

.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-F609B-MC
PRODUCT NAME: CKKTABO 11/44 MEM MGMT PRT A
DATE: APRIL, 1981
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: DAN P. MILLEVILLE

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, 1981 BY DIGITAL EQUIPMENT CORPORATION

43
44
45
46
47
48
49
50
51
52

PROGRAM HISTORY

<u>DATE</u>	<u>REVISION</u>	<u>REASON FOR REVISION</u>
OCTOBER, 1979	A	FIRST RELEASE
APRIL, 1981	B	CORRECT TRACE TRAP COMMENTS, USE NEW SYSMAC CONTAINING CHECK FOR BIT 0 OF CPU ERROR REGIS- TER AND ^Q CHECK IN INPUT ROUTINE.

53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86

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
3.4	POWER MONITOR BIT ERRORS
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

87
88
89
90
91
92
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

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:

CKKAABO 1/44 CPU/EIS
CKKABCO 11/44 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.

139
 140
 141
 142
 143
 144
 145
 146
 147
 148
 149
 150
 151
 152
 153
 154
 155
 156
 157
 158
 159
 160
 161
 162
 163
 164
 165
 166
 167
 168

2.0 OPERATING INSTRUCTIONS

2.1 LOADING PROCEDURES

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

2.2 STARTING PROCEDURES

THE PROGRAM IS STARTED BY LOADING ADDRESS 200 AND STARTING. 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.

169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216

2.3 CONTROL SWITCH SETTINGS

<u>SWITCH</u>	<u>OCTAL VALUE</u>	<u>USE</u>
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 ODD PASS STARTING WITH THE THIRD PASS.
SW10	002000	BELL ON ERROR THIS SWITCH WHEN SET WILL RING THE CONSOLE TERMINAL BELL WHEN AN ERROR HAS BEEN DETECTED.
SW9	001000	LOOP ON ERROR THIS SWITCH WHEN SET WILL CAUSE THE PROGRAM TO LOOP ON THE FIRST FAILURE WHICH IS ENCOUNTERED EVEN IF THE FAILURE IS INTERMITTANT
SW8	000400	LOOP ON TEST IN SWR<7:0> THIS SWITCH WHEN SET WILL CAUSE THE PROGRAM TO LOOP ON THE TEST WHOSE TEST NUMBER IS SET IN BITS 7-0 OF THE SWITCH REG.

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

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
----- 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 8 SECONDS WITH CACHE.

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

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.

315
316
317
318
319
320
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

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.

3.4 POWER MONITOR BIT ERRORS

THE NEW \$ERROR AND \$SCOPE ROUTINES USED CONTAIN A CHECK OF BIT 0 (POWER MONITOR BIT) OF THE CPU ERROR REGISTER. IF THE BIT SHOULD BECOME SET, AN ERROR CALL WILL BE CALLED FROM THE \$SCOPE ROUTINE. THE TEST NUMBER IN THE ERROR PRINTOUT WILL BE THE TEST *BEFORE* THE SCOPE CALL THE SET BIT WAS FOUND. IF THE BIT BECOMES SET AFTER THE SCOPE AND DURING THE TEST, AND AN ERROR SHOULD BE CALLED FOR WHATEVER REASON, *TWO* ERRORS WILL BE CALLED. THE FIRST ERROR WILL BE THE POWER MONITOR BIT ERROR, AND THE SECOND WILL BE THE ERROR ORIGINALLY CALLED. THE ERROR WAS CALLED, LOOP-ON-ERROR IS DISABLED FOR THIS ERROR ONLY. IF THIS ERROR SHOULD APPEAR, IT IS SUGGESTED THAT THE PROBLEM CAUSING THE BIT TO BE SET BE CORRECTED BEFORE USING THE RESULTS OF THIS DIAGNOSTIC AS CONCLUSIVE.

354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391

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
IS PRINTED. FOR EXAMPLE:

END OF PASS #2

THAT WOULD INDICATE THAT PASS TWO WAS JUST COMPLETED
AND NO ERRORS WERE DETECTED DURING THAT PASS.

TOTAL NUMBER OF ERRORS SINCE THE LAST END-OF-PASS ARE
REPORTED ONLY IF THERE ARE ANY IN THE END-OF-PASS MESSAGE
AS FOLLOWS:

END OF PASS #2 TOTAL ERRORS SINCE LAST REPORT 2

THAT WOULD INDICATE THAT PASS TWO WAS JUST COMPLETED
WITH TWO ERRORS 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).

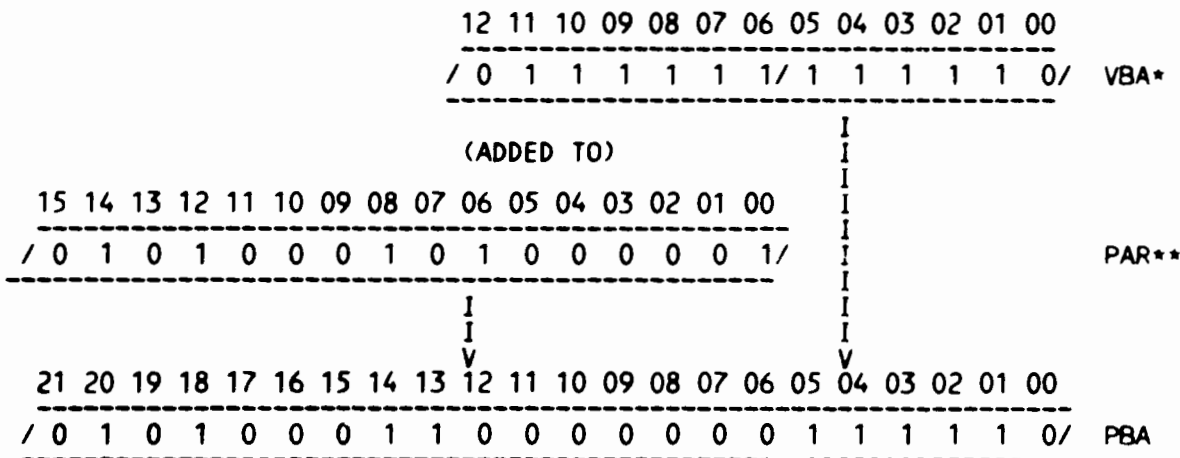
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433

4.4 POWER FAILURE HANDLING

IF A POWER FAIL OCCURS (FOLLOWED BY A POWER UP), THE MESSAGE 'POWER FAILURE-RESTARTING' IS TYPED OUT AND THE PROGRAM WILL RESTART EXECUTION AT 'START:' (THE VERY BEGINNING OF THE PROGRAM. IF THE SOFTWARE SWITCH REGISTER IS BEING USED, ITS CONTENTS WILL BE RESTORED.

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

434
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

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

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

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

464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498

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'.

a

1196
1197

```
.TITLE CKKTABO 11/44 MEM MGMT PRT A
;*COPYRIGHT (C) 1981
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MASS. 01754
;*
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQAC-C5), JAN, 1981.
;*

```

1198

```
.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>

```

1199

001100
104000
000004

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

```

000011
000012
000015
000200
177776
177776
177774
177772
177570
177570

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

```

000000
000001
000002
000003
000004
000005
000006
000007
000006
000007

```
;*GENERAL PURPOSE REGISTER DEFINITIONS
R0= 0           ;;GENERAL REGISTER
R1= 1           ;;GENERAL REGISTER
R2= 2           ;;GENERAL REGISTER
R3= 3           ;;GENERAL REGISTER
R4= 4           ;;GENERAL REGISTER
R5= 5           ;;GENERAL REGISTER
R6= 6           ;;GENERAL REGISTER
R7= 7           ;;GENERAL REGISTER
SP= 6           ;;STACK POINTER
PC= 7           ;;PROGRAM COUNTER

```

000000
000040
000100
000140
000200
000240
000300
000340

```
;*PRIORITY LEVEL DEFINITIONS
PR0= 0          ;;PRIORITY LEVEL 0
PR1= 40         ;;PRIORITY LEVEL 1
PR2= 100        ;;PRIORITY LEVEL 2
PR3= 140        ;;PRIORITY LEVEL 3
PR4= 200        ;;PRIORITY LEVEL 4
PR5= 240        ;;PRIORITY LEVEL 5
PR6= 300        ;;PRIORITY LEVEL 6
PR7= 340        ;;PRIORITY LEVEL 7

```

BASIC DEFINITIONS

```

100000          ;*'SWITCH REGISTER' SWITCH DEFINITIONS
040000          SW15= 100000
020000          SW14= 40000
010000          SW13= 20000
004000          SW12= 10000
002000          SW11= 4000
001000          SW10= 2000
000400          SW09= 1000
000200          SW08= 400
000100          SW07= 200
000040          SW06= 100
000020          SW05= 40
000010          SW04= 20
000004          SW03= 10
000002          SW02= 4
000001          SW01= 2
001000          SW00= 1
000400          SW9=SW09
000200          SW8=SW08
000100          SW7=SW07
000040          SW6=SW06
000020          SW5=SW05
000010          SW4=SW04
000004          SW3=SW03
000002          SW2=SW02
000001          SW1=SW01
000001          SW0=SW00

100000          ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
040000          BIT15= 100000
020000          BIT14= 40000
010000          BIT13= 20000
004000          BIT12= 10000
002000          BIT11= 4000
001000          BIT10= 2000
000400          BIT09= 1000
000200          BIT08= 400
000100          BIT07= 200
000040          BIT06= 100
000020          BIT05= 40
000010          BIT04= 20
000004          BIT03= 10
000002          BIT02= 4
000001          BIT01= 2
001000          BIT00= 1
000400          BIT9=BIT09
000200          BIT8=BIT08
000100          BIT7=BIT07
000040          BIT6=BIT06
000020          BIT5=BIT05
000010          BIT4=BIT04
000004          BIT3=BIT03
000002          BIT2=BIT02
000001          BIT1=BIT01
000001          BIT0=BIT00

000004          ;*BASIC 'CPU' TRAP VECTOR ADDRESSES
000010          ERRVEC= 4          ;;TIME OUT AND OTHER ERRORS
                RESVEC= 10        ;;RESERVED AND ILLEGAL INSTRUCTIONS

```

1200

```

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
1200      .SBTTL MEMORY MANAGEMENT DEFINITIONS
          ;*KT11 VECTOR ADDRESS
          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 REGISTORS
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
:*
MMR0=SR0
MMR1=SR1
MMR2=SR2
MMR3=SR3
KSP=SP
SSP=SP
USP=SP
TBIT=BIT4
WBIT=BIT6
KERSTK=STACK
SUPSTK=STACK-200
USESTK=STACK-300
CPUERR=177766
RMIREG=177770

1201
1202
1203 177572
1204 177574
1205 177576
1206 172516
1207 000006
1208 000006
1209 000006
1210 000020
1211 000100
1212 001100
1213 000700
1214 000600
1215 177766
1216 177770
1271
1273 .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
000174 000174 .=174
000174 000000 DISPREG: .WORD 0 ;;SOFTWARE DISPLAY REGISTER
000176 000000 SWREG: .WORD 0 ;;SOFTWARE SWITCH REGISTER
1274 000200 000137 020000 .SBTTL STARTING ADDRESS(ES)
      JMP @*START ;;JUMP TO STARTING ADDRESS OF PROGRAM
      .SBTTL ACT11 HOOKS
      :******
      :HOOKS REQUIRED BY ACT11
      $SVPC=. ;SAVE PC
      .=46
      $ENDAD ;:1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
      .=52
      .WORD 0 ;:2)SET LOC.52 TO ZERO
      .=$SVPC ;; RESTORE PC
1275 .SBTTL APT PARAMETER BLOCK
      :******
      :SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
      :******
      .SX=. ;;SAVE CURRENT LOCATION
      .=24 ;;SET POWER FAIL TO POINT TO START OF PROGRAM
      200 ;;FOR APT START UP
      .=44 ;;POINT TO APT INDIRECT ADDRESS PNTR.
      $APTHDR ;;POINT TO APT HEADER BLOCK

```

```
000204          .=$X    ;;RESET LOCATION COUNTER
                *****
                ;;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                ;;INTERFACE SPEC.
$APTHD:
$HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MADR:  .WORD $MAIL  ;;ADDRESS OF APT MAILBOX (BITS 0-15)
$STMT:  .WORD 10     ;;RUN TIM OF LONGEST TEST
$PASTM: .WORD 20     ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD 5      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
                .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

1276

.SBTTL COMMON TAGS

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

```
001100 001100          .SCMTAG:          ;;START OF COMMON TAGS
001100 000000          .WORD          0
001102 000          $TSTNM: .BYTE          0          ;;CONTAINS THE TEST NUMBER
001103 000          $ERFLG: .BYTE          0          ;;CONTAINS ERROR FLAG
001104 000000          $ICNT:  .WORD          0          ;;CONTAINS SUBTEST ITERATION COUN'
001106 000000          $LPADR: .WORD          0          ;;CONTAINS SCOPE LOOP ADDRESS
001110 000000          $LPERR: .WORD          0          ;;CONTAINS SCOPE RETURN FOR ERRORS
001112 000000          $ERTTL: .WORD          0          ;;CONTAINS TOTAL ERRORS DETECTED
001114 000          $ITEMB: .BYTE          0          ;;CONTAINS ITEM CONTROL BYTE
001115 001          $ERMAX: .BYTE          1          ;;CONTAINS MAX. ERRORS PER TEST
001116 000000          $ERRPC: .WORD          0          ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120 000000          $GDADR: .WORD          0          ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122 000000          $BDADR: .WORD          0          ;;CONTAINS ADDRESS OF 'BAD' DATA
001124 000000          $GDAT:  .WORD          0          ;;CONTAINS 'GOOD' DATA
001126 000000          $BDAT:  .WORD          0          ;;CONTAINS 'BAD' DATA
001130 000000          .WORD          0          ;;RESERVED--NOT TO BE USED
001132 000000          .WORD          0
001134 000          $AUTOB: .BYTE          0          ;;AUTOMATIC MODE INDICATOR
001135 000          $INTAG: .BYTE          0          ;;INTERRUPT MODE INDICATOR
001136 000000          .WORD          0
001140 177570          $SWR:   .WORD          DSWR          ;;ADDRESS OF SWITCH REGISTER
001142 177570          $DISPLAY: .WORD          DDISP          ;;ADDRESS OF DISPLAY REGISTER
001144 177560          $TKS:   177560          ;;TTY KBD STATUS
001146 177562          $TKB:   177562          ;;TTY KBD BUFFER
001150 177564          $TPS:   177564          ;;TTY PRINTER STATUS REG. ADDRESS
001152 177566          $TPB:   177566          ;;TTY PRINTER BUFFER REG. ADDRESS
001154 000          $NULL: .BYTE          0          ;;CONTAINS NULL CHARACTER FOR FILLS
001155 002          $FILLS: .BYTE          2          ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156 012          $FILLC: .BYTE          12          ;;INSERT FILL CHARS. AFTER A 'LINE FEED'
001157 000          $TPFLG: .BYTE          0          ;;'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
001160 000000          $REGAD: .WORD          0          ;;CONTAINS THE ADDRESS FROM
;;WHICH ($REGO) WAS OBTAINED
001162 000006          .REPT          $CM3
001164 000000          $REGO:  .WORD          0          ;;CONTAINS (($REGAD)+0)
001166 000000          $REG1:  .WORD          0          ;;CONTAINS (($REGAD)+2)
001170 000000          $REG2:  .WORD          0          ;;CONTAINS (($REGAD)+4)
001172 000000          $REG3:  .WORD          0          ;;CONTAINS (($REGAD)+6)
001174 000000          $REG4:  .WORD          0          ;;CONTAINS (($REGAD)+10)
001174 000000          $REG5:  .WORD          0          ;;CONTAINS (($REGAD)+12)
001176 000006          .REPT          6
001200 000000          $TMP0:  .WORD          0          ;;USER DEFINED
001202 000000          $TMP1:  .WORD          0          ;;USER DEFINED
001204 000000          $TMP2:  .WORD          0          ;;USER DEFINED
001206 000000          $TMP3:  .WORD          0          ;;USER DEFINED
001210 000000          $TMP4:  .WORD          0          ;;USER DEFINED
001212 000000          $TMP5:  .WORD          0          ;;USER DEFINED
001214 207          377          $ESCAPE: 0          ;;ESCAPE ON ERROR ADDRESS
001220 077          377          $BELL:  .ASCIZ <207><377><377>          ;;CODE FOR BELL
001221 015          000          $QUES:  .ASCII  /?/          ;;QUESTION MARK
001222 012          000          $CRLF:  .ASCII <15>          ;;CARRIAGE RETURN
001222 012          000          $LF:   .ASCIZ <12>          ;;LINE FEED
*****
```



```
.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
```

001372

\$ERRTB:

1277				
1278				
1279	001372	037460	EM1	:UNEXPECTED CPU TRAP TO LOC. 004
1280	001374	043023	DH1	:OLD PC OLD PSW R6 WAS CPUERR TESTNO ERRORPC
1281	001376	045762	DT1	:TRAPPC,TRAPPS,WASR6,(PUERR,TESTNO,\$ERRPC, 0
1282	001400	046534	DF1	:0,0,0,0,0
1283				
1284				
1285	001402	037520	EM2	:UNEXPECTED MEM. MGMT. TRAP TO LOC. 250
1286	001404	043103	DH2	:OLD PC OLD PSW R6 WAS SR0 SR1 SR2 TESTNO ERR
1287	001406	046000	DT2	:TRAPPC,TRAPPS,WASR6,WASSR0,WASSR1,WASSR2,TESTNO,\$ERRPC,0
1288	001410	046541	DF2	:0,0,0,0,0,0,0,0
1289				
1290				
1291	001412	037567	EM3	:PRIORITY BITS SET WRONG IN PSW
1292	001414	043202	DH3	:WROTE READ TESTNO ERRORPC
1293	001416	046022	DT3	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
1294	001420	046551	DF3	:0,0,0,0
1295				
1296				
1297	001422	037626	EM4	:MODE BITS SET WRONG IN PSW
1298	001424	043202	DH3	:WROTE READ TESTNO ERRORPC
1299	001426	046022	DT3	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
1300	001430	046551	DF3	:0,0,0,0
1301				
1302				
1303	001432	037661	EM5	:DUAL ADDRESSING BETWEEN HI&LO BYTES OF PSW
1304	001434	043202	DH3	:WROTE READ TESTNO ERRORPC
1305	001436	046022	DT3	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
1306	001440	046551	DF3	:0,0,0,0
1307				
1308				
1309	001442	037734	EM6	:KERNEL R6 CHANGED BY WRITING SUPERVISOR/USER R6
1310	001444	043202	DH3	:WROTE READ TESTNO ERRORPC
1311	001446	046022	DT3	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
1312	001450	046551	DF3	:0,0,0,0

1313			;*ITEM 7		
1314	001452	040017	EM7		:A MEMORY MGMT. REG. TIMED OUT
1315	001454	043242	DH7		:ADDRESS TESTNO ERRORPC
1316	001456	046034	DT7		:\$REG0,TESTNO,\$ERRPC,0
1317	001460	046555	DF7		:0,0,0
1318					
1319			;*ITEM 10		
1320	001462	040055	EM10		:SUMMARY OF MEM. MGMT. REG. TIMEOUTS
1321	001464	043272	DH10		:REGISTER-ADDRS NUM. OF
1322					:AND-ED OR-ED TIMOUTS TESTNO ERRORPC
1323	001466	046044	DT10		:ADRAND,ADDROR,TONUM,TESTNO,\$ERRPC,0
1324	001470	046560	DF10		:0,0,1,0,0
1325					
1326			;*ITEM 11		
1327	001472	040121	EM11		:MEM. MGMT. REG. WOULD NOT CLEAR
1328	001474	043372	DH11		:REGISTR READ READ-(BINARY)
1329					:ADDRESS (OCTAL) 5432109876543210 TESTNO ERRORPC
1330	001476	046060	DT11		:\$REG0,\$REG1,\$REG1,TESTNO,\$ERRPC,0
1331	001500	046565	DF11		:0,0,2,0,0
1332					
1333			;*ITEM 12		
1334	001502	040161	EM12		:MEM. MGMT. REG. BITS NOT SET CORRECTLY
1335	001504	043512	DH12		:REGISTR WROTE READ READ
1336					:ADDRESS (OCTAL) (OCTAL) (BINARY) TESTNO ERRORPC
1337	001506	046074	DT12		:\$REG0,\$REG1,\$REG2,\$REG2,TESTNO,\$ERRPC,0
1338	001510	046572	DF12		:0,0,0,2,0,0
1339					
1340			;*ITEM 13		
1341	001512	040230	EM13		:SRO EFFECTED BY WRITE TO PSW
1342	001514	043652	DH13		:READ TESTNO ERRORPC
1343	001516	046034	DT7		:\$REG0,TESTNO,\$ERRPC,0
1344	001520	046555	DF7		:0,0,0
1345					
1346			;*ITEM 14		
1347	001522	040265	EM14		:MMR1 DID NOT TRACK PROPERLY
1348	001524	043702	DH14		:EXPECTD (MMR1) TESTNO ERRORPC
1349	001526	046112	DT14		:\$REG3,\$REG1,TESTNO,\$ERRPC,0
1350	001530	046551	DF3		:0,0,0,0
1351					
1352			;*ITEM 15		
1353	001532	040321	EM15		:MMR3 IS HOLDING THE WRONG DATA
1354	001534	043742	DH15		:LOADED (MMR3) TESTNO ERRORPC
1355	001536	046124	DT15		:\$REG2,\$REG1,TESTNO,\$ERRPC,0
1356	001540	046551	DF3		:0,0,0,0
1357					
1358			;*ITEM 16		
1359	001542	040360	EM16		:DUAL ADDRESSING BETWEEN PAR-PDR GROUPS
1360	001544	044002	DH16		:INDEX INDEX PAR-PDR
1361					:EXPECTD RECEIVD ADDRREAD TESTNO ERRORPC
1362	001546	046136	DT16		:\$REG0,\$REG1,\$REG2,TESTNO,\$ERRPC,0
1363	001550	046534	DF1		:0,0,0,0,0
1364					
1365			;*ITEM 17		
1366	001552	040427	EM17		:PHYS. ADDR. FORMED READ WRONG IN MAINT. MODE
1367	001554	044103	DH17		:VIR ADDR KIPAR4 GDDATA BADDATA TESTNO ERRORPC
1368	001556	046152	DT17		:\$REG0,KIPAR4,\$REG1,\$REG2,TESTNO,\$ERRPC,0
1369	001560	046541	DF2		:0,0,0,0,0,0

1377			;*ITEM 21		
1378	001572	040557	EM21		:SR2 NOT TRACKING CORRECTLY
1379	001574	044312	DH21		:SR2 WAS EXPECTD TESTNO ERRORPC
1380	001576	046212	DT21		:WASSR2,\$REG1,TESTNO,\$ERRPC,0
1381	001600	046551	DF3		:0,0,0,0
1382					
1383			;*ITEM 22		
1384	001602	040612	EM22		:WRONG PDR'S REFERENCED WHILE IN RELOCATE MODE
1385	001604	044352	DH22		:PHYSICL PAR 4
1386					:ADDRESS V.B.A. PAR 4 SRO WAS SR2 WAS PSW TESTNO ERRO
1387	001606	046224	DT22		:PBALO,VIRT1,\$REG4,WASSR0,WASSR2,\$TMP0,TESTNO,\$ERRPC,0
1388	001610	046606	DF20		:3,0,0,0,0,0,0
1389					
1390			;*ITEM 23		
1391	001612	040670	EM23		:PAR OR PDR WAS CHANGED BY A RESET
1392	001614	043372	DH11		:REGISTR READ READ-(BINARY)
1393					:ADDRESS (OCTAL) 5432109876543210 TESTNO ERRORPC
1394	001616	046060	DT11		:\$REG0,\$REG1,\$REG1,TESTNO,\$ERRPC,0
1395	001620	046565	DF11		:0,0,2,0,0
1396					
1397			;*ITEM 24		
1398	001622	040726	EM24		:MAINT MODE (SRO<8>) NOT DISABLED BY A RESET
1399	001624	044470	DH24		:TESTNO ERRORPC
1400	001626	046246	DT24		:TESTNO,\$ERRPC,0
1401	001630	046616	DF24		:0,0
1402					
1403			;*ITEM 25		
1404	001632	041004	EM25		:DATA INCORRECT AFTER A MAINT. MODE WRITE
1405	001634	043202	DH3		:WROTE READ TESTNO ERRORPC
1406	001636	046254	DT25		:\$REG1,\$REG2,TESTNO,\$ERRPC,0
1407	001640	046551	DF3		:0,0,0,0
1408					
1409			;*ITEM 26		
1410	001642	041055	EM26		:SOURCE RELOCATED IN MAINT. MODE
1411	001644	043103	DH2		:OLD PC OLD PSW R6 WAS SRO SR2 TFSTNO ERRORPC
1412	001646	046000	DT2		:TRAPPC,TRAPPS,WASR6,WASSR0,WASSR2,TESTNO,\$ERRPC,0
1413	001650	046541	DF2		:0,0,0,0,0,0,0
1414					
1415			;*ITEM 27		
1416	001652	041115	EM27		:SR2 DIDNOT LOCKUP CORRECT VIRTUAL ADDR.
1417	001654	044312	DH21		:SR2 WAS EXPECTD TESTNO ERRORPC
1418	001656	046266	DT27		:WASSR2,\$REG4,TESTNO,\$ERRPC,0
1419	001660	046551	DF3		:0,0,0,0
1420					
1421			;*ITEM 30		
1422	001662	041171	EM30		:FOLLOWING PAR/PDR WILL NOT ZERO
1423	001664	044510	DH30		:ADDRESS DATA TESTNO ERRORPC
1424	001666	046300	DT30		:\$REG0,\$REG2,TESTNO,\$ERRPC,0
1425	001670	046551	DF3		:0,0,0,0
1426					
1427			;*ITEM 31		
1428	001672	041231	EM31		:SUMMARY OF DUAL ADDRESSING ERRORS
1429	001674	044550	DH31		:ADDROR ADDRAND ADDROR ADDRAND
1430					:LOADED LOADED ENABLED ENABLED TESTNO #ERRORS
1431	001676	046312	DT31		:ADDROR,ADRAND,DATAOR,DATAND,TESTNO,ERRCNT,0
1432	001700	046551	DF3		:0,0,0,0,0,1

1489			:*ITEM 43		
1490	002012	041775	EM43	:	18-BIT MAPPING POSSIBLE HOLE AT TOP OF MEMORY
1491	002014	045327	DH43	:	STARTADR TESTNO ERRORPC
1492	002016	046440	DT43	:	\$TMP1,TESTNO,\$ERRPC,0
1493	002020	046636	DF43	:	4,0,0
1494					
1495			:*ITEM 44		
1496	002022	042053	EM44	:	NO TRAP THRU ERRVEC,AT 18-BIT ADDR. 760000
1497	002024	044470	DH24	:	TESTNO ERRORPC
1498	002026	046246	DT24	:	TESTNO,\$ERRPC,0
1499	002030	046616	DF24	:	0,0
1500					
1501			:*ITEM 45		
1502	002032	042126	EM45	:	DIDN'T GET WRAP AROUND TO ADDRESS ZERO
1503	002034	045357	DH45	:	DATA TESTNO ERRORPC
1504	002036	046450	DT45	:	\$REG1,TESTNO,\$ERRPC,0
1505	002040	046555	DF7	:	0,0,0
1506					
1507			:*ITEM 46		
1508	002042	042175	EM46	:	NO TRAP THRU ERRVEC, ON NON-EXISTANT ADDR.
1509	002044	045407	DH46	:	NEADDR TESTNO ERRORPC \
1510	002046	046034	DT7	:	\$REG0,TESTNO,\$ERRC,0
1511	002050	046555	DF7	:	0,0,0
1512					
1513			:*ITEM 47		
1514	002052	042250	EM47	:	PREMATURE END OF MEMORY FOUND
1515	002054	045436	DH47	:	KIPAR4 LASTBK TESTNO ERRORPC
1516	002056	046460	DT47	:	KIPAR4,\$LSTBK,TESTNO,\$ERRPC,0
1517	002060	046541	DF2	:	0,0,0,0
1518					
1519			:*ITEM 50		
1520	002062	042306	EM50	:	22-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM
1521	002064	045164	DH41	:	STARTBK FINISHBK TESTNO ERRORPC
1522	002066	046412	DT41	:	\$TMP1,\$TMP2,TESTNO,\$ERRPC,0
1523	002070	046541	DF2	:	0,0,0,0
1524					
1525			:*ITEM 51		
1526	002072	042367	EM51	:	BAD RELOCATION,CARRY PROPAGATION 22-BIT MAPPING
1527	002074	045225	DH42	:	PATTERN DATA ADDRESS
1528				:	LOADED FETCHED INTENDED TESTNO ERRORPC
1529	002076	046424	DT42	:	\$REG2,\$REG3,\$REG0,TESTNO,\$ERRPC,0
1530	002100	046541	DF2	:	0,0,0,0,0
1531					
1532			:*ITEM 52		
1533	002102	042447	EM52	:	22-BIT MAPPING POSSIBLE HOLE AT TOP OF MEMORY
1534	002104	045327	DH43	:	STARTADR TESTNO ERRORPC
1535	002106	046440	DT43	:	\$TMP1,TESTNO,\$ERRPC,0
1536	002110	046541	DF2	:	0,0,0
1537					
1538			:*ITEM 53		
1539	002112	042525	EM53	:	DID NOT GET UNIBUS ADDRESS
1540	002114	045225	DH42	:	PATTERN DATA ADDRESS
1541				:	LOADED FETCHED INTENDED TESTNO ERRORPC
1542	002116	046424	DT42	:	\$REG2,\$REG3,\$REG0,TESTNO,\$ERRPC,0
1543	002120	046541	DF2	:	0,0,0,0,0

1544			:*ITEM 54		
1545	002122	042560	EM54		:W-BIT DID NOT GET SET IN PDR
1546	002124	045502	DH54		:PDR VIRTUAL
1547					:TESTED ADDRESS TESTNO ERRORPC
1548	002126	046472	DT54		:\$REG5,\$REG3,TESTNO,\$ERRPC,0
1549	002130	046551	DF3		:0,0,0,0
1550					
1551			:*ITEM 55		
1552	002132	042615	EM55		:W-BIT SET IN MORE THAN ONE PDR
1553	002134	045562	DH55		:PDR IN PDR VIRTUAL
1554					:ERROR TESTED ADDRESS TESTNO ERRORPC
1555	002136	046504	DT55		:\$REG0,\$REG5,\$REG3,TESTNO,\$ERRPC,0
1556	002140	046534	DF1		:0,0,0,0,0
1557					
1558			:*ITEM 56		
1559	002142	042654	EM56		:W-BIT NOT CLEARED BY WRITING TO PDR
1560	002144	045662	DH56		:PDR TESTNO ERRORPC
1561	002146	046520	DT56		:\$REG5,TESTNO,\$ERRPC,0
1562	002150	046555	DF7		:0,0,0
1563					
1564			:*ITEM 57		
1565	002152	042720	EM57		:W-BIT GOT SET DURING ODD ADDR. ABORT
1566	002154	045712	DH57		:PDR WAS EXPECTD TESTNO ERRORPC
1567	002156	046124	DT15		:\$REG2,\$REG1,TESTNO,\$ERRPC,0
1568	002160	046551	DF3		:0,0,0,0
1569					
1570			:*ITEM 60		
1571	002162	042765	EM60		:NO APT SPECIAL HARDWARE FOUND
1572	002164	045752	DH60		:ERRORPC
1573	002166	046530	DT60		:\$ERRPC,0
1574	002170	046534	DF1		:0

```

1575 .SBTTL TURN OFF T-BIT AND SAVE CURRENT PSW
1576 :*****
1577 :***** SUBROUTINES UNIQUE TO THIS PROGRAM *****
1578 :*****
1579 :*
1580 :* THIS SUBROUTINE IS USED TO TURN OFF THE TRACE TRAP BIT IN
1581 :* THE PSW IF IT IS ON. THE PROCESSOR STATUS IS SAVED IN
1582 :* 'TBITPS' SO THAT THE PSW CAN BE RESTORED TO ITS PREVIOUS
1583 :* CONDITION WHEN CONDITIONS WARRANT T-BIT TRAPPING.
1584 :*
1585 :*****
1586 002172 033727 177776 000020 TOFF: BIT PSW,#TBIT ;IS THE T-BIT SET IN THE PSW?
1587 002200 001411 BEQ 1$ ;EXIT IF NO
1588 002202 013746 177776 MOV PSW,-(SP) ;PUSH PRESENT PSW ON THE STACK
1589 002206 011637 001272 MOV (SP),TBITPS ;ALSO SAVE IT IN 'TBITPS' FOR
1590 ;RESTORING LATER
1591 002212 042716 000020 BIC #TBIT,(SP) ;CLEAR THE T-BIT (BIT 4) IN THE PSW
1592 002216 012746 002224 MOV #1$,-(SP) ;PUSH PC OF 'RTS' ON STACK
1593 002222 000006 RTT ;'RETURN' TO 1$ WITH T-BIT OFF
1594 002224 000207 1$: RTS PC ;RETURN TO PROGRAM
    
```

```

1595 .SBTTL TURN ON T-BIT AND RESTORE PREVIOUS PSW
1596 :*****
1597 :*
1598 :* THIS SUBROUTINE IS USED TO RESTORE THE PROCESSOR STATUS
1599 :* TO ITS PREVIOUS CONDITION BY RESTORING THE 'T-BIT PSW'
1600 :* SAVED BY THE 'TOFF' SUBROUTINE IN THE 'TBITPS' LOCATION.
1601 :*
1602 :*****
1603 002226 033727 001272 000020 TON: BIT TBITPS,#TBIT ;WAS T-BIT ON IN THE PREVIOUS PSW?
1604 002234 001410 BEQ 1$ ;EXIT IF NO
1605 002236 013746 001272 MOV TBITPS,-(SP) ;PUSH PREVIOUS PSW ON THE STACK
1606 002242 012737 000340 001272 MOV #340,TBITPS ;RESET THE 'TBITPS' LOCATION
1607 002250 012746 002256 MOV #1$,-(SP) ;PUSH PC OF 'RTS' ON STACK
1608 002254 000006 RTT ;'RETURN' TO 1$ WITH T-BIT RESTORED
1609 002256 000207 1$: RTS PC ;RETURN TO PROGRAM
    
```

```

1610 .SBTTL CLEAR 16 PAR'S OR PDR'S STARTING FROM ADDRESS IN R5
1611 :*****
1612 :* THIS ROUTINE CLEARS 16 CONSECUTIVE MEMORY MANAGEMENT
1613 :* REGISTERS STARTING WITH THE REGISTER POINTED TO BY R5.
1614 :* IT SAVES R0 ON THE STACK, AND LOADS A COUNT IN R0,
1615 :* CLEARS THE PAR'S OR PDR'S, AND THEN RESTORES R0
1616 :* BEFORE RETURNING.
1617 :*
1618 :* CALL: MOV #KIPAR0,R5 ;PUT ADDRESS OF FIRST KERNAL PAR INTO R5
1619 :* JSR PC,CLRREG ;CLEAR ALL THE KERNAL PAR'S
1620 :*****
1621
1622 002260 010046 CLRREG: MOV R0,-(KSP) ;SAVE R0 ON THE STACK
1623 002262 012700 000020 MOV #20,R0 ;PUT COUNT IN R0
1624 002266 005025 1$: CLR (R5)+ ;CLEAR PAR OR PDR POINTED TO BY R5
1625 002270 077002 SOB R0,1$ ;BRANCH BACK 15 DECIMAL TIMES
1626 002272 012600 MOV (KSP)+,R0 ;POP R0 FROM STACK
1627 002274 000207 RTS PC ;RETURN TO TEST
    
```



```
1628 .SBTTL CLEANUP LOCATIONS THAT HOLD LOGICAL 'AND' AND 'OR'  
1629 :*****  
1630 :* THIS SUBROUTINE IS USED TO INITIALIZE ALL LOCATIONS THAT  
1631 :* HOLD THE 'LOGICAL AND' AND 'LOGICAL OR' OF THE DATA AND  
1632 :* ADDRESSES THAT FAILED DURING THE EXECUTION OF A TEST.  
1633 :*****  
1634 :  
1635 CLEANUP:  
1636 CLR DATAOR ;LOCATION OF LOGICAL OR OF BAD DATA  
1637 CLR ADDROR ;LOCATION OF LOGICAL OR OF ADDRESS  
1638 CLR PATTOR ;LOCATION OF LOGICAL OR OF PATTERN LOADED  
1639 MOV #-1,R0 ;LOAD -1 INTO R0 TO INITIALIZE  
1640 MOV R0,DATAND ;LOCATION OF LOGICAL AND OF BAD DATA  
1641 MOV R0,ADRAND ;LOCATION OF LOGICAL AND OF ADDRESS  
1642 MOV R0,PATAND ;LOCATION OF LOGICAL AND OF PATTERN LOADED  
1643 RTS PC ;RETURN TO TEST
```

```

1644 .SBTTL DUAL ADDRESSING WHEN LOADING A PAR OR PDR
1645 :*****
1646 :* THIS SUBROUTINE WILL LOG AND REPORT ALL DUAL ADDRESSING ERRORS
1647 :* FOUND IN BOTH PAR'S AND PDR'S. A 'LOGICAL OR' AND A 'LOGICAL
1648 :* AND' OF THE WRITTEN ADDRESSES WILL BE MAINTAINED IN 'ADDROR',
1649 :* AND 'ADRAND'. THE LOG ON THE ADDITIONAL OR FAILING ADDRESSES
1650 :* WILL BE MAINTAINED IN 'DATAOR' AND 'DATAND'.
1651 :*****
1652
1653 DUALADR:
1654 002334 012637 001326 MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF LOOP
1655 002340 050037 001316 BIS R0,ADDROR ;LOGICAL OR OF WRITTEN ADDRESS
1656 002344 030037 001320 BIT R0,ADRAND ;LOGICAL AND OF WRITTEN ADDRESS
1657 002350 050137 001306 BIS R1,DATAOR ;LOGICAL OR OF DUALED DATA
1658 002354 030137 001310 BIT R1,DATAND ;LOGICAL AND OF DUALED DATA
1659 002360 105737 001103 TSTB $ERFLG ;SEE IF THIS IS FIRST ERROR
1660 002364 001002 BNE 1$ ;BRANCH IF NOT FIRST ERROR
1661 002366 104034 ERROR +34
1662 002370 000401 BR 2$ ;BRANCH TO EXIT
1663 002372 104035 1$: ERROR +35
1664 002374 000177 176726 2$: JMP @OLDPC ;RETURN TO TEST
    
```

```

1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675 002400
1676 002400 012637 001326
1677 002404 050037 001316
1678 002410 030037 001320
1679 002414 050137 001312
1680 002420 030137 001314
1681 002424 050237 001306
1682 002430 030237 001310
1683 002434 105737 001103
1684 002440 001002
1685 002442 104036
1686 002444 000401
1687 002446 104037
1688 002450 000177 176652
1689

```

```

.SBTTL COUNT PATTERN ERRORS IN PAR'S OR PDR'S
:*****
:* THIS SUBROUTINE IS USED TO LOG AND REPORT THE COUNT PATTERN
:* ERRORS OCCURRING WHEN TESTING THE PAR'S AND PDR'S. THE
:* 'LOGICAL OR' AND 'LOGICAL AND' OF VARIOUS DATA WILL BE
:* MAINTAINED AS FOLLOWS:
:* 1. ADDRESSES OF FAILED REGISTERS IN 'ADDROR' AND 'ADRAND'
:* 2. DATA FETCHED FROM REGISTERS IN 'DATAOR' AND 'DATAND'
:* 3. PATTERN LOADED INTO REGISTERS IN 'PATTOR' AND 'PATAND'.
:*****
PARCOUNT:
MOV (KSP)+,OLDPCL ;SAVE RETURN ADDRESS IN CASE OF LOOP
BIS R0,ADDROR ;LOGICAL OR OF FAILING ADDRESS
BIT R0,ADRAND ;LOGICAL AND OF FAILING ADDRESS
BIS R1,PATTOR ;LOGICAL OR OF PATTERN LOADED
BIT R1,PATAND ;LOGICAL AND OF PATTERN LOADED
BIS R2,DATAOR ;LOGICAL OR OF DATA FETCHED
BIT R2,DATAND ;LOGICAL AND OF DATA FETCHED
TSTB $ERFLG ;SEE IF THIS IS THE FIRST ERROR
BNE 1$ ;BRANCH IF NOT FIRST ERROR
ERROR +36
BR 2$ ;BRANCH TO EXIT
1$: ERROR +37
2$: JMP @OLDPCL ;RETURN TO TEST
; *CALL:
; * JSR PC,$SIZE
; * RETURN
; * $LSTAD WILL CONTAIN:
; * WITH KT11 OPTION -- LAST VIRTUAL ADDRESS OF THE LAST BANK
; * WITHOUT KT11 OPTION -- LAST ABSOLUTE ADDRESS OF AVAILABLE MEMORY
; * $LSTBK WILL CONTAIN THE LAST BANK AS A SAF
; * BIT07 = 0 DON'T USE MEMORY MANAGEMENT
; * MUST BE SETUP BEFORE TH CALL
; * BIT15 = 0 DON'T HAVE MEMORY MANAGEMENT OPTION
; * DETERMINED BY ROUTINE
; * $LSTAD WILL CONTAIN THE LAST AVAILABLE MEMORY LOCATION
$SIZE: MOV R0,-(SP) ;;SAVE R0 ON THE STACK
MOV R1,-(SP) ;;SAVE R1 ON THE STACK
MOV R2,-(SP) ;;SAVE R2 ON THE STACK
MOV R3,-(SP) ;;SAVE R3 ON THE STACK
MOV ERRVEC,-(SP) ;;SAVE PRESENT ERROR VECTOR PS & PC
MOV ERRVEC+2,-(SP)
MOV SP,R0 ;;SAVE THE STACK POINTER
;;SET THE ERRVEC PS TO THE PRESENT PS
TRAP ;;PUSH OLD PSW AND PC ON STACK
MOV (SP)+,ERRVEC+2 ;;SAVE THE PSW IN ERRVEC+2
MOV #3776,R1 ;;SETUP ADDRESS
TSTB (PC)+ ;;USE MEMORY MANAGEMENT?
$KT11: .WORD 200 ;;SET TO USE MEMORY MANAGEMENT
BPL $SCORE ;;BR IF NO
MOV # $KTNEX,ERRVEC ;;SET FOR TIMEOUT
TST SR0 ;;KT11 ARE YOU THERE?
BIS #100000,$KT11 ;;YES--SET KT11 KEY
MOV #MGMERR,MMVEC ;;SET IN CASE OF ERROR
MOV #340,MMVEC+2
CLR -(SP) ;;INITIALIZE FOR 'PAR' LOADING
MOV #KIPAR0,R2 ;;ADDRESS OF FIRST 'PAR'

```

```

002560 012703 000010          MOV      #^DB,R3          ;;LOAD EIGHT 'PAR.'S' AND EIGHT 'PDR.'S'
002564 012762 077406 177740 1$:  MOV      #77406,-40(R2)  ;;PDR = 4K, UP, READ/WRITE
002572 011622          MOV      (SP),(R2)+      ;;LOAD 'PAR'
002574 062716 000200          ADD      #200,(SP)      ;;UPDATE FOR NEXT 'PAR'
002600 077307          SOB      R3,1$          ;;LOOP UNTIL ALL EIGHT ARE LOADED
002602 012742 177600          MOV      #177600,-(R2)  ;;SETUP KIPAR7 FOR I/O
002606 005042          CLR      -(R2)          ;;SETUP KIPAR6 FOR TESTING.
002610 012737 002626 000004          MOV      #2$,ERRVEC     ;;CATCH TIMEOUT IF NO SR3
002616 012737 000020 172516          MOV      #20,SR3        ;;ENABLE 22 BIT MODE
002624 000401          BR       3$             ;;THIS PDP-11 HAS A SR3 REGISTER
002626 022626          2$:  CMP      (SP)+,(SP)+    ;;CLEAN OFF TE STAC--NO SR3
002630 005237 177572          3$:  INC      SRO          ;;TURN ON MEMORY MANAGEMENT
002634 012737 002660 000004          MOV      #SKTOUT,ERRVEC ;;SET FOR TIME OUT
002642 005737 143776          4$:  TST      143776      ;;TRAP ON NON-EX-MEM
002646 062712 000040          ADD      #40,(R2)       ;;MAKE A 1K STEP
002652 023712 172356          CMP      KIPAR7,(R2)    ;;LAST ONE?
002656 101371          BHI      4$             ;;NO--TRY IT
002660 011202          SKTOUT: MOV (R2),R2      ;;GET LAST BANK+1
002662 005037 177572          CLR      SRO            ;;TURN OFF MEMORY MANAGEMENT
002666 000421          BR       $$SIZEX
002670 042737 100000 002512  SKTNEX: BIC #100000,$KT11  ;;KT11 NON-EXISTENT
002676 012737 002726 000004  SCORE:  MOV #SCROUT,ERRVEC  ;;SET FOR TIMEOUT
002704 005002          CLR      R2             ;;SET UP BANK
002706 062701 004000          1$:  ADD      #4000,R1      ;;INCREMENT BY 1K
002712 062702 000040          ADD      #40,R2         ;;1K STEP
002716 005711          TST      (R1)           ;;TRAP ON TIME OUT
002720 022701 177776          CMP      #177776,R1    ;;LAST ONE
002724 001370          BNE      1$             ;;NO--TRY AGAIN
002726 162701 004000          SCROUT: SUB #4000,R1
002732 162702 000040          $$SIZEX: SUB #40,R2     ;;DROP BACK
002736 012737 002754 000004          MOV      #2$,ERRVEC     ;;SET FOR TIMEOUT
002744 012701 020000          MOV      #20000,R1     ;;FIRST ADDRESS
002750 005721          1$:  TST      (R1)+        ;;TEST AND STEP TO NEXT ADDRESS
002752 000776          BR       1$             ;;TRY ANOTHER
002754 162701 000002          2$:  SUB      #2,R1        ;;DROP BACK
002760 010006          MOV      R0,SP          ;;RESTORE THE STACK
002762 012637 000006          MOV      (SP)+,ERRVEC+2 ;;RESTORE ERROR VECTOR
002766 012637 000004          MOV      (SP)+,ERRVEC
002772 010137 003020          MOV      R1,$LSTAD     ;;LAST ADDRESS
002776 010237 003022          MOV      R2,$LSTBK     ;;LAST BANK
003002 012603          MOV      (SP)+,R3      ;;RESTORE R3
003004 012602          MOV      (SP)+,R2      ;;RESTORE R2
003006 012601          MOV      (SP)+,R1      ;;RESTORE R1
003010 012600          MOV      (SP)+,R0      ;;RESTORE R0
003012 005037 177766          CLR      CPUERR        ;;CLEAR THE ERROR REGISTER
003016 000207          RTS      PC
003020 000000          $LSTAD: .WORD 0        ;;CONTAINS THE LAST ADDRESS
003022 000000          $LSTBK: .WORD 0        ;;CONTAINS THE LAST BANK
    
```

1691	003024	005737	001206		TIMTST: TST	\$TMP4		:SEE IF THIS IS AN ERROR RETURN
1692	003030	001405			BEQ	1\$:BRANCH TO BEGIN IF NOT
1693	003032	022737	000001	001206	CMP	#1,\$TMP4		:SEE IF THIS IS THE 1ST ERROR CALL RETURN
1694	003040	001453			BEQ	5\$:GO CONTINUE AFTER 1ST ERROR CALL IF SO
1695	003042	000464			BR	6\$:GO CONTINUE AFTER 2ND ERROR CALL IF NOT
1696	003044	012737	003106	001110	1\$: MOV	#2\$,\$LPERR		:SET LOOP ON ERROR POINTER TO 2\$
1697	003052	012737	003146	000004	MOV	#4\$,4		:SET TIMEOUT VECTOR TO 4\$
1698	003060	017600	000000		MOV	@0(SP),R0		:LOAD R0 WITH ADDRESS OF FIRST REG
1699	003064	012701	000020		MOV	#20,R1		:LOAD R1 WITH LOOP COUNT (16)
1700	003070	012737	177777	001320	MOV	#-1,ADRAND		:INITIALIZE 'AND' OF ADDR. LOC
1701	003076	005037	001316		CLR	ADDROR		:INITIALIZE 'OR' OF ADR. LOC.
1702	003102	005037	001274		CLR	TONUM		:INITIALIZE 'TIMEOUTS' COUNTER
1703	003106	005710			2\$: TST	(R0)		:TRY ADDRESSING A REGISTER, TIMEOUTS TO 5\$
1704	003110	005720			3\$: TST	(R0)+		:PUT NEXT REGISTER ADDRESS IN R0
1705	003112	077103			SOB	R1,2\$:LOOP BACK TO 2\$ UNTIL ALL TESTED
1706	003114	012737	003024	001110	MOV	#TIMTST,\$LPERR		:RESET LOOP ON ERROR POINTER TO TIMTST
1707	003122	005737	001274		TST	TONUM		:DID ANY OF THE REGISTERS TIME OUT?
1708	003126	001433			BEQ	7\$:BRANCH IF NOT
1709	003130	011646			MOV	(SP),-(SP)		:MOVE AN EXTRA RETURN PC ONTO STACK FOR POSSIBLE ERROR LOOP
1710	003132	062716	000006		ADD	#6,(SP)		:FUDGE RETURN TO CALL 2ND ERROR
1711	003136	012737	000001	001206	MOV	#1,\$TMP4		:MOVE 1ST ERROR FLAG TO \$TMP4
1712	003144	000207			RTS	PC		:RETURN TO CALL ERROR
1713	003146	062706	000004		4\$: ADD	#4,\$KSP		:CLEAN UP THE STACK
1714	003152	012737	000002	001206	MOV	#2,\$TMP4		:MOVE 2ND ERROR FLAG TO \$TMP4
1715	003160	011646			MOV	(SP),-(SP)		:MOVE AN EXTRA RETURN PC ONTO STACK FOR POSSIBLE ERROR LOOP
1716	003162	062716	000002		ADD	#2,(SP)		:FUDGE RETURN FOR 1ST ERROR CALL IN TEST
1717	003166	000207			RTS	PC		:RETURN TO CALL 2ND ERROR
1718	003170	005726			5\$: TST	(SP)+		:POP EXCESS RETURN OFF STACK - NO LOOP
1719	003172	010002			MOV	R0,R2		:LOAD THE ADDRESS THAT TIMED OUT INTO R2
1720	003174	050237	001316		BIS	R2,ADDROR		: 'OR' IT WITH OTHER ADDRS. THAT TIMED OUT
1721	003200	005102			COM	R2		: 'AND' IT WITH OTHER ADDRS. THAT TIMED OUT
1722	003202	040237	001320		BIC	R2,ADRAND		
1723	003206	005237	001274		INC	TONUM		:INCREMENT THE TIMEOUT COUNTER
1724	003212	000736			BR	3\$:BRANCH BACK TO TEST THE NEXT REGISTER
1725	003214	005726			6\$: TST	(SP)+		:POP EXCESS RETURN OFF STACK - NO LOOP
1726	003216	012737	003232	000004	7\$: MOV	#TIMERR,4		:RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
1727	003224	062716	000010		ADD	#10,(SP)		:FUDGE RETURN OVER BOTH ERROR CALLS
1728	003230	000207			RTS	PC		:RETURN TO TEST

```

1729      .SBTTL CPU TRAP HANDLER ROUTINE
1730      :*****
1731      :***** TRAP HANDLING ROUTINES *****
1732      :*****
1733      :*
1734      :*   THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS THRU
1735      :*   'ERRVEC' (LOC. 004). IF THIS SUBROUTINE IS ENTERED BY A
1736      :*   SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A HALT IS
1737      :*   EXECUTED.
1738      :*
1739      :*****
1740 003232 005227 TIMERR: INC      (PC)+      ;MAKE FLAG ZERO IF FIRST TIME THRU
1741 003234 177777 TIMFLG: .WORD  -1          ;NEGATIVE ONE FOR 'HAVE ENTERED' FLAG
1742 003236 001401      BEQ      1$          ;BRANCH IF FIRST TIME IN
1743 003240 000000      HALT          ;STOP! - I'VE ENTERED THIS ROUTINE
1744      :*
1745      :*   A SECOND TIME BEFORE I FINISHED
1746      :*   REPORTING THE FIRST ERROR. THE
1747      :*   SECOND ENTRY ADDRESS SHOULD BE ON
1748      :*   THE KERNEL STACK.
1748 003242 012637 001260 1$:  MOV      (KSP)+,TRAPPC ;SAVE PC+2 AT TIME OF ABORT
1749 003246 012637 001262      MOV      (KSP)+,TRAPPS ;SAVE PS AT TIME OF ABORT
1750 003252 013737 177766 001332  MOV      CPUERR,PCPUER ;SAVE CPU ERROR REGISTER
1751 003260 010637 001256      MOV      KSP,WASR6   ;SAVE STACK POINTER VALUE
1752 003264 005737 001334      TST      CPUEXP      ;SEE IF ANY ERROR CONDITION WAS EXPECTED
1753 003270 001404      BEQ      2$          ;BRANCH IF NO TRAP EXPECTED
1754 003272 023737 001332 001334  CMP      PCPUER,CPUEXP ;SEE IF EXPECTED CONDITION OCCURED
1755 003300 001401      BEQ      3$          ;BRANCH IF ERROR CODES MATCH
1756 003302 104001 2$:  ERROR  +1          ;UNEXPECTED TRAP OR ABORT TO LOC. 4
1757 003304 005037 177766 3$:  CLR      CPUERR      ;CLEAR CPU ERROR REGISTER
1758 003310 012737 177777 003234  MOV      #-1,TIMFLG   ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
1759 003316 013746 001262      MOV      TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
1760 003322 013746 001260      MOV      TRAPPC,-(KSP)
1761 003326 000000      RTT          ;RETURN FROM INTERRUPT OR ABORT

```

```

1762 .SBT L MEMORY MANAGEMENT TRAP HANDLER ROUTINE
1763 :*****
1764 :
1765 : THIS SUBROUTINE WILL HANDLE ALL UNEXPECTED MEMORY MANAGEMENT
1766 : TRAPS AND ABORTS THRU 'MMVEC' (LOC. 250). IF THIS SUBROUTINE IS
1767 : ENTERED BY A SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A
1768 : HALT IS EXECUTED.
1769 :
1770 :*****
1771 003330 005227 MGMERR: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME THRU
1772 003332 177777 MGMFLG: .WORD -1 ;NEGATIVE ONE FOR 'HAVE ENTERED' FLAG
1773 003334 001401 BEQ 1$ ;BRANCH IF FIRST TIME IN
1774 003336 000000 HALT ;STOP! - I'VE ENTERED THIS ROUTINE
1775 ;A SECOND TIME BEFORE I FINISHED
1776 ;REPORTING THE FIRST ERROR. THE
1777 ;SECOND ENTRY ADDRESS SHOULD BE ON
1778 ;THE KERNEL STACK.
1779 003340 012637 001260 1$: MOV (KSP)+,TRAPPC ;SAVE PC+2 AT TIME OF ABORT
1780 003344 012637 001262 MOV (KSP)+,TRAPPS ;SAVE PS AT TIME OF ABORT
1781 003350 010637 001256 MOV KSP,WASR6 ;SAVE STACK POINTER VALUE
1782 003354 013737 177572 001264 MOV SRO,WASSRO ;SAVE CONTENTS OF KT STATUS REG. 0
1783 003362 013737 177574 001266 MOV SR1,WASSR1 ;SAVE CONTENTS OF KT STATUS REG. 1
1784 003370 013737 177576 001270 MOV SR2,WASSR2 ;SAVE CONTENTS OF KT STATUS REG. 2
1785 003376 042737 160000 177572 BIC #160000,SRO ;CLEAR ERROR BITS IN STATUS REG 0
1786 003404 104002 ERROR +2 ;UNEXPECTED TRAP OR ABORT TO LOC. 250
1787 003406 012737 177777 003332 MOV #-1,MGMFLG ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
1788 003414 013746 001262 MOV TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
1789 003420 013746 001260 MOV TRAPPC,-(KSP)
1790 003424 000006 RTT ;RETURN FROM INTERRUPT OR ABORT.
    
```

CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS

1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807

.SBTTL CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS

```

:*****
:
: THIS SUBROUTINE IS USED TO FORM AN 22-BIT PHYSICAL ADDRESS
: (PBA) FROM THE 16-BIT VIRTUAL ADDRESS (VBA) AND THE APPROPRIATE
: PAGE ADDRESS REGISTER (PAR). THE SAME METHOD USED BY THE MEMORY
: MANAGEMENT LOGIC IS USED. VBA <15:13> SELECTS WHICH PAR/PDR
: IS TO BE USED, VBA <5:0>+PBA <5:0>, AND VBA <12:6> IS ADDED
: TO PAR <15:00> TO GIVE PBA <21:6>. BITS <21:16> OF THE
: PHYSICAL ADDRESS ARE LEFT IN LOC. 'PBAHI' AND BITS <15:00>
: ARE LEFT IN LOC. 'PBALO'. THE PSW'S 'CURRENT MODE' BITS
: ARE USED TO SELECT THE KERNEL,SUPERVISOR OR USER PAR/PDR'S.
: THE ROUTINE IS ENTERED WITH LOC. 'VIRT1' CONTAINING THE 16-BIT
: VIRTUAL ADDRESS.
:*****

```

```

1808 003426 012702 172340 FORMPA: MOV #KIPAR0,R2 ;LOAD ADDRESS OF FIRST KERNEL PAR IN R2
1809 003432 032737 140000 177776 BIT #140000,PSW ;IN USER MODE?
1810 003440 001403 BEQ 1$ ;BRANCH IF NO
1811 003442 012702 177640 MOV #UIPAR0,R2 ;LOAD ADDRESS OF FIRST USER PAR IN R2
1812 003446 000406 BR 2$ ;BRANCH WHEN DONE
1813 003450 032737 040000 177776 1$: BIT #40000,PSW ;IN SUPERVISOR MODE?
1814 003456 001402 BEQ 2$ ;BRANCH IF NO
1815 003460 012702 172240 MOV #SIPAR0,R2 ;LOAD ADDRESS OF FIRST SUPERVISOR PAR IN R2
1816 003464 013700 001276 2$: MOV VIRT1,R0 ;LOAD VIRTUAL ADDR. (VBA) INTO R0
1817 003470 072027 177764 ASH #-14,R0 ;GET BITS <15:13> DOWN TO BITS <3:1>
1818 003474 04270C 177761 BIC #177761,R0 ;MASK OF ALL BITS BUT BITS <3:1>
1819 003500 060002 ADD R0,R2 ;ADD OFFSET TO BASE PAR ADDRESS
1820 003502 011200 MOV (R2),R0 ;GET BITS <15:00> FROM APPROPRIATE PAR
1821 003504 010002 MOV R0,R2 ;COPY PAR BITS <15:00> INTO R2
1822 003506 013737 001276 001302 MOV VIRT1,PBALO ;PUT VIRTUAL ADDR. IN LOC. 'PBALO'
1823 003514 042737 160000 001302 BIC #160000,PBALO ;CLEAR OFF BITS <15:13> OF ORIGINAL VBA
1824 003522 072227 177766 ASH #-12,R2 ;GET PAR <15:00> DOWN TO BITS <1:0> OF R2
1825 003526 042702 177774 BIC #177774,R2 ;CLEAR OFF ALL BITS BUT BITS <1:0>
1826 003532 072027 000006 ASH #6,R0 ;SHIFT PAR<9:0> TO <15:6> OF R0
1827 003536 042700 000077 BIC #77,R0 ;CLEAR BITS <5:0> OF R0
1828 003542 060037 001302 ADD R0,PBALO ;IN EFFECT, ADD VBA<12:0> TO PAR<9:0>
1829 ;(PAR<9:0> IN BITS <15:6> OF R0)
1830 003546 005502 ADC R2 ;ADD ANY CARRY TO R2
1831 003550 010237 001304 MOV R2,PBAHI ;PUT BITS <21:16> OF PHYSICAL ADDR. IN PBAHI
1832 003554 000207 RTS PC ;RETURN TO PROGRAM

```


1833
1834
1835
1836
1837
1838 020000

020000

```

.SBTTL PRE-DIAGNOSTIC SETUP
:*****
:***** STARTING ADDRESS IS 200 *****
:*****
.=20000

START:
.SBTTL INITIALIZE THE COMMON TAGS
::CLEAR THE COMMON TAGS ($CMTAG) AREA
MOV #SCMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
CLR (R6)+ ;;CLEAR MEMORY LOCATION
CMP #SWR,R6 ;;DONE?
BNE -6 ;;LOOP BACK IF NO
MOV #STACK,SP ;;SETUP THE STACK POINTER

::INITIALIZE A FEW VECTORS
MOV #SCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
MOV #340,@IOTVEC+2 ;;LEVEL 7
MOV #ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
MOV #340,@EMTVEC+2 ;;LEVEL 7
MOV #STRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
MOV #340,@TRAPVEC+2;LEVEL 7
MOV #SPWRDN,@PWVVEC ;;POWER FAILURE VECTOR
MOV #340,@PWVVEC+2 ;;LEVEL 7
MOV $ENDCT,$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'.
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$,-(SP) ;;AND PC
RTT ;;TRY THE RTT
020160 012737 000006 033006 64$: MOV #RTT,$RTRN ;;RTT IS LEGAL--SET $RTRN TO A RTT
020166 000402 66$: BR 66$
020170 062706 000010 65$: ADD #10,SP ;;RTT ILLEGAL--CLEAN OFF THE STACK
020174 012737 000012 000010 66$: MOV #RESVEC+2,@RESVEC ;;RESTORE TRAP CATCHER
020202 005037 001346 CLR $TBIT ;;CLEAR 'T' BIT SWITCH
020206 012737 020206 001106 MOV #.,$LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
020214 012737 020214 001110 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.
MOV @ERRVEC,-(SP) ;;SAVE ERROR VECTOR
MOV #67$,@ERRVEC ;;SET UP ERROR VECTOR
MOV #DSWR,$SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
MOV #DDISP,$DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
020250 022777 177777 160662 CMP #-1,$SWR ;;TRY TO REFERENCE HARDWARE SWR
020256 001012 BNE 69$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
;;AND THE HARDWARE SWR IS NOT = -1
020260 000403 BR 68$ ;;BRANCH IF NO TIMEOUT
020262 012716 020270 67$: MOV #68$,(SP) ;;SET UP FOR TRAP RETURN
020266 000002 RTI
020270 012737 000176 001140 68$: MOV #SWREG,$SWR ;;POINT TO SOFTWARE SWR
020276 012737 000174 001142 MOV #DISPREG,$DISPLAY
020304 012637 000004 69$: MOV (SP)+,@ERRVEC ;;RESTORE ERROR VECTOR
020310 005037 001232 CLR $PASS ;;CLEAR PASS COUNT

```

```

020314 132737 000200 001245      BITB  #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
020322 001403                BEQ    70$           ;;YES,USE NON-APT SWITCH
020324 012737 001246 001140      MOV    #$$SWREG,$SWR ;;NO,USE APT SWITCH REGISTER
020332
1839
020332 005227 177777      .SBTTL TYPE PROGRAM NAME
;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
020336 001046      INC    #-1          ;;FIRST TIME?
020340 022737 032742 000042      BNE    71$          ;;BRANCH IF NO
020346 001442      CMP    #$$ENDAD,@#42 ;;ACT-11?
020350 104401 020416      BEQ    71$          ;;BRANCH IF YES
                                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 123727 001244 000001      CMPB   $ENV,#1     ;;ARE WE RUNNING UNDER APT?
020370 001406      BEQ    73$          ;;BRANCH IF YES
020372 023727 001140 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 112737 000001 001134 73$: MOV    #1,$AUTOB   ;;SET AUTO-MODE INDICATOR
020414 000417      74$: BR    71$     ;;GET OVER THE ASCIZ
020454 020454      ;;72$: .ASCIZ <CRLF>#CKKTAB 11/44 MEM MGMT PRT A#<CRLF>
1840 020454      71$: LOOP:
1841 020454 012706 001100      MOV    #STACK,KSP  ;;INITIALIZE THE STACK POINTER
1842 020460 012737 003232 000004      MOV    #TIMERR,ERRVEC ;;LOAD CPU SERVICE ROUTINE INTO TRAP VECTOR
1843 020466 012737 000340 000006      MOV    #340,ERRVEC+2 ;;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
1844 020474 012737 003330 000250      MOV    #MGMERR,MMVEC ;;LOAD MEMORY MANAGENT ROUTINE INTO VECTOR
1845 020502 012737 000340 000252      MOV    #340,MMVEC+2 ;;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
1846 020510 012700 177777      MOV    #-1,R0      ;;PUT -1 INTO R0 TO INITIALIZE FLAGS
1847 020514 010037 003234      MOV    R0,TIMFLG   ;;INITIALIZE CPU ERROR FLAG
1848 020520 010037 003332      MOV    R0,MGMFLG   ;;INITIALIZE MEMORY MANAGEMENT ERROR FLAG
1849 020524 012737 000340 001272      MOV    #340,TBITPS ;;INITIALIZE LOG THAT HOLDS T-BIT PSW
1850 020532 005037 177572      CLR    MMR0        ;;BE SURE MEM. MGMT IS OFF TO START WITH
1851 020536 005037 172516      CLR    MMR3        ;;MAKE SURE ALL MAPPING IS OFF
1852 020542 005037 177766      CLR    CPUERR      ;;MAKE SURE CPU ERROR REG. IS CLEAR
1853 020546 052737 000200 002512      BIS    #BIT7,$KT11 ;;SET M.M. FLAG FOR SIZING ROUTINE
1854 020554 004737 002454      JSR    PC,$SIZE    ;;RUN SIZING ROUTINE
1855 020560 005037 177766      CLR    CPUERR      ;;CLEAR CPU ERROR REG. AFTER SIZING
    
```

1856
1857
1858
1859

.SBTTL ADDRESS AND DATA PATHS TESTS
:*****
: GROUP 1 ADDRESS AND DATA PATHS TESTS
:*****

1866

.SBTTL TEST # 1 - PSW PRIORITY BIT TEST

*TEST 1 PSW PRIORITY BIT TEST

*

THIS TEST READS AND WRITES THE PROCESSOR STATUS WORD <7:5> 'PRIORITY BITS'
TO SEE THAT SOME OF THE BASIC 'DATA PATH' LOGIC IS WORKING.

*

1867	020564	000004		
1868	020574	005000	020576	001110
1869	020576	005001		
1870	020600	010037	177776	
1871	020604	013701	177776	
1872	020610	042701	177437	
1873	020614	020001		
1874	020616	0C1401		
1875	020620	104003		
1876				
1877				
1878				
1879	020622	062700	000040	
1880	020626	022700	000400	
1881	020632	001361		
1882	020634	012737	020566	001110

```

TST1: SCOPE
1$: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
    CLR R0 ;INITIALIZE R0 WITH PRIORITY=0 DATA
2$: CLR R1 ;PREPARE R1 TO ACCEPT DATA READ
    MOV R0, PSW ;WRITE PRIORITY BITS IN THE PSW
    MOV PSW, R1 ;READ BACK THE LOW BYTE OF PSW
    BIC #177437, R1 ;MASK OFF EVERYTHING EXCEPT PRIORITY BITS
    CMP R0, R1 ;WAS CORRECT PRIORITY SET IN THE PSW?
    BEQ 3$ ;BRANCH IF YES
    ERROR +3 ;PRIORITY BITS SET WRONG IN PSW
    ;FOR TIGHTER SCOPE LOOP
    ;REPLACE ERROR CALL WITH
    ;'BR 2$' = 000770
3$: ADD #40, R0 ;CHANGE DATA TO NEXT PRIORITY
    CMP #400, R0 ;HAVE PRIORITIES 0-7 ALL BEEN CHECKED?
    BNF 2$ ;BRANCH IF NO
    MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$

```

1888

```
.SBTTL TEST # 2 - PSW MODE BIT TEST
:*****
:*TEST 2 PSW MODE BIT TEST
:* THIS TEST READS AND WRITES THE PROCESSOR STATUS WORD <15:12> 'MODE BITS'
:* TO FURTHER CHECK THE BASIC CPU DATA PATHS
:*
:*****
```

1889	020642	000004			TST2: SCOPE	
1889	020644	012737	020654	001110	1\$: MOV #2\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 2\$
1890	020652	005000			CLR R0	:INITIALIZE R0 WITH MODE BITS = 0000
1891	020654	005037	177776		2\$: CLR PSW	:INITIALIZE PSW
1892	020660	050037	177776		BIS R0, PSW	:BIT SET THE PSW MODE BITS WITH R0
1893	020664	013701	177776		MOV PSW, R1	:READ BACK THE CONTENTS OF THE PSW
1894	020670	042701	007777		BIC #007777, R1	:MASK OFF EVERYTHING EXCEPT THE MODE BITS
1895	020674	020001			CMP R0, R1	:WERE THE MODE BITS SET CORRECTLY?
1896	020676	001403			BEQ 3\$:BRANCH IF YES
1897	020700	005037	177776		CLR PSW	:CLEAR PSW FOR ERROR REPORT
1898	020704	104004			ERROR +4	:MODE BITS SET WRONG IN PSW
1899						:FOR TIGHTER SCOPE LOOP
1900						:REPLACE ERROR CALL WITH
1901						: 'BR 2\$' = 000763
1902	020706	062700	010000		3\$: ADD #10000, R0	:CHANGE MODE BIT DATA
1903	020712	001360			BNE 2\$:BRANCH IF STILL MORE COMBINATIONS
1904	020714	012737	020644	001110	MOV #1\$, \$LPERR	:RESET LOOP ON ERROR POINTER TO 1\$
1905	020722	005037	177776		CLR PSW	:RESET PSW BEFORE LEAVING

1913

1914	020726	000004		
1915	020730	012737	020736	001110
1916	020736	005037	177776	
1917	020742	012700	000360	
1918	020746	110037	177777	
1919	020752	013701	177776	
1920	020756	042701	007437	
1921	020762	000300		
1922	020764	020001		
1923	020766	001403		
1924	020770	005037	177776	
1925	020774	104005		
1926				
1927				
1928	020776	012737	021004	001110
1929	021004	005037	177776	
1930	021010	012700	000340	
1931	021014	110037	177776	
1932	021020	013701	177776	
1933	021024	042701	007437	
1934	021030	020001		
1935	021032	001403		
1936	021034	005037	177776	
1937	021040	104005		
1938				
1939				
1940				
1941	021042	012737	020730	001110

```

.SBTTL TEST # 3 - BYTE ADDRESSING TEST FOR PSW
:*****
:*TEST 3      BYTE ADDRESSING TEST FOR PSW
:*
:*          THIS TEST WRITES THE HIGH AND LOW BYTES OF THE PROCESSOR STATUS WORD
:*          AND READS THEM BACK TO BE SURE THEY CAN BE WRITTEN INDEPENDENTLY.
:*          THIS CHECKS THE PSW PORTION OF THE ADDRESS DETECTION LOGIC.
:*
:*****
TST3:  SCOPE
1$:   MOV     #2$, $LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2$:   CLR     PSW             ;CLEAR THE PSW
      MOV     #360, R0         ;PUT THE HIGH BYTE DATA INTO R0
      MOVB   R0, PSW+1        ;WRITE THE HIGH BYTE OF THE PSW
      MOV     PSW, R1         ;READ BACK THE ENTIRE PSW
      BIC    #007437, R1      ;MASK OFF THE T & CC BITS
      SWAB   R0              ;GET DATA WRITTEN IN HIGH BYTE OF R0
      CMP    R0, R1          ;WAS THE PSW WRITTEN TO CORRECTLY
      BEQ    3$              ;BRANCH IF YES
      CLR    PSW            ;CLEAR PSW FOR ERROR REPORT
      ERROR  +5              ;LOW BYTE EFFECTED BY WRITE TO HIGH BYTE OF PSW
      ;FOR TIGHTER SCOPE LOOP
      ;REPLACE ERROR CALL WITH
      ;'BR 2$' = 000760
3$:   MOV     #4$, $LPERR      ;SET LOOP ON ERROR POINTER TO 4$
4$:   CLR     PSW             ;CLEAR THE PSW
      MOV     #340, R0         ;PUT THE LOW BYTE DATA INTO R0
      MOVB   R0, PSW         ;WRITE THE LOW BYTE OF THE PSW
      MOV     PSW, R1         ;READ BACK THE ENTIRE PSW
      BIC    #007437, R1      ;MASK OFF THE T&CC BITS
      CMP    R0, R1          ;WAS PSW WRITTEN TO CORRECTLY
      BEQ    5$              ;BRANCH IF YES
      CLR    PSW            ;CLEAR PSW FOR ERROR REPORT
      ERROR  +5              ;HIGH BYTE EFFECTED BY WRITE TO LOW BYTE OF PSW
      ;FOR TIGHTER SCOPE LOOP
      ;REPLACE ERROR CALL WITH
      ;'BR 2$' = 000736
5$:   MOV     #1$, $LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
    
```

1953

.SBTTL TEST # 4 - TEST AND SETUP OF STACK POINTERS

 :*TEST 4 TEST AND SETUP OF STACK POINTERS

 :* THIS TEST SETS THE USER AND KERNEL STACK POINTERS FOR THE
 :* REST OF THE PROGRAM AND MAKES SURE THEY ARE INDEPENDENT OF
 :* EACH OTHER. KERNEL R6 IS SET TO 1100, SUPERVISOR R6 IS SET TO 700,
 :* USER R6 IS SET TO 600, THEN KERNEL R6 IS READ TO BE SURE
 :* IT'S STILL 1100. THE SECOND PART OF THE TEST CHECKS TO SEE
 :* THAT THE ILLEGAL MODE('10') STACK POINTER IS MAPPED TO THE
 :* USER STACK POINTER, WITH MEMORY MANAGEMENT OFF.
 :*

 TST4: SCOPE

1954	021052	005037	177572			CLR	MMRO		:MAKE SURE M.M. IS OFF
1955	021056	005037	177776			CLR	PSW		:GO TO KERNEL MODE
1956	021062	012706	001100			MOV	#KERSTK,KSP		:SET KERNEL STACK POINTER TO 1100
1957	021066	012737	040000	177776		MOV	#40000,PSW		:GO TO SUPERVISOR MODE
1958	021074	012706	000700			MOV	#SUPSTK,SSP		:SET SUPERVISOR STACK POINTER TO 700
1959	021100	012737	140000	177776		MOV	#140000,PSW		:GO TO USER MODE
1960	021106	012706	000600			MOV	#USESTK,USP		:SET USER STACK POINTER TO 600
1961	021112	005037	177776			CLR	PSW		:BACK TO KERNEL MODE
1962	021116	022706	001100			CMP	#KERSTK,KSP		:IS KERNEL R6 STILL 1100?
1963	021122	000404				BR	1\$:BRANCH TO NEXT PART OF TEST
1964	021124	012700	001100			MOV	#KERSTK,R0		:SAVE DATA WRITTEN FOR ERROR REPORT
1965	021130	010601				MOV	KSP,R1		:SAVE DATA READ AFTER USER R6 WAS WRITTEN
1966	021132	104006				ERROR	+6		:KERNEL R6 CHANGED BY WRITING USER R6
1967									:FOR TIGHTER SCOPE LOOP
1968									:REPLACE ERROR CALL WITH
1969									:000756
1970	021134	012737	021134	001110	1\$:	MOV	#1\$,SLPERR		:SET LOOP ON ERROR POINTER TO 1\$
1971	021142	005037	177776			CLR	PSW		:GO TO KERNEL MODE
1972	021146	012746	177777			MOV	#-1,-(KSP)		:PUSH A -1 ONTO STACK
1973	021152	012737	040000	177776		MOV	#40000,PSW		:GO TO SUPERVISOR MODE
1974	021160	012746	000001			MOV	#1,-(SSP)		:PUSH A 1 ONTO THE STACK
1975	021164	012737	140000	177776		MOV	#140000,PSW		:GO TO USER MODE
1976	021172	005046				CLR	-(USP)		:PUSH A ZERO ONTO THE STACK
1977	021174	012737	100000	177776		MOV	#100000,PSW		:PUT ILLEGAL MODE IN PSW
1978	021202	011600				MOV	(SP),R0		:PUT TOP OF STACK INTO R0
1979	021204	001401				BEQ	2\$:WE'RE OK-GO TO NEXT TEST, JUMP OVER ERROR PRINT OUT
1980	021206	104040				ERROR	+40		:ILLEGAL MODE NOT MAPPED TO USER MODE
1981	021210	005037	177776		2\$:	CLR	PSW		:RESET PSW TO KERNEL BEFORE LEAVING

 :* THE NEXT SEVEN (7) TESTS WILL TRY TO ADDRESS ALL OF THE
 :* MEMORY MANAGEMENT REGISTERS (SRO->SR3; I&D SPACE KERNEL,
 :* SUPERVISOR & USER PAR/PDR'S).
 :* EVERY TIME A REGISTER TIMES OUT ITS ADDRESS WILL BE REPORTED.
 :* AT THE END OF EACH TEST A SUMMARY OF THE ADDRESSES THAT TIMED
 :* OUT DURING THAT TEST IS GIVEN. THE RESULTS OF 'AND-ING' AND 'OR-ING'
 :* THEIR ADDRESSES IS GIVEN TO SHOW WHICH ADDRESS LINES MAY BE
 :* STUCK AT 0 OR 1. THE PAR/PDR ADDRESS AND KT MUX'S ARE THE
 :* THINGS BEING CHECKED.
 :*

1994

2002

```
.SBTTL TEST # 5 - SR0,SR1,SR2,SR3 TIMEOUT TEST
*****
*TEST 5      SR0,SR1,SR2,SR3 TIMEOUT TEST
*
*          THIS TEST ADDRESSES THE MEMORY MANAGEMENT STATUS REGISTERS
*          0,1,2, AND 3. DATA WILL BE WRITTEN OR READ FROM THESE REGISTERS
*          IN LATER TESTS, THIS TEST JUST CHECKS FOR A RESPONSE.
*
*****
```

```
TST5:  SCOPE
2003 021214 000004          CLR          PSW          ;MAKE SURE KERNAL MODE IS SELECTED
2004 021216 005037 177776  1$:  MOV          #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
2005 021222 012737 021264 001110  MOV          #6$, 4      ;SET TIMEOUT VECTOR TO 6$
2006 021230 012737 021330 000004  MOV          #SR0, R0    ;LOAD R0 WITH ADDRESS OF FIRST REG.
2007 021236 012700 177572          MOV          #3, R1     ;LOAD R1 WITH THE LOOP COUNT
2008 021242 012701 000003          MOV          #-1,ADRAND ;INITIALIZE 'AND' OF ADDRS. LOC.
2009 021246 012737 177777 001320  ADDROR       ;INITIALIZE 'OR' OF ADDRS. LOC.
2010 021254 005037 001316          CLR          TONUM      ;INITIALIZE 'TIMEOUTS' COUNTER
2011 021260 005037 001274          CLR          (R0)       ;TRY ADDRESSING A STATUS REGISTER
2012 021264 005710          TST          ;IF IT TIMES OUT GO TO 6$
2013 021266 062700 000002          ADD          #2, R0     ;PUT NEXT ADDRESS IN R0
2014 021272 077104          SOB          R1, 2$    ;LOOP BACK TO 2$ UNTIL ALL TESTED
2015 021274 012700 172516          MOV          #SR3, R0  ;LOAD R0 WITH THE ADDRESS OF SR3
2016 021300 005710          TST          (R0)     ;TRY ADDRESSING SR3
2017 021302 012737 021222 001110  4$:  MOV          #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
2018 021310 005737 001274          TST          TONUM    ;DID ANY OF THE STATUS REG.S TIMEOUT?
2019 021314 001401          BEQ          5$       ;BRANCH IF NO
2020 021316 104010          ERROR       +10      ;SUMMARY OF STATUS REG. TIMEOUTS
2021 021320 012737 003232 000004  5$:  MOV          #TIMERR, 4 ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
2022 021326 000417          BR          TST6     ;;GO TO NEXT TEST
2023 021330 062706 000004          ADD          #4, KSP   ;CLEAN UP THE STACK
2024 021334 104007          ERROR       +7       ;ONE OF THE STATUS REGS. TIMED OUT
2025          ;FOR TIGHTER SCOPE LOOP
2026          ;REPLACE ERROR CALL WITH
2027          ;'BR 2$' = 000756
2028 021336 010002          MOV          R0, R2   ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
2029 021340 050237 001316          BIS          R2, ADDROR ;'OR' IT WITH OTHER ADDRS. THAT TIMED OUT
2030 021344 005102          COM          R2      ;'AND' IT WITH OTHER ADDRS. THAT TIMED OUT
2031 021346 040237 001320          BIC          R2, ADRAND
2032 021352 005237 001274          INC          TONUM    ;INCREMENT THE TIMEOUT COUNTER
2033 021356 032700 172516          BIT          #SR3, R0 ;TEST FOR LAST REGISTER
2034 021362 001347          BNE          4$      ;SR3 IS LAST; GO TO FINAL ERROR REPORT
2035 021364 000740          BR          3$       ;BRANCH BACK TO TEST THE NEXT ADDR.
```



```
2039          .SBTTL TEST # 6 - KERNEL I&D SPACE PAR'S TIMEOUT TEST
          :*****
          :*TEST 6      KERNEL I&D SPACE PAR'S TIMEOUT TEST
          :*      THIS TEST ADDRESSES THE SIXTEEN (16) KERNEL PAGE ADDRESS
          :*      REGISTERS (KIPAR0-KIPAR7 AND KDPAR0-KDPAR7)
          :*      AND CHECKS THAT SOMETHING RESPONDS TO THEIR ADDRESSES.
          :*****
          TST6:  SCOPE
          1$:   JSR      PC,TIMTST      ;GO DO THE TEST ON PAGE OF THIS LISTING
          .WORD  KIPAR0      ;PARAMETER USED FOR SUBROUTINE
          ERROR  +7          ;RETURN IS HERE FOR 1ST ERROR - ONE OF THE REGISTERS
          :TIMED OUT.  FOR TIGHTER SCOPE LOOP, REPLACE THE
          :'MOV #1,$TMP4' INSTRUCTION IN THE SUBROUTINE WITH 000756
          BR      1$          ;RETURN TO TEST
          ERROR  +10         ;RETURN IS HERE FOR 2ND ERROR - SUMMARY OF REG TIMEOUTS

          021366 000004
2040 021370 004737 003024
2041 021374 172340
2042 021376 104007
2043
2044
2045 071400 000773
2046 021402 104010
```

2050
2051 021404 000004
2052 021406 004737 003024
2053 021412 172300
2054 021414 104007
2055
2056 021416 000773
2057 021420 104010

```
.SBTTL TEST # 7 - KERNEL I&D SPACE PDR'S TIMEOUT TEST
:*****
:*TEST 7      KERNEL I&D SPACE PDR'S TIMEOUT TEST
:*      THIS TEST ADDRESSES THE SIXTEEN (16) KERNEL PAGE DFSCRIPTOR
:*      REGISTERS (KIPAR0-KIPAR7 AND KDPAR0-KDPAR7)
:*      AND CHECKS THAT SOMETHING RESPONDS TO THEIR ADDRESSES.
:*****
TST7:  SCOPE
1$:    JSR      PC,TIMTST      ;GO DO THE TEST ON PAGE OF THIS LISTING
        .WORD   KIPDR0        ;PARAMETER USED FOR SUBROUTINE
        ERROR  +7             ;RETURN IS HERE FOR 1ST ERROR - ONE OF THE REGISTERS
                                   ;TIMED OUT. FOR TINGTER SCOPE LOOP, REPLACE THE
                                   ;'MOV #1,$TMP4' INSTRUCTION IN THE SUBROUTINE WITH 000756
        BR      1$            ;RETURN TO TEST
        ERROR  +10           ;RETURN IS HERE FOR 2ND ERROR - SUMMARY OF REG TIMEOUTS
```

```
2061 .SBTTL TEST # 10 - SUPERVISOR I&D SPACE PAR'S TIMEOUT TEST
:*****
:TEST 10 SUPERVISOR I&D SPACE PAR'S TIMEOUT TEST
:* THIS TEST ADDRESSES THE SIXTEEN (16) SUPERVISOR PAGE ADDRESS
:* REGISTERS (SIPARO-SIPAR7 AND SDPARO-SDPAR7)
:* AND CHECKS THAT SOMETHING RESPONDS TO THEIR ADDRESSES.
:*****
2062 021422 000004 003024 TST10: SCOPE
2063 021424 004737 JSR PC,TIMTST ;GO DO THE TEST ON PAGE OF THIS LISTING
2064 021430 172240 .WORD SIPARO ;PARAMETER USED FOR SUBROUTINE
2065 021432 104007 ERROR +7 ;RETURN IS HERE FOR 1ST ERROR - ONE OF THE REGISTERS
2066 ;TIMED OUT. FOR TIGHTER SCOPE LOOP, REPLACE THE
2067 021434 000773 BR 1$ ;'MOV #1,STMP4' INSTRUCTION IN THE SUBROUTINE WITH 000756
2068 021436 104010 ERROR +10 ;RETURN TO TEST
;RETURN IS HERE FOR 2ND ERROR - SUMMARY OF REG TIMEOUTS
```

2072

.SBTTL TEST # 11 - SUPERVISOR I&D SPACE PDR'S TIMEOUT TEST

*TEST 11 SUPERVISOR I&D SPACE PDR'S TIMEOUT TEST

* THIS TEST ADDRESSES THE SIXTEEN (16) SUPERVISOR PAGE DESCRIPTOR
* REGISTERS (SIPAR0-SIPAR7 AND SDPAR0-SDPAR7)
* AND CHECKS THAT SOMETHING RESPONDS TO THEIR ADDRESSES.

021440 000004
2073 021442 004737 003024
2074 021446 172200
2075 021450 104007
2076
2077
2078 021452 000773
2079 021454 104010

TST11: SCOPE

1\$: JSR PC,TIMTST
.WORD SIPDRO
ERROR +7

:GO DO THE TEST ON PAGE OF THIS LISTING
:PARAMETER USED FOR SUBROUTINE
:RETURN IS HERE FOR 1ST ERROR - ONE OF THE REGISTERS
:TIMED OUT. FOR TIGHTER SCOPE LOOP, REPLACE THE
: 'MOV #1,\$TMP4' INSTRUCTION IN THE SUBROUTINE WITH 000756
:RETURN TO TEST
:RETURN IS HERE FOR 2ND ERROR - SUMMARY OF REG TIMEOUTS

2083

```
.SBTTL TEST # 12 - USER I&D SPACE PAR'S TIMEOUT TEST
:*****
:*TEST 12      USER I&D SPACE PAR'S TIMEOUT TEST
:*      THIS TEST ADDRESSES THE SIXTEEN (16) USER PAGE ADDRESS
:*      REGISTERS (UIPAR0-UIPAR7 AND UDPAR0-UDPAR7)
:*      AND CHECKS THAT SOMETHING RESPONDS TO THEIR ADDRESSES.
:*****
```

```
021456 000004
2084 021460 004737 003024
2085 021464 177640
2086 021466 104007
2087
2088
2089 021470 000773
2090 021472 104010
```

```
TST12: SCOPE
1$:     JSR      PC,TIMTST      ;GO DO THE TEST ON PAGE OF THIS LISTING
        .WORD   UIPAR0        ;PARAMETER USED FOR SUBROUTINE
        ERROR  +7            ;RETURN IS HERE FOR 1ST ERROR - ONE OF THE REGISTERS
                                ;TIMED OUT. FOR TIGHTER SCOPE LOOP, REPLACE THE
                                ;'MOV #1,$TMP4' INSTRUCTION IN THE SUBROUTINE WITH 000756
                                ;RETURN TO TEST
                                ;RETURN IS HERE FOR 2ND ERROR - SUMMARY OF REG TIMEOUTS
        BR      1$
        ERROR  +10
```

2094

```
.SBTTL TEST # 13 - USER I&D SPACE PDR'S TIMEOUT TEST  
:*****  
:*TEST 13 USER I&D SPACE PDR'S TIMEOUT TEST  
:* THIS TEST ADDRESSES THE SIXTEEN (16) USER PAGE DESCRIPTOR  
:* REGISTERS (UIPAR0-UIPAR7 AND UDPAR0-UDPAR7)  
:* AND CHECKS THAT SOMETHING RESPONDS TO THEIR ADDRESSES.  
:*****
```

```
021474 000004  
2095 021476 004737 003024  
2096 021502 177600  
2097 021504 104007  
2098  
2099  
2100 021506 000773  
2101 021510 104010
```

```
TST13: SCOPE  
1$: JSR PC,TIMTST ;GO DO THE TEST ON PAGE OF THIS LISTING  
.WORD UIPDRO ;PARAMETER USED FOR SUBROUTINE  
ERROR +7 ;RETURN IS HERE FOR 1ST ERROR - ONE OF THE REGISTERS  
;TIMED OUT. FOR TIGHTER SCOPE LOOP, REPLACE THE  
;'MOV #2,$TMP4' INSTRUCTION IN THE SUBROUTINE WITH 000756  
BR 1$ ;RETURN TO TEST  
ERROR +10 ;RETURN IS HERE FOR 2ND ERROR - SUMMARY OF REG TIMEOUTS
```

2114

```
.SBTTL TEST # 14 - MMR0(15:13) BIT TEST & MMR2 TEST
:*****
:*TEST 14      MMR0(15:13) BIT TEST & MMR2 TEST
:*
:*          THIS TEST CHECKS BITS <15:13> OF MEMORY MANAGEMENT
:*          REGISTER 0 TO SEE THAT EACH CAN BE SET AND CLEARED
:*          AND THAT A 'RESET' WILL CLEAR ALL OF THEM.  A TEST
:*          OF THESE THREE ERROR BITS CHECKS PART OF SRO, THE
:*          SRO MUX AND THE KTMUX.  THE REST OF THE BITS IN
:*          SRO WILL BE CHECKED LATER.
:*          ALSO CHECK THAT SR2 IS TRACKING WITH MEM. MGMT.
:*          OFF BUT LOCKS UP WHEN ANY OF SRO ERROR BITS SET.
:*
:*****
```

```
TST14: SCOPE
1$:  MOV      #SRO,R0      ;LOAD ADDRESS OF SRO INTO R0
     MOV      #160000,(R0) ;SET BITS <15:13> IN SRO (ERROR BITS)
     RESET    ;ISSUE AND 'INIT' SIGNAL
     MOV      (R0),R1     ;READ SRO INTO R1 TO SEE IF CLEAR
     BEQ      2$         ;BRANCH IF SRO<15:13> CLEARED BY 'INIT'
     ERROR    +11        ;SRO<15:13> NOT CLEARED BY A 'RESET'
                          ;FOR TIGHTER SCOPE LOOP
                          ;REPLACE ERROR CALL WITH
                          ;'BR 1$' = 000770
2$:  MOV      #2$,SLPERR   ;SET LOOP ON ERROR POINTER TO 2$
     MOV      SR2,WASSR2  ;READ CONTENTS OF SR2
     MOV      #2$,R1     ;LOAD EXPECTED CONTENTS INTO R1
     CMP      R1,WASSR2   ;IS SR2 TRACKING?
     BEQ      3$         ;BRANCH IF YES
     ERROR    +21        ;SR2 NOT 'TRACKING' VIRTUAL ADDRESSES
                          ;FOR TIGHTER SCOPE LOOP
                          ;REPLACE ERROR CALL WITH
                          ;'BR 2$' = 000767
3$:  MOV      #4$,SLPERR   ;SET LOOP ON ERROR POINTER TO 4$
     MOV      #BIT15,R1  ;PUT DATA TO BE WRITTEN IN R1
     MOV      #3,R3      ;SETUP R3 AS A LOOP COUNTER
     CLR      (R0)       ;CLEAR SRO
     BIS      R1,(R0)    ;SET ONE OF THE ERROR BITS IN SRO
     MOV      (R0),R2    ;READ SRO INTO R2
     CMP      R1,R2     ;DID RIGHT ERROR BIT GET SET?
     BEQ      6$         ;BRANCH IF YES
     ERROR    +12        ;BITS WERE SET WRONG IN SRO
                          ;FOR TIGHTER SCOPE LOOP
                          ;REPLACE ERROR CALL WITH
                          ;'BR 4$' = 000772
4$:  MOV      #5$,R4      ;LOAD EXPECTED CONTENTS OF SR2 IN R4
     MOV      SR2,WASSR2  ;READ SR2
     CMP      R4,WASSR2   ;DID SR2 LOCK UP WHEN ERROR
     BEQ      7$         ;BIT SET IN SR1?
     ERROR    +27        ;BRANCH IF YES
                          ;SR2 DID NOT LOCK UP
                          ;FOR TIGHTER SCOPE LOOP
                          ;REPLACE ERROR CALL WITH
                          ;'BR 4$' = 000761
5$:  ROR      R1          ;CHANGE DATA TO CHECK NEXT ERROR BIT
     SOB     R3,4$       ;LOOP BACK UNTIL <15:13> ALL TESTED
     CLR     (R0)        ;CLEAR SRO BEFORE LEAVING
2115 021512 000004
2116 021514 012700 177572
2117 021520 012710 160000
2118 021524 000005
2119 021526 011001
2120 021530 001404
2121 021532 104011
2122
2123
2124 021534 012737 021542 001110
2125 021542 013737 177576 001270 2$:
2126 021550 012701 021542
2127 021554 020137 001270
2128 021560 001401
2129 021562 104021
2130
2131
2132
2133 021564 012737 021602 001110 3$:
2134 021572 012701 100000
2135 021576 012703 000003
2136 021602 005010
2137 021604 050110 4$:
2138 021606 011002 5$:
2139 021610 020102
2140 021612 001401
2141 021614 104012
2142
2143
2144
2145 021616 012704 021604
2146 021622 013737 177576 001270 6$:
2147 021630 020437 001270
2148
2149 021634 001401
2150 021636 104027
2151
2152
2153
2154 021640 006001 7$:
2155 021642 077321
2156 021644 005010
```

2157 021646 012737 021514 001110 MOV #1\$, \$LPERR ;RESET LOOP ON ERROR POINTER TO 1\$

2167

```
.SBTTL TEST # 15 - SRO & PSW DUAL ADDRESSING TEST
*****
:TEST 15      SRO & PSW DUAL ADDRESSING TEST
:
: THIS TEST CHECKS MORE OF THE ADDRESS DETECTION LOGIC BY
: VERIFYING THAT STATUS REGISTER 0 IS NOT EFFECTED BY WRITING
: TO THE PSW AND THAT THE LOW BYTE OF STATUS REGISTER 0
: IS NOT EFFECTED BY WRITING TO ITS HIGH BYTE. THIS IS TO
: SEE IF ADJACENT OUTPUTS ARE SHORTED ON THE ADDRESS DET. LOGIC.
:
:*****
```

```

2168 021654 000004
2168 021656 005037 177776
2169 021662 005037 177572
2170 021666 012737 000340 177776
2171 021674 013700 177572
2172 021700 001401
2173 021702 104013
2174
2175
2176
2177 021704 005037 177572
2178 021710 005037 177776
```

```
TST15: SCOPE
1$: CLR PSW ;CLEAR THE PSW
CLR SRO ;CLEAR STATUS REGISTER 0
MOV #340,PSW ;SET PRIORITY 7 IN LOW BYTE OF PSW
MOV SRO,RO ;READ STATUS REGISTER 0
BEQ 2$ ;BRANCH IF IT WAS STILL 0
ERROR +13 ;SRO EFFECTED BY A WRITE TO THE PSW
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 1$' = 000767
2$: CLR SRO ;BE SURE SRO IS 0 BEFORE LEAVING
CLR PSW ;BE SURE PSW IS 0 BEFORE LEAVING
```

2191

```
.SBTTL TEST # 16 - BIT TEST OF MEMORY MANAGEMENT REGISTER 1
:*****
:*TEST 16 BIT TEST OF MEMORY MANAGEMENT REGISTER 1
:*THIS TEST WILL CAUSE BITS IN MMR1 TO BE LOCKED BY SETTING BITS <15:13>
:*IN MMR0. THE REGISTER ONLY GETS CLOKED ON AN INSTRUCTION THAT AUTO
:*INCREMENTS OR AUTO DECREMENTS A REGISTER. THE LOWER BYTE IS ALWAYS CLOKED
:*FIRST AND THE UPPER BYTE IS ONLY CLOKED IF BOTH SOURCE AND DESTINATION
:*AUTO INCREMENT OR DECREMENT A REGISTER.
:*ALL COUNTS (+1,-1,+2,-2) THAT CAN BE GENERATED ARE CLOKED INTO THE HIGH
:*AND LOW BYTES. ALL REGISTER NUMBERS ARE GENERATED, INCLUDING ALL THREE
:*R6'S, AND ALL THE BITS THAT HOLD THE REGISTER NUMBER IN BOTH BYTES
:*ARE TESTED; HOWEVER, ALL REGISTER NUMBERS ARE CLOKED INTO BOTH HIGH AND
:*LOW BYTES, BUT ENOUGH ARE USED TO TEST THE LOGIC.
:*****
```

```
TST16: SCOPE
20$: MOV #MMR1,R0 ;PUT ADDRESS OF MMR1 INTO R0
MOV #BIT13,R4 ;SET READ ONLY BIT IN R4
MOV #11$,SLPERR ;SET LOOP ON ERROR POINTER TO 11$
2192 021714 000004 177574 11$: MOV #BIT15,MMR0 ;LOCK UP MMR1 LOW BYTE 027,HIGH BYTE 027
2193 021722 012704 020000 ;WORD-013427
2194 021726 012737 021734 001110 ;PUT EXPECTED DATA IN R3
2195 021734 012737 100000 177572 ;READ MMR1 INTO R1
2196 ;SEE IF DATA MATCHES
2197 021742 012703 013427 ;BRANCH IF DATA MATCHES
2198 021746 011001 ;MMR1 DID NOT TRACK PROPERLY
2199 021750 020103 ;SET LOOP ON ERROR POINTER TO 12$
2200 021752 001401 ;ISSUE INIT
2201 021754 104014 ;SEE IF MMR1 IS CLEARED
2202 021756 012737 021764 001110 10$: BEQ 1$ ;BRANCH IF MMR1 IS ZERO
2203 021764 000005 12$: ERROR +11 ;CAN'T CLEAR MMR1
2204 021766 011001 ;SET LOOP ON ERROR POINTER TO 13$
2205 021770 001401 ;PUT ADDRESS OF MMR0 INTO R2
2206 021772 104011 ;LOCK UP MMR1-LOWER BYTE 027
2207 021774 012737 022002 001110 1$: MOV #13$,SLPERR ;UPPER BYTE 022-WORD 011027
2208 022002 012702 177572 13$: MOV #MMR0,R2 ;READ MMR1
2209 022006 012722 040000 ;PUT EXPECTED DATA IN R3
2210 ;SEE IF DATA MATCHES
2211 022012 011001 ;BRANCH IF DATA MATCHES
2212 022014 012703 011027 ;MMR1 DID NOT TRACK PROPERLY
2213 022020 020103 ;CLEAR MMR0
2214 022022 001401 ;SET LOOP ON ERROR POINTER TO 14$
2215 022024 104014 ;PUT ADDRESS OF MMR0 PLUS 2 IN R2
2216 022026 005037 177572 2$: MOV #MMR0 ;LOCK UP MMR1-LOWER BYTE 000
2217 022032 012737 022040 0C1110 14$: MOV #14$,SLPERR ;UPPER BYTE 362-WORD 171000
2218 022040 012702 177574 ;READ MMR1
2219 022044 010442 ;PUT EXPECTED DATA IN R3
2220 ;SEE IF DATA MATCHES
2221 022046 011001 ;BRANCH IF DATA MATCHES
2222 022050 012703 171000 ;MMR1 DID NOT TRACK PROPERLY
2223 022054 020103 ;CLEAR MMR0
2224 022056 001401 ;SET LOOP ON ERROR POINTER TO 15$
2225 022060 104014 ;PUT ADDRESS OF MMR0'S UPPER BYTE IN R5
2226 022062 005012 ;PUT ADDRESS OF READ ONLY BIT <13> IN R1
2227 022064 012737 022072 001110 3$: MOV #15$,SLPERR ;LOCK UP MMR1-LOWER BYTE 011
2228 022072 012705 177573 15$: MOV #MMR0+1,R5 ;UPPER BYTE 015-WORD 006411
2229 022076 012701 001343 ;READ MMR1
2230 022102 112125 ;PUT EXPECTED DATA IN R3
2231 ;SEE IF DATA MATCHES
2232 022104 011001 ;BRANCH IF DATA MATCHES
2233 022106 012703 006411 ;MMR1 DID NOT TRACK PROPERLY
```

2234	022112	020103				CMP	R1,R3	:SEE IF DATA MATCHES
2235	022114	001401				BEQ	4\$:BRANCH IF DATA MATCHES
2236	022116	104014				ERROR	+14	:MMR1 DID NOT TRACK PROPERLY
2237	022120	005012			4\$:	CLR	(R2)	:CLEAR MMR0
2238	022122	012737	022130	001110		MOV	#16\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 16\$
2239	022130	012701	177574		16\$:	MOV	#MMR0+2,R1	:PUT ADDRESS OF MMR0'S UPPER BYTE
2240								:PLUS 1 IN R1
2241	022134	012705	001344			MOV	#READON+2,R5	:PUT ADDRESS OF READ ONLY BIT <13>
2242								:PLUS 2 IN R5
2243	022140	114541				MOVB	-(R5),-(R1)	:LOCK UP MMR1-LOWER BYTE 375
2244								:UPPER BYTE 371-WORD 174775
2245	022142	011001				MOV	(R0),R1	:READ MMR1
2246	022144	012703	174775			MOV	#174775,R3	:PUT EXPECTED DATA IN R3
2247	022150	020103				CMP	R1,R3	:SEE IF DATA MATCHES
2248	022152	001401				BEQ	5\$:BRANCH IF DATA MATCHES
2249	022154	104014				ERROR	+14	:MMR1 DID NOT TRACK PROPERLY
2250	022156	005012			5\$:	CLR	(R2)	:CLEAR MMR0
2251	022160	012737	022200	001110		MOV	#17\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 17\$
2252	022166	012737	040000	177776		MOV	#40000,PSW	:SET SUPERVISOR MODE
2253	022174	010637	001176			MOV	SSP,\$TMP0	:SAVE SUPERVISOR STACK POINTER
2254	022200	012706	177574		17\$:	MOV	#MMR0+2,SSP	:PUT ADDRESS OF MMR0+2 IN SSP
2255	022204	012746	040000			MOV	#40000,-(SSP)	:LOCK UP MMR1-LOWER BYTE 027
2256								:UPPER BYTE 366-WORD 173027
2257	022210	011001				MOV	(R0),R1	:READ MMR1
2258	022212	012703	173027			MOV	#173027,R3	:PUT EXPECTED DATA IN R3
2259	022216	020103				CMP	R1,R3	:SEE IF DATA MATCHES
2260	022220	001401				BEQ	6\$:BRANCH IF DATA MATCHES
2261	022222	104014				ERROR	+14	:MMR1 DID NOT TRACK PROPERLY
2262	022224	013706	001176		6\$:	MOV	\$TMP0,SSP	:RESTORE THE SUPERVISOR STACK POINTER
2263	022230	005012				CLR	(R2)	:CLEAR MMR0
2264	022232	012737	022252	001110		MOV	#18\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 18\$
2265	022240	012737	140000	177776		MOV	#140000,PSW	:SET USER MODE
2266	022246	010637	001176			MOV	USP,\$TMP0	:SAVE USER STACK POINTER
2267	022252	012706	177572		18\$:	MOV	#MMR0,USP	:PUT ADDRESS OF MMR0 IN USP
2268	022256	012726	100000			MOV	#100000,(USP)+	:LOCK UP MMR1-LOWER BYTE 027
2269								:UPPER BYTE 026-WORD 013027
2270	022262	011001				MOV	(R0),R1	:READ MMR1
2271	022264	012703	013027			MOV	#013027,R3	:PUT EXPECTED DATA IN R3
2272	022270	020103				CMP	R1,R3	:SEE IF DATA MATCHES
2273	022272	001401				BEQ	7\$:BRANCH IF DATA MATCHES
2274	022274	104014				ERROR	+14	:MMR1 DID NOT TRACK PROPERLY
2275	022276	013706	001176		7\$:	MOV	\$TMP0,USP	:RESTORE THE USER STACK POINTER
2276	022302	005037	177776			CLR	PSW	:GO BACK TO KERNAL MODE
2277	022306	005037	177572			CLR	MMR0	:LET ALL MEM MGT REG'S TRACK
2278	022312	012737	021716	001110		MOV	#20\$, \$LPERR	:SET LOOP ON ERROR POINTER TO START OF TEST

2286

```
.SBTTL TEST # 17 - BIT TEST OF MEMORY MANAGEMENT REGISTER 3
:*****
:*TEST 17 BIT TEST OF MEMORY MANAGEMENT REGISTER 3
:* THIS TEST SETS AND CLEARS BITS <05:04> AND <02:00> OF MMR3. IT DOES
:* NOT TEST THAT THE BITS FUNCTION PROPERLY SINCE THAT REQUIRES MORE
:* LOGIC; HOWEVER, IT TESTS THE REGISTER AND THE DATA PATH TO AND FROM
:* THE REGISTER. THE PROPER FUNCTIONING OF THESE BITS IS TESTED LATER
:* IN SEVERAL TESTS.
:*****
```

```
TST17: SCOPE
20$: MOV #MMR3,R0 ;PUT ADDRESS OF MMR3 IN R0
MOV #1,R2 ;SET BIT TO FLOAT THRU MMR3
CLR (R0) ;CLEAR MMR3
MOV (R0),R1 ;READ MMR3 INTO R1
BEQ 5$ ;BRANCH IF ZERO
ERROR +11 ;CAN'T CLEAR MMR3
2287 022320 000004
2288 022322 012700 172516
2289 022326 012702 000001
2290 022332 005010
2291 022334 111001
2292 022336 001401
2293 022340 104011
2294 022342 012703 000006
2295 022344 012737 022354 001110
2296 022346 010210
2297 022354 011001
2298 022356 120201
2299 022360 120201
2300 022362 001401
2301 022364 104015
2302 022366 006302
2303 022370 077307
2304 022372 012702 000077
2305 022376 012737 022404 001110
2306 022404 010210
2307 022406 011001
2308 022410 120201
2309 022412 001401
2310 022414 104015
2311 022416 012737 022424 001110
2312 022424 000005
2313 022426 111002
2314 022430 001401
2315 022432 104011
2316 022434 012737 022322 001110
4$: MOV #20$, $LPERR ;SET LOOP ON ERROR POINTER TO START OF TEST
```

2315
2316
2317
2318

```
.SBTTL  PAR AND PDR TESTS  
:*****  
:      GROUP 2  PAR AND PDR TESTS  
:*****
```

/

2332

```
.SBTTL TEST # 20 - DUAL ADDRESS KERNAL PAR'S,ON LOADING
:*****
:*TEST 20      DUAL ADDRESS KERNAL PAR'S,ON LOADING
:*
:*THIS TEST FIRST CLEARS ALL THE KERNAL PAGE ADDRESS REGISTERS,
:*AND CHECKS TO SEE THAT THEY EACH HOLD ZERO. THEN, STARTING
:*WITH I-SPACE ADDRESS REGISTER ZERO, ONE REGISTER AT A TIME IS
:*LOADED WITH A NEGATIVE ONE. ALL KERNAL ADDRESS REGISTERS ARE
:*NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
:*INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN
:*AT THE END OF THIS TEST.
:*****
```

2333 022442 000004
 022444 004737 002276
 022450 012737 022476 001110
 022456 012705 172340
 022462 004737 002260
 022466 012700 172340
 022472 012701 000020
 022476 011002
 022500 001401
 022502 104030
 022504 062700 000002
 022510 077106

```
TST20: SCOPE
20$: JSR PC,CLEANUP ;INITIALIZE THE ERROR LOCATIONS
     MOV #1$, $LPERR ;SET LOOP ON ERROR POINTER TO 1$
     MOV #KIPAR0,R5 ;PUT ADDRESS OF FIRST PAR IN R5
     JSR PC,CLRREG ;CLEAR 16 REGISTERS POINTED TO BY R5
     MOV #KIPAR0,R0 ;PUT ADDRESS OF FIRST PAR IN R0
     MOV #20,R1 ;BRANCH COUNT IS 16 DECIMAL
1$:  MOV (R0),R2 ;READ PAR TO R2
     BEQ 2$ ;BRANCH IF PAR IS 0
     ERROR +30 ;PAR NOT ZERO
2$:  ADD #2,R0 ;POINT TO NEXT REGISTER
     SOB R1,1$ ;BRANCH BACK TO 1$ 15 TIMES
```

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

022512 012737 022524 001110
 022520 012700 172340
 022524 012705 172340
 022530 004737 002260
 022534 012701 172376
 022540 012710 177777
 022544 005037 001176
 022550 011102
 022552 001406
 022554 020001

```
MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
MOV #KIPAR0,R0 ;PUT ADDRESS OF FIRST PAR IN R0
3$: MOV #KIPAR0,R5 ;LOAD STARTING ADDRESS INTO R5
     JSR PC,CLRREG ;CLEAR 16 REGISTERS POINTED TO BY R5
     MOV #KIDPAR7,R1 ;PUT KIDPAR7 ADDRESS INTO R1
     MOV #-1,(R0) ;LOAD REGISTER UNDER TEST
4$:  CLR $TMP0 ;FLAG TO INDICATE THERE WAS A MATCH
     MOV (R1),R2 ;READ ALL REGISTERS
     BEQ 6$ ;BRANCH IF REGISTER IS 0
     CMP R0,R1 ;IS THE ADDRESS OF NON-ZERO REGISTER THE SAME
     ;AS THE REGISTER UNDER TEST?
     BEQ 5$ ;BRANCH IF ADDRESSES MATCH
```

022556 001402
 022560 004737 002334
 022564 005237 001176
 022570 162701 000002
 022574 022701 172340
 022600 101761
 022602 062700 000002
 022606 022700 172376
 022612 103344
 022614 012737 022444 001110
 022622 005737 001322
 022626 000404
 022630 013737 001322 001210
 022636 104031

```
5$: JSR PC,DUALADR ;LOG AND REPORT ERRORS
6$: INC $TMP0 ;SET FLAG WHEN ADDRESSES MATCH
     SUB #2,R1 ;POINT TO NEXT REGISTER
     CMP #KIPAR0,R1 ;SEE IF ALL REGISTERS HAVE BEEN READ
     BLOS 4$ ;BRANCH IF MORE TO READ
     ADD #2,R0 ;NOW LOAD THE NEXT REGISTER
     CMP #KIDPAR7,R0 ;SEE IF THERE ARE MORE REGISTERS TO TEST
     BHIS 3$ ;BRANCH IF MORE REGISTERS TO TEST
     MOV #20$, $LPERR ;SET LOOP ON ERROR POINTER TO START OF TEST
     TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
     BR TST21 ;BRANCH TO NEXT TEST IF NO ERRORS
     MOV ERRCNT,$TMP5 ;SAVE # OF ERRORS FOR PAR OUT
     ERROR +31 ;SUMMARY OF DUAL ADDRESS TEST
```

2337

```
.SBTTL TEST # 21 - DUAL ADDRESS SUPERVISOR PAR'S,ON LOADING
*****
*TEST 21          DUAL ADDRESS SUPERVISOR PAR'S,ON LOADING
*
*THIS TEST FIRST CLEARS ALL THE SUPERVISOR PAGE ADDRESS REGISTERS,
*AND CHECKS TO SEE THAT THEY EACH HOLD ZERO. THEN, STARTING
*WITH I-SPACE ADDRESS REGISTER ZERO, ONE REGISTER AT A TIME IS
*LOADED WITH A NEGATIVE ONE. ALL SUPERVISOR ADDRESS REGISTERS ARE
*NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
*INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN
*AT THE END OF THIS TEST.
*****
```

2338	022640	000004			TST21: SCOPE	
	022642	004737	002276		20\$: JSR	PC,CLEANUP ;INITIALIZE THE ERROR LOCATIONS
	022646	012737	022674	001110	MOV	#1\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 1\$
	022654	012705	172240		MOV	#SIPARO,R5 ;PUT ADDRESS OF FIRST PAR IN R5
	022660	004737	002260		JSR	PC,CLRREG ;CLEAR 16 REGISTERS POINTED TO BY R5
	022664	012700	172240		MOV	#SIPARO,R0 ;PUT ADDRESS OF FIRST PAR IN R0
	022670	012701	000020		MOV	#20,R1 ;BRANCH COUNT IS 16 DECIMAL
	022674	011002			1\$: MOV	(R0),R2 ;READ PAR TO R2
	022676	001401			BEQ	2\$;BRANCH IF PAR IS 0
	022700	104030			ERROR	+30 ;PAR NOT ZERO
	022702	062700	000002		2\$: ADD	#2,R0 ;POINT TO NEXT REGISTER
	022706	077106			SOB	R1,1\$;BRANCH BACK TO 1\$ 15 TIMES
						;NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE REGISTER AND
						;READING THE REST TO SEE ONLY THAT REGISTER IS NON ZERO. (ANY
						;DROPPED BITS WILL BE FOUND IN THE NEXT TEST).
	022710	012737	022722	001110	MOV	#3\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 3\$
	022716	012700	172240		MOV	#SIPARO,R0 ;PUT ADDRESS OF FIRST PAR IN R0
	022722	012705	172240		3\$: MOV	#SIPARO,R5 ;LOAD STARTING ADDRESS INTO R5
	022726	004737	002260		JSR	PC,CLRREG ;CLEAR 16 REGISTERS POINTED TO BY R5
	022732	012701	172276		MOV	#SDPAR7,R1 ;PUT SDPAR7 ADDRESS INTO R1
	022736	012710	177777		MOV	#-1,(R0) ;LOAD REGISTER UNDER TEST
	022742	005037	001176		4\$: CLR	\$TMP0 ;FLAG TO INDICATE THERE WAS A MATCH
	022746	011102			MOV	(R1),R2 ;READ ALL REGISTERS
	022750	001406			BEQ	6\$;BRANCH IF REGISTER IS 0
	022752	020001			CMP	R0,R1 ;IS THE ADDRESS OF NON-ZERO REGISTER THE SAME
						;AS THE REGISTER UNDER TEST?
	022754	001402			BEQ	5\$;BRANCH IF ADDRESSES MATCH
	022756	004737	002334		JSR	PC,DUALADR ;LOG AND REPORT ERRORS
	022762	005237	001176		5\$: INC	\$TMP0 ;SET FLAG WHEN ADDRESSES MATCH
	022766	162701	000002		6\$: SUB	#2,R1 ;POINT TO NEXT REGISTER
	022772	022701	172240		CMP	#SIPARO,R1 ;SEE IF ALL REGISTERS HAVE BEEN READ
	022776	101761			BLOS	4\$;BRANCH IF MORE TO READ
	023000	062700	000002		ADD	#2,R0 ;NOW LOAD THE NEXT REGISTER
	023004	022700	172276		CMP	#SDPAR7,R0 ;SEE IF THERE ARE MORE REGISTERS TO TEST
	023010	103344			BHIS	3\$;BRANCH IF MORE REGISTERS TO TEST
	023012	012737	022642	001110	MOV	#20\$, \$LPERR ;SET LOOP ON ERROR POINTER TO START OF TEST
	023020	005737	001322		TST	ERRCNT ;SEE IF THERE WERE ANY ERRORS
	023024	000404			BR	TST22 ;BRANCH TO NEXT TEST IF NO ERRORS
	023026	013737	001322	001210	MOV	ERRCNT,\$TMP5 ;SAVE # OF ERRORS FOR PAR OUT
	023034	104031			ERROR	+31 ;SUMMARY OF DUAL ADDRESS TEST

2342

```
.SBTTL TEST # 22 - DUAL ADDRESS USER PAR'S,ON LOADING
:*****
:*TEST 22      DUAL ADDRESS USER PAR'S,ON LOADING
:*
:*THIS TEST FIRST CLEARS ALL THE USER PAGE ADDRESS REGISTERS,
:*AND CHECKS TO SEE THAT THEY EACH HOLD ZERO. THEN, STARTING
:*WITH I-SPACE ADDRESS REGISTER ZERO, ONE REGISTER AT A TIME IS
:*LOADED WITH A NEGATIVE ONE. ALL USER ADDRESS REGISTERS ARE
:*NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
:*INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN
:*AT THE END OF THIS TEST.
:*****
```

2343 023036 000004
 023040 004737 002276 001110
 023044 012737 023072
 023052 012705 177640
 023056 004737 002260
 023062 012700 177640
 023066 012701 000020
 023072 011002
 023074 001401
 023076 104030
 023100 062700 000002
 023104 077106

```
TST22: SCOPE
20$: JSR PC,CLEANUP ;INITIALIZE THE ERROR LOCATIONS
      MOV #1$,SLPERR ;SET LOOP ON ERROR POINTER TO 1$
      MOV #UIPAR0,R5 ;PUT ADDRESS OF FIRST PAR IN R5
      JSR PC,CLRREG ;CLEAR 16 REGISTERS POINTED TO BY R5
      MOV #UIPAR0,R0 ;PUT ADDRESS OF FIRST PAR IN R0
      MOV #20,R1 ;BRANCH COUNT IS 16 DECIMAL
1$: MOV (R0),R2 ;READ PAR TO R2
     BEQ 2$ ;BRANCH IF PAR IS 0
     ERROR +30 ;PAR NOT ZERO
2$: ADD #2,R0 ;POINT TO NEXT REGISTER
     SOB R1,1$ ;BRANCH BACK TO 1$ 15 TIMES
;NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE REGISTER AND
;READING THE REST TO SEE ONLY THAT REGISTER IS NON ZERO. (ANY
;DROPPED BITS WILL BE FOUND IN THE NEXT TEST).
      MOV #3$,SLPERR ;SET LOOP ON ERROR POINTER TO 3$
      MOV #UIPAR0,R0 ;PUT ADDRESS OF FIRST PAR IN R0
3$: MOV #UIPAR0,R5 ;LOAD STARTING ADDRESS INTO R5
     JSR PC,CLRREG ;CLEAR 16 REGISTERS POINTED TO BY R5
     MOV #UDPAR7,R1 ;PUT UDPAR7 ADDRESS INTO R1
     MOV #-1,(R0) ;LOAD REGISTER UNDER TEST
4$: CLR $TMP0 ;FLAG TO INDICATE THERE WAS A MATCH
     MOV (R1),R2 ;READ ALL REGISTERS
     BEQ 6$ ;BRANCH IF REGISTER IS 0
     CMP R0,R1 ;IS THE ADDRESS OF NON-ZERO REGISTER THE SAME
     ;AS THE REGISTER UNDER TEST?
     ;BRANCH IF ADDRESSES MATCH
     BEQ 5$ ;LOG AND REPORT ERRORS
5$: JSR PC,DUALADR ;SET FLAG WHEN ADDRESSES MATCH
     INC $TMP0 ;POINT TO NEXT REGISTER
6$: SUB #2,R1 ;SEE IF ALL REGISTERS HAVE BEEN READ
     CMP #UIPAR0,R1 ;BRANCH IF MORE TO READ
     BLOS 4$ ;NOW LOAD THE NEXT REGISTER
     ADD #2,R0 ;SEE IF THERE ARE MORE REGISTERS TO TEST
     CMP #UDPAR7,R0 ;BRANCH IF MORE REGISTERS TO TEST
     BHIS 3$ ;SET LOOP ON ERROR POINTER TO START OF TEST
     MOV #20$,SLPERR ;SEE IF THERE WERE ANY ERRORS
     TST ERRCNT ;BRANCH TO NEXT TEST IF NO ERRORS
     BR TST23 ;SAVE # OF ERRORS FOR PAR OUT
     MOV ERRCNT,$TMP5 ;SUMMARY OF DUAL ADDRESS TEST
     ERROR +31
```

023106 012737 023120 001110
 023114 012700 177640
 023120 012705 177640
 023124 004737 002260
 023130 012701 177676
 023134 012710 177777
 023140 005037 001176
 023144 011102
 023146 001406
 023150 020001
 023152 001402
 023154 004737 002334
 023160 005237 001176
 023164 162701 000002
 023170 022701 177640
 023174 101761
 023176 062700 000002
 023202 022700 177676
 023206 103344
 023210 012737 023040 001110
 023216 005737 001322
 023222 000404
 023224 013737 001322 001210
 023232 104031

2347

```
.SBTTL TEST # 23 - DUAL ADDRESS KERNAL PDR'S,ON LOADING
*****
*TEST 23      DUAL ADDRESS KERNAL PDR'S,ON LOADING
*
*THIS TEST FIRST CLEARS ALL THE KERNEL PAGE DESCRIPTOR REGISTERS,
*AND CHECKS TO SEE THAT THEY EACH HOLD ZERO. THEN, STARTING
*WITH I-SPACE DESCRIPTOR REGISTER ZERO, ONE REGISTER AT A TIME IS
*LOADED WITH A NEGATIVE ONE. ALL KERNEL DESCRIPTOR REGISTERS ARE
*NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
*INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN
*AT THE END OF THIS TEST.
*****
```

2348	023234	000004			TST23: SCOPE	
	023236	004737	002276		20\$: JSR PC,CLEANUP	:INITIALIZE THE ERROR LOCATIONS
	023242	012737	023270	001110	MOV #1\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 1\$
	023250	012705	172300		MOV #KIPDR0,R5	:PUT ADDRESS OF FIRST PDR IN R5
	023254	004737	002260		JSR PC,CLRREG	:CLEAR 16 REGISTERS POINTED TO BY R5
	023260	012700	172300		MOV #KIPDR0,R0	:PUT ADDRESS OF FIRST PDR IN R0
	023264	012701	000020		MOV #20,R1	:BRANCH COUNT IS 16 DECIMAL
	023270	011002			1\$: MOV (R0),R2	:READ PDR TO R2
	023272	001401			BEQ 2\$:BRANCH IF PDR IS 0
	023274	104030			ERROR +30	:PDR NOT ZERO
	023276	062700	000002		2\$: ADD #2,R0	:POINT TO NEXT REGISTER
	023302	077106			SOB R1,1\$:BRANCH BACK TO 1\$ 15 TIMES
					:NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE REGISTER AND	
					:READING THE REST TO SEE ONLY THAT REGISTER IS NON ZERO. (ANY	
					:DROPPED BITS WILL BE FOUND IN THE NEXT TEST).	
	023304	012737	023316	001110	MOV #3\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 3\$
	023312	012700	172300		MOV #KIPDR0,R0	:PUT ADDRESS OF FIRST PDR IN R0
	023316	012705	172300		3\$: MOV #KIPDR0,R5	:LOAD STARTING ADDRESS INTO R5
	023322	004737	002260		JSR PC,CLRREG	:CLEAR 16 REGISTERS POINTED TO BY R5
	023326	012701	172336		MOV #KDPDR7,R1	:PUT KDPDR7 ADDRESS INTO R1
	023332	012710	177777		MOV #-1,(R0)	:LOAD REGISTER UNDER TEST
	023336	005037	001176		4\$: CLR \$TMP0	:FLAG TO INDICATE THERE WAS A MATCH
	023342	011102			MOV (R1),R2	:READ ALL REGISTERS
	023344	001406			BEQ 6\$:BRANCH IF REGISTER IS 0
	023346	020001			CMP R0,R1	:IS THE ADDRESS OF NON-ZERO REGISTER THE SAME
						:AS THE REGISTER UNDER TEST?
	023350	001402			BEQ 5\$:BRANCH IF ADDRESSES MATCH
	023352	004737	002334		JSR PC,DUALADR	:LOG AND REPORT ERRORS
	023356	005237	001176		5\$: INC \$TMP0	:SET FLAG WHEN ADDRESSES MATCH
	023362	162701	000002		6\$: SUB #2,R1	:POINT TO NEXT REGISTER
	023366	022701	172300		CMP #KIPDR0,R1	:SEE IF ALL REGISTERS HAVE BEEN READ
	023372	101761			BLOS 4\$:BRANCH IF MORE TO READ
	023374	062700	000002		ADD #2,R0	:NOW LOAD THE NEXT REGISTER
	023400	022700	172336		CMP #KDPDR7,R0	:SEE IF THERE ARE MORE REGISTERS TO TEST
	023404	103344			BHIS 3\$:BRANCH IF MORE REGISTERS TO TEST
	023406	012737	023236	001110	MOV #20\$, \$LPERR	:SET LOOP ON ERROR POINTER TO START OF TEST
	023414	005737	001322		TST ERRCNT	:SEE IF THERE WERE ANY ERRORS
	023420	000404			BR TST24	:BRANCH TO NEXT TEST IF NO ERRORS
	023422	013737	001322	001210	MOV ERRCNT,\$TMP5	:SAVE # OF ERRORS FOR PDR OUT
	023430	104031			ERROR +31	:SUMMARY OF DUAL ADDRESS TEST

2352

```
.SBTTL TEST # 24 - DUAL ADDRESS SUPERVISOR PDR'S,ON LOADING
:*****
:*TEST 24      DUAL ADDRESS SUPERVISOR PDR'S,ON LOADING
:*
:*THIS TEST FIRST CLEARS ALL THE SUPERVISOR PAGE DESCRIPTOR REGISTERS,
:*AND CHECKS TO SEE THAT THEY EACH HOLD ZERO. THEN, STARTING
:*WITH I-SPACE DESCRIPTOR REGISTER ZERO, ONE REGISTER AT A TIME IS
:*LOADED WITH A NEGATIVE ONE. ALL SUPERVISOR DESCRIPTOR REGISTERS ARE
:*NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
:*INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN
:*AT THE END OF THIS TEST.
:*****
```

2353 023432 000004
 023434 004737 002276
 023440 012737 023466 001110
 023446 012705 172200
 023452 004737 002260
 023456 012700 172200
 023462 012701 000020
 023466 011002
 023470 001401
 023472 104030
 023474 062700 000002
 023500 077106

```
TST24: SCOPE
20$: JSR PC,CLEANUP ;INITIALIZE THE ERROR LOCATIONS
      MOV #1$,SLPERR ;SET LOOP ON ERROR POINTER TO 1$
      MOV #SIPDR0,R5 ;PUT ADDRESS OF FIRST PDR IN R5
      JSR PC,CLRREG ;CLEAR 16 REGISTERS POINTED TO BY R5
      MOV #SIPDR0,R0 ;PUT ADDRESS OF FIRST PDR IN R0
      MOV #20,R1 ;BRANCH COUNT IS 16 DECIMAL
1$: MOV (R0),R2 ;READ PDR TO R2
   BEQ 2$ ;BRANCH IF PDR IS 0
   ERROR +30 ;PDR NOT ZERO
2$: ADD #2,R0 ;POINT TO NEXT REGISTER
   SOB R1,1$ ;BRANCH BACK TO 1$ 15 TIMES
;NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE REGISTER AND
;READING THE REST TO SEE ONLY THAT REGISTER IS NON ZERO. (ANY
;DROPPED BITS WILL BE FOUND IN THE NEXT TEST).
      MOV #3$,SLPERR ;SET LOOP ON ERROR POINTER TO 3$
      MOV #SIPDR0,R0 ;PUT ADDRESS OF FIRST PDR IN R0
3$: MOV #SIPDR0,R5 ;LOAD STARTING ADDRESS INTO R5
   JSR PC,CLRREG ;CLEAR 16 REGISTERS POINTED TO BY R5
   MOV #SDPDR7,R1 ;PUT SDPDR7 ADDRESS INTO R1
   MOV #-1,(R0) ;LOAD REGISTER UNDER TEST
4$: CLR $TMP0 ;FLAG TO INDICATE THERE WAS A MATCH
   MOV (R1),R2 ;READ ALL REGISTERS
   BEQ 6$ ;BRANCH IF REGISTER IS 0
   CMP R0,R1 ;IS THE ADDRESS OF NON-ZERO REGISTER THE SAME
   ;AS THE REGISTER UNDER TEST?
   BEQ 5$ ;BRANCH IF ADDRESSES MATCH
   JSR PC,DUALADR ;LOG AND REPORT ERRORS
5$: INC $TMP0 ;SET FLAG WHEN ADDRESSES MATCH
6$: SUB #2,R1 ;POINT TO NEXT REGISTER
   CMP #SIPDR0,R1 ;SEE IF ALL REGISTERS HAVE BEEN READ
   BLOS 4$ ;BRANCH IF MORE TO READ
   ADD #2,R0 ;NOW LOAD THE NEXT REGISTER
   CMP #SDPDR7,R0 ;SEE IF THERE ARE MORE REGISTERS TO TEST
   BHIS 3$ ;BRANCH IF MORE REGISTERS TO TEST
   MOV #20$,SLPERR ;SET LOOP ON ERROR POINTER TO START OF TEST
   TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
   BR TST25 ;BRANCH TO NEXT TEST IF NO ERRORS
023502 012737 023514 001110 ;MOV #3$,SLPERR ;SET LOOP ON ERROR POINTER TO 3$
023510 012700 172200 ;MOV #SIPDR0,R0 ;PUT ADDRESS OF FIRST PDR IN R0
023514 012705 172200 ;MOV #SIPDR0,R5 ;LOAD STARTING ADDRESS INTO R5
023520 004737 002260 ;JSR PC,CLRREG ;CLEAR 16 REGISTERS POINTED TO BY R5
023524 012701 172236 ;MOV #SDPDR7,R1 ;PUT SDPDR7 ADDRESS INTO R1
023530 012710 177777 ;MOV #-1,(R0) ;LOAD REGISTER UNDER TEST
023534 005037 001176 ;4$: CLR $TMP0 ;FLAG TO INDICATE THERE WAS A MATCH
023540 011102 ;MOV (R1),R2 ;READ ALL REGISTERS
023542 001406 ;BEQ 6$ ;BRANCH IF REGISTER IS 0
023544 020001 ;CMP R0,R1 ;IS THE ADDRESS OF NON-ZERO REGISTER THE SAME
;AS THE REGISTER UNDER TEST?
;BRANCH IF ADDRESSES MATCH
023546 001402 ;BEQ 5$ ;BRANCH IF ADDRESSES MATCH
023550 004737 002334 ;JSR PC,DUALADR ;LOG AND REPORT ERRORS
023554 005237 001176 ;5$: INC $TMP0 ;SET FLAG WHEN ADDRESSES MATCH
023560 162701 000002 ;6$: SUB #2,R1 ;POINT TO NEXT REGISTER
023564 022701 172200 ;CMP #SIPDR0,R1 ;SEE IF ALL REGISTERS HAVE BEEN READ
023570 101761 ;BLOS 4$ ;BRANCH IF MORE TO READ
023572 062700 000002 ;ADD #2,R0 ;NOW LOAD THE NEXT REGISTER
023576 022700 172236 ;CMP #SDPDR7,R0 ;SEE IF THERE ARE MORE REGISTERS TO TEST
023602 103344 ;BHIS 3$ ;BRANCH IF MORE REGISTERS TO TEST
023604 012737 023434 001110 ;MOV #20$,SLPERR ;SET LOOP ON ERROR POINTER TO START OF TEST
023612 005737 001322 ;TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
023616 000404 ;BR TST25 ;BRANCH TO NEXT TEST IF NO ERRORS
023620 013737 001322 001210 ;MOV ERRCNT,$TMP5 ;SAVE # OF ERRORS FOR PDR OUT
023626 104031 ;ERROR +31 ;SUMMARY OF DUAL ADDRESS TEST
```

2357

```
.SBTTL TEST # 25 - DUAL ADDRESS USER PDR'S,ON LOADING
:*****
:TEST 25      DUAL ADDRESS USER PDR'S,ON LOADING
:*
:*THIS TEST FIRST CLEARS ALL THE USER PAGE DESCRIPTOR REGISTERS,
:*AND CHECKS TO SEE THAT THEY EACH HOLD ZERO. THEN, STARTING
:*WITH I-SPACE DESCRIPTOR REGISTER ZERO, ONE REGISTER AT A TIME IS
:*LOADED WITH A NEGATIVE ONE. ALL USER DESCRIPTOR REGISTERS ARE
:*NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
:*INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN
:*AT THE END OF THIS TEST.
:*****
```

2358 023630 000004
 023632 004737 002276
 023636 012737 023664 001110
 023644 012705 177600
 023650 004737 002260
 023654 012700 177600
 023660 012701 000020
 023664 011002
 023666 001401
 023670 104030
 023672 062700 000002
 023676 077106

```
TST25: SCOPE
20$: JSR PC,CLEANUP ;INITIALIZE THE ERROR LOCATIONS
      MOV #1$, $LPERR ;SET LOOP ON ERROR POINTER TO 1$
      MOV #UIPDRO,R5 ;PUT ADDRESS OF FIRST PDR IN R5
      JSR PC,CLRREG ;CLEAR 16 REGISTERS POINTED TO BY R5
      MOV #UIPDRO,R0 ;PUT ADDRESS OF FIRST PDR IN R0
      MOV #20,R1 ;BRANCH COUNT IS 16 DECIMAL
1$: MOV (R0),R2 ;READ PDR TO R2
   BEQ 2$ ;BRANCH IF PDR IS 0
   ERROR +30 ;PDR NOT ZERO
2$: ADD #2,R0 ;POINT TO NEXT REGISTER
   SOB R1,1$ ;BRANCH BACK TO 1$ 15 TIMES
;NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE REGISTER AND
;READING THE REST TO SEE ONLY THAT REGISTER IS NON ZERO. (ANY
;DROPPED BITS WILL BE FOUND IN THE NEXT TEST).
      MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
      MOV #UIPDRO,R0 ;PUT ADDRESS OF FIRST PDR IN R0
3$: MOV #UIPDRO,R5 ;LOAD STARTING ADDRESS INTO R5
   JSR PC,CLRREG ;CLEAR 16 REGISTERS POINTED TO BY R5
   MOV #UDPDR7,R1 ;PUT UDPDR7 ADDRESS INTO R1
   MOV #-1,(R0) ;LOAD REGISTER UNDER TEST
4$: CLR $TMP0 ;FLAG TO INDICATE THERE WAS A MATCH
   MOV (R1),R2 ;READ ALL REGISTERS
   BEQ 6$ ;BRANCH IF REGISTER IS 0
   CMP R0,R1 ;IS THE ADDRESS OF NON-ZERO REGISTER THE SAME
   ;AS THE REGISTER UNDER TEST?
   BEQ 5$ ;BRANCH IF ADDRESSES MATCH
   JSR PC,DUALADR ;LOG AND REPORT ERRORS
5$: INC $TMP0 ;SET FLAG WHEN ADDRESSES MATCH
6$: SUB #2,R1 ;POINT TO NEXT REGISTER
   CMP #UIPDRO,R1 ;SEE IF ALL REGISTERS HAVE BEEN READ
   BLOS 4$ ;BRANCH IF MORE TO READ
   ADD #2,R0 ;NOW LOAD THE NEXT REGISTER
   CMP #UDPDR7,R0 ;SEE IF THERE ARE MORE REGISTERS TO TEST
   BHIS 3$ ;BRANCH IF MORE REGISTERS TO TEST
   MOV #20$, $LPERR ;SET LOOP ON ERROR POINTER TO START OF TEST
   TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
   BR TST26 ;BRANCH TO NEXT TEST IF NO ERRORS
   MOV ERRCNT,$TMP5 ;SAVE # OF ERRORS FOR PDR OUT
   ERROR +31 ;SUMMARY OF DUAL ADDRESS TEST
```

023700 012737 023712 001110
 023706 012700 177600
 023712 012705 177600
 023716 004737 002260
 023722 012701 177636
 023726 012710 177777
 023732 005037 001176
 023736 011102
 023740 001406
 023742 020001
 023744 001402
 023746 004737 002334
 023752 005237 001176
 023756 162701 000002
 023762 022701 177600
 023766 101761
 023770 062700 000002
 023774 022700 177636
 024000 103344
 024002 012737 023632 001110
 024010 005737 001322
 024014 000404
 024016 013737 001322 001210
 024024 104031

2374

```
.SBTTL TEST # 26 - COUNT PATTERN IN KERNAL PAR'S
:*****
:*TEST 26      COUNT PATTERN IN KERNAL PAR'S
:*
:*THIS TEST RUNS A COUNT PATTERN THROUGH THE KERNAL PAGE ADDRESS REGISTERS.
:*IF THE COUNT PATTERN DOES NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS,
:*DATA PATTERN, AND BAD DATA ARE REPORTED.  AT THE END OF THE TEST A SUMMARY
:*OF ERRORS IS GIVEN.
:*****
```

2375	024026	000004				TST26: SCOPE	
	024030	004737	002276			20\$: JSR PC,CLEANUP	:INITIALIZE ERROR LOCATIONS
	024034	012737	024044	001110		MOV #1\$,SLPERR	:SET LOOP ON ERROR POINTER TO 1\$
	024042	005001				CLR R1	:CLEAR REGISTER TO HOLD COUNT PATTERN
	024044	012700	172340		1\$:	MOV #KIPARO,R0	:PUT ADDRESS OF FIRST REGISTER INTO R0
	024050	010110			2\$:	MOV (R0),R2	:LOAD COUNT INTO REGISTER
	024052	011002				MOV R1,R4	:READ REGISTER BACK TO R2
	024054	010104				MOV R4,R2	:PUT PATTERN IN R4
	024056	020402				CMP R4,R2	:SEE IF DATA MATCHES PATTERN
	024060	001402				BEQ 3\$:BRANCH IF DATA IS GOOD
	024062	004737	002400			JSR PC,PARCOUNT	:LOG AND REPORT COUNT ERROR
	024066	062700	000002		3\$:	ADD #2,R0	:POINT TO NEXT REGISTER
	024072	022700	172376			CMP #KDPAR7,R0	:SEE IF YOU PASSED THE KDPAR7 PAR
	024076	103364				BHIS 2\$:BRANCH IF MORE TEST
	024100	022701	177777			CMP #177777,R1	:SEE IF COUNT HAS REACHED 177777
	024104	001403				BEQ 4\$:BRANCH IF SO
	024106	062701	000401			ADD #401,R1	:INCREASE COUNT PATTERN
	024112	000754				BR 1\$:BRANCH TO CONTINUE TEST
	024114	012737	024030	001110	4\$:	MOV #20\$,SLPERR	:SET LOOP POINTER TO START OF TEST
	024122	005737	001322			TST ERRCNT	:SEE IF THERE WERE ANY ERRORS
	024126	000404				BR TST27	:BRANCH TO NEXT TEST IF NO ERRORS
	024130	013737	001322	001210		MOV ERRCNT,\$TMP5	:SAVE # OF ERRORS FOR TYPEOUT
	024136	104032				ERROR +32	:SUMMARY OF COUNT PATTERN FAILURES

2379

```
.SBTTL TEST # 27 - COUNT PATTERN IN SUPERVISOR PAR'S
:*****
:*TEST 27      COUNT PATTERN IN SUPERVISOR PAR'S
:*
:*THIS TEST RUNS A COUNT PATTERN THROUGH THE SUPERVISOR PAGE ADDRESS REGISTERS.
:*IF THE COUNT PATTERN DOES NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS,
:*DATA PATTERN, AND BAD DATA ARE REPORTED.  AT THE END OF THE TEST A SUMMARY
:*OF ERRORS IS GIVEN.
:*****
```

```
2380 024140 000004
024142 004737 002276
024146 012737 024156 001110
024154 005001
024156 012700 172240
024162 010110
024164 011002
024166 010104
024170 020402
024172 001402
024174 004737 002400
024200 062700 000002
024204 022700 172276
024210 103364
024212 022701 177777
024216 001403
024220 062701 000401
024224 000754
024226 012737 024142 001110
024234 005737 001322
024240 000404
024242 013737 001322 001210
024250 104032
```

```
TST27: SCOPE
20$: JSR PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
      MOV #1$, $LPERR   ;SET LOOP ON ERROR POINTER TO 1$
      CLR R1            ;CLEAR REGISTER TO HOLD COUNT PATTERN
1$:   MOV #SIPARO,R0    ;PUT ADDRESS OF FIRST REGISTER INTO R0
2$:   MOV R1,(R0)       ;LOAD COUNT INTO REGISTER
      MOV (R0),R2       ;READ REGISTER BACK TO R2
      MOV R1,R4         ;PUT PATTERN IN R4
      CMP R4,R2        ;SEE IF DATA MATCHES PATTERN
      BEQ 3$           ;BRANCH IF DATA IS GOOD
      JSR PC,PARCOUNT ;LOG AND REPORT COUNT ERROR
3$:   ADD #2,R0         ;POINT TO NEXT REGISTER
      CMP #SDPAR7,R0   ;SEE IF YOU PASSED THE SDPAR7 PAR
      BHIS 2$          ;BRANCH IF MORE TEST
      CMP #177777,R1   ;SEE IF COUNT HAS REACHED 177777
      BEQ 4$           ;BRANCH IF SO
      ADD #401,R1      ;INCREASE COUNT PATTERN
      BR 1$            ;BRANCH TO CONTINUE TEST
4$:   MOV #20$, $LPERR  ;SET LOOP POINTER TO START OF TEST
      TST ERRCNT       ;SEE IF THERE WERE ANY ERRORS
      BR TST30         ;BRANCH TO NEXT TEST IF NO ERRORS
      MOV ERRCNT,$TMP5 ;SAVE # OF ERRORS FOR TYPEOUT
      ERROR +32        ;SUMMARY OF COUNT PATTERN FAILURES
```

2384

```
.SBTTL TEST # 30 - COUNT PATTERN IN USER PAR'S
:*****
:*TEST 30      COUNT PATTERN IN USER PAR'S
:*
:*THIS TEST RUNS A COUNT PATTERN THROUGH THE USER PAGE ADDRESS REGISTERS.
:*IF THE COUNT PATTERN DOES NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS,
:*DATA PATTERN, AND BAD DATA ARE REPORTED.  AT THE END OF THE TEST A SUMMARY
:*OF ERRORS IS GIVEN.
:*****
```

```
2385 024252 000004
      024254 004737 002276
      024260 012737 024270 001110
      024266 005001
      024270 012700 172340
      024274 010110
      024276 011002
      024300 010104
      024302 020402
      024304 001402
      024306 004737 002400
      024312 062700 000002
      024316 022700 172376
      024322 103364
      024324 022701 177777
      024330 001403
      024332 062701 000401
      024336 000754
      024340 012737 024254 001110
      024346 005737 001322
      024352 000404
      024354 013737 001322 001210
      024362 104032
```

```
TST30: SCOPE
20$: JSR PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
      MOV #1$,SLPERR    ;SET LOOP ON ERROR POINTER TO 1$
      CLR R1            ;CLEAR REGISTER TO HOLD COUNT PATTERN
1$:  MOV #KIPAR0,R0     ;PUT ADDRESS OF FIRST REGISTER INTO R0
2$:  MOV R1,(R0)        ;LOAD COUNT INTO REGISTER
      MOV (R0),R2       ;READ REGISTER BACK TO R2
      MOV R1,R4         ;PUT PATTERN IN R4
      CMP R4,R2         ;SEE IF DATA MATCHES PATTERN
      BEQ 3$           ;BRANCH IF DATA IS GOOD
      JSR PC,PARCOUNT ;LOG AND REPORT COUNT ERROR
3$:  ADD #2,R0          ;POINT TO NEXT REGISTER
      CMP #KDPAR7,R0   ;SEE IF YOU PASSED THE KDPAR7 PAR
      BHIS 2$          ;BRANCH IF MORE TEST
      CMP #177777,R1   ;SEE IF COUNT HAS REACHED 177777
      BEQ 4$           ;BRANCH IF SO
      ADD #401,R1      ;INCREASE COUNT PATTERN
      BR 1$            ;BRANCH TO CONTINUE TEST
4$:  MOV #20$,SLPERR    ;SET LOOP POINTER TO START OF TEST
      TST ERRCNT       ;SEE IF THERE WERE ANY ERRORS
      BR TST31         ;BRANCH TO NEXT TEST IF NO ERRORS
      MOV ERRCNT,$TMP5 ;SAVE # OF ERRORS FOR TYPEOUT
      ERROR +32        ;SUMMARY OF COUNT PATTERN FAILURES
```

2389

```
.SBTTL TEST # 31 - COUNT PATTERN IN KERNAL PDR'S
:*****
:*TEST 31      COUNT PATTERN IN KERNAL PDR'S
:*
:*THIS TEST RUNS A COUNT PATTERN THROUGH THE KERNAL PAGE DESCRIPTOR REGISTERS.
:*SINCE BITS <05:04> AND <0> ARE NOT IMPLEMENTED, AND BITS <07:06>
:*CANNOT BE SET BY DIRECT LOAD, THEY ARE MASKED OUT OF THE DATA COMPARE.
:*THESE BITS ARE STILL SENT TO THE DESCRIPTOR REGISTER TO FIND BAD ETCHES.
:*IF THE COUNT PATTERN DOES NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS,
:*DATA PATTERN, AND BAD DATA ARE REPORTED.  AT THE END OF THE TEST A SUMMARY
:*OF ERRORS IS GIVEN.
:*****
```

```
TST31: SCOPE
20$: JSR PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
      MOV #1$, $LPERR   ;SET LOOP ON ERROR POINTER TO 1$
      CLR R1            ;CLEAR REGISTER TO HOLD COUNT PATTERN
1$:   MOV #KIPDRO,R0    ;PUT ADDRESS OF FIRST REGISTER INTO R0
2$:   MOV R1,(R0)       ;LOAD COUNT INTO REGISTER
      MOV (R0),R2       ;READ REGISTER BACK TO R2
      MOV R1,R4         ;PUT PATTERN IN R4
      BIC #000361,R4    ;CLEAR BITS NOT FOUND IN REGISTER
      CMP R4,R2         ;SEE IF DATA MATCHES PATTERN
      BEQ 3$           ;BRANCH IF DATA IS GOOD
      JSR PC,PARCOUNT ;LOG AND REPORT COUNT ERROR
3$:   ADD #2,R0         ;POINT TO NEXT REGISTER
      CMP #KDPDR7,R0   ;SEE IF YOU PASSED THE KDPDR7 PDR
      BHS 2$          ;BRANCH IF MORE TEST
      CMP #177777,R1   ;SEE IF COUNT HAS REACHED 177777
      BEQ 4$          ;BRANCH IF SO
      ADD #401,R1      ;INCREASE COUNT PATTERN
      BR 1$           ;BRANCH TO CONTINUE TEST
4$:   MOV #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
      TST ERRCNT       ;SEE IF THERE WERE ANY ERRORS
      BR TST32        ;BRANCH TO NEXT TEST IF NO ERRORS
      MOV ERRCNT,$TMP5 ;SAVE # OF ERRORS FOR TYPEOUT
      ERROR +32       ;SUMMARY OF COUNT PATTERN FAILURES
```

```
024364 000004
2390 024366 004737 002276
      024372 012737 024402 001110
      024400 005001
      024402 012700 172300
      024406 010110
      024410 011002
      024412 010104
      024414 042704 000361
      024420 020402
      024422 001402
      024424 004737 002400
      024430 062700 000002
      024434 022700 172336
      024440 103362
      024442 022701 177777
      024446 001403
      024450 062701 000401
      024454 000752
      024456 012737 024366 001110
      024464 005737 001322
      024470 000404
      024472 013737 001322 001210
      024500 104032
```

2394

```
.SBTTL TEST # 32 - COUNT PATTERN IN SUPERVISOR PDR'S
*****
*TEST 32      COUNT PATTERN IN SUPERVISOR PDR'S
*
*THIS TEST RUNS A COUNT PATTERN THROUGH THE SUPERVISOR PAGE DESCRIPTOR REGISTERS.
*SINCE BITS <05:04> AND <0> ARE NOT IMPLEMENTED, AND BITS <07:06>
*CANNOT BE SET BY DIRECT LOAD, THEY ARE MASKED OUT OF THE DATA COMPARE.
*THESE BITS ARE STILL SENT TO THE DESCRIPTOR REGISTER TO FIND BAD ETCHES.
*IF THE COUNT PATTERN DOES NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS,
*DATA PATTERN, AND BAD DATA ARE REPORTED.  AT THE END OF THE TEST A SUMMARY
*OF ERRORS IS GIVEN.
*****
```

```
TST32:  SCOPE
20$:   JSR    PC,CLEANUP      ;INITIALIZE ERROR LOCATIONS
      MOV    #1$, $LPERR    ;SET LOOP ON ERROR POINTER TO 1$
      CLR    R1             ;CLEAR REGISTER TO HOLD COUNT PATTERN
1$:   MOV    #SDPDR0,R0     ;PUT ADDRESS OF FIRST REGISTER INTO R0
2$:   MOV    R1,(R0)        ;LOAD COUNT INTO REGISTER
      MOV    (R0),R2        ;READ REGISTER BACK TO R2
      MOV    R1,R4         ;PUT PATTERN IN R4
      BIC    #000361,R4    ;CLEAR BITS NOT FOUND IN REGISTER
      CMP    R4,R2         ;SEE IF DATA MATCHES PATTERN
      BEQ    3$           ;BRANCH IF DATA IS GOOD
      JSR    PC,PARCOUNT  ;LOG AND REPORT COUNT ERROR
3$:   ADD    #2,R0         ;POINT TO NEXT REGISTER
      CMP    #SDPDR7,R0    ;SEE IF YOU PASSED THE SDPDR7 PDR
      BHS   2$           ;BRANCH IF MORE TEST
      CMP    #177777,R1    ;SEE IF COUNT HAS REACHED 177777
      BEQ    4$           ;BRANCH IF SO
      ADD    #401,R1       ;INCREASE COUNT PATTERN
      BR    1$           ;BRANCH TO CONTINUE TEST
4$:   MOV    #20$, $LPERR  ;SET LOOP POINTER TO START OF TEST
      TST    ERRCNT        ;SEE IF THERE WERE ANY ERRORS
      BR    TST33        ;BRANCH TO NEXT TEST IF NO ERRORS
      MOV    ERRCNT,$TMP5  ;SAVE # OF ERRORS FOR TYPEOUT
      ERROR +32          ;SUMMARY OF COUNT PATTERN FAILURES
```

```
2395 024502 000004
      024504 004737 002276
      024510 012737 024520 001110
      024516 005001
      024520 012700 172200
      024524 010110
      024526 011002
      024530 010104
      024532 042704 000361
      024536 020402
      024540 001402
      024542 004737 002400
      024546 062700 000002
      024552 022700 172236
      024556 103362
      024560 022701 177777
      024564 001403
      024566 062701 000401
      024572 000752
      024574 012737 024504 001110
      024602 005737 001322
      024606 000404
      024610 013737 001322 001210
      024616 104032
```


2399

```
.SBTTL TEST # 33 - COUNT PATTERN IN USER PDR'S
:*****
:*TEST 33      COUNT PATTERN IN USER PDR'S
:*
:*THIS TEST RUNS A COUNT PATTERN THROUGH THE USER PAGE ADDRESS REGISTERS.
:*SINCE BITS <05:04> AND <0> ARE NOT IMPLEMENTED, AND BITS <07:06>
:*CANNOT BE SET BY DIRECT LOAD, THEY ARE MASKED OUT OF THE DATA COMPARE.
:*THESE BITS ARE STILL SENT TO THE DESCRIPTOR REGISTER TO FIND BAD ETCHES.
:*IF THE COUNT PATTERN DOES NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS,
:*DATA PATTERN, AND BAD DATA ARE REPORTED.  AT THE END OF THE TEST A SUMMARY
:*OF ERRORS IS GIVEN.
:*****
```

2400	024620	000004			TST33: SCOPE	
	024622	004737	002276		20\$: JSR PC,CLEANUP	:INITIALIZE ERROR LOCATIONS
	024626	012737	024636	001110	MOV #1\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 1\$
	024634	005001			CLR R1	:CLEAR REGISTER TO HOLD COUNT PATTERN
	024636	012700	177600		1\$: MOV #UIPDR0,R0	:PUT ADDRESS OF FIRST REGISTER INTO R0
	024642	010110			2\$: MOV R1,(R0)	:LOAD COUNT INTO REGISTER
	024644	011002			MOV (R0),R2	:READ REGISTER BACK TO R2
	024646	010104			MOV R1,R4	:PUT PATTERN IN R4
	024650	042704	000361		BIC #000361,R4	:CLEAR BITS NOT FOUND IN REGISTER
	024654	020402			CMP R4,R2	:SEE IF DATA MATCHES PATTERN
	024656	001402			BEQ 3\$:BRANCH IF DATA IS GOOD
	024660	004737	002400		JSR PC,PARCOUNT	:LOG AND REPORT COUNT ERROR
	024664	062700	000002		3\$: ADD #2,R0	:POINT TO NEXT REGISTER
	024670	022700	177636		CMP #UDPDR7,R0	:SEE IF YOU PASSED THE UDPDR7 PDR
	024674	103362			BHIS 2\$:BRANCH IF MORE TEST
	024676	022701	177777		CMP #177777,R1	:SEE IF COUNT HAS REACHED 177777
	024702	001403			BEQ 4\$:BRANCH IF SO
	024704	062701	000401		ADD #401,R1	:INCREASE COUNT PATTERN
	024710	000752			BR 1\$:BRANCH TO CONTINUE TEST
	024712	012737	024622	001110	4\$: MOV #20\$, \$LPERR	:SET LOOP POINTER TO START OF TEST
	024720	005737	001322		TST ERRCNT	:SEE IF THERE WERE ANY ERRORS
	024724	000404			BR TST34	:BRANCH TO NEXT TEST IF NO ERRORS
	024726	013737	001322	001210	MOV ERRCNT,\$TMP5	:SAVE # OF ERRORS FOR TIMEOUT
	024734	104032			ERROR +32	:SUMMARY OF COUNT PATTERN FAILURES

2411

```
.SBTTL TEST # 34 - BYTE ADDRESSING OF KERNAL PAR'S
:*****
:*TEST 34      BYTE ADDRESSING OF KERNAL PAR'S
:*
:*      THIS TEST CHECK THE LOGIC ASSOCIATED WITH BYTE ADDRESSING
:*      OF KERNAL PAGE ADDRESS REGISTERS.  IT CAN BE ASSUMED THAT IF
:*      ONE REGISTER WORKS, THEY ALL WORK BECAUSE THE REGISTERS
:*      HAVE ALL BEEN TESTED.
:*****
```

```
TST34: SCOPE
2412 024736 000004
024740 012737 024756 001110 20$:  MOV    #11$, $LPERR    ;SET LOOP ON ERROR POINTER TO 11$
024746 012702 000014          MOV    #14, R2        ;PUT EXPECTED DATA IN R2
024752 012700 172340          MOV    #KIPAR0, R0    ;PUT ADDRESS OF REGISTER IN R0
024756 005037 172340          11$:  CLR    KIPAR0        ;CLEAR THE REGISTER UNDER TEST
024762 112710 172014          MOVB   #172014, (R0)  ;LOAD LOWER BYTE OF REGISTER
024766 013701 172340          MOV    KIPAR0, R1    ;READ REGISTER INTO R1
024772 020102          CMP    R1, R2        ;SEE IF ONLY LOWER BYTE WAS WRITTEN
024774 001401          BEQ    1$            ;BRANCH IF DATA MATCHES
024776 104033          ERROR  +33          ;DIDN'T LOAD CORRECT BYTE
025000 012737 025016 001110 1$:  MOV    #12$, $LPERR    ;SET LOOP ON ERROR POINTER TO 12$
025006 012700 172341          MOV    #KIPAR0+1, R0 ;POINT TO UPPER BYTE OF REGISTER
025012 012702 052014          MOV    #52014, R2    ;LOAD EXPECTED DATA INTO R2
025016 112710 000124          12$:  MOVB   #124, (R0)    ;WRITE UPPER BYTE OF REGISTER
025022 013701 172340          MOV    KIPAR0, R1    ;READ REGISTER INTO R1
025026 020102          CMP    R1, R2        ;SEE IF ONLY UPPER BYTE WAS WRITTEN
025030 001401          BEQ    2$            ;BRANCH TO EXIT IF CORRECT
025032 104033          ERROR  +33          ;DIDN'T LOAD CORRECT BYTE
025034 012737 024740 001110 2$:  MOV    #20$, $LPERR    ;SET LOOP POINTER TO START OF TEST
```

2416

```
.SBTTL TEST # 35 - BYTE ADDRESSING OF SUPERVISOR PAR'S
:*****
:*TEST 35      BYTE ADDRESSING OF SUPERVISOR PAR'S
:*
:*      THIS TEST CHECK THE LOGIC ASSOCIATED WITH BYTE ADDRESSING
:*      OF SUPERVISOR PAGE ADDRESS REGISTERS.  IT CAN BE ASSUMED THAT IF
:*      ONE REGISTER WORKS, THEY ALL WORK BECAUSE THE REGISTERS
:*      HAVE ALL BEEN TESTED.
:*****
```

```
TST35: SCOPE
2417 025042 000004
025044 012737 025062 001110 20$:  MOV    #11$, $LPERR  ;SET LOOP ON ERROR POINTER TO 11$
025052 012702 000014          MOV    #14, R2      ;PUT EXPECTED DATA IN R2
025056 012700 172240          MOV    #SIPARO, R0  ;PUT ADDRESS OF REGISTER IN R0
025062 005037 172240          11$:  CLR    SIPARO      ;CLEAR THE REGISTER UNDER TEST
025066 112710 172014          MOVB  #172014, (R0) ;LOAD LOWER BYTE OF REGISTER
025072 013701 172240          MOV    SIPARO, R1   ;READ REGISTER INTO R1
025076 020102          CMP    R1, R2      ;SEE IF ONLY LOWER BYTE WAS WRITTEN
025100 001401          BEQ    1$          ;BRANCH IF DATA MATCHES
025102 104033          ERROR  +33        ;DIDN'T LOAD CORRECT BYTE
025104 012737 025122 001110 1$:  MOV    #12$, $LPERR ;SET LOOP ON ERROR POINTER TO 12$
025112 012700 172241          MOV    #SIPARO+1, R0 ;POINT TO UPPER BYTE OF REGISTER
025116 012702 052014          MOV    #52014, R2   ;LOAD EXPECTED DATA INTO R2
025122 112710 000124          12$:  MOVB  #124, (R0)   ;WRITE UPPER BYTE OF REGISTER
025126 013701 172240          MOV    SIPARO, R1   ;READ REGISTER INTO R1
025132 020102          CMP    R1, R2      ;SEE IF ONLY UPPER BYTE WAS WRITTEN
025134 001401          BEQ    2$          ;BRANCH TO EXIT IF CORRECT
025136 104033          ERROR  +33        ;DIDN'T LOAD CORRECT BYTE
025140 012737 025044 001110 2$:  MOV    #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
```

2421

```
.SBTTL TEST # 36 - BYTE ADDRESSING OF USER PAR'S
:*****
:*TEST 36      BYTE ADDRESSING OF USER PAR'S
:*
:*      THIS TEST CHECK THE LOGIC ASSOCIATED WITH BYTE ADDRESSING
:*      OF USER PAGE ADDRESS REGISTERS.  IT CAN BE ASSUMED THAT IF
:*      ONE REGISTER WORKS, THEY ALL WORK BECAUSE THE REGISTERS
:*      HAVE ALL BEEN TESTED.
:*****
```

```
2422 025146 000004
025150 012737 025166 001110 20$:  TST36: SCOPE
025156 012702 000014          MOV #11$, $LPERR ;SET LOOP ON ERROR POINTER TO 11$
025162 012700 177640          MOV #14, R2 ;PUT EXPECTED DATA IN R2
025166 005037 177640          MOV #UIPAR0, R0 ;PUT ADDRESS OF REGISTER IN R0
025172 112710 172014          11$: CLR UIPAR0 ;CLEAR THE REGISTER UNDER TEST
025176 013701 177640          MOVB #172014, (R0) ;LOAD LOWER BYTE OF REGISTER
025202 020102          MOV UIPAR0, R1 ;READ REGISTER INTO R1
025204 001401          CMP R1, R2 ;SEE IF ONLY LOWER BYTE WAS WRITTEN
025206 104033          BEQ 1$ ;BRANCH IF DATA MATCHES
025210 012737 025226 001110 1$: MOV #12$, $LPERR ;DIDN'T LOAD CORRECT BYTE
025216 012700 177641          MOV #UIPAR0+1, R0 ;SET LOOP ON ERROR POINTER TO 12$
025222 012702 052014          MOV #52014, R2 ;POINT TO UPPER BYTE OF REGISTER
025226 112710 000124          12$: MOVB #124, (R0) ;LOAD EXPECTED DATA INTO R2
025232 013701 177640          MOV UIPAR0, R1 ;WRITE UPPER BYTE OF REGISTER
025236 020102          CMP R1, R2 ;READ REGISTER INTO R1
025240 001401          BEQ 2$ ;SEE IF ONLY UPPER BYTE WAS WRITTEN
025242 104033          ERROR +33 ;BRANCH TO EXIT IF CORRECT
025244 012737 025150 001110 2$: MOV #20$, $LPERR ;DIDN'T LOAD CORRECT BYTE
;SET LOOP POINTER TO START OF TEST
```

2426

```

.SBTTL TEST # 37 - BYTE ADDRESSING OF KERNAL PDR'S
*****
:TEST 37      BYTE ADDRESSING OF KERNAL PDR'S
:
: THIS TEST CHECK THE LOGIC ASSOCIATED WITH BYTE ADDRESSING
: OF KERNAL PAGE DESCRIPTOR REGISTERS. IT CAN BE ASSUMED THAT IF
: ONE REGISTER WORKS, THEY ALL WORK BECAUSE THE REGISTERS
: HAVE ALL BEEN TESTED.
*****
    
```

```

2427 025252 000004
025254 012737 025272 001110 20$:  MOV    #11$, $LPERR      ;SET LOOP ON ERROR POINTER TO 11$
025262 012702 000014          MOV    #14, R2          ;PUT EXPECTED DATA IN R2
025266 012700 172300          MOV    #KIPDRO, R0      ;PUT ADDRESS OF REGISTER IN R0
025272 005037 172300          11$:  CLR    KIPDRO          ;CLEAR THE REGISTER UNDER TEST
025276 112710 172014          MOVB   #172014, (R0)    ;LOAD LOWER BYTE OF REGISTER
025302 013701 172300          MOV    KIPDRO, R1      ;READ REGISTER INTO R1
025306 020102          CMP    R1, R2          ;SEE IF ONLY LOWER BYTE WAS WRITTEN
025310 001401          BEQ    1$              ;BRANCH IF DATA MATCHES
025312 104033          ERROR  +33            ;DIDN'T LOAD CORRECT BYTE
025314 012737 025332 001110 1$:  MOV    #12$, $LPERR      ;SET LOOP ON ERROR POINTER TO 12$
025322 012700 172301          MOV    #KIPDRO+1, R0   ;POINT TO UPPER BYTE OF REGISTER
025326 012702 052014          MOV    #52014, R2      ;LOAD EXPECTED DATA INTO R2
025332 112710 000124          12$:  MOVB   #124, (R0)      ;WRITE UPPER BYTE OF REGISTER
025336 013701 172300          MOV    KIPDRO, R1      ;READ REGISTER INTO R1
025342 020102          CMP    R1, R2          ;SEE IF ONLY UPPER BYTE WAS WRITTEN
025344 001401          BEQ    2$              ;BRANCH TO EXIT IF CORPECT
025346 104033          ERROR  +33            ;DIDN'T LOAD CORRECT BYTE
025350 012737 025254 001110 2$:  MOV    #20$, $LPERR     ;SET LOOP POINTER TO START OF TEST
    
```

2431

```
.SBTTL TEST # 40 - BYTE ADDRESSING OF SUPERVISOR PDR'S
:*****
:*TEST 40      BYTE ADDRESSING OF SUPERVISOR PDR'S
:*
:*      THIS TEST CHECK THE LOGIC ASSOCIATED WITH BYTE ADDRESSING
:*      OF KERNAL PAGE DESCRIPTOR REGISTERS.  IT CAN BE ASSUMED THAT IF
:*      ONE REGISTER WORKS, THEY ALL WORK BECAUSE THE REGISTERS
:*      HAVE ALL BEEN TESTED.
:*****
```

```
2432 025356 000004
025360 012737 025376 001110 20$:  MOV #11$, $LPERR ;SET LOOP ON ERROR POINTER TO 11$
025366 012702 000014          MOV #14, R2 ;PUT EXPECTED DATA IN R2
025372 012700 172200          MOV #SIPDR0, R0 ;PUT ADDRESS OF REGISTER IN R0
025376 005037 172200          11$: CLR SIPDR0 ;CLEAR THE REGISTER UNDER TEST
025402 112710 172014          MOV #172014, (R0) ;LOAD LOWER BYTE OF REGISTER
025406 013701 172200          MOV SIPDR0, R1 ;READ REGISTER INTO R1
025412 020102          CMP R1, R2 ;SEE IF ONLY LOWER BYTE WAS WRITTEN
025414 001401          BEQ 1$ ;BRANCH IF DATA MATCHES
025416 104033          ERROR +33 ;DIDN'T LOAD CORRECT BYTE
025420 012737 025436 001110 1$:  MOV #12$, $LPERR ;SET LOOP ON ERROR POINTER TO 12$
025426 012700 172201          MOV #SIPDR0+1, R0 ;POINT TO UPPER BYTE OF REGISTER
025432 012702 052014          MOV #52014, R2 ;LOAD EXPECTED DATA INTO R2
025436 112710 000124          12$: MOV #124, (R0) ;WRITE UPPER BYTE OF REGISTER
025442 013701 172200          MOV SIPDR0, R1 ;READ REGISTER INTO R1
025446 020102          CMP R1, R2 ;SEE IF ONLY UPPER BYTE WAS WRITTEN
025450 001401          BEQ 2$ ;BRANCH TO EXIT IF CORRECT
025452 104033          ERROR +33 ;DIDN'T LOAD CORRECT BYTE
025454 012737 025360 001110 2$:  MOV #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
```

2436

```

.SBTTL TEST # 41 - BYTE ADDRESSING OF USER PDR'S
*****
*TEST 41      BYTE ADDRESSING OF USER PDR'S
*
*          THIS TEST CHECK THE LOGIC ASSOCIATED WITH BYTE ADDRESSING
*          OF USER PAGE DESCRIPTOR REGISTERS.  IT CAN BE ASSUMED THAT IF
*          ONE REGISTER WORKS, THEY ALL WORK BECAUSE THE REGISTERS
*          HAVE ALL BEEN TESTED.
*****
    
```

```

2437 025462 000004          TST41: SCOPE
025464 012737 025502 001110 20$: MOV #11$, $LPERR ;SET LOOP ON ERROR POINTER TO 11$
025472 012702 000014      MOV #14, R2 ;PUT EXPECTED DATA IN R2
025476 012700 177600      MOV #UIPDRO, R0 ;PUT ADDRESS OF REGISTER IN R0
025502 005037 177600      11$: CLR UIPDRO ;CLEAR THE REGISTER UNDER TEST
025506 112710 172014      MOV #172014, (R0) ;LOAD LOWER BYTE OF REGISTER
025512 013701 177500      MOV UIPDRO, R1 ;READ REGISTER INTO R1
025516 020102      CMP R1, R2 ;SEE IF ONLY LOWER BYTE WAS WRITTEN
025520 001401      BEQ 1$ ;BRANCH IF DATA MATCHES
025522 104033      ERROR +33 ;DIDN'T LOAD CORRECT BYTE
025524 012737 025542 001110 1$: MOV #12$, $LPERR ;SET LOOP ON ERROR POINTER TO 12$
025532 012700 177601      MOV #UIPDRO+1, R0 ;POINT TO UPPER BYTE OF REGISTER
025536 012702 052014      MOV #52014, R2 ;LOAD EXPECTED DATA INTO R2
025542 112710 000124      12$: MOV #124, (R0) ;WRITE UPPER BYTE OF REGISTER
025546 013701 177600      MOV UIPDRO, R1 ;READ REGISTER INTO R1
025552 020102      CMP R1, R2 ;SEE IF ONLY UPPER BYTE WAS WRITTEN
025554 001401      BEQ 2$ ;BRANCH TO EXIT IF CORRECT
025556 104033      ERROR +33 ;DIDN'T LOAD CORRECT BYTE
025560 012737 025464 001110 2$: MOV #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
    
```

2451

```
.SBTTL TEST # 42 - DUAL ADDRESSING FOR ALL PAR'S & PDR'S
*****
*TEST 42      DUAL ADDRESSING FOR ALL PAR'S & PDR'S
*
*   THIS TEST WILL ENSURE THAT THERE IS NO DUAL ADDRESSING
*   BETWEEN GROUPS OF PAR'S AND PDR'S.  THAT IS, WHEN YOU
*   REFERENCE A KERNAL I-SPACE PAR YOU ARE REALLY REFERENCING
*   THAT PAR.  FIRST EACH I-SPACE PAR OR PDR IS LOADED WITH
*   A UNIQUE NUMBER 0-12 AND THEN THEY ARE EACH CHECKED FOR
*   THE CORRECT DATA.
*   EACH GROUP HAS ALREADY BEEN TESTED FOR DUAL ADDRESSING
*   WITHIN ITS OWN GROUP, SO ONLY ONE REGISTER FROM EACH
*   GROUP NEEDS TO BE TESTED.
*****
```

```
TST42: SCOPE
2452 025566 000004 2452 025570 012737 025604 001110 20$:  MOV    #1$, $LPERR    ;SET LOOP ON ERROR POINTER TO 1$
2453 025576 005000 2454          2455 025600 012704 000006          CLR    R0              ;THIS WILL HOLD THE INDEX OF EACH
                                ;REGISTER, AND THE COUNT LOADED.
                                ;NUMBER OF TIMES TO DO THE LOOP
2456 025604 010070 001356          1$:  MOV    R0, @PARTAB(R0)  ;LOAD PAR OR PDR WITH INDEX NUMBER
2457 025610 062700 000002          ADD    #2, R0          ;CHANGE INDEX TO POINT TO NEXT REGISTER
2458 025614 077405          SOB    R4, 1$         ;BRANCH BACK 5 TIMES
2459 025616 012704 000006          MOV    #6, R4         ;DO NEXT LOOP 6 TIMES
2460 025622 013737 025634 001110          MOV    10$, $LPERR    ;SET LOOP ON ERROR POINTER TO 10$
2461 025630 162700 000002          2$:  SUB    #2, R0        ;ADJUST INDEX FOR REGISTER DESIRED
2462 025634 017001 001356          10$: MOV    @PARTAB(R0), R1 ;READ PAR OR PDR INTO R1
2463 025640 020001          CMP    R0, R1         ;SEE IF INDEX EQUALS DATA
2464 025642 001403          BEQ    3$             ;BRANCH IF INCORRECT REGISTER WAS READ
2465 025644 016002 001356          MOV    PARTAB(R0), R2 ;SAVE ADDRESS OF BAD REGISTER
2466 025650 104016          ERROR +16            ;DUAL ADDRESSING ERROR
2467 025652 077412          3$:  SOB    R4, 2$         ;BRANCH BACK 5 TIMES
2468 025654 012737 025570 001110          MOV    #20$, $LPERR   ;SET LOOP ON ERROR POINTER TO START OF TEST
```


2478

```
.SBTTL TEST # 43 - TEST THAT PAR-PDR'S NOT AFFECTED BY RESET
*****
*TEST 43 TEST THAT PAR-PDR'S NOT AFFECTED BY RESET
*
* THIS TEST CHECKS TO SEE THAT THE KERNEL, SUPERVISOR OR USER
* PAR/PDR'S ARE NOT AFFECTED BY THE EXECUTION OF A 'RESET'
* INSTRUCTION. ALL WRITEABLE BITS ARE SET TO A '1' IN THE
* PAR/PDR'S; THEY ARE THEN READ TO SEE THAT THEY REMAINED
* UNCHANGED.
*****
```

```
TST43: SCOPE
1$: MOV #20,R3 ;LOAD LOOP COUNTER WITH A 16
MOV #KIPDR0,R1 ;LOAD FIRST ADDRESS OF PDR INTO R1
MOV #KIPAR0,R2 ;LOAD FIRST ADDRESS OF PAR INTO R2
21$: MOV #-1,(R1)+ ;SET BITS IN PDR TO 1'S
MOV #-1,(R2)+ ;SET BITS IN PAR'S TO 1'S
SOB R3,21$ ;LOOP UNTIL ALL KERNAL PAR/PDR'S LOADED
MOV #20,R3 ;LOAD LOOP COUNTER WITH A 16
MOV #SIPDR0,R1 ;LOAD FIRST ADDRESS OF PDR INTO R1
MOV #SIPAR0,R2 ;LOAD FIRST ADDRESS OF PAR INTO R2
22$: MOV #-1,(R1)+ ;SET BITS IN PDR TO 1'S
MOV #-1,(R2)+ ;SET BITS IN PAR'S TO 1'S
SOB R3,22$ ;LOOP UNTIL ALL SUPERVISOR PAR/PDR'S LOADED
MOV #20,R3 ;LOAD LOOP COUNTER WITH A 16
MOV #UIPDR0,R1 ;LOAD FIRST ADDRESS OF PDR INTO R1
MOV #UIPAR0,R2 ;LOAD FIRST ADDRESS OF PAR INTO R2
23$: MOV #-1,(R1)+ ;SET BITS IN PDR TO 1'S
MOV #-1,(R2)+ ;SET BITS IN PAR'S TO 1'S
SOB R3,23$ ;LOOP UNTIL ALL USER PAR/PDR'S LOADED
RESET ;ISSUE AN 'INIT' BY EXECUTING A RESET
MOV #KIPDR0,R0 ;LOAD ADDRESS OF FIRST KERNEL PDR IN R0
MOV #20,R4 ;LOAD LOOP COUNTER WITH A 16
2$: MOV (R0),R1 ;READ A KERNEL PDR INTO R1
CMP #177416,R1 ;ARE ALL THE BITS STILL SET?
BEQ 3$ ;BRANCH IF YES
ERROR +23 ;KERNEL PDR AFFECTED BY A RESET
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 2$' = 000773
3$: ADD #2,R0 ;FORM ADDRESS OF NEXT KERNEL PDR
SOB R4,2$ ;LOOP TO 2$ UNTIL ALL KERNEL PDR'S CHECKED
MOV #KIPAR0,R0 ;LOAD ADDRESS OF FIRST KERNEL PAR IN R0
MOV #20,R4 ;LOAD LOOP COUNTER WITH A 16
4$: MOV (R0),R1 ;READ A KERNEL PAR INTO R1
CMP #177777,R1 ;ARE ALL THE BITS STILL SET?
BEQ 5$ ;BRANCH IF YES
ERROR +23 ;KERNEL PAR AFFECTED BY A RESET
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 4$' = 000773
5$: ADD #2,R0 ;FORM ADDRESS OF NEXT KERNEL PAR
SOB R4,4$ ;LOOP TO 4$ UNTIL ALL KERNEL PAR'S CHECKED
MOV #SIPDR0,R0 ;LOAD ADDRESS OF FIRST SUPERVISOR PDR IN R0
MOV #20,R4 ;LOAD LOOP COUNTER WITH A 16
6$: MOV (R0),R1 ;READ A SUPERVISOR PDR INTO R1
CMP #177416,R1 ;ARE ALL THE BITS STILL SET?
```


2576

.SBTTL TEST # 44 - INST. FETCH NOT RELOCATED IN MAINT. MODE

*TEST 44 INST. FETCH NOT RELOCATED IN MAINT. MODE

*:

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

*:

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

*:

2577	026210	000004		
2578	026212	004737	002172	
2579	026216	012700	172300	
2580	026222	012704	000010	
2581	026226	005020		
2582	026230	077402		
2583	026232	012737	000252	000250
2584	026240	005037	000252	
2585				
2586	026244	012737	001006	172300
2587	026252	012737	026260	001110
2588	026260	012737	000400	177572
2589	026266	000005		
2590	026270	032737	000400	177572
2591	026276	001403		
2592	026300	005037	177572	
2593	026304	104024		
2594				
2595				
2596				
2597	026306	012706	001100	
2598	026312	005037	177572	
2599	026316	012737	003330	000250
2600	026324	012737	000340	000252
2601	026332	012737	026212	001110
2602	026340	004737	002226	

TST44: SCOPE

1\$:	JSR	PC,TOFF			;TURN T-BIT TRAPPING OFF FOR THIS TEST
	MOV	#KIPDRO,RO			;LOAD ADDRESS OF FIRST KERNEL PDR INTO RO
	MOV	#10,R4			;LOAD LOOP COUNTER WITH AN 8
2\$:	CLR	(RO)+			;CLEAR PDR - MAPPING PAGE NON-RES, 0 BLKS.
	SOB	R4,2\$;LOOP TO 2\$ UNTIL ALL KERNEL PDR'S CLEARED
	MOV	#MMVEC+2,MMVEC			;LOAD TRAP CATCHER INTO MEM MGMT. VECTOR
	CLR	MMVEC+2			
					;A HALT WILL OCCUR IF RESET FAILS
					;TO DISABLE DEST.-ONLY RELOCATION
	MOV	#1006,KIPDRO			;MAP KERNEL PG 0 R/W, 3 BLOCKS LONG.
	MOV	#3\$,SLPERR			;SET LOOP ON ERROR POINTER TO 3\$
3\$:	MOV	#BIT8,SRO			;TURN ON DEST-ONLY-RELOCATION
	RESET				;SHOULD CLEAR MAINT. BIT - WILL ABORT IF RELOCATED
	BIT	#BIT8,SRO			;WAS MAINT. BIT (BIT 8) OF SRO CLEARED?
	BEQ	4\$;BRANCH IF YES
	CLR	SRO			;CLEAR SRO SO ERROR CAN BE REPORTED
	ERROR	+24			;MAINT. MODE NOT DISABLED BY A RESET
					;FOR A TIGHTER SCOPE LOOP
					;REPLACE ERROR CALL WITH
					; 'BR 3\$' = 000765
4\$:	MOV	#KERSTK,KSP			;RESTORE STACK POINTER
	CLR	SRO			;BE SURE SRO IS CLEAR
	MOV	#MMGMERR,MMVEC			;RESTORE MEM. MGMT. TRAP VECTOR
	MOV	#340,MMVEC+2			;RESTORE MEM. MGMT VECTOR+2
	MOV	#1\$,SLPERR			;RESET LOOP ON ERROR POINTER TO 1\$
	JSR	PC,TON			;TURN T-BIT TRAPPING BACK ON

2616

..SBTTL TEST # 45 - TEST THAT SOURCE NOT RELOCATED IN MAINT. MODE
 ..*****
 ..*TEST 45 TEST THAT SOURCE NOT RELOCATED IN MAINT. MODE
 ..*

..* THIS TEST CHECKS TO SEE THAT WHEN MEMORY MANAGEMENT IS IN
 ..* MAINTENANCE MODE, THE SOURCE IS NOT RELOCATED; ONLY THE
 ..* DESTINATION IS RELOCATED. KERNEL PAR'S 3 & 4 ARE MAPPED TO
 ..* PHYSICAL ADDRESS 60000-77776. PDR4 IS SET TO ALLOW FULL
 ..* READ/WRITE BUT PDR3 IS CLEAR ALLOWING NO ACCESS. VIRTUAL
 ..* ADDRESSES REFERENCING PAR/PDR'S 4 & 3 ARE USED AS
 ..* DESTINATION AND SOURCE RESPECTIVELY. IF THE SOURCE IS
 ..* RELOCATED IN MAINTENANCE MODE A MEM. MGMT. TRAP WILL OCCUR
 ..* AND THE ERROR WILL BE REPORTED. KERNEL PG. 7 IS MAPPED R/W.
 ..*

..*****

026344 000004
 26 7 026346 004737 002172
 2618 026352 012737 026442 001110
 2619 026360 012737 000600 172346
 2620 026366 012737 000600 172350
 2621 026374 012737 007600 172356
 2622 026402 005037 172306
 2623 026406 012737 077406 172310
 2624 026414 012737 077406 172316
 2625 026422 012737 026522 000250
 2626 026430 012737 000377 060000
 2627 026436 012700 177777
 2628 026442 052737 000400 177572
 2629 026450 113737 060000 100001
 2630 026456 000005
 2631 026460 013701 060000
 2632 026464 020001
 2633 026466 001401
 2634 026470 104025
 2635
 2636
 2637
 2638
 2639 026472 012737 003330 000250
 2640 026500 012737 077406 172306
 2641 026506 012737 026346 001110
 2642 026514 004737 002226
 2643 026520 000430
 2644
 2645 026522 042737 000400 177572
 2646 026530 013737 177572 001264
 2647 026536 013737 177576 001270
 2648 026544 010637 001256
 2649 026550 012637 001260
 2650 026554 012637 001262
 2651 026560 042737 160000 177572
 2652 026566 104026
 2653
 2654
 2655
 2656 026570 013746 001262
 2657 026574 013746 001260

TST45: SCOPE
 1\$: JSR PC,TOFF ;TURN OFF T-BIT TRAPPING FOR THIS TEST
 MOV #2\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 2\$
 MOV #600, KIPAR3 ;MAP KERNAL PAGE 3 TO 12-16K
 MOV #600, KIPAR4 ;MAP KERNAL PAGE 4 TO 12-16K
 MOV #7600, KIPAR7 ;MAP KERNAL PAGE 7 TO THE I/O PAGE
 CLR KIPDR3 ;MAP KERNAL PAGE 3 'NO ACCESS'
 MOV #77406, KIPDR4 ;MAP KERNAL PAGE 4 R/W, 128 BLKS
 MOV #77406, KIPDR7 ;MAP KERNAL PAGE 7 R/W, 128 BLKS
 MOV #4\$, MMVEC ;SET M.M. VECTOR TO 4\$ IN CASE OF ABORT
 MOV #377, 60000 ;LOAD ALL 1'S IN LOW BYTE OF TEST LOC.
 MOV #-1, RO ;LOAD EXPECTED CONTENTS OF TEST LOC. IN RO
 2\$: BIS #BIT8, SRO ;TURN ON DEST.-ONLY-RELOCATION
 MOV# 60000, 100001 ;LOAD HI BYTE OF TEST LOC.-ABORT IF SOURCE RELOCATED
 RESET ;TURN OFF DEST.-ONLY-RELOCATION
 MOV 60000, R1 ;READ CONTENTS OF TEST LOC.
 CMP RO, R1 ;WAS TEST LOCATION LOADED PROPERLY?
 BEQ 3\$;BRANCH IF YES
 ERROR +25 ;TEST LOC. NOT LOADED - INST. WAS ABORTED
 ;WHEN SOURCE WAS RELOCATED
 ;FOR TIGHTER SCOPE LOOP REPLACE
 ;ERROR CALL WITH
 ;'BR 2\$' = 000764
 3\$: MOV #MGMERR, MMVEC ;RESTORE MEM. MGMT. VECTOR
 MOV #77406, KIPDR3 ;MAP KERNAL PAGE 3 R/W, 128 BLKS
 MOV #1\$, \$LPERR ;RESET LOOP ON ERROR POINTER TO 1\$
 JSR PC, TON ;TURN T-BIT TRAPPING BACK ON
 BR TST46 ;SKIP TO NEXT TEST
 ;* IF THE PROGRAM TRIES TO RELOCATE THE SOURCE, IT SHOULD TRAP TO 4\$
 4\$: BIC #BIT8, SRO ;TURN OFF DEST.-ONLY-RELOCATION THRU PAR/PDR 7
 MOV SRO, WASSRO ;SAVE REST OF SRO CONTENTS
 MOV SR2, WASSR2 ;SAVE CONTENTS OF SR2
 MOV SP, WASR6 ;SAVE VALUE OF THE STACK POINTER
 MOV (SP)+, TRAPPC ;SAVE PC OF TRAP
 MOV (SP)+, TRAPPS ;SAVE PSW OF TRAP
 BIC #160000, SRO ;CLEAR ERROR BITS IN SRO
 ERROR +26 ;SOURCE APPARENTLY RELOCATED IN MAINT. MODE
 ;FOR TIGHTER SCOPE LOOP
 ;REPLACE ERROR CALL WITH A
 ;'NOP' = 000240
 MOV TRAPPS, -(SP) ;PUT PSW OF TRAP BACK ON THE STACK
 MOV TRAPPC, -(SP) ;PUT PC OF TRAP BACK ON THE STACK

2658 026600 000002

RTI

;RETURN TO TEST

2659
2660
2661
2662

.SBTTL ADDER TESTS
:.....
: GROUP 3 ADDER TESTS
:.....

2674

```
.SBTTL TEST # 46 - 18-BIT MAPPING ADDER TEST
:*****
:*TEST 46      18-BIT MAPPING ADDER TEST
:*
:*      THIS TEST USES 'DESTINATION ONLY' RELOCATION TO CHECK THE
:*      FULL ADD PROPERTIES OF THE RELOCATION ADDER. TWELVE PAIRS
:*      OF NUMBERS ARE ADDED, GENERATING PHYSICAL ADDRESSES FROM 12K
:*      TO ABOVE 16K.
:*
:*      NOTE - PART OF THIS TEST WILL NOT BE RUN IF THERE IS LESS
:*      THAN 16K OF MEMORY ON THE SYSTEM.
:*
:*****
```

```
026602 000004
2675
2676
2677
2678 026604 012705 172200
2679 026610 012704 000100
2680 026614 005025
2681 026616 077402
2682 026620 012705 177600
2683 026624 012704 000040
2684 026630 005025
2685 026632 077402
2686 026634 012702 000010
2687 026640 012701 172300
2688 026644 012721 077406
2689 026650 077203
2690 026652 012737 000200 172342
2691 026660 012737 000400 172344
2692 026666 012737 000600 172346
2693 026674 012737 177600 172356
2694 026702 005003
2695 026704 012704 000004
2696 026710 012705 027024
2697 026714 012737 026740 001110
2698 026722 005203
2699 026724 012537 172350
2700 026730 012500
2701 026732 013537 001176
2702 026736 012501
2703 026740 052737 000400 177572
2704 026746 010110
2705 026750 017502 177774
2706 026754 005037 177572
2707 026760 020102
2708 026762 001401
2709 026764 104017
2710 026766 013775 001176 177774
2711 026774 020304
2712 026776 001351
2713 027000 022704 000020
2714 027004 001507
2715 027006 023727 003022 001000
2716 027014 003503
2717 027016 012704 000020
```

```
TST46: SCOPE
: THE FOLLOWING CODE CLEARS ALL THE PAR'S AND PDR'S SO THAT
: THE RELOCATION ADDERS CAN BE CHECKED. ONLY THE KERNAL
: I-SPACE PAR'S AND PDR'S WILL BE MAPPED RESIDENT AND R/W.
:
:1$: MOV #SIPDR0,R5 ;PUT ADDRESS OF 1ST OF 100 PR'S TO CLEAR TO R5
: MOV #100,R4 ;CLEAR 100 LOCATIONS
: CLR (R5)+ ;CLEAR THE LOCATION
: SOB R4,1$ ;SUBTRACT 1 AND BRANCH IF NOT ZERO
: MOV #UIPDR0,R5 ;PUT ADDRESS OF REMAINING 40 PR'S TO CLEAR TO R5
: MOV #40,R4 ;CLEAR 40 LOCATIONS
: CLR (R5)+ ;CLEAR THE LOCATION
: SOB R4,2$ ;SUBTRACT 1 AND BRANCH IF NOT ZERO
:3$: MOV #10,R2 ;SET COUNTER TO LOAD 8 ADDRESSES
: MOV #KIPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
:4$: MOV #77406,(R1)+ ;LOAD 77406 INTO PDR ADDRESSED BY R1
: SOB R2,4$ ;BRANCH BACK TO 4$ IF R2 IS NOT ZERO
: MOV #200,KIPAR1 ;MAP PAGE 1 TO 4K->8K (PAGE 0 IS ALREADY MAPPED TO 0->4K)
: MOV #400,KIPAR2 ;MAP PAGE 2 TO 8K->12K
: MOV #600,KIPAR3 ;MAP PAGE 3 TO 12K->16K
: MOV #177600,KIPAR7 ;MAP PAGE 7 TO I/O PAGE
: CLR R3 ;CLEAR R3, THE POINTER USED TO INDEX TO THE TABLE
: MOV #4,R4 ;EXECUTE ROUTINE UNTIL R3=4
: MOV #PATRNS,R5 ;MOVE ADDRESS OF THE PATTERN SET (ON NEXT PAGE) TO R5
: MOV #6$, $LPERR ;SET LOOP ON ERROR POINTER TO 4$
:5$: INC R3 ;INCREMENT THE TABLE POINTER
: MOV (R5)+,KIPAR4 ;LOAD PAR4 WITH VALUE IN TABLE, IDENTIFIED BY R3 VALUE
: MOV (R5)+,R0 ;PUT VIRTUAL ADDRESS IN R0
: MOV @ (R5)+,$TMP0 ;SAVE DATA AT TEST LOCATION
: MOV (R5)+,R1 ;PUT DATA PATTERN INTO R1
:6$: BIS #BIT8,MMRO ;TURN ON DESTINATION ONLY RELOCATION
: MOV R1,(R0) ;TRY TO LOAD DATA PATTERN INTO LOCATION IN TABLE
: MOV @-4(R5),R2 ;READ LOCATION INTO R2
: CLR MMRO ;CLEAR MMRO
: CMP R1,R2 ;SEE IF DATA MATCHES PATTERN
: BEQ 7$ ;BRANCH IF DATA MATCHES
: ERROR +17 ;RELOCATION FAILED
:7$: MOV $TMP0,@-4(R5) ;RESTORE ORIGINAL DATA TO TEST LOCATION
: CMP R3,R4 ;SEE IF WE ARE AT END OF THIS SET
: BNE 5$ ;BRANCH BACK IF NOT
: CMP #20,R4 ;SEE IF 2ND BATCH OF DATA IS NOW DONE
: BEQ APTTST ;BRANCH IF SO
: CMP $LSTBK,#1000 ;DO WE HAVE OVER 16K ON THIS SYSTEM?
: BLE APTTST ;BRANCH IF WE DO NOT
: MOV #20,R4 ;EXECUTE ROUTINE UNTIL R3=20
```

2718 027022 000737

BR 5\$

:GO BACK TO THE ROUTINE

2719	027024	000600	100000	060000	PATRNS: .WORD	0600,100000,060000,125200	:R3=1 DATA
2720	027034	000625	105276	067776	.WORD	0625,105276,067776,125201	:R3=2 DATA
2721	027044	000633	105500	071000	.WORD	0633,105500,071000,125202	:R3=3 DATA
2722	027054	000604	111476	072076	.WORD	0604,111476,072076,125203	:R3=4 DATA
2723	027064	001000	100000	100000	.WORD	1000,100000,100000,125204	:R3=5 DATA
2724	027074	001252	112576	137776	.WORD	1252,112576,137776,125205	:R3=6 DATA
2725	027104	001125	105276	117776	.WORD	1125,105276,117776,125206	:R3=7 DATA
2726	027114	001010	101000	102000	.WORD	1010,101000,102000,125207	:R3=10 DATA
2727	027124	001314	110400	142000	.WORD	1314,110400,142000,125210	:R3=11 DATA
2728	027134	001104	111400	122000	.WORD	1104,111400,122000,125211	:R3=12 DATA
2729	027144	001356	104200	142000	.WORD	1356,104200,142000,125212	:R3=13 DATA
2730	027154	001242	115600	142000	.WORD	1242,115600,142000,125213	:R3=14 DATA
2731	027164	001377	102100	142000	.WORD	1377,102100,142000,125214	:R3=15 DATA
2732	027174	001221	117700	142000	.WORD	1221,117700,142000,125215	:R3=16 DATA
2733	027204	001377	100100	140000	.WORD	1377,100100,140000,125216	:R3=17 DATA
2734	027214	001370	101000	140000	.WORD	1370,101000,140000,125217	:R3=20 DATA

2735
 2736
 2737
 2738
 2739
 2740
 2741

```

;*****
;* THIS PIECE OF CODE CHECKS TO SEE IF THE PROGRAM IS RUNNING UNDER
;* APT, AND THEN LOOKS FOR THE APT SPECIAL ADDRESSING HARDWARE. IF
;* THE HARDWARE IS FOUND, THE NEXT TWO TESTS OPERATE AS IF THERE
;* WERE A FULL COMPLEMENT OF MEMORY ON THE SYSTEM.
;*****
    
```

2742	027224	005037	001340		APTST: CLR	HDWFLAG	:RESET THE HARDWARE PRESENT FLAG
2743	027230	105737	001244		TSTB	\$ENV	:ARE WE UNDER APT?
2744	027234	001420			BEQ	TST47	::BRANCH TO NEXT TEST IF NOT
2745	027236	032737	000200	001250	BIT	#BIT7,\$USWR	:IS THE HARDWARE SUPPOSED TO BE THERE?
2746	027244	001414			BEQ	TST47	::BRANCH TO NEXT TEST IF FALSE
2747	027246	012737	027262	000004	MOV	#1\$,ERRVEC	:TRAPS TO 4 GO TO 1\$
2748	027254	005737	177770		TST	RMIREG	:SPECIAL HARDWARE PRESENT?
2749	027260	000404			BR	2\$:IF WE GOT HERE, THE HARDWARE IS THERE
2750	027262	062706	000004		1\$: ADD	#4,\$P	:CLEAN UP STACK AFTER INTERRUPT
2751	027266	104060			ERROR	+60	:APT SPECIAL HARDWARE NOT FOUND
2752	027270	000402			BR	TST47	::GO TO NEXT TEST
2753	027272	005237	001340		2\$: INC	HDWFLAG	:SET SPECIAL HARDWARE PRESENT FLAG

2768

```
.SBTTL TFST # 47 - 18-BIT MAPPING CARRY PROPAGATION
*****
*TEST 47      18-BIT MAPPING CARRY PROPAGATION
*
* THIS TEST USES FULL 18-BIT RELOCATION TO CHECK THE CARRY
* PROPAGATION OF THE RELOCATION ADDER. SINCE THIS TEST SCANS
* MEMORY FROM 00100000 TO 00740000 ON 8K BOUNDARIES, IT WILL
* REPORT ANY HOLES THAT IT FINDS IN MEMORY UP TO THE LAST
* BLOCK. THE INFORMATION GIVEN WILL BE THE ADDRESS WHERE
* THE HOLE WAS DISCOVERED AND THE FIRST GOOD ADDRESS AFTER
* THE HOLE.
*
* NOTE - PART OF THIS TEST WILL NOT BE RUN IF THE SYSTEM
* MEMORY SIZE IS LESS THAN 16K WORDS.
*****
```

```
TST47: SCOPE
2769 027276 000004 012700 100100 MOV #100100,R0 ;LOAD VIRTUAL ADDR FOR PAR4 INTO R0
2770 027300 012700 000001 177572 MOV #1,MMRO ;TURN ON 18-BIT MAPPING
2771 027312 005737 001340 TST HDWFLAG ;SPECIAL HARDWARE?
2772 027316 001004 BNE 20$ ;BRANCH IF TRUE
2773 027320 023727 003022 001000 CMP $LSTBK,#1000 ;IS THERE AT LEAST 16K ON THE SYSTEM?
2774 027326 002510 BLT 4$ ;BRANCH IF LESS THAN 16K TO 4$
2775 027330 005037 001336 20$: CLR HOLFLG ;MAKE SURE HOLE FLAG STARTS AT ZERO
2776 027334 012737 027772 000004 MOV #10$,ERRVEC ;SET ERROR VECTOR POINTER TO 10$
2777 027342 012737 000777 172350 MOV #777,KIPAR4 ;LOAD PAR4 WITH STARTING BASE
2778 027350 012737 001000 172352 MOV #1000,KIPAR5 ;START ADDRESSING WITH 16K
2779 027356 012701 120000 MOV #120000,R1 ;LOAD VIRTUAL ADDR FOR PAR5 INTO R1
2780 027362 012702 000375 MOV #375,R2 ;LOAD DATA PATTERN INTO R2
2781 027366 012737 027374 001110 MOV #1$,SLPERR ;SET LOOP ON ERROR POINTER TO 1$
2782 027374 005037 001332 1$: CLR PCPUER ;CLEAR CPU ERROR FLAG
2783 027400 005737 001340 TST HDWFLAG ;SPECIAL HARDWARE?
2784 027404 001404 BEQ 2$ ;BRANCH IF FALSE
2785 027406 023737 172352 003022 CMP KIPAR5,$LSTBK ;ARE WE OUT OF EXISTING MEMORY?
2786 027414 003023 BGT 21$ ;BRANCH IF TRUE
2787 027416 011137 001176 2$: MOV (R1),$TMP0 ;SAVE DATA AT TEST LOCATION
2788 027422 005737 001332 TST PCPUER ;SEE IF THERE WAS A CPU TRAP
2789 027426 001014 BNE 3$ ;BRANCH IF TRAP OCCURRED
2790 027430 005737 001336 TST HOLFLG ;SEE IF HOLE WAS FOUND IN MEMORY
2791 027434 001411 BEQ 3$ ;BRANCH IF NO HOLE WAS FOUND
2792 027436 013737 172352 001202 MOV KIPAR5,$TMP2 ;SAVE PAR THAT POINTS TO END OF HOLE
2793 027444 012737 027330 001110 MOV #20$,SLPERR ;SET LOOP ON ERROR POINTER TO 20$
2794 027452 104041 ERROR +41 ;HOLE IN MEMORY FROM $TMP1 TO $TMP2
2795 027454 005037 001336 CLR HOLFLG ;CLEAR HOLE FLAG IN CASE THERE ARE MORE
2796 027460 010210 3$: MOV R2,(R0) ;LOAD TEST PATTERN INTO TEST LOCATION
2797 027462 000405 BR 23$ ;BRANCH OVER NEXT INSTRUCTIONS
2798 027464 010204 21$: MOV R2,R4 ;GET COPY OF DATA
2799 027466 052704 000400 BIS #BIT8,R4 ;ADD SPECIAL HARDWARE ENABLE BIT
2800 027472 010437 177770 MOV R4,MMIREG ;PUT DATA IN SPECIAL HARDWARE
2801 027476 011103 23$: MOV (R1),R3 ;READ TEST LOCATION VIA DIFFERENT VIRT. ADDR.
2802 027500 020203 CMP R2,R3 ;SEE IF THE CORRECT LOCATION WAS REFERENCED
2803 027502 001401 BEQ 22$ ;BRANCH IF CORRECT DATA WAS OBTAINED
2804 027504 104042 ERROR +42 ;BAD RELOCATION
2805 027506 00711 001176 22$: MOV $TMP0,(R1) ;RESTORE ORIGINAL DATA TO TEST LOCATION
2806 027512 002737 000400 172350 ADD #400,KIPAR4 ;CHANGE BASE ADDRESS
2807 027520 062737 000400 172352 ADD #400,KIPAR5 ;CHANGE BASE ADDRESS
2808 027526 005302 DEC R2 ;CHANGE DATA PATTERN
```

```

2809 027530 022737 007400 172352      CMP      #7400,KIPAR5      ;SEE IF PAST LAST ADDRESS
2810 027536 103316                      BHIS     1$               ;BRANCH IF NOT PAST LAST ADDRESS
2811 027540 005737 001336              TST     HOLFLG           ;SEE IF YOU ARE IN MIDDLE OF HOLE
2812 027544 001401                      BEQ     4$               ;BRANCH IF NOT IN MIDDLE OF HOLE
2813 027546 104043                      ERROR   +43             ;IN MIDDLE OF HOLE IN MEMORY
2814                                         ;HOLFLG HAS NUMBER OF TIMEOUTS
2815                                         ;$TMP1 HAS PAR OF FIRST TIMEOUT
2816                                         ;*
2817                                         ;*
2818                                         ;*
2819                                         ;*
2820                                         ;*
2821                                         ;*
2822                                         ;*
2823                                         ;*
2824 027550 005737 001340      4$:      TST     HDWFLAG          ;SPECIAL HARDWARE?
2825 027554 001402                      BEQ     45$             ;BRANCH IF NOT
2826 027556 005037 177770              CLR     RMIREG          ;MAKE SURE SPECIAL HARDWARE IS OFF
2827 027562 012737 003232 000004 45$:      MOV     #TIMERR,ERRVEC  ;RESTORE NORMAL ROUTINE FOR TRAPS THRU 4
2828 027570 012737 027622 001110      MOV     #40$,$LPERR     ;SET LOOP ON ERROR POINTER TO 40$
2829 027576 012737 007577 172350      MOV     #7577,KIPAR4    ;LOAD KIPAR4 WITH 007577
2830 027604 012702 007600              MOV     #7600,R2        ;LOAD DATA PATTERN INTO R2
2831 027610 012737 000020 001334      MOV     #20,CPUEXP      ;EXPECTING UNIBUS TIMEOUT
2832 027616 005037 001332              CLR     PCPUER          ;CLEAR CPU TRAP FLAG
2833 027622 010210              40$:      MOV     R2,(R0)         ;LOAD 760000 THRU PAGE 4
2834                                         ;THIS INSTRUCTION SHOULD TIMEOUT
2835                                         ;OVER THE UNIBUS.
2836 027624 005737 001332              TST     PCPUER          ;SEE IF TRAP OCCURRED
2837 027630 001001                      BNE     6$              ;BRANCH IF TRAP
2838 027632 104044                      ERROR   +44             ;NO CPU TRAP
2839                                         ;*
2840                                         ;*
2841                                         ;*
2842                                         ;*
2843                                         ;*
2844                                         ;*
2845 027634 012737 027660 001110      6$:      MOV     #16$,$LPERR     ;SET LOOP ON ERROR POINTER TO 16$
2846 027642 005037 001334              CLR     CPUEXP          ;NO TRAPS THRU ERRVEC EXPECTED HERE
2847 027646 012737 007777 172350      MOV     #7777,KIPAR4    ;LOAD PAR4 WITH HIGHEST VALUE POSSIBLE
2848 027654 005037 000000              CLR     0               ;CLEAR ADDRESS ZERO
2849 027660 013701 100100              16$:      MOV     100100,R1       ;THIS SHOULD READ ADDRESS ZERO INTO R1
2850 027664 005701                      TST     R1              ;SEE IF YOU READ ADDRESS ZERO
2851 027666 001401                      BEQ     7$              ;BRANCH IF ADDRESS ZERO WAS READ
2852 027670 104045                      ERROR   +45             ;DIDN'T READ ADDRESS ZERO
2853 027672 012737 027736 001110 7$:      MOV     #17$,$LPERR     ;SET LOOP ON ERROR POINTER TO 17$
2854 027700 022737 007377 003022      CMP     #7377,$LSTBK    ;IS MEMORY BLOCK SIZE < 7377
2855 027706 101423                      BLOS   8$              ;BRANCH IF MORE THAN 120K ON SYSTEM
2856 027710 012737 000040 001334      MOV     #40,CPUEXP      ;EXPECTING NON-EXISTANT MEMORY ERROR
2857 027716 005037 001332              CLR     PCPUER          ;CLEAR TRAP THRU ERRVEC FLAG
2858 027722 013737 003022 172350      MOV     $LSTBK,KIPAR4   ;GET READY TO GENERATE NON-EXISTANT ADDR.
2859 027730 062737 000100 172350      ADD     #100,KIPAR4     ;MAKE SURE WE'RE IN NON-EXISTANT MEMORY.
2860 027736 013701 100100              17$:      MOV     100100,R1       ;READ FROM NON-EXISTANT ADDRESS
2861 027742 005737 001332              TST     PCPUER          ;SEE IF TRAP THRU ERRVEC OCCURRED
2862 027746 001003                      BNE     8$              ;BRANCH TO EXIT IF TRAP OCCURED
2863 027750 012700 100100              MOV     #100100,R0      ;SAVE VIRTUAL ADDRESS FOR ERROR TYPEOUT
2864 027754 104046                      ERROR   +46             ;NO TRAP THRU ERRVEC
2865 027756 005037 001334      8$:      CLR     CPUEXP          ;NO CPU TRAPS EXPECTED

```

2866	027762	012737	027330	001110	MOV	#20\$, \$LPERR	;SET LOOP ON ERROR POINTER TO START OF TEST
2867	027770	000462			BR	TST50	;:BRANCH TO NEXT TEST
2868					:***** TRAP TO HERE THRU ERRVEC *****		
2869	027772	012637	001326		10\$: MOV	(KSP)+, OLDPC	;SAVE RETURN ADDRESS
2870	027776	012637	001330		MOV	(KSP)+, OLDPS	;SAVE RETURN PSW
2871	030002	013737	177766	001332	MOV	CPUERR, PCPUER	;SAVE CPU ERROR REGISTER FOR TYPING
2872	030010	022737	000040	001332	CMF	#40, PCPUER	;WAS TRAP NON-EXISTANT MEMORY
2873	030016	001012			BNE	12\$;BRANCH IF MEMORY/ EXISTS
2874	030020	023737	172350	003022	CMF	KIPAR4, \$LSTBK	;SEE IF PAR4 MATCHES LAST BLOCK IN MEMORY
2875	030026	002404			BLT	11\$;BRANCH IF NO MATCH-ERROR IN COMPARE CIRCUITS
2876	030030	012737	027550	001326	MOV	#4\$, OLDPC	;CHANGE RETURN ADDRESS IF AT TOP OF MEMORY
2877	030036	000430			BR	14\$;BRANCH TO EXIT
2878	030040	104047			11\$: ERROR	+47	;PREMATURE END OF MEMORY FOUND
2879	030042	000426			BR	14\$;BRANCH TO EXIT
2880	030044	022737	000020	001332	12\$: CMF	#20, PCPUER	;SEE IF ADDRESS TIMED OUT
2881	030052	001011			BNE	13\$;BRANCH IF NO TIME OUT, UNEXPECTED ERROR
2882	030054	005737	001336		TST	HOLFLG	;HAS THIS HAPPENED BEFORE?
2883	030060	001003			BNE	15\$;BRANCH IF NOT FIRST TIMEOUT
2884	030062	013737	172352	001200	MOV	KIPAR5, \$TMP1	;SAVE PAR WHERE HOLE FIRST DISCOVERED
2885	030070	005237	001336		15\$: INC	HOLFLG	;KEEP COUNT OF SUCCESSIVE TIMEOUTS
2886	030074	000411			BR	14\$;BRANCH TO EXIT
2887	030076	013737	001326	001324	13\$: MOV	OLDPC, BADPC	;MOVE PC OF UNEXPECTED ERROR FOR TYPEOUT
2888	030104	104001			ERROR	+1	;UNEXPECTED TRAP THROUGH ERRVEC
2889	030106	013737	001106	001326	MOV	\$LPADR, OLDPC	;RETURN TO BEGINNING OF TEST
2890	030114	005037	001336		CLR	HOLFLG	;CLEAR FLAG IN CASE IT WAS SET
2891	030120	005037	177766		14\$: CLR	CPUERR	;CLEAR CPU ERROR REGISTER
2892	030124	013746	001330		MOV	OLDPS, -(KSP)	;PUSH OLD PSW ONTO STACK
2893	030130	013746	001326		MOV	OLDPC, -(KSP)	;PUSH RETURN ADDRESS ONTO STACK
2894	030134	000002			RTI		;RETURN TO TEST AND CONTINUE

2908

```
.SBTTL TEST # 50 - 22-BIT MAPPING CARRY PROPAGATION
*****
*TEST 50      22-BIT MAPPING CARRY PROPAGATION
*
*      THIS TEST USES FULL 22-BIT RELOCATION TO CHECK THE CARRY
*      PROPAGATION THAT PERTAINS TO 22 BIT ADDRESSES. THIS TEST
*      SCANS MEMORY FROM 00740000 TO 16740000 OR THE LAST BLOCK,
*      ON 8K BOUNDARIES. IF ANY HOLES ARE FOUND, THE ADDRESS
*      WHERE THE HOLE WAS DISCOVERED AND THE FIRST GOOD ADDRESS
*      AFTER THE HOLE WILL BE REPORTED.
*
*      NOTE - PART OF THIS TEST WILL NOT BE RUN IF THERE IS LESS
*      THAN 120K ON THE SYSTEM.
*****
```

```
TST50: SCOPE
2909 030136 000004          MOV      #100100,R0      ;LOAD VIRTUAL ADDR FOR PAR4 INTO R0
2910 030144 012700 100100  MOV      #120000,R1      ;LOAD VIRTUAL ADDR FOR PAR5 INTO R1
2911 030150 012737 000020 172516  MOV      #BIT4,MMR3      ;ENABLE 22 BIT MAPPING
2912 030156 005737 001340          TST      HDWFLAG        ;IS THERE SPECIAL HARDWARE?
2913 030162 001004          BNE      20$           ;BRANCH IF THERE IS
2914 030164 022737 007377 003022  CMP      #7377,$LSTBK    ;IS THERE AT LEAST 120K ON THE SYSTEM?
2915 030172 003112          BGT      4$           ;BRANCH IF LESS THAN 120K
2916 030174 012737 030654 000004 20$:  MOV      #10$,ERRVEC     ;SET ERROR VECTOR POINTER TO 10$
2917 030202 005037 001336          CLR      HOLFLG        ;MAKE SURE HOLE FLAG STARTS AT ZERO
2918 030206 012737 007377 172350  MOV      #7377,KIPAR4    ;LOAD PAR4 WITH STARTING BASE
2919 030214 012737 007400 172352  MOV      #7400,KIPAR5    ;LOAD BASE ADDRESS +100 INTO KIPAR5
2920 030222 012702 000360          MOV      #360,R2        ;LOAD DATA PATTERN INTO R2
2921 030226 012737 030234 001110 1$:  MOV      #21$,SLPERR     ;SET LOOP ON ERROR POINTER TO 21$
2922 030234 005737 001340 21$:  TST      HDWFLAG        ;IS THERE SPECIAL HARDWARE?
2923 030240 001404          BEQ      22$           ;BRANCH IF NOT
2924 030242 023737 172352 003022  CMP      KIPAR5,$LSTBK   ;ARE WE OUT OF MEMORY SPACE?
2925 030250 003025          BGT      23$           ;BRANCH IF WE ARE
2926 030252 005037 001332 22$:  CLR      PCPUER         ;CLEAR CPU ERROR FLAG
2927 030256 011137 001176          MOV      (R1),$TMP0     ;SAVE DATA AT TEST LOCATION
2928 030262 005737 001332          TST      PCPUER         ;SEE IF THERE WAS A CPU TRAP
2929 030266 001014          BNE      2$           ;BRANCH IF TRAP OCCURRED
2930 030270 005737 001336          TST      HOLFLG        ;SEE IF HOLE WAS FOUND IN MEMORY
2931 030274 001411          BEQ      2$           ;BRANCH IF NO HOLE WAS FOUND
2932 030276 013737 172352 001202  MOV      KIPAR5,$TMP2    ;SAVE PAR THAT POINTS TO END OF HOLE
2933 030304 012737 030174 001110  MOV      #20$,SLPERR     ;SET LOOP ON ERROR POINTER TO 20$
2934 030312 104050          ERROR    +50           ;HOLE IN MEMORY FROM $TMP1 TO $TMP2
2935 030314 005037 001336          CLR      HOLFLG        ;CLEAR HOLE FLAG IN CASE THERE ARE MORE
2936 030320 010210 2$:  MOV      R2,(R0)        ;LOAD TEST PATTERN INTO TEST LOCATION
2937 030322 000405          BR      24$           ;BRANCH OVER NEXT FEW INSTRUCTIONS
2938 030324 010204 23$:  MOV      R2,R4          ;GET CURRENT DATA VALUE
2939 030326 052704 000400          BIS      #BIT8,R4       ;SET SPECIAL HARDWARE ENABLE BIT
2940 030332 010437 177770          MOV      R4,RMIREG     ;SETUP SPECIAL HARDWARE
2941 030336 011103 24$:  MOV      (R1),R3       ;READ TEST LOCATION VIA DIFFERENT VIRT. ADDR.
2942 030340 020203          CMP      R2,R3        ;SEE IF THE CORRECT LOCATION WAS REFERENCED
2943 030342 001401          BEQ      3$           ;BRANCH IF CORRECT DATA WAS OBTAINED
2944 030344 104051          ERROR    +51           ;BAD RELOCATION 22-BIT MAPPING
2945 030346 013711 001176 3$:  MOV      $TMP0,(R1)     ;RESTORE ORIGINAL DATA TO TEST LOCATION
2946 030352 062737 000400 172350  ADD      #400,KIPAR4    ;CHANGE BASE ADDRESS
2947 030360 062737 000400 172352  ADD      #400,KIPAR5    ;CHANGE BASE ADDRESS
2948 030366 005302          DEC      R2            ;CHANGE DATA PATTERN
2949 030370 023737 172350 003022  CMP      KIPAR4,$LSTBK  ;IS THE PAR ABOVE THE LAST BLOCK OF MEMORY
```

```

2950 030376 003010          BGT      4$          :BRANCH IF ABOVE LAST MEM. LOC.
2951 030400 022737 167400 172352  CMP      #167400,KIPAR5 :MAKE SURE YOU DON'T GO ON THE UNIBUS
2952 030406 103307          BHS      1$          :BRANCH IF NOT PAST LAST ADDRESS
2953 030410 005737 001336      TST      HOLFLG      :SEE IF MEMORY ENDS WITH A HOLE
2954 030414 001401          BEQ      4$          :BRANCH IF NO HOLE AT END OF MEMORY
2955 030416 104052          ERROR    +52        :HOLE AT END OF MEMORY
2956                                     :*
2957                                     :*
2958                                     :*
2959                                     :*
2960                                     :*
2961 030420 012737 000020 172516 4$:  MOV      #BIT4,MMR3    :ENABLE 22-BIT MAPPING
2962 030426 012737 003232 000004  MOV      #TIMERR,ERRVEC :RESTORE NORMAL ROUTINE FOR TRAPS THRU 4
2963 030434 012737 167777 172350  MOV      #167777,KIPAR4 :GET READY TO TEST U.B. ADDRESS 0
2964 030442 012737 000000 172352  MOV      #000000,KIPAR5 :SHOULD GO TO PHYSICAL ADDRESS 0
2965 030450 012702 017000          MOV      #17000,R2     :LOAD DATA PATTERN INTO R2
2966 030454 012737 030474 001110  MOV      #5$, $LPERR   :SET LOOP ON ERROR POINTER TO 5$
2967 030462 012737 000020 001334  MOV      #20,CPUEXP    :EXPECTING UNIBUS TIMEOUT
2968 030470 011037 001176          MOV      (R0), $TMP0   :SAVE DATA IN LOCATION 0 USING PAGE 4
2969 030474 010210          MOV      R2,(R0)      :LOAD DATA PATTERN INTO TEST LOCATION
2970 030476 011103          MOV      (R1),R3      :READ TEST LOCATION VIA DIFFERENT VIRT ADDR
2971 030500 013710 001176          MOV      $TMP0,(R0)   :RESTORE ORIGINAL DATA USING PAGE 4
2972 030504 020203          CMP      R2,R3        :SEE IF DATA MATCHES
2973 030506 001401          BEQ      6$          :BRANCH IF TRAP
2974 030510 104053          ERROR    +53        :BAD RELOCATION,UNISUS ADDRESS
2975                                     :*
2976                                     :*
2977                                     :*
2978                                     :*
2979                                     :*
2980                                     :*
2981 030512 012737 030536 001110 6$:  MOV      #16$, $LPERR  :SET LOOP ON ERROR POINTER TO 16$
2982 030520 005037 001334          CLR      CPUEXP       :NO TRAPS THRU ERRVEC EXPECTED HERE
2983 030524 012737 177777 172350  MOV      #177777,KIPAR4 :LOAD PAR4 WITH HIGHEST VALUE POSSIBLE
2984 030532 005037 000000          CLR      0            :CLEAR ADDRESS ZERO
2985 030536 013701 100100          MOV      100100,R1    :THIS SHOULD READ ADDRESS ZERO INTO R1
2986 030542 005701          TST      R1           :SEE IF YOU READ ADDRESS ZERO
2987 030544 001401          BEQ      7$          :BRANCH IF ADDRESS ZERO WAS READ
2988 030546 104045          ERROR    +45        :DIDN'T READ ADDRESS ZERO
2989 030550 012737 030614 001110 7$:  MOV      #17$, $LPERR  :SET LOOP ON ERROR POINTER TO 17$
2990 030556 022737 167777 003022  CMP      #167777,$LSTBK :IS MEMORY BLOCK SIZE < 167777
2991 030564 101423          BLOS    8$          :BRANCH IF MORE THAN 120K ON SYSTEM
2992 030566 012737 000040 001334  MOV      #40,CPUEXP    :EXPECTING NON-EXISTANT MEMORY ERROR
2993 030574 005037 001332          CLR      PCPUER      :CLEAR TRAP THRU ERRVEC FLAG
2994 030600 013737 003022 172350  MOV      $LSTBK,KIPAR4 :GET READY TO GENERATE NON-EXISTANT ADDR.
2995 030606 062737 000100 172350  ADD      #100,KIPAR4   :MAKE SURE WE ARE IN NON-EXISTANT MEMORY
2996 030614 013701 100100          MOV      100100,R1    :READ FROM NON-EXISTANT ADDRESS
2997 030620 005737 001332          TST      PCPUER      :SEE IF TRAP THRU ERRVEC OCCURRED
2998 030624 001003          BNE     8$          :BRANCH TO EXIT IF TRAP OCCURED
2999 030626 012700 100100          MOV      #100100,R0   :SAVE VIRTUAL ADDRESS FOR ERROR TYPEOUT
3000 030632 104046          ERROR    +46        :NO TRAP THRU ERRVEC
3001 030634 005037 001334          CLR      CPUEXP       :NO CPU TRAPS EXPECTED
3002 030640 012737 030174 001110 8$:  MOV      #20$, $LPERR  :SET LOOP ON ERROR POINTER TO START OF TEST
3003 030646 005037 172516          CLR      MMR3        :RETURN TO 18-BIT MAPPING
3004 030652 000462          BR      TST51        :BRANCH TO NEXT TEST
3005                                     :***** TRAP TO HERE THRU ERRVEC *****
3006 030654 012637 001326 10$:  MOV      (KSP)+,OLDPC  :SAVE RETURN ADDRESS

```

3007	030660	012637	001330			MOV	(KSP)+,OLDPS	:SAVE RETURN PSW
3008	030664	013737	177766	001332		MOV	CPUERR,PCPUER	:SAVE CPU ERROR REGISTER FOR TYPING
3009	030672	022737	000040	001332		CMP	#40,PCPUER	:WAS TRAP NON-EXISTANT MEMORY
3010	030700	001012				BNE	12\$:BRANCH IF MEMORY EXISTS
3011	030702	023737	172350	003022		CMP	KIPAR4,\$LSTBK	:SEE IF PAR4 MATCHES LAST BLOCK IN MEMORY
3012	030710	002404				BLT	11\$:BRANCH IF NO MATCH-ERROR IN COMPARE CIRCUITS
3013	030712	012737	030420	001326		MOV	#4\$,OLDPC	:CHANGE RETURN ADDRESS IF AT TOP OF MEMORY
3014	030720	000430				BR	14\$:BRANCH TO EXIT
3015	030722	104047			11\$:	ERROR	+47	:PREMATURE END OF MEMORY FOUND
3016	030724	000426				BR	14\$:BRANCH TO EXIT
3017	030726	022737	000020	001332	12\$:	CMP	#20,PCPUER	:SEE IF ADDRESS TIMED OUT
3018	030734	001011				BNE	13\$:BRANCH IF NO TIME OUT,UNEXPECTED ERROR
3019	030736	005737	001336			TST	HOLFLG	:HAS THIS HAPPENED BEFORE?
3020	030742	001003				BNE	15\$:BRANCH IF NOT FIRST TIMEOUT
3021	030744	013737	172352	001200		MOV	KIPAR5,\$TMP1	:SAVE PAR WHERE HOLE FIRST DISCOVERED
3022	030752	005237	001336		15\$:	INC	HOLFLG	:KEEP COUNT OF SUCCESSIVE TIMEOUTS
3023	030756	000411				BR	14\$:BRANCH TO EXIT
3024	030760	013737	001326	001324	13\$:	MOV	OLDPC,BADPC	:MOVE PC OF UNEXPECTED ERROR FOR TYPEOUT
3025	030766	104001				ERROR	+1	:UNEXPECTED TRAP THROUGH ERRVEC
3026	030770	013737	001106	001326		MOV	\$LPADR,OLDPC	:RETURN TO BEGINNING OF TEST
3027	030776	005037	001336			CLR	HOLFLG	:CLEAR FLAG IN CASE IT WAS SET
3028	031002	005037	177766		14\$:	CLR	CPUERR	:CLEAR CPU ERROR REGISTER
3029	031006	013746	001330			MOV	OLDPS,-(KSP)	:PUSH OLD PSW ONTO STACK
3030	031012	013746	001326			MOV	OLDPC,-(KSP)	:PUSH RETURN ADDRESS ONTO STACK
3031	031016	000002				RTI		:RETURN TO TEST AND CONTINUE

3052

```

.SBTTL TEST # 51 - READ AND WRITE WHILE IN RELOCATE MODE
*****
*TEST 51      READ AND WRITE WHILE IN RELOCATE MODE
*
* THE FOLLOWING TEST TURNS ON MEMORY MANAGEMENT AND THEN
* READS AND WRITES LOCATIONS BETWEEN PHYSICAL ADDRESSES
* 060000-067600. ONE LOCATION IN EVERY BLOCK (32. WORDS)
* IS WRITTEN USING PAR4 AND READ USING PAR5. THIS IS
* DONE IN BOTH USER AND KERNEL MODES. THE 'MODE' INPUT TO
* THE PAR/PDR ADDRESS MUX IS CHECKED BY READING AND WRITING
* IN USER MODE. REMEMBER ALSO, THAT SINCE MEMORY MANAGEMENT
* IS ON (IN RELOCATE MODE) THE PROGRAM ITSELF IS USING ITS
* VIRTUAL ADDRESSES AND THE PAR/PDR'S TO EXECUTE.
*
* WHILE TESTING IN KERNEL MODE, USER PAGES 4 & 5 ARE MAPPED
* NON-RESIDENT WITH DIFFERENT PAR VALUES THAN THE KERNEL
* PAR'S TO BE SURE THAT THE KERNEL PAR'S AND PDR'S ARE BEING
* USED WHEN IN KERNEL MODE (AND VICE VERSA WHILE TESTING IN
* USER MODE). IF A MEM. MGMT. TRAP OCCURS, THE PROGRAM GOES
* TO 8$ WHERE THE TRAP IS REPORTED.
*
*****
    
```

```

3053 031020 000004
3053 031022 005037 177776
3054 031026 012704 000577
3055 031032 012705 000600
3056 031036 010437 172350
3057 031042 010537 172352
3058 031046 012700 177640
3059 031052 012703 172240
3060 031056 005001
3061 031060 012702 000007
3062 031064 010120
3063 031066 010123
3064 031070 062701 000200
3065 031074 077205
3066 031076 012710 007600
3067 031102 012713 007600
3068 031106 012700 177600
3069 031112 012703 172200
3070 031116 012701 077406
3071 031122 012702 000010
3072 031126 010120
3073 031130 010123
3074 031132 077203
3075 031134 105037 177610
3076 031140 105037 177612
3077 031144 105037 172210
3078 031150 105037 172212
3079 031154 010537 177650
3080 031160 010437 177652
3081 031164 010537 172250
3082 031170 010437 172252
3083 031174 012737 000001 177572
3084 031202 012737 031234 001110
3085 031210 012737 031556 000250
3086 031216 013737 177776 001176

TST51: SCOPE
1$: CLR PSW ;START IN KERNEL MODE
MOV #577,R4 ;LOAD R4 WITH VALUE FOR PAR4
MOV #600,R5 ;LOAD R5 WITH VALUE FOR PAR5
MOV R4,KIPAR4 ;LOAD KERNEL PAR4
MOV R5,KIPAR5 ;LOAD KERNEL PAR5
MOV #UIPAR0,R0 ;LOAD ADDRESS OF FIRST USER PAR IN R0
MOV #SIPAR0,R3 ;LOAD ADDRESS OF FIRST SUPERVISOR PAR IN R3
CLR R1 ;CLEAR R1
MOV #7,R2 ;LOAD LOOP COUNTER WITH A 7
2$: MOV R1,(R0)+ ;MAP USER PAR'S TO PAGES 0-6 (4K EACH)
MOV R1,(R3)+ ;MAP SUPERVISOR PAR'S TO PAGES 0-6 (4K EACH)
ADD #200,R1
SOB R2,2$ ;LOOP UNTIL UIPAR0-UIPAR6 ARE LOADED
MOV #7600,(R0) ;MAP USER PAR7 TO THE I/O PAGE
MOV #7600,(R3) ;MAP SUPERVISOR PAR7 TO THE I/O PAGE
MOV #UIPDR0,R0 ;LOAD ADDRESS OF FIRST USER PDR IN R0
MOV #SIPDR0,R3 ;LOAD ADDRESS OF FIRST SUPERVISOR PDR IN R3
MOV #77406,R1 ;LOAD PDR DATA INTO R1
3$: MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8
MOV R1,(R0)+ ;MAP ALL 8 PAGES 128 BLOCKS, UPWARD
MOV R1,(R3)+ ;EXPANDABLE,READ/WRITE FOR
SOB R2,3$ ;USER AND SUPERVISOR MODES.
CLRB UIPDR4 ;MAP USER SPACE NON-RESIDENT WHILE
CLRB UIPDR5 ;TESTING KERNEL SPACE
CLRB SIPDR4 ;MAP SUPERVISOR SPACE NON-RESIDENT WHILE
CLRB SIPDR5 ;TESTING KERNEL SPACE
MOV R5,UIPAR4 ;MAP USER PAR'S OPPOSITE OF KIPAR'S
MOV R4,UIPAR5
MOV R5,SIPAR4 ;MAP SUPERVISOR PAR'S OPPOSITE OF KIPAR'S
MOV R4,SIPAR5
MOV #1,SRO ;TURN ON MEMORY MANAGEMENT (RELOCATE MODE)
MOV #5$, $LPERR ;SET LOOP ON ERROR POINTER TO 5$
MOV #8$, $MVEC ;SET M. M. TRAP VECTOR TO 8$
4$: MOV PSW,$TMP0 ;SAVE PSW IN CASE OF ERROR
    
```


3087	031224	012700	100100		MOV	#100100,R0	:	PUT VIRTUAL ADDR. THAT USES PAR4 IN R0
3088	031230	012701	120000		MOV	#120000,R1	:	PUT VIRTUAL ADDR. THAT USES PAR5 IN R1
3089	031234	010010		5\$:	MOV	R0,(R0)	:	WRITE TO TEST LOC. USING PAR4
3090	031236	011102			MOV	(R1),R2	:	READ THE SAME LOC., BUT USING PAR5
3091	031240	020002			CMP	R0,R2	:	DID WE READ WHAT WE WROTE?
3092	031242	001411			BEQ	6\$:	BRANCH IF YES
3093	031244	010137	001300		MOV	R1,VIRT2	:	SAVE VIRTUAL ADDR. THAT SELECTED PAR5
3094	031250	010037	001276		MOV	R0,VIRT1	:	SAVE VIRTUAL ADDR. THAT SELECTED PAR4
3095	031254	004737	003426		JSR	PC,FORMPA	:	GO FROM PHYSICAL ADDRESS BEING USED
3096	031260	104020			ERROR	+20	:	READING LOC. USING PAR5 AND A VIRT.
3097							:	ADDR. DID NOT FIND DATA WRITTEN WHEN USING
3098							:	PAR4 AND VIRT. ADDRESS.
3099							:	OF TIGHTER SCOPE LOOP
3100							:	PLACE ERROR CALL WITH
3101							:	'BR 5\$' = 000765
3102	031262	013700	001276		MOV	VIRT1,R0	:	RESTORE VBA IN R0
3103	031266	062700	000100	6\$:	ADD	#100,R0	:	CHANGE VIRTUAL ADDR. TO POINT TO NEXT BLOCK
3104	031272	062701	000100		ADD	#100,R1		
3105	031276	020127	127700		CMP	R1,#127700		
3106	031302	001354			BNE	5\$:	WERE BLOCKS FROM 60000-67600 ALL TRIED?
3107	031304	032737	140000	177776	BIT	#140000,PSW	:	BRANCH IF NO
3108	031312	001026			BNE	7\$:	HAVE WE DONE TEST IN USER MODE YET?
3109	031314	010437	177650		MOV	R4,UIPAR4	:	BRANCH IF YES
3110	031320	010537	177652		MOV	R5,UIPAR5	:	LOAD USER PAR4
3111	031324	112737	000006	177610	MOVB	#6,UIPDR4	:	LOAD USER PAR5
3112	031332	112737	000006	177612	MOVB	#6,UIPDR5	:	MAP USER SPACE R/W TO TEST IT
3113	031340	105037	172310		CLRB	KIPDR4	:	MAP KERNEL SPACE NON-RESIDENT WHILE
3114	031344	105037	172312		CLRB	KIPDR5	:	TESTING USER SPACE
3115	031350	010537	172350		MOV	R5,KIPAR4	:	MAP KERNEL PAR'S OPPOSITE UIPAR'S
3116	031354	010437	172352		MOV	R4,KIPAR5		
3117	031360	012737	140000	177776	MOV	#140000,PSW		
3118	031366	000713			BR	4\$:	GO TO USER MODE
3119	031370	032737	040000	177776	7\$:	BIT	#40000,PSW	GO BACK AND READ/WRITE IN USER MODE
3120	031376	001026			BNE	10\$:	HAVE WE DONE TEST IN SUPERVISOR MODE YET?
3121	031400	010437	172250		MOV	R4,SIPAR4	:	BRANCH IF YES
3122	031404	010537	172252		MOV	R5,SIPAR5	:	LOAD SUPERVISOR PAR4
3123	031410	112737	000006	172210	MOVB	#6,SIPDR4	:	LOAD SUPERVISOR PAR5
3124	031416	112737	000006	172212	MOVB	#6,SIPDR5	:	MAP SUPERVISOR SPACE R/W TO TEST IT
3125	031424	105037	177610		CLRB	UIPDR4	:	MAP USER SPACE NON-RESIDENT WHILE
3126	031430	105037	177612		CLRB	UIPDR5	:	TESTING USER SPACE
3127	031434	010537	177650		MOV	R5,UIPAR4	:	MAP USER PAR'S OPPOSITE SIPAR'S
3128	031440	010437	177652		MOV	R4,UIPAR5		
3129	031444	012737	040000	177776	MOV	#40000,PSW		
3130	031452	000661			BR	4\$:	GO TO SUPERVISOR MODE
3131	031454	005037	177776	10\$:	CLR	PSW	:	GO BACK AND READ/WRITE IN SUPERVISOR MODE
3132	031460	012737	077406	172310	MOV	#77406,KIPDR4	:	GO BACK TO KERNEL MODE BEFORE LEAVING
3133	031466	012737	077406	172312	MOV	#77406,KIPDR5	:	REMAP KERNEL PAGES READ/WRITE
3134	031474	012737	077406	177610	MOV	#77406,UIPDR4		
3135	031502	012737	077406	177612	MOV	#77406,UIPDR5	:	REMAP USER PAGES READ/WRITE
3136	031510	010537	172350		MOV	R5,KIPAR4	:	MAP KERNEL, SUPERVISOR AND USER PAR'S 4 & 5
3137	031514	010537	172352		MOV	R5,KIPAR5	:	BACK TO 12-16K
3138	031520	010537	177650		MOV	R5,UIPAR4		
3139	031524	010537	177652		MOV	R5,UIPAR5		
3140	031530	010537	172250		MOV	R5,SIPAR4		
3141	031534	010537	172252		MOV	R5,SIPAR5		
3142	031540	012737	003330	000250	MOV	#MGMERR,#MVEC	:	RESTORE ADDR. OF NORMAL M.M. TRAP ROUTINE
3143	031546	012737	031022	001110	MOV	#1\$,\$LPERR	:	RESET LOOP ON ERROR POINTER TO 1\$

```

3144 031554 000427          BR      TST52          ;;BRANCH TO NEXT TEST
3145          ;*****      TRAP TO HERE THRU ERRVEC      *****
3146 031556 012637 001260 8$:      MOV      (KSP)+,TRAPPC      ;SAVE PC & PS OF TRAP
3147 031562 012637 001262          MOV      (KSP)+,TRAPPS
3148          ;PROGRAM WILL TRAP TO HERE IF TRY
3149          ;TO USE USER/SUPERVISOR PDR'S WHEN IN KERNEL MODE
3150          ;OR KERNEL PDR'S WHEN IN USER/SUPERVISOR MODE
3151 031566 010037 001276          MOV      R0,VIRT1      ;SAVE VIRTUAL ADDRESS FOR ERROR REPORT
3152 031572 004737 003426          JSR      PC,FORMPA      ;GO FORM THE PHYSICAL ADDRESS BEING USED
3153 031576 013737 177572 001264      MOV      SR0,WASSR0      ;SAVE SR0 & SR2 FOR ERROR REPORT
3154 031604 013737 177576 001270      MOV      SR2,WASSR2
3155 031612 042737 160000 177572      BIC      #160000,SR0      ;CLEAR ERROR BITS IN SR0
3156 031620 104022          ERROR      +22      ;M.M. TRAP WHILE IN RELOCATE MODE -
3157          ;REFERENCED WRONG SET OF PDR'S
3158          ;FOR TIGHTER SCOPE LOOP
3159          ;REPLACE ERROR CALL WITH
3160          ;A 'NOP' = 000240
3161 031622 013746 001262          MOV      TRAPPS,-(KSP)      ;PUT PC & PS OF TRAP ON STACK
3162 031626 013746 001260          MOV      TRAPPC,-(KSP)
3163 031632 000002          RTI          ;RETURN TO TEST
    
```

3164
3165
3166
3167

```
.SBTTL W-BIT TESTS  
:*****  
: GROUP 4 W-BIT TESTS  
:*****
```

3171

```
.SBTTL TEST # 52 - W-BIT LOGIC TEST, KERNEL PDR'S
:*****
*TEST 52 W-BIT LOGIC TEST, KERNEL PDR'S
* THIS TEST WRITES TO EIGHT (8) DIFFERENT VIRTUAL ADDRESSES
* (VBA'S = 17776,37776,57776,77776,117776,137776,157776 & 177776
* & PBA'S CONSTRUCTED = 17776, 37776, 57776, 77776, 77776,
* 77776, 77776, & 77776 RESPECTIVELY).
* WHICH SHOULD CAUSE THE 'W-BIT' TO SET IN EACH OF THE
* EIGHT (8) KERNEL PAGE DESCRIPTOR REGISTERS. THE PDR'S
* ARE CHECKED TO SEE THAT IT'S W-BIT DOES SET WHEN THE
* PAGE IT IS MAPPED TO IS WRITTEN TO AND THAT THE W-BIT
* DOES NOT SET IN ANY OF THE OTHER PDR'S. KERNEL PDR'S 3,4,5 & 6
* ARE MAPPED TO 12-16K FOR THIS TEST. ALSO THE W-BIT
* SHOULD BE CLEARED WHEN THE PDR IS WRITTEN TO. THE
* W-BIT PORTION OF THE PDR'S AND THE PAR/PDR ADRS MUX
* ARE BEING CHECKED.
:*****
```

3172

```
031634 000004
031636
031642 004737 002172
031646 012702 000004
031652 012700 172346
031652 012701 000600
031656 010120
031660 077202
031662 012705 172300
031666 012704 000010
031672 012703 017776
031676 012737 031704 001110
031704 012700 172300
031710 012702 000010
031714 012701 077406
031720 010120
031722 077202
031724 011313
031726 031527 000100
031732 001002
031734 104054

031736 000422
031740 012702 000010
031744 012700 172300
031750 031027 000100
031754 001403
031756 020500
031760 001401
031762 104055

031764 042700 000002
031770 077211
031772 010115
031774 031527 000100
032000 001401
032002 104056
```

```
TST52: SCOPE
1$: JSR PC,TOFF ;TURN T-BIT TRAPPING OFF FOR THIS TEST
MOV #4,R2 ;SET LOOP COUNTER TO 4
MOV #KIPAR3,R0 ;LOAD ADDRESS OF KIPAR3 INTO R0
MOV #600,R1 ;LOAD '12-16K' PAR VALUE INTO R1
2$: MOV R1,(R0)+ ;MAP PARS 3-6 TO 12-16K
SOB R2,2$ ;LOOP UNTIL ALL 4 OF THEM ARE LOADED
MOV #KIPDRO,R5 ;LOAD ADDRESS OF FIRST PDR TO BF TESTED IN R5
MOV #8,R4 ;SET LOOP COUNTER TO 8
MOV #17776,R3 ;INITIALIZE VIRTUAL ADDRESS TO BE IN R3
MOV #3$,SLPERR ;SET LOOP ON ERROR POINTER TO 3$
3$: MOV #KIPDRO,R0 ;LOAD ADDR. OF FIRST PDR TO BE SETUP IN R0
MOV #8,R2 ;SET LOOP COUNTER TO 8
MOV #77406,R1 ;PUT 'W-BIT OFF DATA' INTO R1
4$: MOV R1,(R0)+ ;CLEAR ALL W-BITS BY WRITING TO ALL PDRS
SOB R2,4$ ;LOOP UNTIL ALL OF THEM ARE SET UP
MOV (R5),(R3) ;DO 'DATO' TO VIRTUAL ADDR.-SETTING A W-BIT
BIT (R5),#WBIT ;DID THAT CAUSE W-BIT TO BE SET?
BNE 5$ ;BRANCH IF YES
ERROR +54 ;W-BIT DID NOT GET SET IN PDR
;FOR TIGHTER SCOPE LOOP, REPLACE ERROR
;CALL WITH 'BR 3$' = 000763
5$: BR 8$ ;SKIP CHECKING OTHER PDR'S-ERROR WILL SET W-BITS
MOV #8,R2 ;SET LOOP COUNTER TO 8
MOV #KIPDRO,R0 ;LOAD ADDR. OF FIRST PDR TO BE CHECKED IN R0
6$: BIT (R0),#WBIT ;DID W-BIT IN OTHER PDRS REMAIN CLEAR?
BEQ 7$ ;BRANCH IF YES
CMP R5,R0 ;IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
BEQ 7$ ;BRANCH IF YES
ERROR +55 ;W-BIT GOT SET IN MORE THAN ONE PDR
;FOR TIGHTER SCOPE LOOP, REPLACE ERROR
;CALL WITH 'BR 3$' = 000750
7$: ADD #2,R0 ;POINT R0 TO NEXT PDR TO BE CHECKED
SOB R2,6$ ;LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
MOV R1,(R5) ;WRITE TO THE PDR TESTED TO CLEAR W-BIT
BIT (R5),#WBIT ;DID WRITING PDR CLEAR THE W-BIT?
BEQ 8$ ;BRANCH IF YES
ERROR +56 ;W-BIT DID NOT CLEAR BY WRITING THE PDR
;FOR TIGHTER SCOPE LOOP, REPLACE ERROR CALL
```

032004	062705	000002		8\$:	ADD	#2,R5	:WITH 'BR 3\$' = 000740
032010	062703	020000			ADD	#20000,R3	:POINT R5 TO THE NEXT PDR TO BE TESTED
032014	077445				SOB	R4,3\$:CHANGE VIRT. ADDR TO REF. NEXT PDR
032016	012737	031636	001110		MOV	#1\$,\$LPERR	:LOOP BACK TO 3\$ UNTIL ALL 8 PDR'S TESTED
032024	004737	002226			JSR	PC,TON	:RESET LOOP ON ERROR POINTER TO 1\$
							:TURN T-BIT BACK ON FOR NEXT TEST

3176

```
.SBTTL TEST # 53 - W-BIT LOGIC TEST, SUPERVISOR PDR'S
*****
:TEST 53      W-BIT LOGIC TEST, SUPERVISOR PDR'S
:*          THIS TEST WRITES TO EIGHT (8) DIFFERENT VIRTUAL ADDRESSES
:*          (VBA'S = 17776, 37776, 57776, 77776, 117776, 137776, 157776 & 177776
:*          & PBA'S CONSTRUCTED = 17776, 37776, 57776, 77776, 77776,
:*          77776, & 77776 RESPECTIVELY).
:*          WHICH SHOULD CAUSE THE 'W-BIT' TO SET IN EACH OF THE
:*          EIGHT (8) SUPERVISOR PAGE DESCRIPTOR REGISTERS. THE PDR'S
:*          ARE CHECKED TO SEE THAT IT'S W-BIT DOES SET WHEN THE
:*          PAGE IT IS MAPPED TO IS WRITTEN TO AND THAT THE W-BIT
:*          DOES NOT SET IN ANY OF THE OTHER PDR'S. SUPERVISOR PDR'S 3,4,5 & 6
:*          ARE MAPPED TO 12-16K FOR THIS TEST. ALSO THE W-BIT
:*          SHOULD BE CLEARED WHEN THE PDR IS WRITTEN TO. THE
:*          W-BIT PORTION OF THE PDR'S AND THE PAR/PDR ADRS MUX
:*          ARE BEING CHECKED.
*****
```

```
TST53: SCOPE
3177 032030 000004          1$:  MOV    #40000,PSW      ;GO TO SUPERVISOR MODE FOR THIS TEST
3178 032032 012737 040000 177776 JSR    PC,TOFF        ;TURN T-BIT TRAPPING OFF FOR THIS TEST
032040 004737 002172          MOV    #4,R2         ;SET LOOP COUNTER TO 4
032044 012702 000004          MOV    --#SIPAR3,RO  ;LOAD ADDRESS OF SIPAR3 INTO RO
032050 012700 172246          MOV    #600,R1      ;LOAD '12-16K' PAR VALUE INTO R1
032054 012701 000600          MOV    R1,(R0)+     ;MAP PARS 3-6 TO 12-16K
032060 010120          SOB   R2,2$        ;LOOP UNTIL ALL 4 OF THEM ARE LOADED
032062 077202          MOV    #SIPDRO,R5   ;LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
032064 012705 172200          MOV    #8,R4        ;SET LOOP COUNTER TO 8
032070 012704 000010          MOV    #17776,R3    ;INITIALIZE VIRTUAL ADDRESS TO BE IN R3
032074 012703 017776          MOV    #3$,SLPERR   ;SET LOOP ON ERROR POINTER TO 3$
032100 012737 032106 001110 3$:  MOV    #SIPDRO,RO    ;LOAD ADDR. OF FIRST PDR TO BE SETUP IN RO
032106 012700 172200          MOV    #8,R2        ;SET LOOP COUNTER TO 8
032112 012702 000010          MOV    #77406,R1    ;PUT 'W-BIT OFF DATA' INTO R1
032116 012701 077406          MOV    R1,(R0)+     ;CLEAR ALL W-BITS BY WRITING TO ALL PDRS
032122 010120          SOB   R2,4$        ;LOOP UNTIL ALL OF THEM ARE SET UP
032124 077202          MOV    (R3),(R3)    ;DO 'DATO' TO VIRTUAL ADDR.-SETTING A W-BIT
032126 011313          BIT   (R5),#WBIT   ;DID THAT CAUSE W-BIT TO BE SET?
032130 031527 000100          BNE   5$           ;BRANCH IF YES
032134 001002          ERROR +54        ;W-BIT DID NOT GET SET IN PDR
032136 104054          ;FOR TIGHTER SCOPE LOOP, REPLACE ERROR
;CALL WITH 'BR 3$' = 000763
;SKIP CHECKING OTHER PDR'S-ERROR WILL SET W-BITS
032140 000422          5$:  BR    8$           ;SET LOOP COUNTER TO 8
032142 012702 000010          MOV    #SIPDRO,RO   ;LOAD ADDR. OF FIRST PDR TO BE CHECKED IN RO
032146 012700 172200          BIT   (RO),#WBIT   ;DID W-BIT IN OTHER PDRS REMAIN CLEAR?
032152 031027 000100          BEQ   7$           ;BRANCH IF YES
032156 001403          CMP   R5,RO        ;IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
032160 020500          BEQ   7$           ;BRANCH IF YES
032162 001401          ;W-BIT GOT SET IN MORE THAN ONE PDR
032164 104055          ;FOR TIGHTER SCOPE LOOP, REPLACE ERROR
;CALL WITH 'BR 3$' = 000750
;POINT RO TO NEXT PDR TO BE CHECKED
032166 062700 000002          7$:  ADD   #2,RO        ;LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
032172 077211          SOB   R2,6$        ;WRITE TO THE PDR TESTED TO CLEAR W-BIT
032174 010115          MOV    (R5),#WBIT  ;DID WRITING PDR CLEAR THE W-BIT?
032176 031527 000100          BEQ   8$           ;BRANCH IF YES
032202 001401          ;W-BIT DID NOT CLEAR BY WRITING THE PDR
032204 104056          ;FOR TIGHTER SCOPE LOOP, REPLACE ERROR CALL
```

032206	062705	000002		8\$:	ADD	#2,R5	;WITH 'BR 3\$' = 000740
032212	062703	020000			ADD	#20000,R3	;POINT R5 TO THE NEXT PDR TO BE TESTED
032216	077445				SOB	R4,3\$;CHANGE VIRT. ADDR TO REF. NEXT PDR
032220	012737	032032	001110		MOV	#1\$,\$LPERR	;LOOP BACK TO 3\$ UNTIL ALL 8 PDR'S TESTED
032226	004737	002226			JSR	PC,TON	;RESET LOOP ON ERROR POINTER TO 1\$
3179 032232	005037	177776			CLR	PSW	;TURN T-BIT BACK ON FOR NEXT TEST
							;BACK TO KERNAL MODE BEFORE LEAVING

3183

```
.SBTTL TEST # 54 - W-BIT LOGIC TEST, USER PDR'S
*****
:TEST 54      W-BIT LOGIC TEST, USER PDR'S
:*          THIS TEST WRITES TO EIGHT (8) DIFFERENT VIRTUAL ADDRESSES
:*          (VBA'S = 17776,37776,57776,77776,117776,137776,157776 & 177776
:*          & PBA'S CONSTRUCTED = 17776, 37776, 57776, 77776, 77776,
:*          77776, 77776, & 77776 RESPECTIVELY).
:*          WHICH SHOULD CAUSE THE 'W-BIT' TO SET IN EACH OF THE
:*          EIGHT (8) USER PAGE DESCRIPTOR REGISTERS. THE PDR'S
:*          ARE CHECKED TO SEE THAT IT'S W-BIT DOES SET WHEN THE
:*          PAGE IT IS MAPPED TO IS WRITTEN TO AND THAT THE W-BIT
:*          DOES NOT SET IN ANY OF THE OTHER PDR'S. USER PDR'S 3,4,5 & 6
:*          ARE MAPPED TO 12-16K FOR THIS TEST. ALSO THE W-BIT
:*          SHOULD BE CLEARED WHEN THE PDR IS WRITTEN TO. THE
:*          W-BIT PORTION OF THE PDR'S AND THE PAR/PDR ADRS MUX
:*          ARE BEING CHECKED.
*****
```

```
TST54: SCOPE
3184 032236 000004 140000 177776 1$: MOV #140000,PSW ;GO TO USER MODE FOR THIS TEST
3185 032240 012737 002172 JSR PC,TOFF ;TURN T-BIT TRAPPING OFF FOR THIS TEST
032246 004737 000004 MOV #4,R2 ;SET LOOP COUNTER TO 4
032252 012702 177646 MOV #UIPAR3,R0 ;LOAD ADDRESS OF UIPAR3 INTO R0
032256 012700 000600 MOV #600,R1 ;LOAD "12-16K" PAR VALUE INTO R1
032262 012701 000600 MOV R1,(R0)+ ;MAP PARS 3-6 TO 12-16K
032266 010120 2$: SOB R2,2$ ;LOOP UNTIL ALL 4 OF THEM ARE LOADED
032270 077202 MOV #UIPDR0,R5 ;LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
032272 012705 177600 MOV #8,R4 ;SET LOOP COUNTER TO 8
032276 012704 000010 MOV #17776,R3 ;INITIALIZE VIRTUAL ADDRESS TO BE IN R3
032302 012703 017776 MOV #3$,SLPERR ;SET LOOP ON ERROR POINTER TO 3$
032306 012737 032314 001110 3$: MOV #UIPDR0,R0 ;LOAD ADDR. OF FIRST PDR TO BE SETUP IN R0
032314 012700 177600 MOV #8,R2 ;SET LOOP COUNTER TO 8
032320 012702 000010 MOV #77406,R1 ;PUT 'W-BIT OFF DATA' INTO R1
032324 012701 077406 4$: MOV R1,(R0)+ ;CLEAR ALL W-BITS BY WRITING TO ALL PDRS
032330 010120 SOB R2,4$ ;LOOP UNTIL ALL OF THEM ARE SET UP
032332 077202 MOV (R3),(R3) ;DO 'DATA' TO VIRTUAL ADDR.-SETTING A W-BIT
032334 011313 BIT (R5),#WBIT ;DID THAT CAUSE W-BIT TO BE SET?
032336 031527 000100 BNE 5$ ;BRANCH IF YES
032342 001002 5$: ERROR +54 ;W-BIT DID NOT GET SET IN PDR
032344 104054 ;FOR TIGHTER SCOPE LOOP, REPLACE ERROR
;CALL WITH 'BR 3$' = 000763
032346 000422 BR 8$ ;SKIP CHECKING OTHER PDR'S-ERROR WILL SET W-BITS
032350 012702 000010 5$: MOV #8,R2 ;SET LOOP COUNTER TO 8
032354 012700 177600 MOV #UIPDR0,R0 ;LOAD ADDR.-OF FIRST PDR TO BE CHECKED IN R0
032360 031027 000100 6$: BIT (R0),#WBIT ;DID W-BIT IN OTHER PDRS REMAIN CLEAR?
032364 001403 BEQ 7$ ;BRANCH IF YES
032366 020500 CMP R5,R0 ;IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
032370 001401 BEQ 7$ ;BRANCH IF YES
032372 104055 ERROR +55 ;W-BIT GOT SET IN MORE THAN ONE PDR
;FOR TIGHTER SCOPE LOOP, REPLACE ERROR
;CALL WITH 'BR 3$' = 000750
032374 062700 000002 7$: ADD #2,R0 ;POINT R0 TO NEXT PDR TO BE CHECKED
032400 077211 SOB R2,6$ ;LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
032402 010115 MOV R1,(R5) ;WRITE TO THE PDR TESTED TO CLEAR W-BIT
032404 031527 000100 BIT (R5),#WBIT ;DID WRITING PDR CLEAR THE W-BIT?
032410 001401 BEQ 8$ ;BRANCH IF YES
032412 104056 ERROR +56 ;W-BIT DID NOT CLEAR BY WRITING THE PDR
;FOR TIGHTER SCOPE LOOP, REPLACE ERROR CALL
```


032414	062705	000002		8\$:	ADD	#2,R5	:WITH 'BR 3\$' = 000740
032420	062703	020000			ADD	#20000,R3	:POINT R5 TO THE NEXT PDR TO BE TESTED
032424	077445				SOB	R4,3\$:CHANGE VIRT. ADDR TO REF. NEXT PDR
032426	012737	032240	001110		MOV	#1\$,\$LPERR	:LOOP BACK TO 3\$ UNTIL ALL 8 PDR'S TESTED
032434	004737	002226			JSR	PC,TON	:RESET LOOP ON ERROR POINTER TO 1\$
3186 032440	005037	177776			CLR	PSW	:TURN T-BIT BACK ON FOR NEXT EST
							:BACK TO KERNEL MODE BEFORE LEAVING

3197

.SBTTL TEST # 55 - TEST 'W-BIT NOT SET' CASES

 *TEST 55 TEST 'W-BIT NOT SET' CASES

* THIS TEST CHECKS TWO SPECIAL CASES WHERE THE W-BIT DOES
 * NOT GET SET ON A WRITE. FIRST CASE IS THAT THE W-BIT
 * SHOULD NOT SET IN PAGE DESCRIPTOR REG. 7 WHEN WRITING TO
 * STATUS REG SRO (KERNEL PDR 7 IS USED). SECOND CASE IS THAT
 * THE W-BIT IS NOT SET IF THE 'DATO' IS ABORTED DUE TO AN
 * ODD ADDRESS ERROR (KERNEL PDR3 & VIRTUAL ADDR 60001 ARE USED).

3198	032444	000004				TST55: SCOPE	
3199	032446	004737	002172			1\$: JSR PC,TOFF	:TURN OFF T-BIT TRAPPING FOR THIS TEST
3200	032452	012701	077406			MOV #77406,R1	:PUT 'W-BIT OFF' VALUE FOR PDR IN R1
3201	032456	012737	032456	001110		2\$: MOV #2\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 2\$
3202	032464	010137	172306			MOV R1,KIPDR3	:LOAD KERNEL PDR3 WITH 77406 TO CLEAR W-BIT
3203	032470	012737	032502	000004		MOV #3\$,ERRVEC	:SET UP LOC. 4 TO 3\$ FOR ODD ADDR. ABORT
3204	032476	005237	060001			INC 60001	:CAUSE ODD ADDRESS ABORT THRU LOC. 4
3205	032502	012706	001100			3\$: MOV #KERSTK,KSP	:RESTORE THE STACK POINTER
3206	032506	013702	172306			MOV KIPDR3,R2	:READ KIPDR3 INTO R2
3207	032512	020102				CMP R1,R2	:WAS W-BIT LEFT CLEARED?
3208	032514	001401				BEQ 4\$:BRANCH IF YES
3209	032516	104057				ERROR +57	:W-BIT GOT SET DURING AN ODD ADDR. ABORT
3210							:FOR TIGHTER SCOPE LOOP
3211							:REPLACE ERROR CALL WITH
3212	032520	012737	003232	000004		4\$: MOV #TIMERR,ERRVEC	:RESTORE NORMAL CPU TRAP ROUTINE TO LOC.4
3213	032526	012737	032446	001110		MOV #1\$, \$LPERR	:RESET LOOP ON ERROR POINTER TO 1\$
3214	032534	004737	002226			JSR PC,TON	:TURN T-BIT TRAPPING BACK ON

3215

```

.SBTTL END OF PASS ROUTINE
:*****
:*INCREMENT THE PASS NUMBER ($PASS)
:*TYPE 'END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYYY'
:*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:
032540          SCOPE
032540 000004   CLR          $STNM          ;;ZERO THE TEST NUMBER
032542 005037   001102   INC          $PASS           ;;INCREMENT THE PASS NUMBER
032546 005237   001232   BIC          #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
032552 042737   100000   001232   DEC          (PC)+          ;;LOOP?
032560 005327   $EOPCT: .WORD      1
032562 000001   BGT          $DOAGN         ;;YES
032564 003072   MOV          (PC)+,@(PC)+   ;;RESTORE COUNTER
032566 012737   $ENDCT: .WORD      1
032570 000001   $EOPCT
032572 032562   TYPE        .65$          ;;TYPE ASCIZ STRING
032574 104401   032602   BR          64$           ;;GET OVER THE ASCIZ
032600 000407   ;;65$: .ASCIZ <12><15>/END PASS #/
032620          64$:
032620 013746   001232   MOV          $PASS,-(SP)    ;;SAVE $PASS FOR TYPEOUT
                                ;;TYPE PASS NUMBER
032624 104405   TYPDS
032626 104401   032634   TYPE        .67$          ;;GO TYPE--DECIMAL ASCII WITH SIGN
032632 000421   BR          66$           ;;TYPE ASCIZ STRING
                                ;;GET OVER THE ASCIZ
                                ;;67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
032676          66$:
032676 013746   001112   MOV          $ERTTL,-(SP)   ;;SAVE $ERTTL FOR TYPEOUT
                                ;;TOTAL NUMBER OF ERRORS
032702 104405   TYPDS
032704 104401   001221   TYPE        ., $CRLF       ;;GO TYPE--DECIMAL ASCII WITH SIGN
032710 005037   001112   CLR          $ERTTL        ;;TYPE CARRIAGE RETURN, LINE FEED
032714 013700   000042   $GET42: MOV          @#42,R0   ;;CLEAR ERROR TOTAL
                                ;;GET MONITOR ADDRESS
032720 001414   BEQ          $DOAGN        ;;BRANCH IF NO MONITOR
032722 005046   CLR          -(SP)         ;;INSURE THE 'T' BIT IS CLEAR
032724 012746   032732   MOV          #$CLR.T,-(SP) ;;SETUP FOR AN RTI OR RTT
032730 000426   BR          $RTRN         ;;GO DO AN RTI OR RTT TO LOAD THE PSW
                                ;;WITH A CLEARED 'T' BIT
032732          $CLR.T:
032732 013700   000042   MOV          @#42,R0       ;;INSURE R0 CONTAINS THE MONITORS
032736 001405   BEQ          $DOAGN        ;;RETURN ADDRESS
032740 000005   RESET
032742 004710   $ENDAD: JSR          PC,(R0) ;;CLEAR THE WORLD
                                ;;GO TO MONITOR
032744 000240   NOP
032746 000240   NOP                       ;;SAVE ROOM
032750 000240   NOP                       ;;FOR
032752          $DOAGN:
032752          TRAP          ;;ACT!!
032754 042716   000020   BIC          #20,(SP)      ;;PUSH OLD PSW AND PC ON STACK
032760 032777   010000   146152   BIT          #BIT12,@SWR   ;;CLEAR THE 'T' BIT
                                ;;RUN WITH TRACE TRAP?
032766 001005   BNE          1$           ;;BR IF NO
032770 005137   001346   CG:        $TBIT         ;;IS IT TIME FOR TRACE TRAP
032774 100402   BMI          1$           ;;BR IF NO
032776 052716   000020   BIS          #20,(SP)     ;;SET TRACE TRAP

```

```
033002 012746 033010 1$: MOV #SLOOP,-(SP) ;; JUMP TO START OF TEST
033006 000002 $RTRN: RTI ;; RETURN--THIS IS CHANGED TO
;; AN 'RTI' IF 'RTI' IS A LEGAL
;; INSTRUCTION

033010 $LOOP:
033010 000137 JMP @PC)+ ;; RETURN
033012 020454 $RTNAD: .WORD LOOP
033014 377 377 000 $ENULL: .BYTE -1,-1,0 ;; NULL CHARACTER STRING
.EVEN
```

3217

.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
    
```

```

033020          $SCOPE:
033020          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
033022          104410          1$: BIT #BIT14,@SWR          ;;LOOP ON PRESENT TEST?
033030          032777 040000 146110 BNE $OVER          ;;YES IF SW14=1
033030          001077          :####START OF CODE FOR THE XOR TESTER####
033032          000416          $XTSTR: BR 6$          ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
          MOV @WERRVEC,-(SP)          ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
033034          013746 000004          MOV #5$,@WERRVEC          ;;SAVE THE CONTENTS OF THE ERROR VECTOR
033040          012737 033060 000004          TST @#177060          ;;SET FOR TIMEOUT
033046          005737 177060          MOV (SP)+,@WERRVEC          ;;TIME OUT ON XOR?
033052          012637 000004          BR $SVLAD          ;;RESTORE THE ERROR VECTOR
033056          000446          5$: CMP (SP)+,(SP)+          ;;GO TO THE NEXT TEST
033060          022626          MOV (SP)+,@WERRVEC          ;;CLEAR THE STACK AFTER A TIME OUT
033062          012637 000004          BR 7$          ;;RESTORE THE ERROR VECTOR
033066          000434          6$:;####END OF CODE FOR THE XOR TESTER####
033070          032777 000400 146042          BIT #BIT08,@SWR          ;;LOOP ON SPEC. TEST?
033076          001404          BEQ 2$          ;;BR IF NO
033100          127737 146034 001102          CMPB @SWR,$STNM          ;;ON THE RIGHT TEST? SWR<7:0>
033106          001450          BEQ $OVER          ;;BR IF YES
033110          013737 177766 033244 2$: MOV 177766,CPSAVE          ;;MOVE CPU ERR REG VALUE TO LOC FOR TST ;DPM001
033116          032737 000001 033244          BIT #BIT00,CPSAVE          ;;SEE IF THE POWER MONITOR BIT IS ON ;DPM001
033124          001406          BEQ 2000$          ;;BRANCH TO CONTINUE ROUTINE IF CLEAR ;DPM001
033126          042737 000001 177766          BIC #BIT00,177766          ;;CLEAR THE BIT FOUND TO BE SET ;DPM001
033134          104177          EMT +177          ;;CALL SPECIAL POWER FAIL BIT ERROR CALL ;DPM001
033136          105037 001103          CLRB $ERFLG          ;;CLEAR THE FLAG FOR THE NEXT CHECK ;DPM001
033142          105737 001103          2000$: TSTB $ERFLG          ;;DID AN ERROR OCCUR?
033146          001412          BEQ $SVLAD          ;;BR IF NO
033150          032777 001000 145762          BIT #BIT09,@SWR          ;;LOOP ON ERROR?
033156          001404          BEQ 4$          ;;BR IF NO
033160          013737 001110 001106 7$: MOV $LPERR,$LPADR          ;;SET LOOP ADDRESS TO LAST SCOPE
033166          000420          BR $OVER
033170          105037 001103          4$: CLRB $ERFLG          ;;ZERO THE ERROR FLAG
033174          105237 001102          $SVLAD: INCB $STNM          ;;COUNT TEST NUMBERS
033200          113737 001102 001230          MOVB $STNM,$TESTN          ;;SET TEST NUMBER IN APT MAILBOX
033206          011637 001106          MOV (SP),$LPADR          ;;SAVE SCOPE LOOP ADDRESS
033212          011637 001110          MOV (SP),$LPERR          ;;SAVE ERROR LOOP ADDRESS
033216          005037 001212          CLR $ESCAPE          ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
033222          112737 000001 001115          MOVB #1,$ERMAX          ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
033230          013777 001102 145704 $OVER: MOV $STNM,@DISPLAY          ;;DISPLAY TEST NUMBER
033236          013716 001106          MOV $LPADR,(SP)          ;;FUDGE RETURN ADDRESS
033242          000002          RTI          ;;FIXES PS
033244          000000          CPSAVE: .WORD 0          ;;LOCATION TO SAVE CPU ERR REG CONTENTS ;DPM001
    
```

3219

.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
  
```

```

033246 105037 033624 $ERROR: CLRB      IBSAVE          ;CLEAR THE ITEM BYTE SAVE LOCATION      ;DPM001
033252 104410          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
033254 010037 001162      MOV      R0,$REG0      ;SAVE THE CONTENTS OF R0
033260 010137 001164      MOV      R1,$REG1      ;SAVE THE CONTENTS OF R1
033264 010237 001166      MOV      R2,$REG2      ;SAVE THE CONTENTS OF R2
033270 010337 001170      MOV      R3,$REG3      ;SAVE THE CONTENTS OF R3
033274 010437 001172      MOV      R4,$REG4      ;SAVE THE CONTENTS OF R4
033300 010537 001174      MOV      R5,$REG5      ;SAVE THE CONTENTS OF R5
033304 113737 001102 001254      MOV      $STNM,TESTNO    ;SAVE THE TEST NUMBER
033312 105237 001103      7$:      INCB      $ERFLG          ;;SET THE ERROR FLAG
033316 001775          BEQ      7$          ;;DON'T LET THE FLAG GO TO ZERO
033320 013777 001102 145614      MOV      $STNM,@DISPLAY  ;;DISPLAY TEST NUMBER AND ERROR FLAG
033326 032777 002000 145604      BIT      #BIT10,@SWR    ;;BELL ON ERROR?
033334 001402          BEQ      1$          ;;NO - SKIP
033336 104401 001214          TYPE      ,SBELL        ;;RING BELL
033342 005237 001112      1$:      INC      $ERTTL        ;;COUNT THE NUMBER OF ERRORS
033346 011637 001116      MOV      (SP),$ERRPC    ;;GET ADDRESS OF ERROR INSTRUCTION
033352 162737 000002 001116      SUB      #2,$ERRPC
033360 117737 145532 001114      MOV      @ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
033366 122737 000177 001114      CMPB    #177,$ITEMB    ;SEE IF THIS IS THE POWER FAIL CALL      ;DPM001
033374 001426          BEQ      1001$        ;BRANCH AROUND ROUTINE IF IT IS          ;DPM001
033376 105737 033624      TSTB    IBSAVE        ;SEE IF THIS IS THE 2ND ERROR CALL      ;DPM001
033402 001021          BNE      1000$        ;BRANCH IF SO                          ;DPM001
033404 013737 177766 033244      MOV      177766,CPSAVE  ;MOVE CPU ERR REG TO CPSAVE FOR TEST    ;DPM001
033412 032737 000001 033244      BIT      #BIT00,CPSAVE  ;SEE IF POWER MONITOR BIT IS SET        ;DPM001
033420 001414          BEQ      1001$        ;BRANCH IF OK                          ;DPM001
033422 042737 000001 177766      BIC      #BIT00,177766  ;CLEAR THE BIT FOUND SET                ;DPM001
033430 113737 001114 033624      MOV      $ITEMB,IBSAVE  ;MAKE IBSAVE NON-ZERO FOR DUAL CALL      ;DPM001
033436 112737 000177 001114      MOV      #177,$ITEMB    ;SET $ITEMB TO SPECIAL POWER FAIL PNTR  ;DPM001
033444 000402          BR      1001$        ;BRANCH OVER IBSAVE CLEARING            ;DPM001
033446 105037 033624      1000$:  CLRB      IBSAVE        ;CLEAR IBSAVE SO AFTER 2ND ERROR, EXIT  ;DPM001
033452          1001$:
033452 032777 020000 145460      BIT      #BIT13,@SWR    ;;SKIP TYPEOUT IF SET
033460 C01004          BNE      20$          ;;SKIP TYPEOUTS
033462 004737 033626      JSR      PC,ERRYP      ;;GO TO USER ERROR ROUTINE
033466 104401 001221          TYPE      ,$CRLF
033472          20$:
033472 122737 000001 001244      CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
033500 001007          BNE      2$          ;;NO,SKIP APT ERROR REPORT
033502 113737 001114 033514      MOV      $ITEMB,21$    ;;SET ITEM NUMBER AS ERROR NUMBER
033510 004737 035752          JSR      PC,$ATY4      ;;REPORT FATAL ERROR TO APT
033514          21$:      .BYTE    0
033515          .BYTE    0
  
```

```

033516 000777          22$: BR      22$      ;;APT ERROR LOOP
033520 105737 033624  2$:  TSTB   IBSAVE  ;;SEE IF POWER FAIL ERROR CALL      ;DPM001
033524 001005          BNE     3$      ;;BRANCH IF NOT - HALT NOT ALLOWED  ;DPM001
033526 005777 145406  TST     @SWR    ;;HALT ON ERROR
033532 100002          BPL     3$      ;;SKIP IF CONTINUE
033534 000000          HALT                    ;;HALT ON ERROR!
033536 104410          CKSWR                    ;;TEST FOR CHANGE IN SOFT-SWR
033540 032777 001000 145372 3$:  BIT     #BIT09,@SWR ;;LOOP ON ERROR SWITCH SET?
033546 001405          BEQ     4$      ;;BR IF NO
033550 105737 033624  TSTB   IBSAVE  ;;SEE IF THIS ERROR CALL IS PWR MON ERR ;DPM001
033554 001256          BNE     7$      ;;BRANCH BACK IF SO - DON'T ALLOW FUDGING;DPM001
033556 013716 001110  MOV     $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
033562 005737 001212  4$:  TST     $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
033566 001405          BEQ     5$      ;;BR IF NONE
033570 105737 033624  TSTB   IBSAVE  ;;SEE IF THIS ERROR CALL IS PWR MON ERR ;DPM001
033574 001246          BNE     7$      ;;BRANCH BACK IF SO - DON'T ALLOW FUDGING;DPM001
033576 013716 001212  MOV     $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
033602 022737 032742 000042 5$:  CMP     #SENDAD,@#42 ;;ACT-11 AUTO-ACCEPT?
033610 001001          BNE     6$      ;;BRANCH IF NO
033612 000000          HALT                    ;;YES
033614 105737 033624  6$:  TSTB   IBSAVE  ;;SEE IF THIS IS THE PWR FAIL ERROR CALL ;DPM001
033620 001234          BNE     7$      ;;BRANCH BACK TO CALL ORIGINAL ERR IF SO ;DPM001
033622 000002          RTI                    ;;RETURN
033624 000000          IBSAVE: .WORD 0 ;;LOC'N TO HOLD $ITEMB DURING DUAL ERR ;DPM001
    
```

```

3221 033626 104401 001221  ERRRTYP: TYPE    , $CRLF      ;'CARRIAGE RETURN' & 'LINE FEED'
3222 033632 010046          MOV    RO,-(KSP)  ;SAVE RO.
3223 033634 005000          CLR    RO         ;PICKUP THE ITEM INDEX
3224 033636 153700 001114  BISB   $ITEMB,RO
3225 033642 001004          BNE   1$         ;IF ITEM NUMBER IS ZERO, JUST
3226                                ;TYPE THE PC OF THE ERROR
3227 033644 013746 001116  MOV    $ERRPC,-(SP) ;:SAVE $ERRPC FOR TYPEOUT
3228                                ;:ERROR ADDRESS
3229 033650 104402          TYPDC          ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
3230 033652 000530          BR    13$       ;GET OUT
3231 033654 122700 000177  1$:  CMPB  #177,RO   ;SEE IF THIS ERROR CALL IS THE POWER MONITOR BIT CALL
3232 033660 001003          BNE   100$     ;BRANCH IF NOT
3233 033662 012700 034150  MOV    #PFECWS,RO ;MOVE ADDRESS OF POWER MONITOR BIT ERROR TO RO
3234 033666 000406          BR    110$     ;BRANCH TO CALL THE ERROR
3235 033670          100$:
3236 033670 005300          DEC    RO       ;ADJUST THE INDEX SO THAT IT WILL
3237 033672 006300          ASL   RO       ;WORK FOR THE ERROR TABLE.
3238 033674 006300          ASL   RO
3239 033676 006300          ASL   RO
3240 033700 062700 001372  ADD    #$ERRTB,RO ;FORM TABLE POINTER
3241 033704 012037 033714  110$: MOV    (RO)+,2$  ;PICKUP 'ERROR MESSAGE' POINTER
3242 033710 001404          BEQ   3$       ;SKIP TYPEOUT IF NO POINTER
3243 033712 104401          TYPE          ;TYPE THE 'ERROR MESSAGE'
3244 033714 000000          .WORD 0       ;'ERROR MESSAGE' POINTER GOES HERE
3245 033716 104401 001221  2$:  TYPE    , $CRLF  ;'CARRIAGE RETURN' & 'LINE FEED'
3246 033722 012037 033732  3$:  MOV    (RO)+,4$  ;PICKUP 'DATA HEADER' POINTER
3247 033726 001404          BEQ   5$       ;SKIP TYPEOUT IF 0
3248 033730 104401          TYPE          ;TYPE THE 'DATA HEADER'
3249 033732 000000          .WORD 0       ;'DATA HEADER' POINTER GOES HERE
3250 033734 104401 001221  4$:  TYPE    , $CRLF  ;'CARRIAGE RETURN' & 'LINE FEED'
3251 033740 010146          MOV    R1,-(KSP) ;SAVE R1
3252 033742 012001          MOV    (R0)+,R1 ;PICKUP 'DATA TABLE' POINTER
3253 033744 001472          BEQ   12$     ;BR IF NO DATA TO BE TYPED
3254 033746 012000          MOV    (R0)+,RO ;PICKUP 'DATA FORMAT' POINTER
3255 033750 105710          6$:  TSTB  (R0)      ;IS IT FORMAT 0?
3256 033752 001003          BNE   7$       ;BR IF NO
3257                                ;*THIS CODE IS FOR OCTAL (16-BIT) FORMAT (DF=0)
3258 033754 013146          MOV    @(R1)+,-(SP) ;:SAVE @(R1)+ FOR TYPEOUT
3259 033756 104402          TYPDC          ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
3260 033760 000456          BR    11$
3261                                ;*THIS CODE IS FOR DECIMAL FORMAT (DF=1)
3262 033762 121027 000001  7$:  CMPB  (R0),#1   ;IS IT FORMAT 1?
3263 033766 001003          BNE   8$       ;BRANCH IF NO
3264 033770 013146          MOV    @(R1)+,-(SP) ;:SAVE @(R1)+ FOR TYPEOUT
3265 033772 104405          TYPDS          ;:GO TYPE--DECIMAL ASCII WITH SIGN
3266 033774 000450          BR    11$
3267                                ;*THIS CODE IS FOR BINARY FORMAT (DF=2)
3268 033776 121027 000002  8$:  CMPB  (R0),#2   ;IS IT FORMAT 2
3269 034002 001003          BNE   9$       ;BRANCH IF NO
3270 034004 013146          MOV    @(R1)+,-(SP) ;:SAVE @(R1)+ FOR TYPEOUT
3271 034006 104406          TYPBN          ;:GO TYPE--BINARY ASCII
3272 034010 000442          BR    11$
3273                                ;*THIS CODE IS FOR OCTAL (22-BIT) FORMAT (DF=3)
3274 034012 121027 000003  9$:  CMPB  (R0),#3   ;IS IT FORMAT 3?
3275 034016 001011          BNE   15$     ;BRANCH IF NO
3276 034020 012146          MOV    (R1)+,-(KSP) ;PUT ADDRESS OF FIRST LOC. ON STACK
3277 034022 004737 037024  JSR   PC,$DB20  ;CONVERT TWO LOCS. TO AN ASCII STRING

```


ERROR HANDLER ROUTINE

```

3278 034026 062716 000003      ADD    #3,(KSP)      ;ONLY NEED 8 CHARACTERS NOT 11
3279 034032 012637 034040      MOV    (KSP)+,10$   ;PUT ADDRESS OF ASCII CHARS. AT 10$
3280 034036 104401                TYPE                                ;TYPE OCTAL VALUE OF 22-BIT BINARY NO.
3281 034040 000000                10$: .WORD 0
3282                                ;*THIS CODE IS FOR OCTAL (22-BIT) FORMAT FOR A PAR LEFT SHIFTED 6 (DF=4)
3283 034042 010246                15$: MOV    R2,-(KSP)   ;SAVE R2 ON STACK
3284 034044 010346                MOV    R3,-(KSP)   ;SAVE R3 ON STACK
3285 034046 013103                MOV    @ (R1)+,R3  ;LOAD DATA WORD INTO R3
3286 034050 005002                CLR    R2          ;R2 HOLDS UPPER SIX BITS OF NUMBER
3287 034052 073227 000006                ASHC  #6,R2        ;SHIFT VALUE LEFT 6 TIMES
3288 034056 010237 001206                MOV    R2,$TMP4    ;HOLDS LOWER 16 BITS OF ADDRESS
3289 034062 010337 001210                MOV    R3,$TMP5    ;HOLDS UPPER 6 BITS OF ADDRESS
3290 034066 012746 001206                MOV    # $TMP4,-(KSP) ;PUT ADDRESS OF LOWER BITS ONTO STACK
3291 034072 004737 037024                JSR    PC,$DB20    ;CONVERT TWO LOCS. TO AN ASCII STRING
3292 034076 062716 000003                ADD    #3,(KSP)   ;ONLY NEED 8 CHARACTERS NOT 11
3293 034102 012637 034110                MOV    (KSP)+,16$  ;PUT ADDRESS OF ASCII CHARS. AT 16$
3294 034106 104401                TYPE                                ;TYPE OCTAL VALUE OF 22-BIT BINARY NO.
3295 034110 000000                16$: .WORD 0
3296 034112 012603                MOV    (KSP)+,R3   ;RESTORE R3
3297 034114 012602                MOV    (KSP)+,R2   ;RESTORE R2
3298 034116 005711                11$: TST    (R1)      ;IS THERE ANOTHER NUMBER?
3299 034120 001404                BEQ    12$         ;BR IF NO
3300 034122 104401 034144                TYPE  ,14$        ;TYPE TWO(2) SPACES
3301 034126 105720                TSTB  (R0)+        ;POINT TO NEW 'DATA FORMAT'
3302 034130 000707                BR    6$          ;LOOP
3303 034132 012601                12$: MOV    (KSP)+,R1 ;RESTORE R1
3304 034134 012600                13$: MOV    (KSP)+,R0 ;RESTORE R0
3305 034136 104401 001221                TYPE  ,$CRLF      ;'CARRIAGE RETURN' & 'LINE FEED'
3306 034142 000207                RTS    PC          ;RETURN
3307 034144 040 040 000 14$: .ASCIZ  / /          ;TWO(2) SPACES
3308                                .EVEN
3309 034150 034160 034220 034250 PFECWS: .WORD  PFECM,PFECDH,PFECDT,PFECDF
3310 034160 120 117 127 PFECM: .ASCIZ  ?POWER MONITOR BIT WAS FOUND SET?
3311 034220 124 105 123 PFECDH: .ASCIZ  ?TESTNO ERR PC CPUERR?
3312                                .EVEN
3313 034250 001230 001116 033244 PFECDT: .WORD  $TESTN,$ERRPC,$CPSAVE,0
3314 034260 000 000 000 PFECDF: .BYTE  0,0,0,0

```

3315

```

.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.
034264 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ;;IS THE SOFT-SWR SELECTED?
034272 001114 BNE 15$ ;;BRANCH IF NO
034274 105777 144644 TSTB @STKS ;;CHAR THERE?
034300 100111 BPL 15$ ;;IF NO, DON'T WAIT AROUND
034302 117746 144640 MOVB @STKB,-(SP) ;;SAVE THE CHAR
034306 042716 177600 BIC #^C177,(SP) ;;STRIP-OFF THE ASCII
034312 022726 000007 CMP #7,(SP)+ ;;IS IT A CONTROL G?
034316 001102 BNE 15$ ;;NO, RETURN TO USER
034320 123727 001134 000001 CMPB $AUTOB,#1 ;;ARE WE RUNNING IN AUTO-MODE?
034326 001476 BEQ 15$ ;;BRANCH IF YES
034330 104401 035231 TYPE ,SCNTLG ;;ECHO THE CONTROL-G (^G)
034334 104401 035236 $GTSWR: TYPE ,SMSWR ;;TYPE CURRENT CONTENTS
034340 013746 000176 MOV SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
034344 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
034346 104401 035247 TYPE ,SMNEW ;;PROMPT FOR NEW SWR
034352 005046 19$: CLR -(SP) ;;CLEAR COUNTER
034354 005046 CLR -(SP) ;;THE NEW SWR
034356 105777 144562 7$: TSTB @STKS ;;CHAR THERE?
034362 100375 BPL 7$ ;;IF NOT TRY AGAIN
034364 117746 144556 MOVB @STKB,-(SP) ;;PICK UP CHAR
034370 042716 177600 BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII
034374 021627 000003 CMP (SP),#3 ;;IS IT A CONTROL-C?
034400 001015 BNE 9$ ;;BRANCH IF NOT
034402 104401 001350 TYPE ,SCNTLC ;;YES, ECHO CONTROL-C (^C)
034406 062706 000006 ADD #6,SP ;;CLEAN UP STACK
034412 123727 001135 000001 CMPB $INTAG,#1 ;;REENABLE TTY KEYBOARD INTERRUPTS?
034420 001003 BNE 8$ ;;BRANCH IF NO
034422 012777 000100 144514 MOV #100,@STKS ;;ALLOW TTY KEYBOARD INTERRUPTS
034430 000137 035260 8$: JMP CNTRLC ;;CONTROL-C RESTART
034434 021627 000025 9$: CMP (SP),#25 ;;IS IT A CONTROL-U?
034440 001005 BNE 10$ ;;BRANCH IF NOT
034442 104401 035224 TYPE ,SCNTLU ;;YES, ECHO CONTROL-U (^U)
034446 062706 000006 20$: ADD #6,SP ;;IGNORE PREVIOUS INPUT
034452 000737 BR 19$ ;;LET'S TRY IT AGAIN
034454 021627 000015 10$: CMP (SP),#15 ;;IS IT A <CR>?
034460 001022 BNE 16$ ;;BRANCH IF NO
034462 005766 000004 TST 4(SP) ;;YES, IS IT THE FIRST CHAR?
034466 001403 BEQ 11$ ;;BRANCH IF YES
034470 016677 000002 144442 MOV 2(SP),@SWR ;;SAVE NEW SWR
034476 062706 000006 11$: ADD #6,SP ;;CLEAN UP STACK
034502 104401 001221 14$: TYPE ,SCRLF ;;ECHO <CR> AND <LF>
034506 123727 001135 000001 CMPB $INTAG,#1 ;;RE-ENABLE TTY KBD INTERRUPTS?
034514 001003 BNE 15$ ;;BRANCH IF NOT
034516 012777 000100 144420 MOV #100,@STKS ;;RE-ENABLE TTY KBD INTERRUPTS
034524 000002 15$: RTI ;;RETURN
034526 004737 035602 16$: JSR PC,$TYPEC ;;ECHO CHAR
034532 021627 000060 CMP (SP),#60 ;;CHAR < 0?
034536 002420 BLT 18$ ;;BRANCH IF YES
034540 021627 000067 CMP (SP),#67 ;;CHAR > 7?

```

```

034544 003015          BGT 18$          ;;BRANCH IF YES
034546 042726 000060  BIC #60,(SP)+    ;;STRIP-OFF ASCII
034552 005766 000002  TST 2(SP)        ;;IS THIS THE FIRST CHAR
034556 001403          BEQ 17$          ;;BRANCH IF YES
034560 006316          ASL (SP)         ;;NO, SHIFT PRESENT
034562 006316          ASL (SP)         ;;CHAR OVER TO MAKE
034564 006316          ASL (SP)         ;;ROOM FOR NEW ONE.
034566 005266 000002  17$: INC 2(SP)    ;;KEEP COUNT OF CHAR
034572 056616 177776  BIC -2(SP),(SP) ;;SET IN NEW CHAR
034576 000667          BR 7$           ;;GET THE NEXT ONE
034600 104401 001220  18$: TYPE $QUES ;;TYPE ?<CR><LF>
034604 000720          BR 20$          ;;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
.
034606 011646          $RDCHR: MOV (SP),-(SP)    ;;PUSH DOWN THE PC
034610 016666 000004 000002 MOV 4(SP),2(SP)  ;;SAVE THE PS
034616 105777 144322  1$: TSTB @TKS    ;;WAIT FOR
034622 100375          BPL 1$           ;;A CHARACTER
034624 117766 144316 000004 MOVB @TKB,4(SP)  ;;READ THE TTY
034632 042766 177600 000004 BIC #^C<177>,4(SP) ;;GET RID OF JUNK IF ANY
034640 026627 000004 000023 CMP 4(SP),#23    ;;IS IT A CONTROL-S?
034646 001013          BNE 3$           ;;BRANCH IF NO
034650 105777 144270  2$: TSTB @TKS    ;;WAIT FOR A CHARACTER
034654 100375          BPL 2$           ;;LOOP UNTIL ITS THERE
034656 117746          MOVB @TKB,-(SP)  ;;GET CHARACTER
034662 042716 177600 BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII
034666 022627 000021  CMP (SP)+,#21    ;;IS IT A CONTROL-Q?
034672 001366          BNE 2$           ;;IF NOT DISCARD IT
034674 000750          BR 1$           ;;YES, RESUME
034676 026627 000004 000021 3$: CMP 4(SP),#$XON ;;IS IT A RANDOM XON? ;RAN001
034704 001744          BEQ 1$           ;;BRANCH IF YES ;RAN001
034706 026627 000004 000140 CMP 4(SP),#140   ;;IS IT UPPER CASE?
034714 002407          BLT 4$           ;;BRANCH IF YES
034716 026627 000004 000175 CMP 4(SP),#175   ;;IS IT A SPECIAL CHAR?
034724 003003          BGT 4$           ;;BRANCH IF YES
034726 042766 000040 000004 BIC #40,4(SP)   ;;MAKE IT UPPER CASE
034734 000002          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
.
034736 010346          $RDLIN: MOV R3,-(SP)  ;;SAVE R3
034740 005046          CLR -(SP)        ;;CLEAR THE RUBOUT KEY
034742 012703 035214  1$: MOV #$TTYIN,R3  ;;GET ADDRESS
034746 022703 035224  2$: CMP #$TTYIN+8.,R3 ;;BUFFER FULL?
034752 101467          BLOS 4$         ;;BR IF YES
034754 104411          RDCHR          ;;GO READ ONE CHARACTER FROM THE TTY
034756 112613          MOVB (SP)+,(R3) ;;GET CHARACTER
034760 122713 000003  CMPB #3,(R3)    ;;IS IT A CONTROL-C?

```

```

034764 001006          BNE      10$          ;;BRANCH IF NO
034766 104401 001350  TYPE      ,%CNTLC      ;;TYPE A CONTROL-C (^C)
034772 005726          TST      (SP)+         ;;CLEAN RUBOUT KEY OFF OF THE STACK
034774 012603          MOV      (SP)+,R3      ;;RESTORE R3
034776 000137 035260  JMP      CNTRLC        ;;GOTO CONTROL-C RESTART
035002 122713 000177 10$:  CMPB    #177,(R3)     ;;IS IT A RUBOUT
035006 001022          BNE      5$           ;;BR IF NO
035010 005716          TST      (SP)         ;;IS THIS THE FIRST RUBOUT?
035012 001007          BNE      6$           ;;BR IF NO
035014 112737 000134 035212 MOVB    #'\,9$        ;;TYPE A BACK SLASH
035022 104401 035212  TYPE      ,9$
035026 012716 177777  MOV      #-1,(SP)     ;;SET THE RUBOUT KEY
035032 005303          DEC      R3           ;;BACKUP BY ONE
035034 020327 035214 6$:  CMP      R3,%$TTYIN   ;;STACK EMPTY?
035040 103434          BLO      4$           ;;BR IF YES
035042 111337 035212  MOVB    (R3),9$      ;;SETUP TO TYPEOUT THE DELETED CHAR.
035046 104401 035212  TYPE      ,9$        ;;GO TYPE
035052 000735          BR       2$           ;;GO READ ANOTHER CHAR.
035054 005716          TST      (SP)         ;;RUBOUT KEY SET?
035056 001406          BEQ      7$           ;;BR IF NO
035060 112737 000134 035212 MOVB    #'\,9$        ;;TYPE A BACK SLASH
035066 104401 035212  TYPE      ,9$
035072 005016          CLR      (SP)         ;;CLEAR THE RUBOUT KEY
035074 122713 000025 7$:  CMPB    #25,(R3)     ;;IS CHARACTER A CTRL U?
035100 001003          BNE      8$           ;;BR IF NO
035102 104401 035224  TYPE      ,%CNTLU     ;;TYPE A CONTROL 'U'
035106 000715          BR       1$           ;;GO START OVER
035110 122713 000022 8$:  CMPB    #22,(R3)     ;;IS CHARACTER A "'R'?"
035114 001011          BNE      3$           ;;BRANCH IF NO
035116 105013          CLRB    (R3)         ;;CLEAR THE CHARACTER
035120 104401 001221  TYPE      ,%CRLF      ;;TYPE A 'CR' & 'LF'
035124 104401 035214  TYPE      ,%TTYIN     ;;TYPE THE INPUT STRING
035130 000706          BR       2$           ;;GO PICKUP ANOTHER CHACTER
035132 104401 001220 4$:  TYPE      ,%QUES      ;;TYPE A '?'
035136 000701          BR       1$           ;;CLEAR THE BUFFER AND LOOP
035140 111337 035212 3$:  MOVB    (R3),9$      ;;ECHO THE CHARACTER
035144 104401 035212  TYPE      ,9$
035150 122723 000015  CMPB    #15,(R3)+     ;;CHECK FOR RETURN
035154 001274          BNE      2$           ;;LOOP IF NOT RETURN
035156 105063 177777  CLRB    -1(R3)       ;;CLEAR RETURN (THE 15)
035162 104401 001222  TYPE      ,%LF        ;;TYPE A LINE FEED
035166 005726          TST      (SP)+         ;;CLEAN RUBOUT KEY FROM THE STACK
035170 012603          MOV      (SP)+,R3     ;;RESTORE R3
035172 011646          MOV      (SP),-(SP)   ;;ADJUST THE STACK AND PUT ADDRESS OF THE
035174 016666 000004 000002 MOV      4(SP),2(SP)   ;; FIRST ASCII CHARACTER ON IT
035202 012766 035214 000004 MOV      %$TTYIN,4(SP)
035210 000002          RTI                ;;RETURN
035212          000          9$:  .BYTE    0            ;;STORAGE FOR ASCII CHAR. TO TYPE
035213          000          .BYTE    0            ;;TERMINATOR
035214          .BLKB   8            ;;RESERVE 8 BYTES FOR TTY INPUT
035224          136          125          015  %CNTLU: .ASCIZ  /^U/<15><12>  ;;CONTROL 'U'
035231          136          107          015  %CNTLG: .ASCIZ  /^G/<15><12>  ;;CONTROL 'G'
035236          015          012          123  SMSWR: .ASCIZ  <15><12>/SWR = /
035247          040          040          116  SMNEW: .ASCIZ  / NEW = /

```

```

3317 .SBTTL CONTROL-C SERVICING ROUTINE
3318 035260 013737 001232 001210 CNTRLC: MOV $PASS,$TMP5 ;GET THE VALUE OF '$PASS'
3319 035266 005237 001210 INC $TMP5 ;FORM CURRENT PASS #
3320 035272 104401 035337 TYPE ,CMMSG ;TYPE THE TEST STOPS HERE
3321 035276 113737 001102 035332 MOVEB $STNM,1$ ;SAVE TEST NUMBER
3322 035304 013746 035332 MOV 1$,-(SP) ;SAVE 1$ FO TYPEOUT
3323 035310 104402 TYPDC ;
3324 035312 104401 035334 TYPE ,2$ ;
3325 035316 013746 001210 MOV $TMP5,-(SP) ;SAVE $TMP5 FOR TYPEOUT
3326 035322 104405 TYPDS ;TYPE ASCII DECIMAL WITH SIGN
3327 035324 104407 GTSWR ;ASK FOR NEW SWR VALUE
3328 035326 000137 032542 JMP $EOP+2 ;JUMP TO END OF PASS + 2
3329 035332 000000 1$: .WORD 0 ;TEST # BUFFER
3330 035334 040 040 000 2$: .ASCII / / ;2 SPACES & STOP MESSAGE
3331 035337 112 125 115 CMMSG: .ASCII /JUMPING TO END OF PASS/<15><12>
3332 035367 000 .BYTE 0
    
```

3333

.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

035370	:05737	001157	\$TYPE:	TSTB	\$TPFLG	:: IS THERE A TERMINAL?
035374	100002			BPL	1\$:: BR IF YES
035376	000000			HALT		:: HALT HERE IF NO TERMINAL
035400	000430			BR	3\$:: LEAVE
035402	010046		1\$:	MOV	RO,-(SP)	:: SAVE RO
035404	017600	000002		MOV	@2(SP),RO	:: GET ADDRESS OF ASCIZ STRING
035410	122737	000001	001244	CMPB	#APTENV,\$ENV	:: RUNNING IN APT MODE
035416	001011			BNE	62\$:: NO,GO CHECK FOR APT CONSOLE
035420	132737	000100	001245	BITB	#APTSPOOL,\$ENVM	:: SPOOL MESSAGE TO APT
035426	001405			BEQ	62\$:: NO,GO CHECK FOR CONSOLE
035430	010037	035440		MOV	RO,61\$:: SETUP MESSAGE ADDRESS FOR APT
035434	004737	035742		JSR	PC,\$ATY3	:: SPOOL MESSAGE TO APT
035440	000000		61\$:	.WORD	0	:: MESSAGE ADDRESS
035442	132737	000040	001245	62\$:	BITB	#APTCSUP,\$ENVM
035450	001003			BNE	60\$:: APT CONSOLE SUPPRESSED
035452	112046		2\$:	MOVB	(RO)+,-(SP)	:: YES,SKIP TYPE OUT
035454	001005			BNE	4\$:: PUSH CHARACTER TO BE TYPED ONTO STACK
035456	005726			TST	(SP)+	:: BR IF IT ISN'T THE TERMINATOR
035460	012600		60\$:	MOV	(SP)+,RO	:: IF TERMINATOR POP IT OFF THE STACK
035462	062716	000002	3\$:	ADD	#2,(SP)	:: RESTORE RO
035466	000002			RTI		:: ADJUST RETURN PC
035470	122716	000011	4\$:	CMPB	#HT,(SP)	:: RETURN
035474	001430			BEQ	8\$:: BRANCH IF <HT>
035476	122716	000200		CMPB	#CRLF,(SP)	:: BRANCH IF NOT <CRLF>
035502	001006			BNE	5\$	
035504	005726			TST	(SP)+	:: POP <CR><LF> EQUIV
035506	104401			TYPE		:: TYPE A CR AND LF
035510	001221			\$CRLF		
035512	105037	035730		CLRB	\$CHARCNT	:: CLEAR CHARACTER COUNT
035516	000755			BR	2\$:: GET NEXT CHARACTER
035520	004737	035602	5\$:	JSR	PC,\$TYPEC	:: GO TYPE THIS CHARACTER
035524	123726	001156	6\$:	CMPB	\$FILLC,(SP)+	:: IS IT TIME FOR FILLER CHARS.?
035530	001350			BNE	2\$:: IF NO GO GET NEXT CHAR.
035532	013746	001154		MOV	\$NULL,-(SP)	:: GET # OF FILLER CHARS. NEEDED
						:: AND THE NULL CHAR.
035536	105366	000001	7\$:	DECB	1(SP)	:: DOES A NULL NEED TO BE TYPED?
035542	002770			BLT	6\$:: BR IF NO--GO POP THE NULL OFF OF STACK
035544	004737	035602		JSR	PC,\$TYPEC	:: GO TYPE A NULL
035550	105337	035730		DECB	\$CHARCNT	:: DO NOT COUNT AS A COUNT
035554	000770			BR	7\$:: LOOP
035556	112716	000040	8\$:	MOVB	#' ,(SP)	:: REPLACE TAB WITH SPACE

:HORIZONTAL TAB PROCESSOR

```

035562 004737 035602 9$: JSR PC,$TYPEC ::TYPE A SPACE
035566 132737 00:007 035730 BITB #7,$CHARCNT ::BRANCH IF NOT AT
035574 001372 BNE 9$ ::TAB STOP
035576 005726 TST (SP)+ ::POP SPACE OFF STACK
035600 000724 BR 2$ ::GET NEXT CHARACTER
035602 $TYPEC:
035602 105777 143336 TSTB @STKS ::CHAR IN KYBD BUFFER? :MJD001
035606 100022 BPL 10$ ::BR IF NOT :MJD001
035610 017746 143332 MOV @STKB,-(SP) ::GET CHAR :MJD001
035614 042716 177600 BIC #177600,(SP) ::STRIP EXTRANEIOUS BITS :MJD001
035620 122716 000023 CMPB #$XOFF,(SP) ::WAS CHAR XOFF :MJD001
035624 001012 BNE 102$ ::BR IF NOT :MJD001
035626 105777 143312 101$: TSTB @STKS ::WAIT FOR CHAR :MJD001
035632 100375 BPL 101$ :MJD001
035634 117716 143306 MOVB @STKB,(SP) ::GET CHAR :MJD001
035640 042716 177600 BIC #177600,(SP) ::STRIP IT :MJD001
035644 122716 000021 CMPB #$XON,(SP) ::WAS IT XON? :MJD001
035650 001366 BNE 101$ ::BR IF NOT :MJD001
035652 102$: TST (SP)+ ::FIX STACK :MJD001
035652 005726 10$: TSTB @STPS ::WAIT UNTIL PRINTER IS READY :MJD001
035654 105777 143270 BPL 10$ :MJD001
035660 100375 CMPB 2(SP),#$XON ::IS CHARACTER A RANDOM XON? :RAN001
035662 126627 000002 000021 BEQ $TYPEX ::BRANCH IF YES :RAN001
035670 001420 MOVB 2(SP),@STPB ::LOAD CHAR TO BE TYPED INTO DATA REG.
035672 116677 000002 143252 CMPB #CR,2(SP) ::IS CHARACTER A CARRIAGE RETURN?
035700 122766 000015 000002 BNE 1$ ::BRANCH IF NO
035706 001003 CLRB $CHARCNT ::YES--CLEAR CHARACTER COUNT
035710 105037 035730 BR $TYPEX ::EXIT
035714 000406 BR $TYPEX ::EXIT
035716 122766 000012 000002 1$: CMPB #LF,2(SP) ::IS CHARACTER A LINE FEED?
035724 001402 BEQ $TYPEX ::BRANCH IF YES
035726 105227 INCB (PC)+ ::COUNT THE CHARACTER
035730 000000 $CHARCNT: WORD 0 ::CHARACTER COUNT STORAGE
035732 000207 $TYPEX: RTS PC
  
```

3335

```

.SBTTL  APT COMMUNICATIONS ROUTINE
*****
035734 112737 000001 036200 $ATY1:  MOVB  #1,$FFLG      ;;TO REPORT FATAL ERROR
035742 112737 000001 036176 $ATY3:  MOVB  #1,$MFLG      ;;TO TYPE A MESSAGE
035750 000403
035752 112737 000001 036200 $ATY4:  MOVB  #1,$FFLG      ;;TO ONLY REPORT FATAL ERROR
035760 $ATYC:
035760 010046                MOV  R0,-(SP)      ;;PUSH R0 ON STACK
035762 010146                MOV  R1,-(SP)      ;;PUSH R1 ON STACK
035764 105737 036176        TSTB  $MFLG        ;;SHOULD TYPE A MESSAGE?
035770 001450                BEQ   $           ;;IF NOT: BR
035772 122737 000001 001244  CMPB  #APTENV,$ENV  ;;OPERATING UNDER APT?
036000 001031                BNE   $           ;;IF NOT: BR
036002 132737 000100 001245  BITB  #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
036010 001425                BEQ   $           ;;IF NOT: BR
036012 017600 000004        MOV  @4(SP),R0      ;;GET MESSAGE ADDR.
036016 062766 000002 000004  ADD  #2,4(SP)      ;;BUMP RETURN ADDR.
036024 005737 001224        1$:   TST  $MSGTYPE      ;;SEE IF DONE W/ LAST XMISSION?
036030 001375                BNE   $           ;;IF NOT: WAIT
036032 010037 001240        MOV  R0,$MSGAD      ;;PUT ADDR IN MAILBOX
036036 105720 2$:   TSTB  (R0)+      ;;FIND END OF MESSAGE
036040 001376                BNE   $           ;;
036042 163700 001240        SUB  $MSGAD,R0      ;;SUB START OF MESSAGE
036046 006200                ASR  R0            ;;GET MESSAGE LGTH IN WORDS
036050 010037 001242        MOV  R0,$MSGGLT     ;;PUT LENGTH IN MAILBOX
036054 012737 000004 001224  MOV  #4,$MSGTYPE   ;;TELL APT TO TAKE MSG.
036062 000413                BR   $           ;;
036064 017637 000004 036110 3$:   MOV  @4(SP),4$      ;;PUT MSG ADDR IN JSR LINKAGE
036072 062766 000002 000004  ADD  #2,4(SP)      ;;BUMP RETURN ADDRESS
036100 013746 177776        MOV  177776,-(SP)   ;;PUSH 177776 ON STACK
036104 004737 035370        JSR  PC,$TYPE      ;;CALL TYPE MACRO
036110 000000        4$:   .WORD  0
036112 5$:
036112 105737 036200        10$:  TSTB  $FFLG        ;;SHOULD REPORT FATAL ERROR?
036116 001416                BEQ   $           ;;IF NOT: BR
036120 005737 001244        TST  $ENV          ;;RUNNING UNDER APT?
036124 001413                BEQ   $           ;;IF NOT: BR
036126 005737 001224        11$:  TST  $MSGTYPE      ;;FINISHED LAST MESSAGE?
036132 001375                BNE   $           ;;IF NOT: WAIT
036134 017637 000004 001226  MOV  @4(SP),$FATAL ;;GET ERROR #
036142 062766 000002 000004  ADD  #2,4(SP)      ;;BUMP RETURN ADDR.
036150 005237 001224        INC  $MSGTYPE      ;;TELL APT TO TAKE ERROR
036154 105037 036200        12$:  CLRB  $FFLG        ;;CLEAR FATAL FLAG
036160 105037 036177        CLRB  $LFLG        ;;CLEAR LOG FLAG
036164 105037 036176        CLRB  $MFLG        ;;CLEAR MESSAGE FLAG
036170 012601                MOV  (SP)+,R1      ;;POP STACK INTO R1
036172 012600                MOV  (SP)+,R0      ;;POP STACK INTO R0
036174 000207                RTS   PC          ;;RETURN
036176 000                $MFLG: .BYTE 0      ;;MESSG. FLAG
036177 000                $LFLG: .BYTE 0      ;;LOG FLAG
036200 000                $FFLG: .BYTE 0      ;;FATAL FLAG
                .EVEN
000200  APTSIZE=200
000001  APTENV=001
000100  APTSPOOL=100
000040  APTCSUP=040

```


3337

.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
036202	010146			\$TYPBN:	MOV	R1,-(SP)	::SAVE R1 ON THE STACK
036204	016601	000006			MOV	6(SP),R1	::GET THE INPUT NUMBER
036210	000261				SEC		::SET 'C' SO CAN KEEP TRACK OF THE NUMBER OF BITS
036212	112737	000060	036254	1\$:	MOVB	#'0,\$BIN	::SET CHARACTER TO AN ASCII '0'.
036220	006101				ROL	R1	::GET THIS BIT
036222	001406				BEQ	2\$::DONE?
036224	105537	036254			ADCB	\$BIN	::NO--SET THE CHARACTER EQUAL TO THIS BIT
036230	104401	036254			TYPE	,\$BIN	::GO TYPE THIS BIT
036234	000241				CLC		::CLEAR 'C' SO CAN KEEP TRACK OF BITS
036236	000765				BR	1\$::GO DO THE NEXT BIT
036240	012601			2\$:	MOV	(SP)+,R1	::POP THE STACK INTO R1
036242	016666	000002	000004		MOV	2(SP),4(SP)	::ADJUST THE STACK
036250	012616				MOV	(SP)+,(SP)	
036252	000002				RTI		::RETURN TO USER
036254	000	000		\$BIN:	.BYTE	0,0	::STORAGE FOR ASCII CHAR. AND TERMINATOR

036432	110337	036476		MOVB	R3,8\$::SAVE FOR TYPING
036436	104401	036476		TYPE	.8\$::GO TYPE THIS DIGIT
036442	105337	036500	7\$:	DECB	\$OCNT	::COUNT BY 1
036446	003347			BGT	2\$::BR IF MORE TO DO
036450	002402			BLT	6\$::BR IF DONE
036452	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
036454	000744			BR	2\$::GO DO THE LAST DIGIT
036456	012605		6\$:	MOV	(SP)+,R5	::RESTORE R5
036460	012604			MOV	(SP)+,R4	::RESTORE R4
036462	012603			MOV	(SP)+,R3	::RESTORE R3
036464	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
036472	012616			MOV	(SP)+,(SP)	
036474	000002			RTI		::RETURN
036476	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
036477	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
036500	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
036501	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
036502	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE

3341

```

.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$:      CLR      R0            ;;ZERO THE CONSTANTS INDEX
MOV      #$DBLK,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$:      CLRB     (R3)          ;;SET THE TERMINATOR
MOV      (SP)+,R5      ;;POP STACK INTO R5
MOV      (SP)+,R3      ;;POP STACK INTO R3
MOV      (SP)+,R2      ;;POP STACK INTO R2
MOV      (SP)+,R1      ;;POP STACK INTO R1
MOV      (SP)+,R0      ;;POP STACK INTO R0
TYPE     ,$DBLK        ;;NOW TYPE THE NUMBER
    
```

```

036504
036504 010046
036506 010146
036510 010246
036512 010346
036514 010546
036516 012746 020200
036522 016605 000020
036526 100004
036530 005405
036532 112766 000055 000001
036540 005000
036542 012703 036720
036546 112723 000040
036552 005002
036554 016001 036710
036560 160105
036562 002402
036564 005202
036566 000774
036570 060105
036572 005702
036574 001002
036576 105716
036600 100407
036602 106316
036604 103003
036606 116663 000001 177777
036614 052702 000060
036620 052702 000040
036624 110223
036626 005720
036630 020027 000010
036634 002746
036636 003002
036640 010502
036642 000764
036644 105726
036646 100003
036650 116663 177777 177776
036656 105013
036660 012605
036662 012603
036664 012602
036666 012601
036670 012600
036672 104401 036720
    
```

```
036676 016666 000002 000004      MOV      2(SP),4(SP)      ;;ADJUST THE STACK
036704 012616                      MOV      (SP)+,(SP)
036706 000002                      RTI          ;;RETURN TO USER
036710 023420                      $DTBL: 10000.
036712 001750                      1000.
036714 000144                      100.
036716 000012                      10.
036720                      $DBLK: .BLKW 4
```

3343

```

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

```

```

036730
036730 010046
036732 010146
036734 010246
036736 010346
036740 010446
036742 010546
036744 016646 000022
036750 016646 000022
036754 016646 000022
036760 016646 000022
036764 000002

```

```

036766
036766 012666 000022
036772 012666 000022
036776 012666 000022
037002 012666 000022
037006 012605
037010 012604
037012 012603
037014 012602
037016 012601
037020 012600
037022 000002

```

3345

```

.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, @#$DB20      ;; CALL THE ROUTINE
:*      RETURN   ;; THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK
$DB20: SAVREG      ;; SAVE ALL REGISTERS
      MOV      2(SP), R1          ;; PICKUP THE POINTER TO LOW WORD
      MOV      #$OCTVL+13., R5   ;; POINTER TO DATA TABLE
      MOV      #12., R4          ;; DO ELEVEN CHARACTERS
      MOV      #^C7, R3          ;; MASK
      MOV      (R1)+, R0         ;; LOWER WORD
      MOV      (R1)+, R1         ;; HIGH WORD
      CLR      R2                ;; TERMINATOR
1$:     MOVB    R2, -(R5)         ;; PUT CHARACTER IN DATA TABLE
      MOV      R0, R2           ;; GET THIS DIGIT
      DEC     R4                ;; COUNT THIS CHARACTER
      BGT     3$               ;; BR IF NOT THE LAST DIGIT
      BEQ     2$               ;; BR IF IT IS THE LAST DIGIT
      INC     R5                ;; ALL DIGITS DONE-ADJUST POINTER FOR FIRST
      MOV     R5, 2(SP)         ;; ASCII CHAR. & PUT IT ON THE STACK
      RESREG   ;; RESTORE ALL REGISTERS
      RTS     PC                ;; RETURN TO USER
2$:     ASR     R3                ;; POSITION THE MASK FOR THE LAST DIGIT
3$:     ROR     R1                ;; POSITION THE BINARY NUMBER FOR
      ROR     R0                ;; THE NEXT OCTAL DIGIT
      ROR     R1
      ROR     R0
      ROR     R1
      ROR     R0
      BIC     R3, R2            ;; MASK OUT ALL JUNK
      ADD     #'0, R2          ;; MAKE THIS CHAR. ASCII
      BR     1$                ;; GO PUT IT IN THE DATA TABLE
$OCTVL: .BLKB 14.            ;; RESERVE DATA TABLE
    
```

```

037024 104413
037026 016601 000002
037032 012705 037143
037036 012704 000014
037042 012703 177770
037046 012100
037050 012101
037052 005002
037054 110245
037056 010002
037060 005304
037062 003007
037064 001405
037066 005205
037070 010566 000002
037074 104414
037076 000207
037100 006203
037102 006001
037104 006000
037106 006001
037110 006000
037112 006001
037114 006000
037116 040302
037120 062702 000060
037124 000753
037126
    
```

3347

.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.

037144 010046
037146 016600 000002
037152 005740
037154 111000
037156 006300
037160 016000 037200
037164 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

037166 011646
037170 016666 000004 000002
037176 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.

ROUTINE

037200 037166
037202 035370
037204 036302
037206 036256
037210 036316
037212 036504
037214 036202
037216 034334
037220 034264
037222 034606
037224 034736
037226 036730
037230 036766

```
$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
```


3349

.SBTTL POWER DOWN AND UP ROUTINES

:POWER DOWN ROUTINE

```

037232 012737 037410 000024 $PWRDN: MOV    #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
037240 012737 000340 000026      MOV    #340,@#PWRVEC+2 ;;PRIO:7
037246 010046           MOV    R0,-(SP)      ;;PUSH R0 ON STACK
037250 010146           MOV    R1,-(SP)      ;;PUSH R1 ON STACK
037252 010246           MOV    R2,-(SP)      ;;PUSH R2 ON STACK
037254 010346           MOV    R3,-(SP)      ;;PUSH R3 ON STACK
037256 010446           MOV    R4,-(SP)      ;;PUSH R4 ON STACK
037260 010546           MOV    R5,-(SP)      ;;PUSH R5 ON STACK
037262 017746 141652     MOV    @SWR,-(SP)    ;;PUSH @SWR ON STACK
037266 010637 037414     MOV    SP,$SAVR6    ;;SAVE SP
037272 012737 037304 000024     MOV    #$PWRUP,@#PWRVEC ;;SET UP VECTOR
037300 000000           HALT
037302 000776           BR     -2           ;;HANG UP
    
```

:POWER UP ROUTINE

```

037304 012737 037410 000024 $PWRUP: MOV    #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
037312 013706 037414     MOV    $SAVR6,SP    ;;GET SP
037316 005037 037414     CLR    $SAVR6      ;;WAIT LOOP FOR THE TTY
037322 005237 037414 1$: INC    $SAVR6    ;;WAIT FOR THE INC
037326 001375           BNE    1$         ;;OF WORD
037330 012677 141604     MOV    (SP)+,@SWR   ;;POP STACK INTO @SWR
037334 012605           MOV    (SP)+,R5    ;;POP STACK INTO R5
037336 012604           MOV    (SP)+,R4    ;;POP STACK INTO R4
037340 012603           MOV    (SP)+,R3    ;;POP STACK INTO R3
037342 012602           MOV    (SP)+,R2    ;;POP STACK INTO R2
037344 012601           MOV    (SP)+,R1    ;;POP STACK INTO R1
037346 012600           MOV    (SP)+,R0    ;;POP STACK INTO R0
037350 012737 037232 000024     MOV    #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
037356 012737 000340 000026     MOV    #340,@#PWRVEC+2 ;;PRIO:7
037364 104401           TYPE
037366 037416 $PWRMG: .WORD  PWRMSG ;;REPORT THE POWER FAILURE
037370 012716           MOV    (PC)+,(SP)  ;;POWER FAIL MESSAGE POINTER
037372 020000 $PWRAD: .WORD  START  ;;RESTART AT START
037374 042766 000020 000002     BIC    #20,2(SP)   ;;RESTART ADDRESS
037402 005037 001346     CLR    $TBIT      ;;CLEAR 'T' BIT
037406 000002           RTI
037410 000000 $ILLUP: HALT
037412 000776           BR     -2           ;;THE POWER UP SEQUENCE WAS STARTED
037414 000000 $SAVR6: 0           ;;BEFORE THE POWER DOWN WAS COMPLETE
3350 037416 012 015 040 PWRMSG: .ASCIZ <12><15>? POWER FAILURE - RESTARTING ?<12><15>
3351           .EVEN
    
```

				.SBTTL ERROR MESSAGES	
3352					
3353	037460	125	116	105	EM1: .ASCIZ /UNEXPECTED CPU TRAP TO LOC. 004/
3354	037520	125	116	105	EM2: .ASCIZ /UNEXPECTED MEM. MGMT. TRAP TO LOC. 250/
3355	037567	120	122	111	EM3: .ASCIZ /PRIORITY BITS SET WRONG IN PSW/
3356	037626	115	117	104	EM4: .ASCIZ /MODE BITS SET WRONG IN PSW/
3357	037661	104	125	101	EM5: .ASCIZ /DUAL ADDRESSING BETWEEN HI&LO BYTES OF PSW/
3358	037734	113	105	122	EM6: .ASCIZ /KERNEL R6 CHANGED BY WRITING SUPERVISOR OR USER R6/
3359	040017	101	040	115	EM7: .ASCIZ /A MEMORY MGMT. REG. TIMED OUT/
3360	040055	123	125	115	EM10: .ASCIZ /SUMMARY OF MEM. MGMT. REG. TIMEOUTS/
3361	040121	115	105	115	EM11: .ASCIZ /MEM. MGMT. REG. WOULD NOT CLEAR/
3362	040161	115	105	115	EM12: .ASCIZ /MEM. MGMT. REG. BITS NOT SET CORRECTLY/
3363	040230	123	122	060	EM13: .ASCIZ /SRO EFFECTED BY WRITE TO PSW/
3364	040265	115	115	122	EM14: .ASCIZ /MMR1 DID NOT TRACK PROPERLY/
3365	040321	115	115	122	EM15: .ASCIZ /MMR3 IS HOLDING THE WRONG DATA/
3366	040360	104	125	101	EM16: .ASCIZ /DUAL ADDRESSING BETWEEN PAR-PDR GROUPS/
3367	040427	102	101	104	EM17: .ASCIZ /BAD RELOCATION, OR STORING DATA 18-BIT MAPPING/
3368	040505	120	110	131	EM20: .ASCIZ /PHYS. ADDR. FORMED WRONG IN RELOCATE MODE/
3369	040557	123	122	062	EM21: .ASCIZ /SR2 NOT TRACKING CORRECTLY/
3370	040612	127	122	117	EM22: .ASCIZ /WRONG PDR'S REFERENCED WHILE IN RELOCATE MODE/
3371	040670	120	101	122	EM23: .ASCIZ /PAR OR PDR CHANGED BY A RESET/
3372	040726	115	101	111	EM24: .ASCIZ /MAINT. MODE (SRO <8>) NOT DISABLED BY A RESET/
3373	041004	104	101	124	EM25: .ASCIZ /DATA INCORRECT AFTER A MAINT. MODE WRITE/
3374	041055	123	117	125	EM26: .ASCIZ /SOURCE RELOCATED IN MAINT. MODE/
3375	041115	123	122	062	EM27: .ASCIZ /SR2 DID NOT LOCK UP CORRECT VIRTUAL ADDRESS/
3376	041171	106	117	114	EM30: .ASCIZ /FOLLOWING PAR-PDR WILL NOT ZERO/
3377	041231	123	125	115	EM31: .ASCIZ /SUMMARY OF DUAL ADDRESSING ERRORS/
3378	041273	123	125	115	EM32: .ASCIZ /SUMMARY OF COUNT PATTERN FAILURES/
3379	041335	105	122	122	EM33: .ASCIZ /ERROR IN BYTE ADDRESSING OF PAR-PDR/
3380	041401	124	110	105	EM34: .ASCIZ /THE FOLLOWING ARE DUAL ADDRESSING ERRORS FOR PAR'S-PDR'S/
3381	041472	124	110	105	EM36: .ASCIZ /THE FOLLOWING ARE COUNT PATTERN ERRORS FOR PAR'S-PDR'S/
3382	041561	111	114	114	EM40: .ASCIZ /ILLEGAL (MODE 10) STACK POINTER NOT MAPPED TO USER/
3383	041644	061	070	055	EM41: .ASCIZ /18-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM/
3384	041725	106	101	125	EM42: .ASCIZ /FAULTY CARRY PROPAGATION 18-BIT MAPPING/
3385	041775	061	070	055	EM43: .ASCIZ /18-BIT MAPPING POSSIBLE HOLE AT TOP OF MEMORY/
3386	042053	116	117	040	EM44: .ASCIZ /NO TRAP THRU ERRVEC, AT 18-BIT ADDR. 760000/
3387	042126	104	111	104	EM45: .ASCIZ /DIDN'T GET WRAP AROUND TO ADDRESS ZERO/
3388	042175	116	117	040	EM46: .ASCIZ /NO TRAP THRU ERRVEC, ON NON-EXISTANT ADDR./
3389	042250	120	122	105	EM47: .ASCIZ /PREMATURE END OF MEMORY FOUND/
3390	042306	062	062	055	EM50: .ASCIZ /22-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM/
3391	042367	102	101	104	EM51: .ASCIZ /BAD RELOCATION, CARRY PROPAGATION 22-BIT MAPPING/
3392	042447	062	062	055	EM52: .ASCIZ /22-BIT MAPPING POSSIBLE HOLE AT TOP OF MEMORY/
3393	042525	104	111	104	EM53: .ASCIZ /DID NOT GET UNIBUS ADDRESS/
3394	042560	127	055	102	EM54: .ASCIZ /W-BIT DID NOT GET SET IN PDR/
3395	042615	127	055	102	EM55: .ASCIZ /W-BIT SET IN MORE THAN ONE PDR/
3396	042654	127	055	102	EM56: .ASCIZ /W-BIT NOT CLEARED BY WRITING TO PDR/
3397	042720	127	055	102	EM57: .ASCIZ /W-BIT GOT SET DURING ODD ADDR. ABORT/
3398	042765	116	117	040	EM60: .ASCIZ /NO APT SPECIAL HARDWARE FOUND/

DATA HEADERS

3399						.SBTTL	DATA HEADERS
3400	043023	117	114	104	DH1:	.ASCIZ	/OLD PC OLD PSW R6 WAS CPUERR TESTNO ERRORPC/
3401	043103	117	114	104	DH2:	.ASCIZ	/OLD PC OLD PSW R6 WAS SRO SR1 SR2 TESTNO ERRORPC/
3402	043202	127	122	117	DH3:	.ASCIZ	/WROTE READ TESTNO ERRORPC/
3403	043242	101	104	104	DH7:	.ASCIZ	/ADDRESS TESTNO ERRORPC/
3404	043272	122	105	107	DH10:	.ASCII	/REGISTER-ADDRS NUM OF/<CRLF>
3405	043322	101	116	104		.ASCIZ	/AND-ED OR-ED TIMOUTS TESTNO ERRORPC/
3406	043372	122	105	107	DH11:	.ASCII	/REGISTR READ READ-(BINARY)/<CRLF>
3407	043430	101	104	104		.ASCIZ	/ADDRESS (OCTAL) 5432109876543210 TESTNO ERRORPC/
3408	043512	122	105	107	DH12:	.ASCII	/REGISTR WROTE READ READ-(BINARY)/<CRLF>
3409	043560	101	104	104		.ASCIZ	/ADDRESS (OCTAL) (OCTAL) 5432109876543210 TESTNO ERRORPC/
3410	043652	122	105	101	DH13:	.ASCIZ	/READ TESTNO ERRORPC/
3411	043702	105	130	120	DH14:	.ASCIZ	/EXPECTD (MMR1) TESTNO ERRORPC/
3412	043742	114	117	101	DH15:	.ASCIZ	/LOADED (MMR3) TESTNO ERRORPC/
3413	044002	111	116	104	DH16:	.ASCII	/INDEX INDEX PAR-PDR/<CRLF>
3414	044032	105	130	120		.ASCIZ	/EXPECTD RECEIVD ADDRREAD TESTNO ERRORPC/
3415	044103	126	111	122	DH17:	.ASCIZ	/VIR ADDR KIPAR4 GDDATA BADDATA TESTNO ERRORPC/
3416	044164	120	110	131	DH20:	.ASCII	/PHYSICL PAR 4 PAR 5/<CRLF>
3417	044212	101	104	104		.ASCIZ	/ADDRESS VBA VBA PAR 4 PAR 5 PSW TESTNO ERRORPC/
3418	044312	123	122	062	DH21:	.ASCIZ	/SR2 WAS EXPECTD TESTNO ERRORPC/
3419	044352	120	110	131	DH22:	.ASCII	/PHYSICL PAR 4/<CRLF>
3420	044370	101	104	104		.ASCIZ	/ADDRESS V.B.A. PAR 4 SRO WAS SR2 WAS PSW TESTNO ERRORPC/
3421	044470	124	105	123	DH24:	.ASCIZ	/TESTNO ERRORPC/
3422	044510	101	104	104	DH30:	.ASCIZ	/ADDRESS DATA TESTNO ERRORPC/
3423	044550	101	104	104	DH31:	.ASCII	/ADDROR ADDRAND ADDROR ADDRAND/<CRLF>
3424	044610	114	117	101		.ASCIZ	/LOADED LOADED ENABLED ENABLED TESTNO #ERRORS/
3425	044667	101	104	104	DH32:	.ASCIZ	/ADDROR ADDRAND PATRNOR PATRNAD DATAOR DATAAND TESTNO #ERRORS/
3426	044767	101	104	104	DH33:	.ASCIZ	/ADDRESS EXPECTD RECEIVD TESTNO ERRORPC/
3427	045037	101	104	104	DH34:	.ASCIZ	/ADDRESS ADDRESS/
3428	045057	101	104	104	DH36:	.ASCIZ	/ADDRESS DATARED PATTERN COUNT TESTNO/
3429	045126	122	105	101	DH40:	.ASCII	/READOFF TESTNO ERRORPC/<CRLF>
3430	045156	123	124	101		.ASCIZ	/STACK/
3431	045164	123	124	101	DH41:	.ASCIZ	/STARTBK FINISHBK TESTNO ERRORPC/
3432	045225	120	101	124	DH42:	.ASCII	/PATTERN DATA ADDRESS/<CRLF>
3433	045255	114	117	101		.ASCIZ	/LOADED FETCHED INTENDED TESTNO ERRORPC/
3434	045327	123	124	101	DH43:	.ASCIZ	/STARTBK TESTNO ERRORPC/
3435	045357	104	101	124	DH45:	.ASCIZ	/DATA TESTNO ERRORPC/
3436	045407	116	105	101	DH46:	.ASCIZ	/NEADDR TESTNO ERRORPC/
3437	045436	113	111	120	DH47:	.ASCIZ	/KIPAR4 LASTBLOCK TESTNO ERRORPC/
3438	045502	120	104	122	DH54:	.ASCII	/PDR VIRTUAL/<CRLF>
3439	045522	124	105	123		.ASCIZ	/TESTED ADDRESS TESTNO ERRORPC/
3440	045562	120	104	122	DH55:	.ASCII	/PDR IN PDR VIRTUAL/<CRLF>
3441	045612	105	122	122		.ASCIZ	/ERROR TESTED ADDRESS TESTNO ERRORPC/
3442	045662	120	104	122	DH56:	.ASCIZ	/PDR TESTNO ERRORPC/
3443	045712	120	104	122	DH57:	.ASCIZ	/PDR WAS EXPECTD TESTNO ERRORPC/
3444	045752	105	122	122	DH60:	.ASCIZ	/ERRORPC/
3445						.EVEN	

Address	Hex	Hex	Hex	Hex	Label	Text
3446					.SETTL	DATA TABLES
3447	045762	001260	001262	001256	DT1:	.WORD TRAPPC,TRAPPS,WASR6,CPUERR,TESTNO,\$ERRPC,0
3448	046000	001260	001262	001256	DT2:	.WORD TRAPPC,TRAPPS,WASR6,WASSR0,WASSR1,WASSR2,TESTNO,\$ERRPC,0
3449	046022	001162	001164	001254	DT3:	.WORD \$REG0,\$REG1,TESTNO,\$ERRPC,0
3450	046034	001162	001254	001116	DT7:	.WORD \$REG0,TESTNO,\$ERRPC,0
3451	046044	001320	001316	001274	DT10:	.WORD ADRAND,ADDROR,TONUM,TESTNO,\$ERRPC,0
3452	046060	001162	001164	001164	DT11:	.WORD \$REG0,\$REG1,\$REG1,TESTNO,\$ERRPC,0
3453	046074	001162	001164	001166	DT12:	.WORD \$REG0,\$REG1,\$REG2,\$REG2,TESTNO,\$ERRPC,0
3454	046112	001170	001164	001254	DT14:	.WORD \$REG3,\$REG1,TESTNO,\$ERRPC,0
3455	046124	001166	001164	001254	DT15:	.WORD \$REG2,\$REG1,TESTNO,\$ERRPC,0
3456	046136	001162	001164	001166	DT16:	.WORD \$REG0,\$REG1,\$REG2,TESTNO,\$ERRPC,0
3457	046152	001162	172350	001164	DT17:	.WORD \$REG0,KIPAR4,\$REG1,\$REG2,TESTNO,\$ERRPC,0
3458	046170	001302	001276	001300	DT20:	.WORD PBALO,VIRT1,VIRT2,\$REG4,\$REG5,\$TMP0,TESTNO,\$ERRPC,0
3459	046212	001270	001164	001254	DT21:	.WORD WASSR2,\$REG1,TESTNO,\$ERRPC,0
3460	046224	001302	001276	001172	DT22:	.WORD PBALO,VIRT1,\$REG4,WASSR0,WASSR2,\$TMP0,TESTNO,\$ERRPC,0
3461	046246	001254	001116	000000	DT24:	.WORD TESTNO,\$ERRPC,0
3462	046254	001164	001166	001254	DT25:	.WORD \$REG1,\$REG2,TESTNO,\$ERRPC,0
3463	046266	001270	001172	001254	DT27:	.WORD WASSR2,\$REG4,TESTNO,\$ERRPC,0
3464	046300	001162	001166	001254	DT30:	.WORD \$REG0,\$REG2,TESTNO,ERRCNT,0
3465	046312	001316	001320	001306	DT31:	.WORD ADDROR,ADRAND,DATAOR,DATAND,TESTNO,ERRCNT,0
3466	046330	001316	001320	001312	DT32:	.WORD ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,TESTNO,ERRCNT,0
3467	046352	001162	001166	001164	DT33:	.WORD \$REG0,\$REG2,\$REG1,TESTNO,\$ERRPC,0
3468	046366	001162	001164	001254	DT35:	.WORD \$REG0,\$REG1,TESTNO,0
3469	046376	001162	001166	001172	DT37:	.WORD \$REG0,\$REG2,\$REG4,\$REG1,TESTNO,0
3470	046412	001200	001202	001254	DT41:	.WORD \$TMP1,\$TMP2,TESTNO,\$ERRPC,0
3471	046424	001166	001170	001162	DT42:	.WORD \$REG2,\$REG3,\$REG0,TESTNO,\$ERRPC,0
3472	046440	001200	001254	001116	DT43:	.WORD \$TMP1,TESTNO,\$ERRPC,0
3473	046450	001164	001254	001116	DT45:	.WORD \$REG1,TESTNO,\$ERRPC,0
3474	046460	172350	003022	001254	DT47:	.WORD KIPAR4,\$LSTBK,TESTNO,\$ERRPC,0
3475	046472	001174	001170	001254	DT54:	.WORD \$REG5,\$REG3,TESTNO,\$ERRPC,0
3476	046504	001162	001174	001170	DT55:	.WORD \$REG0,\$REG5,\$REG3,TESTNO,\$ERRPC,0
3477	046520	001174	001254	001116	DT56:	.WORD \$REG5,TESTNO,\$ERRPC,0
3478	046530	001116	000000		DT60:	.WORD \$ERRPC,0

3479						.SBTTL	DATA FORMATS
3480	046534	000	000	000	DF1:	.BYTE	0,0,0,0,0
3481	046541	000	000	000	DF2:	.BYTE	0,0,0,0,0,0,0,0
3482	046551	000	000	000	DF3:	.BYTE	0,0,0,0
3483	046555	000	000	000	DF7:	.BYTE	0,0,0
3484	046560	000	000	001	DF10:	.BYTE	0,0,1,0,0
3485	046565	000	000	002	DF11:	.BYTE	0,0,2,0,0
3486	046572	000	000	000	DF12:	.BYTE	0,0,0,2,0,0
3487	046600	000	000	000	DF16:	.BYTE	0,0,0,0,0,0
3488	046606	003	000	000	DF20:	.BYTE	3,0,0,0,0,0,0,0
3489	046616	000	000		DF24:	.BYTE	0,0
3490	046620	000	000	000	DF32:	.BYTE	0,0,0,0,0,1
3491	046626	000	000	000	DF33:	.BYTE	0,0,0,0,0,0,0,1
3492	046636	004	000	000	DF43:	.BYTE	4,0,0
3493		000001				.END	

SYMBOL TABLE

ABASE = 000000	BIT00 = 000001	DH14 043702	DT45 046450	ERRTYP 033626
ACDW1 = 000000	BIT01 = 000002	DH15 043742	DT47 046460	ERRVEC= 000004
ACDW2 = 000000	BIT02 = 000004	DH16 044002	DT54 046472	FORMPA 003426
ACPUOP= 000000	BIT03 = 000010	DH17 044103	DT55 046504	GTSWR = 104407
ADDROR 001316	BIT04 = 000020	DH2 043103	DT56 046520	HDWFLA 001340
ADDW0 = 000000	BIT05 = 000040	DH20 044164	DT60 046530	HOLFLG 001336
ADDW1 = 000000	BIT06 = 000100	DH21 044312	DT7 046034	HT = 000011
ADDW10= 000000	BIT07 = 000200	DH22 044352	DUALAD 002334	IBSAVE 033624
ADDW11= 000000	BIT08 = 000400	DH24 044470	EMTVEC= 000030	IOTVEC= 000020
ADDW12= 000000	BIT09 = 001000	DH3 043202	EM1 037460	KDPAR0= 172360
ADDW13= 000000	BIT1 = 000002	DH30 044510	EM10 040055	KDPAR1= 172362
ADDW14= 000000	BIT10 = 002000	DH31 044550	EM11 040121	KDPAR2= 172364
ADDW15= 000000	BIT11 = 004000	DH32 044667	EM12 040161	KDPAR3= 172366
ADDW2 = 000000	BIT12 = 010000	DH33 044767	EM13 040230	KDPAR4= 172370
ADDW3 = 000000	BIT13 = 020000	DH34 045037	EM14 040265	KDPAR5= 172372
ADDW4 = 000000	BIT14 = 040000	DH36 045057	EM15 040321	KDPAR6= 172374
ADDW5 = 000000	BIT15 = 100000	DH40 045126	EM16 040360	KDPAR7= 172376
ADDW6 = 000000	BIT2 = 000004	DH41 045164	EM17 040427	KDPDR0= 172320
ADDW7 = 000000	BIT3 = 000010	DH42 045225	EM2 037520	KDPDR1= 172322
ADDW8 = 000000	BIT4 = 000020	DH43 045327	EM20 040505	KDPDR2= 172324
ADDW9 = 000000	BIT5 = 000040	DH45 045357	EM21 040557	KDPDR3= 172326
ADEVC1 = 000000	BIT6 = 000100	DH46 045407	EM22 040612	KDPDR4= 172330
ADEVN = 000000	BIT7 = 000200	DH47 045436	EM23 040670	KDPDR5= 172332
ADRAND 001320	BIT8 = 000400	DH54 045502	EM24 040726	KDPDR6= 172334
AENV = 000000	BIT9 = 001000	DH55 045562	EM25 041004	KDPDR7= 172336
AENVN = 000000	BPTVEC= 000014	DH56 045662	EM26 041055	KERSTK= 001100
AFATAL = 000000	CKSWR = 104410	DH57 045712	EM27 041115	KIPAR0= 172340
AMADR1 = 000000	CLEANU 002276	DH60 045752	EM3 037567	KIPAR1= 172342
AMADR2 = 000000	CLRREG 002260	DH7 043242	EM30 041171	KIPAR2= 172344
AMADR3 = 000000	CMMSG 035337	DISPLA 001142	EM31 041231	KIPAR3= 172346
AMADR4 = 000000	CNTRLC 035260	DISPRE 000174	EM32 041273	KIPAR4= 172350
AMAMS1 = 000000	CPSAVE 033244	DSWR = 177570	EM33 041335	KIPAR5= 172352
AMAMS2 = 000000	CPUERR= 177766	DT1 045762	EM34 041401	KIPAR6= 172354
AMAMS3 = 000000	CPUEXP 001334	DT10 046044	EM36 041472	KIPAR7= 172356
AMAMS4 = 000000	CR = 000015	DT11 046060	EM4 037626	KIPDR0= 172300
AMSGAD = 000000	CRLF = 000200	DT12 046074	EM40 041561	KIPDR1= 172302
AMSGLG = 000000	DATAND 001310	DT14 046112	EM41 041644	KIPDR2= 172304
AMSGTY = 000000	DATAOR 001306	DT15 046124	EM42 041725	KIPDR3= 172306
AMTYP1 = 000000	DDISP = 177570	DT16 046136	EM43 041775	KIPDR4= 172310
AMTYP2 = 000000	DF1 046534	DT17 046152	EM44 042053	KIPDR5= 172312
AMTYP3 = 000000	DF10 046560	DT2 046000	EM45 042126	KIPDR6= 172314
AMTYP4 = 000000	DF11 046565	DT20 046170	EM46 042175	KIPDR7= 172316
APASS = 000000	DF12 046572	DT21 046212	EM47 042250	KSP = 2000006
APRIOR = 000000	DF16 046600	DT22 046224	EM5 037661	LF = 000012
APTCU = 000040	DF2 046541	DT24 046246	EM50 042306	LOOP 020454
APTENV = 000001	DF20 046606	DT25 046254	EM51 042367	MGMERR 003330
APTSIZ = 000200	DF24 046616	DT27 046266	EM52 042447	MGMFLG 003332
APTSPO = 000100	DF3 046551	DT3 046022	EM53 042525	MMR0 = 177572
APTTST 027224	DF32 046620	DT30 046300	EM54 042560	MMR1 = 177574
ASWREG = 000000	DF33 046626	DT31 046312	EM55 042615	MMR2 = 177576
ATESTN = 000000	DF43 046636	DT32 046330	EM56 042654	MMR3 = 172516
AUNIT = 000000	DF7 046555	DT33 046352	EM57 042720	MMVEC = 000250
AUSWR = 000000	DH1 043023	DT35 046366	EM6 037734	OLDPC 001326
AVECT1 = 000000	DH10 043272	DT37 046376	EM60 042765	OLDPS 001330
AVECT2 = 000000	DH11 043372	DT41 046412	EM7 040017	PARCOU 002400
BADPC 001324	DH12 043512	DT42 046424	ERRCNT 001322	PARTAB 001356
BIT0 = 000001	DH13 043652	DT43 046440	ERROR = 104000	PATAND 001314

SYMBOL TABLE

PATRNS	027024	SIPAR7=	172256	TPVEC =	000064	UDPAR1=	177662	SCM4 =	000006
PATTOR	001312	SIPDR0=	172200	TRAPPC	001260	UDPAR2=	177664	SCNTLC	001350
PBAHI	001304	SIPDR1=	172202	TRAPPS	001262	UDPAR3=	177666	SCNTLG	035231
PBALO	001302	SIPDR2=	172204	TRAPVE=	000034	UDPAR4=	177670	SCNTLU	035224
PCPUER	001332	SIPDR3=	172206	TRIVEC=	000014	UDPAR5=	177672	SCORE	002676
PFECDF	034260	SIPDR4=	172210	TST1	020564	UDPAR6=	177674	SCPUOP	001252
PFECDH	034220	SIPDR5=	172212	TST10	021422	UDPAR7=	177676	SCRLF	001221
PFECDT	034250	SIPDR6=	172214	TST11	021440	UDPDR0=	177620	SCROUT	002726
PFECFM	034160	SIPDR7=	172216	TST12	021456	UDPDR1=	177622	SDBLK	036720
PFECWS	034150	SR0 =	177572	TST13	021474	UDPDR2=	177624	SDB20	037024
PIRQ =	177772	SR1 =	177574	TST14	021512	UDPDR3=	177626	SDEVCT	001234
PIRQVE=	000240	SR2 =	177576	TST15	021654	UDPDR4=	177630	SDOAGN	032752
PRO =	000000	SR3 =	172516	TST16	021714	UDPDR5=	177632	SDTBL	036710
PR1 =	000040	SSP =	X000006	TST17	022320	UDPDR6=	177634	SENDAD	032742
PR2 =	000100	STACK =	001100	TST2	020642	UDPDR7=	177636	SENDCT	032570
PR3 =	000140	START	020000	TST20	022442	UIPAR0=	177640	SENULL	033014
PR4 =	000200	STKLMT=	177774	TST21	022640	UIPAR1=	177642	SENV	001244
PR5 =	000240	SUPSTK=	000700	TST22	023036	UIPAR2=	177644	SENVN	001245
PR6 =	000300	SWR	001140	TST23	023234	UIPAR3=	177646	SEOP	032540
PR7 =	000340	SWREG	000176	TST24	023432	UIPAR4=	177650	SEOPCT	032562
PS =	177776	SW0 =	000001	TST25	023630	UIPAR5=	177652	SERFLG	001103
PSW =	177776	SW00 =	000001	TST26	024026	UIPAR6=	177654	SERMAX	001115
PWRMSG	037416	SW01 =	000002	TST27	024140	UIPAR7=	177656	SERROR	033246
PWRVEC=	000024	SW02 =	000004	TST3	020726	UIPDR0=	177600	SERRPC	001116
RDCHR =	104411	SW03 =	000010	TST30	024252	UIPDR1=	177602	SERRTB	001372
RDLIN =	104412	SW04 =	000020	TST31	024364	UIPDR2=	177604	SERTTL	001112
READON	001342	SW05 =	000040	TST32	024502	UIPDR3=	177606	SESCAP	001212
RESREG=	104414	SW06 =	000100	TST33	024620	UIPDR4=	177610	SETABL	001244
RESVEC=	000010	SW07 =	000200	TST34	024736	UIPDR5=	177612	SETEND	001254
RMIREG=	177770	SW08 =	000400	TST35	025042	UIPDR6=	177614	SFATAL	001226
R6 =	X000006	SW09 =	001000	TST36	025146	UIPDR7=	177616	SFFLG	036200
R7 =	X000007	SW1 =	000002	TST37	025252	USESTK=	000600	SFILLC	001156
SAVREG=	104413	SW10 =	002000	TST4	021050	USP =	X000006	SFILLS	001155
SCOPE =	000004	SW11 =	004000	TST40	025356	VIRT1	001276	SGADR	001120
SDPAR0=	172260	SW12 =	010000	TST41	025462	VIRT2	001300	SGDAT	001124
SDPAR1=	172262	SW13 =	020000	TST42	025566	WASSR6	001256	SGET42	032714
SDPAR2=	172264	SW14 =	040000	TST43	025662	WASSR0	001264	SGTSWR	034334
SDPAR3=	172266	SW15 =	100000	TST44	026210	WASSR1	001266	SHD =	000000
SDPAR4=	172270	SW2 =	000004	TST45	026344	WASSR2	001270	SHIBTS	000204
SDPAR5=	172272	SW3 =	000010	TST46	026602	WBIT =	000100	SICNT	001104
SDPAR6=	172274	SW4 =	000020	TST47	027276	SAPTHD	000204	SILLUP	037410
SDPAR7=	172276	SW5 =	000040	TST5	021214	SATYC	035760	SINTAG	001135
SDPDR0=	172220	SW6 =	000100	TST50	030136	SATY1	035734	SITEMB	001114
SDPDR1=	172222	SW7 =	000200	TST51	031020	SATY3	035742	SKTNEX	002670
SDPDR2=	172224	SW8 =	000400	TST52	031634	SATY4	035752	SKTOUT	002660
SDPDR3=	172226	SW9 =	001000	TST53	032030	SAUTOB	001134	SKT11	002512
SDPDR4=	172230	TBIT =	000020	TST54	032236	SBADR	001122	SLF	001222
SDPDR5=	172232	TBITPS	001272	TST55	032444	SBDDAT	001126	SLFLG	036177
SDPDR6=	172234	TBITVE=	000014	TST6	021366	SBELL	001214	SLOOP	033010
SDPDR7=	172236	TESTNO	001254	TST7	021404	SBIN	036254	SLPADR	001106
SIPAR0=	172240	TIMERR	003232	TYPBN =	104406	SCHARC	035730	SLPERR	001110
SIPAR1=	172242	TIMFLG	003234	TYPDS =	104405	SCKSWR	034264	SLSTAD	003020
SIPAR2=	172244	TIMTST	003024	TYPE =	104401	SCLR.T	032732	SLSTBK	003022
SIPAR3=	172246	TKVEC =	000060	TYPOC =	104402	SCMTAG	001100	SMAIL	001224
SIPAR4=	172250	TOFF	002172	TYPON =	104404	SCM1 =	000006	SMBADR	000206
SIPAR5=	172252	TON	002226	TYPOS =	104403	SCM2 =	000014	SMFLG	036176
SIPAR6=	172254	TONUM	001274	UDPAR0=	177660	SCM3 =	000006	SMNEW	035247

\$MSGAD 001240	\$QUES 001220	\$SCOPE 033020	\$TMP2 001202	\$TYPDS 036504
\$MSGLG 001242	\$RDCHR 034606	\$SETUP= 000137	\$TMP3 001204	\$TYPE 035370
\$MSGTY 001224	\$RDLIN 034736	\$SIZE 002454	\$TMP4 001206	\$TYPEC 035602
\$MSWR 035236	\$RDSZ = 000010	\$SIZEX 002732	\$TMP5 001210	\$TYPEX 035732
\$MXCNT 001344	\$REGAD 001160	\$STUP = 177777	\$TN = 000056	\$TYPOC 036302
\$NULL 001154	\$REGO 001162	\$SVLAD 033174	\$TPB 001152	\$TYPON 036316
\$NWTST= 000001	\$REG1 001164	\$SVPC = 000204	\$TPFLG 001157	\$TYPOS 036256
\$OCNT 036500	\$REG2 001166	\$SWR = 173400	\$TPS 001150	\$UNIT 001236
\$OCTVL 037126	\$REG3 001170	\$SWREG 001246	\$TRAP 037144	\$UNITM 000214
\$OMODE 036502	\$REG4 001172	\$SWRMK= 000000	\$TRAP2 037166	\$USWR 001250
\$OVER 033230	\$REG5 001174	\$TBIT 001346	\$TRP = 000015	\$XOFF = 000023
\$PASS 001232	\$RESRE 036766	\$TESTN 001230	\$TRPAD 037200	\$XON = 000021
\$PASTM 000212	\$RTNAD 033012	\$TKB 001146	\$TSTM 000210	\$XTSTR 033032
\$PWAD 037372	\$RTRN 033006	\$TKS 001144	\$TSTNM 001102	\$\$GET4= 000001
\$PWADN 037232	\$\$AVRE 036730	\$TMP0 001176	\$TTYIN 035214	\$OFILL 036501
\$PWIMG 037366	\$\$AVR6 037414	\$TMP1 001200	\$TYPBN 036202	.\$X = 000204
\$PWUP 037304				

. ABS. 046641 000
000000 001

ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 59464 WORDS (233 PAGES)

DYNAMIC MEMORY: 20034 WORDS (77 PAGES)

ELAPSED TIME: 00:06:06

CKKTAB.BIN,CKKTAB/CR/-SP/NL:TOC-CKKTAB.MLB/ML,CKKTAB.P11

SYMBOL CROSS REFERENCE		REFERENCES								
SYMBOL	VALUE									
ABASE	= 000000	16-1276								
ACDW1	= 000000	16-1276								
ACDW2	= 000000	16-1276								
ACPUOP	= 000000	16-1276	16-1276							
ADDROR	001316	#16-1276	*26-1637	*27-1655	*28-1677	*29-1701	*29-1720	*39-2009	*39-2029	102-3451
		102-3465	102-3466							
ADDW0	= 000000	16-1276								
ADDW1	= 000000	16-1276								
ADDW10	= 000000	16-1276								
ADDW11	= 000000	16-1276								
ADDW12	= 000000	16-1276								
ADDW13	= 000000	16-1276								
ADDW14	= 000000	16-1276								
ADDW15	= 000000	16-1276								
ADDW2	= 000000	16-1276								
ADDW3	= 000000	16-1276								
ADDW4	= 000000	16-1276								
ADDW5	= 000000	16-1276								
ADDW6	= 000000	16-1276								
ADDW7	= 000000	16-1276								
ADDW8	= 000000	16-1276								
ADDW9	= 000000	16-1276								
ADEVCT	= 000000	16-1276	16-1276							
ADEVPM	= 000000	16-1276								
ADRAND	001320	#16-1276	*26-1641	27-1656	28-1678	*29-1700	*29-1722	*39-2008	*39-2031	102-3451
		102-3465	102-3466							
AENV	= 000000	16-1276	16-1276							
AENVPM	= 000000	16-1276	16-1276							
AFATAL	= 000000	16-1276	16-1276							
AMADR1	= 000000	16-1276								
AMADR2	= 000000	16-1276								
AMADR3	= 000000	16-1276								
AMADR4	= 000000	16-1276								
AMAMS1	= 000000	16-1276								
AMAMS2	= 000000	16-1276								
AMAMS3	= 000000	16-1276								
AMAMS4	= 000000	16-1276								
AMSGAD	= 000000	16-1276	16-1276							
AMSGLG	= 000000	16-1276	16-1276							
AMSGTY	= 000000	16-1276	16-1276							
AMTYP1	= 000000	16-1276								
AMTYP2	= 000000	16-1276								
AMTYP3	= 000000	16-1276								
AMTYP4	= 000000	16-1276								
APASS	= 000000	16-1276	16-1276							
APRIOR	= 000000	16-1276								
APTC SU	= 000040	91-3333	#92-3335							
APTENV	= 000001	86-3219	91-3333	92-3335	#92-3335					
APTSIZ	= 000200	33-1838	#92-3335							
APTSPO	= 000100	91-3333	92-3335	#92-3335						
APTTST	027224	74-2714	74-2716	#75-2742						
ASWREG	= 000000	16-1276	16-1276							

SYMBOL CROSS REFERENCE		REFERENCES		REF	VO1					
SYMBOL	VALUE									
AATESTN	= 000000	16-1276	16-1276							
AUNIT	= 000000	16-1276	16-1276							
AUSWR	= 000000	16-1276	16-1276							
AVECT1	= 000000	16-1276								
AVECT2	= 000000	16-1276								
BADPC	= 001324	#16-1276	*76-2887	*77-3024						
BIT0	= 000001	#15-1199								
BIT00	= 000001	#15-1199	15-1199	85-3217	85-3217	86-3219	86-3219			
BIT01	= 000002	#15-1199	15-1199							
BIT02	= 000004	#15-1199	15-1199							
BIT03	= 000010	#15-1199	15-1199							
BIT04	= 000020	#15-1199	15-1199							
BIT05	= 000040	#15-1199	15-1199							
BIT06	= 000100	#15-1199	15-1199							
BIT07	= 000200	#15-1199	15-1199							
BIT08	= 000400	#15-1199	15-1199	85-3217						
BIT09	= 001000	#15-1199	15-1199	85-3217	86-3219					
BIT1	= 000002	#15-1199								
BIT10	= 002000	#15-1199	86-3219							
BIT11	= 004000	#15-1199								
BIT12	= 010000	#15-1199	84-3215							
BIT13	= 020000	#15-1199	48-2193	86-3219						
BIT14	= 040000	#15-1199	48-2209	85-3217						
BIT15	= 100000	#15-1199	46-2134	48-2195						
BIT2	= 000004	#15-1199								
BIT3	= 000010	#15-1199								
BIT4	= 000020	#15-1199	15-1210	77-2911	77-2961					
BIT5	= 000040	#15-1199								
BIT6	= 000100	#15-1199	15-1211							
BIT7	= 000200	#15-1199	33-1853	75-2745						
BIT8	= 000400	#15-1199	71-2588	71-2590	72-2628	72-2645	74-2703	76-2799	77-2939	
BIT9	= 001000	#15-1199								
BPTVEC	= 000014	#15-1199								
CKSWR	= 104410	85-3217	86-3219	86-3219	#98-3347					
CLEANU	002276	#26-1635	51-2333	52-2338	53-2343	54-2348	55-2353	56-2358	57-2375	58-2380
		59-2385	60-2390	61-2395	62-2400					
CLRREG	002260	#25-1622	51-2333	51-2333	52-2338	52-2338	53-2343	53-2343	54-2348	54-2348
		55-2353	55-2353	56-2358	56-2358					
CMSG	035337	90-3320	#90-3331							
CNTRLC	035260	89-3315	89-3315	#90-3318						
CPSAVE	033244	*85-3217	85-3217	#85-3217	*86-3219	86-3219	88-3313			
CPUEER	= 177766	*15-1215	*28-1689	30-1750	*30-1757	*33-1852	*33-1855	76-2871	*76-2891	77-3008
		*77-3028	102-3447							
CPUEXP	001334	#16-1276	30-1752	30-1754	*76-2831	*76-2846	*76-2856	*76-2865	*77-2967	*77-2982
		*77-2992	*77-3001							
CR	= 000015	#15-1199	91-3333	91-3333						
CRLF	= 000200	#15-1199	33-1839	91-3333	91-3333	91-3333	101-3404	101-3406	101-3408	101-3413
		101-3416	101-3419	101-3423	101-3429	101-3432	101-3438	101-3440		
DATAID	001310	#16-1276	*26-1640	27-1658	28-1682	102-3465	102-3466			
DATAOR	001306	#16-1276	*26-1636	*27-1657	*28-1681	102-3465	102-3466			
DDISP	- 177570	#15-1199	16-1276	33-1838						
DF1	046534	17-1282	18-1363	20-1456	22-1556	22-1574	#103-3480			

SYMBOL	CROSS REFERENCE VALUE	REFERENCES	SEQUENCE	CREF	V01
DT10	046044	18-1323 #102-3451			
DT11	046060	18-1330 19-1394 #102-3452			
DT12	046074	18-1337 #102-3453			
DT14	046112	18-1349 #102-3454			
DT15	046124	18-1355 22-1567 #102-3455			
DT16	046136	18-1362 #102-3456			
DT17	046152	18-1368 #102-3457			
DT2	046000	17-1287 19-1412 #102-3448			
DT20	046170	18-1375 #102-3458			
DT21	046212	19-1380 #102-3459			
DT22	046224	19-1387 #102-3460			
DT24	046246	19-1400 21-1498 #102-3461			
DT25	046254	19-1406 #102-3462			
DT27	046266	19-1418 #102-3463			
DT3	046022	17-1293 17-1299 17-1305 17-1311 #102-3449			
DT30	046300	19-1424 #102-3464			
DT31	046312	19-1431 #102-3465			
DT32	046330	20-1436 #102-3466			
DT33	046352	20-1442 #102-3467			
DT35	046366	20-1455 #102-3468			
DT37	046376	20-1467 #102-3469			
DT41	046412	20-1480 21-1522 #102-3470			
DT42	046424	20-1487 21-1529 21-1542 #102-3471			
DT43	046440	21-1492 21-1535 #102-3472			
DT45	046450	21-1504 #102-3473			
DT47	046460	21-1516 #102-3474			
DT54	046472	22-1548 #102-3475			
DT55	046504	22-1555 #102-3476			
DT56	046520	22-1561 #102-3477			
DT60	046530	22-1573 #102-3478			
DT7	046034	18-1316 18-1343 20-1474 21-1510 #102-3450			
DUALAD	002334	#27-1653 51-2333 52-2338 53-2343 54-2348 55-2353 56-2358			
EMTVEC	= 000030	#15-1199 *33-1838 *33-1838			
EM1	037460	17-1279 #100-3353			
EM10	040055	18-1320 #100-3360			
EM11	040121	18-1327 #100-3361			
EM12	040161	18-1334 #100-3362			
EM13	040230	18-1341 #100-3363			
EM14	040265	18-1347 #100-3364			
EM15	040321	18-1353 #100-3365			
EM16	040360	18-1359 #100-3366			
EM17	040427	18-1366 #100-3367			
EM2	037520	17-1285 #100-3354			
EM20	040505	18-1372 #100-3368			
EM21	040557	19-1378 #100-3369			
EM22	040612	19-1384 #100-3370			
EM23	040670	19-1391 #100-3371			
EM24	040726	19-1398 #100-3372			
EM25	041004	19-1404 #100-3373			
EM26	041055	19-1410 #100-3374			
EM27	041115	19-1416 #100-3375			
EM3	037567	17-1291 #100-3355			

SYMBOL	CROSS REFERENCE VALUE	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES
GTSWR	= 104407	98-3347	98-3347							
HDWFLA	001340	33-1839	90-3327	#98-3347						
HOLFLG	001336	#16-1276	*75-2742	*75-2753	76-2771	76-2783	76-2824	77-2912	77-2922	
		#16-1276	*76-2775	76-2790	*76-2795	76-2811	76-2882	*76-2885	*76-2890	*77-2917
		77-2930	*77-2935	77-2953	77-3019	*77-3022	*77-3027			
HT	= 000011	#15-1199	91-3333	91-3333						
IBSAVE	033624	*86-3219	86-3219	*86-3219	*86-3219	86-3219	86-3219	86-3219	86-3219	#86-3219
IOTVEC	= 000020	#15-1199	*33-1838	*33-1838						
KDPAR0	= 172360	#15-1200								
KDPAR1	= 172362	#15-1200								
KDPAR2	= 172364	#15-1200								
KDPAR3	= 172366	#15-1200								
KDPAR4	= 172370	#15-1200								
KDPAR5	= 172372	#15-1200								
KDPAR6	= 172374	#15-1200								
KDPAR7	= 172376	#15-1200	51-2333	51-2333	57-2375	59-2385				
KDPDR0	= 172320	#15-1200								
KDPDR1	= 172322	#15-1200								
KDPDR2	= 172324	#15-1200								
KDPDR3	= 172326	#15-1200								
KDPDR4	= 172330	#15-1200								
KDPDR5	= 172332	#15-1200								
KDPDR6	= 172334	#15-1200								
KDPDR7	= 172336	#15-1200	54-2348	54-2348	60-2390					
KERSTK	= 001100	#15-1212	38-1956	38-1962	38-1964	71-2597	83-3204			
KIPAR0	= 172340	#15-1200	28-1689	32-1808	40-2041	51-2333	51-2333	51-2333	51-2333	51-2333
		57-2375	59-2385	63-2412	*63-2412	63-2412	63-2412	63-2412	70-2481	70-2509
KIPAR1	= 172342	#15-1200	*74-2690							
KIPAR2	= 172344	#15-1200	*74-2691							
KIPAR3	= 172346	#15-1200	*72-2619	*74-2692	80-3172					
KIPAR4	= 172350	#15-1200	*72-2620	*74-2699	*76-2777	*76-2806	*76-2829	*76-2847	*76-2858	*76-2859
		76-2874	*77-2918	*77-2946	77-2949	*77-2963	*77-2983	*77-2994	*77-2995	77-3011
		*78-3056	*78-3115	*78-3136	102-3457	102-3474				
KIPAR5	= 172352	#15-1200	*76-2778	76-2785	76-2792	*76-2807	76-2809	76-2884	*77-2919	77-2924
		77-2932	*77-2947	77-2951	*77-2964	77-3021	*78-3057	*78-3116	*78-3137	
KIPAR6	= 172354	#15-1200								
KIPAR7	= 172356	#15-1200	28-1689	*72-2621	*74-2693					
KIPDR0	= 172300	#15-1200	41-2052	54-2348	54-2348	54-2348	54-2348	54-2348	60-2390	66-2427
		*66-2427	66-2427	66-2427	66-2427	70-2480	70-2498	71-2578	*71-2586	74-2687
		80-3172	80-3172	80-3172						
KIPDR1	= 172302	#15-1200								
KIPDR2	= 172304	#15-1200								
KIPDR3	= 172306	#15-1200	*72-2622	*72-2640	*83-3201	83-3205				
KIPDR4	= 172310	#15-1200	*72-2623	*78-3113	*78-3132					
KIPDR5	= 172312	#15-1200	*78-3114	*78-3133						
KIPDR6	= 172314	#15-1200								
KIPDR7	= 172316	#15-1200	*72-2624							
KSP	=X000006	#15-1207	*25-1622	*25-1626	*27-1654	*28-1676	*29-1713	*30-1748	*30-1749	30-1751
		*30-1759	*30-1760	*31-1779	*31-1780	31-1781	*31-1788	*31-1789	*33-1841	*38-1956
		38-1962	38-1965	*38-1972	*39-2023	*71-2597	*76-2869	*76-2870	*76-2892	*76-2893
		*77-3006	*77-3007	*77-3029	*77-3030	*78-3146	*78-3147	*78-3161	*78-3162	*83-3204
		*88-3222	*88-3251	*88-3276	88-3278	*88-3279	*88-3283	*88-3284	*88-3290	88-3292

SYMBOL	CROSS REFERENCE VALUE	REFERENCES
LF	= 000012	*88-3293 *88-3296 *88-3297 *88-3303 *88-3304
LOOP	020454	#15-1199 91-3333 91-3333
MGMERR	003330	#33-1840 84-3215
MGMFLG	003332	28-1689 #31-1771 33-1844 71-2599 72-2639 78-3142
MMRO	= 177572	#31-1772 *31-1787 *33-1848
MMR1	= 177574	#15-1203 *33-1850 *38-1954 *48-2195 48-2208 *48-2216 48-2218 48-2228 48-2239
MMR2	= 177576	48-2254 48-2267 *48-2277 *74-2703 *74-2706 *76-2770
MMR3	= 172516	#15-1204 48-2192
MMVEC	= 000250	#15-1205
OLDPC	001326	#15-1206 *33-1851 49-2287 *77-2911 *77-2961 *77-3003
OLDPS	001330	#15-1200 *28-1689 *28-1689 *33-1844 *33-1845 71-2582 *71-2582 *71-2583 *71-2599
PARCOU	002400	*71-2600 *72-2625 *72-2639 *78-3085 *78-3142
PARTAB	001356	#16-1276 *27-1654 27-1664 *28-1676 28-1688 *76-2869 *76-2876 76-2887 *76-2889
PATAND	001314	76-2893 *77-3006 *77-3013 77-3024 *77-3026 77-3030
PATRNS	027024	#16-1276 *76-2870 76-2892 *77-3007 77-3029
PATTOR	001312	#16-1276 *28-1675 57-2375 58-2380 59-2385 60-2390 61-2395 62-2400
PBAHI	001304	#16-1276 69-2456 69-2462 69-2465
PBALO	001302	#16-1276 *26-1642 28-1680 102-3466
PCPUER	001332	74-2696 #75-2719
PFECDF	034260	#16-1276 *26-1638 *28-1679 102-3466
PFECDH	034220	#16-1276 *32-1831
PFECDT	034250	#16-1276 *32-1822 *32-1823 *32-1828 102-3458 102-3460
PFECM	034160	#16-1276 *30-1750 30-1754 *76-2782 76-2788 *76-2832 76-2836 *76-2857 76-2861
PFECWS	034150	*76-2871 76-2872 76-2880 *77-2926 77-2928 *77-2993 77-2997 *77-3008 77-3009
PIRQ	= 177772	77-3017
PIRQVE	= 000240	88-3309 #88-3314
PRO	= 000000	88-3309 #88-3311
PR1	= 000040	88-3309 #88-3313
PR2	= 000100	88-3309 #88-3310
PR3	= 000140	88-3309 #88-3309
PR4	= 000200	88-3233 #88-3309
PR5	= 000240	#15-1199
PR6	= 000300	#15-1199
PR7	= 000340	#15-1199
PS	= 177776	#15-1199 15-1199
PSW	= 177776	#15-1199 23-1586 23-1588 32-1809 32-1813 *35-1870 35-1871 *36-1891 *36-1892
		36-1893 *36-1897 *36-1905 *37-1915 *37-1917 37-1918 *37-1923 *37-1929 *37-1931
		37-1932 *37-1936 *38-1955 *38-1957 *38-1959 *38-1961 *38-1971 *38-1973 *38-1975
		*38-1977 *38-1981 *39-2003 *47-2168 *47-2170 *47-2178 *48-2252 *48-2265 *48-2276
		*78-3053 78-3086 78-3107 *78-3117 78-3119 *78-3129 *78-3131 *81-3177 *81-3179
		*82-3184 *82-3186
PWRMSG	= 037416	99-3349 #99-3350
PWRVEC	= 000024	#15-1199 *33-1838 *33-1838 *99-3349 *99-3349 *99-3349 *99-3349 *99-3349 *99-3349
RDCHR	= 104411	89-3315 #98-3347
RDLIN	= 104412	#98-3347
READON	001342	#16-1276 48-2229 48-2241

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES										
RESREG	=	104414	97-3345	#98-3347									
RESVEC	=	000010	#15-1199	*33-1838	33-1838	*33-1838							
RMIREG	=	177770	#15-1216	75-2748	*76-2800	*76-2826	*77-2940						
R6	=	2000006	#15-1199	*33-1838	*33-1838	33-1838							
R7	=	2000007	#15-1199										
SAVREG	=	104413	97-3345	#98-3347									
SCOPE	=	000004	#15-1199	35-1866	36-1888	37-1913	38-1953	39-2002	40-2039	41-2050	42-2061		
			43-2072	44-2083	45-2094	46-2114	47-2167	48-2191	49-2286	51-2332	52-2337		
			53-2342	54-2347	55-2352	56-2357	57-2374	58-2379	59-2384	60-2389	61-2394		
			62-2399	63-2411	64-2416	65-2421	66-2426	67-2431	68-2436	69-2451	70-2478		
			71-2576	72-2616	74-2674	76-2768	77-2908	78-3052	80-3171	81-3176	82-3183		
			83-3197	84-3215									
SDPAR0	=	172260	#15-1200										
SDPAR1	=	172262	#15-1200										
SDPAR2	=	172264	#15-1200										
SDPAR3	=	172266	#15-1200										
SDPAR4	=	172270	#15-1200										
SDPAR5	=	172272	#15-1200										
SDPAR6	=	172274	#15-1200										
SDPAR7	=	172276	#15-1200	52-2338	52-2338	58-2380							
SDPDR0	=	172220	#15-1200										
SDPDR1	=	172222	#15-1200										
SDPDR2	=	172224	#15-1200										
SDPDR3	=	172226	#15-1200										
SDPDR4	=	172230	#15-1200										
SDPDR5	=	172232	#15-1200										
SDPDR6	=	172234	#15-1200										
SDPDR7	=	172236	#15-1200	55-2353	55-2353	61-2395							
SIPARO	=	172240	#15-1200	32-1815	42-2063	52-2338	52-2338	52-2338	52-2338	52-2338	58-2380		
			64-2417	*64-2417	64-2417	64-2417	64-2417	70-2487	70-2531	78-3059			
SIPAR1	=	172242	#15-1200										
SIPAR2	=	172244	#15-1200										
SIPAR3	=	172246	#15-1200	81-3178									
SIPAR4	=	172250	#15-1200	*78-3081	*78-3121	*78-3140							
SIPAR5	=	172252	#15-1200	*78-3082	*78-3122	*78-3141							
SIPAR6	=	172254	#15-1200										
SIPAR7	=	172256	#15-1200										
SIPDR0	=	172200	#15-1200	43-2074	55-2353	55-2353	55-2353	55-2353	55-2353	55-2353	61-2395	67-2432	
			*67-2432	67-2432	67-2432	67-2432	70-2486	70-2520	74-2678	78-3069	81-3178		
			81-3178										
SIPDR1	=	172202	#15-1200										
SIPDR2	=	172204	#15-1200										
SIPDR3	=	172206	#15-1200										
SIPDR4	=	172210	#15-1200	*78-3077	*78-3123								
SIPDR5	=	172212	#15-1200	*78-3078	*78-3124								
SIPDR6	=	172214	#15-1200										
SIPDR7	=	172216	#15-1200										
SRC	=	177572	#15-1200	15-1203	28-1689	*28-1689	*28-1689	31-1782	*31-1785	39-2006	46-2115		
			*47-2169	47-2171	*47-2177	*71-2588	71-2590	*71-2592	*71-2598	*72-2628	*72-2645		
			72-2646	*72-2651	*78-3083	78-3153	*78-3155						
SR1	=	177574	#15-1200	15-1204	31-1783								
SR2	=	177576	#15-1200	15-1205	31-1784	46-2125	46-2146	72-2647	78-3154				

SYMBOL	CROSS REFERENCE	REFERENCES								
SYMBOL	VALUE									
SR3	= 172516	#15-1200	15-1206	*28-1689	39-2015	39-2033				
SSP	=X000006	#15-1208	*38-1958	*38-1974	48-2253	*48-2254	*48-2255	*48-2262		
STACK	= 001100	#15-1199	15-1212	15-1213	15-1214	33-1838	33-1841			
START	020000	15-1273	#33-1838	99-3349						
STKLMT	= 177774	#15-1199								
SUPSTK	= 000700	#15-1213	38-1958							
SWR	001140	#16-1276	33-1838	*33-1838	33-1838	*33-1838	*33-1838	33-1839	84-3215	85-3217
		85-3217	85-3217	85-3217	86-3219	86-3219	86-3219	86-3219	89-3315	89-3315
		99-3349	99-3349							
SWREG	000176	#15-1273	33-1838	33-1839	89-3315	89-3315				
SW0	= 000001	#15-1199								
SW00	= 000001	#15-1199	15-1199							
SW01	= 000002	#15-1199	15-1199							
SW02	= 000004	#15-1199	15-1199							
SW03	= 000010	#15-1199	15-1199							
SW04	= 000020	#15-1199	15-1199							
SW05	= 000040	#15-1199	15-1199							
SW06	= 000100	#15-1199	15-1199							
SW07	= 000200	#15-1199	15-1199							
SW08	= 000400	#15-1199	15-1199							
SW09	= 001000	#15-1199	15-1199							
SW1	= 000002	#15-1199								
SW10	= 002000	#15-1199								
SW11	= 004000	#15-1199								
SW12	= 010000	#15-1199								
SW13	= 020000	#15-1199								
SW14	= 040000	#15-1199								
SW15	= 000000	#15-1199								
SW2	= 000004	#15-1199								
SW3	= 000010	#15-1199								
SW4	= 000020	#15-1199								
SW5	= 000040	#15-1199								
SW6	= 000100	#15-1199								
SW7	= 000200	#15-1199								
SW8	= 000400	#15-1199								
SW9	= 001000	#15-1199								
TBIT	= 000020	#15-1210	23-1586	23-1591	24-1603					
TBITPS	001272	#16-1276	*23-1589	24-1603	24-1605	*24-1606	*33-1849			
TBITVE	= 000014	#15-1199	*33-1838	*33-1838						
TESTNO	001254	#16-1276	*86-3219	102-3447	102-3448	102-3449	102-3450	102-3451	102-3452	102-3453
		102-3454	102-3455	102-3456	102-3457	102-3458	102-3459	102-3460	102-3461	102-3462
		102-3463	102-3464	102-3465	102-3466	102-3467	102-3468	102-3469	102-3470	102-3471
		102-3472	102-3473	102-3474	102-3475	102-3476	102-3477			
		29-1726	*30-1740	33-1842	39-2021	76-2827	77-2962	83-3212		
TIMERR	003232	#30-1741	*30-1758	*33-1847						
TIMFLG	003234	#29-1691	29-1706	40-2040	41-2051	42-2062	43-2073	44-2084	45-2095	
TIMTST	003024	#15-1199								
TKVEC	= 000060	#15-1199								
TOFF	002172	#23-1586	71-2577	72-2617	80-3172	81-3178	82-3185	83-3198		
TON	002226	#24-1603	71-2602	72-2642	80-3172	81-3178	82-3185	83-3214		
TONUM	001274	#16-1276	*29-1702	29-1707	*29-1723	*39-2010	39-2018	*39-2032	102-3451	
TPVEC	= 000064	#15-1199								
TRAPP	001260	#16-1276	*30-1748	30-1760	*31-1779	31-1789	*72-2649	72-2657	*78-3146	78-3162

SYMBOL	CROSS REFERENCE	REFERENCES							
SYMBOL	VALUE								
TRAPPS	001262	102-3447 102-3448 #16-1276 *30-1749 102-3447 102-3448	30-1759	*31-1780	31-1788	*72-2650	72-2656	*78-3147	78-3161
TRAPVE	= 000034	#15-1199 *33-1838	*33-1838						
TRTVEC	= 000014	#15-1199							
TST1	020564	#35-1866							
TST10	021422	#42-2061							
TST11	021440	#43-2072							
TST12	021456	#44-2083							
TST13	021474	#45-2094							
TST14	021512	#46-2114							
TST15	021654	#47-2167							
TST16	021714	#48-2191							
TST17	022320	#49-2286							
TST2	020642	#36-1888							
TST20	022442	#51-2332							
TST21	022640	51-2333	#52-2337						
TST22	023036	52-2338	#53-2342						
TST23	023234	53-2343	#54-2347						
TST24	023432	54-2348	#55-2352						
TST25	023630	55-2353	#56-2357						
TST26	024026	56-2358	#57-2374						
TST27	024140	57-2375	#58-2379						
TST3	020726	#37-1913							
TST30	024252	58-2380	#59-2384						
TST31	024364	59-2385	#60-2389						
TST32	024502	60-2390	#61-2394						
TST33	024620	61-2395	#62-2399						
TST34	024736	62-2400	#63-2411						
TST35	025042	#64-2416							
TST36	025146	#65-2421							
TST37	025252	#66-2426							
TST4	021050	#38-1953							
TST40	025356	#67-2431							
TST41	025462	#68-2436							
TST42	025566	#69-2451							
TST43	025662	#70-2478							
TST44	026210	#71-2576							
TST45	026344	#72-2616							
TST46	026602	72-2643	#74-2674						
TST47	027276	75-2744	75-2746	75-2752	#76-2768				
TST5	021214	#39-2002							
TST50	030136	76-2867	#77-2908						
TST51	031020	77-3004	#78-3052						
TST52	031634	78-3144	#80-3171						
TST53	032030	#81-3176							
TST54	032236	#82-3183							
TST55	032444	#83-3197							
TST6	021366	39-2022	#40-2039						
TST7	021404	#41-2050							
TYPBN	= 104406	88-3271	#98-3347						
TYPDS	= 104405	84-3215	84-3215	88-3265	90-3326	#98-3347			

SYMBOL CROSS REFERENCE

SYMBOL VALUE

TYPE = 104401

REFERENCES

SYMBOL	CROSS REFERENCE	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES
TYPE	VALUE									
	= 104401	33-1839	84-3215	84-3215	84-3215	86-3219	86-3219	88-3221	88-3243	88-3245
		88-3248	88-3250	88-3280	88-3294	88-3300	88-3305	89-3315	89-3315	89-3315
		89-3315	89-3315	89-3315	89-3315	89-3315	89-3315	89-3315	89-3315	89-3315
		89-3315	89-3315	89-3315	89-3315	89-3315	90-3320	90-3324	91-3333	93-3337
		94-3339	95-3341	*98-3347	99-3349					
		88-3229	88-3259	89-3315	90-3323	*98-3347				
		*98-3347								
TYPON	= 104404	#98-3347								
TYPON	= 104404	#98-3347								
TYPON	= 104403	#98-3347								
UDPAR0	= 177660	#15-1200								
UDPAR1	= 177662	#15-1200								
UDPAR2	= 177664	#15-1200								
UDPAR3	= 177666	#15-1200								
UDPAR4	= 177670	#15-1200								
UDPAR5	= 177672	#15-1200								
UDPAR6	= 177674	#15-1200								
UDPAR7	= 177676	#15-1200	53-2343	53-2343						
UDPDR0	= 177620	#15-1200								
UDPDR1	= 177622	#15-1200								
UDPDR2	= 177624	#15-1200								
UDPDR3	= 177626	#15-1200								
UDPDR4	= 177630	#15-1200								
UDPDR5	= 177632	#15-1200								
UDPDR6	= 177634	#15-1200								
UDPDR7	= 177636	#15-1200	56-2358	56-2358	62-2400					
UIPAR0	= 177640	#15-1200	32-1811	44-2085	53-2343	53-2343	53-2343	53-2343	53-2343	65-2422
		*65-2422	65-2422	65-2422	65-2422	70-2493	70-2553	78-3058		
UIPAR1	= 177642	#15-1200								
UIPAR2	= 177644	#15-1200								
UIPAR3	= 177646	#15-1200	82-3185							
UIPAR4	= 177650	#15-1200	*78-3079	*78-3109	*78-3127	*78-3138				
UIPAR5	= 177652	#15-1200	*78-3080	*78-3110	*78-3128	*78-3139				
UIPAR6	= 177654	#15-1200								
UIPAR7	= 177656	#15-1200								
UIPDR0	= 177600	#15-1200	45-2096	56-2358	56-2358	56-2358	56-2358	56-2358	62-2400	68-2437
		*68-2437	68-2437	68-2437	68-2437	70-2492	70-2542	74-2682	78-3068	82-3185
		82-3185	82-3185							
UIPDR1	= 177602	#15-1200								
UIPDR2	= 177604	#15-1200								
UIPDR3	= 177606	#15-1200								
UIPDR4	= 177610	#15-1200	*78-3075	*78-3111	*78-3125	*78-3134				
UIPDR5	= 177612	#15-1200	*78-3076	*78-3112	*78-3126	*78-3135				
UIPDR6	= 177614	#15-1200								
UIPDR7	= 177616	#15-1200								
USESTK	= 000600	#15-1214	38-1960							
USP	= X000006	#15-1209	*38-1960	*38-1976	48-2266	*48-2267	*48-2268	*48-2275		
VIRT1	001276	#16-1276	32-1816	32-1822	*78-3094	78-3102	*78-3151	102-3458	102-3460	
VIRT2	001300	#16-1276	*78-3093	102-3458						
WASR6	001256	#16-1276	*30-1751	*31-1781	*72-2648	102-3447	102-3448			
WASSR0	001264	#16-1276	*31-1782	*72-2646	*78-3153	102-3448	102-3460			
WASSR1	001266	#16-1276	*31-1783	102-3448						
WASSR2	001270	#16-1276	*31-1784	*46-2125	46-2127	*46-2146	46-2147	*72-2647	*78-3154	102-3448
		102-3459	102-3460	102-3463						

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES	CREF	V01						
WBIT	=	000100	#15-1211	80-3172	80-3172	80-3172	81-3178	81-3178	81-3178	82-3185	82-3185
			82-3185								
SAPTHD		000204	15-1275	#15-1275							
SASTAT	=	*****	92-3335	#92-3335							
SATYC		035760	92-3335	#92-3335							
SATY1		035734	#92-3335								
SATY3		035742	91-3333	#92-3335							
SATY4		035752	86-3219	#92-3335							
SAUTOB		001134	#16-1276	*33-1839	89-3315	89-3315	89-3315				
SBADR		001122	#16-1276								
SBDDAT		001126	#16-1276								
SBELL		001214	#16-1276	86-3219	87-3219	87-3219					
SBIN		036254	*93-3337	*93-3337	93-3337	#93-3337					
SCHARC		035730	*91-3333	*91-3333	91-3333	*91-3333	#91-3333				
SCKSWR		034264	#89-3315	98-3347	98-3347						
SCLR.T		032732	84-3215	#84-3215							
SCMTAG		001100	#16-1276	33-1838	33-1838	33-1838	33-1838	33-1838			
SCM1	=	000006	#16-1276	16-1276	16-1276	#16-1276	16-1276	16-1276	#16-1276	16-1276	16-1276
			#16-1276	16-1276	16-1276	#16-1276	16-1276	16-1276	#16-1276	16-1276	16-1276
			#16-1276	16-1276	16-1276	#16-1276	16-1276	16-1276	#16-1276	16-1276	16-1276
SCM2	=	000014	#16-1276	16-1276	16-1276	#16-1276	16-1276	16-1276	#16-1276	16-1276	16-1276
			#16-1276	16-1276	16-1276	#16-1276	16-1276	16-1276	#16-1276	16-1276	16-1276
			#16-1276	16-1276	16-1276	#16-1276	16-1276	16-1276	#16-1276	16-1276	16-1276
SCM3	=	000006	#16-1276	16-1276	16-1276	#16-1276	16-1276	16-1276	#16-1276	16-1276	16-1276
SCM4	=	000006	#16-1276	16-1276	16-1276	#16-1276	16-1276	16-1276	#16-1276	16-1276	16-1276
			#16-1276	16-1276	16-1276	#16-1276	16-1276	16-1276	#16-1276	16-1276	16-1276
SCNTLC		001350	#16-1276	89-3315	89-3315	89-3315	89-3315				
SCNTLG		035231	89-3315	#89-3315							
SCNTLU		035224	89-3315	#89-3315	#89-3315						
SCORE		002676	28-1689	#28-1689							
SCPUOP		001252	#16-1276								
SCRFL		001221	#16-1276	84-3215	86-3219	87-3219	87-3219	88-3221	88-3245	88-3250	88-3305
			89-3315	89-3315	89-3315	89-3315	91-3333	91-3333	91-3333		
SCROUT		002726	28-1689	#28-1689							
SDBLK		036720	95-3341	95-3341	#95-3341						
SDB20		037024	88-3277	88-3291	#97-3345						
SDEVCT		001234	#16-1276								
SDOAGN		032752	84-3215	84-3215	#84-3215						
SDTBL		036710	95-3341	#95-3341							
SENDAD		032742	15-1274	33-1839	#84-3215	86-3219					
SENDCT		032570	33-1838	#84-3215							
SENULL		033014	#84-3215								
SENV		001244	#16-1276	33-1839	75-2743	86-3219	91-3333	92-3335	92-3335		
SENVN		001245	#16-1276	33-1838	91-3333	91-3333	92-3335				
SEOP		032540	#84-3215	90-3328							
SEOPCT		032562	*33-1838	#84-3215	84-3215						
SERFLG		001103	#16-1276	27-1659	28-1683	85-3217	*85-3217	85-3217	*85-3217	85-3217	85-3217
			*86-3219	87-3219	87-3219						
SERMAX		001115	#16-1276	*33-1838	*85-3217	85-3217	85-3217				
SERROR		033246	33-1838	#86-3219							
SERRPC		001116	#16-1276	*86-3219	*86-3219	86-3219	87-3219	88-3227	88-3313	102-5447	

SYMBOL	CROSS REFERENCE VALUE	REFERENCES	REF	V01						
		102-3448	102-3449	102-3450	102-3451	102-3452	102-3453	102-3454	102-3455	102-3456
		102-3457	102-3458	102-3459	102-3460	102-3461	102-3462	102-3463	102-3467	102-3470
		102-3471	102-3472	102-3473	102-3474	102-3475	102-3476	102-3477	102-3478	
\$ERRTB	001372	#17-1276	88-3240							
\$ERTTL	001112	#16-1276	84-3215	*84-3215	*86-3219	87-3219	87-3219			
\$ESCAP	001212	#16-1276	*33-1838	*85-3217	86-3219	86-3219	87-3219	87-3219		
\$ETABL	001244	#16-1276								
\$ETEND	001254	15-1275	#16-1276							
\$FATAL	001226	#16-1276	*92-3335							
\$FFLG	036200	*92-3335	*92-3335	92-3335	*92-3335	#92-3335				
\$FILLC	001156	#16-1276	91-3333	91-3333	91-3333					
\$FILLS	001155	#16-1276	91-3333	91-3333						
\$GDADR	001120	#16-1276								
\$GDDAT	001124	#16-1276								
\$GET42	032714	#84-3215								
\$GTSWR	034334	#89-3315	98-3347	98-3347						
\$HD	= 000000	15-1197	15-1197	15-1197						
\$HIBTS	000204	#15-1275								
\$ICNT	001104	#16-1276								
\$ILLUP	037410	99-3349	99-3349	#99-3349						
\$INTAG	001135	#16-1276	89-3315	89-3315	89-3315	89-3315				
\$ITEMB	001114	#16-1276	*86-3219	86-3219	86-3219	*86-3219	86-3219	87-3219	87-3219	88-3224
\$KTNEX	002670	28-1689	#28-1689							
\$KTOUT	002660	28-1689	#28-1689							
\$KT11	002512	#28-1689	*28-1689	*28-1689	*33-1853					
\$LF	001222	#16-1276	87-3219	87-3219	89-3315	89-3315	89-3315	91-3333	91-3333	
\$LFLG	036177	*92-3335	#92-3335							
\$LOOP	033010	84-3215	#84-3215							
\$LPADR	001106	#16-1276	*33-1838	76-2889	77-3026	*85-3217	*85-3217	85-3217	85-3217	85-3217
\$LPERR	001110	#16-1276	*29-1696	*29-1706	*33-1838	*35-1867	*35-1882	*36-1889	*36-1904	*37-1914
		*37-1928	*37-1941	*38-1970	*39-2004	*39-2017	*46-2124	*46-2133	*46-2157	*48-2194
		*48-2202	*48-2207	*48-2217	*48-2227	*48-2238	*48-2251	*48-2264	*48-2278	*49-2294
		*49-2303	*49-2309	*49-2314	*51-2333	*51-2333	*51-2333	*52-2338	*52-2338	*52-2338
		*53-2343	*53-2343	*53-2343	*54-2348	*54-2348	*54-2348	*55-2353	*55-2353	*55-2353
		*56-2358	*56-2358	*56-2358	*57-2375	*57-2375	*58-2380	*58-2380	*59-2385	*59-2385
		*60-2390	*60-2390	*61-2395	*61-2395	*62-2400	*62-2400	*63-2412	*63-2412	*63-2412
		*64-2417	*64-2417	*64-2417	*65-2422	*65-2422	*65-2422	*66-2427	*66-2427	*66-2427
		*67-2432	*67-2432	*67-2432	*68-2437	*68-2437	*68-2437	*69-2452	*69-2460	*69-2468
		*71-2587	*71-2601	*72-2618	*72-2641	*74-2697	*76-2781	*76-2793	*76-2828	*76-2845
		*76-2853	*76-2866	*77-2921	*77-2933	*77-2966	*77-2981	*77-2989	*77-3002	*78-3084
		*78-3143	*80-3172	*80-3172	*81-3178	*81-3178	*82-3185	*82-3185	*83-3200	*83-3213
		85-3217	*85-3217	85-3217	85-3217	86-3219				
\$LSTAD	003020	*28-1689	#28-1689							
\$LSTBK	003022	*28-1689	#28-1689	74-2715	76-2773	76-2785	76-2854	76-2858	76-2874	77-2914
		77-2924	77-2949	77-2990	77-2994	77-3011	102-3474			
\$MAIL	001224	15-1275	15-1275	#16-1276	33-1838	33-1839	85-3217	86-3219	91-3333	
\$MEADR	000206	#15-1275								
\$MFLG	036176	*92-3335	92-3335	*92-3335	#92-3335					
\$MNEW	035247	89-3315	#89-3315							
\$MSGAD	001240	#16-1276	*92-3335	92-3335						
\$MSGLG	001242	#16-1276	*92-3335							
\$MSGTY	001224	#16-1276	92-3335	*92-3335	92-3335	*92-3335				

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
\$MSWR		035236	89-3315 #89-3315
\$MXCNT		001344	#16-1276
\$NULL		001154	#16-1276 91-3333 91-3333 91-3333
\$NWTST	=	000001	#34-1866 #35-1866 35-1866 #35-1888 35-1888 #36-1888 36-1888 #36-1913 36-1913 #37-1913 37-1913 #37-1953 37-1953 #38-1953 38-1953 #38-2002 38-2002 #39-2002 39-2002 #39-2039 39-2039 #40-2039 40-2039 #40-2050 40-2050 #41-2050 41-2050 #41-2061 41-2061 #42-2061 42-2061 #42-2072 42-2072 #43-2072 43-2072 #43-2083 43-2083 #44-2083 44-2083 #44-2094 44-2094 #45-2094 45-2094 #45-2114 45-2114 #46-2114 46-2114 #46-2167 46-2167 #47-2167 47-2167 #48-2191 48-2191 #48-2286 48-2286 #49-2286 49-2286 #50-2332 50-2332 #51-2332 51-2332 #51-2337 51-2337 #52-2337 52-2337 #52-2342 52-2342 #53-2342 53-2342 #53-2347 53-2347 #54-2347 54-2347 #54-2352 54-2352 #55-2352 55-2352 #55-2357 55-2357 #56-2357 56-2357 #56-2374 56-2374 #57-2374 57-2374 #57-2379 57-2379 #58-2379 58-2379 #58-2384 58-2384 #59-2384 59-2384 #59-2389 59-2389 #60-2389 60-2389 #60-2394 60-2394 #61-2394 61-2394 #61-2399 61-2399 #62-2399 62-2399 #62-2411 62-2411 #63-2411 63-2411 #63-2416 63-2416 #64-2416 64-2416 #64-2421 64-2421 #65-2421 65-2421 #65-2426 65-2426 #66-2426 66-2426 #66-2431 66-2431 #67-2431 67-2431 #67-2436 67-2436 #68-2436 68-2436 #68-2451 68-2451 #69-2451 69-2451 #69-2478 69-2478 #70-2478 70-2478 #70-2576 70-2576 #71-2576 71-2576 #71-2616 71-2616 #72-2616 72-2616 #73-2674 73-2674 #74-2674 74-2674 #75-2768 75-2768 #76-2768 76-2768 #76-2908 76-2908 #77-2908 77-2908 #77-3052 77-3052 #78-3052 78-3052 #79-3171 79-3171 #80-3171 80-3171 #80-3176 80-3176 #81-3176 81-3176 #81-3183 81-3183 #82-3183 82-3183 #82-3197 82-3197 #83-3197 83-3197
\$OCNT		036500	*94-3339 *94-3339 #94-3339
\$OCTVL		037126	97-3345 #97-3345
\$OMODE		036502	*94-3339 *94-3339 94-3339 *94-3339 *94-3339 #94-3339
\$OVER		033230	85-3217 85-3217 #85-3217
\$PASS		001232	#16-1276 *33-1838 *84-3215 *84-3215 84-3215 84-3215 84-3215 90-3318
\$PASTM		000212	#15-1275
\$PWAD		037372	#99-3349
\$PWADN		037232	33-1838 #99-3349 99-3349
\$PWIMG		037366	#99-3349
\$PWIRUP		037304	99-3349 #99-3349
\$QUES		001220	#16-1276 87-3219 87-3219 89-3315 89-3315 89-3315 89-3315 91-3333 91-3333
\$RDCHR		034606	#89-3315 98-3347 98-3347
\$RDDEC	=	*****	98-3347
\$RDLIN		034736	#89-3315 98-3347 98-3347
\$RDOCT	=	*****	98-3347
\$RDSZ	=	000010	#89-3315 89-3315
\$REGAD		001160	#16-1276
\$REGO		001162	#16-1276 *86-3219 102-3449 102-3450 102-3452 102-3453 102-3456 102-3457 102-3464 102-3467 102-3468 102-3469 102-3471 102-3476
\$REG1		001164	#16-1276 *86-3219 102-3449 102-3452 102-3452 102-3453 102-3454 102-3455 102-3456 102-3457 102-3459 102-3462 102-3467 102-3468 102-3469 102-3473
\$REG2		001166	#16-1276 *86-3219 102-3453 102-3453 102-3455 102-3456 102-3457 102-3462 102-3464 102-3467 102-3469 102-3471
\$REG3		001170	#16-1276 *86-3219 102-3454 102-3471 102-3475 102-3476
\$REG4		001172	#16-1276 *86-3219 102-3458 102-3460 102-3463 102-3469
\$REG5		001174	#16-1276 *86-3219 102-3458 102-3475 102-3476 102-3477
\$RESRE		036766	#96-3343 98-3347
\$RTNAD		033012	#84-3215
\$RTRN		033006	33-1838 *33-1838 *33-1838 84-3215 #84-3215

SYMBOL	CROSS REFERENCE VALUE	REFERENCES								
\$R2A	= *****	98-3347								
\$SAVRE	036730	#96-3343	98-3347	98-3347						
\$SAVR6	037414	*99-3349	99-3349	*99-3349	*99-3349	#99-3349				
\$SCOPE	033020	33-1838	#85-3217							
\$SETUP	= 000137	#15-1272	15-1272	#15-1272	15-1272	#15-1272	15-1272	#15-1272	15-1272	#15-1272
		15-1272	#15-1272	15-1272	#15-1272	33-1838	33-1838	#15-1272	33-1838	33-1838
		33-1838	33-1838	33-1838	33-1838	33-1838	33-1838	33-1838	33-1838	33-1839
		33-1839	84-3215	84-3215	85-3217	86-3219	86-3219	86-3219	86-3219	89-3315
		89-3315	99-3349							
\$SIZE	002454	#28-1689	33-1854							
\$SIZEX	002732	28-1689	#28-1689							
\$STUP	= 177777	#15-1272	#15-1272	15-1272	#15-1272	#15-1272	15-1272	#15-1272	#15-1272	15-1272
		#15-1272	#15-1272	15-1272	#15-1272	#15-1272	15-1272	#15-1272	#15-1272	15-1272
\$SVLAD	033174	85-3217	85-3217	#85-3217						
\$SVPC	= 000204	#15-1274	15-1274							
\$SWR	= 173400	#15-1193	15-1197	15-1198	15-1198	15-1198	15-1198	15-1198	15-1198	15-1198
		15-1198	16-1276	16-1276	16-1276	33-1838	33-1838	33-1838	33-1838	33-1838
		35-1866	36-1888	37-1913	38-1953	39-2002	40-2039	41-2050	42-2061	43-2072
		44-2083	45-2094	46-2114	47-2167	48-2191	49-2286	51-2332	52-2337	53-2342
		54-2347	55-2352	56-2357	57-2374	58-2379	59-2384	60-2389	61-2394	62-2399
		63-2411	64-2416	65-2421	66-2426	67-2431	68-2436	69-2451	70-2478	71-2576
		72-2616	74-2674	76-2768	77-2908	78-3052	80-3171	81-3176	82-3183	83-3197
		84-3215	84-3215	84-3215	84-3215	84-3215	85-3217	85-3217	85-3217	85-3217
		85-3217	85-3217	85-3217	85-3217	85-3217	85-3217	85-3217	85-3217	85-3217
		85-3217	85-3217	85-3217	85-3217	85-3217	85-3217	85-3217	85-3217	85-3217
		86-3219	86-3219	86-3219	86-3219	86-3219	86-3219	86-3219	86-3219	87-3219
		99-3349								
\$SWREG	001246	#16-1276	33-1838							
\$SWRMK	= 000000	15-1198	15-1198	15-1198	15-1198	15-1198	15-1198	15-1198	15-1198	15-1198
		85-3217	85-3217	85-3217	85-3217	85-3217	85-3217	85-3217	85-3217	85-3217
		85-3217								
\$TBIT	001346	#16-1276	*33-1838	*84-3215	84-3215	84-3215	*99-3349			
\$TESTN	001230	#16-1276	*85-3217	88-3313						
\$TKB	001146	#16-1276	89-3315	89-3315	89-3315	89-3315	89-3315	89-3315	91-3333	91-3333
		91-3333	91-3333							
\$TKS	001144	#16-1276	89-3315	89-3315	89-3315	89-3315	89-3315	89-3315	89-3315	89-3315
		91-3333	91-3333	91-3333	91-3333					
\$TMP0	001176	#16-1276	*48-2253	48-2262	*48-2266	48-2275	*51-2333	*51-2333	*52-2338	*52-2338
		*53-2343	*53-2343	*54-2348	*54-2348	*55-2353	*55-2353	*56-2358	*56-2358	*74-2701
		74-2710	*76-2787	76-2805	*77-2927	77-2945	*77-2968	77-2971	*78-3086	102-3458
		102-3460								
\$TMP1	001200	#16-1276	*76-2884	*77-3021	102-3470	102-3472				
\$TMP2	001202	#16-1276	*76-2792	*77-2932	102-3470					
\$TMP3	001204	#16-1276								
\$TMP4	001206	#16-1276	29-1691	29-1693	*29-1711	*29-1714	*88-3288	88-3290		
\$TMP5	001210	#16-1276	*51-2333	*52-2338	*53-2343	*54-2348	*55-2353	*56-2358	*57-2375	*58-2380
		*59-2385	*60-2390	*61-2395	*62-2400	*88-3289	*90-3318	*90-3319	90-3325	
		#15-1194	15-1197	34-1866	35-1866	#35-1866	35-1888	36-1888	#36-1888	36-1913
\$TN	- 000056	37-1913	#37-1913	37-1953	38-1953	#38-1953	38-2002	39-2002	#39-2002	39-2022
		39-2039	40-2039	#40-2039	40-2050	41-2050	#41-2050	41-2061	42-2061	#42-2061
		42-2072	43-2072	#43-2072	43-2083	44-2083	#44-2083	44-2094	45-2094	#45-2094
		45-2114	46-2114	#46-2114	46-2167	47-2167	#47-2167	47-2191	48-2191	#48-2191

SYMBOL	CROSS REFERENCE VALUE	REFERENCES	48-2286	49-2286	#49-2286	50-2332	51-2332	#51-2332	51-2333	51-2337	52-2337
			#52-2337	52-2338	52-2342	53-2342	#53-2342	53-2343	53-2347	54-2347	#54-2347
			54-2348	54-2352	55-2352	#55-2352	55-2353	55-2357	56-2357	#56-2357	56-2358
			56-2374	57-2374	#57-2374	57-2375	57-2379	58-2379	#58-2379	58-2380	58-2384
			59-2384	#59-2384	59-2385	59-2389	60-2389	#60-2389	60-2390	60-2394	61-2394
			#61-2394	61-2395	61-2399	62-2399	#62-2399	62-2400	62-2411	63-2411	#63-2411
			63-2416	64-2416	#64-2416	64-2421	65-2421	#65-2421	65-2426	66-2426	#66-2426
			66-2431	67-2431	#67-2431	67-2436	68-2436	#68-2436	68-2451	69-2451	#69-2451
			69-2478	70-2478	#70-2478	70-2576	71-2576	#71-2576	71-2616	72-2616	#72-2616
			72-2643	73-2674	74-2674	#74-2674	75-2744	75-2746	75-2752	75-2768	76-2768
			#76-2768	76-2867	76-2908	77-2908	#77-2908	77-3004	77-3052	78-3052	#78-3052
			78-3144	79-3171	80-3171	#80-3171	80-3176	81-3176	#81-3176	81-3183	82-3183
			#82-3183	82-3197	83-3197	#83-3197					
\$TPB	001152		#16-1276	91-3333	91-3333	91-3333	91-3333				
\$TPFLG	001157		#16-1276	91-3333	91-3333	91-3333	91-3333				
\$TPS	001150		#16-1276	91-3333	91-3333	91-3333	91-3333				
\$TRAP	037144		33-1838	#98-3347							
\$TRAP2	037166		#98-3347	98-3347							
\$TRP	= 000015		#98-3347	98-3347	98-3347	98-3347	98-3347	#98-3347	98-3347	98-3347	98-3347
			98-3347	#98-3347	98-3347	98-3347	98-3347	#98-3347	98-3347	98-3347	98-3347
			98-3347	#98-3347	98-3347	98-3347	98-3347	#98-3347	98-3347	98-3347	98-3347
			98-3347	#98-3347	98-3347	98-3347	98-3347	#98-3347	98-3347	98-3347	98-3347
			98-3347	#98-3347	98-3347	98-3347	98-3347	#98-3347	98-3347	98-3347	98-3347
			98-3347	#98-3347	98-3347	98-3347	98-3347	#98-3347	98-3347	98-3347	98-3347
			98-3347	#98-3347	98-3347	98-3347	98-3347	#98-3347	98-3347	98-3347	98-3347
\$TRPAD	037200		98-3347	#98-3347							
\$STSM	000210		#15-1275								
\$STSNM	001102		#16-1276	*84-3215	85-3217	85-3217	*85-3217	85-3217	85-3217	85-3217	85-3217
			86-3219	86-3219	87-3219	87-3219	90-3321				
\$TTYIN	035214		89-3315	89-3315	89-3315	89-3315	89-3315	#89-3315			
\$TYPBN	036202		#93-3337	98-3347	98-3347	98-3347					
\$TYPDS	036504		#95-3341	98-3347	98-3347	98-3347					
\$TYPE	035370		#91-3333	92-3335	98-3347	98-3347					
\$TYPEC	035602		89-3315	91-3333	91-3333	91-3333	#91-3333				
\$TYPEX	035732		91-3333	91-3333	91-3333	#91-3333					
\$TYPOC	036302		#94-3339	98-3347	98-3347	98-3347					
\$TYPON	036316		94-3339	#94-3339	98-3347	98-3347					
\$TYPOS	036256		#94-3339	98-3347							
\$UNIT	001236		#16-1276								
\$UNITM	000214		#15-1275								
\$USWR	001250		#16-1276	75-2745							
\$XOFF	= 000023		91-3333	91-3333							
\$XON	= 000021		89-3315	91-3333	91-3333	91-3333					
\$XTSTR	033032		#85-3217								
\$SGET4	= 000001		#84-3215	#84-3215	84-3215						
\$OFILL	036501		*94-3339	*94-3339	94-3339	#94-3339					
\$LOCAT	= *****		85-3217	86-3219							
.\$ASTA	= *****		92-3335	92-3335							
.\$X	= 000204		#15-1275	*15-1275							

MACRO CROSS REFERENCE

MACRO NAME	REFERENCES	CREF	/01
BYTADR	#15-1162	#63-2412	#64-2417
BYTEMG	#62-2401	#63-2411	#64-2416
COMLEN	#15-1199		
COUNT1	#15-1074	57-2375	58-2380
DUAL	#15-1033	51-2333	52-2338
DUALMG	#50-2319	#51-2332	#52-2337
ENDCOM	#15-1199		
ESCAPE	#15-1199		
GETPRI	#15-1199	84-3215	
GETSWR	#15-1199	#33-1839	33-1839
MSG1	#34-1860	35-1866	
MSG10	#43-2080	#44-2083	
MSG11	#44-2091	45-2094	
MSG12	#45-2102	46-2114	
MSG13	#46-2158	47-2167	
MSG14	#47-2179	48-2191	
MSG14A	#48-2279	#49-2286	
MSG15	#50-2329	#51-2332	
MSG15A	#51-2334	#52-2337	
MSG15B	#52-2339	#53-2342	
MSG15C	#53-2344	54-2347	
MSG15D	#54-2349	#55-2352	
MSG15E	#55-2354	56-2357	
MSG16	#56-2371	#57-2374	
MSG16A	#57-2376	#58-2379	
MSG16B	#58-2381	#59-2384	
MSG16C	#59-2386	#60-2389	
MSG16D	#60-2391	#61-2394	
MSG16E	#61-2396	#62-2399	
MSG17	#62-2408	#63-2411	
MSG17A	#63-2413	#64-2416	
MSG17B	#64-2418	65-2421	
MSG17C	#65-2423	66-2426	
MSG17D	#66-2428	67-2431	
MSG17E	#67-2433	#68-2436	
MSG2	#35-1883	36-1888	
MSG20	#68-2438	69-2451	
MSG21A	#69-2469	#70-2478	
MSG21B	#70-2564	71-2576	
MSG21C	#71-2603	#72-2616	
MSG22	#73-2663	74-2674	
MSG23	#75-2754	76-2768	
MSG23A	#76-2895	77-2908	
MSG24	#77-3032	#78-3052	
MSG25	#79-3168	80-3171	
MSG25A	#80-3173	81-3176	
MSG26	#81-3180	#82-3183	
MSG27	#82-3187	83-3197	
MSG3	#36-1906	#37-1913	
MSG4	#37-1942	38-1953	
MSG5	#38-1995	#39-2002	
MSG6	#39-2036	40-2039	

MACRO CROSS REFERENCE

MACRO NAME	REFERENCES									
MSG7	#40-2047	41-2050								
MSG7A	#41-2058	42-2061								
MSG7B	#42-2069	#43-2072								
MULT	#15-1199									
NEWTST	#15-1188	#15-1199	#34-1866	#35-1888	#36-1913	#37-1953	#38-2002	#39-2039	#40-2050	#41-2061
	#42-2072	#43-2083	#44-2094	#45-2114	#46-2167	#47-2191	#48-2286	#50-2332	#51-2337	#52-2342
	#53-2347	#54-2352	#55-2357	#56-2374	#57-2379	#58-2384	#59-2389	#60-2394	#61-2399	#62-2411
	#63-2416	#64-2421	#65-2426	#66-2431	#67-2436	#68-2451	#69-2478	#70-2576	#71-2616	#73-2674
	#75-2768	#76-2908	#77-3052	#79-3171	#80-3176	#81-3183	#82-3197			
PARMSG	#56-2359	57-2374	58-2379	59-2384	60-2389	61-2394	62-2399			
POP	#15-1199	92-3335	92-3335	95-3341	96-3343	99-3349	99-3349			
PUSH	#15-1199	92-3335	92-3335	92-3335	95-3341	96-3343	99-3349	99-3349		
REPORT	#15-1199									
SAVR	#15-936	#86-3219								
SETPRI	#15-1199									
SETTRA	#98-3347	98-3347	98-3347	98-3347	98-3347	98-3347	98-3347	98-3347	98-3347	98-3347
	98-3347	98-3347	98-3347							
SETUP	#15-1199	33-1838								
SKIP	#15-1199	39-2022	51-2333	52-2338	53-2343	54-2348	55-2353	56-2358	57-2375	58-2380
	59-2385	60-2390	61-2395	62-2400	72-2643	75-2744	75-2746	75-2752	76-2867	77-3004
	78-3144									
SLASH	#15-1199									
SPACE	#15-933	#15-1199								
STARS	#15-1199	15-1274	15-1275	15-1275	15-1275	16-1276	16-1276	16-1276	23-1576	23-1578
	23-1585	24-1596	24-1602	25-1611	25-1620	26-1629	26-1633	27-1645	27-1651	28-1666
	28-1674	30-1730	30-1732	30-1739	31-1763	31-1770	32-1792	32-1806	33-1834	33-1836
	34-1857	34-1859	35-1866	35-1866	36-1888	36-1888	37-1913	37-1913	38-1953	38-1953
	38-1982	38-1994	39-2002	39-2002	40-2039	40-2039	41-2050	41-2050	42-2061	42-2061
	43-2072	43-2072	44-2083	44-2083	45-2094	45-2094	46-2114	46-2114	47-2167	47-2167
	48-2191	48-2191	49-2286	49-2286	50-2316	50-2318	51-2332	51-2332	52-2337	52-2337
	53-2342	53-2342	54-2347	54-2347	55-2352	55-2352	56-2357	56-2357	57-2374	57-2374
	58-2379	58-2379	59-2384	59-2384	60-2389	60-2389	61-2394	61-2394	62-2399	62-2399
	63-2411	63-2411	64-2416	64-2416	65-2421	65-2421	66-2426	66-2426	67-2431	67-2431
	68-2436	68-2436	69-2451	69-2451	70-2478	70-2478	71-2576	71-2576	72-2616	72-2616
	73-2660	73-2662	74-2674	74-2674	76-2768	76-2768	77-2908	77-2908	78-3052	78-3052
	79-3165	79-3167	80-3171	80-3171	81-3176	81-3176	82-3183	82-3183	83-3197	83-3197
	84-3215	85-3217	86-3219	89-3315	89-3315	89-3315	89-3315	91-3333	92-3335	93-3337
	94-3339	95-3341	96-3343	97-3345	98-3347	99-3349	99-3349			
SWRSU	#15-1199	#33-1838	#33-1838							
TIMMSG	#15-1028	#40-2039	#41-2050	#42-2061	#43-2072	#44-2083	#45-2094			
TRMTRP	#98-3347									
TYPBIN	#15-1199									
TYPDEC	#15-1199	84-3215	84-3215							
TYPNAM	#15-1199	33-1839								
TYPNUM	#15-1199									
TYPOCS	#15-1199									
TYPOCT	#15-1199	#89-3315								
TYPTXT	#15-1199	84-3215	84-3215							
USER	#15-1225	#16-1276								
WMSG	#15-1101	#80-3171	#81-3176	#82-3183						
WTST	#15-1116	80-3172	81-3178	82-3185						
SSCMRE	#15-1276	#16-1276	#16-1276	#16-1276	#16-1276	#16-1276	#16-1276			

