  ;IF SUPERVISOR MODE HAS BEEN CHECKED,
1659 023466 001403                BEQ    5$         ;GO TO END OF TEST
1660 023470 012702 050000        MOV    #50000,R2  ;ELSE, SET SUPERVISOR MODE,
1661 023474 000752                BR     2$         ;AND BRANCH BACK TO TEST IT.
1662 023476 012767 023410 155404 5$:      MOV    #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
1663

```

1676

```

*****
*TEST 12      KT ERROR NOT SERVICED IF ODD ADDR. ERROR
*
*   THIS TEST CHECKS TO SEE THAT IF A CERTAIN VIRTUAL ADDRESS THAT
*   WOULD CAUSE A MEMORY MANAGEMENT ERROR CAUSES AN ODD ADDRESS
*   ERROR FIRST, THE ODD ADDRESS ERROR IS SERVICED BUT THE MEMORY
*   MANAGEMENT ERROR ISN'T. THIS MEANS THAT SRO AND SR2
*   SHOULD NOT REPORT THE ERROR OR LOCK UP ITS VIRTUAL ADDRESS.
*   A READ-ONLY VIOLATION IS USED AS THE POTENTIAL MEMORY MANAGEMENT
*   ERROR
*
*****
  
```

```

*****
TST12: SCOPE
1677 023504 000004          1677 023506 012767 000600 146634 1$:  MOV    #600,KIPAR4    ;MAP KERNEL PAGE 4 TO 12-16K
1678 023514 012705 077402          1678 023514 012705 077402 1678 023514 012705 077402 1$:  MOV    #77402,R5      ;LOAD PDR4 DATA INTO R5
1679 023520 010567 146564          1679 023520 010567 146564 1679 023520 010567 146564 1$:  MOV    R5,KIPDR4     ;MAP PAGE 4 READ-ONLY
1680 023524 012737 023554 000004          1680 023524 012737 023554 000004 1680 023524 012737 023554 000004 1$:  MOV    #4$,@#4       ;SET CPU TRAP VECTOR TO ADDRESS OF 4$
1681 023532 012737 023552 000250          1681 023532 012737 023552 000250 1681 023532 012737 023552 000250 1$:  MOV    #3$,@#250     ;SET M.M. TRAP VECTOR TO ADDRESS OF 3$
1682 023540 012767 023546 155342          1682 023540 012767 023546 155342 1682 023540 012767 023546 155342 2$:  MOV    #2$, $LPERR   ;SET LOOP ON ERROR POINTER TO 2$
1683 023546 005267 034227          1683 023546 005267 034227 1683 023546 005267 034227 2$:  INC    60001         ;CAUSE ODD ADDR. ERROR & POTENTIAL R/O ABORT
1684 023552 104020          1684 023552 104020          1684 023552 104020          3$:  ERROR  +20         ;TRAPPED THRU M.M. VECTOR BUT SHOULDN'T HAVE
1685                                     ;FOR TIGHTER SCOPE LOOP
1686                                     ;REPLACE ERROR CALL WITH
1687                                     ;'BR 2$' = 000776
1688 023554 012706 001100          1688 023554 012706 001100 1688 023554 012706 001100 4$:  MOV    #KERSTK,KSP   ;RESTORE STACK POINTER AFTER TRAPPING
1689 023560 005067 155414          1689 023560 005067 155414 1689 023560 005067 155414 4$:  CLR    $TMP1         ;CLEAR ERROR INDICATOR
1690 023564 016767 154002 155472          1690 023564 016767 154002 155472 1690 023564 016767 154002 155472 5$:  MOV    SRO,WASSRO    ;READ STATUS REG. 0
1691 023572 016767 154000 155470          1691 023572 016767 154000 155470 1691 023572 016767 154000 155470 5$:  MOV    SR2,WASSR2    ;READ STATUS REG. 2
1692 023600 012700 000017          1692 023600 012700 000017 1692 023600 012700 000017 5$:  MOV    #17,R0        ;LOAD EXPECTED SRO CONTENTS INTO R0
1693 023604 020067 155454          1693 023604 020067 155454 1693 023604 020067 155454 5$:  CMP    RC,WASSRO     ;SRO ERROR BITS LEFT CLEAR BY TRAPPING?
1694 023610 001402          1694 023610 001402          1694 023610 001402          5$:  BEQ    6$           ;BRANCH IF YES
1695 023612 005267 155362          1695 023612 005267 155362 1695 023612 005267 155362 5$:  INC    $TMP1         ;SRO ERROR BITS SET WHEN ODD ADDR. SERVICED
1696 023616 012701 023572          1696 023616 012701 023572 1696 023616 012701 023572 6$:  MOV    #5$,R1        ;LOAD EXPECTED SR2 CONTENTS INTO R1
1697 023622 020167 155442          1697 023622 020167 155442 1697 023622 020167 155442 6$:  CMP    R1,WASSR2     ;WAS SR2 LEFT UNLOCKED BY TRAPPING?
1698 023626 001402          1698 023626 001402          1698 023626 001402          6$:  BEQ    7$           ;BRANCH IF YES
1699 023630 005267 155344          1699 023630 005267 155344 1699 023630 005267 155344 6$:  INC    $TMP1         ;SR2 LOCKED UP BY ODD ADDR. ERROR
1700 023634 005767 155340          1700 023634 005767 155340 1700 023634 005767 155340 7$:  TST    $TMP1         ;WHERE SRO OR SR2 EFFECTED?
1701 023640 001404          1701 023640 001404          1701 023640 001404          7$:  BEQ    8$           ;BRANCH IF NO
1702 023642 104021          1702 023642 104021          1702 023642 104021          7$:  ERROR  +21         ;SRO OR SR2 CHANGED BY ODD ADDR. ERROR
1703                                     ;FOR TIGHTER SCOPE LOOP
1704                                     ;REPLACE ERROR CALL WITH
1705                                     ;'BR 2$' = 000741
1706 023644 042767 160000 153720          1706 023644 042767 160000 153720 1706 023644 042767 160000 153720 8$:  BIC    #160000,SRO   ;CLEAR ERROR BITS THAT MAY BE SET IN SRO
1707 023652 012737 002404 000004          1707 023652 012737 002404 000004 1707 023652 012737 002404 000004 8$:  MOV    #TIMERR,@#4   ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
1708 023660 012737 002456 000250          1708 023660 012737 002456 000250 1708 023660 012737 002456 000250 8$:  MOV    #MMGMERR,@#250 ;RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER
1709 023666 012767 077406 146414          1709 023666 012767 077406 146414 1709 023666 012767 077406 146414 8$:  MOV    #77406,KIPDR4 ;REMAP PAGE 4 TO READ/WRITE
1710 023674 012767 023506 155206          1710 023674 012767 023506 155206 1710 023674 012767 023506 155206 8$:  MOV    #1$, $LPERR   ;RESET LOOP ON ERROR POINTER TO 1$
1711
1726
1727
*****
  
```

```

*****
*TEST 13      PC & PSW SAVED FOR KT ERROR ON ODD ADDR.
*
*   THIS TEST CHECKS THE PC AND PROCESSOR STATUS WORD SAVED WHEN
*   A KT ERROR OCCURS DURING THE SECOND PUSH ON THE STACK DURING
*   SERVICING OF AN ODD ADDR. ERROR. DURING A 'DOUBLE ERROR'
*   SEQUENCE SUCH AS THIS, THE PSW SAVED WILL BE THE ONE PICKED UP
*   FROM VECTOR+2 (LOC. 6 IN THIS CASE) AFTER THE FIRST TRAP,
*
*****
  
```

```

:* NOT THE PSW PRESENT BEFORE THE FIRST TRAP. SRO AND SR2
:* SHOULD RECORD THE KT ERROR (A R/O VIOLATION BY THE USER STACK PTR.)
:*
:* NOTE THAT THE PREVIOUS MODE BITS <13:12> OF THE PSW
:* WILL BE SET IN THE PSW THAT IS SAVED.

```

```

*****
TST13: SCOPE
1728 023702 000004      1728 023704 004767 156406      1$: JSR PC,TOFF ;TURN T-BIT TRAPPING OFF FOR THIS TEST
1729 023710 012767 000600 153730      MOV #600,UIPAR3 ;MAP USER PAGE 3 TO 12-16K
1730 023716 012767 000600 153724      MOV #600,UIPAR4 ;MAP USER PAGE 4 TO 12-16K
1731 023724 012767 077402 153654      MOV #77402,UIPDR3 ;MAP USER PAGE 3 READ-ONLY
1732 023732 012767 077406 153650      MOV #77406,UIPDR4 ;MAP USER PAGE 4 READ/WRITE
1733 023740 012737 024014 000004      MOV #4$,@#4 ;LOAD ADDRESS OF 4$ IN CPU (ODD ADDR.) VECTOR
1734 023746 012737 140017 000006      MOV #140017,@#6 ;LOAD PSW THAT SHOULD BE PUT ON STACK IN VECTOR+2
1735 023754 012737 024014 000250      MOV #4$,@#250 ;LOAD ADDRESS OF 4$ IN M.M. TRAP VECTOR
1736 023762 012737 000340 000252      MOV #340,@#252 ;LOAD A KERNEL PSW IN MMVEC+2
1737 023770 012767 023776 155112      MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
1738 023776 012767 140000 153772      2$: MOV #140000,PSW ;GO TO USER MODE
1739 024004 012706 100002      MOV #100002,USP ;SET USER STACK PTR. SO SECOND PUSH IS IN PG. 3
1740 024010 005267 053771      3$: INC 100005 ;CAUSE ODD ADDRESS ERROR THAT WILL CAUSE
1741 ;R/O ERROR WHEN TRY TO SAVE OLD PC
1742 024014 016601 000002      4$: MOV 2(KSP),R1 ;PUT PSW SAVED ON KERNEL STACK INTO R1
1743 024020 011603      MOV (KSP),R3 ;PUT PC SAVED ON KERNEL STACK INTO R3
1744 024022 016767 153544 155234      MOV SRO,WASSRO ;READ THE CONTENTS OF M.M. STATUS REG. 0
1745 024030 016767 153542 155232      MOV SR2,WASSR2 ;READ THE CONTENTS OF M.M. STATUS REG. 2
1746 024036 042767 160000 153526      BIC #160000,SRO ;CLEAR THE ERROR BITS IN SRO
1747 024044 005067 153726      CLR PSW ;BE SURE IN KERNEL MODE
1748 024050 012706 001100      MOV #KERSTK,KSP ;RESTORE KERNEL STACK POINTER
1749 024054 012767 140000 153714      MOV #140000,PSW ;GO TO USER MODE
1750 024062 012706 000600      MOV #USESTK,USP ;RESTORE USER STACK POINTER
1751 024066 005067 153704      CLR PSW ;GO BACK TO KERNEL MODE
1752 024072 005067 155100      CLR $TMP0 ;CLEAR ERROR INDICATOR
1753 024076 020127 170017      CMP R1,#170017 ;WAS THE PSW SAVED THE ONE PICKED UP BY THE
1754 ;ODD ADDR. TRAP FROM ERRVEC+2?
1755 ;VALUE 170017 = PSW FROM LOC. 6 WITH
1756 ;PREVIOUS MODE BITS = USER
1757 024102 001402      BEQ 5$ ;BRANCH IF YES
1758 024104 005267 155066      INC $TMP0 ;WRONG PSW SAVED DURING 'DOUBLE ERROR' SEQUENCE
1759 024110 020327 024014      5$: CMP R3,#3$+4 ;WAS THE PC AT THE TIME OF THE ODD ADDR. ERROR
1760 ;SAVED ON THE STACK?
1761 024114 001402      BEQ 6$ ;BRANCH IF YES
1762 024116 005267 155054      INC $TMP0 ;WRONG PC SAVED DURING TRAP SEQUENCE
1763 024122 026727 155136 020147      6$: CMP WASSRO,#20147 ;DID SRO REPORT - USER, PAGE 3, R/O ABORT?
1764 024130 001402      BEQ 7$ ;BRANCH IF YES
1765 024132 005267 155040      INC $TMP0 ;SRO DID NOT REPORT R/O ABORT
1766 024136 026727 155126 024010      7$: CMP WASSR2,#3$ ;DID SR2 LOCK UP VIRTUAL ADDR. OF LAST
1767 ;INSTRUCTION SUCCESSFULLY FETCHED?
1768 024144 001402      BEQ 8$ ;BRANCH IF YES
1769 024146 005267 155024      INC $TMP0 ;SR2 DID NOT LOCK UP ADDR. OF ODD ADDR. INST.
1770 024152 005767 155020      8$: TST $TMP0 ;ANY 'ERRORS' DURING TRAP SEQUENCE?
1771 024156 001401      BEQ 9$ ;BRANCH IF NO
1772 024160 104022      ERROR +22 ;THE WRONG PC OR PSW WERE SAVED
1773 ;OR SRO OR SR2 DID NOT REPORT R/O
1774 ;ERROR DURING ODD ADDR. - KT TRAP
1775 ;SEQUENCE
1776 ;FOR TIGHTER SCOPE LOOP

```

```
1777  
1778  
1779 024162 012737 002404 000004 9$: MOV #TIMERR,@#4 ;REPLACE ERROR CALL WITH  
1780 024170 012737 000340 000006 MOV #340,@#6 ;'BR 2$' = 000710  
1781 024176 012737 002456 000250 MOV #MGMERR,@#250 ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER  
1782 024204 012767 077406 153374 MOV #77406,UIPDR3 ;RELOAD ERRVEC+2 WITH KERNEL PSW  
1783 024212 012767 023704 154670 MOV #1$, $LPERR ;RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER  
1784 024220 004767 156126 JSR PC,TON ;REMAP USER PAGE 3 READ/WRITE  
;RESET LOOP ON ERROR POINTER TO 1$  
;TURN T-BIT TRAPPING BACK ON
```



```
1833 024334 005300          DEC    R0
1834 024336 006000          ROR    R0
1835 024340 006100          ROL    R0
1836 024342 005067 153224  CLR    MMRO ;TURN OFF MEMORY MANAGEMENT
1837          :*          TEST SOB NON-MOD WITH DSTM-2,3,6,7; DSTF=7
1838          :*
1839 024346 012737 100002 060000 MOV    #100002,@#60000 ;SET UP TEST LOCATION
1840 024354 012767 024366 154526 MOV    #11$,SLPERR ;SET LOOP ON ERROR POINTER TO 11$
1841 024362 005267 153204          INC    MMRO ;TURN ON MEMORY MANAGEMENT
```



```

1843 024366 005727 060000 11$: TST #60000 ;DSTM=2 SOP NON-MOD
1844 024372 005737 100000 TST @#100000 ;DSTM=3 SOP NON-MOD
1845 024376 005767 053376 TST 100000 ;DSTM=6 SOP NON-MOD
1846 024402 005777 053372 TST @100000 ;DSTM=7 SOP NON-MOD
1847 024406 005067 153160 CLR MMR0 ;TURN OFF MEMORY MANAGEMENT
1848 :* TEST SOB MOD WITH DSTM=2,3,6,7; DSTF=7
1849 :*
1850 024412 012767 024432 154470 MOV #12$, $LPERR ;SET LOOP ON ERROR POINTER TO 12$
1851 024420 012737 100000 060002 MOV #100000, @#60002 ;SET UP TEST VALUES
1852 024426 005267 153140 INC MMR0 ;TURN ON MEMORY MANAGEMENT
1853 024432 005027 060000 12$: CLR #60000 ;DSTM=2 SOP MOD
1854 024436 005037 100000 CLR @#100000 ;DSTM=3 SOP MOD
1855 024442 005067 053332 CLR 100000 ;DSTM=6 SOP MOD
1856 024446 005077 053330 CLR @100002 ;DSTM=7 SOP MOD
1857 024452 005067 153114 CLR MMR0 ;TURN OFF MEMORY MANAGEMENT
1858 :*
1859 :* THE NEXT THREE TESTS ARE CONCERNED WITH TESTING
1860 :* DOP AND NOT(SRCM=DSTM=0)
1861 :*
1862 :* TEST DOP DEST NON-MOD SRCM=DSTM=2,3,6,7; SRCF=DSTF=7
1863 :*
1864 024456 012767 024476 154424 MOV #13$, $LPERR ;SET LOOP ON ERROR POINTER TO 13$
1865 024464 012737 100000 060002 MOV #100000, @#60002 ;SET UP TEST VALUE
1866 024472 005267 153074 INC MMR0 ;TURN ON MEMORY MANAGEMENT
1867 024476 022727 060000 060000 13$: CMP #60000, #60000 ;SRCM=2 DSTM=2 DOP NON-MOD
1868 024504 023737 100000 100000 CMP @#100000, @#100000 ;SRCM=3 DSTM=3 DOP NON-MOD
1869 024512 026767 053262 053260 CMP 100000, 100000 ;SRCM=6 DSTM=6 DOP NON-MOD
1870 024520 027777 053256 053254 CMP @100002, @100002 ;SRCM=7 DSTM=7 DOP NON-MOD
1871 024526 005067 153040 CLR MMR0 ;TURN OFF MEMORY MANAGEMENT
1872 :* TEST MOV DEST AND NOT(SRCM=DSTM=0)
1873 :* SRCM=DSTM=2,3,6,7; SRCF=DSTF=7
1874 :*
1875 024532 012767 024552 154350 MOV #14$, $LPERR ;SET LOOP ON ERROR POINTER TO 14$
1876 024540 012737 100000 060002 MOV #100000, @#60002 ;SET UP TEST VALUE
1877 024546 005267 153020 INC MMR0 ;TURN ON MEMORY MANAGEMENT
1878 024552 012727 060002 060002 14$: MOV #60002, #60002 ;SRCM=2 DSTM=2 MOV
1879 024560 012737 060000 100000 MOV #60000, @#100000 ;SRCM=2 DSTM=3 MOV
1880 024566 012767 060000 053204 MOV #60000, 100000 ;SRCM=2 DSTM=6 MOV
1881 024574 012777 060000 053200 MOV #60000, @100002 ;SRCM=2 DSTM=7 MOV
1882 024602 005067 152764 CLR MMR0 ;TURN OFF MEMORY MANAGEMENT
1883 :* TEST DOP DEST MOD AND NOT SUB
1884 :* SRCM=DSTM=2,3,6,7; SRCF=DSTF=7
1885 :*
1886 024606 012767 024620 154274 MOV #15$, $LPERR ;SET LOOP ON ERROR POINTER TO 15$
1887 024614 005267 152752 INC MMR0 ;TURN ON MEMORY MANAGEMENT
1888 024620 052727 060000 060000 15$: BIS #60000, #60000 ;SRCM=2 DSTM=2 DOP MOD
1889 024626 052737 000000 100000 BIS #00000, @#100000 ;SRCM=2 DSTM=3 DOP MOD
1890 024634 052767 000000 053136 BIS #00000, 100000 ;SRCM=2 DSTM=6 DOP MOD
1891 024642 052777 000000 053132 BIS #00000, @100002 ;SRCM=2 DSTM=7 DOP MOD
1892 024650 005067 152716 CLR MMR0 ;TURN OFF MEMORY MANAGEMENT
1893 :* TEST SWAB WITH DSTM=2,3,6,7; DSTF=7
1894 :*
1895 024654 012767 024674 154226 MOV #16$, $LPERR ;SET LOOP ON ERROR POINTER TO 16$
1896 024662 012737 100002 060000 MOV #100002, @#60000 ;SET UP TEST VALUES
1897 024670 005267 152676 INC MMR0 ;TURN ON MEMORY MANAGEMENT
1898 024674 000327 060000 16$: SWAB #60000 ;DSTM=2 SWAB
1899 024700 000337 100002 SWAB @#100002 ;DSTM=3 SWAB
  
```

```

1900 024704 000367 053072          SWAB 100002          ;DSTM=6 SWAB
1901 024710 000377 053064          SWAB @100000        ;DSTM=7 SWAB
1902 024714 005067 152652          CLR  MMR0           ;TURN OFF MEMORY MANAGEMENT
1903          :*          TEST ROT/SHFT WITH DSTM=2,3,6,7; DSTF=7
1904          :*
1905 024720 012767 024740 154162    MOV  #17$,SLPERR    ;SET LOOP ON ERROR POINTER TO 17$
1906 024726 012737 100000 060002    MOV  #100000,@#60002 ;SET UP TEST VALUES
1907 024734 005267 152632          INC  MMR0           ;TURN ON MEMORY MANAGEMENT
1908 024740 006127 060000          17$: ROL  #60000       ;DSTM=2 ROT/SHFT
1909 024744 006137 100000          ROL  @#100000      ;DSTM=3 ROT/SHFT
1910 024750 006167 053024          ROL  100000        ;DSTM=6 ROT/SHFT
1911 024754 006177 053022          ROL  @100002       ;DSTM=7 ROT/SHFT
1912 024760 005067 152606          CLR  MMR0           ;TURN OFF MEMORY MANAGEMENT
1913          :*          TEST ASH/ASHC WITH DSTM=2,3,6,7; DSTF=7
1914          :*
1915 024764 012767 025012 154116    MOV  #18$,SLPERR    ;SET LOOP ON ERROR POINTER TO 18$
1916 024772 012737 000001 060000    MOV  #1,@#60000    ;SET UP TEST VALUES
1917 025000 012737 100000 060002    MOV  #100000,@#60002
1918 025006 005267 152560          INC  MMR0           ;TURN ON MEMORY MANAGEMENT
1919 025012 072027 000001          18$: ASH #1,R0       ;DSTM=2 ASH/ASHC
1920 025016 072037 100000          ASH  @#100000,R0   ;DSTM=3 ASH/ASHC
1921 025022 072067 052752          ASH  100000,R0     ;DSTM=6 ASH/ASHC
1922 025026 072077 052750          ASH  @100002,R0    ;DSTM=7 ASH/ASHC
1923 025032 005067 152534          CLR  MMR0           ;TURN OFF MEMORY MANAGEMENT
1924          :*          TEST MUL/DIV WITH DSTM=2,3,6,7; DSTF=7
1925          :*
1926 025036 012767 025050 154044    MOV  #19$,SLPERR    ;SET LOOP ON ERROR POINTER TO 19$
1927 025044 005267 152522          INC  MMR0           ;TURN ON MEMORY MANAGEMENT
1928 025050 070027 000002          19$: MUL #2,R0       ;DSTM=2 MUL/DIV
1929 025054 070037 100000          MUL  @#100000,R0   ;DSTM=3 MUL/DIV
1930 025060 070067 052714          MUL  100000,R0     ;DSTM=6 MUL/DIV
1931 025064 070077 052712          MUL  @100002,R0    ;DSTM=7 MUL/DIV
1932 025070 005067 152476          CLR  MMR0           ;TURN OFF MEMORY MANAGEMENT
1933          :*          TEST JMP WITH DSTM=3,6,7; DSTF=7
1934          :*
1935 025074 012767 025114 154006    MOV  #20$,SLPERR    ;SET LOOP ON ERROR POINTER TO 20$
1936 025102 012737 025130 060000    MOV  #23$,@#60000  ;SET UP TEST VALUES
1937 025110 005267 152456          INC  MMR0           ;TURN ON MEMORY MANAGEMENT
1938 025114 000137 025120          20$: JMP @#21$       ;DSTM=3 JMP
1939 025120 000167 000000          21$: JMP 22$        ;DSTM=6 JMP
1940 025124 000177 052650          22$: JMP @100000    ;DSTM=7 JMP
1941 025130 005067 152436          23$: CLR MMR0       ;TURN OFF MEMORY MANAGEMENT
1942          :*          TEST SUB WITH DSTM=2,3,6,7; DSTF=7
1943          :*
1944 025134 012767 025156 153746    MOV  #28$,SLPERR    ;SET LOOP ON ERROR POINTER TO 28$
1945 025142 005000          CLR  R0             ;SET UP TEST VALUES
1946 025144 012737 100000 060002    MOV  #100000,@#60002
1947 025152 005267 152414          INC  MMR0           ;TURN ON MEMORY MANAGEMENT
1948 025156 160027 060002          28$: SUB R0,#60002   ;DSTM=2 SUB
1949 025162 160037 100000          SUB  R0,@#100000   ;DSTM=3 SUB
1950 025166 160067 052606          SUB  R0,100000     ;DSTM=6 SUB
1951 025172 160077 052604          SUB  R0,@100002    ;DSTM=7 SUB
1952 025176 005067 152370          CLR  MMR0           ;TURN OFF MEMORY MANAGEMENT
1953 025202 012767 024312 153700    MOV  #10$,SLPERR    ;SET LOOP ON ERROR POINTER TO START OF TEST
1964          :*
:*****
:*TEST 15          ENABLE D-SPACE AND SEE I-SPACE IS NOT FORCED
:*

```

:* THIS TEST SHOWS THAT I-SPACE IS NOT FORCED IF THE REGISTER FIELD
 IS NOT 7, BUT THE OTHER CONDITIONS ARE MET.

:* ALL ERRORS FOUND IN THIS TEST ARE REPORTED WHEN THE CPU ABORTS
 THROUGH 'MMVEC' TO SUBROUTINE 'MODSPAC'. THIS SUBROUTINE WILL
 REPORT THAT D-SPACE WAS NOT ENABLED PROPERLY.

```

025210 000004
1965 025212 012700 077406
1966 025216 010067 145066
1967 025222 010067 145074
1968 025226 010067 145072
1969 025232 010067 145070
1970 025236 105067 145044
1971
1972
1973 025242 012700 060000
1974 025246 010037 060000
1975 025252 012737 060002 060002
1976 025260 012737 060004 060004
1977 025266 012737 060006 060006
1978 025274 012767 025336 153606
1979 025302 012737 060000 060000
1980 025310 012737 060002 060002
1981 025316 012737 060004 060004
1982 025324 012737 060006 060006
1983 025332 005267 152234
1984 025336 005710
1985 025340 005720
1986 025342 005730
1987 025344 005750
1988 025346 005770 000000
1989 025352 005067 152214
1990
1991
1992 025356 012767 025370 153524
1993 025364 005267 152202
1994 025370 005010
1995 025372 005020
1996 025374 005030
1997 025376 005050
1998 025400 005070 000000
1999 025404 005067 152162
2000
2001
2002
2003 025410 012767 025452 153472
2004 025416 012702 000032
2005 025422 012700 060000
2006 025426 012701 060000
2007 025432 010021
2008 025434 062700 000002
2009 025440 077204
2010 025442 012700 060000
2011 025446 005267 152120
2012 025452 021010
  
```

```

TST15: SCOPE
MOV #77406,R0
MOV R0,KIPDR4 ;MAKE KIPDR4 R/W,4K,200 BLOCKS
MOV R0,KDPDR1 ;MAKE KDPDR1 R/W,4K,200 BLOCKS
MOV R0,KDPDR2 ;MAKE KDPDR2 R/W,4K,200 BLOCKS
MOV R0,KDPDR3 ;MAKE KDPDR3 R/W,4K,200 BLOCKS
CLRB KIPDR3 ;MAKE KIPDR3 NON-RESIDENT
;* TEST SOP NON-MOD; DSTM=1,2,3,5,7
;*
20$: MOV #60000,R0 ;SET UP CONSTANTS FOR TEST
MOV R0,@#60000
MOV #60002,@#60002
MOV #60004,@#60004
MOV #60006,@#60006
MOV #1$,SLPERR ;SET LOOP ON ERROR POINTER TO 1$
MOV #60000,@#60000
MOV #60002,@#60002
MOV #60004,@#60004
MOV #60006,@#60006
1$: INC MMRO ;TURN ON MEMORY MANAGEMENT
TST (R0) ;DSTM=1 SOP NON-MOD
TST (R0)+ ;DSTM=2 SOP NON-MOD
TST @(R0)+ ;DSTM=3 SOP NON-MOD
TST @-(R0) ;DSTM=5 SOP NON-MOD
TST @0(R0) ;DSTM=7 SOP NON-MOD
CLR MMRO ;TURN OFF MEMORY MANAGEMENT
;* TEST SOP MOD; DSTM=1,2,3,5,7
;*
MOV #2$,SLPERR ;SET LOOP ON ERROR POINTER TO 2$
INC MMRO ;TURN ON MEMORY MANAGEMENT
2$: CLR (R0) ;DSTM=1 SOP MOD
CLR (R0)+ ;DSTM=2 SOP MOD
CLR @(R0)+ ;DSTM=3 SOP MOD
CLR @-(R0) ;DSTM=5 SOP MOD
CLR @0(R0) ;DSTM=7 SOP MOD
CLR MMRO ;TURN OFF MEMORY MANAGEMENT
;* TEST DOP DEST NON-MOD WITH SRCM=1,2,3,5,7 AND DSTM=1,2,3,5,7
;* ALL SOURCE MODES TO BE TESTED ARE TESTED HERE
;*
MOV #3$,SLPERR ;SET LOOP ON ERROR POINTER TO 3$
MOV #32,R2 ;SET UP ADDRESSES 60000-60064 FOR TEST
MOV #60000,R0
MOV #60000,R1
21$: MOV R0,(R1)+
ADD #2,R0
SOB R2,21$
MOV #60000,R0
3$: INC MMRO ;TURN ON MEMORY MANAGEMENT
CMP (R0),(R0) ;SRCM=1 DSTM=1 DOP DEST NON-MOD
  
```

```

2013 025454 021020      CMP      (R0),(R0)+      ;SRCM=1 DSTM=2 DOP DEST NON-MOD
2014 025456 021030      CMP      (R0),@(R0)+    ;SRCM=1 DSTM=3 DOP DEST NON-MOD
2015 025460 021050      CMP      (R0),@(R0)-    ;SRCM=1 DSTM=5 DOP DEST NON-MOD
2016 025462 021070 000000  CMP      (R0),@0(R0)    ;SRCM=1 DSTM=7 DOP DEST NON-MOD
2017 025466 022010      CMP      (R0)+,(R0)     ;SRCM=2 DSTM=1 DOP DEST NON-MOD
2018 025470 022020      CMP      (R0)+,(R0)+    ;SRCM=2 DSTM=2 DOP DEST NON-MOD
2019 025472 022030      CMP      (R0)+,@(R0)+   ;SRCM=2 DSTM=3 DOP DEST NON-MOD
2020 025474 022050      CMP      (R0)+,@(R0)-   ;SRCM=2 DSTM=5 DOP DEST NON-MOD
2021 025476 022070 000000  CMP      (R0)+,@0(R0)   ;SRCM=2 DSTM=7 DOP DEST NON-MOD
2022 025502 023010      CMP      @(R0)+,(R0)    ;SRCM=3 DSTM=1 DOP DEST NON-MOD
2023 025504 023020      CMP      @(R0)+,(R0)+   ;SRCM=3 DSTM=2 DOP DEST NON-MOD
2024 025506 023030      CMP      @(R0)+,@(R0)+   ;SRCM=3 DSTM=3 DOP DEST NON-MOD
2025 025510 023050      CMP      @(R0)+,@(R0)-   ;SRCM=3 DSTM=5 DOP DEST NON-MOD
2026 025512 023070 000000  CMP      @(R0)+,@0(R0)   ;SRCM=3 DSTM=7 DOP DEST NON-MOD
2027 025516 025010      CMP      @-(R0),(R0)    ;SRCM=5 DSTM=1 DOP DEST NON-MOD
2028 025520 025020      CMP      @-(R0),(R0)+   ;SRCM=5 DSTM=2 DOP DEST NON-MOD
2029 025522 025030      CMP      @-(R0),@(R0)+   ;SRCM=5 DSTM=3 DOP DEST NON-MOD
2030 025524 025050      CMP      @-(R0),@(R0)-   ;SRCM=5 DSTM=5 DOP DEST NON-MOD
2031 025526 025070 000000  CMP      @-(R0),@0(R0)   ;SRCM=5 DSTM=7 DOP DEST NON-MOD
2032 025532 027010 000000  CMP      @0(R0),(R0)    ;SRCM=7 DSTM=1 DOP DEST NON-MOD
2033 025536 027020 000000  CMP      @0(R0),(R0)+   ;SRCM=7 DSTM=2 DOP DEST NON-MOD
2034 025542 027030 000000  CMP      @0(R0),@(R0)+   ;SRCM=7 DSTM=3 DOP DEST NON-MOD
2035 025546 027050 000000  CMP      @0(R0),@(R0)-   ;SRCM=7 DSTM=5 DOP DEST NON-MOD
2036 025552 027070 000000 000000  CMP      @0(R0),@0(R0)   ;SRCM=7 DSTM=7 DOP DEST NON-MOD
2037 025560 005067 152006  CLR      MMR0          ;TURN OFF MEMORY MANAGEMENT
2038          ;*          TEST DOP DEST MOD AND NOT SUB; DSTM=1,2,3,5,7
2039          ;*
2040 025564 005000      CLR      R0            ;SET UP CONSTANTS FOR TEST
2041 025566 012701 060000  MOV      #60000,R1     ;SET UP CONSTANTS FOR TEST
2042 025572 012767 025604 153310  MOV      #4$, $LPERR   ;SET LOOP ON ERROR POINTER TO 4$
2043 025600 005267 151766  INC      MMR0          ;TURN ON MEMORY MANAGEMENT
2044 025604 050011 4$:    BIS      R0,(R1)      ;DSTM=1 DOP DEST MOD
2045 025606 050021      BIS      R0,(R1)+     ;DSTM=2 DOP DEST MOD
2046 025610 050031      BIS      R0,@(R1)+    ;DSTM=3 DOP DEST MOD
2047 025612 050051      BIS      R0,@-(R1)    ;DSTM=5 DOP DEST MOD
2048 025614 050071 000000  BIS      R0,@0(R1)    ;DSTM=7 DOP DEST MOD
2049 025620 005067 151746  CLR      MMR0          ;TURN OFF MEMORY MANAGEMENT
2050          ;*          TEST MOV DEST AND NOT(SM0 AND DM0); DSTM=3,5,7
2051          ;*
2052 025624 012701 060000  MOV      #60000,R1     ;SET UP CONSTANTS FOR TEST
2053 025630 012737 060002 060000  MOV      #60002,@#60000 ;SET UP CONSTANTS FOR TEST
2054 025636 012767 025650 153244  MOV      #5$, $LPERR   ;SET LOOP ON ERROR POINTER TO 5$
2055 025644 005267 151722  INC      MMR0          ;TURN ON MEMORY MANAGEMENT
2056 025650 010031 5$:    MOV      R0,@(R1)+    ;DSTM=3 MOV DEST
2057 025652 010051      MOV      R0,@-(R1)    ;DSTM=5 MOV DEST
2058 025654 010071 000000  MOV      R0,@0(R1)    ;DSTM=7 MOV DEST
2059 025660 005067 151706  CLR      MMR0          ;TURN OFF MEMORY MANAGEMENT
2060          ;*          TEST SWAB; DSTM=1,2,3,5,7
2061          ;*
2062 025664 012701 060000  MOV      #60000,R1     ;SET UP CONSTANTS FOR TEST
2063 025670 012767 025702 153212  MOV      #6$, $LPERR   ;SET LOOP ON ERROR POINTER TO 6$
2064 025676 005267 151670  INC      MMR0          ;TURN ON MEMORY MANAGEMENT
2065 025702 000311 6$:    SWAB     (R1)         ;DSTM=1 SWAB
2066 025704 000321      SWAB     (R1)+        ;DSTM=2 SWAB
2067 025706 000331      SWAB     @(R1)+       ;DSTM=3 SWAB
2068 025710 000351      SWAB     @-(R1)       ;DSTM=5 SWAB
2069 025712 000371 000000  SWAB     @0(R1)       ;DSTM=7 SWAB
    
```

```

2070 025716 005067 151650          CLR      MMR0          ;TURN OFF MEMORY MANAGEMENT
2071          :*          TEST ROT/SHFT; DSTM=1,2,3,5,7
2072          :*
2073 025722 012701 060000          MOV      #60000,R1     ;SET UP CONSTANTS FOR TEST
2074 025726 012702 060006          MOV      #60006,R2
2075 025732 010203          MOV      R2,R3
2076 025734 012737 060000 060000  MOV      #60000,@#60000
2077 025742 012737 060002 060002  MOV      #60002,@#60002
2078 025750 012737 060004 060004  MOV      #60004,@#60004
2079 025756 012737 060006 060006  MOV      #60006,@#60006
2080 025764 012767 025776 153116  MOV      #7$, $LPERR   ;SET LOOP ON ERROR POINTER TO 7$
2081 025772 005267 151574          INC      MMR0          ;TURN ON MEMORY MANAGEMENT
2082 025776 006111          7$: ROL      (R1)       ;DSTM=1 ROT/SHFT
2083 026000 006121          ROL      (R1)+        ;DSTM=2 ROT/SHFT
2084 026002 006131          ROL      @ (R1)+      ;DSTM=3 ROT/SHFT
2085 026004 006152          ROL      @-(R2)       ;DSTM=5 ROT/SHFT
2086 026006 006173 000000          ROL      @0(R3)       ;DSTM=7 ROT/SHFT
2087 026012 005067 151554          CLR      MMR0          ;TURN OFF MEMORY MANAGEMENT
2088          :*          TEST MUL/DIV; DSTM=1,2,3,5,7
2089          :*
2090 026016 012767 026032 153064  MOV      #8$, $LPERR   ;SET LOOP ON ERROR POINTER TO 8$
2091 026024 005003          CLR      R3           ;SET UP CONSTANT FOR TEST
2092 026026 005267 151540          INC      MMR0          ;TURN ON MEMORY MANAGEMENT
2093 026032 070310          8$: MUL      (R0),R3    ;DSTM=1 MUL/DIV
2094 026034 070320          MUL      (R0)+,R3     ;DSTM=2 MUL/DIV
2095 026036 070330          MUL      @ (R0)+,R3   ;DSTM=3 MUL/DIV
2096 026040 070350          MUL      @-(R0),R3   ;DSTM=5 MUL/DIV
2097 026042 070370 000000          MUL      @0(R0),R3   ;DSTM=7 MUL/DIV
2098 026046 005067 151520          CLR      MMR0          ;TURN OFF MEMORY MANAGEMENT
2099          :*          TEST ASH/ASHC; DSTM=1,2,3,5,7
2100          :*
2101 026052 012737 000001 060000  MOV      #1,@#60000    ;SET UP CONSTANTS FOR THE TEST
2102 026060 012737 060000 060002  MOV      #60000,@#60002
2103 026066 012700 060000          MOV      #60000,R0
2104 026072 012767 026104 153010  MOV      #9$, $LPERR   ;SET LOOP ON ERROR POINTER TO 9$
2105 026100 005267 151466          INC      MMR0          ;TURN ON MEMORY MANAGEMENT
2106 026104 072310          9$: ASH      (R0),R3    ;DSTM=1 ASH/ASHC
2107 026106 072320          ASH      (R0)+,R3     ;DSTM=2 ASH/ASHC
2108 026110 072330          ASH      @ (R0)+,R3   ;DSTM=3 ASH/ASHC
2109 026112 072350          ASH      @-(R0),R3   ;DSTM=5 ASH/ASHC
2110 026114 072370 000000          ASH      @0(R0),R3   ;DSTM=7 ASH/ASHC
2111 026120 005067 151446          CLR      MMR0          ;TURN OFF MEMORY MANAGEMENT
2112          :*          TEST JMP; DSTM=3,7
2113          :*
2114 026124 012767 026156 152756  MOV      #10$, $LPERR  ;SET LOOP ON ERROR POINTER TO 10$
2115 026132 012737 026160 060000  MOV      #11$,@#60000  ;SET UP CONSTANTS FOR THE TEST
2116 026140 012737 026164 060002  MOV      #12$,@#60002
2117 026146 012701 060000          MOV      #60000,R1
2118 026152 005267 151414          INC      MMR0          ;TURN ON MEMORY MANAGEMENT
2119 026156 000131          10$: JMP      @ (R1)+     ;DSTM=3 JMP
2120 026160 000171 000000          11$: JMP      @0(R1)    ;DSTM=7 JMP
2121 026164 005067 151402          12$: CLR      MMR0          ;TURN OFF MEMORY MANAGEMENT
2122          :*          TEST JSR; DSTM=3,7
2123          :*
2124 026170 012767 026222 152712  MOV      #13$, $LPERR  ;SET LOOP ON ERROR POINTER TO 13$
2125 026176 012737 026224 060000  MOV      #14$,@#60000  ;SET UP CONSTANTS FOR THE TEST
2126 026204 012737 026230 060002  MOV      #15$,@#60002

```

```

2127 026212 012701 060000      MOV    #60000,R1
2128 026216 005267 151350      INC    MMR0           ;TURN ON MEMORY MANAGEMENT
2129 026222 004731           13$:  JSR    PC,@(R1)+   ;DSTM=3 JSR
2130 026224 004771 000000      14$:  JSR    PC,@(R1)   ;DSTM=7 JSR
2131 026230 005067 151336      15$:  CLR    MMR0           ;TURN OFF MEMORY MANAGEMENT
2132 026234 112767 000006 144044  MOVB   #6,KIPDR3     ;MAKE KIPDR3 RESIDENT
2133 026242 012706 001100      MOV    #KERSTK,KSP   ;RESET STACK POINTER
2134 026246 012767 025242 152634  MOV    #20$,$LPERR   ;SET LOOP ON ERROR POINTER TO START OF TEST
2135
2144
  
```

```

:*****
:TEST 16      PROPER ENABLING OF SUPER. D-SPACE
:
:      THIS TEST CHECKS FOR PROPER ENABLING OF THE SUPERVISOR D-SPACE.
:
:      ANY ERRORS ENCOUNTERED WILL BE REPORTED THROUGH 'MMVEC' TO
:      SUBROUTINE 'NODSPAC'.
:*****
  
```

```

2145 026254 000004           TST16: SCOPE
2146 026256 105067 143740      20$:  CLRB   SDPDR1       ;MAKE SDPDR1 NON-RESIDENT
2147 026262 012767 026310 152620  MOV    #1$,$LPERR     ;SET LOOP ON ERROR POINTER TO 1$
2148 026270 012767 000022 144220  MOV    #22,MMR3       ;ENABLE 22-BIT SUPERVISOR D-SPACE
2149 026276 052767 040000 151472  BIS    #40000,PSW     ;ENABLE SUPERVISOR MODE
2150 026304 005267 151262      INC    MMR0           ;TURN ON MEMORY MANAGEMENT
2151           ;*      THE NEXT INSTRUCTIONS SHOULD NEVER INVOKE D-SPACE
2152 026310 000400           1$:  BR     2$
2153 026312 005700           2$:  TST    R0
2154 026314 005200           INC    R0
2155 026316 005067 151250      CLR    MMR0           ;TURN OFF MEMORY MANAGEMENT
2156 026322 112767 000006 143672  MOVB   #6,SDPDR1     ;MAKE SDPDR1 RESIDENT
2157 026330 105067 143652      CLRB   SIPDR3        ;MAKE SIPDR3 NON-RESIDENT
2158           ;*      TEST SOP INSTRUCTIONS
2159           ;*
2160 026334 012767 026346 152546  MOV    #3$,$LPERR     ;SET LOOP ON ERROR POINTER TO 3$
2161 026342 005267 151224      INC    MMR0           ;TURN ON MEMORY MANAGEMENT
2162 026346 005737 060000      3$:  TST    @#60000      ;DSTM=3 DSTF=7 SOP NON-MOD
2163 026352 005037 060000      CLR    @#60000      ;DSTM=3 DSTF=7 SOP MOD
2164 026356 005067 151210      CLR    MMR0           ;TURN OFF MEMORY MANAGEMENT
2165           ;*      TEST DOP INSTRUCTIONS
2166           ;*
2167 026362 012767 026400 152520  MOV    #4$,$LPERR     ;SET LOOP ON ERROR POINTER TO 4$
2168 026370 012700 060000      MOV    #60000,R0      ;SET UP CONSTANT FOR TEST
2169 026374 005267 151172      INC    MMR0           ;TURN ON MEMORY MANAGEMENT
2170 026400 023710 060000      4$:  CMP    @#60000,(R0)   ;SRCM=3 DSTM=1 DOP DEST NON-MOD
2171 026404 052730 000000      BIS    #0,@(R0)+     ;SRCM=2 DSTM=3 DOP DEST MOD
2172 026410 013737 060002 060002  MOV    @#60002,@#60002 ;SRCM=3 DSTM=3 MOV DEST
2173 026416 005067 151150      CLR    MMR0           ;TURN OFF MEMORY MANAGEMENT
2174           ;*      TEST ROT/SHFT AND ASH/ASHC INSTRUCTIONS
2175           ;*
2176 026422 012767 026446 152460  MOV    #5$,$LPERR     ;SET LOOP ON ERROR POINTER TO 5$
2177 026430 012737 000001 060000  MOV    #1,@#60000     ;SET CONSTANT FOR TEST
2178 026436 012700 000001      MOV    #1,R0          ;SET UP CONSTANT FOR TEST
2179 026442 005267 151124      INC    MMR0           ;TURN ON MEMORY MANAGEMENT
2180 026446 006137 060000      5$:  ROL    @#60000      ;DSTM=3 ROT/SHFT
2181 026452 072337 060000      ASH    @#60000,R3     ;DSTM=3 ASH/ASHC
2182 026456 005067 151110      CLR    MMR0           ;TURN OFF MEMORY MANAGEMENT
  
```

```

2183 :* TEST MUL/DIV AND SWAB INSTRUCTIONS
2184 :*
2185 026462 012767 026474 152420 :* MOV #6$, $LPERR ;SET LOOP ON ERROR POINTER TO 6$
2186 026470 005267 151076 :* INC MMR0 ;TURN ON MEMORY MANAGEMENT
2187 026474 070037 060000 6$: * MUL @#60000,R0 ;DSTM=3 MUL/DIV
2188 026500 000337 060004 :* SWAB @#60004 ;DSTM=3 SWAB
2189 026504 005067 151062 :* CLR MMR0 ;TURN OFF MEMORY MANAGEMENT
2190 026510 042767 000002 144000 :* BIC #2,MMR3 ;DISABLE SUPERVISOR D-SPACE
2191 026516 112767 000006 143462 :* MOVB #6,SIPDR3 ;MAKE SIPDR3 RESIDENT
2192 026524 012767 026256 152356 :* MOV #20$, $LPERR ;SET LOOP ON ERROR POINTER TO START OF TEST
2193
2202
  
```

```

:*****
:TEST 17 PROPER ENABLING OF USER D-SPACE
:
: THIS TEST CHECKS FOR PROPER ENABLING OF THE USER D-SPACE.
:
: ANY ERRORS ENCOUNTERED WILL BE REPORTED THROUGH 'MMVEC' TO
: SUBROUTINE 'NODSPAC'.
:*****
  
```

```

2203 026532 000004 :* TST17: SCOPE
2204 026534 105067 151062 20$: * CLRB UDPDR1 ;MAKE UDPDR1 NON-RESIDENT
2205 026540 012767 026566 152342 :* MOV #1$, $LPERR ;SET LOOP ON ERROR POINTER TO 1$
2206 026546 012767 000021 143742 :* MOV #21,MMR3 ;ENABLE 22-BIT USER D-SPACE
2207 026554 052767 140000 151214 :* BIS #140000,PSW ;ENABLE USER MODE
2208 :* INC MMR0 ;TURN ON MEMORY MANAGEMENT
2209 :* THE NEXT INSTRUCTIONS SHOULD NEVER INVOKE D-SPACE
2210 026566 000400 1$: * BR 2$
2211 026570 005700 2$: * TST R0
2212 026572 005200 :* INC R0
2213 026574 005067 150772 :* CLR MMR0 ;TURN OFF MEMORY MANAGEMENT
2214 026600 112767 000006 151014 :* MOVB #6,UDPDR1 ;MAKE UDPDR1 RESIDENT
2215 026606 105067 150774 :* CLRB UIPDR3 ;MAKE UIPDR3 NON-RESIDENT
2216 :* TEST SOP INSTRUCTIONS
2217 :*
2218 026612 012767 026624 152270 :* MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
2219 026620 005267 150746 :* INC MMR0 ;TURN ON MEMORY MANAGEMENT
2220 026624 005737 060000 3$: * TST @#60000 ;DSTM=3 DSTF=7 SOP NON-MOD
2221 026630 005037 060000 :* CLR @#60000 ;DSTM=3 DSTF=7 SOP MOD
2222 026634 005067 150732 :* CLR MMR0 ;TURN OFF MEMORY MANAGEMENT
2223 :* TEST DOP INSTRUCTIONS
2224 :*
2225 026640 012767 026656 152242 :* MOV #4$, $LPERR ;SET LOOP ON ERROR POINTER TO 4$
2226 026646 012700 060000 :* MOV #60000,R0 ;SET UP CONSTANT FOR TEST
2227 026652 005267 150714 :* INC MMR0 ;TURN ON MEMORY MANAGEMENT
2228 026656 023710 060000 4$: * CMP @#60000,(R0) ;SRCM=3 DSTM=1 DOP DEST NON-MOD
2229 026662 052730 000000 :* BIS #0,@(R0)+ ;SRCM=2 DSTM=3 DOP DEST MOD
2230 026666 013737 060002 060002 :* MOV @#60002,@#60002 ;SRCM=3 DSTM=3 MOV DEST
2231 026674 005067 150672 :* CLR MMR0 ;TURN OFF MEMORY MANAGEMENT
2232 :* TEST ROT/SHFT AND ASH/ASHC INSTRUCTIONS
2233 :*
2234 026700 012767 026720 152202 :* MOV #5$, $LPERR ;SET LOOP ON ERROR POINTER TO 5$
2235 026706 012737 000001 060000 :* MOV #1,@#60000 ;SET CONSTANT FOR TEST
2236 026714 005267 150652 :* INC MMR0 ;TURN ON MEMORY MANAGEMENT
2237 026720 006137 060000 5$: * ROL @#60000 ;DSTM=3 ROT/SHFT
2238 026724 072337 060000 :* ASH @#60000,R3 ;DSTM=3 ASH/ASHC
  
```

```

2239 026730 005067 150636          CLR      MMR0          ;TURN OFF MEMORY MANAGEMENT
2240          : *      TEST MUL/DIV AND SWAB INSTRUCTIONS
2241          : *
2242 026734 012767 026752 152146    MOV      #6$,SLPERR   ;SET LOOP ON ERROR POINTER TO 6$
2243 026742 012700 000001          MOV      #1,R0        ;SET UP CONSTANT FOR TEST
2244 026746 005267 150620          INC      MMR0         ;TURN ON MEMORY MANAGEMENT
2245 026752 070037 060000 6$      MUL      @#60000,R0   ;DSTM=3 MUL/DIV
2246 026756 000337 060004          SWAB    @#60004      ;DSTM=3 SWAB
2247 026762 005067 150604          CLR      MMR0         ;TURN OFF MEMORY MANAGEMENT
2248 026766 042767 000001 143522    BIC     #1,MMR3       ;DISABLE USER D-SPACE
2249 026774 112767 000006 150604    MOVB    #6,UIPDR3    ;MAKE UIPDR3 RESIDENT
2250 027002 012767 026534 152100    MOV     #20$,SLPERR  ;SET LOOP ON ERROR POINTER TO START OF TEST
2251 027010 005067 150762          CLR      PSW         ;RESET TO KERNAL SPACE
2252
2261
  
```

```

:*****
:*TEST 20      TRAPPING IN D-SPACE KERNAL MODE
:
:  THIS TEST VERIFIES THAT THE ABORT VECTOR IS TAKEN FROM
:  D-SPACE AND NOT I-SPACE. THE I-SPACE VECTOR POINTS TO
:  10$ AND THE D-SPACE VECTOR POINTS TO 15$. EACH PSW IN
:  VIRTUAL 252 IS DIFFERENT SO THE PROGRAM CAN TELL WHICH
:  AREA IT IS PICKED UP FROM.
:*****
  
```

```

027014 000004
2262 027016 004767 153274          JSR     PC,TOFF      ;TURN OF T-BIT FOR THIS TEST
2263 027022 012767 027056 152060 20$:  MOV     #1$,SLPERR   ;SET LOOP ON ERROR POINTER TO 1$
2264 027030 012737 027124 000250    MOV     #10$,@MMVEC  ;SET M.M. VEC. TO HOLD BAD VECTOR
2265 027036 005037 000252          CLR     @MMVEC+2    ;PSW IN 252 HAS PRIORITY OF ZERO
2266 027042 012737 027132 000350    MOV     #15$,@#350   ;SET D-SPACE M.M VECTOR TO 350
2267 027050 012737 000340 000352    MOV     #340,@#352   ;SET PSW PRIORITY TO 7
2268 027056 012706 001000 1$:    MOV     #1000,KSP    ;SET UP KERNAL VECTOR
2269 027062 005267 150504          INC     MMR0         ;TURN ON MEMORY MANAGEMENT
2270          ;NOW SET UP FOR AN ABORT IN KERNAL MODE WITH D-SPACE ENABLED
2271          :
2272 027066 012767 077402 143234    MOV     #77402,KDPDR4 ;KERNAL D-SPACE PAGE 4 IS READ ONLY
2273 027074 012767 000600 143266    MOV     #600,KDPAR4  ;MAP D-SPACE PAGE 4 TO 12K
2274 027102 052767 000004 143406    BIS     #BIT2,MMR3   ;ENABLE KERNAL D-SPACE MAPPING
2275 027110 012767 000001 143242    MOV     #1,KDPAR0    ;MAP KERNAL D PAGE 0 TO 000100
2276 027116 012737 177777 100000    MOV     #-1,@#100000 ;TRY TO WRITE TO PAGE 4
2277 027124 016700 150646 10$:    MOV     PSW,R0       ;SAVE PSW FOR COMPARE
2278 027130 000402          BR      16$         ;BRANCH TO D-SPACE READ CODE
2279 027132 016700 150640 15$:    MOV     PSW,R0       ;SAVE PSW FOR COMPARE
2280 027136 005067 143216 16$:    CLR     KDPAR0       ;REMAP KERNAL D PAGE 0 TO PHYSICAL 0
2281 027142 012706 001100          MOV     #KERSTK,KSP  ;RESET STACK POINTER AFTER D-SPACE ABORT
2282 027146 042767 000004 143342    BIC     #BIT2,MMR3   ;TURN OFF KERNAL D-SPACE ENABLE
2283 027154 016701 150412          MOV     MMR0,R1      ;SAVE MMR0 FOR COMPARE
2284 027160 016702 150410          MOV     MMR1,R2      ;SAVE MMR1 FOR COMPARE
2285 027164 016703 150406          MOV     MMR2,R3      ;SAVE MMR2 FOR COMPARE
2286 027170 122700 000340          CMPB   #340,R0       ;DID YOU PICK CORRECT PSW
2287 027174 001401          BEQ    2$           ;BRANCH IF PSW IS 340
2288 027176 104033          ERROR  +33          ;WRONG PSW PICKED IN ABORT SEQUENCE
2289 027200 022701 020031 2$:    CMP    #020031,R1   ;EXPECTING READ ONLY ABORT
2290          ;KERNAL MODE D-SPACE PAGE 4
2291 027204 001401          BEQ    3$           ;BRANCH IF CONDITION IS CORRECT
2292 027206 104027          ERROR  +27          ;WRONG M.M. ABORT CUNDITION
2293 027210 005067 150356 3$:    CLR     MMR0         ;CLEAR OFF MMR0 FOR EXIT OF TEST
2294 027214 012767 002456 151026    MOV     #MGMERR,MMVEC ;RESTORE NORMAL M.M. TRAP VECTOR
  
```



```
2295 027222 012767 000340 151022      MOV      #340,MMVEC+2      ;RESTORE TRAP PSW (PRIORITY 7)
2296 027230 012767 027022 151652      MOV      #20$,SLPERR      ;SET LOOP ON ERROR POINTER TO START OF TEST
2297
```

.....


```

2355 027366 012767 010340 150402 9$: MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
2356 027374 012702 100000 MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
2357 027400 006512 MFPI (R2) ;READ FROM PHYSICAL 60000
2358 027402 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
2359 027404 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
2360 027406 001401 BEQ 10$ ;BRANCH IF CORRECT DATA WAS FETCHED
2361 027410 104023 ERROR +23 ;WRONG DATA WAS FETCHED
2362 ;FOR TIGHTER SCOPE LOOP
2363 ;REPLACE ERROR CALL WITH
2364 ;'BR 9$' = 000766
2365 027412 10$: ;THE FOLLOWING WILL TEST DSTM=2 MFPI.
2366 027412 012767 027420 151470 MOV #11$,SLPERR ;SET LOOP ON ERROR POINTER TO 11$
2367 027420 012767 010340 150350 11$: MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
2368 027426 012702 100000 MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
2369 027432 006522 MFPI (R2)+ ;READ FROM PHYSICAL 60000
2370 027434 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
2371 027436 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
2372 027440 001401 BEQ 12$ ;BRANCH IF CORRECT DATA WAS FETCHED
2373 027442 104023 ERROR +23 ;WRONG DATA WAS FETCHED
2374 ;FOR TIGHTER SCOPE LOOP
2375 ;REPLACE ERROR CALL WITH
2376 ;'BR 11$' = 000766
2377 027444 12$: ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
2378 027444 012767 027452 151436 MOV #13$,SLPERR ;SET LOOP ON ERROR POINTER TO 13$
2379 027452 012767 010340 150316 13$: MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
2380 027460 006537 100000 MFPI @#100000 ;READ FROM PHYSICAL 60000
2381 027464 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
2382 027466 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
2383 027470 001401 BEQ 14$ ;BRANCH IF CORRECT DATA WAS FETCHED
2384 027472 104023 ERROR +23 ;WRONG DATA WAS FETCHED
2385 ;FOR TIGHTER SCOPE LOOP
2386 ;REPLACE ERROR CALL WITH
2387 ;'BR 13$' = 000767
2388 027474 14$: ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
2389 027474 012767 027502 151406 MOV #15$,SLPERR ;SET LOOP ON ERROR POINTER TO 15$
2390 027502 012767 010340 150266 15$: MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
2391 027510 012702 100002 MOV #100002,R2 ;LOAD VIRTUAL ADDRESS INTO R2
2392 027514 006542 MFPI -(R2) ;READ FROM PHYSICAL 60000
2393 027516 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
2394 027520 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
2395 027522 001401 BEQ 16$ ;BRANCH IF CORRECT DATA WAS FETCHED
2396 027524 104023 ERROR +23 ;WRONG DATA WAS FETCHED
2397 ;FOR TIGHTER SCOPE LOOP
2398 ;REPLACE ERROR CALL WITH
2399 ;'BR 15$' = 000766
2400 027526 16$: ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
2401 ;
2402 ;
2403 027526 012767 027534 151354 MOV #17$,SLPERR ;SET LOOP ON ERROR POINTER TO 17$
2404 027534 012767 010340 150234 17$: MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
2405 027542 012767 100000 151432 MOV #100000,$TMP2 ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
2406 027550 012702 001204 MOV #<$TMP2+2>,R2 ;LOAD ADDR. OF $TMP2+2 INTO R2
2407 027554 006552 MFPI @-(R2) ;READ FROM PHYSICAL 60000
2408 027556 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
2409 027560 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
2410 027562 001401 BEQ 18$ ;BRANCH IF CORRECT DATA WAS FETCHED
2411 027564 104023 ERROR +23 ;WRONG DATA WAS FETCHED
  
```

```

2412                                     :FOR TIGHTER SCOPE LOOP
2413                                     :REPLACE ERROR CALL WITH
2414                                     :'BR 17$' = 000763
2415 027566          18$:      ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
2416                                     :
2417 027566 012767 027574 151314      MOV      #19$, $LPERR      ;SET LOOP ON ERROR POINTER TO 19$
2418 027574 012767 010340 150174 19$:  MOV      #010340, PSW      ;MAKE PREVIOUS MODE SUPERVISOR
2419 027602 005002                                     CLR      R2              ;MAKE REGISTER 2 A ZERO
2420 027604 006562 100000      MFPI     100000(R2)      ;READ FROM PHYSICAL 60000
2421 027610 012601      MOV      (KSP)+, R1      ;POP KERNEL STACK INTO R1
2422 027612 020001      CMP      R0, R1          ;WAS DATA FETCHED SAME AS STORED
2423 027614 001401      BEQ      20$            ;BRANCH IF CORRECT DATA WAS FETCHED
2424 027616 104023      ERROR    +23          ;WRONG DATA WAS FETCHED
2425                                     :FOR TIGHTER SCOPE LOOP
2426                                     :REPLACE ERROR CALL WITH
2427                                     :'BR 19$' = 000766
2428 027620          20$:      ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
2429                                     :
2430 027620 012767 027626 151262      MOV      #21$, $LPERR      ;SET LOOP ON ERROR POINTER TO 21$
2431 027626 012767 010340 150142 21$:  MOV      #010340, PSW      ;MAKE PREVIOUS MODE SUPERVISOR
2432 027634 012767 100000 151340      MOV      #100000, $TMP2    ;LOAD TEST LOC. V.A. INTO $TMP2
2433 027642 012702 001202      MOV      # $TMP2, R2       ;LOAD ADDRESS OF $TMP2 INTO R2
2434 027646 006572 000000      MFPI     @0(R2)          ;USE $TMP2 TO FETCH VIRTUAL
2435                                     :ADDRESS OF 60000
2436 027652 012601      MOV      (KSP)+, R1      ;POP KERNEL STACK INTO R1
2437 027654 020001      CMP      R0, R1          ;WAS DATA FETCHED SAME AS STORED
2438 027656 001401      BEQ      22$            ;BRANCH IF CORRECT DATA WAS FETCHED
2439 027660 104023      ERROR    +23          ;WRONG DATA WAS FETCHED
2440                                     :FOR TIGHTER SCOPE LOOP
2441                                     :REPLACE ERROR CALL WITH
2442                                     :'BR 21$' = 000762
2443 027662 012767 002456 150360 22$:  MOV      #MGMERR, MMVEC    ;SET M.M. VECTOR TO NORMAL ROUTINE
2444 027670 012767 02725c 151212      MOV      #1$, $LPERR      ;SET LOOP POINTER TO START OF TEST
2445 027676 112767 000006 142404      MOVF    #6, KIPDR4        ;MAKE KIPDR4 RESIDENT
2446 027704 000423      BR       TST22          ;:BRANCH TO NEXT TEST
2447                                     :
2448                                     :
2449 027706 012667 151346          23$:  MOV      (KSP)+, TRAPPC    ;SAVE PC & PS OF TRAP
2450 027712 012667 151344      MOV      (KSP)+, TRAPPS
2451 027716 016767 147650 151340      MOV      SR0, WASSRO      ;SAVE SR0 FOR ERROR TYPEOUT
2452 027724 016767 147646 151336      MOV      SR2, WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
2453 027732 042767 160000 147632      BIC     #160000, SR0      ;CLEAR ERROR BITS IN SR0 AND LEAVE
2454 027740 104026      ERROR    +26          ;TRIED TO READ NON-RESIDENT PAGE
2455                                     :FOR TIGHTER SCOPE LOOP
2456                                     :REPLACE ERROR CALL WITH
2457                                     :A 'NOP' = 000240
2458 027742 016746 151314      MOV      TRAPPS, -(KSP)    ;PUT PC & PS OF TRAP ON STACK
2459 027746 016746 151306      MOV      TRAPPL, -(KSP)
2460 027752 000002      RTI
2461

```

2474

```

.....
*TEST 22      MOVE FROM PREVIOUS (USER) I-SPACE
*
*   THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT THE
*   PREVIOUS MODE IS CLOCKED CORRECTLY
*   THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.
*
*   IF THE CORRECT MODE (USER) IS NOT ENABLED A NON-RESIDENT ABORT
*   WILL OCCUR AND TRAP TO 23$, WHERE THE ERRORS ARE REPORTED.
*
.....
  
```

```

027754 000004
2475 027756 012700 036514
2476 027762 012767 000600 147660
2477 027770 010037 100000
2478 027774 012767 030404 150246
2479 030002 105067 142302
2480
2481
2482 030006 012767 030014 151074
2483 030014 012767 030340 147754
2484 030022 006506
2485 030024 022706 001100
  
```

```

TST22: SCOPE
1$: MOV #36514,R0 ;LOAD DATA PATTERN INTO R0
MOV #600,UIPAR4 ;MAP UIPAR4 TO 12K
MOV R0,#100000 ;LOAD DATA PATTERN INTO PHY 60000
MOV #23$,MMVEC ;SET M.M. VECTOR TO 23$
CLRB KIPDR4 ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
;THE FOLLOWING WILL TEST DSTM=0 MFPI
.
MOV #5$,SLPERR ;SET LOOP ON ERROR POINTER TO 5$
5$: MOV #030340,PSW ;MAKE PREVIOUS MODE USER
6$: MFPI USP ;PUT USER STACK POINTER ON KERNEL STACK
CMP #KERSTK,KSP ;WAS SOMETHING PUSHEI ON STACK AT 6$
  
```

2487 030030 001407
2488 030032 012600
2489 030034 012701 000600
2490 030040 020001
2491 030042 001403
2492 030044 104023
2493
2494

BEQ 7\$
MOV (KSP)+,R0
MOV #USESTK,R1
CMP R0,R1
BEQ 8\$
ERROR +23

:BRANCH IF NOTHING WAS PUSHED
:POP KERNEL STACK INTO R0
:EXPECTING TO GET 600 AS USP
:DID YOU GET THE RIGHT POINTER?
:BRANCH IF YOU DID
:WRONG THING WAS PUSHED ON STACK
:FOR TIGHTER SCOPE LOOP
:REPLACE ERROR CALL WITH

```

2496
2497 030046 000401
2498 030050 104025
2499
2500
2501
2502 030052
2503 030052 012767 030064 151030
2504 030060 012700 036514
2505 030064 012767 030340 147704
2506 030072 012702 100000
2507 030076 006512
2508 030100 012601
2509 030102 020001
2510 030104 001401
2511 030106 104023
2512
2513
2514
2515 030110
2516 030110 012767 030116 150772
2517 030116 012767 030340 147652
2518 030124 012702 100000

      BR      8$
      ERROR   +25
      7$:
      8$:
      9$:
      10$:
      11$:

      ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
      MOV     #9$, $LPERR
      MOV     #36514, R0
      MOV     #030340, PSW
      MOV     #100000, R2
      MFPI   (R2)
      MOV     (KSP)+, R1
      CMP    R0, R1
      BEQ    10$
      ERROR   +23

      ;THE FOLLOWING WILL TEST DSTM=2 MFPI.
      MOV     #11$, $LPERR
      MOV     #030340, PSW
      MOV     #100000, R2

      ;'BR 5$' = 000763
      ;BRANCH TO NEXT TRY
      ;NOTHING PUSHED ON STACK
      ;FOR TIGHTER SCOPE LOOP
      ;REPLACE ERROR CALL WITH
      ;'BR 5$' = 000761
      ;SET LOOP ON ERROR POINTER TO 9$
      ;RELOAD DATA PATTERN IN R0
      ;MAKE PREVIOUS MODE USER
      ;LOAD VIRTUAL ADDRESS INTO R2
      ;READ FROM PHYSICAL 60000
      ;POP KERNEL STACK INTO R1
      ;WAS DATA FETCHED SAME AS STORED
      ;BRANCH IF CORRECT DATA WAS FETCHED
      ;WRONG DATA WAS FETCHED
      ;FOR TIGHTER SCOPE LOOP
      ;REPLACE ERROR CALL WITH
      ;'BR 9$' = 000766
      ;SET LOOP ON ERROR POINTER TO 11$
      ;MAKE PREVIOUS MODE USER
      ;LOAD VIRTUAL ADDRESS INTO R2
  
```

```
2520 030130 006522 MFPI (R2)+ :READ FROM PHYSICAL 60000
2521 030132 012601 MOV (KSP)+,R1 :POP KERNEL STACK INTO R1
2522 030134 020001 CMP R0,R1 :WAS DATA FETCHED SAME AS STORED
2523 030136 001401 BEQ 12$ :BRANCH IF CORRECT DATA WAS FETCHED
2524 030140 104023 ERROR +23 :WRONG DATA WAS FETCHED
2525 :FOR TIGHTER SCOPE LOOP
2526 :REPLACE ERROR CALL WITH
2527 :'BR 11$' = 000766
2528 030142 12$: :THE FOLLOWING WILL TEST DSTM=3 MFPI.
2529 030142 012767 030150 150740 MOV #13$,SLPERR :SET LOOP ON ERROR POINTER TO 13$
2530 030150 012767 030340 147620 13$: MOV #030340,PSW :MAKE PREVIOUS MODE USER
```


2532 030156 006537 100000
2533 030162 012601
2534 030164 020001
2535 030166 001401
2536 030170 104023
2537
2538
2539
2540 030172

MFPI @#100000
MOV (KSP)+,R1
CMP R0,R1
BEQ 14\$
ERROR +23

:READ FROM PHYSICAL 60000
:POP KERNEL STACK INTO R1
:WAS DATA FETCHED SAME AS STORED
:BRANCH IF CORRECT DATA WAS FETCHED
:WRONG DATA WAS FETCHED
:FOR TIGHTER SCOPE LOOP
:REPLACE ERROR CALL WITH
: 'BR 13\$' = 000767

14\$: ;THE FOLLOWING WILL TEST DSTM=4 MFPI.

```

2542 030172 012767 030200 150710      MOV      #15$,SLPERR      ;SET LOOP ON ERROR POINTER TO 15$
2543 030200 012767 030340 147570 15$:  MOV      #030340,PSW     ;MAKE PREVIOUS MODE USER
2544 030206 012702 100002                MOV      #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
2545 030212 006542                MFPI     -(R2)           ;READ FROM PHYSICAL 60000
2546 030214 012601                MOV      (KSP)+,R1      ;POP KERNEL STACK INTO R1
2547 030216 020001                CMP      R0,R1          ;WAS DATA FETCHED SAME AS STORED
2548 030220 001401                BEQ      16$            ;BRANCH IF CORRECT DATA WAS FETCHED
2549 030222 104023                ERROR   +23            ;WRONG DATA WAS FETCHED
2550                                ;FOR TIGHTER SCOPE LOOP
2551                                ;REPLACE ERROR CALL WITH
2552                                ;'BR 15$' = 000766
2553 030224                                16$:
2554                                ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
2555                                ;
2556 030224 012767 030232 150656      MOV      #17$,SLPERR     ;SET LOOP ON ERROR POINTER TO 17$
2557 030232 012767 030340 147536 17$:  MOV      #030340,PSW     ;MAKE PREVIOUS MODE USER

```

2559 030240 012767 100000 150734 MOV #100000,STMP2 ;LOAD TEST LOC. VIRT. ADDR INTO LOC. STMP2

2561 030246 012702 001204
2562 030252 006552
2563 030254 012601

MOV #(<\$TMP2+2>),R2 ;LOAD ADDR. OF \$TMP2+2 INTO R2
MFPI @-(R2) ;READ FROM PHYSICAL 60000
MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1

```

2565 030256 020001      CMP      R0,R1      ;WAS DATA FETCHED SAME AS STORED
2566 030260 001401      BEQ      18$        ;BRANCH IF CORRECT DATA WAS FETCHED
2567 030262 104023      ERROR    +23       ;WRONG DATA WAS FETCHED
2568                      ;FOR TIGHTER SCOPE LOOP
2569                      ;REPLACE ERROR CALL WITH
2570                      ;'BR 17$' = 000763
2571 030264          18$: ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
2572                      ;
2573 030264 012767 030272 150616      MOV      #19$, $LPERR ;SET LOOP ON ERROR POINTER TO 19$
2574 030272 012767 030340 147476 19$: MOV      #030340,PSW  ;MAKE PREVIOUS MODE USER
2575 030300 005002                      CLR      R2          ;MAKE REGISTER 2 A ZERO
2576 030302 006562 100000      MFPI    100000(R2)  ;READ FROM PHYSICAL 60000
2577 030306 012601      MOV      (KSP)+,R1   ;POP KERNEL STACK INTO R1
2578 030310 020001      CMP      R0,R1      ;WAS DATA FETCHED SAME AS STORED
2579 030312 001401      BEQ      20$        ;BRANCH IF CORRECT DATA WAS FETCHED
2580 030314 104023      ERROR    +23       ;WRONG DATA WAS FETCHED
2581                      ;FOR TIGHTER SCOPE LOOP
2582                      ;REPLACE ERROR CALL WITH
2583                      ;'BR 19$' = 000766
2584 030316          20$: ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
2585                      ;
2586 030316 012767 030324 150564      MOV      #21$, $LPERR ;SET LOOP ON ERROR POINTER TO 21$
2587 030324 012767 030340 147444 21$: MOV      #030340,PSW  ;MAKE PREVIOUS MODE USER
2588 030332 012767 100000 150642      MOV      #100000,$TMP2 ;LOAD TEST LOC. V.A. INTO $TMP2
2589 030340 012702 001202      MOV      #$TMP2,R2   ;LOAD ADDRESS OF $TMP2 INTO R2
2590 030344 006572 000000      MFPI    @0(R2)      ;USE $TMP2 TO FETCH VIRTUAL
2591                      ;ADDRESS OF 60000
2592 030350 012601      MOV      (KSP)+,R1   ;POP KERNEL STACK INTO R1
2593 030352 020001      CMP      R0,R1      ;WAS DATA FETCHED SAME AS STORED
2594 030354 001401      BEQ      22$        ;BRANCH IF CORRECT DATA WAS FETCHED
2595 030356 104023      ERROR    +23       ;WRONG DATA WAS FETCHED
2596                      ;FOR TIGHTER SCOPE LOOP
2597                      ;REPLACE ERROR CALL WITH
2598                      ;'BR 21$' = 000762
2599 030360 012767 002456 147662 22$: MOV      #MGMERR,MMVEC ;SET M.M. VECTOR TO NORMAL ROUTINE
2600 030366 012767 027756 150514      MOV      #1$, $LPERR ;SET LOOP POINTER TO START OF TEST
2601 030374 112767 000006 141706      MOVB    #6,KIPDR4   ;MAKE KIPDR4 RESIDENT
2602 030402 000423      BR      TST23      ;BRANCH TO NEXT TEST
2603
2604
2605 030404 012667 150650          23$: MOV      (KSP)+,TRAPPC ;SAVE PC OF TRAP
2606 030410 012667 150646      MOV      (KSP)+,TRAPPS
2607 030414 016767 147152 150642      MOV      SR0,WASSRO ;SAVE SR0 FOR ERROR TYPEOUT
2608 030422 016767 147150 150640      MOV      SR2,WASSR2 ;SAVE SR2 FOR ERROR TYPEOUT
2609 030430 042767 160000 147134      BIC     #160000,SR0 ;CLEAR ERROR BITS IN SR0 AND LEAVE
2610 030436 104026      ERROR    +26       ;TRIED TO READ NON-RESIDENT PAGE
2611                      ;FOR TIGHTER SCOPE LOOP
2612                      ;REPLACE ERROR CALL WITH
2613                      ;A 'NOP' = 000240
2614 030440 016746 150616      MOV      TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
2615 030444 016746 150610      MOV      TRAPPC,-(KSP)
2616 030450 000002      RTI
  
```

2618
 2619
 2631

```

*****
*TEST 23      MOVE TO PREVIOUS(SUPERVISOR) I-SPACE
*
*   THIS TEST USES THE 'MTPI' INSTRUCTION TO ENSURE THAT THE
*   PREVIOUS MODE IS CLOKED CORRECTLY
*   THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.
*
*   IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
*   WILL OCCUR AND TRAP TO 20$, WHERE THE ERRORS ARE REPORTED.
*
*****
  
```

```

TST23: SCOPE
1$:  MOV      #77406,SIPDR4  ;SUPERVISOR I-SPACE PAGE 4 READ/WRITE
    MOV      #20$,MMVEC    ;SET M.M. VECTOR TO 20$
    ;THE FOLLOWING WILL TEST DSTM=0 MTPI
    .
2$:  MOV      #010340,PSW   ;MAKE PREVIOUS MODE SUPERVISOR
    MOV      #7777,-(KSP)  ;PUSH DATA ON KERNEL STACK
    MTPI     SSP           ;LOAD SUPERVISOR STACK POINTER
    MFPI     SSP           ;READ SUPERVISOR STACK POINTER
    MOV      (KSP)+,R1     ;POP KERNEL STACK INTO R1
    CMP      #7777,R1     ;WAS SUPERVISOR STACK POINTER CHANGED
    BEQ      3$           ;BRANCH IF IT WAS
    ERROR    +25         ;SUPERVISOR STACK POINTER NOT CHANGED
    ;FOR TIGHTER SCOPE LOOP
    ;REPLACE ERROR CALL WITH
    ;'BR 2$' = 000764
3$:  MOV      #010340,PSW   ;MAKE PREVIOUS MODE SUPERVISOR
    MOV      #SUPSTK,-(KSP);GET READY TO RESTORE SUPERVISOR S. POINT
    MTPI     SSP           ;RESTORE SUPERVISOR STACK POINTER
    ;THIS WILL TEST DSTM = 1 MTPI.
4$:  MOV      #5$,SLPERR    ;SET LOOP ON ERROR POINTER TO 5$
    MOV      #100000,R2    ;LOAD VIRTUAL ADDRESS INTO R2
    MOV      #125252,R0    ;LOAD TEST DATA INTO R0
    MOV      R0,-(KSP)     ;PUSH TEST DATA ON KERNEL STACK
    CLRB     KIPDR4       ;MAKE KERNEL I PAGE 4 NON-RESIDENT
    MTPI     (R2)         ;LOAD TEST DATA INTO PHYSICAL 60000
    MOV      #006,KIPDR4  ;MAKE KERNEL PAGE 4 RESIDENT
    MOV      (R2),R1      ;READ FROM ADDRESS 60000
    CMP      R0,R1        ;SEE IF DATA WAS STORED AT CORRECT PLACE
    BEQ      6$           ;BRANCH IF STORE WAS CORRECT
    ERROR    +24         ;INCORRECT STORE
    ;FOR TIGHTER SCOPE LOOP
    ;REPLACE ERROR CALL WITH
    ;'BR 5$' = 000765
5$:  ;THE FOLLOWING WILL TEST DSTM=2 MTPI.
    .
6$:  MOV      #8$,SLPERR    ;SET LOOP ON ERROR POINTER TO 8$
    MOV      #010340,PSW   ;MAKE PREVIOUS MODE SUPERVISOR
    MOV      #125252,R0    ;LOAD TEST DATA INTO R0
    MOV      #100000,R2    ;LOAD VIRTUAL ADDRESS INTO R2
    MOV      R0,-(KSP)     ;PUSH TEST DATA ON KERNEL STACK
    CLRB     KIPDR4       ;MAKE KERNEL PAGE 4 NON-RESIDENT
    MTPI     (R2)         ;LOAD TEST DATA INTO PHYSICAL 60000
  
```

```

2674 030634 112767 000006 141446      MOVB #006,KIPDR4      ;MAKE KERNEL PAGE 4 RESIDENT
2675 030642 013701 100000              MOV @#100000,R1      ;READ FROM ADDRESS 60000
2676 030646 020001              CMP R0,R1            ;SEE IF DATA WAS STORED CORRECTLY
2677 030650 001401              BEQ 9$               ;BRANCH IF STORE WAS CORRECT
2678 030652 104024              ERROR +24           ;INCORRECT STORE
2679                                ;FOR TIGHTER SCOPE LOOP
2680                                ;REPLACE ERROR CALL WITH
2681                                ;'BR 8$' = 000764
2682 030654                                9$: ;THIS WILL TEST DSTM = 3 MTPI.
2683 030654 012767 030674 150226      MOV #10$,SLPERR      ;SET LOOP ON ERROR POINTER TO 10$
2684 030662 012767 010340 147106      MOV #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
2685 030670 012700 052525              MOV #52525,R0       ;LOAD TEST DATA INTO R0
2686 030674 010046                                10$: MOV R0,-(KSP)        ;PUSH TEST DATA ON KERNEL STACK
2687 030676 105067 141406      CLRB KIPDR4         ;MAKE KERNEL I PAGE 4 NON-RESIDENT
2688 030702 006637 100000      MTP1 @#100000       ;LOAD TEST DATA INTO PHYSICAL 60000
2689 030706 112767 000006 141374      MOVB #006,KIPDR4    ;MAKE KERNEL PAGE 4 RESIDENT
2690 030714 013701 100000              MOV @#100000,R1     ;READ FROM ADDRESS 60000
2691 030720 020001              CMP R0,R1            ;SEE IF DATA WAS STORED CORRECTLY
2692 030722 001401              BEQ 11$              ;BRANCH IF STORE WAS CORRECT
2693 030724 104024              ERROR +24           ;INCORRECT STORE
2694                                ;FOR TIGHTER SCOPE LOOP
2695                                ;REPLACE ERROR CALL WITH
2696                                ;'BR 10$' = 000763
2697 030726                                11$: ;THIS WILL TEST DSTM = 4 MTPI.
2698 030726 012767 030746 150154      MOV #12$,SLPERR     ;SET LOOP ON ERROR POINTER TO 12$
2699 030734 012767 010340 147034      MOV #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
2700 030742 012700 125252              MOV #125252,R0      ;LOAD TEST DATA INTO R0
2701 030746 010046                                12$: MOV R0,-(KSP)        ;PUSH TEST DATA ON KERNEL STACK
2702 030750 012702 100002              MOV #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
2703 030754 105067 141330      CLRB KIPDR4         ;MAKE KERNEL I PAGE 4 NON-RESIDENT
2704 030760 006642              MTP1 -(R2)          ;LOAD TEST DATA INTO PHYSICAL 60000
2705 030762 112767 000006 141320      MOVB #006,KIPDR4    ;MAKE KERNEL PAGE 4 RESIDENT
2706 030770 013701 100000              MOV @#100000,R1     ;READ FROM ADDRESS 60000
2707 030774 020001              CMP R0,R1            ;SEE IF DATA WAS STORED CORRECTLY
2708 030776 001401              BEQ 13$              ;BRANCH IF STORE WAS CORRECT
2709 031000 104024              ERROR +24           ;INCORRECT STORE
2710                                ;FOR TIGHTER SCOPE LOOP
2711                                ;REPLACE ERROR CALL WITH
2712                                ;'BR 12$' = 000762
2713 031002                                13$: ;THE FOLLOWING WILL TEST DSTM=5 MTPI.
2714
2715 031002 012767 031034 150100      MOV #14$,SLPERR     ;SET LOOP ON ERROR POINTER TO 14$
2716 031010 012767 010340 146760      MOV #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
2717 031016 012700 052525              MOV #52525,R0       ;LOAD TEST DATA INTO R0
2718 031022 012702 001204              MOV #<STMP2+2>,R2   ;LOAD ADDR. OF LOC. STMP2+2 INTO R2
2719 031026 012767 100000 150146      MOV #100000,STMP2   ;LOAD VIRT. ADDR. OF TEST LOC. INTO STMP2
2720 031034 010046                                14$: MOV R0,-(KSP)        ;PUSH TEST DATA ON KERNEL STACK
2721 031036 105067 141246      CLRB KIPDR4         ;MAKE KERNEL PAGE 4 NON-RESIDENT
2722 031042 006652              MTP1 @-(R2)         ;LOAD TEST DATA INTO PHYSICAL 60000
2723 031044 112767 000006 141236      MOVB #006,KIPDR4    ;MAKE KERNEL PAGE 4 RESIDENT
2724 031052 013701 100000              MOV @#100000,R1     ;READ FROM ADDRESS 60000
2725 031056 020001              CMP R0,R1            ;SEE IF DATA WAS STORED CORRECTLY
2726 031060 001401              BEQ 15$              ;BRANCH IF STORE WAS CORRECT
2727 031062 104024              ERROR +24           ;INCORRECT STORE
2728                                ;FOR TIGHTER SCOPE LOOP
2729                                ;REPLACE ERROR CALL WITH
2730                                ;'BR 14$' 000764
  
```

```

2731 031064          15$: ;THIS WILL TEST DSTM - 6 MTPI.
2732
2733 031064 012767 031106 150016      MOV    #16$, $LPERR      ;SET LOOP ON ERROR POINTER TO 16$
2734 031072 012767 010340 146676      MOV    #010340, PSW     ;MAKE PREVIOUS MODE SUPERVISOR
2735 031100 012700 052525              MOV    #52525, R0       ;LOAD TEST DATA INTO R0
2736 031104 005002                      CLR    R2               ;MAKE REGISTER 2 ZERO
2737 031106 010046          16$: MOV    R0, -(KSP)        ;PUSH TEST DATA ON KERNEL STACK
2738 031110 105067 141174              CLRB  KIPDR4           ;MAKE KERNEL I PAGE 4 NON-RESIDENT
2739 031114 006662 100000              MTP1  100000(R2)       ;LOAD TEST DATA INTO PHYSICAL 60000
2740 031120 112767 000006 141162      MOVB  #006, KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
2741 031126 013701 100000              MOV    @#100000, R1     ;READ FROM ADDRESS 60000
2742 031132 020001                      CMP    R0, R1          ;SEE IF DATA WAS STORED CORRECTLY
2743 031134 001401                      BEQ    17$             ;BRANCH IF STORE WAS CORRECT
2744 031136 104024                      ERROR  +24            ;INCORRECT STORE
2745
2746
2747
2748 031140          17$: ;THE FOLLOWING WILL TEST DSTM=7 MTPI.
2749
2750 031140 012767 031172 147742      MOV    #18$, $LPERR     ;SET LOOP ON ERROR POINTER TO 18$
2751 031146 012767 010340 146622      MOV    #010340, PSW     ;MAKE PREVIOUS MODE SUPERVISOR
2752 031154 012700 125252              MOV    #125252, R0      ;LOAD TEST DATA INTO R0
2753 031160 012767 100000 150014      MOV    #100000, $TMP2   ;LOAD VIRT. ADDR. OF TEST LOCATION
2754
2755 031166 012702 001202              MOV    #$TMP2, R2       ;LOAD ADDRESS OF $TMP2 INTO R2
2756 031172 010046          18$: MOV    R0, -(KSP)        ;PUSH TEST DATA ON KERNEL STACK
2757 031174 105067 141110              CLRB  KIPDR4           ;MAKE KERNEL PAGE 4 NON-RESIDENT
2758 031200 006672 000000              MTP1  @0(R2)          ;LOAD TEST DATA INTO PHYSICAL 60000
2759 031204 112767 000006 141076      MOVB  #006, KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
2760 031212 013701 100000              MOV    @#100000, R1     ;READ FROM ADDRESS 60000
2761 031216 020001                      CMP    R0, R1          ;SEE IF DATA WAS STORED CORRECTLY
2762 031220 001401                      BEQ    19$             ;BRANCH IF STORE WAS CORRECT
2763 031222 104024                      ERROR  +24            ;INCORRECT STORE
2764
2765
2766
2767 031224 012767 030454 147656 19$: MOV    #1$, $LPERR     ;SET LOOP POINTER TO START OF TEST
2768 031232 012767 002456 147010      MOV    #MGMERR, MMVEC   ;RESTORE M.M. VECTOR TO NORMAL ROUTINE
2769 031240 000423                      BR     TST24           ;;BRANCH TO NEXT TEST
2770
2771
2772 031242 012667 150012          20$: MOV    (KSP)+, TRAPPC    ;SAVE PC & PS OF TRAP
2773 031246 012667 150010              MOV    (KSP)+, TRAPPS
2774 031252 016767 146314 150004      MOV    SR0, WASSR0     ;SAVE SR0 FOR ERROR TYPEOUT
2775 031260 016767 146312 150002      MOV    SR2, WASSR2     ;SAVE SR2 FOR ERROR TYPEOUT
2776 031266 042767 160000 146276      BIC   #160000, SR0     ;CLEAR ERROR BITS IN SR0
2777 031274 104024                      ERROR  +24            ;TRIED TO LOAD A N.R. PAGE 4
2778
2779
2780
2781 031276 016746 147760              MOV    TRAPPS, -(KSP)   ;PUT PC & PS OF TRAP ON STACK
2782 031302 016746 147752              MOV    TRAPPC, -(KSP)
2783 031306 000002                      RTI                    ;RETURN TO TEST
  
```


2785

```

*****
*TEST 24      MOVE TO PREVIOUS (USER) I-SPACE
*
*   THIS TEST USES THE 'MTPI' INSTRUCTION TO ENSURE THAT THE
*   PREVIOUS MODE IS CLOKED CORRECTLY
*   THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.
*
*   IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
*   WILL OCCUR AND TRAP TO 20$, WHERE THE ERRORS ARE REPORTED.
*
*****
  
```

```

TST24: SCOPE
1$:  MOV      #77406,UIPDR4      ;USER I-SPACE PAGE 4 READ/WRITE
    MOV      #600,UIPAR4      ;MAP USER I PAGE 4 TO 12K
    MOV      #20$,MMVEC      ;SET M.M. VECTOR TO 20$
    ;THE FOLLOWING WILL TEST DSTM=0 MTPI
    .
2$:  MOV      #030340,PSW      ;MAKE PREVIOUS MODE USER
    MOV      #7777,-(KSP)      ;PUSH DATA ON KERNEL STACK
    MTPI     USP              ;LOAD USER STACK POINTER
    MFPI     USP              ;READ USER STACK POINTER
    MOV      (KSP)+,R1         ;POP KERNEL STACK INTO R1
    CMP      #7777,R1         ;WAS USER STACK POINTER CHANGED
    BEQ      3$              ;BRANCH IF IT WAS
    ERROR    +25             ;USER STACK POINTER NOT CHANGED
    .
    ;FOR TIGHTER SCOPE LOOP
    ;REPLACE ERROR CALL WITH
    ;'BR 2$' = 000764
3$:  MOV      #030340,PSW      ;MAKE PREVIOUS MODE USER
    MOV      #USESTK,-(KSP)    ;GET READY TO RESTORE USER S. POINT
    MTPI     USP              ;RESTORE USER STACK POINTER
    .
4$:  ;THIS WILL TEST DSTM = 1 MTPI.
    MOV      #5$,SLPERR      ;SET LOOP ON ERROR POINTER TO 5$
    MOV      #100000,R2       ;LOAD VIRTUAL ADDRESS INTO R2
    MOV      #125252,R0       ;LOAD TEST DATA INTO R0
    .
5$:  MOV      R0,-(KSP)        ;PUSH TEST DATA ON KERNEL STACK
    CLRB    KIPDR4           ;MAKE KERNEL I PAGE 4 NON-RESIDENT
    MTPI    (R2)             ;LOAD TEST DATA INTO PHYSICAL 60000
    MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
    MOV     (R2),R1          ;READ FROM ADDRESS 60000
    CMP     R0,R1           ;SEE IF DATA WAS STORED AT CORRECT PLACE
    BEQ     6$              ;BRANCH IF STORE WAS CORRECT
    ERROR   +24             ;INCORRECT STORE
    .
    ;FOR TIGHTER SCOPE LOOP
    ;REPLACE ERROR CALL WITH
    ;'BR 5$' = 000765
6$:  ;THE FOLLOWING WILL TEST DSTM=2 MTPI.
    .
8$:  MOV      #8$,SLPERR      ;SET LOOP ON ERROR POINTER TO 8$
    MOV      #030340,PSW      ;MAKE PREVIOUS MODE USER
    MOV      #125252,R0       ;LOAD TEST DATA INTO R0
    MOV      #100000,R2       ;LOAD VIRTUAL ADDRESS INTO R2
    .
    MOV      R0,-(KSP)        ;PUSH TEST DATA ON KERNEL STACK
    CLRB    KIPDR4           ;MAKE KERNEL PAGE 4 NON-RESIDENT
    MTPI    (R2)             ;LOAD TEST DATA INTO PHYSICAL 60000
    MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
  
```

```

031310 000004
2786 031312 012767 077406 146270
2787 031320 012767 000600 146322
2788 031326 012767 032106 146714
2789
2790
2791 031334 012767 030340 146434
2792 031342 012746 007777
2793 031346 006606
2794 031350 006506
2795 031352 012601
2796 031354 022701 007777
2797 031360 001401
2798 031362 104025
2799
2800
2801
2802 031364 012767 030340 146404
2803 031372 012746 000600
2804 031376 006606
2805 031400
2806 031400 012767 031416 147502
2807 031406 012702 100000
2808 031412 012700 125252
2809 031416 010046
2810 031420 105067 140664
2811 031424 006612
2812 031426 112767 000006 140654
2813 031434 011201
2814 031436 020001
2815 031440 001401
2816 031442 104024
2817
2818
2819
2820 031444
2821
2822 031444 012767 031470 147436
2823 031452 012767 030340 146316
2824 031460 012700 125252
2825 031464 012702 100000
2826 031470 010046
2827 031472 105067 140612
2828 031476 006612
2829 031500 112767 000006 140602
  
```

2830	031506	013701	100000		MOV	@#100000,R1	:READ FROM ADDRESS 60000
2831	031512	020001			CMP	R0,R1	:SEE IF DATA WAS STORED CORRECTLY
2832	031514	001401			BEQ	9\$:BRANCH IF STORE WAS CORRECT
2833	031516	104024			ERROR	+24	:INCORRECT STORE
2834							:FOR TIGHTER SCOPE LOOP
2835							:REPLACE ERROR CALL WITH
2836							: 'BR 8\$' = 000764
2837	031520			9\$:		:THIS WILL TEST DSTM = 3	MTPI.
2838	031520	012767	031540	147362	MOV	#10\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 10\$
2839	031526	012767	030340	146242	MOV	#030340,PSW	:MAKE PREVIOUS MODE USER
2840	031534	012700	052525		MOV	#52525,R0	:LOAD TEST DATA INTO R0
2841	031540	010046		10\$:	MOV	R0,-(KSP)	:PUSH TEST DATA ON KERNEL STACK
2842	031542	105067	140542		CLRB	KIPDR4	:MAKE KERNEL I PAGE 4 NON-RESIDENT
2843	031546	006637	100000		MTPI	@#100000	:LOAD TEST DATA INTO PHYSICAL 60000
2844	031552	112767	000006	140530	MOVB	#006,KIPDR4	:MAKE KERNEL PAGE 4 RESIDENT
2845	031560	013701	100000		MOV	@#100000,R1	:READ FROM ADDRESS 60000
2846	031564	020001			CMP	R0,R1	:SEE IF DATA WAS STORED CORRECTLY
2847	031566	001401			BEQ	11\$:BRANCH IF STORE WAS CORRECT
2848	031570	104024			ERROR	+24	:INCORRECT STORE
2849							:FOR TIGHTER SCOPE LOOP
2850							:REPLACE ERROR CALL WITH
2851							: 'BR 10\$' = 000763
2852	031572			11\$:		:THIS WILL TEST DSTM = 4	MTPI.
2853	031572	012767	031612	147310	MOV	#12\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 12\$
2854	031600	012767	030340	146170	MOV	#030340,PSW	:MAKE PREVIOUS MODE USER
2855	031606	012700	125252		MOV	#125252,R0	:LOAD TEST DATA INTO R0
2856	031612	010046		12\$:	MOV	R0,-(KSP)	:PUSH TEST DATA ON KERNEL STACK
2857	031614	012702	100002		MOV	#100002,R2	:LOAD VIRTUAL ADDRESS INTO R2
2858	031620	105067	140464		CLRB	KIPDR4	:MAKE KERNEL I PAGE 4 NON-RESIDENT
2859	031624	006642			MTPI	-(R2)	:LOAD TEST DATA INTO PHYSICAL 60000
2860	031626	112767	000006	140454	MOVB	#006,KIPDR4	:MAKE KERNEL PAGE 4 RESIDENT
2861	031634	013701	100000		MOV	@#100000,R1	:READ FROM ADDRESS 60000
2862	031640	020001			CMP	R0,R1	:SEE IF DATA WAS STORED CORRECTLY
2863	031642	001401			BEQ	13\$:BRANCH IF STORE WAS CORRECT
2864	031644	104024			ERROR	+24	:INCORRECT STORE
2865							:FOR TIGHTER SCOPE LOOP
2866							:REPLACE ERROR CALL WITH
2867							: 'BR 12\$' = 000762
2868	031646			13\$:		:THE FOLLOWING WILL TEST DSTM=5	MTPI.
2869							
2870	031646	012767	031700	147234	MOV	#14\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 14\$
2871	031654	012767	030340	146114	MOV	#030340,PSW	:MAKE PREVIOUS MODE USER
2872	031662	012700	052525		MOV	#52525,R0	:LOAD TEST DATA INTO R0
2873	031666	012702	001204		MOV	#<\$TMP2+2>,R2	:LOAD ADDR. OF LOC. \$TMP2+2 INTO R2
2874	031672	012767	100000	147302	MOV	#100000,\$TMP2	:LOAD VIRT. ADDR. OF TEST LOC. INTO \$TMP2
2875	031700	010046		14\$:	MOV	R0,-(KSP)	:PUSH TEST DATA ON KERNEL STACK
2876	031702	105067	140402		CLRB	KIPDR4	:MAKE KERNEL PAGE 4 NON-RESIDENT
2877	031706	006652			MTPI	@-(R2)	:LOAD TEST DATA INTO PHYSICAL 60000
2878	031710	112767	000006	140372	MOVB	#006,KIPDR4	:MAKE KERNEL PAGE 4 RESIDENT
2879	031716	013701	100000		MOV	@#100000,R1	:READ FROM ADDRESS 60000
2880	031722	020001			CMP	R0,R1	:SEE IF DATA WAS STORED CORRECTLY
2881	031724	001401			BEQ	15\$:BRANCH IF STORE WAS CORRECT
2882	031726	104024			ERROR	+24	:INCORRECT STORE
2883							:FOR TIGHTER SCOPE LOOP
2884							:REPLACE ERROR CALL WITH
2885							: 'BR 14\$' = 000764
2886	031730			15\$:		:THIS WILL TEST DSTM = 6	MTPI.

2952

```
*****
*TEST 25 MFPI (KERNAL) TO SUPERVISOR MODE
*
* THIS TEST CHECKS THAT IF THE PREVIOUS MODE IS KERNEL THE
* FETCH IS FROM KERNEL SPACE.
* THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.
*
* IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
* WILL OCCUR AND TRAP TO 21$, WHERE THE ERRORS ARE REPORTED.
*****
```

```
TST25: SCOPE
2953 032154 000004
2953 032156 012767 032164 146724
2954 032164 012767 040340 145604
2955 032172 012700 036514
2956 032176 010037 100000
2957 032202 012702 100000
2958
2959
2960 032206 012767 032616 146034
2961 032214 105067 137770
2962 032220 012767 040340 145550
2963 032226 006506
2964 032230 022706 000700
2965 032234 001407
2966 032236 012600
2967 032240 012701 001100
2968 032244 020001
2969 032246 001403
2970 032250 104023
2971
2972
2973
2974 032252 000401
2975 032254 104025
2976
2977
2978
2979 032256
2980 032256 012767 032264 146624
2981 032264 012767 040340 145504
2982 032272 012700 036514
2983 032276 012702 100000
2984 032302 006512
2985 032304 012601
2986 032306 020001
2987 032310 001401
2988 032312 104023
2989
2990
2991
2992 032314
2993 032314 012767 032322 146566
2994 032322 012767 040340 145446
2995 032330 012702 100000
2996 032334 006522

MOV #1$, $LPERR ;SET LOOP ON ERROR TO 1$
MOV #040340, PSW ;GO TO SUPERVISOR MODE FOR THIS TEST
MOV #36514, R0 ;LOAD DATA PATTERN INTO R0
MOV R0, #100000 ;LOAD DATA PATTERN INTO PHY 60000
MOV #100000, R2 ;LOAD VIRTUAL ADDRESS INTO R2
;THE FOLLOWING WILL TEST DSTM=0 MFPI
MOV #21$, MMVEC ;SET M.M. VECTOR TO 21$
CLRB SIPDR4 ;MAKE SUPERVISOR I-SPACE PAGE 4 NON-RESIDENT
MOV #040340, PSW ;MAKE PREVIOUS MODE KERNEL PRESENT SUPERVISOR
MFPI KSP ;PUT KERNEL STACK POINTER ON SUPERVISOR STACK
CMP #SUPSTK, SSP ;WAS SOMETHING PUSHED ON STACK AT 1$
BEQ 5$ ;BRANCH IF NOTHING WAS PUSHED
MOV (SSP)+, R0 ;POP SUPERVISOR STACK INTO R0
MOV #KERSTK, R1 ;EXPECTING 1100 AS KSP
CMP R0, R1 ;DID YOU GET THE RIGHT POINTER?
BEQ 6$ ;BRANCH IF YOU DID
ERROR +23 ;WRONG THING WAS PUSHED ON STACK
;FOR TIGHTER SCOPE LOOP
REPLACE ERROR CALL WITH
;BR 4$ = 000766
BR 6$ ;BRANCH TO NEXT TRY
ERROR +25 ;NOTHING PUSHED ON STACK
;FOR TIGHTER SCOPE LOOP
REPLACE ERROR CALL WITH
;BR 4$ = 000764
;THE FOLLOWING WILL TEST DSTM=1 MFPI.
MOV #7$, $LPERR ;SET LOOP ON ERROR POINTER TO 7$
MOV #040340, PSW ;MAKE PREVIOUS MODE KERNEL PRESENT SUPERVISOR
MOV #36514, R0 ;LOAD DATA EXPECTED INTO R0
MOV #100000, R2 ;LOAD VIRTUAL ADDRESS INTO R2
MFPI (R2) ;READ FROM PHYSICAL 60000
MOV (SSP)+, R1 ;POP SUPERVISOR STACK INTO R1
CMP R0, R1 ;WAS DATA FETCHED SAME AS STORED
BEQ 8$ ;BRANCH IF CORRECT DATA WAS FETCHED
ERROR +23 ;WRONG DATA WAS FETCHED
;FOR TIGHTER SCOPE LOOP
REPLACE ERROR CALL WITH
;BR 7$ = 000764
;THE FOLLOWING WILL TEST DSM=2 MFPI.
MOV #9$, $LPERR ;SET LOOP ON ERROR POINTER TO 9$
MOV #040340, PSW ;MAKE PREVIOUS MODE KERNEL PRESENT SUPERVISOR
MOV #100000, R2 ;LOAD VIRTUAL ADDRESS INTO R2
MFPI (R2)+ ;READ FROM PHYSICAL 60000
```

```

2997 032336 012601      MOV      (SSP)+,R1      ;POP SUPERVISOR STACK INTO R1
2998 032340 020001      CMP      R0,R1         ;WAS DATA FETCHED SAME AS STORED
2999 032342 001401      BEQ     10$           ;BRANCH IF CORRECT DATA WAS FETCHED
3000 032344 104023      ERROR   +23          ;WRONG DATA WAS FETCHED
3001                                ;FOR TIGHTER SCOPE LOOP
3002                                ;REPLACE ERROR CALL WITH
3003                                ;'BR 9$' = 000766
3004 032346                                10$:      ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
3005 032346 012767 032354 146534      MOV      #11$, $LPERR   ;SET LOOP ON ERROR POINTER TO 11$
3006 032354 012767 040340 145414 11$:      MOV      #040340,PSW    ;MAKE PREVIOUS MODE KERNEL PRESENT SUPERVISOR
3007 032362 006537 100000      MFPI    @100000        ;READ FROM PHYSICAL 60000
3008 032366 012601      MOV      (SSP)+,R1      ;POP SUPERVISOR STACK INTO R1
3009 032370 020001      CMP      R0,R1         ;WAS DATA FETCHED SAME AS STORED
3010 032372 001401      BEQ     12$           ;BRANCH IF CORRECT DATA WAS FETCHED
3011 032374 104023      ERROR   +23          ;WRONG DATA WAS FETCHED
3012                                ;FOR TIGHTER SCOPE LOOP
3013                                ;REPLACE ERROR CALL WITH
3014                                ;'BR 11$' = 000767
3015 032376                                12$:      ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
3016 032376 012767 032404 146504      MOV      #13$, $LPERR   ;SET LOOP ON ERROR POINTER TO 13$
3017 032404 012767 040340 145364 13$:      MOV      #040340,PSW    ;MAKE PREVIOUS MODE DERNEL PRESENT SUPERVISOR
3018 032412 012702 100002      MOV      #100002,R2     ;LOAD VIRTUAL ADDRESS INTO R2
3019 032416 006542      MFPI    -(R2)          ;READ FROM PHYSICAL 60000
3020 032420 012601      MOV      (SSP)+,R1      ;POP SUPERVISOR STACK INTO R1
3021 032422 020001      CMP      R0,R1         ;WAS DATA FETCHED SAME AS STORED
3022 032424 001401      BEQ     14$           ;BRANCH IF CORRECT DATA WAS FETCHED
3023 032426 104023      ERROR   +23          ;WRONG DATA WAS FETCHED
3024                                ;FOR TIGHTER SCOPE LOOP
3025                                ;REPLACE ERROR CALL WITH
3026                                ;'BR 13$' = 000766
3027 032430                                14$:      ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
3028                                ;
3029 032430 012767 032436 146452      MOV      #15$, $LPERR   ;SET LOOP ON ERROR POINTER TO 15$
3030 032436 012767 040340 145332 15$:      MOV      #040340,PSW    ;MAKE PREVIOUS MODE KERNEL PRESENT SUPERVISOR
3031 032444 012767 100000 146530      MOV      #100000,$TMP2  ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
3032 032452 012702 001204      MOV      #<$TMP2+2>,R2  ;LOAD ADDRESS OF $TMP2+2 INTO R2
3033 032456 006552      MFPI    @-(R2)          ;READ FROM PHYSICAL 60000
3034 032460 012601      MOV      (SSP)+,R1      ;POP SUPERVISOR STACK INTO R1
3035 032462 020001      CMP      R0,R1         ;WAS DATA FETCHED SAME AS STORED
3036 032464 001401      BEQ     16$           ;BRANCH IF CORRECT DATA FETCHED
3037 032466 104023      ERROR   +23          ;WRONG DATA WAS FETCHED
3038                                ;FOR TIGHTER SCOPE LOOP
3039                                ;REPLACE ERROR CALL WITH
3040                                ;'BR 15$' = 000763
3041 032470                                16$:      ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
3042                                ;
3043 032470 012767 032476 146412      MOV      #17$, $LPERR   ;SET LOOP ON ERROR POINTER TO 17$.
3044 032476 012767 040340 145272 17$:      MOV      #040340,PSW    ;MAKE PREVIOUS MODE KERNEL PRESENT SUPERVISOR
3045 032504 005002      CLR     R2             ;MAKE REGISTER 2 A ZERO
3046 032506 006562 100000      MFPI    100000(R2)     ;READ FROM PHYSICAL 60000
3047 032512 012601      MOV      (SSP)+,R1      ;POP SUPERVISOR STACK INTO R1
3048 032514 020001      CMP      R0,R1         ;WAS DATA FETCHED SAME AS STORED
3049 032516 001401      BEQ     18$           ;BRANCH IF CORRECT DATA FETCHED
3050 032520 104023      ERROR   +23          ;WRONG DATA WAS FETCHED
3051                                ;FOR TIGHTER SCOPE LOOP
3052                                ;REPLACE ERROR CALL WITH
3053                                ;'BR 17$' = 000766

```

```

3054 032522      18$:      ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
3055
3056 032522 012767 032530 146360      MOV      #19$, $LPERR      ;SET LOOP ON ERROR POINTER TO 19$
3057 032530 012767 040340 145240 19$:      MOV      #040340, PSW      ;MAKE PREVIOUS MODE KERNEL PRESENT SUPERVISOR
3058 032536 012767 100000 146436      MOV      #100000, $TMP2    ;LOAD TEST LOC. VIRT. ADDR. INTO $TMP2
3059 032544 012702 001202      MOV      # $TMP2, R2      ;LOAD ADDRESS OF $TMP2 INTO R2
3060 032550 006572 000000      MFPI     @0(R2)           ;READ FROM PHYSICAL 60000
3061 032554 012601      MOV      (SSP)+, R1      ;POP SUPERVISOR STACK INTO R1
3062 032556 020001      CMP      R0, R1          ;WAS DATA FETCHED SAME AS STORED
3063 032560 001401      BEQ     20$             ;BRANCH IF CORRECT DATA FETCHED
3064 032562 104023      ERROR   +23            ;WRONG DATA WAS FETCHED
3065
3066
3067
3068 032564 012767 002456 145456 20$:      MOV      #MMGMERR, MMVEC   ;SET M.M. VECTOR TO NORMAL ROUTINE
3069 032572 012767 000340 145176      MOV      #00340, PSW      ;GO BACK TO KERNEL MODE, PREVIOUS KERNEL
3070 032600 012767 032164 146302      MOV      #1$, $LPERR      ;SET LOOP POINTER TO START OF TEST
3071 032606 112767 000006 137374      MOVB    #6, $IPDR4        ;MAKE $IPDR4 RESIDENT
3072 032614 000423      BR      TST26           ;;BRANCH TO NEXT TEXT
3073
3074
3075 032616 012667 146436      21$:      MOV      (KSP)+, TRAPPC    ;SAVE PC & PS OF TRAP
3076 032622 012667 146434      MOV      (KSP)+, TRAPPS    ;SAVE PC & PS OF TRAP
3077 032626 016767 144740 146430      MOV      SR0, WASSRO      ;SAVE SR0 FOR ERROR TYPEOUT
3078 032634 016767 144736 146426      MOV      SR2, WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
3079 032642 042767 160000 144722      BIC     #160000, SR0      ;CLEAR ERROR BITS IN SR0
3080 032650 104026      ERROR   +26            ;TRIED TO READ NON-RESIDENT PAGE
3081
3082
3083
3084 032652 016746 146404      MOV      TRAPPS, -(KSP)    ;PUT PC & PS OF TRAP ON STACK
3085 032656 016746 146376      MOV      TRAPPC, -(KSP)
3086 032662 000002      RTI                    ;RETURN TO TEST
3087

```

```

*****
;TEST 26      MFPI (KERNEL) TO USER MODE
;
;      THIS TEST CHECKS THAT IF THE PREVIOUS MODE IS KERNEL THE
;      FETCH IS FROM KERNEL SPACE.
;      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.
;
;      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
;      WILL OCCUR AND TRAP TO 21$, WHERE THE ERRORS ARE REPORTED.
*****

```

```

3088 032664 000004      TST26:  SCOPE
3089 032666 012767 032674 146214      MOV      #1$, $LPERR      ;SET LOOP ON ERROR TO 1$
3090 032674 012767 140340 145074 1$:      MOV      #140340, PSW      ;GO TO USER MODE FOR THIS TEST
3091 032702 012700 036514      MOV      #36514, R0       ;LOAD DATA PATTERN INTO R0
3092 032706 010037 100000      MOV      R0, @#100000     ;LOAD DATA PATTERN INTO PHY 60000
3093 032712 012702 100000      MOV      #100000, R2      ;LOAD VIRTUAL ADDRESS INTO R2
3094
3095 032716 012767 033326 145324      ;THE FOLLOWING WILL TEST DSTM=0 MFPI
3096 032724 105067 144660      MOV      #21$, MMVEC      ;SET M.M. VECTOR TO 21$
3097 032730 012767 140340 145040      CLRB    $IPDR4           ;MAKE USER I-SPACE PAGE 4 NON-RESIDENT
3098 032736 006506      4$:      MOV      #140340, PSW      ;MAKE PREVIOUS MODE KERNEL PRESENT USER
;      MFPI     KSP          ;PUT KERNEL STACK POINTER ON USER STACK

```

```

3099 032740 022706 000600      CMP      #USESTK,USP      ;WAS SOMETHING PUSHED ON STACK AT 'S
3100 032744 001407      BEQ      5$              ;BRANCH IF NOTHING WAS PUSHED
3101 032746 012600      MOV      (USP)+,R0      ;POP USER STACK INTO R0
3102 032750 012701 001100      MOV      #KERSTK,R1     ;EXPECTING 1100 AS KSP
3103 032754 020001      CMP      R0,R1         ;DID YOU GET THE RIGHT POINTER?
3104 032756 001403      BEQ      6$              ;BRANCH IF YOU DID
3105 032760 104023      ERROR   +23           ;WRONG THING WAS PUSHED ON STACK
3106                                ;FOR TIGHTER SCOPE LOOP
3107                                ;REPLACE ERROR CALL WITH
3108                                ;'BR 4$' = 000766
3109 032762 000401      BR       6$              ;BRANCH TO NEXT TRY
3110 032764 104025      5$:     ERROR   +25           ;NOTHING PUSHED ON STACK
3111                                ;FOR TIGHTER SCOPE LOOP
3112                                ;REPLACE ERROR CALL WITH
3113                                ;'BR 4$' = 000764
3114 032766                                6$:     ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
3115 032766 012767 032774 146114      MOV      #7$,SLPERR     ;SET LOOP ON ERROR POINTER TO 7$
3116 032774 012767 140340 144774      7$:     MOV      #140340,PSW    ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3117 033002 012700 036514      MOV      #36514,R0     ;LOAD DATA EXPECTED INTO R0
3118 033006 012702 100000      MOV      #100000,R2    ;LOAD VIRTUAL ADDRESS INTO R2
3119 033012 006512      MFPI    (R2)           ;READ FROM PHYSICAL 60000
3120 033014 012601      MOV      (USP)+,R1     ;POP USER STACK INTO R1
3121 033016 020001      CMP      R0,R1         ;WAS DATA FETCHED SAME AS STORED
3122 033020 001401      BEQ      8$              ;BRANCH IF CORRECT DATA WAS FETCHED
3123 033022 104023      ERROR   +23           ;WRONG DATA WAS FETCHED
3124                                ;FOR TIGHTER SCOPE LOOP
3125                                ;REPLACE ERROR CALL WITH
3126                                ;'BR 7$' = 000764
3127 033024                                8$:     ;THE FOLLOWING WILL TEST DSM=2 MFPI.
3128 033024 012767 033032 146056      MOV      #9$,SLPERR     ;SET LOOP ON ERROR POINTER TO 9$
3129 033032 012767 140340 144736      9$:     MOV      #140340,PSW    ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3130 033040 012702 100000      MOV      #100000,R2    ;LOAD VIRTUAL ADDRESS INTO R2
3131 033044 006522      MFPI    (R2)+         ;READ FROM PHYSICAL 60000
3132 033046 012601      MOV      (USP)+,R1     ;POP USER STACK INTO R1
3133 033050 020001      CMP      R0,R1         ;WAS DATA FETCHED SAME AS STORED
3134 033052 001401      BEQ      10$            ;BRANCH IF CORRECT DATA WAS FETCHED
3135 033054 104023      ERROR   +23           ;WRONG DATA WAS FETCHED
3136                                ;FOR TIGHTER SCOPE LOOP
3137                                ;REPLACE ERROR CALL WITH
3138                                ;'BR 9$' = 000766
3139 033056                                10$:    ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
3140 033056 012767 033064 146024      MOV      #11$,SLPERR    ;SET LOOP ON ERROR POINTER TO 11$
3141 033064 012767 140340 144704      11$:    MOV      #140340,PSW    ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3142 033072 006537 100000      MFPI    @#100000      ;READ FROM PHYSICAL 60000
3143 033076 012601      MOV      (USP)+,R1     ;POP USER STACK INTO R1
3144 033100 020001      CMP      R0,R1         ;WAS DATA FETCHED SAME AS STORED
3145 033102 001401      BEQ      12$            ;BRANCH IF CORRECT DATA WAS FETCHED
3146 033104 104023      ERROR   +23           ;WRONG DATA WAS FETCHED
3147                                ;FOR TIGHTER SCOPE LOOP
3148                                ;REPLACE ERROR CALL WITH
3149                                ;'BR 11$' = 000767
3150 033106                                12$:    ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
3151 033106 012767 033114 145774      MOV      #13$,SLPERR    ;SET LOOP ON ERROR POINTER TO 13$
3152 033114 012767 140340 144654      13$:    MOV      #140340,PSW    ;MAKE PREVIOUS MODE DERNEL PRESENT USER
3153 033122 012702 100002      MOV      #100002,R2    ;LOAD VIRTUAL ADDRESS INTO R2
3154 033126 006542      MFPI    -(R2)         ;READ FROM PHYSICAL 60000
3155 033130 012601      MOV      (USP)+,R1     ;POP USER STACK INTO R1

```

```

3156 033132 020001      CMP      R0,R1      ;WAS DATA FETCHED SAME AS STORED
3157 033134 001401      BEQ      14$        ;BRANCH IF CORRECT DATA WAS FETCHED
3158 033136 104023      ERROR   +23        ;WRONG DATA WAS FETCHED
3159                      ;FOR TIGHTER SCOPE LOOP
3160                      ;REPLACE ERROR CALL WITH
3161                      ;'BR 13$' = 000766
3162 033140          14$: ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
3163                      ;
3164 033140 012767 033146 145742      MOV      #15$,SLPERR ;SET LOOP ON ERROR POINTER TO 15$
3165 033146 012767 140340 144622      MOV      #140340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3166 033154 012767 100000 146020      MOV      #100000,$TMP2 ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
3167 033162 012702 001204          MOV      #<$TMP2+2>,R2 ;LOAD ADDRESS OF $TMP2+2 INTO R2
3168 033166 006552          MFPI     @-(R2)      ;READ FROM PHYSICAL 60000
3169 033170 012601          MOV      (USP)+,R1    ;POP USER STACK INTO R1
3170 033172 020001      CMP      R0,R1      ;WAS DATA FETCHED SAME AS STORED
3171 033174 001401      BEQ      16$        ;BRANCH IF CORRECT DATA FETCHED
3172 033176 104023      ERROR   +23        ;WRONG DATA WAS FETCHED
3173                      ;FOR TIGHTER SCOPE LOOP
3174                      ;REPLACE ERROR CALL WITH
3175                      ;'BR 15$' = 000763
3176 033200          16$: ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
3177                      ;
3178 033200 012767 033206 145702      MOV      #17$,SLPERR ;SET LOOP ON ERROR POINTER TO 17$.
3179 033206 012767 140340 144562      MOV      #140340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3180 033214 005002          CLR      R2          ;MAKE REGISTER 2 A ZERO
3181 033216 006562 100000          MFPI     100000(R2) ;READ FROM PHYSICAL 60000
3182 033222 012601          MOV      (USP)+,R1    ;POP USER STACK INTO R1
3183 033224 020001      CMP      R0,R1      ;WAS DATA FETCHED SAME AS STORED
3184 033226 001401      BEQ      18$        ;BRANCH IF CORRECT DATA FETCHED
3185 033230 104023      ERROR   +23        ;WRONG DATA WAS FETCHED
3186                      ;FOR TIGHTER SCOPE LOOP
3187                      ;REPLACE ERROR CALL WITH
3188                      ;'BR 17$' = 000766
3189 033232          18$: ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
3190                      ;
3191 033232 012767 033240 145650      MOV      #19$,SLPERR ;SET LOOP ON ERROR POINTER TO 19$
3192 033240 012767 140340 144530      MOV      #140340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3193 033246 012767 100000 145726      MOV      #100000,$TMP2 ;LOAD TEST LOC. VIRT. ADDR. INTO $TMP2
3194 033254 012702 001202          MOV      #$TMP2,R2    ;LOAD ADDRESS OF $TMP2 INTO R2
3195 033260 006572 000000          MFPI     @0(R2)      ;READ FROM PHYSICAL 60000
3196 033264 012601          MOV      (USP)+,R1    ;POP USER STACK INTO R1
3197 033266 020001      CMP      R0,R1      ;WAS DATA FETCHED SAME AS STORED
3198 033270 001401      BEQ      20$        ;BRANCH IF CORRECT DATA FETCHED
3199 033272 104023      ERROR   +23        ;WRONG DATA WAS FETCHED
3200                      ;FOR TIGHTER SCOPE LOOP
3201                      ;REPLACE ERROR CALL WITH
3202                      ;'BR 19$' = 000762
3203 033274 012767 002456 144746      MOV      #MGMERR,MMVEC ;SET M.M. VECTOR TO NORMAL ROUTINE
3204 033302 012767 000340 144466      MOV      #00340,PSW  ;GO BACK TO KERNEL MODE, PREVIOUS KERNEL
3205 033310 012767 032674 145572      MOV      #1$,SLPERR  ;SET LOOP POINTER TO START OF TEST
3206 033316 112767 000006 144264      MOV      #6,UIPDR4   ;MAKE UIPDR4 RESIDENT
3207 033324 000423      BR      TST27      ;:BRANCH TO NEXT TEXT
3208
3209
3210 033326 012667 145726          21$: MOV      (KSP)+,TRAPPC ;SAVE PC & PS OF TRAP
3211 033332 012667 145724          MOV      (KSP)+,TRAPPS
3212 033336 016767 144230 145720      MOV      SR0,WASSRO  ;SAVE SR0 FOR ERROR TYPEOUT

```



```

3213 033344 016767 144226 145716      MOV      SR2,WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
3214 033352 042767 160000 144212      BIC      #160000,SRO     ;CLEAR ERROR BITS IN SRO
3215 033360 104026                ERROR    +26             ;TRIED TO READ NON-RESIDENT PAGE
3216                                ;FOR TIGHTER SCOPE LOOP
3217                                ;REPLACE ERROR CALL WITH
3218                                ;A 'NOP' = 000240
3219 033362 016746 145674      MOV      TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
3220 033366 016746 145666      MOV      TRAPPC,-(KSP)
3221 033372 000002                RTI                    ;RETURN TO TEST
3222
3223
3233

```

: *TEST 27 MFPI (SUPERVISOR) WITH SUPER D-SPACE ENABLED
: *

: * THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT PREVIOUS
: * MODE IS CLOCKED CORRECTLY, AND THAT D-SPACE IS NOT ENABLED.

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

```

033374 000004
3234 033376 012767 033412 145502      TST27: SCOPE
3235 033404 012767 033412 145476      MOV      #20$, $LPADR   ;SET LOOP POINTER TO 20$
3236 033412 012700 077400      20$:    MOV      #20$, $LPERR   ;SET LOOP ON ERROR POINTER TO 20$
3237                                MOV      #77400,R0      ;MAKE PAGE 4 IN ALL BUT SUPERVISOR I
3238 033416 010067 136706      ;AND KERNAL I NON-RESIDENT
3239 033422 010067 136602      MOV      R0,KDPDR4     ;KERNAL D-SPACE PAGE 4
3240 033426 010067 144176      MOV      R0,SDPDR4     ;SUPERVISOR D-SPACE PAGE 4
3241 033432 010067 144152      MOV      R0,UDPDR4     ;USER D-SPACE PAGE 4
3242 033436 012700 036514      MOV      R0,UIPDR4     ;USER I-SPACE PAGE 4
3243 033442 010037 060000      MOV      #36514,R0     ;LOAD DATA PATTERN INTO R0
3244 033446 012767 033672 144574      MOV      R0,#60000     ;LOAD DATA PATTERN INTO PHYS. 60000
3245 033454 052767 000002 137034      MOV      #10$,MMVEC    ;SET M.M. VECTOR TO 10$
3246 033462 105067 136622      BIS      #BIT1,MMR3    ;ENABLE SUPERVISOR D-SPACE
3247                                CLR      KIPDR4        ;MAKE KERNAL I-SPACE PAGE 4 NON-RESIDENT
3248                                ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
3249 033466 012767 033474 145414      MOV      #12$, $LPERR   ;SET LOOP ON ERROR POINTER TO 12$
3250 033474 012767 010340 144274      12$:    MOV      #010340,PSW   ;MAKE PREVIOUS MODE SUPERVISOR
3251 033502 012702 100000      MOV      #100000,R2    ;LOAD VIRTUAL ADDRESS INTO R2
3252 033506 006512                MFPI      (R2)          ;READ FROM PHYSICAL 60000
3253 033510 012601                MOV      (KSP)+,R1     ;POP KERNAL STACK INTO R1
3254 033512 020001                CMP      R0,R1         ;WAS DATA FETCHED SAME AS DATA STORED
3255 033514 001401                BEQ      4$            ;BRANCH IF CORRECT DATA FETCHED
3256 033516 104037                ERROR    +37           ;WRONG DATA FETCHED
3257                                ;THE FOLLOWING WILL TEST DSTM=2 MFPI.
3258                                ;
3259 033520 012767 033526 145362      4$:    MOV      #14$, $LPERR   ;SET LOOP ON ERROR POINTER TO 14$
3260 033526 012767 010340 144242      14$:    MOV      #010340,PSW   ;MAKE PREVIOUS MODE SUPERVISOR
3261 033534 012702 100000      MOV      #100000,R2    ;LOAD VIRTUAL ADDRESS INTO R2
3262 033540 006522                MFPI      (R2)+        ;READ FROM PHYSICAL 100000
3263 033542 012601                MOV      (KSP)+,R1     ;POP KERNAL STACK INTO R1
3264 033544 020001                CMP      R0,R1         ;WAS DATA FETCHED SAME AS DATA STORED
3265 033546 001401                BEQ      5$            ;BRANCH IF CORRECTDATA FETCHED
3266 033550 104037                ERROR    +37           ;WRONG DATA FETCHED
3267                                ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
3268                                ;

```

```
3269 033552 012767 033560 145330 5$: MOV #15$,SLPERR ;SET LOOP ON ERROR POINTER TO 15$
3270 033560 012767 010340 1442'0 15$: MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
3271 033566 012702 100000 MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
3272 033572 006537 100000 MFPI @100000 ;READ FROM PHYSICAL 100000
```

```

3274 033576 012601      MOV      (KSP)+,R1      ;POP KERNAL STACK INTO R1
3275 033600 020001      CMP      R0,R1         ;WAS DATA FETCHED SAME AS DATA STORED
3276 033602 001401      BEQ     6$             ;BRANCH IF CORRECTDATA FETCHED
3277 033604 104037      ERROR   +37           ;WRONG DATA FETCHED
3278                      ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
3279                      ;
3280 033606 012767 033614 145274 6$:      MOV      #16$, $LPERR   ;SET LOOP ON ERROR POINTER TO 16$
3281 033614 012767 010340 144154 16$:     MOV      #010340,PSW    ;MAKE PREVIOUS MODE SUPERVISOR
3282 033622 012702 100002      MOV      #100002,R2    ;LOAD VIRTUAL ADDRESS INTO R2
3283 033626 006542      MFPI    -(R2)         ;READ FROM PHYSICAL 100000
3284 033630 012601      MOV      (KSP)+,R1    ;POP KERNAL STACK INTO R1
3285 033632 020001      CMP      R0,R1         ;WAS DATA FETCHED SAME AS DATA STORED
3286 033634 001401      BEQ     7$             ;BRANCH IF CORRECTDATA FETCHED
3287 033636 104037      ERROR   +37           ;WRONG DATA FETCHED
3288 033640 012767 002456 144402 7$:      MOV      #MMGMERR,MMVEC ;SET M.M. VECTOR TO NORMAL ROUTINE
3289 033646 012767 033412 145234      MOV      #20$, $LPERR   ;SET LOOP ON ERROR POINTER TO START OF TEST
3290 033654 112767 000006 136426      MOV     #6,KIPDR4      ;MAKE KIPDR4 RESIDENT
3291 033662 042767 000002 136626      BIC     #BIT1,MMR3     ;DISABLE SUPERVISOR D-SPACE
3292 033670 000426      BR      TST30         ;:BRANCH TO NEXT TEST
3293
3294 033672 016767 143674 145364 10$:     MOV     MMR0,WASSR0    ;SAVE MMR0 FOR ERROR TYPEOUT
3295 033700 016767 143670 145360      MOV     MMR1,WASSR1    ;SAVE MMR1 FOR ERROR TYPEOUT
3296 033706 016767 143664 145354      MOV     MMR2,WASSR2    ;SAVE MMR2 FOR ERROR TYPEOUT
3297 033714 016767 136576 145350      MOV     MMR3,WASSR3    ;SAVE MMR3 FOR ERROR TYPEOUT
3298 033722 012667 145332      MOV     (KSP)+,TRAPPC  ;SAVE PC & PS OF TRAP
3299 033726 012667 145330      MOV     (KSP)+,TRAPPS  ;
3300 033732 104040      ERROR   +40           ;TRIED TO READ NON-RESIDENT PAGE
3301 033734 016746 145322      MOV     TRAPPS,-(KSP)  ;PUT PC & PS OF TRAP ON STACK
3302 033740 016746 145314      MOV     TRAPPC,-(KSP)  ;
3303 033744 000002      RTI
3304
3314

```

```

:*****
:*TEST 30      MTP1(SUPERVISOR) WITH SUPER. D-SPACE ENABLED
:*
:*      THIS TEST USES THE 'MTP1' INSTRUCTION TO ENSURE THAT THE
:*      PREVIOUS MODE IS CLOKED CORRECTLY, AND THAT D-SPACE IS NOT
:*      ENABLED.
:*      IF THE CORRECT MODE IS NOT ENABLED, A NON-RESIDENT ABORT
:*      WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
:*
:*****

```

```

3315 033746 000004      TST30: SCOPE
3316 033750 012767 034264 144272 20$:      MOV     #10$,MMVEC     ;SET M.M. VECTOR TO 10$
3317 033756 052767 000002 136532      BIS     #BIT1,MMR3     ;ENABLE SUPERVISOR D-SPACE
3318                      ;THIS WILL TEST DSTM = 1 MTP1
3319 033764 012767 034010 145116      MOV     #13$, $LPERR   ;SET LOOP ON ERROR POINTER TO 13$
3320 033772 012767 010340 143776      MOV     #010340,PSW    ;MAKE PREVIOUS MODE SUPERVISOR
3321 034000 012705 100000      MOV     #100000,R5     ;LOAD VIRTUAL ADDRESS INTO R5
3322 034004 012703 125252      MOV     #125252,R3     ;LOAD TEST DATA INTO R3
3323 034010 010346 13$:      MOV     R3,-(KSP)     ;PUSH TEST DATA ONTO KERNAL STACK
3324 034012 105067 136272      CLRB   KIPDR4         ;MAKE KERNAL PAGE 4 NON-RESIDENT
3325 034016 006615      MTP1   (R5)           ;LOAD TEST DATA INTO PHYSICAL 100000
3326 034020 112767 000006 136262      MOV     #006,KIPDR4    ;MAKE KERNAL I PAGE 4 RESIDENT
3327 034026 011504      MOV     (R5),R4       ;READ FROM ADDRESS 100000
3328 034030 020304      CMP     R3,R4         ;SEE IF DATA WAS STORED AT CORRECT PLACE
3329 034032 001401      BEQ     4$             ;BRANCH IF STORE WAS CORRECT

```

```

3330 034034 104035          ERROR +35          :INCORRECT STORE
3331          :THIS WILL TEST DSTM = 3 MTP1
3332 034036 012767 034056 145044 4$: MOV #14$, $LPERR :SET LOOP ON ERROR POINTER TO 14$
3333 034044 012767 010340 143724 MOV #010340, PSW :MAKE PREVIOUS TYPE SUPERVISOR
3334 034052 012703 052525 MOV #52525, R3 :LOAD TEST DATA INTO R3
3335 034056 010346 14$: MOV R3, -(KSP) :PUSH TEST DATA ON KERNAL STACK
3336 034060 105067 136224 CLR# KIPDR4 :MAKE KERNAL I PAGE 4 NON-RESIDENT
3337 034064 006637 100000 MTP1 @#100000 :LOAD TEST DATA INTO PHYSICAL 100000
3338 034070 112767 000006 136212 MOV# #006, KIPDR4 :MAKE KERNAL I PAGE 4 RESIDENT
3339 034076 013704 100000 MOV @#100000, R4 :READ FROM ADDRESS 100000
3340 034102 020304 CMP R3, R4 :SEE IF DATA WAS STORED CORRECTLY
3341 034104 001401 BEQ 5$ :BRANCH IF STORED CORRECTLY
3342 034106 104035          ERROR +35          :INCORRECT STORE
3343          :THIS WILL TEST DSTM = 4 MTP1
3344 034110 012767 034130 144772 5$: MOV #15$, $LPERR :SET LOOP ON ERROR POINTER TO 15$
3345 034116 012767 010340 143652 MOV #010340, PSW :MAKE PREVIOUS TYPE SUPERVISOR
3346 034124 012703 125252 MOV #125252, R3 :LOAD TEST DATA INTO R3
3347 034130 010346 15$: MOV R3, -(KSP) :PUSH TEST DATA ON KERNAL STACK
3348 034132 012705 100002 MOV #100002, R5 :LOAD VIRTUAL ADDRESS INTO R5
3349 034136 105067 136146 CLR# KIPDR4 :MAKE KERNAL I PAGE 4 NON-RESIDENT
3350 034142 006645 MTP1 -(R5) :LOAD TEST DATA INTO PHYSICAL 100000
3351 034144 112767 000006 136136 MOV# #006, KIPDR4 :MAKE KERNAL I PAGE 4 RESIDENT
3352 034152 013704 100000 MOV @#100000, R4 :READ FROM ADDRESS 100000
3353 034156 020304 CMP R3, R4 :SEE IF DATA WAS STORED CORRECTLY
3354 034160 001401 BEQ 6$ :BRANCH IF STORED CORRECTLY
3355 034162 104035          ERROR +35          :INCORRECT STORE
3356          :THIS WILL TEST DSTM = 6 MTP1
3357 034164 012767 034206 144716 6$: MOV #16$, $LPERR :SET LOOP ON ERROR POINTER TO 16$
3358 034172 012767 010340 143576 MOV #010340, PSW :MAKE PREVIOUS TYPE SUPERVISOR
3359 034200 012703 052525 MOV #52525, R3 :LOAD TEST DATA INTO R3
3360 034204 005005 CLR R5 :MAKE R5 ZERO
3361 034206 010346 16$: MOV R3, -(KSP) :PUSH TEST DATA ON KERNAL STACK
3362 034210 105067 136074 CLR# KIPDR4 :MAKE KERNAL I PAGE 4 NON-RESIDENT
3363 034214 006665 100000 MTP1 100000(R5) :LOAD TEST DATA INTO PHYSICAL 100000
3364 034220 112767 000006 136062 MOV# #006, KIPDR4 :MAKE KERNAL I PAGE 4 RESIDENT
3365 034226 013704 100000 MOV @#100000, R4 :READ FROM ADDRESS 100000
3366 034232 020304 CMP R3, R4 :SEE IF DATA WAS STORED CORRECTLY
3367 034234 001401 BEQ 7$ :BRANCH IF STORED CORRECTLY
3368 034236 104035          ERROR +35          :INCORRECT STORE
3369 034240 012767 033750 144642 7$: MOV #20$, $LPERR :SET LOOP ON ERROR POINTER TO START OF TEST
3370 034246 012767 002456 143774 MOV #MMGMERR, MMVEC :RESTORE M.M. VECTOR TO NORMAL ROUTINE
3371 034254 042767 000002 136234 BIC #BIT1, MMR3 :DISABLE SUPERVISOR D-SPACE
3372 034262 000410 BR TST31 :BRANCH TO NEXT TEST
3373 034264 016700 143302 10$: MOV MMR0, R0 :SAVE MMR0 FOR ERROR TYPEOUT
3374 034270 016701 143300 MOV MMR1, R1 :SAVE MMR1 FOR ERROR TYPEOUT
3375 034274 016702 143276 MOV MMR2, R2 :SAVE MMR2 FOR ERROR TYPEOUT
3376 034300 104036          ERROR +36          :TRIED TO LOAD A NON-RESIDENT PAGE 4
3377 034302 000002          RTI :RETURN TO TEST
3378
3379

```

```

:*****
:*TEST 31          MTP1 (USER) WITH USER D-SPACE ENABLED
:*
:*          THIS TEST USES THE 'MTP1' INSTRUCTION TO ENSURE THAT THE
:*          PREVIOUS MODE IS CLOKED CORRECTLY, AND THAT D-SPACE IS NOT
:*          ENABLED.
:*          IF THE CORRECT MODE IS NOT ENABLED, A NON-RESIDENT ABORT
:*          WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
:*

```

```

:*****
:
TST31: SCOPE
:
3380 034304 000004          20$: MOV      #77400,SIPDR4      ;MAKE SIPDR4 NON-RESIDENT
3381 034306 012767 077400 135674      MOV      #77406,UIPDR4      ;MAKE UIPDR4 RESIDENT
3382 034314 012767 077406 143266      MOV      #10$,MMVEC        ;SET M.M. VECTOR TO 10$
3383 034322 012767 034636 143720      BIS      #BIT0,MMR3        ;ENABLE USER D-SPACE
3384 034330 052767 000001 136160      ;THIS WILL TEST DSTM = 1 MTP1
3385 034336 012767 034362 144544      MOV      #13$,SLPERR       ;SET LOOP ON ERROR POINTER TO 13$
3386 034344 012767 030340 143424      MOV      #030340,PSW       ;MAKE PREVIOUS MODE USER
3387 034352 012705 100000          MOV      #100000,R5        ;LOAD VIRTUAL ADDRESS INTO R5
3388 034356 012703 125252          MOV      #125252,R3        ;LOAD TEST DATA INTO R3
3389 034362 010346          13$: MOV      R3,-(KSP)        ;PUSH TEST DATA ONTO KERNAL STACK
3390 034364 105067 135720          CLRB    KIPDR4            ;MAKE KERNAL PAGE 4 NON-RESIDENT
3391 034370 006615          MTP1    (R5)              ;LOAD TEST DATA INTO PHYSICAL 100000
3392 034372 112767 000006 135710      MOV     #006,KIPDR4       ;MAKE KERNAL I PAGE 4 RESIDENT
3393 034400 011504          MOV     (R5),R4           ;READ FROM ADDRESS 100000
3394 034402 020304          CMP     R3,R4            ;SEE IF DATA WAS STORED AT CORRECT PLACE
3395 034404 001401          BEQ    4$                ;BRANCH IF STORE WAS CORRECT
3396 034406 104035          ERROR  +35              ;INCORRECT STORE
3397          ;THIS WILL TEST DSTM = 3 MTP1
3398 034410 012767 034430 144472      4$: MOV      #14$,SLPERR       ;SET LOOP ON ERROR POINTER TO 14$
3399 034416 012767 030340 143352      MOV      #030340,PSW       ;MAKE PREVIOUS TYPE USER
3400 034424 012703 052525          MOV      #52525,R3        ;LOAD TEST DATA INTO R3
3401 034430 010346          14$: MOV      R3,-(KSP)        ;PUSH TEST DATA ON KERNAL STACK
3402 034432 105067 135652          CLRB    KIPDR4            ;MAKE KERNAL I PAGE 4 NON-RESIDENT
3403 034436 006637 100000          MTP1    @#100000         ;LOAD TEST DATA INTO PHYSICAL 100000
3404 034442 112767 000006 135640      MOV     #006,KIPDR4       ;MAKE KERNAL I PAGE 4 RESIDENT
3405 034450 013704 100000          MOV     @#100000,R4       ;READ FROM ADDRESS 100000
3406 034454 020304          CMP     R3,R4            ;SEE IF DATA WAS STORED CORRECTLY
3407 034456 001401          BEQ    5$                ;BRANCH IF STORED CORRECTLY
3408 034460 104035          ERROR  +35              ;INCORRECT STORE
3409          ;THIS WILL TEST DSTM = 4 MTP1
3410 034462 012767 034502 144420      5$: MOV      #15$,SLPERR       ;SET LOOP ON ERROR POINTER TO 15$
3411 034470 012767 030340 143300      MOV      #030340,PSW       ;MAKE PREVIOUS TYPE USER
3412 034476 012703 125252          MOV      #125252,R3        ;LOAD TEST DATA INTO R3
3413 034502 010346          15$: MOV      R3,-(KSP)        ;PUSH TEST DATA ON KERNAL STACK
3414 034504 012705 100002          MOV      #100002,R5        ;LOAD VIRTUAL ADDRESS INTO R5
3415 034510 105067 135574          CLRB    KIPDR4            ;MAKE KERNAL I PAGE 4 NON-RESIDENT
3416 034514 006645          MTP1    -(R5)            ;LOAD TEST DATA INTO PHYSICAL 100000
3417 034516 112767 000006 135564      MOV     #006,KIPDR4       ;MAKE KERNAL I PAGE 4 RESIDENT
3418 034524 013704 100000          MOV     @#100000,R4       ;READ FROM ADDRESS 100000
3419 034530 020304          CMP     R3,R4            ;SEE IF DATA WAS STORED CORRECTLY
3420 034532 001401          BEQ    6$                ;BRANCH IF STORED CORRECTLY
3421 034534 104035          ERROR  +35              ;INCORRECT STORE
3422          ;THIS WILL TEST DSTM = 6 MTP1
3423 034536 012767 034560 144344      6$: MOV      #16$,SLPERR       ;SET LOOP ON ERROR POINTER TO 16$
3424 034544 012767 030340 143224      MOV      #030340,PSW       ;MAKE PREVIOUS TYPE USER
3425 034552 012703 052525          MOV      #52525,R3        ;LOAD TEST DATA INTO R3
3426 034556 005005          CLR     R5                ;MAKE R5 ZERO
3427 034560 010346          16$: MOV      R3,-(KSP)        ;PUSH TEST DATA ON KERNAL STACK
3428 034562 105067 135522          CLRB    KIPDR4            ;MAKE KERNAL I PAGE 4 NON-RESIDENT
3429 034566 006665 100000          MTP1    100000(R5)       ;LOAD TEST DATA INTO PHYSICAL 100000
3430 034572 112767 000006 135510      MOV     #006,KIPDR4       ;MAKE KERNAL I PAGE 4 RESIDENT
3431 034600 013704 100000          MOV     @#100000,R4       ;READ FROM ADDRESS 100000
3432 034604 020304          CMP     R3,R4            ;SEE IF DATA WAS STORED CORRECTLY
3433 034606 001401          BEQ    7$                ;BRANCH IF STORED CORRECTLY

```

```

3434 034610 104035          ERROR +35          :INCORRECT STORE
3435 034612 012767 034306 144270 7$: MOV #20$,SLPERR :SET LOOP ON ERROR POINTER TO START OF TEST
3436 034620 012767 002456 143422 MOV #MGMERR,MMVEC :RESTORE M.M. VECTOR TO NORMAL ROUTINE
3437 034626 042767 000001 135662 BIC #BIT0,MMR3 :DISABLE USER D-SPACE
3438 034634 000410          BR TST32 :BRANCH TO NEXT TEST
3439 034636 016700 142730 10$: MOV MMR0,R0 :SAVE MMR0 FOR ERROR TYPEOUT
3440 034642 016701 142726 MOV MMR1,R1 :SAVE MMR1 FOR ERROR TYPEOUT
3441 034646 016702 142724 MOV MMR2,R2 :SAVE MMR2 FOR ERROR TYPEOUT
3442 034652 104036          ERROR +36          :TRIED TO LOAD A NON-RESIDENT PAGE 4
3443 034654 000002          RTI :RETURN TO TEST
3452
  
```

```

:*****
:TEST 32 MFPI(PREVIOUS=CURRENT=KERNEL)
:
: THIS TEST CHECKS THAT IF BOTH PREVIOUS AND CURRENT MODES
: ARE KERNEL, AND THE SOURCE MODE IS 0, THE DESTINATION
: STACK IS NOT DECREMENTED BEFORE ACCESS.
: 'MFPI KSP' SHOULD PUSH THE NON-DECREMENTED VALUE
: OF KSP (1100) ONTO THE STACK (AT LOC. 1076).
:*****
  
```

```

3453 034656 000004          TST32: SCOPE
3453 034660 112767 000006 135442 1$: MOVB #6,KDPDR4 :MAKE KDPDR4 RESIDENT
3454 034666 005037 177776 CLR @PSW :SET PREVIOUS = CURRENT = KERNEL
3455 034672 012700 001100 MOV #STACK,R0 :SETUP VALUE FOR STACK POINTER
3456 034676 010006 MOV R0,KSP :LOAD STACK POINTER
3457 034700 006506 MFPI KSP :THE VALUE 'STACK' SHOULD BE PUSHED
3458 :BEFORE BEING DECREMENTED
3459 034702 011601 MOV (KSP),R1 :READ DATA WHICH WAS PUSHED
3460 034704 020001 CMP R0,R1 :WAS THE ORIGINAL VALUE OF THE
3461 :STACK POINTER PUSHED?
3462 034706 001401 BEQ 2$ :BRANCH IF YES
3463 034710 104037 ERROR +37 :MFPI FETCHED WRONG DATA
3464 :FOR TIGHTER SCOPE LOOP
3465 :REPLACE ERROR CALL WITH
3466 :'BR 1$' = 000766
3467 034712 005740 2$: TST -(R0) :SETUP EXPECTED STACK POINTER VALUE
3468 034714 020600 CMP KSP,R0 :WAS THE STACK POINTER DECREMENTED?
3469 034716 001401 BEQ 3$ :BRANCH IF YES
3470 034720 104025 ERROR +25 :STACK NOT PUSHED BY THE MFPI
3471 :FOR TIGHTER SCOPE LOOP
3472 :REPLACE ERROR CALL WITH
3473 :'BR 1$' = 000762
3474 034722 012706 001100 3$: MOV #STACK,KSP :RESTORE STACK POINTER
3484
  
```

```

:*****
:TEST 33 MFPI (SUPERVISOR) WITH SUPER D-SPACE ENABLED
:
: THIS TEST CHECKS TO SEE THAT THE REFERENCE IS TO D-SPACE
: IF THE INSTRUCTION IS AN MFPI.
:
: IF THE CORRECT MODE IS NOT ENABLED, A NON-RESIDENT ABORT
: WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
:*****
  
```

```

3485 034726 000004          TST33: SCOPE
3485 034730 012767 034752 144150 MOV #20$,SLPADR :SET LOOP POINTER TO 20$
3486 034736 012767 034752 144144 MOV #20$,SLPERR :SET LOOP ON ERROR POINTER TO 20$
3487 034744 012767 000600 135316 MOV #600,SDPAR4 :MAP SDPAR4 TO 12K
3488 034752 012700 077400 20$: MOV #77400,R0 :MAKE PAGE 4 IN ALL BUT SUPERVISOR D
  
```

```

3489                                     :AND KERNAL I NON-RESIDENT
3490 034756 010067 135346                MOV    R0,KDPDR4                :KERNAL D-SPACE PAGE 4
3491 034762 010067 135222                MOV    R0,SIPDR4                :SUPERVISOR I-SPACE PAGE 4
3492 034766 010067 142636                MOV    R0,UDPDR4                :USER D-SPACE PAGE 4
3493 034772 010067 142612                MOV    R0,UIPDR4                :USER I-SPACE PAGE 4
3494 034776 012767 077406 135224        MOV    #77406,SDPDR4            :MAKE SDPDR4 RESIDENT
3495 035004 012700 036514                MOV    #36514,R0                :LOAD DATA PATTERN INTO R0
3496 035010 010037 100000                MOV    R0,#100000              :LOAD DATA PATTERN INTO PHYS. 100000
3497 035014 012767 035246 143226        MOV    #10$,MMVEC               :SET M.M. VECTOR TO 10$
3498 035022 052767 000002 135466        BIS    #BIT1,MMR3               :ENABLE SUPERVISOR D-SPACE
3499 035030 105067 135254                CLR    KIPDR4                   :MAKE KERNAL I-SPACE PAGE 4 NON-RESIDENT
3500                                     :THE FOLLOWING WILL TEST DSTM=1 MFPD
3501                                     :
3502 035034 012767 035042 144046 2$:    MOV    #12$,SLPERR              :SET LOOP ON ERROR POINTER TO 12$
3503 035042 012767 010340 142726 12$:    MOV    #010340,PSW             :MAKE PREVIOUS MODE SUPERVISOR
3504 035050 012702 100000                MOV    #100000,R2              :LOAD VIRTUAL ADDRESS INTO R2
3505 035054 106512                        MFPD    (R2)                    :READ FROM PHYSICAL 100000
3506 035056 012601                        MOV    (KSP)+,R1               :POP KERNAL STACK INTO R1
3507 035060 020001                        CMP    R0,R1                   :WAS DATA FETCHED SAME AS DATA STORED
3508 035062 001401                        BEQ    4$                      :BRANCH IF CORRECT DATA WAS FETCHED
3509 035064 104037                        ERROR  +37                     :WRONG DATA WAS FETCHED
3510                                     :THE FOLLOWING WILL TEST DSTM=2 MFPD
3511                                     :
3512 035066 012767 035074 144014 4$:    MOV    #14$,SLPERR              :SET LOOP ON ERROR POINTER TO 14$
3513 035074 012767 010340 142674 14$:    MOV    #010340,PSW             :MAKE PREVIOUS MODE SUPERVISOR
3514 035102 012702 100000                MOV    #100000,R2              :LOAD VIRTUAL ADDRESS INTO R2
3515 035106 106522                        MFPD    (R2)+                  :READ FROM PHYSICAL 100000
3516 035110 012601                        MOV    (KSP)+,R1               :POP KERNAL STACK INTO R1
3517 035112 020001                        CMP    R0,R1                   :WAS DATA FETCHED SAME AS DATA STORED
3518 035114 001401                        BEQ    5$                      :BRANCH IF CORRECT DATA WAS FETCHED
3519 035116 104037                        ERROR  +37                     :WRONG DATA WAS FETCHED
3520                                     :THE FOLLOWING WILL TEST DSTM=3 MFPD
3521                                     :
3522 035120 012767 035126 143762 5$:    MOV    #15$,SLPERR              :SET LOOP ON ERROR POINTER TO 15$
3523 035126 012767 010340 142642 15$:    MOV    #010340,PSW             :MAKE PREVIOUS MODE SUPERVISOR
3524 035134 012702 100000                MOV    #100000,R2              :LOAD VIRTUAL ADDRESS INTO R2
3525 035140 106537 100000                MFPD    @#100000              :READ FROM PHYSICAL 100000
3526 035144 012601                        MOV    (KSP)+,R1               :POP KERNAL STACK INTO R1
3527 035146 020001                        CMP    R0,R1                   :WAS DATA FETCHED SAME AS DATA STORED
3528 035150 001401                        BEQ    6$                      :BRANCH IF CORRECT DATA WAS FETCHED
3529 035152 104037                        ERROR  +37                     :WRONG DATA WAS FETCHED
3530                                     :THE FOLLOWING WILL TEST DSTM=4 MFPD
3531                                     :
3532 035154 012767 035162 143726 6$:    MOV    #16$,SLPERR              :SET LOOP ON ERROR POINTER TO 16$
3533 035162 012767 010340 142606 16$:    MOV    #010340,PSW             :MAKE PREVIOUS MODE SUPERVISOR
3534 035170 012702 100002                MOV    #100002,R2              :LOAD VIRTUAL ADDRESS INTO R2
3535 035174 106542                        MFPD    -(R2)                  :READ FROM PHYSICAL 100000
3536 035176 012601                        MOV    (KSP)+,R1               :POP KERNAL STACK INTO R1
3537 035200 020001                        CMP    R0,R1                   :WAS DATA FETCHED SAME AS DATA STORED
3538 035202 001401                        BEQ    7$                      :BRANCH IF CORRECT DATA WAS FETCHED
3539 035204 104037                        ERROR  +37                     :WRONG DATA WAS FETCHED
3540 035206 012767 002456 143034 7$:    MOV    #MMGMERR,MMVEC          :RESTORE M.M. VECTOR
3541 035214 012767 034752 143666        MOV    #20$,SLPERR              :SET LOOP ON ERROR POINTER TO START OF ROUTINE
3542 035222 042767 000002 135266        BIC    #BIT1,MMR3               :DISABLE SUPERVISOR D-SPACE
3543 035230 012700 077406                MOV    #77406,R0                :SET UP R0 FOR 4K RESIDENT R/W
3544 035234 010067 135050                MOV    R0,KIPDR4               :MAKE KIPAR4 RESIDENT
3545 035240 010067 142364                MOV    R0,UDPDR4               :MAKE UDPDR4 RESIDENT
  
```

```

3546 035244 000423          BR      TST34          ;:BRANCH TO NEXT TEST
3547 035246 016767 142320 1440*0 10$: MOV    MMR0,WASSRO    ;:SAVE MMR0 FOR ERROR TYPEOUT
3548 035254 016767 142314 144004      MOV    MMR1,WASSR1    ;:SAVE MMR1 FOR ERROR TYPEOUT
3549 035262 016767 142310 144000      MOV    MMR2,WASSR2    ;:SAVE MMR2 FOR ERROR TYPEOUT
3550 035270 012667 143764      MOV    (KSP)+,TRAPPC ;:SAVE PC & PS OF TRAP
3551 035274 012667 143762      MOV    (KSP)+,TRAPPS
3552 035300 104040          ERROR  +40          ;:TRIED TO READ NON-RESIDENT PAGE
3553 035302 016746 143754      MOV    TRAPPS,-(KSP) ;:PUT PC & PS OF TRAP ON STACK
3554 035306 016746 143746      MOV    TRAPPC,-(KSP)
3555 035312 000002          RTI              ;:RETURN TO TEST
3556
3568
  
```

 *TEST 34 MFPI (USER/PREV USER) WITH USER D-SPACE ENABLED

THIS TEST CHECKS THAT IF THE INSTRUCTION IS EITHER MFPI OR
 MTPI AND BOTH THE PRESENT AND PREVIOUS MODES ARE USER, THEN
 D-SPACE IS USED IF IT IS ENABLED. IN THIS WAY AN OPERATING
 SYSTEM CAN MAKE PROPRIETARY CODE 'EXECUTE ONLY' FOR THE USER.

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

```

          035314 000004          TST34: SCOPE
3569 035316 012767 035332 143562      MOV    #20$,SLPADR    ;:SET LOOP POINTER TO 20$
3570 035324 012767 035332 143556      MOV    #20$,SLPERR    ;:SET LOOP ON ERROR POINTER TO 20$
3571 035332 012767 000600 142330 20$: MOV    #600,UDPAR4    ;:MAP UDPAR4 TO 12K
3572 035340 012700 036514          MOV    #36514,R0      ;:LOAD DATA PATTERN INTO R0
3573 035344 010037 100000          MOV    R0,#100000     ;:LOAD DATA PATTERN INTO PHYS. 100000
3574 035350 012767 035606 142672      MOV    #10$,MMVEC     ;:SET M.M. VECTOR TO 10$
3575 035356 052767 000001 135132      BIS    #BIT0,MMR3     ;:ENABLE USER D-SPACE
3576 035364 105067 134640          CLRB   SDPDR4        ;:MAKE SDPDR4 NON-RESIDENT
3577 035370 105067 134714          CLRB   KIPDR4        ;:MAKE KERNAL I-SPACE PAGE 4 NON-RESIDENT
3578
3579          ;:THE FOLLOWING WILL TEST DSTM=1 MFPI
3580 035374 012767 035402 143506 2$: MOV    #12$,SLPERR    ;:SET LOOP ON ERROR POINTER TO 12$
3581 035402 012767 170340 142366 12$: MOV    #170340,PSW    ;:SET PREVIOUS MODE TO USER
3582 035410 012702 100000          MOV    #100000,R2     ;:LOAD VIRTUAL ADDRESS INTO R2
3583 035414 006512          MFPI   (R2)          ;:READ FROM PHYSICAL 100000
3584 035416 012601          MOV    (USP)+,R1     ;:POP USER STACK INTO R1
3585 035420 020001          CMP    R0,R1         ;:WAS DATA FETCHED SAME AS DATA STORED
3586 035422 001401          BEQ    4$            ;:BRANCH IF CORRECT DATA WAS FETCHED
3587 035424 104037          ERROR  +37          ;:WRONG DATA WAS FETCHED
3588          ;:THE FOLLOWING WILL TEST DSTM=2 MFPI
3589
3590 035426 012767 035434 143454 4$: MOV    #14$,SLPERR    ;:SET LOOP ON ERROR POINTER TO 14$
3591 035434 012767 170340 142334 14$: MOV    #170340,PSW    ;:SET PREVIOUS MODE TO USER
3592 035442 012702 100000          MOV    #100000,R2     ;:LOAD VIRTUAL ADDRESS INTO R2
3593 035446 006522          MFPI   (R2)+         ;:READ FROM PHYSICAL 100000
3594 035450 012601          MOV    (USP)+,R1     ;:POP USER STACK INTO R1
3595 035452 020001          CMP    R0,R1         ;:WAS DATA FETCHED SAME AS DATA STORED
3596 035454 001401          BEQ    5$            ;:BRANCH IF CORRECT DATA WAS FETCHED
3597 035456 104037          ERROR  +37          ;:WRONG DATA WAS FETCHED
3598          ;:THE FOLLOWING WILL TEST DSTM=3 MFPI
3599
3600 035460 012767 035466 143422 5$: MOV    #15$,SLPERR    ;:SET LOOP ON ERROR POINTER TO 15$
3601 035466 012767 170340 142302 15$: MOV    #170340,PSW    ;:SET PREVIOUS MODE TO USER
  
```



```

3602 035474 012702 100000      MOV      #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3603 035500 006537 100000      MFPI     @#100000      ;READ FROM PHYSICAL 100000
3604 035504 012601      MOV      (USP)+,R1     ;POP USER STACK INTO R1
3605 035506 020001      CMP      R0,R1        ;WAS DATA FETCHED SAME AS DATA STORED
3606 035510 001401      BEQ     6$           ;BRANCH IF CORRECT DATA WAS FETCHED
3607 035512 104037      ERROR   +37         ;WRONG DATA WAS FETCHED
3608                                ;THE FOLLOWING WILL TEST DSTM=4 MFPI
3609                                ;
3610 035514 012767 035522 143366 6$:  MOV      #16$,SLPERR   ;SET LOOP ON ERROR POINTER TO 16$
3611 035522 012767 170340 142246 16$: MOV      #170340,PSW   ;SET PREVIOUS MODE TO USER
3612 035530 012702 100002      MOV      #100002,R2   ;LOAD VIRTUAL ADDRESS INTO R2
3613 035534 006542      MFPI     -(R2)        ;READ FROM PHYSICAL 100000
3614 035536 012601      MOV      (USP)+,R1     ;POP USER STACK INTO R1
3615 035540 020001      CMP      R0,R1        ;WAS DATA FETCHED SAME AS DATA STORED
3616 035542 001401      BEQ     7$           ;BRANCH IF CORRECT DATA WAS FETCHED
3617 035544 104037      ERROR   +37         ;WRONG DATA WAS FETCHED
3618 035546 012767 002456 142474 7$:  MOV      #MMGMERR,MMVEC ;RESTORE M.M. VECTOR
3619 035554 012767 035332 143326      MOV      #20$,SLPERR  ;SET LOOP ON ERROR POINTER TO START OF ROUTINE
3620 035562 042767 000001 134726      BIC      #BIT0,MMR3   ;DISABLE USER D-SPACE
3621 035570 012767 000340 142200      MOV      #340,PSW    ;MAKE PRESENT MODE KERNAL
3622 035576 112767 000006 134504      MOV      #6,KIPDR4   ;MAKE KIPDR4 RESIDENT
3623 035604 000423      BR      $EOP         ;BRANCH TO END-OF-PASS
3624 035606 016767 141760 143450 10$: MOV      MMR0,WASSRO  ;SAVE MMR0 FOR ERROR TYPEOUT
3625 035614 016767 141754 143444      MOV      MMR1,WASSR1  ;SAVE MMR1 FOR ERROR TYPEOUT
3626 035622 016767 141750 143440      MOV      MMR2,WASSR2  ;SAVE MMR2 FOR ERROR TYPEOUT
3627 035630 012667 143424      MOV      (KSP)+,TRAPPC ;SAVE PC & PS OF TRAP
3628 035634 012667 143422      MOV      (KSP)+,TRAPPS
3629 035640 104040      ERROR   +40         ;TRIED TO READ NON-RESIDENT PAGE
3630 035642 016746 143414      MOV      TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
3631 035646 016746 143406      MOV      TRAPPC,-(KSP)
3632 035652 000002      RTI
3633
    
```

3635

```

.SBTTL END OF PASS ROUTINE
:*****
:*INCREMENT THE PASS NUMBER ($PASS)
:*TYPE 'END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYY'
:*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
:*IF SW12=1 INHIBIT TRACE TRAP
:*IF THERES A MONITOR GO TO IT
:*IF THERE ISN'T JUMP TO LOOP
$EOP:
035654                                SCOPE
035654 000004                          CLR      $STNM      ;;ZERO THE TEST NUMBER
035656 005067 143220                    INC      $PASS      ;;INCREMENT THE PASS NUMBER
035662 005267 143344                    BIC      #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
035666 042767 100000 143336            DEC      (PC)+      ;;LOOP?
035674 005327                          $EOPCT: .WORD     1
035676 000001                          BGT      $DOAGN     ;;YES
035700 003072                          MOV      (PC)+,@(PC)+ ;;RESTORE COUNTER
035702 012737                          $ENDCT: .WORD     1
035704 000001                          $EOPCT
035706 035676                          TYPE     ,65$      ;;TYPE ASCIZ STRING
035710 104401 035716                    BR       64$        ;;GET OVER THE ASCIZ
035714 000407                          ;;65$: .ASCIZ <12><15>/END PASS #/
035734                                64$:
035734 016746 143272                    MOV      $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT
;;TYPE PASS NUMBER
035740 104405                          TYPDS
035742 104401 035750                    TYPE     ,67$      ;;GO TYPE--DECIMAL ASCII WITH SIGN
035746 000421                          BR       66$        ;;TYPE ASCIZ STRING
;;67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
036012                                66$:
036012 016746 143074                    MOV      $ERTTL,-(SP) ;;SAVE $ERTTL FOR TYPEOUT
;;TOTAL NUMBER OF ERRORS
036016 104405                          TYPDS
036020 104401 001221                    TYPE     ,SCLRF     ;;GO TYPE--DECIMAL ASCII WITH SIGN
036024 005067 143062                    CLR      $ERTTL     ;;TYPE CARRIAGE RETURN, LINE FEED
036030 013700 000042                    $GET42: MOV      @#42,R0 ;;CLEAR ERROR TOTAL
036034 001414                          BEQ      $DOAGN     ;;GET MONITOR ADDRESS
036036 005046                          CLR      -(SP)      ;;BRANCH IF NO MONITOR
036040 012746 036046                    MOV      #$CLR.T,-(SP) ;;INSURE THE 'T' BIT IS CLEAR
036044 000426                          BR       $RTRN      ;;SETUP FOR AN RTI OR RTT
;;GO DO AN RTI OR RTT TO LOAD THE PSW
;;WITH A CLEARED 'T' BIT
036046                                $CLR.T:
036046 013700 000042                    MOV      @#42,R0   ;;INSURE R0 CONTAINS THE MONITORS
036052 001405                          BEQ      $DOAGN     ;;RETURN ADDRESS
036054 000005                          RESET
036056 004710                          $ENDAD: JSR      PC,(R0) ;;CLEAR THE WORLD
036060 000240                          NOP      ;;GO TO MONITOR
036062 000240                          NOP      ;;SAVE ROOM
036064 000240                          NOP      ;;FOR
036066                                ;;ACT11
036066 104400                          $DOAGN: TRAP
036070 042716 000020                    BIC      #20,(SP)   ;;PUSH OLD PSW AND PC ON STACK
036074 032777 010000 143036            BIT      #BIT12,@SWR ;;CLEAR THE 'T' BIT
036102 001005                          BNE     1$         ;;RUN WITH TRACE TRAP?
036104 005167 143200                    COM     $TBIT      ;;BR IF NO
036110 100402                          BMI     1$         ;;IS IT TIME FOR TRACE TRAP
036112 052716 000020                    BIS     #20,(SP)   ;;BR IF NO
;;SET TRACE TRAP

```

3636

```
036116 012746 036124 1$: MOV #SLOOP,-(SP) ;;JUMP TO START OF TEST
036122 000002 $RTN: RTI ;;RETURN--THIS IS CHANGED TO
;;AN 'RTI' IF 'RTI' IS A LEGAL
;;INSTRUCTION

036124 $LOOP: JMP @(PC)+ ;;RETURN
036124 000137 $RTNAD: .WORD LOOP
036126 020456 $EMULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
036130 377 377 000 .EVEN

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

036134 104410 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
036134 032777 040000 142774 1$: BIT #BIT14,@SWR ;;LOOP ON PRESENT TEST?
036144 001062 BNE $OVER ;;YES IF SW14=1
:*****START OF CODE FOR THE XOR TESTER*****
036146 000416 $XTSTR: BR 6$ ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
036150 013746 000004 MOV @ERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
036154 012737 036174 000004 MOV #5$,@ERRVEC ;;SET FOR TIMEOUT
036162 005737 177060 TST @177060 ;;TIME OUT ON XOR?
036166 012637 000004 MOV (SP)+,@ERRVEC ;;RESTORE THE ERROR VECTOR
036172 000431 BR $SVLAD ;;GO TO THE NEXT TEST
036174 022626 5$: CMP (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
036176 012637 000004 MOV (SP)+,@ERRVEC ;;RESTORE THE ERROR VECTOR
036202 000417 BR 7$ ;;LOOP ON THE PRESENT TEST
036204 6$:*****END OF CODE FOR THE XOR TESTER*****
036204 032777 000400 142726 BIT #BIT08,@SWR ;;LOOP ON SPEC. TEST?
036212 001404 BEQ 2$ ;;BR IF NO
036214 127767 142720 142660 CMPB @SWR,$STNM ;;ON THE RIGHT TEST? SWR<7:0>
036222 001433 BEQ $OVER ;;BR IF YES
036224 105767 142653 2$: TSTB $ERFLG ;;HAS AN ERROR OCCURRED?
036230 001412 BEQ $SVLAD ;;BR IF NO
036232 032777 001000 142700 BIT #BIT09,@SWR ;;LOOP ON ERROR?
036240 001404 BEQ 4$ ;;BR IF NO
036242 016767 142642 142636 7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
036250 000420 BR $OVER
036252 105067 142625 4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
036256 105267 142620 $SVLAD: INCB $STNM ;;COUNT TEST NUMBERS
036262 116767 142614 142740 MOVB $STNM,$STNM ;;SET TEST NUMBER IN APT MAILBOX
036270 011667 142612 MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
036274 011667 142610 MOV (SP),$LPERR ;;SAVE ERROR LOOP ADDRESS
036300 005067 142706 CLR $ESCAPE ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
036304 112767 000001 142603 MOVB #1,$ERMAX ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
036312 016777 142564 142622 $OVER: MOV $STNM,@DISPLAY ;;DISPLAY TEST NUMBER
036320 016716 142562 MOV $LPADR,(SP) ;;FUDGE RETURN ADDRESS
036324 000002 RTI ;;FIXES PS
```

3637

.SBTTL ERROR HANDLER ROUTINE

```

*****
*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
*AND GO TO ERRYP ON ERROR
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW15=1      HALT ON ERROR
*SW13=1      INHIBIT ERROR TYPEOUTS
*SW10=1      BELL ON ERROR
*SW09=1      LOOP ON ERROR
*CALL
*          ERROR N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
$ERROR:
036326      104410
036326      010067      142626      CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
036330      010167      142624      MOV        R0,$REG0      ;;SAVE THE CONTENTS OF R0
036334      010267      142622      MOV        R1,$REG1      ;;SAVE THE CONTENTS OF R1
036340      010367      142620      MOV        R2,$REG2      ;;SAVE THE CONTENTS OF R2
036344      010467      142616      MOV        R3,$REG3      ;;SAVE THE CONTENTS OF R3
036350      010567      142614      MOV        R4,$REG4      ;;SAVE THE CONTENTS OF R4
036354      010667      142612      MOV        R5,$REG5      ;;SAVE THE CONTENTS OF R5
036360      116767      142516      142666      MOVB       $TSTNM,TESTNO ;;SAVE THE TEST NUMBER
036366      105267      142511      7$:      INCB       $ERFLG      ;;SET THE ERROR FLAG
036372      001775      BEQ        7$      ;;DON'T LET THE FLAG GO TO ZERO
036374      016777      142502      142540      MOV        $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
036402      032777      002000      142530      BIT        #BIT10,@SWR    ;;BELL ON ERROR?
036410      001402      BEQ        1$      ;;NO - SKIP
036412      104401      001214      TYPE      ,SBELL      ;;RING BELL
036416      005267      142470      1$:      INC        $ERTTL      ;;COUNT THE NUMBER OF ERRORS
036422      011667      142470      MOV        (SP),$ERRPC    ;;GET ADDRESS OF ERROR INSTRUCTION
036426      162767      000002      142462      SUB        #2,$ERRPC
036434      117767      142456      142452      MOVB       @$ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
036442      032777      020000      142470      BIT        #BIT13,@SWR    ;;SKIP TYPEOUT IF SET
036450      001004      BNE        20$      ;;SKIP TYPEOUTS
036452      004767      000106      JSR        PC,ERRYP      ;;GO TO USER ERROR ROUTINE
036456      104401      001221      TYPE      ,$CRLF
036462
036462      122767      000001      142554      20$:      CMPB       #APTENV,$ENV    ;;RUNNING IN APT MODE
036470      001007      BNE        2$      ;;NO,SKIP APT ERROR REPORT
036472      116767      142416      000004      MOVB       $ITEMB,21$    ;;SET ITEM NUMBER AS ERROR NUMBER
036500      004767      002052      JSR        PC,$ATY4      ;;REPORT FATAL ERROR TO APT
036504      000      21$:      .BYTE     0
036505      000      .BYTE     0
036506      000777      22$:      BR        22$      ;;APT ERROR LOOP
036510      005777      142424      2$:      TST       @SWR      ;;HALT ON ERROR
036514      100002      BPL        3$      ;;SKIP IF CONTINUE
036516      000000      HALT      ;;HALT ON ERROR!
036520      104410      CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
036522      032777      001000      142410      3$:      BIT        #BIT09,@SWR    ;;LOOP ON ERROR SWITCH SET?
036530      001402      BEQ        4$      ;;BR IF NO
036532      016716      142352      MOV        $LPERR,(SP)   ;;FUDGE RETURN FOR LOOPING
036536      005767      142450      4$:      TST       $ESCAPE      ;;CHECK FOR AN ESCAPE ADDRESS
036542      001402      BEQ        5$      ;;BR IF NONE
036544      016716      142442      MOV        $ESCAPE,(SP)  ;;FUDGE RETURN ADDRESS FOR ESCAPE
036550      5$:
036550      022737      036056      000042      CMP        #SENDAD,@#42  ;;ACT-11 AUTO-ACCEPT?
036556      001001      BNE        6$      ;;BRANCH IF NO
036560      000000      HALT      ;;YES
036562      6$:

```

```

036562 000002          RTI          ;;RETURN
3638
3639 036564 104401 001221  ERR_TYP: TYPE  , $CRLF          ;'CARRIAGE RETURN' & 'LINE FEED'
3640 036570 010046          MOV      R0,-(KSP)      ;SAVE R0.
3641 036572 005000          CLR      R0          ;PICKUP THE ITEM INDEX
3642 036574 153700 001114  BISB    @#SITEMB,R0
3643 036600 001004          BNE     1$          ;IF ITEM NUMBER IS ZERO, JUST
3644                                ;TYPE THE PC OF THE ERROR
3645 036602 016746 142310  MOV     $ERRPC,-(SP)  ;SAVE $ERRPC FOR TYPEOUT
3646                                ;ERROR ADDRESS
3647 036606 104402          TYPDC          ;GO TYPE--OCTAL ASCII(ALL DIGITS)
3648 036610 000522          BR      13$        ;GET OUT
3649 036612 005300 1$:    DEC     R0          ;ADJUST THE INDEX SO THAT IT WILL
3650 036614 006300          ASL     R0          ;WORK FOR THE ERROR TABLE.
3651 036616 006300          ASL     R0
3652 036620 006300          ASL     R0
3653 036622 062700 001320  ADD     #SERRTB,R0  ;FORM TABLE POINTER
3654 036626 012067 000004  MOV     (R0)+,2$    ;PICKUP 'ERROR MESSAGE' POINTER
3655 036632 001404          BEQ     3$          ;SKIP TYPEOUT IF NO POINTER
3656 036634 104401          TYPE          ;TYPE THE 'ERROR MESSAGE'
3657 036636 000000 2$:    .WORD  0          ;'ERROR MESSAGE' POINTER GOES HERE
3658 036640 104401 001221  TYPE    , $CRLF      ;'CARRIAGE RETURN' & 'LINE FEED'
3659 036644 012067 000004 3$:    MOV     (R0)+,4$    ;PICKUP 'DATA HEADER' POINTER
3660 036650 001404          BEQ     5$          ;SKIP TYPEOUT IF 0
3661 036652 104401          TYPE          ;TYPE THE 'DATA HEADER'
3662 036654 000000 4$:    .WORD  0          ;'DATA HEADER' POINTER GOES HERE
3663 036656 104401 001221  TYPE    , $CRLF      ;'CARRIAGE RETURN' & 'LINE FEED'
3664 036662 010146 5$:    MOV     R1,-(KSP)    ;SAVE R1
3665 036664 012001          MOV     (R0)+,R1   ;PICKUP 'DATA TABLE' POINTER
3666 036666 001472          BEQ     12$         ;BR IF NO DATA TO BE TYPED
3667 036670 012000          MOV     (R0)+,R0   ;PICKUP 'DATA FORMAT' POINTER
3668 036672 105710 6$:    TSTB   (R0)         ;IS IT FORMAT 0?
3669 036674 001003          BNE     7$          ;BR IF NO
3670                                ;*THIS CODE IS FOR OCTAL (16-BIT) FORMAT (DF=0)
3671 036676 013146          MOV     @ (R1)+,-(SP) ;SAVE @ (R1)+ FOR TYPEOUT
3672 036700 104402          TYPDC          ;GO TYPE--OCTAL ASCII(ALL DIGITS)
3673 036702 000456          BR      11$
3674                                ;*THIS CODE IS FOR DECIMAL FORMAT (DF=1)
3675 036704 121027 000001 7$:    CMPB   (R0),#1     ;IS IT FORMAT 1?
3676 036710 001003          BNE     8$          ;BRANCH IF NO
3677 036712 013146          MOV     @ (R1)+,-(SP) ;SAVE @ (R1)+ FOR TYPEOUT
3678 036714 104405          TYPDS          ;GO TYPE--DECIMAL ASCII WITH SIGN
3679 036716 000450          BR      11$
3680                                ;*THIS CODE IS FOR BINARY FORMAT (DF=2)
3681 036720 121027 000002 8$:    CMPB   (R0),#2     ;IS IT FORMAT 2
3682 036724 001003          BNE     9$          ;BRANCH IF NO
3683 036726 013146          MOV     @ (R1)+,-(SP) ;SAVE @ (R1)+ FOR TYPEOUT
3684 036730 104406          TYPBN          ;GO TYPE--BINARY ASCII
3685 036732 000442          BR      11$
3686                                ;*THIS CODE IS FOR OCTAL (22-BIT) FORMAT (DF=3)
3687 036734 121027 000003 9$:    CMPB   (R0),#3     ;IS IT FORMAT 3?
3688 036740 001011          BNE     15$         ;BRANCH IF NO
3689 036742 012146          MOV     (R1)+,-(KSP) ;PUT ADDRESS OF FIRST LOC. ON STACK
3690 036744 004767 002660  JSR     PC,$DB20    ;CONVERT TWO LOCS. TO AN ASCII STRING
3691 036750 062716 000003  ADD     #3,(KSP)    ;ONLY NEED 8 CHARACTERS NOT 11
3692 036754 012667 000002  MOV     (KSP)+,10$  ;PUT ADDRESS OF ASCII CHARS. AT 10$
3693 036760 104401          TYPE          ;TYPE OCTAL VALUE OF 22-BIT BINARY NO.

```

```

3694 036762 000000
3695
3696 036764 010246
3697 036766 010346
3698 036770 013103
3699 036772 005002
3700 036774 073227 000006
3701 037000 010267 142202
3702 037004 010367 142200
3703 037010 012746 001206
3704 037014 004767 002610
3705 037020 062716 000003
3706 037024 012667 000002
3707 037030 104401
3708 037032 000000
3709 037034 012603
3710 037036 012602
3711 037040 005711
3712 037042 001404
3713 037044 104401 037066
3714 037050 105720
3715 037052 000707
3716 037054 012601
3717 037056 012600
3718 037060 104401 001221
3719 037064 000207
3720 037066 040 040 000
3721 037071 000
3722
    
```

```

10$: .WORD 0
:*THIS CODE IS FOR OCTAL (22-BIT) FORMAT FOR A PAR LEFT SHIFTED 6 (DF=4)
15$: MOV R2,-(KSP) ;SAVE R2 ON STACK
MOV R3,-(KSP) ;SAVE R3 ON STACK
MOV @R1+,R3 ;LOAD DATA WORD INTO R3
CLR R2 ;R2 HOLDS UPPER SIX BITS OF NUMBER
ASHC #6,R2 ;SHIFT VALUE LEFT 6 TIMES
MOV R2,$TMP4 ;HOLDS LOWER 16 BITS OF ADDRESS
MOV R3,$TMP5 ;HOLDS UPPER 6 BITS OF ADDRESS
MOV #$TMP4,-(KSP) ;PUT ADDRESS OF LOWER BITS ONTO STACK
JSR PC,$DB20 ;CONVERT TWO LOCS. TO AN ASCII STRING
ADD #3,(KSP) ;ONLY NEED 8 CHARACTERS NOT 11
MOV (KSP)+,16$ ;PUT ADDRESS OF ASCII CHARS. AT 16$
TYPE ;TYPE OCTAL VALUE OF 22-BIT BINARY NO.
16$: .WORD 0
MOV (KSP)+,R3 ;RESTORE R3
MOV (KSP)+,R2 ;RESTORE R2
11$: TST (R1) ;IS THERE ANOTHER NUMBER?
BEQ 12$ ;BR IF NO
TYPE ,14$ ;TYPE TWO(2) SPACES
TSTB (R0)+ ;POINT TO NEW 'DATA FORMAT'
BR 6$ ;LOOP
12$: MOV (KSP)+,R1 ;RESTORE R1
13$: MOV (KSP)+,R0 ;RESTORE R0
TYPE ,$CRLF ;'CARRIAGE RETURN' & 'LINE FEED'
RTS PC ;RETURN
14$: .ASCIZ / / ;TWO(2) SPACES
.BYTE 0
.SBTTL TTY INPUT ROUTINE
:*****
.ENABL LSB
:*****
:*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
:*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
:*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
:*WHEN OPERATING IN TTY FLAG MODE.
    
```

```

037072 022767 000176 142040
037100 001114
037102 105777 142036
037106 100111
037110 117746 142032
037114 042716 177600
037120 022726 000007
037124 001102
037126 126727 142002 000001
037134 001476
037136 104401 040027
037142 104401 040034
037146 016746 141024
037152 104402
037154 104401 040045
037160 005046
037162 005046
037164 105777 141754
037170 100375
037172 117746 141750
037176 042716 177600
    
```

```

$CKSWR: CMP #SWREG,SWR ;:IS THE SOFT-SWR SELECTED?
BNE 15$ ;:BRANCH IF NO
TSTB @$TKS ;:CHAR THERE?
BPL 15$ ;:IF NO, DON'T WAIT AROUND
MOVB @$TKB,-(SP) ;:SAVE THE CHAR
BIC #^C177,(SP) ;:STRIP-OFF THE ASCII
CMP #7,(SP)+ ;:IS IT A CONTROL G?
BNE 15$ ;:NO, RETURN TO USER
CMPB $AUTOB,#1 ;:ARE WE RUNNING IN AUTO-MODE?
BEQ 15$ ;:BRANCH IF YES
TYPE ,$CNTLG ;:ECHO THE CONTROL-G (^G)
$GTSWR: TYPE ,SMSWR ;:TYPE CURRENT CONTENTS
MOV SWREG,-(SP) ;:SAVE SWREG FOR TYPEOUT
TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
TYPE ,SMNEW ;:PROMPT FOR NEW SWR
19$: CLR -(SP) ;:CLEAR COUNTER
CLR -(SP) ;:THE NEW SWR
7$: TSTB @$TKS ;:CHAR THERE?
BPL 7$ ;:IF NOT TRY AGAIN
MOVB @$TKB,-(SP) ;:PICK UP CHAR
BIC #^C177,(SP) ;:MAKE IT 7-BIT ASCII
    
```

```

037202 021627 000003          CMP      (SP),#3          ;; IS IT A CONTROL-C?
037206 001015                BNE      9$              ;; BRANCH IF NOT
037210 104401 001312          TYPE     ,SCNTLC         ;; YES, ECHO CONTROL-C (^C)
037214 062706 000006          ADD      #6,SP           ;; CLEAN UP STACK
037220 126727 141711 000001  CMPB     $INTAG,#1       ;; REENABLE TTY KEYBOARD INTERRUPTS?
037226 001003                BNE      8$              ;; BRANCH IF NO
037230 012777 000100 141706  MOV      #100,@$TKS      ;; ALLOW TTY KEYBOARD INTERRUPTS
037236 000167 000614          JMP      CNTRLC          ;; CONTROL-C RESTART
037242 021627 000025          8$:     CMP      (SP),#25       ;; IS IT A CONTROL-U?
037246 001005                BNE      10$             ;; BRANCH IF NOT
037250 104401 040022          TYPE     ,SCNTLU         ;; YES, ECHO CONTROL-U (^U)
037254 062706 000006          20$:    ADD      #6,SP           ;; IGNORE PREVIOUS INPUT
037260 000737                BR       19$             ;; LET'S TRY IT AGAIN
037262 021627 000015          10$:    CMP      (SP),#15       ;; IS IT A <CR>?
037266 001022                BNE      16$             ;; BRANCH IF NO
037270 005766 000004          TST     4(SP)           ;; YES, IS IT THE FIRST CHAR?
037274 001403                BEQ     11$             ;; BRANCH IF YES
037276 016677 000002 141634  MOV      2(SP),@$SWR     ;; SAVE NEW SWR
037304 062706 000006          11$:    ADD      #6,SP           ;; CLEAR UP STACK
037310 104401 001221          14$:    TYPE     ,SCRLF         ;; ECHO <CR> AND <LF>
037314 126727 141615 000001  CMPB     $INTAG,#1       ;; RE-ENABLE TTY KBD INTERRUPTS?
037322 001003                BNE      15$             ;; BRANCH IF NOT
037324 012777 000100 141612  MOV      #100,@$TKS      ;; RE-ENABLE TTY KBD INTERRUPTS
037332 000002          15$:    RTI                    ;; RETURN
037334 004767 001060          16$:    JSR      PC,$TYPEC      ;; ECHO CHAR
037340 021627 000060          CMP      (SP),#60        ;; CHAR < 0?
037344 002420                BLT     18$             ;; BRANCH IF YES
037346 021627 000067          CMP      (SP),#67        ;; CHAR > 7?
037352 003015                BGT     18$             ;; BRANCH IF YES
037354 042726 000060          BIC     #60,(SP)+        ;; STRIP-OFF ASCII
037360 005766 000002          TST     2(SP)           ;; IS THIS THE FIRST CHAR
037364 001403                BEQ     17$             ;; BRANCH IF YES
037366 006316                ASL     (SP)             ;; NO, SHIFT PRESENT
037370 006316                ASL     (SP)             ;; CHAR OVER TO MAKE
037372 006316                ASL     (SP)             ;; ROOM FOR NEW ONE.
037374 005266 000002          17$:    INC     2(SP)           ;; KEEP COUNT OF CHAR
037400 056616 177776          BIS     -2(SP),(SP)      ;; SET IN NEW CHAR
037404 000667                BR      7$              ;; GET THE NEXT ONE
037406 104401 001220          18$:    TYPE     ,QUES         ;; TYPE ?<CR><LF>
037412 000720                BR      20$             ;; SIMULATE CONTROL-U
.DSABL  LSB
;*****
;THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
;CALL:
;* RDCHR          ;; INPUT A SINGLE CHARACTER FROM THE TTY
;* RETURN HERE   ;; CHARACTER IS ON THE STACK
;*              ;; WITH PARITY BIT STRIPPED OFF
;
037414 011646          $RDCHR: MOV      (SP),-(SP)      ;; PUSH DOWN THE PC
037416 016666 000004 000002  MOV      4(SP),2(SP)     ;; SAVE THE PS
037424 105777 141514          1$:    TSTB     @$TKS         ;; WAIT FOR
037430 100375                BPL     1$              ;; A CHARACTER
037432 117766 141510 000004  MOVB     @$TKB,4(SP)     ;; READ THE TTY
037440 042766 177600 000004  BIC     #^C<177>,4(SP)  ;; GET RID OF JUNK IF ANY
037446 026627 000004 000023  CMP      4(SP),#23       ;; IS IT A CONTROL-S?
037454 001013                BNE     3$              ;; BRANCH IF NO
037456 105777 141462          2$:    TSTB     @$TKS         ;; WAIT FOR A CHARACTER

```

```

037462 100375          BPL      2$          ;; LOOP UNTIL ITS THERE
037464 117746 141456  MOVB    @STKB,-(SP)  ;; GET CHARACTER
037470 042716 177600  BIC    #^C177,(SP)  ;; MAKE IT 7-BIT ASCII
037474 022627 000021  CMP    (SP)+,#21    ;; IS IT A CONTROL-0?
037500 001366          BNE    2$          ;; IF NOT DISCARD IT
037502 000750          BR     1$          ;; YES, RESUME
037504 026627 000004 000140 3$:  CMP    4(SP),#140  ;; IS IT UPPER CASE?
037512 002407          BLT    4$          ;; BRANCH IF YES
037514 026627 000004 000175  CMP    4(SP),#175  ;; IS IT A SPECIAL CHAR?
037522 003003          BGT    4$          ;; BRANCH IF YES
037524 042766 000040 000004  BIC    #40,4(SP)   ;; MAKE IT UPPER CASE
037532 000002          4$:  RTI          ;; GO BACK TO USER
*****
*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
*CALL:
*      RDLIN          ;; INPUT A STRING FROM THE TTY
*      RETURN HERE   ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
*                      ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
$RDLIN: MOV    R3,-(SP)  ;; SAVE R3
        CLR    -(SP)   ;; CLEAR THE RUBOUT KEY
1$:  MOV    #$TTYIN,R3  ;; GET ADDRESS
2$:  CMP    #$TTYIN+8.,R3  ;; BUFFER FULL?
        BLOS  4$          ;; BR IF YES
        RDCHR          ;; GO READ ONE CHARACTER FROM THE TTY
        MOVB  (SP)+,(R3)  ;; GET CHARACTER
        CMPB  #3,(R3)    ;; IS IT A CONTROL-C?
        BNE  10$         ;; BRANCH IF NO
        TYPE  ,SCNTLC    ;; TYPE A CONTROL-C (^C)
        TST  (SP)+      ;; CLEAN RUBOUT KEY OFF OF THE STACK
        MOV  (SP)+,R3   ;; RESTORE R3
        JMP  CNTRLC     ;; GOTO CONTROL-C RESTART
10$:  CMPB  #177,(R3)   ;; IS IT A RUBOUT
        BNE  5$          ;; BR IF NO
        TST  (SP)       ;; IS THIS THE FIRST RUBOUT?
        BNE  6$          ;; BR IF NO
037612 112767 000134 000170  MOVB  #' \ ,9$      ;; TYPE A BACK SLASH
037620 104401 040010          TYPE  ,9$
037624 012716 177777          MOV  #-1,(SP)      ;; SET THE RUBOUT KEY
037630 005303          6$:  DEC    R3          ;; BACKUP BY ONE
037632 020327 040012  CMP    R3,$TTYIN  ;; STACK EMPTY?
037636 103434          BLO    4$          ;; BR IF YES
037640 111367 000144  MOVB  (R3),9$      ;; SETUP TO TYPEOUT THE DELETED CHAR.
037644 104401 040010          TYPE  ,9$
037650 000735          BR     2$          ;; GO READ ANOTHER CHAR.
037652 005716          5$:  TST  (SP)       ;; RUBOUT KEY SET?
037654 001406          BEQ  7$          ;; BR IF NO
037656 112767 000134 000124  MOVB  #' \ ,9$      ;; TYPE A BACK SLASH
037664 104401 040010          TYPE  ,9$
037670 005016          CLR  (SP)         ;; CLEAR THE RUBOUT KEY
037672 122713 000025  7$:  CMPB  #25,(R3)   ;; IS CHARACTER A CTRL U?
037676 001003          BNE  8$          ;; BR IF NO
037700 104401 040022  TYPE  ,SCNTLU     ;; TYPE A CONTROL 'U'
037704 000715          BR     1$          ;; GO START OVER
037706 122713 000022  8$:  CMPB  #22,(R3)   ;; IS CHARACTER A '^R'?
037712 001011          BNE  3$          ;; BRANCH IF NO
037714 105013          CLRB (R3)        ;; CLEAR THE CHARACTER
037716 104401 001221  TYPE  ,SCRLF     ;; TYPE A 'CR' & 'LF'

```



```

037722 104401 040012          TYPE      $TTYIN          ;;TYPE THE INPUT STRING
037726 000706          BR          2$          ;;GO PICKUP ANOTHER CHACTER
037730 104401 001220      4$:      TYPE      $QUES          ;;TYPE A '?'
037734 000701          BR          1$          ;;CLEAR THE BUFFER AND LOOP
037736 111367 000046      3$:      MOV      (R3),9$          ;;ECHO THE CHARACTER
037742 104401 040010          TYPE      9$
037746 122723 000015      (MP      #15,(R3)+          ;;CHECK FOR RETURN
037752 001274          BNE         2$          ;;LOOP IF NOT RETURN
037754 105063 177777      (LR      -1(R3)          ;;CLEAR RETURN (THE 15)
037760 104401 001222          TYPE      $LF          ;;TYPE A LINE FEED
037764 005726          TST        (SP)+          ;;CLEAN RUBOUT KEY FROM THE STACK
037766 012603          MOV        (SP)+,R3          ;;RESTORE R3
037770 011646          MOV        (SP),-(SP)          ;;ADJUST THE STACK AND PUT ADDRESS OF THE
037772 016666 000004 000002  MOV        4(SP),2(SP)          ;; FIRST ASCII CHARACTER ON IT
040000 012766 040012 000004  MOV        #TTYIN,4(SP)
040006 000002          RTI          ;;RETURN
040010 000          9$:      .BYTE    0          ;;STORAGE FOR ASCII CHAR. TO TYPE
040011 000          .BYTE    0          ;;TERMINATOR
040012          $TTYIN: .BLKB   8          ;;RESERVE 8 BYTES FOR TTY INPUT
040022 136 125 015 $CNTLU: .ASCIZ /^U/<15><12>          ;;CONTROL 'U'
040025 012 000          .ASCIZ /^G/<15><12>          ;;CONTROL 'G'
040027 136 107 015 $CNTLG: .ASCIZ /^G/<15><12>
040032 012 000          .ASCIZ <15><12>/SWR - /
040034 015 012 123 $MSWR: .ASCIZ
040037 127 122 040
040042 075 040 000
040045 040 040 116 $MNEW: .ASCIZ / NEW /
040050 105 127 040
040053 075 040 000

3723
3724          .SBTTL CONTROL-C SERVICING ROUTINE
3725
3726 040056 016767 141150 141124 CNTRLC: MOV      $PASS,$TMP5          ;;GET THE VALUE OF '$PASS'
3727 040064 005267 141120          INC      $TMP5          ;;FORM CURRENT PASS #
3728 040070 104401 040135          TYPE      $MSG          ;;TYPE THE TEST STOPS HERE
3729 040074 116767 141002 000026  MOV      $TSTNM,1$          ;;SAVE TEST NUMBER
3730 040102 016746 000022          MOV      1$,-(SP)          ;;SAVE 1$ FO TYPEOUT
3731 040106 104402          TYP      $
3732 040110 104401 040132          TYPE      2$
3733 040114 016746 141070          MOV      $TMP5,-(SP)          ;;SAVE $TMP5 FOR TYPEOUT
3734 040120 104405          TYP      ASCII DECIMAL WITH SIGN
3735 040122 104407          GTSWR          ;;ASK FOR NEW SWR VALUE
3736 040124 000167 175526          JMP      $EOP+2          ;;JUMP TO END OF PASS + 2
3737 040130 000000          1$:      .WORD    0          ;;TEST # BUFFER
3738 040132 040 040 000 2$:      .ASCIZ / /          ;;2 SPACES & STOP MESSAGE
3739 040135 112 125 115 CMMSG: .ASCII /JUMPING TO END OF PASS/<15><12>
040140 120 111 116
040143 107 040 124
040146 117 040 105
040151 116 104 040
040154 117 106 040
040157 120 101 123
040162 123 015 012
3740 040165 124 105 123          .ASCIZ /TESTNO PASSNO/<15><12>
040170 124 116 117
040173 011 120 101
040176 123 123 116
  
```

040201	117	015	012
040204	000		

3742 040205 000
3743

```

        .BYTE 0
        .SBTTL TYPE ROUTINE
        *****
        *ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
        *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
        *NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
        *NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
        *NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
        *
        *CALL:
        *1) USING A TRAP INSTRUCTION
        *      TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
        *OR
        *      TYPE
        *      MESADR
        *
        040206 105767 140745 $TYPE: TSTB $TFPLG      ;; IS THERE A TERMINAL?
        040212 100002      BPL 1$      ;; BR IF YES
        040214 000000      HALT      ;; HALT HERE IF NO TERMINAL
        040216 000430      BR 3$      ;; LEAVE
        040220 010046      1$: MOV R0,-(SP)      ;; SAVE R0
        040222 017600 000002 MOV @2(SP),R0      ;; GET ADDRESS OF ASCIZ STRING
        040226 122767 000001 141010 CMPB #APTENV,$ENV      ;; RUNNING IN APT MODE
        040234 001011      BNE 62$      ;; NO,GO CHECK FOR APT CONSOLE
        040236 132767 000100 141001 BITB #APTSPOOL,$ENVM      ;; SPOOL MESSAGE TO APT
        040244 001405      BEQ 62$      ;; NO,GO CHECK FOR CONSOLE
        040246 010067 000004      MOV R0,61$      ;; SETUP MESSAGE ADDRESS FOR APT
        040252 004767 000270      JSR PC,$ATY3      ;; SPOOL MESSAGE TO APT
        040256 000000      61$: .WORD 0      ;; MESSAGE ADDRESS
        040260 132767 000040 140757 62$: BITB #APTCSUP,$ENVM      ;; APT CONSOLE SUPPRESSED
        040266 001003      BNE 60$      ;; YES,SKIP TYPE OUT
        040270 112046      2$: MOVB (R0)+,-(SP)      ;; PUSH CHARACTER TO BE TYPED ONTO STACK
        040272 001005      BNE 4$      ;; BR IF IT ISN'T THE TERMINATOR
        040274 005726      TST (SP)+      ;; IF TERMINATOR POP IT OFF THE STACK
        040276 012600      60$: MOV (SP)+,R0      ;; RESTORE R0
        040300 062716 000002      3$: ADD #2,(SP)      ;; ADJUST RETURN PC
        040304 000002      RTI      ;; RETURN
        040306 122716 000011      4$: CMPB #HT,(SP)      ;; BRANCH IF <HT>
        040312 001430      BEQ 8$
        040314 122716 000200      CMPB #CRLF,(SP)      ;; BRANCH IF NOT <CRLF>
        040320 001006      BNE 5$
        040322 005726      TST (SP)+      ;; POP <CR><LF> EQUIV
        040324 104401      TYPE      ;; TYPE A CR AND LF
        040326 001221      $CRLF
        040330 105067 000200      CLRB $CHARCNT      ;; CLEAR CHARACTER COUNT
        040334 000755      BR 2$      ;; GET NEXT CHARACTER
        040336 004767 000056      5$: JSR PC,$TYPEC      ;; GO TYPE THIS CHARACTER
        040342 126726 140610      6$: CMPB $FILLC,(SP)+      ;; IS IT TIME FOR FILLER CHARS.?
        040346 001350      BNE 2$      ;; IF NO GO GET NEXT CHAR.
        040350 016746 140600      MOV $NULL,-(SP)      ;; GET # OF FILLER CHARS. NEEDED
        ;; AND THE NULL CHAR.
        040354 105366 000001      7$: DECB 1(SP)      ;; DOES A NULL NEED TO BE TYPED?
        040360 002770      BLT 6$      ;; BR IF NO--GO POP THE NULL OFF OF STACK
        040362 004767 000032      JSR PC,$TYPEC      ;; GO TYPE A NULL
        040366 105367 000142      DECB $CHARCNT      ;; DO NOT COUNT AS A COUNT
        040372 000770      BR 7$      ;; LOOP
        ;HORIZONTAL TAB PROCESSOR
    
```

```

040374 112716 000040      8$:  MOVB  #' (SP)      ;; REPLACE TAB WITH SPACE
040400 004767 000014      9$:  JSR  PC,$TYPEC    ;; TYPE A SPACE
040404 132767 000007 000122  BITB  #7,$CHARCNT  ;; BRANCH IF NOT AT
040412 001372          BNE  9$           ;; TAB STOP
040414 005726          TST  (SP)+        ;; POP SPACE OFF STACK
040416 000724          BR   2$           ;; GET NEXT CHARACTER
040420 105777 140524      $TYPEC: TSTB @STPS      ;; WAIT UNTIL PRINTER IS READY
040424 100375          BPL  $TYPEC
040426 116677 000002 140516  MOVB  2(SP),@STPB  ;; LOAD CHAR TO BE TYPED INTO DATA REG.
040434 105777 140504          TSTB @STKS      ;; SEE IF KEYBOARD IS TALKING.
040440 100021          BPL  2$           ;; BRANCH IF IT ISN'T.
040442 017746 140500          MOV  @STKB,-(SP)   ;; PUSH CHARACTER ONTO STACK.
040446 042716 177600          BIC  #177600,(SP) ;; BIT CLEAR TOP BYTE AND PARITY BIT.
040452 022726 000023          CMP  #23,(SP)+   ;; SEE IF THIS IS A ^S.
040456 001012          BNE  2$           ;; BRANCH TO CONTINUE IF IT ISN'T.
040460 105777 140460      3$:  TSTB @STKS      ;; WAIT FOR ANOTHER INPUT.
040464 100375          BPL  3$           ;; BRANCH BACK IF NOT READY.
040466 017746 140454          MOV  @STKB,-(SP)   ;; PUSH NEXT CHARACTER ON STACK.
040472 042716 177600          BIC  #177600,(SP) ;; BIT CLEAR TOP BYTE AND PARITY BIT.
040476 022726 000021          CMP  #21,(SP)+   ;; SEE IF THIS IS A ^Q.
040502 001366          BNE  3$           ;; BRANCH BACK FOR MORE WAIT IF NOT.
040504 122766 000015 000002  2$:  CMPB  #CR,2(SP)   ;; IS CHARACTER A CARRIAGE RETURN?
040512 001003          BNE  1$           ;; BRANCH IF NO
040514 105067 000014          CLRB $CHARCNT    ;; YES--CLEAR CHARACTER COUNT
040520 000406          BR   $TYPEX     ;; EXIT
040522 122766 000012 000002  1$:  CMPB  #LF,2(SP)   ;; IS CHARACTER A LINE FEED?
040530 001402          BEQ  $TYPEX     ;; BRANCH IF YES
040532 105227          INCB (PC)+     ;; COUNT THE CHARACTER
040534 000000      $CHARCNT: .WORD 0      ;; CHARACTER COUNT STORAGE
040536 000207      $TYPEX: PC
3744
.SBTTL APT COMMUNICATIONS ROUTINE
*****
040540 112767 000001 000236  $ATY1: MOVB  #1,$FFLG  ;; TO REPORT FATAL ERROR
040546 112767 000001 000226  $ATY3: MOVB  #1,$MFLG  ;; TO TYPE A MESSAGE
040554 000403          BR   $ATYC
040556 112767 000001 000220  $ATY4: MOVB  #1,$FFLG  ;; TO ONLY REPORT FATAL ERROR
040564          $ATYC:
040564 010046          MOV  R0,-(SP)     ;; PUSH R0 ON STACK
040566 010146          MOV  R1,-(SP)     ;; PUSH R1 ON STACK
040570 105767 000206          TSTB $MFLG      ;; SHOULD TYPE A MESSAGE?
040574 001450          BEQ  5$           ;; IF NOT: BR
040576 122767 000001 140440  CMPB  #APTENV,$ENV  ;; OPERATING UNDER APT?
040604 001031          BNE  3$           ;; IF NOT: BR
040606 132767 000100 140431  BITB  #APTSPOOL,$ENVM ;; SHOULD SPOOL MESSAGES?
040614 001425          BEQ  3$           ;; IF NOT: BR
040616 017600 000004          MOV  @4(SP),R0    ;; GET MESSAGE ADDR.
040622 062766 000002 000004  ADD  #2,4(SP)      ;; BUMP RETURN ADDR.
040630 005767 140370      1$:  TST  $MSGTYPE     ;; SEE IF DONE W/ LAST XMISSION?
040634 001375          BNE  1$           ;; IF NOT: WAIT
040636 010067 140376          MOV  R0,$MSGAD    ;; PUT ADDR IN MAILBOX
040642 105720      2$:  TSTB (R0)+        ;; FIND END OF MESSAGE
040644 001376          BNE  2$           ;;
040646 166700 140366          SUB  $MSGAD,R0    ;; SUB START OF MESSAGE
040652 006200          ASR  R0          ;; GET MESSAGE LNTH IN WORDS
040654 010067 140362          MOV  R0,$MSGGLT   ;; PUT LENGTH IN MAILBOX
040660 012767 000004 140336  MOV  #4,$MSGTYPE  ;; TELL APT TO TAKE MSG.
040666 000413          BR   5$
    
```

```

040670 017667 000004 000016 3$: MOV @4(SP),4$ ::PUT MSG ADDR IN JSR LINKAGE
040676 062766 000002 000004 ADD #2,4(SP) ::BUMP RETURN ADDRESS
040704 016746 137066 MOV 177776,-(SP) ::PUSH 177776 ON STACK
040710 004767 177272 JSR PC,$TYPE ::CALL TYPE MACRO
040714 000000 4$: .WORD 0
040716 5$:
040716 105767 000062 10$: TSTB $FFLG ::SHOULD REPORT FATAL ERROR?
040722 001416 BEQ 12$ ::IF NOT: BR
040724 005767 140314 TST $ENV ::RUNNING UNDER APT?
040730 001413 BEQ 12$ ::IF NOT: BR
040732 005767 140266 11$: TST $MSGTYPE ::FINISHED LAST MESSAGE?
040736 001375 BNE 11$ ::IF NOT: WAIT
040740 017667 000004 140260 MOV @4(SP),$FATAL ::GET ERROR #
040746 062766 000002 000004 ADD #2,4(SP) ::BUMP RETURN ADDR.
040754 005267 140244 INC $MSGTYPE ::TELL APT TO TAKE ERROR
040760 105067 000020 12$: CLRB $FFLG ::CLEAR FATAL FLAG
040764 105067 000013 CLRB $LFLG ::CLEAR LOG FLAG
040770 105067 000006 CLRB $MFLG ::CLEAR MESSAGE FLAG
040774 012601 MOV (SP)+,R1 ::POP STACK INTO R1
040776 012600 MOV (SP)+,R0 ::POP STACK INTO R0
041000 000207 RTS PC ::RETURN
041002 000 $MFLG: .BYTE 0 ::MESSG. FLAG
041003 000 $LFLG: .BYTE 0 ::LOG FLAG
041004 000 $FFLG: .BYTE 0 ::FATAL FLAG

```

```

000200 APTSIZE=200
000601 APTENV=001
000100 APTSPool=100
000040 APTCSUP=040

```

3745

```

.SBTTL BINARY TO ASCII AND TYPE ROUTINE
*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
*BINARY-ASCII NUMBER AND TYPE IT.
*CALL:

```

```

* MOV NUMBER,-(SP) ::NUMBER TO BE TYPED
* TYPBN ::TYPE IT
041006 010146 $TYPBN: MOV R1,-(SP) ::SAVE R1 ON THE STACK
041010 016601 000006 MOV 6(SP),R1 ::GET THE INPUT NUMBER
041014 000261 SEC ::SET 'C' SO CAN KEEP TRACK OF THE NUMBER OF BITS
041016 112767 000060 000034 1$: MOV# #'0,$BIN ::SET CHARACTER TO AN ASCII '0'.
041024 006101 ROL R1 ::GET THIS BIT
041026 001406 BEQ 2$ ::DONE?
041030 105567 000024 ADCB $BIN ::NO--SET THE CHARACTER EQUAL TO THIS BIT
041034 104401 041060 TYPE , $BIN ::GO TYPE THIS BIT
041040 000241 CLC ::CLEAR 'C' SO CAN KEEP TRACK OF BITS
041042 000765 BR 1$ ::GO DO THE NEXT BIT
041044 012601 2$: MOV (SP)+,R1 ::POP THE STACK INTO R1
041046 016666 000002 000004 MOV 2(SP),4(SP) ::ADJUST THE STACK
041054 012616 MOV (SP)+,(SP)
041056 000002 RTI ::RETURN TO USER
041060 000 000 $BIN: .BYTE 0,0 ::STORAGE FOR ASCII CHAR. AND TERMINATOR

```

3746

```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE
*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:

```

```

    *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
    *      TYPOS    ;;CALL FOR TYPEOUT
    *      .BYTE   N                ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
    *      .BYTE   M                ;;M=1 OR 0
    *                                     ;;1=TYPE LEADING ZEROS
    *                                     ;;0=SUPPRESS LEADING ZEROS
    *
    *$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
    *$TYPOS OR $TYPOC
    *CALL:
    *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
    *      TYPON    ;;CALL FOR TYPEOUT
    *
    *$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
    *CALL:
    *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
    *      TYPOC    ;;CALL FOR TYPEOUT
    *
    041062 017646 000000 000211 $TYPOS: MOV      @ (SP),-(SP)      ;;PICKUP THE MODE
    041066 116667 000001          MOV      1(SP), $OFILL      ;;LOAD ZERO FILL SWITCH
    041074 112667 000207          MOV      (SP)+, $OMODE+1      ;;NUMBER OF DIGITS TO TYPE
    041100 062716 000002          ADD      #2,(SP)          ;;ADJUST RETURN ADDRESS
    041104 000406          BR      $TYPON
    041106 112767 000001 000171 $TYPOC: MOV      #1, $OFILL      ;;SET THE ZERO FILL SWITCH
    041114 112767 000006 000165          MOV      #6, $OMODE+1      ;;SET FOR SIX(6) DIGITS
    041122 112767 000005 000154 $TYPON: MOV      #5, $OCNT      ;;SET THE ITERATION COUNT
    041130 010346          MOV      R3,-(SP)          ;;SAVE R3
    041132 010446          MOV      R4,-(SP)          ;;SAVE R4
    041134 010546          MOV      R5,-(SP)          ;;SAVE R5
    041136 116704 000145          MOV      $OMODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE
    041142 005404          NEG      R4
    041144 062704 000006          ADD      #6,R4          ;;SUBTRACT IT FOR MAX. ALLOWED
    041150 110467 000132          MOV      R4, $OMODE      ;;SAVE IT FOR USE
    041154 116704 000125          MOV      $OFILL,R4      ;;GET THE ZERO FILL SWITCH
    041160 016605 000012          MOV      12(SP),R5      ;;PICKUP THE INPUT NUMBER
    041164 005003          CLR      R3          ;;CLEAR THE OUTPUT WORD
    041166 006105          1$: ROL      R5          ;;ROTATE MSB INTO 'C'
    041170 000404          BR      3$          ;;GO DO MSB
    041172 006105          2$: ROL      R5          ;;FORM THIS DIGIT
    041174 006105          ROL      R5
    041176 006105          ROL      R5
    041200 010503          MOV      R5,R3
    041202 006103          3$: ROL      R3          ;;GET LSB OF THIS DIGIT
    041204 105367 000076          DECB    $OMODE          ;;TYPE THIS DIGIT?
    041210 100016          BPL     7$          ;;BR IF NO
    041212 042703 177770          BIC     #177770,R3      ;;GET RID OF JUNK
    041216 001002          BNE     4$          ;;TEST FOR 0
    041220 005704          TST     R4          ;;SUPPRESS THIS 0?
    041222 001403          BEQ     5$          ;;BR IF YES
    041224 005204          4$: INC     R4          ;;DON'T SUPPRESS ANYMORE 0'S
    041226 052703 000060          BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
    041232 052703 000040          5$: BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
    041236 110367 000040          MOV      R3,8$          ;;SAVE FOR TYPING
    041242 104401 041302          TYPE    ,8$          ;;GO TYPE THIS DIGIT
    041246 105367 000032          7$: DECB    $OCNT          ;;COUNT BY 1
    041252 003347          BGT     2$          ;;BR IF MORE TO DO
    041254 002402          BLT     6$          ;;BR IF DONE
    041256 005204          INC     R4          ;;INSURE LAST DIGIT ISN'T A BLANK
    
```

```
041260 000744          BR      2$          ;;GO DO THE LAST DIGIT
041262 012605          6$:  MOV    (SP)+,R5      ;;RESTORE R5
041264 012604          MOV    (SP)+,R4      ;;RESTORE R4
041266 012603          MOV    (SP)+,R3      ;;RESTORE R3
041270 016666 000002 000004 MOV    2(SP),4(SP)    ;;SET THE STACK FOR RETURNING
041276 012616          MOV    (SP)+,(SP)
041300 000002          RTI          ;;RETURN
041302 000          8$:  .BYTE 0          ;;STORAGE FOR ASCII DIGIT
041303 000          .BYTE 0          ;;TERMINATOR FOR TYPE ROUTINE
041304 000          $OCNT: .BYTE 0        ;;OCTAL DIGIT COUNTER
041305 000          $OFILL: .BYTE 0       ;;ZERO FILL SWITCH
041306 000000          $OMODE: .WORD 0        ;;NUMBER OF DIGITS TO TYPE
```

3748

```

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
:*****
:THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
:SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
:NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
:BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
:REPLACED WITH SPACES.
:CALL:
:*      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
:*      TYPDS    ;;GO TO THE ROUTINE
$TYPDS:
MOV      R0,-(SP)      ;;PUSH R0 ON STACK
MOV      R1,-(SP)      ;;PUSH R1 ON STACK
MOV      R2,-(SP)      ;;PUSH R2 ON STACK
MOV      R3,-(SP)      ;;PUSH R3 ON STACK
MOV      R5,-(SP)      ;;PUSH R5 ON STACK
MOV      #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
MOV      20(SP),R5      ;;GET THE INPUT NUMBER
BPL      1$            ;;BR IF INPUT IS POS.
NEG      R5            ;;MAKE THE BINARY NUMBER POS.
MOVB     #'-,1(SP)      ;;MAKE THE ASCII NUMBER NEG.
1$:      CLR      R0            ;;ZERO THE CONSTANTS INDEX
MOV      #SDBLK,R3      ;;SETUP THE OUTPUT POINTER
MOVB     #' ,(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
2$:      CLR      R2            ;;CLEAR THE BCD NUMBER
MOV      $DTBL(R0),R1    ;;GET THE CONSTANT
3$:      SUB      R1,R5          ;;FORM THIS BCD DIGIT
BLT      4$            ;;BR IF DONE
INC      R2            ;;INCREASE THE BCD DIGIT BY 1
BR       3$
4$:      ADD      R1,R5          ;;ADD BACK THE CONSTANT
TST      R2            ;;CHECK IF BCD DIGIT=0
BNE      5$            ;;FALL THROUGH IF 0
TSTB     (SP)          ;;STILL DOING LEADING 0'S?
BMI      7$            ;;BR IF YES
5$:      ASLB     (SP)          ;;MSD?
BCC      6$            ;;BR IF NO
MOVB     1(SP),-1(R3)    ;;YES--SET THE SIGN
6$:      BIS      #'0,R2        ;;MAKE THE BCD DIGIT ASCII
7$:      BIS      #' ,R2        ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
MOVB     R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
TST      (R0)+          ;;JUST INCREMENTING
CMP      R0,#10        ;;CHECK THE TABLE INDEX
BLT      2$            ;;GO DO THE NEXT DIGIT
BGT      8$            ;;GO TO EXIT
MOV      R5,R2          ;;GET THE LSD
BR       6$            ;;GO CHANGE TO ASCII
8$:      TSTB     (SP)+        ;;WAS THE LSD THE FIRST NON-ZERO?
BPL      9$            ;;BR IF NO
MOVB     -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
9$:      CLRB     (R3)          ;;SET THE TERMINATOR
MOV      (SP)+,R5        ;;POP STACK INTO R5
MOV      (SP)+,R3        ;;POP STACK INTO R3
MOV      (SP)+,R2        ;;POP STACK INTO R2
MOV      (SP)+,R1        ;;POP STACK INTO R1
MOV      (SP)+,R0        ;;POP STACK INTO R0
MOV      #SDBLK,R0      ;;NOW TYPE THE NUMBER
TYPE
    
```

```

041310
041310 010046
041312 010146
041314 010246
041316 010346
041320 010546
041322 012746 020200
041326 016605 000020
041332 100004
041334 005405
041336 112766 000055 000001
041344 005000 1$:
041346 012703 041524
041352 112723 000040
041356 005002 2$:
041360 016001 041514
041364 160105 3$:
041366 002402
041370 005202
041372 000774
041374 060105 4$:
041376 005702
041400 001002
041402 105716
041404 100407
041406 106316 5$:
041410 103003
041412 116663 000001 177777
041420 052702 000060 6$:
041424 052702 000040 7$:
041430 110223
041432 005720
041434 020027 000010
041440 002746
041442 003002
041444 010502
041446 000764
041450 105726 8$:
041452 100003
041454 116663 177777 177776
041462 105013 9$:
041464 012605
041466 012603
041470 012602
041472 012601
041474 012600
041476 104401 041524
    
```



```

041502 016666 000002 000004      MOV      2(SP),4(SP)      ;;ADJUST THE STACK
041510 012616                    MOV      (SP)+,(SP)
041512 000002                    RTI                      ;;RETURN TO USER
041514 023420                    $DTBL: 10000.
041516 001750                    1000.
041520 000144                    100.
041522 000012                    10.
041524                    $DBLK: .BLKW 4
3749 .SBTTL SAVE AND RESTORE R0-R5 ROUTINES
:*****
:*SAVE R0-R5
:*CALL:
:* SAVREG
:*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
:*
:*TOP---(+16)
:* +2---(+18)
:* +4---R5
:* +6---R4
:* +8---R3
:*+10---R2
:*+12---R1
:*+14---R0
$SAVREG:
041534 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
041536 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
041540 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
041542 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
041544 010446      MOV      R4,-(SP)      ;;PUSH R4 ON STACK
041546 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
041550 016646 000022      MOV      22(SP),-(SP)  ;;SAVE PS OF MAIN FLOW
041554 016646 000022      MOV      22(SP),-(SP)  ;;SAVE PC OF MAIN FLOW
041560 016646 000022      MOV      22(SP),-(SP)  ;;SAVE PS OF CALL
041564 016646 000022      MOV      22(SP),-(SP)  ;;SAVE PC OF CALL
041570 000002      RTI
:*RESTORE R0-R5
:*CALL:
:* RESREG
$RESREG:
041572 012666 000022      MOV      (SP)+,22(SP)  ;;RESTORE PC OF CALL
041576 012666 000022      MOV      (SP)+,22(SP)  ;;RESTORE PS OF CALL
041602 012666 000022      MOV      (SP)+,22(SP)  ;;RESTORE PC OF MAIN FLOW
041606 012666 000022      MOV      (SP)+,22(SP)  ;;RESTORE PS OF MAIN FLOW
041612 012605      MOV      (SP)+,R5      ;;POP STACK INTO R5
041614 012604      MOV      (SP)+,R4      ;;POP STACK INTO R4
041616 012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
041620 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
041622 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
041624 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
041626 000002      RTI
3750 .SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
:*****
:*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
:*UNSIGNED OCTAL ASCII NUMBER.
:*CALL
:* MOV      #PNTR,-(SP)      ;;POINTER TO LOW WORD OF BINARY NUMBER
:* JSR      PC,@$DDB20      ;;CALL THE ROUTINE
    
```

```

041630 104413          ;* RETURN
041632 016601 000002 $DB20: SAVREG
041636 012705 041747 MOV 2(SP),R1
041642 012704 000014 MOV #SOCTVL+13.,R5
041646 012703 177770 MOV #12.,R4
041652 012100 MOV #^C7,R3
041654 012101 MOV (R1)+,R0
041656 005002 MOV (R1)+,R1
041660 110245 1$: CLR R2
041662 010002 MOVB R2,-(R5)
041664 005304 MOV R0,R2
041666 003007 DEC R4
041670 001405 BGT 3$
041672 005205 BEQ 2$
041674 010566 000002 INC R5
041700 104414 MOV R5,2(SP)
041702 000207 RESREG
041704 006203 RTS PC
041706 006001 2$: ASR R3
041710 006000 3$: ROR R1
041712 006001 ROR R0
041714 006000 ROR R1
041716 006001 ROR R0
041720 006000 ROR R1
041722 040302 BIC R3,R2
041724 062702 000060 ADD #'0,R2
041730 000753 BR 1$
041732 $OCTVL: .BLKB 14.

```

```

::THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK
::SAVE ALL REGISTERS
::PICKUP THE POINTER TO LOW WORD
::POINTER TO DATA TABLE
::DO ELEVEN CHARACTERS
::MASK
::LOWER WORD
::HIGH WORD
::TERMINATOR
::PUT CHARACTER IN DATA TABLE
::GET THIS DIGIT
::COUNT THIS CHARACTER
::BR IF NOT THE LAST DIGIT
::BR IF IT IS THE LAST DIGIT
::ALL DIGITS DONE-ADJUST POINTER FOR FIRST
::ASCII CHAR. & PUT IT ON THE STACK
::RESTORE ALL REGISTERS
::RETURN TO USER
::POSITION THE MASK FOR THE LAST DIGIT
::POSITION THE BINARY NUMBER FOR
::THE NEXT OCTAL DIGIT
::MASK OUT ALL JUNK
::MAKE THIS CHAR. ASCII
::GO PUT IT IN THE DATA TABLE
::RESERVE DATA TABLE

```

3752

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

```
041750 010046
041752 016600 000002
041756 005740
041760 111000
041762 006300
041764 016000 042004
041770 000200
```

```
$TRAP: MOV R0,-(SP) ;;SAVE R0
MOV 2(SP),R0 ;;GET TRAP ADDRESS
TST -(R0) ;;BACKUP BY 2
MOVB (R0),R0 ;;GET RIGHT BYTE OF TRAP
ASL R0 ;;POSITION FOR INDEXING
MOV $TRPAD(R0),R0 ;;INDEX TO TABLE
RTS R0 ;;GO TO ROUTINE
```

```
;;THIS IS USE TO HANDLE THE 'GETPRI' MACRO
```

```
041772 011646
041774 016666 000004 000002
042002 000002
```

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

```
.SBTTL TRAP TABLE
:THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
:BY THE 'TRAP' INSTRUCTION.
```

```
042004 041772
042006 040206
042010 041106
042012 041062
042014 041122
042016 041310
042020 041006
042022 037142
042024 037072
042026 037414
042030 037534
042032 041534
042034 041572
```

```
ROUTINE
-----
$TRPAD: .WORD $TRAP2
$TYPE ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
$TYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
$TYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
$TYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
$TYPDS ;;CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
$TYPBN ;;CALL=TYPBN TRAP+6(104406) TYPE BINARY (ASCII) NUMBER
$GTSWR ;;CALL=GTSWR TRAP+7(104407) GET SOFT-SWR SETTING
$CKSWR ;;CALL=CKSWR TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
$RDCHR ;;CALL=RDCHR TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
$RDLIN ;;CALL=RDLIN TRAP+12(104412) TTY TYPEIN STRING ROUTINE
$SAVREG ;;CALL=SAVREG TRAP+13(104413) SAVE R0-R5 ROUTINE
$RESREG ;;CALL=RESREG TRAP+14(104414) RESTORE R0-R5 ROUTINE
```

3753

```
.SBTTL POWER DOWN AND UP ROUTINES
:*****
```

```
042036 012737 042214 000024
042044 012737 000340 000026
042052 010046
042054 010146
042056 010246
042060 010346
042062 010446
042064 010546
042066 017746 137046
042072 010667 000122
042076 012737 042110 000024
042104 000000
042106 000776
```

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

```
*****
```

```
042110 012737 042214 000024
042116 016706 000076
042122 005067 000072
042126 005267 000066
```

```
POWER UP ROUTINE
$PWRUP: MOV #SILLUP,@PWRVEC ;;SET FOR FAST DOWN
MOV $SAVR6,SP ;;GET SP
CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
1$: INC $SAVR6 ;;WAIT FOR THE INC
```

```
042132 001375          BNE      1$          ;;OF WORD
042134 012677 137000   MOV      (SP)+,@SWR   ;;POP STACK INTO @SWR
042140 012605          MOV      (SP)+,R5    ;;POP STACK INTO R5
042142 012604          MOV      (SP)+,R4    ;;POP STACK INTO R4
042144 012603          MOV      (SP)+,R3    ;;POP STACK INTO R3
042146 012602          MOV      (SP)+,R2    ;;POP STACK INTO R2
042150 012601          MOV      (SP)+,R1    ;;POP STACK INTO R1
042152 012600          MOV      (SP)+,R0    ;;POP STACK INTO R0
042154 012737 042036 000024 MOV      #SPWRDN,@PWRVEC ;;SET UP THE POWER DOWN VECTOR
042162 012737 000340 000026 MOV      #340,@PWRVEC+2 ;;PRIO:7
042170 104401          TYPE                      ;;REPORT THE POWER FAILURE
042172 042222          $PWRMG: .WORD PWRMSG    ;;POWER FAIL MESSAGE POINTER
042174 012716          MOV      (PC)+,(SP) ;;RESTART AT START
042176 020000          $PWRAD: .WORD START    ;;RESTART ADDRESS
042200 042766 000020 000002 BIC      #20,2(SP)    ;;CLEAR 'T' BIT
042206 005067 137076   CLR      $TBIT       ;;CLEAR THE 'T' BIT FLAG
042212 000002          RTI
042214 000000          $ILLUP: HALT        ;;THE POWER UP SEQUENCE WAS STARTED
042216 000776          BR      .-2        ;; BEFORE THE POWER DOWN WAS COMPLETE
042220 000000          $$SAVR6: 0        ;;PUT THE SP HERE
3754 042222 012 015 040 PWRMSG: .ASCIIZ <12><15>? POWER FAILURE - RESTARTING ?<12><15>
042225 120 117 127
042230 105 122 040
042233 106 101 111
042236 114 125 122
042241 105 040 055
042244 040 122 105
042247 123 124 101
042252 122 124 111
042255 116 107 040
042260 012 015 000

3755          .EVEN
3756
3757
```

```

3759          .SBTTL  ERROR MESSAGES, DATA HEADERS-TABLES & FORMATS
3760          .NLIST  BEX
3761 042264      125      116      105  EM1:  .ASCIZ  /UNEXPECTED CPU TRAP TO LOC. 004/
3762 042324      125      116      105  EM2:  .ASCIZ  /UNEXPECTED MEM. MGMT. TRAP TO LOC. 250/
3763 042373      115      105      115  EM10: .ASCIZ  /MEMORY MGMT. ACCESS ABORT DID NOT OCCUR/
3764 042443      101      103      103  EM11: .ASCIZ  /ACCESS ERROR DID NOT ABORT INSTRUCTION/
3765 042512      123      122      060  EM12: .ASCIZ  /SR0 DID NOT REPORT ACCESS ERROR CORRECTLY/
3766 042564      123      122      062  EM13: .ASCIZ  /SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR./
3767 042635      120      101      107  EM14: .ASCIZ  /PAGE LGTH. ABORT OCCURRED WHEN IT SHOULDN'T HAVE/
3768 042716      120      101      107  EM15: .ASCIZ  /PAGE LGTH. ABORT DID NOT OCCUR WHEN IT SHOULD HAVE/
3769 043001      123      122      060  EM16: .ASCIZ  /SR0 DID NOT REPORT PAGE LGTH. ABORT CORRECTLY/
3770 043057      123      122      060  EM21: .ASCIZ  /SR0 OR SR2 CHANGED BY A SECOND ABORT/
3771 043124      123      122      060  EM22: .ASCIZ  /SR0 OR SR2 WERE NOT 'RESET' BY A RESET/
3772 043173      123      122      062  EM23: .ASCIZ  /SR2 NOT TRACKING CORRECTLY/
3773 043226      104      111      104  EM24: .ASCIZ  /DID NOT TRAP THRU KERNEL SPACE/
3774 043265      113      124      040  EM25: .ASCIZ  /KT ERROR SERVICED ON ODD ADDR. ERROR/
3775 043332      123      122      060  EM26: .ASCIZ  /SR0 OR SR2 CHANGED BY ODD ADDR. ERROR/
3776 043400      105      122      122  EM27: .ASCIZ  /ERROR DURING 'DOUBLE ERROR' (KT & ODD ADDR.)/
3777 043455      115      106      120  EM30: .ASCIZ  /MFPI INSTRUCTION PUSHED WRONG DATA/
3778 043520      115      124      120  EM31: .ASCIZ  /MTPI INSTRUCTION LOADED WRONG DATA/
3779 043563      123      124      101  EM32: .ASCIZ  /STACK NOT PUSHED BY MFPI-MTPI/
3780 043621      113      105      122  EM33: .ASCIZ  /KERNEL PAGE ACCESS INSTEAD OF USER: MFPI-MTPI/
3781 043677      115      056      115  EM34: .ASCIZ  /M.M. ABORT IN KERNAL D-SPACE HAD WRONG CONDITION/
3782 043760      111      114      114  EM35: .ASCIZ  /ILLEGAL MODE 10 NOT ABORTED/
3783 044014      123      122      060  EM36: .ASCIZ  /SR0 DID NOT REPORT ILLEGAL MODE 10 CORRECTLY/
3784 044071      120      123      127  EM37: .ASCIZ  /PSW CHANGED BY AN RTI IN USER MODE/
3785 044134      101      102      117  EM40: .ASCIZ  /ABORT I KERNAL D-SPACE PICKED UP VECTOR FROM I-SPACE/
3786 044221      104      040      123  EM41: .ASCIZ  /D SPACE ENABLE CIRCUITRY HAS FAILED/
3787 044265      111      116      103  EM42: .ASCIZ  /INCORRECT STORE BY MTP INSTRUCTION/
3788 044330      124      122      111  EM43: .ASCIZ  /TRIED TO REFERENCE NON-RESIDENT PAGE/
3789 044375      127      122      117  EM44: .ASCIZ  /WRONG DATA FETCHED BY MFP INSTRUCTION/
3790 044443      111      114      114  EM45: .ASCIZ  /ILLEGAL CSM DID NOT TRAP TO 10/
3791 044502      103      123      115  EM46: .ASCIZ  /CSM DID NOT ENTER SUPERVISOR MODE/
3792 044544      103      123      115  EM47: .ASCIZ  /CSM SET UP WRONG PREVIOUS MODE/
3793 044603      103      123      115  EM50: .ASCIZ  /CSM SET UP STACK WRONG/
3794 044632      103      123      115  EM51: .ASCIZ  /CSM PUSHED INCORRECT ARGUMENT/
3795 044670      103      123      115  EM52: .ASCIZ  /CSM PUSHED WRONG PC/
3796 044714      103      123      115  EM53: .ASCIZ  /CSM DID NOT CLEAR OLD PSW BITS <3:0>/
3797 044761      103      123      115  EM54: .ASCIZ  /CSM ACCESSED WRONG SUPERVISOR SPACE/
3798          .EVEN
  
```

3800	045026	117	114	104	DH1:	.ASCIZ	/OLD PC	OLD PSW	R6 WAS	CPUERR	TESTNO	ERRORPC/
3801	045106	117	114	104	DH2:	.ASCIZ	/OLD PC	OLD PSW	R6 WAS	SR0	SR2	TESTNO ERRORPC/
3802	045176	120	104	122	DH10:	.ASCIZ	/PDR 4	PSW	TESTNO	ERRORPC/		
3803	045236	123	122	060	DH12:	.ASCIZ	/SR0 WAS	EXPECTED	PDR 4	PSW	TESTNO	ERRORPC/
3804	045316	123	122	062	DH13:	.ASCIZ	/SR2 WAS	EXPECTED	PDR 4	PSW	TESTNO	ERRORPC/
3805	045376	126	056	102	DH14:	.ASCIZ	/V.B.A.	KIPDR4	SR0 WAS	SR2 WAS	TESTNO	ERRORPC/
3806	045456	126	056	102	DH15:	.ASCIZ	/V.B.A.	KIPDR4	TESTNO	ERRORPC/		
3807	045516	126	056	102	DH16:	.ASCIZ	/V.B.A.	KIPDR4	SR0 WAS	EXPECTED	TESTNO	ERRORPC/
3808	045576	126	056	102	DH17:	.ASCIZ	/V.B.A.	KIPDR4	SR2 WAS	EXPECTED	TESTNO	ERRORPC/
3809	045656	123	122	062	DH20:	.ASCIZ	/SR2 WAS	EXPECTED	TESTNO	ERRORPC/		
3810	045716	106	111	122	DH21:	.ASCII	/FIRST ABORT	SECOND ABORT/	<CRLF>			
3811	045753	123	122	060		.ASCIZ	/SR0 WAS	SR2 WAS	SR0 WAS	SR2 WAS	TESTNO	ERRORPC/
3812	046033	123	122	060	DH22:	.ASCIZ	/SR0 WAS	SR2 WAS	TESTNO	ERRORPC/		
3813	046073	120	123	127	DH24:	.ASCIZ	/PSW WAS	R6 WAS	TESTNO	ERRORPC/		
3814	046133	105	130	120	DH26:	.ASCII	/EXPECTED	RECEIVED/	<CRLF>			
3815	046165	123	122	060		.ASCIZ	/SR0	SR2	SR0 WAS	SR2 WAS	TESTNO	ERRORPC/
3816	046245	105	130	120	DH27:	.ASCII	/EXPECTED:	<CRLF>				
3817	046257	120	123	127		.ASCII	/PSW	PC	SR0	SR2/	<CRLF>	
3818	046313	061	067	060		.ASCII	/170017	(3\$+4)	020147	(3\$)/	<CRLF>	
3819	046350	122	105	103		.ASCII	/RECEIVED:	<CRLF>				
3820	046362	120	123	127		.ASCIZ	/PSW	PC	SR0	SR2	TESTNO	ERRORPC/
3821	046442	104	101	124	DH30:	.ASCII	/DATA	DATA/	<CRLF>			
3822	046457	105	130	120		.ASCIZ	/EXPECTED	RECEIVED	TESTNO	ERRORPC/		
3823	046517	124	105	123	DH32:	.ASCIZ	/TESTNO	ERRORPC/				
3824	046537	123	122	060	DH33:	.ASCIZ	/SR0 WAS	SR2 WAS	TESTNO	ERRORPC/		
3825	046577	050	115	115	DH34:	.ASCIZ	/(MMR0)	(MMR1)	(MMR2)	TESTNO	ERRORPC	EXPECTING 020031/
3826	046670	123	122	060	DH36:	.ASCIZ	/SR0 WAS	EXPECTED	TESTNO	ERRORPC/		
3827	046730	120	123	127	DH37:	.ASCIZ	/PSW WAS	EXPECTED	TESTNO	ERRORPC/		
3828	046770	050	120	123	DH40:	.ASCIZ	/(PSW)	TESTNO	ERRORPC	EXPECTING	XXX340/	
3829	047041	105	122	122	DH41:	.ASCII	/ERROR	AUTOI-D	VIRTUAL/	<CRLF>		
3830	047071	122	105	107		.ASCIZ	/REGISTR	REGISTR	ADDRESS	TESTNO	PC AT ABORT/	
3831	047145	107	104	104	DH42:	.ASCIZ	/GDDATA	STORED	TESTNO	ERRORPC/		
3832	047205	050	115	115	DH43:	.ASCIZ	/(MMR0)	(MMR1)	(MMR2)	TESTNO	ERRORPC/	
3833	047255	050	120	123	DH44:	.ASCIZ	/(PSW)	TESTNO	ERRORPC/			
3834	047305	117	114	104	DH45:	.ASCIZ	/OLDPSW	TESTNO	ERRORPC/			
3835						.EVEN						

ERROR MESSAGES, DATA HEADERS-TABLES & FORMATS

3837	047336	001260	001262	001256	DT1:	.WORD	TRAPPC,TRAPPS,WASR6,CPUERR,TESTNO,\$ERRPC,0
3838	047354	001260	001262	001256	DT2:	.WORD	TRAPPC,TRAPPS,WASR6,WASSR0,WASSR2,TESTNO,\$ERRPC,0
3839	047374	001166	001176	001254	DT10:	.WORD	\$REG2,\$TMP0,TESTNO,\$ERRPC,0
3840	047406	001264	001170	001166	DT12:	.WORD	WASSR0,\$REG3,\$REG2,\$TMP0,TESTNO,\$ERRPC,0
3841	047424	001270	001172	001166	DT13:	.WORD	WASSR2,\$REG4,\$REG2,\$TMP0,TESTNO,\$ERRPC,0
3842	047442	001162	001172	001264	DT14:	.WORD	\$REG0,\$REG4,WASSR0,WASSR2,TESTNO,\$ERRPC,0
3843	047460	001162	001172	001254	DT15:	.WORD	\$REG0,\$REG4,TESTNO,\$ERRPC,0
3844	047472	001162	001172	001264	DT16:	.WORD	\$REG0,\$REG4,WASSR0,\$REG2,TESTNO,\$ERRPC,0
3845	047510	001162	001172	001270	DT17:	.WORD	\$REG0,\$REG4,WASSR2,\$REG3,TESTNO,\$ERRPC,0
3846	047526	001270	001164	001254	DT20:	.WORD	WASSR2,\$REG1,TESTNO,\$ERRPC,0
3847	047540	001176	001202	001264	DT21:	.WORD	\$TMP0,\$TMP2,WASSR0,WASSR2,TESTNO,\$ERRPC,0
3848	047556	001264	001270	001254	DT22:	.WORD	WASSR0,WASSR2,TESTNO,\$ERRPC,0
3849	047570	001164	001166	001254	DT24:	.WORD	\$REG1,\$REG2,TESTNO,\$ERRPC,0
3850	047602	001162	001164	001264	DT26:	.WORD	\$REG0,\$REG1,WASSR0,WASSR2,TESTNO,\$ERRPC,0
3851	047620	001164	001170	001264	DT27:	.WORD	\$REG1,\$REG3,WASSR0,WASSR2,TESTNO,\$ERRPC,0
3852	047636	001162	001164	001254	DT30:	.WORD	\$REG0,\$REG1,TESTNO,\$ERRPC,0
3853	047650	001254	001116	000000	DT32:	.WORD	TESTNO,\$ERRPC,0
3854	047656	001164	001166	001170	DT34:	.WORD	\$REG1,\$REG2,\$REG3,TESTNO,\$ERRPC,0
3855	047672	001264	001164	001254	DT36:	.WORD	WASSR0,\$REG1,TESTNO,\$ERRPC,0
3856	047704	001162	001254	001116	DT40:	.WORD	\$REG0,TESTNO,\$ERRPC,0
3857	047714	001264	001266	001270	DT41:	.WORD	WASSR0,WASSR1,WASSR2,TESTNO,BADPC,0
3858	047730	001162	001164	001254	DT42:	.WORD	\$REG0,\$REG1,TESTNO,\$ERRPC,0
3859	047742	001162	001164	001166	DT43:	.WORD	\$REG0,\$REG1,\$REG2,TESTNO,\$ERRPC,0
3860	047756	001264	001266	001270	DT45:	.WORD	WASSR0,WASSR1,WASSR2,TESTNO,\$ERRPC,0
3861	047772	001256	001254	001116	DT46:	.WORD	WASR6,TESTNO,\$ERRPC,0
3862	050002	001174	001256	001254	DT47:	.WORD	\$REG5,WASR6,TESTNO,\$ERRPC,0
3863	050014	001176	001256	001254	DT50:	.WORD	\$TMP0,WASR6,TESTNO,\$ERRPC,0

```
3865 050026      000      000      000 DF1:  .BYTE  0,0,0,0,0
3866 050033      000      000      000 DF2:  .BYTE  0,0,0,0,0,0,C
3867 050042      000      000      000 DF3:  .BYTE  0,0,0,0
3868 050046      000      000      000 DF5:  .BYTE  0,0,0
3869 050051      000      000      000 DF12: .BYTE  0,0,0,0,0,0
3870 050057      000      000      000 DF32: .BYTE  0,0
3871
3872                .EVEN
3873                .LIST  BEX
3874
3875      000001      .END
```


ABASE = 000000
 ACDW1 = 000000
 ACDW2 = 000000
 ACPUDP = 000000
 ADDW0 = 000000
 ADDW1 = 000000
 ADDW10 = 000000
 ADDW11 = 000000
 ADDW12 = 000000
 ADDW13 = 000000
 ADDW14 = 000000
 ADDW15 = 000000
 ADDW2 = 000000
 ADDW3 = 000000
 ADDW4 = 000000
 ADDW5 = 000000
 ADDW6 = 000000
 ADDW7 = 000000
 ADDW8 = 000000
 ADDW9 = 000000
 ADEVCT = 000000
 ADEVN = 000000
 AENV = 000000
 AENVN = 000000
 AFATAL = 000000
 AMADR1 = 000000
 AMADR2 = 000000
 AMADR3 = 000000
 AMADR4 = 000000
 AMAMS1 = 000000
 AMAMS2 = 000000
 AMAMS3 = 000000
 AMAMS4 = 000000
 AMSGAD = 000000
 AMSGLG = 000000
 AMSGTY = 000000
 AMTYP1 = 000000
 AMTYP2 = 000000
 AMTYP3 = 000000
 AMTYP4 = 000000
 APASS = 000000
 APRINI = 002020
 APRIOR = 000000
 APTCSU = 000040
 APTENV = 000001
 APTSIZ = 000200
 APTSPO = 000100
 ASWREG = 000000
 ATESTN = 000000
 AUNIT = 000000
 AUSWR = 000000
 AVECT1 = 000000
 AVECT2 = 000000
 BADPC = 001304
 BIT0 = 000001
 BIT00 = 000001
 BIT01 = 000002

BIT02 = 000004
 BIT03 = 000010
 BIT04 = 000020
 BIT05 = 000040
 BIT06 = 000100
 BIT07 = 000200
 BIT08 = 000400
 BIT09 = 001000
 BIT1 = 000002
 BIT10 = 002000
 BIT11 = 004000
 BIT12 = 010000
 BIT13 = 020000
 BIT14 = 040000
 BIT15 = 100000
 BIT2 = 000004
 BIT3 = 000010
 BIT4 = 000020
 BIT5 = 000040
 BIT6 = 000100
 BIT7 = 000200
 BIT8 = 000400
 BIT9 = 001000
 BPTVEC = 000014
 CKSWR = 104410
 MSG = 040135
 CNTRLC = 040056
 CPUERR = 177766
 CR = 000015
 CRLF = 000200
 DDISP = 177570
 DF1 = 050026
 DF12 = 050051
 DF2 = 050033
 DF3 = 050042
 DF32 = 050057
 DF5 = 050046
 DH1 = 045026
 DH10 = 045176
 DH12 = 045236
 DH13 = 045316
 DH14 = 045376
 DH15 = 045456
 DH16 = 045516
 DH17 = 045576
 DH2 = 045106
 DH20 = 045656
 DH21 = 045716
 DH22 = 046033
 DH24 = 046073
 DH26 = 046133
 DH27 = 046245
 DH30 = 046442
 DH32 = 046517
 DH33 = 046537
 DH34 = 046577
 DH36 = 046670

DH37 = 046730
 DH40 = 046770
 DH41 = 047041
 DH42 = 047145
 DH43 = 047205
 DH44 = 047255
 DH45 = 047305
 DISPLA = 001142
 DISPRE = 000174
 DSWR = 177570
 DT1 = 047336
 DT10 = 047374
 DT12 = 047406
 DT13 = 047424
 DT14 = 047442
 DT15 = 047460
 DT16 = 047472
 DT17 = 047510
 DT2 = 047354
 DT20 = 047526
 DT21 = 047540
 DT22 = 047556
 DT24 = 047570
 DT26 = 047602
 DT27 = 047620
 DT30 = 047636
 DT32 = 047650
 DT34 = 047656
 DT36 = 047672
 DT40 = 047704
 DT41 = 047714
 DT42 = 047730
 DT43 = 047742
 DT45 = 047756
 DT46 = 047772
 DT47 = 050002
 DT50 = 050014
 EM*VEC = 000030
 EM1 = 042264
 EM10 = 042373
 EM11 = 042443
 EM12 = 042512
 EM13 = 042564
 EM14 = 042635
 EM15 = 042716
 EM16 = 043001
 EM2 = 042324
 EM21 = 043057
 EM22 = 043124
 EM23 = 043173
 EM24 = 043226
 EM25 = 043265
 EM26 = 043332
 EM27 = 043400
 EM30 = 043455
 EM31 = 043520
 EM32 = 043563

EM33 = 043621
 EM34 = 043677
 EM35 = 043760
 EM36 = 044014
 EM37 = 044071
 EM40 = 044134
 EM41 = 044221
 EM42 = 044265
 EM43 = 044330
 EM44 = 044375
 EM45 = 044443
 EM46 = 044502
 EM47 = 044544
 EM50 = 044603
 EM51 = 044632
 EM52 = 044670
 EM53 = 044714
 EM54 = 044761
 ERROR = 104000
 ERRYP = 036564
 ERRVEC = 000004
 GTSWR = 104407
 HT = 000011
 INIT = 020564
 IOTVEC = 000020
 KDPAR0 = 172360
 KDPAR1 = 172362
 KDPAR2 = 172364
 KDPAR3 = 172366
 KDPAR4 = 172370
 KDPAR5 = 172372
 KDPAR6 = 172374
 KDPAR7 = 172376
 KDPDR0 = 172320
 KDPDR1 = 172322
 KDPDR2 = 172324
 KDPDR3 = 172326
 KDPDR4 = 172330
 KDPDR5 = 172332
 KDPDR6 = 172334
 KDPDR7 = 172336
 KERSTK = 001100
 KIPAR0 = 172340
 KIPAR1 = 172342
 KIPAR2 = 172344
 KIPAR3 = 172346
 KIPAR4 = 172350
 KIPAR5 = 172352
 KIPAR6 = 172354
 KIPAR7 = 172356
 KIPDR0 = 172300
 KIPDR1 = 172302
 KIPDR2 = 172304
 KIPDR3 = 172306
 KIPDR4 = 172310
 KIPDR5 = 172312
 KIPDR6 = 172314

KIPDR7 = 172316
 KSP = %000006
 LF = 000012
 LOOP = 020456
 MGMERR = 002456
 MGMFLG = 002460
 MMR0 = 177572
 MMR1 = 177574
 MMR2 = 177576
 MMR3 = 172516
 MMVEC = 000250
 NDFLAG = 002202
 NODSPA = 002172
 PBAHI = 001302
 PBALO = 001300
 PIRQ = 177772
 PIRQVE = 000240
 PRO = 000000
 PR1 = 000040
 PR2 = 000100
 PR3 = 000140
 PR4 = 000200
 PR5 = 000240
 PR6 = 000300
 PR7 = 000340
 PS = 177776
 PSW = 177776
 PURMSG = 042222
 PURVEC = 000024
 RDCHR = 104411
 RDLIN = 104412
 RESREG = 104414
 RESVEC = 000010
 R6 = %000006
 R7 = %000007
 SAVREG = 104413
 SCOPE = 000004
 SDPAR0 = 172260
 SDPAR1 = 172262
 SDPAR2 = 172264
 SDPAR3 = 172266
 SDPAR4 = 172270
 SDPAR5 = 172272
 SDPAR6 = 172274
 SDPAR7 = 172276
 SDPDR0 = 172220
 SDPDR1 = 172222
 SDPDR2 = 172224
 SDPDR3 = 172226
 SDPDR4 = 172230
 SDPDR5 = 172232
 SDPDR6 = 172234
 SDPDR7 = 172236
 SIPAR0 = 172240
 SIPAR1 = 172242
 SIPAR2 = 172244
 SIPAR3 = 172246

SIPAR4= 172250	TON 002352	UIPAR1= 177642	SENVN 001245	\$REGAD 001160
SIPAR5= 172252	TPVEC = 000064	UIPAR2= 177644	\$EOP 035654	\$REG0 001162
SIPAR6= 172254	TRAPPC 001260	UIPAR3= 177646	\$EOPCT 035676	\$REG1 001164
SIPAR7= 172256	TRAPPS 001262	UIPAR4= 177650	\$ERFLG 001103	\$REG2 001166
SIPDR0= 172200	TRAPVE= 000034	UIPAR5= 177652	\$ERMAX 001115	\$REG3 001170
SIPDR1= 172202	TRTVEC= 000014	UIPAR6= 177654	\$ERROR 036326	\$REG4 001172
SIPDR2= 172204	TST1 020602	UIPAR7= 177656	\$ERRPC 001116	\$REG5 001174
SIPDR3= 172206	TST10 023076	UIPDR0= 177600	\$ERRTB 001320	\$RESRE 041572
SIPDR4= 172210	TST11 023406	UIPDR1= 177602	\$ERTTL 001112	\$RTNAD 036126
SIPDR5= 172212	TST12 023504	UIPDR2= 177604	\$ESCAP 001212	\$RTRN 036122
SIPDR6= 172214	TST13 023702	UIPDR3= 177606	\$ETABL 001244	\$SAVRE 041534
SIPDR7= 172216	TST14 024224	UIPDR4= 177610	\$ETEND 001254	\$SAVR6 042220
SRO = 177572	TST15 025210	UIPDR5= 177612	\$FATAL 001226	\$SCOPE 036134
SR1 = 177574	TST16 026254	UIPDR6= 177614	\$FFLG 041004	\$SETUP= 000137
SR2 = 177576	TST17 026532	UIPDR7= 177616	\$FILLC 001156	\$STUP = 177777
SR3 = 172516	TST2 021126	USESTK= 000600	\$FILLS 001155	\$SVLAD 036256
SSP = %000006	TST20 027014	USP = %000006	\$GDADR 001120	\$SVPC = 000204
STACK = 001100	TST21 027236	VIRT1 001276	\$GDDAT 001124	\$SWR = 173400
START 020000	TST22 027754	WASR6 001256	\$GET42 036030	\$SWREG 001246
STKMT= 177774	TST23 030452	WASSR0 001264	\$GTSWR 037142	\$SWRMK= 000000
SUPSTK= 000700	TST24 031310	WASSR1 001266	\$HD = 000000	\$TBIT 001310
SWR 001140	TST25 032154	WASSR2 001270	\$HIBTS 000204	\$TESTN 001230
SWREG 000176	TST26 032664	WASSR3 001272	\$ICNT 001104	\$TKB 001146
SW0 = 000001	TST27 033374	WBIT = 000100	\$ILLUP 042214	\$TKS 001144
SW00 = 000001	TST3 021370	\$APTHD 000204	\$INTAG 001135	\$TMP0 001176
SW01 = 000002	TST30 033746	\$ATYC 040564	\$ITEMB 001114	\$TMP1 001200
SW02 = 000004	TST31 034304	\$ATY1 040540	\$LF 001222	\$TMP2 001202
SW03 = 000010	TST32 034656	\$ATY3 040546	\$LFLG 041003	\$TMP3 001204
SW04 = 000020	TST33 034726	\$ATY4 040556	\$LOOP 036124	\$TMP4 001206
SW05 = 000040	TST34 035314	\$AUTOB 001134	\$LPADR 001106	\$TMP5 001210
SW06 = 000100	TST4 021520	\$BDADR 001122	\$LPERR 001110	\$TN = 000035
SW07 = 000200	TST5 022022	\$BDDAT 001126	\$MAIL 001224	\$TPB 001152
SW08 = 000400	TST6 022310	\$BELL 001214	\$MADR 000206	\$TPFLG 001157
SW09 = 001000	TST7 022476	\$BIN 041060	\$MFLG 041002	\$TPS 001150
SW1 = 000002	TYPBN = 104406	\$CHARC 040534	\$MNEW 040045	\$TRAP 041750
SW10 = 002000	TYPDS = 104405	\$CKSWR 037072	\$MSGAD 001240	\$TRAP2 041772
SW11 = 004000	TYPE = 104401	\$CLR.T 036046	\$MSGLG 001242	\$TRP = 000015
SW12 = 010000	TYPOC = 104402	\$CMTAG 001100	\$MSGTY 001224	\$TRPAD 042004
SW13 = 020000	TYPON = 104404	\$CM1 = 000006	\$MSWR 040034	\$TSTM 000210
SW14 = 040000	TYPOS = 104403	\$CM2 = 000014	\$MXCNT 001306	\$TSTM 001102
SW15 = 100000	UDPAR0= 177660	\$CM3 = 000006	\$NULL 001154	\$TTYIN 040012
SW2 = 000004	UDPAR1= 177662	\$CM4 = 000006	\$NWTST= 000001	\$TYPBN 041006
SW3 = 000010	UDPAR2= 177664	\$CNTLC 001312	\$OCNT 041304	\$TYPDS 041310
SW4 = 000020	UDPAR3= 177666	\$CNTLG 040027	\$OCTVL 041732	\$TYPE 040206
SW5 = 000040	UDPAR4= 177670	\$CNTLU 040022	\$OMODE 041306	\$TYPEC 040420
SW6 = 000100	UDPAR5= 177672	\$CPUOP 001252	\$OVER 036312	\$TYPEX 040536
SW7 = 000200	UDPAR6= 177674	\$CRLF 001221	\$PASS 001232	\$TYPOC 041106
SW8 = 000400	UDPAR7= 177676	\$DBLK 041524	\$PASTM 000212	\$TYPON 041122
SW9 = 001000	UDPDR0= 177620	\$DB20 041630	\$PWRAD 042176	\$TYPOS 041062
TBIT = 000020	UDPDR1= 177622	\$DEVCT 001234	\$PWRDN 042036	\$UNIT 001236
TBITPS 001274	UDPDR2= 177624	\$DOAGN 036066	\$PWRMG 042172	\$UNITM 000214
TBITVE= 000014	UDPDR3= 177626	\$DTBL 041514	\$PWRUP 042110	\$USWR 001250
TESTNO 001254	UDPDR4= 177630	\$ENDAD 036056	\$QUES 001220	\$XTSTR 036146
TIMERR 002404	UDPDR5= 177632	\$ENDCT 035704	\$RDCHR 037414	\$GET4= 000001
TIMFLG 002406	UDPDR6= 177634	\$ENULL 036130	\$RDLIN 037534	\$OFILL 041305
TKVEC = 000060	UDPDR7= 177636	\$ENV 001244	\$RDS? = 000010	.\$X 000204
TOFF 002316	UIPAR0= 177640			

. ABS. 050062 000
000000 001

ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 51496 WORDS (202 PAGES)

DYNAMIC MEMORY: 20434 WORDS (78 PAGES)

ELAPSED TIME: 00:10:21

CKKTBAO.BIN,CKKTBAO.SEQ/CRF=CKKTBAO.MLB/ML,CKKTBAO.P11

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
ABASE	=	000000	10-578
ACDW1	=	000000	10-578
ACDW2	=	000000	10-578
ACPUOP	=	000000	10-578 10-578
ADDW0	=	000000	10-578
ADDW1	=	000000	10-578
ADDW10	=	000000	10-578
ADDW11	=	000000	10-578
ADDW12	=	000000	10-578
ADDW13	=	000000	10-578
ADDW14	=	000000	10-578
ADDW15	=	000000	10-578
ADDW2	=	000000	10-578
ADDW3	=	000000	10-578
ADDW4	=	000000	10-578
ADDW5	=	000000	10-578
ADDW6	=	000000	10-578
ADDW7	=	000000	10-578
ADDW8	=	000000	10-578
ADDW9	=	000000	10-578
ADEVCT	=	000000	10-578 10-578
ADEVN	=	000000	10-578 10-578
AENV	=	000000	10-578 10-578
AENVN	=	000000	10-578 10-578
AFATAL	=	000000	10-578 10-578
AMADR1	=	000000	10-578
AMADR2	=	000000	10-578
AMADR3	=	000000	10-578
AMADR4	=	000000	10-578
AMAMS1	=	000000	10-578
AMAMS2	=	000000	10-578
AMAMS3	=	000000	10-578
AMAMS4	=	000000	10-578
AMSGAD	=	000000	10-578 10-578
AMSGLG	=	000000	10-578 10-578
AMSGTY	=	000000	10-578 10-578
AMTYP1	=	000000	10-578
AMTYP2	=	000000	10-578
AMTYP3	=	000000	10-578
AMTYP4	=	000000	10-578
APASS	=	000000	10-578 10-578
APRINI	=	002020	#12-842 16-1044 25-2324
APRIOR	=	000000	10-578
APTCSU	=	000040	40-3743 #40-3744
APTENV	=	000001	39-3637 40-3743 40-3744 #40-3744
APTSIZ	=	000200	16-1024 #40-3744
APTSP0	=	000100	40-3743 40-3744 #40-3744
ASWREG	=	000000	10-578 10-578
ATESTN	=	000000	10-578 10-578
AUNIT	=	000000	10-578 10-578
AUSWR	=	000000	10-578 10-578
AVECT1	=	000000	10-578

CKKTBAO		CREATED BY MALRO ON 25-SEP-79 AT 12:29		PAGE 2		M 9					
SYMBOL CROSS REFERENCE		REFERENCES		CREF	D9D	YZ	SEQ 0116				
SYMBOL	VALUE										
AVECT2	= 000000	10-578									
BADPC	= 001304	#10-578	*13-903	45-3857							
BIT0	= 000001	#8-525	38-3383	38-3437	38-3575	38-3620					
BIT00	= 000001	#8-525	8-525								
BIT01	= 000002	#8-525	8-525								
BIT02	= 000004	#8-525	8-525								
BIT03	= 000010	#8-525	8-525								
BIT04	= 000020	#8-525	8-525								
BIT05	= 000040	#8-525	8-525								
BIT06	= 000100	#8-525	8-525								
BIT07	= 000200	#8-525	8-525								
BIT08	= 000400	#8-525	8-525	39-3636							
BIT09	= 001000	#8-525	8-525	39-3636	39-3637						
BIT1	= 000002	#8-525	37-3245	38-3291	38-3317	38-3371	38-3498	38-3542			
BIT10	= 002000	#8-525	39-3637								
BIT11	= 004000	#8-525									
BIT12	= 010000	#8-525	39-3635								
BIT13	= 020000	#8-525	39-3637								
BIT14	= 040000	#8-525	39-3636								
BIT15	= 100000	#8-525									
BIT2	= 000004	#8-525	13-894	13-918	24-2274	24-2282					
BIT3	= 000010	#8-525									
BIT4	= 000020	#8-525	8-536	16-1043							
BIT5	= 000040	#8-525									
BIT6	= 000100	#8-525	8-537								
BIT7	= 000200	#8-525									
BIT8	= 000400	#8-525									
BIT9	= 001000	#8-525									
BPTVEC	= 000014	#8-525									
CKSWR	= 104410	39-3636	39-3637	39-3637	#42-3752						
CMSG	= 040135	39-3728	#39-3739								
CNTRLC	= 040056	39-3722	39-3722	#39-3726							
CPUEER	= 177766	#8-538	*15-984	45-3837							
CR	= 000015	#8-525	40-3743	40-3743							
CRLF	= 000200	#8-525	16-1025	16-1025	40-3743	40-3743	44-3810	44-3814	44-3816	44-3817	
		44-3818	44-3819	44-3821	44-3829						
DDISP	= 177570	#8-525	10-578	16-1024							
DF1	= 050026	11-724	11-755	11-767	11-780	#46-3865					
DF12	= 050051	11-584	11-608	11-614	11-620	11-632	11-638	11-651	11-682	11-692	
		#46-3869									
DF2	= 050033	11-590	11-786	11-792	11-798	11-825	11-831	#46-3866			
DF3	= 050042	11-596	11-602	11-626	11-644	11-657	11-663	11-669	11-699	11-706	
		11-718	11-736	11-742	11-761	11-774	11-805	11-819	#46-3867		
DF32	= 050057	11-712	11-730	#46-3870							
DF5	= 050046	11-675	11-748	#46-3868							
DH1	= 045026	11-582	#44-3800								
DH10	= 045176	11-594	11-600	#44-3802							
DH12	= 045236	11-606	#44-3803								
DH13	= 045316	11-612	#44-3804								
DH14	= 045376	11-618	#44-3805								
DH15	= 045456	11-624	#44-3806								
DH16	= 045516	11-630	#44-3807								

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
DH17		045576	11-636 #44-3808
DH2		045106	11-588 #44-3801
DH20		045656	11-642 11-661 11-673 #44-3809
DH21		045716	11-648 #44-3810
DH22		046033	11-655 #44-3812
DH24		046073	11-667 #44-3813
DH26		046133	11-679 #44-3814
DH27		046245	11-686 #44-3816
DH30		046442	11-696 11-703 11-771 11-802 11-809 11-816 #44-3821
DH32		046517	11-710 11-728 11-784 11-829 #44-3823
DH33		046537	11-716 #44-3824
DH34		046577	11-722 #44-3825
DH36		046670	11-734 #44-3826
DH37		046730	11-740 #44-3827
DH40		046770	11-746 #44-3828
DH41		047041	11-752 #44-3829
DH42		047145	11-759 #44-3831
DH43		047205	11-765 11-778 #44-3832
DH44		047255	11-790 11-796 11-823 #44-3833
DH45		047305	#44-3834
DISPLA		001142	#10-578 *16-1024 *16-1024 39-3636 39-3637
DISPRE		000174	#9-575 16-1024
DSWR	=	7570	#8-525 10-578 16-1024
DT1		047336	11-583 #45-3837
DT10		047374	11-595 11-601 #45-3839
DT12		047406	11-607 #45-3840
DT13		047424	11-613 #45-3841
DT14		047442	11-619 #45-3842
DT15		047460	11-625 #45-3843
DT16		047472	11-631 #45-3844
DT17		047510	11-637 #45-3845
DT2		047354	11-589 #45-3838
DT20		047526	11-643 11-662 11-674 #45-3846
DT21		047540	11-650 #45-3847
DT22		047556	11-656 11-717 #45-3848
DT24		047570	11-668 11-741 #45-3849
DT26		047602	11-681 #45-3850
DT27		047620	11-691 #45-3851
DT30		047636	11-698 11-705 11-773 11-804 #45-3852
DT32		047650	11-711 11-729 11-785 11-830 #45-3853
DT34		047656	11-723 #45-3854
DT36		047672	11-735 #45-3855
DT40		047704	11-747 #45-3856
DT41		047714	11-754 #45-3857
DT42		047730	11-760 #45-3858
DT43		047742	11-766 #45-3859
DT45		047756	11-779 #45-3860
DT46		047772	11-791 11-797 11-824 #45-3861
DT47		050002	11-811 #45-3862
DT50		050014	11-818 #45-3863
EMTVEC	=	000030	#8-525 16-1024 16-1024
EM1		042264	11-581 #43-3761

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES								
EM10		042373	11-593	#43-3763							
EM11		042443	11-599	#43-3764							
EM12		042512	11-605	#43-3765							
EM13		042564	11-611	11-635	11-641	#43-3766					
EM14		042635	11-617	#43-3767							
EM15		042716	11-623	#43-3768							
EM16		043001	11-629	#43-3769							
EM2		042324	11-587	#43-3762							
EM21		043057	11-647	#43-3770							
EM22		043124	11-654	#43-3771							
EM23		043173	11-660	#43-3772							
EM24		043226	11-666	#43-3773							
EM25		043265	11-672	#43-3774							
EM26		043332	11-678	#43-3775							
EM27		043400	11-685	#43-3776							
EM30		043455	11-695	#43-3777							
EM31		043520	11-702	#43-3778							
EM32		043563	11-709	#43-3779							
EM33		043621	11-715	#43-3780							
EM34		043677	11-721	#43-3781							
EM35		043760	11-727	#43-3782							
EM36		044014	11-733	#43-3783							
EM37		044071	11-739	#43-3784							
EM40		044134	11-745	#43-3785							
EM41		044221	11-751	#43-3786							
EM42		044265	11-758	#43-3787							
EM43		044330	11-764	11-777	#43-3788						
EM44		044375	11-770	#43-3789							
EM45		044443	11-783	#43-3790							
EM46		044502	11-789	#43-3791							
EM47		044544	11-795	#43-3792							
EM50		044603	11-801	#43-3793							
EM51		044632	11-808	#43-3794							
EM52		044670	11-815	#43-3795							
EM53		044714	11-822	#43-3796							
EM54		044761	11-828	#43-3797							
ERROR	=	104000	#8-525	13-911	13-913	15-982	15-1013	17-1093	17-1101	17-1109	17-1116
			18-1167	18-1175	19-1184	19-1191	19-1225	19-1236	19-1244	19-1303	19-1314
			19-1321	19-1335	19-1377	19-1388	19-1395	19-1409	19-1453	19-1491	19-1500
			19-1524	19-1540	19-1549	20-1595	20-1621	20-1654	21-1684	21-1702	21-1772
			24-2288	24-2292	25-2343	25-2348	25-2361	25-2373	25-2384	25-2396	25-2411
			25-2424	25-2439	25-2454	27-2492	28-2498	28-2511	29-2524	30-2536	31-2549
			34-2567	34-2580	34-2595	34-2610	35-2643	35-2661	35-2678	35-2693	35-2709
			35-2727	35-2744	35-2763	35-2777	36-2798	36-2816	36-2833	36-2848	36-2864
			36-2882	36-2899	36-2918	36-2932	37-2970	37-2975	37-2988	37-3000	37-3011
			37-3023	37-3037	37-3050	37-3064	37-3080	37-3105	37-3110	37-3123	37-3135
			37-3146	37-3158	37-3172	37-3185	37-3199	37-3215	37-3256	37-3266	38-3277
			38-3287	38-3300	38-3330	38-3342	38-3355	38-3368	38-3376	38-3396	38-3408
			38-3421	38-3434	38-3442	38-3463	38-3470	38-3509	38-3519	38-3529	38-3539
			38-3552	38-3587	38-3597	38-3607	38-3617	38-3629			
ERRTYP	036564		39-3637	#39-3639							
ERRVEC	000004		#8-525	16-1024	16-1024	16-1024	*16-1034	*16-1035	39-3636	39-3636	39-3636

SYMBOL	CROSS REFERENCE VALUE	REFERENCES								
GNS	= *****	39-3636								
		9-575	9-575	16-1025	39-3635	39-3635	42-3752	42-3752	42-3752	42-3752
		42-3752	42-3752	42-3752	42-3752	42-3752	42-3752	42-3752	42-3752	42-3752
		42-3752	42-3752	42-3752	42-3752	42-3752	42-3752	42-3752	42-3752	42-3752
		42-3752	42-3752							
GTSWR	= 104407	16-1025	39-3735	#42-3752						
HT	= 000011	#8-525	40-3743	40-3743						
INIT	020564	#16-1042								
IOTVEC	= 000020	#8-525	16-1024	16-1024						
KDPA0	= 172360	#8-526	12-858	*24-2275	*24-2280					
KDPA1	= 172362	#8-526								
KDPA2	= 172364	#8-526								
KDPA3	= 172366	#8-526								
KDPA4	= 172370	#8-526	*22-1813	*24-2273						
KDPA5	= 172372	#8-526								
KDPA6	= 172374	#8-526								
KDPA7	= 172376	#8-526								
KDPDR0	= 172320	#8-526								
KDPDR1	= 172322	#8-526	*22-1810	*24-1967						
KDPDR2	= 172324	#8-526	*22-1811	*24-1968						
KDPDR3	= 172326	#8-526	*22-1812	*24-1969						
KDPDR4	= 172330	#8-526	*24-2272	*37-3238	*38-3453	*38-3490				
KDPDR5	= 172332	#8-526								
KDPDR6	= 172334	#8-526								
KDPDR7	= 172336	#8-526	12-849							
KERSTK	- 001100	#8-539	19-1230	19-1291	19-1295	19-1308	19-1365	19-1369	19-1382	19-1449
		19-1508	19-1513	20-1578	20-1600	20-1626	21-1688	21-1748	24-2133	24-2281
		25-2337	26-2485	37-2967	37-3102					
KIPAR0	172340	#8-526	12-857							
KIPAR1	= 172342	#8-526								
KIPAR2	= 172344	#8-526								
KIPAR3	= 172346	#8-526	*17-1074	*19-1436						
KIPAR4	= 172350	#8-526	*17-1075	*19-1437	*21-1677	*25-2326				
KIPAR5	= 172352	#8-526	*19-1483							
KIPAR6	= 172354	#8-526								
KIPAR7	= 172356	#8-526	12-867							
KIPDR0	= 172300	#8-526	12-845							
KIPDR1	= 172302	#8-526								
KIPDR2	= 172304	#8-526								
KIPDR3	= 172306	#8-526	*19-1284	*19-1438	*24-1970	*24-2132				
KIPDR4	- 172310	#8-526	*17-1085	*18-1159	*19-1289	*19-1363	*19-1439	*19-1464	*19-1484	*19-1554
		*21-1679	*21-1709	*22-1809	*24-1966	*25-2331	*25-2445	*26-2479	*34-2601	*35-2655
		*35-2657	*35-2672	*35-2674	*35-2687	*35-2689	*35-2703	*35-2705	*35-2721	*35-2723
		*35-2738	*35-2740	*35-2757	*35-2759	*36-2810	*36-2812	*36-2827	*36-2829	*36-2842
		*36-2844	*36-2858	*36-2860	*36-2876	*36-2878	*36-2893	*36-2895	*36-2912	*36-2914
		*37-3246	*38-3290	*38-3324	*38-3326	*38-3336	*38-3338	*38-3349	*38-3351	*38-3362
		*38-3364	*38-3390	*38-3392	*38-3402	*38-3404	*38-3415	*38-3417	*38-3428	*38-3430
		*38-3499	*38-3544	*38-3577	*38-3622					
KIPDR5	- 172312	#8-526	*19-1285	*19-1485	*19-1555					
KIPDR6	= 172314	#8-526								
KIPDR7	- 172316	#8-526								
KSP	- %000006	#8-533	13-903	*13-904	*13-905	*13-916	*13-917	*15-979	*15-980	15-981

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
PR2	=	000100	#8-525
PR3	=	000140	#8-525
PR4	=	000200	#8-525
PR5	=	000240	#8-525
PR6	=	000300	#8-525
PR7	=	000340	#8-525
PS	=	177776	#8-525
PSW	=	177776	#8-525 8-525
			14-931 14-933 *16-1029 *16-1031 *16-1033 17-1091 *17-1124 *17-1129
			*17-1135 *17-1137 18-1165 *19-1199 *19-1204 *19-1206 *19-1224 *20-1577 *20-1585
			20-1592 *20-1594 *20-1599 *20-1602 *20-1604 *20-1611 20-1618 *20-1620 *20-1625
			*20-1628 *20-1630 *20-1645 20-1649 *20-1651 *21-1738 *21-1747 *21-1749 *21-1751
			*24-2148 *24-2206 *24-2251 24-2277 24-2279 *25-2335 *25-2355 *25-2367 *25-2379
			*25-2390 *25-2404 *25-2418 *25-2431 *26-2483 *28-2505 *28-2517 *29-2530 *31-2543
			*31-2557 *34-2574 *34-2587 *35-2636 *35-2647 *35-2668 *35-2684 *35-2699 *35-2716
			*35-2734 *35-2751 *36-2791 *36-2802 *36-2823 *36-2839 *36-2854 *36-2871 *36-2889
			*36-2906 *37-2954 *37-2962 *37-2981 *37-2994 *37-3006 *37-3017 *37-3030 *37-3044
			*37-3057 *37-3069 *37-3089 *37-3097 *37-3116 *37-3129 *37-3141 *37-3152 *37-3165
			*37-3179 *37-3192 *37-3204 *37-3250 *37-3260 *37-3270 *38-3281 *38-3320 *38-3333
			*38-3345 *38-3358 *38-3386 *38-3399 *38-3411 *38-3424 38-3454 *38-3503 *38-3513
			*38-3523 *38-3533 *38-3581 *38-3591 *38-3601 *38-3611 *38-3621
			42-3753 #42-3754
PWRMSG	042222		
PWRVEC	000024	#8-525	16-1024 16-1024 42-3753 42-3753 42-3753 42-3753 42-3753 42-3753
RDCHR	- 104411	39-3722	#42-3752
RDLIN	= 104412	#42-3752	
RESREG	- 104414	41-3750	#42-3752
RESVEC	- 000010	#8-525	16-1024 16-1024 16-1024
R6	%000006	#8-525	*16-1024 *16-1024 16-1024
R7	-%000007	#8-525	
SAVREG	104413	41-3750	#42-3752
SCOPE	- 000004	#8-525	17-1071 18-1151 19-1220 19-1283 19-1359 19-1435 19-1482 20-1574
			20-1641 21-1676 21-1727 22-1805 24-1964 24-2144 24-2202 24-2261 25-2323
			26-2474 35-2631 36-2785 37-2952 37-3087 37-3233 38-3314 38-3379 38-3452
			38-3484 38-3568 39-3635
SDPAR0	= 172260	#8-526	12-870
SDPAR1	= 172262	#8-526	
SDPAR2	= 172264	#8-526	
SDPAR3	= 172266	#8-526	
SDPAR4	= 172270	#8-526	*38-3487
SDPAR5	= 172272	#8-526	
SDPAR6	= 172274	#8-526	
SDPAR7	= 172276	#8-526	
SDPDR0	= 172220	#8-526	
SDPDR1	= 172222	#8-526	*24-2145 *24-2156
SDPDR2	= 172224	#8-526	
SDPDR3	= 172226	#8-526	
SDPDR4	= 172230	#8-526	*37-3239 *38-3494 *38-3576
SDPDR5	= 172232	#8-526	
SDPDR6	= 172234	#8-526	
SDPDR7	= 172236	#8-526	12-853
SIPAR0	= 172240	#8-526	12-869 *20-1608 *20-1627
SIPAR1	= 172242	#8-526	
SIPAR2	= 172244	#8-526	

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
SIPAR3	=	172246	#8-526 *17-1076
SIPAR4	=	172250	#8-526 *17-1077 *25-2327
SIPAR5	=	172252	#8-526
SIPAR6	=	172254	#8-526
SIPAR7	=	172256	#8-526 12-872
SIPDR0	=	172200	#8-526 12-851
SIPDR1	=	172202	#8-526
SIPDR2	=	172204	#8-526
SIPDR3	=	172206	#8-526 *24-2157 *24-2191
SIPDR4	=	172210	#8-526 *17-1086 *18-1160 *35-2632 *37-2961 *37-3071 *38-3380 *38-3491
SIPDR5	=	172212	#8-526
SIPDR6	=	172214	#8-526
SIPDR7	=	172216	#8-526
SRO	=	177572	#8-526 8-529 15-1010 *15-1012 17-1105 19-1180 19-1232 *19-1248 19-1309 *19-1325 19-1332 *19-1334 19-1383 *19-1399 19-1406 *19-1408 *19-1457 19-1509 19-1514 19-1530 *19-1544 21-1690 *21-1706 21-1744 *21-1746 25-2451 *25-2453 34-2607 *34-2609 35-2774 *35-2776 36-2929 *36-2931 37-3077 *37-3079 37-3212 *37-3214
SR1	=	177574	#8-526 8-530
SR2	=	177576	#8-526 8-531 15-1011 17-1106 19-1181 19-1241 19-1310 19-1333 19-1384 19-1407 19-1450 19-1487 19-1496 19-1510 19-1515 19-1534 19-1545 21-1691 21-1745 25-2452 34-2608 35-2775 36-2930 37-3078 37-3213
SR3	=	172516	#8-526 8-532
SSP	=	%000006	#8-534 *16-1030 *20-1612 *20-1629 25-2336 *35-2638 35-2639 *35-2649 37-2964 *37-2966 *37-2985 *37-2997 *37-3008 *37-3020 *37-3034 *37-3047 *37-3061 #8-525 8-539 8-540 8-541 16-1024 16-1028 38-3455 38-3474
STACK	=	001100	#8-525
START	=	020000	9-575 #16-1024 42-3753
STKLMT	=	177774	#8-525
SUPSTK	=	000700	#8-540 16-1030 20-1612 20-1629 25-2340 35-2648 37-2964
SWR	=	001140	#10-578 16-1024 *16-1024 16-1024 *16-1024 *16-1024 16-1025 39-3635 39-3636 39-3636 39-3636 39-3637 39-3637 39-3637 39-3637 39-3722 39-3722 42-3753 42-3753
SWREG	=	000176	#9-575 16-1024 16-1025 39-3722 39-3722
SW0	=	000001	#8-525
SW00	=	000001	#8-525 8-525
SW01	=	000002	#8-525 8-525
SW02	=	000004	#8-525 8-525
SW03	=	000010	#8-525 8-525
SW04	=	000020	#8-525 8-525
SW05	=	000040	#8-525 8-525
SW06	=	000100	#8-525 8-525
SW07	=	000200	#8-525 8-525
SW08	=	000400	#8-525 8-525
SW09	=	001000	#8-525 8-525
SW1	=	000002	#8-525
SW10	=	002000	#8-525
SW11	=	004000	#8-525
SW12	=	010000	#8-525
SW13	=	020000	#8-525
SW14	=	040000	#8-525
SW15	=	100000	#8-525
SW2	=	000004	#8-525

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
SW3	=	000010	#8-525
SW4	=	000020	#8-525
SW5	=	000040	#8-525
SW6	=	000100	#8-525
SW7	=	000200	#8-525
SW8	=	000400	#8-525
SW9	=	001000	#8-525
TBIT	=	000020	#8-536 14-931 14-936 14-949
TBITPS		001274	#10-578 *14-934 14-949 14-951 *14-952 *16-1041
TBITVE	=	000014	#8-525 16-1024 16-1024
TESTNO		001254	#10-578 *39-3637 45-3837 45-3838 45-3839 45-3840 45-3841 45-3842 45-3843 45-3844 45-3845 45-3846 45-3847 45-3848 45-3849 45-3850 45-3851 45-3852 45-3853 45-3854 45-3855 45-3856 45-3857 45-3858 45-3859 45-3860 45-3861 45-3862 45-3863
TIMERR		002404	#15-971 16-1034 20-1631 21-1707 21-1779
TIMFLG		002406	#15-972 *15-983 *16-1039
TKVEC	=	000060	#8-525
TOFF		002316	#14-931 20-1575 21-1728 24-2262
TON		002352	#14-949 20-1633 21-1784
TPVEC	=	000064	#8-525
TRAPPC		001260	#10-578 *13-904 13-917 *15-979 15-986 *15-1007 15-1016 *19-1330 19-1340 *19-1404 19-1414 *25-2449 25-2459 *34-2605 34-2615 *35-2772 35-2782 *36-2927 36-2937 *37-3075 37-3085 *37-3210 37-3220 *38-3298 38-3302 *38-3550 38-3554 *38-3627 38-3631 45-3837 45-3838
TRAPPS		001262	#10-578 *13-905 13-916 *15-980 15-985 *15-1008 15-1015 *19-1331 19-1339 *19-1405 19-1413 *25-2450 25-2458 *34-2606 34-2614 *35-2773 35-2781 *36-2928 36-2936 *37-3076 37-3084 *37-3211 37-3219 *38-3299 38-3301 *38-3551 38-3553 *38-3628 38-3630 45-3837 45-3838
TRAPVE	-	000034	#8-525 16-1024 16-1024
TRTVEC	-	000014	#8-525
TST1		020602	#17-1071
TST10		023076	#20-1574
TST11		023406	#20-1641
TST12		023504	#21-1676
TST13		023702	#21-1727
TST14		024224	#22-1805
TST15		025210	#24-1964
TST16		026254	#24-2144
TST17		026532	#24-2202
TST2		021126	#18-1151
TST20		027014	#24-2261
TST21		027236	#25-2323
TST22		027754	25-2446 #26-2474
TST23		030452	34-2602 #35-2631
TST24		031310	35-2769 #36-2785
TST25		032154	36-2924 #37-2952
TST26		032664	37-3072 #37-3087
TST27		033374	37-3207 #37-3233
TST3		021370	#19-1220
TST30		033746	38-3292 #38-3314
TST31		034304	38-3372 #38-3379
TST32		034656	38-3438 #38-3452

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
TST33		034726	#38-3484
TST34		035314	38-3546 #38-3568
TST4		021520	#19-1283
TST5		022022	#19-1359
TST6		022310	#19-1435
TST7		022476	#19-1482
TYPBN	=	104406	39-3684 #42-3752
TYPDS	=	104405	39-3635 39-3678 39-3734 #42-3752
TYPE		104401	16-1025 39-3635 39-3635 39-3635 39-3637 39-3639 39-3656 39-3658
			39-3661 39-3663 39-3693 39-3707 39-3713 39-3718 39-3722 39-3722 39-3722
			39-3722 39-3722 39-3722 39-3722 39-3722 39-3722 39-3722 39-3722 39-3722 39-3722
			39-3722 39-3722 39-3722 39-3722 39-3722 39-3722 39-3722 39-3722 39-3722 39-3722
			40-3746 41-3748 #42-3752 42-3753 39-3728 39-3732 40-3743 40-3745
			39-3647 39-3672 39-3722 39-3731 #42-3752
TYPOC	=	104402	#42-3752
TYPON	=	104404	#42-3752
TYPOS	=	104403	#42-3752
UDPAR0	=	177660	#8-526 12-876
UDPAR1	=	177662	#8-526
UDPAR2	=	177664	#8-526
UDPAR3	=	177666	#8-526
UDPAR4	=	177670	#8-526 *38-3571
UDPAR5	=	177672	#8-526
UDPAR6	=	177674	#8-526
UDPAR7	=	177676	#8-526
UDPDR0	=	177620	#8-526
UDPDR1	=	177622	#8-526 *24-2203 *24-2214
UDPDR2	=	177624	#8-526
UDPDR3	=	177626	#8-526
UDPDR4	=	177630	#8-526 *37-3240 *38-3492 *38-3545
UDPDR5	=	177632	#8-526
UDPDR6	=	177634	#8-526
UDPDR7	=	177636	#8-526
UIPAR0	=	177640	#8-526 12-875 *20-1582 *20-1601
UIPAR1	=	177642	#8-526
UIPAR2	=	177644	#8-526
UIPAR3	=	177646	#8-526 *17-1078 *21-1729
UIPAR4	=	177650	#8-526 *17-1079 *21-1730 *26-2476 *36-2787
UIPAR5	=	177652	#8-526
UIPAR6	=	177654	#8-526
UIPAR7	=	177656	#8-526
UIPDR0	=	177600	#8-526 12-855
UIPDR1	=	177602	#8-526
UIPDR2	=	177604	#8-526
UIPDR3	=	177606	#8-526 *21-1731 *21-1782 *24-2215 *24-2249
UIPDR4	=	177610	#8-526 *17-1087 *18-1161 *21-1732 *36-2786 *37-3096 *37-3206 *37-3241 *38-3381
			*38-3493
UIPDR5	=	177612	#8-526
UIPDR6	=	177614	#8-526
UIPDR7	=	177616	#8-526
USESTK	=	000600	#8-541 16-1032 20-1586 20-1603 21-1750 27-2489 36-2803 37-3099
USP	=	%000006	#8-535 *16-1032 *20-1586 *20-1603 *21-1739 *21-1750 26-2484 *36-2793 36-2794
			*36-2804 37-3099 *37-3101 *37-3120 *37-3132 *37-3143 *37-3155 *37-3169 *37-3182

CKKTBAO		CREATED BY	MACRO	ON	25-SEP-79	AT	12:29	PAGE	11	I	10	SEQ	0125
SYMBOL CROSS REFERENCE													
SYMBOL	VALUE	REFERENCES											
VIRT1	001276	*37-3196	*38-3584	*38-3594	*38-3604	*38-3614							
WASR6	001256	#10-578	*15-981	*15-1009	45-3837	45-3838	45-3861	45-3862	45-3863				
WASSRO	001264	#10-578	*13-906	13-909	*15-1010	*17-1105	17-1107	*19-1180	19-1182	*19-1232			
		19-1234	*19-1309	19-1312	*19-1332	*19-1383	19-1386	*19-1406	*19-1514	19-1516			
		*19-1530	19-1531	*21-1690	21-1693	*21-1744	21-1763	*25-2451	*34-2607	*35-2774			
		*36-2929	*37-3077	*37-3212	*38-3294	*38-3547	*38-3624	45-3838	45-3840	45-3842			
		45-3844	45-3847	45-3848	45-3850	45-3851	45-3855	45-3857	45-3860				
WASSR1	001266	#10-578	*13-907	*38-3295	*38-3548	*38-3625	45-3857	45-3860					
WASSR2	001270	#10-578	*13-908	*15-1011	*17-1106	17-1114	*19-1181	19-1189	*19-1241	19-1242			
		*19-1310	19-1319	*19-1333	*19-1384	19-1393	*19-1407	*19-1450	19-1451	*19-1487			
		19-1489	*19-1496	19-1498	*19-1515	19-1519	*19-1534	19-1535	*19-1545	19-1547			
		*21-1691	21-1697	*21-1745	21-1766	*25-2452	*34-2608	*35-2775	*36-2930	*37-3078			
		*37-3213	*38-3296	*38-3549	*38-3626	45-3838	45-3841	45-3842	45-3845	45-3846			
		45-3847	45-3848	45-3850	45-3851	45-3857	45-3860						
WASSR3	001272	#10-578	*38-3297										
WBIT	= 000100	#8-537											
SAPTHD	000204	9-577	#9-577										
SASTAT	= *****	40-3744	40-3744										
SATYC	040564	40-3744	#40-3744										
SATY1	040540	#40-3744											
SATY3	040546	40-3743	#40-3744										
SATY4	040556	39-3637	#40-3744										
SAUTOB	001134	#10-578	*16-1025	39-3722	39-3722	39-3722							
SBADR	001122	#10-578											
SBDDAT	001126	#10-578											
SBELL	001214	#10-578	39-3637	39-3637	39-3637								
SBIN	041060	*40-3745	*40-3745	40-3745	#40-3745								
SCHARC	040534	*40-3743	*40-3743	40-3743	*40-3743	#40-3743							
SCKSWR	037072	#39-3722	42-3752	42-3752									
SCLR.T	036046	39-3635	#39-3635										
SCM1AG	001100	#10-578	16-1024	16-1024	16-1024	16-1024	16-1024	16-1024					
SCM1	- 000006	#10-578	10-578	10-578	#10-578	10-578	10-578	#10-578	10-578	10-578			
		#10-578	10-578	10-578	#10-578	10-578	10-578	#10-578	10-578	10-578			
		#10-578	10-578	10-578	#10-578	10-578	10-578	#10-578	10-578	10-578			
SCM2	- 000014	#10-578	10-578	10-578	#10-578	10-578	10-578	#10-578	10-578	10-578			
		#10-578	10-578	10-578	#10-578	10-578	10-578	#10-578	10-578	10-578			
SCM3	000006	#10-578	10-578	10-578									
SCM4	- 000006	#10-578	10-578	10-578	#10-578	10-578	10-578	#10-578	10-578	10-578			
		#10-578	10-578	10-578	#10-578	10-578	10-578	#10-578	10-578	10-578			
SCNTLC	001312	#10-578	39-3722	39-3722	39-3722	39-3722							
SCNTL3	040027	39-3722	#39-3722										
SCNTLU	040022	39-3722	39-3722	#39-3722									
SCPUOP	001252	#10-578											
SCRLF	001221	#10-578	39-3635	39-3637	39-3637	39-3637	39-3639	39-3658	39-3663	39-3718			
		39-3722	39-3722	39-3722	39-3722	40-3743	40-3743	40-3743					
SDBLK	041524	41-3748	41-3748	#41-3748									
SDB20	041630	39-3690	39-3704	#41-3750									
SDEVCT	001234	#10-578											
SDOAGN	036066	39-3635	39-3635	39-3635	#39-3635								

CKKTBAO
SYMBOL CROSS REFERENCE

CREATED BY MACRO ON 25-SEP-79 AT 12:29

PAGE 12
CREF D9D YZ

J 10

SEQ 0126

SYMBOL	VALUE	REFERENCES										
\$DTBL	041514	41-3748	#41-3748									
\$ENDAD	036056	9-576	16-1025	#39-3635	39-3637							
\$ENDCT	035704	16-1024	#39-3635									
\$ENULL	036130	#39-3635										
\$ENV	001244	#10-578	16-1025	39-3637	40-3743	40-3744	40-3744					
\$ENV*	001245	#10-578	16-1024	40-3743	40-3743	40-3744						
\$EOP	035654	38-3623	#39-3635	39-3736								
\$EOPCT	035676	*16-1024	#39-3635	39-3635								
\$ERFLG	001103	#10-578	39-3636	39-3636	*39-3636	39-3636	39-3636	*39-3637	39-3637	39-3637		
\$ERMAX	001115	#10-578	*16-1024	*39-3636	39-3636	39-3636						
\$ERROR	036326	16-1024	#39-3637									
\$ERRPC	001116	#10-578	*39-3637	*39-3637	39-3637	39-3637	39-3637	39-3645	45-3837	45-3838		
		45-3839	45-3840	45-3841	45-3842	45-3843	45-3844	45-3845	45-3846	45-3847		
		45-3848	45-3849	45-3850	45-3851	45-3852	45-3853	45-3854	45-3855	45-3856		
		45-3858	45-3859	45-3860	45-3861	45-3862	45-3863					
\$ERRTB	001320	#11-578	39-3653									
\$ERTTL	001112	#10-578	39-3635	*39-3635	*39-3637	39-3637	39-3637					
\$ESCAP	001212	#10-578	*16-1024	*39-3636	39-3637	39-3637	39-3637	39-3637				
\$ETABL	001244	#10-578										
\$ETEND	001254	9-577	#10-578									
\$FATAL	001226	#10-578	*40-3744									
\$FFLG	041004	*40-3744	*40-3744	40-3744	*40-3744	#40-3744						
\$FILLC	001156	#10-578	40-3743	40-3743	40-3743							
\$FILLS	001155	#10-578	40-3743	40-3743								
\$GDADR	001120	#10-578										
\$GDDAT	001124	#10-578										
\$GET42	036030	#39-3635										
\$GTSWR	037142	#39-3722	42-3752	42-3752								
\$HD	= 000000	8-523	8-523	8-523								
\$HIBTS	000204	#9-577										
\$ICNT	001104	#10-578										
\$ILLUP	042214	42-3753	42-3753	#42-3753								
\$INTAG	001135	#10-578	39-3722	39-3722	39-3722	39-3722						
\$ITEMB	001114	#10-578	*39-3637	39-3637	39-3637	39-3637	39-3642					
\$LF	001222	#10-578	39-3637	39-3637	39-3722	39-3722	39-3722	40-3743	40-3743			
\$LFLG	041003	*40-3744	#40-3744									
\$LOOP	036124	39-3635	#39-3635									
\$LPADR	001106	#10-578	*16-1024	*37-3234	*38-3485	*38-3569	*39-3636	*39-3636	39-3636	39-3636		
		39-3636										
\$LPERR	001110	#10-578	*16-1024	*17-1088	*17-1138	*18-1162	*19-1207	*19-1221	*19-1250	*19-1290		
		*19-1294	*19-1300	*19-1343	*19-1364	*19-1368	*19-1374	*19-1417	*19-1443	*19-1463		
		*19-1486	*19-1495	*19-1504	*19-1553	*20-1576	*20-1632	*20-1643	*20-1662	*21-1682		
		*21-1710	*21-1737	*21-1783	*22-1815	*23-1840	*24-1850	*24-1864	*24-1875	*24-1886		
		*24-1895	*24-1905	*24-1915	*24-1926	*24-1935	*24-1944	*24-1953	*24-1978	*24-1992		
		*24-2003	*24-2042	*24-2054	*24-2063	*24-2080	*24-2090	*24-2104	*24-2114	*24-2124		
		*24-2134	*24-2146	*24-2160	*24-2167	*24-2176	*24-2185	*24-2192	*24-2204	*24-2218		
		*24-2225	*24-2234	*24-2242	*24-2250	*24-2263	*24-2296	*25-2334	*25-2353	*25-2366		
		*25-2378	*25-2389	*25-2403	*25-2417	*25-2430	*25-2444	*26-2482	*28-2503	*28-2516		
		*29-2529	*31-2542	*31-2556	*34-2573	*34-2586	*34-2600	*35-2651	*35-2667	*35-2683		
		*35-2698	*35-2715	*35-2733	*35-2750	*35-2767	*36-2806	*36-2822	*36-2838	*36-2853		
		*36-2870	*36-2888	*36-2905	*36-2922	*37-2953	*37-2980	*37-2993	*37-3005	*37-3016		
		*37-3029	*37-3043	*37-3056	*37-3070	*37-3088	*37-3115	*37-3128	*37-3140	*37-3151		

REFERENCES

SYMBOL	VALUE	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	
		*37-3164	*37-3178	*37-3191	*37-3205	*37-3235	*37-3249	*37-3259	*37-3269	*38-3280
		*38-3289	*38-3319	*38-3332	*38-3344	*38-3357	*38-3369	*38-3385	*38-3398	*38-3410
		*38-3423	*38-3435	*38-3486	*38-3502	*38-3512	*38-3522	*38-3532	*38-3541	*38-3570
		*38-3580	*38-3590	*38-3600	*38-3610	*38-3619	39-3636	*39-3636	39-3636	39-3636
		39-3637								
SMAIL	001224	9-577	9-577	#10-578	16-1024	16-1025	39-3636	39-3637	40-3743	
SMBADR	000206	#9-577								
SMFLG	041002	*40-3744	40-3744	*40-3744	#40-3744					
SMNEW	040045	39-3722	#39-3722							
SMSGAD	001240	#10-578	*40-3744	40-3744						
SMSGLG	001242	#10-578	*40-3744							
SMSGTY	001224	#10-578	40-3744	*40-3744	40-3744	*40-3744				
SMSWR	040034	39-3722	#39-3722							
SMXCNT	001306	#10-578								
SNULL	001154	#10-578	40-3743	40-3743	40-3743					
SNWTST	= 000001	#17-1071	17-1071	#17-1071	17-1071	#18-1151	18-1151	#18-1151	18-1151	#19-1220
		19-1220	#19-1220	19-1220	#19-1283	19-1283	#19-1283	19-1283	#19-1359	19-1359
		#19-1359	19-1359	#19-1435	19-1435	#19-1435	19-1435	#19-1482	19-1482	#19-1482
		19-1482	#20-1574	20-1574	#20-1574	20-1574	#20-1641	20-1641	#20-1641	20-1641
		#21-1676	21-1676	#21-1676	21-1676	#21-1727	21-1727	#21-1727	21-1727	#22-1805
		22-1805	#22-1805	22-1805	#24-1964	24-1964	#24-1964	24-1964	#24-2144	24-2144
		#24-2144	24-2144	#24-2202	24-2202	#24-2202	24-2202	#24-2261	24-2261	#24-2261
		24-2261	#25-2323	25-2323	#25-2323	25-2323	#26-2474	26-2474	#26-2474	26-2474
		#35-2631	35-2631	#35-2631	35-2631	#36-2785	36-2785	#36-2785	36-2785	#37-2952
		37-2952	#37-2952	37-2952	#37-3087	37-3087	#37-3087	37-3087	#37-3233	37-3233
		#37-3233	37-3233	#38-3314	38-3314	#38-3314	38-3314	#38-3379	38-3379	#38-3379
		38-3379	#38-3452	38-3452	#38-3452	38-3452	#38-3484	38-3484	#38-3484	38-3484
		#38-3568	38-3568	#38-3568	38-3568					
SOCNT	041304	*40-3746	*40-3746	#40-3746						
SOCIVL	041732	41-3750	#41-3750							
SOMODE	041306	*40-3746	*40-3746	40-3746	*40-3746	*40-3746	#40-3746			
SOVER	036312	39-3636	39-3636	39-3636	#39-3636					
SPASS	001232	#10-578	*16-1024	*39-3635	*39-3635	39-3635	39-3635	39-3635	39-3726	
SPASTM	000212	#9-577								
SPWRAD	042176	#42-3753								
SPWRDN	042036	16-1024	#42-3753	42-3753						
SPWRMG	042172	#42-3753								
SPWRUP	042110	42-3753	#42-3753							
SQUES	001220	#10-578	39-3637	39-3637	39-3722	39-3722	39-3722	39-3722	40-3743	40-3743
SRDCHR	037414	#39-3722	42-3752	42-3752						
SRDEC	= *****	42-3752								
SRDLIN	037534	#39-3722	42-3752	42-3752						
SRDOCT	= *****	42-3752								
SRDSZ	- 000010	#39-3722	39-3722							
SREGAD	001160	#10-578								
SREGO	001162	#10-578	*39-3637	45-3842	45-3843	45-3844	45-3845	45-3850	45-3852	45-3856
		45-3858	45-3859							
SREG1	001164	#10-578	*39-3637	45-3846	45-3849	45-3850	45-3851	45-3852	45-3854	45-3855
		45-3858	45-3859							
SREG2	001166	#10-578	*39-3637	45-3839	45-3840	45-3841	45-3844	45-3849	45-3854	45-3859
SREG3	001170	#10-578	*39-3637	45-3840	45-3845	45-3851	45-3854			
SREG4	001172	#10-578	*39-3637	45-3841	45-3842	45-3843	45-3844	45-3845		

CKKTBAO		CREATED BY MACRO ON 25-SEP-79 AT 12:29			PAGE 14	L 10				
SYMBOL CROSS REFERENCE		REFERENCES			CREF	D9D YZ				
SYMBOL	VALUE	REFERENCES								
\$REGS	001174	#10-578	*39-3637	45-3862						
\$RESRE	041572	#41-3749	42-3752							
\$RTNAD	036126	#39-3635								
\$RTRN	036122	16-1024	*16-1024	*16-1024	39-3635	#39-3635				
\$R2A	= *****	42-3752								
\$SAVRE	041534	#41-3749	42-3752	42-3752						
\$SAVR6	042220	*42-3753	42-3753	*42-3753	*42-3753	#42-3753				
\$SCOPE	036134	16-1024	#39-3636							
\$SETUP	= 000137	#9-574	9-574	#9-574	9-574	#9-574	9-574	#9-574	9-574	#9-574
		9-574	#9-574	9-574	#9-574	16-1024	16-1024	16-1024	16-1024	16-1024
		16-1024	16-1024	16-1024	16-1024	16-1024	16-1024	16-1024	16-1025	16-1025
		16-1025	39-3635	39-3635	39-3636	39-3637	39-3637	39-3637	39-3637	39-3722
		39-3722	42-3753							
\$STUP	= 177777	#9-574	#9-574	9-574	#9-574	#9-574	9-574	#9-574	#9-574	9-574
		#9-574	#9-574	9-574	#9-574	#9-574	9-574	#9-574	#9-574	9-574
\$SVLAD	036256	39-3636	39-3636	#39-3636						
\$SVPC	= 000204	#9-576	9-576							
\$SWR	- 173400	#8-519	8-523	8-524	8-524	8-524	8-524	8-524	8-524	8-524
		8-524	10-578	10-578	10-578	16-1024	16-1024	16-1024	16-1024	16-1024
		17-1071	18-1151	19-1220	19-1283	19-1359	19-1435	19-1482	20-1574	20-1641
		21-1676	21-1727	22-1805	24-1964	24-2144	24-2202	24-2261	25-2323	26-2474
		35-2631	36-2785	37-2952	37-3087	37-3233	38-3314	38-3379	38-3452	38-3484
		38-3508	39-3635	39-3635	39-3635	39-3635	39-3635	39-3636	39-3636	39-3636
		39-3636	39-3636	39-3636	39-3636	39-3636	39-3636	39-3636	39-3636	39-3636
		39-3636	39-3636	39-3636	39-3636	39-3636	39-3636	39-3636	39-3636	39-3637
		39-3637	39-3637	39-3637	39-3637	39-3637	39-3637	39-3637	39-3637	39-3637
		39-3637	42-3753							
\$SWREG	001246	#10-578	16-1024							
\$SWRMK	= 000000	8-524	8-524	8-524	8-524	8-524	8-524	8-524	8-524	8-524
		39-3636	39-3636	39-3636	39-3636	39-3636	39-3636	39-3636	39-3636	39-3636
		39-3636								
\$TBIT	001310	#10-578	*16-1024	*39-3635	39-3635	39-3635	*42-3753			
\$TESTN	001230	#10-578	*39-3636							
\$TKB	001146	#10-578	39-3722	39-3722	39-3722	39-3722	39-3722	39-3722	40-3743	40-3743
\$TKS	001144	#10-578	39-3722	39-3722	39-3722	39-3722	39-3722	39-3722	39-3722	39-3722
		40-3743	40-3743							
\$TMP0	001176	#10-578	*17-1091	17-1121	17-1126	*18-1165	19-1196	19-1201	*19-1509	19-1516
		*21-1752	*21-1758	*21-1762	*21-1765	*21-1769	21-1770	45-3839	45-3840	45-3841
		45-3847	45-3863							
\$TMP1	001200	#10-578	*19-1506	*19-1518	*19-1521	19-1522	*19-1528	*19-1533	*19-1537	19-1538
		*21-1689	*21-1695	*21-1699	21-1700					
\$TMP2	001202	#10-578	*19-1510	19-1519	*25-2405	25-2406	*25-2432	25-2433	*32-2559	33-2561
		*34-2588	34-2589	35-2718	*35-2719	*35-2753	35-2755	36-2873	*36-2874	*36-2908
		36-2910	*37-3031	37-3032	*37-3058	37-3059	*37-3166	37-3167	*37-3193	37-3194
		45-3847								
\$TMP3	001204	#10-578								
\$TMP4	001206	#10-578	*39-3701	39-3703						
\$TMP5	001210	#10-578	*39-3702	*39-3726	*39-3727	39-3733				
\$TN	- 000035	#8-520	8-523	17-1071	17-1071	#17-1071	18-1151	18-1151	#18-1151	19-1220
		19-1220	#19-1220	19-1283	19-1283	#19-1283	19-1359	19-1359	#19-1359	19-1435
		19-1435	#19-1435	19-1482	19-1482	#19-1482	20-1574	20-1574	#20-1574	20-1641
		20-1641	#20-1641	21-1676	21-1676	#21-1676	21-1727	21-1727	#21-1727	22-1805

CKKTBAO
SYMBOL CROSS REFERENCE
SYMBOL VALUE

CREATED BY MACRO ON 25-SEP-79 AT 12:29

PAGE 15
M 10
REF D9D YZ

SEQ 0'29

REFERENCES

		22-1805	#22-1805	24-1964	24-1964	#24-1964	24-2144	24-2144	#24-2144	24-2202
		24-2202	#24-2202	24-2261	24-2261	#24-2261	25-2323	25-2323	#25-2323	25-2446
		26-2474	26-2474	#26-2474	34-2602	35-2631	35-2631	#35-2631	35-2769	36-2785
		36-2785	#36-2785	36-2924	37-2952	37-2952	#37-2952	37-3072	37-3087	37-3087
		#37-3087	37-3207	37-3233	37-3233	#37-3233	38-3292	38-3314	38-3314	#38-3314
		38-3372	38-3379	38-3379	#38-3379	38-3438	38-3452	38-3452	#38-3452	38-3484
		38-3484	#38-3484	38-3546	38-3568	38-3568	#38-3568			
\$TPB	001152	#10-578	40-3743	40-3743	40-3743					
\$TPFLG	001157	#10-578	40-3743	40-3743	40-3743					
\$TPS	001150	#10-578	40-3743	40-3743	40-3743					
\$TRAP	041750	16-1024	#42-3752							
\$TRAP2	041772	#42-3752	42-3752							
\$TRP	= 000015	#42-3752	42-3752	42-3752	42-3752	42-3752	#42-3752	42-3752	42-3752	42-3752
		42-3752	#42-3752	42-3752	42-3752	42-3752	42-3752	#42-3752	42-3752	42-3752
		42-3752	42-3752	#42-3752	42-3752	42-3752	42-3752	42-3752	#42-3752	42-3752
		42-3752	42-3752	42-3752	#42-3752	42-3752	42-3752	42-3752	42-3752	#42-3752
		42-3752	42-3752	42-3752	42-3752	#42-3752	42-3752	42-3752	42-3752	42-3752
		#42-3752	42-3752	42-3752	42-3752	42-3752	#42-3752	42-3752	42-3752	42-3752
		42-3752	#42-3752	42-3752	42-3752	42-3752	42-3752	#42-3752	42-3752	42-3752
\$TRPAD	042004	42-3752	#42-3752							
\$STSM	000210	#9-577								
\$STSTM	001102	#10-578	*39-3635	39-3636	39-3636	*39-3636	39-3636	39-3636	39-3636	39-3636
		39-3637	39-3637	39-3637	39-3637	39-3729				
\$TTYIN	040012	39-3722	39-3722	39-3722	39-3722	39-3722	#39-3722			
\$TYPBN	041006	#40-3745	42-3752	42-3752						
\$TYPDS	041310	#41-3748	42-3752	42-3752						
\$TYPE	040206	#40-3743	40-3744	42-3752	42-3752					
\$TYPEC	040420	39-3722	40-3743	40-3743	40-3743	#40-3743	40-3743			
\$TYPEX	040536	40-3743	40-3743	#40-3743						
\$TYPOC	041106	#40-3746	42-3752	42-3752						
\$TYPON	041122	40-3746	#40-3746	42-3752						
\$TYPOS	041062	#40-3746	42-3752							
\$UNIT	001236	#10-578								
\$UNITM	000214	#9-577								
\$USWR	001250	#10-578								
\$XTSTR	036146	#39-3636								
\$GET4	= 000001	#39-3635	#39-3635	39-3635						
\$OF!LL	041305	*40-3746	*40-3746	40-3746	#40-3746					
\$LOCAT	= *****	39-3636	39-3637							
.\$ASTA	*****	40-3744	40-3744							
.\$X	000204	#9-577	9-577							

MACRO NAME	REFERENCES									
COMMEN	#8-525									
ENDCOM	#8-525									
ESCAPE	#8-525									
GETPRI	#8-525	39-3635								
GETSWR	#8-525	#16-1025	16-1025							
MSG30	#17-1063	#17-1071								
MSG31	#17-1142	#18-1151								
MSG31A	#19-1210	#19-1220								
MSG32	#19-1271	#19-1283								
MSG33	#19-1347	#19-1359								
MSG34	#19-1421	19-1435								
MSG35	#19-1468	#19-1482								
MSG36	#19-1559	20-1574								
MSG36A	#20-1635	#20-1641								
MSG37	#20-1664	21-1676								
MSG40	#21-1712	21-1727								
MSG40A	#24-2253	#24-2261								
MSG40B	#22-1790	#22-1805								
MSG40C	#24-1954	24-1964								
MSG40D	#24-2136	#24-2144								
MSG40E	#24-2194	#24-2202								
MSG41	#25-2312	#25-2323								
MSG41A	#25-2462	26-2474								
MSG42	#35-2620	#35-2631	#36-2785							
MSG43	#36-2940	#37-2952	#37-3087							
MSG43A	#37-3224	37-3233								
MSG44	#38-3305	#38-3314	#38-3379							
MSG45	#38-3444	#38-3452								
MSG46	#38-3475	38-3484								
MSG47	#38-3557	#38-3568								
MULT	#8-525									
NEWTST	#8-525	17-1071	18-1151	19-1220	19-1283	19-1359	19-1435	19-1482	20-1574	20-1641
		21-1676	21-1727	22-1805	24-1964	24-2144	24-2202	24-2261	25-2323	26-2474
		36-2785	37-2952	37-3087	37-3233	38-3314	38-3379	38-3452	38-3484	38-3568
POP	#8-525	40-3744	40-3744	41-3748	41-3749	42-3753	42-3753			
PUSH	#8-525	#40-3744	#40-3744	#40-3744	#41-3748	#41-3749	#42-3753	#42-3753		
REPORT	#8-525									
SAVR	#8-502	39-3637								
SETPRI	#8-525									
SETTRA	#42-3752	42-3752	42-3752	42-3752	42-3752	42-3752	42-3752	42-3752	42-3752	42-3752
		42-3752	42-3752	42-3752						
SETUP	#8-525	#16-1024								
SKIP	#8-525	25-2446	34-2602	35-2769	36-2924	37-3072	37-3207	38-3292	38-3372	38-3438
		38-3546								
SLASH	#8-525									
SPACE	#8-499	#8-525								
STARS	#8-525	#9-576	#9-577	#9-577	#9-577	#10-578	#10-578	#10-578	#12-836	#12-841
	#13-881	#13-892	#14-923	#14-930	#14-942	#14-948	#15-963	#15-970	#15-991	#15-998
	#17-1051	#17-1061	#17-1071	#17-1071	#18-1151	#18-1151	#19-1220	#19-1220	#19-1253	#19-1268
	#19-1283	#19-1283	#19-1359	#19-1359	#19-1435	#19-1435	#19-1482	#19-1482	#20-1574	#20-1574
	#20-1641	#20-1641	#21-1676	#21-1676	#21-1727	#21-1727	#22-1805	#22-1805	#24-1964	#24-1964
	#24-2144	#24-2144	#24-2202	#24-2202	#24-2261	#24-2261	#25-2304	#25-2309	#25-2323	#25-2323

CKKTBAO CREATED BY MACRO ON 25-SEP-79 AT 12:29

PAGE 18

CREF D9D YZ

SEQ 0132

MACRO CROSS REFERENCE

MACRO NAME	REFERENCES
.STYPE	#8-515 40-3743
.STYPC	#8-516 40-3746